# 广州大学华软软件学院

# 毕业论文(设计)

课题名称　机票预订管理系统设计与实现

系　　别　软件工程系

专业班级　12 级数本 1 班

学生姓名　吴财雄

学　　号　1240120115

指导教师　吴向荣

日　　期　2016 年 4 月 1 日

**摘要** 随着现代社会经济的高速发展和生活水平的提高，人们的生活节奏也在逐渐加快，越来越多的旅客选择以更为舒适和快捷的民航出行；在追求更高层次的精神需求的满足时，人们往往会选择休闲度假等方式，廉价航空的发展与特价机票的增多使更多游客选择坐飞机出行。民航客运正逐步完成由高端化服务向平民化服务的转变。另一方面，科技的飞速发展，特别是互联网与电子支付的发展与普及对人们的出行方式和消费习惯产生了深远的影响。旅客对出行的要求以及机票的电子化，使得建设机票预订管理系统有了需求。

目前面向服务的架构已经成为企业 IT 系统实施的首选方案,而 REST 模式的 Web 服务与复杂的 SOAP 和 XML-RPC 对比显然更加简洁。本系统将基于 REST 风格设计结合 MVC 模式实现，采用 Spring IOC 和 AOP 来提高系统的可拓展性和维护性，选择 Spring MVC 框架来提高可用性和伸缩性，使用 Apache Shiro 提供安全保障，利用 Hibernate ORM 技术对数据进行持久化，运用面向对象编程思维来操纵数据库，提高开发效率。

**关键词** 机票预订；信息管理；REST；MVC 模式

**ABSTRACT** With the rapid development of modern social economy and the growth in living standards, people's pace of life is gradually accelerated, and the more comfortable and efficient the civil aviation travel has been becoming more popular; people tend to choose vacation and leisure taking for pursuit of the satisfaction of spiritual needs, the development of low-cost airlines and Special ticket make more tourists choose to travel by plane. Civil aviation passenger transportation is being completed by the high-end services to transition to civilian services. On the other hand, the rapid development of science and technology, in particular the development and popularization of Internet and electronic payment, are considered highly influential in people's trip mode and consumption habits. Passenger transport demand for travel and electronic ticket, make building flight ticket booking information system necessary.

Service-oriented architecture has become a preferred solution for implementation of enterprise IT system, and the REST mode of Web services is obviously more concise than complex SOAP and xml-rpc. System is built based on the REST style design and MVC pattern, using Spring IOC and AOP to enhance expansibility and maintainability, using Spring MVC framework to improve the availability and scalability，using Apache Shiro for obtain security，and using Hibernate ORM technology to persistent data，manipulate the database with object oriented programming thinking and improve the development efficiency.

**KEY WORDS** Ticket Booking; Information Management; REST; MVC pattern

# 目录

# 前 言

## 背景

随着现代社会经济的高速发展和人民生活水平的提高，人们的生活节奏也在逐渐加快，在出行方式上越来越多的旅客选择更为舒适和快捷的民航出行，民航事业在不断壮大；在追求更高层次的精神需求的满足时，人们往往会选择休闲度假等方式，而随着国民整体休闲度假意愿提升，国家加快落实带薪休假制度促进了旅游业的发展，尤其成为中远途旅行的首选，廉价航空的发展与特价机票的增多使更多游客选择坐飞机出行。搭乘飞机出行的老百姓的数量在呈明显的上升趋势，机票预订在各航空公司的机票销售中占据着主导地位，民航客运正逐步完成由高端化服务向平民化服务的转变。

另一方面，科技的飞速发展，特别是互联网与电子支付的发展与普及对人们的出行方式和消费习惯产生了深远的影响。随着航空公司用户的迅猛增长和人们对便捷性要求的提高，原有的机票预订方式已经无法满足人们的需求，严重制约了航空公司的工作效率，也耽误了用户的宝贵时间，因此，引入高效的机票预订系统，来协助处理机票预订工作是计算机技术高速发展的必然趋势。

## 目的与意义

传统线下销售机票价格不够透明，且存在许多不规范作为，给航空公司内部管理带来了很大障碍，繁杂的线下程序耽误了用户的宝贵时间，航空公司的工作效率也受到了严重制约，机票预订的服务质量大打折扣，无法再满足人们的需求。

本系统旨在提高航空公司的机票预订服务质量和效率，满足旅客对出行的要求，实现真正的机票电子化，使航空公司机票预订工作规范化、系统化、程序化，避免管理的随意性，提高信息处理的速度和准确性，能够及时、准确、有效地进行信息查询和修改；降低售票服务中错误的发生率，减少信息交流的烦琐过程及其带来的开销；

提升票务管理和行政管理水平，增强服务竞争力，实现优质服务，直接或间接提高经济效益。

## 论文结构

本文共分为七章，具体内容安排如下：

**第一章** 相关技术理论介绍，并论述其在本系统中的使用；

**第二章** 系统的需求分析，主要从系统角色和系统的规定与要求等方面对系统的功能性和非功能性需求进行分析；

**第三章** 系统的总体设计，描述系统在运行环境中的总体网络结构，介绍系统中各个功能模块或子模块之间的层次关系；

**第四章** 数据库设计，简单描述数据库的外部设计，着重对数据库的概念结构设计、逻辑结构设计进行定义和陈述；

**第五章** 系统主要功能模块的设计与实现，主要针对核心功能介绍其模块的基本流程、代码的实现以及展示效果进行描述；

**第六章** 结束语，对本系统设计与实现中所做的工作进行总结，给出本课题取得的成果，并指出系统存在不足的地方以及相应的改进方向。

# 1. 相关技术概述

本章主要介绍 REST 软件架构风格的设计原则，系统开发过程中各个模块应用到的一些技术框架等。为后续的开发工作其指导作用。

## 1.1 REST 架构风格

表述性状态转移（Representational State Transfer，REST）是 Roy Thomas Fielding 博士于 2000 年在他的博士论文《Architectural Styles and the Design of Network-based Software Architectures》中提出来的一种以网络系统为基础的架构样式和架构风格。它凭借互联网取得的成功用以说明如何正确地使用 Web 标准，通常满足论文中提出的“客户-服务器”、“无状态”、“缓存”、“统一接口”、“分层系统”以及“按需代码”等约束条件和设计原则的应用程序或设计都可以称为 REST ful。REST 并没有一个明确的标准，它更像是一种设计风格。

REST 的基本设计原则是建立在 SOA 的基础之上的更进一步的发展。主要包括以下几点：

1、资源的抽象

REST 服务中需要暴露出来的所有内容都将被抽象为资源。在网络中这些资源都具有唯一的统一资源标识符(URI：Uniform Resource Identiter)，且都是自我们描述的。客户端无法直接获取一个资源本身，而只能获取该资源的一种表述。表述是具有特定数据格式的用于描述该资源信息的文档，数据格式使用 HTTP 内容标头类型指定。如：XML、JSON、HTML、PNG 等；

2、统一接口

REST 中服务的消费者服务端资源之间的所有交互，都是通过几个 HTTP 协议的标准动作(GET、PUT、POST、DELETE)进行的。各个方法在访问不同资源的时候存在着相同的语义，也就是所谓统一的接口。其中 GET 方法对应对服务端资源的查询请求，它不会导致服务器资源的状态的任何改变，换句话说它的操作是安全的；PUT 方

法对应更新请求，通常它是幂等的，即对同一资源进行的多次操作和对其进行一次操作是等效的；POST 方法对应资源新建请求，它的操作会造成服务端资源状态的改变，是不幂等也是不安全的；而 DELETED 方法则用于删除服务端的资源，通常是幂等的。统一接口使得所有理解 HTTP 应用协议的组件之间能够以同样的方式同样的语义进行正确的交互，达到资源与资源访问者之间的相互协作。

3、超媒体驱动

超媒体驱动（Hypermedia as the engine of application state，HATEOAS）被当做当前应用状态引擎，这个概念的核心是超媒体概念，即通过超媒体转换应用的状态。在服务消费者与提供者之间的每次交互中，并不直接交换应用的状态，而是通过资源状态的表述来进行交换的。

4、无状态服务

REST 要求客户端、服务端之间的交互是没有状态的，即服务无法记住当前请求之外的任何客户端的状态。由于这种无状态行，服务端不需要为每个客户端维护 Context。一方面省去维护客户端状态带来性能消耗。另一方面将很容易提高服务的可扩展性，由于服务器不保存每次请求的状态信息，每次请求对于服务端来说都是一次新的请求，因此可以很容易的通过对服务器的横向扩展来提高系统的整体并发能力。

## 1.2 Spring

Spring 是为了解决企业应用程序开发复杂性而创建的一个 JAVA 开发框架，它由 Rod Johnson 在其著作《Expert One-On-One J2EE Development and Design》中阐述的部分理念和原型衍生而来，其核心是控制反转（Inversion of control ，IoC）和面向切面编程（Aspect Oriented Programming ，AOP）。使用 Spring 主要有以下好处：

1．Spring 采用低侵入式设计，代码污染极低；

2．通过使用 Spring 为我们提供的 IoC 容器，对象之间的依赖关系可交由 Spring 进行管理，降低了业务对象替换的复杂性，提高了组件之间的解耦，简化开发；

3．运用 Spring AOP 开发者可方便进行面向切面的编程，利用它我们可以将一些通用任务如安全、事务、日志等进行集中式管理，从而提供了更好的复用；

4．Spring 的声明式事务，使我们可以从单调烦闷的事务管理代码中解脱出来，灵

活地进行事务的管理，提高开发效率和质量。

5．Spring 不排斥各种优秀的开源框架，它提供了对各种优秀框架的直接支持，并且降低了各种框架的使用难度。其 ORM 和 DAO 还提供了与第三方持久层框架的良好整合，并简化了底层的数据库访问。

6．通过 Spring 我们可以用非容器依赖的编程方式进行几乎所有的测试工作，而且在 Spring 里，测试不再是昂贵的操作，而是随手可做的事情。例如：Spring 对 Junit4 支持，可以通过注解方便的测试 Spring 程序。

7．Spring 采用分层架构，并不强制应用完全依赖于它，开发者可自由选用其部分或全部。

### 1.3 Spring MVC

Spring MVC 是一个基于 Java 的实现了 Web MVC 设计模式的请求驱动类型的轻量级 Web 框架，其使用了 MVC 架构模式的思想，将 web 层进行职责解耦，基于请求驱动指的就是使用请求-响应模型，其与 Spring 有着天然的无缝集成的优势，也简化了我们日常的 Web 开发。

### 1.4 Hibernate

Hibernate是一个基于Java的持久化中间件，它不仅提供ORM映射服务，还提供数据查询和数据缓存等功能，通过使用Hibernate API可以在很方便地运用面向对象编程思维来对数据库的数据进行操作。此外，Hibernate可以和多种Web服务器、应用服务器良好集成，并且支持多种缓存技术和连接池技术以及几乎所有流行的数据库服务器。由于对JDBC做了轻量级的封装，在任何使用JDBC的场合，无论在Java的客户端程序还是在Servlet/JSP的Web应用中都可以使用Hibernate来完成持久化的重任。

# 2. 需求分析

## 2.1 用户角色及特点

系统中的用户根据权限的不同大致可以分为三类，分别是旅客、客服以及管理员。

1、旅客：

互联网用户即需要预定机票的旅客，可以通过系统进行航班查询、机票预订、订单支付、订单查询、订单操作申请、用户账号基本信息和常用旅客信息的维护等，在下文中简称“旅客”。

2、客服：

客服中心业务员，主要负责接收客户的来电，根据其需求为客户提供航班查询和机票预订以及出票服务，还需根据客户的申请，对客户的订单进行退票、改期、签转、升舱、废票和换开等操作，并对客户订单状态进行维护。在下文中简称“客服”。

3、管理员：

管理员主要负责系统中如机场信息、机型信息、航空公司、航空客规等基础信息的维护、对客服信息和旅客信息的管理维护以及周期性更新票务相关数据的更新，另外还负责操作日志管理等。在下文中简称“管理员”。

