

**信息科学技术学院**

**文档名称：《需求分析说明书》**

**项目名称：《网上订票系统》**

**学生姓名：魏鹏超**

**学号：2208010423**

**专业：计算机科学与技术**

**班级：224班**

**2025年01月25日**

Contents

[1 系统用户 1](#_Toc6443)

[2 系统需求 2](#_Toc18823)

[2.1 功能需求 2](#_Toc7597)

[2.1.1 购票者功能需求 2](#_Toc12922)

[2.1.2 管理员功能需求 2](#_Toc15752)

[2.1.3 售票窗口工作人员功能需求 3](#_Toc2153)

[2.1.4 系统维护人员功能需求 3](#_Toc19033)

[2.2 性能需求 3](#_Toc10434)

[2.2.1 响应时间 3](#_Toc15628)

[2.2.2 并发处理能力 5](#_Toc23824)

[2.2.3 系统稳定性 6](#_Toc25610)

[2.2.4 数据安全性 7](#_Toc24829)

[2.2.5 可扩展性 7](#_Toc8113)

[2.2.6 兼容性 8](#_Toc16734)

[2.2.7 备份与恢复机制 9](#_Toc1963)

[3 系统设计思路 9](#_Toc9401)

[3.1 系统架构 9](#_Toc26593)

[3.2 开发工具 10](#_Toc20708)

[3.2.1 开发环境 10](#_Toc14874)

[3.2.2 开发语言 11](#_Toc7851)

[3.2.3 数据库 11](#_Toc10635)

[3.2.4 操作系统平台 11](#_Toc21413)

[3.3 系统主要开发技术 11](#_Toc22492)

[4 开发计划 13](#_Toc23447)

# 系统用户

**系统用户**是指与系统进行交互，并通过系统完成特定任务的个体或角色。不同类型的系统用户在系统中具有不同的权限和操作需求。

本系统主要面向的用户为：购票者，管理员，售票窗口工作人员，系统维护人员。

* + - * 1. 用户角色及其主要功能表

|  |  |
| --- | --- |
| **用户角色** | **主要功能** |
| 购票者 | 浏览车票信息 |
| 查询票价 |
| 预订车票 |
| 在线支付 |
| 查看订单 |
| 退票改签 |
| 管理员 | 管理车次信息 |
| 维护票价 |
| 处理订单 |
| 管理用户权限 |
| 生成统计报表 |
| 售票窗口工作人员 | 人工售票 |
| 订单查询 |
| 退票处理 |
| 异常情况处理 |
| 系统维护人员 | 服务器维护 |
| 数据库管理 |
| 系统升级与优化 |
| 安全漏洞修复 |

