**工程硕士学位论文**

机票退改签系统的设计与实现

**THE DESIGN AND IMPLEMENTATION**

**OF TICKET REFUND AND ENDORSE SYSTEM**

**王文娟**

**哈尔滨工业大学**

**2015 年 06 月**

国内图书分类号：TP311 学校代码：10213

国际图书分类号：621.3 密级：公开

**工程硕士学位论文**

机票退改签系统的设计与实现

|  |  |
| --- | --- |
| 硕 士 研 究 Th | ：王文娟 |
| 导 师 | ：王宏志 副教授 |
| 副 导 师 | ：李文芳 高级工程师 |
| 申 请 学 位 | ：工程硕士 |
| 学 科 、 专 业 | ：软件工程 |
| 所 在 单 位 | ：软件学院 |
| 答 辩 日 期 | ：2015 年 06 月 |
| 授予学位单位 | ：哈尔滨工业大学 |

Classified Index: TP311 U.D.C.: 621.3

Dissertation for the Master Degree in Engineering

**THE DESIGN AND IMPLEMENTATION**

**OF TICKET REFUND AND ENDORSE SYSTEM**

|  |  |
| --- | --- |
| **Candidate: Supervisor: Associate Supervisor:**  **Academic Degree Applied for: Speciality:**  **Affiliation：**  **Date of Defence:**  **Degree-Conferring-Institution:** | Wenjuan Wang A.Prof. Hongzhi Wang Engineer Wenfang Li Master of Engineering Software Engineering School of Software June, 2015  Harbin Institute of Technology |

摘 要

由于互联网的蓬勃发展，网络购票出行成为人们日常生活的常见行为。当购票后对机票进行退改签操作时，机票由航空公司和代理商多方同时控制，造成退改手续费不明确的情况。这种情况常常导致用户的经济损失。基于这种情况，本文的研究内容是如何根据机票判断机票真实情况，获取真实的航空公司和代理商的机票退改签规则，为用户进行退款改签操作。

本文主要实现机票真实票号的获取，根据票号查询机票的真实仓位，并且获取航空公司及代理商此时刻对该仓位的退改签规则，对获取的信息进行整合，计算出准确的退改签手续费，提高金融服务的准确性。同时，由于网络情况的不确定性，网络间消息传递存在延时、传输错误、丢失等多种异常情况，为保证与金额相关的信息传递能够准确且及时，本系统中采用消息队列方式进行消息传输。对于不同类型的消息进行区分，存入定时消息、延迟消息、重发消息等多个队列中，不断轮询各个队列，多方面保证消息的传递正确有效。

本文设计与实现了一次退款模块，改签模块，多次退款模块，消息队列管理模块，消息队列服务端模块，消息队列客户端模块，延迟消息模块和定时任务模块。在一次退款模块，改签模块，多次退款模块中实现对业务逻辑的控制，根据机票的不同情况判断对应机票的业务操作。在消息队列管理模块，消息队列服务端模块，消息队列客户端模块，延迟消息模块和定时任务模块中实现了对退改签消息的管理，协助业务模块运行，保障业务模块的消息传递可靠。

通过本文的对机票退改签系统和消息队列的设计与实现，用户能够准确的得到机票的准确退改签手续费金额，该系统在线上服务器中能够正常运行，很好地解决了不同退改签规则在退改操作中的相互影响问题，满足了公司需求，保护了用户的利益不受损害。

关键词：退票；改签；消息队列

**Abstract**

Due to the rapid development of Internet, to buy travel tickets from Internet becomes the people's common behavior in the daily life. When people want to refund or endorse the ticket, the fee is often wrong, because the fee prices is decided by both airlines companies and ticket agencies. This kind of situation often leads the user's economic loss. According to this situation, this paper presents a solution that can achieve the ticket refund and endorse rule of the real airlines companies and ticket agencies based on real ticket information complied with exact ticket number. With this information the user can do the refund and endorse operations by themselves.

In this paper, users can get the real ticket number, and use this number to get the real cabin, and then get the real method how to refund or endorse this ticket immediately. Afterwards this system can calculate the accurate cost, improve the accuracy of financial services by integrate these ticket information. Meanwhile, because of the uncertainty of network conditions, messaging between distributed systems always have delay, transmission error, package loss, and other anomalies. To ensure the transmission of information related to the amount can be accurately and immediately, this system uses the message queue to transmission. To distinguish different types of messages, and then store timing messages, delay messages, resend messages in different message queues. System continue visit each queue, to make sure the delivery of messages is correct and effective.

This article has designed and implemented refund module, endorse module, message queue management module, message queue server module, message queue client module, delay message module and resend message module. System can handle the business logic control and manage the messages of refund or endorse ticket .

By design and implementation the ticket refund and endorse system, systems can provide an accurate fee when the ticket change is required. Up to now, these service modules car running wall online and the system can solve the problem there were different methods to calculate the ticket price before. This system meets the company's requirements and protecting the interests of users.

**Keywords:** ticket refund; Ticket endorse; Message queue

目 录

[摘 要](#_Toc686532422) 3

**[Abstract](#_Toc686532423)** 3

[第1章 绪论](#_Toc686532424) 5

[1.1 课题来源](#_Toc686532425) 5

[1.2 项目开发的目的与意义](#_Toc686532426) 5

[1.3 与课题相关的国内外研究现状](#_Toc686532427) 5

[1.3.1 机票业服务系统情况](#_Toc686532428) 5

[1.3.2 消息队列发展情况](#_Toc686532429) 5

[1.4 本论文的主要工作内容](#_Toc686532430) 6

[1.5 本论文的组织结构](#_Toc686532431) 6

[第2章 机票退改签系统需求分析](#_Toc686532432) 6

[2.1 业务需求分析](#_Toc686532433) 6

[2.2 功能需求分析](#_Toc686532434) 6

[2.2.1 匹配规则需求](#_Toc686532435) 6

[2.2.2 服务平台退改操作控制需求](#_Toc686532436) 7

[2.2.3 消息队列功能需求](#_Toc686532437) 7

[2.2.4 二次退款退至乘机人银行卡功能需求](#_Toc686532438) 7

[2.2.5 可信用户立即退款需求](#_Toc686532439) 8

[2.4 本章小结](#_Toc686532440) 8

[第3章 机票退改签系统设计](#_Toc686532441) 8

[3.1 机票退改签系统总体设计](#_Toc686532442) 8

[3.2 退改签业务系统系统模块设计](#_Toc686532443) 9

[3.2.1 机票退改签计算方案](#_Toc686532444) 9

[3.2.2 机票退改签计算方案数据库设计](#_Toc686532445) 9

[3.3 消息队列子系统模块设计](#_Toc686532446) 19

[3.3.1 消息队列子系统模块功能划分](#_Toc686532447) 19

[3.3.2 消息队列子系统数据库设计](#_Toc686532448) 19

[3.4 本章小结](#_Toc686532449) 23

[第4章 机票退改签系统实现](#_Toc686532450) 23

[4.1 退改签业务系统设计与实现](#_Toc686532451) 23

[4.1.1 业务处理主要类说明](#_Toc686532452) 24

[4.1.2 乘机人可信度计算](#_Toc686532453) 29

[4.2.2 消息队列子系统性能及特点分析](#_Toc686532454) 30

[4.3主要类和接口说明](#_Toc686532455) 31

[4.3.1消息队列主要类说明](#_Toc686532456) 31

[4.3.2消息队列主要接口说明](#_Toc686532457) 32

[4.4 本章小结](#_Toc686532458) 34

[第5章 机票退改签系统运行与测试](#_Toc686532459) 34

[5.1 测试环境](#_Toc686532460) 34

[5.2 功能性测试](#_Toc686532461) 34

[5.2.3 sonar检查](#_Toc686532462) 40

[5.3 系统性能测试](#_Toc686532463) 41

[结论](#_Toc686532464) 43

[参考文献](#_Toc686532465) 43

[哈尔滨工业大学学位论文原创性声明及授权使用说明](#_Toc686532466) 45

[个人简历](#_Toc686532467) 45

# 第1章 绪论

## 1.1 课题来源

随着电子商务的蓬勃发展，人们逐渐转变传统购物方式，从传统的真实物品交易转变为网上购物，银行等第三方支撑产业的发展也大力支持了电子商务的需求。在这样的市场环境下，机票行业将大部分的机票转到网络销售渠道进行售卖，机票售卖后的退款和改签等业务操作也由原来的线下支持转到线上[1]。基于这样的情况，完善网络购票售前售后服务的市场需求逐渐明朗。

本项目来源于北京去哪儿网对其机票事业部创新研发退改组的项目要求。机票退改签系统是去哪儿网线上系统售后部分的核心系统，需要根据机票业务实际情况做出正确响应。由于机票退改签系统尚存在金额计算错误或非实时更新的情况，因此提出更改退改签业务逻辑，完善去哪儿网线上售后系统的业务需求。

## 1.2 项目开发的目的与意义

去哪儿网作为全球最大的中文旅行平台，与携程不同的是，去哪儿网自身非机票代理商，公司系统只提供网络售票的平台而非代理经营。去哪儿网为用户抓取网络上各机票站点的机票报价，为用户提供全面的机票比价服务[2]。搜索与服务是去哪儿网的业务核心。

去哪网以机票为最大主营业务，每日机票生单量：10W单。设国内与国际机票平均每张1000元，即每日机票部交易额达到1亿以上。这样巨大的数字足以说

明机票市场之庞大。每日机票退改签量约1000单，是否及时响应用户的请求，是否能够准确计算出航空公司手续费及代理商手续费，准确给用户退款，是退改系统中最重要的部分。本系统中，我们要根据用户订单号，查询出此机票在各系统中的手续费。

本课题中，要着重确保这些金额的实时性和准确性，确保用户与代理商的利益，为用户提供优质的服务。为确保金额流转的准确且无遗漏无重复，本课题中采取消息队列方式传输消息，消息队列是本课题的技术重点。

## 1.3 与课题相关的国内外研究现状

### 1.3.1 机票业服务系统情况

根据2013年１月艾瑞对旅行类网站的访问次数统计，去哪儿网超过携程、艺

龙等网站，排名第一。自2005年成立至今，去哪儿网通过其卓越的搜索能力发展

为国内一线旅游网站，截止至2013年6月，去哪儿网可搜索航班航线超近130000条，酒店数量超过480000家，旅游度假线路超过200000条[3, 4]。去哪儿网APP客户端下载量过亿，拥有在线注册用户量4300万，是旅游行业中不可替代的领军企业。

随着互联网的发展，网络销售机票行业也随之兴起。电子商务的出现提高了商业贸易的交易透明度，也大大提高了交易效率。对于机票这种产品而言，其使用方式也适用于电子商务交易。用户凭借票号到达机场即可更换登机牌，并不需要提前将这种商品真实地拿在手上。因此机票的生产和销售环节都非常适用于电子商务[5-7]。加上现有的银行等支付行业的发展，用户可以足不出户购买机票，电子商务发展旅游业的条件已经非常成熟。

现在，电子商务占有市场总交易额的20%，预计在未来两三年内，电子商务占有额将达到30%以上。国内机票每年网络交易额约为40~50亿元人民币，现有网络交易量占机票总量70%以上，按照电子商务增长比例计算，两三年内，网络销售机票的占有量将超过机票总量的80%，约为40亿元人民币[8, 9]。其市场份额之大是其他行业不可比拟的。

航空公司将机票销售渠道分为直销和分销。直销是航空公司自营机票，按照机票真实票价出售，分销是航空公司抽取部分佣金后，让利销售给机票代理商，由机票代理商加价售卖的销售方式[10-12]。所以代理商加价比例决定了分销机票的最终价格。目前，机票市场的分销机票占机票总额的80%以上，因此分销机票是机票市场的主力军。机票分销市场供应链形式为五级：第一级，航空公司；第二级，区域大型机票代理商；第三级，B2B机票平台；第四级，中小型机票采购商；第五级，机票分销机构[13-16]。去哪儿网和携程均属于第五级分销机构。最近，航空公司发起大规模的降低佣金的销售措施，原有3%的佣金比例被下调到1%，西部航空将佣金降到了0%。航空公司大比例的下调佣金的行为也标志着机票分销行业的发展趋势，航空公司将大力发展分销行业，这对于整个机票市场供应链都是个转型提升的机会[17, 18]。

航空公司降低佣金后，让利于区域大型批发商，批发商将有更多的资金去收购机票。同时，这些成本降低的机票也会在接下来的供应链中呈现低价格的姿态，市场价格更具有优势。区域大型批发商自身销售能力有限，会将部分低价机票投放至B2B平台，小型采购商和机票分销机构能够采购这些低价机票的机会就会大幅增加。

国内机票代理商上千家，代理商数量庞大是市场庞大的体现，同时也是市场

混乱的一种体现，目前，机票代理商仍处于垄断竞争阶段，每个代理商都在努力地扩大自己的影响范围，增加代理产品数量，但仍摆脱不了规模小、竞争力薄弱、可替代的现状，没有哪个代理商能够在机票分销市场中占有绝对优势的地位。这样的情况就造成了机票分销市场的混乱，管理散乱不规则成为市场发展的瓶颈[19]。在这样的情况下，去哪儿网对机票统一的平台管理有利于市场的发展。去哪儿网对于所有OTA(Online Travel Agent在线旅游社)代理商提出TTS管理模式，即Total

Solution（统一管理模式），将所有类型的机票用统一的方式进行销售，统一管理售前和售后服务，在同一的服务平台操作机票的整个流程。

去哪儿网与携程的竞争始终存在，2014年第四季度去哪儿网机票生单率超过携程，但机遇与挑战并存，去哪儿网希望筛选出存票量高的市场价值高的机票供应商成为稳定合作伙伴，但被筛选出局的小型代理商手中的库存就会流入B2B平台，使竞争对手的存票量增加[20, 21]。在航空公司降低佣金的期间，存票量的降低使得机票市场形势更加严峻。

对于用户而言，最重要的是在保证产品质量的情况下拿到更低的价格。当代理商繁多时，无法在短时间内找到价格低服务好的机票产品。去哪儿网提供的全网搜索服务为用户搜索到大量的机票产品，将产品集中展现也促进了产品市场的销售。

### 1.3.2 消息队列发展情况

消息队列的意义：消息传输机制的设计目的是在应用程序之间构建一座“桥梁”，通过消息传输系统提高高效可靠的存储转发机制，将消息从一个应用发送到另一个应用中，实现应用系统间无缝的数据交换和共享[22, 23]。设计和应用消息传输系统有一定的现实意义。

实现应用系统间资源共享，为应用系统间的数据交换提供一种解决方法。消息传输系统是一个相对独立的消息中间件服务程序，它屏蔽了网络平台，操作协议的差异[12]。

目前业界有很多MQ（Message Queue消息队列）产品，我们作如下对比：

#### 1) RabbitMQ

RabbitMQ支持多种协议，如AMQP, XMPP, SMTP, STOMP等。其支持的广泛性使其本身具有较重量级的特点，因此RabbitMQ在大多数情况下更适合与企业级开发。RabbitMQ的实现原理是在发送消息时进入到中心队列，排队等候消息发送，其优点是使用负载均衡和路由将消息分发，并且支持数据持久化[24]。RabbitMQ

重量级的特点使得它自身的扩展性较差，封装后的消息体也较大，适合大型消息的传输[25]。

#### 2) Redis

Redis本身只是一个数据库，但是它支持数据量较小的消息队列功能。其特点是：当入队消息较小时，Redis展现出较好的性能优势，其速度超过了RabbitMQ；但随着消息体的增大，Redis性能大幅度降低[26,27]。除此之外，Redis还具有轻量级，出队速度快等优点。支持小规模的信息传输，Redis是很不错的选择。

对于Redis和RabbitMQ的入队及出队的操作统计，各执行100万次，每10万次记录一次执行时间。测试数据分为128Bytes、512Bytes、1K和10K四个不同大小的数据。实验表明：入队时，当数据比较小时Redis的性能要高于RabbitMQ，而如果数据大小超过了10K, Redis性能就会大幅度下降；出队时，无论数据大小，

