软件系统分析与设计

**WeShare学术成果分享平台**

**系统架构设计说明书**

|  |  |  |  |
| --- | --- | --- | --- |
| **项目组成员信息** | | | |
| **小组编号** | **15** | | |
| **小组名称** | **老师多给点分吧** | | |
| **小组联系人** | **何宇峰** | | |
| **学号** | **姓名** | **联系方式** | **本次实践中主要承担的工作内容** |
| 18182107 | 李睿楷 | [ricklee0624@gmail.com](mailto:ricklee0624@gmail.com) | 编写质量属性和待解决问题，参与讨论 |
| 18373041 | 王力 | [742650485@qq.com](mailto:742650485@qq.com) | 编写架构调研分析报告，参与讨论 |
| 18373056 | 罗驭 | [861392049@qq.com](mailto:861392049@qq.com) | 编写技术选型，参与讨论 |
| 18373082 | 罗志贤 | [1367678572@qq.com](mailto:1367678572@qq.com) | 编写横切关注点，参与讨论 |
| 18373229 | 姚树基 | [3183903748@qq.com](mailto:3183903748@qq.com) | 整合修改小组成果，完善格式，参与讨论 |
| 18373355 | 张洋 | [zhangyang946@163.com](mailto:zhangyang946@163.com) | 绘制包图和部署图，参与讨论 |
| 18373386 | 刘义浩 | lyh3024@163.com | 编写架构调研分析报告，参与讨论 |
| 18373669 | 赵致远 | [512063422@qq.com](mailto:512063422@qq.com) | 开展爬取数据工作，参与讨论 |
| 18373672 | 何宇峰 | [18373672@buaa.edu.cn](mailto:18373672@buaa.edu.cn) | 修改部分文档，编写ppt并展示，协调分工，主持讨论 |
| 18373761 | 吴涛 | [1422757727@qq.com](mailto:1422757727@qq.com) | 编写架构调研分析报告，参与讨论 |
| 78076003 | 金真率 | [jinsol000103@gmail.com](mailto:jinsol000103@gmail.com) | 设计界面，参与讨论 |

2020

版本变更历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 提交日期 | 主要编制人 | 审核人 | 版本说明 |
| V1.0 | 2020.11.15 | 姚树基 | 何宇峰 | 整合小组成员成果，完善格式 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 录

[1. 引言 1](#_Toc56335128)

[1.1 项目背景 1](#_Toc56335129)

[1.2 编写目的 1](#_Toc56335130)

[1.3 参考资料 1](#_Toc56335131)

[1.4 相关文档 1](#_Toc56335132)

[2. 整体架构 2](#_Toc56335133)

[3. 应用类型（Application Type） 2](#_Toc56335134)

[4. 技术选型（Appropriate Technologies） 3](#_Toc56335135)

[4.1 技术框架表格 3](#_Toc56335136)

[4.2 重要技术使用描述 3](#_Toc56335137)

[4.2.1 后端框架 3](#_Toc56335138)

[4.2.2 前端框架 4](#_Toc56335139)

[5. 部署策略（Deployment Strategy） 5](#_Toc56335140)

[6. 质量属性（Quality Attributes） 5](#_Toc56335141)

[7. 横切关注点（Crosscutting Concerns） 8](#_Toc56335142)

[7.1 安全 8](#_Toc56335143)

[7.1.1 数据库安全 8](#_Toc56335144)

[7.1.2 Web安全 9](#_Toc56335145)

[7.1.3 源码安全 9](#_Toc56335146)

[7.2 速度 9](#_Toc56335147)

[7.2.1 前端页面加载速度 9](#_Toc56335148)

[7.2.2 数据库连接速度 10](#_Toc56335149)

[7.2.3 后端处理速度 10](#_Toc56335150)

[8. 待解决问题和解决方案（Problems to be solved and Solutions） 10](#_Toc56335151)

[8.1 待解决问题列表 10](#_Toc56335152)

[8.2 高并发大数据访问情况下系统的正常运行 11](#_Toc56335153)