# 系统需求

## 功能需求

### 购票者功能需求

购票者是系统的主要用户，他们的需求主要围绕查询车票、购票、订单管理等功能展开。

1. **车票查询**

* 购票者可以通过出发地、目的地、日期等条件查询车票信息。
* 支持筛选车次、票价、座位类型等信息。

1. **票价查询**

* 购票者可以查看不同车次、不同座位等级的票价信息。
* 购票者可以比较同一线路不同车次的票价差异。

1. **车票预定**

* 购票者可选择车次、座位类型，并填写乘客信息进行预订。
* 支持单人或多人购票。

1. **在线支付**

* 购票者可通过支付宝、微信、银行卡等方式完成支付。
* 需提供支付状态查询和订单支付成功通知功能。

1. **订单管理**

* 购票者可在个人中心查看已购车票详情，包括订单号、乘车信息等。
* 购票者可查询历史订单记录，并提供筛选和排序功能。

1. **退票与改签**

* 购票者可根据退票规则申请退票，并自动计算退款金额。
* 购票者可修改订单，改签至其他车次或座位类型。

### 管理员功能需求

管理员负责管理系统核心数据，确保购票流程的正常运行。

1. **车次信息管理**

* 管理员可添加、修改、删除车次信息，包括出发地、目的地、发车时间等。
* 支持批量导入或导出车次信息。

1. **票价维护**

* 管理员可设置和调整车票价格，并配置不同座位等级的定价策略。

1. **订单管理**

* 管理员可查询所有订单信息，并处理异常订单。
* 支持按照订单状态（已支付、已退票、已改签）进行筛选。

1. **用户权限管理**

* 管理员可创建和管理不同类型的用户账户（如窗口工作人员、系统维护人员）。
* 支持分配不同的权限等级，确保安全性。

1. **统计报表生成**

* 系统可自动生成购票数据、销售额、订单情况等报表。
* 支持数据可视化，提供折线图、柱状图等统计方式。

### 售票窗口工作人员功能需求

窗口工作人员主要负责人工售票、订单查询、退票处理等操作。

1. **人工售票**

* 窗口工作人员可直接在系统中查询车次，并为购票者购买车票。
* 支持现金支付或第三方支付渠道。

1. **订单查询**

* 可查询购票者的订单信息，核对车票状态。

1. **退票处理**

* 窗口工作人员可手动为用户办理退票，并按照系统规则计算退款金额。

1. **异常情况处理**

* 支持处理支付失败、票务超售等异常情况。
* 可手动更改订单状态，并提供人工介入处理机制。

### 系统维护人员功能需求

系统维护人员主要负责系统稳定性、安全性和升级优化。

1. **服务器维护**

* 监控服务器状态，确保系统正常运行。
* 提供服务器日志分析和故障预警机制。

1. **数据库管理**

* 负责数据库的定期备份、恢复和优化。
* 保障数据一致性，防止数据丢失。

1. **系统升级与优化**

* 定期优化系统性能，减少查询延迟，提高响应速度。
* 部署新功能或修复系统漏洞。

1. **安全漏洞修复**

* 监测系统安全漏洞，并进行修复。
* 提供用户数据加密、访问权限控制等安全措施。

## 性能需求

本系统在设计和实现过程中需要满足高效、稳定、安全的性能要求，以确保用户在购票、查询、支付等关键操作中获得良好的体验。性能需求主要包括**响应时间、并发处理能力、系统稳定性、安全性和可扩展性**等方面。

### 响应时间

系统的响应时间是影响用户体验的重要因素，应尽可能优化系统性能，使用户操作流畅无阻，减少等待时间。

1. **页面加载时间**

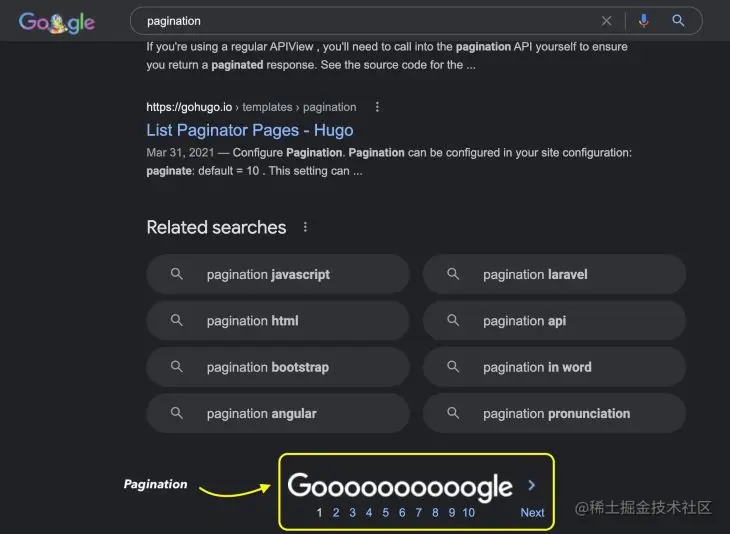
* 普通页面（如首页、车次查询页面）加载时间应**小于 3 秒**，数据量较大的页面（如订单管理、统计报表）加载时间应**小于 5 秒**。
* 采用 **CDN（内容分发网络）** 加速静态资源的加载，提升访问速度。



