# 项目计划书2.0

## 一、团队架构：

组长：刘思远

副组长：宇通

组员：袁诚，朱庆夫，谢华祥。

|  |  |  |
| --- | --- | --- |
| 姓名 | 角色 | 工作描述 |
| 刘思远 | 组长 | 软件工程师，质量控制人员，软件架构师 |
| 宇通 | 副组长 | 项目经理，软件工程师，文档人员 |
| 袁诚 | 组员 | 软件工程师，测试人员，UI工程师 |
| 朱庆夫 | 组员 | 软件工程师，测试人员，需求工程师 |
| 谢华祥 | 组员 | 软件工程师，前端工程师，需求工程师 |

## 二、工作流程：

系统设计-需求分析-设计-编码-测试-支持

采用瀑布模型作为软件过程模型，因为这种模型是最早也是应用最广泛的软件过程模型.

## 三、拟采用的软件技术组合：

后端：Spring Boot + JPA +thymeleaf模板

数据库：MySQL

前端UI：Semantic UI框架

## 四、项目进度表:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **阶段** | **主要工作** | **应完成的文档** | **起始时间** | **开发周期（天）** |
| **系统需求** | 讨论项目可行性 |  | 2021.03.25 | 3 |
| 制定项目初步计划 |  | 2021.03.28 | 3 |
| **需求分析** | 确定系统运行环境 |  | 2021.04.01 | 3 |
| 编写规格说明 | 规格说明书 | 2021.04.04 | 4 |
| 确定项目开发 | 项目计划书 | 2021.04.08 | 3 |
| **概要设计** | 建立系统总体结构，划分功能模块 |  | 2021.04.11 | 3 |
| 定义模块接口 |  | 2021.04.14 | 3 |
| 数据库设计 |  | 2021.04.17 | 5 |
| **详细设计** | 设计各模块具体算法 |  | 2021.04.22 | 5 |
| 确定模块间详细接口 |  | 2021.04.27 | 4 |
| **实现代码** | 环境搭建 |  | 2021.05.01 | 5 |
| 配置文件实现 |  | 2021.05.06 | 5 |
| 前端页面实现 |  | 2021.05.11 | 12 |
| 登录拦截实现 |  | 2021.05.23 | 4 |
| 后端逻辑实现 |  | 2021.05.27 | 15 |
| **测试** | 模块测试 |  | 2021.06.13 | 5 |
| 系统总体测试 |  | 2021.06.17 | 5 |
| **部署维护** | 将系统上线部署 |  | 2021.06.22 | 1 |

## 五、软件质量保证:

1）《代码编写规范说明》

**个人博客系统编码规范**

1. 代码编写

1. 开发人员应保证工程中不存在无用的资源（如代码、图片文件等）。
2. 每个需要 import 的类都应使用一行 import 声明，不得使用 import xxx.\* 。
3. System.out.println() 仅在调试时使用，正式代码里不应出现。
4. 开发人员编写代码时应遵循以下命名规则：

● Package 名称应该都是由一组小写字母组成；

● Class 名称中的每个单词的首字母必须大写；

● Static Final 变量的名称全用大写，并且名称后加注释；

● 参数的名称必须和变量的命名规范一致；

● 使用有意义的参数命名，如果可能的话，使用和要赋值的字段一样的名称。

5) exit 除了在 main 中可以被调用外，其他的地方不应被调用。

6) 代码中应尽量使用 interfaces ，不要使用 abstract 类。

7) 在需要换行的情况下，尽量使用 println 来代替在字符串中使用的 "\n" 。

8) 涉及 HTML 的文档，尽量使用 XHTML1.0 transitional 文件类型，其中所有 HTML 标签都应关闭。

9) 在 HTML 、 JavaScript 、 XML 代码中，缩进应为两个空格，不得使用 Tab 。

10) HTML 标签的 name 和 id 属性的命名方式应与 Java 变量名相同。

11) 在需要经常创建开销较大的对象时，开发人员应考虑使用对象池。

12) 产品中不要包含后门代码，隔离系统中的后门代码，确保其不能出现在产品中。作为一种特殊的调试代码，后门访问代码是为了使开发者和测试工程师访问一部分终端用户不能访问的程序代码。但是，如果后门代码被留到产品中，对攻击者来说，它就是一条不需要通过正常安全手段来攻陷系统的通路。

2. JAVA 安全

遵循下面列出的准则有利于编写更加安全的代码。但是总体来说，这些准则不能对安全性做出任何保证。遵循这些准则可能好的实践，但是即使遵循了这些准则，写出的代码仍然可能是不安全的。但仍应当仔细的考虑如下准则：