本系统运行在互联网上，系统用户应具备有关机票预订的基本概念及流程。旅客只需要通过 PC 机或只能移动设备就可完成简单的操作与系统交互；客服则需要一定程度上熟悉航空机票预订的详细流程和相关概念；而管理员还需要有一定的系统管理知识以及数据库相关概念。客服和管理员需要经过培训。

## 2.2 系统用例

经过前期对航空票务的市场调研，结合机票预订流程对系统各角色参与的用例进行分析、统计，最后分析系统边界进行过滤后，勾勒出本系统所包含用例的用例图如下：


图 3-1 系统用户角色

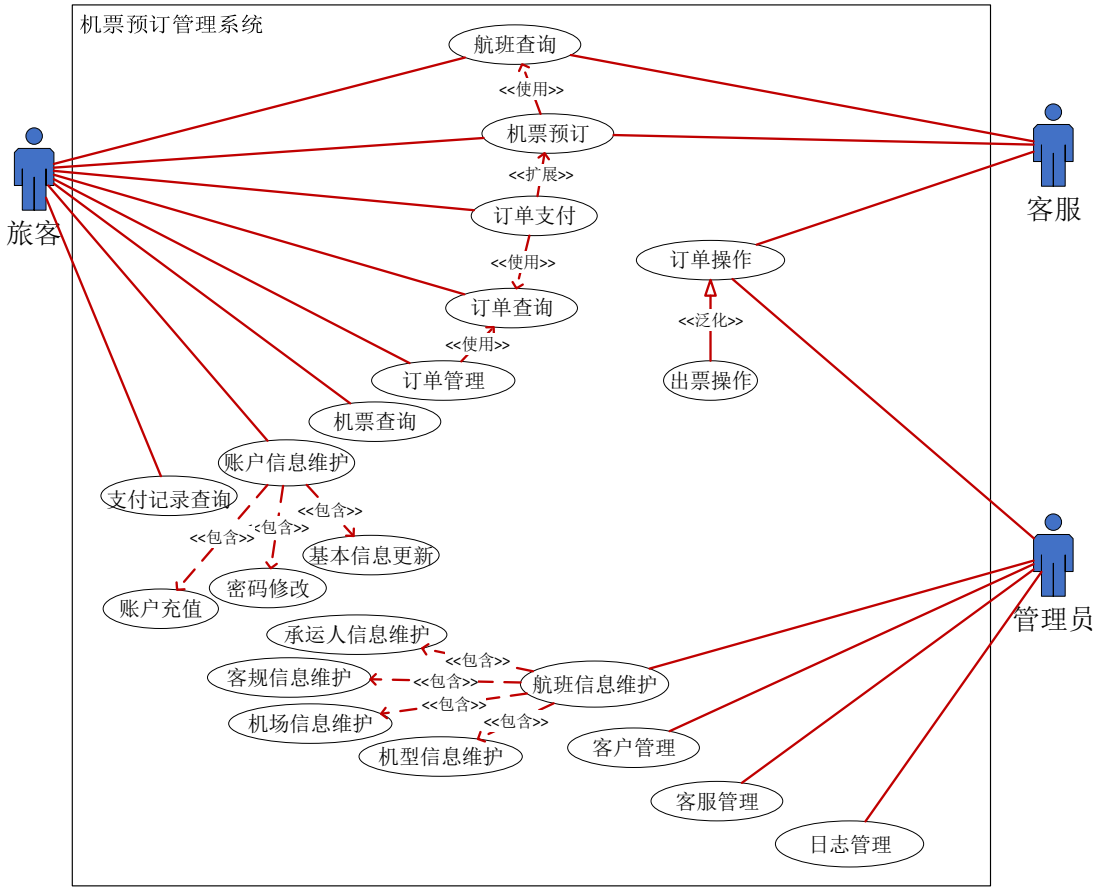


图 3-2 系统总用例图

### 2.2.1 系统主要用例

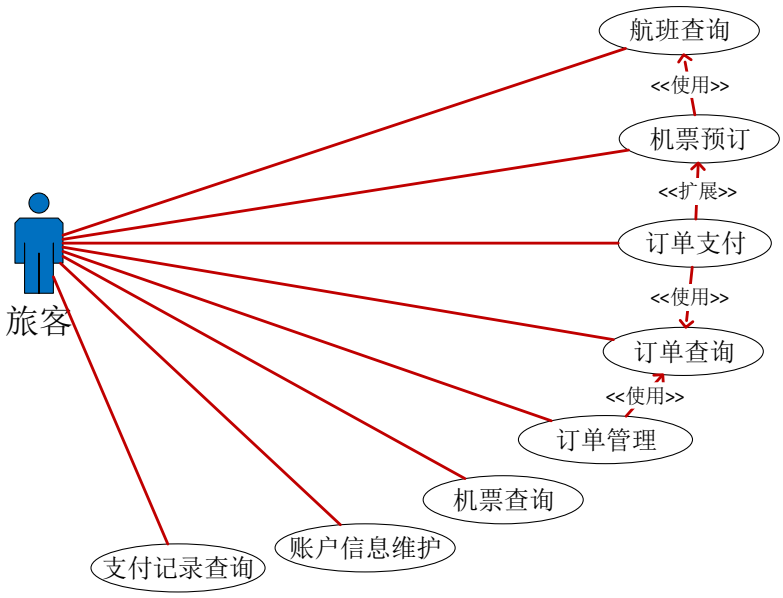


图 3-3 与旅客相关联的用例图

#### 2.2.1.1 航班查询

旅客在进入首页后可在相应页面模块选择出发城市、到达城市和出行日期等查询信息后进行航班查询。系统完成查询后将航班时刻列表返回给页面，若查不到满足查询条件的航班这提示暂无航班满足条件；否则显示结果的每一条包括承运人、航班号、出发机场、出发时间、到达机场、到达时间、机型、总航程和旅途时长、经停点、中转航班(如果有)、有无餐食服务以及所提供最低折扣舱位服务等级、最低折扣及其相应的价格等；默认查询结果按最低价从低到高排序，旅客可选择按出发时间或所提供最低折扣对查询结果进行升降序排序，还可以根据航空公司、有无经停点和是否直达对查询结果进行过滤；

旅客可以点击列表项右侧的“查看”按钮，或单击航班列表中其中一项航班时刻记录可看各舱位的详细信息。下拉选项列表中将包括该航班所提供的各舱位的舱位服务等级、舱位折扣率以及对应的机票票面价等。旅客还可以在下拉选项列表中任一项点击“退改详情”超链接来浏览对应的退改签详情。

#### 2.2.1.2 机票预订

旅客在选定要预订的航班及相应的舱位后，其中显示剩余票数不为 0 且“预订”按钮非不可用状态则旅客可单击“预订”按钮进行机票预订，系统进入预订信息填写页面，旅客将可在该页面看到详细的航班信息以及票价、机建+燃油费和航班当天的备注说明等。旅客需要在该页面填写乘机人信息、联系人信息等信息，其中联系人信息包括姓名和手机号码，乘机人信息包括乘机人姓名、乘机人类型、证件类型、证件号码等，如果旅客已登录且事先存有常用乘机人或者联系人都可直接选择而无需重复输入，多个乘机人可单击添加乘机人按钮添加乘机人。系统将根据乘机人数和乘机人类型以及航班信息，自动计算并显示该订单的总价。

在完成乘机人信息、联系人信息后旅客可以点击“提交”按钮，系统将进入订单确认页面。旅客确认订单信息无误后点击“确定”按钮，系统将在后台根据用户提交的信息与相应的数据提供商(全球分销系统 GDS，例如国内的中航信)交互，并获取返回的 PNR,如果订座成功则将进行相应处理然后将处理后信息存入数据库并提示机票预订成功，系统将进入订单支付环节；如果 PNR 生成失败则提示旅客订单生成失败请重新提交订单。如果出现系统异常则将事务回滚并提示旅客预订失败。

#### 2.2.1.3 订单支付

旅客在确认订单信息后进入订单支付环节，系统将跳转到相应的付款页面，旅客完成支付后系统将生成相应支付记录并更新订单状态，记录返回的支付方式、支付卡号以及支付流水号等支付信息，并显示付款明细、提示付款成功请耐心等待机票的出票。

旅客也可以点击“稍后支付”选择暂不支付。系统将提示 3 天，航班起飞前 24 小时内未付款该订单将自动取消，最后跳转回首页。

#### 2.2.1.4 订单查询

旅客在订单查询页面可直接根据订单号来查找订单信息。在查询文本框内输入订单号点击“查找”按钮后，系统将查找并返回相应的订单信息。若查找不到则提示订单不存在；若查找出来则将对旅客权限进行验证。如果订单号对应的旅客账户并非

本登录账户则系统提示无权限查看，若被查询订单存在且由匿名旅客下单则系统提示要求输入如下单时的联系人信息来验证该旅客是否有权限查看该订单信息，有则系统返回查询到的订单信息并将其显示出来，包括订单号、下单时间、PNR、以及相应的行程信息列表、订单总价格和订单状态等信息，没有则提示无权限查看。

旅客还可以选择要查询订单的下单日期区间，根据订单的下单时间来查询本账户下的单。在订单状态多选框中旅客还可以勾选要查询订单的状态来过滤掉不想被查找到的订单。在确认查询条件后旅客单击“查询”按钮后，系统将返回符合查询条件的订单列表，每一订单会在订单列表中显示订单号、下单时间、PNR、以及相应的行程信息列表、订单总价格和订单状态等信息。其中行程列表中每段行程的具体信息将包括出发机场(航站楼)和抵达机场(航站楼)、出发日期时间和到达日期时间、乘机人数等。如果没有满足查询条件的订单则提示未找到相应的订单记录。

#### 2.2.1.5 订单管理

旅客在登录状态下进入个人主页，点击导航栏中我的订单按钮进入订单浏览界面后可有选择性过滤地浏览个人账户下的所有订单，指定具体订单后可对其进行管理，删除后该订单信息将不会再被浏览。

在具体订单中删除按钮为可用的状态下旅客点击删除按钮则可进行订单的删除，系统将提示旅客确认是否进行订单删除。若旅客确定则系统后台将进行订单的非物理删除及相关操作，并提示旅客删除成功且将该订单信息从订单列表中移除，若出现异常则提示删除失败请稍后重试并在后台进行相应处理。

#### 2.2.1.6 机票查询

旅客在首页导航栏中可单击机票查询按钮进行机票查询，输入机票票号后系统将进行相关查询。若查询不到则提示没有该机票，否则若旅客未登录则提示请先登录并将页面跳转到旅客登录界面；若查询到机票对应订单的旅客账户并非本旅客账户则提示无权限查看，否则系统将返回机票详细信息，显示结果包括机票票号、所属订单编号、航班类型、行程列表、乘机人数及机票价格明细等详细信息。若出现异常则提示查询失败并在后台进行相应处理。

#### 2.2.1.7 账户信息维护

旅客在登录状态下进入个人主页，点击导航栏中账户按钮则可进行账户信息的维护。账户信息的维护用例又包括基本信息的更新、密码修改和余额充值三个子用例。

旅客单击基本信息按钮可进行基本信息的更新，系统将用可编辑控件显示账户当前包括手机号码和邮箱地址等基本信息。旅客进行基本信息的编辑后更新按钮将从不可用状态变为可用状态，旅客点击更新按钮提交信息更新，系统后台将做相应的更新操作并提示更新成功，若出现异常则提示更新失败请稍后再试；

旅客单击密码修改按钮可进行账户密码的修改，系统将密码修改页面并提示要求输入旧密码、新密码、重复输入新密码。旅客在正确输入旧密码和新密码等信息后点击修改按钮可完成密码的修改，系统将对旧密码进行验证。若旧密码不正确则提示旧密码不正确请重新输入，否则进行账户密码的更新并提示修改成功。
若出现异常则提示修改失败请稍后再试；

