软件系统分析与设计

PaperIsAllYouNeed学术成果分享平台

架构设计报告

|  |  |  |  |
| --- | --- | --- | --- |
| 项目组成员信息 | | | |
| 小组编号 | 17 | | |
| 小组名称 | 请填写团队名称 | | |
| 小组联系人 | 18351015 张津赫 | | |
| 学号 | 姓名 | 联系方式 | 本次实践中主要承担的工作内容 |
| 19373239 | 王子涵 | 18519208373 | 质量属性相关设计 |
| 18351040 | 高野淇 | 15141810175 | 汇总文档，答辩 |
| 18351015 | 张津赫 | 17812032698 | 横切关注点 |
| 18351021 | 冯宇扬 | 15359226037 | 技术选型，部署策略 |
| 18351008 | 高嘉兴 | 15210224408 | 横切关注点 |
| 19373308 | 邹文祥 | 18811797692 | 技术选型，部署策略 |

2021.11.10

目录

[一、整体架构 3](#_Toc13979)

[二、部署策略 5](#_Toc8823)

[2.1 部署概述 5](#_Toc21866)

[2.2 流程概述 5](#_Toc174)

[2.3 设计中的注重点 6](#_Toc31609)

[2.3.1 使用Nginx来实现反向代理和负载均衡 6](#_Toc31362)

[2.3.2数据库读写分离 6](#_Toc16282)

[2.3.3 Redis缓存 7](#_Toc14438)

[2.3.4 Elasticsearch搜索引擎 7](#_Toc2486)

[三、技术选型 8](#_Toc26333)

[3.1 架构目标 8](#_Toc3990)

[3.1.1 高并发 8](#_Toc31378)

[3.1.2 高可用 8](#_Toc22807)

[3.1.3 强一致性 8](#_Toc12871)

[3.1.4 Devops 8](#_Toc3819)

[3.1.5 微服务 8](#_Toc11540)

[3.2 前端技术选型 8](#_Toc12360)

[3.2.1 Vue 8](#_Toc24529)

[3.2.2 Nginx 9](#_Toc28474)

[3.2.3 Axios 9](#_Toc19105)

[3.2.4 Jest 9](#_Toc12411)

[3.3 后端技术选型 9](#_Toc6128)

[3.3.1 sprintboot 9](#_Toc21235)

[3.3.2 Spring Data JPA 9](#_Toc22641)

[3.3.3 Elasticsearch搜索引擎 9](#_Toc17490)

[3.3.4微服务（springcloud-alibaba） 10](#_Toc27601)

[3.4 数据库技术选型 10](#_Toc21345)

[3.4.1 MySQL 10](#_Toc25659)

[3.4.2 Redis 10](#_Toc30207)

[3.5 其他技术选型 10](#_Toc25324)

[3.5.1 Python爬虫技术 10](#_Toc10215)

[3.5.2 Log4j2 10](#_Toc7776)

[3.5.3 Gradle 11](#_Toc28842)

[3.5.4 Jenkins 11](#_Toc1756)

[3.5.5 Scrapy 11](#_Toc21129)

[3.5.6 Lombok 11](#_Toc25703)

[3.5.7 YApi 11](#_Toc12562)

[3.5.8 Swagger 11](#_Toc24942)

[3.5.9 Postman 11](#_Toc30125)

[四、质量属性 12](#_Toc20393)

[4.1设计属性 12](#_Toc10616)

[4.1.1概念完整性 12](#_Toc8440)

[4.1.2可复用性 13](#_Toc14115)

[4.2 运行时属性 13](#_Toc7245)

[4.2.1 安全性 13](#_Toc23737)

[4.2.2 可靠性 15](#_Toc22252)

[4.2.3 可用性 15](#_Toc21604)

[4.3 系统质量 – 保障性 15](#_Toc31863)

[4.4 用户质量 – 易用性 15](#_Toc13498)

[五、横切关注点 17](#_Toc1196)

[5.1 安全 17](#_Toc19479)

[5.1.1 使用Spring Security + OAuth2提供安全的认证和访问控制 17](#_Toc11197)

[5.1.2 数据加密 17](#_Toc6152)

[5.2 缓存 17](#_Toc27296)

[5.2.1 前端数据缓存 17](#_Toc16602)

[5.2.2 Redis数据库缓存 18](#_Toc22235)

[5.3 日志 18](#_Toc704)

[5.4 异常处理 19](#_Toc19847)

[5.4.1 服务器宕机 19](#_Toc31038)

