**项 目 计 划 书**

姓名：周宣善

学号：SA20225083

目录

[一 项目简介 1](#_Toc55753349)

[二 项目范围 1](#_Toc55753350)

[2.1 人员 1](#_Toc55753351)

[2.2 产品 1](#_Toc55753352)

[2.3 过程 1](#_Toc55753353)

[2.4 项目 2](#_Toc55753354)

[三 项目进度 2](#_Toc55753355)

[3.1 WBS结构图 2](#_Toc55753356)

[3.2 规模估算 2](#_Toc55753357)

[3.3 甘特图 4](#_Toc55753358)

[3.4 里程碑图 5](#_Toc55753359)

[3.5 关键路径 5](#_Toc55753360)

[四 风险管理 6](#_Toc55753361)

[4.1 风险标识 6](#_Toc55753362)

[4.2 风险预测 6](#_Toc55753363)

[4.4 风险管理和控制 7](#_Toc55753364)

[五 质量管理 8](#_Toc55753365)

[5.1 质量控制QC 8](#_Toc55753366)

[5.2 质量保证QA 8](#_Toc55753367)

[六 配置管理 8](#_Toc55753368)

[6.1 软件配置项 9](#_Toc55753369)

[6.2 配置空间管理 9](#_Toc55753370)

## 一 项目简介

21世纪是一个互联网时代，大家日常生活中接触得比较多的就是网上购物，随着各大电商平台的兴起，无论是平台还是商家之间的竞争也日渐激烈，很多商家都会做一些秒杀的活动来吸引顾客，但是秒杀活动是爽了商家和消费者，苦了程序员，因为秒杀活动对系统的性能要求非常高，一个不小心服务器就会崩掉或者数据不一致，针对于这个问题，我打算开发一个基于JavaWeb的秒杀系统，该系统能够有效的保证系统可用性和数据一致性。

## 二 项目范围

### **2.1 人员**

本项目是独立开发项目，项目组成员只有周宣善一人

### 2.2 产品

1)目的：开发一个简单的支持高并发的秒杀系统

2)范围：包括商品展示、商品秒杀、订单查询以及上新(店家)

3)解决方案：前端采用layui进行页面设计，后端使用springboot框架，使用redis+rabbitmq处理高并发，数据库采用mysql

### 2.3 过程

采用软件生命周期的方法，系统的开发包括如下几个子过程：

1) 问题定义

2) 可行性研究

3) 需求分析

4) 概要设计

5) 详细设计

6) 编码和单元测试

7) 集成测试

8) 后期维护

### 2.4 项目

采用科学的方法及工具对项目基本内容进行管理，使用WBS对项目进行分解、使用甘特图和关键路径等方法估计项目进度等；开发过程中使用idea+git进行代码的开发和托管。