旅客在菜单栏中单击余额按钮系统将跳转到余额相关页面，旅客可查看当前账户余额，单击充值按钮系统将显示相应的表单供旅客编辑，旅客在输入充值额度后点击支付按钮系统将跳转到支付页面，完成支付后系统将形成相关的支付记录，记录返回的支付方式、支付卡号以及支付流水号等支付信息，更新账户余额并显示付款明细等。若出现异常则提示充值失败请稍后再试。

#### 2.2.1.8 支付记录查询

旅客在首页导航栏中可单击机票查询按钮进行机票查询，输入机票票号后系统将进行相关查询。若查询不到则提示没有该机票，否则若旅客未登录则提示请先登录并将页面跳转到旅客登录界面；若查询到机票对应订单的旅客账户并非本旅客账户则提示无权限查看，否则系统将返回机票详细信息，显示结果包括机票票号、所属订单编号、航班类型、行程列表、乘机人数及机票价格明细等详细信息。若出现异常则提示查询失败并在后台进行相应处理。

## 2.3 规定与要求

跟普通的电子商务网站想比机票预订系统并无自己的库存，而是航空公司与诸多代理人之间共享一个库存，无论哪个代理人或承运人出售机票都会造成总库存的减少。虽然目前航空出行已逐渐平民化，但由于价格、天气以及其它元素，目前航空出行还未成为普遍大众出行的首选方式，特别是短距离的出行。因此机票预订系统并不像时下比较热的火车票以及高铁票或者其它电子商务网站搞活动大促销时那样需要瞬时处理非常多的需求。相对于瞬时并发，机票预订系统更关注不间断的业务处理能力。

机票预订系统需保证在 7*24 小时内全天候运行。仅在航信、承运人或者数据提供商进行系统维护时停止对外提供服务。因此依赖于数据提供商的实时航班查询和预订，允许 200 旅客可以同时进行操作；而订单查询等仅需要本地服务即可完成的服务则可同时支持 1000 用户同时访问。

系统安全方面输入域需严格防止脚本或 SQL 注入，用户客服端(浏览器)与服务器之间不能以明文方式传输敏感数据，与数据提供商之间的数据传输必须通过加密的方式进行传输。机票预订信息以及用户账户的登录密码、支付信息或其他设计用户敏感信息的数据需经过加密处理。

# 3. 总体设计

本章主要介绍系统的总体结构设计，包括系统的软件层次结构以及各功能模块的设计。

## 3.1 软件层次架构设计

整个系统主要包括机票预订、后台管理两大功能模块，为最终用户提供页面的访问功能，实现旅客在互联网的环境中机票的预订和管理人员对数据的管理。以及日志和安全两个系统辅助模块，安全模块负责协助后台管理以及保证系统正常运行，各功能均采用统一的日志模块进行日志管理。系统的正常运行除了数据库系统的支持外还需由数据库提供商提供相应的数据源以及第三方支付平台的支持。系统的逻辑结构如图 **3-1** 所示：

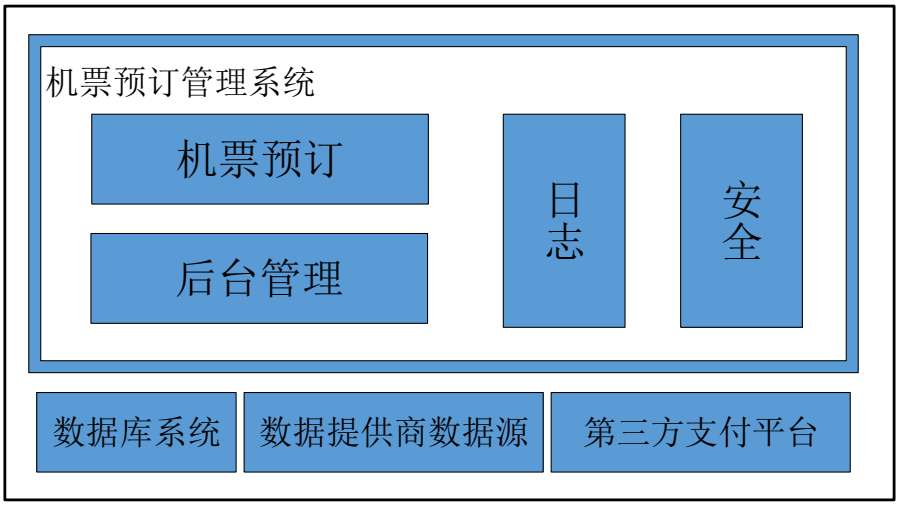


图 3-1 系统逻辑结构图

## 3.2 功能模块设计

机票预订模块包含的子模块有航班查询、机票预订、机票查询、订单管理以及账号管理。其中，航班查询主要提供系统用户对航班时刻表的查询功能，用户在查询前除必要信息外可输入最少剩余票数、起飞时间段、航空公司、舱位服务等级等约束过滤掉自己不想要的航班信息，查询到相应结果后用户还可以根据航空公司、起飞时间、舱位服务等级以及价格高低等对查询结果进行排序或过滤；机票预订则提供系统用户在航班查询的基础上进行在线机票预订；机票查询提供旅客在进行机票预订后随时根据机票票号或 PNR 进行查询功能，用户在查询前若未登入系统则需先进行登录认证，验证有相应的权限后才予浏览机票详细信息；订单管理指旅客对自己账户内订单的管理，旅客无法直接修改订单信息，只能对订单进行删除操作，若旅客想对订单进行退票、改期和签转等操作则需想系统提交更改申请，由有相关权限的客服人员代为操作。最后账号管理为旅客提供了账号密码修改、账号基本信息更新以及账号充值等功能。

后台管理模块主要包括机场信息管理、机型信息管理、航空公司及其客规等基础信息管理，以及旅客和客服的账号管理，当然还有订单的管理。其中基础信息的管理允许拥有权限的管理员对机场、机型、航空公司、航空客规这些资源进行除了增删改查功能，客服和旅客只有查询的权限；管理员对旅客账户只能进行锁定操作，无法删除旅客账户，也无法修改其账号密码和基本信息，但管理员对客服账号可进行增删改查所有操作。

安全模块主要包括实现系统用户的登录认证授权以及各操作的权限检查，一次来保障用户账户信息以及系统的安全。

日志模块主要是使用配置的方式对系统其他各个模块生成日志的统一管理。

系统的功能结构如图 3-2：

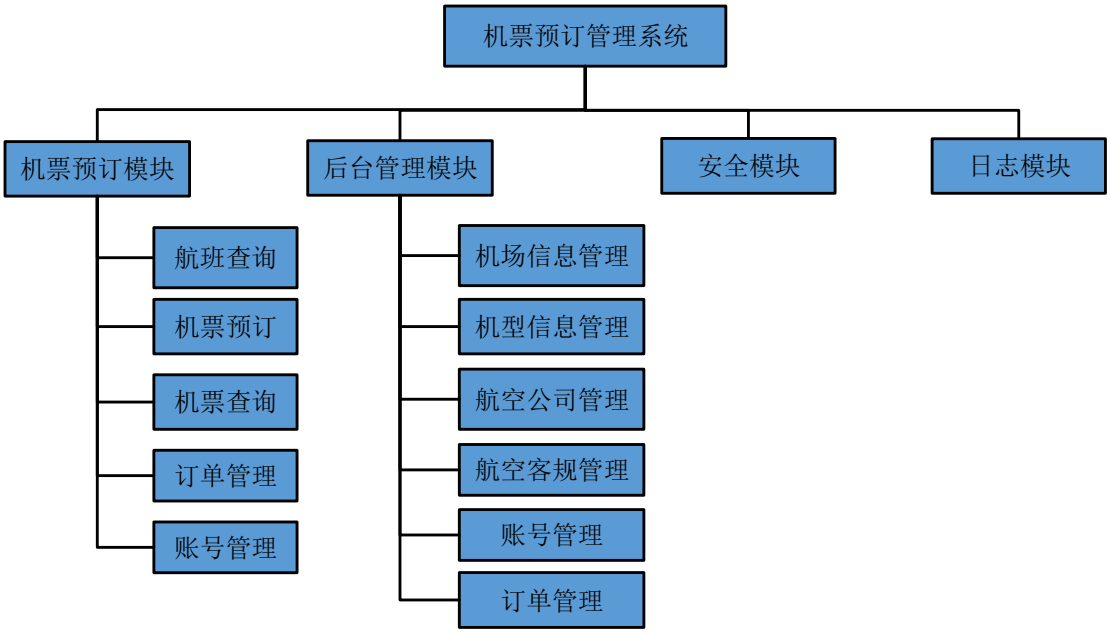


图 3-2 机票预订信息系统功能结构图

# 4. 数据库设计

## 4.1 外部设计

### 4.1.1 标识符和状态

数据库服务器版本： Oracle 12C

### 4.1.2 约定

#### 4.1.2.1 命名：

1、命名使用富有意义的英文词汇，尽量避免使用缩写，多个单词组成，中间以下划线“_”分割；

2、避免使用 Oracle 的保留字如 LEVEL、关键字如 TYPE（见 Oracle 保留字和关键字）；

3、各表之间相关列名尽量同名；

4、除数据库名称长度为 1-8 个字符，其余为 1-30 个字符，Database link 名称也不要超过 30 个字符；

5、不定长字符类型除全是中文用 NVARCHAR2 外，其余的用 VARCHAR2，避免使用 VARCHAR；

6、命名只使用英文字母，数字和下划线，不使用汉语拼音，表名不使用复数；

7、表名命名规则为 TRS_yyy_TableName。其中 TRS 表示该（子）系统，yyy 表示子系统中的某子模块的名称（可以没有），TableName 为表的含义。视图则为 TRS_yyy_ViewName_V；

8、存储过程等尽量放在包内，命名以“PRO_”开头，函数则以“FUC_”开头，触发器以“TRG_”开头，序列命名以“SEQ_”开头，索引不论什么索引统一以“TableName_ColumnName_idx”的形式命名。当然，默认的序列以“TableName_SEQ”

命名，默认的表的 Before Insert for each row 触发器则以 TableName_TRG 命名；

9、无实际意义的单主键统一命名为 ID，特别例子如订单编号可命名为 NO;

#### 4.1.2.2 数据类型和取值：

1、任何表示密码字段都采用摘要算法进行转化为 16 进制编码后存储;

2、包含业务逻辑的所有实体表新增一个 IS_DELETED 字段，用于表示该记录所对应的实体是否已被删除。为保证数据安全性以及供后期日志处理和申诉对照等，需对数据进行非物理删除。该字段数据类型为 CHAR(1 BYTE)，并用小写 y 和 n 代替布尔类型表示是否已被删除；

3、可能被操作的实体表新增一个 IS_LOCKED 字段，用于表示该记录所对应的实体是否已被锁定，例如异常的账号、异常的订单等在异常的情况下需进行锁定并需由特定权限的角色进行解锁。该字段数据类型为 CHAR(1 BYTE)，并用小写 l 和 u 来代替 locked 和 unlocked 表示是否被锁住。

4、诸如备注这样的字段以空字符串''来替代 null，避免在查询语句中出现 is null 或 is not null 等字样；

#### 4.1.2.3 约束：

所有字段值的约束属于范围约束的统一在 check 约束中用范围表达式表示出来，无法用范围表达式表达出来的则用正则表达式进行约束，正则表达式都无法约束的则通过触发器进行对记录的 insert 和 update 进行约束。

## 4.2 结构设计

### 4.2.1 概念结构设计

根据系统核心功能模块的划分，机票预订模块主要围绕着订单实体展开，包括订单、旅客、行程、机票、支付信息等实体，后台管理模块包括航班时刻表、运价缓存、机场、机型、航班公司和航空客规等实体，另外日志模块还需要对操作日志实体进行操作。

