

 	文档编号		2016 级	公开
	其他			
	数据库课程设计			

## 《数据库课程设计·期末报告》

### 民宿管理系统

学生姓名： 何长鸿

学生学号： 2016141482154

电子邮件： geekhch@qq.com

指导老师： 袁道华

报告日期： 2018.12.10

## 章节目录

一、 实验目的.....	1
二、 开发环境及工具.....	1
1. 设备.....	1
2. 支撑软件.....	1
三、 需求分析.....	2
1. 需求软件功能说明.....	2
2. 对功能的一般性规定.....	2
3. 对性能的一般性规定.....	2
四、 任务概述.....	3
1. 目标.....	3
2. 系统的特点.....	3
3. 假定和约束.....	3
五、 设计.....	4
1. ER 模型图.....	4
2. 数据表结构定义.....	4
六、 功能模块结构图.....	6
1. 功能模块介绍.....	6
七、 程序功能模块处理流程.....	7
1. 客户端功能流程图.....	7
2. 服务器端功能流程图.....	9
3. 关键功能算法流程.....	10
八、 服务器架构.....	11
1. 主要代码包和类结构.....	11
九、 用户界面.....	12
1. 游客.....	12
2. 管理员.....	15
十、 部分核心代码.....	18
1. 服务器后端.....	18
2. 分页显示.....	26
3. 图片上传实现代码.....	28
十一、 参考文献.....	30

图目录

图 1 ER 模型图.....4

图 2 表结构定义..... 5

图 3 功能模块..... 6

图 4 客户端游客功能模块处理流程..... 7

图 5 客户端商家功能操作流程..... 8

图 6 服务器请求处理流程..... 9

图 7 剩余房间计算控制流程..... 10

图 8 项目代码包结构..... 11

图 9 用户注册页面..... 12

图 10 用户登录页面..... 13

图 11 房间浏览界面..... 13

图 12 预定房间界面..... 14

图 13 输入预定客户信息界面..... 14

图 14 确认支付界面..... 15

图 15 查看预定订单..... 15

图 16 房间分类管理..... 16

图 17 房间对象管理..... 16

图 18 用户管理..... 17

图 19 订单管理..... 17

表目录

表 1 民宿旅游需求规定..... 2

# 民宿管理系统设计报告

何长鸿  
2016141482154

## 一、 实验目的

开发完成一个小型的民宿管理。完成从需求分析、数据模式设计到编码实现、系统调试的所有流程；通过此一图书管理系统的实现，在实践中掌握数据库系统设计开发的特点、方法和步骤。

## 二、 开发环境及工具

### 1. 设备

本系统采用 B/S 模式开发，开服务器发语言使用 java

- 1) 服务器运行于 Windows 或 Linux 系统
- 2) 客户端全平台支持

### 2. 支撑软件

#### (1) 服务器

- 1) MySQL 数据库
- 2) Tomcat 8.5
- 3) Java SDK1.8

#### (2) 客户端

- 1) 主流浏览器均可，例如 Firefox、Chrome、Safari 等

### 三、需求分析

#### 1. 需求软件功能说明

分类	功能
管理	<ol style="list-style-type: none"><li>1. 房间类型资源上架，修改，统计，SKU 管理。</li><li>2. 民宿资源展示</li><li>3. 房间类型所含属性修改</li><li>4. 房间分类图片上传</li><li>5. 对订单的管理：查询，修改，删除。</li><li>6. 用户信息管理、增删改查</li><li>7. 房间信息管理：房间数量、名称、属性修改、原价、活动价</li><li>8. 订单信息修改</li><li>9. 订单信息多条件组合查询</li><li>10. 一键查询已购买超过三天未评论的订单</li><li>11. 房间存量同步统计</li><li>12. 各种列表清单的导出 excel 文件和打印</li></ol>
游客功能	<ol style="list-style-type: none"><li>1. 预览商家</li><li>2. 民宿资源信息</li><li>3. 预订下单</li><li>4. 空闲房间不足时拒绝购买</li><li>5. 预定订单查看</li><li>6. 评论已购买商品</li><li>7. 注册</li><li>8. 登录</li></ol>

表 1 民宿旅游需求规定

#### 2. 对功能的一般性规定

软件采用可视化图形界面，界面格式统一，界面功能键排版能使用户能较快找到所需功能按钮，设定默认值以防每次操作要求输入太多，统一的错误提示风格。

#### 3. 对性能的一般性规定

##### (1)精度

浮点型数据保存 2 位以上小数，百分比数据小数点后 2 位小数。

##### (2)输入输出要求

信息录入时，数字、字符、时间日期的格式和长度应遵照提示要求，否则不能提交或提示出错。系统输出时，与金额相关的数值都保留 2 位小数；百分比数值小数点后保留 2 位小数。

(3)数据管理能力要求

需要管理的问卷

(4)其他专门要求

- 1) 用户的信息加密、信息认证（登陆访问）
- 2) 使用方便
- 3) 可维护性、可补充性、易读性、可靠性

## 四、 任务概述

### 1. 目标

本项目旨在开发一款满足用户商户需求的民宿旅游平台。为用户提供更好的房屋以及售后服务，同时为商户提供更多地出租机会。同时营造一个游客们的交流平台，为平台提供一定的人文气息。

### 2. 系统的特点

- 1) 易用性：人性化设计，面向社会大众使用，简单操作的功能，美观和谐的用户交互界面
- 2) 实用性：以实际应用为出发点，向旅游大众，名宿商家，旅游管理局做出详细调研，做出最优化设计，满足各方面需求。
- 3) 高效性：通过互联网技术，使得民宿信息实时更新，让游客随时随地货比三家，选择自己最满意的名宿，提高出行的满意度，通过网络支付手段减少交易流程，使得出行变得高效快捷。
- 4) 安全性：外部安全性：能够使得旅游局有序管理，用户的意见也能得到第一时间反馈，保证用户出行的安全，做到有记录可循。内部安全性：利用 javaweb 优秀的封装机制，提供多种安全保护方法，保证了客户资料隐私的安全和完整。
- 5) 开发性：可以用其他软件，平台完成必要的整合应用。

### 3. 假定和约束

1) 假定 I 经费限制

选择 Web 服务器与内存服务器只能选择内存较小的服务器，则开发完成以后系统的浏览速度不高，并行操作人数也受限。

2) 假定 II 开发期限

本项目的开发周期为 17 个周，由于参与项目开发的学生都是第一次着手开发，能力尚不足，在这规定的开发周期内面临着不能成功开发出一个完整的系统。

## 五、 设计

### 1. ER 模型图

在本系统中，涉及的实体性有游客、商家、订单、房间、图片、用户评论等，但为了房间类别关系，去掉大量房间类别重复字段，并且对此形成约束，所以将房间类型抽象为实体型，整个系统数据库 ER 模型图如下所示：

