

杭州电子科技大学

本科毕业设计

(2018 届)

题 目	基于 SpringBoot 的响应式技术博客的设计和实现
学 院	计算机学院
专 业	软件工程
班 级	14108413
学 号	14108337
学生姓名	朱文赵
指导教师	马虹
完成日期	2018 年 5 月

诚 信 承 诺

笔者谨在此承诺：本人所写的毕业论文《基于 SpringBoot 的响应式技术博客的设计和实现》均系本人独立完成，没有抄袭行为，凡涉及其他作者的观点和材料，均作了注释，若有不实，后果由本人承担。

承诺人（签名）：

年 月

摘 要

随着现在信息化时代的发展，越来越多的人喜欢把自己的一些见解和心得记下来，放到一个平台上，希望别人来观看和交流。所以博客孕育而生，特别是技术类的博客，更是给开发者带来更多学习他人经验与技术的机会和与提供和别人交流和分享的一个平台。

本课题将许多当前比较热门的技术框架有机的集合起来，比如 **Spring boot**、**Spring data**、**Elasticsearch** 等。同时采用 **Java8** 作为主要开发语言，利用新型 **API**，改善传统的开发模式和代码结构，实现了具有实时全文搜索、博客编辑、分布式文件存贮和能够在浏览器中适配移动端等功能的响应式技术博客。

在保证功能能够全部实现的同时，使用一些比如回归测试和后台测试驱动开发，以及最后的 **postman** 接口测试和浏览器适配移动端等成熟的测试技术和方法。测试结果符合预期，系统功能完善，有良好的用户体验。确保系统上线后是一个能够给用户分享经验和心得，以及学习他人优秀成果的系统。

关键字：Spring Boot; Blog; Java; Elasticsearch;

ABSTRACT

With the development of the information age, more and more people like to put some of their views and thoughts down, put them on a platform, and hope to see and communicate with others. So blogs are born, especially technical blogs, to give developers more opportunities to learn from others' experience and technology, and a platform to communicate and share with others.

This topic organically integrates many popular technical frameworks, such as Spring boot, Spring data, Elasticsearch and so on. At the same time, Java8 is used as the main development language, and the new API is used to improve the traditional development mode and code structure. A response technology blog with real-time full text search, blog editing, distributed file storage and the ability to fit mobile terminal in browsers is realized.

While guaranteeing the full realization of the work, some mature testing techniques and methods such as regression testing and back-end test drive development, and the final postman interface test and browser fit mobile end are used. The test results meet the expectations, the system is functional and has a good user experience. To ensure that the system is on-line, it is a system that can provide users with experience and experience, and learn from others' outstanding achievements.

Keywords: Spring Boot; Blog; Java; Elasticsearch;

目录

1 绪论	1
1.1 项目背景	1
1.2 本课题的国内外研究现状	1
1.3 本课题研究意义	2
1.4 课题研究内容	3
1.5 本论文组织结构	3
2 开发环境与相关技术	4
2.1 Windows 系统介绍	4
2.2 Java 语言	4
2.3 Spring boot 框架	5
2.4 Spring security 框架	5
2.5 Spring data 框架	5
2.5.1 Spring data jpa	5
2.5.2 Spring data elasticsearch	6
2.5.3 Spring data mongodb	6
2.6 Maven 工具	7
2.7 Lombok 工具	7
2.8 设计模式	7
3. 系统分析	8
3.1 可行性分析	8
3.1.1 技术可行性分析	8
3.1.2 经济可行性分析	8
3.1.3 操作可行性分析	8
3.1.4 法律可行性分析	8
3.2 用户需求分析	9
3.2.1 用户管理模块	10
3.2.2 全文搜索模块	11
3.2.3 博客管理模块	13
3.2.4 评论管理模块	14

3.2.5 点赞管理模块.....	15
3.2.6 权限设置模块.....	16
3.2.7 个性推荐模块.....	17
3.2.8 分类管理模块.....	18
3.2.9 标签管理模块.....	19
3.3 非功能需求分析.....	20
3.3.1 安全性.....	20
3.3.2 简约规范.....	20
3.3.3 流畅高效.....	20
3.3.4 良好的用户体验.....	20
3.4 开发和运行环境.....	20
3.4.1 开发环境.....	20
3.4.2 运行环境.....	21
4.系统设计.....	22
4.1 系统架构设计.....	22
4.1.1 模型层.....	22
4.2.2 控制层.....	23
4.2.3 视图层.....	23
4.2 功能模块设计.....	24
4.2.1 用户管理模块.....	25
4.2.2 全文搜索模块.....	25
4.2.3 博客管理模块.....	25
4.2.4 评论管理模块.....	26
4.2.5 点赞管理模块.....	26
4.2.6 权限设置模块.....	26
4.2.7 个性推荐模块.....	26
4.3 数据模型设计.....	26
4.3.1 博客模型.....	27
4.3.2 用户模型.....	28
4.3.3 博客评论模型.....	28
4.3.4 评论模型.....	29
4.3.5 分类模型.....	29
4.3.6 博客点赞模型.....	29
4.3.7 点赞模型.....	29

4.3.8 权限模型	30
4.3.9 用户权限模型	30
5. 系统实现	31
5.1 数据获取层	31
5.1.1 用户管理模块	31
5.1.2 博客管理模块	32
5.1.3 点赞管理模块	36
5.1.4 评论管理模块	37
5.1.5 分类管理模块	40
5.2 数据构建层	43
5.2.1 用户管理模块	43
5.2.2 博客管理模块	44
5.2.3 分类管理模块	44
5.2.4 点赞管理模块	45
5.2.5 评论管理模块	45
5.2.6 权限管理模块	46
5.3 数据展示层	47
5.5.1 登录界面	47
5.5.2 注册界面	47
5.5.3 博客主页界面	48
5.4.4 创建博客界面	49
5.5.5 查看博客界面	50
5.5.6 点赞界面	50
5.5.7 评论界面	51
5.5.8 用户主页界面	51
5.5.9 个人设置界面	52
5.5.10 头像变更切图界面	52
5.5.11 分类管理界面	53
6. 软件测试	54
6.1 测试环境	54
6.2 测试方法	54
6.3 测试用例设计	56
6.4 测试结果	59
7. 总结和展望	60

7.1 毕业设计工作总结	60
7.2 展望	60
致谢	61
参考文献	62

1 绪论

1.1 项目背景

早期的博客为了给不同的人提供一些不同的东西，会挑选一些特别的网站，并作简单的介绍。久而久之，慢慢的形成了不同种类，从不同的角度，又可以分为多种类型，如用户角度，可以分为个人博客、企业博客；或者按功能分，可以分为技术类的博客，微型博客^[1]。

而技术博客在众多博客类型中又有着一个不一样的地位。这些博客记录了博主曾经走过的路，是如何考研，做项目，学习的历程，算法心得，思考的过程，面试的技巧，如何进入谷歌，Facebook，微软这样的大公司实习或在学习或者工作中学到的新技术或是碰到的技术难题以及解决的方法。每个人对待各种技术的看法，已经消耗的时间可能都不一样，但他们对此都有相同的态度。那就是都希望通过自己的坚持和努力，解决一个个难点后，把这种喜悦或者说是成就感分享出来，让别人也能都一起学习和借鉴这个历程，分享自己的快乐，分享自己的收获。如笔者一样，相信很多人在遇到一些“疑难杂症”的时候，都会去网上苦苦寻求，当他们读到了这些优秀的文字的时候，就能够快速解决问题或者得到一个问题的解决思路。因此这些人在学会知识的同时，也懂得了记录和分享的重要性^[2]。

1.2 本课题的国内外研究现状

国内外有很多优秀的技术博客，比如说国内的 CSDN，简书；国外的 Tumblr, Github Pages, The Hacker Blog, Stack OverFlow。同时还有一些个人站点，如：程序猿 DD，廖雪峰的官方网站和阮一峰博客等。这些都不失为值得学习借鉴的博客，对比他们的技术特点，不难总结出以下几点：

（1）出众的设计界面

注重“表面”的开发，页面相当于一个门面。用户的体验，笔者觉得是第一要义，良好的设计能够给自己的门户网站带来不错的流量。在保证前端美观的同时，还提供了性能良好的后台，能够把内容更好的展示出来。一般国外的技术博客都有这个特点；相比之下国内的有些博客做的不是很好，究其原因国内技术太过于守旧，没有拥抱新的技术和框架，不愿做出一些新的尝试^[3]。

（2）友好程度不一

说到友好程度，那必须得说一下国内的一些优秀的博客站点，总是有人能够在一定的时间，去学习一些比较热门的技术，然后分享出来。其中不乏有英文的文档，他们大多会翻译一部分，然后结合自己的见解，分享出来给大家学习和借鉴。当然用户也可以第一时间采用谷歌搜索或者 Stack OverFlow 中搜索自己的想要的答案，也可以在相关的社区提问，或者在 Github 中提一个 issue，通过别人交流，得到一些解答的思路^[4]。不过后者会对用户上网会有一定的要求，同时也需要有一点英文功底。

（3）自定义的成分

国外很多有一些博客能够支持用户自定义代码块，让用户小规模建站，让他们能够自由发挥，得到一个与众不同的 Blog。而国内的博客目前通常还不支持这一小功能。

（4）机器学习

在这一点上国内外很多站点都是支持的，因为他们有庞大的数据量，能够有足够的训练集，来丰富他们的机器学习模型^[5]。以此来构建用户画像，智能推荐给用户真正感兴趣的东西，而不是一旦用户搜索了一个东西，就疯狂地、无意义地的推荐这些信息。

（5）是否提供注册

有的技术博客不提供注册，只能博主使用，如一些个人站点。人们只能以游客的方式访问，但是绝大部分都提供了评论的功能，能够保证博主和其他人的交流度^[6]。

1.3 本课题研究意义

随着现在信息化时代的发展，越来越多的人喜欢把自己的一些见解和心得记下来，放到一个平台上，希望别人来观看和交流。很多人都会选择一些现有的博客站点，当然还是有很大一部分人选择自己建站，那建站就会有很多方式^[7]。比如采用开源的博客框架，利用他人的服务器，快速搭建，不过这样会有一些限制，如存储容量是固定值，另外还存在网络问题，有一些部署在国外，国内访问会很有一点棘手，同时也无法自定义一些十分个性化的功能。于是就有了另一种方式，自己从零开始搭建，采用自己喜爱的技术和框架，制作一个真正意义上的属于自己的博客。