下面将以订单为中心，描述机票预订模块中各实体及它们之间的关系：

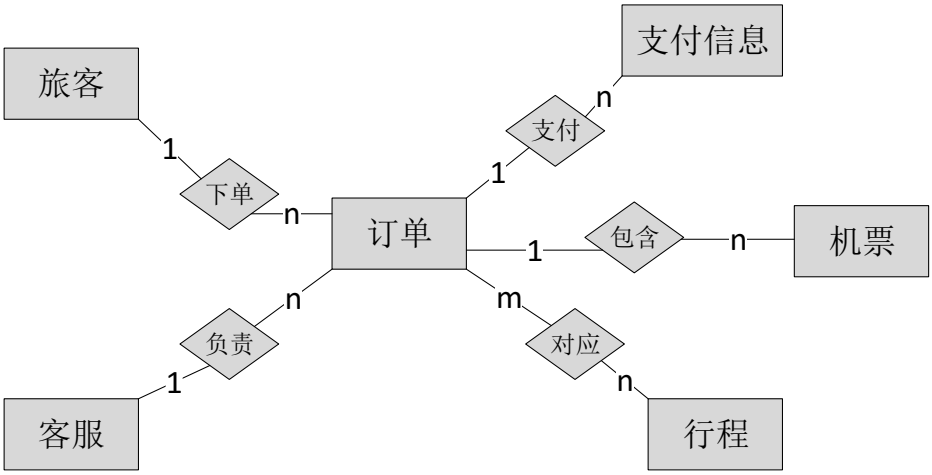


图 4-1 机票预订模块主要 E-R

1、每个旅客可多次下单，每个订单最多属于一个注册旅客（原则上旅客不注册为系统用户也可下单，只要保存联系人姓名和手机号码）；

2、每个订单中支付时可选择一种或多种方式支付（混合支付，例如系统本地账户余额不足时，可消费完本地账户余额后由其他类型的可使用账号支付补足），每条支付信息能且只能描述其支付于某一订单；

3、每个订单对应完整出行旅途中包括一至多趟有先后次序的行程，例如订单号为 1101200604190122 的订单中去程为北京-上海-广州，那么其包含序号为 1 的北京-上海，序号为 2 的上海-广州的 2 个行程。多个不同的订单可同时包含同一行程，例如订单号为 1101200604190122 的订单中包含航班号 CA821 出发时间 201606180930 去程北京-上海-广州，而订单号为 1101200604190133 的订单中包含航班号 CA821 出发时间 201606180930 去程出发北京-上海，那么这两个订单将同时对应着航班号 CA821 出发时间 201606180930 北京-上海的行程；

4、每个订单最终预订到一或多张（行程列表一样，不一样拆单）机票，每张机票被预定有有且只存在某一订单下单中；

5、每个订单在下单后都将被分配由某一客服(经手人)负责，每个订单同时最多只能由 1 位客服负责，而每个客服同时可负责多个订单。

另外目前国内机票一个票号最多包含 2 个航段(每一航段包含可 1 至 2 个行程，去程和返程各为 1 个航段，中转行程与其他行程合并为 1 个航段)，国际机票一个票号最多包含 4 个航段（包含境外航段或港澳台航段整张机票视为国际票）。而且行程

中的缺口在票号中也是占一个航段的。具体参照实体行程的描述。

### 4.2.2 逻辑结构设计

前面围绕着订单实体讨论了机票预订模块各实体的关系，下面将对系统比较核心的一些实体的具体逻辑结构进行详细说明：

1、订单表。订单编号作为唯一标识每一订单的主键，由年月日时分秒 12 位+4 位随机数总共 16 位数字组成。联系人姓名和联系人手机号码组成联系人信息，方便订单异常时联系下单旅客。旅客订座记录 PNR 标识着该旅客下单预订机票的信息，供用户查询预订记录。订单的状态字段 STATUS 取值 unpaid、cancelled、to_refund、refunded、to_reschedule、rescheduled、to_endorse、endorsed 和 finished，分别对应待支付的、已取消的、申请退票的、已退票的、申请改期的、已改期的、申请转签的、已转签的以及已完成的。

表 4-1 订单表的逻辑结构

| 序号 | 字段名 | 数据类型 | 允许为空 | 默认值 | 字段说明 |
|---|---|---|---|---|---|
| 1 | NO | NUMBER(20,0) | | | 订单编号 |
| 2 | CSTM_ID | NUMBER(11,0) | 是 | | 所属旅客 ID |
| 3 | ORDER_DATE | DATE | | | 下单时间 |
| 4 | CONTACT_NAME | VARCHAR2(32 BYTE) | | | 联系人姓名 |
| 5 | CONTACT_PHONE | CHAR(11 BYTE) | | | 联系人手机 |
| 6 | TOTAL_COST | NUMBER(6,0) | | | 总金额 |
| 7 | REMARK | VARCHAR2(600 BYTE) | | (空串) | 订单备注 |
| 8 | PNR | CHAR(6 BYTE) | | | 旅客订座记录 |

续表 4-1

| 序号 | 字段名 | 数据类型 | 允许为空 | 默认值 | 字段说明 |
|---|---|---|---|---|---|
| 9 | STATUS | VARCHAR2(24 BYTE) | | UNPAID | 订单状态 |
| 10 | HANDLER_NO | NUMBER(8 BYTE) | 是 | | 经手人员工编号 |
| 11 | LAST_MODIFIED | DATE | | | 最后操作时间 |
| 12 | IS_LOCKED | CHAR(1 BYTE) | | U | 是否已被锁定 |
| 13 | IS_DELETED | CHAR(1 BYTE) | | N | 是否已被删除 |

2、支付信息表。支付流水号为唯一标识，根据数据库序列自增。支付方式指明此次支付使用的那种类型的账号进行支付，除了使用本系统旅客账户之外还支持例如银行卡、支付宝和微信支付等，对应取值 local、bank card、alipay 和 wechat；支付时间以本系统接受返回的交易成功信息的时间为。准；支付银行流水号作为对账的依据。

表 4-2 支付信息表

| 序号 | 字段名 | 数据类型 | 允许为空 | 默认值 | 字段说明 |
|---|---|---|---|---|---|
| 1 | ID | NUMBER(22,0) | | | 支付流水号 |
| 2 | ORDER_NO | NUMBER(20,0) | | | 所属订单号 |
| 3 | MODE | VARCHAR2(24 BYTE) | | BANK CARD | 支付方式 |
| 4 | ACCOUNT | VARCHAR2(32 BYTE) | | | 支付账号 |
| 5 | AMOUNT | NUMBER(12,2) | | | 支付总金额 |
| 6 | TRANS_SEQ_NO | VARCHAR2(16 BYTE) | 是 | | 支付银行流水号 |
| 7 | PAYMENT_TIME | DATE | | | 支付时间 |
| 8 | IS_DELETED | CHAR(1 BYTE) | | N | 是否已被删除 |

3、行程信息表，指明旅客于某一天出行某一航段。航班号标识着此次行程乘坐航班的航班号，在此习惯将其拆分为“承运人代号”和“航线号”两个字段加以外键约束，因此由承运人代号 2 为字符+实际航班序号 3-4 位数字组成，末位单数表示往程复数表示返程；出发时间、出发机场代号、登机航站楼、到达时间、到达机场代号和接机航站楼说明行程的具体时间地点，供旅客临时查询。

表 4-3 行程信息表

| 序号 | 字段名 | 数据类型 | 允许为空 | 默认值 | 字段说明 |
|---|---|---|---|---|---|
| 1 | ID | NUMBER(22,0) | | | ID |
| 2 | CARRIER_CODE | CHAR(2 BYTE) | | | 承运人代号 |
| 3 | FLIGHT_NO | VARCHAR2(4 BYTE) | | | 航班号 |
| 4 | DEP_DATETIME | DATE | | | 出发时间 |
| 5 | DEP_AIRPORT_CODE | CHAR(3 BYTE) | | | 出发机场代码 |
| 6 | DEP_TERMINAL | CHAR(1 BYTE) | 是 | | 登机航站楼 |
| 7 | ARR_DATETIME | DATE | | | 到达时间 |
| 8 | ARR_AIRPORT_CODE | CHAR(3 BYTE) | | | 到达机场代码 |
| 9 | ARR_TERMINAL | CHAR(1 BYTE) | 是 | | 接机航站楼 |
| 10 | AC_CODE | CHAR(3 BYTE) | | | 机型代号 |
| 11 | MEAL_SERVICE | CHAR(1 BYTE) | | | 是否提供餐食 |
| 12 | IS_DELETED | CHAR(1 BYTE) | | N | 是否已被删除 |

4、机票信息表，客票票号由各航空公司的客票代号前缀 3 位字符+实际票号 11 位数字总共 14 位字符组成，唯一标识每一张电子机票；乘机人姓名、乘机人类型、乘机人出生日期、乘机人证件类型和乘机人证件号等属性共同构成一位旅客的基本信息，乘客类型字段 PAX_TYPE 取值 adult、child 和 infant，分别对应成人、儿童和婴儿；乘客证件类型字段 PAX_ID_TYPE 取值 ID card、passport 和 other，分别对应身份证、护照和其他类型，当乘机人类型为“婴儿”时乘机人出生日期不得为空；舱位代码指示旅客最终登机后乘坐的实际座位，由座位所在排数 1-2 两位数字+该舱位的舱位等级 1 位字母，总共 2-3 位字符组成；票面价、机场建设费和燃油附加费等根据旅客预订机票时该航班该舱位等级对应座位的价格决定，且货币单位统一为人民币，因此该字段的值不包含货币符号。机票状态字段 STATUS 取值 OPEN、HK、RR、RQ、SUSPENDED、NO、REFUNDED、USED、VOID 和 CHANGE，分别对应状态开放使用、预订成功、已出票、候补票、已被挂起，无法使用、已被航空公司取消、已退票、已使用、已作废以及已换开。

表 4-4 机票信息表

| 序号 | 字段名 | 数据类型 | 允许为空 | 默认值 | 字段说明 |
|---|---|---|---|---|---|
| 1 | ID | NUMBER(22,0) | | | 机票 ID |
| 2 | ORDER_NO | NUMBER(20,0) | | | 所属订单编号 |
| 3 | TICKET_NO | VARCHAR2(14 BYTE) | 是 | | 客票票号 |
| 4 | PAX_NAME | VARCHAR2(32 BYTE) | | | 乘机人姓名 |
| 5 | PAX_TYPE | CHAR(1 BYTE) | | ADULT | 乘机人类型 |
| 6 | PAX_BIRTHDAY | DATE | 是 | | 乘机人出生日期 |
| 7 | PAX_ID_TYPE | CHAR(1 BYTE) | | ID CARD | 乘机人证件类型 |
| 8 | PAX_ID_NO | VARCHAR2(18 BYTE) | | | 乘机人证件号 |
| 9 | SPACE_CODE | VARCHAR2(3 BYTE) | 是 | | 舱位代码 |
| 10 | TICKET_FEE | NUMBER(6,0) | | | 票价 |

续表 4-4

| 序号 | 字段名 | 数据类型 | 允许为空 | 默认值 | 字段说明 |
|---|---|---|---|---|---|
| 11 | AIRPORT_TAX | NUMBER(4,0) | | 50 | 机场建设费 |
| 12 | BAF | NUMBER(4,0) | | 0 | 燃油附加费 |
| 13 | STATUS | VARCHAR2(4 BYTE) | | | 机票状态 |
| 14 | REMARK | VARCHAR2(600 BYTE) | | (空串) | 备注 |
| 15 | LAST_MODIFIED | TIMESTAMP | | | 最后操作时间 |
| 16 | IS_LOCKED | CHAR(1 BYTE) | | U | 是否已被锁定 |
| 17 | IS_DELETED | CHAR(1 BYTE) | | N | 是否已被删除 |

5、订单-旅程表，主要用于描述实体订单与行程之间多对多的对应关系，订单编号、行程 ID 和订单内行程序号作为联合主键唯一标识一个对应关系。

表 4-5 订单-旅程表

| 序号 | 字段名 | 数据类型 | 允许为空 | 默认值 | 字段说明 |
|---|---|---|---|---|---|
| 1 | ORDER_NO | NUMBER(22,0) | | | 订单 ID |
| 2 | VOYAGE_ID | NUMBER(22,0) | | | 行程 ID |
| 3 | VOYAGE_ORDER | NUMBER(1,0) | | 1 | 订单内行程序号 |
| 4 | VOYAGE_TYPE | VARCHAR2(24 BYTE) | | | 去程、返程、中转 |
| 5 | IS_DELETED | CHAR(1 BYTE) | | N | 是否已被删除 |