[8.3 认领门户的审核机制 11](#_Toc56335154)

[8.4 非结构化数据的存储形式 11](#_Toc56335155)

[8.5 数据源的选取与数据的更新机制 11](#_Toc56335156)

[8.5.1 数据源的选取 11](#_Toc56335157)

[8.5.2 数据的抓取方式 11](#_Toc56335158)

[8.5.3 数据的更新机制 12](#_Toc56335159)

# 引言

## 项目背景

随着国家对科技创新的日益重视，学术成果正变成日益宝贵的企业和社会资源，如何有效地实现学术成果分享并进一步推动科研合作正成为科学技术领域的热门话题。现如今的互联网上存在着很多学术成果分享平台，它们各有各的特点，能够满足很多人的需要。但是我们经过需求调研，发现这些平台并没有较多的温情，这并不能完全满足“分享”这一精神的要求。

因此，在我们进行充足的需求调研之后，我们计划开发WeShare平台，着重体现share这一精神，确定了“FEEL”这一主题风格，即：

Free（免费）：更为普通人考虑，让每一个人都能分享科学的喜悦。

Efficient（高效）：更快更实用的高级搜索，让好文章不再难以寻找。

Easy（简单）：更简洁的页面，解放你的眼睛，更解放你的心情。

Legal（合法）：更好的版权政策，给每一位科研人员我们最大的敬意。

我们力求给每一位用户带来最好的体验！

## 编写目的

在建立需求模型后，为规划项目、组织软件开发、方便数据库设计，我们进一步分析了需求。经过小组内的讨论分工，结合之前成果和上次作业《15-03-分析模型-01-分析模型文档》的类模型和关键用例交互模型，我们最终撰写了这份《系统架构设计说明书》。本文档用于描述WeShare学术成果分享平台的架构设计，包括应用类型（Application Type）、技术选型（Appropriate Technologies）、部署策略（Deployment Strategy）、质量属性（Quality Attributes）、横切关注点（Crosscutting Concerns）、待解决问题和解决方案（Problems to be solved and Solutions），作为软件开发工作的基础和依据以及确认测试和验收的依据。

## 参考资料

1. 谭火彬. UML2面向对象分析与设计[M].第二版.清华大学出版社,2019

## 相关文档

1. 《学术成果分享平台-背景陈述》
2. 《15-01-需求调研-01-需求调研报告》
3. 《15-02-需求模型-01-系统需求规格说明书》
4. 《15-03-分析模型-01-分析模型文档》
5. 《15-04-架构设计-01-Android架构调研分析报告》

# 整体架构

为了开发过程中明确职能，降低不同职能之间的相互依赖、提高程序的可扩展性和可复用性，我们采用分层架构的思想，确定了经典的三级架构模型，具体包括：。

界面层：用户界面或表现层，主要实现和用户的交互，接收用户请求或返回用户请求的数据结果的展现；

业务逻辑层（Service层）：实现业务处理逻辑，业务逻辑层承上启下，用于对上下交互的数据进行逻辑处理，实现业务目标；

数据持久层（Dao层）：数据库访问层，用于实现与数据库的交互和访问，从数据库获取数据或保存数据到数据库的部分。

根据分层架构思想，采用“B-C-E”（即“边界-控制-实体”）策略分包，画出本系统的包图（如图2.1所示）。



**图2-1 科技专家资源共享平台包图**

为处理版权问题，平台将引入第三方网站。

# 应用类型（Application Type）

经过小组讨论，我们认为Web端具有效率高、应用面广、学习成本低、开发周期短、开发成本低的特点，决定在Web端进行开发，具体理由如下：

1. 需求调研阶段问卷调查结果显示，大部分用户选择网页端作为资料查找第一选择
2. 查阅大量的资料后发现，Web应用对于开发者、用户都更加友好
3. Web应用开发、部署成本较低，适合学生开发学习
4. 之前有过两次Web应用经历，更熟悉Web应用的开发工具