于是本毕业设计本着给大家提供一个更多技术交流平台思路，采用一些目前比较热门的框架，深入用户的需求，给用户带来一个易于使用的博客。因此选用 Spring boot 来作为技术支持，让建站变得更加容易，更易于部署，更加容易和其他技术栈结合，让人们能够专注于开发逻辑，技术的选用，而不在搭建和繁杂的配置项上^[8]。同时希望能够实现将用户的信息同步到一些大数据搜索平台上，给用户带来一种快速，响应式的体验。

1.4 课题研究内容

本毕业设计选用 SpringBoot 框架，结合 Thymeleaf, SpringData, SpringSecurity, Elasticsearch 等技术，旨在为技术人员设计并实现一款用于记录并分享技术文档的技术博客。通过该技术博客，方便技术人员记录自己工作和学习过程中的点滴，不断地进行技术的总结和积累，从而提升自己的综合能力，并通过博客这一平台，把自己的知识、经验、教训分享给大家，为志同道合者提供一个相互交流、共同学习的平台，促使更多的人共同进步^[9]。学习到别人的一些良好的设计思路、编码风格和优秀的技术能力，使笔者的设计初衷。本系统主要面向 web 端的用户，希望能给用户更多的学习和交流的选择。

本课题进行了网络问卷调查和梳理用户的一些想法后，明确了用户对于 web 端的博客有着一种简单快速优质的愿望。此次设计中将主要实现用户模块、角色模块、权限模块、博客模块、评论模块、点赞模块、分类模块、标签模块、搜索模块等功能模块。会根据用户平常阅读的喜好，搜索的习惯，智能推荐给用户相应的优质内容。方便技术人员能够更加自由的书写博客，以及分享自己的学习心得，结识一些有相同爱好的极客^[10]。后期也可根据用户的需求做适当调整。

1.5 本论文组织结构

论文各个章节安排如下：

第一章：绪论。主要介绍本课题的项目背景、国内外的研究现状，课题的研究意义，并阐述了本论文的主要研究内容。

第二章：开发环境与相关技术介绍。本系统为 Web 端系统，虽在 Windows 下完成开发，但基本没有平台的约束性。主要语言采用 Java8，结合 Spring 的技术栈和 Maven 等一些工具进行开发。

第三章：系统分析。对系统的开发可行性进行了分析，描述了用户的需求和介绍了开发和运行环境。

第四章：系统设计。对整个课题结合用户需求，来有效的进行拆解，拆解为各个功能模块，并对数据库、界面和系统架构进行了设计。

第五章：系统实现。对各个模块的实现过程进行细化的描述，阐述其中的设计原理和思想以及具体的逻辑，将最终实现的成品展现出来。

第六章：软件测试。对各个模块进行相应的测试用例设计，已经采用相应的工具进行自动化测试和压力测试。

第七章：总结与展望。对整个课题的开发过程进行一个总结和展望。

2 开发环境与相关技术

2.1 Windows 系统介绍

Windows 系统的建立，是为了人们能够简单，快捷的操作人们的操作系统，所有的事情都只需要人们通过 GUI 来进行，而不是像程序员们需要通过命令行来进行交互。比如极其强大的资源管理器，能够将不同的内容穿梭在不同的文件夹中。此外 window8 开始，已经支持屏幕触控，不过要想进行该操作，只能在微端，类似平板或者支持可触控的显示器等条件下，才能正常开启。

Windows8 中加入了一种新的开机模式，叫做混合启动，能够保证用户在下次开机时能够快速进入，类似于一种休眠的状态，大幅度提高了开机的时候硬盘读取和初始化的时间。

当然，人们的开发者通常不会满足于这些简单的需求，因为总是有一些东西是 GUI 无法满足的，所以微软的 Windows 提供了一套 cmd 命令来供开发者使用。比如查看两个不同的网络是否互通的 ping 操作和查看当前的网络连接状态的 netstat 的命令^[11]。

2.2 Java 语言

Java 是一种能够运行在 JVM 虚拟机中语言规范，它是一种高级程序设计语言，可以在不同的平台下运行^[12]。具有很多优秀的特性，它结合了一些语言的优点，抛弃了一些语言的设计不足，比如说 C#语言中的菱形集成，同时也包装了一些比较难以理解的原理，比如指针。可谓是取其精华，去其糟粕。

它是一种面向对象的设计语言，比起一些面向过程的语言，有很好的可阅读性，能够让程序员们进行简单和复杂中进行自由组合出一套优雅的程序。

而本课题主要采用 Java8 编写，大量采用了 Java8 的新特性，比如时间 API 和 Stream 流式操作（Lambda 表达式），比起以往的代码（以前的 Java 版本）代码量能够在保持简洁下而不丧失良好的语义^[12]。其中 Lambda 表达式算是一个核心突破，它吸取了一些先驱如 JavaScript 或者符合 JVM 语言规范的后来者如 Scala 等设计理念，从而产生了现在这一个优秀的 API，极大地简化了程序的逻辑复杂度。另外 Java8 也在改进以前的一些 API，比如集合中的 map，在 hash 碰撞之后在链表和红黑树的切换机制，这个是在 Java7 及以前版本是没有的。

2.3 Spring boot 框架

本课题采用了当前 Java 最热门的框架 Spring boot 来进行编写，主要目的是利用它的开箱即用的特性，它能够快速的集成其他框架，让开发者能够专注于自己的业务逻辑，而不是陷入一大堆的配置文件中或者项目搭建中^[13]。

Spring boot 本质上并不是一个新的框架，他只是在原有的 Spring 框架中进行了进一步的封装，将它打造成为一个具有良好开闭原则的分布式框架。

该框架现在保持了极大的活跃度，其衍生产品 Spring cloud 也是在微服务中站住一个不可动摇的位置，比起阿里的 Dubbo 有着一些良好的生态支持^[14]。

总的来说该框架在将来必将取代现有的开发模式，类似于 SSM（Spring+Spring MVC+Mybatis）的开发模式，这将成为一种主流，所以笔者选择了拥抱将来，拥抱新的技术。

2.4 Spring security 框架

Spring security 在本课题中占了很大的部分，有着不可动摇的地位。它主要用来控制用户的访问权限，相比以往的进行 Filter 过滤器、Interceptor 拦截器或者 AOP 面向切面编程，更加的灵活和自由，同时也更加简单。它支持 Spring EL 表达式，能够结合注解的同时做到更加细粒度的方法层面的控制^[15]。

作为同为开源权限控制框架的 Shiro，它相比 Spring security 来说就会有点太过于“粗糙”，不能达到后者的粒度控制大小，而且在生态支持力度，也没有后者这么良好。后者能够和 Spring 有着天然良好的集成，是作为开发者权限控制的首要选择。

2.5 Spring data 框架

2.5.1 Spring data jpa

Spring data jpa 作为 Spring 整合 Hibernate 的一套优质的 ORM（Object Relational Mapping）框架，它即保持了 JPA（Java Persistence Api）的设计规范，比起原始 JDBC（Java DataBase Connectivity）具有良好的操作性和可读性，极大的提高了开发者的开发效率^[16]。

该框架在持久层中占据着一个重要的地位，它即支持简单的单表操作也支持复杂的多表连接已经手写 Sql 的支持。同时和 Mybatis 都有同样的设计风格，那就是面向接口编程，能够让用户只需要定义接口，而不用去管他的实现。只不过前者能够支持通过方法名的命名规则来进行 Sql 的转化，具有非常好的可扩展性^[17]。

同时该框架也是在微服务中首选的框架之一，因为微服务中因为需要各种各样的链路调用，所以它需要保证各个服务之间延迟时间在可控范围，然后该框架能够很好保持这种良好特性，同时也能让编程更加容易。

2.5.2 Spring data elasticsearch

Spring data elasticsearch 同样是 Spring 开源出来的针对大数据搜索平台 Elasticsearch 的 API 封装。它将一个文档化和结构化的数据变成，人们开发者能够通过人们熟悉的方式，类似于 Spring data jpa 的操作，进行操作，从而屏蔽掉一些复杂的实现细节，这也是一种优秀的设计模式构建方式。

它提供了和 elastic 的官方文档同步的操作操作 API，保证开发者在阅读文档后能够利用 Java elasticsearch api 进行平滑的操作。同样他也支持原始的 JSON 查询体进行查询，给用户提供了极大的便利^[18]。

当然该框架是对 Elasticsearch 的封装，其关键还是在 Elasticsearch 的本身，该搜索平台是现在的优秀分布式搜索平台之一，同比还有 Solr 等。但前者具有更加灵活的 API 和搜索设计，后者呢语法会稍微复杂一点，且实时搜索效率不高，没有像 Elasticsearch 那么高速的分布式搜索效率。Elasticsearch 相比较传统的关系型数据库（Relational Database）在搜索效率上有着本质上的提升，因为前者在插入数据的时候，都是将该数据变为非结构化的数据，如 JSON，没有复杂的表关联，而是一些平铺开的数据，同时加入索引（Index），加快了数据的访问速度。不在是关系型数据库中的结构化检索，同时搜索语法也比较后者有着本质的提升^[19]。

2.5.3 Spring data mongodb

本课题采用 Spring data mongodb 来操作 Mongodb 存储一些图片文件信息，利用它对二进制文件良好存储性能构建人们的文件服务器。它和上面提到的 Elasticsearch 都有一个共同点，那就是存储结构都是非结构化的数据，都是采用 key-value 的 JSON 来存储，从某种意义上来说，提高了可读性和程序的适配性。相比 xml 的方式具有很大传输优势，这就是现在基本主流程程序都是通过 JSON 来传输的原因。同样都是以文档（Document）来定义属性，保持了 NoSQL（Not Only SQL）的高效性和灵活性^[20]。

在当前的大数据时代，这种非结构化的数据能够帮助开发者进行一些良好的数据分析操作，Mongodb 也是同样具有全覆盖索引操作，对检索的支持也是十分良好。Spring data mongodb 在保证不变该数据库的操作原理的同时，封装出对 Java 程序员友好的操作 API，这也体现了 Java 多态的设计原则，极大简化了开发者开发的效率。

2.6 Maven 工具

Maven 这个工具简直是 Java 开发者的福音，开发者再也不用手动去晚上一个一个的去下载 jar 包，只需要在 pom 文件中添加 jar 的坐标，然后点击引入，Maven 就会自动从你的 setting.xml 文件中配置的仓库去查找 jar 包，并进行下载。

