

**《数据库课程设计》**

**实验报告**

**实验名称： 宾馆管理系统**

**班 级： 软件Q185**

**组 员： 朱行军 张建军 马瑞 谭清流**

**指导老师： 王磊**

**日 期： 2020 年 7 月 14 日**

**评阅成绩：**

**教师签名：**

# 目 录

[目 录 2](#_Toc45613889)

[引 言 3](#_Toc45613890)

[1数据库概念模型 4](#_Toc45613891)

[1.1 需求分析 4](#_Toc45613892)

[1.1.1总经理： 4](#_Toc45613893)

[1.1.2会计 4](#_Toc45613894)

[1.1.3前台 4](#_Toc45613895)

[1.2 概念结构设计 5](#_Toc45613896)

[1.2.1抽象出实体 5](#_Toc45613897)

[1.2.2局部E-R图 5](#_Toc45613898)

[1.2.3整体E-R图 9](#_Toc45613899)

[2数据库逻辑模型 9](#_Toc45613900)

[2.1将E-R图转化为关系模式（关系的码用下划线标出） 9](#_Toc45613901)

[3数据库物理设计与实施 10](#_Toc45613902)

[3.1创建数据库 10](#_Toc45613903)

[3.2创建和管理基本表 12](#_Toc45613904)

[3.2.1创建基本表 12](#_Toc45613905)

[3.2.2管理基本表 16](#_Toc45613906)

[3.2.3访问数据库 17](#_Toc45613907)

[4程序设计 17](#_Toc45613908)

[4.1创建应用程序 17](#_Toc45613909)

[4.2连接数据库 18](#_Toc45613910)

[4.3设计应用程序 18](#_Toc45613911)

[5测试用例 25](#_Toc45613912)

[6作业总结 36](#_Toc45613913)

[附 录 38](#_Toc45613914)

# 引 言

（正文小四号宋体，1.5倍行距，首行缩进2字符，下同）

（将本项目的选题的题目、项目背景、需求功能内容及选题意义写进来）

近年来，随着改革开放步伐的加快和中西方文化的交流与发展，我国人民生活水平显著提高，旅游业日益繁荣发展，伴随着消费的持续增长，宾馆行业已经成为现代社会中发展最快的行业之一。根据调查发现，只有大型的宾馆才有规范、合理的管理系统，大部分中小型宾馆都还是人工服务，因此，中小型宾馆要想在激烈的市场竞争中生存，就必须在经营管理、服务质量等方面提高管理意识，即建立一套合理、高效的管理系统。应势而出的宾馆管理系统有着很好的应用性。一直以来人们都采用传统人工的方式管理账单，这种管理方式存在着许多缺点，如：效率低，另外时间一长，将产生大量的文件和数据，这对于查找更新和维护都带来了很多的困难，同时，也加大了管理者的工作量和工作难度。本系统很好的克服了这些缺点，并且有检索迅速、查找方便、可靠性高、存储量大、寿命长、成本低等优点。这些优点能够极大地提高宾馆管理的效率，该宾馆管理系统以Microsoft SQL Server 2008作为后台数据库，Java语言为前台开发工具，运用面向对象的设计方法，完成了宾馆管理系统的主要模块。现如今随着经济的不断发展，旅游出国等带来的宾馆市场营业量大大增加，如果设计一款专门为宾馆管理人员和宾馆员工查询本宾馆信息，以及各种查询、入住、核账、结算等的专业软件，将会大大减少宾馆内工作人员的工作量并提高工作效率同时提升宾馆的营业额。因此，该系统的开发不仅解决了中小型宾馆管理复杂、不规范等问题，而且优化了管理流程，实现了宾馆管理工作的无纸化办公。

# 1数据库概念模型

## 需求分析

### 1.1.1总经理：

1）维护职员的个人基本信息，实现对职员个人信息的添加、删除和更新等，职员信息包括职员的职工号、职务、姓名、性别、年龄、电话等。

2）维护VIP客户的个人基本信息，实现对VIP客户个人信息的添加、删除和更新等。VIP客户信息包括客户的身份证号、姓名、性别、身份证号、电话等。

3）维护房间信息，实现对房间信息的添加、删除和修改等。房间信息包括房间号、房间类型、当前房间状态、金额等

4）维护订单信息，实现对订单信息的添加、删除和修改等。订单信息包括订单号、身份证号、消费项目，消费金额，订单状态等

### 1.1.2会计

1）查询订单信息，如订单号、消费额等。

2）核实销售额是否有误

### 1.1.3前台

1）查询房间信息：如当前房间状态（已预定，已入住，转房，未入住等）、入住人员等。

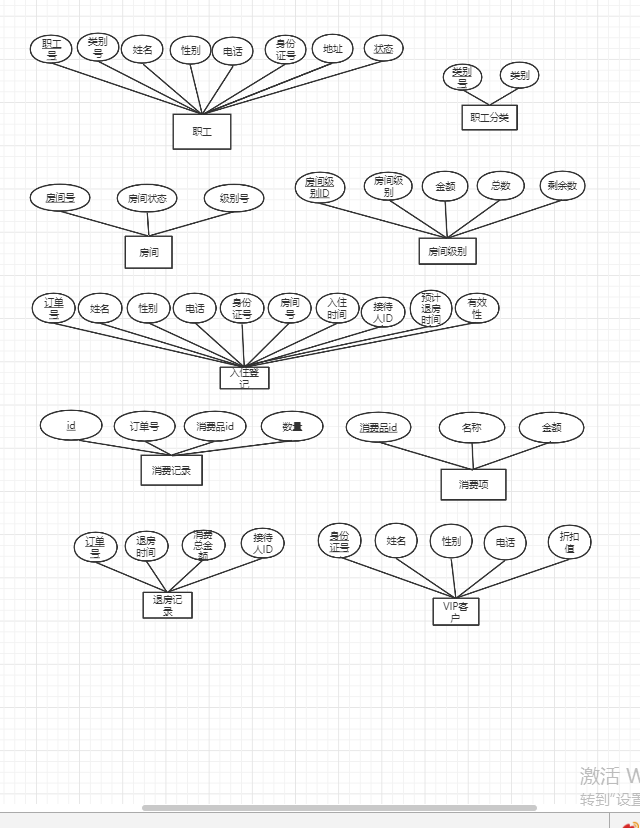
2）添加订单信息（退房结算），如订单号、房间号、入住时间、退房时间、消费金额，订单状态等

## 1.2 概念结构设计

### 1.2.1抽象出实体

根据分析，宾馆管理系统主要包含职工，职工类别，房间，房间类别，入住登记信息，退房记录，消费项，消费清单，VIP客户九个实体，下列为这九个实体的局部E-R图，图中加下划线的属性为各实体的主键。

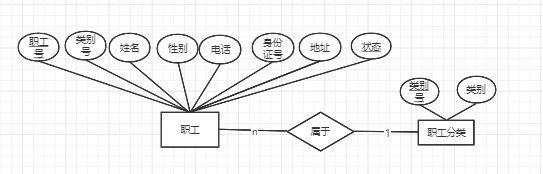
（各实体及属性）



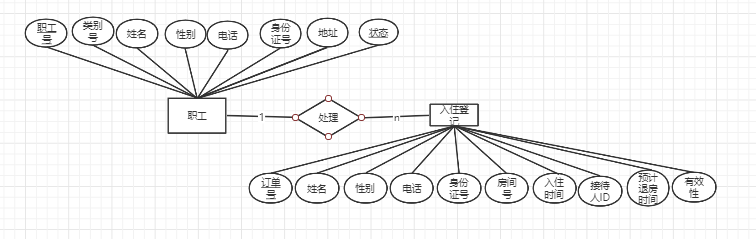
### 1.2.2局部E-R图

（各子系统/子部门的E-R图，含实体、联系和属性）

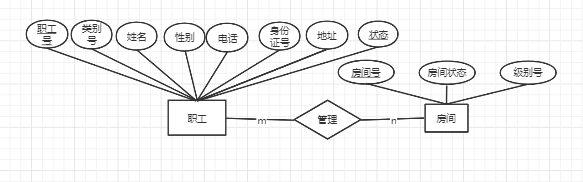
职工与职工分类：一个职工只属于一个类别，一个类别拥有多个员工，所以二者是一对多的关系



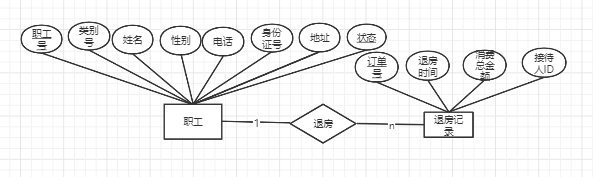
职工与入住登记：一个职工可以处理多个入住登记信息，一个入住登记信息只能由一个职工处理，所以二者是一对多的关系



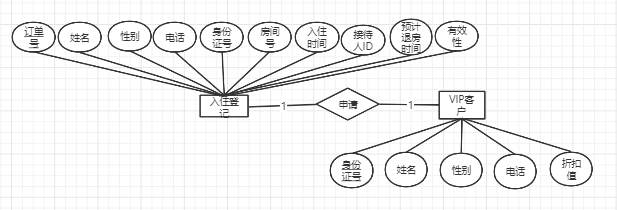
职工与房间：一个职工可以管理多个房间，一个房间也可由多个个职工处理，所以二者是多对多的关系



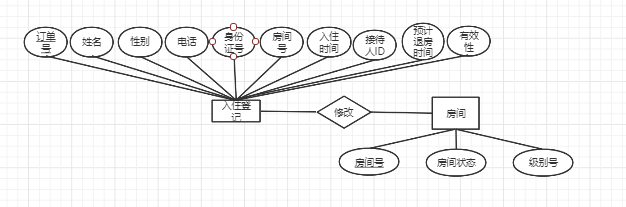
职工与退房记录：一个职工可以处理多个退房记录，一个退房记录只能由一个职工处理，所以二者是一对多的关系



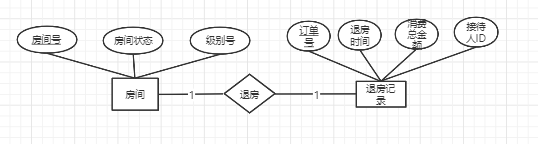
入住登记与VIP客户：一次入住登记可以申请成为一个VIP客户，一个VIP客户只能由一次入住登记生成，所以二者是一对一的关系



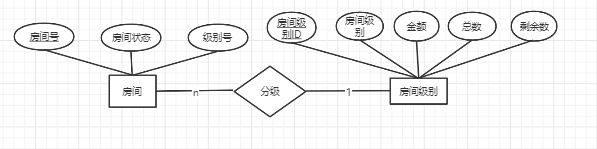
入住登记与房间：一个房间的变动会修改一个入住登记的信息，一个入住登记的生产会修改一个房间的状态，所以二者是一对一的关系