6、航班时刻表。航班时刻表主要用于航班查询中的初步查询，航空公司及所属航班编号构成航段的航班号，与出发机场代码(三字码)、抵达机场代码及出发时间组成联合唯一键唯一标识某一航段信息。出发机场代码、登机航站楼和出发时间描述旅客登机时的具体时间地点。抵达机场代码、接机航站楼、抵达时间描述旅客抵达时的具体时间地点。机型代码与机型表相关联指示此次飞行乘坐的飞机机型，提供舱位等级描述了次航段中所有可能提供的舱位的等级，经停情况指示次航段飞行中飞机中途由于需要加油等进行短时间内经停的所有机场，有先后顺序。某些航空公司为促进销量可能会将两个航段合并成一个，并降低总价格一次来吸引旅客，因此若该航段为合并航段时还需加上中转航班号(航空公司代号与主航段一致)、中转机场三字码、中转机场抵达时间、中转机场接机航站楼、中转机场出发时间、中转机场登机航站楼、中转航段经停情况、中转班机机型和中转舱位等级等来表述中转航段的信息。总时长和总航程描述了此次飞行花的时间和飞行的总距离，所属航班班期指示该航段在一周七天内哪几天是否有飞行，有则以数字 1234567 对应周日至周一表示，没有则以‘.’表示。航班有效期表示年度该航段从哪天起生效至哪天为止无效，配合班期可指明本年度中具体某一天是否飞行。

表 5-1 航段信息表

| 序号 | 字段名 | 数据类型 | 允许为空 | 默认值 | 字段说明 |
|---|---|---|---|---|---|
| 1 | ID | NUMBER(12,0) | | | 航段 ID |
| 2 | CARRIER_CODE | CHAR(2 BYTE) | | | 航空公司代号 |
| 3 | FLIGHT_NO | VARCHAR2(4 BYTE) | | | 所属航班编号 |
| 4 | DEP_AIRPORT_CODE | CHAR(3 BYTE) | | | 出发机场代码 |
| 5 | DEP_TERMINAL | CHAR(1 BYTE) | | 1 | 登机航站楼 |
| 6 | DEP_TIME | CHAR(4 BYTE) | | | 出发时间 |
| 7 | AC_CODE | CHAR(3 BYTE) | | | 机型代码 |
| 8 | CLS_CODES | VARCHAR2(3 BYTE) | | | 提供舱位等级 |

续表 5-1

| 序号 | 字段名 | 数据类型 | 允许为空 | 默认值 | 字段说明 |
|---|---|---|---|---|---|
| 9 | STOPS | VARCHAR2(18 BYTE) | | 0 | 经停情况 |
| 10 | MEAL_SERVICE | CHAR(1 BYTE) | | M | 餐食标志 |
| 11 | ARR_TIME | CHAR(5 BYTE) | | | 抵达时间 |
| 12 | ARR_AIRPORT_CODE | CHAR(3 BYTE) | | | 抵达机场代码 |
| 13 | ARR_TERMINAL | CHAR(1 BYTE) | | 1 | 接机航站楼 |
| 14 | TRA_FLIGHT_NO | VARCHAR2(4 BYTE) | 是 | | 中转航班号 |
| 15 | TRA_AIRPORT_CODE | CHAR(3 BYTE) | 是 | | 中转机场三字码 |
| 16 | TRA_ARR_TIME | CHAR(5 BYTE) | 是 | | 中转抵达时间 |
| 17 | TRA_ARR_TERMINAL | CHAR(1 BYTE) | 是 | | 中转接机航站楼 |
| 18 | TRA_DEP_TIME | CHAR(5 BYTE) | 是 | | 中转出发时间 |
| 19 | TRA_DEP_TERMINAL | CHAR(1 BYTE) | 是 | | 中转登机航站楼 |
| 20 | TRA_STOPS | VARCHAR2(18 BYTE) | 是 | | 中转航段经停 |
| 21 | TRA_AC_CODE | CHAR(3 BYTE) | 是 | | 中转班机机型 |
| 22 | TRA_CL_CODES | VARCHAR2(3 BYTE) | 是 | | 中转舱位等级 |
| 23 | DURATION | CHAR(4 BYTE) | | | 总时长 |
| 24 | FLIGHT_SHORT | NUMBER(5,1) | | | 总航程 |
| 25 | DAYS_OF_OPERATION | CHAR(7 BYTE) | | .23456. | 所属航班班期 |
| 26 | VALIDITY | CHAR(16 BYTE) | | | 航班有效期 |

7、运价缓存表。航空公司代号和所属航班编号组合成完整的航班号，和出发机场代码、抵达机场代码以及出发日期等组成联合主键唯一标识具体哪家航空公司的哪个航班哪天的运价。票面价描述了以全价经济舱 100%的价格，因统一价格单位为 RMB 该字段的值不包括货币符号。更新时间戳用于描述此运价缓存记录的更新时间，结合系统当前时间可判断其是否过期。

表 5-2 运价缓存表

| 序号 | 字段名 | 数据类型 | 允许为空 | 默认值 | 字段说明 |
|---|---|---|---|---|---|
| 1 | CARRIER_CODE | CHAR(2 BYTE) | | | 航空公司代号 |
| 2 | FLIGHT_NO | VARCHAR2(4 BYTE) | | | 所属航班编号 |
| 3 | DEP_AIRPORT_CODE | CHAR(3 BYTE) | | | 出发机场代码 |
| 4 | ARR_AIRPORT_CODE | CHAR(3 BYTE) | | | 抵达机场代码 |
| 5 | DEP_DATE | VARCHAR2(18 BYTE) | | | 出发日期 |
| 6 | TICKET_FEE | CHAR(32 BYTE) | | | 票面价 |
| 7 | AIRPORT_TAX | NUMBER(4,2) | | 50 | 机场建设费 |
| 8 | BAF | NUMBER(4,2) | | 0 | 燃油附加费 |
| 9 | UPDATE_TIME | | | | 更新时间戳 |

8、操作日志记录了系统角色对大部分表的具体操作信息，其中操作者 ID 可以是旅客实体的 ID 也可以是员工实体的 ID，若操作者为某员工实体则操作者类型为其角色对应的字符串，否则为“customer”；操作时间记录了操作的具体日期和时间已备审查；所操作表名称指明了被操作的表；操作语句中的条件为记录的(联合)主键，用来定位到具体被操作的哪条记录；操作类型包括插入记录、修改记录和删除(非物理删除)记录，查询操作不做日志处理；操作细节记录了操作的具体字段内容变化前后的值，方便审查和恢复。

表 4-6 操作日志表

| 序号 | 字段名 | 数据类型 | 允许为空 | 默认值 | 字段说明 |
|---|---|---|---|---|---|
| 1 | ID | NUMBER(38,0) | | | 日志设置 ID |
| 2 | OPERATOR_ID | NUMBER(11,0) | | | 操作者的 ID |
| 3 | OPERATOR_TYPE | CHAR(24 BYTE) | | CUSTOMER | 操作者类型 |
| 4 | OPERATING_TIME | DATE | | | 操作日期时间 |
| 5 | TABLE_NAME | VARCHAR2(30 BYTE) | | | 所操作表名称 |
| 6 | TABLE_PK_VALUE | VARCHAR2(200 BYTE) | | | 操作条件 |
| 7 | TYPE | CHAR(1 BYTE) | | INSERT | 操作类型 |
| 8 | DETAILS | VARCHAR2(4000 BYTE) | | | 具体操作细节 |

# 5. 主要功能模块的设计与实现

本章着重介绍系统几个核心的功能模块的设计与实现，包括系统在安全模块对用户登录认证的基本设计思路以及基础数据管理模块，最后详细介绍一下航班查询与机票预订模块。

## 5.1 用户安全认证的设计

保障系统安全一直是一个很重要的课题，在此将以系统中用户登录认证为例介绍系统如何配合其他模块来保障系统中用户账户的安全。

通常，保障用户浏览器与服务器之间信道安全的方式是通过 https 证书对传输层的数据进行加密处理。然而这也意味着应用需要拥有一个确定的域名，还需要向可信的 CA 机构进行申请，并周期性地支付一定的费用。对于一个运营的系统来讲在安全方面进行投入确实是值得考虑的，但在系统刚刚起步或者还未形成稳定收益用于支撑该部分投入的情况下，也可以考虑采用软件的方式进行处理。

下面不妨对保障客户端浏览器与服务端之间信道传输安全进行逐步的推演，来说明系统在账户信息安全保护方面所做的工作。

1、用户在浏览器的 web 页面输入账户登录信息（包括 username 和 password），此时存在一个中间人（试图获取传输的敏感信息，已达到其非法目的的入侵者）在网络上侦听传输消息。倘若中间人在截取了用户在浏览器与服务器之间的消息后成功得到了用户的登录信息，那此时无论系统数据库中采用明文还是加密后的字符串来存储密码都无法实现对账户信息的保护。

2、先用哈希算法对密码进行摘要处理然后再传输。此时在浏览器与服务器之间传输的是用户名和经过哈希处理的密码（username，hash（password））。中间人截取到用户的登录信息后，虽无法得到用户的明文密码但其可以伪装成合法用户直接使用用户名连同经哈希算法处理后的密码提交到后台进行认证，由于服务器的处理是基于无状态的 HTTP 协议，无法得知此时提交数据的是否为真正的合法用户，因此

中间人依然可以成功地以合法用户的身份进入系统。

3、考虑进行登录信息提交前在用户的密码中混入随机盐（此时需要在数据库中旅客表中添加一个字段用户记录该随机盐），这样一来旅客在提交登录信息进行认证前需先从服务器获取账号 username 对应的随机盐 salt1，之后将密码进行哈希处理得到 H1=hash(password)，再混入该随机盐 salt1 进行哈希处理得到 H2=hash(salt+H1)，此时向服务器提交的登录信息为(username,H2)，假设中间人通过网络侦听获取登录信息并得到了 H2。服务端接受到登录验证的请求后，根据登录信息生成一个 token(username,H2)，接着根据 token 中的 username 找到数据库中对应旅客账户信息记录并生成相应的认证信息 authcInfo(userid,hash(password),salt1)，再交由相应的密码匹配器对 token 中的 H2 与 authcInfo 中进行处理后得到的 H3=hash(salt1+hash(password))进行匹配。由于 H3 =hash(salt1+hash(password) )=H2 因此匹配成功认证通过，一旦认证通过，在进行相应的授权后系统将重新生成随机盐 salt2 来覆盖数据库中 salt1 的值，下次请求时获取的和认证时 authcInfo 中用于匹配的随机盐都将一致变更为 salt2。

当中间人再次以(username，H2)的登录信息提交进行验证时，服务器依然生成相应的 token(username,H2) 以及对应的 authcInfo(userid,hash(password),salt2)，此时 authcInfo 中的随机盐已由 salt1 变为 salt2，因此 H4= hash(salt2+hash(password) )!= (salt1+hash(password) )=H2，最终匹配不成功认证失败，中间人得不到相应的授权。至此中间人无法通过直接提交传输过程中截取的登录信息来伪装成旅客进入系统。

4、以上虽然做到了不明文传输且防止了重复认证，但一旦数据库沦陷攻击者可以很轻松地对旅客账户信息表中 hash(password)字段的值进行破译从而得到 password，此时攻击者在浏览器以正常旅客的登录方式输入 password 仍然可以伪装成旅客进入系统。这时可考虑加大其破译难度，由于通常无法很容易地实现从 hash(password)到 hash(salt+password)的计算，而之前存储在数据库中的密码只是进行了哈希处理却没有加盐，这时可以在存入数据库之前对其进行不加盐的哈希处理后再混入固定随机盐再次哈希处理，当然数据库中需要加入相应的字段来存储该静态随机盐。同之间的动态随机盐一样，该静态随机盐在旅客注册账户时生成，且旅客请求登录认证前也需获取该随机盐，只不过在之后的处理过程中该静态随机盐都不再发生改变。这样一来，数据库存储的密码字段将包括 salt0，H0=hash(salt0+hash(password))和 salt1，客