1) 静态字段

避免使用非 final 的公共静态变量 ， 应尽可能地避免使用非 final 公共静态变量，因为无法判断代码有无权限改变这些静态变量的值。。

2) 缩小作用域

尽可能缩小成员方法和成员变量的作用域。检查包访问权限成员能否改成私有成员，保护访问成员可否改成包访问权限成员/ 私有成员等等。

3) 公共方法 / 字段

公共变量应当避免使用，访问这些变量时应当通过 getter/setter 法。在这种方式下，必要时可以增加集中的安全检查。任何能够访问或修改任何敏感内部状态的公共方法，务必包含安全检查。

4) 尽可能使对象不可变

尽可能使对象不可变。如果对象必须改变，使得它们可以克隆并在方法调用时返回副本。如果方法调用的返回对象是数组、向量或哈希表等，牢记这些对象并非不可变，调用者可以修改这些对象的内容并导致安全漏洞。此外，不可变的对象因为不用上锁所以能够提高并发性。

5) 不要直接在用户提供的数组里存储

这个数据存储到内部之前应当克隆，并保存克隆后的数据，而不是直接将数组引用赋给同样类型的内部变量。

1. 加密

如果决定加密，需要管理好密钥，注意密钥的存储以及密钥交付的方式等等。

7) 清除敏感信息

当保存敏感信息时，如信用信息，尽量保存在如数组这样的可变数据类型中，而不是保存在字符串这样的不可变对象中，这样使得敏感信息可以尽早显式地被清除。

3. 数据库安全

1) 开发人员应严格遵守Mysql编码规则编写代码。

2) 只给数据库用户授予其需要的最小权限，以保障数据库服务器的安全 。

3) 数据库表名、字段名必须大写。

4) 对于返回较大结果集的查询，必须禁止 SELECT \* ，在其他查询中也应避免使用。

5) 编写可以移植的 SQL 语句，原则如下：

● 不得使用某个数据库专用的关键字、函数等；

● 当必须要使用某个数据库特定的特性时，需在程序运行时，先判断当前数据库的类型，然后再根据数据的不同使用其特性；

● 可以使用各种数据库都支持的函数包括 MIN 、 MAX 、 AVG 、 COUNT ；

● 尽量使用简单的 SQL 语句，当因为特殊情况需要使用非常见 SQL 语句时，应该在多种数据库下测试。

4. WEB安全

1) 独立、完整且集中的输入验证

2) 校验全部的程序输入

3) 校验全部的输入长度

4) 校验全部的输入类型

5) 对 HTTP 所有内容进行校验

6) 校验向用户输出的数据

7) 除了完全公开的 Web 页面， 对于 其他 所有 Web 页面，需要验证访问用户是否具有访问权限。

5. 日志安全

1) 对每个重要的行为都记录日志。如认证尝试、重要传输、重要数据更改、管理行为等。

2) 保护日志文件。安全地保存日志文件，主要方法是将日志文件独立保存于应用程序目录外，同时通过严格的权限设置来控制对日志文件的访问。

2）《代码检查表》

**个人博客项目代码检查表**

|  |  |  |  |
| --- | --- | --- | --- |
| 待查项目 | 检查项 | 是/否/不适用 | 注释 |
| 1. 命名、注释及风格 |  |  |  |
| 1.1 文件、类/接口、静态变量、成员变量、方法及关键代码是否都有格式良好、简明扼要、的注释？注释是否是对设计思路的说明而不仅仅是代码行为的描述？是否存在过时的注释或废代码注释？ |  |  |  |
| 1.2 文件中各种段落布局是否合理、是否用恰当的空行分隔？代码的断行、对齐、缩进、空行是否恰当？ |  |  |  |
| 1.3 代码中的变量、属性、参数、方法、类是否恰当命名？有无易相互混淆的命名？ |  |  |  |
| 1.4 其他编码风格是否符合规范要求？如if、else永远加{}；for、while的格式；switch永远有default；case有break等等 |  |  |  |
| 2. 设计原则 |  |  |  |
| 2.1 成员变量和方法public/private/protected关键字是否恰当？内部类、非公共类及其内部成员、方法的访问权限是否合适？ |  |  |  |
| 2.2 嵌套内部类是否超过2层？ |  |  |  |
| 2.3 没有使用任何实例类成员（包括方法和成员变量）的方法是否被声明为静态的？ |  |  |  |
| 2.4 异常发生时，记录错误日志是否存在使用 System.out.println 而不是日志模块记录日志的情况发生？ |  |  |  |
| 2.5 参数、变量等的类型是否定义的合适？精度是否足够？方法的返回值是否定义恰当？ |  |  |  |
| 2.6 使用已有设计模式时，该模式要求的技术细节是否实现正确、完整？ |  |  |  |
| 2.7 是否存在废代码？ |  |  |  |
| ● 是否存在没有使用的参数、变量、对象实例？ |  |  |  |
| ● 是否存在重复、无效的方法、语句或 子条件 表达式？ |  |  |  |