房间与退房记录：一个房间的退房会产生一个退房记录，一个退房记录只会退掉一间房，所以二者是一对一的关系

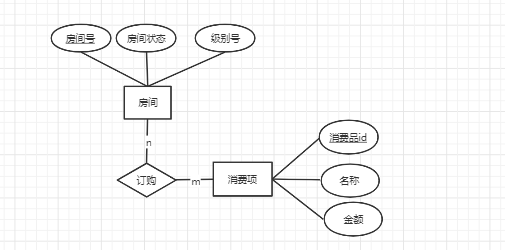


房间与房间级别：一个房间只属于一种级别的房间，一种级别的房间拥有多个房间，所以二者是一对多的关系

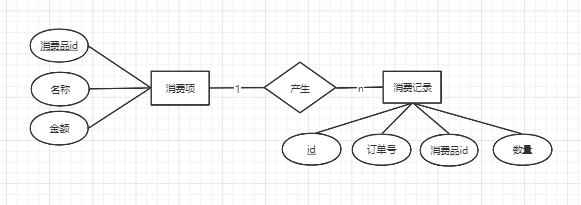


房间，消费项和消费记录三个实体均存在联系

房间与消费项：一个房间可以订购多个消费项，一个消费项也可以由多个房间订购，所以二者是多对多的关系

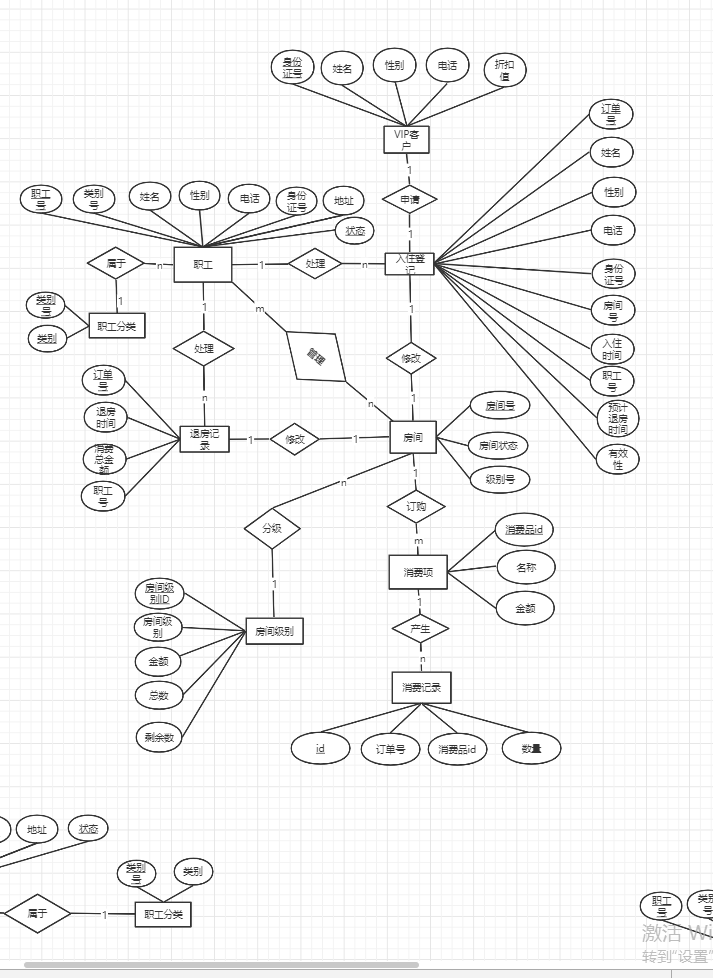


消费项与消费记录：一个消费项可以产生多个消费记录，一个消费记录只能对于一个消费项，所以二者是一对多的关系



### 1.2.3整体E-R图

（项目整体E-R图）



# 2数据库逻辑模型

## 2.1将E-R图转化为关系模式（关系的码用下划线标出）

职工（职工号，类别号，姓名，性别，电话，身份证号，地址，状态）为职工实体对应的关系模式。

职工分类（类别号，类别）为职工分类实体对应的关系模式。

VIP客户（身份证号，姓名，性别，电话，折扣值，VIP状态）为客户实体对应的关系模式。

房间（房间号，房间状态，级别号）为房间实体对应的关系模式。

房间级别（房间级别ID，房间级别，金额，总数，剩余数）为房间级别实体对应的关系模式。

入住登记（订单号，姓名，性别，电话，身份证号，房间号，入住时间，预计退房时间，职工号，有效性）为入住登记实体对应的关系模式。

退房记录（订单号，退房时间，消费总金额，职工号）为退房记录实体对应的关系模式

消费项（消费品id，名称，金额）为消费项实体对应的关系模式

消费记录（id，订单号，消费品id，数量）为消费记录实体对应的关系模式

（列出转化后的关系模式，关系的码用下划线标出）

）

# 3数据库物理设计与实施

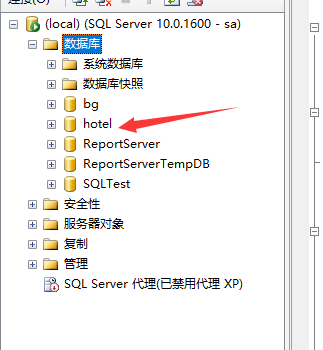
## 3.1创建数据库

（数据库创建过程和结果截图）

数据库过程如下



结果如下

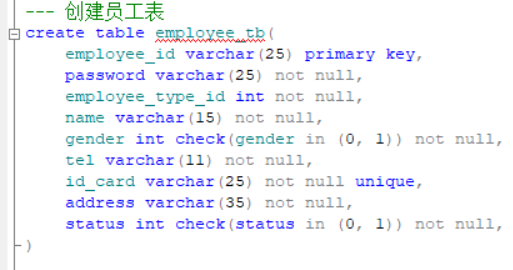


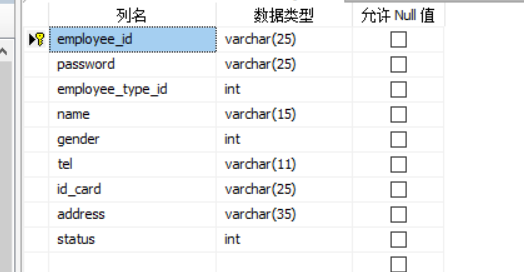
## 3.2创建和管理基本表

### 3.2.1创建基本表

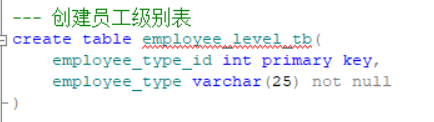
利用SQL语句在查询分析器中建表

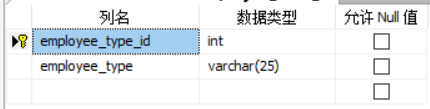
建立职工表





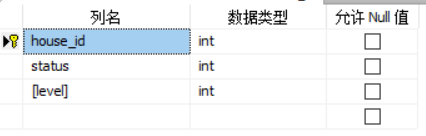
建立职工分类表



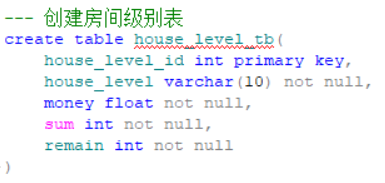


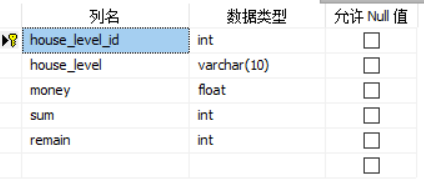
建立房间表



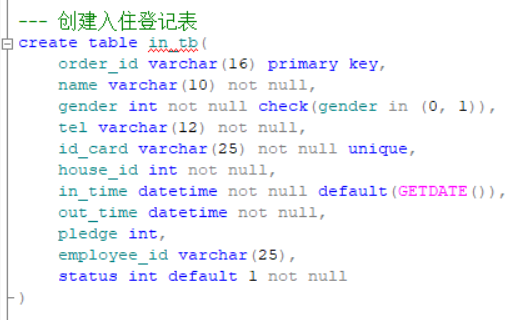


建立房间级别表

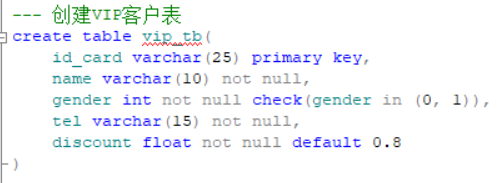




建立入住登记表

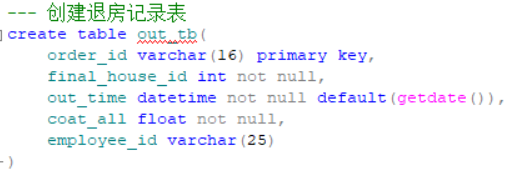


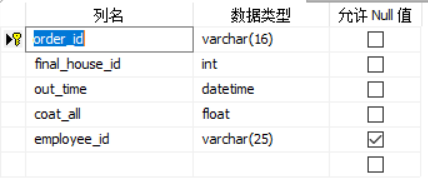
建立VIP客户表



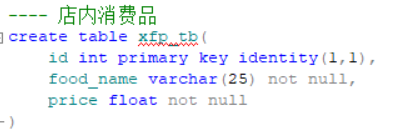


建立退房记录表



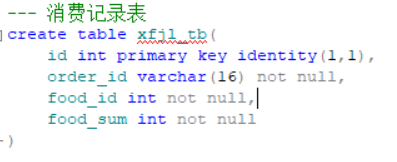


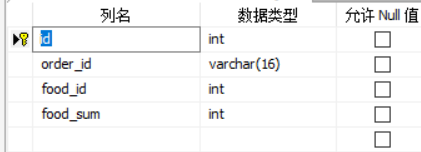
建立消费项表





建立消费记录表





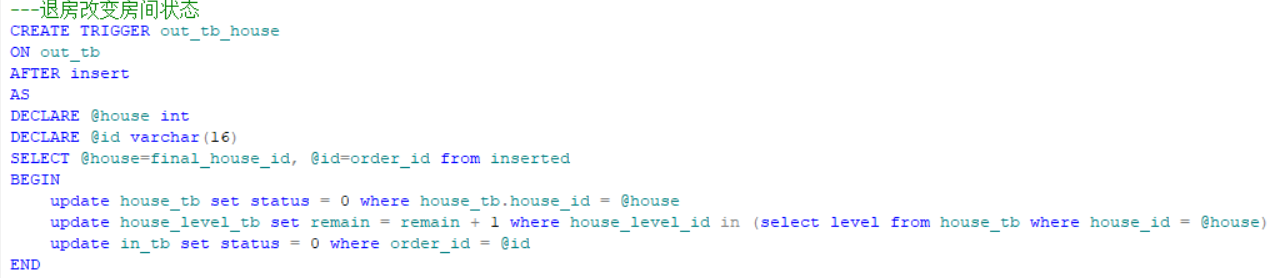
（将各基本表的创建过程和结果截图，可尝试不同的创建方式，如代码创建、对象管理器创建）

### 3.2.2管理基本表

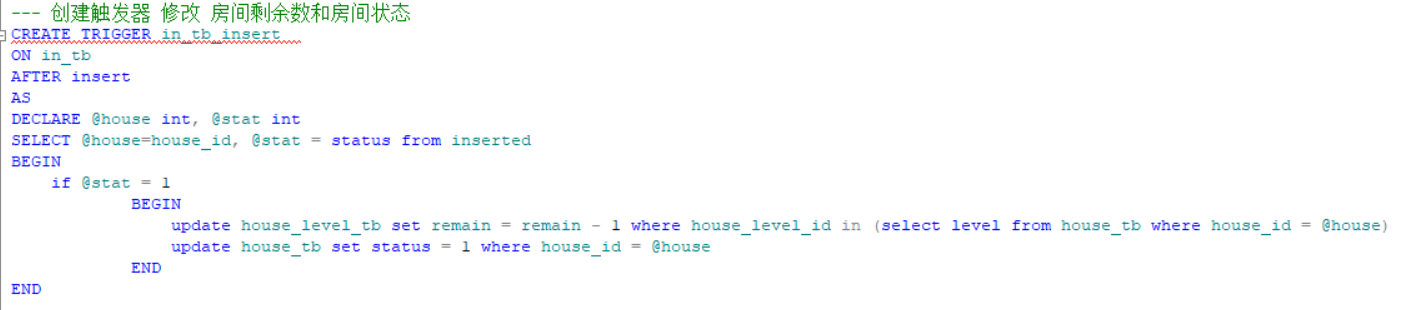
（各基本表的约束设置、索引建立、创建存储过程、创建触发器等内容，过程和结果截图）

创建触发器

1.如果新添退房记录，会修改房间表的对应房间的状态，修改房间级别的房间剩余数



2. 如果新添入住登记，会修改房间表的对应房间的状态，修改房间级别的房间剩余数

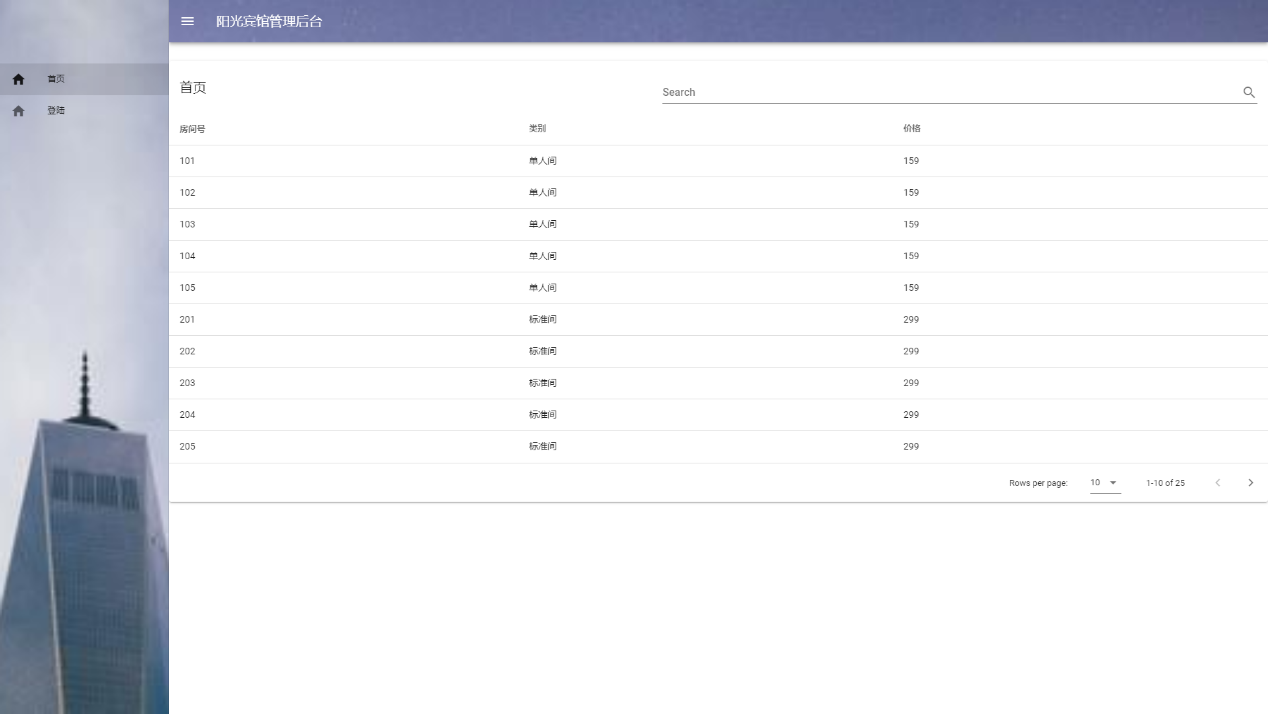


### 3.2.3访问数据库

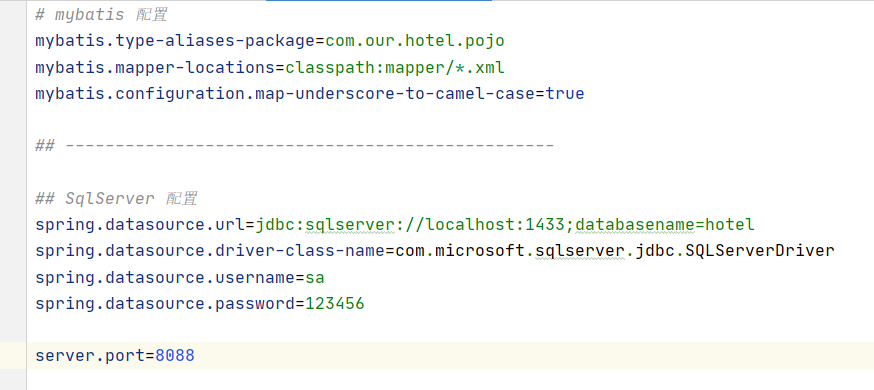
（在DBMS上实现对数据库的增、删、查、改操作，过程和结果截图）

# 4程序设计

## 4.1创建应用程序



## 4.2连接数据库

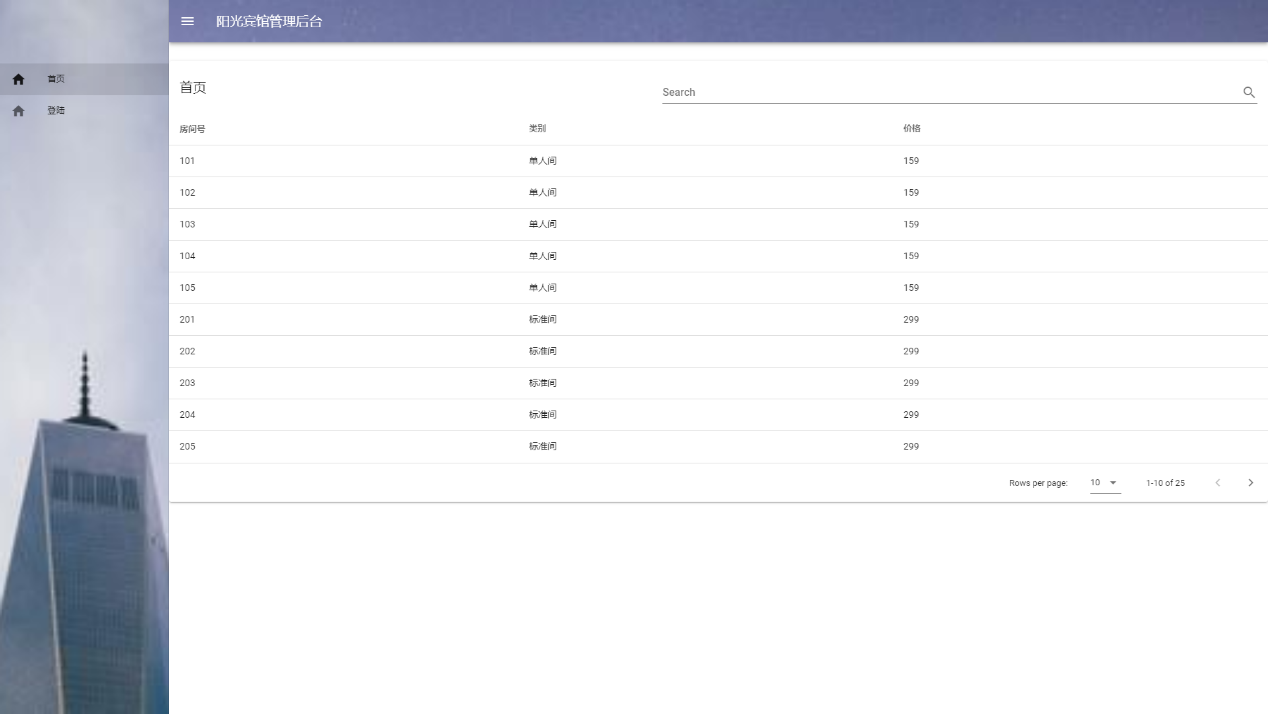


## 4.3设计应用程序

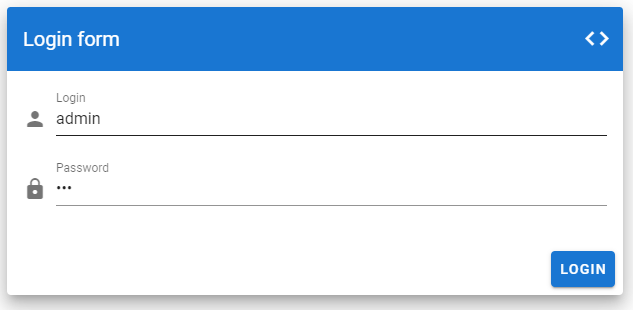
根据系统功能的要求，把程序分为九个页面，即登陆页面，前台展示页面（首页），预约申请页面，预约管理页面，入住申请页面，会员管理页面，人员管理页面，房间管理页面，数据报表页面

1. 游客页面

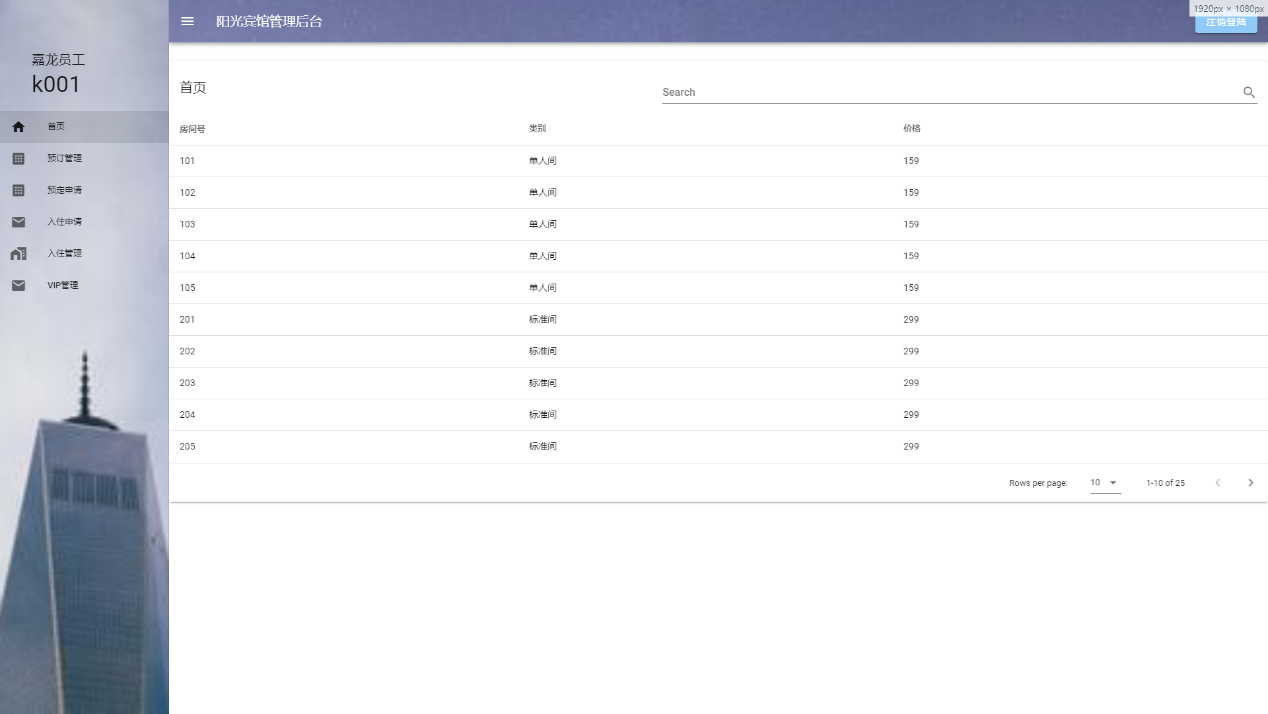
1.前台展示页面（首页）。首页只展示可以入住的房间信息，包括房间号，房间类别，金额。职工可以通过该页面查询房间是否空余