户端浏览器与服务端传输的数据将为(username, H1=hash(salt1+hash(salt0+hash(password)))) ，密码匹配时将使用H2=hash(salt1+H0)与 H1 进行匹配。

整个登录认证过程的时序图如下：

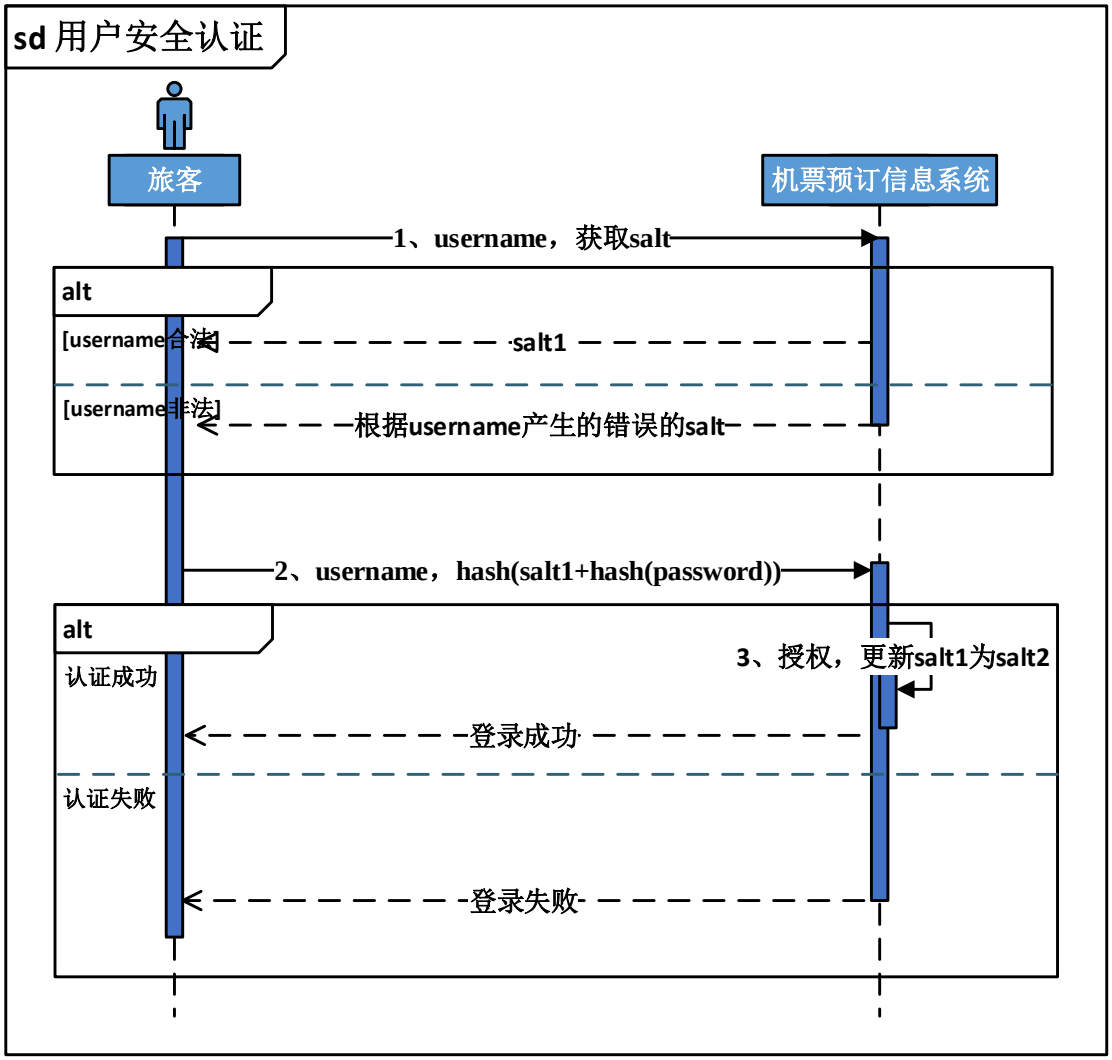


图 5-1 用户安全认证的时序图

下面简单介绍一下以代码实现的步骤：

1、请求认证前获取盐值，代码如图 5-2：

```
116         if (customer != null) {
117             //如果用户名对应用户存在则返回数据库中该用户账户对应的随机盐，
118             staticSalt = customer.getStaticSalt();
119             dynamicSalt = customer.getDynamicSalt();
120         }else{
121             //否则返回新生成的随机盐
122             staticSalt = RandmSaltGenerator.nextSalt();
123             dynamicSalt = RandmSaltGenerator.nextSalt();
124         }
125 
126         result.put("status", "succeed");
127         result.put("staticSalt", staticSalt);
128         result.put("dynamicSalt", dynamicSalt);
129         return result;
130     }
```

图 5-2 接受登录认证请求时返回随机盐的代码块

2、浏览器取得盐后通过 javascript 脚本对密码进行 hash 处理后再提交，核心代码如图 5-3：

```
136 if(data.status == "succeed"){
137     var staticSalt = data.staticSalt;
138     var dynamicSalt = data.dynamicSalt;
139     //对密码根据混入的随机盐进行运算
140     var password = $.md5(dynamicSalt+$.md5(staticSalt+$.md5($("#password").val())))
141     var captcha = $("#captchaToLogin").val();
142     var keepLoggedIn = $("#keepLoggedIn").is(':checked') ;
143     var urlToDoLogin = "<%=contextPath %>/login.do";
144     var paramsToDoLogin = {
145         username : username
146         , password : password
147         , captcha : captcha
148         , keepLoggedIn : keepLoggedIn
149     };
150     //将混入随机盐的密码与其他登录信息一起提交后台验证
151     $.post(
152         urlToDoLogin,
153         paramsToDoLogin,
```

图 5-3 js 脚本处理登录密码的代码块

3、后台接受到验证请求后用提交的登录名和密码构建 token，如图 5-3：

```
143         Subject currentUser = SecurityUtils.getSubject();
144 
145         String username = logininfo.getUsername();
146         String password = logininfo.getPassword();
147         String captcha = logininfo.getCaptcha();
148         //用提交的用户名和密码生成token
149         CustomUsernamePasswordTocken token
150         = new CustomUsernamePasswordTocken(username, password, captcha);
151 
152         try {
153 
154             currentUser.login(token);    //认证
155 
```

图 5-4 构建 token 的代码块

4、根据 token 中的登录名查找用户储存在数据库的密码和盐，并根据它们构建认证信息，如图 5-5 和图 5-6；

```
64  //取token中的用户名并尝试查找用户信息
65  String username = (String)token.getPrincipal();
66  Customer customer = null;
67  if ( RegexpValidator.isPhone(username) ) {
68      customer = customerService.findByPhone(username);
69  }
70  else if ( RegexpValidator.isEmail(username) ) {
71      customer = customerService.findByEmail(username);
72  }
73  else {                          //登录名格式错误
74      throw new IllegalArgumentException("Illegal  username[" + username+ "]");
75  }
```

图 5-5 根据 token 中登录名查找对应用户的代码块

```
87  //若找到该用户则，该用户的id、密码和动态盐构建相应的认证信息
88  String principal = String.valueOf(customer.getId());
89  String credentials = customer.getPassword();
90  ByteSource dynamicSalt = ByteSource.Util.bytes(customer.getDynamicSalt());
91
92  SimpleAuthenticationInfo authcInfo
93  = new SimpleAuthenticationInfo(principal,credentials,dynamicSalt,getName());
94
95  //接下来由凭证匹配器匹配
96  return authcInfo;
```

图 5-6 取用户 id、密码和动态盐构建认证信息的代码块

5、由密码匹配器对认证信息中的密码进行 hash 处理后，与 token 中的密码进行匹配，如图 5-7 和 5-8：

```
163 @Override
164 public boolean doCredentialsMatch(AuthenticationToken token, AuthenticationInfo info)
165 {
166     Object tokenCredentials = token.getCredentials();
167     Object hashedAccountCredentials = hashAccountCredentials(info);
168
169     //将token里的密码与 authcInfo中hash处理后的结果进行匹配
170     return equals(tokenCredentials, hashedAccountCredentials);
171 }
```

图 5-7 认证匹配规则的核心代码块

```
191 protected Object hashAccountCredentials(AuthenticationInfo info) {
192
193     Object salt = null;
194     if (info instanceof SaltedAuthenticationInfo) {
195         salt = ((SaltedAuthenticationInfo) info).getCredentialsSalt();
196     }
197
198     //对 authcInfo中的密码进行加盐hash处理，盐值取之前生成authcInfo时去数据库取的动态盐，默认迭代次数1
199     Hash computed = hashAccountCredentials(info.getCredentials(), salt, getHashIterations());
200     return ByteSource.Util.bytes(computed.toHex());
201 }
```

图 5-8 匹配前对认证信息的哈希处理代码块

6、匹配正确后进行授权，认证通过，刷新数据库中当前用户对应的动态盐，具体代码如图 5-9 和 5-10：

```
99  //授权
00  @Override
01  protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principals) {
02      // TODO
03      String userID = (String)principals.getPrimaryPrincipal();
04      SimpleAuthorizationInfo authzInfo = new SimpleAuthorizationInfo();
05      authzInfo.addRole("cust");
06      authzInfo.addStringPermission("cust:*:"+userID);
07      return authzInfo;
08  }
```

图 5-9 认证匹配正确后授权的代码块

```
183         //认证成功后，刷新当前用于数据库中的动态盐
184         String dynamicSalt = RandmSaltGenerator.nextSalt();
185         long custId = Long.parseLong(currentUser.getPrincipal().toString());
186         Customer cust = customerService.findByID(custId);
187         cust.setDynamicSalt(dynamicSalt);
188         customerService.update(cust);
189
```

图 5-10 认证通过刷新当前用户于数据库中动态盐的代码块

## 5.2 基础数据管理模块的设计与实现

基础数据模块将采用 REST 方式想外提供服务，服务消费者可通过 URL 和 HTTP 方法的组合的方式来访问这些资源。在设计时，先由对象实体映射框架将数据库记录实体映射为相应的对象，在将这些对象抽象成资源，再由 URL 对其进行描述。对于/airport/表示机场这个实体的全体对象，及对象的集合；而/airport/{code}则对应某一具体的机场，例如/airport/SHA 表示上海虹桥国际机场；而提交 HTTP 方法则对应操作对实体的操作，DELETE 对应对实体的删除、GET 对应对具体实体的查询、PUT 对应对某一实体的修改、POST 则对应具体实体的添加。

系统前端控制器采用 Spring MVC 来实现，利用其对 RESTful 风格的支持，使用@RestController 对控制器类进行注解便可轻松完成，如图 5-11。

```
@RestController
@RequestMapping("/airport")
public class AirportRestController {

    @Resource
    IAirportService airportService;

    @RequestMapping(value="/", method={RequestMethod.GET})
    @ResponseStatus(HttpStatus.OK)
    public List<Airport> getAll() {
        return airportService.findAll();
    }
```

