**目录**

[**1 绪 论 1**](#_Toc6666)

[1.1 研究的背景及意义 1](#_Toc31676)

[1.1.1 选题的背景 1](#_Toc32544)

[1.1.2 国内外研究现状 1](#_Toc8760)

[1.1.3 研究的意义 2](#_Toc32026)

[1.2 系统目标 2](#_Toc31782)

[**2 需求分析 3**](#_Toc31574)

[2.1 业务需求 3](#_Toc18042)

[2.1.1主要业务流程 3](#_Toc4575)

[2.2 功能需求 10](#_Toc21613)

[2.2.1 角色分析 10](#_Toc23805)

[2.2.2 业务功能 10](#_Toc22354)

[2.3 非功能需求 22](#_Toc24118)

[2.3.1环境需求 22](#_Toc4621)

[2.3.2 性能需求 23](#_Toc24180)

[2.3.3 安全需求 23](#_Toc15662)

[**3 总体设计 24**](#_Toc8740)

[3.1系统设计的原则 24](#_Toc18691)

[3.2 系统体系结构设计 24](#_Toc1935)

[3.3 系统功能结构设计 25](#_Toc8260)

[**4 数据库设计 27**](#_Toc11708)

[4.1逻辑设计 27](#_Toc1179)

[4.1.1实体关系设计 27](#_Toc725)

[4.1.2实体属性设计 27](#_Toc15763)

[4.1.3逻辑表格设计 29](#_Toc29510)

[4.2物理结构设计 31](#_Toc19928)

[4.2.1数据库存取方式 31](#_Toc723)

[4.2.1存储过程设计 32](#_Toc11965)

[**5 界面设计 33**](#_Toc16283)

[5.1 界面关系图或工作流图 33](#_Toc12185)

[5.2 界面设计成果 34](#_Toc5768)

[5.2.1 主界面 34](#_Toc8435)

[5.2.2 子界面 34](#_Toc22589)

[**6 详细设计 38**](#_Toc17127)

[6.1 系统主要功能模块介绍 38](#_Toc24117)

[6.2小程序端功能模块设计 38](#_Toc10165)

[6.2.1功能结构设计 38](#_Toc17128)

[6.2.2 功能类设计 39](#_Toc27750)

[6.2.3 时序图设计 49](#_Toc6166)

[6.3网站后台管理功能模块设计 52](#_Toc4155)

[6.3.1功能结构设计 52](#_Toc6943)

[6.3.2功能类设计 54](#_Toc12714)

[6.3.3时序图设计 61](#_Toc21712)

[**7 编码 64**](#_Toc20924)

[7.1 代码实现与核心算法 64](#_Toc6101)

[7.2 代码优化分析 66](#_Toc32541)

[7.2.1 代码优化的原则 66](#_Toc3063)

[7.2.2 系统优化示例 67](#_Toc3848)

[**8 测试 68**](#_Toc29674)

[8.1 测试方案设计 68](#_Toc31284)

[8.1.1 测试策略 68](#_Toc2812)

[8.1.2 测试进度安排 68](#_Toc26207)

[8.1.3 测试资源 68](#_Toc239)

[8.1.4 关键测试点 69](#_Toc26947)

[8.2 测试用例构建 70](#_Toc3503)

[8.2.1 测试用例编写约定 70](#_Toc14306)

[8.2.2 测试用例设计 70](#_Toc9170)

[8.2.3 测试用例维护 72](#_Toc19861)

[**9 总结与展望 74**](#_Toc16387)

[9.1 设计工作总结 74](#_Toc12360)

[9.2 未来工作展望 74](#_Toc1214)

[**谢 辞 75**](#_Toc25532)

[**参考文献 76**](#_Toc3345)

[**附录A 外文翻译—原文部分 78**](#_Toc27602)

[**附录B 外文翻译—译文部分 83**](#_Toc30269)

[**附录C 软件使用说明 8**](#_Toc30269)**6**

[**附录D 主要源代码 8**](#_Toc30269)**8**

1 绪 论

## 1.1 研究的背景及意义

### 1.1.1选题的背景

随着社会的不断发展，人们生活水平的不断提高，人们出行的交通工具也跟随社会的发展不断革新。乘坐轮渡外出旅游的人数也不断提升，单凭线下购票的渠道已经不能满足人们的出行需求，所以在线购票的开发势在必行。而开发该微信小程序，不仅可以为广大用户的日常出行提供了极大的便利，还可以加速轮渡行业的发展，为当地的经济发展提供一份动力。除此之外，小程序可以转发到微信圈，微信群，微博等社交平台分享给好友，通过客户各自的社交渠道推广小程序，可以以此提高品牌推广；小程序还具有数据统计分析功能，可以帮助企业建立自己的大数据库，精准掌握用户消费数据，市场发展动向，为企业运营决策提供数据支持。

又随着网络的普及，中国的网民数量已超过九亿。现在很多人都选择了方便快捷的网络购票方式，足不出户即可买到火车票、船票及飞机票等。相比传统的购票方式，网络购票脱离了时空的园囿，很多老百姓因此而受益，他们充分享受到了网络购票所给予的便利。网络购票做到了公平公正，没有了插队、窗口没有了整夜排队的现象，如今网络购票中，频频出现了网络抢票这个很流行的词，高科技的抢票软件弄得人心慌慌。也许这样的事情也存在，但仅仅是少一部分，对于大多数人来讲网络购票带来很大的收益，网购的方便是事实存在的，这一新型购票的方式其主要就是为了让旅客顺利购到出行的票。  
　　随着网络逐渐成为人们生活不可缺少的一部分，微信小程序的飞速发展，相关购票微信小程序成为了大家买车票的最方便，最快捷的工具，告别了以往逢年过节买车票成天连夜的排长队、托关系、找熟人的混乱局面。由于票可以在网上预定，是直销方式，所以运营成本很低，这样网络运营者，站台，我们三方就是一种共赢局面。站台的工作量也因此减少很多，而且环境保安等各方面都可以得到最佳控制。我们出行时可以悠然地走进站台，而且很舒适。

### 1.1.2国内外研究现状

电子商务于 90 年代初兴起于美国，加拿大等国，但在近几年电子支付才被人们普遍接受。各厂商如 IBM，惠普，微软，SUN 等纷纷推出自己的电子商务产品和各自 的解决方案。随着电子商务的发展， 各种法规也随之健全，德国，韩国，意大利，西班牙和美国的许多州已经通过数字签名和身份认证法律。1996 年下半年，美国财政部颁布有关《全球电子商务选择税收政策》 白皮书；联合国国际贸易法委员会（ Uncitral）已经完成模型电子商务法的制定工作，为电子交易制订出统一通用的规则。另外，两大国际信用卡组织 Visa 和 MasterCard合作制订的安全电子交易（Set）协议定义了一种电子支付过程标准，其目的就是保护互联网上支付卡交易的每一个环节。Set 是专为网上支付卡业务安全所制定的标准。

我国在八十年代以前随着计算机的普及和各种商业软件的发展、电子商务初见端倪， 出现了商业数据的无纸化处理。 在这一阶段许多商业数据通过磁介质而非纸介质进行交换。总体而言，我国电子商务的宣传和兴起比国外晚一、两年，但发展十分迅速。最新数字统计，时至 2020 年，我国上网人数超过了9 亿，国内 Internet 用户的发展速度非常快， 这种增长速度在国际上也是绝对领先的。

国外电子商务于 90 年代初兴起于美国等，国外的诸多电子支付系统发展的相对比较完善。 许多知名公司不仅开发了功能强大的电子支付系统而且还提供比较完善的服务。 如IBM、惠普、微软、SUN 等纷纷推出自己的电子商务产品和各自的解决方案。

### 1.1.3 研究的意义

通过实施该项目 ，为当地的经济发展提供一份动力，提高公司信息化水平，帮助企业建立自己的大数据库，精准掌握用户消费数据、市场发展动向，提供人性化客户服务，深入挖掘市场潜力，为企业运营决策提供数据支持，为企业创造更大的价值。

