**铁路售票系统**

**架构设计文档**

**组长：张付俊**

**组员：孙帅、黄东鹏**

目录

[一、项目背景 2](#_Toc497122444)

[1.概述 2](#_Toc497122445)

[2.目标 2](#_Toc497122446)

[二、需求分析 2](#_Toc497122447)

[1.功能性需求 2](#_Toc497122448)

[2.非功能性需求 3](#_Toc497122449)

[3.系统定义 3](#_Toc497122450)

[4.细化用例 3](#_Toc497122451)

[5.用例图 4](#_Toc497122452)

[三、系统架构设计 8](#_Toc497122453)

[1.体系结构 8](#_Toc497122454)

[2.质量场景与策略选择 9](#_Toc497122455)

[1）性能场景 9](#_Toc497122456)

[2）可用性场景 9](#_Toc497122457)

[3）安全性场景 9](#_Toc497122458)

[4）灵活性场景 9](#_Toc497122459)

[5）易用性场景 9](#_Toc497122460)

[3.类图设计 10](#_Toc497122461)

[4.数据表设计 10](#_Toc497122462)

# 一、项目背景

## 1.概述

现如今，铁路在人们的生活中扮演了很重要的地位。同时，票务的管理也随着票务种类的增加而越来越复杂。早期的人力和纸张管理由于各种不定因素会产生很多信息上的错误，并且安全性也不高，而查找信息却又存在很大不便。

随着计算机网络的发展，一种基于B/S模式的票务系统提出了需求。由于票务管理的繁多与特殊性，票务系统需要极高的安全性，极快的反应速度，同时除了基本的购票、等功能外还要满足响应多点请求等功能。

## 2.目标

在满足售票系统的基本登录、注册、购票、改签、退票、查询等功能的同时，充分考虑其安全性和稳定性。

# 二、需求分析

## 1.功能性需求

|  |  |
| --- | --- |
| 功能 | 说明 |
| 用户信息管理 | 1.对注册的用户进行身份信息认证  2.对注册成功的用户的身份信息进行管理  3.对用户的联系人的信息进行管理 |
| 票务信息管理 | 票务信息添加、删除等操作 |
| 票务查询 | 根据用户对始发站、发车时间、列车类型的选择，对符合条件的车票的余票量、价格，列车的车次、座位类型，途经站地进行查询和显示 |
| 预订购票 | 根据用户的车票选择进行验证、生成相应订单、提交支付操作 |
| 退票 | 支持用户对已经支付的车票进行退订，并根据相关规定扣除手续费后进行退款操作 |
| 改签 | 支持用户对已经支付的车票进行改签，对自己的行程进行修改，并根据相关规定扣除手续费后进行收款或退款操作 |
| 订单查询 | 支持用户对自己的已支付订单，未支付订单，历史订单的信息进行查询 |

## 2.非功能性需求

|  |  |
| --- | --- |
| 质量属性 | 说明 |
| 高并发性 | 由于票务管理的特殊性和复杂性，在购票高峰期（春运、节假日期间等）能够对用户的查询、购票操作进行尽可能快速处理，如果在短时间未能相应用户操作需求，则显示相应信息 |
| 安全性 | 1.对注册用户的身份信息进行验证  2.给管理用户信息的部分信息高权限管理  3. 在web数据库客户端，web服务器和数据库服务器之间应设有防火墙保护，防止网络上的非法数据请求  4.对用户购票进行验证码验证处理，防止机器刷票行为  5.用户修改密码时进行手机验证码验证 |
| 可用性 | 售票系统尽可能24小时正常运作，减少系统故障率，存在故障时减少系统故障时间，尽可能减少维护时间，使用户购票更为便利 |
| 灵活性 | 售票系统应该具有可移植性，可以在不同平台上运行，方便用户以不同形式进行使用 |
| 易用性 | 售票系统面向人群广泛，基数较大，因此要方便使用，并且做到界面简洁易操作，方便各类人群进行使用 |