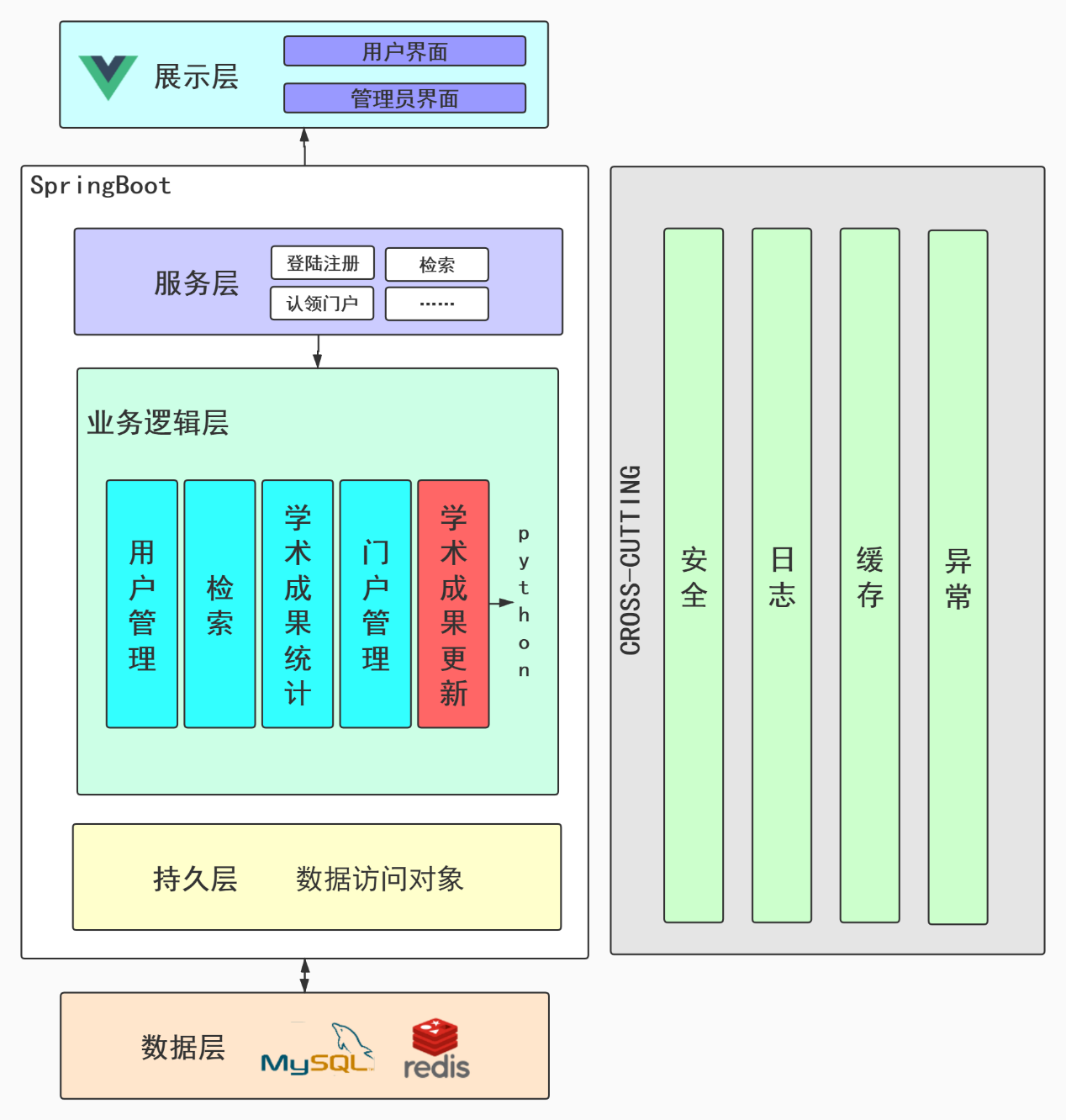
[5.4.2 请求未返回 19](#_Toc11175)

[5.4.2 记录异常 19](#_Toc9901)