Redis都表现出非常好的性能，而RabbitMQ的出队性能则远低于Redis。

#### 3) Jafka/Kafka

Kafka是跨语言的高性能分布式系统，Jafka在Kafka基础上发展得来。具有持久化速度快、吞吐量大等特点。其速度可以在O（1）的系统开销下进行数据持久化，即使对TB级以上数据也能保证常数时间复杂度的访问性能，即使在普通商用机器上也能做到单机支持每秒100K条以上消息的传输，这是其他消息队列无法做到的[28-30]。并且Jafka是分布式消息队列系统，能够自动实现负载均衡。同时，Jafka还具有轻量级的优点，是性能优越的消息队列系统。

#### 4) TongLINK/Q

TongLINK/Q支持文件传输功能，在许多业务场景中，单纯的消息传递是无法满足的业务需求的，因此TongLINK/Q的文件传输功能就显得十分重要了。同时，

TongLINK/Q具有稳定、可靠的性能优势，在市场中占有一定的市场份额。如中国银行的支票交换系统就采用了TongLINK/Q系统支持，TongLINK/Q的大规模消息分发能力使其在消息队列产品中脱颖而出[31]。

## 1.4 本论文的主要工作内容

为用户提供机票退改签服务，方便代理商的机票售后服务，提高代理商的服务质量和服务效率，航空公司提供的的机票退改签信息必须实时地反映在用户申请页面和代理商的服务平台上。退改签系统的订单信息要准确及时地传递给支付中心，为用户进行改签及退款操作。相关数据会持久化在公司内部的服务器上。

用户端发起退改签申请时，系统自动查询航空公司黑屏，获取当前用户机票的退改签规则，为用户展示，同时，系统获取展示销售此机票的代理商此刻的退改签规则。重点解决的问题是各个模块的连接，对数据库的实时连接，以及将退改消息地传输给支付中心进行打款操作。预期结果是为用户端和代理商的工作人员提供可靠的机票退改签信息和及时退款操作，实现机票售后系统的高效处理。

## 1.5 本论文的组织结构

本论文从组织结构上面可以分为下面几个部分：

第1章为绪论部分，绪论首先阐明了课题来源，然后分析机票退改签系统开发目的和意义，并对机票行业和消息队列的国内外发展现状进行分析，主要介绍了机票行业的发展趋势和几种常见消息队列的特点。明确了本论文的主要工作内容。

第2章为退改签需求分析部分，首先对系统的业务需求进行了分析，功能需求包括匹配规则需求，服务平台操作需求，消息队列功能需求，退至乘机人银行卡需求，以及可信度用户判断需求。这部分内容给出了消息队列整个产品中的位置，并在此部分对模块整体结构进行了描述。

第3章为退改签系统的设计部分，该部分对机票退改签方案的总体架构和消息队列子系统的总体架构做了明确阐释，将每个部分分为几个模块实现，各模块需要实现的主要功能做了明确的分工，并且明确了数据库的设计。

第4章为退改签系统的实现部分，该部分用具体的类说明了代码实现的方式，对乘机人可信度算法进行了描述，并对其进行了实现。分析消息子队列模块的设计分析其性能特点，给出了主要类和接口的实现方式。

第5章为退改签系统测试部分，对系统运行测试环境进行了说明，然后对本模块的功能进行了测试，给出了测试用例，并使用工具对系统进行了性能测试。

最后为结论部分，对整个论文进行了总结概括，给出了工作内容和研究成果，同时对该领域进行了展望。

# 第2章 机票退改签系统需求分析

本章主要阐述机票退改签系统的需求分析，根据系统实际业务流程，明确提出了匹配规则、服务平台退改操作控制、消息队列、可信乘机人判断、退款至乘机人等功能需求，通过细化各业务线的流程图，明确系统功能需求，最后明确系统性能需求。

## 2.1 业务需求分析

因为代理商为用户实际出票和用户购买政策中的产品存在差异，因此机票无法在机票购买时将退改签规则做得准确，也无法根据用户订单为用户查询到真实有效的退改签规则。因此要求在代理商为用户出票后，根据实际出票情况重新获取退改签规则。增加FD舱位票价不一致的计算模式，要求如下：

1）建立NFD（非航空公司公布运价票）退改签缓存库，以及匹配逻辑；

2）实现NFN（航空公司退改签规则解析）匹配、解析金额与逻辑计算；

3）修改订单为维度的规则为票维度规则；

4）代理商使用服务平台对用户退改请求的操作控制；

5）对于可信用户执行立即退款操作；

6）对于多次退款的行为，要求金额退至乘机人银行卡中。

## 2.2 功能需求分析

### 2.2.1 匹配规则需求

该文所研发的系统，主要实现的是针对去哪儿软件科技有限公司票务管理系统中的退改模块，机票退改签规则复杂且由多方共同决定。总体分为航空公司退改签规则和代理商退改签规则，金额不匹配航空公司退改签规则的退款全部认为是代理商人为手工操作的，因此退改签金额计算准确与实时具有随机性。规范退改规则建立如下退款流程，从业务的角度分为可匹配航空公司和不可匹配航空公司的退款流程。可匹配航空公司规则的退款流程图如图2-1所示，不可匹配航空公

司规则的退款流程图如图2-2所示。

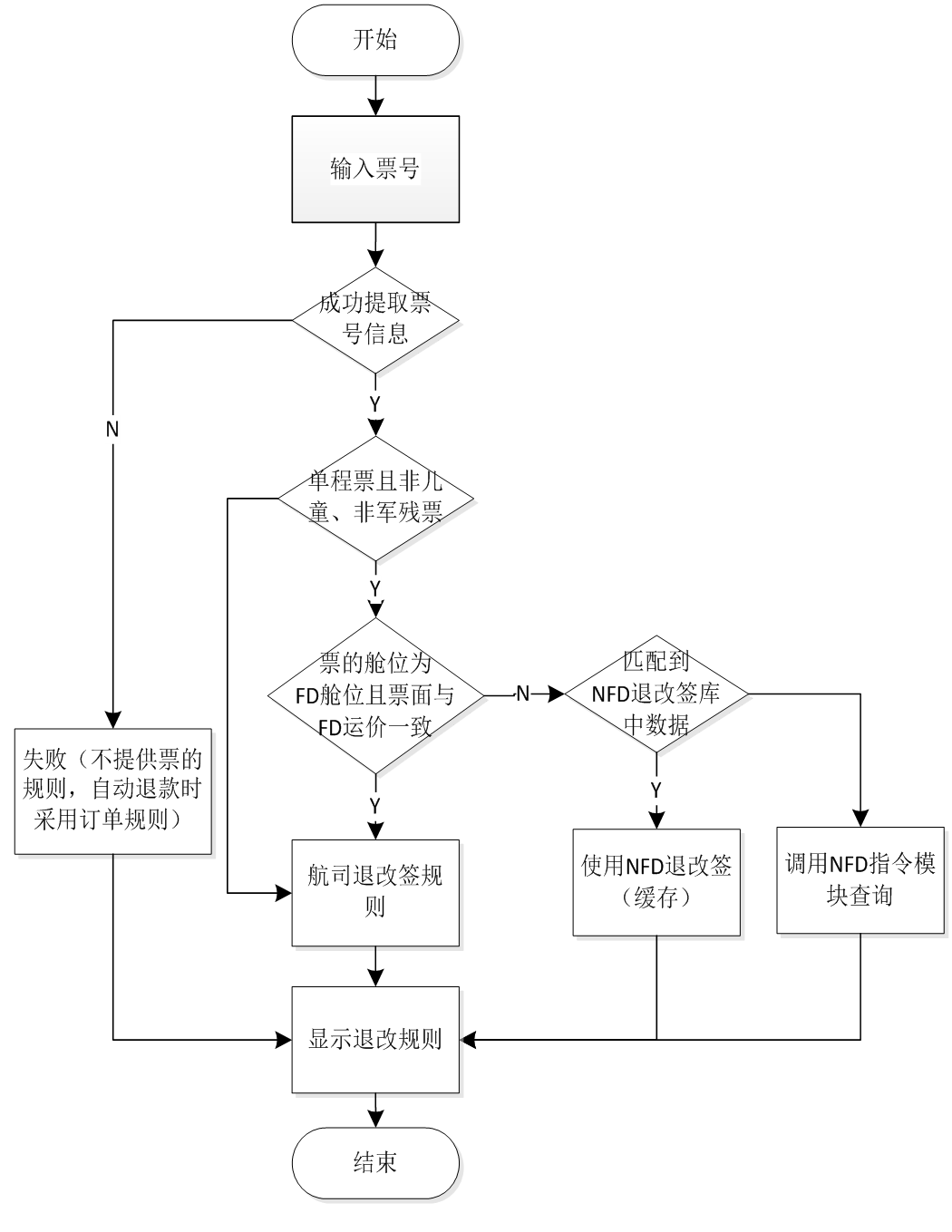


图2-1 可匹配航空公司规则的退款流程图

在NFD退改缓存库中未匹配到有效规则操作如下：

执行NFD指令，NFD:航班航班航段、起飞日期、航空公司英文简写三字码，

NFD:航班航段组/起飞日期/航空公司英文简写三字码及舱位；若指令1执行失败、返回为空、没有权限等异常情况，返回无规则，并将此条加入NFD退改签获取失败列表。

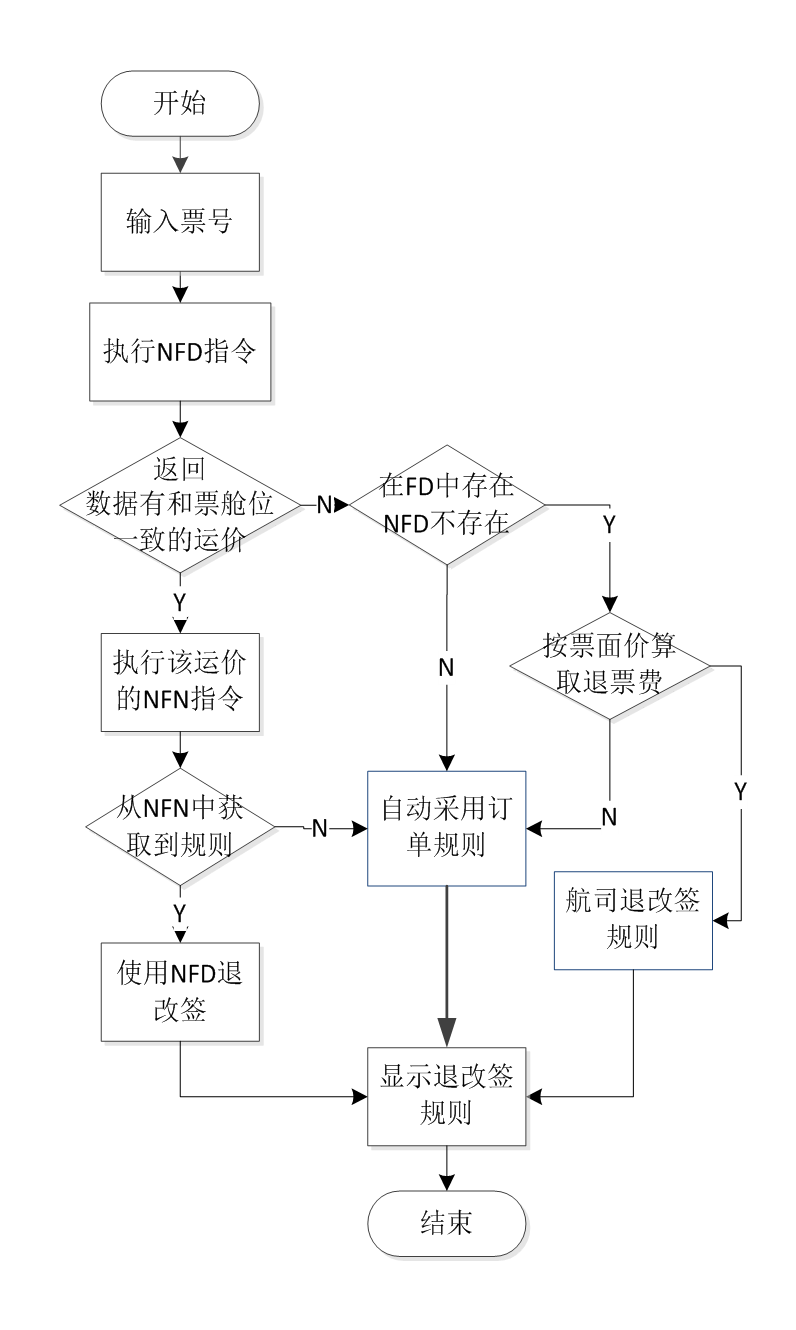


图2-2 不可匹配航空公司规则的退款业务流程图

判断NFD中返回的舱位、票价、FBC/TC是否与票号中一致是，则进行下一步；

否，则返回无规则，将此条加入NFD退改签获取失败列表；

舱位不存在NFD中：如果在FD中存在，则使用票面价与航空公司标准计算退票手续费；如果在FD中也不存在，则返回无规则，加入NFD退改签获取失败列表；

票面价不一致；

FBC/TC不一致；

执行NFN: N//08指令（N=相同票价和FBC中的第1条），提取NFN退票规则；

如果提取规则返回为空值，则系统返回查询此票号无票，将此条加入NFD退改签获取失败列表；

调用NFN模版库，并逐条（根据隔行分条）解析当前NFN中的条目，解析的条目最终有三种状态“已解析、未解析、无效”，未解析成功的退票规则条目进入NFN未解析列表；

获取NFN退票规则文本后，先将非自愿退票部分去除（无效数据）；根据隔行符，逐条调用模版并解析。

### 2.2.2 服务平台退改操作控制需求

机票服务平台代理商使用的订单原有页面有多个，不同页面的信息功能各不相同，代理商操作员需要自己去考虑和判断需要进入哪个页面进行操作，且可能需要关闭后重新打开页面；现在一个订单处理完后，页面自动关闭，并不会进入下一个状态的页面，本系统将实现订单的退改签连贯操作，订单修改后进入下一状态的页面继续操作。据统计代理商在出票中贴票号的平均操作时长18s；代理商操作鼠标滑至页面下方贴票号区域，点击修改链接，在文本输入框中粘贴票号，点击出票完成按钮；新页面信息整合后，减少代理商滑动页面、直接粘贴、信息对比查看更清晰，统计代理商在退款待审核页面，平均操作时长137s；代理商操作：进入退款待审核页面，验证票号状态正确，核对退款金额，切换到订单查看页找出票渠道，去上级出票点提交退票，再返回退款待审核页面，修改退票状态，提交确认退款；信息整合后，不需代理商切换页面查看出票渠道，不需代理商黑屏提票号状态，减少代理商滑动页面、信息对比查看更清晰。系统提供快速定位功能，提高信息认知度，提高代理商处理订单效率，减少订单页面操作时间，减少因切换页面、反复查找订单等操作时间，减少订单页鼠标滑动。服务平台业务操作流程图如图2-3所示。

?

图2-3 服务平台业务处理流程图

服务平台为呼叫中心提供的订单页面接口给出的，信息不全且缺失很多操作，通过新增各种接口来提供呼叫中心所需的各种信息与操作。本系统将统一订单页面，针对不同角色的订单页面使用者，根据权限控制相应信息的展示与操作，使订单页面使用者无须判断进入哪个订单页面即可完成所需操作；增加呼叫中心对

机票订单的操作权利，为用户提供准确实时的退改签服务。

### 2.2.3 消息队列功能需求

系统需要在现实网络上运行，因此真实的网络情况很有可能有不准确或阻塞的情况发生。在这样的情况下，系统需要保证消息传递的真实可靠，要使用消息队列进行传递。本系统中要求消息队列可以立即实时读取请求的消息，对于不能及时送达的消息，要保留在服务器端等待重发。

当有消息产生时，客户端发起发送请求，发送至服务器端等待服务器为其转发，若服务器发送成功，则向客户端发送一个确认报文，通知客户端它请求的消息已经转发。若发送不成功，则在服务器端存储这条消息，不断重发直至发送成功，再向客户端确认。有些特殊业务情况有特殊的消息发送需求，有些系统消息需要延时发送。消息队列对消息处理的业务流程图如图2-4所示。

?