## 3.系统定义

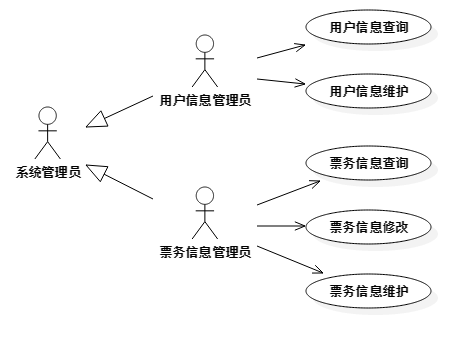
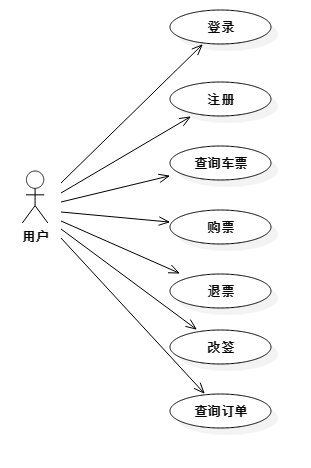
根据功能需求，可得本系统主要面向人群为普通用户和系统管理人员，系统管理人员分为用户信息管理人员和票务信息管理人员。

|  |  |
| --- | --- |
| 系统角色 | 要求 |
| 用户信息管理人员 | 根据信息登录获得系统用户信息管理权限，并可以使用系统对用户的信息进行查询和管理 |
| 票务信息管理人员 | 根据信息登录获得系统票务信息管理权限，并可以对票务信息进行更改，查询和管理 |
| 普通用户 | 根据身份验证进行注册，登录成功后可对票务信息进行查询，进行正常购票，退票，改签操作 |

## 4.细化用例

|  |  |
| --- | --- |
| 用例 | 说明 |
| 注册 | 用户根据身份信息进行注册，对用户的身份信息进行验证，并可以进行学生身份认证 |
| 登录 | 用户注册成功后使用身份信息进行登录 |
| 查询 | 用户根据始发站，发车时间，列车类型对车票余量，价格，车次途经站地，起始时间进行查询 |
| 购票 | 用户选择车票后系统自动生成订单，用户需在30分钟内进行支付，超过30分钟订单取消 |
| 改签 | 用户对支付成功的订单行程进行修改，系统自动生成新的订单，收取相应手续费后进行退款或支付操作 |
| 退票 | 用户对支付成功的订单行程进行删除，系统自动删除相应订单，收取相应手续费后进行退款操作 |
| 用户信息查询 | 用户信息管理员登录获得权限后，可对用户身份信息进行查询 |
| 用户信息维护 | 用户信息管理员可对用户的身份信息进行维护和备份 |
| 票务信息修改 | 票务信息管理员在登录获得权限后，可对票务信息，车次信息，发车时间进行修改 |
| 票务信息查询 | 票务信息管理员可对票务信息，车次信息进行查询 |
| 票务信息维护 | 票务信息管理员在登录获得权限后，可对票务信息进行维护和备份 |

## 5.用例图



|  |  |
| --- | --- |
| 要素 | 说明 |
| 用例名称 | 用户注册 |
| 简要描述 | 用户根据自己真实身份信息进行注册 |
| 事件流 | 基本事件流：  1.用户进入系统注册界面  2.用户填写真实姓名，身份证号，电话，邮箱信息，用户名，密码，确认密码  3.用户提交身份信息  4.该用例结束 |
| 备选事件流 | 用户身份证号，电话，邮箱格式不正确，则  1.弹出错误提示界面  2.返回注册界面，用户更改用户信息后重新提交  用户请求进行学生认证，则  1.进入学生认证界面  2.用户填写学校，学号，乘车区间  3.用户提交学生认证信息 |
| 特殊需求 | 系统对用户提交身份证号，电话，邮箱格式进行检测 |
| 前置条件 | 无 |
| 后置条件 | 系统将信息传给服务器进行身份认证，并返回身份认证结果 |

|  |  |
| --- | --- |
| 要素 | 说明 |
| 用例名称 | 用户登录 |
| 简要描述 | 已经成功注册的用户登录系统 |
| 事件流 | 基本事件流：  1.用户填写用户名，密码  2.用户登录进入主界面 |
| 备选事件流 | 用户密码与用户名不匹配或用户名不存在，则  1.弹出错误提示界面  2.返回登录界面，用户重新进行登录  用户忘记密码，则  1.系统向注册时使用的邮箱发送验证邮件  2.用户使用验证邮件中的验证码进行验证  3.验证成功后，用户进入密码修改页面输入新密码并确认新密码 |
| 特殊需求 | 无 |
| 前置条件 | 用户已经成功注册 |
| 后置条件 | 进入用户界面 |