# 一、整体架构

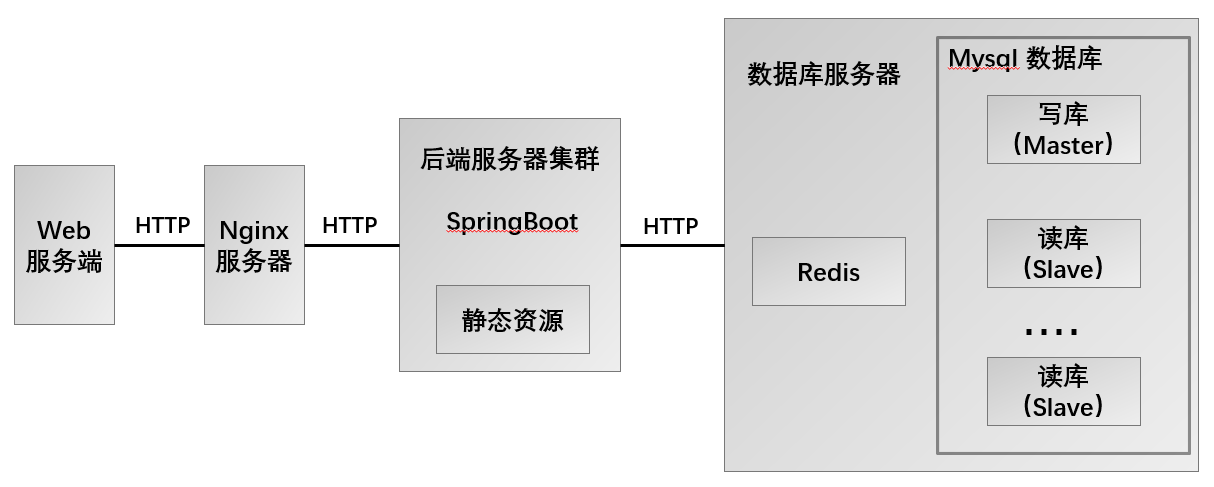
考虑到我们学术成果分享平台的界面需求和功能需要，以及开发过程中低耦合性的需要，我们平台将会采取前后端分离的开发方法。为了使整体平台的架构清晰，结合实际的功能特点，我们依托基本的分层架构，设计了我们的平台的系统架构，从而实现前后端分离，同步开发，也为后续的拓展做好准备。

参考Web Application原型，基于Spring的经典分层架构，我们平台的整体架构图如下，将整个平台大致分为五层：展示层，服务层，业务逻辑层，持久层和数据层。需要说明的是，由于实现的技术相同，我们将将服务层，业务逻辑层和持久层划分到一起，但其中学术成果更新的部分虽然功能上属于此模块，但实现的语言有所不同，已在图上特别标明。



**图1-1 架构设计图**

# 二、部署策略



**图2-1 部署策略图**

## 2.1 部署概述

设置ngnix作为前端服务器。

使用springboot部署后端在后端服务器集群中，并存储相应的静态资源。

在数据库服务器中，分别部署Redis缓存数据库和Mysql数据库，实现主从复制，实现读写分离。

## 2.2 流程概述

1、用户通过Web服务端发送HTTP请求，将该请求发送给配有Nginx的服务器上做反向代理，将请求平均转发到后端的服务器上。

2、Spring服务器接收到请求后，进行业务逻辑的相关判断，如果是静态资源，则直接通过axios将静态资源返回；否则进一步调用数据库API。

3、查看请求类型：

对于读请求，首先访问redis集群查看缓存，若存在所需数据，则直接读取，若不存在则去请求读库中的数据（Slave），并将其添加进缓存之中。

若为写请求，则直接对写库（Master）进行操作，并相应的与缓存进行同步。

## 2.3 设计中的注重点

### 2.3.1 使用Nginx来实现反向代理和负载均衡

Nginx是一款轻量级的Web服务器、反向代理服务器，它部署简单，成本低，内存占用少，高并发能力强。

反向代理（Reverse Proxy）方式是指以代理服务器来接受Internet上的连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给Internet上请求连接的客户端，此时代理服务器对外就表现为一个服务器。通过使用反向代理，能够起到保护网站安全的作用，因为任何来自Internet的请求都必须先经过代理服务器。同时能够通过缓存静态资源，加速web请求，还能够实现负载均衡。而Ngnix的反向代理比较容易，修改配置文件即可实现。

负载均衡的目的是为了解决单个节点压力过大，造成Web服务响应过慢，严重的情况下导致服务瘫痪，无法正常提供服务。Nginx负载均衡是通过upstream模块来实现的，默认情况下使用轮询模式，Nginx根据请求次数，将每个请求均匀分配到每台服务器上。这也方便了我们选择多台后端服务器。

### 2.3.2数据库读写分离

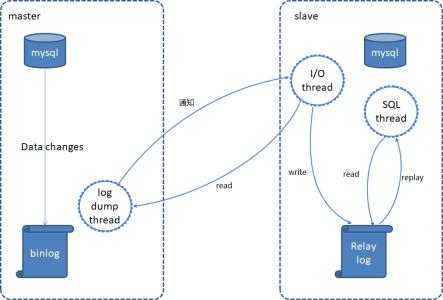
由于单台MySQL数据库服务器不能满足数据备份和读写频率高的要求，此时我们需要搭建一个数据库集群来来解决这一问题。数据库之间的同步可以采用主从复制来完成。MYSQL数据库本身所带的主从复制技术比较成熟，配置也不复杂，我们决定在项目中采用它。 MySQL数据库提供的主从复制的机制能够方便的实现数据的多处自动备份和读写分离。数据可以从一个Master数据库服务器主节点复制到一个或多个Slave节点。

我们在项目中将根据实际情况选择一主一从或一主多从的形式，将主库作为写库，其余从库作为读库。从而实现读写分离。具体实现上，是两台以上的服务器，操作系统选择ubuntu16.04；