## 三 项目进度

### 3.1 WBS结构图

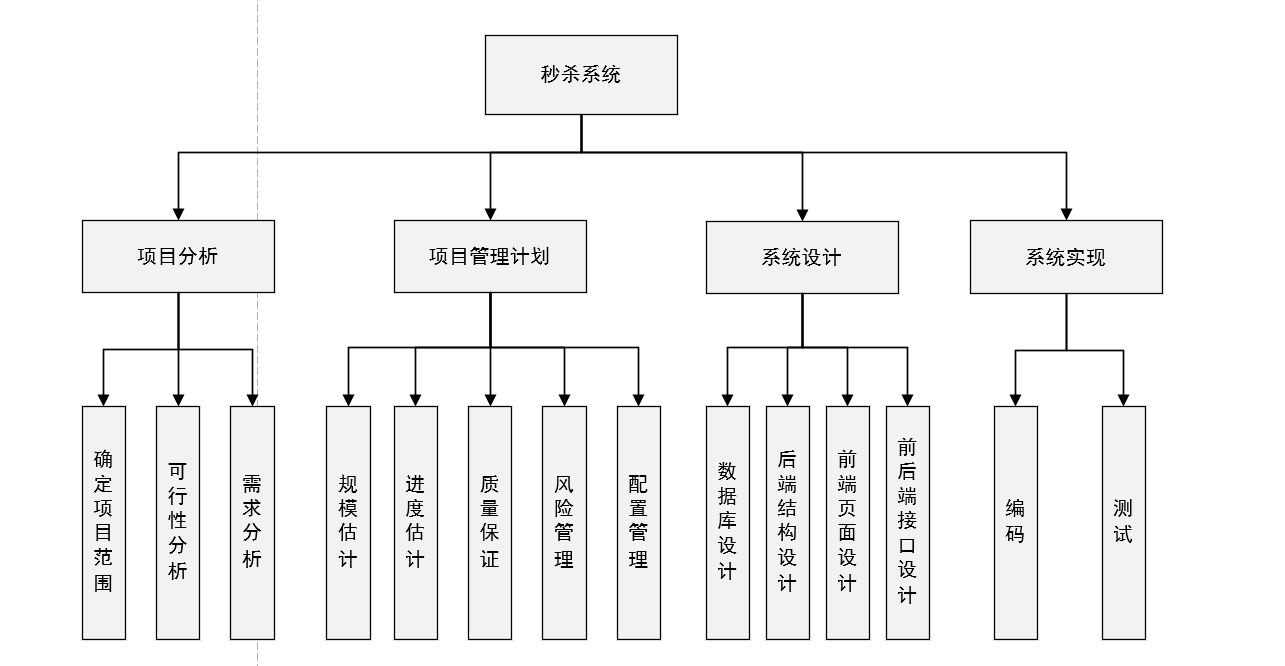


图3.1 WBS结构图

### 3.2 规模估算

#### 3.2.1 未经调整的功能点估算

1 ILF

登录用户信息：用户id 上一次操作时间

用户提交的秒杀信息：用户id 商品id 提交时间

商品信息：商品id 商品名字 商品价格 商品图片 商品库存

2 EIF

无

3 EI

用户登录信息：用户名 密码

用户秒杀商品提交：用户id 商品id 提交时间

管理员发布商品：商品名称 商品价格 商品数量 商品图片 秒杀时间

4 EO

用户秒杀结果：成功或者失败

用户登录结果：成功或者失败

5 EQ

用户查看可秒杀商品：商品名称 商品价格 商品图片 商品库存 秒杀时间

用户查看个人订单：订单号 商品名称 商品价格 提交时间

表3.1 ILF表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ILF | RET个数 | DET个数 | 复杂度 | 未调整的FP个数 |
| 秒杀信息 | 1 | 3 | 低 | 7 |
| 商品信息 | 1 | 5 | 低 | 7 |
| 登录用户信息 | 1 | 2 | 低 | 7 |

表3.2 EF表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EI | FTR个数 | DET个数 | 复杂度 | 未调整的FP个数 |
| 用户登录 | 0 | 2 | 低 | 3 |
| 秒杀提交 | 2 | 8 | 中 | 4 |
| 发布商品 | 0 | 5 | 低 | 3 |

表3.3 EO表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EO | FTR个数 | DET个数 | 复杂度 | 未调整的FP个数 |
| 用户秒杀结果 | 2 | 8 | 中 | 4 |
| 用户登录结果 | 1 | 2 | 低 | 3 |

表3.4 EQ表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EQ | FTR个数 | DET个数 | 复杂度 | 未调整的FP个数 |
| 浏览商品 | 0 | 5 | 低 | 3 |
| 查看订单 | 0 | 4 | 低 | 3 |

#### 3.1.2 系统特性调整

表3.5特性调整表

|  |  |  |
| --- | --- | --- |
| 序号 | 模块名称 | 最终结果 |
| 1 | 系统需要可靠的备份和和恢复吗？ | 5 |
| 2 | 需要数据通信吗？ | 5 |
| 3 | 有分布式处理的功能吗 | 0 |
| 4 | 性能是否关键 | 5 |
| 5 | 系统是否在一个已有的、很实用的操作环境中运行？ | 5 |
| 6 | 系统需要联机数据项嘛？ | 0 |
| 7 | 联机数据项是否需要在多屏幕或多操作之间切换以完成输入？ | 0 |
| 8 | 需要联机更新主文件嘛？ | 0 |
| 9 | 输入、输出、文件或查询很复杂吗？ | 2 |
| 10 | 内部处理复杂吗？ | 5 |
| 11 | 代码需要被设计成可复用嘛？ | 1 |
| 12 | 设计中需要包括转换及安装吗？ | 0 |
| 13 | 系统的设计支持不同组织的多次安装吗？ | 0 |
| 14 | 应用的设计方便用户修改和使用吗？ | 1 |
| 合计 |  | 29 |