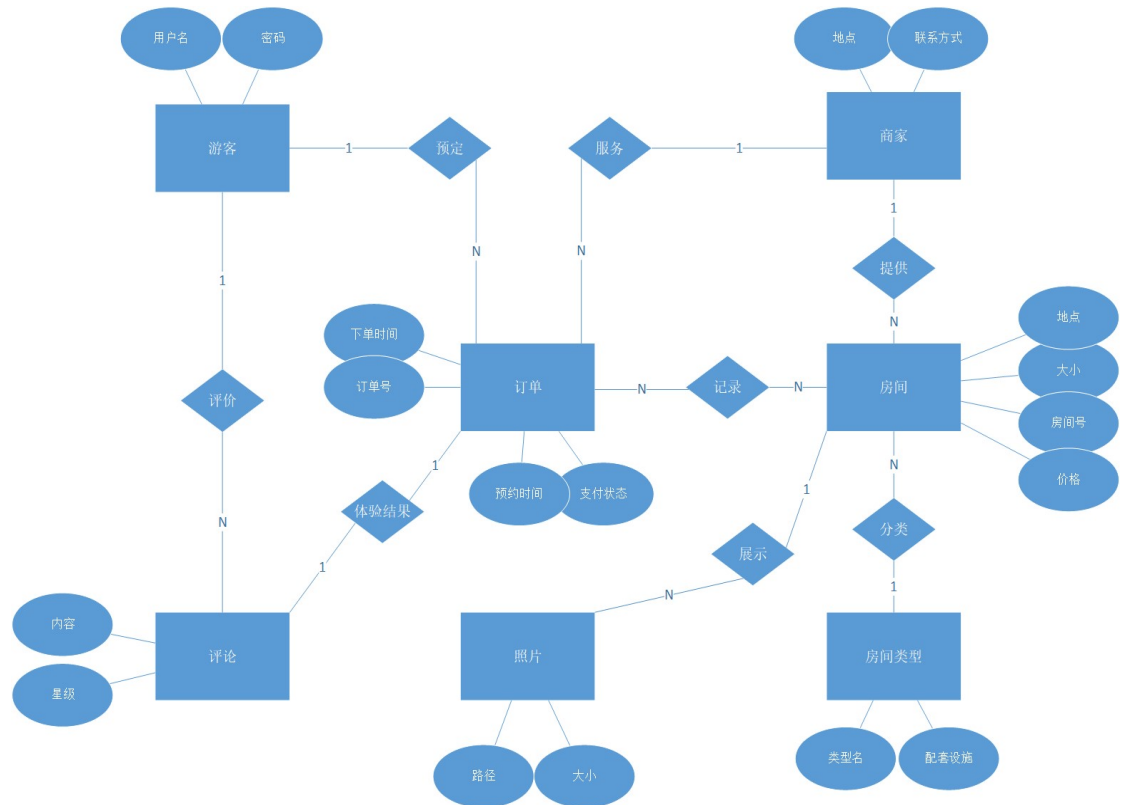


图 1 ER 模型图

### 2. 数据表结构定义

在下图中给出了数据表名，字段名、字段类型、数据长度以及数据表说明

<b>User</b> id int(11) name varchar(255) password varchar(255) ----- 游客/商家用户	<b>Review</b> id int(11) content varchar(4000) uid int(11) pid int(11) createDate timestamp ----- 评论
<b>Property</b> id int(11) cid int(11) name varchar(255) ----- 房间属性	<b>Product</b> id int(11) name varchar(255) subTitle varchar(255) originalPrice float createDate timestamp cid int(11) stock float promotePrice float ----- 房间
<b>OrderItem</b> id int(11) oid int(11) pid int(11) uid int(11) number int(11) ----- 订单购买内容	<b>Order_</b> id int(11) orderCode varchar(255) total float status varchar(255) uid int(11) confirmDate timestamp deliveryDate timestamp payDate timestamp createDate timestamp userMessage varchar(255) mobile varchar(255) receiver varchar(255) post varchar(255) address varchar(255) ----- 订单
<b>Category</b> id int(11) name varchar(255) ----- 房间类型	

图 2 表结构定义



## 六、 功能模块结构图

### 1. 功能模块介绍

本系统分为商家与游客两大部分，使用 B/S 模式开发，两者功能相互对应。主要功能模块包括用户登录与注销、用户信息管理、房源管理、订单管理、图片库管理、预定等流程。同一个用户账户既可以是游客也可以是商家，但会根据用户发布的房源情况自动判断是否有相关功能的操作权限。