除了刚刚讲的下载功能，它还有三个生命周期，默认（Default），清洁（Clean）和站点（Site），能够帮助开发者进行不同的操作^[21]。这个在传统的项目中是做不到的，开发者只能手动去用 Java 命令去构建，发布到相应的站点。这在现在提倡自动化发布的时代，很明显是不符的。

同时在 Maven 中由于加 jar 包比较自由，所以会出现 jar 包冲突的情况，不过这个依然可以很优雅的解决，现在的各个 IDE 都可以添加 Maven 插件，来查看当前的 jar 包是否重复。这个在原始的 Java Web 项目中简直是一个噩梦，开发者必须要去查看 Tomact 中打包的项目中的 lib 目录 jar 包是否有重复或者有高低版本问题，当那种 jar 包内部有冲突的时候，这种情况将很难查找。

2.7 Lombok 工具

本课题采用 Lombok 来简化代码，目的是减少 java 中实体类的“模板”信息，比如 get、set 方法和构造函数，通过注解形式，在类编译之后自动生成，不用手动维护。这样做能够保证在改字段声明的时候，不用手动去改其他的关联信息。

同时该 jar 包，还支持 Builder 构建者模式，能够通过添加 @Builder 注解，动态生成构建者信息，在类的字段特别多的时候，将会解救开发者的代码，不再看起来那么的愚拙。如果没有这个 jar 包的支持，开发者要想实现这种构建器模式，也必须要十分呆滞的再代码中添加样板代码，这样费时也费力，可以说是一个苦力活。

2.8 设计模式

本课题全部采用面向接口的设计理念，同时采用策略模式来构建不同情况下不同的代码分支，保证代码清晰可读。同时代码保持开闭原则，尽量做到用实体类来进行操作，这样防止今后该栋接口，只需要在类中加减字段，而不需要改动方法传输结构^[22]。

3. 系统分析

3.1 可行性分析

3.1.1 技术可行性分析

技术操作上，本课题采用 spring boot 为框架主体结合 Thymeleaf, SpringData, SpringSecurity, Elasticsearch 等技术。这些都是一些开源技术，各个社区都十分的活跃，互联网和书籍中都有十分成功的案例教学。难点是如何才能保证它们在有机的结合起来，发挥出自己的作用，做到各司其职^[23]。

整体课题都基于 mvc 设计模式来构建，前后端传输采用通用数据格式 JSON，做到前后端分离，降低耦合度^[24]。

由于自己在平时有在学习这些框架，平时会去阅读官方文档和在社区中进行交流和别人进行讨论，对整体要用到的技术内容，都有着一定了解和使用率。因此整个案例的构建在技术上可行的。

3.1.2 经济可行性分析

由于整个课题用到的技术都是开源的，所以基本不会存在购买资源的问题。同时编译器也是用的 IDEA，目前享受学生正版优惠，因此也不会有版权问题。

同时开发本项目虽然会有一定的服务器开销，但系统一旦完成，将会给各个开发者提供一个新的选择来进行接触和交流，具有不错的用户流量。因此本课题在经济上是可行的。

3.1.3 操作可行性分析

本课题是用 web 端展现给用户，界面美观，具有良好的操作风格，简单容易上手，提示信息良好。因此，在操作上是可行的。

3.1.4 法律可行性分析

本项目在法律范围内进行部署，不会触碰到法律的底线，传播积极向上的内容，为净化网络环境做出贡献。因此在法律层面是可行的。

3.2 用户需求分析

摸清用户的需求在开发过程中是十分重要的，具有是非常关键的地位，一旦偏离了用户的需求和用户的痛点，那么开发的结果将会不尽人意。因此做好用户需求分析，将会给后续的开发过程中，树立一个目标。用户的整体活动图如图 3-1 所示：

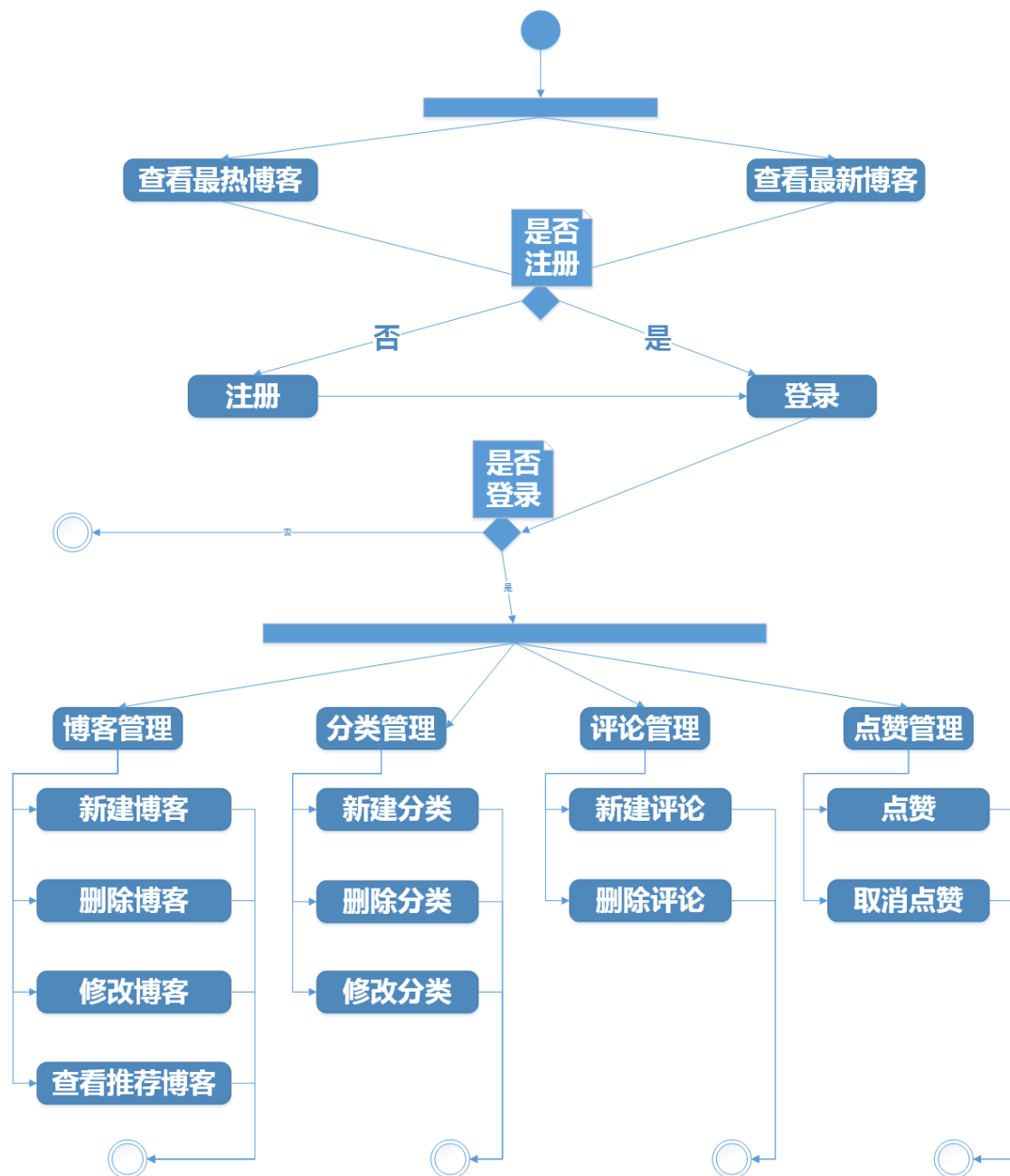


图 3-1 系统整体活动图

根据需求分析，可以分为用户管理，全文搜索，博客管理，评论管理，点赞管理，安全管理，权限管理，个性推荐等用例，用例描述如图 3-2 所示：

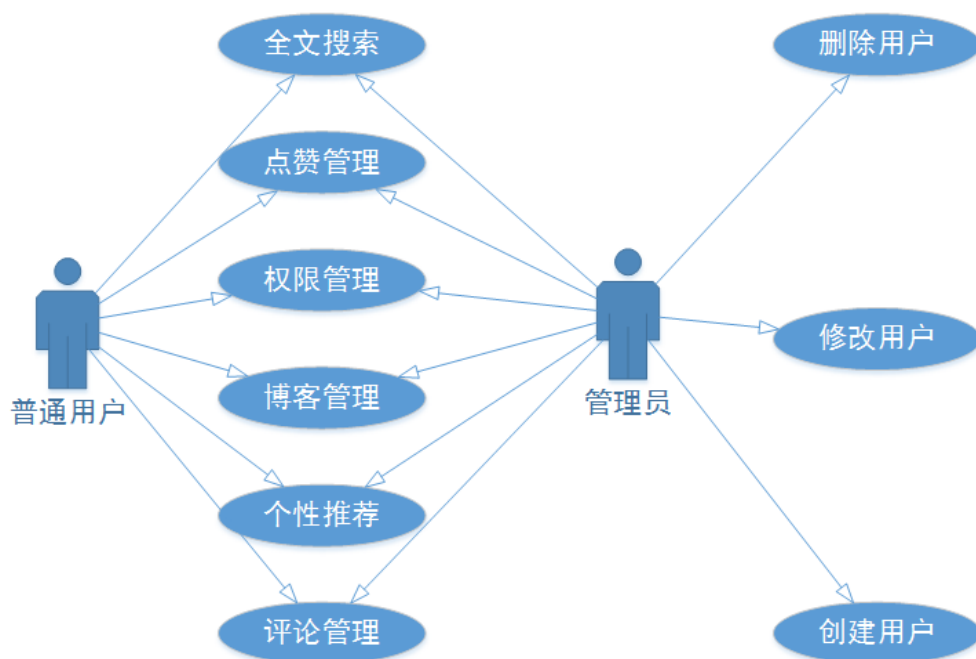


图 3-2 系统总体用例图

3.2.1 用户管理模块

用户管理是在本系统中虽然只是一小部分，但是却把控着用户的操作权限，决定这用户是否权限操作一些模块。首先用户应当先注册，然后登录，登录后可以对自己的个人信息进行修改，如头像，昵称等个人信息。用户的简单活动图如图 3-3 所示：