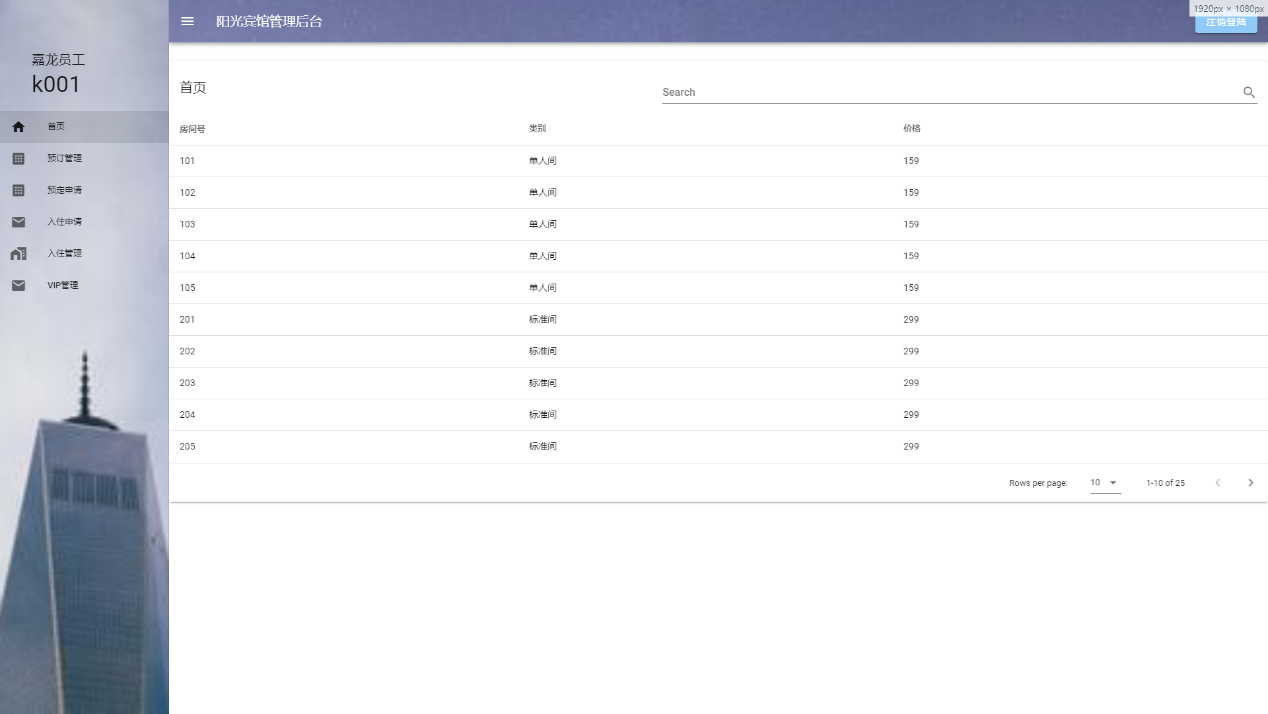


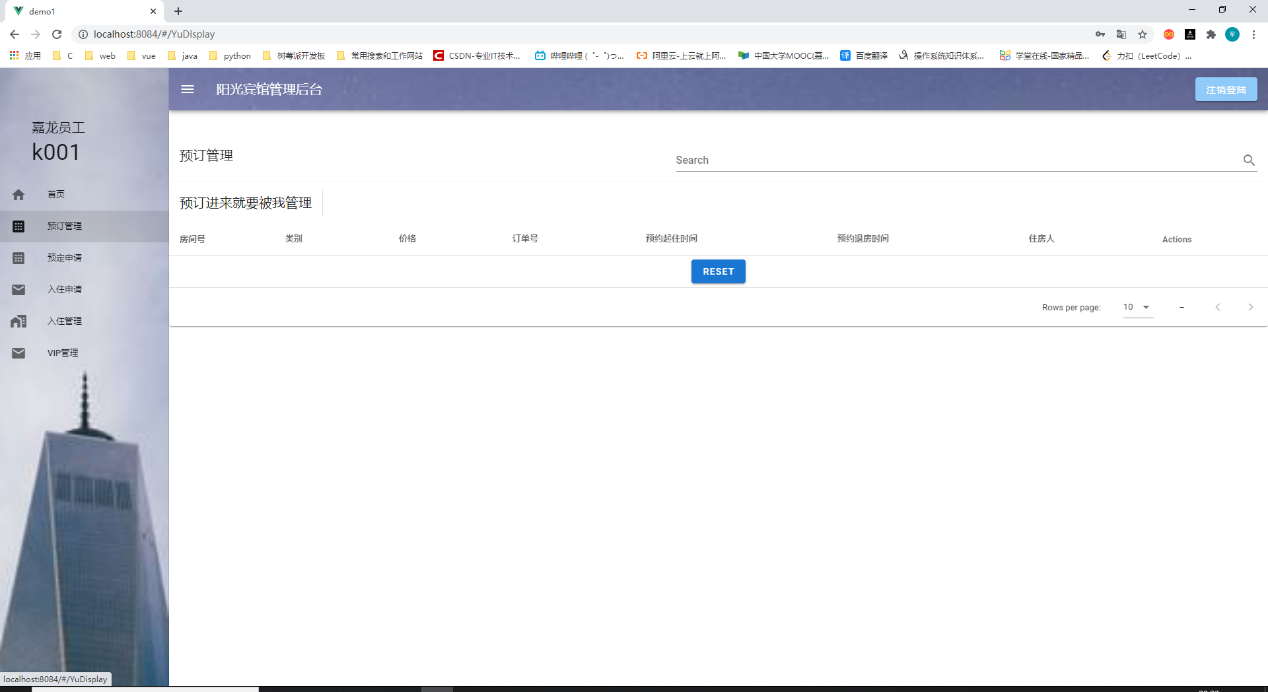
2.登陆页面。职工通过职工号和密码进行登录。



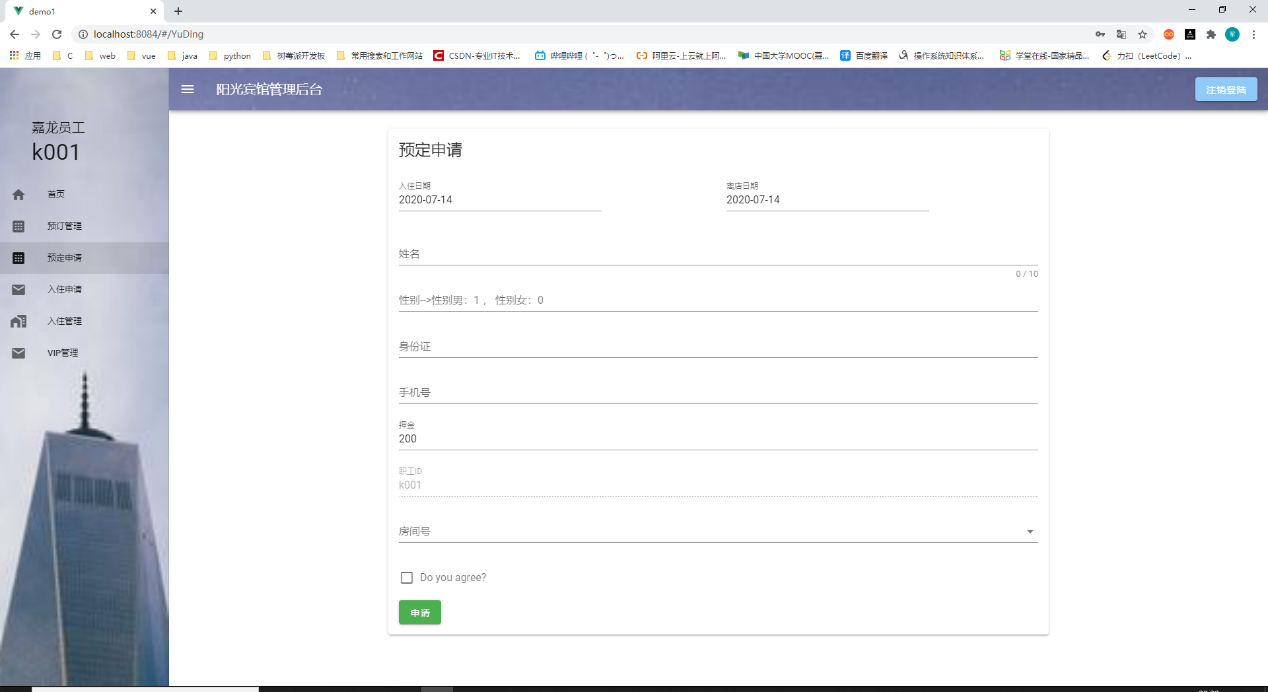
1. 员工（前台）页面（前台具有首页查看，预约管理，预约申请，入住申请，入住管理，VIP管理，五个功能，分为五个）

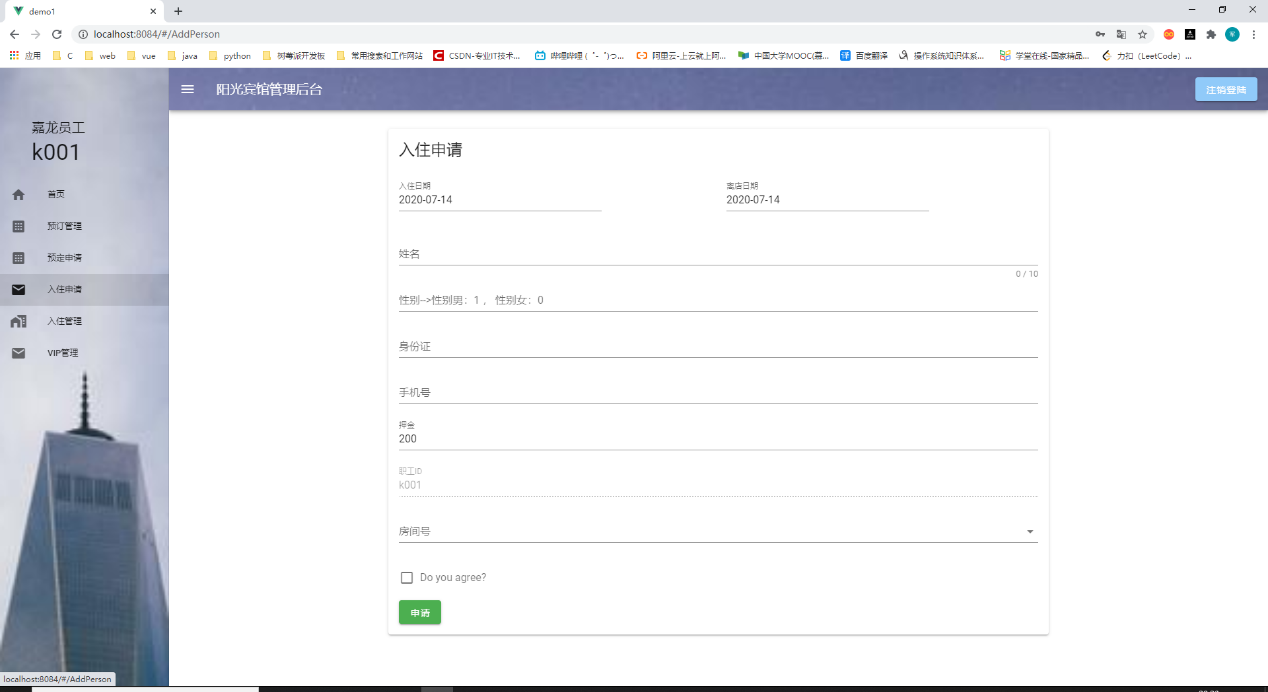
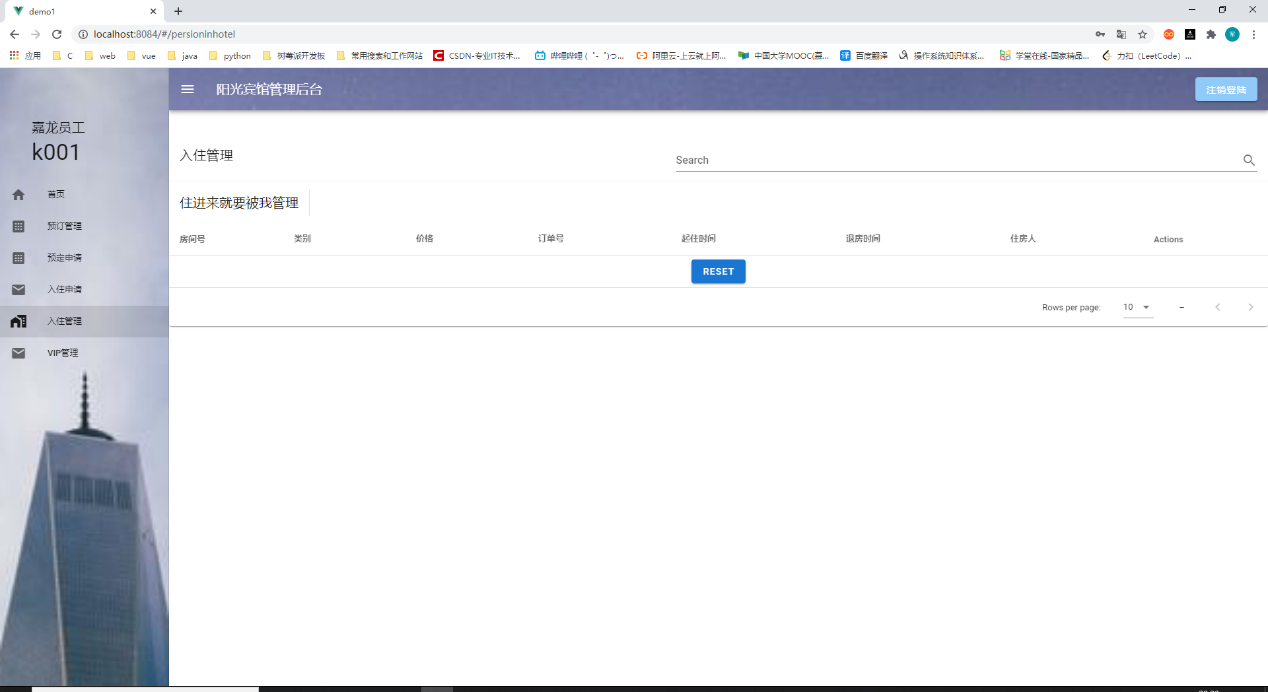
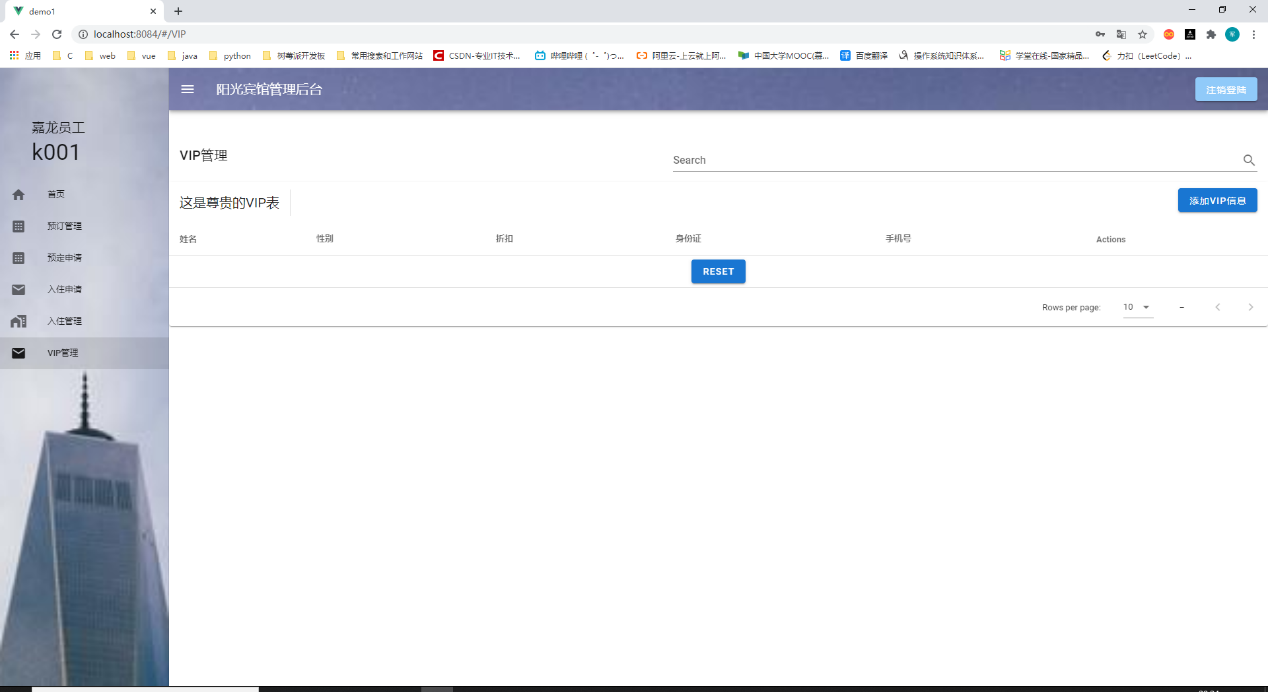