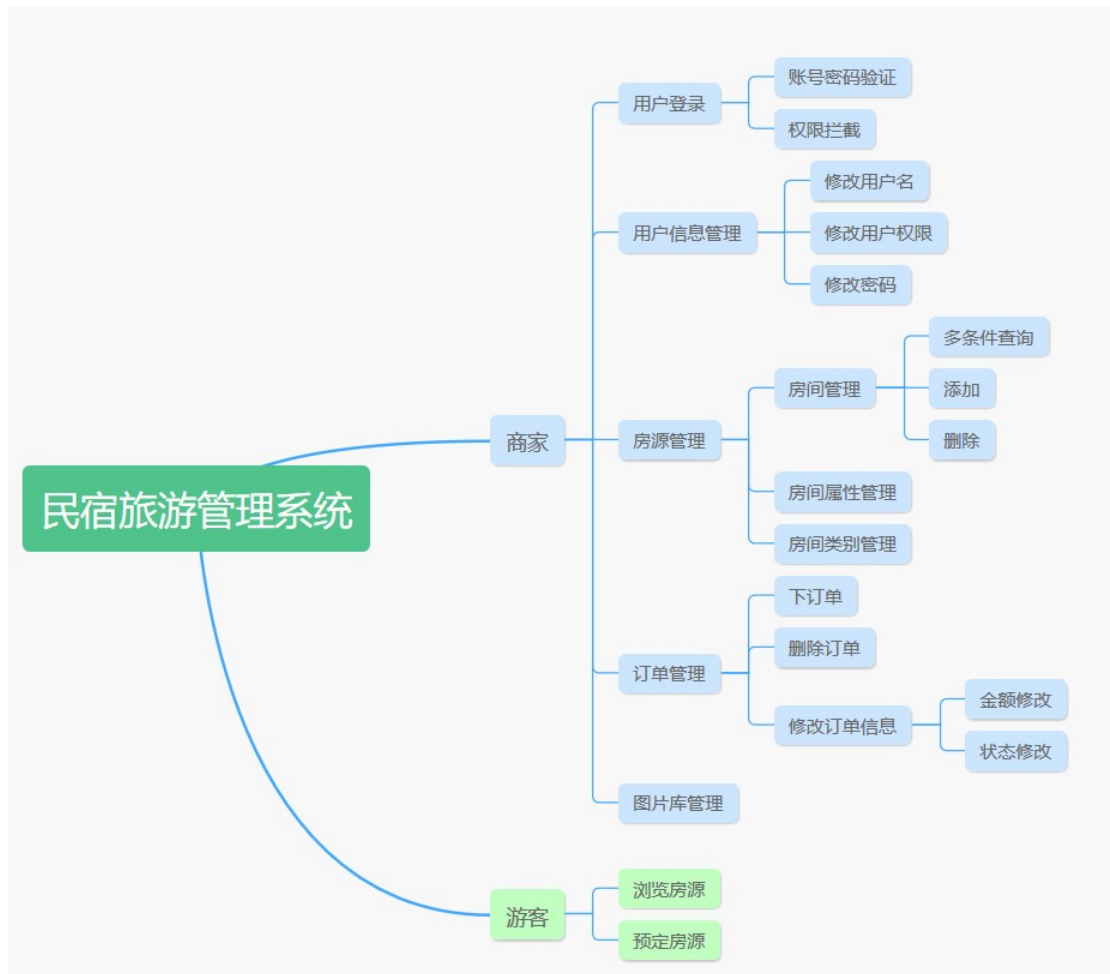


图 3 功能模块

## 七、 程序功能模块处理流程

### 1. 客户端功能流程图

#### (1) 客户端游客注册及登陆、预定功能流程图

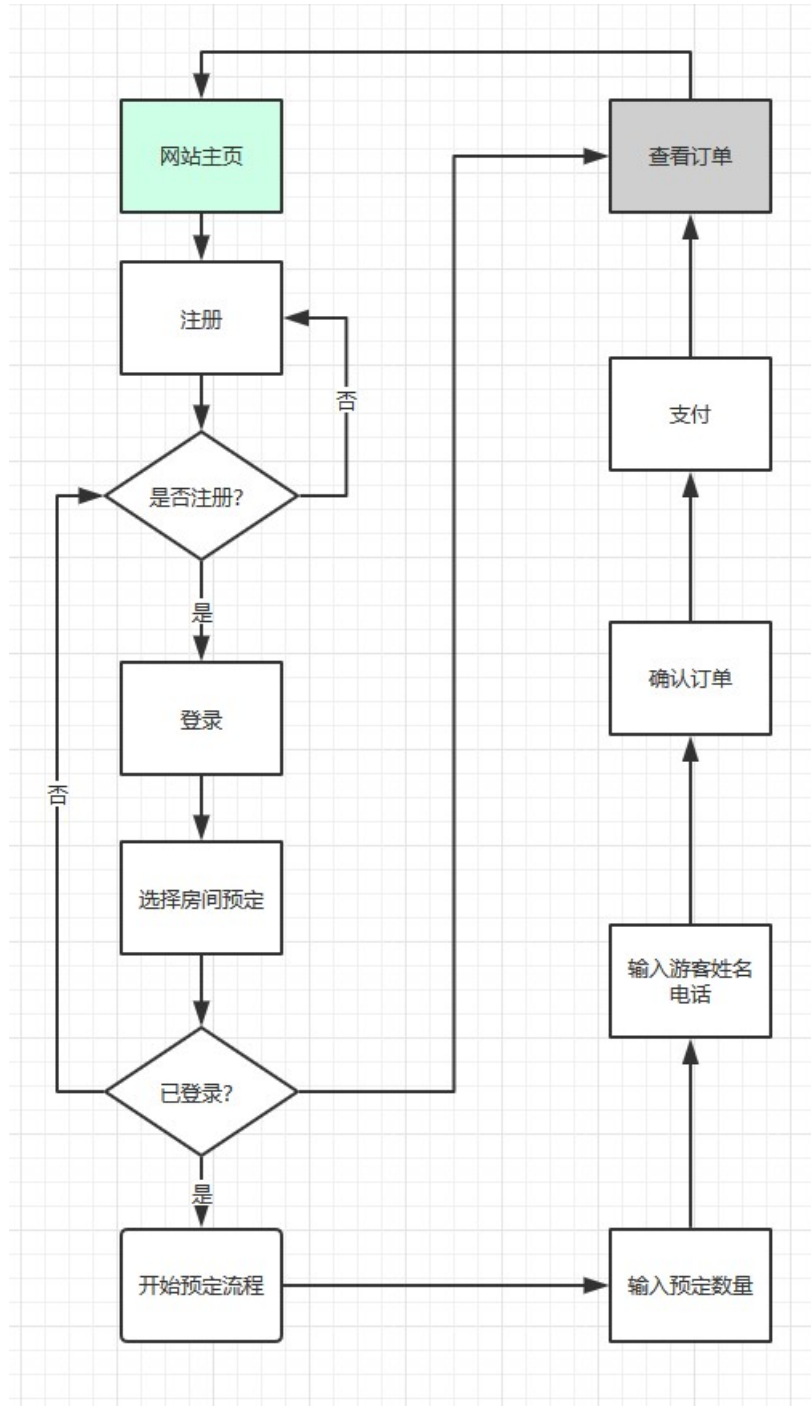


图 4 客户端游客功能模块处理流程

(2) 客户端商家功能流程图

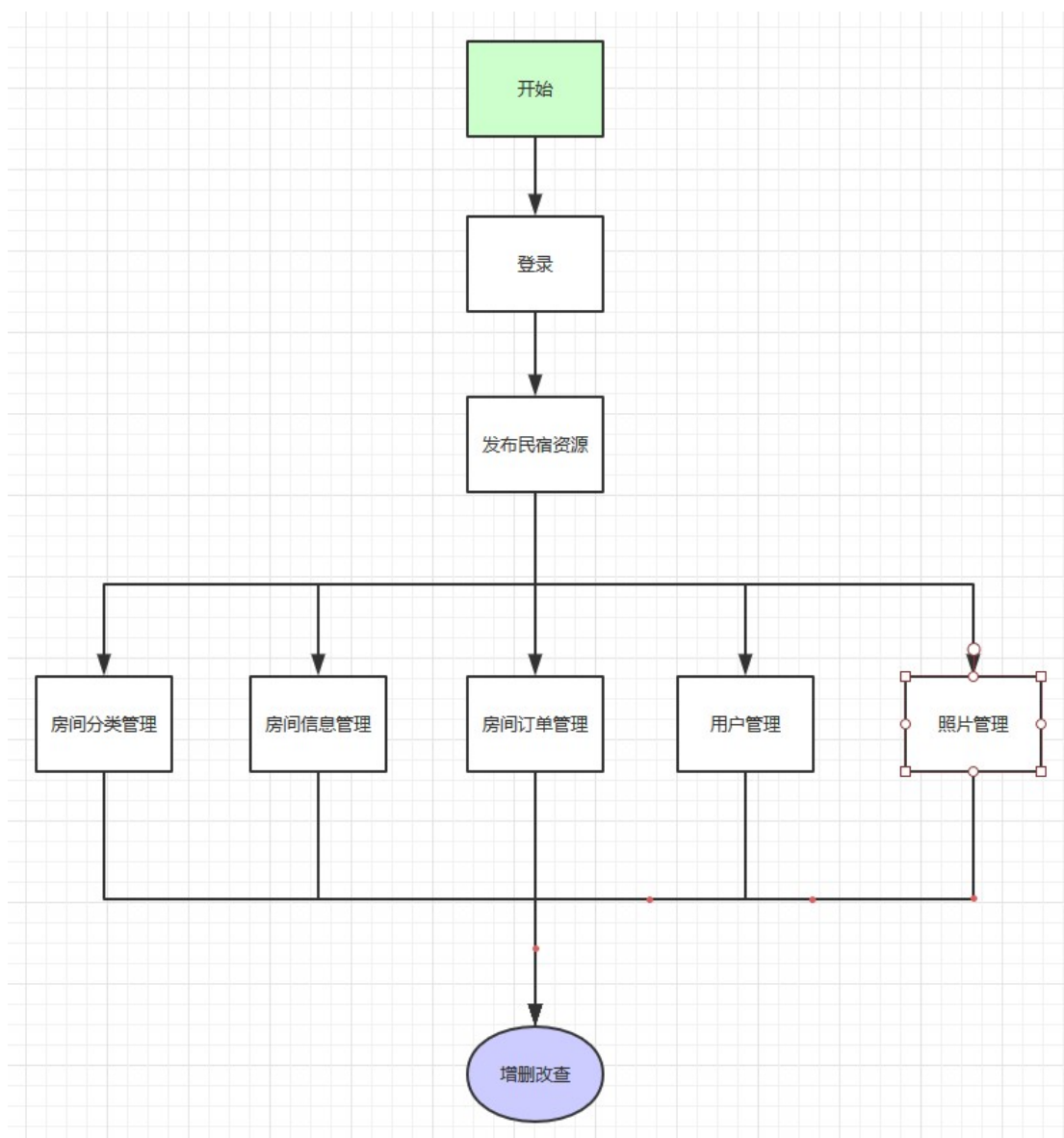


图 5 客户端商家功能操作流程

## 2. 服务器端功能流程图

### (1) 请求处理流程

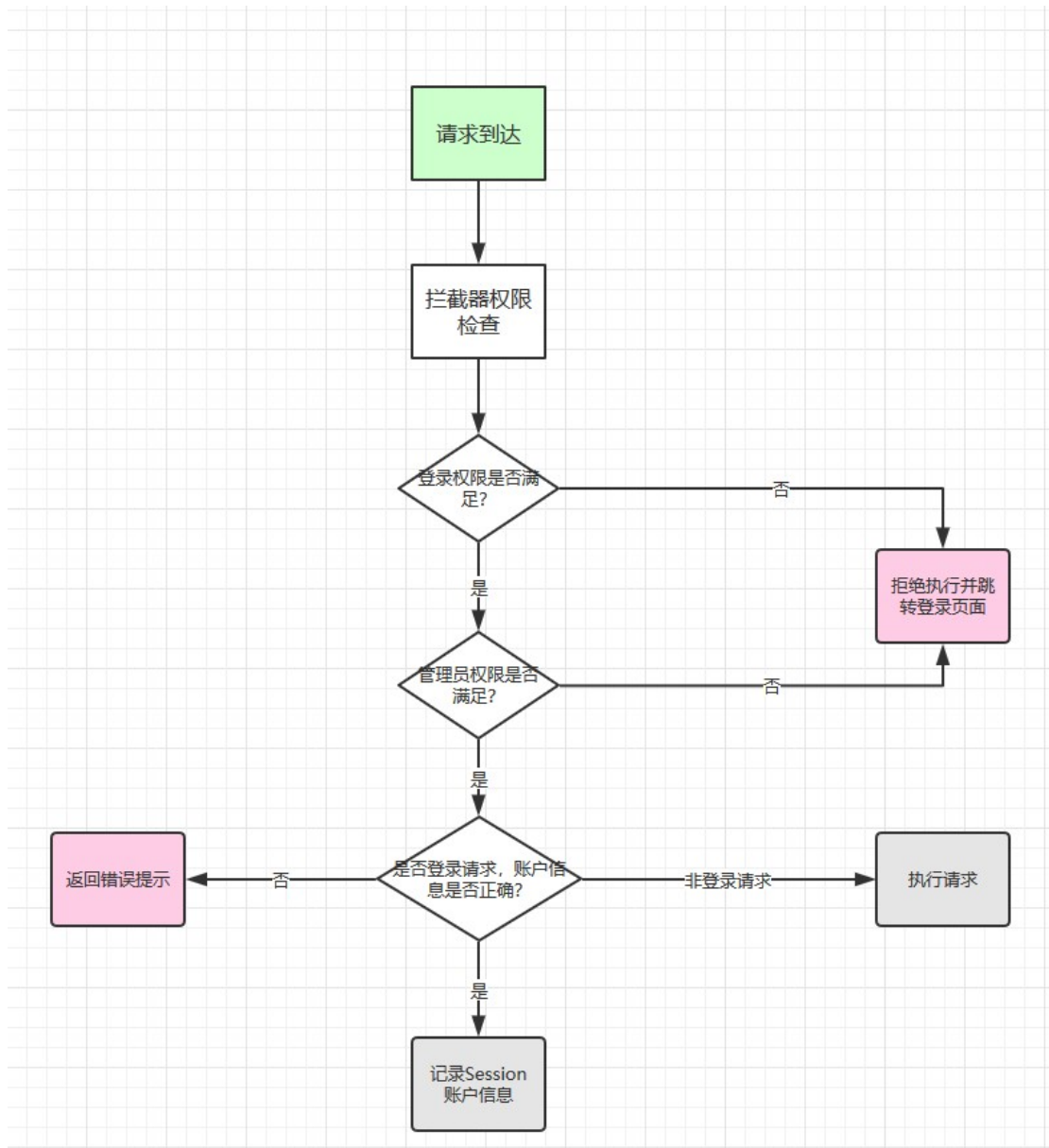


图 6 服务器请求处理流程

### 3. 关键功能算法流程

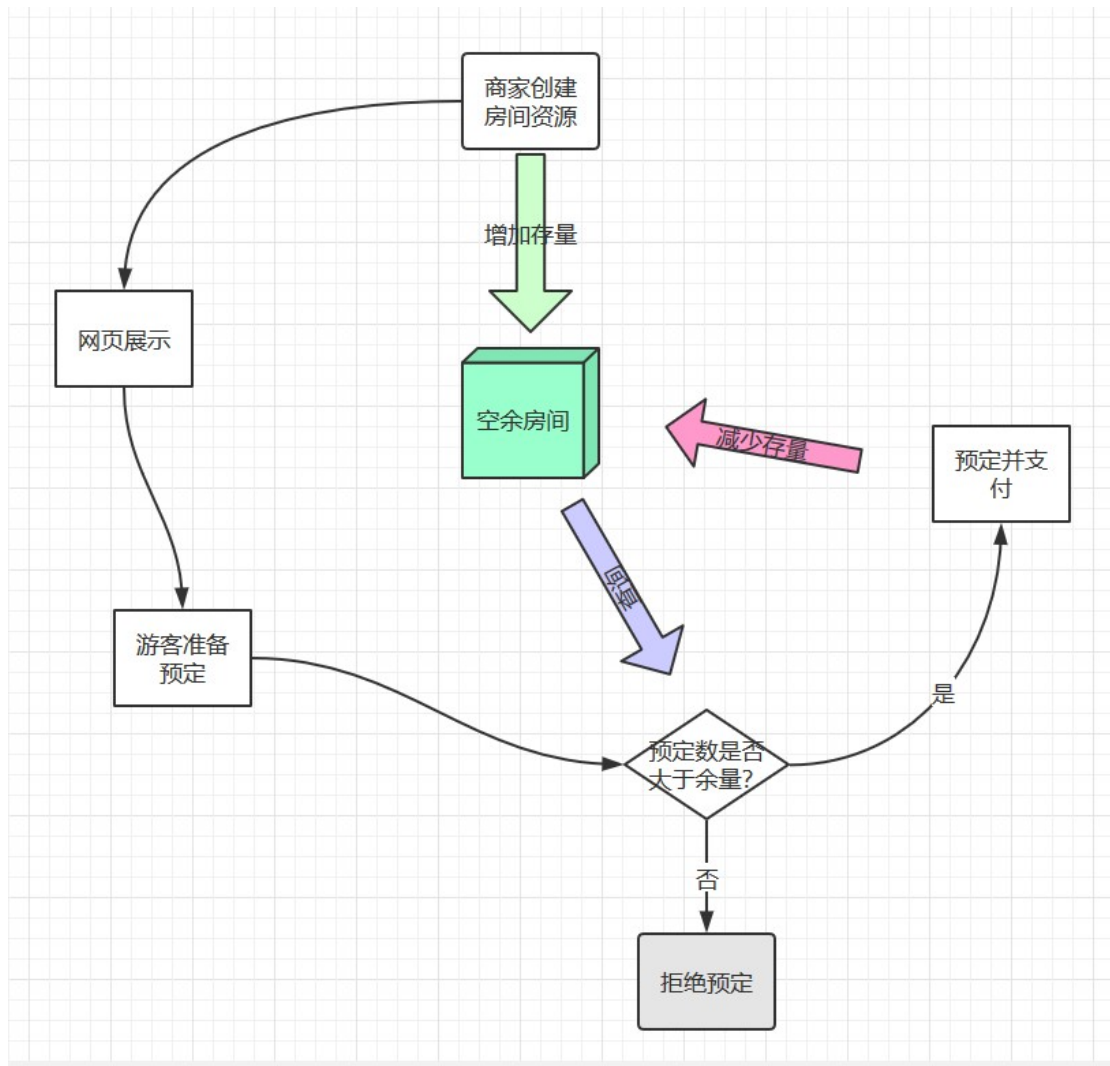


图 7 剩余房间计算控制流程

## 八、 服务器架构

### 1. 主要代码包和类结构

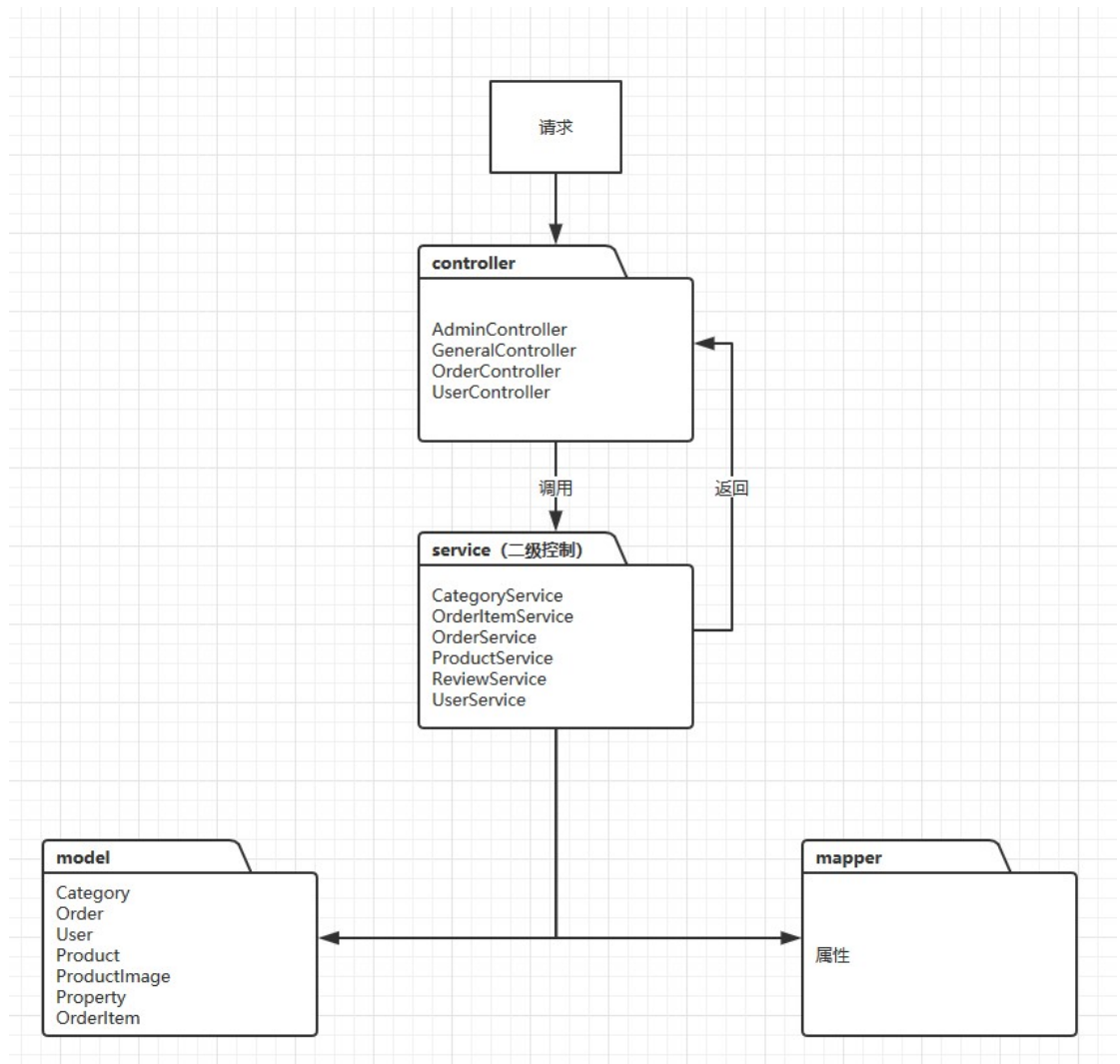


图 8 项目代码包结构

## 九、 用户界面

以下使用用户故事的方式来描述软件功能使用流程。

### 1. 游客

- (1) 打开网站主页
