# 酒店预定系统详细设计报告

题目：**酒店预订系统**

|  |  |  |  |
| --- | --- | --- | --- |
| 学号 | 姓名 | 分工 | 小组 |
| 2017214817 | 王琛 | 后端开发、前端设计、系统设计 | 组长 |
| 2017214806 | 郑海洋 | 前端设计、需求分析、测试报告 | 组员 |
| 2017214807 | 陈之昂 | 前端设计、详细设计报告 | 组员 |
| 2017214820 | 陈李欣 | 前端设计、需求分析、系统设计报告 | 组员 |
| 2017214822 | 董飞飞 | 后端开发、系统设计报告 | 组员 |

**版本修订记**

|  |  |  |
| --- | --- | --- |
| 版本标识 | 注 释 | 作 者 |
| 1.0 | 初始版本 | 小组成员 |
| 1.1 | 加入管理员功能 | 小组成员 |
| 1.2 | 前端加入关于和主页 | 小组成员 |
| 1.3 | 进行检查与修改 | 小组成员 |
| 1.4 | 取消订单功能失误，已修改 | 小组成员 |
| 1.5 | 服务器时区出现问题，已经修改 | 小组成员 |

目录

**1.引言 3**

1.1编写目的 3

1.2背景 3

1.3参考资料 3

1.5术语与缩略语 4

**2.系统总体概述 4**

2.1任务与目标 4

2.2运行环境 5

2.3条件与限制 5

2.5详细设计方法与方式 6

**3.系统详细需求分析 6**

3.1功能和性能 6

3.1.1详细功能需求分析 6

3.1.2详细性能需求分析 7

3.2详细系统运行环境及限制条件分析接口需求分析 8

**4.总体方案的确认 8**

4.1系统总体结构的确认 8

4.2系统内部详细结构的确认 11

4.1.1技术选型 11

4.1.2数据持久层设计 11

**5.系统功能模块详细设计 13**

5.1注册模块 13

5.1.1功能 13

5.1.2性能 14

5.1.3输入项目 14

5.1.4输出项目 15

5.1.5算法 15

5.1.6程序逻辑 16

5.2登录模块 16

5.2.1功能 17

5.2.2性能 17

5.2.3输入项目 18

5.2.4输出项目 18

5.2.5算法 18

5.2.6程序逻辑 18

5.2.7限制条件 18

5.3查询房间模块 19

5.3.1功能 19

5.3.2性能 20

5.3.3输入项目 20

5.3.4输出项目 20

5.3.5算法 21

5.3.6程序逻辑 21

5.4订单控制模块 21

5.4.1功能 21

5.4.2性能 22

5.4.3输入项目 22

5.4.4输出项目 22

5.4.5算法 23

5.4.6程序逻辑 23

5.5管理员控制模块 23

5.5.1功能 22

5.5.2性能 23

5.5.3输入项目 23

5.5.4输出项目 24

5.5.5算法 24

5.5.6程序逻辑 25

**6.数据库详细设计 26**

6.1设计依据 26

6.2数据库种类及特点 26

6.3数据库逻辑结构 26

6.4数据字典 27

6.5数据库安全 29

**一．引言**

* 1. **编写目的**

本系统设计报告编制的目的是说明一个软件系统各个层次中的每个程序（每个模块或子程序）和数据库系统的设计考虑，为程序员编码提供依据。

本系统设计报告对概要设计进一步细化然后再集成，是具体的实现细节。是“程序”的蓝图，确定每个模块采用的算法、数据结构、接口的实现、属性、参数。在本详细设计报告中详细文档中最重要的部分是模块的流程图、状态图、局部变量及相应的文字说明等。

* 1. **背景**

随着时代的发展，互联网的也在不断的进步，人民的生活水平大大提高。同样的，交通设施的完善也导致人们在节假日期间大量外出，而在人们在外出的时候往往需要去租住一个酒店，在这个旅游旺季，如果不预定而在当天直接去酒店租住的话，可能并没有空闲的房间，对于出行体验造成极大的不便。而且，用户可能会对当地的酒店并不了解，无法找到符合自己心里价位的酒店，如果找到超出用户心理预期的酒店，那么可能会影响用户的剩余时间的游玩规划。酒店预订系统油然而 生，通过酒店预订，可以完美的解决上述问题。

* 1. **参考资料**

软件需求工程原理和方法，金芝编写科学出版社

需求分析与系统设计 LESZEK.A 马素霞 等译

百度百科

维基百科

百度文库

* 1. **术语定义及说明**

Active：激活状态，即正在处理中或处理完成。

Cancel：取消状态，该动作以及取消或无效。

Select： 数据库查询

Update：数据库更新

Insert： 数据库插入

**二．设计概述**

**2.1. 任务和目标**

本酒店预订管理系统，可以让用户足不出户，选择适合自己的酒店。在网页中，用户可以查看每个酒店的详情，并根据自己的目的地以及需求、经济状况等内容来选择适合的酒店。进入酒店主页之后可以看到该酒店中不同的房间类型以及房间详情包括是否包含早餐等等。如有特殊情况，用户可以通过本系统进行房间的退订。用户可以在本系统之中查看、编辑自己的个人信息、查看自己的订单、与酒店交互，而酒店则通过管理员模块与用户互动。

管理员模块有两个功能，就是他们能够进行酒店信息的添加和套房信息的添加，因为管理员的存在，使得未加入本系统的酒店能够加入本系统与我们合作。同时管理员也能够对酒店信息和套房信息进行删除。

**2.2运行环境概述**

硬件环境：

Cpu：i5及以上，笔记本电脑，可以连接互联网

软件环境：

操作系统：windows 10

应用软件：idea

浏览器 ：谷歌浏览器

数据库软件：mysql 8.0

**2.3条件与限制**

由于时间原因，本次设计并没有选取酒店的实际照片，采取的是网络上的一些酒店的照片，并且因为用户是自主注册的，所以并没有删除功能，即不能删除用户和酒店的房间。对于订单，同样的因为时间的原因只把订单的状态由active修改为cancel状态而不是清空。因为本系统搭建在远端服务器上，所以没有互联网的用户无法访问。如果用户忘记了密码，我们只提供“忘记密码？”此选项，碍于时间原因但是并没有实现向用户注册时所输入的邮箱发送验证码来进行密码的重置工作。