图2-4 消息队列消息处理流程图



例如：系统希望在退款时给用户发送短信提醒，但有时支付中心的打款成功回调会选择在夜晚网络空闲时，退改签系统收到支付中心的回调信息会触发发短信的功能，但是实际上这个时候并不适合为用户发送短信提醒，会打扰到用户的休息，因此，我们希望系统能够把这些信息存储下来，等到白天再为用户发送。这样的信息就需要消息队列暂存。因此消息队列需要有暂存延时消息的功能。

### 2.2.4 二次退款退至乘机人银行卡功能需求

机票退款手续费由系统计算，但仍保留代理商手动输入功能，原因是有些特殊情况需要根据实际情况判断退款金额。退款至乘机人银行卡业务流程图如图2-5所示。



开始





Y

核实通过

N

N

结束

代理商发起二次退款请求

请求核实

填写银行卡信息

调用支付中心接口核实银行卡信息

请求驳回需重新申请

持卡人错误， 提示用户修改

银行卡信息错误，提示用户修改

按照所填支行进行打款操作

图2-5 退至乘机人银行卡业务流程图

然而人工操作即存在输入错误或金额争议，原有“退款完成”状态为订单终



持卡人姓名

为乘机人 N

Y

开卡省市填写正确

Y

止状态，即无法再次退款，只能由代理商线下采用支付宝等第三方方式给用户打款，这样的操作在公司的交易记录中没有明确记录，给公司与代理商后续的对账工作造成很大困难，也无法保证代理商会及时给用户退款，保障用户的利益优先。因此，系统开发二次退款功能，即“退款完成”状态非订单终止状态，用户与代理商协商后，代理商可在服务平台发起二次退款请求，只要订单金额未完全退回，则订单始终可进行再次退款操作。系统增加二次退款操作，提供更清晰的业务操作流程，提升使用率，减少线下退款。

在去哪儿网页面输入卡号相关信息支付（卡号信息包含卡号、证件号、有效期、开户名、手机号等，但只需要卡号、开户名、支付银行）。仲裁退款到乘机人范围扩大到只要用户申请按航空公司规则退款，一律进行仲裁退款到乘机人，仲裁是否退回乘机人的逻辑，由数据应用组给出。

### 2.2.5 可信用户立即退款需求

机票退款时判断乘机人是否是本公司的可信用户，对于可信用户可采取立即退款操作处理，可信用户立即退款业务流程图如图2-6所示。

目前订单无法支持可信乘机人立即退款，普通的退款速度：借记卡约3-7天/

信用卡约7-21天，若用户需要立即订另外一张机票，那么退款未到帐就会影响用户的购买。因此系统提出对于可信度较高的用户能够在代理商退款完成后可以在

30分钟内进行退款。系统需要根据用户的个人信息，原支付时是否使用常用卡支付，在公司购买产品的金额，历史记录等信息判断该用户是否是可信用户，若可信度满足一定要求，则该用户算作可信用户，其申请的退款操作将优先处理，即代理商确定退款时，系统可在30分钟内为用户进行退款。

?

2.3非功能需求

图2-6 仲裁退款业务流程图

为了达到系统在线上复杂网络环境中能够正常运行，系统需要满足以下几方面性能、安全等方面的非功能性需求。

(1)系统运行稳定，用户访问无页面无返回的错误。

(2)客户端响应速度，3秒之内响应基本的页面请求。

(3)消息队列传输消息准确实时，丢失率和失败率低，退改签金额计算准确。

(4)获取退改签规则为实时规则，确保金额计算的实时有效性。

(5)容错和抗干扰能力，在某台机器停止服务时，使用负载均衡能够自动响应打到其他机器上提供服务，系统终端能够正常运行。

(6)数据加密，对用户身份证电话号等敏感数据进行加密，保证数据安全。

## 2.4 本章小结

本章主要对机票退改签系统和消息队列子系统进行了需求分析。首先给出业务需求分析，明确系统整体功能。然后从功能和性能上分别对系统做出了明确的要求，细化了系统具体要完善的各个功能和形式并给出每个功能的业务流程图，最后对系统性能做出要求，对后续的设计与实现有一定的指向性作用。

# 第3章 机票退改签系统设计

本章阐述机票退改签系统的设计方式。系统总体分为退改签业务系统和消息队列子系统两部分内容。退改签业务系统中主要完成总系统的业务功能，包括退款和改签两大模块；消息队列子系统用于辅助完成业务过程中消息的处理功能；这三部分共同组成机票退改签系统。

## 3.1 机票退改签系统总体设计

本系统要在业务层提供退款和改签功能，因此系统总体设计为两大模块，退票模块和改签模块，退款模块和改签模块为系统业务处理模块，消息队列模块为系统支持模块。在退改签模块中完成退改签规则匹配功能，二次退款退至乘机人银行卡功能以及可信用户立即退款功能。在消息队列模块中完成退改签消息的生产和发送，业务中产生的消息都使用消息队列进行转发。

总体设计功能结构图如图3-1所示。

?

图3-1 退改签系统功能模块图

用户从浏览器进行退款或改签操作，表示层仅作浏览器页面处理。业务逻辑层进行退款与改签模块的业务处理，每一个业务逻辑都需要调用消息队列进行消息传递，从消息队列的客户端发起调用请求，消息队列进行业务处理，最后将消息持久化同步至数据库。

?

图3-2 退改签系统体系结构图

## 3.2 退改签业务系统系统模块设计

### 3.2.1 机票退改签计算方案

机票按票号去航空公司黑屏查询此票号状态，给出退改签金额计算结果。具体步骤：提取指令detr tn ticketNum提取票号信息，返回open for use且具有航班航段组（如from CTU to NNG），若返回信息错误，DETR提取不到票信息、非open for use、无航班航段组（如：void、open以及多航班航段等），则返回无票的规则[32]。根据机票信息判断是否军残或儿童票（此种机票信息进入代理商规则模块）。根据指令detr tn ticketNum,提取票面价，判断票面价是否与FD运价相同，由此判断是否为FD舱位，FD仓位使用infocenter航空公司退改签，非FD仓位根据航空公司、航班航段、舱位、起飞日期、价格、FBC去匹配NFD改签库中的退票规则，匹配多条采用创建时间最新，NFD中舱位和票价均一致，执行NFN指令，获取NFN中退票规则（NFN的指令格式，NFN:运价在NFD中的位置//08，示例

NFN:03//08.08 表示获取退票规则），解析退票规则文本，取出航空公司手续费

规则，计算退改手续费，给出金额。

本系统中，我们根据用户订单中每张机票的票号，查询出这些机票在各系统中的手续费，用户选择不同的退改原因，会相应计算不同的退改手续费，所涉及的金额部分有：用户支付金额、一次退款金额、二次退款金额、行程单差额、已返现金额、航空公司手续费、代理商手续费、各种红包费用。

### 3.2.2 机票退改签计算方案数据库设计

机票退票模块数据库设计包括航空公司退款手续费表，自动退款订单队列表，仲裁退款表数据库表，二次退款订单信息表等，具体数据库表设计如表3-1,3-2,3-3,3-4所示。

表3-1 passenger\_refund\_info仲裁退款表数据库表设计

| 字段名 | 字段类型 | 是否可空 | 含义 |
| --- | --- | --- | --- |
| id | int(11) | Unsigned NOT NULL | AUTO\_INCREMENT |
| arbitration\_refund\_id | int(10) | Unsigned NOT NULL | 仲裁退款 id |
| order\_no | varchar(23) | NOT NULL | 订单号 |
| domain | varchar(23) | NOT NULL | 域名 |
| cardholder\_name | varchar(20) | NOT NULL | 持卡人姓名 |
| card\_no | varchar(40) | NOT NULL | 银行卡号 |
| bank\_code | varchar(50) | NOT NULL | 银行编码 |
| branch\_bank\_province | varchar(50) | NOT NULL | 支行所在省 |
| branch\_bank\_city | varchar(50) | NOT NULL | 支行所在市 |
| branch\_bank\_name | varchar(50) | NOT NULL | 支行名字 |
| branch\_bank\_code | varchar(50) | NOT NULL | 支行 code |
| target\_merchant\_code | varchar(50) | NOT NULL | 实际出款商户号 |
| remit\_request\_no | varchar(50) | NOT NULL | 打款订单号 |

表3-2 repeat\_refund\_order二次退款订单信息表数据库表设计

| 字段名 | 字段类型 | 是否可空 | 含义 |
| --- | --- | --- | --- |
| id | int(11) | Unsigned NOT NULL | AUTO\_INCREMENT |
| order\_no | varchar(23) | NOT NULL | 订单号 |
| order\_id | bigint(20) | NOT NULL | tts 库存储的tts 订单 ID |
| payed\_price | decimal(8,2) | NOT NULL | 订单支付金额 |
| apply\_price | decimal(8,2) | NOT NULL | 二次退款申请退款金额 |
| confirm\_price | decimal(8,2) | NOT NULL | 二次退款确认退款金额 |
| status | tinyint(4) | NOT NULL | 二次退款订单状态 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | （续表 3-2 |  |
|  | 字段名 | 字段类型 | 是否可空 | 含义 |  |
|  | apply\_time | datetime | NOT NULL | 二次退款申请时间 |  |
|  | last\_update\_time | datetime | NOT NULL | 二次退款上次状态变更时间 |  |
|  | refund\_reason | tinyint(4) | NOT NULL | 二次退款申请原因 |  |
|  | domain | varchar(30) | NOT NULL | 订单的代理商域名 |  |

）

二次退款订单状态，1：二次退款申请中，2：二次退款审核中，3：二次退款完成，4：二次退款拒绝'

表3-3 hs\_refund\_fee航空公司退款手续费表数据库表设计

| 字段名 | 字段类型 | 是否可空 | 含义 |
| --- | --- | --- | --- |
| id | int(11) | Unsigned NOT NULL | AUTO\_INCREMENT |
| order\_no | varchar(23) | NOT NULL | 订单号 |
| apply\_refund\_time | datetime | NOT NULL | 申请退款时间 |
| passenger\_id | bigint(20) | NOT NULL | N\_passenger id |
| ticket\_no | varchar(30) | NOT NULL | 票号 |
| fee | decimal(12,2) | NOT NULL | 手续费 |
| refund\_rule | varchar(255) | NOT NULL | 退款规则 |
| source | tinyint(4) | NOT NULL | 计算来源 |
| create\_time | datetime | NOT NULL | 创建时间 |

计算来源，0 FD，1 NFD, 2 FD upper

主键：PRIMARY KEY (`id`),

表3-4 auto\_refund\_queue自动退款订单队列表数据库表设计

| 字段名 | 字段类型 | 是否可空 | 含义 |
| --- | --- | --- | --- |
| id | int(11) | Unsigned NOT NULL | AUTO\_INCREMENT |
| order\_no | varchar(23) | NOT NULL | 订单号 |
| status | tinyint(4) | NOT NULL | 订单状态 |
| refund\_price | decimal(8,2) | NOT NULL | 应退金额 |
| express\_fee | decimal(6,2) | NOT NULL | 快递费 |
| apply\_refund\_time | datetime | NOT NULL | 申请退款时间 |
| domain | varchar(255) | NOT NULL | 录入用户对应的域名 |
| deadline\_time | datetime | NOT NULL DEFAULT time | 自动退款截至时间 |
| auto\_refund\_type | tinyint(4) | NOT NULL DEFAULT '1' | 自动退款类型 |

DEFAULT time: DEFAULT '1970-01-01 00:00:00'

索引：KEY `uniq\_orderno` (`order\_no`, `domain`),

机票改签模块数据库表设计包括：改签航班信息表，改签支付信息表，改签乘机人信息表，改签退款表，退款解冻表等。具体数据库表设计如表

3-5,3-6,3-7,3-8,3-9所示。

表3-5 tb\_gq\_flight改签航班信息表数据库表设计

| 字段名 | 字段类型 | 是否可空 | 含义 |
| --- | --- | --- | --- |
| id | int(11) | Unsigned NOT NULL | AUTO\_INCREMENT |
| gq\_id | int(11) | NOT NULL | 改签 id |
| departure\_day | varchar(10) | NOT NULL | 起飞日期 |
| departure\_time | varchar(8) | NOT NULL | 起飞时间 |
| arrival\_day | varchar(10) | NOT NULL | 到达日期 |
| 字段名 | 字段类型 | 是否可空 | 含义 |
| arrival\_time | varchar(8) | NOT NULL | 到达时间 |
| flight\_no | varchar(12) | NOT NULL | 航班号 |
| gq\_fee | int(11) | NOT NULL | 改签费 |
| upgrade\_fee | int(11) | NOT NULL | 升舱费 |
| cabin | varchar(20) | NOT NULL | 舱位 |
| flight\_type | tinyint(1) | NOT NULL | 改签航程类型 |
| passenger\_count | tinyint(1) | NOT NULL | 改签乘机人数目 |
| gq\_flight\_time | timestamp | NOT NULL | 改签航班具体时间 |
| ticket\_price\_difference | int(11) | NOT NULL | 票面差 |

改签航程类型：0：单程；1：去程；2：返程

表3-6 tb\_gq\_pay改签支付信息表数据库表设计

| 字段名 | 字段类型 | 是否可空 | 含义 |
| --- | --- | --- | --- |
| id | int(11) | Unsigned NOT NULL | AUTO\_INCREMENT |
| pay\_no | varchar(30) | NOT NULL | 交易号 |
| pay\_status | tinyint(1) | NOT NULL DEFAULT '0' | 交易状态 |
| gq\_id | int(11) | NOT NULL | 改签 id |
| order\_no | varchar(30) | NOT NULL | 订单号 |
| pay\_type | tinyint(1) | NOT NULL DEFAULT '0' | 改签费用类型 |
| amount | decimal(11,2) | NOT NULL | 支付金额 |
| pay\_time | timestamp | NOT NULL | 支付时间 |
| pay\_remark | varchar(200) | NOT NULL | 改签支付备注 |
| share\_data | varchar(100) | NOT NULL | share\_data 用于支付和退款 |
| card\_type | tinyint(2) | NOT NULL DEFAULT '1' | 支付卡类型 |
| refund\_mode | tinyint(3) | NOT NULL DEFAULT '1' | 退款类型 |

交易状态：0：失败fail，1：成功success，2：accept或者unknown

改签费用类型：0：全部；1：升舱费；2：改签费支付卡类型：余额支付（4），信用卡（1），储蓄卡（0）

退款类型：1：BANK原路退回2：QUNAR返回到余额

主键：PRIMARY KEY (`id`),

索引：KEY `idx\_pay\_no\_status` (`pay\_no`, `pay\_status`), KEY `idx\_gq\_id\_pay\_status` (`gq\_id`, `pay\_status`)

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 是否可空 | 含义 |
| id | int(11) | unsigned NOT NULL | AUTO\_INCREMENT |
| tts\_id | int(11) | NOT NULL | ttsId |
| gq\_id | int(11) | NOT NULL | 改签 id |
| name | varchar(30) | NOT NULL | 乘机人姓名 |
| passenger\_type | tinyint(1) | NOT NULL | 乘机人类型 |
| id\_card | varchar(50) | NOT NULL | 乘机人身份证号码 |
| ticket\_no | varchar(15) | NOT NULL | 回帖的票号 |
| platform | varchar(20) | NOT NULL | 出票平台 |
| gq\_fee | int(11) | NOT NULL DEFAULT '0' | 单程改签手续费 |
| upgrade\_fee | int(11) | NOT NULL DEFAULT '0' | 单程升舱费 |
| gq\_fee\_in | int(11) | NOT NULL DEFAULT '0' | 返程改签手续费 |
| upgrade\_fee\_in | int(11) | NOT NULL DEFAULT '0' | 返程升舱费 |

表3-7 tb\_gq\_passenger改签乘机人信息表数据库表设计

乘机人类型：0成人；1儿童主键：PRIMARY KEY (`id`),

索引：KEY `idx\_product\_no` (`gq\_id`), KEY `idx\_name` (`name`)

表3-8 tb\_gq\_refund改签退款表数据库表设计

| 字段名 | 字段类型 | 是否可空 | 含义 |
| --- | --- | --- | --- |
| id | int(11) | Unsigned NOT NULL | AUTO\_INCREMENT |
| gq\_id | int(11) | NOT NULL | 改签 id |
| refund\_no | varchar(30) | NOT NULL | 退款号 |
| refund\_status | tinyint(1) | NOT NULL | 退款状态 |
| refund\_pay\_no | varchar(30) | NOT NULL | 支付流水号 |
| amount | decimal(11,2) | NOT NULL | 退款金额 |
| refund\_type | tinyint(1) | NOT NULL | 0: 担保，1：退款 |
| refund\_time | timestamp | NOT NULL | 改签退款时间 |
| order\_no | varchar(30) | NOT NULL | 订单号 |

主键：PRIMARY KEY (`id`),

索引：KEY `uniq\_refund\_no` (`refund\_no`), KEY `idx\_gq\_id` (`gq\_id`)

|  |  |  |  |
| --- | --- | --- | --- |
|  | 表 3-9 t | b\_gq\_unfreeze 退款解冻表数 | 库表设计 |
| **字段名** | **字段类型** | **是否可空** | **含义** |
| id | int(11) | unsigned NOT NULL | AUTO\_INCREMENT |
| gq\_id | int(11) | NOT NULL | 改签 id |
| pay\_no | varchar(30) | NOT NULL | 支付流水号 |
| pay\_time | datetime | NOT NULL | 支付流水号生成时间 |
| amount | decimal(11,2) | NOT NULL | 退款金额 |
| unfreeze\_no | varchar(30) | NOT NULL | 解冻流水号 |
| unfreeze\_time | timestamp | NOT NULL | 解冻支付流水创建时间 |
| unfreeze\_status | tinyint(1) | NOT NULL DEFAULT '2' | 解冻状态 |

据

解冻状态(0：失败fail, 1：成功success, 2：accept或者unknown)

主键：PRIMARY KEY (`id`),

索引：KEY `uniq\_unfreeze\_no` (`unfreeze\_no`), KEY `idx\_gq\_id` (`gq\_id`)

## 3.3 消息队列子系统模块设计

### 3.3.1 消息队列子系统模块功能划分

消息队列功能结构划分如图3-3所示：

?