- (2) 注册并登录账号

---



故事  
体验

房源

发布

免费注册

请登录

联系

☎13928712412

注册

账号

登陆密码

密码确认

提交

图 9 用户注册页面

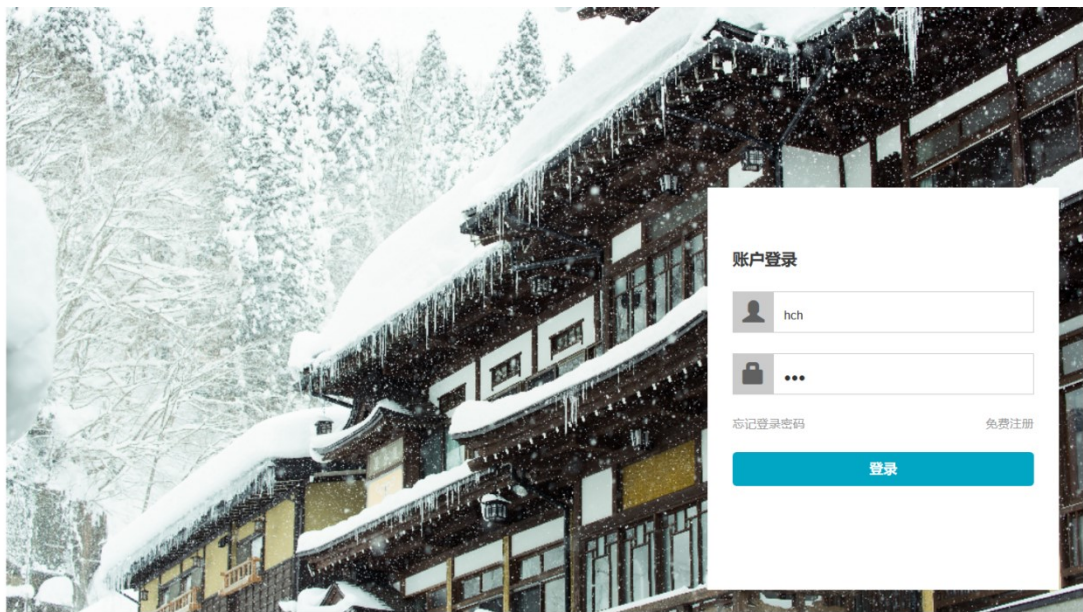


图 10 用户登录页面

(3) 浏览并选择预定房间

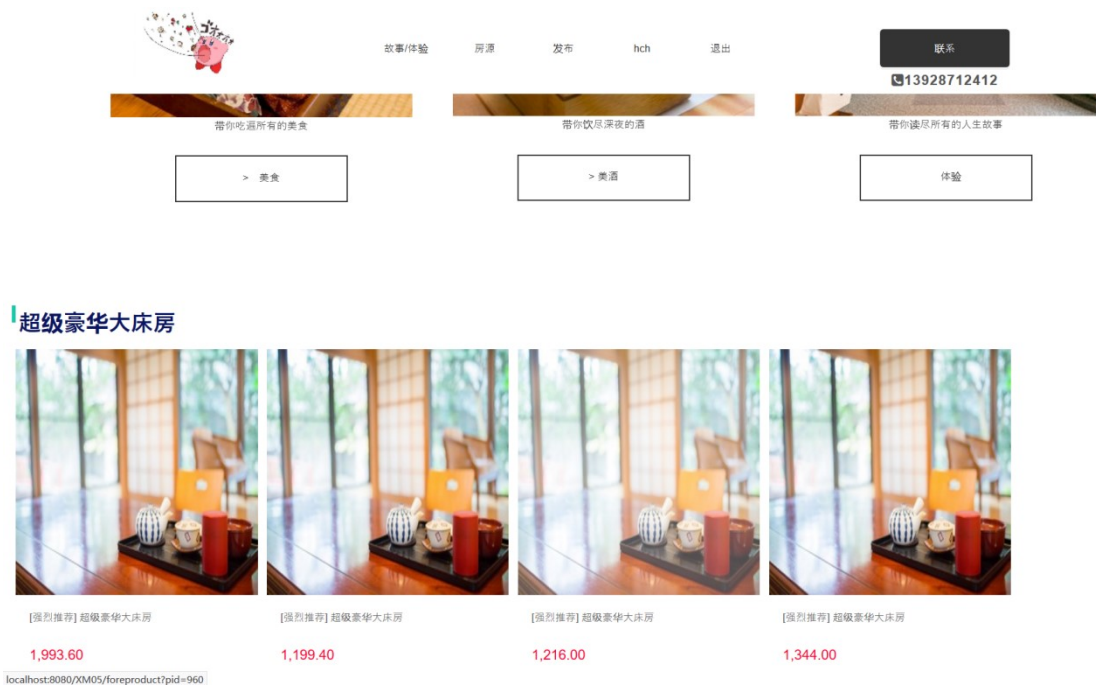


图 11 房间浏览界面

- (4) 如果此时系统检测到用户未登录，则跳转到步骤（2）
- (5) 输入预定房间数量并确定





图 12 预定房间界面

#### (6) 填入客户信息并确认提交订单

故事/体验 房源 发布 hch 退出 联系 13928712412

客户姓名\* test  
 手机号码\* 15465451465

数量	小计
超级豪华大床房 🇨🇳 🇻🇳 🇹🇼	¥1,993.60

总费用 ¥1,993.60

实付款：**¥1,993.60**


提交订单

图 13 输入预定客户信息界面

#### (7) 确认支付

我已入住，同意支付宝付款

订单信息

房间	单价	数量	房间总价
 超级豪华大床房	¥ 2,848.00	1	¥ 1,993.60