|  |  |
| --- | --- |
| 要素 | 说明 |
| 用例名称 | 用户查询票务 |
| 简要描述 | 用户根据选定的起始站，发车时间，列车类型查询票务信息 |
| 事件流 | 基本事件流：  1.用户输入起始站点，发车时间，选定列车类型  2.系统将符合条件的列车信息，行车时间，途经站地，座位信息，车票余量，车票价格信息形成列表显示在主界面  3. 此用例结束 |
| 备选事件流 | 没有与用户输入的信息相匹配的列车，则界面显示相应提示信息 |
| 特殊需求 | 无 |
| 前置条件 | 用户已成功登陆系统 |
| 后置条件 | 系统调用数据库中数据进行显示 |

|  |  |
| --- | --- |
| 要素 | 说明 |
| 用例名称 | 用户购买车票 |
| 简要描述 | 用户在查询的列表中选择购买车票 |
| 事件流 | 基本事件流：  1.用户选定查询列表中选定车票  2.用户添加乘客  3.用户提交订单并进入支付界面  4.用户选择支付方式并完成相应支付  5.此用例结束 |
| 备选事件流 | 用户选择车票后没有及时进行支付，则  1.系统生成相应订单等待用户支付，并在未支付订单项中进行生成  2.若30分钟后未进行支付则取消相应订单 |
| 特殊需求 | 无 |
| 前置条件 | 用户输入相应筛选条件 |
| 后置条件 | 完成支付后，系统更新数据库 |

|  |  |
| --- | --- |
| 要素 | 说明 |
| 用例名称 | 用户查询订单 |
| 简要描述 | 用户对自己的订单进行查询 |
| 事件流 | 基本事件流：  1.用户选择查询未出行订单、未支付订单、历史订单中的一项  2.系统对订单中车次，发车时间，乘客形成列表进行显示  3.此用例结束 |
| 备选事件流 | 没有相应的订单，则显示相应提示信息 |
| 特殊需求 | 无 |
| 前置条件 | 用户已成功登陆系统 |
| 后置条件 | 无 |

|  |  |
| --- | --- |
| 要素 | 说明 |
| 用例名称 | 用户退票 |
| 简要描述 | 用户对已经支付的车票进行退订 |
| 事件流 | 基本事件流：  1.用户点击进入未出行订单项  2.用户选择想要退订的订单  3.系统计算相应手续费后提示1-3个工作日内进行剩余退款  4.此用例结束 |
| 备选事件流 | 用户选择订单发车时间已经超过当前时间，则产生相应提示信息 |
| 特殊需求 | 无 |
| 前置条件 | 用户已成功支付订单 |
| 后置条件 | 系统对数据库中数据进行修改 |

|  |  |
| --- | --- |
| 要素 | 说明 |
| 用例名称 | 用户改签 |
| 简要描述 | 用户对已经支付的车票行程进行修改 |
| 事件流 | 基本事件流：  1.用户点击进入未出行订单项  2.用户选择想要改签的订单  3.用户进入查询界面重新选择行程  4.系统形成新的订单，计算相应手续费后进入支付界面或提示1-3个工作日内退款  5.此用例结束 |
| 备选事件流 | 用户选择订单发车时间已经超过当前时间，则产生相应提示信息 |
| 特殊需求 | 无 |
| 前置条件 | 用户已成功支付订单 |
| 后置条件 | 系统更新数据库中信息 |

