民航售票系统架构设计文档

架构设计文档版本

|  |  |  |
| --- | --- | --- |
| 日期 | 版本号 | 说明 |
| 2016-10-10 | 1.0 | 最终版本 |

**目录**

1. 概述------------------------------------------------------2
2. 目标------------------------------------------------------2
3. 需求分析--------------------------------------------------2
   1. 功能性需求---------------------------------------------2
   2. 用例图-------------------------------------------------4
   3. 非功能性需求-------------------------------------------4
   4. 质量场景及策略选择-------------------------------------5
4. 系统设计--------------------------------------------------6

4.1细化用例-----------------------------------------------6

4.2模块划分-----------------------------------------------9

4.3数据表设计---------------------------------------------9

4.4类图---------------------------------------------------11

1. 体系结构--------------------------------------------------14
2. 参考资料--------------------------------------------------14
3. **概述**

本文档为民航售票系统的架构设计文档，在文档中对民航售票系统做了系统的分析，确定了其基本的功能，根据要实现的质量属性采取了相应的架构策略，旨在指导项目的开发过程。

此文档的读者对象：

需求分析人员（请参考需求分析部分）

UI设计人员（请参考非功能性需求分析部分）

程序员（请参考系统设计、体系结构部分）

数据库设计人员（请参考系统设计部分）

测试人员（请参考目标部分）

1. **目标**

实现民航售票系统的客户端功能，即提供基本的查票、购票业务

在大的并发条件下，保证系统的响应速度

实现系统的安全性

1. **需求分析**

**3.1功能性需求**

民航售票系统旨在为用户提供网上查询航班、购票的服务，为实现此功能需要实现以下几点基本功能：用户管理、查票、买票及退票。

|  |  |
| --- | --- |
| 功能 | 详细说明 |
| 用户管理 | 1. 用户第一次使用系统时需要进行注册，注册时需要提供必要的身份信息。 2. 用户登录。 3. 订单管理：完成订票的用户可以查看自己订单的详细信息（班次、时间、出票情况等），同时用户也能查看历史订单记录。 |
| 查票 | 用户输入出发地、目的地以及出行日期，系统输出满足上述条件的全部班次以及班次的具体信息（班次的起飞时间，是否有座位剩余等）  用户也可以只选择出发地和目的地来进行查询。 |
| 买票 | 选择某一班次进行购票 |
| 退票 | 在航班起飞前可以进行退票(按具体情况扣取一定的费用) |
| 改签 | 用户根据自身实际情况修改行程 |

**3.2用例图**

**3.3非功能性需求**

* 高并发性：由于售票系统的用户数量大，因此当系统处于访问高峰期时，必须保证每个用户的有效操作（查询、选择）能在2秒的时间内得到响应，如若无法在2秒的时间内响应请求，需要用户等待，则要给出友好的提示信息。
* 可用性：售票系统在相当长得到时间能一直正常的运行，系统平均故障时间尽可能短，不需要经常进行维护，尽可能减少因为维护给用户造成的不利影响
* 安全性：每次进行操作前需要登陆，而且用户进行买票、改签、退票操作时需要进行验证，用户信息不会泄露。
* 灵活性：售票系统要具有可移植性，方便在不同平台上运行，而且模块要便于修改
* 易用性：售票系统是实用性很强的软件系统，因此必须操作简单，方便使用。用户界面风格也要简洁明了，适合不同年龄和身份的人群。

**3.4质量场景及策略选择**

* 性能场景：在系统处于高峰时期，保证登陆的每个顾客所作的选择和查询的响应时间能在2s以内，如果需要等待则给出有友好的提示。系统可以保证以最快速度同时响应多个用户的操作。

采用的策略：限制队列大小和缓冲池战术。

* 可用性场景：在正常的工作时间内，系统必须具有较高的可用性，保证出故障几率最低。出现故障时系统有相应的处理机制，而且处理时间必须较短。

采用的策略：异常检查，故障恢复机制、资源调度机制。

* 安全性场景： 杜绝非法用户试图绕过应用服务器直接连接到数据库服务器的端口上，防止非法窃取注册用户个人息；屏蔽某IP短时间内的大量无意义的访问，以防被挤爆，使正常用户无法使用，保证系统数据的机密性和完整性。

采用的策略：身份验证、权限、数据机密性、验证码、维护完整性。

* 灵活性场景：售票系统可能会在之后推出pc端、手机客户端，因此需要系统具有良好的可移植性与可重用性，减少工作量。

采用的策略： MVC的设计模式，将视图层与业务逻辑层、数据访问层分离。

* 易用性场景：在该系统中，用户希望在运行时能尽快取消某操作使错误的影响降到最低，取消在1秒内发生；要求具有基本电脑操作常识的人，可以根据界面迅速学会使用方法，让熟手用户使用快捷键。

采用的策略： 单独的用户接口、简洁的界面设计。

1. **系统设计**

**4.1细化用例**

用户注册：

|  |  |
| --- | --- |
| 要素 | 说明 |
| 用例名称 | 用户注册 |
| 简要描述 | 用户输入必要信息进行注册 |
| 事件流 | 基本事件流：  1、用户点击“注册“按钮  2、根据提示输入相关信息 |