实付款：¥ 1,993.60

订单编号：763549b6721d4deaa91469be2cf2d8d2

卖家昵称：GoTrip

成交时间：2018-12-13 10:55:14

确认支付

图 14 确认支付界面

## (8) 查看已预定房间











	故事 体验	房源	发布	订 住	退出	联系 13928712412
所有订单	待付款	待评价				
房间	单价	数量	实付款	交易操作		
2018-07-11 20:31:24 订单号: eb8c1621a7b74c0bb4de77c10c1156b7  超级豪华大床房	GoTrip ¥ 1,280.00 ¥ 1,216.00	0	¥ 0.00			
2018-07-11 20:52:24 订单号:  超级豪华大床房	GoTrip ¥ 1,280.00 ¥ 1,216.00	0	¥ 0.00			
2018-12-13 10:34:58 订单号:  超级豪华大床房	GoTrip ¥ 2,848.00 ¥ 1,993.60	0	¥ 0.00			
2018-12-13 10:35:12 订单号:  超级豪华大床房	GoTrip ¥ 2,848.00 ¥ 1,993.60	0	¥ 0.00			

图 15 查看预定订单

## 2. 管理员

- (1) 注册、登录
- (2) 房源分类管理（增删改）



房源管理

用户管理

订单管理

故事管理

免费注册

联系

13928712412

房源管理

ID	图片	分类名称	属性管理	房间管理	编辑	删除
60		超级豪华大床房				
64		可爱公主房				
77		亲子套房				
78		单人房				
79		青年旅间				
80		特价小包间				
81		榻榻米				
82		地下旅店				
83		合租间				

新增分类

分类名称

分类图片

浏览... 未选择文件。

提交

图 16 房间分类管理

(3) 某个分类下的房间管理

ID	图片	房间名称	房间描述	原价格	优惠价格	剩余房间	图片管理	设置属性	编辑	删除
958		超级豪华大床房	超级大的床	2848.0	1993.6	5				
959		超级豪华大床房	超级大的床	1999.0	1199.4	8				
960		超级豪华大床房	超级大的床	1280.0	1216.0	5				
961		超级豪华大床房	超级大的床	1680.0	1344.0	8				
962		超级豪华大床房	超级大的床	980.0	882.0	5				

