**迭代计划（简化版）**

　　　　　　　　　　　　　制定日期：7.1

|  |  |  |  |
| --- | --- | --- | --- |
| 组号 | 10 | 项目名称 | 在线车票服务系统 |
| 迭代名称 | 技术原型迭代 | 计划起止日期 | 7.1-7.10 |
| 任务、进度安排和人员分配：   |  |  |  |  | | --- | --- | --- | --- | | No | 任务 | 起止日期 | 人员 | | 1 | 制定本迭代的《迭代计划》 | 7.1 | 黄星铭 | | 2 | 选择架构风格，确定语言、框架、工具 | 7.1 | 黄星铭 | | 3 | 设计多个架构视图 | 7.1-7.3 | 黄星铭、彭若扬 | | 4 | (若需要)设计关键算法 | 7.1-7.3 | 所有人 | | 5 | 撰写和评审软件架构文档 | 7.1-7.3 | 黄星铭 | | 6 | 选定或撰写编程规范(如google Java编程规范) | 7.1-7.3 | 黄星铭 | | 7 | 实现技术原型 |  |  | | 7.1 | 数据库设计（设计ER图：用户表、车次表、订单表、车票表、支付表等；字段定义（身份证、车次状态等）） | 7.2 | 张弋洋、黄星铭 | | 7.2 | 高并发架构和方案设计（抢票架构：Redis缓存余票 + RabbitMQ异步下单 + 分布式锁） | 7.2-7.3 | 张弋洋 | | 7.3 | 接口规范制定（定义RESTful API（URL/请求体/状态码），生成Swagger文档） | 7.2-7.3 | 黄星铭 | | 7.4 | 编程规范确认（采用Google Java规范，配置Checkstyle插件） | 7.3 | 黄星铭 | | 7.5 | 用户模块（User/Auth Controller：注册/登录接口（JWT签发）、UserService：实名认证逻辑（身份证校验、密码加密）、UserDao：用户数据CRUD） | 7.4 | 季桐辉 | | 7.6 | 数据库搭建（创建MySQL表结构（users/trains/orders等）、编写数据导入脚本（初始化10k+车次数据（脚本导入））、配置Redis连接池） | 7.3-7.4 | 张弋洋 | | 7.7 | 基础架构（搭建Spring Boot+MyBatis框架、集成Redis+RabbitMQ） | 7.2-7.3 | 黄星铭 | | 7.8 | 车次查询（TrainController：按时间/地点查询车次、TrainService：中转逻辑（进阶需求预留接口）、TrainDao：SQL优化（索引）） | 7.5-7.6 | 彭若扬 | | 7.9 | 订单模块（OrderController：订单生成/详情接口、OrderService：状态机（待支付/已完成/已取消）、购票事务方法（@Transactional）、退票库存回滚逻辑） | 7.5-7.6 | 张弋洋 | | 7.10 | 票务核心（TicketController：购票/退票/改签接口、关键逻辑：购票：MySQL事务扣减余票、退票：回滚库存+日志、改签：旧票作废+新票生成） | 7.5-7.6 | 张弋洋 | | 7.11 | 抢票模块（实时动态路由/全量高并发（12306模式）、RushController：抢票入口、高并发设计：RedisService：预缓存余票（Atomic操作）、RedisService.decrStock()：原子扣减库存、QueueService：RabbitMQ订单队列排队下单、超时处理：xx分钟未支付自动释放） | 7.7 | 张弋洋 | | 7.12 | 支付模块（订单状态机（待支付→已完成）、PaymentController：模拟银行支付回调、TicketService：生成电子票证（预留二维码签名接口）、PersonalController：个人中心数据加载） | 7.7-7.8 | 季桐辉、张弋洋 | | 7.13 | 订单管理（OrderQueryService：按状态筛选订单、车票详情关联查询（联表SQL）） | 7.6-7.7 | 彭若扬 | | 7.14 | 个人中心（ProfileController：个人信息获取/更新、数据脱敏处理（身份证号）） | 7.6-7.8 | 黄星铭 | | 7.15 | 高并发优化（压力测试抢票接口（JMeter）、修复可能的超卖问题（Redis+Lua脚本）） | 7.8 | 张弋洋、黄星铭 | | 7.16 | 前后端联调（前端移除Mock数据，调用真实API（Fetch API/Axios拦截器配置）、重点验证：登录态传递（JWT）、订单状态流转） | 7.9 | 所有人 | | 7.17 | 性能压测（使用JMeter模拟1000并发：抢票接口响应时间 ≤3s、余票更新一致性校验） | 7.8-7.10 | 黄星铭、张弋洋 | | 7.18 | （可选）中转查票功能 | 7.10 | - | | 7.19 | （非必要）竞态条件测试（甚至再安全审计） | 7.10 | - | | 8 | Bug修复、测试是否达到需求中预定的技术要求，并根据测试结果进行改进（如修复联调中的逻辑错误或优化SQL查询性能（Explain分析）等） | 7.9-7.10 | 所有人 | | 9 | 评审和测试记录，开发总结并编写《迭代评估报告》 | 7.10 | 黄星铭 | | | | |
| 预期成果：  文档：《第二次迭代计划》、《软件架构文档》、《Java后端编程规范》、《迭代评估报告》  代码：含抢票服务/订单服务等完整流程的后端核心、动态查询SQL+乐观锁实现的数据访问层、Redis队列操作+缓存预热的缓存层、真实API调用模块（移除Mock）的前端适配  数据库：含表结构+索引的SQL脚本、包括车次数据的测试数据  原型/组件：抢票服务、订单服务、前端应用 | | | |
| 主要的风险和应对方案：   1. 抢票性能不达标   风险描述：1000并发下响应时间>3s，或出现超卖/数据不一致  缓解措施：  Redis预优化：完成队列分片设计（SHARDED\_QUEUE）  限流降级：实现Guava RateLimiter（每秒500请求限流）  应急预案：  若仍不达标：采用简易库存扣减（同步DB事务）牺牲部分性能  启动日志采样：实时监控Redis队列深度   1. 数据一致性风险   风险描述：支付成功但出票失败，或订单/车票状态不一致  缓解措施：  事务补偿：实现订单状态机（OrderStateMachine）  对账任务：开发校验脚本  幂等设计：为支付接口添加idempotency\_key  应急预案：  人工修复脚本：准备data\_fix\_tool.sql（基于索引优化经验）  紧急回滚：支付失败时自动调用compensationService.revertStock()   1. 前后端联调阻塞   风险描述：接口协议不一致导致联调延期  缓解措施：  契约先行：定义Swagger接口文档  Mock备用：保留前端Mock开关（USE\_MOCK=true）  每次接口检查：每次合并验证3个核心接口  应急预案：  简化接口：暂不实现改签/退票复杂状态流转  聚焦核心：优先保证"查询-抢票-支付"主链路   1. 技术债务累积   风险描述：为赶进度忽略异常处理/日志规范  缓解措施：  强制Code Review：编程规范中要求关键服务CR  日志规范：统一错误码（ERROR\_CODE\_MAP）  自动化扫描：集成SonarQube基础规则  应急预案：  预留时间：预留一定时间的技术债务修复窗口  文档标注：在架构文档中明确待优化项 | | | |