1.建立一个Master节点，开启binlog，设置唯一的服务器id；

2.建立一个Slave节点，同样唯一的服务器id；

3.通过命令将Slave节点连接到Master节点上。即可完成。



**图2-2 数据库读写分离示意图**

这一措施使得数据库性能大幅提高

### 2.3.3 Redis缓存

详见5.2.2

### 2.3.4 Elasticsearch搜索引擎

Elasticsearch 是一个分布式、高扩展、高实时的搜索与数据分析引擎。它能很方便的使大量数据具有搜索、分析和探索的能力。充分利用Elasticsearch的水平伸缩性，能使数据在生产环境变得更有价值。Elasticsearch 的实现原理主要分为以下几个步骤，首先用户将数据提交到Elasticsearch 数据库中，再通过分词控制器去将对应的语句分词，将其权重和分词结果一并存入数据，当用户搜索数据时候，再根据权重将结果排名，打分，再将返回结果呈现给用户。

Elasticsearch是与名为Logstash的数据收集和日志解析引擎以及名为Kibana的分析和可视化平台一起开发的。这三个产品被设计成一个集成解决方案，称为“Elastic Stack”（以前称为“ELK stack”）。

Elasticsearch可以用于搜索各种文档。它提供可扩展的搜索，具有接近实时的搜索，并支持多租户。”Elasticsearch是分布式的，这意味着索引可以被分成分片，每个分片可以有0个或多个副本。每个节点托管一个或多个分片，并充当协调器将操作委托给正确的分片。再平衡和路由是自动完成的。“相关数据通常存储在同一个索引中，该索引由一个或多个主分片和零个或多个复制分片组成。一旦创建了索引，就不能更改主分片的数量。

Elasticsearch使用Lucene，并试图通过JSON和Java API提供其所有特性。它支持facetting和percolating，如果新文档与注册查询匹配，这对于通知非常有用。另一个特性称为“网关”，处理索引的长期持久性；例如，在服务器崩溃的情况下，可以从网关恢复索引。

# 三、技术选型

## 3.1 架构目标

### 3.1.1 高并发

分库分表：垂直拆分 水平拆分

读写分离：一主多从

负载均衡：轮询

动静分离：静态资源直接由Nginx转发

CDN：边缘计算，离用户更近的缓存

对象存储：云上的静态资源服务

缓存: Redis

### 3.1.2 高可用

集群，主从，备份容灾

### 3.1.3 强一致性

分布式事务控制

### 3.1.4 Devops

Jenkins + git hook一键部署

Docker + K8S 容器化，批量编排

### 3.1.5 微服务

解耦 弹性

## 3.2 前端技术选型

### 3.2.1 Vue

Vue是一套用于构建用户界面的渐进式框架。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。在vue中，结合css/html5可以形成更加丰富的前端页面，结合element-UI可以方便快捷地开发。单向数据流动，双向数据绑定。相比react上手更简单。

### 3.2.2 Nginx

Nginx是一款轻量级的Web 服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器，在BSD-like 协议下发行。其特点是占有内存少，并发能力强，成本低廉，节省带宽，稳定性好。

### 3.2.3 Axios

高性能异步ajax请求库。

### 3.2.4 Jest

js测试框架

## 3.3 后端技术选型

### 3.3.1 sprintboot

基于Spring4.0设计，不仅继承了Spring框架原有的优秀特性，而且还通过简化配置来进一步简化了Spring应用的整个搭建和开发过程。另外SpringBoot通过集成大量的框架使得依赖包的版本冲突，以及引用的不稳定性等问题得到了很好的解决。

### 3.3.2 Spring Data JPA

Spring Data JPA是Spring Data家族的一部分，可以轻松实现基于JPA的存储库。 此模块处理对基于JPA的数据访问层的增强支持。 它使构建使用数据访问技术的Spring驱动应用程序变得更加容易。

### 3.3.3 Elasticsearch搜索引擎

Elasticsearch 是一个分布式、高扩展、高实时的搜索与数据分析引擎。它能很方便的使大量数据具有搜索、分析和探索的能力。充分利用Elasticsearch的水平伸缩性，能使数据在生产环境变得更有价值。Elasticsearch 的实现原理主要分为以下几个步骤，首先用户将数据提交到Elasticsearch 数据库中，再通过分词控制器去将对应的语句分词，将其权重和分词结果一并存入数据，当用户搜索数据时候，再根据权重将结果排名，打分，再将返回结果呈现给用户。

### 3.3.4微服务（springcloud-alibaba）

Nacos：服务注册，服务发现，配置中心

RabbitMQ: 消息队列，发布订阅，灵活的routing

OpenFeign：远程调用，fallback，负载均衡

## 3.4 数据库技术选型

### 3.4.1 MySQL

MySQL是一种关系型数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。其体积小、速度快、总体拥有成本低，十分受欢迎。