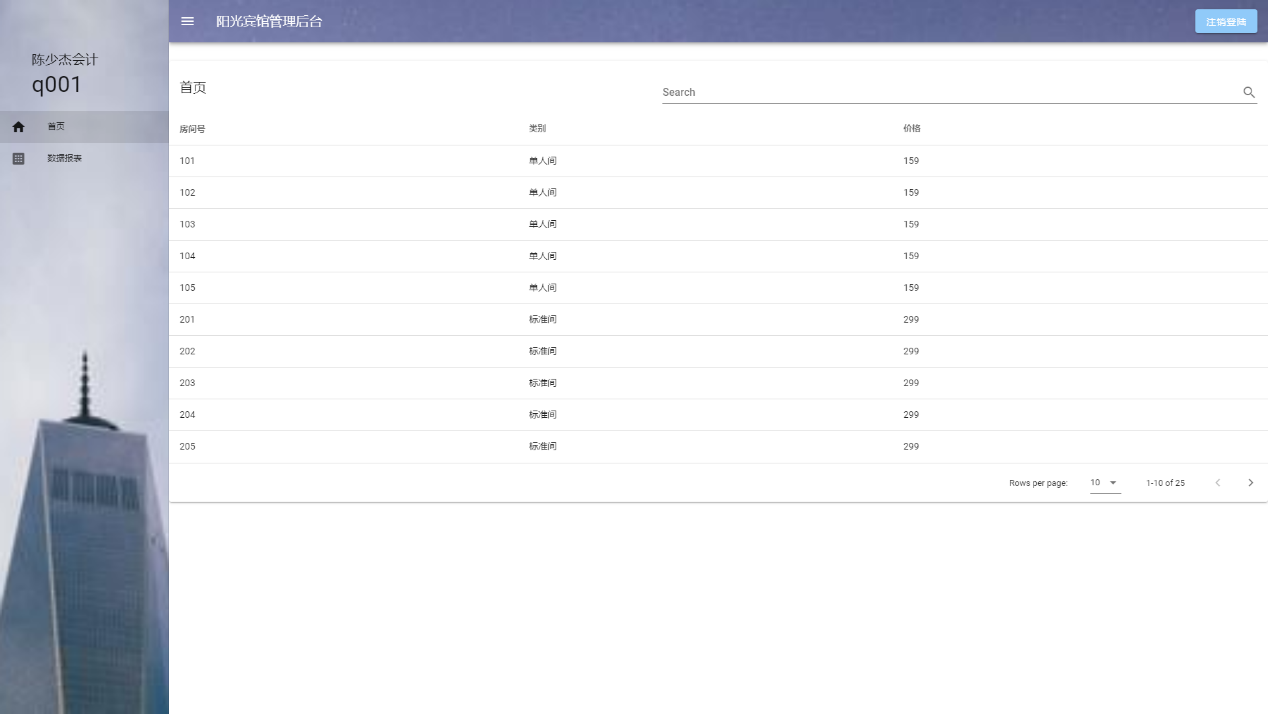
1. 前台展示页面（首页）。首页只展示可以入住的房间信息，包括房间号，房间类别，金额。职工可以通过该页面查询房间是否空余
2. 预约管理页面。此页面用来登记预约申请人的信息，包括入住日期，离店日期，姓名，性别，身份证号等等



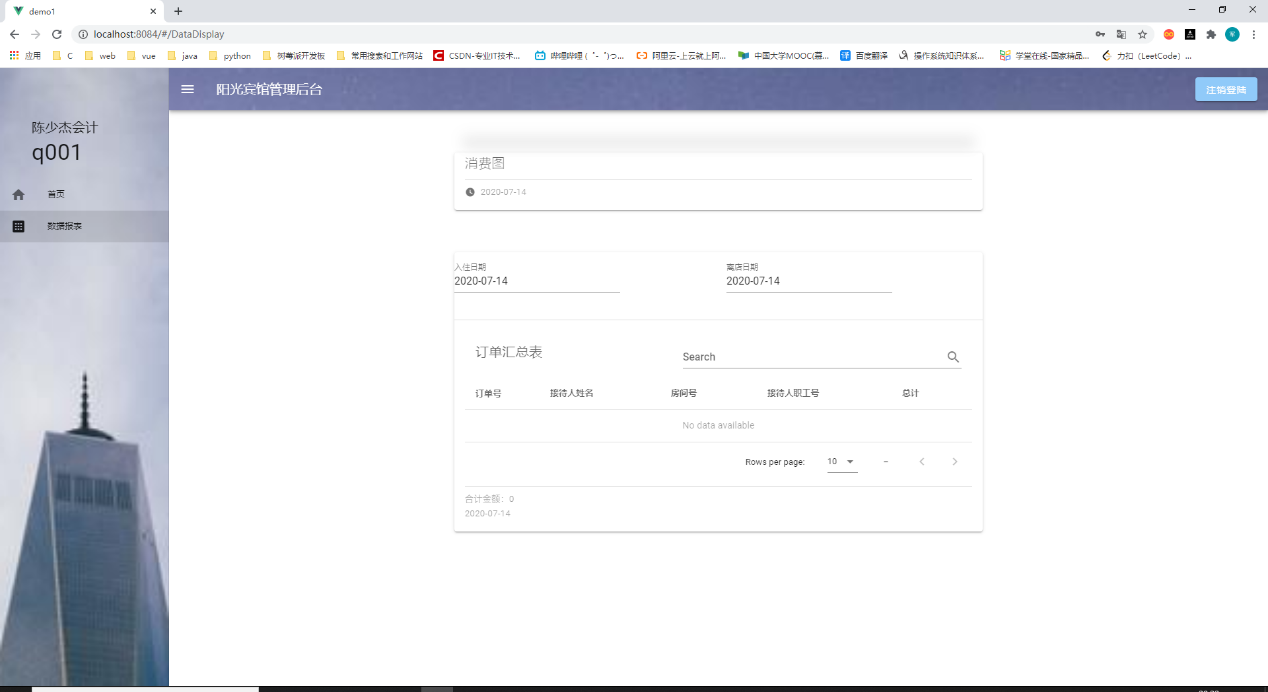
1. 预约管理页面。预约申请页面。此页面用来登记预约申请人的信息，包括入住日期，离店日期，姓名，性别，身份证号等等



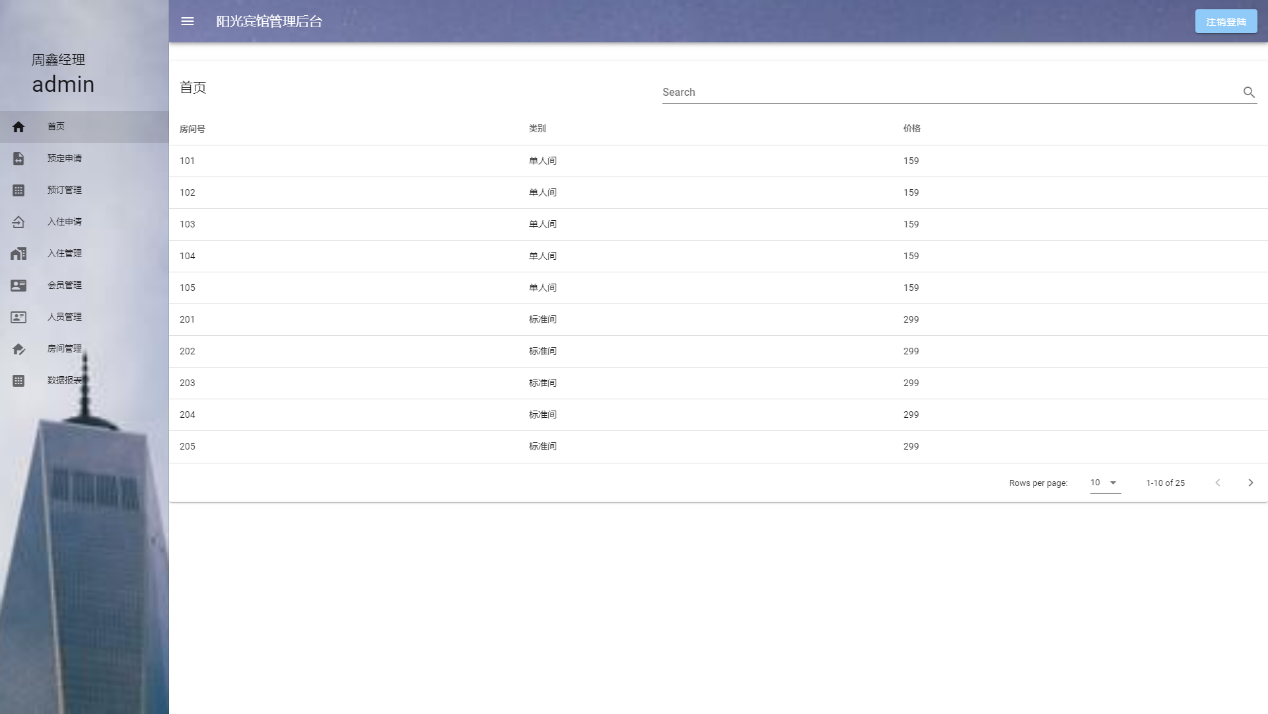
1. 入住申请页面。此页面用来登记入住人的信息，包括入住日期，离店日期，姓名，性别，身份证号等等。
2. 
3. 入住管理页面。此页面用来管理已入住人员，包括房间号，类别，价格，订单号，起住时间等等，职工可通过此页面为客户办理转房和退房服务
4. 
5. 会员管理页面。此页面展示的二是尊贵的VIP用户，包括姓名，性别，折扣值，身份证号等等，职工可通过此页面对VIP客户表进行增删改查。
6. 会计页面（具有查看首页和数据报表的功能）

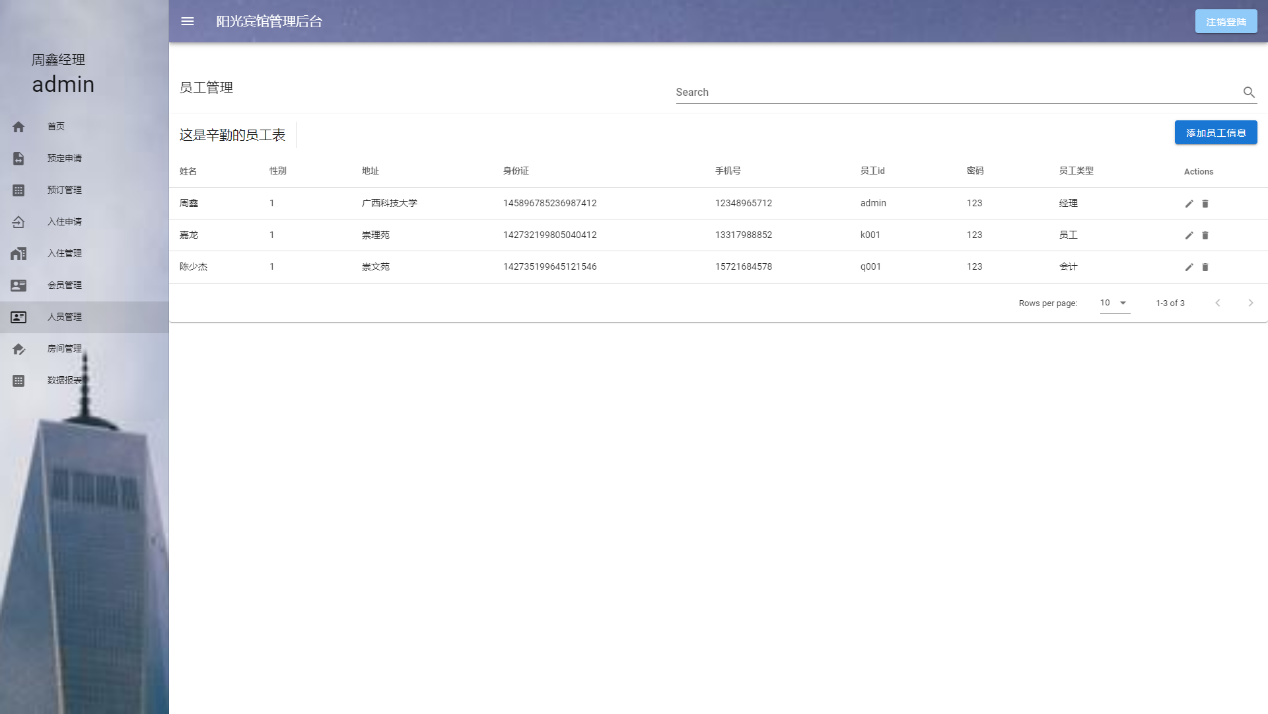


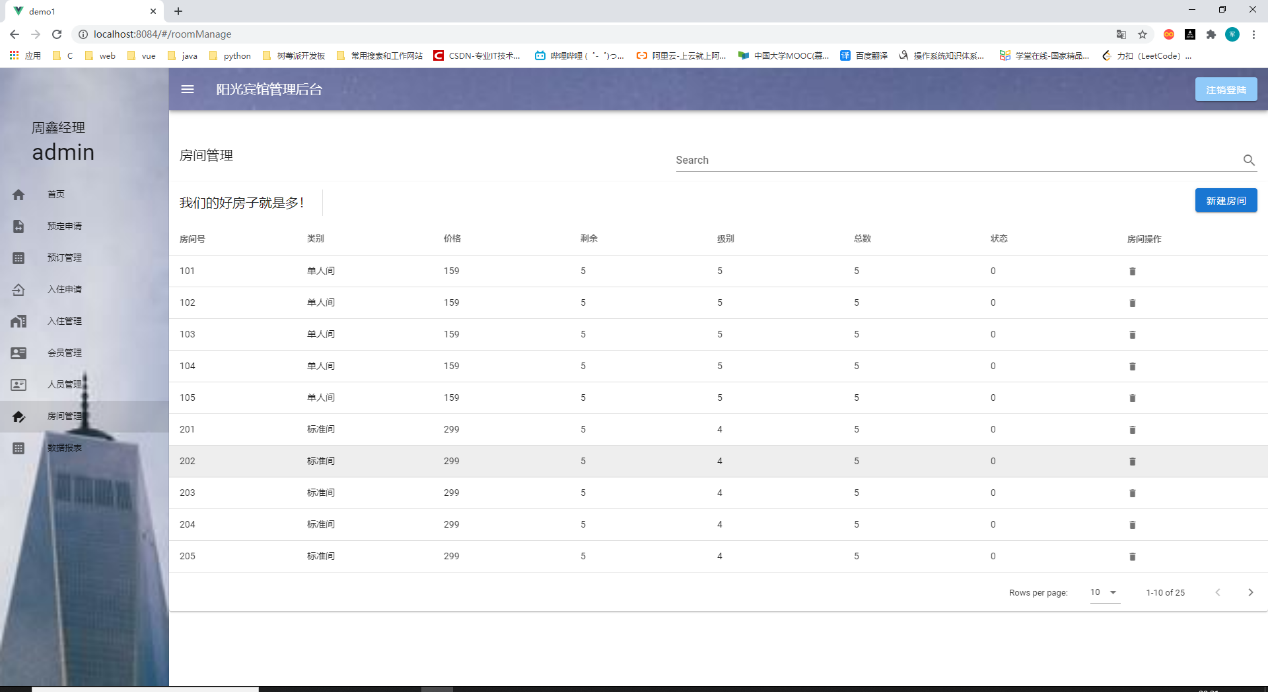
1.数据报表页面。此页面展示的是订单信息，



1. 经理页面（经理拥有会计和前台的功能，下面只展示不重叠功能）



1. 人员管理页面。此页面展示的是职工信息，包括姓名，性别，地址，身份证号，手机号等等，经理可以对通过此页面对职工表进行增删查改。
2. 房间管理页面。此页面展示的是房间信息，包括房间号，类别，价格，状态等等，经理可以对通过此页面对房间表进行增删查改。