| ● 子类或数据处理下游代码中是否重复设计 了父类或 数据处理上游代码中已有的功能？ |  |  |  |
| 3. 缺陷检查 |  |  |  |
| 3.1 程序逻辑是否符合详细设计？ |  |  |  |
| 3.2 输入参数、调用其他函数的返回值是否有必要校验合法性、完整性、依赖关系？如有必要，是否做了恰当的校验？如无必要，是否有多余的代码？ |  |  |  |
| 3.3 是否避免了直接抛出Exception类异常，而没有抛出恰当的由Exception派生的异常类？ |  |  |  |
| 3.4 try catch的结构是否合理？catch语句处理是否恰当？异常转抛时是否携带了嵌套异常？ |  |  |  |
| 3.5 打开的流、连接等资源是否在finally语句块或恰当的地方关闭或释放了？临时资源使用完后是否及时释放了？如临时文件要及时删除。各种资源释放的顺序是否正确？ |  |  |  |
| 3.6 条件、循环中的判断边界值是否恰当？ |  |  |  |
| 3.7 程序块的break、return、throw是否恰当？ |  |  |  |
| 3.8 相似的代码 块是否 是拷贝过来的？如果是，则需检查拷贝的代码中每处需要修改的地方是否都修改了。 |  |  |  |
| 3.9 是否存在其他可能造成缺陷的代码？ |  |  |  |
| ● 在if条件中赋值 |  |  |  |
| ● 精度丢失 |  |  |  |
| ● 浮点数判等 |  |  |  |
| ● 循环体中修改循环变量 |  |  |  |
| ● 字符串比较没有使用equals或 compareTo |  |  |  |
| ● 不恰当的static变量 |  |  |  |
| 4. 代码优化 |  |  |  |
| 4.1 在不影响可读性和易维护性的前提下，对象是否可重复利用？如 StringBuffer 可以通过 setLength (0)重复利用，无需每次重复创建新实例。 |  |  |  |
| 4.2 是否有这样的代码：new String( ”” )？ |  |  |  |
| 4.3 嵌套的条件判断、循环是否可优化？ |  |  |  |
| ● 嵌套的条件判断、循环是否可替换为较简单的结构？ |  |  |  |
| ● 是否存在可以合并的循环或条件判断？ |  |  |  |
| ● 循环体内是否存在可以提到循环体外的代码？如实例化对象或声明变量等 |  |  |  |
| 4.4 循环条件中或重复出现的如.length() .size() . getClass (). getName ()等取值方法的返回值是否使用了临时变量代替？ |  |  |  |
| 4.5 频繁拼装字符串时，是否使用了 StringBuffer ，并事先分配恰当空间？ |  |  |  |
| 5. 目录结构 |  |  |  |
| 5.1 Jsp,Action ，service ， dao ,逻辑分层是否明确 |  |  |  |
| 5.2 配置文件（struts，spring， sql ）是否条理清晰，目录规范 |  |  |  |
| 5.3 页面资源文件（ JavaScript,CSS,images ）目录是否规范 |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

3）《代码检查工具》

JArchitect：

JArchitect主要致力于Java语言中的代码分析。JArchitect是用于分析的最详尽的Java代码分析工具。

软件功能：

1、问题管理

只需要点击几下鼠标就可以找到相应的JAVA代码，然后进行管理操作。

2、插件集成

可以与PMD、FindBugs、CheckStyle和CPD等插件兼容。

3、探索现有的体系结构

查看现代的代码体系结构，然后找到相应的缺陷。

4、测试覆盖数据

导入测试覆盖数据，并在JArchitect特性丰富的上下文中充分利用它。

5、保持代码简洁性

JArchitect帮助用户清除无效和无用的代码，以保持代码的简洁性。

6、趋势监测

获取关于所有代码度量的趋势图，以掌握应用程序的发展。

7、生成定制报告

在最终导致摩擦和疼痛之前，请注意潜在的构建过程问题。

软件特点：

1、一个新的仪表板，可使用户对代码库的当前状态一目了然，并提供了与基线的对照。

2、趋势监控会监控50个默认“趋势指标”的趋势，也可以监控自定义趋势指标，所有这些都可以通过趋势图进行展示。

3、集成插件以支持导入来自Pmd、FindBugs、CheckStyle和其它Java静态分析工具的XML结果文件。

4、关注最近添加或重构代码元素时与某个基线相比违反规则的情况(借助过滤器)。

5、根据通用标准列出规则和查询，以及快速列出所违反的规则。

6、对用户界面进行了重大改进，使菜单的组织更现代化。

7、对报告进行增强和重新设计，包括趋势指标图和更多信息。