### 3.4.2 Redis

Redis 是一个高性能的key-value数据库。可以作为关系型数据库mysql的补充，作为mysql的缓冲。mysql作为主存储，redis作为辅助存储被用作缓存，加快访问读取的速度，提高性能。二者配合，同时兼顾了性能和存储成本。

## 3.5 其他技术选型

### 3.5.1 Python爬虫技术

网络爬虫，是一种按照一定的规则，自动地抓取万维网信息的程序或者脚本。通过程序去获取web页面上自己想要的数据，也就是自动抓取数据。Python作为时下流行的语言，相比与其他静态编程语言，如java，c#，C++，python抓取网页文档的接口更简洁；相比其他动态脚本语言，如perl，shell，python的urllib2包提供了较为完整的访问网页文档的API。抓取的网页通常需要处理，比如过滤html标签，提取文本等。python的beautifulsoap等提供了简洁的文档处理功能，能用极短的代码完成大部分文档的处理。本项目利用爬虫技术获取网络上的论文及学者数据。

### 3.5.2 Log4j2

Apache Log4j 2是对Log4j的升级，它比其前身Log4j 1.x提供了重大改进，并提供了Logback中可用的许多改进，同时修复了Logback架构中的一些问题。是目前最优秀的Java日志框架。Log4j2虽然在各个方面都与Logback非常相似，但是却提供了更强的性能和并发性，下一代异步Logger，易于拓展自定义需求的架构，是目前使用十分广泛的日志框架。