# 技术选型（Appropriate Technologies）

## 技术框架表格

|  |  |  |
| --- | --- | --- |
| 应用框架 | Spring boot | 见下文 |
| Vue | 见下文 |
| 代码整合管理 | Git | 目前最主流、最通用的代码管理工具，便捷，应用广泛 |
| 数据框架 | Json |  |
| 数据库 | Mysql | 开发人员熟悉，安装方便，免费。  性能优秀，能够满足用户需求。 |
| 数据备份 | Veritas Backup | 提供安全可靠的数据备份服务，支持阿里云服务器，对试用服新手友好，提供免费务，解决经费  限制 |
| Web前端开发工具 | VSCode | 学习成本低，开发人员熟悉，且提供免费社区版以及专业版试用。 |
| WebStorm | 学习成本低，开发人员熟悉，且提供免费社区版以及专业版试用。 |
| Web服务器端开发工具 | eclipse | 学习成本低，开发人员熟悉，且提供免费社区版以及专业版试用。 |
| IDEA | 学习成本低，开发人员熟悉，且提供免费社区版以及专业版试用。 |
| VSCode | 学习成本低，开发人员熟悉，且提供免费社区版以及专业版试用。 |
| 爬虫 | Python | Python语言友好，启动成本低。框架成熟简单，编写成本低。 |

**表3.1 技术框架表**

## 重要技术使用描述

### 后端框架

后端计划采用Spring Boot框架。Spring Boot是由Pivotal团队提供的全新框架，其设计目的是用来简化Spring应用初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置，方便开发人员快速搭建和开发。

其具有以下一些优点：

* 1. 搭建项目快，几秒钟就可以搭建完成m33；
  2. 让测试变得简单。内置了JUnit、Spring Boot Test等多种测试框架，方便测试；
  3. 让配置变得简单。通过采用约定大约配置的方式，大量的减少了配置文件的使用；
  4. 内嵌容器。省去了配置Tomcat的繁琐；
  5. 方便监控。使用Spring Boot Actuator组件提供的系统监控，可以查看应用配置的详细信息。