## 1.2 系统目标

开发一个简单、稳定、易用的购票平台，为广大用户提供高效、快捷的轮渡出行服务。本小程序应满足的主要功能有：

1、旅客信息输入：能将旅客的信息，如:姓名、乘客人数、各乘客的证件类型和证件号码、联系电话、出发时间、出发地点、抵达地点等。

2、旅客信息修改：能对以上的旅客各信息的修改或删除功能。

3、信息核对功能：系统要能通过对公安部门的信息联网对旅客各项信息进行核对。

4、信息查询功能：旅客能查询到各个班次的轮渡情况。

5、打印输出功能：旅客交付出行费用后，系统即可打印船票给旅客。

6、变更权限功能：管理员可以变更相应系统权限。

7、旅客查看出行记录：能将旅客的出行记录展现出来并可以通过条件查询展示相应的出行记录。

8、订单展示功能：能将旅客的已结算订单信息展现出来并可以通过条件查询展示相应的已结算订单信息。

9、票务展示功能：能将轮渡的订单信息展现出来并可以通过条件查询展示相应的票务信息。

10、天气查询功能：能将旅客所在地的天气信息展现出来并可以通过条件查询展示相应目的地的信息。

2 需求分析

## 2.1 业务需求

（1）能够实现船票展示、船票检索、船票选择、网上订货、网上支付和船票打印等功能，对网上购票的全过程进行管理。

（2）实现对网上购票过程中产生的所有业务数据的管理，如订货单、支付记录、发货信息的管理与维护。

（3）具有配套的系统后台管理维护功能，能够对船票信息、用户信息、系统日志等信息进行管理与维护，并能够进行对应的权限管理。

（4）响应速度合理，安全性较高。

（5）系统运行稳定，并且应易于维护。

### 2.1.1主要业务流程

（1）游客添加订单业务

如图2-1所示为游客添加订单业务图：

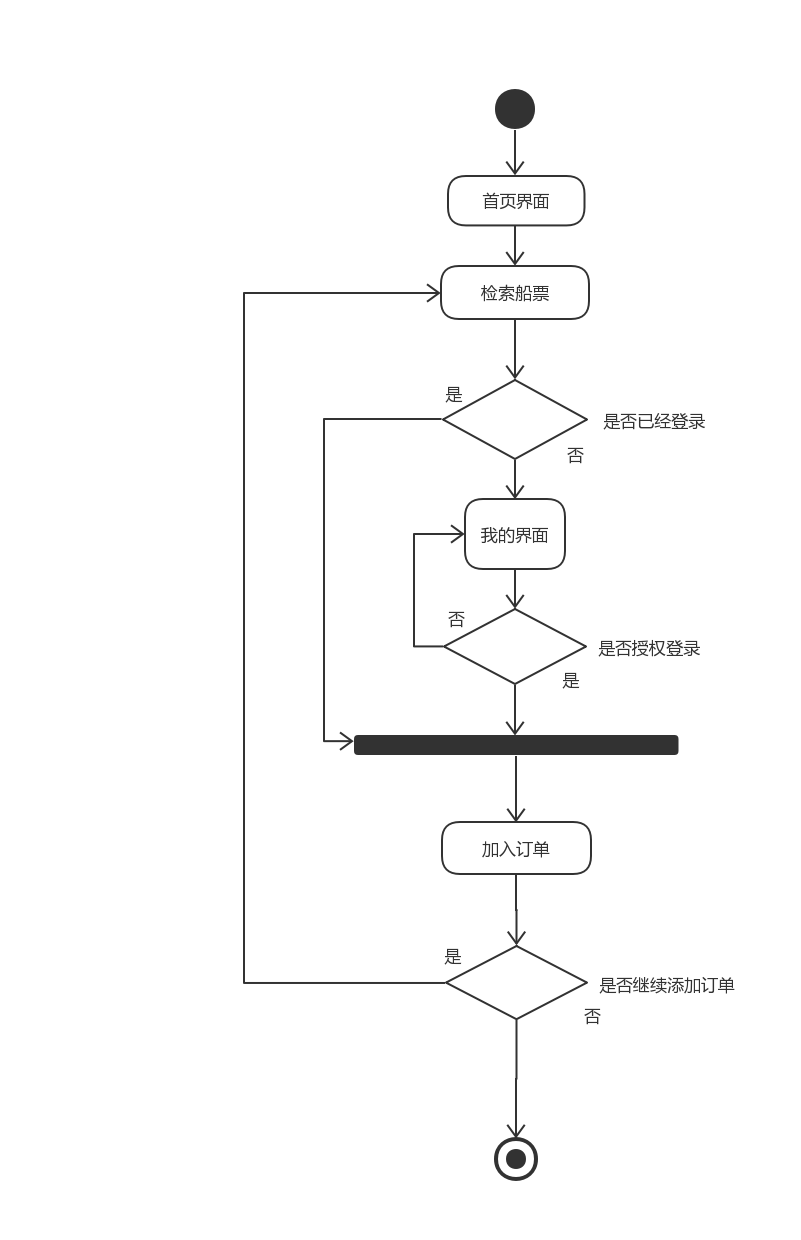


图2-1添加订单业务流程

1. 游客在线改签业务

如图2-2所示为游客在线改签业务图：

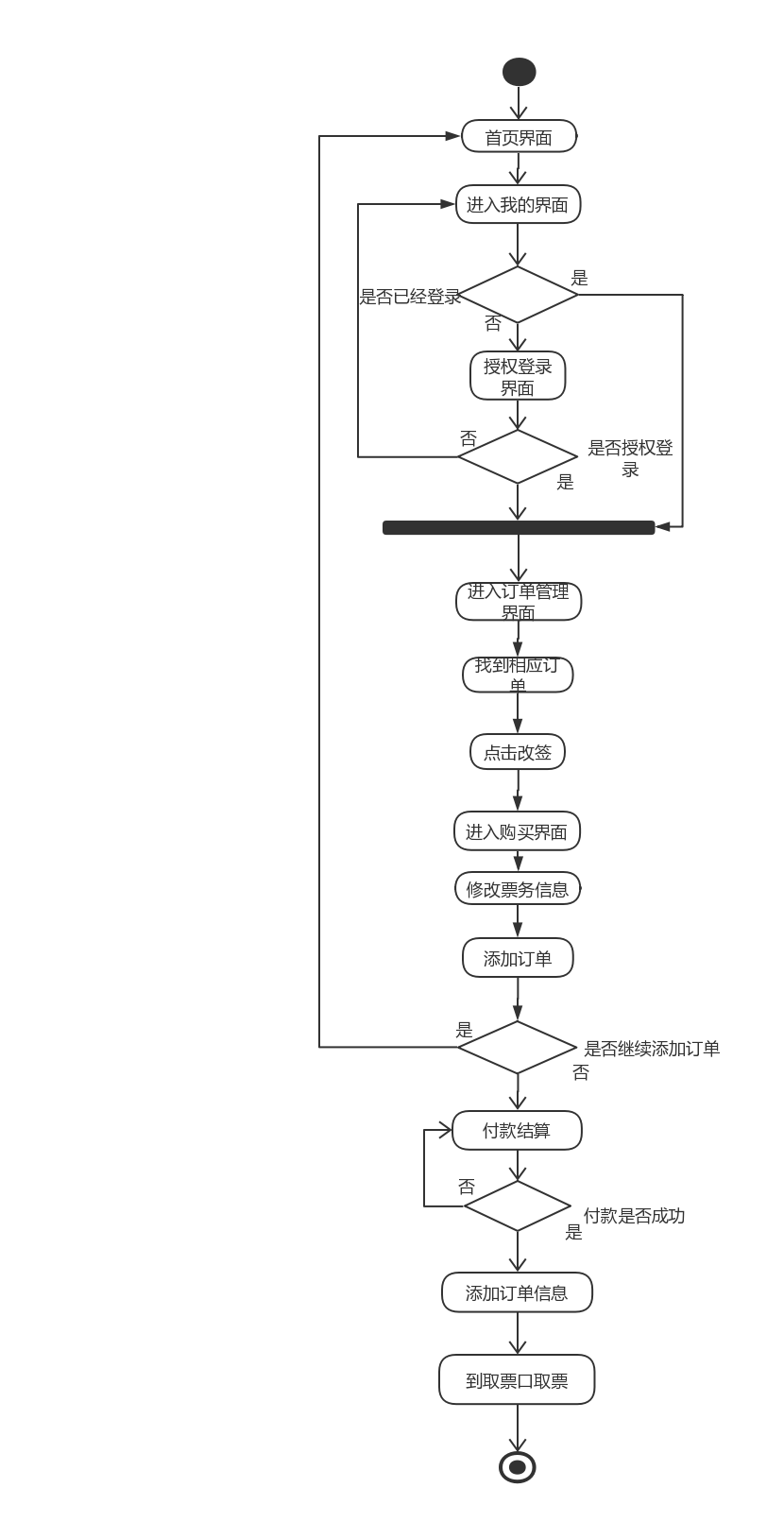


图2-2网上改签业务流程

（3）游客信息修改业务

如图2-3所示为游客信息修改业务图：

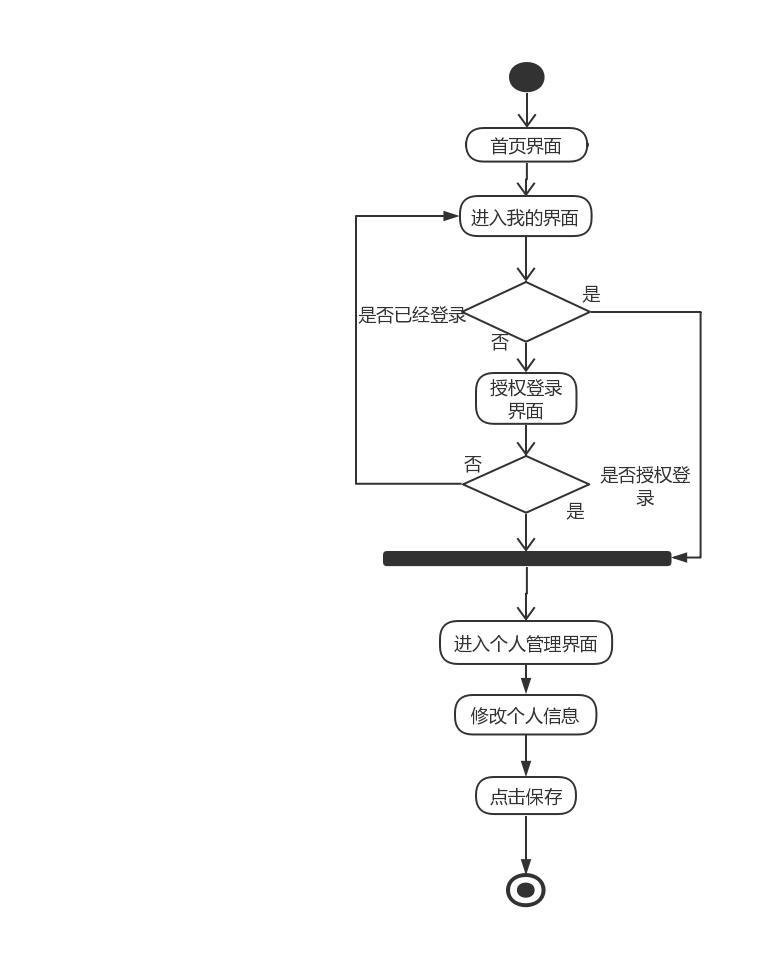


图2-3游客信息修改业务图

（4）订单支付业务

如图2-4所示为订单支付业务图：

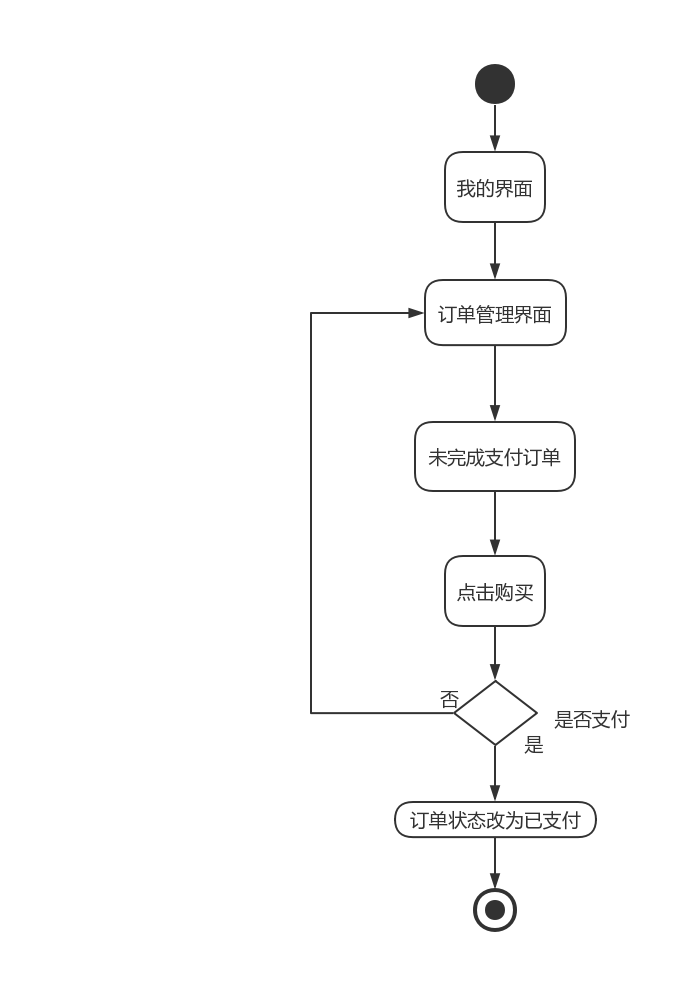


图2-4订单支付流程图

（5）授权登录业务

如图2-5所示为授权登录业务图：

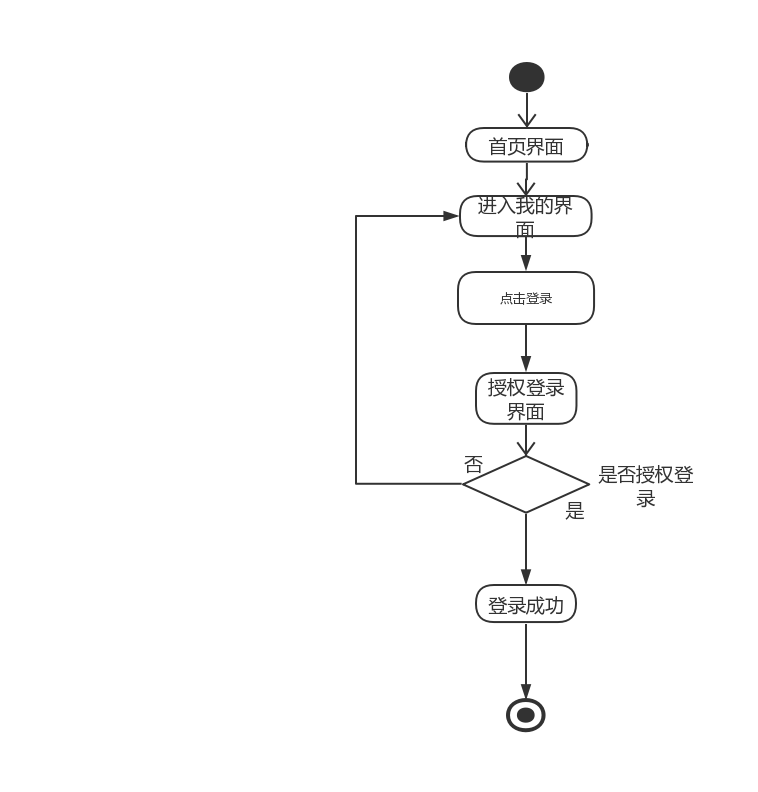


图2-5授权登录流程图

（6）票务信息查询业务

如图2-6所示为票务信息查询业务图：

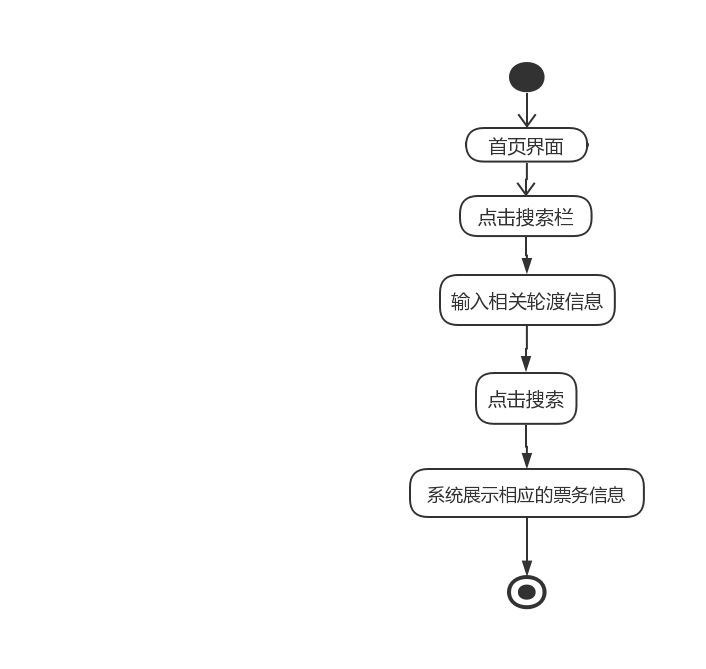


图2-6票务信息查询流程图

（7）查看出行记录业务

如图2-7所示为查看出行记录业务图：

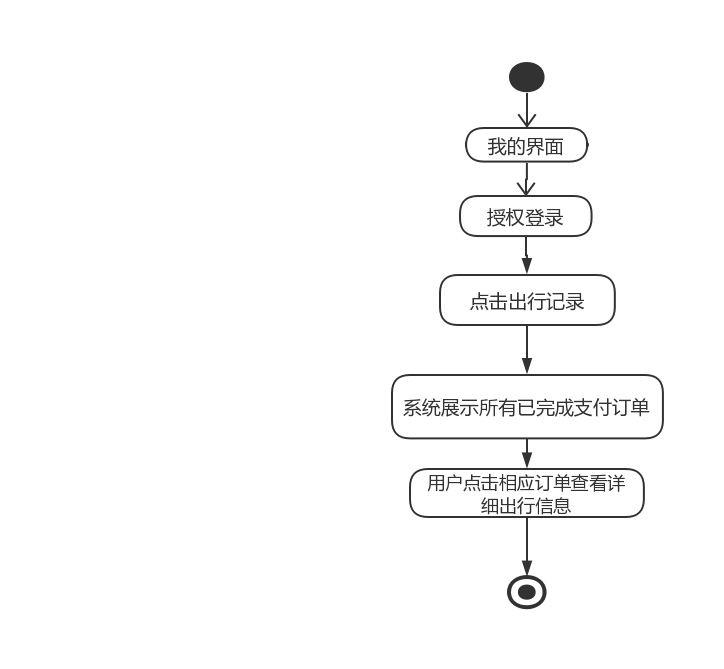


图2-7查看出行记录业务流程图

（8）天气查询业务

如图2-8所示为天气查询业务图：

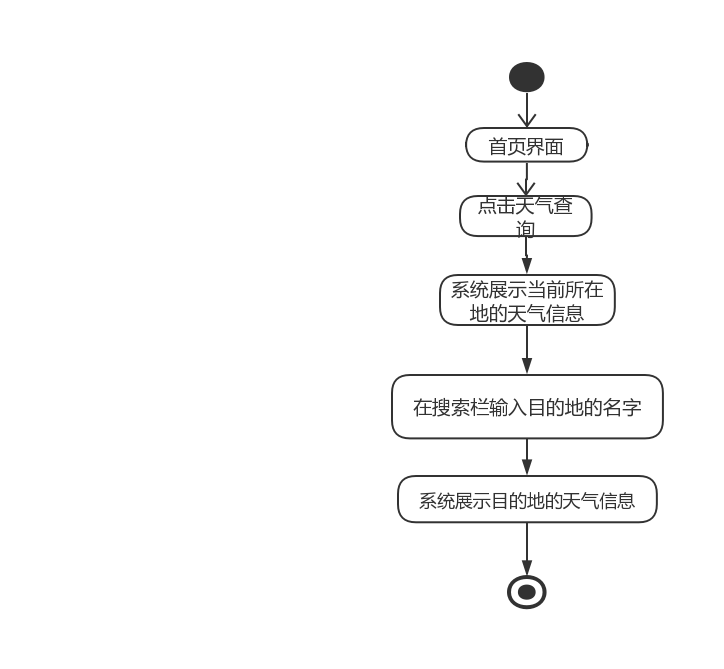


图2-8天气查询业务流程图

## 2.2 功能需求

### 2.2.1 角色分析

从网上购票系统的实际需求分析，系统涉及到角色如下表2-1所示：

|  |  |
| --- | --- |
| 角色 | 职责或功能 |
| 客户(买家) | 该系统的服务对象，通过游客的操作推动系统的主要运作 |
| 系统管理员 | 管理和维护整个系统的用户组织结构，负责对用户、角色、用户级别的增、删、改、查等管理。 |

表2-1 系统角色表

### 2.2.2 业务功能

从业务角度出发，图2-9给出了系统的总体用例图，包含用户信息管理、授权登录、天气查询、票务查询、票务查询、订单管理、出行记录、轮渡信息管理以及系统权限管理等用例，如下图2-9所示：

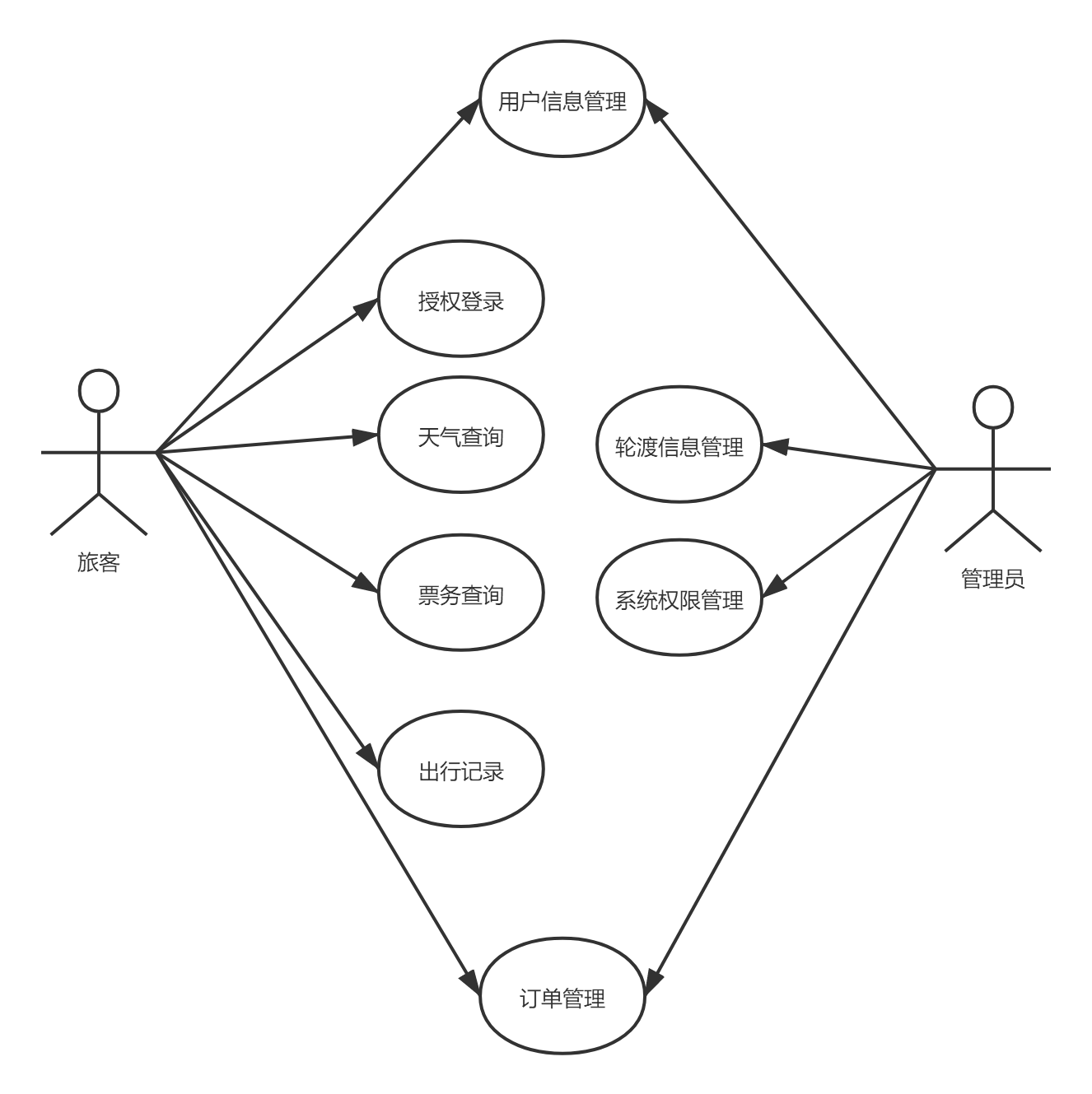
**

图2-9系统总体用例图

2.2.2.1小程序端子系统

如下图2-10所示为小程序端子系统用例图：

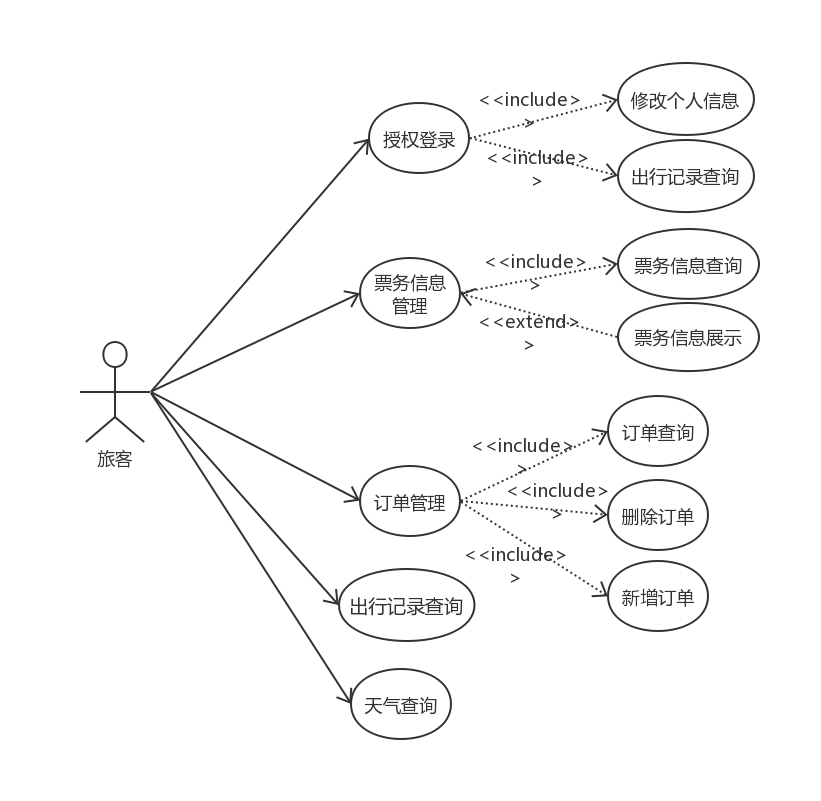
**

图2-10小程序端子系统用例图

（1）授权登录用例表如表2-2所示：

表2-2 授权登录用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0001 | 用例名称 | 授权登录 |
| 用例描述 | 授权登录是游客通过自己绑定的微信进行登录操作，只需自己授予权限，相应的数据信息会从微信中获取并保存在小程序的数据库中。 | | |
| 执行者 | 游客 | | |
| 前置条件 | 无 | | |
| 后置条件 | 返回“首页”界面 | | |
| 涉众利益 | 1.使用者担心个人信息泄露  2.用户希望登录成功后能够记住账号密码 | | |
| 用例场景 | 系统授权登录界面 | | |

续表2-2

|  |  |
| --- | --- |
| 基本事件流 | 基本流程（Base Flow）  1.游客授权登录  2.系统从微信中获取相应的数据  3.系统将获取的数据新增到小程序的用户数据库中  4.将获取的数据提交到服务器验证  5.系统根据服务器验证结果进行相应操作  6.如果验证是not found 404，直接提示刷新页面，重新授权  7.如果验证通过，直接登录到系统首页界面。 |
| 编写人 | 莫平平 |

（2）票务信息管理用例表如表2-3所示：

表2-3 票务信息管理用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0002 | 用例名称 | 票务信息管理 |
| 用例描述 | 游客进入“首页“”界面，点击“票务信息”，进入票务信息界面，进入票务信息界面后，系统向游客展示所有的票务信息，像剩余的票的张数，始发时间等，并且向游客提供查询指定票务信息的功能。 | | |
| 执行者 | 游客 | | |
| 前置条件 | 进入“首页”界面，点击“票务信息”，进入票务信息界面 | | |
| 后置条件 | 点击“购买”按钮进入购买界面 | | |
| 涉众利益 | 1.使用者希望能够有多种方法快速查询到自己想要的票 | | |
| 用例场景 | 游客查询购买票务 | | |
| 基本事件流 | 基本流程（Base Flow）  1.游客进入票务信息界面  2.系统展示所有的票务信息  3.游客选择自己的票并点击购买  4.系统跳转到购买界面 | | |
| 编写人 | 莫平平 | | |

（3）订单管理用例表如表2-4所示：

表2-4 订单管理用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0003 | 用例名称 | 订单管理 |
| 用例描述 | 游客通过进入“我的”界面，点击“订单管理”，进入订单管理界面；进入订单管理界面后，游客可以查看到自己所有的订单情况，并可以查询自己的订单信息、删除订单和新增订单。 | | |
| 执行者 | 游客 | | |
| 前置条件 | 进入“我的”界面，点击“订单管理”，进入订单管理界面； | | |
| 后置条件 | 点击“查找”按钮展示查找的票；点击“删除”按钮删除相应的订单；点击“新增”按钮跳转到票务信息界面；点击“改签”按钮改签订单：  点击“购买”按钮完成未支付的订单 | | |
| 涉众利益 | 1. 游客希望能够有多种方法快速查询到自己想要的订单信息 2. 游客能够快速删除自己不需要的订单信息 3. 游客可以快速改签票 | | |
| 用例场景 | 游客查询订单信息 | | |
| 基本事件流 | 基本流程（Base Flow）  1.游客进入订单管理界面  2.系统展示游客所有的订单信息  3.游客输入相应的信息，点击“查找”按钮  4.系统查找并展示符合信息的订单  5.点击“删除”按钮删除相应的订单  6.点击“新增”按钮跳转到票务信息界面，添加新的订单  7.点击“改签”按钮跳转到票务信息页面，修改相应信息提交订单完成支付即可  8.点击“购买”按钮跳转到支付界面，完成支付，订单从未支付状态转变为已支付状态 | | |
| 编写人 | 莫平平 | | |

（4）票务信息管理用例表如表2-5所示：

表2-5 天气查询用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0004 | 用例名称 | 天气查询 |
| 用例描述 | 游客通过进入“首页”界面，点击“天气查询”，进入天气查询界面；进入天气查询界面后，游客可以查看到自己现在所在地的天气，并可以通过在搜索框输入目的地的名字，点击“搜索”按钮可以查询到目的地的天气信息。 | | |
| 执行者 | 游客 | | |
| 前置条件 | 进入“首页”界面，点击“天气查询”，进入天气查询界面； | | |
| 后置条件 | 点击“搜索”按钮展示目的地的天气信息 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 游客查询天气信息 | | |
| 基本事件流 | 基本流程（Base Flow）  1.游客进入天气查询界面  2.系统展示游客现在所在地的天气信息  3.游客输入目的地的名称并点击“搜索”按钮  4.系统搜索并展示目的地的天气信息 | | |
| 编写人 | 莫平平 | | |

（5）票务信息管理用例表如表2-6所示：

表2-6 用户管理用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0007 | 用例名称 | 用户管理 |
| 用例描述 | 游客在个人管理界面可以查看到自己的信息，并可以对自己的信息作出修改 | | |
| 执行者 | 游客 | | |
| 前置条件 | 进入我的界面，点击“个人管理”，进入个人管理界面； | | |
| 后置条件 | 点击“保存”按钮更新用户的信息； | | |
| 涉众利益 | 无 | | |
| 用例场景 | 游客更改个人信息 | | |

续表2-6

|  |  |
| --- | --- |
| 基本事件流 | 基本流程（Base Flow）  1.用户进入小程序“我的”界面，点击“个人管理”  2.进入个人管理界面  3.游客在个人管理界面修改自己的个人信息  4.游客点击“保存”按钮  5.系统更新相应的信息 |
| 编写人 | 莫平平 |