### 3.5.3 Gradle

新一代构建工具，提供的DSL比maven的XML配置方式更加简洁。

### 3.5.4 Jenkins

老牌CI/CD工具，配合git hook可以实现开发过程中的一键部署。

### 3.5.5 Scrapy

成熟而完备的分布式爬虫框架，效率高，可定制化强。

### 3.5.6 Lombok

提供了丰富的注解，以非侵入的方式，给原生java bean扩展了丰富的功能。比如一个@Slf4j即可在类中使用log记录日志。

### 3.5.7 YApi

接口设计，接口测试

### 3.5.8 Swagger

OpenAPI的优秀实现，代码即文档。减少前后端沟通成本。

### 3.5.9 Postman

Workspace可以协同开发，大大提高接口测试效率。

# 四、质量属性

为了多维度分析平台的质量属性，我们将从设计属性，运行属性，系统属性和用户属性四个方面进行分析。

## 4.1设计属性

### 4.1.1概念完整性

概念完整性定义了整体设计的一致性和连贯性。这包括组件或模块的设计方式，以及编码风格和变量命名等因素。

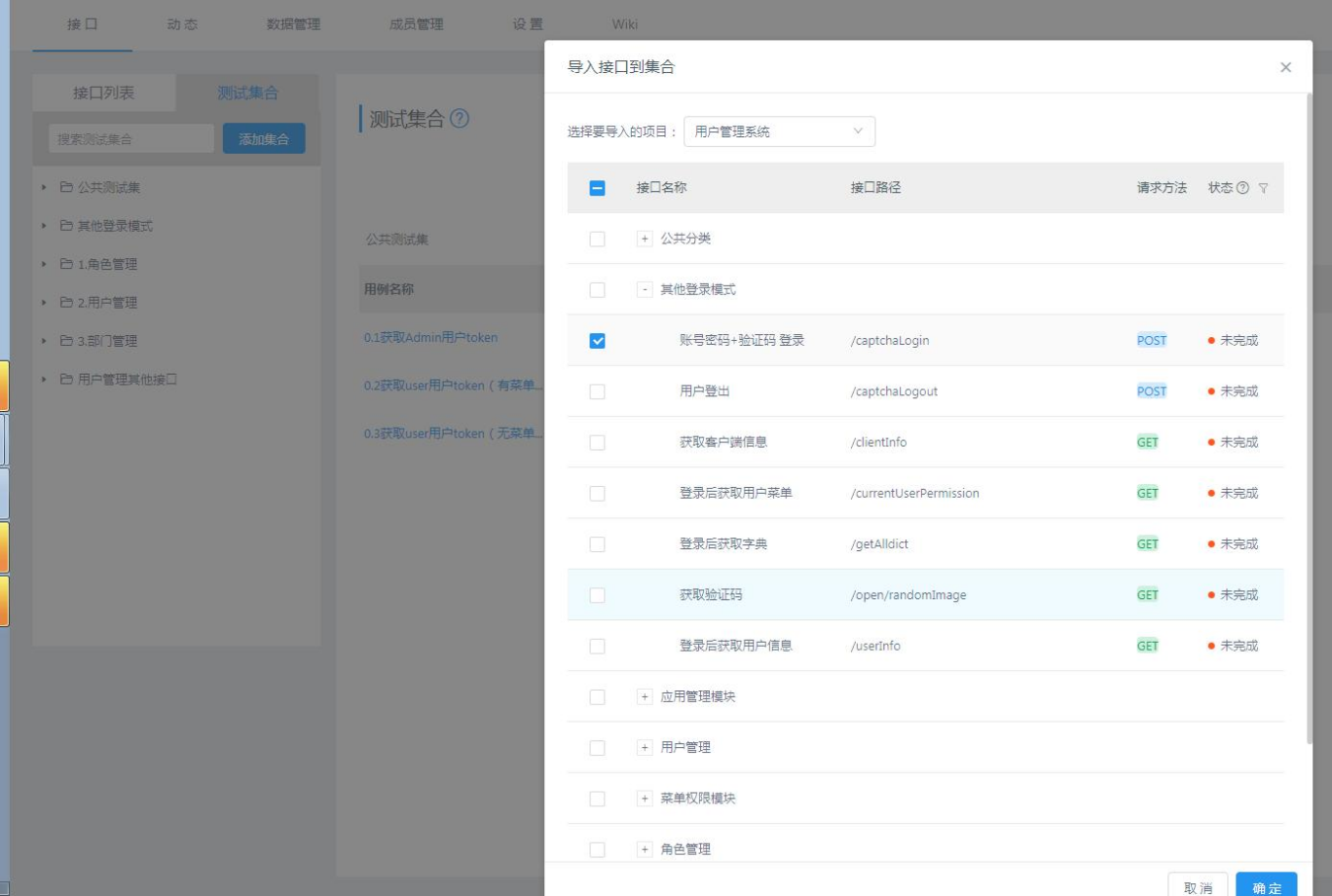
1. 前后端分离



前后端分离技术是当今web开发的主流技术，通过前后端分离开发的手段可以减少

每名开发者的职责范围，也可以提高开发效率。

我们计划通过springboot+vue框架进行开发，在对接时，为了确保前端javascript与后端controller层中API接口的统一，我们计划通过Yapi提供的服务来进行接口的统一，测试，与开发进度的监测：



### 4.1.2可复用性

可重用性定义了组件和子系统适合在其他应用程序和其他场景中使用的能力。可重用性最小化了组件的重复性和实现时间。

针对前端较为常用的显示部分，我们将其编写为独立的组件，以便于频繁的复用，并且方便简单。在后端中，可能出现大量使用的子功能，我们将其处理为全局功能，以便于减少代码重复，方便使用。

## 4.2 运行时属性

### 4.2.1 安全性

前后端分离的开发方式，我们以接口为标准来进行推动，定义好接口，各自开发自己的功能，最后进行联调整合。无论是开发原生的APP还是webapp还是PC端的软件,只要是前后端分离的模式，就避免不了调用后端提供的接口来进行业务交互。

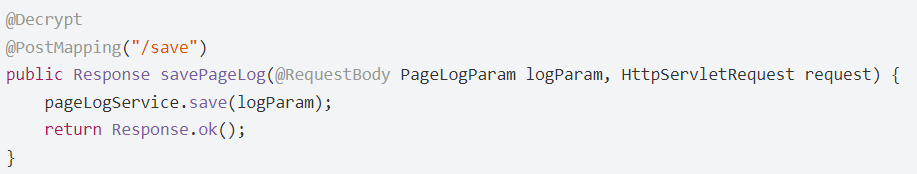
网页或者app，只要抓下包就可以清楚的知道这个请求获取到的数据，因此数据很容易被他人爬取：

数据的安全性非常重要，特别是用户相关的信息，稍有不慎就会被不法分子盗用，所以我们对这块要非常重视，通过以下方法可以有效提高本系统的安全性。

1. 对请求和响应都进行加解密操作

这种方法可以有效对抗爬虫以及调用接口，因为即使成果爬取到数据或者调用了接口，

获得的依然是加密后的数据，有效防止了核心数据泄露。

由于我们后端使用了Springboot框架，其本身提供了相应的配置包用于实现前后端传输的加密，只需要引入相关包，之后在每个Controller前添加@Decrypt注释，之后前端提交的数据需要按照AES加密算法，进行加密，然后提交到后端，后端这边会自动解密，然后再映射到参数对象中。  


同时加密key可以手动配置，通过添加：



对于前端，其需要做到：

在渲染界面前解密后端传来的加密数据&&在POST请求前加密数据

以axios来作为请求数据的框架，前端可以用axios的拦截器来统一处理加密解密操作。

2. 接口安全

为防止接口被恶意调用，用户在成果登录后后端将为其分配一个Session，之后用户所有高级功能在涉及接口调用时都首先需要检查该Session，这样可以有效避免

3. 账户安全与Dos攻击防范

在注册时我们使用邮箱认证，通过邮箱认证保障用户账号安全，实现密码修改和忘记密码找回密码等操作。登录时需要输入验证码并在前端进行验证。只有验证成功才能发出HTTP请求。