图3-3 消息队列功能结构图

主要功能模块功能介绍：

Admin 是消息管理后台，用户可登录系统，申请消息上线，查看消息内容和

消费情况。遇到消息消费失败可以人工干预，手动重发消息。

Server是服务器端，主要负责接收client producer投递的消息。当Server端接收到Client Producer投递的信息后，要发送给Client端一个确认的报文，通知Client端它所投递的信息已被接收。然后Server端查找订阅该消息的client consumer，将消息转发出去。同时将Message发送的日志记录到Backup库中去。Server内部提供各种机制保证消息准确高效的送达。

Client是一个jar包，其中Producer API用于发送消息，Consumer API用于接收消息。其中包括消息的生产者和消费者，它们都属于消息队列的Client端，负责生产和使用消息。Producer端将消息投递给Server服务器端，报文中带有ack确认帧，等待Server返回的确认报文。若服务器端返回了确认，则认为消息已投放成功，否则重发本次消息[33]。Consumer端在接收到Server服务器发送来的消息后，也给服务器端返回确认报文，Server知道发送成功，将在自己的消息队列中删除这条信息，也不需要再向Producer端确认。这种设计隔离了消息的生产者和消费者之间的直接通信，使用Server为他们建立联系，保证了消息传递的可靠性。

为保证系统安全可靠性，系统设计Backup，Delay，Task模块进行队列管理：

Backup向hbase中备份历史消息和写入消息发送日志，并提供查询历史消息和消息发送日志的接口给其他模块调用。备份数据的目的是可以在网络出现异常的情况下查询备份数据，根据数据标志状态获悉信息是否需要系统再次发送。

Delay负责接收client producer发送的延迟消息做delay操作，当到达消息的预定投递时间时，才发送给server。延迟消息的特殊处理满足了系统特殊业务的需求。

Task执行定时任务。如将client producer发送失败的消息重推给server，另一个功能是将历史消息扫出来，交给backup备份。定时重扫消息的目的是管理队列中消息，确保消息不会长时间滞留在队列中，影响业务操作。

消息队列使用zookeeper管理消息类型，在消息系统里面的作用主要是存储基础的数据。

Broker用于维持消息订阅关系，consumer启动的时候会把自己的订阅关系注册到zookeeper上，broker会监听这个订阅关系，消息由subject唯一标示，broker通过对subject的监听管理生产者与消费者的订阅关系[34]。为了避免横向扩展broker的时候需要修改配置文件，把broker的基础信息（包括该node连接的数据库库，和对应的服务队列）都放在了zookeeper上，启动的时候去zookeeper获取这些信息。

消息队列注册机制如图3-4所示。



图3-4 消息队列注册机制

broker在接收到消息后会立即将消息持久化，并push到一个内存队列中，然后返回给producer. broker会有多个线程从内存队列里获取消息，然后通过查询

zookeeper获取该消息需要发送的consumer group，之后将消息与consumer group的关系持久化。然后将消息发送的职责转交给一个专门发送的线程池，发送的时候会使用指定的负载均衡算法从consumer group里选取一个consumer发送（对于每个consumer group只会选取一个consumer）。

消息在broker中的流程是分阶段进行的，各个阶段组合成一个流水线。对于特定流水线是串行执行的，流水线的上一步完成之后，交给流水线的下一步。多个流水线并行执行的。

在消息队列的broker中消息以及消息与consumer的对应关系会持久化到数据库中，以json的形式存储。在broker中有专门的定时任务对已经发送完毕或者无

consumer订阅的消息进行定时清理，清理的消息会转储到数据库中以备历史查询使用。在broker后台有专门的线程扫描消息状态，对于发送出错或超时的消息进行重新发送[35]。消息队列与数据库交互方式如图3-5所示。



图3-5 消息队列与数据库交互方式

### 3.3.2 消息队列子系统数据库设计

#### 1) Client 端

主要存储消息发送情况，根据消息Id，状态，重发次数判断消息的发送情况，并记录发送时间，以便决定后续是否再次发送。

消息队列Client端数据库表设计如表3-10所示：

表3-10 msg\_queue记录业务系统消息表数据库表设计

| 字段名 | 字段类型 | 是否可空 | 含义 |
| --- | --- | --- | --- |
| id | BIGINT | NOT NULL AUTO\_INCREMENT | 主键 |
| message\_id | VARCHAR(10 0) | NOT NULL | 消息编号 |
| status | smallint | NOT NULL | 消息状态 |
| error | unsigned | NOT NULL | 错误次数 |
| create\_time | DATETIME | NOT NULL DEFAULT CURRENT\_TIMESTAMP | 创建时间 |
| update\_time | TIMESTAMP | NOT NULL | 更新时间 |
| content | TEXT | NOT NULL | 消息体 |

#### 2) Server 端

消息队列Server端数据库表包括：记录业务系统消息表，消息发送状态表，消息备份表等，根据消息id连接多张表。具体数据库表设计如表3-11,3-12,3-13所示。

表3-11 messages记录业务系统消息表数据库表设计

| 字段名 | 字段类型 | 是否可空 | 含义 |
| --- | --- | --- | --- |
| id | BIGINT | NOT NULL | 主键 |
| message\_id | VARCHAR(100) | NOT NULL | 消息编号 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | （续表 3-11） |
| **字段名** | **字段类型** | **是否可空** | **含义** |
| subject | varchar(100) | NOT NULL | 主题 |
| sender | Int unsigned | NOT NULL | Producer ip |
| status | smallint | NOT NULL | 消息状态 |
| content | TEXT | NOT NULL | 消息的 json 表示 |
| create\_time | Timestamp DEFAULT 0 | NOT NULL | 接收时间 |
| update\_time | TIMESTAMP | NOT NULL | 更新时间 |

表3-12 onsumer\_state消息发送状态表数据库表设计

| 字段名 | 字段类型 | 是否可空 | 含义 |
| --- | --- | --- | --- |
| id | BIGINT | NOT NULL | 主键 |
| message\_id | VARCHAR(100) | NOT NULL | 消息编号 |
| consumer\_group | varchar(50) | NOT NULL | consumer 分组 |
| prefix | varchar(100) | NOT NULL | consumer 订阅 prefix |
| message\_status | smallint | NOT NULL | 消息状态 |
| retries | tinyint | NOT NULL | 重试次数 |
| current\_endpoint | varchar(22) | NOT NULL | Consumer 的 ip:port |
| log | text | NOT NULL | 发送日志 |
| created\_time | timestamp | NOT NULL | 消息创建时间 |
| update\_time | timestamp | NOT NULL | 更新时间 |

表3-13 backup消息备份表数据库表设计

| 字段名 | 字段类型 | 是否可空 | 含义 |
| --- | --- | --- | --- |
| id | BIGINT | NOTNULL | 主键 |
| message\_id | VARCHAR(100) | NOT NULL | 消息编号 |
| records | text | NOT NULL | 消息发送记录 |
| content | text | NOT NULL | 消息内容 |

## 3.4 本章小结

本章主要对机票退改签系统和消息队列子系统进行了系统设计。总体分为三大模块，对于每个模块介绍了设计点，退款与改签模块负责系统业务实现，消息队列模块负责系统消息的传递，三个模块的协作完成系统需求。对系统整体的框架进行了设计，并在设计基础上，对消息队列子系统的模块进行了功能上面的划分，明确每个模块实现的具体功能及协作方式。最后对数据库进行了设计。

# 第4章 机票退改签系统实现

本章主要阐述机票退改签系统的实现过程。分别对退改签业务系统和消息队列子系统的主要功能类进行了描述，并通过细化的代码流程图，阐述退改签系统的具体工作方式。同时阐述消息队列的性能及特点，最后给出主要接口的配置说明。

## 4.1 退改签业务系统设计与实现

当系统收到退款请求时，首先获取请求中的orderNo信息，在数据库中查询此

orderNo的订单信息以及对应的ticketNo,校验此ticketNo是否有效。订单信息中包含乘机人id，系统根据id查询乘机人表中的乘机人信息，判断passange\_age\_type是否是儿童，判断为true即计算代理商规则退改签金额。

若判断为false，则ttsPayRecords调取tts售前接口获得payment值，即订单支付金额，同时refundService执行NFD指令获取当前规则文本，然后使用NFN指令对文本进行解析[36, 37]。解析出来格式为“20-2-50”的计算规则，由此计算机票的真实退改签金额。将退款金额记入数据库内。根据乘机人可信度若此乘机人为可信用户，则立即退款。退款处理程序流程图如图4-1所示。

文本解析时首先查询NFD退改签规则缓存库，若能得到匹配的规则，则直接获取解析数据。若无法获得NFD匹配规则，则使用NFD指令获取航空公司黑屏数据，然后使用NFN指令匹配与当前航班匹配的规则，调用NFN模板库，对所获取的信息性解析[38]。

解析成功可拼凑出退改签规则，若无法正常解析或解析形成了规则闭环，无法正常拼凑规则内容，则将此条记录到失败列表中，由运营人员手动确认机票的航空公司规则。规则匹配程序流程如图4-2所示。

?

图 4-1 退款处理程序流程图

### 4.1.1 业务处理主要类说明

OrderPriceService类提供订单维度的相关金额，其中：

queryOrderPayedPrice方法可查询订单已支付金额。

queryOrderHadRefundPrice方法查询机票订单中已退款金额，其中包括：一次退款金额，二次退款金额，行程单差价，返现金额，查询结果返回price类型

的金额数值。

?

图4-2 规则解析匹配程序流程图

fetchOrderRefundPrice方法计算出此订单当前可退金额。

calHSRefundPrice方法计算出航空公司退款金额：其方法内涉及信息包括：获取订单信息，获取快递信息，获取机场建设费及燃油费，获取保险费，获取快递费等。计算方法如下：

应退金额=Σ（机场建设费+燃油费+ max(（票号订单票面价-票号行程单差价- 返现金额- 票号退款手续费）, 0））+订单快递费

退款手续费=Σ（min（票号订单票面价-票号行程单差价-返现金额，票号退款手续费））

fetchAllRefundPrice方法计算全退金额，返回RefundPrice类型金额。其中，全退金额=应退金额+快递费金额。

OrderRefundPriceService类提供订单已退款部分的信息。其中：

fetchShouldRefundAmount方法获取订单的应退金额，首先判断订单的状态，目前只支持退款待确认状态，然后根据订单的用户退票申请类别，分为自愿和非自愿进行处理。

fetchInvolunteerShouldRefundAmount方法获取非自愿退款应退金额，首先判断时候满足航变，航变为航空业自处理状态，若查为航变、9C或者往返，则系统不再自动计算。非自愿退款应退金额=订单总价-快递费-保险费（纯机票费用，不扣除任何手续费）

fetchBxFee方法判断保险金额，判断标准是首先判断是否已经出保，其次判断申请时间是否在起飞之前，根据返回结果决定退款金额是否包含保险费用。

RefundOperateService类进行真实退款操作。退款主要分为退款和确认退款（核准退款）两部分，因为在机票代理业，不同角色的权限有所不同，操作退款的工作人员不可以有真实打款操作的权限，因此，从业务的层面来看，退款操作已触发，但从数据真实处理的层面来看，需要等到有更高权限的管理员进行真实的确认退款。

comfirmRefund方法为确认退款操作，此方法中查询退改签信息，判断退改签信息是否合法，根据代理商domain和订单号获取退保类型，判断是否成功确认退款，增加确认退款保险日志。

Refund方法为退款操作，此方法查询退款信息，调取自动退款逻辑，根据代理商domain和订单号获取退保类型，判断是否退款成功，添加自动退款日志，对于自动退款成功的订单在库中添加一条自动退款完成记录。

repeatRefund方法为二次退款操作，方法中首先查询二次退款记录，若记录为空或二次退款状态错误，则跳出方法。库中记录表里有二次退款申请已核准的记录，则继续进行退款操作。二次退款成功后，设置退款金额和退款成功状态，库中插入一条自动二次退款的记录，从消息队列中删除此次退款操作申请。

isSuccessRefundInfoResult方法为退款成功后的更新操作，更改库中各表单状

态。

TicketCheckService类用于检查票号状态

commonMatch 方法查询航班基本状态，包括出发日期， 出发时间，航班号，

起飞机场，到达机场

isFromSegmentMatch方法核准去程票票号状态，

isReturnSegmentMatch方法核准返程票票号状态，

isStatusConsist方法判断票号状态符合要求，即票号为“已出票”状态，才是有效状态。

isPassengerNameConsist方法判断乘客姓名是否真实；

AutoRefundService类进行自动退款操作

checkAutoRefund是否订单是否支持自动退款，对于自愿退票的订单，计算其退款截至时间；对于非自愿退票的机票订单，退票原因选择：航班延误或取消、航班时刻变更等，系统通过调用航变系统接口返回得到的结果，若真实航变，则免除退票费用进行自动退款。若非航变，但用户提交原因为航变，则不进入自动退款队列，等待手动退款。

queryDeadline方法查询自动退款截止时间，自动退款的订单根据最晚截止时间进行打款操作。方法中查询自动退款截止时间、查询退款原因类型、获取该订单的服务等级等，然后计算截至时间，如果投诉时间早于服务等级的截止时间则更新投诉时间至截止时间。

queryRefundPrice方法查询订单应退金额。

queryOrderAirChangeInfo方法查询订单航变信息和参数。

comfirmRefund方法自动确认退款，获取订单号，确认的退款金额，确认退款的结果进行退款。

autoRepeatRefund方法与comfirmRefund类似，获取获取订单号，确认的退款金额，确认退款的结果进行退款等信息进行自动二次退款。

系统资源类图如图4-3所示。

OrderPriceService

-ttsPayRecordDao : TTSPayRecordDao

-ttsnOrderInfoDao : TTSNOrderInfoDao

-ttsnFlightSegmentDao : TTSNFlightSegmentDao

-ttsnPassengerDao : TTSNPassengerDao

|  |
| --- |
| OrderRefundPriceService |
| -orderInfoDao : TTSNOrderInfoDao  -ttsRefundRecordDao : TTSRefundRecordDao  -refundWaitConfirmService : RefundPrice  -insuranceInfoDao : InsuranceInfoDao  -cashBackService : CashBackService |
| +fetchShouldRefundAmount() : ShouldRefundAmountResult  +fetchInvolunteerShouldRefundAmount() : Integer  +fetchBxFee() : BigDecimal |

|  |
| --- |
| RefundOperateService |
| -refundQueryService : RefundQueryService  -refundInsuranceService : RefundInsuranceService  -refundDao : RefundDao  -repeatRefundOrderDao : RepeatRefundOrderDao  -autoRefundQueueDao : AutoRefundQueueDao |
| +comfirmRefund() : String  +refund() : String  +repeatRefund() : String  +isSuccessRefundInfoResult() : Boolean |

|  |
| --- |
| AutoRefundService |
| -orderRefundPriceService : OrderRefundPriceService  -airChangeInfoQueryService : AirChangeInfoQueryService  -orderFullRefundService : OrderFullRefundService  -refundStatisticService : RefundStatisticService  -autoRefundQueueDao : AutoRefundQueueDao |
| +checkAutoRefund() : AutoRefundCheckResult  +queryDeadline() : DeadlineResult  +queryOrderAirChangeInfo() : String  +comfirmRefund() : String  +autoRepeatRefund () : String  +refund () : String |