开发过程中运用了前后端分离的技术，前后端分别进行开发，前端HTML页面通过AJAX调用后端的RESTFUL API接口并使用JSON数据进行交互。

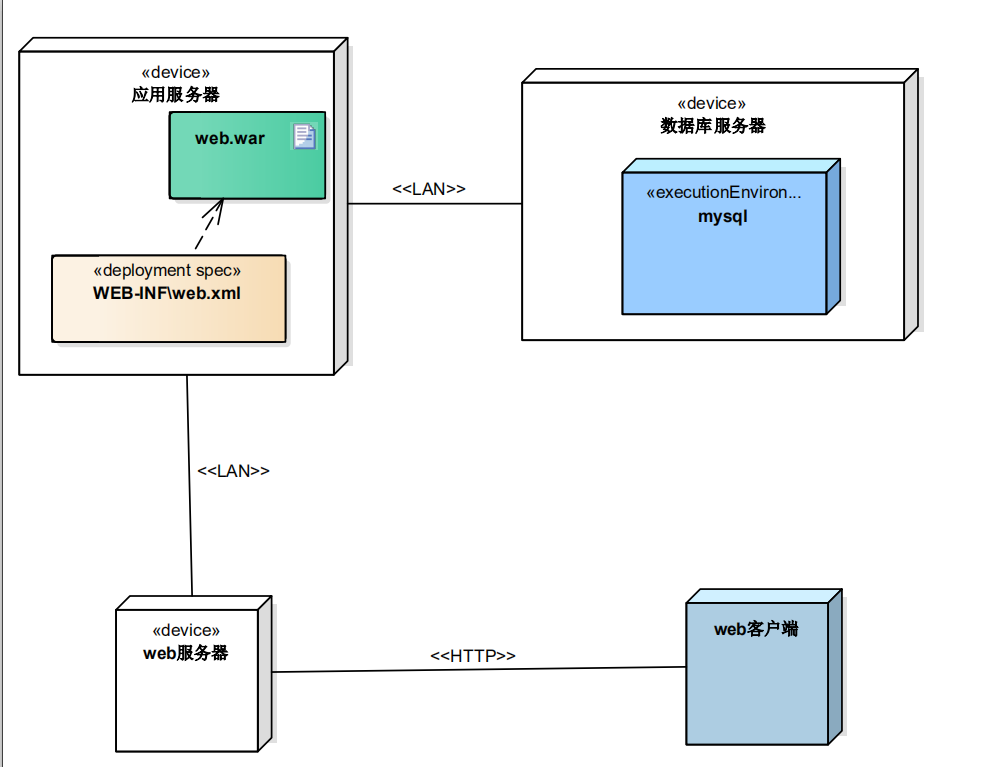
### 前端框架

前端计划采用Vue框架。Vue.js 是一套构建用户界面的渐进式框架。与其他重量级框架不同的是，Vue 采用自底向上增量开发的设计。虽然只是轻量级框架，但Vue 完全有能力驱动采用单文件组件和 Vue 生态系统支持的库开发的复杂单页应用。

其主要优点如下：

1. 轻量级的框架，只关注视图层，是一个构建数据的视图集合，大小只有几十KB；
2. 简单易学，国人开发，中文文档，不存在语言障碍，便于理解和学习；
3. 双向数据绑定，在数据操作中较为简单；
4. 组件化，采用可复用、解耦的组件来构建页面；
5. 数据驱动：自动计算属性和追踪依赖的模板表达式；
6. 虚拟DOM：DOM操作是十分耗费性能的，不再使用原生的dom操作节点，极大地提升了性能。

# 部署策略（Deployment Strategy）



**图3.1 weshare论文共享平台部署模型**

图3.1描述了weshare论文共享平台部署模型，该模型包括客户端节点（Web客户端）和处理器节点（服务器节点），节点间的通信路径通过图中的关联关系表示，比如Web客户端通过HTTP的方式访问Web服务器。此外，在数据库服务器节点内部定义了一个执行环境节点说明所采用的数据库管理系统为MySQL。同时我们将工件web.war直接放在应用服务器节点内部，表明了工件和节点的部署关系。而WEB-INF目录下的web.xml文件，他和web.war工件之间存在依赖关系，我们通过部署依赖关系将web.war部署到应用服务器上，与部署相关的配置信息采用类似关联类的方式被附加到部署关系。

# 质量属性（Quality Attributes）

## 可用性

### 可用性一般场景描述

|  |  |
| --- | --- |
| 场景的部分 | 可能的值 |
| 刺激源 | 系统内部、系统外部 |
| 刺激 | 输入、响应不当、崩溃 |
| 制品 | 系统、存储器、进程 |
| 环境 | 正常操作 |
| 响应 | 系统将检测时间并通知用户或系统管理员 |
| 相应度量 | 系统可用时间、异常或崩溃的修复时间 |

### 可用性具体场景描述

|  |  |
| --- | --- |
| 场景的部分 | 可能的值 |
| 刺激源 | 用户 |
| 刺激 | 大量用户同时访问网站，系统在短时间内无法处理这样的高并发访问从而崩溃 |
| 制品 | 系统 |
| 环境 | 正常操作 |
| 响应 | 系统检测到事件，通知系统管理员 |
| 相应度量 | 系统管理员维护后，系统可以正常使用 |

### 可用性战术

#### 错误检测

* 当出现异常时，异常处理程序开始执行。
* 当服务器出现崩溃，系统将及时做出响应并通知系统管理员。

#### 错误恢复

* 通过数据库的回滚，使系统返回到出现错误之前的状态。
* 通过事务日志进行恢复。

#### 错误预防

使用事务来保证数据的一致性，即几个相关的步骤或请求，要么全部成功，要么全部不成功。

## 可修改性

### 可修改性一般场景描述

|  |  |
| --- | --- |
| 场景的部分 | 可能的值 |
| 刺激源 | 用户、开发人员、系统管理员 |
| 刺激 | 希望修改功能、质量属性或系统容量 |
| 制品 | 系统界面、环境或与系统交互的页面 |
| 环境 | 设计系统时、构建系统时、系统运行时 |
| 响应 | 查找需要修改的模块在代码中的位置并在不会影响其他模块的前提下进行修改，修改完成后需要进行测试以及部署 |
| 相应度量 | 度量修改的成本和修改的影响 |