**2.4详细设计方法与工具**

本文档采用E-R图方法、状态图方法、时序图方法、数据流程图、业务流程图方法。

**三．系统详细需求分析**

**3.1.1详细功能需求分析**

根据对任务及目标的分析可以得出需要的功能为：

**登陆注册功能**：用户根据自己是否是新用户，若是，进行注册；若不是，直接登陆即可

**个人信息查看和修改功能：**用户可以在个人中心那里查看自己的个人信息，并且能够修改自己的个人信息

**个人订单查看功能：**用户可以在个人中心那里查看自己的个人订单

**酒店总览：**用户可以在主页查看系统收录的所有酒店的信息

**酒店筛选：**用户可以在主页输入旅行的目的地，从而筛选出只位于当地的酒店

**酒店具体信息查询：**用户可以在主页选择一个酒店，然后点击具体**信息**即可查看该酒店的具体信息了。

**酒店预订：**用户选择一个酒店查看具体详情之后，弱若对酒店房间的价格感到满意，并且酒店存在空闲旅馆，就可以进行预订

**酒店退订功能：**用户预订之后，若是因为有其他事务或者其他原因，即可进行退订

**酒店信息添加功能：**管理员能够根据要加入本系统的具体酒店的消息，来将本酒店加入到本系统中，并且将酒店的详细信息录入。

**套房信息添加功能：**管理员能够根据要加入本系统的具体酒店的套房消息，来添加新的套房信息进入本酒店 。

**酒店信息修改功能：**管理员能够修改已经录入的酒店信息,修改信息也要符合规定，哪些数据不能为空，注意什么格式等等。

**套房信息修改功能：**管理员能够修改已经录入的套房信息，修改信息也要符合规定，哪些数据不能为空，注意什么格式等等。

**3.1.2详细性能需求分析**

**可靠性和可移植性：**为了满足不同客户端用户的需要，本系统满足不同客户端用户登录的需求，不管是用手机登录还是电脑登录都简介美观。

**开放性需求：**为了满足客户的更多需求，不同ip可以同时登陆同一个账号且不会出现顶号的现象

**系统安全性需求：**在系统的安全性需求上，密码上采取了密文录入，而不是明文，确保了系统的安全性

只有授权的用户才可以修改信息，即只有用户自己才可以修改自己的信息，旁人都无法修改

**现有资源利用需求：**充分利用现有的资源，将数据库搭建在服务器上，使用户可以多地登陆，足不出户即可使用本系统。

**3.2 详细系统运行环境及限制条件分析接口需求分析**

**软件接口：**

描述该软件产品与其它外部组件的连接，这些外部组件必须 明确他们的名称和版本号以资识别，可能的外部组件包括：

● 操作系统；win10

● 数据库；sql sever

● 工具；idea pingendo

**硬件接口：**

● 通过usb接口进行硬件之间的连接

**通信接口：**

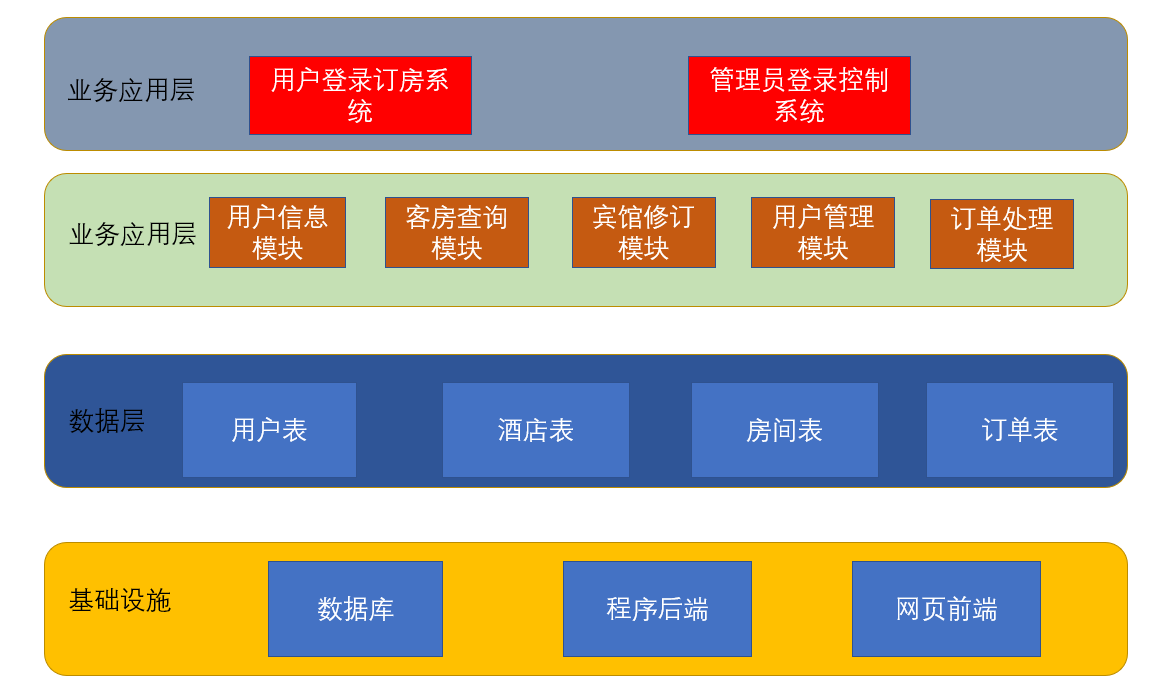
● WEB浏览器；

● 网络通讯标准或者协议；

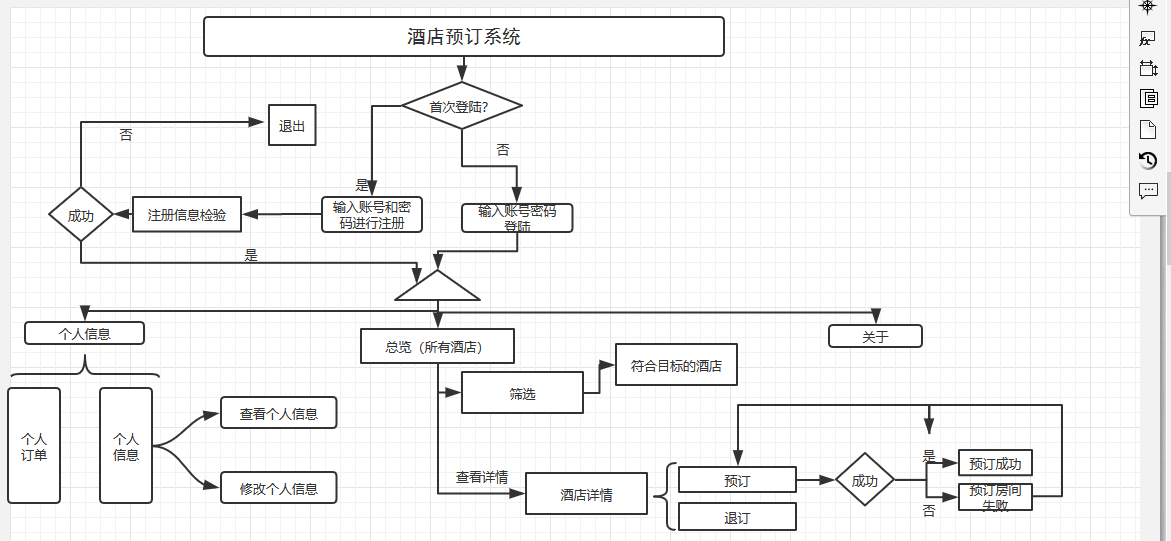
**四．总体方案的确认**

**4.1系统总体结构的确认**

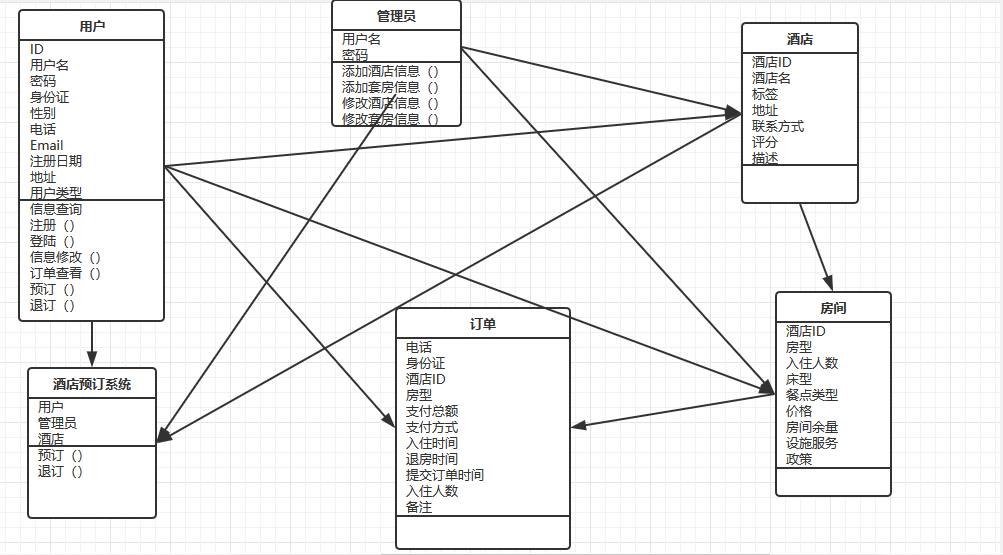
**系统组成、逻辑结构图：**



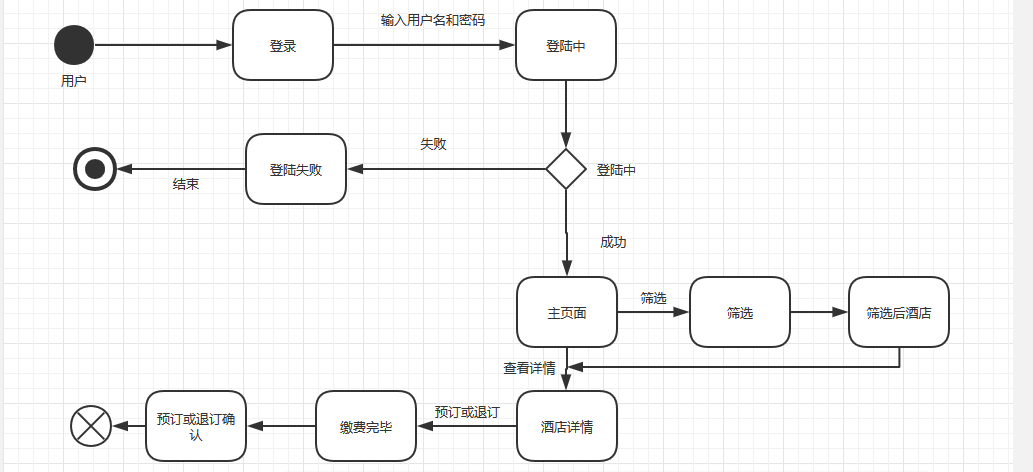
**系统流程图**

****

**UML类图**

****

**UML状态图**

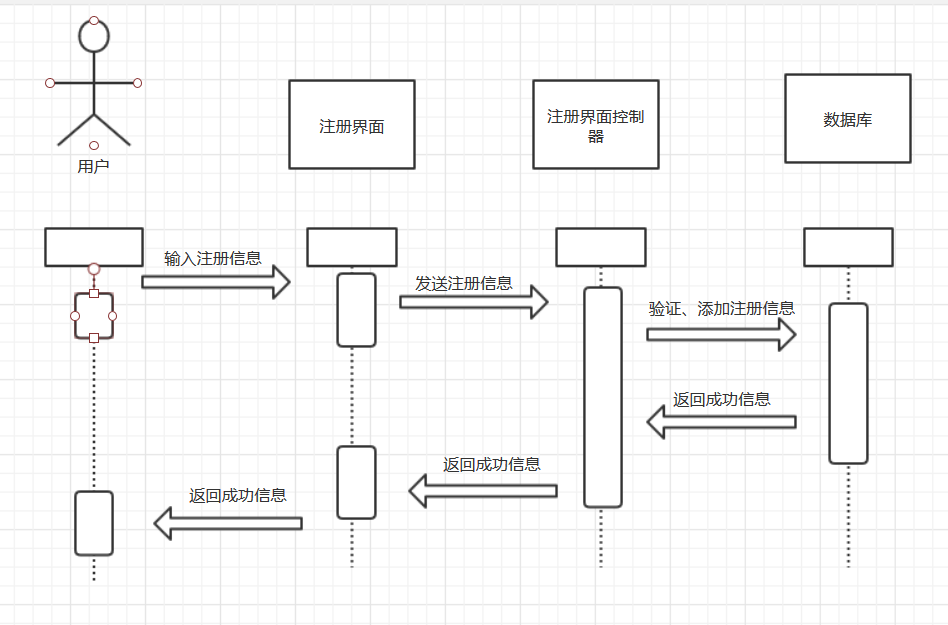