用户登录：

|  |  |
| --- | --- |
| 要素 | 说明 |
| 用例名称 | 用户登陆 |
| 简要描述 | 用户输入邮箱及密码进行登陆 |
| 事件流 | 基本事件流：  1、用户点击“登陆“按钮，或体统弹出登录框。  2、根据提示输入邮箱及密码。 |

查票：

|  |  |
| --- | --- |
| 要素 | 说明 |
| 用例名称 | 查票 |
| 简要描述 | 用户输入出发地、目的地、日期进行查票 |
| 事件流 | 基本事件流：  1、用户根据提示输入了出发地、目的地、日期（可选）这些信息  2、用户点击“查询“按钮 |

买票：

|  |  |
| --- | --- |
| 要素 | 说明 |
| 用例名称 | 买票 |
| 简要描述 | 用户根据当前票务信息购买相应的票 |
| 事件流 | 基本事件流：  1、用户选择了某一班次  2、用户点击了“购票“按钮 |
| 前置事件 | 用户已经登陆 |

改签：

|  |  |
| --- | --- |
| 要素 | 说明 |
| 用例名称 | 改签 |
| 简要描述 | 用户根据自身需要更改行程 |
| 事件流 | 基本事件流：  1、用户选择了某一订单  2、用户点击“改签“按钮  3、选择改签后的航班起止地、日期 |
| 前置条件 | 用户已经登录，且其至少定了一张票 |
| 后置事件 | 系统给出两次航班之间的差价，根据情况确定要用户补足费用还是向用户退还部分费用 |

退票：

|  |  |
| --- | --- |
| 要素 | 说明 |
| 用例名称 | 退票 |
| 简要描述 | 用户取消订单 |
| 事件流 | 基本事件流：  1、用户选择了某一订单  2、用户点击“退票“按钮 |
| 前置事件 | 用户已经登录，且其至少定了一张票 |
| 后置事件 | 系统判定用户是否可以退票，如能退票则退还相应的金额，若不能则返回不能退票的原因 |

**4.2模块划分**

用户管理

查票

购票

改签

退票

**4.3数据表设计**

用户表：

|  |  |  |
| --- | --- | --- |
| 属性 | 数据类型 | 限制 |
| pwd | varchar2(20) | not null |
| username | varchar2(20) | not null |
| userID | varchar2(20) | unique,not null |
| email | varchar2(30) | primary key |
| phonenum | varchar2(11) | not null |

航班信息表：

|  |  |  |
| --- | --- | --- |
| 属性 | 数据类型 | 限制 |
| planenum | varchar2(20) | not null |
| planetime | varchar2(20) | not null |
| starting | varchar2(20) | not null |
| ending | varchar2(20) | not null |
| consumetime | varchar2(30) | not null |
| price | varchar2(10) | not null |
| planedate | varchar2(30) | not null |

备注：planenum,planedate,planetime三者联合作为主码

订票表：

|  |  |  |
| --- | --- | --- |
| 属性 | 数据类型 | 限制 |
| ordernum | varchar2(20) | not null |
| email | varchar2(20) | not null |
| userID | varchar2(20) | not null |
| planenum | varchar2(20) | not null |
| planedate | varchar2(30) | not null |
| planetime | varchar2(20) | not null |
| seat | char(20) | not null |

备注：ordernum,userID 作为联合主码

座位信息表：

|  |  |  |
| --- | --- | --- |
| 属性 | 数据类型 | 限制 |
| planenum | varchar2(20) | not null |
| planetime | varchar2(20) | not null |
| planedate | varchar2(30) | not null |
| ticketnum | int | not null |

备注：planenum,planedate,planetime作为联合主码

**4.4类图**

UserDaoImpl

inserUser()

updateUser

deleteUser()

getUser()

User

sex

pwd

userID

email

phonenum

<<interface>>

UserDao

insetUser()

UpdateUser()

deleteUser()

getUser()

PlaneDaoImpl

insetPlane()

UpdatePane()

deletePlane()

getPlanes()

getPlanesInfo()

Plane

planenum

planedate

planetime

consumetime

staring

ending

price

<<interface>>

PlaneDao

insetPlane()

UpdatePane()

deletePlane()

getPlanes()

TicketOrderDaoImpl

insetTicketOrder ()

UpdateTicketOrder ()

deleteTicketOrder ()

getTicketOrders()

TicketOrder

 Ordernum email

userID

planenum

planedate

planetime

seat

<<interface>>

TicketOrderDao

insetTicketOrder ()

UpdateTicketOrder ()

deleteTicketOrder ()

getTicketOrders()

SeatDaoImpl

insetTicketnum()

UpdateTicketnum()

deleteTicketnum()

getTicketnum

subTicketnum()

Seat

planenum

planedate

planetime

ticketnum

<<interface>>

SeatDao

insetTicketnum()

UpdateTicketnum()

deleteTicketnum()

getTicketnum

subTicketnum()

1. **体系架构**

本系统采用基于B/S的分层结构。这种结构有如下特点：节省投资，跨地域广；维护和升级方式简单，如果想对功能进行修改，可以方便地进行修改，大大减少维护成本。

系统结构视图：

Web浏览器

应用服务器

系统采用的设计模式：MVC设计模式

视图层：提供显示界面，利用HTML、CSS、javascript实现

模型层：处理基本的业务逻辑，利用java实现

控制器：处理视图层发来的请求，完成业务流程之间的跳转，利用Servlet、Jsp实现。

1. **参考资料**

*Software Architecture in Practice(第三版)*