在线车票服务系统

软件架构文档

版本 <1.1>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <04/07/2025> | <1.1> | 设计在线车票服务系统（Mini-12306），根据项目特点，我将选择9个视图进行设计并编写软件架构文档。 | 黄星铭 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 参考资料 4

2. 用例视图 5

3. 逻辑视图 6

4. 部署视图 7

5. 实现视图 8

6. 技术视图 8

7. 数据视图 8

8. 算法视图 8

9. 性能视图 9

10. 可靠性视图 9

11. 安全性视图 9

12. 易用性视图 9

13. 可维护性视图 9

软件架构文档

# 简介

## 目的

本文档描述在线车票服务系统（Mini-12306）的软件架构设计，将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策，涵盖核心模块划分、技术选型、部署方案及关键质量属性保障策略。目标读者为开发团队、测试人员及项目评审方。

## 参考资料

## [1] 刘玉,王永峰,张志强,等.铁路计次定期票产品的改良设计[J].铁路计算机用,2025,34(05):33-37. [2] 李明炀,白一凡.基于AI的火车票订票系统设计与实现[J].电脑编程技巧与护,2025,(04):141-144.

# 用例视图

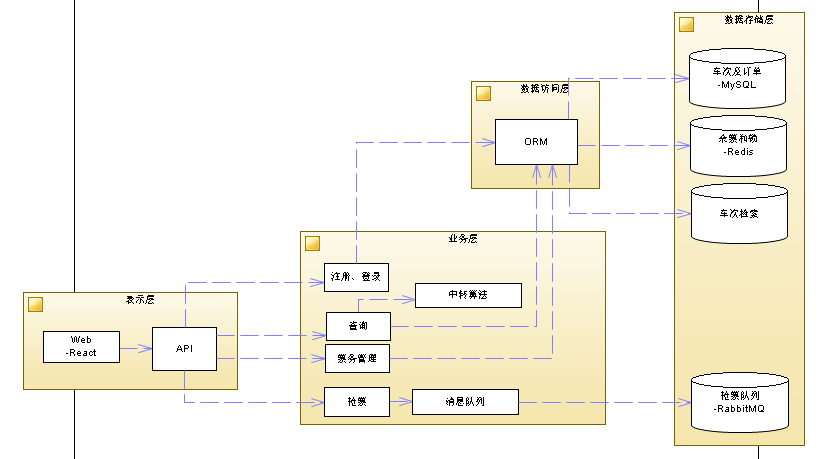
核心用例：

用户注册/登录、车次查询（含中转查询）、购票/退票/改签、抢票（高并发场景）、订单与车票管理、支付回调等

架构显著用例：

抢票用例：验证高并发架构能力（1000并发，响应≤3s）；

车次查询：验证大数据量（10k+车次）检索性能；

支付流程：保障事务一致性（购票+扣减库存）。

# 逻辑视图

分层架构：表示层🡪业务服务层🡪业务对象层🡪数据访问层🡪数据存储层

关键包与类：

User Interface(处理前端交互):LoginController, TicketSearchView

Business Services (核心业务逻辑):TicketService, OrderService, RushService

Business Objects (领域模型定义):User, Train, Order, Ticket

Data Access(数据库/缓存操作):TrainDao, OrderDao, RedisCache

类关系图要点：

TicketService 依赖 TrainDao 和 RedisCache 实现票务操作;RushService 通过消息队列解耦抢票请求（异步处理）等

# 部署视图

物理架构图：A[客户端] 🡪 B[Nginx负载均衡]、B 🡪 C[Web服务器集群]、C 🡪 D[业务微服务]、D 🡪 E[MySQL主从]、D 🡪 F[Redis集群]、 D 🡪 G[RabbitMQ]、D 🡪 H[Elasticsearch]

节点说明：

客户端（Web/APP)：用户交互入口

Nginx（反向代理）：流量分发、静态资源缓存

Web服务器（Tomcat + Spring Boot）：运行Controller层逻辑

业务微服务（Spring Cloud）：独立服务：用户/票务/订单

MySQL（主从复制）：持久化核心业务数据

Redis（集群分片）：缓存余票库存、会话数据

RabbitMQ(镜像队列):异步削峰（抢票请求排队）

Elasticsearch（集群）：车次快速检索（时间/地点）

# 实现视图

系统仅一个安装包。

# 技术视图

技术栈选型：

后端框架：Spring Boot 3.4 + Spring Cloud，微服务支持、生态成熟

数据库：MySQL 8.4 + Redis 7.0，事务一致性、高并发缓存

消息队列：RabbitMQ，异步解耦、流量削峰

搜索引擎：Elasticsearch 7.x，车次模糊检索性能优化

前端：React 18 + Ant Design，组件化、响应式布局

# 数据视图

ER模型已融入逻辑视图，单独视图冗余。

# 算法视图

核心算法（如抢票排队等）在性能视图中描述。

# 性能视图

# 设计策略：

# 缓存优化：Redis存储余票库存，Lua脚本实现原子扣减；热点数据预加载（如热门车次）。

# 异步处理：抢票请求入RabbitMQ队列，消费者限流处理（如500 TPS）。

# 数据库分片：按车次ID分库分表，单表数据≤50000。

# 无状态服务：JWT令牌实现会话管理，支持服务横向扩展。

# 性能指标：

# 车次查询：1000并发下响应≤2.5s（Elasticsearch索引优化）。

# 抢票接口：1000并发下响应≤3s（Redis+队列削峰）。

# 可靠性视图

# 设计策略：

# 冗余部署：MySQL主从切换（Keepalived）；Redis集群多副本（AOF持久化）。

# 故障隔离：微服务拆分：票务服务异常不影响用户登录。

# 事务补偿：Seata实现分布式事务（购票失败回滚库存）。

# 监控告警：Prometheus监控QPS/延迟，Grafana可视化。

# SLA目标：可用性≥99.95%（全年故障时间≤5小时）。

# 安全性视图

# 设计策略：

# 身份认证：JWT令牌 + RSA256非对称加密。

# 数据安全：敏感字段（身份证/银行卡）AES加密存储。

# 接口防护：Nginx限流（100 IP/秒）、防SQL注入。

# 支付安全：模拟银行回调验证签名，隔离支付沙箱环境。

# 易用性视图

通过React组件化及Spring Boot模块化保障，无需独立视图。

# 可维护性视图

通过React组件化及Spring Boot模块化保障，无需独立视图。