**项目总结报告**

日期：2025.7.18

|  |  |  |  |
| --- | --- | --- | --- |
| 组号 | 10 | 项目名称 | 在线车票服务系统 |
| 编程语言 | Java、Javascript | 开发平台和框架 | Spring Boot、React、Ant Design、Maven、npm |

|  |
| --- |
| **项目工作小结** |
| 1.采用哪种架构风格？哪些设计模式？  项目采用前后端分离的架构风格，后端基于RESTful API提供服务，前端通过API调用实现数据交互。这种风格确保了系统的可扩展性和模块化开发。同时，引入了微服务思想，例如我们项目中订单服务、票务服务独立模块化等。  设计模式方面：   * 1. MVC模式   后端Controller层处理请求，Service层实现业务逻辑，DAO层访问数据库。   * 1. 状态模式   用于订单和车票状态管理，例如我们的订单从待支付到已完成的流转，使用状态机避免复杂if-else。   * 1. 策略   对于高并发的实现，我们组没使用实时动态路由，而是采用了12306模式的全量高并发的策略。   * 1. 单例模式   Redis连接池和分布式锁的配置使用单例确保资源唯一性。   * 1. 创新模式   针对座位区间占用问题，自定义了“位操作策略模式”，使用二进制位表示站点占用状态，避免传统方法的数据冗余和性能瓶颈。  2.技术方案有哪些亮点？  2.1. 高并发抢票优化  采用Redis预缓存余票 + RabbitMQ异步队列 + 分布式锁（Redis Lua脚本），确保在1000并发场景下无超卖，响应时间<3s。  2.2. 座位区间占用创新  使用二进制位操作表示行程区间（始发站10、终到站01、中间站11），将整个区间压缩为一个整数字段，避免了数百万冗余数据，提高查询效率20%以上。  2.3. 数据拟真  虽未使用真实车次数据，但导入了约30000条爬取自某一时段的车次数据，并集成电子票证二维码的进阶需求。  2.4. 候补订单  实现了候补机制，定时任务自动兑现，增强用户体验。  3.是否做了单元测试？是否做了系统的功能测试、性能测试、兼容性测试和易用性测试等？  是的，我们进行了全面测试：  单元测试：使用JUnit覆盖核心模块，如订单服务（测试状态流转）和票务服务（测试库存扣减）等，覆盖率达93%以上。  系统测试：  功能测试 测试正常用户注册流程、注册非法数据是否成功、注册非法数据是否成功、跨天车票冲突检测、用户改签正常流程、用户候补正常流程等50余个业务模块  性能测试 测试1000并发用户下的购票请求压力测试、1000并发用户下的查询请求压力测试和3万级车次查询响应  可靠性测试 测试12h可靠性和超卖问题可靠性  安全性测试 测试权限控制、SQL注入攻击防御测试和支付凭证篡改测试  兼容性测试 测试不同浏览器和不同设备分辨率  易用性测试 测试信息显示和反馈是否准确和用户不需要培训就能使用系统  部署测试 首次安装部署系统  4.是否采用大模型来辅助开发？对开发生产率约有百分之几的提升？  是的，我们组采用了大模型辅助开发，主要用于乐观锁相关技术难题和优化SQL查询等。整体而言，提升了开发生产率约30%，因为它加速了问题解决，减少了手动调试时间，但我们仍需人工审查以确保质量。 |
| **项目组成员对项目的贡献度（%）** |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 姓名 | 需求 | 设计 | 编码 | 测试 | 项目管理\* | 小计 | | 黄星铭 | 70 | 45 | 28 | 32 | 100 | 30 | | 张弋洋 | 10 | 15 | 30 | 30 | 0 | 30 | | 彭若扬 | 10 | 20 | 20 | 18 | 0 | 20 | | 季桐辉 | 10 | 20 | 22 | 20 | 0 | 20 |   注： 项目管理包括项目计划、报告、沟通与协调等。 |

|  |  |
| --- | --- |
| **软件规模** | |
| 前端的代码行数（不包括注解行、空行和复用代码）： | 约9800 |
| 后端的代码行数（不包括注解行、空行和复用代码）： | 约22400 |

|  |
| --- |
| **经验、教训和建议** |
| 经验：  前后端分离架构大大提高了开发效率，便于并行工作。高并发场景下的Redis + RabbitMQ组合证明了其在抢票模块的可靠性。创新的座位区间位操作方案不仅解决了数据膨胀问题，还提升了查询性能，这为类似交通系统的票务管理提供了可借鉴的思路。Postman和JMeter等工具的使用，确保了测试的系统性和可重复性。  教训：  早期数据库设计时，未充分考虑改签和候补等需求以及高并发下的锁机制，导致数据库表结构欠佳或缺少考虑，后修改和新增字段与表等以及通过分布式锁修复，但浪费了一些调试时间。Mock数据在第一次迭代中便利，但切换到真实数据时，需要更多联调测试，以避免前后端不一致。团队协作中，编程规范的严格执行虽好，但初次配置Checkstyle插件时导致了一些格式化冲突，需要提前统一学习。  建议：  针对进阶需求（指中转查票），如果之后还有机会的话可以尝试考虑图算法优化路径；可引入自动化测试和部署，以进一步提升效率；加强安全测试，例如JWT token验证等，以防止潜在漏洞。体而言，项目已为组员奠定部分开发基础并使我们积累宝贵经验。 |

项目组各成员签字：黄星铭 张弋洋 彭若扬 季桐辉