****

**4.2系统内部详细结构的确认**

**4.2.1 技术选型**  
  使用Maven进行第三方jar包管理是非常流行的方式，它能够帮助我们解决大部分第三方jar包的相互依赖问题，因此，为了简化环境配置的复杂性，我们整合了Maven作为项目的开发包管理器  
 **4.2.2 数据持久层设计**  
  **4.2.2.1 技术选型**  
  为了简化开发流程，加强数据访问安全性，我们使用了Mybatis的动态代理技术、映射技术以及动态sql技术。第一点，虽然Mybatis本质使用JDBC技术，但是Mybatis的动态sql技术有效的减少了sql注入攻击的危险。第二，映射技术的使用消除了数据库元组与JavaBean之间的阻碍（属性命名规则不相同问题，比如在数据库中用户名属性被命名user\_name，而JavaBean中将属性命名位userName）。第三，动态代理技术让我们只用专注于接口的定义以及sql语句的编写，而不用去实现接口的实现类，极大的减少了代码量。  
  **4.2.2.2 整合Mybatis**  
   本项目中，因为还使用了Spring框架，我们将Mybatis的核心类SqlSessionFactory交给了SpringIOC容器管辖。我们仅配置了对数据持久层的接口扫描以及编写了对应接口的xml文件。  
**4.2.2.3 数据持久层映射**  
   根据上文中数据库的设计，本系统中一共存在6个实体。因此我们分别编写了6个对应的JavaBean实体，并将它们放入model包中保存，在对应的xml文件中，我们根据Mybatis的官方说明，设计了6个XxxxMap来对应JavaBean中的各个属性  
**4.2.2.4 数据库事务管理**  
   我们定义了一个Mybatis提供的事务管理器，并将它交给IOC容器管理，在每个需要的业务函数中我们使用了@Transactional注解，表示这是一个数据库事务，每当程序运行出现异常，事务管理器就将出现异常前发送的sql语句回滚，保证数据库中数据的正确、有效、相容性  
**4.2.3 IOC容器**  
**4.2.3.1 技术选型**为了简化变量声明与初始化的过程，增强代码可阅读性、健壮性，我们使用了Spring框架的IOC容器管理需要的bean，并使用了注解实现自动装配、自动注入，为了快速定位错误并调试，还实现了向IOC容器发布Bean时的监听器  
**4.2.3.2 整合Spring**  
   本项目中，还使用了SpringMVC框架，得益于dispatcherServlet的强大，我们能够在web容器初始化时初始化IOC容器，因此我们在web.xml中声明了这一servlet。本项目使用xml文件方式配置SpringIOC容器，而不是使用JavaBean方式配置，相较于JavaBean方式配置，xml配置方式更能够让我们在一处快速定位容器初始化问题。  
**4.2.3.3 依赖注入DI**  
   众所周知，一个简单的POJO/JavaBean都有getter和setter函数，虽然一定程度破坏了面向对象的封装性，但是却方便了我们对一个对象进行业务操作，而不是把关注点放在对象的内部安全性。我们利用了Spring的自动注入功能为我们的service层以及controller层对象注入了一些业务操作对象  