新增房间

房间名称

房间描述

原价格

1999

优惠价格

1899

剩余房间

9

提交

图 17 房间对象管理

(4) 用户管理

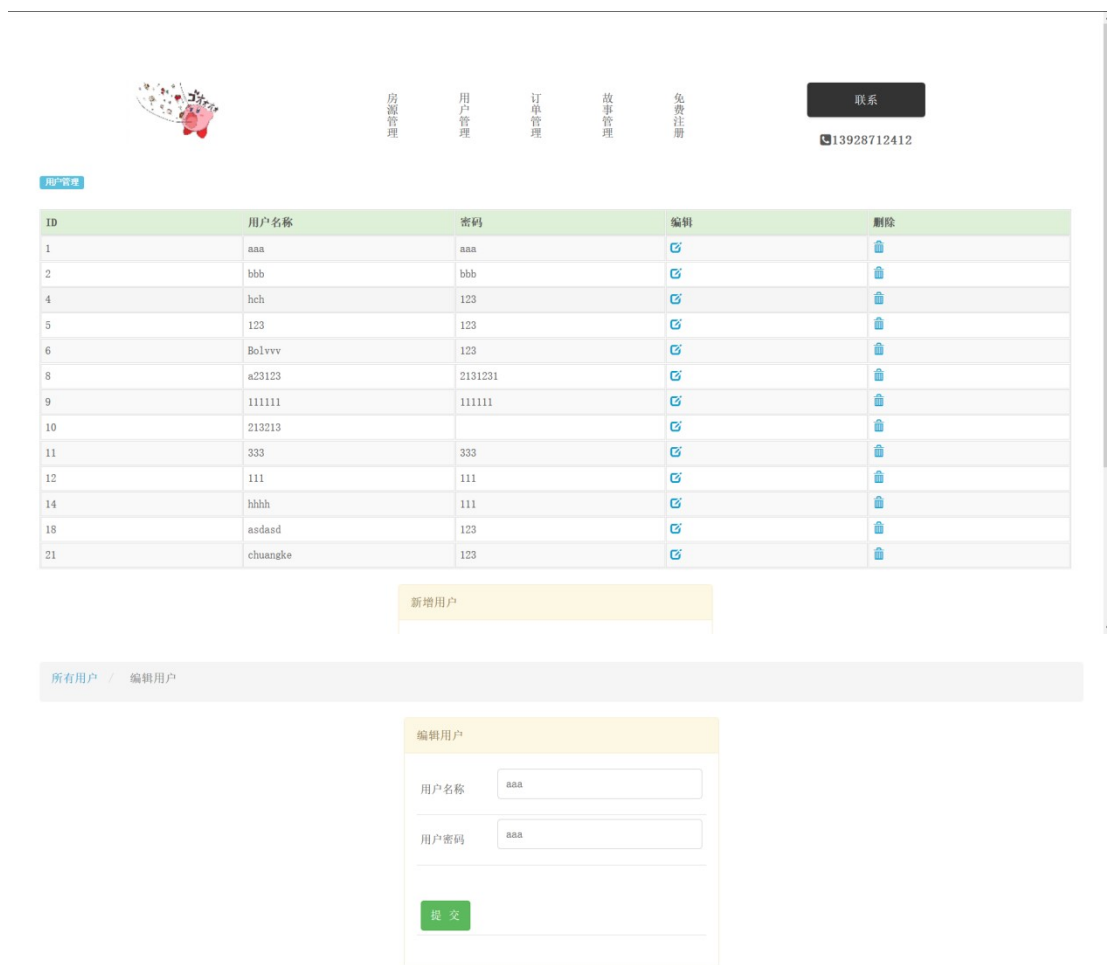


图 18 用户管理

### (5) 订单管理

包括支付状态、金额、订单中预定的房间数量、买家信息、支付时间等数据项。管理功能提供了增删改查等操作。

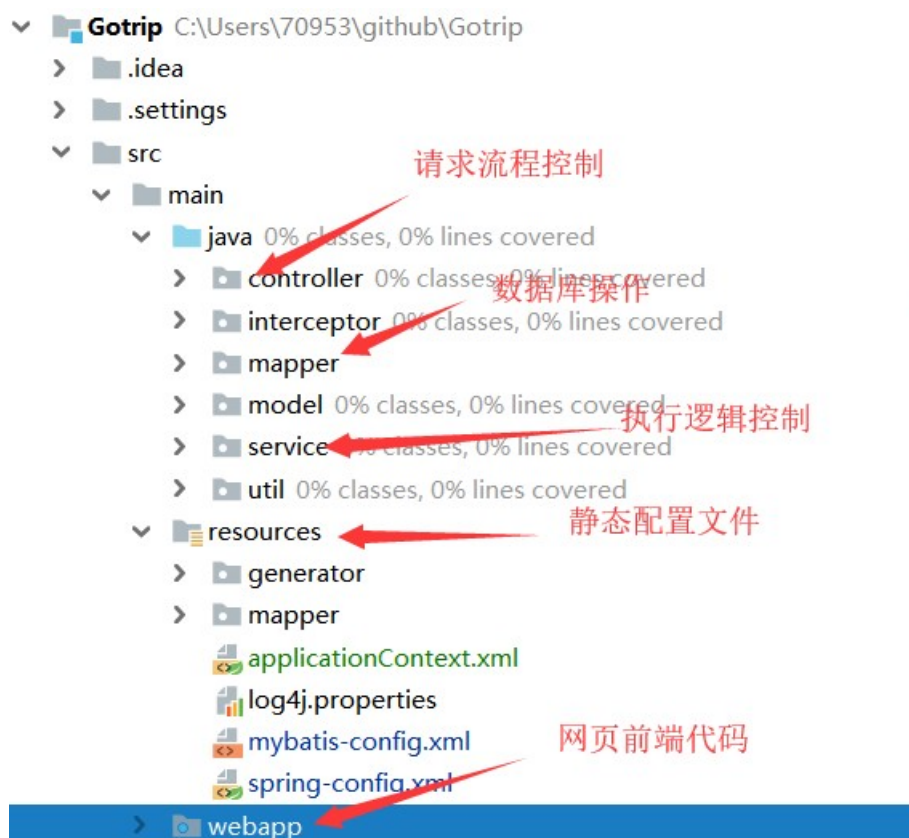


图 19 订单管理

## 十、 部分核心代码

### 1. 服务器后端

后端采用 JAVA 语言, SSM 框架编写, 主要有 model、mapper、service、controller 等包结构。一下主要代码中仅列举每个主要包中的一个类的代码。



#### (1) 用户登录检查实现代码

```
package interceptor;

public class AdminInterceptor implements HandlerInterceptor {

    private static Logger logger = Logger.getLogger(AdminController.class);

    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws IOException {
        logger.info("执行 spring 拦截器");
        //检查当前 sessions 是否有登录用户
        User user = (User) request.getSession().getAttribute("user");
        if(user != null)
            return true;
        else
            response.sendRedirect(request.getContextPath()+"/login.jsp");
    }
}
```

```

        return false;
    }

    @Override
    public void postHandle(javax.servlet.http.HttpServletRequest request,
        javax.servlet.http.HttpServletResponse response, Object handler, ModelAndView modelAndView)
        throws Exception {

        logger.info("postHandle");
    }

    @Override
    public void afterCompletion(HttpServletRequest request, HttpServletResponse response,
        Object handler, Exception e) throws Exception{
        logger.info("spring 拦截结束执行");
    }
}

```

(2) 客户预定服务端处理代码 (GeneralController.java)

```

package controller;

@Controller
@RequestMapping("/")
public class GeneralController {

    private static Logger logger = Logger.getLogger(OrderController.class);
    private ModelAndView modelAndView = new ModelAndView();

    @Autowired
    UserService userService;

    @Autowired
    CategoryService categoryService;

    @Autowired
    OrderItemService orderItemService;

    @Autowired
    OrderService orderService;

    @Autowired
    ProductService productService;

    @Autowired
    ReviewService reviewService;
}

```

```

    @RequestMapping(value = "/forelogin", produces = "text/html;charset=UTF-8")
    @ResponseBody
    public ModelAndView logIn(HttpServletRequest request, HttpServletResponse
response, String name, String password) throws Exception{
        User user = userService.getUser(name);
        if(user != null && user.getPassword().equals(password)){
            //登陆成功
            request.getSession(true).setAttribute("user", user);
            logger.info(request.getPathInfo());
            return home(request);
        }
        return null;
    }

    @RequestMapping(value = "/forelogout", produces = "text/html;charset=UTF-8")
    @ResponseBody
    public void logOut(HttpServletRequest request, HttpServletResponse response)
throws Exception{
        request.getSession().invalidate();
        response.sendRedirect("home.jsp");
    }

    @RequestMapping(value = "/foresearch", produces = "text/html;charset=UTF-8")
    @ResponseBody
    public ModelAndView search(HttpServletRequest request, HttpServletResponse
response, String keyword) throws Exception{
        List<Product> ps = productService.listProducts(keyword);

        modelAndView.addObject("ps", ps);
        modelAndView.setViewName("/searchResult");
        return modelAndView;
    }

    @RequestMapping(value = "/foreproduct", produces = "text/html;charset=UTF-8")
    @ResponseBody
    public ModelAndView product(Integer pid) throws Exception{
        Product p = productService.getProduct(pid);
        List<Propertyvalue> pvs = productService.listProprotyValue(p.getId());
        modelAndView.addObject(pvs);
        modelAndView.addObject("p", p);
        modelAndView.setViewName("/product");
    }

```

```

        return modelAndView;
    }

    @RequestMapping(value = "/forecheckLogin", produces = "text/html;charset=UTF-8")
    @ResponseBody
    public int check(HttpServletRequest request) throws Exception{
        User user = (User) request.getSession().getAttribute("user");
        if(user != null)
            return 1;
        return 0;
    }

    @RequestMapping(value = "/forebuyone", produces = "text/html;charset=UTF-8")
    @ResponseBody
    public ModelAndView forebuyone(HttpServletRequest request, Orderitem orderitem)
    throws Exception{
        User user = (User) request.getSession().getAttribute("user");
        Map map = orderItemService.buyOne(user, orderitem);
        request.getSession(true).setAttribute("order", map.get("order"));
        request.getSession(true).setAttribute("pid", orderitem.getPid());
        Product product = productService.getProduct(orderitem.getPid());
        product.setSaleCount(product.getSaleCount()+orderitem.getNumber());
        productService.update(product);
        modelAndView.addObject("ois", map.get("ois"));
        modelAndView.addObject("total", ((Order)map.get("order")).getTotal());
        modelAndView.setViewName("/buy");
        return modelAndView;
    }

    @RequestMapping(value = "/home", produces = "text/html;charset=UTF-8")
    @ResponseBody
    public ModelAndView home(HttpServletRequest request) throws Exception{
        List<Category> c = categoryService.listTwoCategory();
        for(Category cs: c){
            cs.setProducts(productService.listProducts(cs.getId()));
        }
        modelAndView.addObject("cs", c);
        modelAndView.setViewName("home");
        return modelAndView;
    }

```



```

    }

    @RequestMapping(value = "/forecreateOrder", produces = "text/html; charset=UTF-8")
    @ResponseBody
    public ModelAndView buy(Order order, HttpServletRequest request) throws
Exception{
        /**确定支付**/
        Order origin_order = (Order)request.getSession().getAttribute("order");
        origin_order.setOrderCode(UUID.randomUUID().toString().replace("-",
"").toLowerCase());
        origin_order.setReceiver(order.getReceiver());
        origin_order.setMobile(order.getMobile());
        origin_order.setUserMessage(order.getUserMessage());
        origin_order.setStatus("waitReview");
        origin_order.setPayDate(new Date());

        orderService.update(origin_order);

        modelAndView.addObject("o", origin_order);
        modelAndView.setViewName("/confirmPay");
        return modelAndView;
    }

    @RequestMapping(value = "/foreorderConfirmed", produces = "text/html;
charset=UTF-8")
    @ResponseBody
    public ModelAndView confirm() throws Exception{
        modelAndView.setViewName("orderConfirmed");
        return modelAndView;
    }

    @RequestMapping(value = "/forebought", produces = "text/html; charset=UTF-8")
    @ResponseBody
    public ModelAndView bought(HttpServletRequest request) throws Exception{
        modelAndView.setViewName("bought");
        User user = (User) request.getSession().getAttribute("user");
        modelAndView.addObject("os", orderService.getOrders(user));
        return modelAndView;
    }

    @RequestMapping(value = "/deleteorder", produces = "text/html; charset=UTF-8")