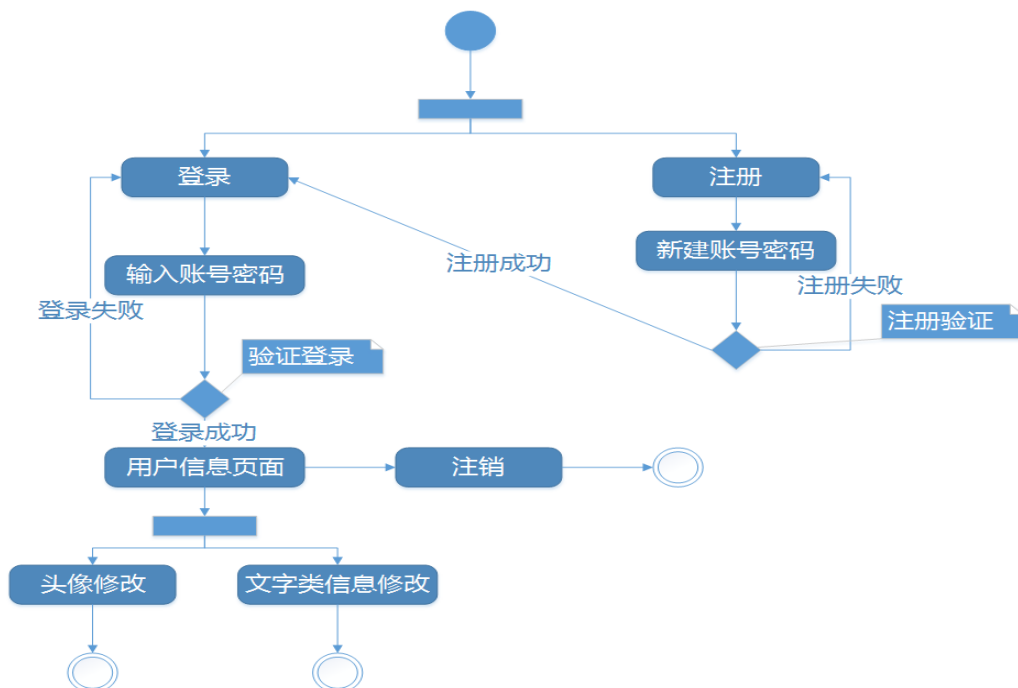


图 3-3 用户管理活动图

用户管理模块包含用户登录、注册、注销、还有查看用户信息，用例描述如图 3-4 所示：

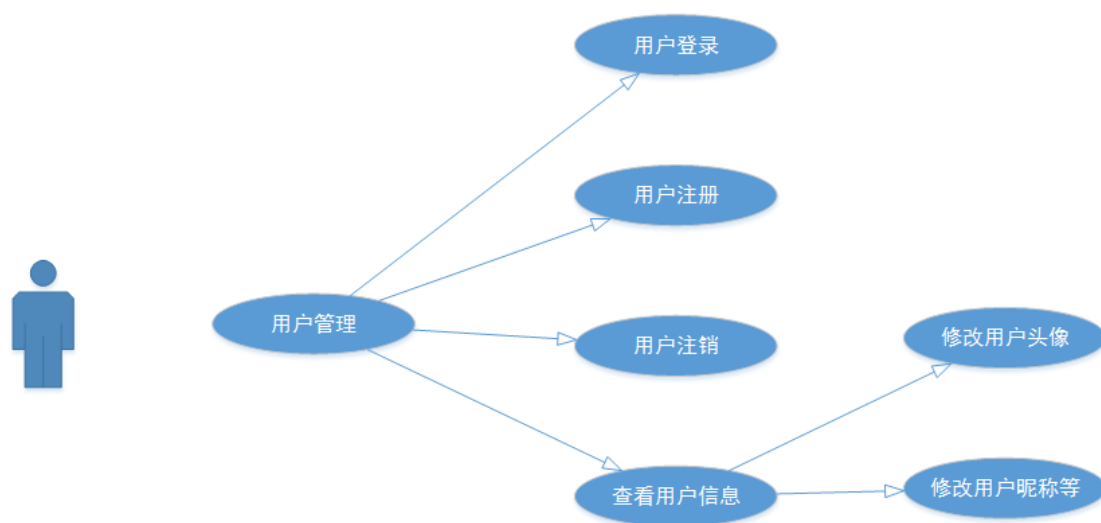


图 3-4 用户管理用例图

用户登录：用户在登录时输入邮箱和密码，验证信息是否和数据库匹配，匹配则登录至主页，否则则登录失败。登录成功后，系统会写入用户信息到 cookie 中，方便用户在不注销的情况下，下次能够正常登录进来。

用户注册：用户首次进入本系统，可以选择注册一个账号，成为本系统的用户，方便查看更为丰富的内容。用户输入自己的账号和密码，验证通过后，即可成为本站的用户。

用户注销：用户登录后，若想要退出该系统，可以选择注销操作，注销后将会清除 Cookie 和 Threadlocal 中用户信息的值。然后跳转到用户登录页，下次进入本站，要想查看一些比较丰富的内容，必须要进行登录。

查看用户信息：用户登录后可以选择查看或修改自己的用户信息，其中包括对自己昵称等信息的修改和修改头像。

3.2.2 全文搜索模块

全文搜索模块是本课题中具有特色的一个模块，主要包括最新、最热查询、以及输入关键字查询和查看自己推荐的博客内容。同时还有显示活跃的用户和热门标签等。全文搜索活动图如图 3-5 所示：