（6）票务信息管理用例表如表2-7所示：

表2-7 出行记录查询用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0010 | 用例名称 | 出行记录查询 |
| 用例描述 | 游客通过进入“首页”界面，点击“出行记录查询”，进入出行记录查询界面；进入出行记录查询界面后，游客可以查看到自己所有已完成支付的订单信息，点击订单，可以查看到详细的出行信息。 | | |
| 执行者 | 游客 | | |
| 前置条件 | 进入“首页”界面，点击“出行记录查询”，进入出行记录查询界面； | | |
| 后置条件 | 点击订单查看详细的出行信息 | | |
| 涉众利益 | 无 | | |
| 用例场景 | 游客查询出行记录 | | |
| 基本事件流 | 基本流程（Base Flow）  1.游客进入出行记录查询界面  2.系统展示游客所有已完成支付的订单信息  3.游客在搜索栏输入订单的相关信息并点击“搜索”按钮  4.系统搜索并展示相应的订单信息  5.点击订单，系统展示出详细的出行记录信息 | | |
| 编写人 | 莫平平 | | |

2.2.2.2网站后台管理子系统

如下图2-11所示位网站后台管理子系统总用例图：

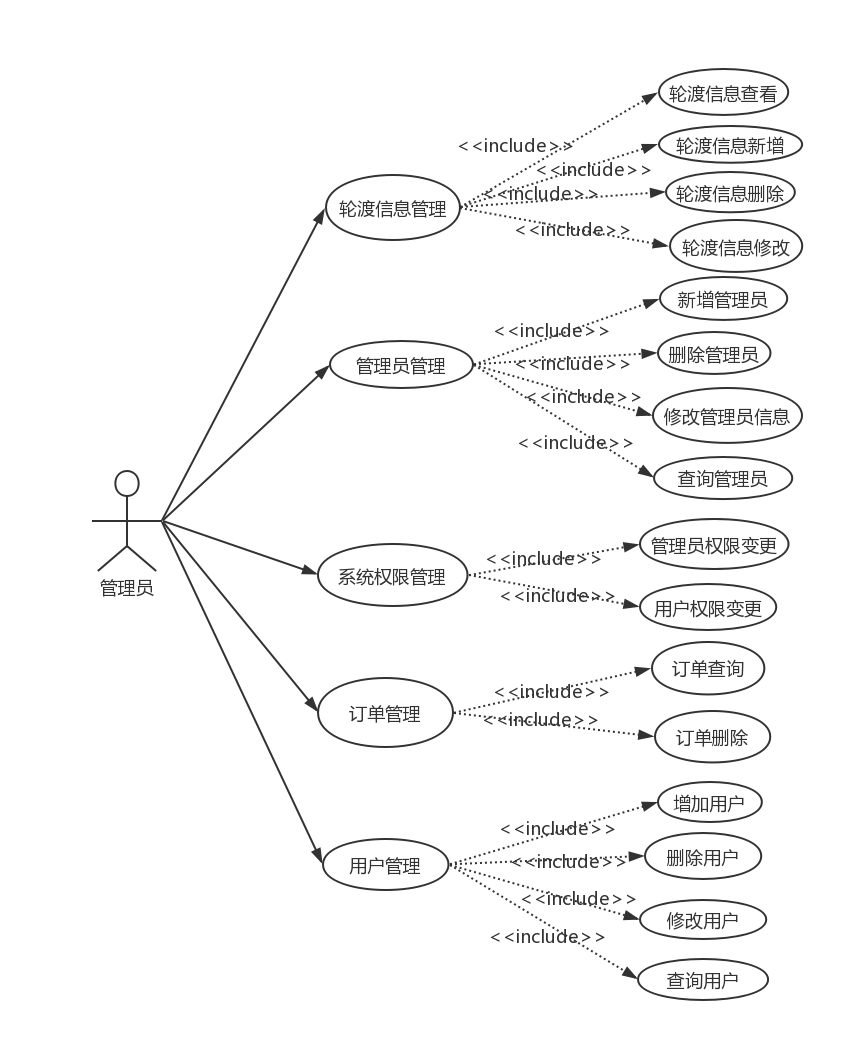


图2-11网站后台子系统用例图

（1）轮渡信息管理用例表如表2-8所示：

表2-8 轮渡信息管理用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0005 | 用例名称 | 轮渡信息管理 |

续表2-8

|  |  |
| --- | --- |
| 用例描述 | 管理员通过网站后台可以查看到所有轮渡的信息，可以对轮渡的相关信息作出修改和删除，并可以增加新的轮渡信息。 |
| 执行者 | 管理员 |
| 前置条件 | 进入后台管理界面，点击“轮渡信息管理”，进入轮渡信息管理界面； |
| 后置条件 | 点击“查找”按钮展示查找的轮渡信息；点击“删除”按钮删除相应的轮渡信息；点击“新增”按钮增加新的轮渡信息；点击“修改”按钮可以修改相应的轮渡信息 |
| 涉众利益 | 无 |
| 用例场景 | 管理员日常管理轮渡信息 |
| 基本事件流 | 基本流程（Base Flow）  1.管理员进入网站后台，输入账号密码  2.系统验证账号密码  3.管理员进入网站后台管理界面，点击“轮渡管理”进入轮渡管理界面  4.管理员输入查找的轮渡信息，点击“查找”按钮  5.系统展示相应的轮渡信息  6.管理员找到符合信息后，点击“删除”按钮删除相应的订单；点击“修改”按钮可以修改相应的轮渡信息  7.管理员点击“新增”按钮增加新的轮渡信息 |
| 编写人 | 莫平平 |

（2）订单管理用例表如表2-9所示：

表2-9 订单管理用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0003 | 用例名称 | 订单管理 |
| 用例描述 | 管理员通过网站后台可以查看到游客所有的订单情况，并可以查询游客的订单信息，可以删除那些违法的订单。 | | |
| 执行者 | 管理员 | | |
| 前置条件 | 进入后台管理界面，点击“订单管理”，进入订单管理界面； | | |

续表2-9

|  |  |
| --- | --- |
| 后置条件 | 点击“查找”按钮展示查找的票；点击“删除”按钮删除相应的订单； |
| 涉众利益 | 快速查找并删除相应的违法订单 |
| 用例场景 | 管理员删除订单 |
| 基本事件流 | 基本流程（Base Flow）  1.管理员进入网站后台，输入账号密码  2.系统验证账号密码  3.管理员进入网站后台管理界面，点击“订单管理”进入订单管理界面  4.管理员输入查找的订单信息，点击“查找”按钮  5.系统展示相应的订单  6.管理员找到符合信息后，点击“删除”按钮删除相应的订单 |
| 编写人 | 莫平平 |

（3）票务信息管理用例表如表2-10所示：

表2-10 管理员管理用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0006 | 用例名称 | 管理员管理 |
| 用例描述 | 管理员通过网站后台可以查看到所有的管理员，并可以对管理员作出增删改查操作 | | |
| 执行者 | 管理员 | | |
| 前置条件 | 进入后台管理界面，点击“管理员管理”，进入管理员管理界面； | | |
| 后置条件 | 点击“查找”按钮展示查找管理员；点击“删除”按钮删除相应的管理员；点击“新增”按钮增加新的管理员；点击“修改”按钮修改管理员的信息； | | |
| 涉众利益 | 无 | | |
| 用例场景 | 管理员管理 | | |

续表2-10

|  |  |
| --- | --- |
| 基本事件流 | 1.管理员进入网站后台，输入账号密码  2.系统验证账号密码  3.管理员进入网站后台管理界面，点击“管理员管理”进入管理员管理界面  4.管理员输入查找的管理员信息，点击“查找”按钮  5.系统展示相应的管理员信息基本流程（Base Flow）  6.管理员找到符合信息后，点击“修改”按钮修改相应的管理员信息  7.点击“新增”按钮添加新的管理员  8.点击“删除”按钮删除相应的管理员 |
| 编写人 | 莫平平 |

（4）票务信息管理用例表如表2-11所示：

表2-11 用户管理用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0007 | 用例名称 | 用户管理 |
| 用例描述 | 管理员通过网站后台可以查看到所有的用户，并可以对用户作出CURD操作 | | |
| 执行者 | 管理员 | | |
| 前置条件 | 进入后台管理界面，点击“用户管理”，进入用户管理界面； | | |
| 后置条件 | 点击“查找”按钮展示查找用户；点击“删除”按钮删除相应的用户；点击“新增”按钮增加新的用户；点击“修改”按钮修改用户的信息； | | |
| 涉众利益 | 无 | | |
| 用例场景 | 管理员管理用户 | | |

续表2-11

|  |  |
| --- | --- |
| 基本事件流 | 基本流程（Base Flow）  1.管理员进入网站后台，输入账号密码  2.系统验证账号密码  3.管理员进入网站后台管理界面，点击“用户管理”进入用户管理界面  4.管理员输入查找的用户信息，点击“查找”按钮  5.系统展示相应的管理员信息  6.管理员找到符合信息后，点击“修改”按钮修改相应的用户信息  7.点击“新增”按钮添加新的用户  8.点击“删除”按钮删除相应的用户 |
| 编写人 | 莫平平 |

（5）票务信息管理用例表如表2-12所示：

表2-12 系统权限管理用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 0008 | 用例名称 | 系统权限管理 |
| 用例描述 | 管理员通过网站后台可以查看到所有用户以及管理员的权限，并可以对相应人员的权限作出CURD操作 | | |
| 执行者 | 管理员 | | |
| 前置条件 | 进入后台管理界面，点击“系统权限管理”，进入系统权限管理界面； | | |
| 后置条件 | 点击“查找”按钮展示查找相应的权限信息；点击“删除”按钮删除相应的权限信息；点击“新增”按钮增加新的权限；点击“修改”按钮修改相应的权限信息； | | |
| 涉众利益 | 无 | | |
| 用例场景 | 管理员管理 | | |

续表2-12

|  |  |
| --- | --- |
| 基本事件流 | 基本流程（Base Flow）  1.管理员进入网站后台，输入账号密码  2.系统验证账号密码  3.管理员进入网站后台管理界面，点击“系统权限管理”进入系统权限管理界面  4.管理员输入查找的信息，点击“查找”按钮  5.系统展示相应的信息  6.管理员找到符合信息后，点击“修改”按钮修改相应的系统权限信息  7.点击“新增”按钮添加新的系统权限信息  8.点击“删除”按钮删除相应的系统权限信息 |
| 编写人 | 莫平平 |

## 2.3 非功能需求

### 2.3.1环境需求

服务器端环境如表2-13所示：

表 2-13 服务器端环境

|  |  |
| --- | --- |
| 系统所使用数据库 | MySQL8.0 |
| 操作系统 | Windows 7 及其以上 |
| Web服务器 | IIS 7.5 |
| 数据库 | MySQL8.0 |

客户端环境如表2-14所示：

表 2-14 客户端环境

|  |  |
| --- | --- |
| 操作系统 | Windows 7 或其以上版本操作系统 |
| 浏览器 | IE 6.0以及其以上版本（IE7.0、IE8.0） |
| 分辨率 | 最佳效果为1024×768像素 |

### 2.3.2 性能需求

（1）可维护性

维护方法：系统能真正做到使自身在数据损坏、丢失等情况下将备份数据倒回，实现数据的恢复。提供对系统自身的集中操作维护的功能。

（2）易操作性

界面设计：系统应提供美观实用，方便和直观的图形用户管理界面，充分考虑员工的习惯，简单易学，操作方便，所有菜单驱动的处理和各种快捷键，一键功能以确保多数达到。

（3）可扩展性

功能扩展：系统从系统结构、功能设计、管理对象等各方面的功能扩展来考虑，以满足用户今后系统扩充和扩大使用范围的要求。

软硬件升级：系统应采取的硬件和软件平台，软件和硬件的负载平衡机制的可扩展性充分考虑。系统要具有灵活的扩展能力，来适应关键的软件和硬件的开发及管理能力的上升。

（4）开放性

系统的数据格式应符合国家相关标准及行业标准，以此确保应用程序具有良好的互操作性和移植的可能。

### 2.3.3 安全需求

（1）容错性

当用户输入或误操作导致非法数据产生时，系统应具有一定的容错机制。在这种情况下，系统应给出友好的提示，提示用户重新输入或者进行自动的修复校正。

（2）系统的外在环境安全

安全系统要以充分考虑网络的高级别，多层次的安全性措施为前提，包括系统的备份，防火墙，用户权限和其他措施，以确保数据安全和机密信息不被泄露；考虑到系统的硬件和软件故障恢复等应急措施，以保障网络的安全和处理安全性。形成相对独立的安全机制，以防止来自系统外的未经授权的访问。

（3）系统内部安全

确保外部系统安全的同时，该系统还必须确保授权用户的合法使用。

（4）系统运行安全

从逻辑上讲，该系统应具有抵抗非法入侵的能力；在物理方面，该系统应确保没有潜在的单点故障，并提供资源的数据备份功能。系统支持定期自动和手动数据备份，能够在数据损坏或数据丢失的情况下找回数据，实现一定程度的数据恢复。

3 总体设计

## 3.1 系统设计的原则

在充分考虑到本系统的总体规划目标，以及系统的应用环境前提下，本系统的设计将严格遵循以下的设计原则：

1. 实用性原则。网上购票系统是ERP的典型代表，它的好坏直接影响到企业的经营效益。所以在进行该系统的设计时，我将系统的实用性要求放在了首位。本系统力求能够紧密切合企业和用户的实际业务需求，以求彻底解决传统购物方式的效率低的通病。
2. 先进性原则。在强调实用性的同时，我们还要考虑系统的先进性。今时今日，信息化的发展速度令世人惊叹，每一次技术上的更新与进步都会推动信息化建设的跳跃式发展。所以，为了与时俱进，有一个较高的起点，在本系统的设计过程中我采用目前较为先进和成熟的技术和理念，使系统在性价比上拥有一定的优势。
3. 规范性原则。系统中的所有设计都遵循现有的国家标准、行业标准，符合软件工程中的设计原则与规范。所有文档的书写都遵照国标《软件需求说明书（GB856T—88）》要求。
4. 安全性原则。系统提供安全机制以防止非法授权操作，为各子系统提供严格灵活的用户和权限划分，对于数据库管理，建立起完备的数据备份恢复机制。对关键的信息的操作提供日志记录。
5. 易于集成的原则。出于对信息化建设的全盘考虑和整体规划，本系统要能够与其他业务系统进行集成。本系统为其他系统预留有应用接口和数据传输规范，方便其他系统与我系统间的信息交互。本系统在数据库设计上同样遵循此原则，在数据访问层以WebServices的方式提供数据访问服务。
6. 可扩展性原则。出于对后续升级工作的考虑，系统的所有功能模块的设计都预留了扩展接口。因为人们对网上购物系统的需求和应用目标会随着技术进步和商业模式的变换而发生转变，在设计初期就考虑到可能发生的变化，是十分必要的。

## 3.2 系统体系结构设计

目前的系统设计中，主要采用的体系结构是B/S结构和微信小程序体系结构。每个小程序的结构都是由两个主要部分构成：主体部分 + 各个页面。类似于许多框架，主体部分主要用于核心的配置，各个页面主要用于不同业务场景。

1.主体部分主要由3个文件构成

1）app.js：小程序逻辑，初始化APP

2）app.json ：小程序配置，比如导航、窗口、页面http请求跳转等

3）app.wxss：公共样式配置

主体配置完成之后，就是对应的业务开发了，也就是开发者最常操作的页面。小程序页面设计基本上也是遵循 MVC 结构进行构建。

2.页面由4个文件构成

1）js：页面逻辑，相当于控制层（C）；也包括部分的数据（M）

2）wxml：页面结构展示，相当于视图层（V）

3）wxss：页面样式表，纯前端，用于辅助wxml展示

4）json：页面配置，配置一些页面展示的数据，充当部分的模型（M）

如图3-1所示：

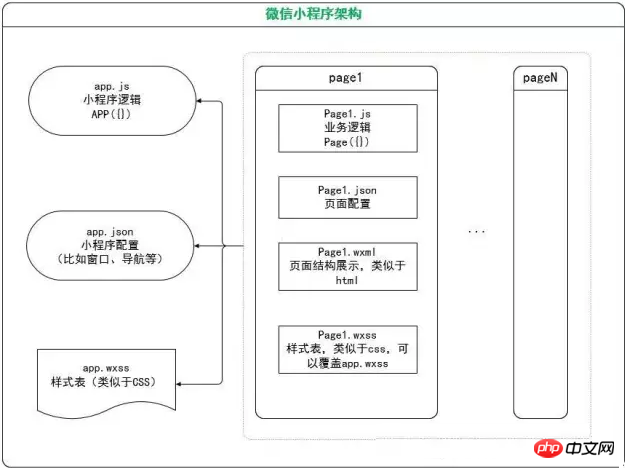
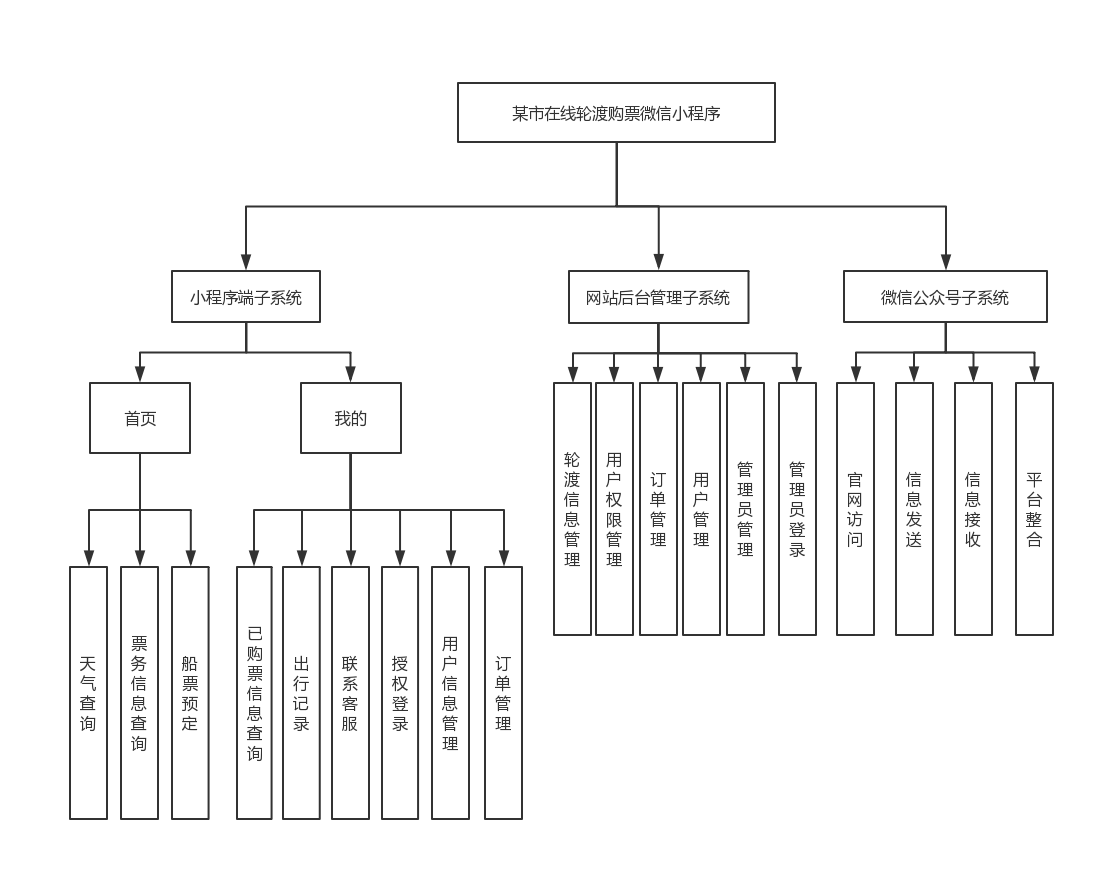