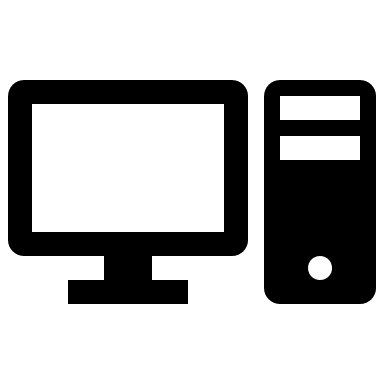
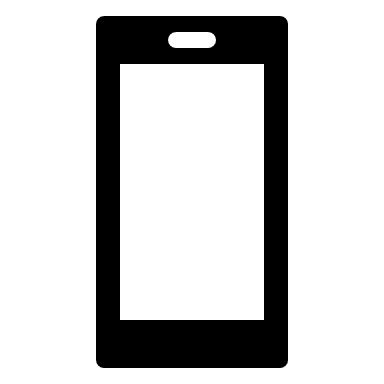
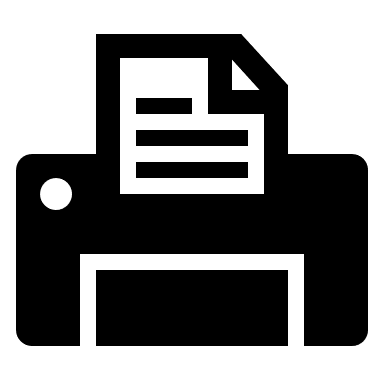
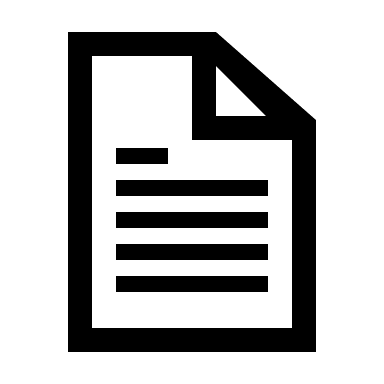
|  |  |
| --- | --- |
| 要素 | 说明 |
| 用例名称 | 用户信息管理员查询 |
| 简要描述 | 用户信息管理员对用户信息进行查询 |
| 事件流 | 基本事件流：  1.用户信息管理员输入身份信息登录系统  2.用户信息管理员输入权限码进入用户信息查询界面  3.用户信息管理员输入用户名及身份证号对用户信息进行查询  4..此用例结束 |
| 备选事件流 | 用户信息管理员输入权限码3次错误，则  暂时封锁该管理员账号 |
| 特殊需求 | 无 |
| 前置条件 | 用户信息管理员正确输入权限码 |
| 后置条件 | 无 |

|  |  |
| --- | --- |
| 要素 | 说明 |
| 用例名称 | 票务信息管理员修改票务信息 |
| 简要描述 | 票务信息管理员对车票信息，车次信息进行修改 |
| 事件流 | 基本事件流：  1.票务信息管理员输入身份信息登录系统  2.票务信息管理员输入权限码进入修改票务信息界面  3.票务信息管理员对票务信息进行修改  4.票务信息管理员对票务信息进行保存  5.此用例结束 |
| 备选事件流 | 票务信息管理员输入权限码3次错误，则  暂时封锁该管理员账号 |
| 特殊需求 | 无 |
| 前置条件 | 票务信息管理员正确输入权限码 |
| 后置条件 | 系统自动更新数据库，并备份数据 |

# 三、系统架构设计

## 1.体系结构

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，如果想对功能修改，可以方便的进行更改，大大减少维护成本。

表示层 业务逻辑 数据访问层

系统采用的设计模式：三层设计模式

视图层：提供显示界面，利用MFC实现

业务逻辑层：处理基本的业务逻辑以及实体类设计，利用C++实现

数据访问层：处理底层与数据库的交互，利用C++实现。

## 2.质量场景与策略选择

1）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500个用户的操作。

策略选择：限制队列大小和**LRU**缓冲战术。

2）可用性场景：售票系统尽可能24小时正常运作，减少系统故障率，存在故障时减少系统故障时间，尽可能减少维护时间，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。