### 可修改性具体场景描述

|  |  |
| --- | --- |
| 场景的部分 | 可能的值 |
| 刺激源 | 开发人员 |
| 刺激 | 修改系统界面 |
| 制品 | 系统界面 |
| 环境 | 构建系统时 |
| 响应 | 在不影响其他模块的同时进行修改并完成测试、部署 |
| 相应度量 | 6小时之内完成对系统界面的修改 |

### 可修改性战术

#### 局部化修改

在修改之前将预期的变更限制在一定的范围中，通过局部化修改的方式降低修改成本。

#### 减少对其他模块的影响

系统的模块具有依赖性，当我们对某一模块进行修改的时候，可能会影响到其他的模。，因此，我们在修改之前应首先分析修改的部分将会影响哪些其他的模块，并在修改的过程中注意维持这些模块原有的功能。同时，我们需要尽可能地维持现有接口的稳定性，如果不得不更改接口，我们可以通过添加接口的方式实现。

## 性能

### 性能一般场景描述

|  |  |
| --- | --- |
| 场景的部分 | 可能的值 |
| 刺激源 | 系统内部或系统外部 |
| 刺激 | 定期事件、随机事件 |
| 制品 | 系统 |
| 环境 | 正常模式 |
| 响应 | 处理事件 |
| 相应度量 | 度量等待时间、吞吐量、缺失率 |

### 性能具体场景描述

|  |  |
| --- | --- |
| 场景的部分 | 可能的值 |
| 刺激源 | 用户 |
| 刺激 | 上传论文 |
| 制品 | 系统 |
| 环境 | 正常模式 |
| 响应 | 请求被处理，论文上传成功，数据库更新成功 |
| 相应度量 | 响应时间不超过3秒 |

### 性能战术

#### 资源需求

* 在编写代码的时候采用更加优秀的算法、减少计算开销。
* 在不影响功能实现的前提下，减少所处理的时间的数量。

### 综合描述

* 在浏览器端，通过使用浏览器缓存、更合理的布局平台界面等手段，优化性能。
* 在代码层，通过使用多线程、改善内存管理、代码优化等手段优化性能。
* 预留充足的数据存储空间。

## 安全性

### 安全性一般场景描述

|  |  |
| --- | --- |
| 场景的部分 | 可能的值 |
| 刺激源 | 对敏感资源进行访问的用户 |
| 刺激 | 试图访问敏感资源或试图使用系统提供的服务 |
| 制品 | 系统中的数据与服务 |
| 环境 | 正常模式 |
| 响应 | 对用户进行验证并阻止或允许该用户访问敏感资源或使用系统提供的服务 |
| 相应度量 | 使用安全措施所需要的时间或资源；增加安全性所需的成本 |

### 安全性具体场景描述

|  |  |
| --- | --- |
| 场景的部分 | 可能的值 |
| 刺激源 | 未认证科研人员的用户 |
| 刺激 | 试图进入门户系统 |
| 制品 | 系统提供的服务 |
| 环境 | 正常模式 |
| 响应 | 对用户身份进行验证并阻止其访问访问门户系统 |
| 相应度量 | 操作被拒绝 |

### 安全性战术

* 对用户身份进行验证。
* 对用户的访问进行控制管理。
* 系统采取权限设置和数据库加密等措施，只有管理员可以更新修改数据库。
* 对敏感信息进行加密。如：用户的密码。
* 定期备份数据库，当系统出现故障导致信息丢失的时候，可通过备份对数据库进行恢复。

## 可测试性

### 可测试性一般场景描述

|  |  |
| --- | --- |
| 场景的部分 | 可能的值 |
| 刺激源 | 各类测试人员（单元测试、集成测试、验收、用户） |
| 刺激 | 测试 |
| 制品 | 代码段或完整的应用 |
| 环境 | 设计时、开发时、编译时、部署时 |
| 响应 | 提供测试的状态值、测试环境与案例的准备 |
| 相应度量 | 测试成本、出现故障的概率、执行的时间等 |