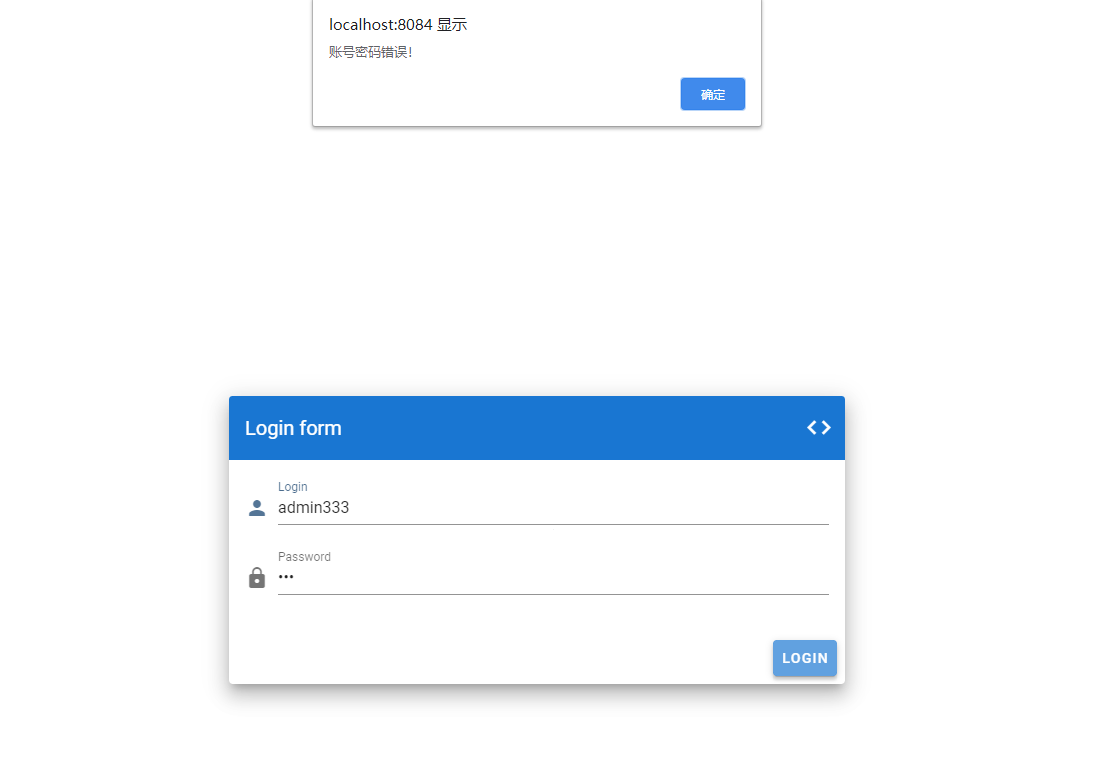
# 5测试用例

（按项目需求使用设计好的应用程序进行实现）

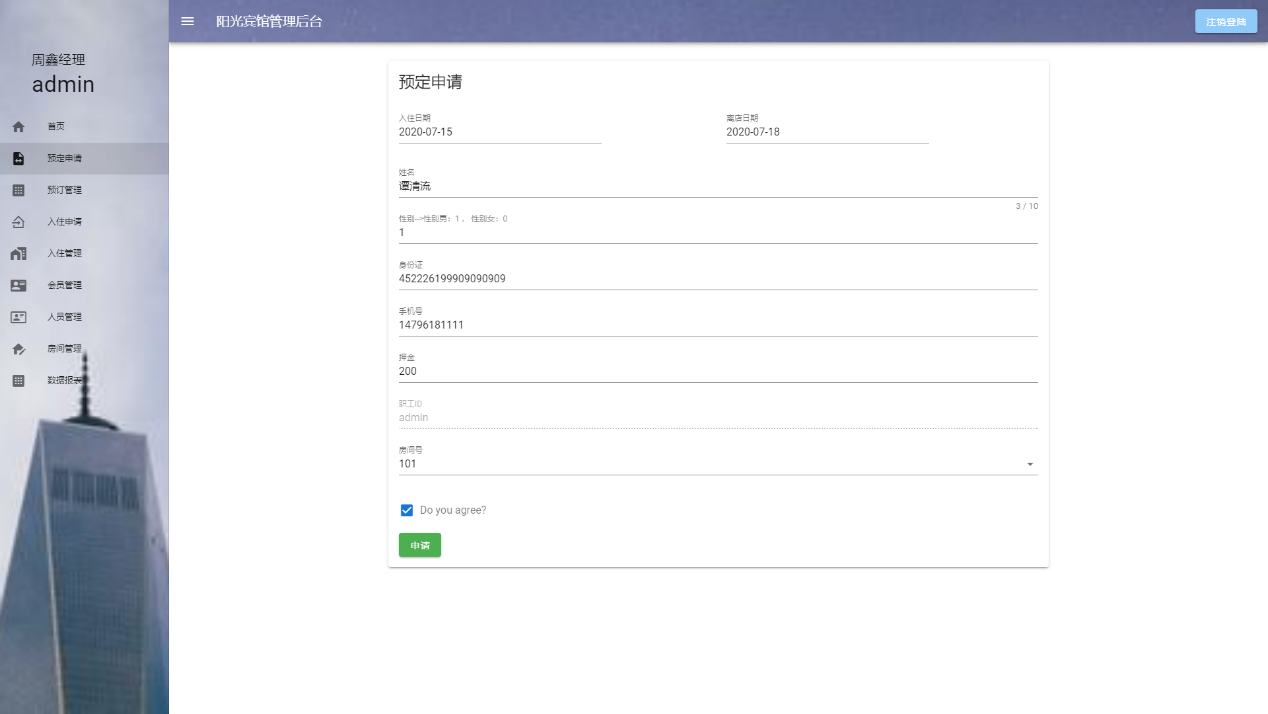
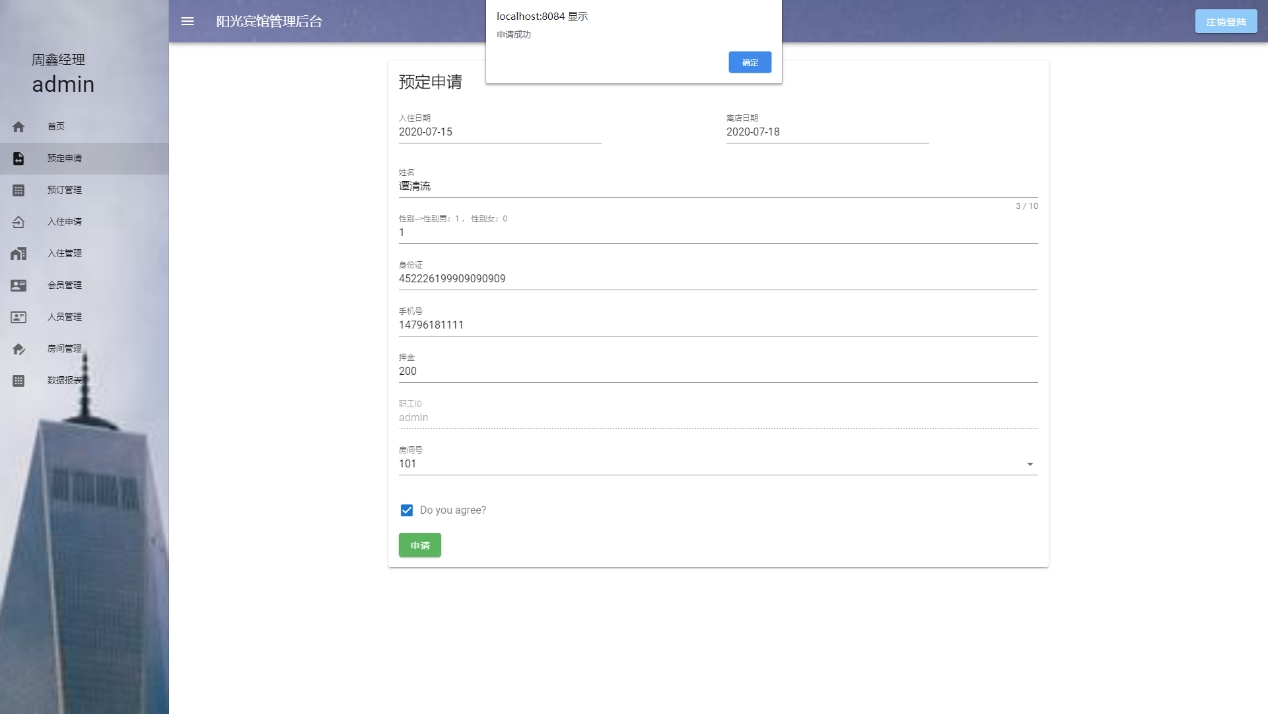
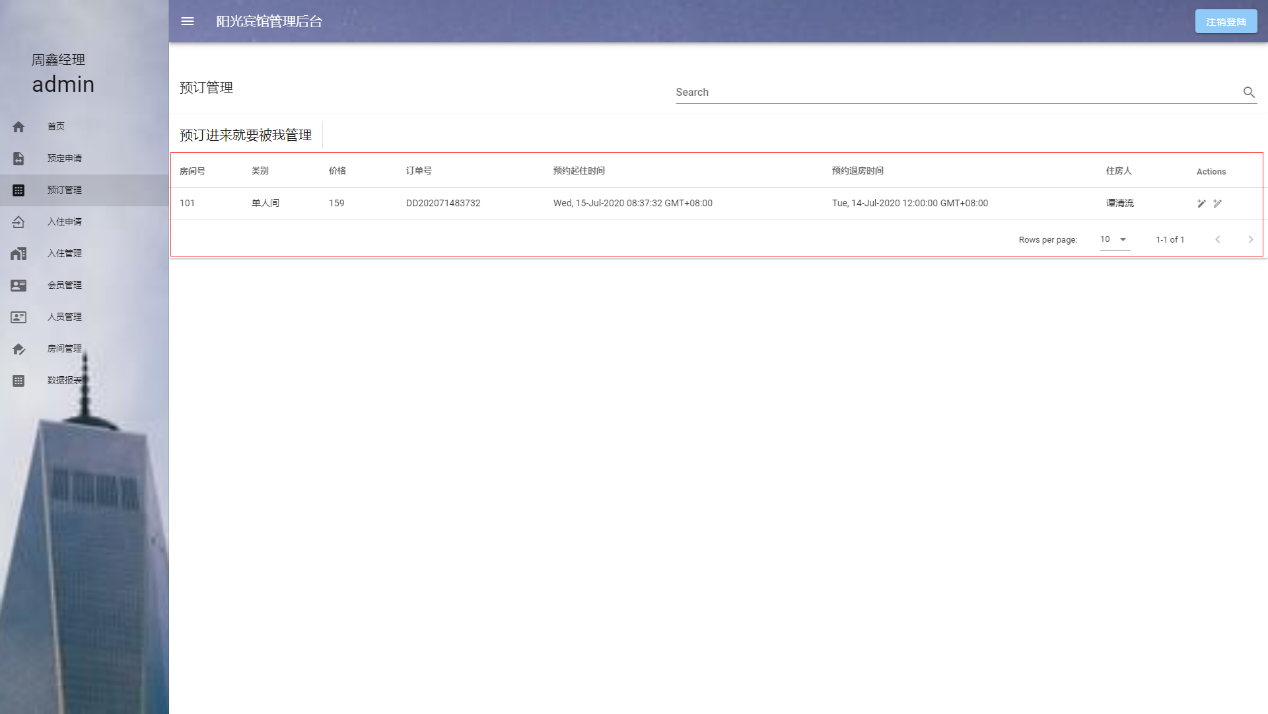
1. 登录测试，输入账号，密码，判断是否存在，是否有误

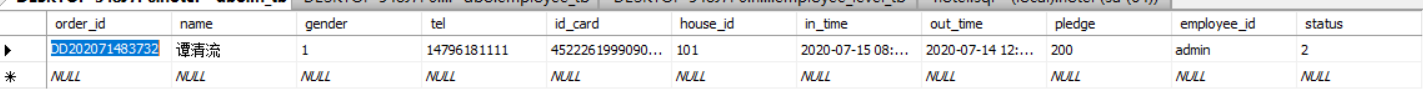
如果没有错误直接进入

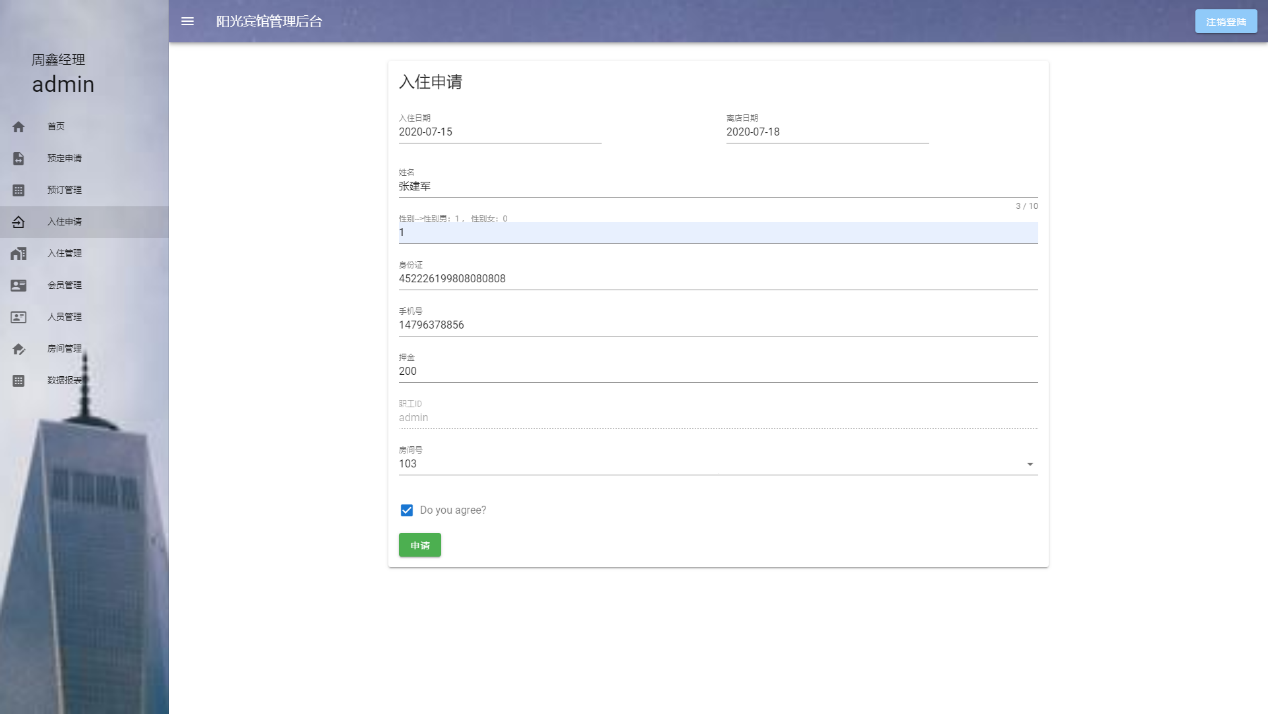
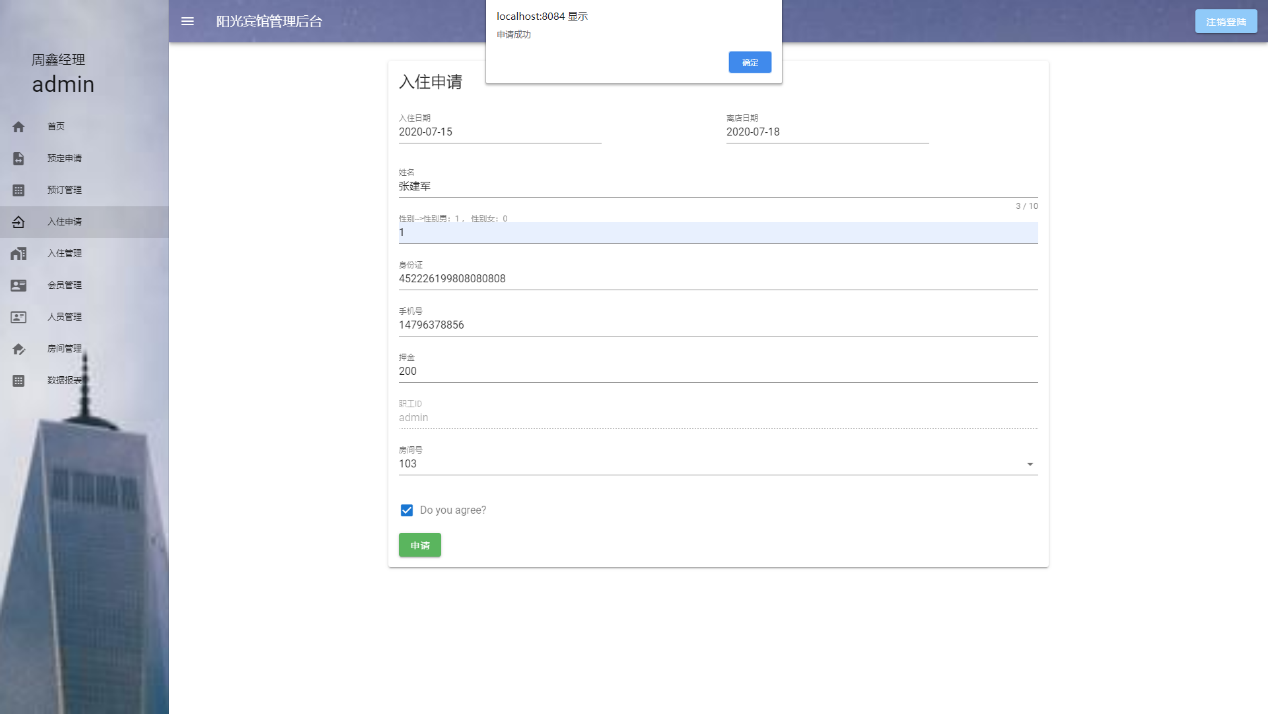
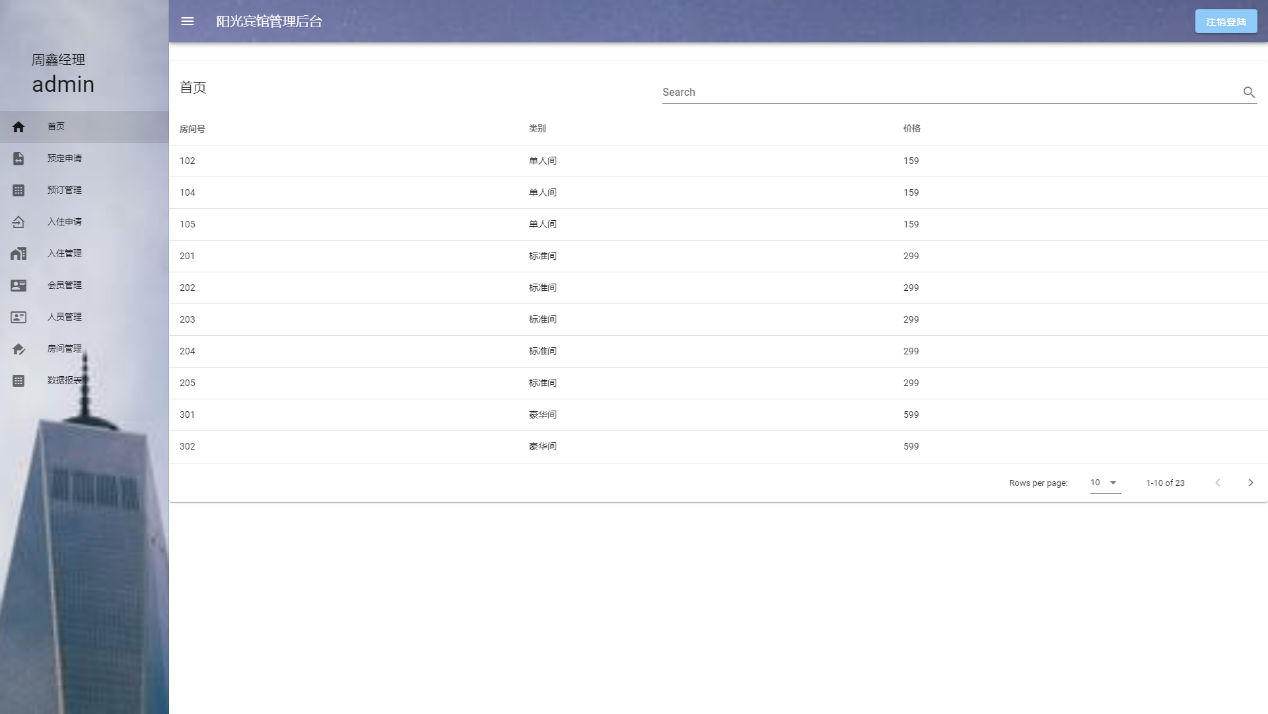
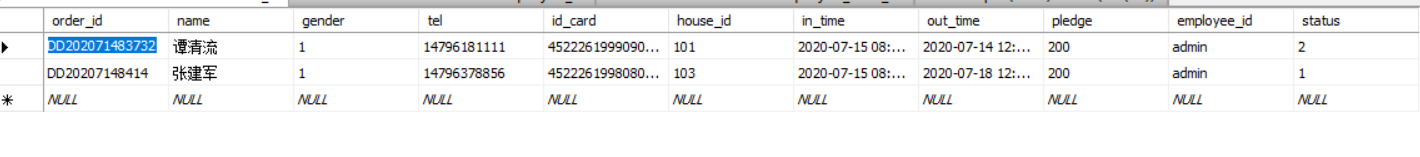
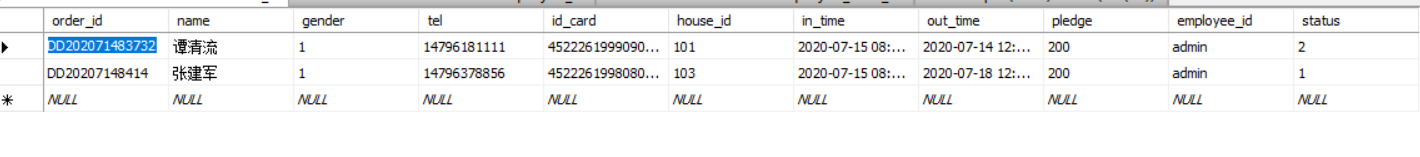
如果有错误如下图所示