```

```

@ResponseBody
public void deleteorder(HttpServletRequest request, HttpServletResponse response,
Integer oid) throws Exception{
    User user = (User) request.getSession().getAttribute("user");
    orderService.deleteOrder(oid);
    response.sendRedirect("forebought");
}

@RequestMapping(value = "/forestory", produces = "text/html; charset=UTF-8")
@ResponseBody
public ModelAndView story() throws Exception{
    modelAndView.setViewName("story");
    return modelAndView;
}

@RequestMapping(value = "/forereview", produces = "text/html; charset=UTF-8")
@ResponseBody
public ModelAndView review(int oid, HttpServletRequest request) throws Exception{
    Product product =
productService.getProduct((int)request.getSession().getAttribute("pid"));
    List<Review> reviews = reviewService.listReview(product.getId());
    modelAndView.addObject("re", reviews);
    modelAndView.addObject("p", product);
    modelAndView.addObject("u", request.getSession().getAttribute("user"));
    modelAndView.setViewName("review");
    return modelAndView;
}

@RequestMapping(value = "/foredoreviewed", produces = "text/html; charset=UTF-8")
@ResponseBody
public ModelAndView reviewed(Review review, HttpServletRequest request) throws
Exception{
    reviewService.insert(review);
    Product product = productService.getProduct(review.getPid());
    product.setReviewCount(product.getReviewCount()+1);
    productService.update(product);
    return bought(request);
}
}

```

(3) 对 User 表的数据库操作 (UserMapper.java)

本系统采用 mybatis 自动生成基本数据库操作代码, 并且被封装在 xml 文件中, 因此, 仅对部分复杂数据库操作才需要重写 SQL 语句, 并且以注解方式写在 Mapper 的接口中。

```
package mapper;

import java.util.List;
import model.User;
import model.UserExample;
import org.apache.ibatis.annotations.Param;
import org.apache.ibatis.annotations.Select;
import org.springframework.stereotype.Repository;

@Repository
public interface UserMapper {

    long countByExample(UserExample example);

    int deleteByExample(UserExample example);

    int deleteByPrimaryKey(Integer id);

    int insert(User record);

    int insertSelective(User record);

    List<User> selectByExample(UserExample example);

    @Select("select * from User")
    List<User> selectAll();

    User selectByPrimaryKey(Integer id);

    //使用注解方式执行查询代码
    @Select("select * from User where name = #{name}")
    User selectByName(String name);

    int updateByExampleSelective(@Param("record") User record, @Param("example")
UserExample example);

    int updateByExample(@Param("record") User record, @Param("example") UserExample
example);
```

```
int updateByPrimaryKeySelective(User record);

int updateByPrimaryKey(User record);
}
```

(4) User 模型类，用于 Spring 框架的注入 (User.java)

```
package service;

import mapper.*;
import model.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class UserService implements UserInt {

    @Autowired()
    ProductMapper productMapper;

    @Autowired
    ProductImageMapper productImageMapper;
    @Autowired
    OrderItemMapper orderItemMapper;
    @Autowired
    OrderMapper orderMapper;
    @Autowired
    ReviewMapper reviewMapper;
    @Autowired
    UserMapper userMapper;

    //根据用户名查询用户
    @Override
    public User getUser(String name) {
        return userMapper.selectByName(name);
    }

    //根据用户 id 查询用户
}
```

```

@Override
public User getUser(int id) {
    return userMapper.selectByPrimaryKey(id);
}

//查询所有用户
@Override
public List<User> listAll() {
    return userMapper.selectAll();
}

//通过 id 删除用户
@Override
public void deleteByPrimaryKey(Integer id) {
    userMapper.deleteByPrimaryKey(id);
}

//新增用户
@Override
public void insert(User user) {
    userMapper.insert(user);
}

//更新用户信息
@Override
public void update(User user) {
    userMapper.updateByPrimaryKey(user);
}
}

```

## 2. 分页显示

```

package util;

public class Page {
    private int start;
    private int count;
    private int total;
    private String param;
    public int getStart() {
        return start;
    }
}

```

```

    }

    public void setStart(int start) {
        this.start = start;
    }

    public int getCount() {
        return count;
    }

    public void setCount(int count) {
        this.count = count;
    }

    public Page(int start, int count) {
        super();
        this.start = start;
        this.count = count;
    }

    public boolean isHasPrevious() {
        return start != 0;
    }

    public boolean isHasNext() {
        return start != getLast();
    }

    public int getTotalPage() {
        int totalPage;
        // 假设总数是 50, 是能够被 5 整除的, 那么就有 10 页
        if (0 == total % count)
            totalPage = total / count;
        // 假设总数是 51, 不能够被 5 整除的, 那么就有 11 页
        else
            totalPage = total / count + 1;

        if(0==totalPage)
            totalPage = 1;
        return totalPage;
    }

    public int getLast() {
        int last;

```

```

        // 假设总数是 50, 是能够被 5 整除的, 那么最后一页的开始就是 45
        if (0 == total % count)
            last = total - count;
        // 假设总数是 51, 不能够被 5 整除的, 那么最后一页的开始就是 50
        else
            last = total - total % count;

        last = last < 0 ? 0 : last;
        return last;
    }

    public int getTotal() {
        return total;
    }

    public void setTotal(int total) {
        this.total = total;
    }

    public String getParam() {
        return param;
    }

    public void setParam(String param) {
        this.param = param;
    }
}

```

### 3. 图片上传实现代码

```

//添加分类
//特别注意, 由于本操作涉及图片保存, 更换运行环境是需要重新配置路径
@RequestMapping(value = "/admin_category_add", produces =
"text/html;charset=UTF-8")
@ResponseBody
public ModelAndView admin_category_add(@RequestParam("name") String name,
@RequestParam("filepath") MultipartFile filepath) throws IOException {
    Category category = new Category();
    category.setName(name);
    Integer cid = categoryService.insert(category);
    String path1 =
"C:\\Users\\70953\\github\\Gotrip\\src\\main\\webapp\\img\\category\\"+cid.toS

```

```
tring()+".jpg";
    String path2 =
"C:\\Users\\70953\\github\\Gotrip\\target\\Gotrip\\img\\category\\"+cid.toStri
ng()+".jpg";
    filepath.transferTo(new File(path1));
    FileUtils.copyFile(new File(path1), new File(path2));
    mav = listCategory();
    return mav;
}
```



## 十一、参考文献

- (1) 安亚琴, 基于地方性视角的民宿发展研究——以台州市黄岩区为例[J], 经贸实践, 2018 年 23 期
- (2) 张祖铭、张月忠、袁一平, 基于 SSM 框架的科研管理系统的分析与设计[J], 民营科技, 2018 年 12 期
- (3) 吴集林、莫颖均, 数据库关系范式的几个注记及图解规范化方法[J], 电脑知识与技术, 2018 年 27 期
- (4) 王茜, 数据库优化技术的分析与探讨[J], 中国管理信息化, 2016 年 24 期
- (5) 孙军、袁梓焜、王佳、陈晴、马宝龙、刘蕊, 基于 UI 设计的社交软件界面优化项目研究[J], 项目管理技术, 2018 年 10 期