### 可测试性具体场景描述

|  |  |
| --- | --- |
| 场景的部分 | 可能的值 |
| 刺激源 | 开发人员 |
| 刺激 | 单元测试 |
| 制品 | 代码段 |
| 环境 | 开发阶段 |
| 响应 | 准备集成环境 |
| 相应度量 | 执行测试的时间 |

### 可测试性战术

* 记录/回放：指捕获跨接口的信息，并将其作为测试专用软件的输入。
* 将接口与实现分离：允许使用实现的替代（模拟器）来支持各种测试目的。
* 优化访问线路/接口：用测试工具来捕获或赋予构件的变量值。

### 综合描述

编写代码时做到代码简洁、清晰，尽可能多的让代码具有可复用性。

## 易用性

### 易用性一般场景描述

|  |  |
| --- | --- |
| 场景的部分 | 可能的值 |
| 刺激源 | 用户 |
| 刺激 | 使用系统 |
| 制品 | 系统 |
| 环境 | 系统运行时 |
| 响应 | 根据用户所触发的事件提供对应的服务 |
| 相应度量 | 完成服务所需时间，错误数量，用户满意度 |

### 易用性具体场景描述

|  |  |
| --- | --- |
| 场景的部分 | 可能的值 |
| 刺激源 | 用户 |
| 刺激 | 检索科研人员信息 |
| 制品 | 系统 |
| 环境 | 系统正常运行时 |
| 响应 | 返回对应的科研人员信息 |
| 相应度量 | 99%以上的用户对搜索结果满意 |

### 易用性战术

* 维护用户的信息，使系统了解用户试图做什么，并提供各种协助。
* 支持用户主动操作。如支持“取消”、“撤销”等操作。

# 横切关注点（Crosscutting Concerns）

## 安全

### 数据库安全

（1）设置防火墙策略，只允许信任的服务器连接数据库。

（2）定期的对数据库进行备份。定期对数据库进行备份可以在很大程度上减少因硬件故障导致的数据丢失。

（3）为数据库的root用户设置强力的密码（长度长，且密码字符种类多），并且更改root用户的用户名。Mysql安装时都会默认设置一个root用户，很多黑客会root用户作为突破点。

（4）安排专门的数据库管理人员，尽量减少接触数据库的人员数量。接触数据库的人员过于多且复杂会给数据库的安全带来许多不确定性。在实际项目中，大家就通常不会让运维人员接触数据库。

（5）针对于不同的数据库管理人员，需要根据其实际的管理需求为其账户设置不同的用户权限

（6）移除多余的数据库用户，以及多余的数据库。避免存在一些密码为空默认的数据库用户，以及一些默认允许所有人访问的数据库。

（7）禁用LOAD DATA LOCAL INFILE指令。这个指令可以被黑客用来读取服务器的本地文件，对数据库安全有着很大的威胁。

（8）删除mysql\_history文件。

（9）避免网站的管理运营人员直接接触数据库系统，需要我们通过jdbc连接数据库，并且将必要的数据管理操作封装为网站的按钮提供给运营人员，且涉及到删除数据等敏感操作时，需要区分不同的权限。

### Web安全

（1）对用户密码设置的安全性进行要求，最好要保证密码至少由数字、字母、特殊字符组成。

（2）数据库中不要存储用户的明文密码，要利用比如MD5之类的技术对用户的密码进行加密，且当我们需要通过SESSION向浏览器传输用户信息时，我们需要保证先将用户的密码设置为空值。

（3）用户登录注册时密码要利用表单中的Password类型对数据进行隐藏，避免被他人直接从界面查看到密码。

（4）当用户注册时，如果用户名密码检验失败，我们反馈的提示信息需要注意，不能将密码错误独立作为一种情况来给出提示，需要将比如用户名不存在与密码错误合并作为一种情况同一提示信息。避免被黑客通过登录时的密码不存在提示，判断出哪些用户名是已经存在的。