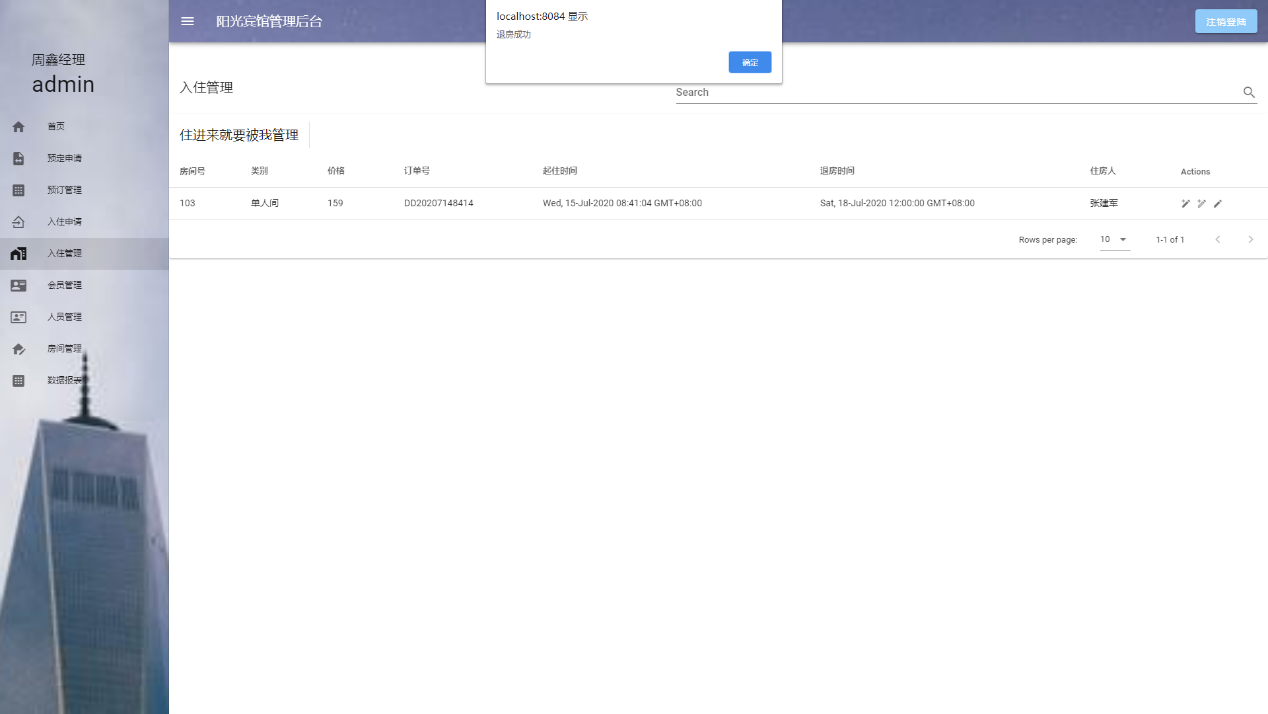
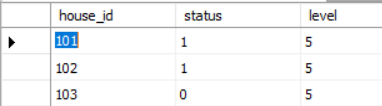


1. 预约申请测试，选取时间段，显示该时间段可用房间号，提交申请，形成申请订单，订单状态标识为已预约（2）

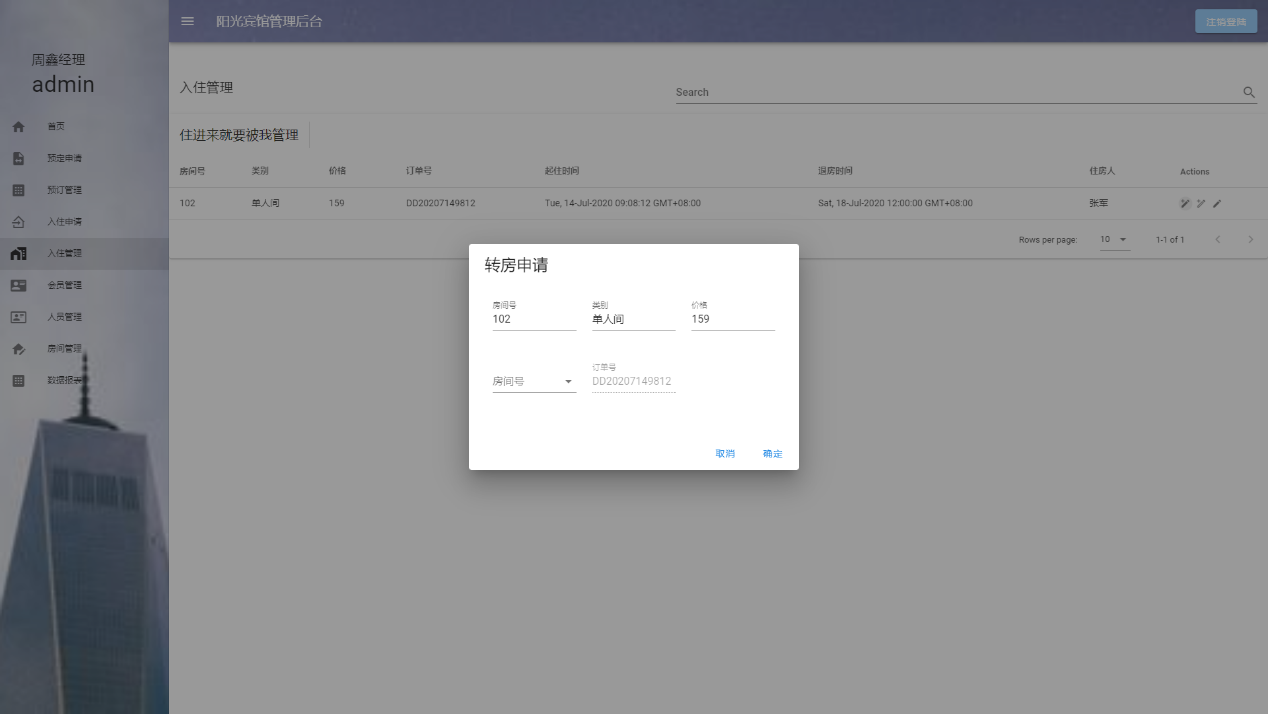
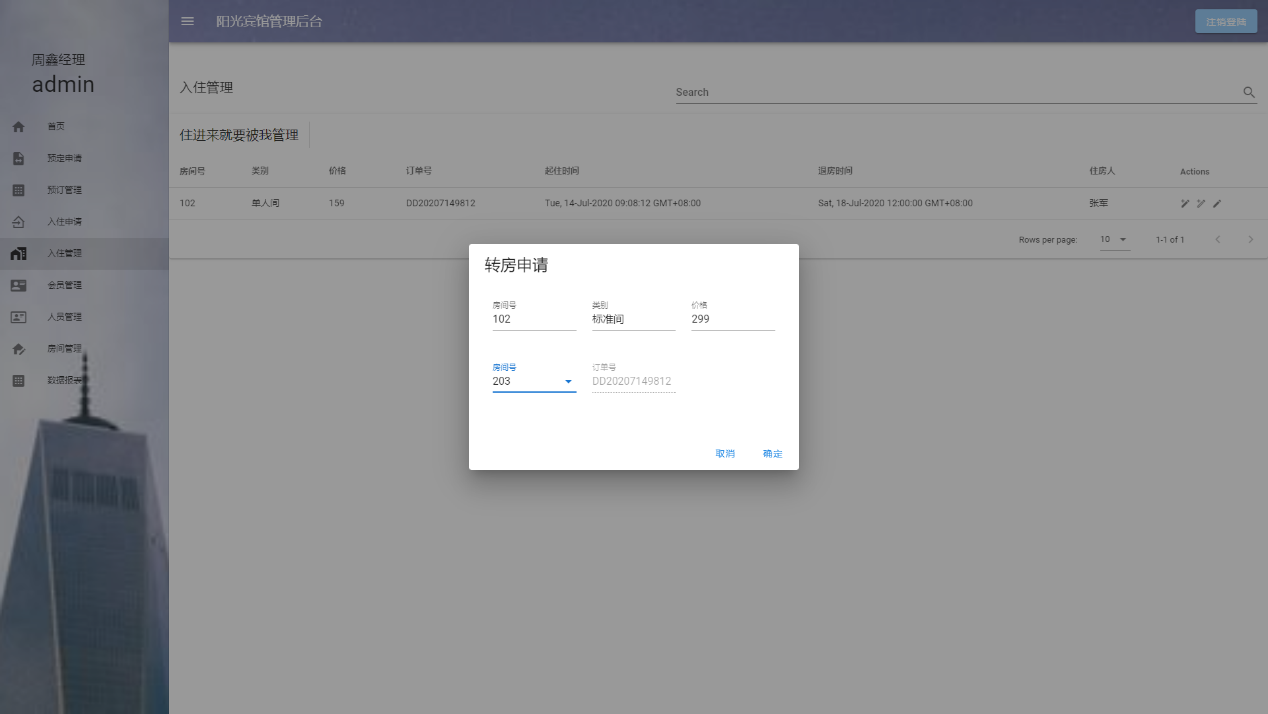
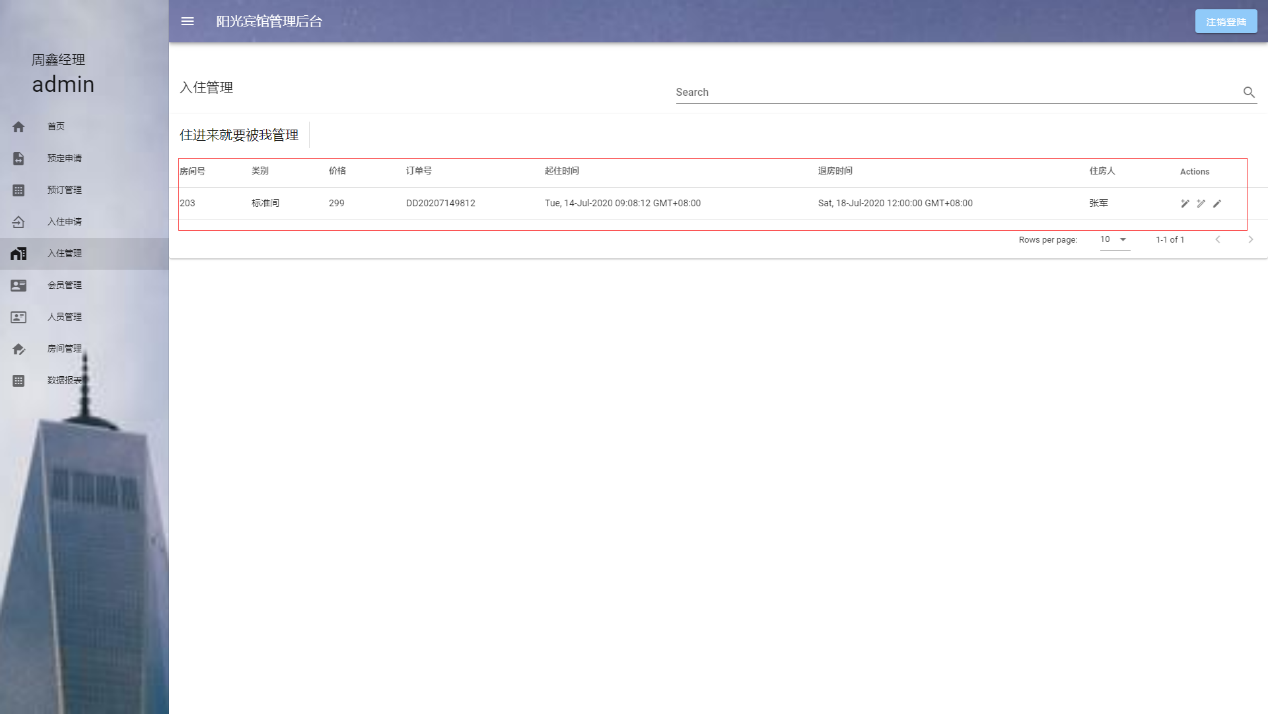
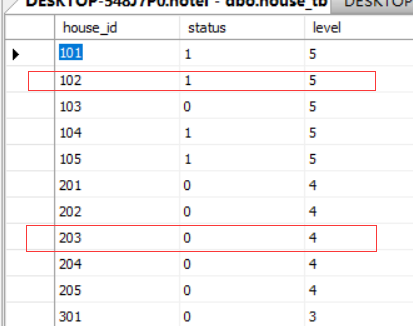
  