图3-1 微信小程序开发小程序架构篇图解

## 3.3 系统功能结构设计

系统功能架构设计定义软件系统各主要部件及成分之间的关系，本系统具体关系如下图3-2所示，本系统主要分为小程序端子系统和网站后台管理系统两大模块，其中小程序端子系统分为天气查询、票务信息管理、订单管理、用户信息管理、已购票信息管理、授权登录、出行记录以及联系客服八大模块，其中天气查询模块只负责天气信息的查询与展示；信息管理包含票务信息的展示和查询功能；订单管理包含订单的CURD功能以及支付订单的功能；用户信息管理包含用户信息的修改和查看操作；已购票信息管理和出行记录都只包含订单的查找功能；授权登录模块主要是从微信获取权限登录小程序；出行记录模块包含对订单的查询功能以及展示功能；联系客服模块包含与后台管理员进行会话的功能。网站后台管理子系统分为轮船信息管理、订单管理、用户管理、管理员管理、管理员登录以及权限管理六大模块，其中轮船信息管理包含轮船信息的CURD功能；订单管理包含订单的CURD功能；用户信息管理包含CURD功能；管理员管理包含管理员的添加和删除功能；管理员登录包含管理员输入账号密码登录网站后台的功能；权限管理包含权限的赋予和删除功能。

图3-2系统总包图

# 4 数据库设计

## 4.1逻辑设计

### 4.1.1实体关系设计

该小程序实体关系图如下图4-1所示:

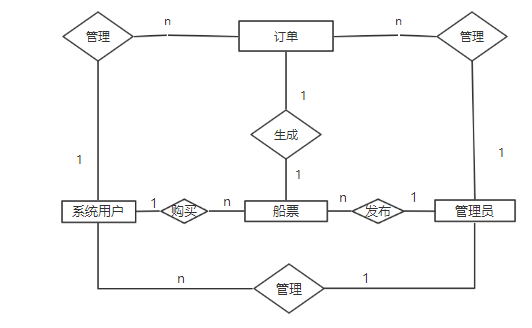


图4-1 实体关系图

### 4.1.2实体属性设计

（1）系统用户实体属性

如下图4-2所示为系统用户实体属性图：

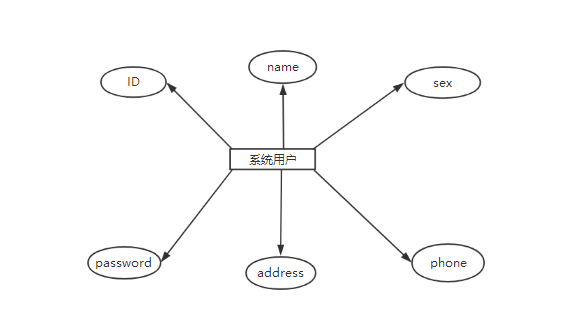


图4-2 系统用户实体属性图

（2）管理员实体属性

如下图4-3所示为管理员实体属性图：

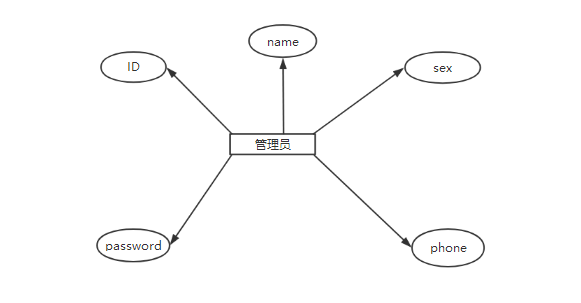


图4-3 管理员实体属性图

（3）船票实体属性

如下图4-4所示为船票实体属性图：

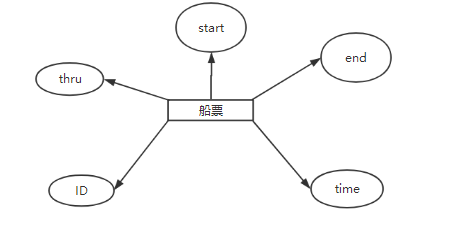


图6-4 船票实体属性图

（4）订单实体属性

如下图6-5所示为订单实体属性图：

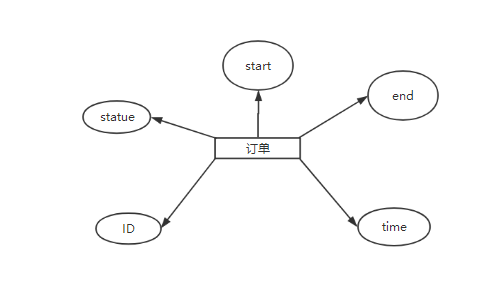


图4-5 订单实体属性图

### 4.1.3逻辑表格设计

（1）系统用户表

系统用户表如表4-1所示：

表4-1 系统用户表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 字段长度 | 是否主键 | 说明 |
| 1 | User\_id | 字符型 | 20 | 是 | 系统用户编号 |
| 2 | User\_name | 字符型 | 20 | 否 | 系统用户名称 |
| 3 | Sex | 字符型 | 10 | 否 | 系统用户性别 |

续表4-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | Phone | 字符型 | 20 | 否 | 系统用户电话 |
| 5 | Password | 字符型 | 20 | 否 | 系统用户密码 |
| 6 | Address | 字符型 | 20 | 否 | 系统用户住址 |
| 7 | Order\_id | 字符型 | 20 | 否 | 订单编号 |

（2）管理员表

管理员表如表4-2所示：

表4-2 管理员表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 字段长度 | 是否主键 | 说明 |
| 1 | Admin\_id | 字符型 | 20 | 是 | 管理员编号 |
| 2 | Admin\_name | 字符型 | 20 | 否 | 管理员名称 |
| 3 | User\_id | 字符型 | 20 | 否 | 系统用户编号 |
| 4 | Sex | 字符型 | 10 | 否 | 管理员性别 |
| 5 | Phone | 字符型 | 20 | 否 | 管理员电话 |
| 6 | Password | 字符型 | 20 | 否 | 管理员密码 |

（3）船票表

船票表如表4-3所示：

表4-3 船票表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 字段长度 | 是否主键 | 说明 |
| 1 | Tick\_id | 字符型 | 20 | 是 | 船票编号 |
| 2 | Start | 字符型 | 20 | 否 | 船票出发点 |
| 3 | End | 字符型 | 10 | 否 | 船票抵达点 |
| 4 | Thru | 字符型 | 50 | 否 | 船票途经点 |
| 5 | Time | 字符型 | 20 | 否 | 轮船出发时间 |

（4）订单表

订单表如表4-4所示：

表4-4 订单表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 字段长度 | 是否主键 | 说明 |
| 1 | Order\_id | 字符型 | 20 | 是 | 订单编号 |
| 2 | Start | 字符型 | 20 | 否 | 船票出发点 |
| 3 | End | 字符型 | 10 | 否 | 船票抵达点 |
| 4 | Statue | 字符型 | 50 | 否 | 订单状态 |
| 5 | Time | 字符型 | 20 | 否 | 订单生成时间 |
| 6 | Tick\_id | 字符型 | 20 | 是 | 船票编号 |

## 4.2物理结构设计

### 4.2.1数据库存取方式

本系统使用MySql数据库，版本为5.7.21，采用索引存取的方式对数据进行存取，在每个表中设置主键字段和经常需要查找的字段的B+树索引，提高数据库的查找效率。

1、系统用户表（User）

系统用户表如表4-5所示：

表4-5 系统用户表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 字段长度 | 是否主键 | 说明 |
| 1 | User\_id | 字符型 | 20 | 是 | 系统用户编号 |
| 2 | User\_name | 字符型 | 20 | 否 | 系统用户名称 |
| 3 | Sex | 字符型 | 10 | 否 | 系统用户性别 |
| 4 | Phone | 字符型 | 20 | 否 | 系统用户电话 |
| 5 | Password | 字符型 | 20 | 否 | 系统用户密码 |
| 6 | Address | 字符型 | 20 | 否 | 系统用户住址 |
| 7 | Order\_id | 字符型 | 20 | 否 | 订单编号 |

2、管理员表（Admin）

管理员表如表4-6所示：

表4-6 管理员表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 字段长度 | 是否主键 | 说明 |
| 1 | Admin\_id | 字符型 | 20 | 是 | 管理员编号 |
| 2 | Admin\_name | 字符型 | 20 | 否 | 管理员名称 |
| 3 | User\_id | 字符型 | 20 | 否 | 系统用户编号 |
| 4 | Sex | 字符型 | 10 | 否 | 管理员性别 |
| 5 | Phone | 字符型 | 20 | 否 | 管理员电话 |
| 6 | Password | 字符型 | 20 | 否 | 管理员密码 |

1. 船票表（Ticket）

船票表如表4-7所示：

表4-7 船票表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 字段长度 | 是否主键 | 说明 |
| 1 | Tick\_id | 字符型 | 20 | 是 | 船票编号 |
| 2 | Start | 字符型 | 20 | 否 | 船票出发点 |
| 3 | End | 字符型 | 10 | 否 | 船票抵达点 |
| 4 | Thru | 字符型 | 50 | 否 | 船票途经点 |
| 5 | Time | 字符型 | 20 | 否 | 轮船出发时间 |

4、订单表（Order）

订单表如表4-8所示：

表4-8 订单表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 字段长度 | 是否主键 | 说明 |
| 1 | Order\_id | 字符型 | 20 | 是 | 订单编号 |
| 2 | Start | 字符型 | 20 | 否 | 船票出发点 |
| 3 | End | 字符型 | 10 | 否 | 船票抵达点 |
| 4 | Statue | 字符型 | 50 | 否 | 订单状态 |
| 5 | Time | 字符型 | 20 | 否 | 订单生成时间 |
| 6 | Tick\_id | 字符型 | 20 | 是 | 船票编号 |

### 4.2.2存储过程设计

针对数据库系统中数据的结构，本系统采用mysql作为存储数据库，并且采用了索引顺序存储、数据分离和缓存优化技术。系统的数据库表存储采取顺序存储并对行字段建立对应索引，对经常查询的表进行数据分离，加快系统的查询速度，并且对于系统中的一些不经常更新的查询结果，将其存放在缓存数据库中，节省了重复查询的时间开销。

5 界面设计

## 5.1 工作流图

如下图5-1所示为工作流程图：

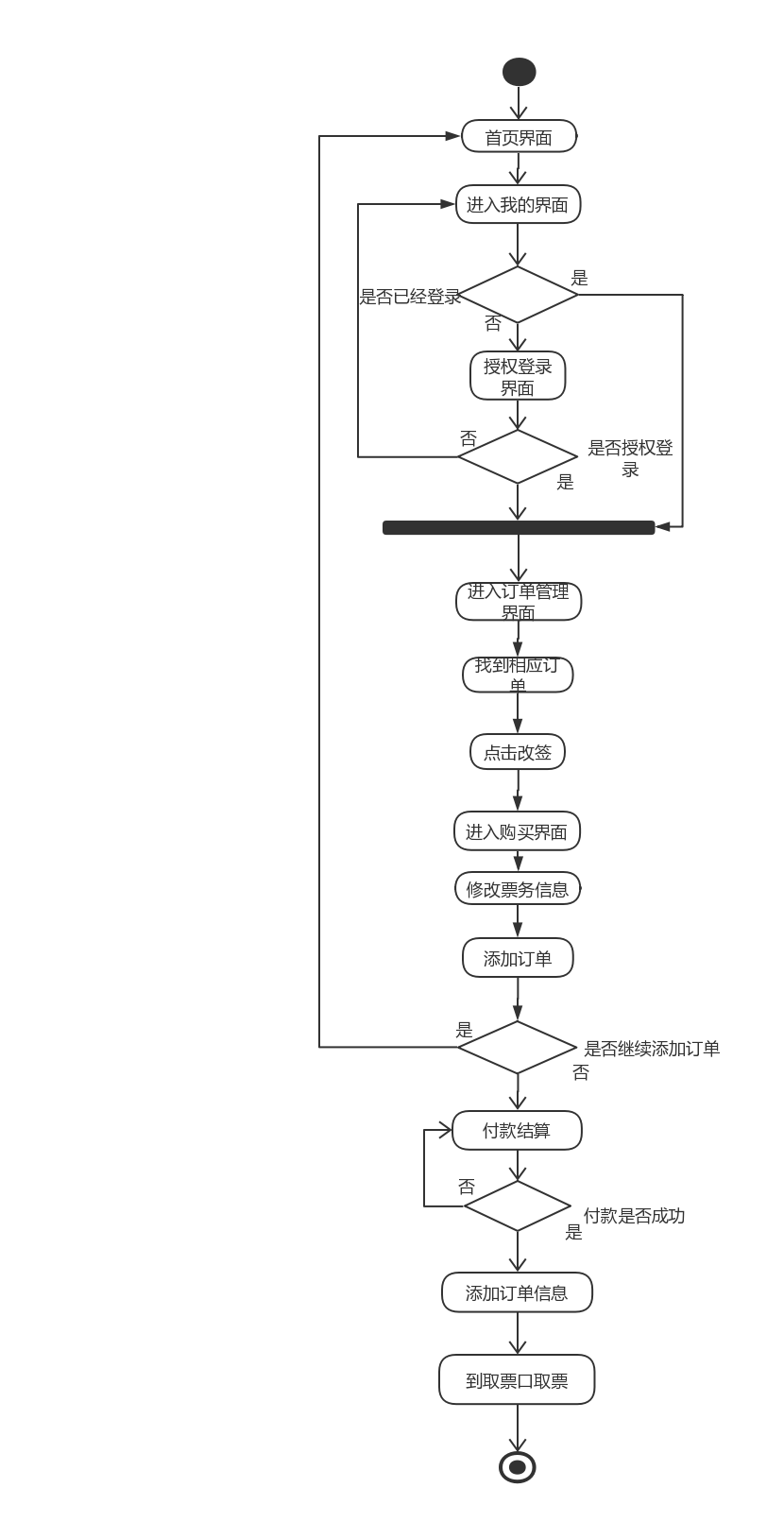


图5-1 工作流程图

## 5.2 界面设计成果

### 5.2.1 主界面

如下图5-2所示为微信小程序主界面图：



图5-2 微信小程序主界面图

### **5.2.2 子界面**

（1）添加订单子界面图

如下图5-3所示为添加订单子界面图：



图5-3 添加订单子界面图

（2）支付订单子界面图

如下图5-4所示为支付订单子界面图：



图5-4 支付订单子界面图

（3）订单详情子界面图

如下图5-5所示为订单详情子界面图：



图5-5 订单详情子界面图

（4）酒店预订子界面图

如下图5-6所示为酒店预订子界面图：



图5-6 酒店预订子界面图

# **6 详细设计**

## 6.1 系统主要功能模块介绍

本系统主要分为小程序端子系统和网站后台管理系统两大模块，其中小程序端子系统分为天气查询、票务信息管理、订单管理、用户信息管理、已购票信息管理、授权登录、出行记录以及联系客服八大模块，其中天气查询模块只负责天气信息的查询与展示；信息管理包含票务信息的展示和查询功能；订单管理包含订单的CURD功能以及支付订单的功能；用户信息管理包含用户信息的修改和查看操作；已购票信息管理和出行记录都只包含订单的查找功能；授权登录模块主要是从微信获取权限登录小程序；出行记录模块包含对订单的查询功能以及展示功能；联系客服模块包含与后台管理员进行会话的功能。网站后台管理子系统分为轮船信息管理、订单管理、用户管理、管理员管理、管理员登录以及权限管理六大模块，其中轮船信息管理包含轮船信息的CURD功能；订单管理包含订单的CURD功能；用户信息管理包含CURD功能；管理员管理包含管理员的添加和删除功能；管理员登录包含管理员输入账号密码登录网站后台的功能；权限管理包含权限的赋予和删除功能。

## 6.2小程序端功能模块设计

### 6.2.1功能结构设计

如下图6-1所示为小程序端子系统功能结构示意图，从图中我们看出小程序端子系统分为天气查询、票务信息管理、订单管理、用户信息管理、已购票信息管理、授权登录、出行记录以及联系客服八大模块。其中票务信息管理包含票务信息的展示和查询功能；订单管理包含订单的CURD功能以及支付订单的功能；用户信息管理包含用户信息的修改和查看操作；已购票信息管理和出行记录都只包含订单的查找功能。

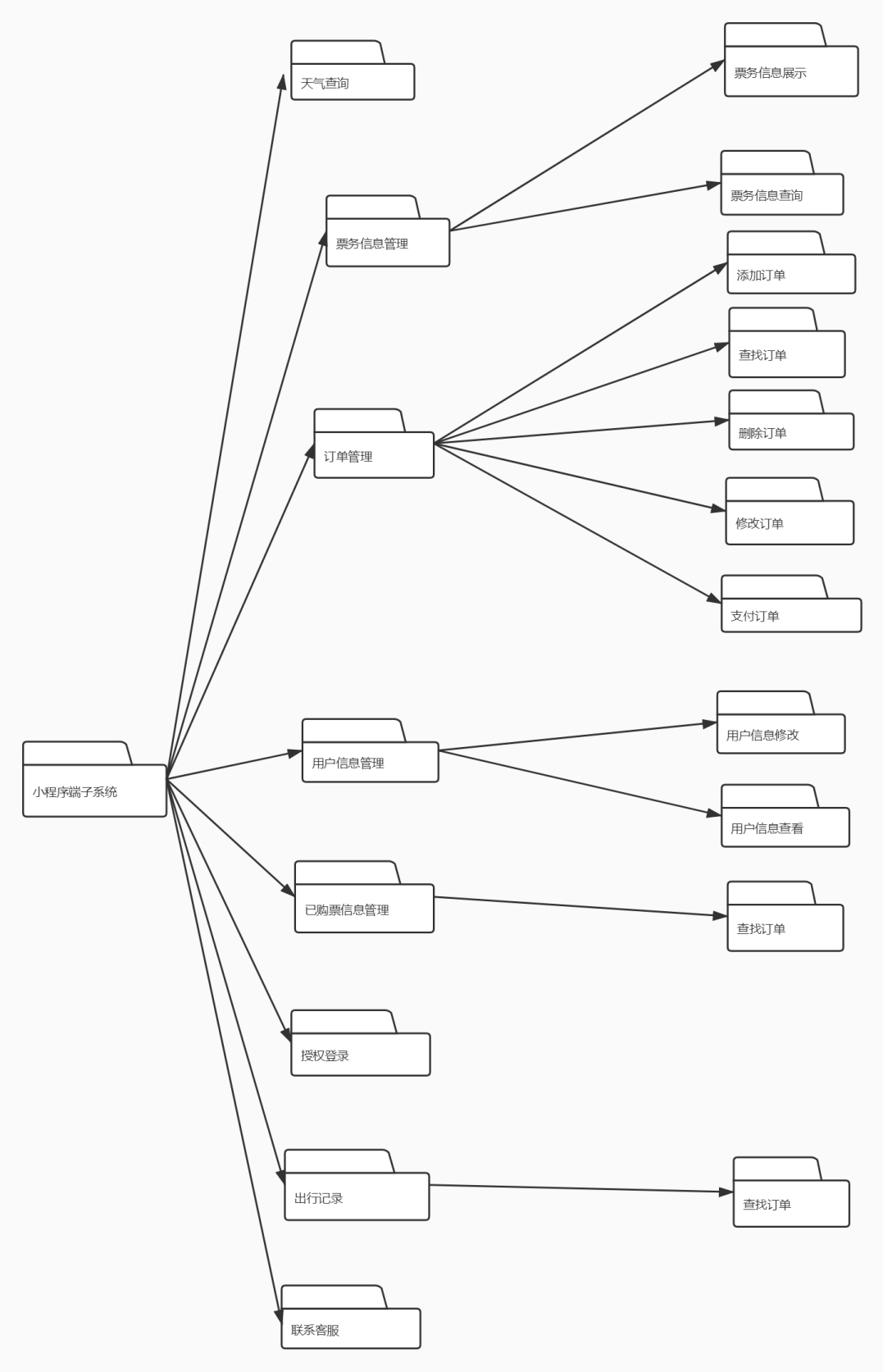


图6-1 小程序端子系统功能结构设计图

### 6.2.2 功能类设计

（1）票务信息管理类图

票务信息管理类图如下图6-2所示：

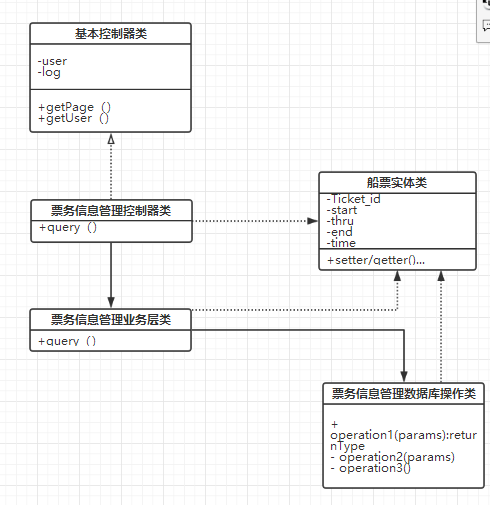


图6-2 票务信息管理类图

表6-1 票务信息管理-基本控制器类表

|  |  |  |
| --- | --- | --- |
| **基本控制器类** | | |
| 类的描述：负责管理所有与控制器的公用功能和公用数据，以及获取用户信息 | | |
| 属性描述 | log | 系统日志 |
| user | 已登录的用户信息 |
| 操作描述 | getPage() | 通过springMVC获取路径参数，组装分页对象 |
| getUser() | 从shiro中获取当前登录用户的对象 |