（5）对用户所有的数据输入都要进行数据合法性检查。

（6）后端需要对报错信息进行统一处理，避免存在将报错信息直接通过页面显示给用户的可能性。

（7）后端在处理数据库访问层需要避免使用“+”来拼接SQL语句。

（8）实现用户上传论文功能时一定要对用户上传的文件进行类型检查，避免黑客将一些攻击程序（比如一些.jsp , .php , .asp等文件）通过用户文件上传功能传输到我们的服务器上，威胁服务器的安全。

（9）利用日志、监控等方法对服务器中的事件进行记录，以便为管理员事后的合理处理提供足够的信息。且如果监控到出现异常日志的情况，系统要及时通知管理员对异常情况进行处理。

### 源码安全

（1）利用终端安全管理技术，在存储源码的电脑上设置源代码终端安全区。安全区内源代码无法通过磁盘、端口、网络等任意途径泄露。

（2）对于源代码中的关键模块：如密码的加密算法、数据库访问层等，采用对文件进行重命名，加密，权限控制等方法进行保护。

## 速度

### 前端页面加载速度

（1）减少HTTP的请求。前端页面加载所花的时间中，HTTP请求的下载所花费的时间占了总时间的80%-90%。减少HTTP的请求可以有效的减少HTTP请求下载的时间，对减少加载速度十分有效。常用的减少HTTP请求的方法是减少请求的图片数量，将能够合并成一张大图的小图片都进行合并。

（2）将样式表放在前部。虽然这种方法对实际加载速度几乎没有什么影响，但是这样可以避免加载未完成时一直白屏的情况，可以先优先显示出一些网页的内容，在用户的感官上会觉得加载速度变快了。

（3）避免使用CSS表达式。CSS表达式虽然功能十分的强大，但是这会对性能产生巨大的负担，有些得不偿失。

（4）使用外部的JS和CSS。理论上来说，这样方法会导致HTTP请求增加，延长前端页面加载速度，但是实际上来说，浏览器会缓存外部的JS和CSS，客户只需要在第一次访问时发送HTTP请求，加载速度变长。在之后的访问中，网页加载速度反而会因为HTML文档大小的减小而加快。

（5）精简JS代码，移除代码中不必要的内容（如不必要的空白符），这样可以减少JS代码的加载时间。

（6）后端对功能进行实现时需要尽可能的减少对重定向的使用，重定向会对性能尝试一定的损耗。

（7）使用可缓存的Ajax，传统WEB应用经常会出现这样一种情况：用户提交表单后向服务器发送请求，然后服务器返回一个新的HTML页面，但是这两个页面又有许多的代码是一样的。这使我们就可以利用Ajax对其进行优化，我们利用Ajax向服务器取得必要的数据，然后再结合之前的HTML页面，即可形成一个新的HTML页面。

### 数据库连接速度

（1）设置最优的预取值，JDBC中预取条数默认值，触发查询操作的批量请求值这两个值的默认值均为10，在实际情况中一般偏小，我们可以通过增大这两个参数值来减少数据库请求，提高查询的效率。

（2）建立连接池。数据库连接是非常花费时间的，且我们需要重复使用的仅仅为对数据库数据进行操作的功能，频繁的新建、销毁数据库连接非常没有必要，所以我们可以建立连接池来避免此情况。

（3）设置检索时的批量值，服务器对数据库进行检索时并不是一次性将检索全部完成，而是根据检索批量值分批进行检索。提高批量值可以直接的减少检索批次，从而加快检索的速度。

（4）优化SQL的检索，比如严格保证实际检索范围是实际需要的检索范围，不要自行增大检索的范围，这虽然不会对结果产生影响，但是会明显影响到数据库检索的速度。

### 后端处理速度

（1）优化搜索算法，在数据量较大时，算法时间复杂的一个微小差距就可以导致执行时间的巨大差距。

（2）对代码的细节进行优化。根据计算机底层的具体实现对代码进行一些相应的性能优化，这样可以节省一定的时间。

# 待解决问题和解决方案（Problems to be solved and Solutions）

## 待解决问题列表

1. 如何保证系统在高并发大数据访问的情况下正常运行。
2. 科研人员认领门户时的审核机制。
3. 系统中的非结构化数据的存储形式。
4. 数据源的选取与数据的更新机制。