图 5-11 Controller 示例

通过上面例子中由于该资源在系统全局中普通游客的身份便可查询，因此无需进行任何权限方面的认证，从浏览器输入相应的 URL 即可获得如图 5-12 的结果：


```
[{"code":"AAT","cityName":"阿勒泰","cityNameInitial":"A","airportName":"阿勒泰机场","dist":null,"isHot":"n"},
{"code":"YIE","cityName":"阿尔山","cityNameInitial":"A","airportName":"阿尔山伊尔施机场","dist":null,"isHot":"n"},
{"code":"AKA","cityName":"安康","cityNameInitial":"A","airportName":"安康机场","dist":null,"isHot":"n"},
{"code":"AKU","cityName":"阿克苏","cityNameInitial":"A","airportName":"阿克苏机场","dist":null,"isHot":"n"},
{"code":"AOG","cityName":"鞍山","cityNameInitial":"A","airportName":"鞍山机场","dist":null,"isHot":"n"},
{"code":"AQG","cityName":"安庆","cityNameInitial":"A","airportName":"安庆天柱山机场","dist":null,"isHot":"n"},
{"code":"AVA","cityName":"安顺","cityNameInitial":"A","airportName":"贵州安顺黄果树机场","dist":null,"isHot":"n"},
{"code":"NGQ","cityName":"阿里","cityNameInitial":"A","airportName":"阿里昆莎机场","dist":null,"isHot":"n"},
{"code":"AEB","cityName":"百色","cityNameInitial":"B","airportName":"百色田阳机场","dist":null,"isHot":"n"},
{"code":"BAV","cityName":"包头","cityNameInitial":"B","airportName":"包头二里半机场","dist":null,"isHot":"n"},
{"code":"BHY","cityName":"北海","cityNameInitial":"B","airportName":"北海福城机场","dist":null,"isHot":"n"},
{"code":"BPL","cityName":"博乐","cityNameInitial":"B","airportName":"新疆博乐机场","dist":null,"isHot":"n"},
{"code":"BSD","cityName":"保山","cityNameInitial":"B","airportName":"保山云瑞机场","dist":null,"isHot":"n"},
{"code":"BJS","cityName":"北京","cityNameInitial":"B","airportName":"首都机场","dist":null,"isHot":"y"},
```


图 5-12 从浏览器获取所有机场资源

## 5.3 航班查询与机票预订模块的设计

航班查询与机票预订模块是系统的核心功能模块，这两个模块的功能是完成机票预订的整个流程。由于整个流程相对复杂，这里将分为查询和预订两个部分来分别介绍。

### 5.3.1 航班查询部分的流程设计

旅客在进入系统首页即可在航班查询标签页进行航班查询，基本流程如图 5-13：

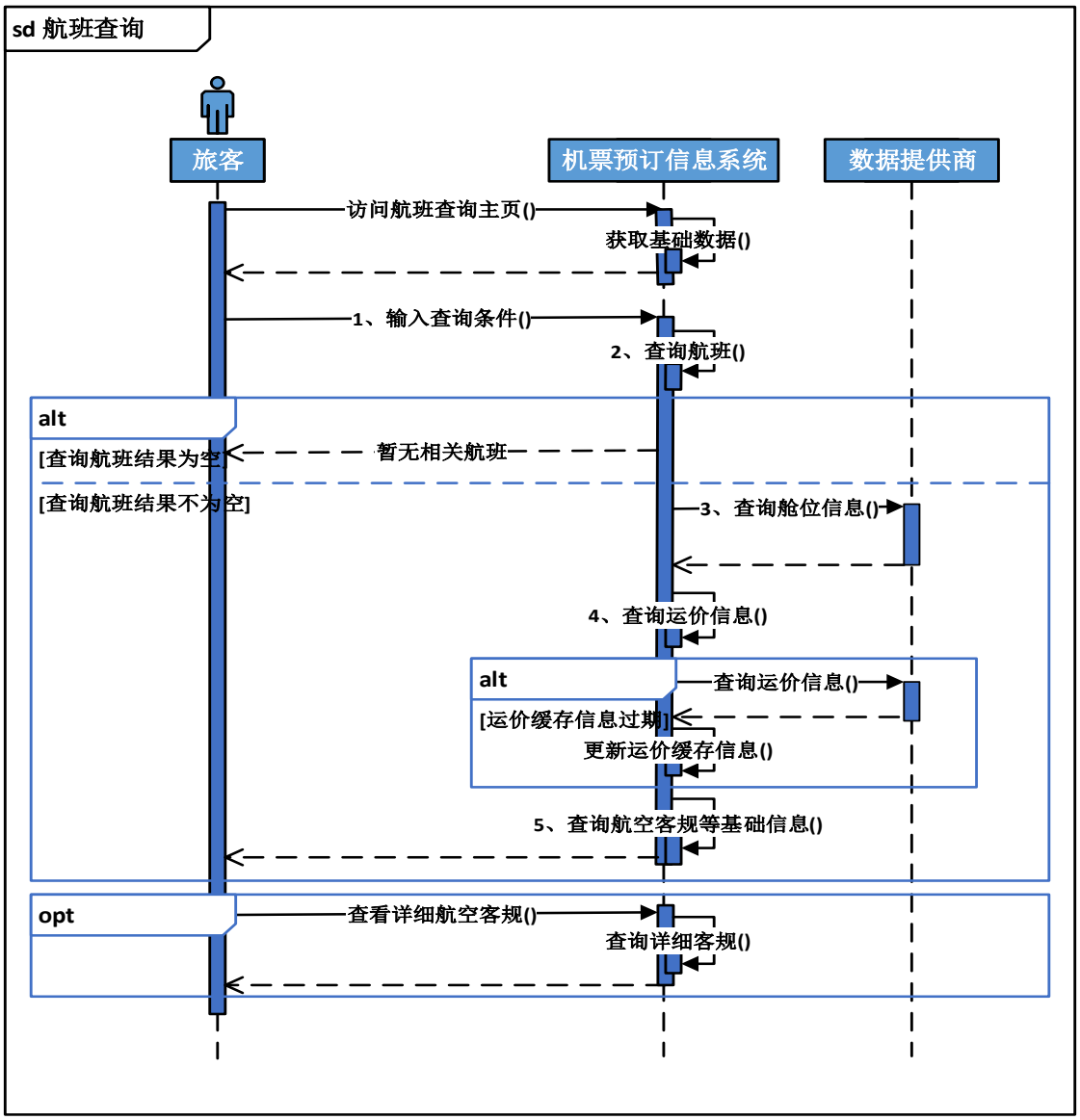


图 5-13 航班查询的时序图

航班查询的基本流程如下：

1、进入系统首页后，选择国内航班(默认)或国际航班后旅客可选择单程、返程等航班类别(国内机票最多 2 程，国际机票最多 4 程，因此选择国内航班可以有‘联程’类别的航班可选择，选择国际航班则有含缺口程的‘多程’类别航班选择)，然后单击出发城市和抵达城市的文本框即可选择相应城市，当然也可以输入城市的拼音首字母或机场三字码来选择，接着旅客还需指定出发的日期即可进行提交查询数据进行航班查询。如果有需要旅客还可以单击左下角的‘高级查询’来进行航班的过滤，可对航班出发时间段、舱位服务等级、承运人以及最少剩余票数等进行限制来过滤掉不想被查看的航班，如图 5-14 所示。如果查询不到符合旅客输入查询条件的航班，则页面将提示暂无相应的航班。如果存在满足查询条件的航班，则将查询到的航班列表数据通过 ajax 的方式传回查询页面并展示出来。


图 5-14 航班查询条件输入页面

2、客户端浏览器提交航班查询请求后，系统将根据提交的查询信息到数据库查询航班时刻表并返回所有满足查询条件的航班。若查询结果为空则直接通知旅客暂无相关航班信息，否则航班将查询舱位的信息。

3、根据前面查出来的航班列表中各个航班的航班号、出发城市、到达城市、出发日期以及起飞时间等信息由数据提供商提供的服务接口想数据提供商发起舱位信息查询，并将查询的结果合并到对应航班信息中。若旅客的查询条件中包括最少剩余票数的要求则还需对合并后的结果进行进一步过滤，否则将进行运价信息查询。

4、由于运价信息是固定周期性更新的，对其进行缓存并周期性更新可提高服务器查询效率，减缓数据提供商系统的压力，因此系统会先查询数据库的运价缓存。如果运价缓存已经过期，则需先根据数据提供商提供的查询接口向数据提供商查询运价信息，然后更新到数据库中并重置过期时间。将查询结果合并到前面经过过滤后的航班信息结果中。

5、最后，经过以上的整合后所得的数据还不是最终要展示在浏览器的，因此需要查询基础数据，并整合到上面的航班信息中去。最终将整合后的结果返回前台。结果如图 5-15 所示：


图 5-15 航班查询列表

### 5.3.2 机票预订部分的流程设计

旅客在完成航班的查询后，选择需要预定的航班并单击预定按钮即可进入预定状态。下面将介绍其基本流程，如图 5-16：

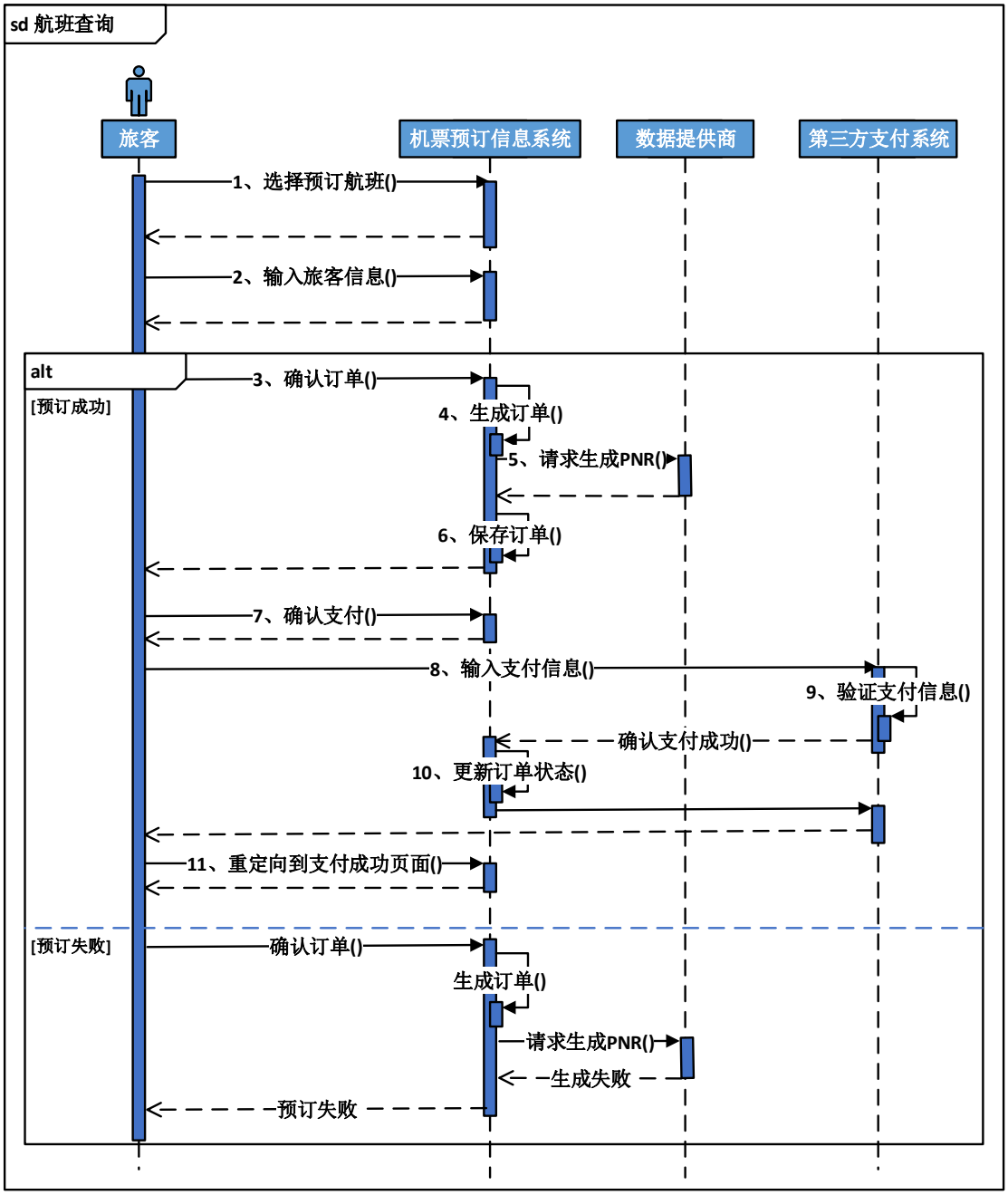


图 5-16 机票预订时序图

如上图，机票预订的基本流程如下：

1、 旅客完成航班查询后选定需要进行预订的航班之后系统将返回相应的界面，并提示旅客填入联系人信息以及乘机人信息等必要的旅客信息，如图 5-17 所示。


图 5-17 用户信息输入页面

2、 旅客完成相应信息的填写后单击下一步按钮后，系统将生成相应的汇总页面，并提示旅客进行订单确认。

3、 旅客在确认页面再次确认自己所选择的航班信息、票面价、订单总价格、联系人以及乘机人等信息后点击确认按钮，系统将进入订单生成阶段。

4、 系统将从旅客提交的信息中提取出航程、乘机人、价格等信息并根据它们生成相应的订单。

5、 订单生成后，系统将从提交的预订信息中提取出航程信息、乘机人信息后，组成数据提供商要求的格式后，使用这些数据调用数据提供商提供的服务接口请求生成旅客订座记录 PNR。若 PNR 生成成功，则进入订单保存阶段，否则将直接通知旅客预订失败。