对于防范可能的Dos或者DDos攻击，我们打算通过一下两点进行防范：

1. 在部署时选择租用阿里云等商业化云服务器，通过其自带的安全防范体系避免攻击带来的损伤。
2. 在短时间收到一个ip过量请求时，对其进行一段时间的封禁。

4. 防范XSS攻击

XSS攻击就是攻击者通过各种办法，在用户访问的网页中插入自己的脚本，让其在用户访问网页时在其浏览器中进行执行。

XSS防御的总体思路是：对输入(和URL参数)进行过滤，对输出进行编码。也就是对提交的所有内容进行过滤，对url中的参数进行过滤，过滤掉会导致脚本执行的相关内容；然后对动态输出到页面的内容进行html编码，使脚本无法在浏览器中执行。虽然对输入过滤可以被绕过，但是也还是会拦截很大一部分的XSS攻击。

### 4.2.2 可靠性

可靠性是指系统在一段时间内保持运行的能力。可靠性是指一个系统在规定的时间间隔内不会无法执行其预期功能的概率。

提高系统可靠性的方法，主要是两种：避错和容错。测试是最常用的一种避错技术，而容错则一般使用冗余来实现。冗余技术是容错的主要手段。主是通过对资源的冗余，包括硬件、软件、信息、时间等，可以使系统的容错性得到较大的提高。

### 4.2.3 可用性

可用性指的是在某个考察时间，系统能够正常运行的概率或时间占有率期望值。换句话来说就是提高系统高并发的能力。

1、通过使用Nginx实现反向代理、负载均衡，以此来降低延迟，提升吞吐量，避免出现单个节点过多请求的情况。具体实现请参考文档的部署部分。

2、平台通过使用MySQL主从复制的方式增加系统的可用性。由于单台MySQL数据库服务器不能满足数据备份和读写频率高的要求，此时我们需要搭建一个数据库集群来来解决这一问题。数据库之间的同步可以采用主从复制来完成。

## 4.3 系统质量 – 保障性

首先，我们在编写代码时要做到程序有足够的鲁棒性，具体表现为：

1. 对大多数可能的异常通过try-catch模块进行相应，保证系统不会因为一些异常导致运行崩溃，同时在catch相应模块需要对异常情况进行足够完整的说明，方便debug时快速发现问题的关键与解决问题
2. 通过日志，各模块功能运行状态检测线程等手段保证在本项目部署运行后，维护人员可以快速获得当前系统各部分运行的状态信息，对可能发生问题的部分提前预警，防患于未然。
3. 建立责任分配制度，在个人负责的代码出现问题时做到快速相应与纠错。

## 4.4 用户质量 – 易用性

我们的目标在于提升用户的使用体验，因此我们可以从以下几个角度出发：

1、前端界面首先要在尽量简洁清爽的前提下保证主要核心功能（即搜索）的突出显示，各界面的渲染要注重设计美学，并且做到风格的统一。

2、对于新用户，应该在其注册后通过“新手指引”快速引导其掌握本站相关功能的使用。同时页面下方应提供“设置”与“常见问题模块”，前者可以帮助用户进行个性化设置，后者可以为常见问题提供非人工的解答。

3、用算法或者机器学习手段分析用户的使用习惯，争取做到智能推送其感兴趣的学术内容

# 五、横切关注点

## 5.1 安全

### 5.1.1 使用Spring Security + OAuth2提供安全的认证和访问控制

我们将权限按照游客，已登录用户，已认证用户，管理员四种身份划定相关权限，不允许跨越权限，下述框架实现安全的认证和授权功能。。

Spring Security是针对Spring项目的安全框架，也是Spring Boot底层安全模块默认的技术选型。他可以实现强大的web安全控制。对于安全控制，我们仅需引入spring-boot-starter-security模块，进行少量的配置，即可实现强大的安全管理。

OAuth2.0则是目前流行的授权机制。致力于简化客户端开发人员的工作，同时为Web应用程序，桌面应用程序，移动电话和客厅设备提供特定的授权流程。在同时保护用户的数据以及权限。

二者结合可以保证用户按照安全规则，以安全策略可以访问且只能访问自己权限内的资源。

### 5.1.2 数据加密

为了保护用户的敏感数据，我们需要对信息进行加密处理，其中我们主要以SHA256散列加密函数为主，非对称加密（RSA）为辅。

单向散列加密是指通过对不同输入长度的信息进行散列计算，得到固定长度的输出。这个散列计算过程是单向的，即不能对固定长度的输出进行计算从而获得输入信息，是不可逆的。

我们可以应用这个特性来保存用户的密码等敏感信息，只在数据库中存储加密过的数据。同时在从前端到后端传输数据的过程中，也进行相应的加密，最大程度上在传输过程中保护用户的信息安全。