**4.2.4 MVC模式**          
  **4.2.4.1 技术选型**  
   显然，上述的内容只能实现对数据对象以及数据库的操作，我们还缺一个控制器以及视图解析器来实现控制逻辑和视图渲染。因此我们使用了与Spring无缝整合的SpringMVC，它的控制器逻辑浅显易懂易于实现，对拦截器的实现和配置也相对简单，而传统的servlet+filter代码量相对较多且控制逻辑与视图交杂。除了控制器，SpringMVC还为我们提供了多种视图模板，使用统一的视图解析器就能够对多种视图进行分析渲染，我们的视图基本上都是以jsp为主，同时也使用了jstl标签库来简化视图设计  
**4.2.4.2 整合SpringMVC**  
   本项目前端使用了多种脚本，包括js、ajax、jquery，我们前后端传递数据**大部分**都是以json格式传递的，因此我们引入了jackson来将后端的数据转化为json数据并设计了简单的响应数据容器ResponseVessel来存储状态码、后台消息以及后台数据  
**五、系统功能模块详细设计**

**5.1 注册模块**

**5.1.1功能**



时序图

**5.1.2性能**

用户通过输入自己的身份信息来进行注册，已经使用过的手机号和身份证信息无法重复注册，保证了安全性。

**5.1.3输入项目**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **类型** | **长度** | **是否为空** |
| Name | Char | 50 | 否 |
| Password | Char | 50 | 否 |
| Confirmpassword | Char | 50 | 否 |
| Tel | Char | 12 | 否 |
| ID-card | Int | 20 | 否 |
| Email | Char | 20 | 否 |

**5.1.4输出项目**

根据用户所输入的信息，在数据库中首先查询是否存在相同的邮箱或者手机号，如果不存在则将用户所输入的信息加入到数据库当中，如果存在则返回已经存在。

**5.1.5算法**

该模块通过访问数据库查询用户所输入的注册信息，提取用户在html界面上输入的用户名、密码、身份信息、手机号、邮箱等首先自动在页面上检查是否满足格式，如果不满足则提示格用户式错误，如果满足则封装成insert语句，到数据库中搜索相关信息，如果数据库中已经存在相同的邮箱或者手机号，则返回已经存在，如果不存在则并将信息返回到用户界面。

身份证正则匹配：

第一位不可能是0

第二位到第六位可以是0-9

第七位到第十位是年份，所以七八位为19或者20

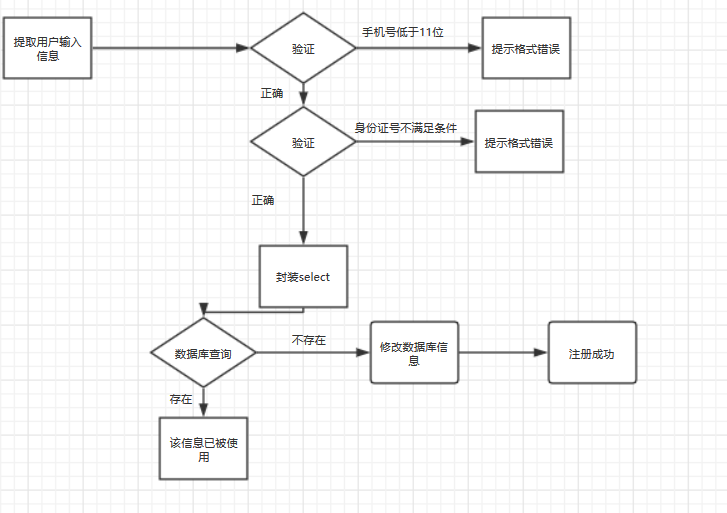
十一位和十二位是月份，这两位是01-12之间的数值

十三位和十四位是日期，是从01-31之间的数值

十五，十六，十七都是数字0-9

十八位可能是数字0-9，也可能是X

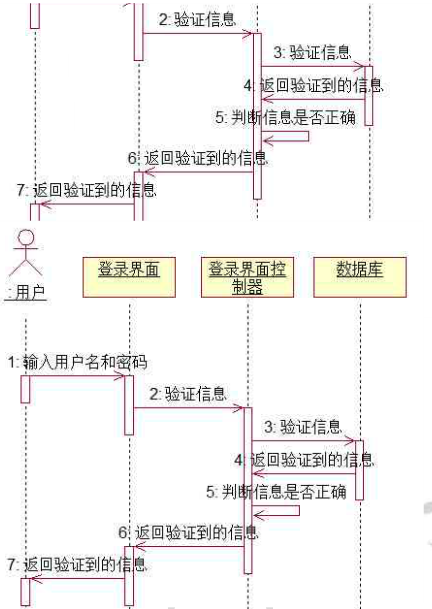
**5.1.6程序逻辑**



封装insert语句

注册过程流程图

**5.2登录模块**

**5.2.1功能**  


时序图

**5.2.2性能**

用户通过用户名和密码进入系统，可以确保系统的安全，由于系统管理员和用户具有不同的权限，可以保证系统的权限分明。

**5.2.3输入项目**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 允许空 | 主键 |
| Tel | Int | 11 | 不允许 | 是 |
| 密码 | Char | 50 | 不允许 |  |