6、 系统成功获取 PNR 后，将 PNR 保存到相应的订单中，并将订单信息保存到数据库中，成功保存订单后系统返回需要支付的金额等信息，并提示旅客进入支付阶段，如图 5-18：

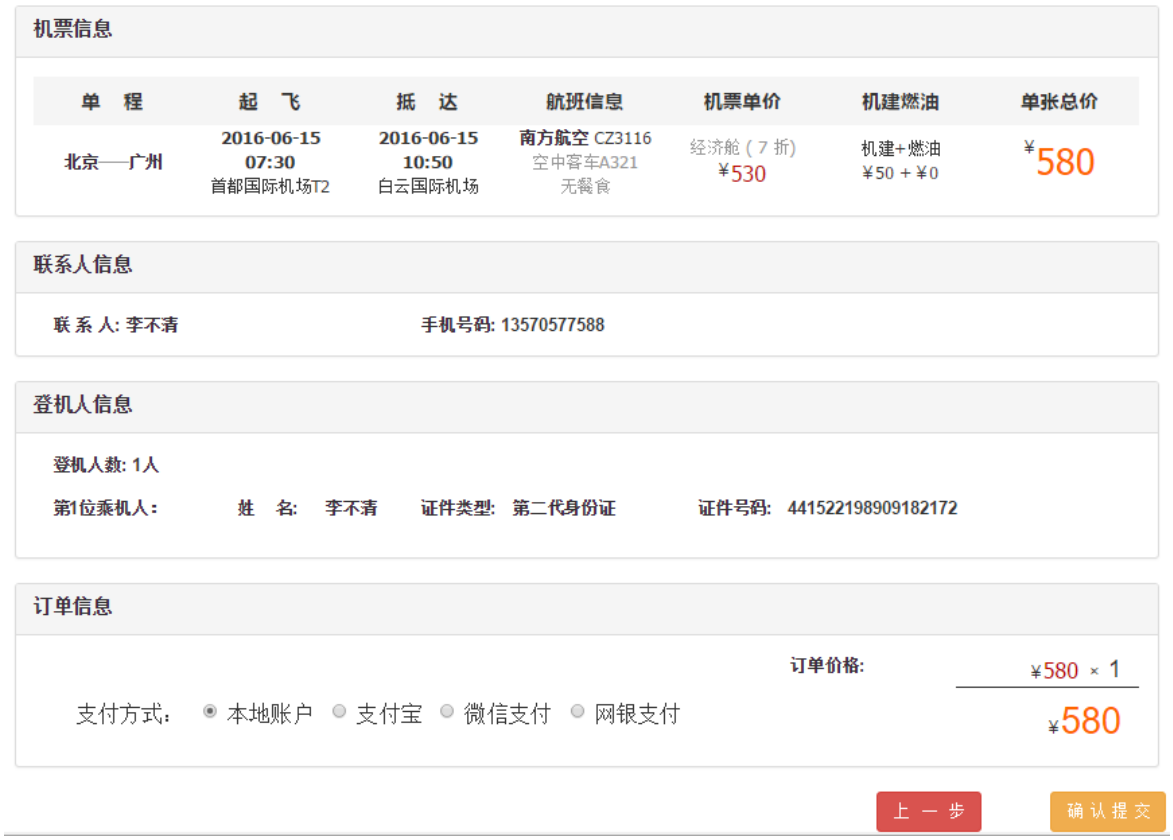


图 5-18 订单确认页面

7、 旅客确认支付金额等信息后确认支付，系统将页面链接到相应的第三方支付平台。

8、 旅客在支付页面输入银行账号、密码等支付信息后，提交第三方支付平台。

9、 第三方支付平台接受旅客输入的信息后将对这些信息进行验证，若验证通过银行系统在扣款将通知系统支付成功。

10、 系统接受到第三方支付平台返回的支付结果后将支付账号、支付金额以及银行流水号等信息保存到订单中，并更新订单的状态。更新完订单状态后系统将向第三方支付平台反馈订单状态改变结果。第三方支付平台得到系统反馈结果后将向旅客支付页面返回成功支付。

11、 旅客浏览器支付页面接受到第三方支付平台反馈的结果后进行重定向，将页面重定向到成功支付页面。系统将提示预订成功，并返回相关的支付信息，供旅客日后对账使用，如图 5-19。


机票预订成功
机票预订成功。请留意稍后下发的短信，以免耽搁您的行程。
订单编号： 201606100956591110
查看详情
交易金额： ¥ 580
支付流水号： 20160610095659111003

图 5-19 预订成功页面

# 6. 总结

## 6.1 论文总结

本文在深入了解分析了目前国内机票市场的发展情况下，结合自身所学知识，论述了 REST 架构风格的特性及设计原则，针对目前自身所具备的开发技术，设计了机票预订系统的架构，并阐述了如何设计其中各个组成部分与关键部分，最后论述了系统如何保障用户信息安全、基于 REST 使用 SpringMVC 作为前端控制器结合 Apache Shiro 进行权限管理对基础信息服务模块的实现、以及航班查询和机票预订模块的基本流程的设计。

具体本文的工作内容主要包括如下：

1、本文首先讨论了面向服务的基本概念与其特性，接着引入 REST 的相关内容及其基本的设计原则，然后讨论了目前比较流行的 Spring 框架在开发中的优势以及与其无缝集成且对 RESTful 架构风格支持较好的轻量级的 SpringMVC 框架，最后还介绍了在安全管理方面对 shiro 的选择和 ORM 框架 Hibernate 能为我们实现什么、给开发带来什么好处。

2、接着讨论了系统的需求分析，主要从系统的角色及其特点入手，然后在对系统的用例进行了详细的分析。

3、之后开始对系统的总体设计进行思考，从系统的各功能模块及外部系统之间的层次架构入手，讨论了系统各功能模块的设计。

4、再后来又阐述了从数据库的外部设计到概念结构设计再到逻辑结构的设计，并针对主要的实体及其对应逻辑模型进行了描述，最后给出系统总体的逻辑结构图。

5、然后讨论了系统中主要模块部分的详细设计和具体的实现，包括如何实现用户账户信息安全的保障特别是客户端浏览器与服务器之间的信道安全保护，还有结合 REST 编程风格和 Shiro 权限控制的基础数据模块的实现以及航班查询模块和机票预订模块的流程设计。

## 6.2 问题和展望

经过之前对国内航空票务市场及相关业务的调研，结合自身关于面向服务体系架构相关技术的积累，再综合系统的需求分析、各个部分的设计、编码实现和最后的测试等步骤，最后终于使得机票预订系统运行良好。然而受技术和时间约束，项目完善程度在刚完成初期恐怕都会差强人意，即使花的时间精力再多也很难一口气完美实现。随着时间的推移，无论是在业务层面还是技术层面，很多地方都会有所变更发展，系统自然也会存在很多需要扩展和改进的地方。

1、针对机票预订这种跨地域性比较强的且查询效率要求比较高的票务需求，技术上采用分布式系统的模式架构设计会更有优势，尤其在数据库系统方面，尽管一些用于分片的字段考虑到了，但受技术支持影响目前仍未实现。

2、目前收支持条件限制，只能提供单机服务器作为后台服务器，多服务器集群能提供负载均衡带来性能上的提升。

3、目前系统对旅客订票成功后的售后服务显然不是特别完善、诸如退票、改期、签转、废票、换开和升舱等业务的支持有待扩展，虽原则上保持不介入旅客与航空公司及代理人之间的负责关系，但至少应提供相关的渠道让旅客更好地沟通与他们沟通，从而提高旅客对本系统的满意度。

4、系统业务功能目前还有很大的扩展空间。现代生活中随着经济水平的提高，旅游、差旅等也都在蓬勃发展，人们对机票、酒店和快递等活动的需求通常都存在着一定的关联性。就想商业巨头们都忙着建立旗下的生态圈一样，除了机票以外，酒店、旅游门票、甚至婚礼举办等产品都可以继承到一个更大的商业平台，通过进行各种产品的组合来使用户在一个平台消费便可满足日常生活的各种需求，达到培养用户对平台的依赖性甚至是其消费习惯的目的，可考虑扩展相关业务完成“一站式服务”。

# 参考文献

[1] 张屹,吴向荣. 企业级 Java Web 编程技术——Servlet&JSP[M]. 大连:大连理工大学出版社,2012.7. p270-p286

[2] 张屹,蔡木生. Java 核心编程技术[M]. 大连: 大连理工大学出版社,2010.9. p70-p91

[3] 陈志泊,王春玲. 数据库原理及应用教程(第二版)[M]. 北京: 人民邮电出版社,2008.3. p18-p23

[4] 杨永健,刘尚毅. Oracle 数据库管理、开发与实践[M]. 北京: 人民邮电出版社,2012.12. p10-p31

[5] Guy Harrison. Oracle 性能优化求生指南[M]. 北京: 人民邮电出版社,2012.9. p11-p361

[6] 计文柯. Spring 技术内幕：深入解析 Spring 架构与设计原理[M]. 北京: 机械工业出版社,2011.12. p17-p189

[7] Seth Ladd. 深入解析 Spring MVC 与 Web Flow[M]. 北京: 人民邮电出版社,2008.11. p18-198

[8] 孙卫琴. 精通 Hibernate：Java 对象持久化技术详解(第二版) [M]. 北京:电子工业出版社 ,2010. p21-p491

[9] 秦小波. 设计模式之禅[M]. 机械工业出版社,2013.10. p384-p437

[10] 詹舒波. WAP—移动互联网解决方案[M]. 北京： 北京邮电大学出版社, 2000.6. p10-p28

# 致 谢

在本课题的设计、实现和论文阶段，我得到了很多人的帮助，在此谨对帮助本课题以及对我有帮助和关心我的老师、同学以及我的家人致以最由衷的感谢。

首先我要感谢我的导师吴向荣老师，吴老师无论是严谨的治学态度还是渊博的学识都使得其成为我的榜样。吴老师对我的项目和论文都给予了我耐心的知道和帮助，并提出了许多宝贵的意见。从吴老师身上我不仅学到了扎实的专业知识，还学到了做人的道理。在此我要向吴老师致以诚挚的谢意。

其次，我要感谢大学四年中给我影响的老师们，是他们教会我如何正确地对待学习，他们对只是的渴望和追求，对教学的一丝不苟深深影响着我。在学习道路上是他们给我传道授业解惑才有了我的成长。我还要感谢班级的同学，他们的努力和刻苦精神以及对知识的渴望同样深深影响着我，使我处在一个良好的学习氛围中。

我还要感谢我的家人，有了家人的支持和鼓励我才能顺利地完成本科阶段的学习，并使我的在生活上没有后顾之忧。

在此，向所有关心和帮助过我的老师、同学、家人以及其他朋友们表示衷心的感谢！同时也衷心地感谢各位专家和老师们在百忙之中评阅本文和指导答辩！