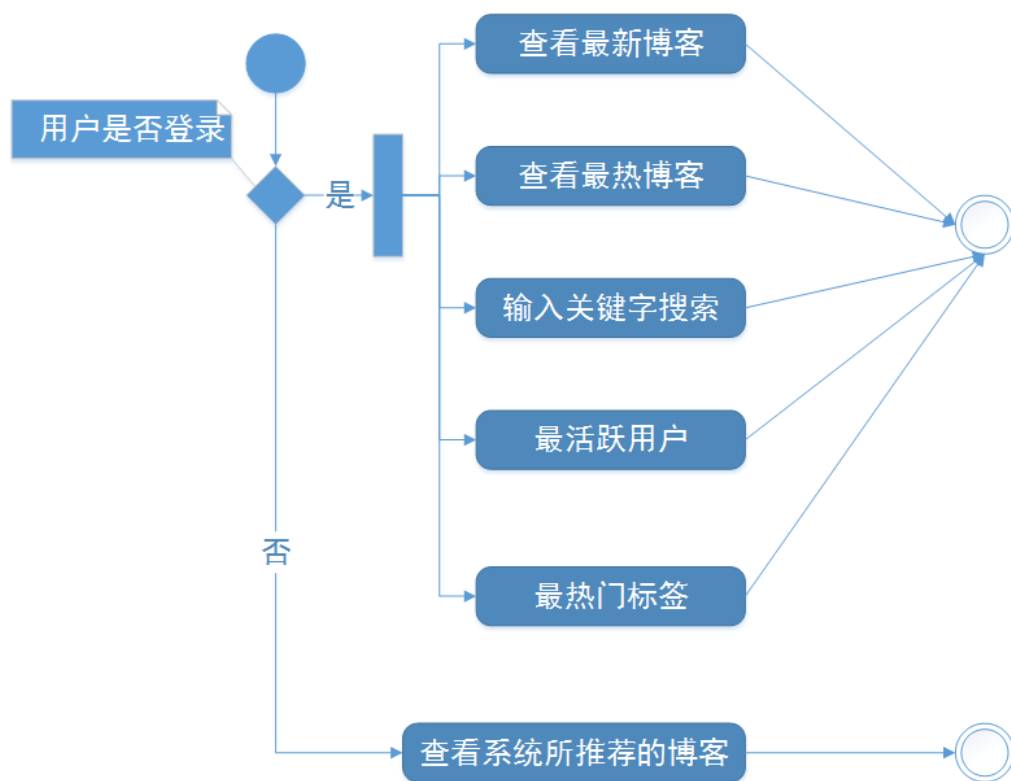


图 3-5 全文搜索模块活动图

用户有些操作必须要在登录后才能操作，比如查看智能推荐的内容等。简单用例图如图 3-6 所示：

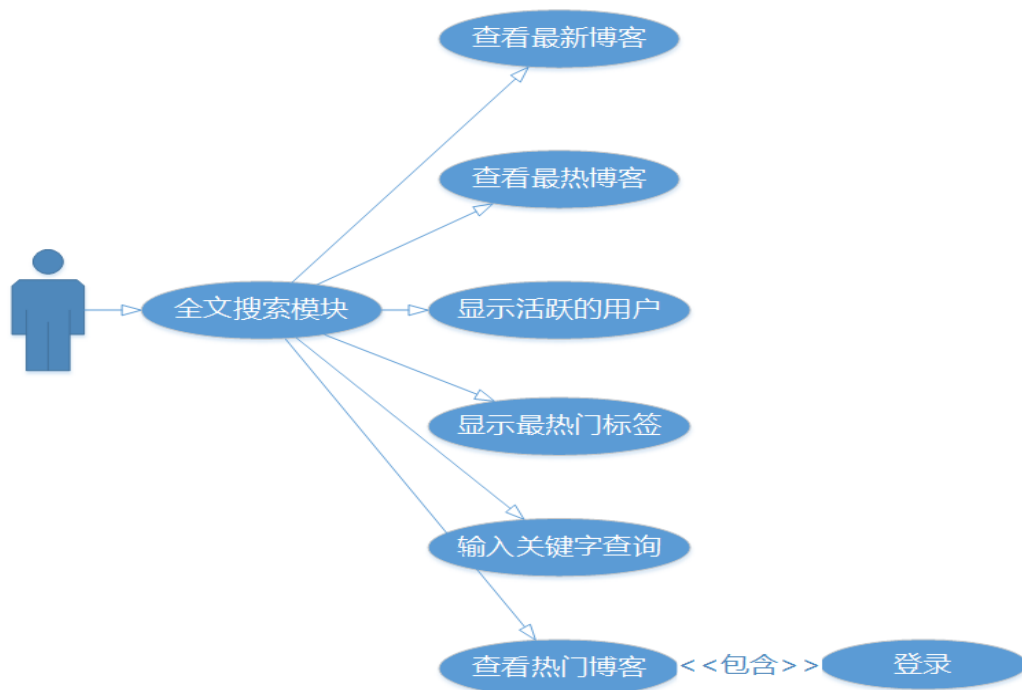


图 3-6 全文搜索模块用例图

查看最新博客：用户可以点自己最新按钮查看目前博客中最新创建的博客，看到热门新鲜出炉的分享内容。

查看最热门博客：用户可以点自己最热按钮查看目前博客中最火热的博客列表，看到现在热门最关注的的博客内容。

输入关键字搜索：用户可以输入自己想要查找的内容，系统会根据关键字查找内容和标题匹配的博客内容，然后显示给用户。

显示热门标签：本课题将会显示最热门的用户自定义标签，同时也可以点击这些标签进行搜索。

显示活跃的用户：本课题将会显示最活跃的用户，根据创建博客的多少进行聚和统计。

查看推荐的内容：用户必须要先登录才能查看该模块下的内容，系统会根据用户的喜好，推荐给用户一些他自己喜欢看的内容。

3.2.3 博客管理模块

用户在登录之后，可以创建、修改和删除博客，对自己的博客进行一个管理。博客管理活动图如图 3-7 所示：

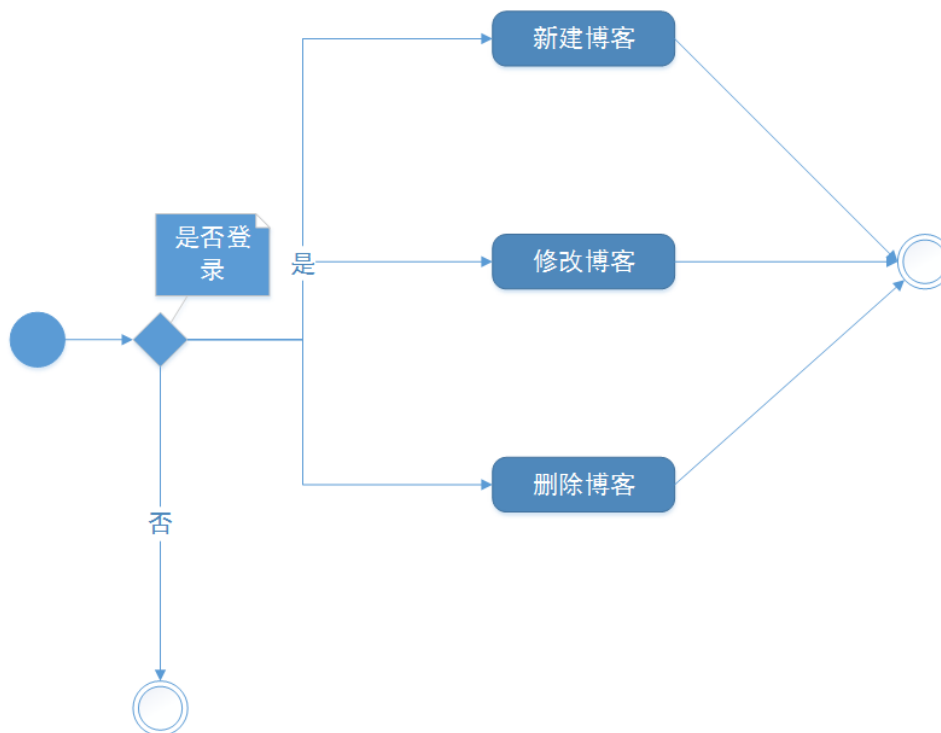


图 3-7 博客管理活动图

用例图如图 3-8 所示：

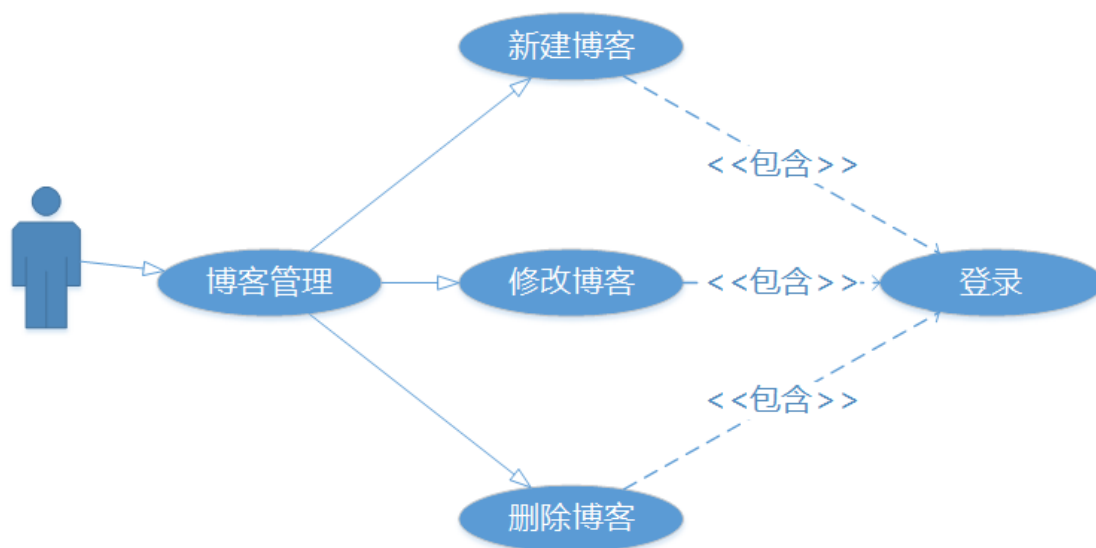


图 3-8 博客管理用例图

新建博客：用户登录后，可以新建博客，与别人分享自己的经验和快乐。