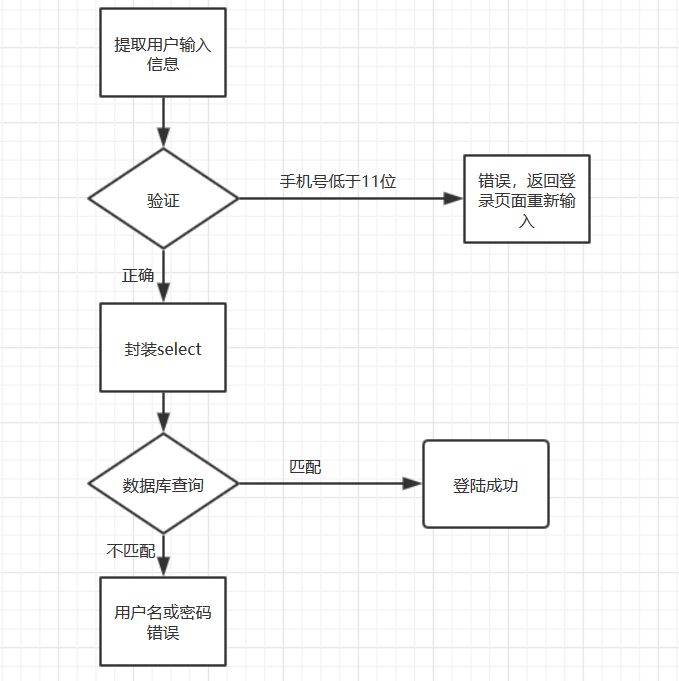
**5.2.4输出项目**

根据数据库中存在的表的信息进行查询，判断用户名、密码是否正确，正确则显示登陆成功并跳转，若用户名和密码不匹配则显示输入错误

**5.2.5算法**

该模块通过访问数据库查询用户登录信息，提取用户在html界面上输入的用户名和密码，封装成select语句，到数据库中搜索相关信息，并将信息返回到用户界面。若用户输入错误，则弹出提示。

**5.2.6程序逻辑**



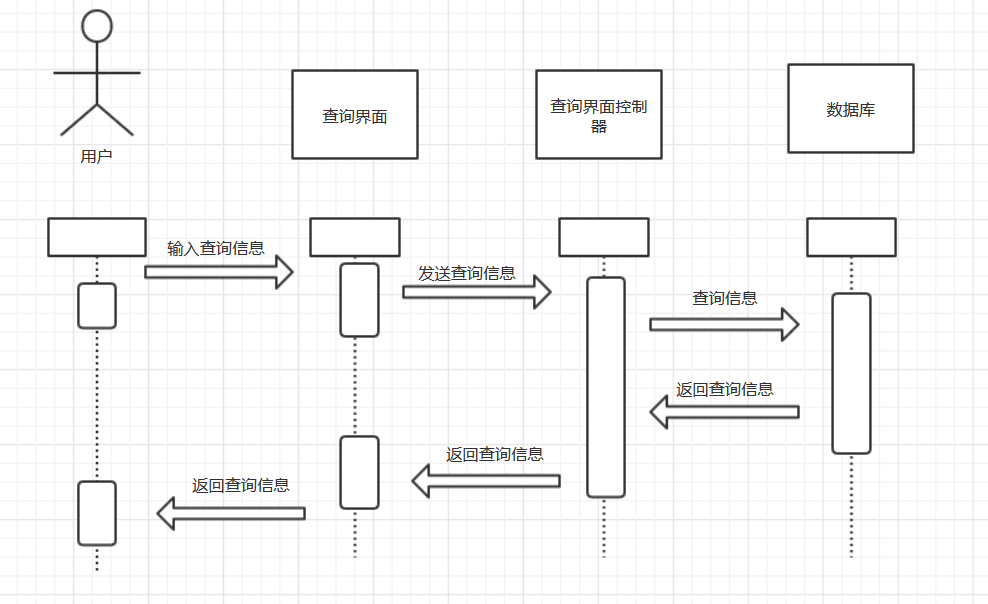
登陆过程流程图

**5.2.7限制条件**

由于登录系统是需要用户名和密码，因此需要用户输入用户名和密码，并且是已经注册过的用户。

**5.3 查询房间模块**

**5.3.1功能**



时序图

**5.3.2性能**

用户通过输入目的地以及出发时间，回归时间，关键字进行搜索，出发时间和回归时间由系统给定的日历选择，并且提供了模糊搜索（关键字），大幅提高了用户使用时的效率。

**5.3.3输入项目**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 是否为空 |
| 目的地 | String | ------- | 是 |
| 关键字 | String | ------- | 是 |
| 评分 | Int | 10 | 是 |