而在条件允许的情况下，我们可以采用非对称加密（RSA）用于前端与服务器之间的数据加密。前端获取公钥，后端生成对应的私钥，传输时提高安全性。密码只在入库前进行进行SHA256加密。多种加密方式提升了数据的安全性。

## 5.2 缓存

### 5.2.1 前端数据缓存

在前端的数据缓存上，我们主要使用Json web token来确定用户的状态，保存包括id在内的重要信息，同时确保了前后端的快速访问。

而对于在前端中传递的一些不重要的信息，我们则采用html技术里面的localStorage来保存一些不太重要的信息，作为前端的数据缓存。

### 5.2.2 Redis数据库缓存

我们使用Redis分布式缓存作为数据库缓存。Redis是一个高性能的key-value的数据库，可以存储多种样式的数据。其可以处理高并发的情况，比之Mysql最高500-1000的连接量，Redis的性能有极大的提高。在实际使用中，当我们有数据查询请求时，先到Redis缓存中查寻，查找不到再去Mysql中查询。因此Redis会存储最近的，被多次使用的数据。为了实现Redis和Mysql的同步问题，有包括Redis设置缓存时间，Mysql自动同步Redis等等操作。使用Redis数据库缓存可以很好的处理高并发的问题，同时提高系统整体性能。

## 5.3 日志

log4j2作为spring官方推荐的一种日志管理系统，和spring有很好的兼容性。在log4j2中，一共有五种log level，分别为TRACE, DEBUG,INFO, WARN, ERROR 以及FATAL。

log4j2优点的优点包括：

1. 异常处理，在logback中，Appender中的异常不会被应用感知到，但是在log4j2中，提供了一些异常处理机制。

2. 性能提升， log4j2相较于log4j 1和logback都具有很明显的性能提升，后面会有官方测试的数据。

3. 自动重载配置，参考了logback的设计，当然会提供自动刷新参数配置，最实用的就是我们在生产上可以动态的修改日志的级别而不需要重启应用——那对监控来说，是非常敏感的。

4. 无垃圾机制，log4j2在大部分情况下，都可以使用其设计的一套无垃圾机制，避免频繁的日志收集导致的jvm gc。

其中日志等级包括六种：

1. FATAL：用在极端的情形中，即必须马上获得注意的情况。

2. ERROR：显示一个错误，或一个通用的错误情况，但还不至于会将系统挂起。这种程度的错误一般会触发邮件的发送，将消息发送到alert list中，运维人员可以在文档中记录这个bug并提交。

3. WARN：不一定是一个bug，但是有人可能会想要知道这一情况。如果有人在读log文件，他们通常会希望读到系统出现的任何警告。

4. INFO：用于基本的、高层次的诊断信息。在长时间运行的代码段开始运行及结束运行时应该产生消息，以便知道现在系统在干什么。但是这样的信息不宜太过频繁。

5. DEBUG：用于协助低层次的调试。

6. TRACE：用于展现程序执行的轨迹。

对于日志的良好使用可以帮助我们更好的监控到可能出现的bug，提升系统的可靠性等等。

## 5.4 异常处理

### 5.4.1 服务器宕机

服务器宕机时，判断宕机原因，并作出相应处理。

1、服务器环境的客观原因。比如机房断电导致的服务器断电、机房温度过高，导致的服务器死机、关机等。这种情况相对少见，若遇到可联系与服务器公司相关的人员。

2、服务器不堪负重。这种情况是比较常见的主要原因，网站流量暴增、程序中毒、遭受攻击等大规模高消耗服务器资源情况，而导致的服务器资源耗尽不敢负重，终无法响应和死机。

这时应该获取第一现场日志，不仅能更快的将宕机数据推送给业务或运营感知并处理，还能降当前服务器状态给保存下来，便于时候分析及后续的故障预测。

### 5.4.2 请求未返回

前端向后端发送请求信息时，未能接收到后端发出的返回信息，这时大致可分为两种情况。

1.前后端接口编写错误。这主要是因为在编写前后端接口时，接口本身编写错误或者接口不匹配。这时应该检查前后端接口的正确性，并相互协商保证接口的一致性。同时前端跳转到一个错误页面，提醒用户系统出现问题，请稍后再试。

2.后端服务器出现问题。由于后端服务器出现问题，平台功能无法正常运转。这时应排查服务器的问题，同时前端跳转到一个错误页面，提醒用户系统出现问题，请稍后再试。

### 5.4.2 记录异常

发生异常之后，需要保存异常发生时的情况，记录在日志系统中，供开发人员后续根据错误发生时的数据进行相应的维护和升级。尽量避免此类情况的再次发生。