修改博客：用户登录后，可以点击自己之前的博客，进行修改。

删除博客：如果用户不想在管理这个博客，那么可以选择删除它。

3.2.4 评论管理模块

用户在点击一个博客后可以对改博客进行评论，同时也可以删除该评论。评论模块活动图如图 3-9 所示：

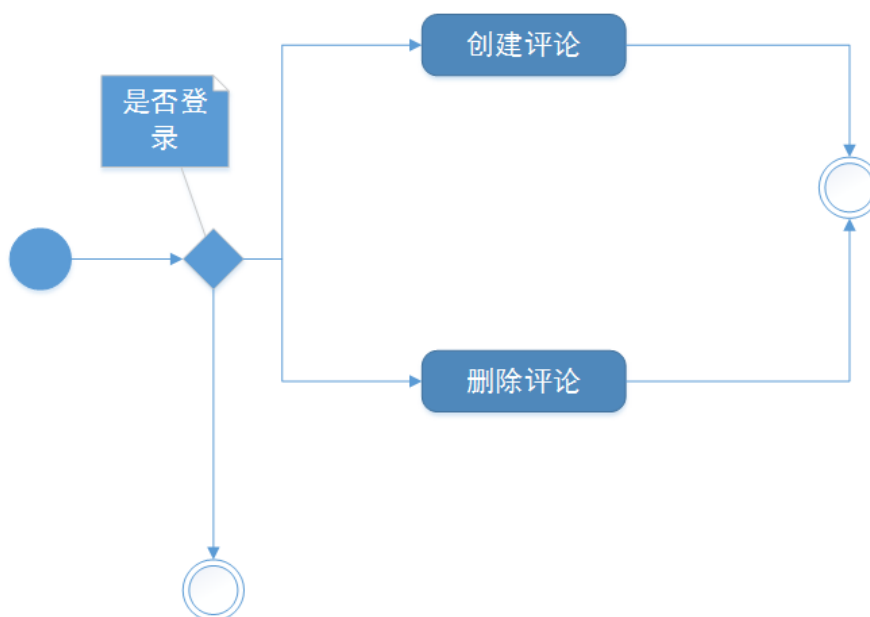


图 3-9 评论管理活动图

用例图如图 3-10 所示：

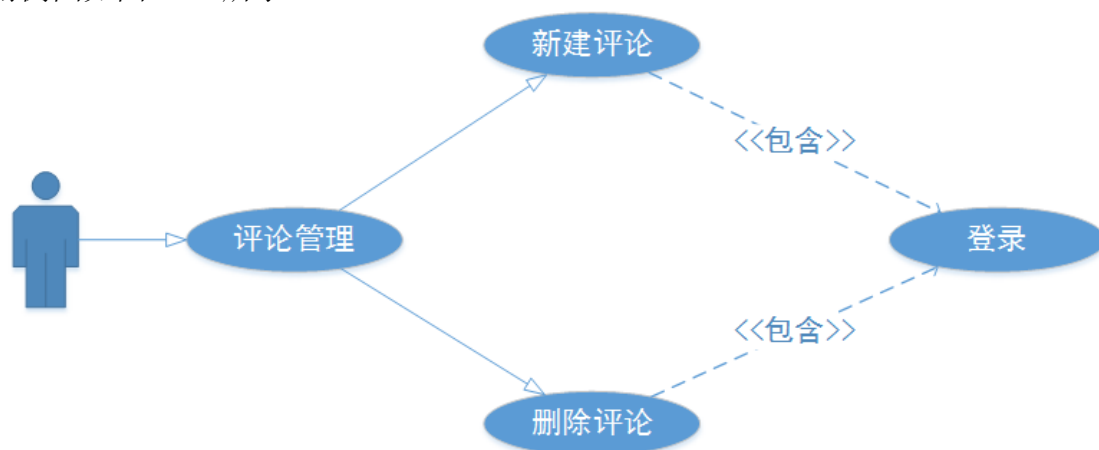


图 3-10 评论管理用例图

新建评论：用户可以对有想法的博客进行评论，和博主进行互动。

删除评论：用户可以删除自己不满意的评论。

3.2.5 点赞管理模块

点赞管理模块分为取消点赞和点赞，用户可以对自己喜欢的博客进行点赞操作，点赞管理活动图如图 3-11 所示：

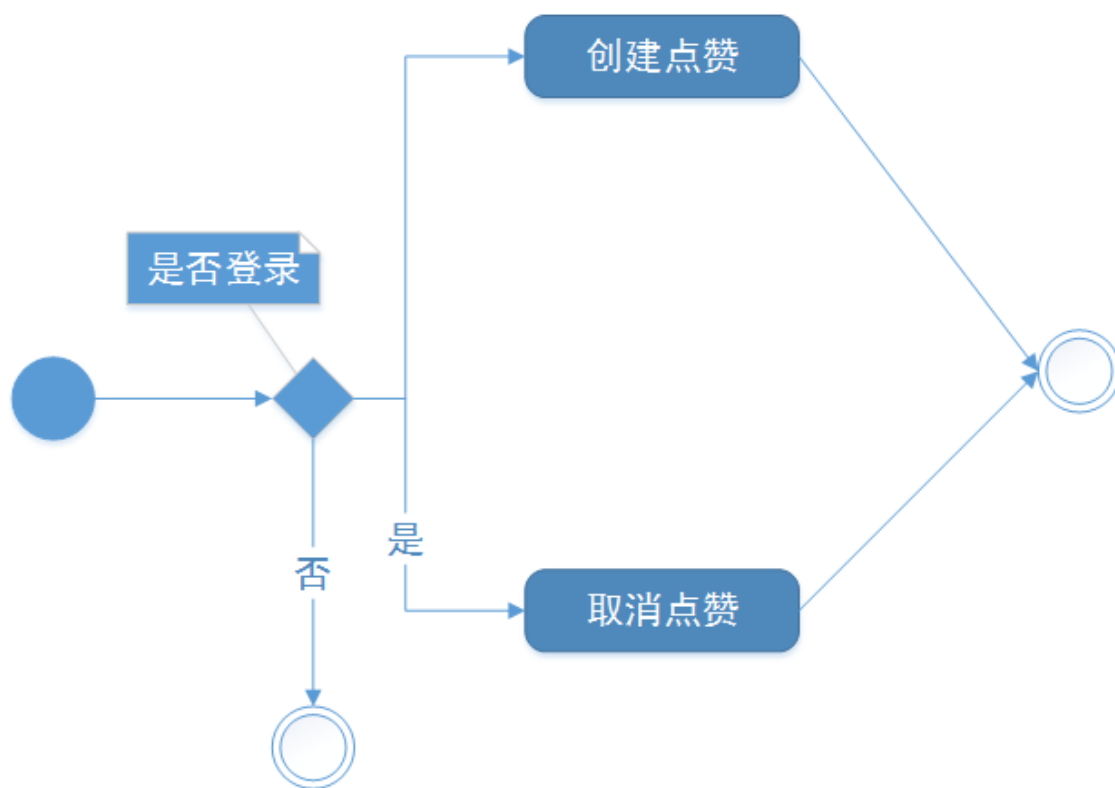


图 3-11 评论管理活动图

点赞用例图如图 3-12 所示：

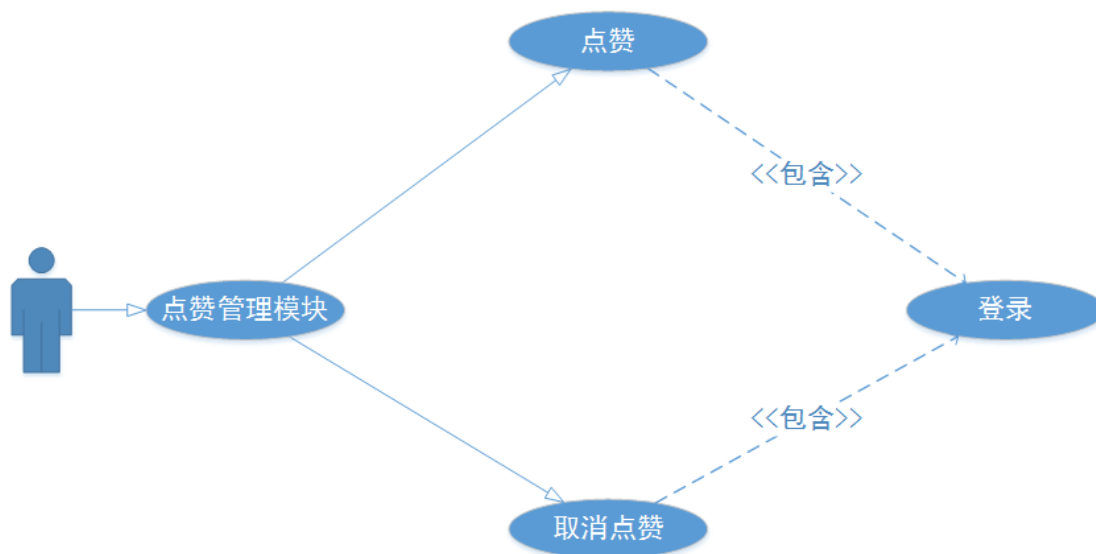


图 3-12 评论管理用例图

点赞：用户可以对自己喜欢的博客进行点赞，为博主的博客贡献一份力量。

取消点赞：同时用户可以对自己不再感兴趣的blog进行取消点赞。

3.2.6 权限设置模块

本文权限控制是非常关键的一部分，是过滤用户请求的重要环节，保证不同状态的用户能进行不同的操作。主要的权限分为管理员权限和用户权限。权限模块活动图如图 3-13 所示：