CDN技术示意图

1. **查询响应时间**

* 车票查询结果应在 **2 秒内**返回，并支持**分页加载**，以提高查询效率。
* 采用**数据库索引优化和缓存机制**，减少数据库查询压力。



分页加载示意图

1. **订单处理时间**

* 订单提交后的确认和支付状态更新应在 **2 秒内**完成，确保购票流程的顺畅。
* 采用**异步任务**处理订单，减少用户等待时间。

1. **支付流程**

* 支付成功或失败的反馈时间应在 **3 秒内**返回，以减少用户焦虑。
* 采用 **第三方支付接口（如支付宝、微信支付）** 提高支付效率，同时做好支付回调机制，确保订单状态同步。



一些支付API接口

### 并发处理能力

本系统需支持高并发访问，尤其是在节假日或高峰期，需具备较强的负载能力，以避免系统崩溃或响应过慢的问题。

1. **普通情况**

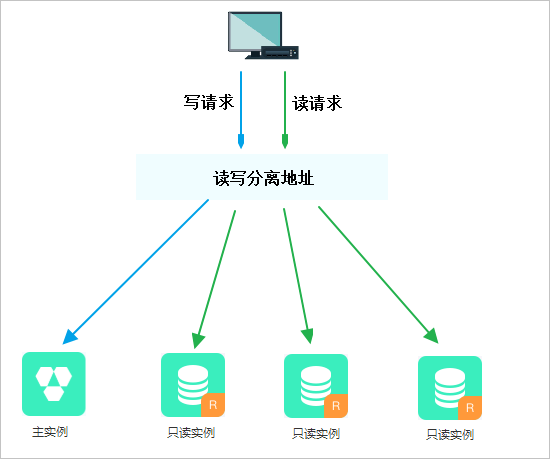
* 支持 **5000** 名用户同时在线访问，其中 **1000** 名用户进行购票操作。

1. **高峰期（如春运、国庆）**

* 支持 **20000** 名用户同时在线，购票请求峰值可达 **5000** 笔/分钟。

1. **数据库处理能力**

* 支持每秒至少 **1000** 条数据库查询或更新操作，避免因数据库瓶颈影响系统性能。
* 采用 **数据库读写分离、分库分表** 技术，减少单个数据库的压力。



数据库读写分离技术

1. **负载均衡**

* 采用 **Nginx 或 HAProxy 进行负载均衡**，均衡分配流量，提升系统稳定性。



NGINX

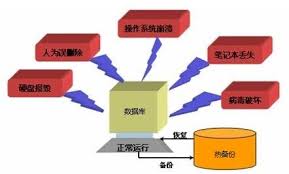
### 系统稳定性

1. **系统可用性**

* 全年可用性需达到 **99.9%**，即全年不可用时间不超过 **8.76 小时**。

1. **自动恢复**

* 服务器故障时可快速切换到备用服务器，实现 **热备份**。
* 采用 **微服务架构**，确保部分模块故障不影响整体系统运行。



热备份技术

1. **异常处理**

* 具备 **实时监控与报警**，在服务器出现异常时，系统可立即通知管理员进行处理。
* 采用 **日志记录与分析**，记录异常情况，便于事后排查和优化。

### 数据安全性

系统需采取严格的数据安全策略，确保用户数据、交易信息不被泄露或篡改，保障用户隐私和交易安全。

1. **数据加密**

* 用户密码、支付信息应采用 **SHA-256** 或更高级别的加密算法存储。
* 传输过程中使用 **HTTPS + TLS 1.3** 加密，防止中间人攻击。