表6-2 票务信息管理-控制器类表

|  |  |  |
| --- | --- | --- |
| **票务信息管理控制器类** | | |
| 类的描述：负责票务管理相关请求的处理和转发 | | |
| 操作描述 | query() | 查询票务信息 |

表6-3 票务信息管理-服务层类表

|  |  |  |
| --- | --- | --- |
| **票务信息管理服务层类** | | |
| 类的描述：负责票务管理相关操作的服务层方法封装 | | |
| 操作描述 | query() | 数据的封装 |

表6-4 票务信息管理-数据库操作类表

|  |  |  |
| --- | --- | --- |
| **票务信息管理数据库操作类** | | |
| 类的描述：负责票务管理相关操作的数据库操作层方法封装 | | |
| 操作描述 | query() | 返回查询结果 |

表6-5 船票实体类

|  |  |  |
| --- | --- | --- |
| **船票实体类** | | |
| 类的描述：船票的实体对象类 | | |
| 属性描述 | Ticket\_id | 船票编号 |
| Start | 出发点 |
| Thru | 途经点 |
| End | 目的地 |
| Time | 购票时间 |
| 操作描述 | setter/getter() | 对船票属性的set和get方法 |

（2）订单管理类图

订单管理类图如下图6-3所示：

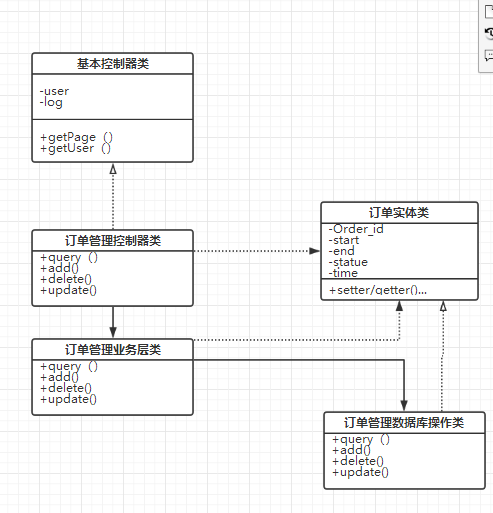


图6-3 订单管理类图

表6-6 订单管理-基本控制器类表

|  |  |  |
| --- | --- | --- |
| **基本控制器类** | | |
| 类的描述：负责管理所有与控制器的公用功能和公用数据，以及获取用户信息 | | |
| 属性描述 | log | 系统日志 |
| user | 已登录的用户信息 |
| 操作描述 | getPage() | 通过springMVC获取路径参数，组装分页对象 |
| getUser() | 从shiro中获取当前登录用户的对象 |

表6-7 订单管理-控制器类表

|  |  |  |
| --- | --- | --- |
| **订单管理控制器类** | | |
| 类的描述：负责订单管理相关请求的处理和转发 | | |
| 操作描述 | query() | 查询订单 |
| add() | 添加订单 |
| delete() | 删除订单 |

续表6-7

|  |  |  |
| --- | --- | --- |
|  | update() | 更改订单 |

表6-8 订单管理-服务层类表

|  |  |  |
| --- | --- | --- |
| **订单管理服务层类** | | |
| 类的描述：负责订单管理相关操作的服务层方法封装 | | |
| 操作描述 | query() | 数据的封装 |
| add() | 数据的封装 |
| delete() | 数据的封装 |
| update() | 数据的封装 |

表6-9 订单管理-数据库操作类表

|  |  |  |
| --- | --- | --- |
| **订单管理数据库操作类** | | |
| 类的描述：负责订单管理相关操作的数据库操作层方法封装 | | |
| 操作描述 | query() | 返回查询结果 |
| add() | 返回添加结果 |
| delete() | 返会删除结果 |
| update() | 返回修改结果 |

表6-10 订单实体类

|  |  |  |
| --- | --- | --- |
| **订单实体类** | | |
| 类的描述：订单的实体对象类 | | |
| 属性描述 | Order\_id | 订单编号 |
| Start | 出发点 |
| End | 目的地 |
| Time | 订单生成时间 |
| Statue | 订单状态 |
| 操作描述 | setter/getter() | 对船票属性的set和get方法 |

1. 已购票信息管理类图

已购票信息管理类图如下图6-4所示：

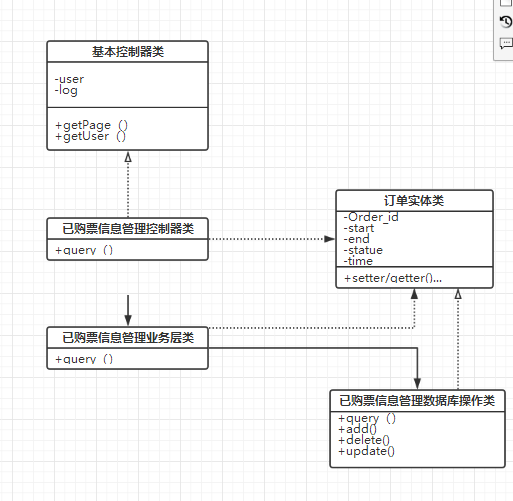


图6-4 已购票信息管理类图

表6-11 已购票信息管理-基本控制器类表

|  |  |  |
| --- | --- | --- |
| **基本控制器类** | | |
| 类的描述：负责管理所有与控制器的公用功能和公用数据，以及获取用户信息 | | |
| 属性描述 | log | 系统日志 |
| user | 已登录的用户信息 |
| 操作描述 | getPage() | 通过springMVC获取路径参数，组装分页对象 |
| getUser() | 从shiro中获取当前登录用户的对象 |

表6-12 已购票信息管理-控制器类表

|  |  |  |
| --- | --- | --- |
| **已购票信息管理控制器类** | | |
| 类的描述：负责已购票管理相关请求的处理和转发 | | |
| 操作描述 | query() | 查询票务信息 |

表6-13 已购票信息管理-服务层类表

|  |  |  |
| --- | --- | --- |
| **已购票信息管理服务层类** | | |
| 类的描述：负责已购票管理相关操作的服务层方法封装 | | |
| 操作描述 | query() | 数据的封装 |

表6-14 已购票信息管理-数据库操作类表

|  |  |  |
| --- | --- | --- |
| **已购票信息管理数据库操作类** | | |
| 类的描述：负责已购票相关操作的数据库操作层方法封装 | | |
| 操作描述 | query() | 返回查询结果 |

表6-15 订单实体类

|  |  |  |
| --- | --- | --- |
| **订单实体类** | | |
| 类的描述：订单的实体对象类 | | |
| 属性描述 | Order\_id | 订单编号 |
| Start | 出发点 |
| End | 目的地 |
| Time | 订单生成时间 |
| Statue | 订单状态 |
| 操作描述 | setter/getter() | 对船票属性的set和get方法 |

（4）出行记录类图

出行记录类图如图6-5所示：

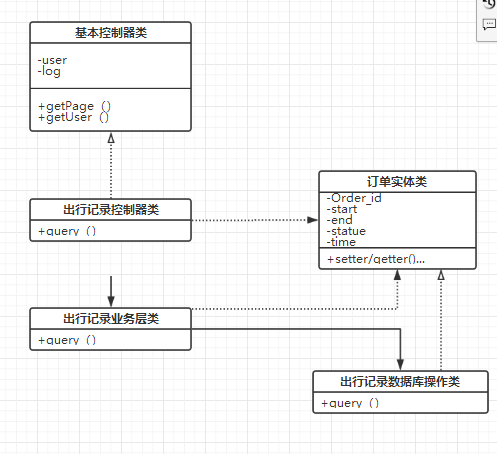


图6-5 出行记录类图

表6-16 出行记录-基本控制器类表

|  |  |  |
| --- | --- | --- |
| **基本控制器类** | | |
| 类的描述：负责管理所有与控制器的公用功能和公用数据，以及获取用户信息 | | |
| 属性描述 | log | 系统日志 |
| user | 已登录的用户信息 |
| 操作描述 | getPage() | 通过springMVC获取路径参数，组装分页对象 |
| getUser() | 从shiro中获取当前登录用户的对象 |

表6-17 出行记录-控制器类表

|  |  |  |
| --- | --- | --- |
| **出行记录控制器类** | | |
| 类的描述：负责出行记录相关请求的处理和转发 | | |
| 操作描述 | query() | 查询票务信息 |

表6-18 出行记录-服务层类表

|  |  |  |
| --- | --- | --- |
| **出行记录服务层类** | | |
| 类的描述：负责出行记录相关操作的服务层方法封装 | | |
| 操作描述 | query() | 数据的封装 |

表6-19 出行记录-数据库操作类表

|  |  |  |
| --- | --- | --- |
| **出行记录数据库操作类** | | |
| 类的描述：负责出行记录相关操作的数据库操作层方法封装 | | |
| 操作描述 | query() | 返回查询结果 |

表6-20 订单实体类

|  |  |  |
| --- | --- | --- |
| **订单实体类** | | |
| 类的描述：船票的实体对象类 | | |
| 属性描述 | Order\_id | 订单编号 |
| Start | 出发点 |
| End | 目的地 |
| Time | 订单生成时间 |
| Statue | 订单状态 |
| 操作描述 | setter/getter() | 对船票属性的set和get方法 |

（5）用户信息管理类图

用户信息管理类图如下图6-6所示：

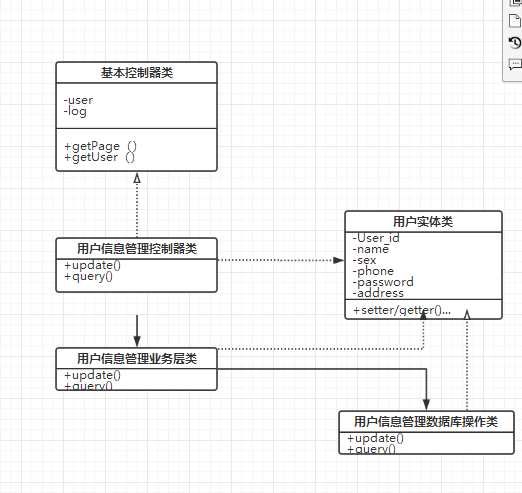


图6-6 用户信息管理类图

表6-21 用户信息管理-基本控制器类表

|  |  |  |
| --- | --- | --- |
| **基本控制器类** | | |
| 类的描述：负责管理所有与控制器的公用功能和公用数据，以及获取用户信息 | | |
| 属性描述 | log | 系统日志 |
| user | 已登录的用户信息 |
| 操作描述 | getPage() | 通过springMVC获取路径参数，组装分页对象 |
| getUser() | 从shiro中获取当前登录用户的对象 |

表6-22 用户信息管理-控制器类表

|  |  |  |
| --- | --- | --- |
| **用户信息管理控制器类** | | |
| 类的描述：负责用户信息管理相关请求的处理和转发 | | |
| 操作描述 | query() | 查询系统用户信息 |
| update() | 更改系统用户信息 |

表6-23 用户信息管理-服务层类表

|  |  |  |
| --- | --- | --- |
| **用户信息管理服务层类** | | |
| 类的描述：负责用户信息管理相关操作的服务层方法封装 | | |
| 操作描述 | query() | 数据的封装 |
| update() | 数据的封装 |

表6-24 用户信息管理-数据库操作类表

|  |  |  |
| --- | --- | --- |
| **用户信息管理数据库操作类** | | |
| 类的描述：负责用户信息管理相关操作的数据库操作层方法封装 | | |
| 操作描述 | query() | 返回查询结果 |
| update() | 返回修改结果 |

表6-25 用户实体类

|  |  |  |
| --- | --- | --- |
| **用户实体类** | | |
| 类的描述：系统用户的实体对象类 | | |
| 属性描述 | User\_id | 用户id |
| Name | 姓名 |
| Password | 密码 |
| Sex | 性别 |
| Phone | 电话 |
| Address | 地址 |
| 操作描述 | setter/getter() | 对用户属性的set和get方法 |

### 6.2.3 时序图设计

（1）票务信息管理时序图

票务信息管理时序图如下图6-7所示：

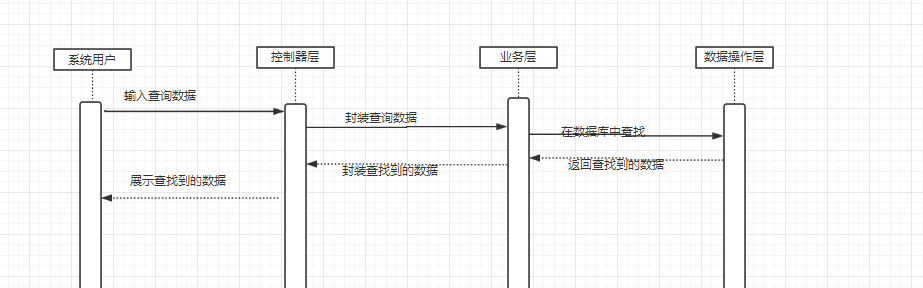


图6-7票务信息管理时序图

（2）订单管理时序图

订单管理时序图如下图6-8所示：

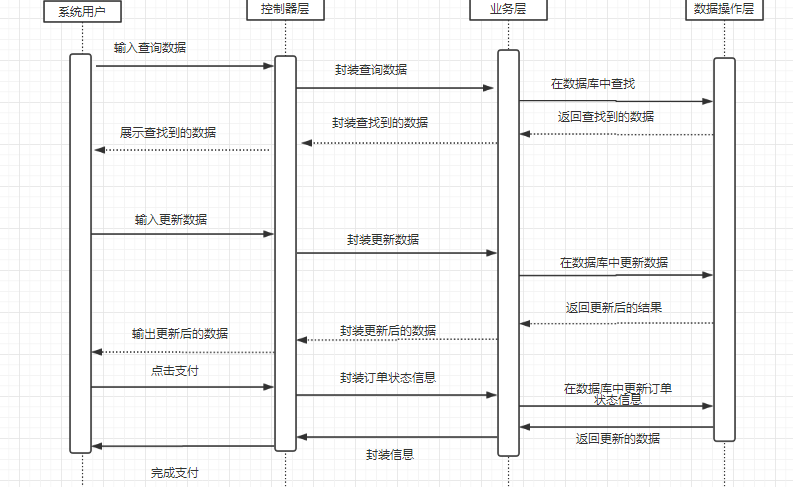


图6-8订单管理时序图

（3）已购票信息管理时序图

已购票信息管理时序图如图6-9所示：

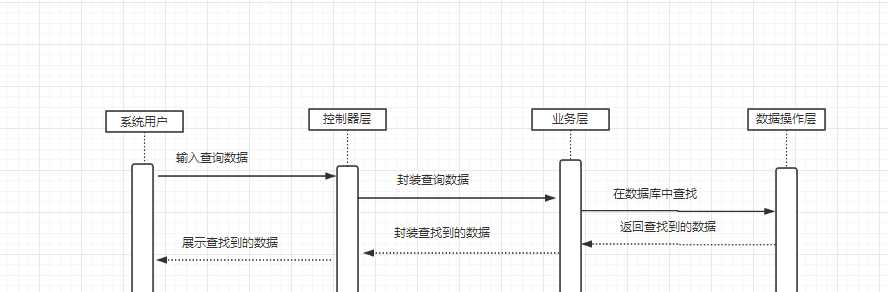


图6-9已购票信息管理时序图

1. 出行记录时序图

出行记录时序图如下图6-10所示：

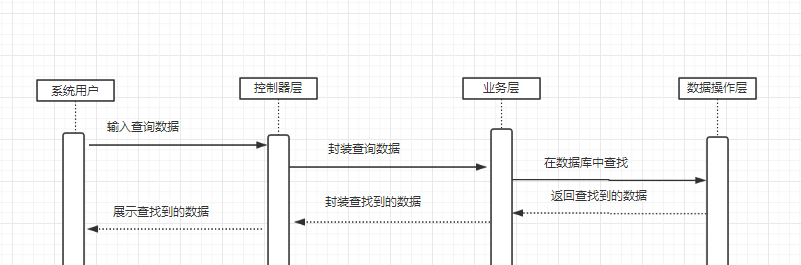


图6-10出行记录时序图

1. 用户信息管理时序图

用户信息管理时序图如下图6-11所示：

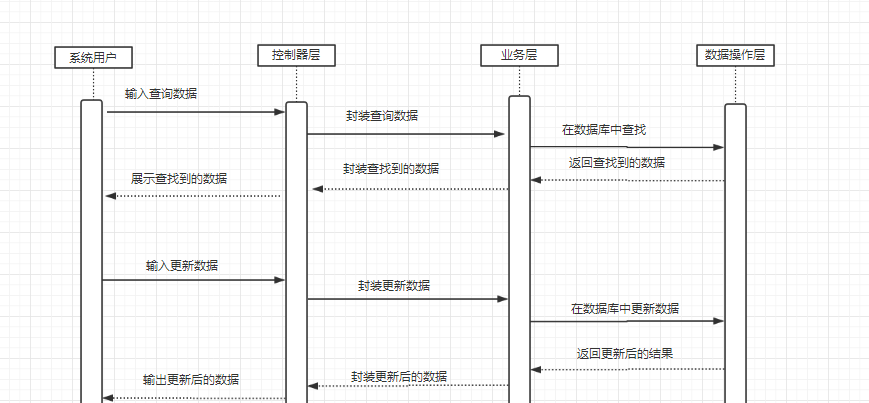


图6-11用户信息管理时序图

## 6.3网站后台管理功能模块设计

### 6.3.1功能结构设计

如下图6-12所示为网站后台管理子系统功能结构示意图，从图中我们看出网站后台管理子系统分为轮船信息管理、订单管理、用户管理、管理员管理、管理员登录以及权限管理六大模块。其中轮船信息管理包含轮船信息的CURD功能；订单管理包含订单的CURD功能；用户信息管理包含CURD功能；管理员管理包含管理员的添加和删除功能；权限管理包含权限的赋予和删除功能。