## 高并发大数据访问情况下系统的正常运行

首先数据库的设计需要更加严谨，尽可能降低数据的冗余同时保证数据库的完整性；其次，使用缓存，减少对数据库的访问次数；在后端的代码编写阶段更多地使用单例模式，减少不必要的对象创建并使用效率更高的容器。

## 认领门户的审核机制

本平台的科研人员申请门户的审核机制为自动审核和人工审核和并行。

科研人员通过前端提供身份信息后，首先，我们进行自动审核。针对身份证数据，后端可以通过使用腾讯云的ocr.tencentcloudapi.com接口请求域名实现对身份证的识别，接着将身份证识别的关键信息（如姓名、出生日期、籍贯等信息）与科研人员的信息进行核对和审查，最后向前端返回比对结果。针对认证邮箱数据，后端PHP通过基于PHP的邮箱地址验证API检查邮箱存在与否，接口调用地址为http://v.juhe.cn/verifyemail/query。并且可以通过PHPMailer这个现成的类文件实现验证码的发送。用户登录认证邮箱获取验证码，在平台输入验证码。若验证码正确，则通过审核。

如果用户通过了自动审核，则进行人工审核。

## 非结构化数据的存储形式

我们将资源文件存储在服务器的文件系统中，同时在数据库对应的表中存储资源文件在服务器文件系统中的绝对路径，当我们需要取用资源时，根据绝对路径即可访问。

考虑到数据量较大，且为了防止数据丢失，我们将在远程服务端花费相同的额外存储空间对数据库内容进行备份。

## 数据源的选取与数据的更新机制

### 数据源的选取

数据源选择中国知网的学术资源，在数据源选择过程遵循专指性原则、专业性原则、全面性原则、整体化原则、通用性原则。

“中国知网”系列数据库(简称CNKI)为一系列大规模集成整合传播我国期刊、博硕士学位论文、工具书、会议论文、报纸、年鉴、专利、标准、科技成果、古籍等各类文献资源的大型全文数据库和二次文献数据库，以及由文献内容挖掘产生的知识元数据库。至今为止收录了10000余种学术期刊、37万余篇博士论文、357万余篇优秀硕士论文、国内公开发行的500多种重要报纸等，拥有大量有价值的文献资源。

综上，数据源确定为中国知网的学术资源。

### 数据的抓取方式

采用爬虫的方式抓取需要的数据源。首先，确定需要爬取的网页的url地址，通过HTTP/HTTP协议获取对应的HTML页面，提取HTML页面中的目标数据。

数据爬取采用Python语言，利用scrapy框架进行开发。Scrapy运行流程大概如下：

1. 引擎从调度器中取出一个链接(URL)用于接下来的抓取
2. 引擎把URL封装成一个请求(Request)传给下载器
3. 下载器把资源下载下来，并封装成应答包(Response)
4. 爬虫解析Response
5. 解析出实体（Item）,则交给实体管道进行进一步的处理
6. 解析出的是链接（URL）,则把URL交给调度器等待抓取

对于可能遇到反爬机制，相应的解决方案如下：

1.若所爬取网站以检查User-Agent作为反爬虫机制，则通过设定Request Headers中的User-Agent，突破这种机制。

2.若网站限制访问频率，或通过检测访问时间间隔判断是否为爬虫。则选择代理IP爬虫的方式，代理IP为通过设置维护一个可用代理IP池进而规避网站对访问频率的限制。

### 数据的更新机制

为保证数据的时效性，平台将定期更新数据。考虑到我们的平台含有论文发表功能，即：若用户希望在短期分享自己的学术成果，可以自行上传。且考虑到更新一次数据库将花费较多的时间成本，因此平台将定期进行数据更新，而非与知网同步更新。