图4-3 系统资源类图

+queryOrderPayedPrice() : BigDecimal

+queryOrderHadRefundPrice1() : BigDecimal

+fetchOrderRefundPrice() : RefundPrice

+calHSRefundPrice() : RefundPrice

+fetchAllRefundPrice() : RefundPrice

TicketCheckService

-pidService : PIDService

-nPassengerDao : NPassengerDao

-ttsnFlightSegmentDao

-officeCodeService : OfficeCodeService

+commonMatch() : Boolean

+isFromSegmentMatch() : Boolean

+isReturnSegmentMatch() : Boolean

+isStatusConsist() : Boolean

+isTicketInfoConsist() : String

+isPassengerNameConsist() : String

+isTicketNoConsist() : String

根据业务形态，退款状态枚举类型分为：二次退款，行程单差价退款，返现，未知。枚举类型在RefundType类中列出。

public enum RefundType {REPEAT\_REFUND(0,"二次退款“),

XCD\_DIFF\_REFUND(1,"行程单差价退款“), CASH\_BACK(2,"返现”),

UNKNOWN(-1,"未知“);

}

订单状态枚举类型分为：订座成功等待支付，订座成功等待价格确认，支付成功，等待出票，未出票，出票中，出票完成，订单取消，等待座位确认，改签申请中，改签完成，申请退款，退款待确认，待退款，退款完成等。枚举类型在

OrderStatus类中列出。

Public enum OrderStatus {

BOOK\_OK(0,"订座成功等待支付“),

PAY\_OK(1,"支付成功等待出票"), TICKET\_OK(2,"出票完成"),

TICKET\_LOCK(5,"出票中"),

CANCEL\_OK(12,"订单取消"),

WAIT\_CONFIRM(20,"等待座位确认"), APPLY\_REFUNDMENT(30,"退款待确认"),

WAIT\_REFUNDMENT(31,"待退款"), REFUND\_OK(39," 退 款 完 成 "), APPLY\_CHANGE(40,"改签申请中"),

CHANGE\_OK(42,"改签完成"),

APPLY\_4\_RETURN\_PAY(50,"未出票申请退款"),

ORDER\_SUCCESS\_WAIT\_4\_PRICE\_CONFIRM(51,"订座成功等待价格

确认“）；

}

自动退款失败原因枚举类型分为：快递已寄出，无法获取office号“，解析detr，

f失败，乘客姓名不一致，票号不一致，行程单已经打印，票号状态不符，航班航段信息不符，调用确认退款接口异常，调用退款接口异常，无法获取应退金额，域名白名单过滤。枚举类型在AutoRefundFailReason类中列出。

public enum AutoRefundFailReason {XCD\_HAVE\_POSTED(0,"快递已寄出“),

CAN\_NOT\_FIND\_OFFICE\_CODE(1,"无法获取office号“), PARSE\_DETRF\_FAIL(2,"解析detr, f失败"), PASSENGER\_NAME\_NOT\_MATCH(3,"乘客姓名不一致”), TICKET\_NO\_NOT\_CONSIST(4,"票号不一致"), XCD\_HAVE\_PRINTED(5,"行程单已经打印"), TICKET\_STATUS\_NOT\_CONSISIT(6,"票号状态不符“), FLIGHT\_SEGMENT\_NOT\_MATCH(7,"航班航段信息不符”),

DUBBO\_CONFIRM\_REFUND\_FAIL(8,"调用确认退款接口异常“), DUBBO\_REFUND\_FAIL(9,"调用退款接口异常"), CAN\_NOT\_FETCH\_SHOULD\_REFUND\_AMOUNT(10,"无法获取应退金

额“），

WHITE\_DOMAIN\_FILTER(12,"域名白名单过滤“);

}

退款方式枚举类型分为：代理商退款，自动退款，呼叫中心退款，未知四种类型。枚举类型在RefundWay类中列出。

Public enum RefundWay {

OTA(0,"代理商退款“),

AUTO(1,"自动退款“),

CALLCENTER(2,"呼叫中心退款“), UNKNOWN(-1,"未知”),

}

获取航班的退改签规则解析方式，有NFD指令获取该航段全部退改退则明细，如图4-4所示。

?

图 4-4 NFD指令获取退改类型明细

获取航班的退改签规则解析方式，有NFD指令获取该航段全部退改退则明细，如图4-4所示。根据仓位RBD、机票金额公布运价OW, FBC/TC核对乘机人所持机票类型，获取其NFN编号，进一步获取模板信息。

使用NFN指令获得所持机票退改签规则。如图4-5所示。

?

图4-5 NFN指令获取机票退改签规则

通用格式，通用格式是指大部分航空公司录入退票的标准格式[39]。解析内容旅客类型、航程种类、适用舱位、运价基础；客票使用情况包括全部未使用、出票后未使用，首个航班起飞后计算已用航段票价、计算未使用航段退票费（比例100%）最低收费金额，未用税款[40-42]。通用格式需要提取的数据有：旅客类型（不限OR int）、航程种类（目前仅使用OW，忽略RT）、适用舱位（指定舱位OR不限）、未使用的首个航班起飞前（后）以及后面数字、计算未使用航段退票费以及后面比例数字、最低收费金额（数字OR为空忽略）。

需要解析的文本格式如下：

（1） 起飞后：0-100

（2） 起飞前：100-0

(3)当“未使用的首个航班起飞前（后）”后面有明确时间时：

(4)如：2小时到无穷小时，输出为100-2，意为起飞前2小时前退票费为100%\*

票面价

(5)当最低收费金额后有数字时，则与退票费计费比例取值取大，输出格式为：如max{最低收费金额，退票费计费比例} -min

(6)例：最低收费金额为100，则输出max{100，票面价\*100%}

(7)如果未使用航段退票费为固定金额后有数字时，则输出固定金额

(8)例：固定金额200，则输出0-200cny

解析结果分类：

#### 1） 通用格式Y 舱

格式与通用格式相同，计算退票费时使用Y舱或F舱计算；

旅客类型：不限航程种类：不限适用舱位：ZJASRWP运价基础：不限

客票使用情况：全部未使用出票后：不限未使用的首个航班起飞前：不限计算已用航段票价：无计算未使用航段退票费：按指定舱位公布运价乘以指定比例收取(Y舱,20%)未用税款：可退

解析为：

(1)当“计算未使用航段退票费”后含有“指定舱位”，“公布运价”，“乘 以指定比例”or（Y舱, 20%），表示使用Y运价计算手续费

(2)输出：20Y-0（指起飞前退票费为Y\*20%）

#### 2) MU-Z格式

(1)航班起飞7天以前，按票面价收取40%退票费

(2)航班起飞时间至航班起飞7天内，按票价收取80%退票费

(3)航班起飞后，票价不可退，退还机建燃油税。

(4)客票部分使用，未使用航段票价和燃油税不可退，退还机建税解析为：

（1） 当含有“航班起飞”AND" N天以前“，后”按票面价“，收取后含”40%“；

（2） 则输出：40-7\*N

（3） 当含有“航班起飞”AND" N天内“，后”按票面价“，收取后含”80%“

（4） 则输出：7\*N-80-0

（5）当含有“航班起飞（以）后”，后“票价不”可“退”OR“（仅）退（还）基建燃油税”则输出：0-100

（6） 最终输出：40-168-80-0-100

#### 3) NS-K格式

(1)若销售3.6折（含）以上运价，按照该舱位客票类别所对应的规定退票

(2)若销售3.6折以下舱位运价，客票类别为YK, YN, YZ, YD时，则仅退机建费和燃油费，机票款不退

解析为：

(1)当含有“该舱位客票类别所对应的规定退票”，则视为无效；

(2)当含有“N折（含）以下”AND“仅退机建费和燃油费”OR“机票款不退”

(3)输出：0

#### 4) NS-E格式

(1)多航段组合特殊运价客票已使用部分航段，退还未使用航段的实际订座舱

位运价，特殊产品未使用航段实际订座舱位运价

(2)航班规定离站时间2小时前收取客票价的40%作为退票费；航班规定离站时间2小时（含）之内及起飞后收取客票价的50%作为退票费.

解析为：

(1)当含有“多航段”AND“已使用部分航段”，则视为无效；

(2)当含有“航班规定离站时间”AND" N小时前“AND”客票价40%“，输出：40-N

(3)当含有“航班规定离站时间”AND" N小时（含）之内及起飞后“AND”客票价的50%“，则输出：N-50

（4） 最终输出：40-2-50

解析文本内容存入NFN库中，当再次有相同类型的机票进行退改签操作时，优先查询NFN库，若存在相同航段、相同时间、相同航班的机票，就可以直接过去到退款规则，不必再次调取指令查询。NFD信息列表如表4-1所示。

表4-1 NFD获取航班规则信息表

| 创建时间 | 航  空公司 | 出  发 | 到  达 | 有效期  起始 | 有效期截止 | 舱  位 | OW | 退款规则 | 状  态 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2014-07-31  12: 23 | M U | P E  K | S Y  Z | 2014-07  -23 | -- | Q | 200.00 | 40Y-2-80Y-0-80Y | 有  效 |
| 2014-07-31  12: 23 | M U | X  I Y | X  M N | 2014-07  -23 | 2015-07-23 | Q | 280.00 | 40-168-80-0-100 | 失效 |
| 2014-07-31  12: 23 | M U | N K  G | X M  N | 2014-07  -23 | 2015-07-23 | Z | 300.00 | 0 | 失效 |
| 2014-07-31  12: 23 | C A | T A  O | C A  N | 2014-07  -23 | 2015-07-23 | T | 320.00 | 600cny-0-600cny | 有效 |
| 2014-07-31  12: 23 | CZ | P  E K | C  A N | 2014-07  -23 | 2015-07-23 | T | 930.00 | 40-2-80-0-80-m-50  -50 | 有效 |

### 4.1.2 乘机人可信度计算

为提高用户体验，对于高诚信用户，提交退款后立即将金额退回，缩短自动退款时间，能够提升用户体验，减少代理商人力成本。

根据监控显示做数据统计，11月24日-11月30日退款申请至退款完成时间统计，退款完成订单数量总计：68346单，平均处理时长为11.37小时。

其中，代理商人工退款数量：46724单，平均时长11.94h，系统自动退了数量

21622单，平均时长10.16h。

由此可见，无论是代理商人工操作还是系统自动处理，对退改签订单的处理时长都略长。机票的平均价格都较高，如果用户着急出行，那么这笔退款将直接影响用户接下来的购买。也正因为机票价格高昂，迅速准确的退款，会大大提升用户对网站的信任度，因此，我们增加可信用户立即退款判断。可信用户立即退款逻辑图如图4-6所示。

?

图4-6 可信用户立即退款逻辑图

为研究可信度用户数据，我们建立可信度计算模型。可信度计算方法如公式4-1所示，给出部分参考参数：

F=** ( **Fs,*Fr*,*Hp*) (4-1)

1）Fs表示用户订单金额，权重因子包括：活跃度权重，产品类型权重，机票舱位权重。

2）Fr表示退票金额，权重因子包括：活跃度权重，产品类型权重，机票舱位权重，投诉权重。

3）Hp表示用户总消费金额，金额权重因子。

4.2消息队列子系统实现

4.2.1消息队列子系统实现方式

消息队列为了在分布式环境中确保消息一定送达，所以采用重试机制直到确认消息被确定的消费成功，由于分布式环境的不确定性，例如：1）。 网络抖动，导致ack消息丢失或者ack消息消息超时[43]。2）. 消息队列server正在部署或重启，导致没有接到ack消息，或者接到ack消息了但还没来得及更新ack状态，有的时候实际消费成功了，但broker并没有收到ACK确认，所以也会存在消息重发的情况[44]。Consumer端应该做幂等设计来消除这种重复对业务的影响，消息队列不对不发出重复消息做承诺。所以在数据库表中使用唯一性约束杜绝由于ack重复而导致的消息重复发送。

消息队列通过注册消息发送器进行消息发送，消息发送通过不同的途径发送给不同的用户；内部调用的方式将消息发送是内部的某个接口调用来将消息发送，不需要外部调用，也就是不需要定义外部通信协议。向外部提供的接口通过外部的调用来进行消息的发送，这就涉及到外部接口之间的通信协议[45]。若使用的是没部接口的调用，就不需要进行协议的定义，仅将发送消息的接口封装好，等其他人来调用发送消息。但是如果是使用的对外提供的接口，通过外部接口的调用来发送消息的话就得注册协议是http协议或者dubbo协议。消息队列转发方式如图4-7所示。

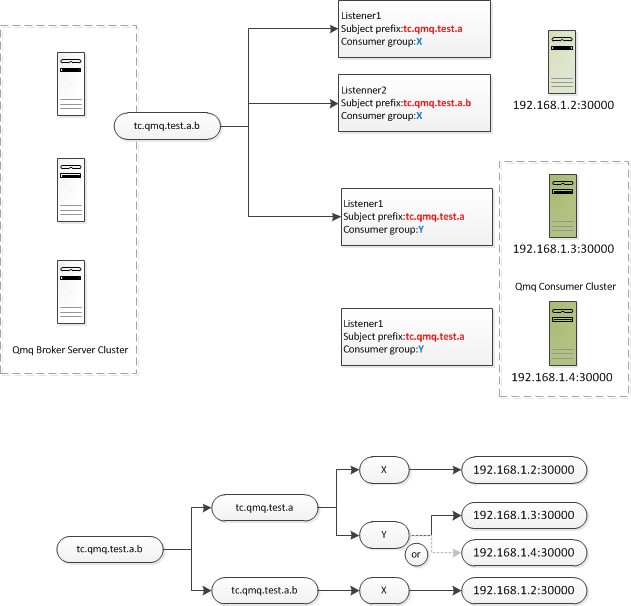


图4-7 消息队列转发模拟图

消息生产者按发送消息可靠性分为：事务持久型消息生成者，（非事务）持久型消息生产者，非持久型生产者和非可靠消息生产者。事务持久型消息生产者：可靠性由消息队列保证，非常可靠，可与业务绑定在一个事务中，业务处理成功消息发送，业务处理失败业务回滚消息不发送，但并发量不高。持久型消息生产者：可靠性由消息队列保证，非常可靠，业务处理和发消息不在一个事务中，可能会导致业务处理失败但消息发送出去了，但并发量不高。非持久型消息生产者：消息在client端不持久化也不保证事务，在server端持久化，可靠性中等，client端可通过捕获异常和响应监听实现重发提高可靠性，并发量中等。非可靠消息生产者：消息在client端和server端都不持久化，而且消息消费者消费失败也不会重推消息，可靠性最低，但是并发量大，性能高，一般用在非可靠通知场景。

消息生产者按消息投递时间分为：实时消息生产和延迟（或定时）消息生产者。消息在client端发送过程会经历4个步骤：校验、存储、队列缓冲和RPC远程调用。如果某consumer group（A）对某subject（S）从未订阅过，并且broker在接收

到S下的消息时，A中没有任何consumer在线，则该消息自动丢弃（所以为了确保不丢失任何消息，在首次上线时先发布consumer端）。如果某consumer group（A）对某subject（B）订阅过，但是在broker接收到S下的消息时，A中的consumer因为某些原因离线，则在该consumer再次上线时broker会主动将未过期的消息推送给A[46]。已经过期的消息即使发送失败，系统也不会调用broker再次推送。

配置消息队列的方式：

#### 1) 配置MessageConsumer

MessageConsumer接口的实现类是MessageConsumerProvider，配置如图4-8

所示：

?