SHA-256加密技术

1. **访问权限控制**

* 购票者、管理员、售票窗口工作人员、系统维护人员需进行角色划分，并采取 **最小权限原则（Least Privilege）** 进行访问控制。

1. **防攻击机制**

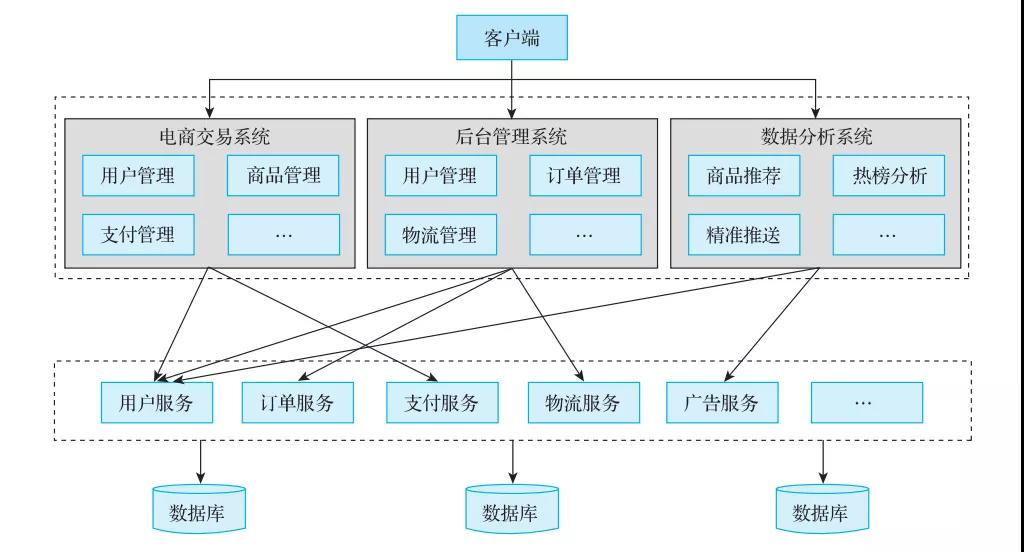
* 采用 **防火墙、DDoS 防御、验证码、人机验证** 等手段防止恶意攻击。
* 采用 **SQL 注入防护、XSS（跨站脚本攻击）防御** 机制，避免数据泄露。