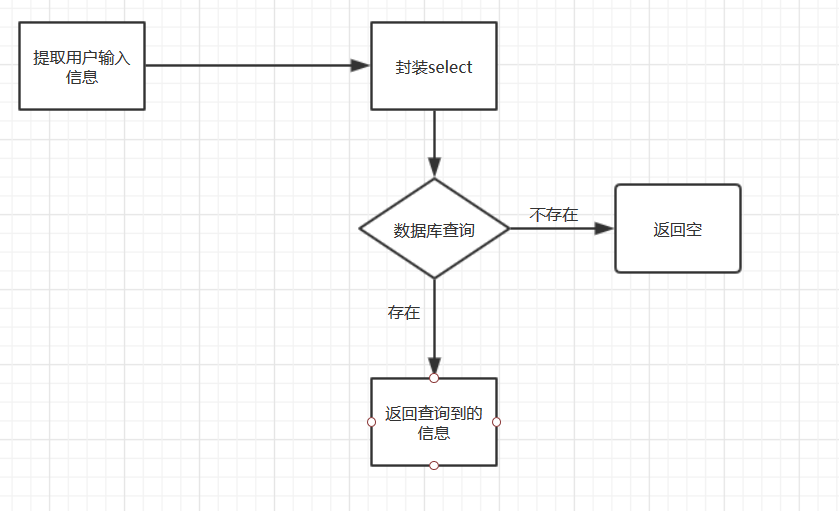
**5.3.4输出项目**

根据数据库中存在的表的信息进行查询，判断是否存在于用户所输入的条件所相符的酒店，并显示出来。

**5.3.5算法**

该模块通过访问数据库查询用户所输入的酒店信息，提取用户在html界面上输入的目的地、关键字、评分等，之后根据数据库所提供的模糊查询返回值，并跳转。

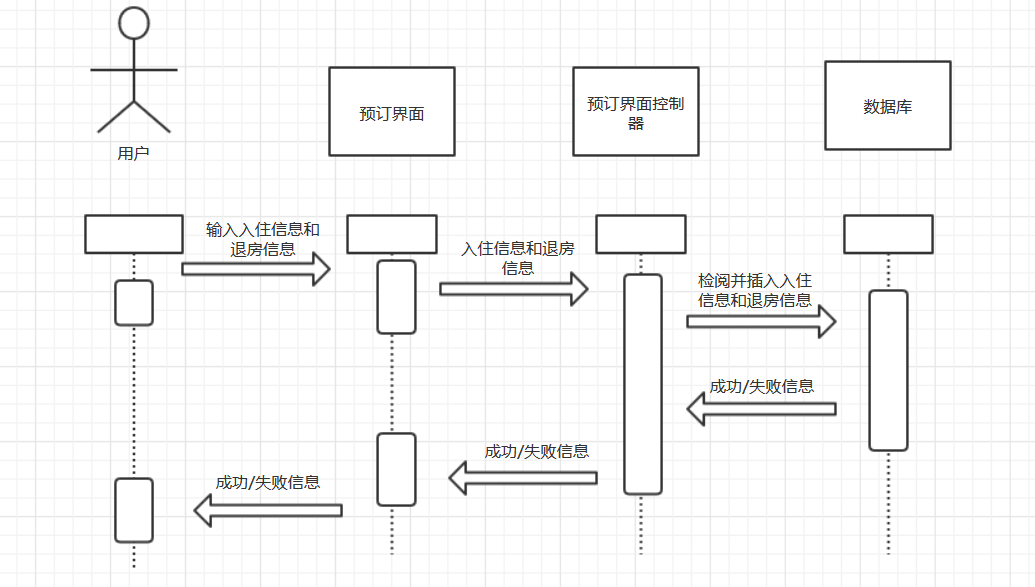
**5.3.6程序逻辑**



查询过程流程图

**5.4 订单控制模块**

**5.4.1功能**



预订模块时序图

**5.4.2性能**

**登陆之后，用户根据自己的情况选择订房时间和退房时间，订房人数而不是输入信息，方便快捷。**

**5.4.3输入项目**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 是否为空 |
| 订房人数 | Int | 10 | 否 |

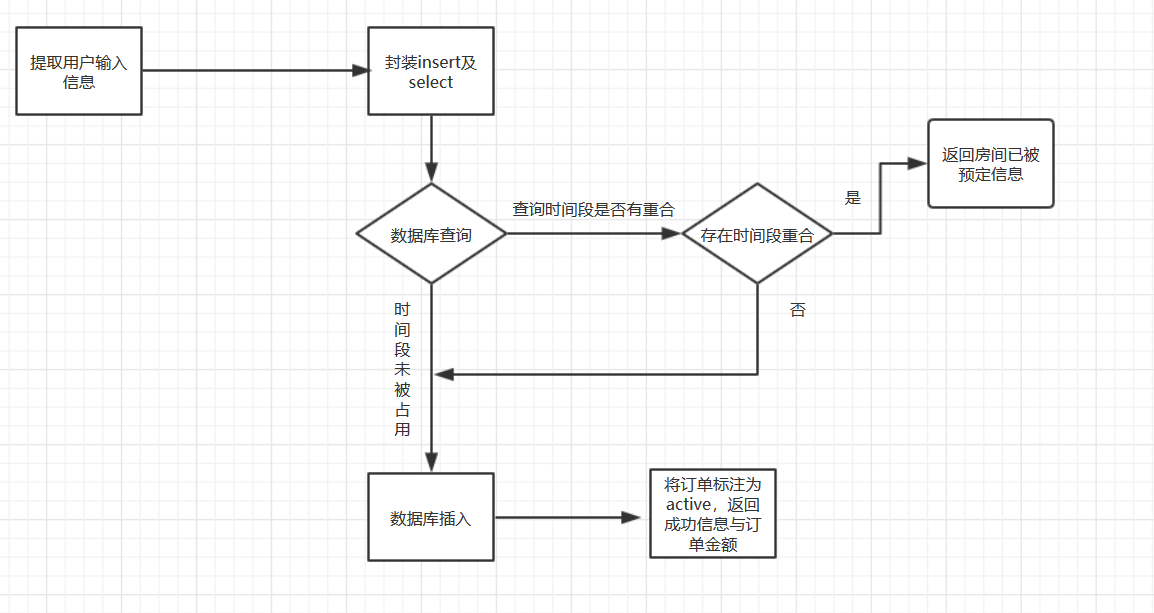
**5.4.4输出项目**

该模块通过提取用户在html界面上输入的订房人数及选择的入住时间、退房时间，封装成insert语句，添加到数据库中相关信息，并将信息返回到用户界面。若用户输入错误，则弹出提示。

**5.4.5算法**

计算用户入住到退房的天数，并与单位时间的价格进行乘法运算，计算出最终金额，在数据库中如果用户输入的几天时间段被占用，则返回时间段被占用

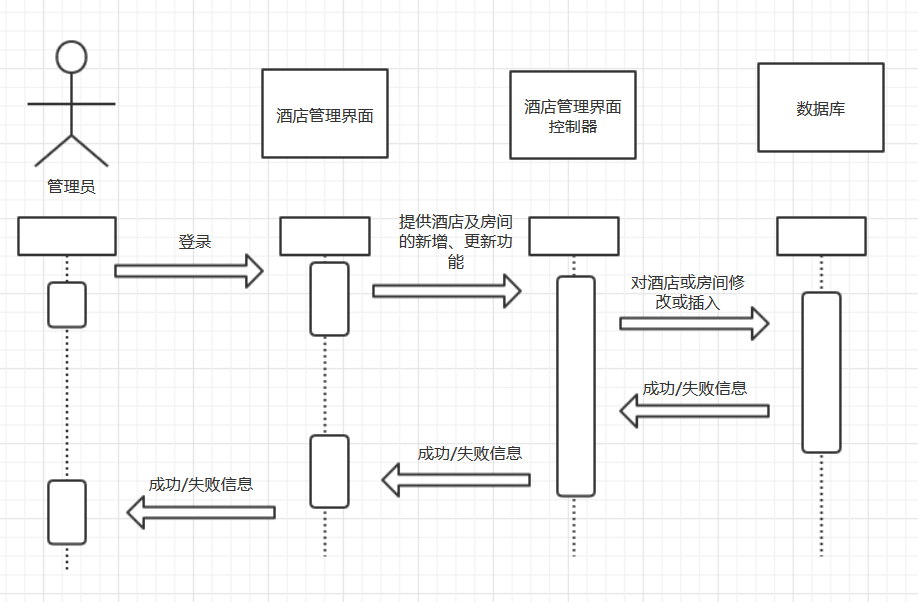
**5.4.6程序逻辑**



房间预订流程图

**5.5 管理员控制模块**

**5.5.1功能**



管理员登录后台时序图

**5.5.2性能**

禁止管理员的一切删除操作，防止数据库订单系统、酒店和房间的关系紊乱，提供高效查询的手段，可以在 “酒店管理”以及“添加房间”选项中检索对应的旅馆，模糊查询项包括酒店地址、酒店电话、酒店ID以及关键字等