图4-8 MessageConsumer接口配置参数

#### 2) 调用MessageConsumer的addListener方法订阅消息

抽象消息接收监听器AbstractMessageReceiverListener实现了消息接收监听的注册与注销，接收消息线程池的初始化和关闭功能，但非实现如何处理接收到的消息。

事务管理器TransactionManager通过重写Spring的DataSourceTransaction

Manager来确保消息的持久化和业务操作在同一个事务中，如果目前的应用重新实现了Spring的TransactionManager则需要继承TransactionManager。

创建MessageProducerProvider在使用延迟和定时消息投递功能时，需另创建一个MessageProducer。

addListener的第一个参数是subjectPrefix. subjectPrefix并不等价于消息的

subject. 当有两个consumer时，一个订阅了test. a的消息(subjectPrefix为test. a)，另一个订阅了test的消息(subjectPrefix为test)，如果producer发送一条subject 为

test. a的消息过来，这两个consumer都会收到该消息，因此subject是有层次的。

addListener的第二个参数是consumer集群标识。消息队列消息推送方式是对于一条消息，给每个订阅的集群推送一条。当subject为a时，有两个consumer集群都订阅了这个消息（集群a，集群b）。每个集群里有3台机器，则a这个subject有一条消息来的时候，会给集群a里一台机器发送一条消息，同时也会给集群b里一台机器推送一条消息。消息队列来做缓存过期通知时，需要采用广播的模式，即每台机器都收到消息。将addListener的第二个参数留空（null或空串），这样每台机

器就是一个集群，每台机器都会收到消息。

由于broker在推送消息到consumer失败后有重试机制，所以consumer端的

MessageListener应该是幂等的。在listener的onMessage执行完毕之后，消息队列

client会根据listener执行结果发送ack回消息队列broker，如果onMessage方法执行出现异常，则返回ack error，那么后续该消息会重发。在保证接口的幂等上，我们往往依赖数据库的唯一约束来确保接口的幂等性，但实现的时候要注意，如果消息队列第一次发消息发送成功，由于网络等原因导致消息重发，这个时候可能引起DuplicateKey异常，这个时候业务方捕获该异常，记入日志或监控，系统不抛出该异常，否则消息队列会认为该消息处理失败，导致再次重发。

### 4.2.2 消息队列子系统性能及特点分析

程序之间异步通信，不需要双方同时运行，也不需要双方建立起完整的通信连接，只要发送方将消息放入到消息队列中，由消息队列负责转发，因此对收发双方要求降低，对环境的依赖也相对降低。消息存入消息队列时不需要接收方在线运行，同理，也不需要接收方立即接受处理，消息队列会有暂存功能将其暂存。在复杂的业务环境中，这种通信方式还可以支持一对多、多对多的通信，各方均不受其他业务的影响，可以做到各自隔离稳定运行[47]。某个程序的中断或异常也不会造成消息阻塞，影响其他程序消息的发送与接收。

当生产者产生一条消息时，调用admin模块中controller层的Message Controller方法，到zookeeper中注册，然后调用SendMessageController将此消息发送出去，订阅这条消息的监听者根据task不断扫描消息队列，当有新消息到达时，client模块中的ConsumerMessage方法会消费掉这个消息。

使用消息队列的目的是保证退改签消息的实时性和准确性，其三大性能为：

（1）可靠性

采用事务、重试机制、完整状态机、Ack确认方式保证消息可靠发送与接收。事务：为避免消息传输失败时与业务逻辑不一致，把业务操作和消息发送封

装成事务。事务具有原子性，不可拆分，保证业务操作和消息发送结果一致，若消息发送失败，则所有相关操作和数据回滚，确保操作与数据的一致性，避免金额不一致现象。

@Transactional

Public void payOrder(Order order)

{ //发送消息通知系统支付成功producer. sendMessage(buildMessage(order));

//更新数据库状态表示支付成功updatePayState(order);

}

重试机制：消息发送需要通过网络传输，网络状况不可预测，会存在网络阻塞，断网断电等异常情况，导致消息发送失败或延时。因此设置重试机制，将消息持久化到数据库，即使硬件系统暂时出现故障，也可在重启后从数据库中获取未发送成功的数据重新发送。

状态机：将消息状态区分清晰，已发送、未发送、发送中、发送失败等，消息分发使用集群，可能有多台机器同时处理数据，消息状态及时更新，确保消息发送成功且不重复。

Ack确认：当client端收到消息，发送ack确认给server端，ack中包含此消息是否已成功处理，若client端系统阻塞无法及时处理，server端可根据ack确认帧中的内容重新发送此消息。

（2）可用性

采用集群方式防止单点故障，持久化可恢复保证系统不会因硬件问题无法支持请求。

集群防单点故障：系统硬件使用多台机器集群，确保当硬件出现故障时消息无法及时发送。集群技术可以采用负载均衡机制，将新加入的进程打到负载较轻的机器上，避免了某台机器高频高压的工作，使机房的资源得到充分利用。即使某台机器出现故障，负载均衡机制可以将请求打到集群中其他机器上，保证对外提供的服务正常运行，提高系统的可靠性和稳定性。

持久化可恢复：将消息持久化到数据库，即使机房整体断电，全部机器无法发送处理消息，也可在重启后去数据库获取未发送的消息进行重发。

保存并转发消息：如果消息要求标记为持久性，则应由消息队列提供者负责利用“保存并转发”机制，实施它和发送者之间的协议。存储机制用于将消息持久保存在磁盘上，以确保在提供者发生故障或消费客户端发生故障的情况下，消息仍然可以恢复正常。退改签系统的存储机制实现依靠消息队列中的backup模块。使用各个发送或接受的客户端，消息可以实现集中存储，转发机制负责从数据库中取回消息，随后再按照规定的时间进行重发。

消息确认和故障处理机制：消息队列的提供者监测一条消息的整个过程，以便了解是否成功地生产和消费了该条消息，消息队列的提供者通过监控这些消息就能够管理消息的重发，并保证他们可靠传送。

（3）高性能

通过系统异步性调用、批处理、合理使用线程池、Hbase大数据存储保证系统性能。

异步性：线上系统具有高并发性，机票系统的通常在下午14点左右达到峰值，当消息发送给server时，使用Dubbo接口，将数据备份，也使用异步操作。避免饥饿和等待现象。

批处理：机票系统由于其自身特殊性的要求，实时性是机票退改签的重点，因此数据的实时处理是系统考虑的重点。线上系统会出现大规模并发的情况，单位时间内需要处理的数据量规模巨大，因此逐条记录到数据库会影响系统应答速度。因此本系统采用批处理解决数据问题。其优点是批处理可以大大减少IO操作，

IO操作耗时较大，减少IO操作可以大幅度减少时间，有效地避免数据传输过程中产生的大量通信开销对系统的影响，提高系统性能。

线程池：机票系统的业务数据具有高并发性，系统运行时会产生大量的线程来完成任务，且要求任务完成的时间较短。例如类似于WEB服务器完成网页请求等任务，就适合采用线程池技术。若单位时间内请求过多，应答的线程过多，就会造成系统内存分配异常，产生“OutOfMemory”的错误，采用线程池技术可以避免这样的错误，在短时间内接受大量的外部请求，并且不耗费大量的服务器资源。因此本系统中消息出队入队，均使用线程池，多线程处理模式保证各请求都能及时得到响应。

大数据存储：使用hbase进行存储。Hbase是一个面向列存储的分布式存储系统，Hbase可以实现数据切分的透明存储，使得存储本身具有延展性，并且Hbase卓越的吞吐量可以支持系统的高并发业务情况。Hbase写入速度是每秒万级，满足系统每天存储的数据量极大的现实需求，保障了系统的数据安全。

在消息队列发送消息时，由于类型不同，因此需要消息注册到注册中心注册主题，不同类型的消息具有不同的group类型，对于消息订阅者，只需要订阅某个

group类型的消息，就可以不关心是谁产生的消息，只要接受这种group类型的消息进行消费和处理即可。通过类型注册，将消息分类，便于消息生产和消费双方使用。若消息未能正常发送，那么它仍然存在于这个队列中，等到下次消息轮询时，仍可将其发送。

在消息队列模式中，消息根据不同的主题发送到不同的队列里，消息队列传输质量得到保证，系统使用消息队列传输消息的优点在于，生产者只需要关注自己生产消息即可，生产出的消息放入消息队列，不需要与消费方建立通信连接，这样在网络中避免和很多不必要的麻烦。消息队列管理模块会定时将存入的消息发送到消费方，不需要生产者干预。

## 4.3主要类和接口说明

### 4.3.1消息队列主要类说明

消息队列运行类图如图4-9所示。

ProduceMessage类负责生产消息，根据业务线的需要，用户的操作触发时，系统生产一条消息发送出去，供其他模块消费。其中：

send设置这个方法的目的是未来引入多实例.当未指定消息存储时，使用同步发送，发送失败会抛出异常。当消息进入发送队列发现消息发送队列已满时，此消息将暂时丢弃，等待task处理。消息在用户进程阻塞，等待队列激活.重新入队时成功进入发送队列.由于无法入队，发送失败！取消发送！异常时，同一条消息不能被入队两次。

error方法用来处理消息发送结果。若消息发送失败，系统使用tryCount的值控制重新发送。

当尝试次数超过tryCount上限，则认为是网络或系统异常，提示尝试发送消

-messageService : MessageService

-subjectService : SubjectService

-zkClientService : ZookeeperClientService

-retryService : MessageRetryService

+index() : void

Sender

-scheduleMessageStore : IScheduleMessageStore

-dubboMessageServicebroker : BrokerMessageService

-batchDelayMessageExecutor : BatchDelayQueueExecutorService

+send() : void

+process() : void

+initNeedSendMessages() : void

+handleResult() : void

息。

MessageController

|  |
| --- |
| ProduceMessage |
| -base : BaseMessage  -queue : QueueSender  -sendStateListener : MessageSendStateListener |
| +send () : void  +error () : void |

|  |
| --- |
| ConsumerMessage |
| -registryURL : String  -inProcessHolder : MessageInProcessHolder  -processThread : Thread |
| +ConsumerMessage() : void  +ack() : void  +setProcessThread() : void  +autoAck() : void |

-executor : ThreadPoolExecutor

-listener : MessageListener

-serializer : Serializer

-brokerGroupRoot : String

-zkAddress : String

图4-9 消息队列类图

MessageHandler

+handle() : void

MessageHandler类负责管理消息。

Handle方法用来将当前线程设置给message，当业务调用message的autoAck

（boolean）的时候，会判断这个线程与当时的当前线程是不是一致的，如果不一致，说明业务有可能是在别的线程里设置这个，这是不允许的，系统会抛出异常。同样，当客户端线程已满超过上限时，系统也会给出异常提示。

Sender类用于发送消息

Send方法发送消息到延迟队列，延迟消息，若入队失败，等待任务调度重发。

Process方法用于去掉队列中的重复msg，调用dubbo，将baseMessages发送给broker，检查调用结果，错误消息等待定时任务重推，错误消息等待定时任务重推。

handleResult方法中设置broker dubbo超时有可能发送成功却返回重复消息异常，success(message)；方法用于更新消息状态，等待定时调度任务重复。

### 4.3.2消息队列主要接口说明

#### 1) Message接口消息设置

Message接口设置如表4-2所示，用于配置message信息，提供给各模块调用。

表4-2 Message接口消息设置

| 参数名称 | 参数设置信息 |
| --- | --- |
| maxMessageSize | 最大消息容量 60 \* 1K |
| maxQueueSize | 最大队列设置，默认 10000 |
| sendThreads | 发送消息开启线程数 |
| sendTryCount | 消息发送次数设置，默认 5 |
| minExpiredTime | 最短消息过期时间，默认 15 分钟 |

#### 2) MessageProducer接口设置

MessageProducer接口用于生产者生产消息时使用，系统中生产消息的模块需初始化此接口后，才可以发送消息。在发送消息之前系统调用该接口生成消息，该接口会生成唯一的消息id。消息的过期时间为默认的15分钟。参数设置如表4-3所示。

表4-3 MessageProducer生产消息接口设置

| 参数名称 | 参数设置信息 |
| --- | --- |
| messageId | 用户提供的消息 ID |
| subject | 消息 subject |

|  |  |
| --- | --- |
|  | （续表 4-3） |
| 参数名称 | 参数设置信息 |
| expire | 多长时间后过期 过期时间不能超过 24 小时 |
| timeUnit | 时间的单位 |
| MessageConsumer | 注册消息处理程序 |
| 接口名称 | 接口设置信息 |
| subjectPrefix | 订阅的消息分类 |
| consumerGroup | consumer 分组 |
| listener | 消息处理程序 |
| executor | 消息处理线程池 |

其中consumerGroup参数用来标志consumer分组，用于consumer的负载均衡

（broker只会给每个consumer group发送一条消息）。如果想每个consumer进程都收到消息（广播模式），只需要给group参数传空字符串即可。

#### 3） MessageConsumer接口，消费者订阅消息时需初始化此接口。参数设置如表4-4

所示。

表4-4 MessageConsumer接口参数设置

| 参数名称 | 参数设置信息 |
| --- | --- |
| subjectPrefix | 订阅的消息分类 |
| consumerGroup | consumer 分组，用于 consumer 的负载均衡 |
| listener | 消息处理程序 |
| rejectPolicy | 处理队列满时的拒绝策略 |
| executor | 消息处理线程池 |
| eturn | 返回的 ListenerHolder, 表示注册关系 |

## 4.4 本章小结

本章根据系统设计的内容，实现了系统调度模块的各个子模块的具体代码流程，并阐述了每个模块的实现方式以及采取的策略。在本章中，对本系统模块的实现进行详细的描述，主要包括基本的可信乘机人计算方法，退款规则模板解析方式，接口实现参数设置等进行阐释了。并对消息队列功能类进行了设计与实现，最后则列出了一些主要类的实现以及接口。

# 第5章 机票退改签系统运行与测试

本章主要阐述机票退改签系统的运行及测试情况。通过具体的测试用例检测系统的功能性需求是否完善，采用公司通用的mock平台测试系统的对外接口，对代码进行sonar规范性检查，最后对系统进行压力测试及浏览器兼容性测试，确认系统性能需求。

## 5.1 测试环境

本系统测试环境部署：测试服务器4台，通过nginx负载均衡将请求打到每台服务器上。服务器使用Linux系统。

本地测试机采用目前市场上常见的windows 7系统，PC笔记本，8G内存。根据需求文档人工执行预先设定的case，浏览器兼容性测试覆盖Google chrome、火狐、360极速、IE7、IE8、IE9等主流浏览器。

本系统压力测试采用公司统一的Siege，用于测试多线程HTTP请求的负载情况。代码规范采用sonar进行检查，确保没有阻断和严重的规范问题上线。

## 5.2 功能性测试

本系统的功能测试部分分为接口测试和系统测试。接口测试的目的是检测对外提供服务及调用的接口是否做了安全性考虑，防止Sql注入、参数越界、越权访问的现象的发生[48]。系统测试的目的是根据需求文档预先对系统的功能要求，人工执行case，确保需求中所提出的系统功能实现完整。最后用sonar进行代码检查，解决代码规范问题。

5.2.1接口测试

通过模拟用户使用系统时的操作对系统的场景调用，对系统提供服务的接口进行测试。此时不需要输入正确的入参，也不考虑系统的真实逻辑，因为用户的使用具有随机性，因此我们单纯根据接口定义的参数设计用例，对接口进行测试。

通过公司统一mock接口测试平台，调用外部接口可能产生异常情况，因此，

mock异常情况对系统进行测试，需要更改服务器的host，如图5-1所示：

?

图5-1 服务器host修改

Mock平台操作页面如图5-2所示。

?

图5-2 Mock平台模拟接口调用

?

图5-3 接口调用异常时，后台日志显示错误信息

有时系统会受到黑客的非法访问入侵，因此要考虑SQL注入等漏洞。用户可以通过拼接HTTP非法请求，越权访问数据库。这些都需要进行测试。输入框的通常测试自动过滤中英文空格、大小写检查、特殊字符串验证(~@#$%^&\*() \_+| {}

[]: ；'" /？《》<>），类型验证、边界值验证、超长字符验证、null或NULL的检查。输入次数的限制、敏感词验证（显示无结果）、密码密文显示且存库后要加密。字符串首尾包含空格的验证、脚本录入检查(<br>< script> alert(/xss/) </script>)[49]。

后台日志会提示访问异常，系统异常日志如图5-3所示。系统页面提示用户填写错

误，如图5-4所示。

?

图5-4 填写错误时，页面弹框提示内容错误

为防止用户知晓别人的订单号进行越权查询，拼接URL访问他人订单，如：

[http: //kmair. qunar. com/kmair/viewOrderorderNo=493637150427](http://kmair.qunar.com/kmair/viewOrder?orderNo=493637150427)，系统提示订单非本人订单。如图5-5所示。

?