综上所述：

FP = CT\*［0.65+0.01\*∑Fi］

=44 \* [0.65 + 0.01 \* 29] = 41.36

### 3.3 甘特图

图3.2 甘特图

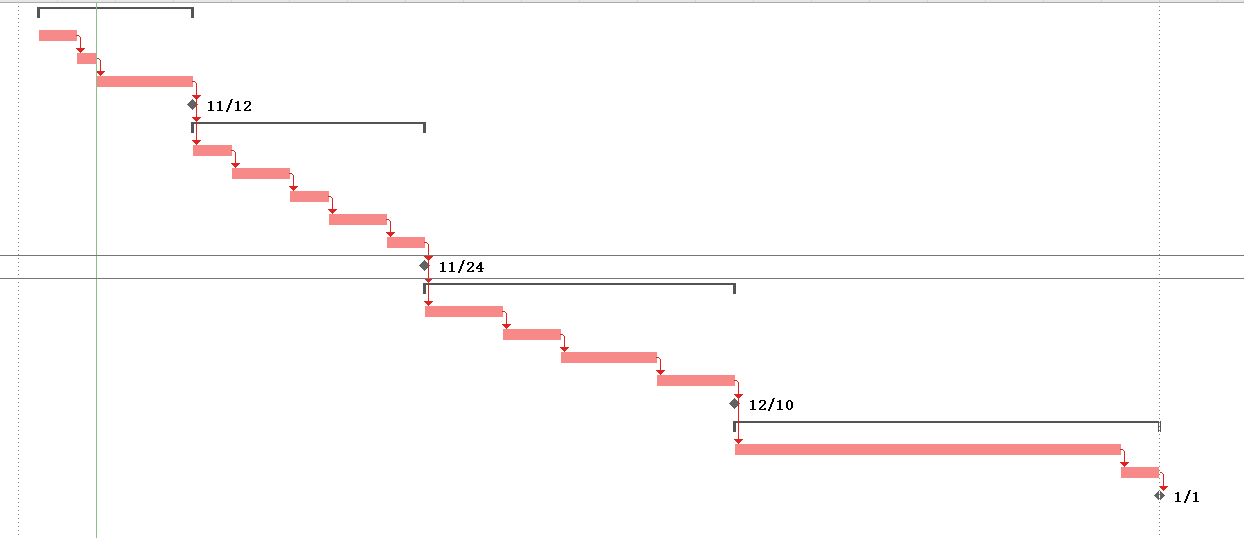


图3.3 项目进度图

### 3.4 里程碑图

图3.4 里程碑图

### 3.5 关键路径

图3.5关键路径

关键路径：项目分析→项目管理→系统设计→系统实现→交付，共56天。

## 四 风险管理

### 4.1 风险标识

表4.1 人员配备风险检测表

|  |  |
| --- | --- |
| 问题 | 风险程度（0~5分，5分风险最高） |
| 开发人员的水平如何 | 3 |
| 开发人员在技术上是否配套 | 3 |
| 开发人员数量如何 | 2 |
| 开发人员能够自始至终地参加软件开发工作 | 0 |
| 开发人员能否集中全部精力投入软件开发工作 | 2 |
| 开发人员对自己的工作是否有正确的期望 | 1 |
| 开发人员是否受过必要的培训 | 5 |
| 开发人员的流动是否能够保证工作的连续性 | 0 |

表4.2 组织和管理风险检测表

|  |  |
| --- | --- |
| 问题 | 风险程度（0~5分，5分风险最高） |
| 对项目的规模、难度估计是否比较正确 | 3 |
| 人力资源（开发人员、管理人员）够用吗？合格吗？ | 3 |
| 项目所需的软件、硬件能按时到位吗？ | 0 |
| 项目的经费够用吗？ | 0 |
| 进度安排是否过于紧张？有合理的缓冲时间嘛？ | 0 |
| 进度安排是否考虑了关键路径 | 0 |
| 是否可能出现某一项工作延误导致其他一连串的工作也被延误 | 2 |
| 缺乏必要的规范，导致工作失误与重复工作 | 1 |
| 低效的项目组结构降低生产率 | 1 |

### 4.2 风险预测

表4.3风险表

|  |  |  |  |
| --- | --- | --- | --- |
| 风险 | 风险类别 | 风险发生的概率 | 风险产生的影响 |
| 代码设计不规范，导致代码重复冗余 | 技术风险 | 0.3 | 项目结构复杂，后期维护麻烦 |
| 项目进度估算不准确 | 项目风险 | 0.3 | 项目延期 |
| 开发人员水平不足 | 技术风险 | 0.2 | 项目延期、成果达不到预期 |
| 大量的纸面工作导致进程比预期慢 | 项目风险 | 0.2 | 工程延期，最终质量下降，细节没有时间打磨 |