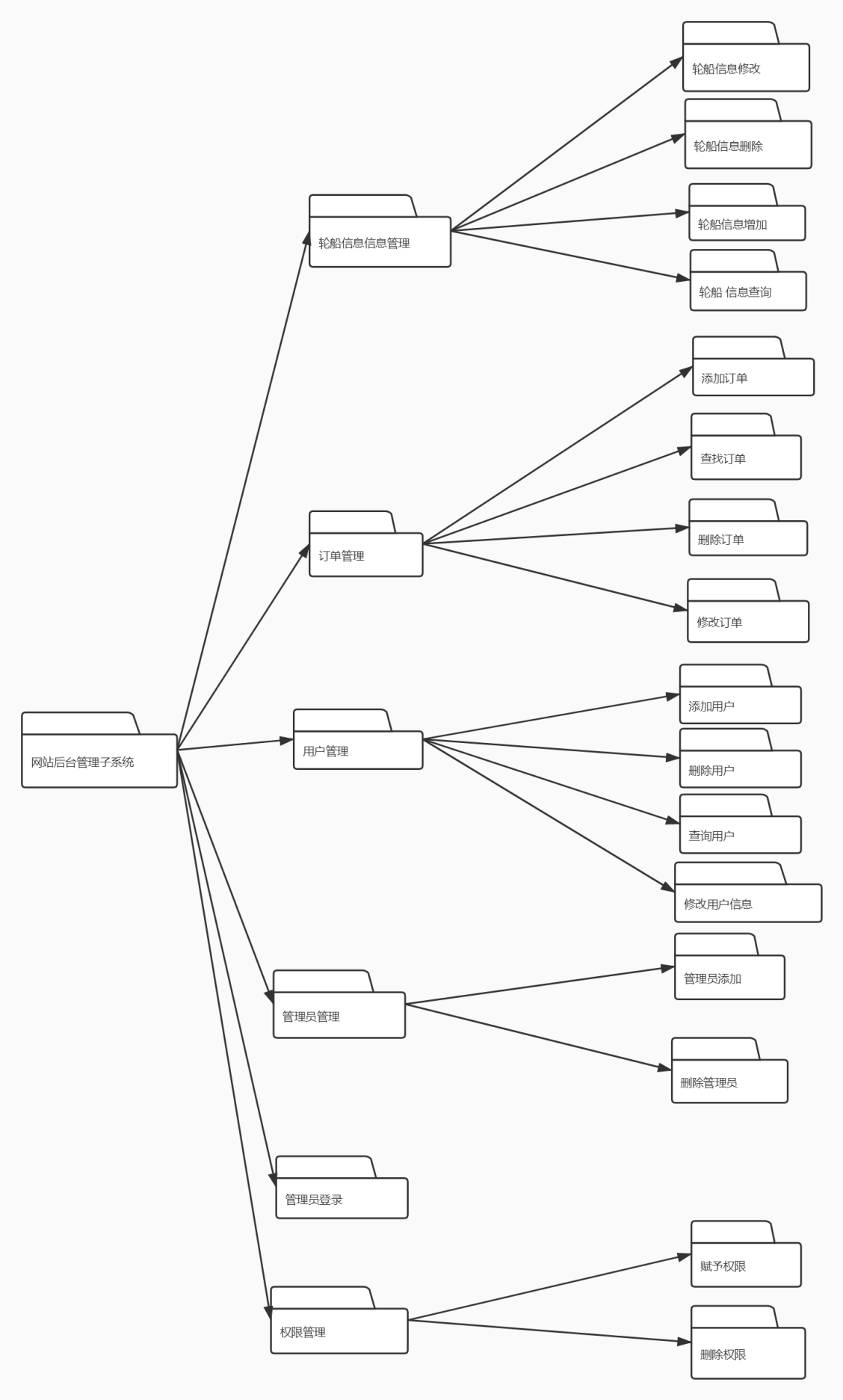


图6-12 网站后台管理子系统功能结构示意图

### 6.3.2功能类设计

（1）轮船信息管理类图

轮船信息管理类图如图6-13所示：

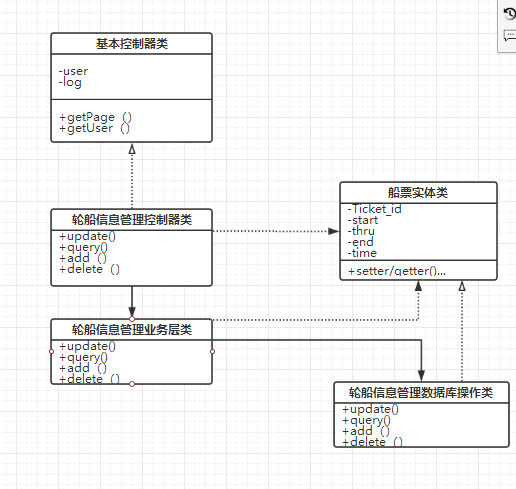


图6-13 轮船信息管理类图

表6-26 轮船信息管理-基本控制器类表

|  |  |  |
| --- | --- | --- |
| **基本控制器类** | | |
| 类的描述：负责管理所有与控制器的公用功能和公用数据，以及获取用户信息 | | |
| 属性描述 | log | 系统日志 |
| user | 已登录的用户信息 |
| 操作描述 | getPage() | 通过springMVC获取路径参数，组装分页对象 |
| getUser() | 从shiro中获取当前登录用户的对象 |

表6-27 轮船信息管理-控制器类表

|  |  |  |
| --- | --- | --- |
| **轮船信息管理控制器类** | | |
| 类的描述：负责轮船信息管理相关请求的处理和转发 | | |
| 操作描述 | query() | 查询轮船信息 |
| add() | 添加轮船信息 |

续表6-27

|  |  |  |
| --- | --- | --- |
|  | update() | 更改轮船信息 |
| delete() | 删除轮船信息 |

表6-28 轮船信息管理-服务层类表

|  |  |  |
| --- | --- | --- |
| **轮船信息管理服务层类** | | |
| 类的描述：负责轮船信息管理相关操作的服务层方法封装 | | |
| 操作描述 | query() | 数据的封装 |
| add() | 数据的封装 |
| update() | 数据的封装 |
| delete() | 数据的封装 |

表6-29 轮船信息管理-数据库操作类表

|  |  |  |
| --- | --- | --- |
| **轮船信息管理数据库操作类** | | |
| 类的描述：负责轮船信息管理相关操作的数据库操作层方法封装 | | |
| 操作描述 | query() | 返回查询结果 |
| add() | 返回添加结果 |
| update() | 返回更新结果 |
| delete() | 返回删除结果 |

表6-30 船票实体类

|  |  |  |
| --- | --- | --- |
| **船票实体类** | | |
| 类的描述：船票的实体对象类 | | |
| 属性描述 | Ticket\_id | 船票编号 |
| Start | 出发点 |
| Thru | 途经点 |
| End | 目的地 |
| Time | 购票时间 |
| 操作描述 | setter/getter() | 对船票属性的set和get方法 |

（2）订单管理类图

订单管理类图如下图6-14所示：

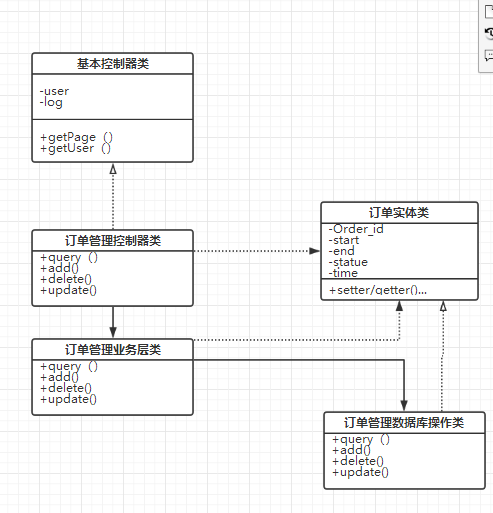


图6-14 订单管理类图

表6-31 订单管理-基本控制器类表

|  |  |  |
| --- | --- | --- |
| **基本控制器类** | | |
| 类的描述：负责管理所有与控制器的公用功能和公用数据，以及获取用户信息 | | |
| 属性描述 | log | 系统日志 |
| user | 已登录的用户信息 |
| 操作描述 | getPage() | 通过springMVC获取路径参数，组装分页对象 |
| getUser() | 从shiro中获取当前登录用户的对象 |

表6-32 订单管理-控制器类表

|  |  |  |
| --- | --- | --- |
| **订单管理控制器类** | | |
| 类的描述：负责订单管理相关请求的处理和转发 | | |
| 操作描述 | query() | 查询订单 |
| add() | 添加订单 |
| delete() | 删除订单 |

续表6-32

|  |  |  |
| --- | --- | --- |
|  | update() | 更改订单 |

表6-33 订单管理-服务层类表

|  |  |  |
| --- | --- | --- |
| **订单管理服务层类** | | |
| 类的描述：负责订单管理相关操作的服务层方法封装 | | |
| 操作描述 | query() | 数据的封装 |
| add() | 数据的封装 |
| delete() | 数据的封装 |
| update() | 数据的封装 |

表6-34 订单管理-数据库操作类表

|  |  |  |
| --- | --- | --- |
| **订单管理数据库操作类** | | |
| 类的描述：负责订单管理相关操作的数据库操作层方法封装 | | |
| 操作描述 | query() | 返回查询结果 |
| add() | 返回添加结果 |
| delete() | 返会删除结果 |
| update() | 返回修改结果 |

表6-35 订单实体类

|  |  |  |
| --- | --- | --- |
| **订单实体类** | | |
| 类的描述：订单的实体对象类 | | |
| 属性描述 | Order\_id | 订单编号 |
| Start | 出发点 |
| End | 目的地 |
| Time | 订单生成时间 |
| Statue | 订单状态 |
| 操作描述 | setter/getter() | 对船票属性的set和get方法 |

（3）用户管理类图

用户管理类图如下图6-15所示：

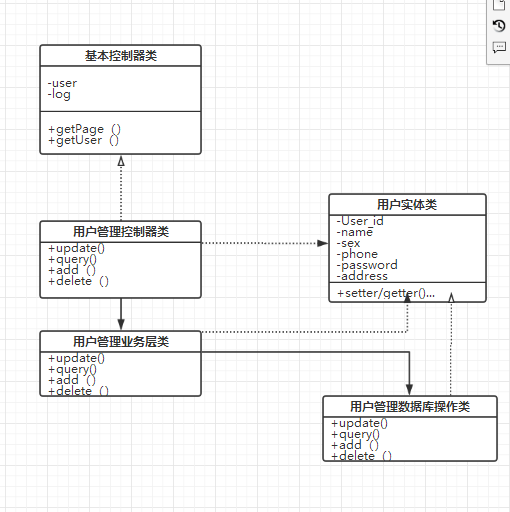


图6-15 用户管理类图

表6-36 用户管理-基本控制器类表

|  |  |  |
| --- | --- | --- |
| **基本控制器类** | | |
| 类的描述：负责管理所有与控制器的公用功能和公用数据，以及获取用户信息 | | |
| 属性描述 | log | 系统日志 |
| user | 已登录的用户信息 |
| 操作描述 | getPage() | 通过springMVC获取路径参数，组装分页对象 |
| getUser() | 从shiro中获取当前登录用户的对象 |

表6-37 用户管理-控制器类表

|  |  |  |
| --- | --- | --- |
| **用户信息管理控制器类** | | |
| 类的描述：负责用户管理相关请求的处理和转发 | | |
| 操作描述 | query() | 查询用户 |
| update() | 更改用户 |
| add() | 添加用户 |
| delete() | 删除用户 |

表6-38 用户管理-服务层类表

|  |  |  |
| --- | --- | --- |
| **用户管理服务层类** | | |
| 类的描述：负责用户管理相关操作的服务层方法封装 | | |
| 操作描述 | query() | 数据的封装 |
| update() | 数据的封装 |
| add() | 数据的封装 |
| delete() | 数据的封装 |

表6-39 用户管理-数据库操作类表

|  |  |  |
| --- | --- | --- |
| **用户管理数据库操作类** | | |
| 类的描述：负责用户信息管理相关操作的数据库操作层方法封装 | | |
| 操作描述 | query() | 返回查询结果 |
| update() | 返回修改结果 |
| add() | 返回添加结果 |
| delete() | 返回删除结果 |

表6-40 用户实体类表

|  |  |  |
| --- | --- | --- |
| **用户实体类** | | |
| 类的描述：系统用户的实体对象类 | | |
| 属性描述 | User\_id | 用户id |
| Name | 姓名 |
| Password | 密码 |
| Sex | 性别 |
| Phone | 电话 |
| Address | 地址 |
| 操作描述 | setter/getter() | 对用户属性的set和get方法 |

（4）管理员管理类图

管理员管理类图如下图6-16所示：

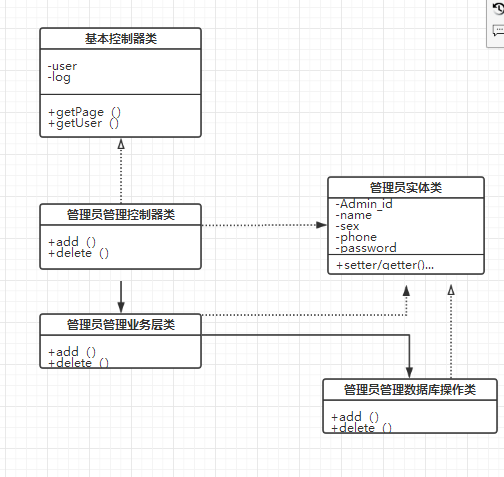


图6-16 管理员管理类图

表6-41 管理员管理-基本控制器类表

|  |  |  |
| --- | --- | --- |
| **基本控制器类** | | |
| 类的描述：负责管理所有与控制器的公用功能和公用数据，以及获取用户信息 | | |
| 属性描述 | log | 系统日志 |
| user | 已登录的用户信息 |
| 操作描述 | getPage() | 通过springMVC获取路径参数，组装分页对象 |
| getUser() | 从shiro中获取当前登录用户的对象 |

表6-42 管理员管理-控制器类表

|  |  |  |
| --- | --- | --- |
| **用户信息管理控制器类** | | |
| 类的描述：负责管理员管理相关请求的处理和转发 | | |
| 操作描述 | add() | 添加管理员 |
| delete() | 删除管理员 |

表6-43 管理员管理-服务层类表

|  |
| --- |
| **用户管理服务层类** |
| 类的描述：负责管理员管理相关操作的服务层方法封装 |

续表6-43

|  |  |  |
| --- | --- | --- |
| 操作描述 | add() | 数据的封装 |
| delete() | 数据的封装 |

表6-44 管理员管理-数据库操作类表

|  |  |  |
| --- | --- | --- |
| **管理员管理数据库操作类** | | |
| 类的描述：负责管理员管理相关操作的数据库操作层方法封装 | | |
| 操作描述 | add() | 返回添加结果 |
| delete() | 返回删除结果 |

表6-45 管理员实体类

|  |  |  |
| --- | --- | --- |
| **管理员实体类** | | |
| 类的描述：管理员的实体对象类 | | |
| 属性描述 | Admin\_id | 管理员id |
| Name | 姓名 |
| Password | 密码 |
| Sex | 性别 |
| Phone | 电话 |
| 操作描述 | setter/getter() | 对用户属性的set和get方法 |

### 6.3.3时序图设计

（1）轮船信息管理时序图

轮船信息管理时序图如下图6-17所示：

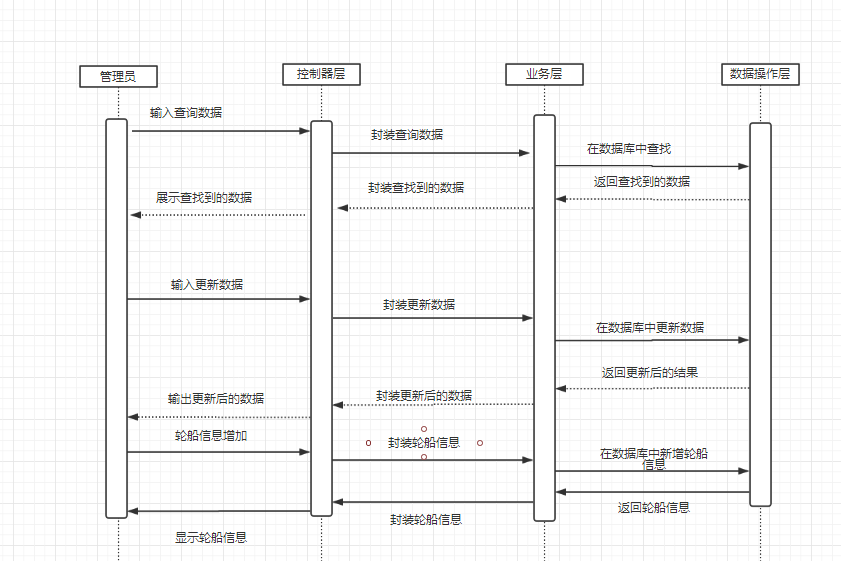


图6-17 轮船信息管理时序图

1. 订单管理时序图

订单管理时序图如下图6-18所示：

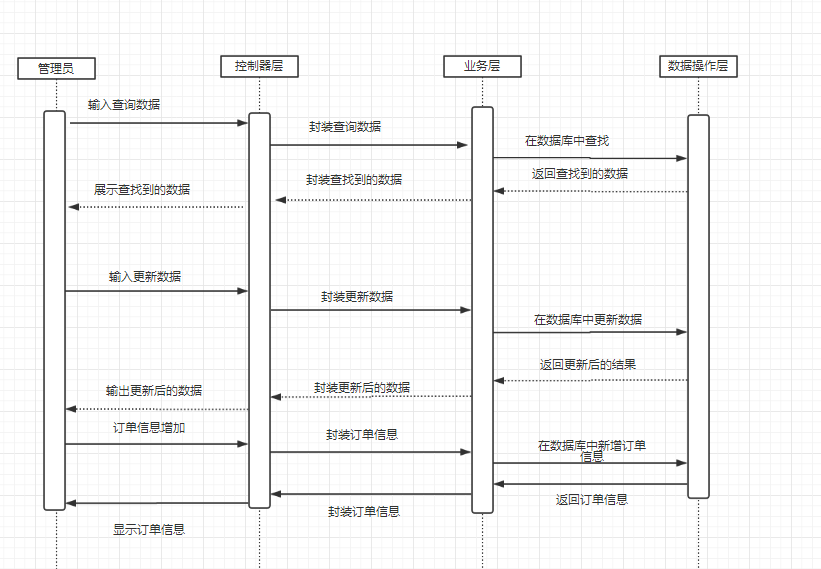


图6-18订单管理时序图

1. 用户管理时序图

用户管理时序图如下图6-19所示：

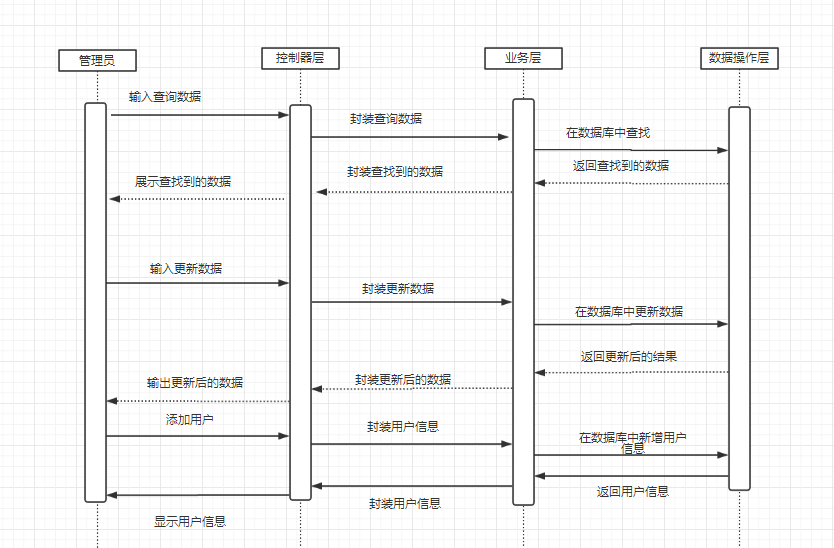


图6-19用户管理时序图

1. 管理员管理时序图

管理员管理时序图如下图6-20所示：

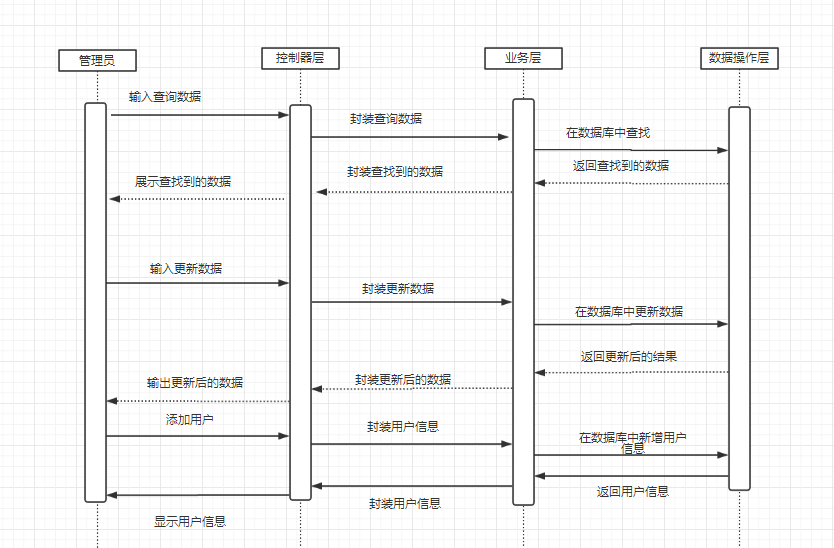


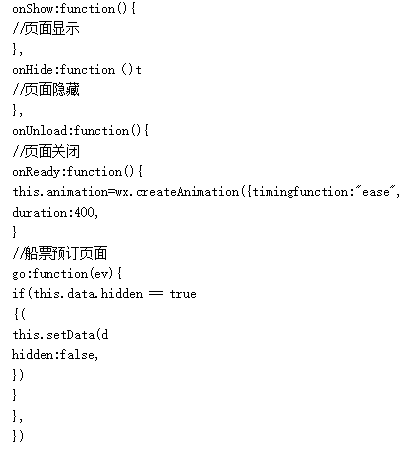
图6-20管理员管理时序图

7 编码

## 7.1 代码实现与核心算法

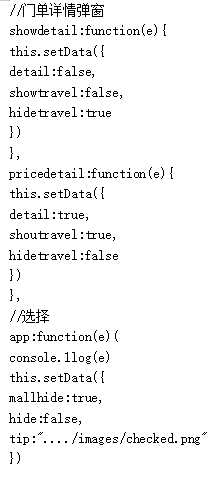
（1）船票预订代码

船票预订代码如下所示：



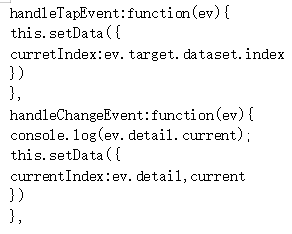
1. 订单详情代码

订单详情代码如下所示：



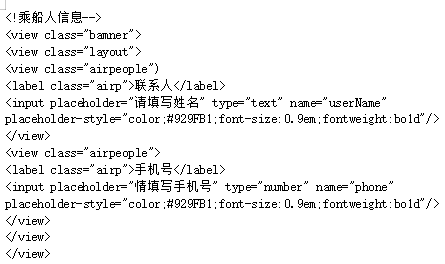
1. 天气查询搜索城市代码

天气查询搜索城市代码如下所示：



1. 添加乘客信息代码

添加乘客信息代码如下所示：



## 7.2 代码优化分析

### **7.2.1 代码优化的原则**

1. 首先确定优化的目标

在优化工作开始时，应该清楚地研究和确定系统要达到额效果，以及和优化相关的各类问题，而且在接下来的几个阶段中，要坚持以这些内容为目标。在项目开发的过程中，很多过程时无法完全按照计划中进行。即使一开始确定了优化的计划，但是很可能因为在项目实现的过程中进行改变，所以在对目标优化的研究中应该足够透彻和对项目的一些计划外的变化掌握的足够全面。