5.2.2系统功能测试

图5-5 越权访问弹框提示语

系统功能测试使用预定的测试用例进行测试。测试用例如表5-1至5-9所示。

表5-1 验证航空公司退改签可查询的一次退款测试用例

| 用例编号 | 1.1 |
| --- | --- |
| 用例描述 | 验证航空公司退改签可查询的一次退款 |
| 用例目的 | 验证系统可以正确获得航空公司退改签规则 |
| 用例 | 1. 首先在 beta 环境下一张国内单程机票订单。乘机人姓名使用白名单  服务平台为用户出票，票号采用万能票号  用户端提出退款申请，退款原因选择自愿退款  2. 系统调用查询航空公司退改签接口，查询这张机票真实的退改签规则  3. 服务平台操作完整退票流程 |
| 预期结果 | 系统能够获取此刻航空公司退改签规则 |
| 测试结果 | 页面显示航空公司退改手续费金额，后台日志显示退改签详细退规则，可以手动计算，检查页面展示的金额是否按获取的规则计算。 |
| 状态 | 通过 |

表5-2 验证儿童订单手续费无法查询测试用例

| 用例编号 | 1.2 |
| --- | --- |
| 用例描述 | 验证含有儿童的订单，航空公司手续费无法查询情况 |
| 用例目的 | 验证系统可以因订单内有儿童票，无法获得航空公司退改签规则 |
| 用例 | 1. 首先在 beta 环境生成一张国内单程机票订单。  乘机人姓名使用白名单，订单内一名成人，一名儿童服务平台为用户出票，票号采用万能票号  用户端提出退款申请，退款原因选择自愿退款  2. 由于订单中有儿童票，因此系统不调用查询航空公司退改签接口查询这张机票真实的退改签规则  3. 服务平台操作完整退票流程 |
| 预期结果 | 因儿童票使用规则特殊，系统不获取航空公司退改签规则 |
| 测试结果 | 页面显示代理商退改手续费金额，整张订单成人儿童均使用代理商规则进行退款，后台日志显示退改签详细规则，可以手动计算，检查页面展示的金额是否按获取的规则计算。 |
| 状态 | 通过 |

表5-3 验证拆单测试用例

| 用例编号 | 1.3 |
| --- | --- |
| 用例描述 | 验证订单拆单后，儿童不可单独乘机 |
| 用例目的 | 验证将原有订单拆单后，乘机人只剩儿童，业务要求儿童不可单独乘机， 因此成人票不可退改签 |
| 用例 | 1. 首先在 beta 环境生成一张国内单程机票订单。  乘机人姓名使用白名单，订单内一名成人，一名儿童服务平台为用户出票，票号采用万能票号  用户端申请成人退款，理由选择自愿退款，乘机人保留一名儿童  2. 业务要求儿童不可单独乘机，因此系统不调用查询退改签规则的接口，成人不进行退票  3. 服务平台操作完整退票流程 |
| 预期结果 | 系统不调用查询退改签规则的接口，成人不进行退票 |
| 测试结果 | 页面提示儿童不可单独乘机弹框提示语 |
| 状态 | 通过 |

表5-4 验证未出票申请退款测试用例

| 用例编号 | 1.4 |
| --- | --- |
| 用例描述 | 验证未出票申请退款 |
| 用例目的 | 验证系统为未出票申请退款的订单全额退款 |

|  |  |
| --- | --- |
|  | （续表 5-4） |
| 用例 | 1. 首先在 beta 环境下一张国内单程机票订单。乘机人姓名使用白名单  系统不需要在服务平台出票，用户端显示订单状态为“支付成功等待出票”用户端提出退款申请，退款原因选择自愿退款  2. 系统为这张订单全额退款  3. 服务平台操作完整退票流程 |
| 预期结果 | 系统查询订单金额，不获取航空公司或代理商退改签规则 |
| 测试结果 | 页面显示全额退款金额，后台日志显示金额明细，可以手动计算，检查页面展示的金额是否正确。 |
| 状态 | 通过 |

表5-5 验证二次退款金额上限测试用例

| 用例编号 | 1.5 |
| --- | --- |
| 用例描述 | 验证二次退款申请金额上限 |
| 用例目的 | 验证系统可以计算二次退款上限金额 |
| 用例 | 1. 首先在 beta 环境下一张国内单程机票订单。乘机人姓名使用白名单  服务平台为用户出票，票号采用万能票号  用户端提出退款申请，退款原因选择自愿退款  2. 系统调用查询航空公司退改签接口，查询这张机票真实的退改签规则  3. 服务平台操作完整退票流程  4. 退票后服务平台再次发起二次退款申请，申请金额上限=订单全额-已退款金额，代理商可以手动填入金额 |
| 预期结果 | 系统能够计算出退款上限，填入金额后可完成再次退款 |
| 测试结果 | 页面显示退款金额上限及计算方式，可以手动计算，检查页面展示的金额是否按获取的规则计算。 |
| 状态 | 通过 |

表5-6 验证多次退款测试用例

| 用例编号 | 1.6 |
| --- | --- |
| 用例描述 | 验证多次退款退至乘机人银行卡 |
| 用例目的 | 验证系统核实乘机人信息的功能是否正确 |
| 用例 | 1. 首先在 beta 环境下一张国内单程机票订单。乘机人姓名使用白名单  服务平台为用户出票，票号采用万能票号  用户端提出退款申请，退款原因选择自愿退款  2. 系统调用查询航空公司退改签接口，查询这张机票真实的退改签规则  3. 服务平台操作完整退票流程  4. 退票后服务平台再次发起二次退款申请，申请金额上限=订单全额-已退款金额。同时需要填入本张订单退款的乘机人银行卡号，退款将打入本张银行卡中 |

|  |  |
| --- | --- |
|  | （续表 5-6） |
| 预期结果 | 系统准确将退款金额打入到用户填写的银行卡中 |
| 测试结果 | 页面显示填写乘机人姓名及银行卡的区域，填写错误时有弹框提示，信息提交至用户中心进行核实，核实准确返回正常，退款打入填写的银行卡中。 |
| 状态 | 通过 |

二次退款处理页面如图5-6所示。

?

图5-6 二次退款页面展示

表5-7 验证往返订单测试用例

| 用例编号 | 1.7 |
| --- | --- |
| 用例描述 | 验证往返订单 |
| 用例目的 | 验证系统可以正确获得航空公司退改签规则 |
| 用例 | 1. 首先在 beta 环境下一张国内往返机票订单。乘机人姓名使用白名单  服务平台为用户出票，往返票号均采用万能票号用户端提出退款申请，退款原因选择自愿退款  2. 系统调用查询航空公司退改签接口，查询两张机票退改签规则  3. 服务平台操作完整退票流程 |
| 预期结果 | 系统能够获取此刻航空公司退改签规则 |

|  |  |
| --- | --- |
|  | （续表 5-7） |
| 测试结果 | 页面显示航空公司退改手续费金额，后台日志显示退改签详细退则，可以手动计算，检查页面展示的金额是否按获取的规则计算。 |
| 状态 | 通过 |

表5-8 验证订单中多乘机人测试用例

| 用例编号 | 1.8 |
| --- | --- |
| 用例描述 | 验证一次退款订单中多乘机人情况 |
| 用例目的 | 验证系统可以正确获得航空公司退改签规则 |
| 用例 | 1. 首先在 beta 环境订国内单程机票订单。乘机人姓名使用白名单，多成人乘机  服务平台为用户出票，票号采用多种仓位票号用户端提出退款申请，退款原因选择自愿退款  2. 系统调用查询航空公司退改签接口，查询这张订单内每张机票的退改签规则  3. 服务平台操作完整退票流程 |
| 预期结果 | 系统能够获取此刻航空公司退改签规则 |
| 测试结果 | 页面显示航空公司退改手续费金额，后台日志显示退改签详细退则，可以手动计算，检查页面展示的金额是否按获取的规则计算。 |
| 状态 | 通过 |

表5-9 验证航空公司可查询的测试用例

| 用例编号 | 1.1 |
| --- | --- |
| 用例描述 | 验证航空公司退改签可查询的一次退款 |
| 用例目的 | 验证系统的可以正确获得航空公司退改签规则 |
| 用例 | 1. 首先在 beta 环境下一张国内单程机票订单。乘机人姓名使用白名单  服务平台为用户出票，票号采用万能票号  用户端提出退款申请，退款原因选择自愿退款  2. 系统调用查询航空公司退改签接口，查询这张机票真实的退改签规则  3. 服务平台操作完整退票流程 |
| 预期结果 | 系统能够获取此刻航空公司退改签规则 |
| 测试结果 | 页面显示航空公司退改手续费金额，后台日志显示退改签详细退则，可以手动计算，检查页面展示的金额是否按获取的规则计算。 |
| 状态 | 通过 |

### 5.2.3 sonar检查

通过公司统一的代码规范工具sonar进行检查，Sonar是一个开源平台，用于管理源代码的质量。常见集成工具只能够把FindBugs等检查结果展示出来，但

Sonar能够通过插件对检查结果进行整合处理，代码进行量化检查，对不同类型的

工程代码都可以起到检查作用。同时Sonar检查不但可以支持IDE，还可以支持公司的开发工具IntelliJ IDEA，本系统的开发过程就是在IntelliJ IDEA中完成，Sonar也可以联机查看代码检查结果[50]。本系统部分代码sonar检查结果如图5-7所示。

?

图5-7 sonar检查结果图

其中，包耦合指数（Package tangle index）是检测包的耦合性，Java中要求低耦合，因此耦合度越低证明代码间耦合关系越松散，越有利于后期维护，本系统代码耦合度7.3%。该指数计算方式：2 \* (package\_tangles / package\_edges\_weight) \*

100, 其中：package\_edges\_weight =包之间的文件依赖总数，package\_tangles =可以去除文件依赖的数量。

## 5.3 系统性能测试

5.3.1压力测试

对系统进行压力测试时，根据系统需求初期规划的性能指标进行测试，对系统不断增加压力，在系统资源允许的情况下，验证系统可以达到预期的性能要求。公司对压力测试环境设定：

(1)测试环境与线上环境尽量保持一致，这样可以最大限度保证beta环境的测试对线上有相同的验证效果。

(2)请求发起方和服务方不在同一台机器，避免相互影响。

(3)尽量不跨机房，确保测试环境稳定趋于线上环境。

测试环境设置：参数并发20个，每次增加4个，循环6次，根据吞吐量，响应时间，资源消耗，判断系统综合性能指标是否合格。耗时检测结果如图5-8所示。

?

图5-8 耗时监控

根据真实业务场景，将负载由20%加压至80%，监控对系统平均负载情况

（Load Average）如图5-9所示。系统CPU使用率（CPU Usage）由5%升至50%，负载情况如图5-10所示。TCP连接数（TCP Connections）由10/秒增加到60/秒，负载情况如图5-11所示。内存使用率（Memory Usage）由2G升至6G，系统监控如图5-12所示。

?

图5-9 平均负载（Load Average）监控图

?

图5-10 CPU使用率（CPU Usage）监控图

?

?

图5-11 TCP连接数（TCP Connections）监控图

图5-12 内存使用率（Memory Usage）监控图

系统上线运行需要考虑网络负载情况，虽然退改签模块不会有常见的集中并发的请况出现，但是也要考虑在一定负载下，系统能够正常给予相应。因此采用

Siege进行轻量级的负载测试。Siege可以对WEB站点进行压力检测，模拟多用户请求，设置用户数量，重复次数等值，在单位时间下对系统进行高频访问，结果输出访问成功次数，测试用时等指标，可根据实际情况判断是否满足系统需求[51]。

系统指标参数说明如表5-10所示。

表5-10 Siege参数说明表

| 输出名称 | 解释说明 |
| --- | --- |
| Transactions: | 访问次数 |
| Availability: | 成功比例 |
| Elapsed time: | 测试用时 |
| Data transferred: | 测试传输数据量 |
| Response time: | 平均响应时间 |

|  |  |
| --- | --- |
|  | （续表 5-10） |
| 输出名称 | 解释说明 |
| Transaction rate: | 每秒事务处理量 |
| Throughput: | 吞吐率 |
| Concurrency: | 并发用户数 |
| Successful transactions: | 成功传输次数 |
| Failed transactions: | 失败传输次数 |
| Longest transaction: | 最长响应时间 |
| Shortest transaction: | 最短响应时间 |

根据真实业务场景，退改签日申请量1000单，真实最大并发量小于10/秒，

考虑个别特殊情况，测试设置最大并发量40单/秒~50单/秒之间，Siege负载压力测试设置结果图如图5-13所示。请求成功率100%，响应时间1.29秒。

?

5.3.2浏览器兼容性测试

图5-13 Siege负载压力结果图

本系统提供给用户和代理商使用，不确定用户会使用什么浏览器进行操作，由于浏览器内核的差异，浏览器对页面的展示也存在差异。所以前端设计要考虑到各种主流浏览器的解析方式，保证用户使用时不出现变形等不兼容错误。

浏览器内核分析：Trident内核：IE最先开发或使用的，也称IE内核。Webkit内核：谷歌[chrome浏览器](http://zhidao.baidu.com/search?word=chrome%E6%B5%8F%E8%A7%88%E5%99%A8&amp;fr=qb_search_exp&amp;ie=utf8)最先开发并且投入使用，也成为谷歌内核。Presto内核：目前只有[Opera浏览器](http://zhidao.baidu.com/search?word=Opera%E6%B5%8F%E8%A7%88%E5%99%A8&amp;fr=qb_search_exp&amp;ie=utf8)采用该内核[52]。

浏览器测试内容：

#### 1） 页面样式

要进行页面缩放（75%-400%），做到样式不乱功能正常。开发时chrome浏览器，测试时优先使用IE6、IE7和火狐浏览器缩放快捷键：Ctrl+鼠标滚轮观察页面内容。页面样式不变形，页面文字无错别字，语法正常，描述准确。

#### 2） 页面功能

IE6、IE7、IE8对CSS的解析方式不一样，因此会导致生成的页面展示效果不一样，因此对IE6,7单独进行处理。IE6处理如图5-14所示。

?

图5-14 IE6页面特殊处理

报文压缩：大于1KB的response报文都需要压缩；使用HttpWatch可以看到每个请求和返回的报文大小。HttpWatch抓取数据包结果如图5-15所示。

?

图5-15 HttpWatch抓取数据包结果图

将资源或接口等数据提供方的host指向调整成invalid ip，测试中为了不影响其他功能，ip指向另一个环境的host，页面正常填充不显示部分整个页面加载时间<5s，不影响用户感受。服务器的host指向地址修改如图5-16所示。

?

图5-16 服务器的host指向地址修改

接口的敏感信息包含：用户名、密码、电话号码、订单号、身份证号、银行卡等信息，需要进行加密或者特殊处理。有敏感信息的接口要设置白名单，只允许固定机器进行访问。为防止信息别删改通常会对全部内容进行MD5加密，服务端通过解密MD5值后与出入的值进行比对验证。例如：传输时将日期、金额、酒店进行加密，服务端接到请求后再次进行加密判断是否有修改。订单详情页源码中不能出现keywrods和description避免搜索引擎抓到。将订单详情页URL地址在其他浏览器中打开，要进行验证。验证结果如图5-17,5-18所示。

?

图5-17 乘机人身份信息加密

?