### 可扩展性

为了适应未来业务增长和技术升级，系统需要具备良好的可扩展性，以支持更多的用户和功能。

1. **支持横向扩展**

采用 **分布式架构**，支持新增服务器提升系统吞吐量。



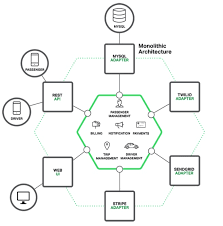
分布式架构系统示例

1. **数据库优化**

采用 **分库分表、索引优化、Redis 缓存** 等手段，提高大数据量场景下的查询效率。

1. **模块化设计**

采用 **微服务架构**，使各功能模块独立，便于后期增加新功能（如支持更多支付方式、增加智能推荐等）。



微服务架构

### 兼容性

系统需兼容不同设备和浏览器，保证用户在不同环境下均可正常使用，提高系统的可访问性。

1. **浏览器兼容性**

* 支持 Chrome、Firefox、Edge、Safari 等主流浏览器，并兼容最新的 **3 个版本**。

1. **移动端适配**

* 支持 **响应式设计**，适配手机、平板等不同屏幕尺寸的设备。
* 提供 **APP 版本**，提升用户体验。

1. **操作系统支持**

* 可在 Windows、MacOS、Linux 等常见操作系统上正常运行。

### 备份与恢复机制

系统需支持数据备份与快速恢复，防止意外数据丢失，确保业务连续性。

1. **定期备份**

数据库每天自动备份，保留最近 **30 天**的备份记录。

采用 **多地存储**，确保数据安全。

1. **紧急恢复**

当发生数据损坏或服务器宕机时，可在 **1 小时**内恢复至最近一次备份状态。

1. **日志记录**

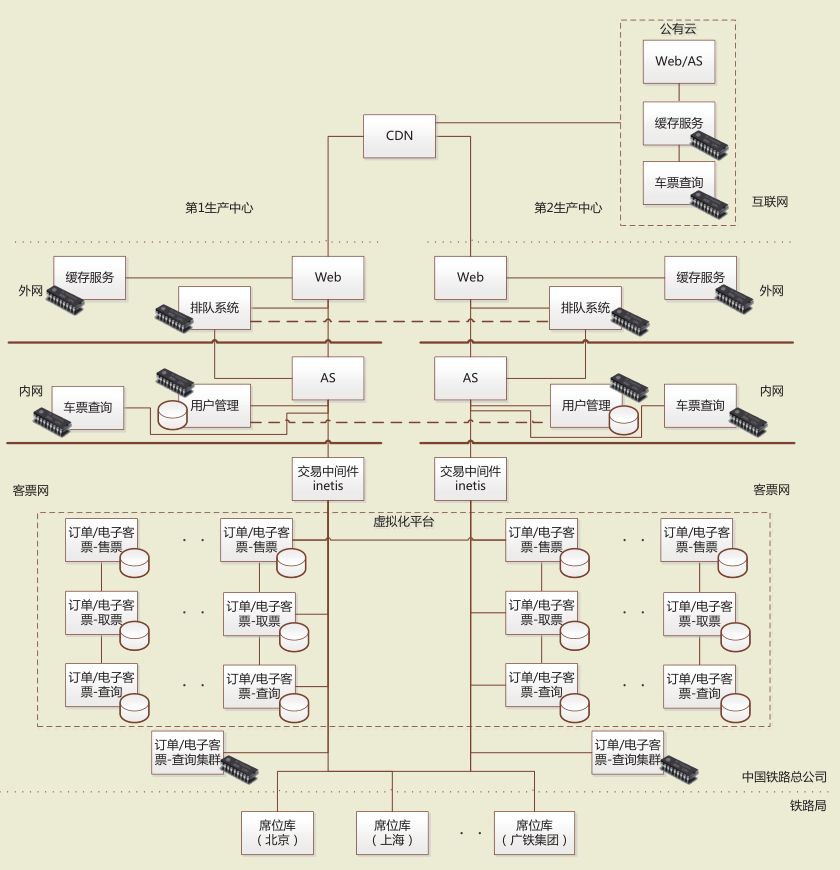
所有操作需记录日志，便于追踪和审计，日志需保存 **至少 6 个月**。

# 系统设计思路

## 系统架构

* **CDN（内容分发网络）** 作为流量入口，提高访问速度并减少服务器压力。
* **生产中心（第1、第2）** 处理购票核心业务，包括**Web 层（处理用户请求）、AS 层（业务逻辑）、缓存服务（车票余量查询）、排队系统（高并发购票）、用户管理（账户认证）、车票查询（数据存储）**。
* **客票网** 负责**订单/电子客票处理**，涵盖**售票、查询、取票、退票、改签**等功能，并通过**交易中间件** 确保数据传输稳定。
* **铁路总公司数据中心** 管理**席位库（北京、上海、广州等）**，确保购票信息的准确性和座位分配。
* **公有云** 提供**轻量级查询和缓存服务**，提升系统整体效率。

该架构采用**CDN 加速、高并发排队系统、分布式缓存、冗余备份、虚拟化平台**，支持**全国性票务系统**的高并发访问，保证**高可用性、高可靠性**。



系统整体结构图

## 开发工具

### 开发环境

* + - * 1. 开发环境表

|  |  |
| --- | --- |
| **开发环境** | **用途** |
| **IntelliJ IDEA** | 主要用于 Java 开发 |
| **Visual Studio Code** | 前端开发与 API 调试 |
| **Postman** | 接口调试与测试 |
| **Git/GitHub** | 版本控制与协作 |
| **Maven/Gradle** | 项目构建管理 |