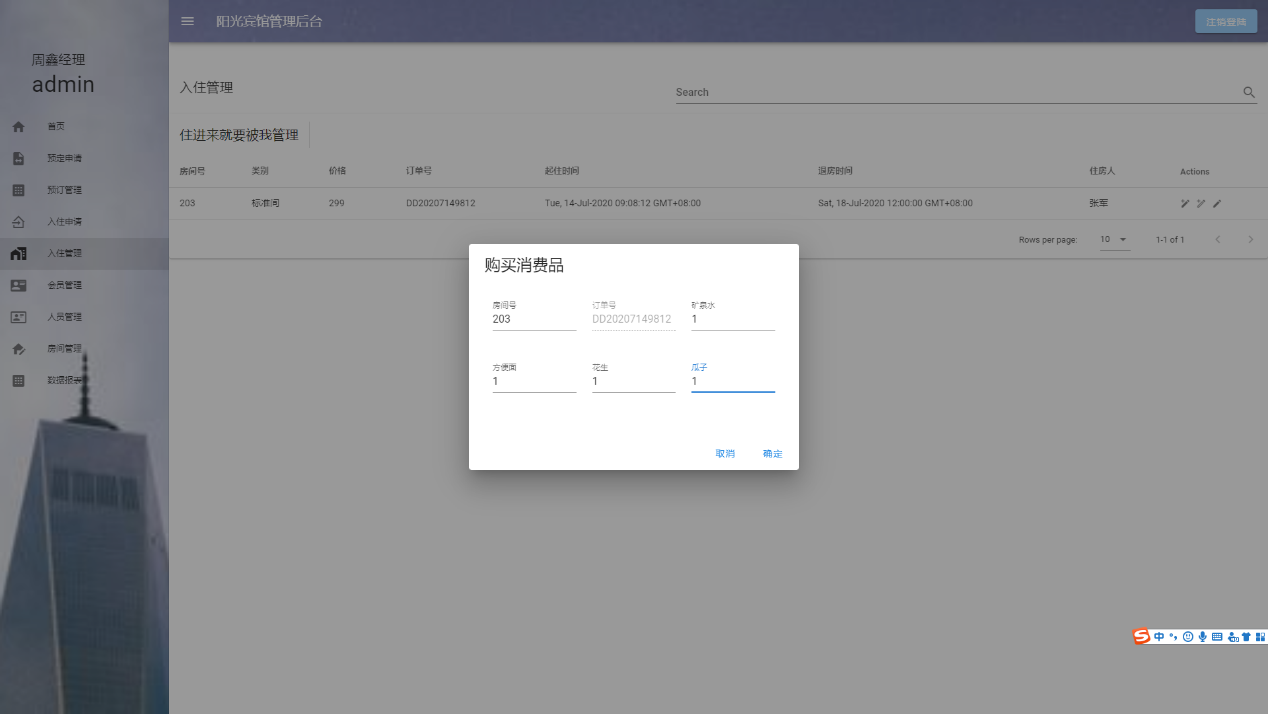
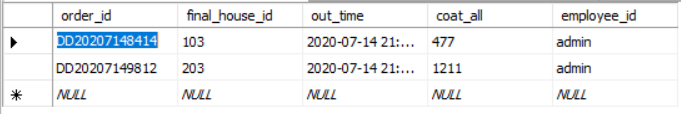
1. 入住登记测试，提交入住登记，房间状态标识为已入住（1），首页将不显示此房间，订单标识为已入住（1），级别相应剩余房间数减1     
2. 退房测试，所住房间状态标识为未入住（0），级别相应剩余房间数加1

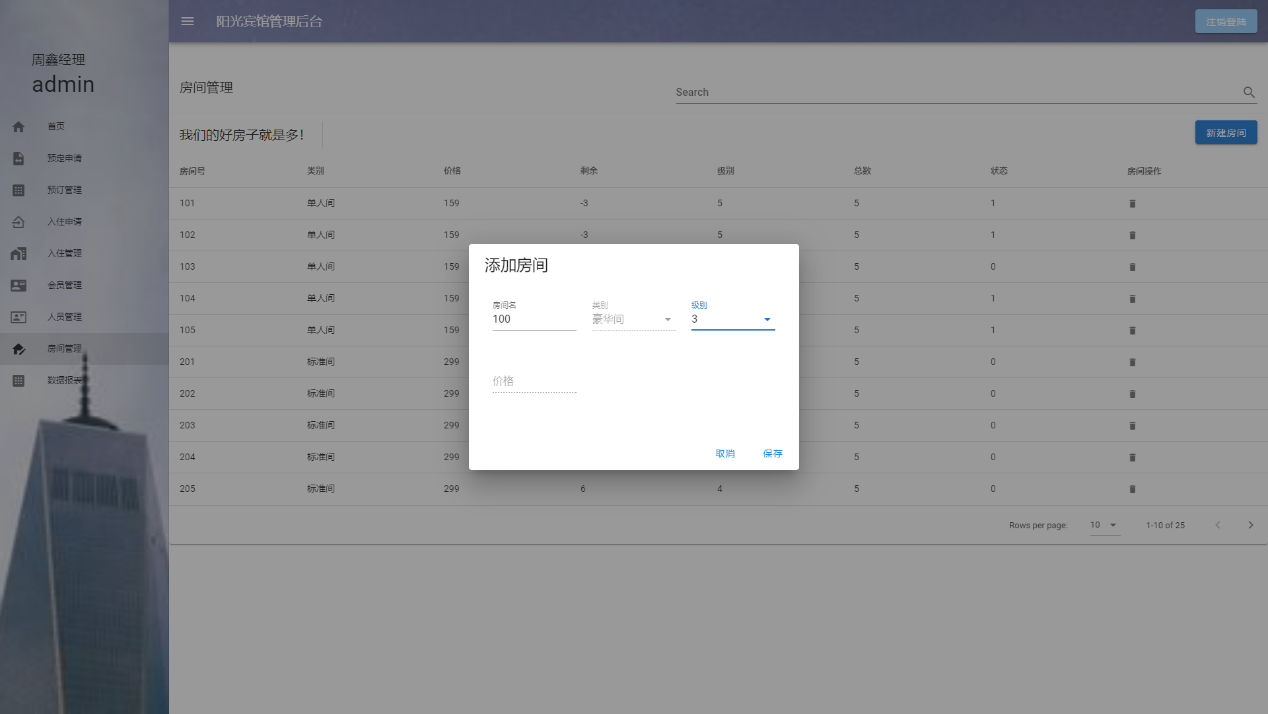
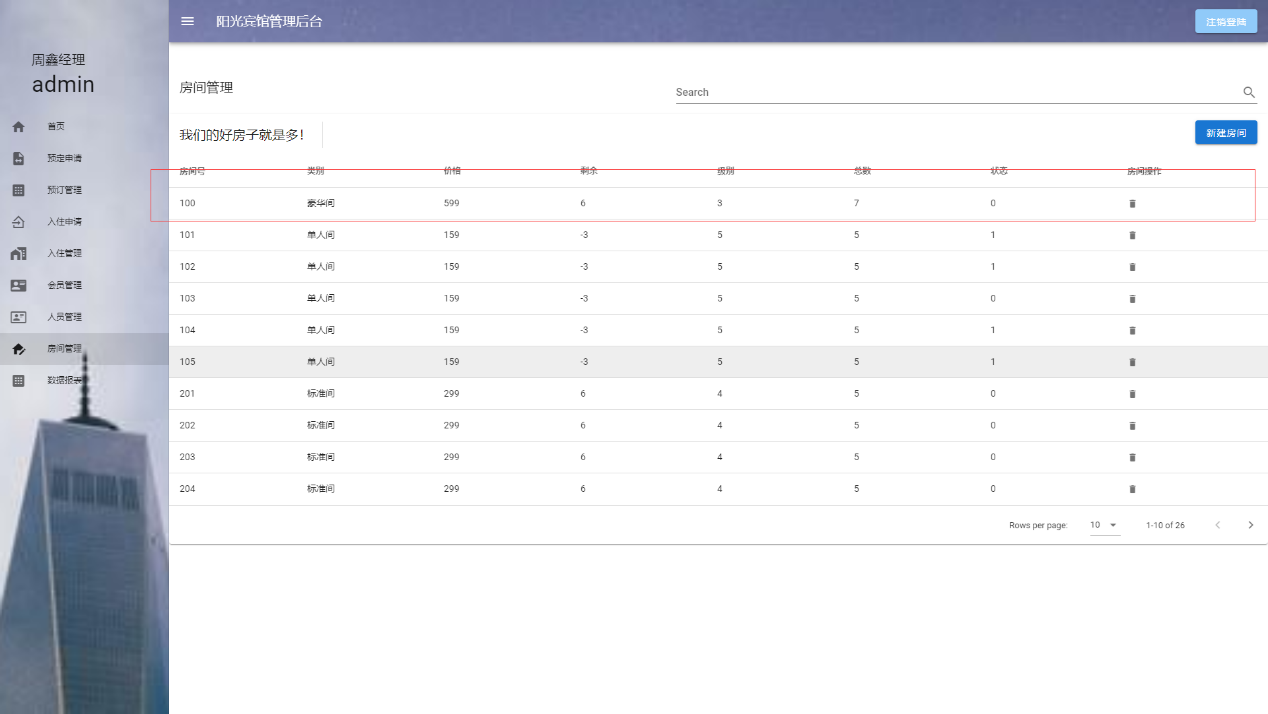
1. 转房测试，选择转入房间号，转前房间状态标识为未入住（0），级别相应剩余房间数加1，转后房间状态标识为已入住（1），级别相应剩余房间数减1

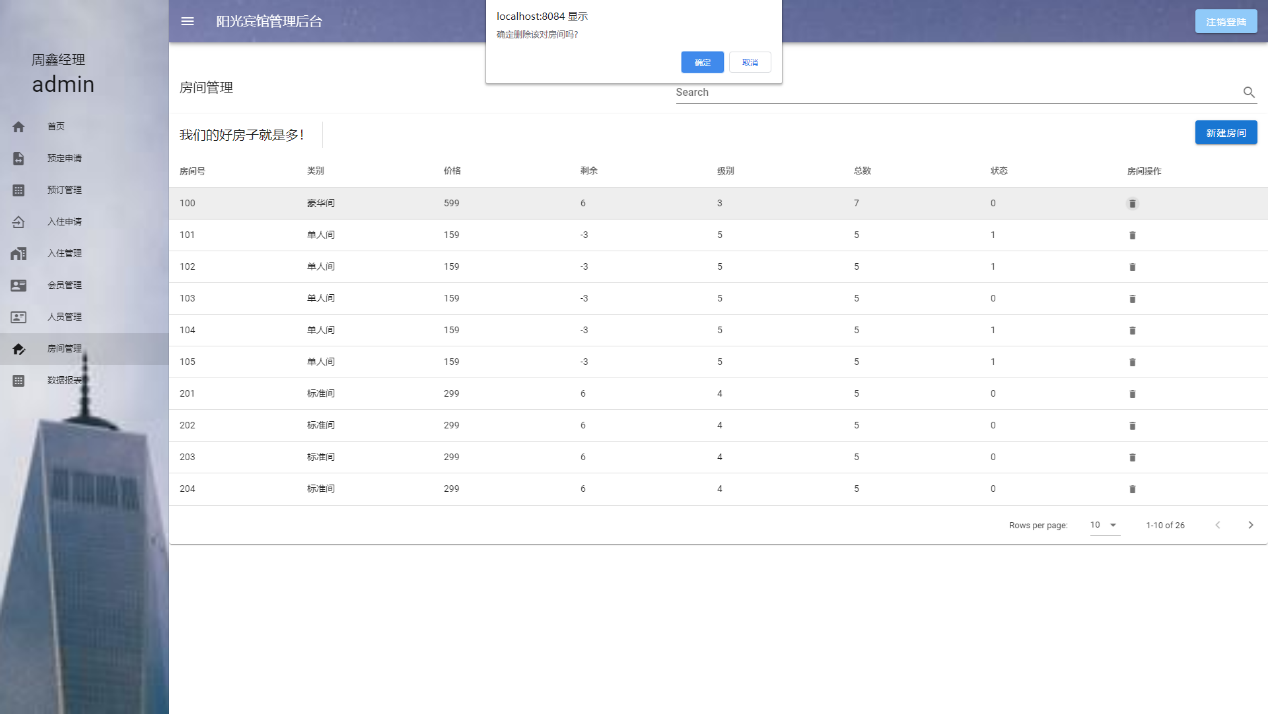
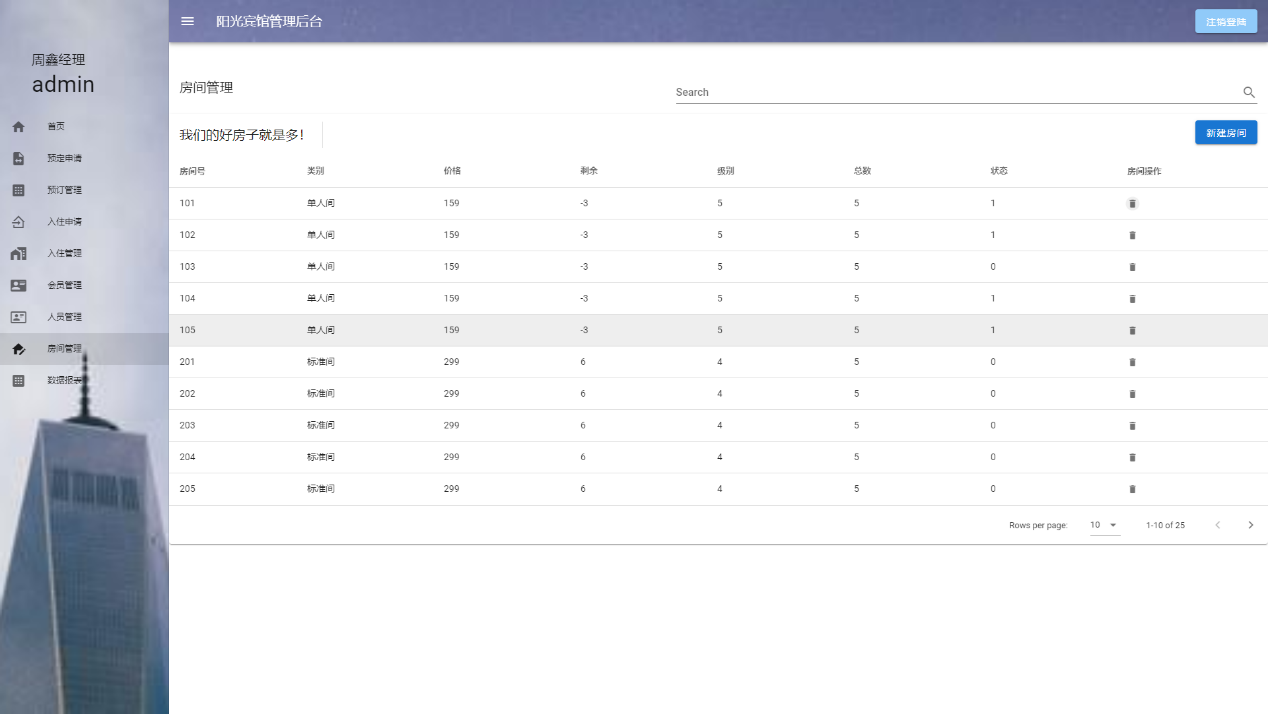
1. 添加消费品记录和退房结算消费品