**5.5.3输入项目**

**5.5.3.1 酒店信息更新**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 是否可为空 |
| 酒店名 | String | -------- | 否 |
| 地址 | String | 255 | 否 |
| 电话 | Varchar | 20 | 否 |
| Ace评分 | Float | 3 | 否 |
| 标签 | Varchar | 255 | 是 |
| 描述 | Varchar | 255 | 是 |

**5.5.3.2 插入新酒店**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 是否可为空 |
| 酒店名 | String | -------- | 否 |
| 地址 | String | 255 | 否 |
| 电话 | Varchar | 20 | 否 |
| Ace评分 | Float | 3 | 否 |
| 标签 | Varchar | 255 | 是 |
| 描述 | Varchar | 255 | 是 |

**5.5.3.3 为酒店插入房间**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 是否可为空 |
| 最大入住人数 | Int | 3 | 否 |
| 床型 | Varchar | 20 | 否 |
| 价格 | Float | 10 | 否 |
| 餐点 | Varchar | 255 | 否 |
| 退房政策 | Varchar | 255 | 否 |

**5.5.3.4 修改某一酒店的房间信息**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 是否可为空 |
| 最大入住人数 | Int | 3 | 否 |
| 床型 | Varchar | 20 | 否 |
| 价格 | Float | 10 | 否 |
| 餐点 | Varchar | 255 | 否 |
| 退房政策 | Varchar | 255 | 是 |
| 房型 | Varchar | 255 | 否 |

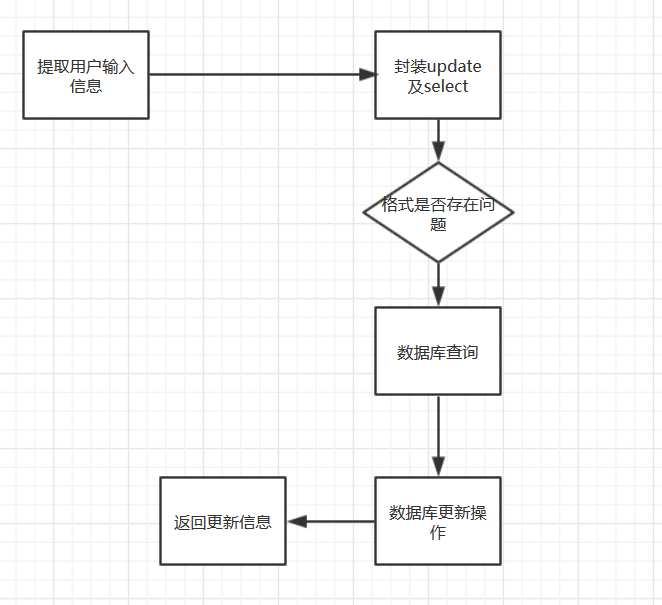
**5.5.4输出项目**

酒店、房间的基本信息。

**5.5.5算法**

管理员通过在页面上录入新的酒店或者宾馆信息，封装到select语句与update中，数据库通过select语句找到表中对应的数据进行修改最后返回。

**5.5.6程序逻辑**



管理员控制模块流程图

**六、数据库详细设计**

**6.1设计依据**

被访问最多的是房间表，该表的数据存储量也是最多的。数据增长量最多的是订单表，该表数据的存储时间最短。酒店及用户表数据存储时间最长

**6.2数据库种类及特点**

数据库种类为关系数据库

特点：

1.数据集中控制，在文件管理方法中，文件是分散的，这些文件之间一般是没有联系的，因此不能按照统一的方法来控制、维护和管理。而数据库则可以集中控制、维护和管理有关数据。