4.3 风险评估

考虑到项目进度和软件性能会对整个项目产生比较大的影响，给出进度和性能的风险参考水准

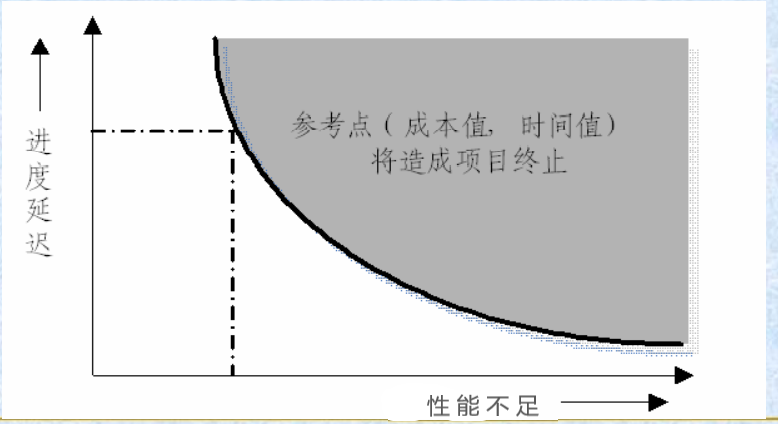


图3.5 进度和性能参考水准图

### 4.4 风险管理和控制

#### 4.4.1 风险避免：

1、对于代码不规范问题：学习相关的设计模式，在开发之前先进行代码框架结构设计，定义代码规范。

2、对于项目进度估算不准确：在项目进行过程中，根据项目的进度和自身的时间重新规划安排。

3、开发人员水平不足：去查阅相关的项目，熟悉项目中需要用到的技术，向老师和有经验的同学请教。

4、大量的纸面工作导致项目延期：对纸面工作有一个正确的认知，对于非必要的纸面工作可以采取简化，去适应项目变化，而不是完全跟随项目计划。

#### 4.4.2 风险监控

1、对于项目代码规范问题，做如下监控：

--监控代码的组织结构

--监控代码的接口定义

--监控代码中类或函数的代码行长度

2、对于项目进度估算不准确，做如下监控：

--监控项目的实施进度

--监控开发人员的工作效率

3、对于开发人员水平不足,做如下监控：

--监控开发人员的学习情况

--监控开发人员的技术栈

## 五 质量管理

### 5.1 质量控制QC

1、使用idea集成环境进行代码开发，开发过程中借助git进行代码托管，确保软件质量。

2、在软件开发每个阶段结束，都要组织技术评审，对质量评价，提交相关的评审报告。

3、从设计阶段就要指定测试计划，并且严格按照测试计划，在每个阶段或者模块开发完成后，进行相关测试与反馈。

4、在详细设计阶段确定项目的整体框架，代码规范以及命名约束等。

5、严格控制版本更新，每一次的修改提交都要产生版本号已经操作记录。

6、可度量规约：每个模块的完成都要有一定的工作产品，最好使用可视化的方式展现。

### 5.2 质量保证QA

软件质量保证主要由软件工程师和SQA小组两类角色承担。

#### 5.2.1 软件工程师

在本项目中，软件工程师为周宣善。为了保证软件质量，在项目开发的每个模块或者阶段完成之后，严格按照初期制定的软件测试计划，对每一个模块进行测试，填写测试文档和评审报告。

#### 5.2.2 SQA小组

1)小组人员：由于本项目是一个单人独立完成项目，SQA小组暂定为组外人员张振。

2)评审时间：每个阶段和模块完成之后

3)评审标准:

--项目是否符合预期进度

--软件功能是否当前阶段预期

--软件性能如何

--相关的文档是否完整

## 六 配置管理

### 6.1 软件配置项

本项目的软件配置项分为两大类

1. 产品组成部分的工作成果

--需求文档

--设计文档

--源代码

--测试用例

--数据库文档

2) 管理过程中产生的文档

--项目计划书

--项目进度管理文档

--版本日志

### 6.2 配置空间管理

使用git管理项目，整个项目的所有配置项都放到Github上托管。开发过程中，每次本地开发完成一个模块，经过测试之后，通过git提交上去，并填写相关的操作记录。版本更新是非常严格的，每一个版本都对应相应的版本号、发布日期、相关内容更新等，只有必要的时候才进行更新。