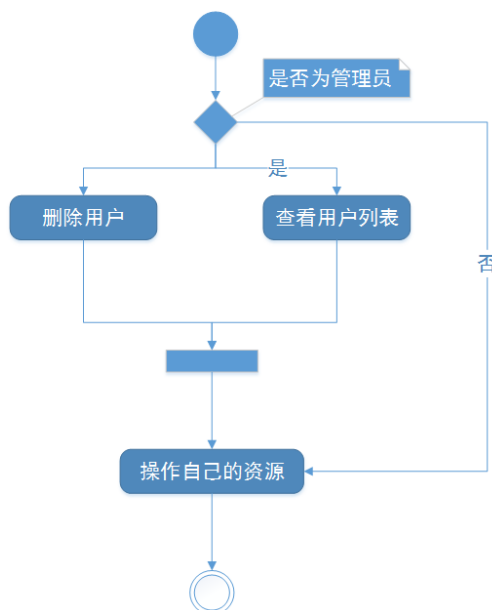


图 3-13 权限管理活动图

用例图如图 3-14 所示：

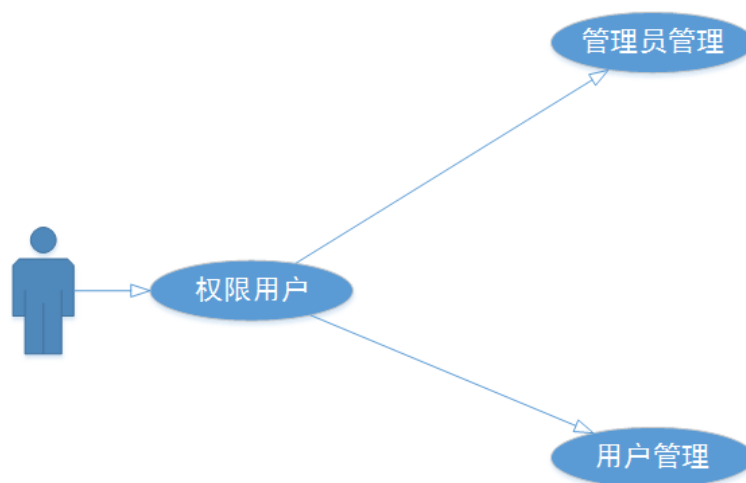


图 3-14 权限管理用例图

管理员权限：管理员可以查看整个用户列表，对用户进行一个管理操作，能够删除用户等。

用户权限：用户可以查看所有的博客，但是只能修改和删除自己的博客，同样点赞和评论也是只能删除自己锁创建的。保证了用户的权限的隔离。

3.2.7 个性推荐模块

个性推荐模块，是本课题的一个重点，能够在用户登录后，根据用户的喜好，智能推荐出按照匹配率排序的博客列表。个性推荐活动图如图 3-15 所示：

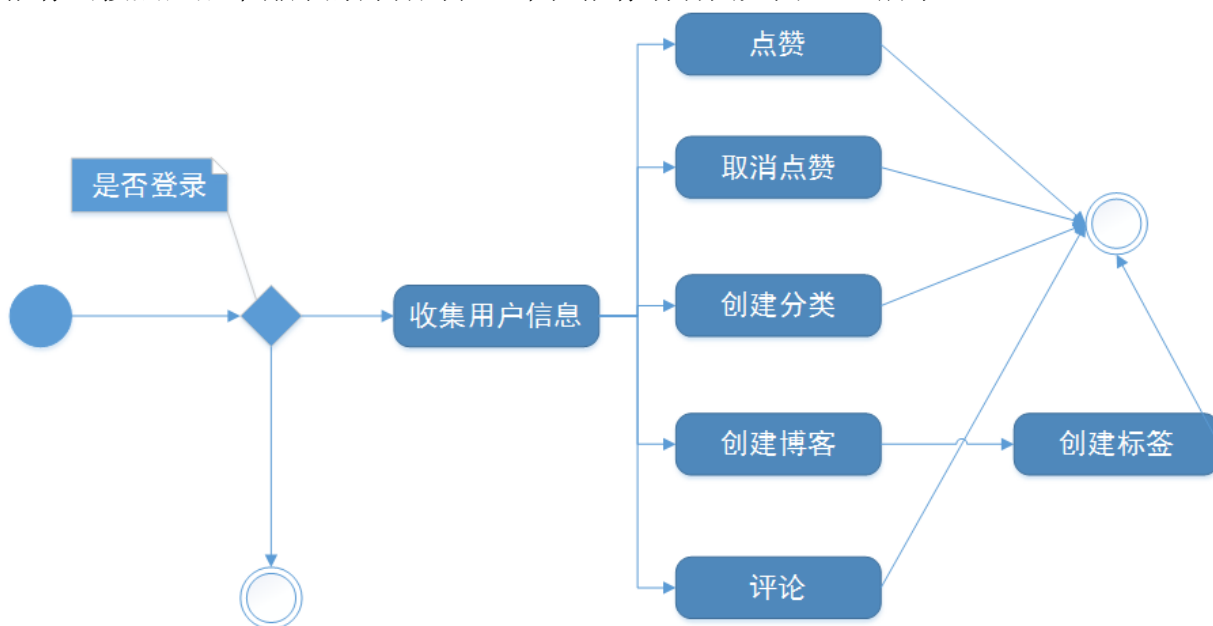


图 3-15 个性推荐模块活动图

用例图如图 3-16 所示：



图 3-16 个性推荐模块用例图

个性推荐模块，人们需要采集用户的信息，后台会根据用户的点赞、评论、创建的博客和创建的标签来动态的解析，生成一个推荐列表给用户。

3.2.8 分类管理模块

用户可以在创建博客的时候，选择创建好的分类，用来绑定每个博客属于什么分类。便于用户进行管理自己的博客列表。分类管理活动图如图 3-17 所示：

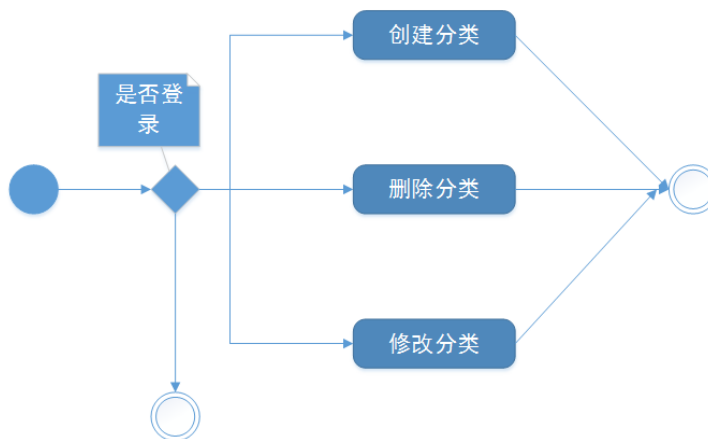


图 3-17 分类管理模块活动图

用例图如图 3-18 所示：

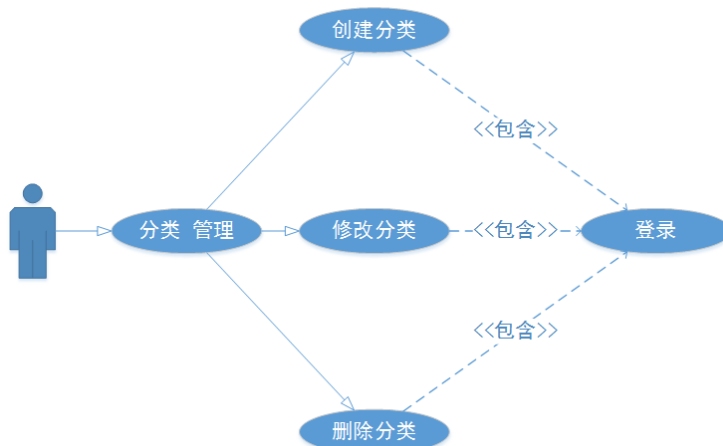


图 3-18 分类管理模块用例图

新建分类：用户可以根据自己的喜好，新建博客分类，用来管理自己的博客，让自己的博客看起来更加的条理化。

删除分类：用户如果对之前的分类不喜欢，那么可以删除该分类，该分类下的所有的博客列表会到一个未分类的栏目中。

修改分类：用户可以对自己所创建的博客分类进行修改，将他改为自己所想要的名字。

3.2.9 标签管理模块

标签管理区别于用户管理，它主要是为了细化每个博客的所属标签，让同一个博客可以根据用户的喜欢，添加自己的标签。同时也方便后台进行数据的清洗和过滤。标签管理活动图如图 3-19 所示：

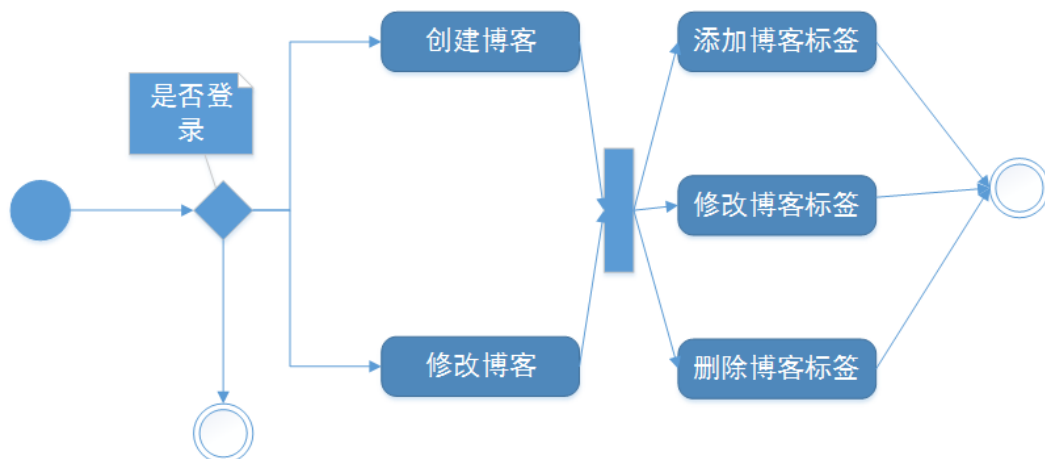


图 3-19 标签管理模块活动图

用例图如图 3-20 所示：

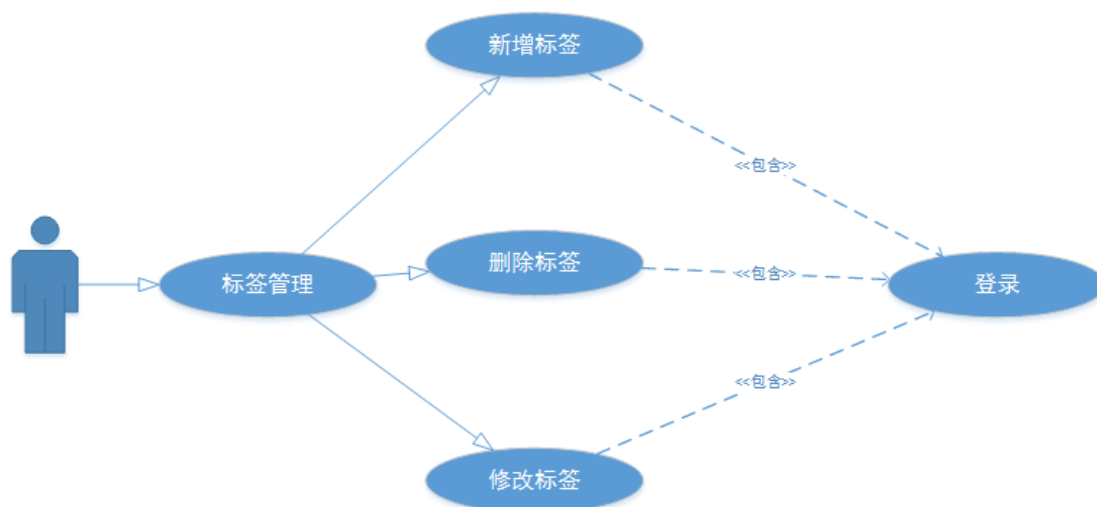


图 3-20 标签管理模块用例图

新建标签：用户在新建或者修改博客的时候可以点击标签模块，进行新建标签，丰富博客的内容。

删除标签：用户同样可以删除标签，剔除不合适的内容。

修改标签：同样用户也可以对原有的标签进行修改，使其更加语义明确和丰满。

3.3 非功能需求分析

3.3.1 安全性

对用户的密码都进行加密的处理，避免明文传输，同时采用 `spring security` 作为权限管理框架，对用户的权限进行了方法级别的隔离，保证每个请求都是经过校验的，同时也保持了代码的简洁性。

3.3.2 简约规范

在保证系统能够正常运行的时候，同时也保证了前后台代码的简洁性，保证后续重构和优化的便捷性。对用户的界面也采用响应式的规范来编写，界面美观。满足简约高效的规范。

3.3.3 流畅高效

同时，本课题在保证用户能够正常运行的同时，也保证了性能的问题，采用 `elasticsearch` 进行全文搜索，相较于关系型数据库就会有很大的性能提升，保证了用户流畅的操作感觉。同时也能准确的搜索到用户想要的。

3.3.4 良好的用户体验

满足以上几点条件的同时，本课题也保证用户能够拥有良好的用户体验，无论是在视觉（前端）还是在性能（后台）都能够抓住用户的心。

3.4 开发和运行环境

这里对开发本课题的开发和运行环境做出简单的介绍。

3.4.1 开发环境

设备：联想 G510

CPU：Intel(R) Core(TM) i5-4210M CPU @ 2.60GHz 2.60 GHz

内存：12G

硬盘：512G

系统：Windows 8.1 专业版

网络调试：Chrome 66.0.3359.139（正式版本）（64 位）

JDK：Java8

调试设备：IDEA 2018.1

3.4.2 运行环境

系统要求：没有要求

设备要求：Chrome55 及以上

屏幕：无要求，可自动适配

硬盘：无要求

网络：wifi 或者 3G 网络或者更快

4.系统设计

本课题遵循 MVC 设计模式和常用设计模式如策略模式和工厂模式设计理念，主要可以分为系统架构设计、功能模块设计、数据模型设计、数据模型设计、界面设计、系统框架设计。因此本章将会从这几个方面进行描述。

4.1 系统架构设计

系统的架构采用 MVC 的设计模式分为三层，主要为数据模型层、视图层和控制层，如图 4-1 所示。系统的数据集模型层主要有 mysql、elasticsearch 和 mongodb 提供支持，视图层采用 web 界面，同时适配移动端和网页端，控制层采用 spring boot 作为主体框架，采用 java 编写业务支撑。系统结构图如图 4-1 所示：

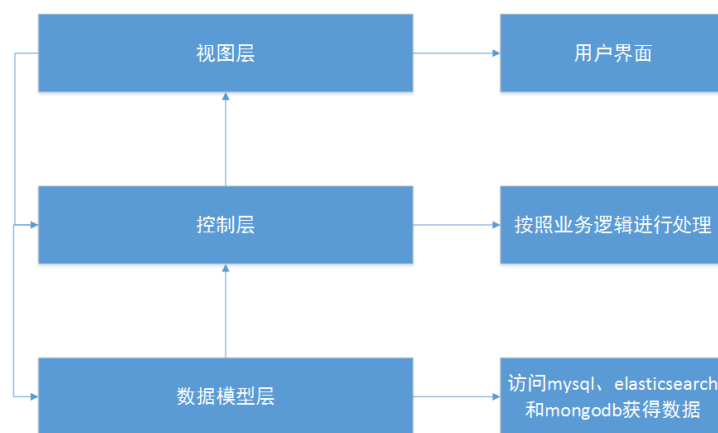


图 4-1 系统结构图

4.1.1 模型层

数据模型层主要是为了支撑整个系统，是一个非常重要的部分，它作为数据库和 Java 代码之间的连接桥梁，可以说是至关重要的。采用多个数据库来支持和维护该数据模型层，利用不同数据库的特性，比如关系型数据库的事务性和非关系型数据库的高效性等，发挥各个数据库的特长，为人们的系统带来非常不错的性能和功能的支持。