5.4本章小结

图5-18 联系人手机号加密

本章介绍了测试环境，及其运行过程中所需参数的意义。然后对本模块进行了功能和性能方面的测试。从功能上面来说，主要是接口测试与系统整体功能测试2个方面的测试。通过对基本的测试用例进行测试以及测试完成的结果来验证功能需求中的基本功能得到了实现。最后用压力测试工具对系统进行了压力测试，系统达到了公司系统上线的性能要求。同时对浏览器兼容性做了验证，确保系统上线后用户在各主流浏览器上使用正常。

结论

本论文对机票退改签系统和消息队列子系统的结合给出了解决方案，完成了项目的需求分析、概要设计、详细设计、代码实现以及系统的部署上线测试。系统通过了功能性测试以及性能测试，能够正常稳定运行，满足了用户线上使用的需求。

本文主要的工作内容有：

（1）设计和实现了机票退改签系统。机票按票号去航空公司黑屏查询此票号状态，给出退改签金额计算结果。建立NFD退改签缓存库，以及匹配逻辑；实现

NFN匹配、解析逻辑；建立NFD退票规则管理（人工维护规则部分）；建立NFN

未解析列表；将航空公司退改签规则修改为本公司的票维度退改签规则。

（2）提出了可信用户计算方法，根据用户在系统中的使用操作流程判断用户可信度，在系统中加入快速退款的业务逻辑。对于高诚信用户，提交退款后立即将金额退回，缩短自动退款时间，能够提升用户体验，减少代理商人力成本。

（3）消息队列的实现解决了系统间各模块的调用问题，使得退改消息准确发送及接收，包括与用户中心、支付中心的协作关系，都使用到消息队列进行传递信息。消息队列的设计和实现解决了在复杂网络中数据无法保证准确到达的问题，使系统的安全性可靠性达到了一定的标准。在上线验证后也发现，使用消息队列的模块消息转达正确性高于未使用消息队列的功能模块。

本系统解决了原有线上金额不准确或丢失等问题。提高了用户体验，也保证了代理商、航空公司、用户等多方的利益。本系统的上线使原本复杂的人工查询金额操作变得简易，系统自动查询到的退改规则和业务逻辑，保证了金额计算的准确，同时也降低了用户投诉率，减少了运营人员的工作成本，大幅度提高了退改部门对订单的处理能力。

综上所述，本系统较为良好的完成了本系统的业务需求功能，但是本系统还有以下需要改进和完善的地方，以适应更复杂的业务逻辑。解析模板文案的方式较为简单，当航空公司有新的模板产生时，系统不能够及时变更，仍需要运营人员手动处理，这成为了系统大规模推广的瓶颈，因此接下来应当完善模板解析的方法，使其能够适应更多更复杂的文案解析。消息队列解决了系统信息传输不安全因素，但健壮性和稳定性仍有待提高，后续应当加强消息队列的稳定性设计，相信使用消息队列的开端能够引领整体系统走向稳定的发展。

参考文献

[[1] 王强](http://search.cnki.com.cn/Search.aspx?q=author%3A%E7%8E%8B%E5%BC%BA). 关于机票代理行业规范化管理及业务发展趋势的调研[D]. 西南财经大学, 2010: 8-11.

[2] 勾志鹏. 机票分销企业的电子商务化研究[D]. 北京交通大学, 2010: 7-9.

[[3] 刘晓华](http://social.wanfangdata.com.cn/Locate.ashx?ArticleId=Y2201632&amp;Name=%e5%88%98%e6%99%93%e5%8d%8e). L航空公司机票销售代理费用管理的研究[D]. [北京工业大学](http://s.wanfangdata.com.cn/Paper.aspx?f=detail&amp;q=School%3A%e5%8c%97%e4%ba%ac%e5%b7%a5%e4%b8%9a%e5%a4%a7%e5%ad%a6%2BDBID%3AWF_XW), 2012: 13-18.

[[4] 倪正卿](http://social.wanfangdata.com.cn/Locate.ashx?ArticleId=Y1846148&amp;Name=%e5%80%aa%e6%ad%a3%e5%8d%bf). 东航B2B机票销售系统的设计与实现[D]. [华东师范大学](http://s.wanfangdata.com.cn/Paper.aspx?f=detail&amp;q=School%3A%e5%8d%8e%e4%b8%9c%e5%b8%88%e8%8c%83%e5%a4%a7%e5%ad%a6%2BDBID%3AWF_XW), 2010: 11-13.

[5] 丁欣. 东方航空公司机票营销渠道优化策略研究[D]. 复旦大学, 2013: 22-25.

[[6] 肖伯春](http://search.cnki.com.cn/Search.aspx?q=author%3A%E8%82%96%E4%BC%AF%E6%98%A5), [彭杰](http://search.cnki.com.cn/Search.aspx?q=author%3A%E5%BD%AD%E6%9D%B0). 中国民航客运机票定价策略研究[J]. 四川大学学报, 2000(6): 11-13.

[[7] 王星月](http://search.cnki.com.cn/Search.aspx?q=author%3A%E7%8E%8B%E6%98%9F%E6%9C%88). SCHK公司大客户系统的设计与实现[D][. 电子科技大学, 2013:](http://cdmd.cnki.com.cn/Area/CDMDUnitArticle-10614-2013-1.htm) 19-22.

[8] 陈振坤. 民用航空器基于案例推理的故障诊断系统设计[D]. 上海交通大学, 2011: 25-31.

[[9] 韩璐](http://search.cnki.com.cn/Search.aspx?q=author%3A%E9%9F%A9%E7%92%90). 中国机票销售市场纵向关系研究[D]. [东北财经大学， 2012](http://cdmd.cnki.com.cn/Area/CDMDUnitArticle-10173-2012-1.htm): 13-19.

[10] [刘洋妹](http://social.wanfangdata.com.cn/Locate.ashx?ArticleId=Y1757426&amp;Name=%e5%88%98%e6%b4%8b%e5%a6%b9). 国旅在线机票分销系统的设计与实现[D]. 北京邮电大学, 2010: 17-22.

[11] 赵晓峰. 基于消息队列的Web 应用异步通信的设计与实现[J]. 计算机工程, 2011, 11(6): 23-29.

[12] 臧冬松, Vincent Garonne, 孙功量. 一种大规模分布式应用性能分析系统[J]. [计算机工程](http://c.wanfangdata.com.cn/Periodical-jsjgc.aspx), [2012, 38(24)](http://c.wanfangdata.com.cn/periodical/jsjgc/2012-24.aspx): 12-16.

[13] AbadiD J, Carney D. Aurora: A New Model and Architecture for Data Stream Management[J]. The VLDB Journal, 2003, 12(2): 120-139.

[14] 李立宏, 李浩昱, 张福恩. 消息中间件的设计与实现[J]. 计算机工程, 2000, 21(01): 17-19.

[15] 何红波, 陆建德, 丁卫平. 基于中间件MQSeries的消息队列异步通信服务的应用研究[J]. 南通大学学报(自然科学版), 2006, 33(04): 23-25.

[16] 周泽华, 黄涛, 李京. 消息中间件管理器的设计和实现[J]. 计算机研究与发展, 2002, 39(3): 56-59.

[17] 尤天舒. 基于Agent的集群负载均衡模型及其实验研究[D]. 吉林大学, 2014: 35-39.

[18] Wolfgang Emmerich. software Engineer and middleware[J]. 2000, 19(03): 117-129.

[19] J. Charles. Middleware Moves to the Forefront[J]. IEEE Computer, 1999, 8(2):

17-19.

[20] Vinoski S. Advanced Message Queuing Protocol[D]. IEEE Internet Computing, 2006, 31(07): 14-21.

[21] 黄姝娟, 杜承烈, 尤涛. 中间件技术实时性能的比较[J]. 计算机工程, 2009, 35(11): 32- 37.

[22] S Zhou. LSF: Load Sharing in Large Heterogeneous Distributed Systems[J] Workshop on Cluster Computing, 2009, 29(06): 122-125.

[23] Willebeek- LeMair M H, Reeves A P. Strategies for dynamic 1oad balancingon highly paraUd computer[J]. IEEETransactionsonParallel and Distributed System, 2005, 4(9): 979-993.

[24] 王传胜, 李乔儒. 基于JMS的消息服务的研究与开发[J]. 计算机工程与设计, 2005, 26(12): 3423-3426.

[25] Feiteison D G. Met. tic and Workload Effects on Computer Systems[J] Evaluation Computer, 2003, 36(9): 18-25.

[26] KRU IJFM D, SANKARAL INGAM K. MapReduce for the cell B. E. architecture, CS2TR2200721625 [J]. U2niversity ofWisconsin2Madison, 2007, 38(12): 11-13.

[27] 曾东海, 刘海, 金士尧. 集群负载调度算法性能评价[J]. 计算机工程, 2006, 32(11): 78-79.

[28] 王旭新. 消息中间件研究与实现[D]. 南京理工大学, 2005: 45-53.

[29] PapazoglouM P, C, eorgakopodosD. Service-OrientedCompufing[J]. Service Oriente Computing Communications of the ACM, 2003, 46(10): 25-28.

[30] 徐晶, 许炜. 消息中间件综述[J]. 计算机工程, 2005, 31(16): 73- 76.

[31] 杜彭, 赵荣彩, 董春丽. MPI通信代码自动生成算法[J]. 计算机应用, 2007, 27( 3): 759- 761.

[32] 王莉莉. 基于XML 的消息中间件的研究及实现[D]. 华北电力大学, 2007: 51-55.

[33] CHEN Xing, LIU Zhao, HUANG Gang. A method of the long distance deployment of Middleware[J], Computer Science, 2008, 109(27): 19-31.

[34] ZHOU Minghui, WANG Qianxiang, JIAO Wenpin, MEI Hong. A perspective on evolution of Middleware technology supporting Internetware[J], Journal of Frontiers of Computer Science an Technology, 2008, 112(16): 98-104.

[35] 景晓玺, 葛玮, 郝克刚. 基于JMS的企业即时通讯系统的设计与实现[J]. 计算机应用与软件, 2009, 26(3): 22-25.

[36] 徐俊杰. 消息中间件在数据仓库中的应用[D]. 复旦大学, 2005: 63-66.

[37] 文杰. 支持信息安全传输的消息中间件研究及实现[D]. 西北工业大学, 2006: 31-37.

[38] 康汶, 何耀光, 詹先信, 邓伦强. 跨部门数据共享平台总体架构与功能分析[J]. 计算机与现代化, 2011, 14(3): 43-48.

[39] [王德民](http://epub.cnki.net/kns/popup/knetsearchNew.aspx?sdb=CJFQ&amp;sfield=%e4%bd%9c%e8%80%85&amp;skey=%e7%8e%8b%e5%be%b7%e6%b0%91&amp;scode), [何立东](http://epub.cnki.net/kns/popup/knetsearchNew.aspx?sdb=CJFQ&amp;sfield=%e4%bd%9c%e8%80%85&amp;skey=%e4%bd%95%e7%ab%8b%e4%b8%9c&amp;scode), [刘菲菲](http://epub.cnki.net/kns/popup/knetsearchNew.aspx?sdb=CJFQ&amp;sfield=%e4%bd%9c%e8%80%85&amp;skey=%e5%88%98%e8%8f%b2%e8%8f%b2&amp;scode), [苏男](http://epub.cnki.net/kns/popup/knetsearchNew.aspx?sdb=CJFQ&amp;sfield=%e4%bd%9c%e8%80%85&amp;skey=%e8%8b%8f%e7%94%b7&amp;scode). [基于消息的加权负载均衡算法](http://epub.cnki.net/kns/detail/detail.aspx?QueryID=2&amp;CurRec=2&amp;recid&amp;FileName=JLGY201201025&amp;DbName=CJFD2012&amp;DbCode=CJFQ&amp;pr)[J]. 吉林大学学报, 2012, 19(01): 15-17.

[40] [蒋鹏](http://epub.cnki.net/kns/popup/knetsearchNew.aspx?sdb=CJFQ&amp;sfield=%e4%bd%9c%e8%80%85&amp;skey=%e8%92%8b%e9%b9%8f&amp;scode), [黄声勇](http://epub.cnki.net/kns/popup/knetsearchNew.aspx?sdb=CJFQ&amp;sfield=%e4%bd%9c%e8%80%85&amp;skey=%e9%bb%84%e5%a3%b0%e5%8b%87&amp;scode). [虚拟多端口的短信分发系统设计与实现](http://epub.cnki.net/kns/detail/detail.aspx?QueryID=0&amp;CurRec=15&amp;recid&amp;FileName=WJFZ201412052&amp;DbName=CJFDLAST2015&amp;DbCode=CJFQ&amp;pr)[J]. [计算机技术与发展](http://epub.cnki.net/kns/Navi/ScdbBridge.aspx?DBCode=CJFD&amp;BaseID=WJFZ&amp;UnitCode&amp;NaviLink=%e8%ae%a1%e7%ae%97%e6%9c%ba%e6%8a%80%e6%9c%af%e4%b8%8e%e5%8f%91%e5%b1%95), 2014, 43(12): 21-23.

[41] Piqtek M. Real-Time Application Interface and Xenomai modified Linux real-time operating systems dedicated to control[J]. Computer Methods and Systems, 2007, 89(11): 24-30.

[42] Fitzpatrick B. Distributed caching with memcache[J]. Linux journal, 2004, 124(07): 5.

[43] 陈祖元, 基于消息队列的某异构数据交互系统的设计与实现[D]. 中国科学院, 2014: 68-79.

[44] 于彩荣. 基于XML的消息队列中间件的设计与实现[D]. 西北工业大学, 2004: 44-51.

[45] Ning Ma, Zhonghai Lu, Liong Zheng et al. System design of full HD MVC decoding on mesh-based multicore NoCs[J]. Microprocessors and microsystems, 2011, 35(2): 217-229.

[46] Fang Miao, Ken Chen, Wenhui Yang, Xirong Guo. Research on a New Digital Earth Platform Service Mechanism Based on G/S Model[J]. 7th International Symposium on Digital Earth Secretariat, 2011, 37(2): 210-217..

[47] CHEN Bingxing, QIU Baozhi. Construction technology of cluster message-oriented middleware[J], Application Research of Computers 2012, 26(05): 47-48.

[48] 杜彭, 赵荣彩, 董春丽. MPI通信代码自动生成算法[J]. 计算机应用, 2007, 27(3): 759- 761.

[49] 黄姝娟, 杜承烈, 尤涛. 中间件技术实时性能的比较[J]. 计算机工程, 2009, 35(11): 32- 37.

[50] 沈雁. 基于消息队列的分布式图像处理系统的设计与实现[D]. 浙江大学, 2013: 11-12.

[51] 朱少民. 敏捷测试的方法和实践[J]. 程序员, 2010, 23(10): 22-24.

### 哈尔滨工业大学学位论文原创性声明及授权使用说明

学位论文原创性声明

本人郑重声明：此处所提交的学位论文《机票退改签系统的设计与实现》，是本人在导师指导下，在哈尔滨工业大学攻读学位期间独立进行研究工作所取得的成果。据本人所知，论文中除已注明部分外不包含他人已发表或撰写过的研究成果。对本文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明。本声明的法律结果将完全由本人承担。

?

学位论文使用权限

学位论文是研究生在哈尔滨工业大学攻读学位期间完成的成果，知识产权归属哈尔滨工业大学。学位论文的使用权限如下：

（1）学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文，并向国家图书馆报送学位论文；（2）学校可以将学位论文部分或全部内容编入有关数据库进行检索和提供相应阅览服务；（3）研究生毕业后发表与此学位论文研究成果相关的学术论文和其他成果时，应征得导师同意，且第一署名单位为哈尔滨工业大学。

保密论文在保密期内遵守有关保密规定，解密后适用于此使用权限规定。本人知悉学位论文的使用权限，并将遵守有关规定。

?

致谢

值此论文完成之际，向无私的老师，亲爱的父母，可爱的同学以及实习的同事们致以最诚挚的谢意。

短暂的硕士阶段学习生活即将结束，短短两年很快过去，但在这两年内我学到了很多，能够顺利完成硕士阶段的学习，取得今天的成绩，离不开老师的精心指导，学校的栽培，离不开同学同事们的帮助和支持，以及父母的关爱。

感谢我可敬的父母，在我的学习和生活上给予一贯的支持，是你们给了我鼓励，给了我信心，鼓舞我不断地前进，在迷茫的时刻告诉我坚持下去就会有曙光在前方，在欣喜的时刻告诉我要继续踏实前行才能走得更远。感谢父母多年来的关怀和支持，使我有了今天的成绩。

感谢哈尔滨工业大学对我的培养，特别要感谢我的导师王宏志副教授的指导，王老师在学习和实习中都给予我很大的帮助，在我的硕士论文写作与课题研究中提出了很多指导性的建议，王老师的指导是我论文完成过程中最有力的支持。

感谢实习单位的同事，是你们教会了我如何工作，如何在工作中与其他同事沟通，如何把控项目的进度和发展。同时，在实习过程中认识了很多努力认真的人，知道每个人都有他闪光的一面值得我学习。感谢我的校外指导老师、我的

teamleader，无论从生活上还是工作上都帮助了我很多，使我快速地适应公司环境并做出贡献。

衷心地感谢我的老师与同学们的一路相伴，感谢各位！

### 个人简历

2007年9月考入哈尔滨工程大学计算机科学与技术专业。

2011年7月毕业获得工学学士学位。

2013年9月考入哈尔滨工业大学软件学院攻读硕士学位。

2014年6月至2015年5月在北京趣拿软件科技有限公司机票事业部实习。