策略选择：异常检查，故障恢复机制、资源调度机制。

3）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；防止软件抢票。

策略选择：身份验证，验证码，服务器防火墙，用户信息的高权限管理，用户信息加密存储，用户密码MD5加密验证

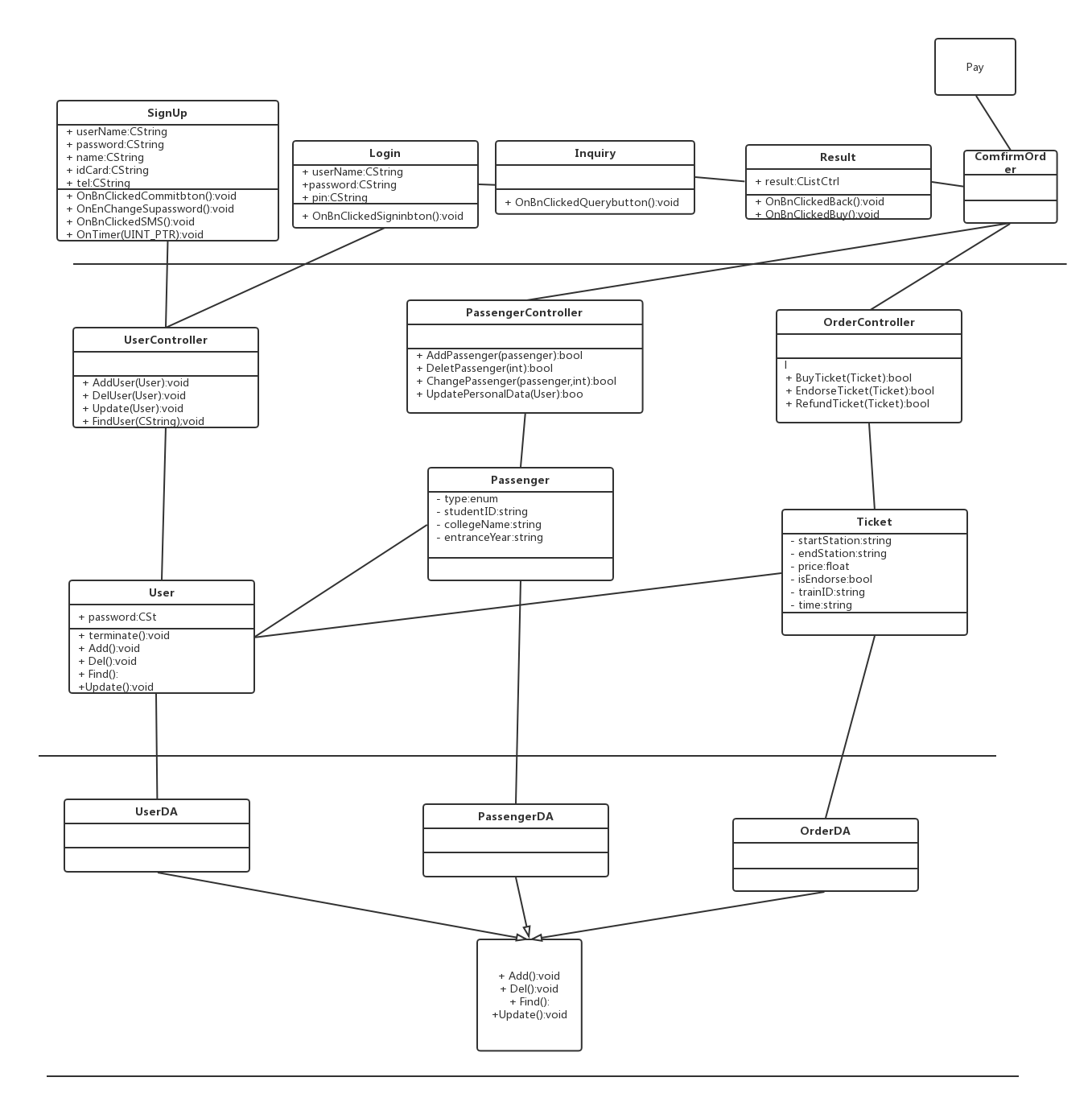
4）灵活性场景：售票系统可能会在之后推出pc端、手机客户端，因此需要系统具有良好的可移植性与可重用性，减少工作量。

策略选择：三层设计模式，使视图层，业务逻辑层，数据访问层分来

5）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。

策略选择：简化界面，快捷键双键设定，用户独立接口

## 3.类图设计



## 4.数据表设计

用户类：

|  |  |  |
| --- | --- | --- |
| 属性名 | 数据类型 | 限制 |
| 姓名 | String | Not null |
| 用户名 | String | Not null |
| 密码 | String | Not null |
| 身份证 | String | Not null |
| 电话 | String | Not null |

乘客类：

|  |  |  |
| --- | --- | --- |
| 属性 | 数据类型 | 限制 |
| 姓名 | String | Not null |
| 身份证 | String | Not null |
| 乘客类型 | String | Not null |
| 用户名 | String | Not null |

列车类：

|  |  |  |
| --- | --- | --- |
| 属性 | 数据类型 | 限制 |
| 车次 | String | Not null |
| 余票 | Int | Not null |
| 票价 | Float | Not null |

站点类：

|  |  |  |
| --- | --- | --- |
| 属性名 | 数据类型 | 限制 |
| 站点名 | String | Not null |
| 经过车次 | Key |  |