1. 进行关键模块的优化

项目的优化和项目的开发一样，需要很多的人力物力，项目的开发原则为在不影响功能的前提下节省开发的成本，而代码优化的成本也属于项目的开发成本。所以只进行关键模块的优化。对于一些使用频率低且使用压力小的模块在不存在明显问题的情况下无需进行过多的优化。

1. 优化应在项目开发的后期

过早的进行优化无疑在增加项目的开发成本。项目在不断的变化之中，而过早的进行优化有很大的可能通过项目开发的变化而变得用处极小或失效，这将白白浪费时间和经历。本系统认为，使项目开发的过程和优化的过程分开的进行。

1. 优化不能解决所有的问题

优化的一般作用是简化系统对硬件进行的多余的变更操作，甚至在一些不理想的情况下在一个方面问题的优化中，牺牲了另一个方面。如：进行了一个模块中速度的优化，可能要增加其对资源利用的数量。所以系统出现的所有问题不能都通过优化进行解决，可以在其他的方面进行考虑。

### **7.2.2 系统优化示例**

1. 数据库优化

<select id="findArticles" parameterType="Map" resultMap="ArticleResult">

select id,article\_title,article\_create\_date,article\_content,add\_name from ssm\_article

<where>

<if test="articleTitle!=null and articleTitle!='' ">

and article\_title like #{articleTitle}

</if>

</where>

<if test="start!=null and size!=null">

limit #{start},#{size}

</if>

</select>

关于Mapper文件中的sql语句，修改了select语句，原本的select语句中使用的为select\*查询所有字段，使其查询时的所有数据都会出现在结果集中，而这些数据在使用时显然不会全部使用。所以在这种情况下系统的工作量在无形的加大。而优化后的语句根据使用条件进行查询，对于无需使用的数据无需进行查询，这将系统性能进行了一定的改善。所有模块都进行了这样的优化。

8 测试

## 8.1 测试方案设计

### 8.1.1 测试策略

1. 本系统中测试的目标
2. 测试是为了发现运行中的程序错误而执行程序的过程。
3. 成功的测试是观测到了至今为止尚未发现的错误的测试。
4. 好的测试方案是发现迄今为止尚未发现的错误的测试方案。
5. 系统中的测试方法

白盒测试：它的前提是可以把程序看成装在一个透明的白盒子里，测试者完全知道程序的结构和处理算法。这种方法按照程序内部的逻辑测试程序，检测程序中的主要执行通路是否都能按预定要求正确工作。白盒测试又称为结构测试。

黑盒测试：正好相反，黑盒测试是把程序看作一个黑盒子，完全不考虑程序的内部结构和处理过程。就是说，黑盒测试是在程序接口进行的测试，它只检查程序功能是否按照需求规格说明书的规定正常使用，程序是否能适当地接收输入数据并产生正确的输出信息，程序运行过程中能否保持外部信息的完整性。黑盒测试又称为功能测试。

本系统同期不同时使用白盒测试和黑盒测试两种方法对软件各模块进行完整的功能测试，是系统中的bug尽量的减少。

### 8.1.2 测试进度安排

1. 模块测试

在开发完成一些模块功能后，对其进行模块测试，单个模块的进行各类测试，直到按照需求说明书进行运行。

1. 子系统测试

完成每一个子系统后进行子系统测试，使各个子系统完成需求说明书中的要求功能进行运行，排除逻辑问题和结果错误。

1. 系统测试

整个系统完成之后，最后进行系统的测试，运行整个系统使用各个功能进行数据操作，直到一段时间内均不存在问题。

### 8.1.3 测试资源

1. 硬件和软件资源
2. 2台计算机
3. 最新版的谷歌浏览器，火狐浏览器，搜狗浏览器和Microsoft Edge浏览器
4. IntelliJ IDEA 2019.3.1 x64 开发工具
5. eclipse开发工具
6. Navicat for MySQL数据库
7. 人力资源

项目开发人员两名，和其他一些同学老师。

### 8.1.4 关键测试点

1. Web页面

主要包括url地址、页面展示、动画切换显示、首页图标、按钮或链接跳转、后退功能、刷新功能的测试

1. 文本框
2. 正确输入，在限制条件内取几条用例进行正确值进行测试。一般最小、大边界值各一条，参数类型一条
3. 未输入，在未输入内容的时候，检查显示是否正常
4. 输入，在输入内容的时候，检查显示是否正常
5. 为空，其他内容输入正确，输入为空，点击登录/提交
6. 为空格，其他内容输入正确，输入为空格，点击登录/提交
7. 包含空格，其他内容输入正确，输入包含空格（首、中、尾包含空格分别测试），点击登录/提交
8. 为空白符其他内容输入正确，输入或复制使用搜狗输入法v1选d输出的空白字符，点击登录/提交
9. 特殊字符，其他内容输入正确，输入除\_以外的特殊字符，点击登录/提交
10. 小于最小值，其他内容输入正确，输入少于最小位，点击登录/提交

10)大于最大值，其他内容输入正确，输入大于最大位，点击登录/提交

11)中文汉字，其他内容输入正确，输入中文汉字，点击登录/提交

12)符合格式的错误内容，其他内容输入正确，输入错误的内容，点击登录/提交

13)粘贴复制修改，其他内容输入正确，在输入框中通过复制/粘贴/修改的方式填写内容

14)区分大小写，其他内容输入正确，区分大小写校验

1. 数据库数据处理
2. 新增字段是否对应表属性正确
3. 删除记录id等属性是否对应正确
4. 修改记录表属性是否对应正确

4）查找目标属性记录是否正确

## 8.2 测试用例构建

### 8.2.1 测试用例编写约定

1. 系统性：对于系统业务流程，应充分描述整个系统的业务需求、多个子系统的组成及其相互关系；对于模块业务流程，应描述内部功能、关键功能及其相互关系
2. 连惯性：对于系统业务流程，有必要解释子系统是如何连接在一起的。如果需要接口，各子系统之间是否有正确的接口。如果依赖页面链接，则页面链接是否正确。对于模块业务流程，需要说明同一级模块与上下级模块如何形成子系统，内部功能接口是否一致
3. 全面性：应尽量覆盖各种路径和业务点，考虑跨年跨月数据的准备和大量并发测试
4. 正确性：输入接口后的数据应与测试文件中记录的数据一致，预期的结果也应与测试数据的业务一致
5. 遵守正常的业务规则：测试数据应符合用户实际工作中的业务流程，并考虑到各种业务的变化和行业现行的法律法规
6. 人名、地名、电话号码等应具有模拟功能，符合通用命名惯例；不允许与名人、小说中的人物名等有相同情况。
7. 可操作性：在测试用例中，清楚地写出测试的操作步骤和不同操作步骤对应的测试结果。

### 8.2.2 测试用例设计

测试用例的设计主要以等价类划分法、边界值分析法、错误推测法、因果图方法、正交表分析法为主。测试了各个界面的需求。

1. 消息页面

消息管理页面测试需求如下表8-1所示：

表8-1消息页面需求和测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 页面需求 | | | | | |
| 输入 | 有效 | | | 无效 | |
| messageTitle | 长度小于255个字符，非空 | | | 长度多于255个字符 | |
| messageContent | 长度小于255个字符，非空 | | | 长度多于255个字符 | |
| 页面测试用例 | | | | | |
| 输入目标 | | 1 | 2 | | 3 |

续表8-1

|  |  |  |  |
| --- | --- | --- | --- |
| 输入messageTitle | 测试 | 计算机（computer）[俗称](https://baike.baidu.com/item/%E4%BF%97%E7%A7%B0/6538506)[电脑](https://baike.baidu.com/item/%E7%94%B5%E8%84%91/124859)，是现代一种用于高速计算的电子计算机器，可以进行数值计算，又可以进行逻辑计算，还具有存储记忆功能。是能够按照程序[运行](https://baike.baidu.com/item/%E8%BF%90%E8%A1%8C/5480697)，自动、高速处理海量数据的现代化智能电子设备。 | Message test |
| 输入messageContent | 123456 | Figure 4.1 does not look much like efficient automated testing and indeed it is not. It would be nice if the test execution tool were responsible for running the comparator, but unless we tell it to do so, and tell it how to do so, it is not. | 由[硬件](https://baike.baidu.com/item/%E7%A1%AC%E4%BB%B6" \t "https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA/_blank)系统和[软件](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6/12053)系统所[组成](https://baike.baidu.com/item/%E7%BB%84%E6%88%90/10881662)，没有安装任何软件的计算机称为裸机。可分为超级计算机、工业控制计算机、[网络计算机](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E8%AE%A1%E7%AE%97%E6%9C%BA/101471)、[个人计算机](https://baike.baidu.com/item/%E4%B8%AA%E4%BA%BA%E8%AE%A1%E7%AE%97%E6%9C%BA/3731770)、嵌入式计算机五类，较先进的计算机有[生物计算机](https://baike.baidu.com/item/%E7%94%9F%E7%89%A9%E8%AE%A1%E7%AE%97%E6%9C%BA/363245)、[光子计算机](https://baike.baidu.com/item/%E5%85%89%E5%AD%90%E8%AE%A1%E7%AE%97%E6%9C%BA/1880523)、[量子计算机](https://baike.baidu.com/item/%E9%87%8F%E5%AD%90%E8%AE%A1%E7%AE%97%E6%9C%BA/363335)等。 |
| 测试结果：  正确 | | | |

（2）用户管理页面

用户管理页面测试需求如下表8-2所示：

表8-2用户管理页面需求和测试用例

|  |  |  |
| --- | --- | --- |
| 页面需求 | | |
| 输入 | 有效 | 无效 |
| User\_id | 长度小于30的数字，非空 | 长度多于30个字符 |

续表8-2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| name | 长度小于255个字符，非空 | | | 长度多于255个字符 | |
| phone | 长度小于20的数字 | | | 长度多于20个字符 | |
| address | 长度小于255个字符 | | | 不属于范围的数字；字母；特殊字符；汉字 | |
| password | 长度小于20的数字 | | | 不属于范围的数字；字母；特殊字符；汉字 | |
| sex | “男”或“女” | | | 除有效中两种情况外的其他任何字符 | |
| 页面测试用例 | | | | | |
| 输入目标 | | 1 | 2 | | 3 |
| User\_id | | 964857 | 978546 | | 925487 |
| name | | 张三 | 李四 | | 王二 |
| phone | | 13570296537 | 13970262587 | | 13870262258 |
| address | | 浙江省宁波市 | 省南昌市 | | 湖南省长沙市 |
| password | | 123456789 | 456789321 | | 25048976 |
| sex | | 男 | 女 | | 男 |
| 测试结果：  正确 | | | | | |

以上为主要的测试用例列举，实际测试用例数量较多，无法完全列举。

### 8.2.3 测试用例维护

1. 删除过时的测试用例

由于需求的变化等原因，一个基线测试用例可能不再适合被测系统，那么这些测试用例就会过时，需要及时删除这些测试用例。在删除过程中，不能删除整行测试用例，需要删除的整行测试用例灰显，该行用例计数器清空；当需要删除能量模块的整个工作时，整页状态灰显，用例计数器将被清除

1. 修改的测试用例

随着软件项目的进展，测试需求可能会发生一些甚至大规模的变化。此时，我们将根据需求的变化来维护测试用例，修改不符合当前需求的内容，并在备注栏中进行说明

1. 删除多余的测试用例

如果有两个或多个测试用例要测试同一组输入和输入，则需要删除它们，只需要保留其中一个

1. 添加新测试用例

对于新增的功能，在评审过程和测试过程中发现缺少的测试用例，或者系统有缺陷但没有相应的测试用例，需要按照测试用例的设计标准进行添加。添加测试用例时，新添加的测试用例需要插入相应功能模块的底部，并在备注栏中进行说明。

9 总结与展望

## 9.1 设计工作总结

这次的，是对我大学四年以来学习的实践总结。它不仅让我感受到了我这大学四年的学习之中还存在的很多的不足之处，也让我清楚地知晓了自身的发展方向，给了我以后的就业方向。

通过自己、同学以及老师对毕设课题的应用，有效证明了系统设计基本达到了本课题的开发的初始目的；同时通过对该系统的不断熟悉，在系统的实际使用工程中也发现了部分问题，提出了很多的修改意见和建议，这对系统的稳定性、人性化设计提供了很大的帮助。本系统还应有更多的完善之处，但是由于时间和经验的限制，本人无法继续完善，希望以后有机会能够继续完善。

## 9.2 未来工作展望

本次毕设答辩之后，我就要踏出校门走向社会。在走向社会上找工作时，我希望能够找到和软件开发相关的工作，能为自己多积累经验。在找到工作后，作为一名新人加入，我觉得我首先要做到多向公司里的前辈们以及同事们多学习工作经验。我认为这是非常重要的，学习工作经验有利于我们在刚开始的工作中少走弯路，提高工作效率和工作质量，而且这还有利于我快速融入这个集体。

在本次论文的设计中，该小程序还有一些不足之处，如微信公众号由于技术和时间原因没有开发出来，后台管理系统的界面不够美观等等。在答辩之后，如果自己有能力，我一定会完善这个小程序。

# **谢 辞**

参考文献

1. 王学鸣,黄秋波.微信公众号的现状、类型及发展趋势研究[J].无线互联科技,2018.
2. 龚让声,李素桂,林敏.微信公众平台二次开发关键技术[J].电子技术与软件工程,2018.
3. 王俊欢,周如旗.响应式布局在微网站中的应用研究[J].现代计算机(专业版),2017.
4. 张德申,秦红亮.微信公众平台开发——订阅号功能开发研究[J].电子技术与软件工程,2013.
5. 尤鸿霞.基于HTML5的企业微网站建设[J].电脑知识与技术,2018.
6. 张海潘等编著.软件工程导论（第6版）[M].清华大学出版社,2018.
7. 严蔚敏，吴伟民著.数据结构[M].清华大学出版社,2017.
8. 萨师煊等编著.数据库系统概论（第5版）.高等教育出版社,2017.
9. 莎丽·劳伦斯·弗里格.软件工程(第4版)(修订版)[M].人民邮电出版社,2016.

[10] Efraim Turban.Electronic Commerce 2016 A Managerial Perspective[M].

Prentice Hall,2015.

[11] Brahm Canzer. E~business:Strategic Thinkingand Practice[M].Houghton

Mifflin College Div,2015.

[12] M.Papazoglou.E~Business:Organizational And Technical Foundations[M].

John Wiley&SonsInc,2016.

[13] Mehdi Khosrowpour.E~Commerce Security:Advice from Experts[M].Idea

GroupPub,2014.

[14] Paul Beynon~Davies.E~Business[M].Palgrave Macmillan,2014.

[15] Yang-Hua Chu,Joan Feigenbaum,Brian LaMacchia,PaulResnick and Martin

Strauss.Trust management for Web applications.Computer Networks and ISDN

Systems,2017.

[16] Beth T,Borcherding M,Klein B.Valuation of trust in open net-works.

Proceedings of the European Sympo-sium on Research in Security(Esorics).

Brighton:Springer-Verlag,2014.

[17] Jsang A.The right type of trust for distributed systems. Proc of the 2016 New Security Paradigms Workshop. Lake Arrowhead, CA: ACM Press,2016.

[18] Dean Povey.Developing Electronic Trust Policies Using a Risk Management

Model[C].Proceedings of the1999 Cqre Congress,2015.

[19] C.Ellison, B.Schneier.Ten Risks of PKI: What You are not Being Told about Public Key Infrastructure.Computer Security Journal,2017.

[20] Philo Juang, Hidekazu Okiand D. Rubenstein《MIS of Containers Basedon C/S

Structure》.Computer Engineering And Applications,2014.

[21] William G.Special Edition Using Oracle8/8i. American Indianapolis,2015.

[22] Jose G.Carrasco.“From EDI to ebXML–Rocking The Boat”.IAME,2017.

[23] Gooley,Toby B.“How to Choose an Ocean Carrier”.Traffic Management,

Denver,2019.

[24] Kumar K and Dissel HG.“Sustainable Collaboration:Managing Conflict

and Cooperation in Interorganizational Systems”.MIS Quarterly,2016.

[25] 孙君.我国网上支付存在的问题与建议. 江苏商论,2015.

[26] 陈娟.我国电子商务发展趋势[J]. 创新科技,2016.

[27] 陈力行.我国电子商务建设所面临的问题与对策[J]. 中国科技信息,2016.

[28] 杨坚争.电子商务安全与电子支付[M]. 北京. 机械工业出版社,2017.

[29] 胡锐,郭俊华.浅谈国内电子商务网上支付问题及对策[J]. 集团经济研究,2017.

[30] 肖林. 电子商务网上支付系统的研究[D], 贵州贵阳: 贵州大学,2016.

[31] 唐堃,王明哲,彭怀军,孙健,王运霞.铁路自动售票系统的研究与实现.铁路计算机应用,2018.[26]陈娟.我国电子商务发展趋势[J]. 创新科技,2016.

附录A 外文翻译—原文部分

原文出处：<https://www.doc88.com/p-9763109802100.html>

What is testware architecture?

Testware is the term we use to describe all of the artifacts required for testing, including documentation, scripts, data, and expected outcomes, and all the artifacts generated by testing, including actual outcomes, difference reports, and summary reports. Architecture is the arrangement of all of these artifacts; that is, where they are stored and used, how they are grouped and referenced, and how they are changed and maintained. Of course, some testware is specific to automated testing (such as the scripts) and some is specific to manual testing (such as the test procedures). However, much testware is common to both automated and manual testing (such as the test plans and test data). In this chapter we cover testware architecture specifically in relation to automated testing but the points we raise and the recommendations we make can be applied equally to manual testing. In fact we would recommend that the testware architecture be developed to cover all testware generically. It should make no difference whether the artifacts are related to manual or automated testing, unit or system testing, or functional or non-functional testing. We begin this chapter by highlighting four key issues that we believe have to be addressed if long-term success in test automation is to be achieved. The chapter ends with an approach to testware architecture that we have found works well. It may not suit everyone, as different environments and systems proffer different problems that require different solutions. However, even if the general approach is not suitable for you, we believe that there may be some ideas that could prove useful if adopted or at least adapted.