2.数据独立，数据库中的数据独立于应用程序，包括数据的物理独立性和逻辑独立性，给数据库的使用、调整、优化和进一步扩充提供了方便。

3.数据共享，数据库中的数据可以供多个用户使用，每个用户只与库中的一部分数据发生联系；用户数据可以重叠，用户可以同时存取数据而互不影响。

4.减少数据冗余，数据库中的数据不是面向应用，而是面向系统。数据统一定义、组织和存储，集中管理，避免了不必要的数据冗余。

5.数据结构化，整个数据库按一定的结构形式构成，数据在记录内部和记录类型之间相互关联，用户可通过不同的路径存取数据。

6.统一的数据保护功能，在多用户共享数据资源的情况下，对用户使用数据有严格的检查，对数据库规定密码或存取权限，以确保数据的安全性、并发控制。

**6.3数据库逻辑结构**

用户（id, 身份证，用户名，密码，性别，电话，Email，地址，注册日期, 用户类型）实体表

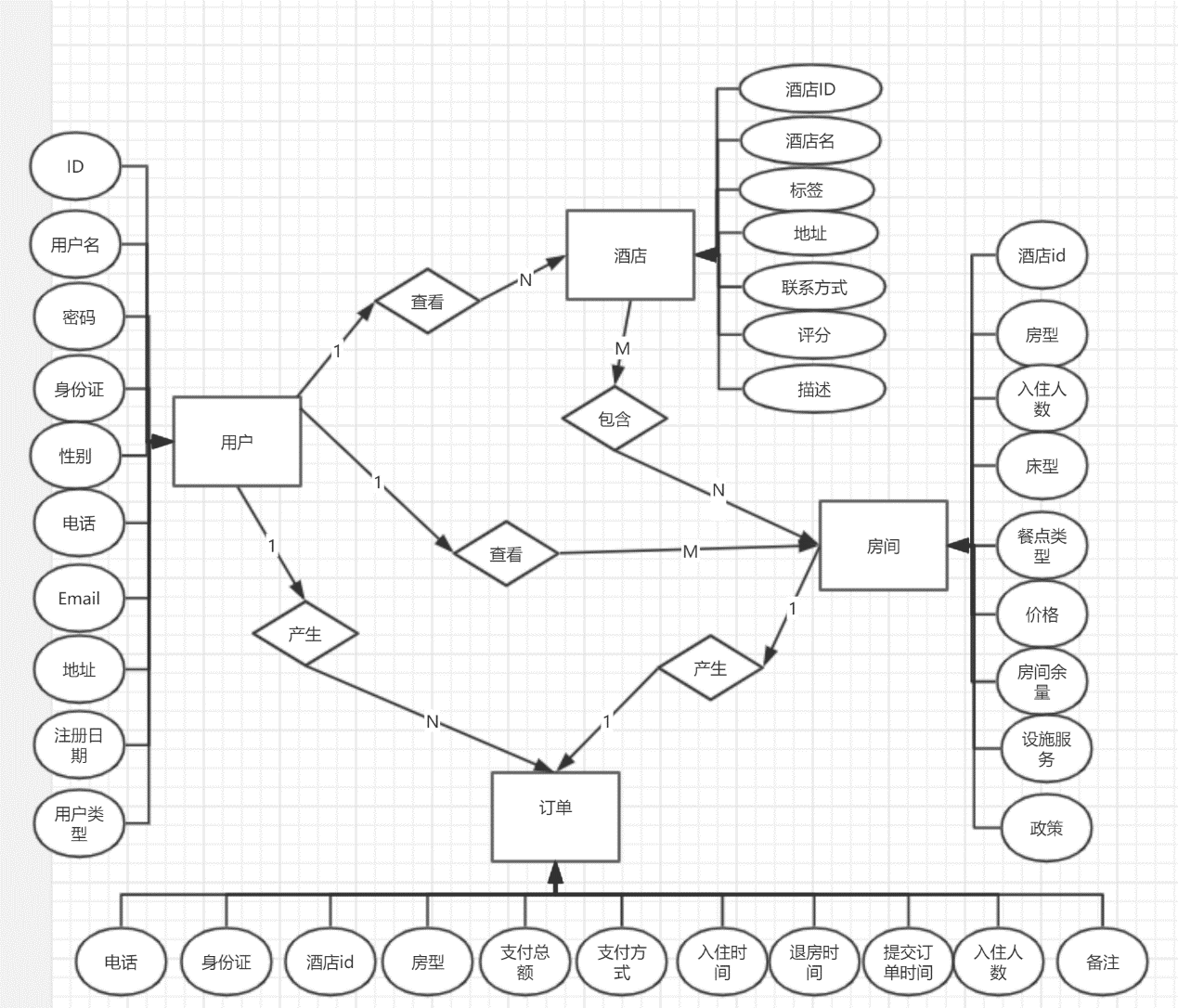
酒店 （id，酒店名，标签，地址，联系方式，评分，描述）实体表

房间 （酒店id，房间id,房型，入住人数，床型，餐点类型，价格，房间余量，设施服务，政策）实体表

入住订单 (电话，身份证，酒店id，房型，支付总额，支付方式，入住时间，退房时间，提交订单时间，入住人数，备注）关系表

酒店图片（id，酒店id，地址，文件名，大小，宽度，高度，描述）实体表

在转换为关系模型时，一对多的联系都在相应的多方实体的关系中增加一个外键。

****

**6.4数据字典**

用户表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 宽度 | 小数 | 主键/索引 | 参照表 | 约束 |
| Id | int |  |  | 主键 |  |  |
| 身份证 | Char | 20 |  |  |  |  |
| 用户名 | Varchar | 50 |  |  |  | 不为null |
| 密码 | Varchar | 50 |  |  |  | 不为null |
| 性别 | Char | 4 |  |  |  | 男or女 |
| 电话 | Char | 12 |  |  |  |  |
| Email | Char | 20 |  |  |  |  |
| 地址 | Varchar | 50 |  |  |  |  |
| 注册日期 | Char | 10 |  |  |  |  |
| 用户类型· | Varchar | 5 |  |  |  | 不为null |

酒店表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 宽度 | 小数 | 主键/索引 | 参照表 | 约束 |
| Id | Int |  |  | 主键 |  |  |
| 酒店名 | Varchar | 20 |  |  |  | 不为null |
| 标签 | Varchar | 50 |  |  |  |  |
| 地址 | Varchar | 50 |  |  |  |  |
| 联系方式 | Varchar | 12 |  |  | 用户表 | 不为null |
| 评分 | double |  | 1 |  |  |  |
| 描述 | Varchar | 100 |  |  |  |  |

房间表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 宽度 | 小数 | 主键/索引 | 参照表 | 约束 |
| 房间id | Int |  |  | 主键（索引） |  | 不为null |
| 酒店id | int |  |  | 主键 | 酒店表 | 不为null |
| 房型 | Varchar | 20 |  |  |  | 不为null |
| 入住人数 | Int |  |  |  |  | 不为null |
| 床型 | Varchar | 20 |  |  |  | 不为null |
| 餐点类型 | Varchar | 20 |  |  |  | 不为null |
| 价格 | Double |  | 2 |  |  | 不为null |
| 房间余量 | int |  |  |  |  | 不为null |
| 设施服务 | varchar |  |  |  |  |  |
| 政策 | varchar |  |  |  |  |  |

订单表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 宽度 | 小数 | 主键/索引 | 参照表 | 约束 |
| 电话 | Char | 12 |  |  | 用户表 | 不为null |
| 身份证 | Char | 20 |  | 主键（索引） | 用户表 | 不为null |
| 酒店id | int |  |  | 主键（索引） | 房间表 | 不为null |
| 房间id | Int |  |  | 主键（索引） | 房间表 | 不为null |
| 房型 | Varchar | 50 |  |  | 房间表 | 不为null |
| 支付总额 | Double |  | 2 |  | 房间表 | 不为null |
| 支付方式 | Varchar | 20 |  |  |  | 不为null |
| 入住时间 | Varchar | 20 |  |  |  | 不为null |
| 退房时间 | Char | 10 |  |  |  | 不为null |
| 提交订单时间 | char | 10 |  |  |  | 不为null |
| 入住人数 | varchar | 10 |  |  |  | 不为null |
| 备注 | varchar | 100 |  |  |  |  |

酒店图片表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 宽度 | 小数 | 主键/索引 | 参照表 | 约束 |
| id | Int |  |  | 主键 |  |  |
| 酒店id | Int |  |  |  | 酒店表 | 不为null |
| 地址 | varchar | 50 |  |  |  | 不为null |
| 文件名， | varchar | 20 |  |  |  | 不为null |
| 大小 | varchar | 10 |  |  |  | 不为null |
| 宽度 | varchar | 10 |  |  |  | 不为null |
| 高度 | varchar | 10 |  |  |  | 不为null |
| 描述 | varchar | 200 |  |  |  |  |

**6.5数据库安全**

用户身份鉴别是数据库管理系统提供的最外层安全保护措施，每个用户在系统中都有一个用户标识（由用户名和用户标识号组成），用户标识号（UID）在系统的生命周期内是唯一的，并且系统内部记录这所有合法用户的标识，系统鉴别是指由系统提供一定的方式让用户标识自己的身份或名字。用户进入系统时由系统进行核对，通过鉴定后才提供使用数据库管理系统的权限。提供了初步的静态口令鉴别，同时数据库中的密码以及用户类别进行了加密。通过对用户类别的识别对不同用户授予不同的权限，低用户权限的用户无法进行高权限的操作。同时数据库经常进行备份