### 开发语言

* + - * 1. 开发语言表

|  |  |
| --- | --- |
| **开发语言** | **用途** |
| **Java** | 后端核心业务逻辑开发 |
| **JavaScript/TypeScript** | 前端交互开发 |
| **HTML + CSS** | 页面结构与样式 |
| **SQL** | 数据库查询与管理 |
| **Shell 脚本** | 服务器维护与自动化运维 |

### 数据库

* + - * 1. 数据库

|  |  |
| --- | --- |
| **数据库** | **用途** |
| **MySQL** | 关系型数据库，存储车票信息、用户数据、订单记录 |
| **Redis** | 缓存数据库，加速查询，提高系统响应速度 |
| **MongoDB** | 非关系型数据库，存储日志和大规模非结构化数据 |

### 操作系统平台

* + - * 1. 操作系统平台表

|  |  |
| --- | --- |
| **操作系统平台** | **用途** |
| **Linux** | 服务器端操作系统 |
| **Win11** | 开发环境 |

## 系统主要开发技术

本系统采用多种技术栈来支持高效、稳定的购票功能，涵盖**前端、后端、数据库、缓存、消息队列、分布式架构**等多个方面。

**1. 后端开发技术**

* **Spring Boot**：基于 Java 的轻量级框架，负责业务逻辑和接口管理。
* **Spring Cloud**：用于微服务架构，实现服务注册、负载均衡、熔断等功能。
* **MyBatis/Hibernate**：提供数据库访问和 ORM（对象关系映射）功能，简化 SQL 操作。
* **Redis**：缓存车票信息，优化查询速度，减少数据库压力。
* **RabbitMQ/Kafka**：用于订单处理和消息通知，支持高并发场景。

后端开发技术

**2. 前端开发技术**

* **Vue.js/React**：用于开发购票系统的前端界面，提供用户友好的交互体验。
* **Axios**：实现前端与后端的 API 通信。
* **Element UI/Ant Design**：提供丰富的 UI 组件，优化用户体验。

前端开发技术

**3. 数据库技术**

* **MySQL**：核心数据库，存储用户、订单、车次等数据。
* **Redis**：作为 NoSQL 数据库，加速热点数据查询，减少数据库负载。
* **MongoDB**：存储日志和大规模非结构化数据。

数据库技术

**4. 分布式与高可用技术**

* **Nginx**：作为反向代理服务器，提高并发能力。
* **Docker/Kubernetes**：实现系统容器化部署，支持弹性扩展。
* **CDN（内容分发网络）**：加速静态资源加载，提高用户访问速度。

分布式技术

**5. 安全与优化技术**

* **HTTPS + JWT 认证**：保护用户数据安全，防止信息泄露。
* **SQL 注入与防火墙**：防止恶意攻击，确保数据库安全。
* **分布式锁（Redisson/ZooKeeper）**：防止超卖问题，提高数据一致性。

本系统通过**微服务架构 + 分布式缓存 + 高并发处理**，保障购票体验的流畅性和稳定性。

# 开发计划

* + - * 1. 开发计划

|  |  |
| --- | --- |
| **开发计划** | **任务分析** |
| 第一阶段：需求分析与系统设计 (2周)  2025.01.12 ~ 2025.1.25 | 需求收集, 技术选型, 系统架构设计, 并发性分析 |
| 第二阶段：核心模块开发 (4周)  2025.02.01 ~ 2025.02.28 | 数据库设计与优化, 接口设计与开发, 异步处理与消息队列, 缓存系统实现, 负载均衡配置 |
| 第三阶段：并发性与性能测试 (2周)  2025.03.01 ~ 2025.03.14 | 压力测试, 性能分析与调优, 高可用性测试 |
| 第四阶段：系统优化与安全性加强 (2周)  2025.03.15 ~ 2025.03.28 | 并发控制与限流, 数据库优化, 安全性设计与加固 |
| 第五阶段：运维与持续优化  2025.03.29 ~ \ | 持续监控与优化, 更新与迭代 |