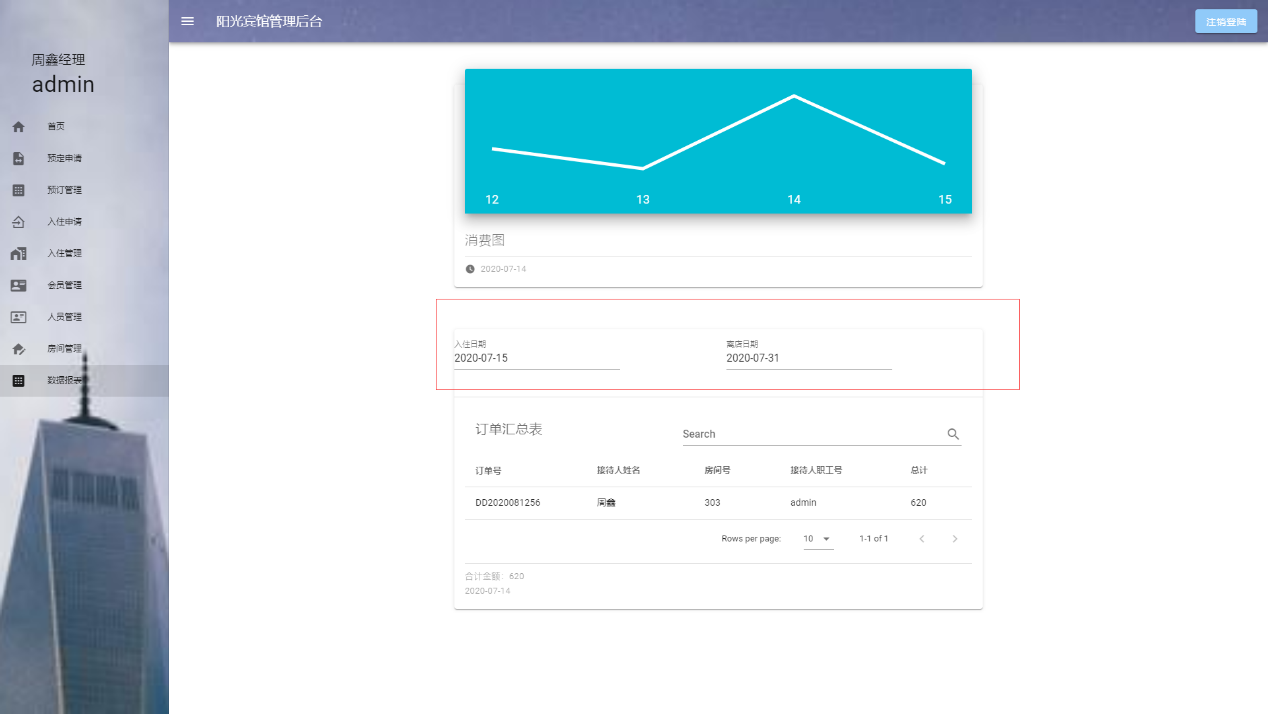
1. 添加房间测试，级别相应剩余房间数加1，级别相应房间总数加1

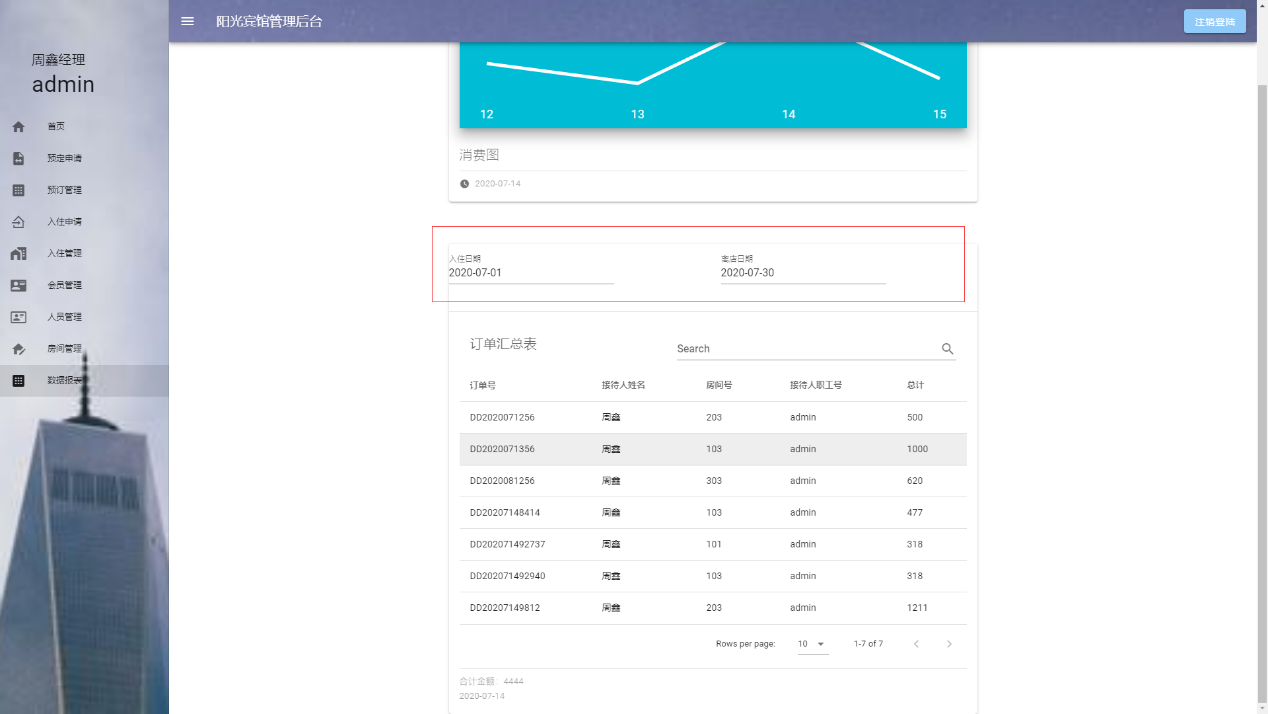
 

1. 删除房间测试，级别相应剩余房间数减1，级别相应房间总数减1

1. 订单查询测试，选取时间段，显示该时间段已结束的订单，计算销售额





# 6作业总结

（写出实践过程中的困难、用到的知识、心得体会）

我们组做的宾馆管理系统是运用。。。。语言和SQL Server 2008 数据库开发的，目前项目要求的功能基本都可实现。

这个系统也有些不足之处，比如界面外观还可以继续优化，整个系统的并发性如何，用户信息的保密性和安全性还没有考虑。

通过这次数据库实践，我们学到了很多知识，受益匪浅。技术上，我们巩固了数据库的相关知识，并将其运用，理论与实践相结合，达到了对所学知识的灵活运用。团队精神上，我们有很大的提升，我们可以有效的和成员沟通，出现大的问题也可以及时解决。

# 附 录

（带注释的源程序。只要经典代码、核心代码。带注释的存储过程和触发器的代码）

1. 触发器代码如下

--- 创建触发器修改房间剩余数和房间状态

CREATE TRIGGER in\_tb\_insert

ON in\_tb

AFTER insert

AS

DECLARE @house int, @stat int

SELECT @house=house\_id, @stat = status from inserted

BEGIN

if @stat = 1

BEGIN

update house\_level\_tb set remain = remain - 1 where house\_level\_id in (select level from house\_tb where house\_id = @house)

update house\_tb set status = 1 where house\_id = @house

END

END

---退房改变房间状态

CREATE TRIGGER out\_tb\_house

ON out\_tb

AFTER insert

AS

DECLARE @house int

DECLARE @id varchar(16)

SELECT @house=final\_house\_id, @id=order\_id from inserted

BEGIN

update house\_tb set status = 0 where house\_tb.house\_id = @house

update house\_level\_tb set remain = remain + 1 where house\_level\_id in (select level from house\_tb where house\_id = @house)

update in\_tb set status = 0 where order\_id = @id

END

1. 源程序代码如下

*//提交订单* @PostMapping("/putDD")  
 public Integer InsertDD(@RequestBody In\_tb DDMessage){  
*// 设置订单号* DDMessage.setOrderId(getOrder\_id());  
 *//设置状态* DDMessage.setStatus(1);  
*// 设置退房时间日期* Calendar date = Calendar.*getInstance*();  
 date.setTime(DDMessage.getOutTime());  
 Calendar cal4 = Calendar.*getInstance*();  
 cal4.set(date.get(Calendar.*YEAR*),date.get(Calendar.*MONTH*),date.get(Calendar.*DATE*),12,0,0);  
 Date date1 = cal4.getTime();  
 DDMessage.setOutTime(date1);  
*// 起* Calendar getHMS = Calendar.*getInstance*();  
 getHMS.setTime(new Date());  
 System.*out*.println("小时"+getHMS.get(Calendar.*HOUR*));  
 Calendar date3 = Calendar.*getInstance*();  
 date3.setTime(DDMessage.getInTime());  
 Calendar cal5 = Calendar.*getInstance*();  
 cal5.set(date3.get(Calendar.*YEAR*),date3.get(Calendar.*MONTH*),date3.get(Calendar.*DATE*),getHMS.get(Calendar.*HOUR*),getHMS.get(Calendar.*MINUTE*),getHMS.get(Calendar.*SECOND*));  
 Date date2 = cal5.getTime();  
 DDMessage.setInTime(date2);  
 System.*out*.println(DDMessage.getInTime()+"入住时间");  
  
 System.*out*.println("订单信息"+DDMessage);  
 int res = in\_tbMapper.insert(DDMessage);  
 *//返回1表示成功* if(res == 1){  
 return 1;  
 }else {  
 return 0;  
 }  
 }

对应SQL语句：

<insert id="insert" parameterType="com.our.hotel.pojo.In\_tb" >  
 insert into in\_tb (order\_id, name, gender,   
 tel, id\_card, house\_id, in\_time, out\_time, pledge,  
 employee\_id, status)  
 values (#{orderId,jdbcType=VARCHAR}, #{name,jdbcType=VARCHAR}, #{gender,jdbcType=INTEGER},   
 #{tel,jdbcType=VARCHAR}, #{idCard,jdbcType=VARCHAR}, #{houseId,jdbcType=INTEGER}, #{inTime,jdbcType=TIMESTAMP},  
 #{outTime,jdbcType=TIMESTAMP}, #{pledge,jdbcType=INTEGER},  
 #{employeeId,jdbcType=VARCHAR}, #{status,jdbcType=INTEGER})  
</insert>

*//获得可以预定的房间列表* @PostMapping("/putyuDT")  
 public List putyuDT(@RequestBody YuDingTime yuDingTime){  
 System.*out*.println("输出预定入住时间"+yuDingTime.getStart()+"==========="+yuDingTime.getEnd());  
 Calendar c = Calendar.*getInstance*();  
 c.setTime(yuDingTime.getStart());  
 String start = c.get(Calendar.*YEAR*)+"-"+(c.get(Calendar.*MONTH*)+1)+"-"+c.get(Calendar.*DATE*)+" "+"14"+":"+c.get(Calendar.*MINUTE*)+":"+c.get(Calendar.*SECOND*);  
 Calendar c2 = Calendar.*getInstance*();  
 c2.setTime(yuDingTime.getEnd());  
 String end = c2.get(Calendar.*YEAR*)+"-"+(c2.get(Calendar.*MONTH*)+1)+"-"+c2.get(Calendar.*DATE*)+" "+"12"+":"+c2.get(Calendar.*MINUTE*)+":"+c2.get(Calendar.*SECOND*);  
 System.*out*.println(start+"===="+end);  
 yuDingTime.setStarts(start);  
 yuDingTime.setEnds(end);  
  
 List<HouseID> houseIDS = in\_tbMapper.selectAllYuDingHouseID(yuDingTime);  
 System.*out*.println(houseIDS);  
 ArrayList<Integer> arrayList = new ArrayList<>();  
 for (HouseID houseID : houseIDS) {  
 arrayList.add(houseID.getHouseID());  
 }  
 return arrayList;  
 }

对应SQL语句：

<select id="selectAllYuDingHouseID" resultType="HouseID" parameterType="YuDingTime">  
 select h\_tb.house\_id from house\_tb h\_tb where  
 h\_tb.house\_id not in (select house\_id from in\_tb  
 <where>  
*<!-- <if test="start != null and end != null">-->* in\_time between #{starts} and #{ends}  
 </where>)  
 </select>

*//提交预定订单* @PostMapping("/putyuDD")  
 public Integer putYuDD(@RequestBody In\_tb DDMessage){  
 *// 设置订单号* DDMessage.setOrderId(getOrder\_id());  
 *// 设置状态* DDMessage.setStatus(2);  
*// 设置退房时间日期* Calendar date = Calendar.*getInstance*();  
 System.*out*.println("入住时间"+DDMessage.getInTime()+"=-=============退房时间"+DDMessage.getOutTime());  
 date.setTime(DDMessage.getOutTime());  
 Calendar cal4 = Calendar.*getInstance*();  
 cal4.set(date.get(Calendar.*YEAR*),date.get(Calendar.*MONTH*),date.get(Calendar.*DATE*),12,0,0);  
 Date date1 = cal4.getTime();  
 DDMessage.setOutTime(date1);  
  
 *// 起* Calendar getHMS = Calendar.*getInstance*();  
 getHMS.setTime(new Date());  
 System.*out*.println("小时"+getHMS.get(Calendar.*HOUR*));  
 Calendar date3 = Calendar.*getInstance*();  
 date3.setTime(DDMessage.getInTime());  
 Calendar cal5 = Calendar.*getInstance*();  
 cal5.set(date3.get(Calendar.*YEAR*),date3.get(Calendar.*MONTH*),date3.get(Calendar.*DATE*),getHMS.get(Calendar.*HOUR*),getHMS.get(Calendar.*MINUTE*),getHMS.get(Calendar.*SECOND*));  
 Date date2 = cal5.getTime();  
 DDMessage.setInTime(date2);  
 System.*out*.println(DDMessage.getInTime()+"入住时间");  
  
 System.*out*.println("这里是预定订单提交"+DDMessage);  
 int res = in\_tbMapper.insertYuDD(DDMessage);  
 *//返回1表示成功* if(res == 1){  
 return 1;  
 }else {  
 return 0;  
 }  
 }

对应SQL语句：  
<insert id="insertYuDD" parameterType="com.our.hotel.pojo.In\_tb" >  
 insert into in\_tb (order\_id, name, gender,  
 tel, id\_card, house\_id, out\_time, in\_time, pledge,  
 employee\_id, status)  
 values (#{orderId,jdbcType=VARCHAR}, #{name,jdbcType=VARCHAR}, #{gender,jdbcType=INTEGER},  
 #{tel,jdbcType=VARCHAR}, #{idCard,jdbcType=VARCHAR}, #{houseId,jdbcType=INTEGER},  
 #{outTime,jdbcType=TIMESTAMP},#{inTime,jdbcType=TIMESTAMP}, #{pledge,jdbcType=INTEGER},  
 #{employeeId,jdbcType=VARCHAR}, 2)  
</insert>

*//返回预定订单列表* @GetMapping("/getyudinghotel")  
 public List<Inhotel> getyudinghotel(){  
 List<Inhotel> inhotels = inhotelMapper.selectYudDingInOrder();  
*// System.out.println(ShowMessageController.getGMT(inhotels.get(0).getStart()));* for (Inhotel inhotel : inhotels) {  
 inhotel.setStarts(ShowMessageController.*getGMT*(inhotel.getStart()));  
 inhotel.setEnds(ShowMessageController.*getGMT*(inhotel.getEnd()));  
 }  
 return inhotels;  
 }

对应SQL语句：

<select id="selectYudDingInOrder" resultMap="DDH">  
 select in\_tb.house\_id houseID, order\_id, house\_level\_tb.house\_level houesType, house\_level\_tb.money price, in\_tb.in\_time, in\_tb.out\_time,in\_tb.name from in\_tb, house\_level\_tb, house\_tb where  
 in\_tb.status = 2 and in\_tb.house\_id = house\_tb.house\_id and house\_tb.level = house\_level\_tb.house\_level\_id  
</select>

*//确认预定订单* @PostMapping("/qurenOrder")  
 public Integer qurenOrder(@RequestBody Inhotel inhotel){  
 System.*out*.println(inhotel);  
 int i = inhotelMapper.querenOrderStatus(inhotel);  
 return i;  
 }

对应SQL语句：

<update id="querenOrderStatus" parameterType="Inhotel">  
 update in\_tb set status = 1 where order\_id = #{order\_id}  
</update>

*//取消订单* @PostMapping("/quxiaoOrder")  
 public Integer quxiaoOrder(@RequestBody Inhotel inhotel){  
 int i = inhotelMapper.quxiaoOrderStatus(inhotel);  
 return i;  
 }

对应SQL语句：

<delete id="quxiaoOrderStatus" parameterType="Inhotel">  
 delete from in\_tb where order\_id = #{order\_id}  
</delete>

*//返回房间对应的类型和价格* @PostMapping("/gettp")  
 public House\_level\_tb gettp(@RequestBody HouseID id){  
 House\_level\_tb house\_level\_tb = house\_level\_tbMapper.selectByhouseID(id.getHouseID());  
 return house\_level\_tb;  
 }

对应SQL语句：  
<select id="selectByhouseID" resultType="House\_level\_tb">  
 select house\_level\_tb.\* from house\_level\_tb, house\_tb where house\_tb.house\_id = #{houseLevelId} and house\_level\_tb.house\_level\_id = house\_tb.level  
</select>

@GetMapping("/getData")  
 public ShuJuBaoBiao getData(){  
 ShuJuBaoBiao shuJuBaoBiao1 = new ShuJuBaoBiao();  
 List<ShuJuBaoBiao> shuJuBaoBiaos = out\_tbMapper.selectDataByMothDay();  
 List<Integer> dayList = new ArrayList<>();  
 List<Float> moneyList = new ArrayList<>();  
 for (ShuJuBaoBiao shuJuBaoBiao : shuJuBaoBiaos) {  
 dayList.add(shuJuBaoBiao.getDay());  
 moneyList.add(shuJuBaoBiao.getMoney());  
 }  
 shuJuBaoBiao1.setDayList(dayList);  
 shuJuBaoBiao1.setMoneyList(moneyList);  
 return shuJuBaoBiao1;  
 }

对应SQL语句：

<select id="selectDataByMothDay" resultType="ShuJuBaoBiao">  
 select *sum*(coat\_all) money,day(out\_time) day from out\_tb  
 where out\_time &gt;= (select convert(datetime,convert(varchar(8),getdate(),120)+'01',120)) and out\_time &lt;=(select dateadd(day,-1,convert(datetime,convert(varchar(8),dateadd(month,1,getdate()),120)+'01',120))) group by day(out\_time)  
</select>

@GetMapping("/getAllOrder")  
 public List<Out\_tb> getAllOrder(){  
 List<Out\_tb> out\_tbs = out\_tbMapper.selectAllOutOrder();  
 return out\_tbs;  
 }

对应SQL语句：

<select id="selectAllOutOrder" resultType="Out\_tb">  
 select out\_tb.\* , name from out\_tb, employee\_tb where out\_tb.employee\_id = employee\_tb.employee\_id;  
</select>

@PostMapping("/getOrderByTime")  
 public List<Out\_tb> getOrderByTime(@RequestBody YuDingTime startEnd){  
  
 System.*out*.println("输出选择的时间段"+startEnd.getStart()+"==========="+startEnd.getEnd());  
 Calendar c = Calendar.*getInstance*();  
 c.setTime(startEnd.getStart());  
 String start = c.get(Calendar.*YEAR*)+"-"+(c.get(Calendar.*MONTH*)+1)+"-"+c.get(Calendar.*DATE*)+" "+"0"+":"+c.get(Calendar.*MINUTE*)+":"+c.get(Calendar.*SECOND*);  
 Calendar c2 = Calendar.*getInstance*();  
 c2.setTime(startEnd.getEnd());  
 String end = c2.get(Calendar.*YEAR*)+"-"+(c2.get(Calendar.*MONTH*)+1)+"-"+c2.get(Calendar.*DATE*)+" "+"0"+":"+c2.get(Calendar.*MINUTE*)+":"+c2.get(Calendar.*SECOND*);  
 System.*out*.println(start+"===="+end);  
 startEnd.setStarts(start);  
 startEnd.setEnds(end);  
 List<Out\_tb> out\_tbs = out\_tbMapper.selectByTime(startEnd);  
 return out\_tbs;  
 }

对应SQL语句：

<select id="selectByTime" parameterType="YuDingTime" resultType="Out\_tb">  
 select out\_tb.\*, name from out\_tb, employee\_tb  
 <where>out\_time between #{starts} and #{ends} and out\_tb.employee\_id = employee\_tb.employee\_id;  
 </where>  
</select>

小组成员任务清单

|  |  |  |
| --- | --- | --- |
| **组员** | **任务分工** | **评分** |
| **朱行军** |  |  |
| **张建军** |  |  |
| **马瑞** |  |  |
| **谭清流** |  |  |