Platform and environment independence

This fourth testware architecture issue applies only where the same software has to be tested across different environments or hardware platforms. Ideally, all tests would be platform independent such that the same files, scripts, etc. can be used on every platform or in every environment. In practice there will inevitably be some differences. Some expected outcomes will likely be different. Due to the way a particular operating system implements a 'standard' package, different numerical results may result from the same calculation using the same input values. Different platforms may display things on the screen in a different order, in different colors or fonts, etc. Some of these differences are 'invisible' to the human tester, but can affect an automated test. This is one reason why the use of filters, described in Chapter 4, is useful. Perhaps test set-up instructions will change for different environments, such as for different database management systems. If a test database needs to be set up and populated for a given test, the commands to create the database and even the way in which data is entered may be quite different from one proprietary database to another. If the automated tests are required across different platforms, the testware architecture should provide a means of holding the necessary copies of platform-dependent information without requiring all the testware to be repeated.

Configuration management

There are two alternative ways of managing the configuration of the test-ware. The method that we favor is for the Testware Sets to be stored in the Testware Library as configuration items (that is, having a version number). The individual testware artifacts that make up the content of each type of set do not have their own version numbers. The effect of this is that whenever anything in the Testware Set is changed, a new version of the Testware Set is created containing the changed artifacts and the unchanged artifacts. A baseline is something that has a version number but comprises a number of configuration items, each version of a baseline being built from specific versions of the configuration items. In Figure 5.10, our baseline is a Test Suite, made up of the Testware Sets, which are the configuration items. An alternative way of managing the testware configuration would be for the individual testware artifacts to have their own version number. In this case, whenever anything in a Testware Set is changed, only those artifacts that change have new versions. The lowest level of the diagram would then become the configuration items, with version numbers. The Testware Sets then become baselines instead of configuration items. The Test Suite remains as a baseline but is a baseline comprising other baselines (rather than configuration items directly). A new version of the Testware Set baseline will then comprise those artifacts that have not changed and the new versions of those that have changed. This alternative way of managing the configuration may be better in some situations (particularly where a good configuration management system is in place). More than one version of a Testware Set may coexist in the Testware Library but only one version of a Testware Set can be held in a Test Suite.

Physical implementation

Our description of the Testware Sets and Test Suite has been confined to a logical view of them. That is, we have described the relationship between the various artifacts but have not specifically stated how these relationships might be implemented. Of course any one implementation is unlikely to suit everyone but the one that we favor makes extensive use of a hierarchy of

directories (folders). We do not necessarily expect readers to implement verbatim the structure we describe, but rather adapt it to better suit their own needs.

5.3.7.1 Testware Set Directories We strictly maintain the same physical structure for every Testware Set regardless of where it is located (in the Testware Library or in a Test Suite). This is to facilitate the use of standard utilities for performing a range of useful tasks and avoids any restructuring of the Testware Sets as they are copied from one place to another. We propose a basic physical structure of a Testware Set Directory as shown in Figure 5.12. The whole Testware Set is held in a directory (folder) that contains a series of subdirectories, one for each category of testware. The same subdirectory names are used consistently throughout all Testware Sets. This hierarchical structure of directories makes them very easy to copy and move about. In most environments it is possible to copy or move a directory hierarchy (sometimes this is referred to as a directory tree) with a single command or user interaction (such as 'drag and drop'). This is not the only way that the testware in the Testware Sets could be organized physically, but this implementation has worked well in a number of organizations. We are basically taking the logical files of a Testware Set (for example t\_ScribbleBreadth vl, a Test Set in Figure 5.4) and arranging them into various physical subdirectories based on the type of file (this is shown in Figure 5.14). So all of the expected results from the Test Set would be put into a subdirectory called Expected, for example.

Note that not every lower-level subdirectory would be present for all Testware Set Directories. A Test Set Directory could contain all of the subdirectories shown in Figure 5.12; a Data Set Directory would not contain script, source, or utility subdirectories, a Utility Set Directory would not contain scripts, data, or expected results, and a Script Set Directory would contain only scripts and documentation. Figure 5.13 shows an example of the physical structure of each type of Testware Set. The Test Set t\_ScribbleList contains one document (the test specification), which goes into the document subdirectory 'Doc.' The four scripts go into the Scripts subdirectory, and five expected outcome files (in this case all are Scribble documents) go into the Expected subdirectory. There are no data files in this Test Set (t\_ScribbleList) as these test cases use shared data files, which are held in the separate Data Set d\_ScribbleTypical. There are no utilities in this Test Set, as they are held in the Utility Set u\_GeneralCompare. Therefore the Test Set subdirectory only contains the Doc, Script and Expected subdirectories; the other subdirectories are omitted. Figure 5.13 also shows an example physical implementation of a Script Set, s\_ScribbleDocument, which contains a documentation subdirectory (Doc) and a script subdirectory (Scripts), but no other subdirectories. Similarly, the physical implementation of the Data Set contains a documentation subdirectory (Doc) and a data subdirectory (Data). The Utility Set is implemented by three subdirectories, including one for documentation (Doc). The Utility Set includes two source files UStoUKDate.frm and UStoUKDate.vbp. These are used to build the executable image UStoUKDate.exe. In our test automation regime the executable image is not held in the Testware Library but is generated from the source after the Utility Set has been copied into a Test Suite. This is a task that can be undertaken as a Test Suite or Testware Set pre-processing task (see Chapter 6). The subdirectory for all the executables (Utilities) includes the compiled program (.exe suffix) and two command files (.cmd suffix) which are interpreted rather than compiled. This means that we have all executable utilities in one subdirectory.

Test tool problems

Two different test tools were used for Projects A and B. Later, I saw similar problems to what was experienced in these projects in other projects using different tools. One of the problems we encountered was that the test tool did not support the development tool being used. This caused several problems, but the main problem was that the tool would not recognize the elements of the screens (e.g. radio buttons) as objects, so we had to rely on relative positioning at the pixel level to identify what is on the screen. If one tester then resized a window after the tests had been automated the first time, all of those tests would fail in the next regression. This was one of the major problems in both projects. It is obvious that different tools have different benefits and problems, but if you do automation 'wrong,' selecting the 'right' tool will not save you.

19,3.2 The testers and their problems The testers in both projects faced several problems. It was a conflict for the product people now being used as testers to concentrate on both testing the system (manually) and recording the tests for later automated regression testing. Testing the server transactions sometimes included several steps to build test data to be used later in the test. For example, one function would add a record to the database that would be used by the next function to be tested. This type of function, whose only purpose was to build test data to be used by the function actually being tested, was called a

'build data transaction.' Very often these 'build data transactions' had already been tested, so recording them for audit trail purposes was therefore not necessary. However, to be able to convert the recorded scripts into an automatic regression test, all 'build data transactions' must be able to be replayed, and therefore do need to be recorded. Very often the product specialist forgot to record these data building transactions, and therefore recorded an incomplete test, making life miserable for the test automation specialists (and later again for the testers). The tester had to keep track of his or her test data requirement to be able to test the function he or she was concentrating on at the moment, but also had to remember to record all these transactions to be able to do a playback later. This conflict of interest (i.e. having to concentrate on building a complete set of test data and also on recording for playback purpose) demotivated the testers and made them less productive as manual testers. The main problem for the testers focusing on test automation in Project B (home banking) was the instability of the application when it came to user acceptance testing. This was a RAD project, and unfortunately, too many problems were detected during the user testing and too many major changes were implemented in the application. All these major changes caused severe problems to the attempts to automate any testing. The reason for this is that the automated testing was based on recording of the tests of one version of the software. Those tests were played back automatically when the next release was received. If a field had moved or some text had changed, this would then be reported as a difference (and a potential error). Because of the number of these types of changes in every release (which was done on a weekly basis), the maintenance of the Test Suite became a big problem.

附录B 外文翻译—译文部分

原文出处：<https://www.doc88.com/p-9763109802100.html>

什么是测试软件架构？

Testware是我们用来描述测试所需的所有工件的术语，包括文档、脚本、数据和预期结果，以及测试生成的所有工件，包括实际结果、差异报告和摘要报告。体系结构是所有这些构件的排列；也就是说，它们存储和使用的位置、它们是如何分组和引用的，以及它们是如何更改和维护的。当然，有些测试软件特定于自动化测试（如脚本），有些则特定于手动测试（如测试过程）。然而，许多测试软件对于自动化和手动测试（如测试计划和测试数据）都是通用的。在本章中，我们将专门介绍与自动化测试相关的测试软件体系结构，但我们提出的要点和建议同样适用于手动测试。事实上，我们建议开发测试软件体系结构来覆盖所有的测试软件。不管工件是与手动测试还是自动测试、单元测试还是系统测试、功能测试还是非功能测试相关，都应该没有区别。本章首先强调了四个关键问题，我们认为如果要在测试自动化方面取得长期成功，就必须解决这些问题。本章最后介绍了一种测试软件体系结构的方法，我们发现这种方法非常有效。它可能不适合每个人，因为不同的环境和系统提供不同的问题，需要不同的解决方案。然而，即使一般的方法不适合你，我们相信，可能有一些想法，可以证明是有用的，如果采取或至少适应。

平台和环境独立性

第四个测试软件体系结构问题仅适用于同一软件必须跨不同环境或硬件平台进行测试的情况。理想情况下，所有测试都是独立于平台的，这样相同的文件、脚本等就可以在每个平台或每个环境中使用。在实践中难免会有一些分歧。一些预期的结果可能会有所不同。由于特定操作系统实现“标准”包的方式，使用相同输入值的相同计算可能会产生不同的数值结果。不同的平台可能会以不同的顺序、不同的颜色或字体等显示在屏幕上。这些差异中的一些对人类测试人员来说是“看不见的”，但会影响自动化测试。这就是为什么使用第4章中描述的过滤器是有用的。可能测试设置指令会因不同的环境而改变，例如针对不同的数据库管理系统。如果需要为给定的测试设置和填充测试数据库，则创建数据库的命令，甚至输入数据的方式，可能与一个专有数据库到另一个专有数据库有很大不同。如果需要跨不同平台进行自动化测试，那么测试软件体系结构应该提供一种方法，即在不需要重复所有测试软件的情况下保存与平台相关的信息的必要副本。

配置管理

有两种管理测试软件配置的替代方法。我们喜欢的方法是将测试软件集作为配置项（即具有版本号）存储在测试软件库中。组成每种类型集合内容的单个测试软件工件没有自己的版本号。这样做的效果是，每当测试软件集中的任何内容发生更改时，都会创建一个新版本的测试软件集，其中包含已更改的工件和未更改的工件。基线是具有版本号但包含许多配置项的东西，基线的每个版本都是根据配置项的特定版本生成的。在图5.10中，我们的基线是一个测试套件，由作为配置项的测试软件集组成。管理测试软件配置的另一种方法是让各个测试软件构件具有自己的版本号。在这种情况下，只要测试软件集中的任何内容发生更改，只有更改的工件才有新版本。图的最底层将成为配置项，并带有版本号。然后，测试软件集成为基线，而不是配置项。测试套件仍然作为基线，但它是包含其他基线（而不是直接配置项）的基线。一个新版本的测试软件集基线将包含那些没有改变的工件和那些已经改变的新版本。这种管理配置的替代方法在某些情况下可能更好（特别是在配置管理系统良好的情况下）。测试软件库中可能存在多个版本的测试软件集，但测试套件中只能保存一个版本的测试软件集。

物理实现

我们对测试软件集和测试套件的描述仅限于对它们的逻辑视图。也就是说，我们已经描述了各种工件之间的关系，但还没有具体说明如何实现这些关系。当然，任何一个实现都不可能适合所有人，但是我们喜欢的那个实现广泛地使用

目录（文件夹）。我们不一定期望读者能够一字不差地实现我们描述的结构，而是要使其更适合自己的需要。

测试软件集目录我们严格地为每个测试软件集维护相同的物理结构，不管它位于何处（在测试软件库或测试套件中）。这是为了方便使用标准实用程序来执行一系列有用的任务，并避免在将测试软件集从一个地方复制到另一个地方时对它们进行任何重新构造。我们提出了一个测试软件集目录的基本物理结构。整个测试软件集保存在一个目录（文件夹）中，该目录（文件夹）包含一系列子目录，每个目录对应一类测试软件。相同的子目录名在所有的测试软件集中被一致地使用。这种目录的层次结构使它们很容易复制和移动。在大多数环境中，可以使用单个命令或用户交互（如“拖放”）复制或移动目录层次结构（有时称为目录树）。这并不是测试软件集中的测试软件可以物理组织的唯一方法，但是这个实现在许多组织中都很有效。我们基本上是取一个测试软件集的逻辑文件并根据文件的类型将它们排列成不同的物理子目录。例如，测试集的所有预期结果都将放入名为expected的子目录中。

请注意，并非所有测试软件集目录的所有下级子目录都存在。数据集目录不包含脚本、源或实用程序子目录，实用程序集目录不包含脚本、数据或预期结果，脚本集目录只包含脚本和文档。显示了每种类型的测试软件集的物理结构示例。测试集túu ScribbleList包含一个文档（测试规范），该文档进入文档子目录“Doc”。四个脚本进入脚本子目录，五个预期结果文件（在本例中都是Scribble文档）进入预期子目录。此测试集中没有数据文件（t\_ScribbleList），因为这些测试用例使用共享数据文件，这些文件保存在单独的数据集中d\_ScribbleTypical中。此测试集中没有实用程序，因为它们保存在通用比较实用程序集中。因此，测试集子目录只包含Doc、Script和预期的子目录；其他子目录则被省略。它还显示了脚本集的一个物理实现示例，即ScribbleDocument，它包含一个文档子目录（Doc）和一个脚本子目录（Scripts），但没有其他子目录。类似地，数据集的物理实现包含一个文档子目录（Doc）和一个数据子目录（Data）。实用程序集由三个子目录实现，其中一个子目录用于文档（Doc）。实用程序集包括两个源文件UStoUKDate.frm和UStoUKDate.vbp。它们用于生成可执行映像UStoUKDate.exe。在我们的测试自动化机制中，可执行映像不保存在测试软件库中，而是在实用程序集复制到测试套件后从源代码生成的。这是一个可以作为测试套件或测试软件集预处理任务执行的任务。所有可执行文件（实用程序）的子目录包括编译的程序（.exe后缀）和两个命令文件（.cmd后缀），这些文件是解释的，而不是编译的。这意味着我们在一个子目录中有所有的可执行实用程序。

测试工具问题

项目A和项目B使用了两种不同的测试工具。后来，我看到了与使用不同工具的其他项目在这些项目中遇到的类似问题。我们遇到的问题之一是测试工具不支持正在使用的开发工具。这导致了几个问题，但主要问题是工具无法将屏幕元素（如单选按钮）识别为对象，因此我们必须依赖像素级的相对定位来识别屏幕上的内容。如果一个测试人员在测试第一次自动完成后调整了窗口的大小，那么所有这些测试都将在下一次回归中失败。这是两个项目的主要问题之一。很明显，不同的工具有不同的好处和问题，但如果你做自动化“错误”，选择“正确”的工具不会拯救你。

测试人员及其问题两个项目中的测试人员都面临一些问题。对于现在被用作测试人员的产品人员来说，集中精力测试系统（手动）和记录测试以供以后的自动回归测试是一个冲突。测试服务器事务有时包括几个步骤来构建测试数据，以便稍后在测试中使用。例如，一个函数将向数据库中添加一条记录，供下一个要测试的函数使用。这种类型的函数被称为“生成数据事务”。通常这些“生成数据事务”已经过测试，因此不必出于审计跟踪目的记录它们。但是，要将记录的脚本转换为自动回归测试，必须能够重播所有“生成数据事务”，因此确实需要记录。产品专家常常忘记记录这些数据构建事务，因此记录了一个不完整的测试，这使测试自动化专家（以及后来的测试人员）的生活很痛苦。测试人员必须跟踪他或她的测试数据需求，以便能够测试他或她目前正在关注的功能，但也必须记住记录所有这些事务，以便稍后能够进行回放。这种利益冲突（即必须集中精力建立一套完整的测试数据，并记录下来以便回放）降低了测试人员的积极性，降低了他们作为手动测试人员的工作效率。在项目B（家庭银行）中，测试人员关注测试自动化的主要问题是应用程序在进行用户验收测试时的不稳定性。这是一个RAD项目，不幸的是，在用户测试期间检测到太多的问题，并且在应用程序中实现了太多的主要更改。所有这些主要的变化都给自动化测试的尝试带来了严重的问题。原因是，自动化测试是基于记录软件的一个版本的测试。当收到下一个版本时，这些测试会自动回放。如果某个字段已移动或某个文本已更改，则会将其报告为差异（和潜在错误）。由于每一个版本中都有很多这样的更改（每周都有），所以测试套件的维护就成了一个大问题。

附录C 软件使用说明

1.系统介绍

在线轮渡购票微信小程序主要是为了方便人们的日常出行而开发的一款微信小程序。随着社会的不断发展，人们生活水平的不断提高，人们出行的交通工具也跟随社会的发展不断革新。乘坐轮渡外出旅游的人数也不断提升，单凭线下购票的渠道已经不能满足人们的出行需求，所以在线购票的开发势在必行。而开发该微信小程序，不仅可以为广大用户的日常出行提供了极大的便利，还可以加速轮渡行业的发展，为当地的经济发展提供一份动力。除此之外，小程序可以转发到微信圈，微信群，微博等社交平台分享给好友，通过客户各自的社交渠道推广小程序，可以以此提高品牌推广。

2.环境要求

在线轮渡购票微信小程序主要运行于能够支持微信8.0.0及其以上版本的机器上。

3.操作指南

游客需在微信搜索栏里搜索在线轮渡购票，找到小程序后点击进入小程序。进入小程序后，游客需进入我的界面点击授权登录按钮，按照相应提示完成授权登录步骤。授权登录后，游客返回首页界面如图1-1所示，游客可根据自身购票需求在地名框查找相应地名信息进行搜索如图1-2所示，之后系统会显示出搜索结果如图1-3所示，游客在显示结果中找到自己所需要的票并点击进入购票环节如图1-4所示，按照系统提示输入自身相关信息后点击购买按钮，按照提示完成支付即可。



图1-1 首页界面



图1-2 地名选择框



图1-3搜索结果图

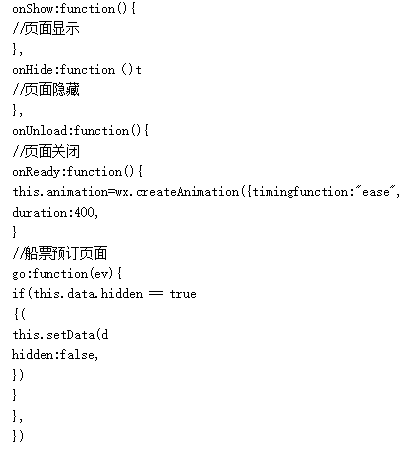


图1-4 订单支付图

附录D 主要源代码

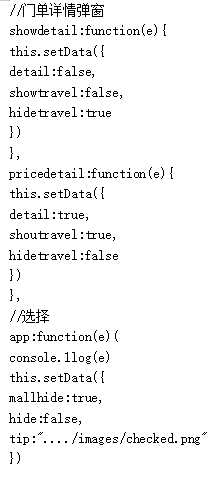
1. 船票预订代码

船票预订代码如下所示：



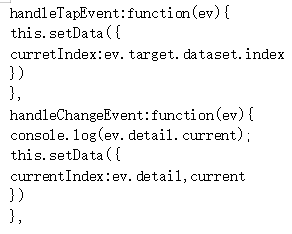
1. 订单详情代码

订单详情代码如下所示，通过这些代码，我们可以查看到订单的详细信息。



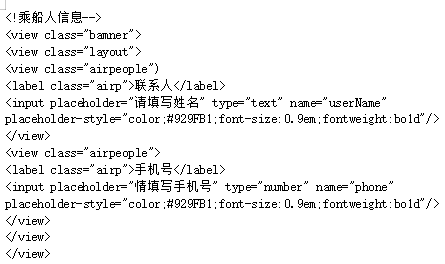
1. 天气查询搜索城市代码

天气查询搜索城市代码如下所示：



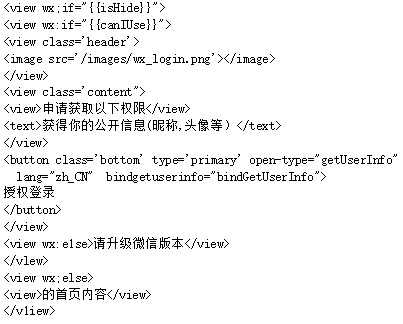
1. 添加乘客代码

添加乘客信息代码如下所示：



1. 授权登录代码

授权登录代码如下所示：



6.提交订单验证代码

提交订单验证代码如下所示：



7.改签代码

改签代码如所示：