它作为整个系统中属于最为底层的模块，通过 Java 中类的概念和数据库中的表一一对应，屏蔽了一些具体的细节，开发者只需要通过类与类之间的调用即可操作数据库中的表。其中类和数据库表具体的对应关系如下所示：

- 1) Blog 类对应数据库中 blog 表。
- 2) Catalog 类对应数据库表中 catalog 表。
- 3) Comment 类对应数据库表中 comment 表。
- 4) User 类对应数据库表中的 user 表。
- 5) Vote 类对应数据库表中的 vote 表。
- 6) EsBlog 类对应 elasticsearch 中的 blog 索引 (index)。
- 7) File 类对应 mongodb 中的 File 文档 (document)。

4.2.2 控制层

控制层是整个模块的核心，他连接了模型层和视图层，采用 JSON 的数据格式进行传输，便于移植和拓展。它负责将视图层传过来的请求分发到各个路由下，使用相应的方法进行解析和返回需要的数据。具体的控制类如下所示：

- 1) AdminController: 管理员控制器，用例处理管理员相关逻辑。
- 2) BlogController: 博客控制器，用例处理博客相关逻辑。
- 3) CatalogController: 分类控制器，用例处理分类相关逻辑。
- 4) CommentController: 评论控制器，用例处理评论相关逻辑。
- 5) MainController: 主控制器，用来处理一些公共的页面跳转。
- 6) UserController 用户控制器，用来处理用户相关逻辑。
- 7) UserspaceController: 用户控件控制器，用来处理用户主页的控制器。
- 8) VoteController: 点赞控制器：用来处理点赞的相关逻辑。

4.2.3 视图层

视图层主要是采用 HTML5、CSS3 和 ES5 以及 thymeleaf 来编写，同时采用 bootstrap 来适配移动端，给用户带来更好的体验。整个视图层包括用户的视觉上的效果，点击事件的处理和数据的渲染三个部分。本课题中的视图主要包括如下几个：

- 1) admin
index.html 主要用来显示管理员的一些操作的界面
- 2) users
add.html 主要用来为管理员进行添加用户的界面
edit.html 主要用来为管理员进行编辑用户的界面
list.html 主要用来显示所有的用户列表。

3) userspace

avatar.html 主要用来编辑头像

blog.html 主要用来显示每个用户下的博客列表

blogedit.html 主要用来标记自己的博客信息

catalog.html 主要用来创建自己的分类信息

profile.html 主要用来编辑自己的个人信息

u.html 查看自己的所有信息

4) fragments

foot.html 公共部分用来处理整个系统的底部

header.html 公共部分用来处理整个系统的头部

page.html 公共分页的模块

5) other

index.html 主要用来处理博客主页页面

login.html 主要用来处理登录的页面

register.html 主要用来处理注册的页面

search.html 主要用来处理搜索的页面

4.2 功能模块设计

本系统的定位是一个供技术人员交流和分享的技术博客。通过该技术博客，方便技术人员记录自己工作和学习过程中的点滴，不断地进行技术的总结和积累，从而提升自己的综合能力，并通过博客这一平台，把自己的知识、经验、教训分享给大家，为志同道合者提供一个相互交流、共同学习的平台，促使更多的人共同进步。

本毕业设计中将主要实现用户模块、角色模块、权限模块、博客模块、评论模块、点赞模块、分类模块、标签模块、搜索模块等功能模块。会根据用户平常阅读的喜好，搜索的习惯，智能推荐给用户相应的优质内容。方便技术人员能够更加自由的书写博客，以及分享自己的学习心得，结识一些有相同爱好的极客。后期也可根据用户的需求做适当调整。

具体的功能模块图如图 4-2 所示：



图 4-2 系统功能结构图

4.2.1 用户管理模块

用户可以进行登录、注册和搜索其他用户，超级管理员具有用户的增删改查的超级权限。同时本课题可以实现自动登录的功能，只要用户不注销，那么下次他可以免登录，直接变成已经登录的状态。

4.2.2 全文搜索模块

此模块为本项目的一个亮点，采用大数据分析平台 `elasticsearch`，将后台数据库中的博客信息，用户信息，标签信息。用户在全文搜索的时候，后台直接从 `elasticsearch` 中进行查询，增加了响应速度和响应精确性。比直接用 `sql` 语句中的 `like` 查询面积要广，而且还具有分词效果，匹配度高的显示在前面，低的显示在后面，真正达到全文搜索。全文覆盖的效果。

4.2.3 博客管理模块

此模块是本系统的一个重点。用户可以自由的对博客增删改查，同时可以对博客设置相应的标签。还可以对博客进行最新最热排序。同时系统还会对每一个博客进行

阅读量的统计，让别的技术人员能够更加直观的感受该博文的受欢迎程度，同时也给全文搜索给了一个指标。

4.2.4 评论管理模块

用户可以对博客进行评论，让更多的人看到自己的想法，能够与别人进行交流和互动。同时也能和博主有着思想的碰撞。

4.2.5 点赞管理模块

通评论模块一样，用户可以对自己喜欢的博文进行点赞，或者对自己已经不感兴趣的博文进行取消点赞。同时也给全文搜索提供一个硬性指标。

4.2.6 权限设置模块

用户只能对自己编写的博客才有删除的权限，对其他的无法进行删除操作。用户只能在自己拥有该权限的时候，才能进行相应的操作，这个通过 `SpringSecurity` 来进行管理。且权限粒度除了区分到方法级别上，还设置到了类级别，比如“`/admin`”和“`/u`”这两个路由下的资源必须要具有相应的角色才能够访问。

4.2.7 个性推荐模块

主要是靠后台进行数据筛选，比如必要在改用户是为登录的状态，而不是为匿名用户（未登录）的状态。具体要采集的数据主要为该用户自己创建的博客分类和具体属于的标签，还有就是用户电子和评论过的博客列表，并对这些博客列表进行筛选，选择具体属于的分类和标签。再去 `elasticsearch` 中进行查询该特征值的博客列表。

4.3 数据模型设计

根据本课题的需求，确定其数据模型如图 4-3 所示。



图 4-3 数据模型图

4.3.1 博客模型

博客模型是用来存储和博客相关的一些属性，主要包括博客的编号、评论量、博客内容（markdown）、创建时间、网页内容、阅读量、摘要、标签、标题、点赞量、目录编号和用户编号。博客模型主要属性如表 4-1 所示。

表 4-1 博客模型表

字段名称	字段类型	字段说明
id	bigint	博客编号
comment_size	int	评论量
content	longtext	博客内容

续表 4-1

字段名称	字段类型	字段说明
create_time	datetime	创建时间
html_content	longtext	网页内容
summary	varchar	摘要
tags	varchar	标签
title	varchar	标题
vote_size	int	点赞量
catalog_id	bigint	分类编号
user_id	bigint	用户编号

4.3.2 用户模型

用户模型是用来存贮用户的一些基本信息，包括用户编号、头像地址、电子邮箱、用户名、密码。昵称等信息。用户模型主要属性如表 4-2 所示。

表 4-2 用户模型表

字段名称	字段类型	字段说明
id	bigint	用户编号
avatar	varchar	用户头像
email	varchar	用户邮箱
name	varchar	昵称
password	varchar	密码
username	varchar	用户名

4.3.3 博客评论模型

博客评论模型主要是为了存放博客的评论的信息，这里没有放在 vote 表中，而是单独抽离出来一张表格。主要包括博客编号和评论编号。博客评论模型如表 4-3 所示。

表 4-3 博客评论模型表

字段名称	字段类型	字段说明
blog_id	bigint	博客编号
comment_id	bigint	评论编号

4.3.4 评论模型

评论模型主要包括评论编号、评论内容、创建时间和用户编号。评论模型的主要属性如表 4-4 所示。

表 4-4 评论模型表

字段名称	字段类型	字段说明
id	bigint	评论编号
content	varchar	评论内容
create_time	datetime	创建时间
user_id	bigint	用户编号

4.3.5 分类模型

分类模型主要为了存放每个用户的具体的分类信息，方便用户按照分类进行管理博客。主要包括分类编号、分类名字和用户编号。分类模型主要属性如表 4-5 所示。

表 4-5 分类模型表

字段名称	字段类型	字段说明
id	bigint	分类编号
name	varcahr	分类名字
user_id	bigint	用户编号

4.3.6 博客点赞模型

博客点赞模型主要为了存放博客的点赞信息，具体是那个用户进行的点赞操作。主要有博客编号和点赞编号。博客点赞模型主要属性如表 4-6 所示。

表 4-6 博客点赞模型表

字段名称	字段类型	字段说明
blog_id	bigint	博客编号
vote_id	bigint	点赞编号

4.3.7 点赞模型

点赞模型主要包括点赞编号、创建时间和用户编号。点赞模型主要属性如表 4-7 所示。

表 4-7 点赞模型表

字段名称	字段类型	字段说明
id	bigint	点赞编号
create_time	datetime	创建时间
user_id	bigint	用户编号

4.3.8 权限模型

权限模型主要包括权限编号和权限名字。权限模型的主要属性如表 4-8 所示。

表 4-8 权限模型表

字段名称	字段类型	字段说明
id	bigint	权限编号
name	varchar	权限名字

4.3.9 用户权限模型

用户权限模型主要为了存放具体每个用户对应的权限信息。权限模型的主要属性如表 4-9 所示。

表 4-9 用户权限模型表

字段名称	字段类型	字段说明
user_id	bigint	用户编号
authority_id	bigint	权限编号

5. 系统实现

本章节主要对实现细节进行介绍，主要从三个方面进行叙述。

5.1 数据获取层

本节主要对开发中的数据传输进行描述，主要可以归纳为包括用户管理、博客管理、点赞管理，评论管理，分类管理等模块。

5.1.1 用户管理模块

1) 登录接口：

由于人们采用 Spring security 作为权限框架，所以人们的登录将会变得很简单，人们只需要调用框架中的“/login”接口即可。

请求地址：/login

请求方式：POST

接口参数：该接口直接采用 Spring security 中的 login 接口进行操作，所以人们不需要管具体实现的细节，只需要传输参数即可。具体如表 5-1 所示

表 5-1 用户登录请求参数

字段	意义	是否可选	案例
username	用户名	必选	“username”： “root”
password	密码	必选	“password”： 123456

接口返回：验证用户信息是否正确，正确后先往 cookie 中存放用户数据，然后跳转到用户刚刚请求的地址。

2) 注册接口：

方法签名如图 5-1 所示：

```
/**
 *
 * @param user
 * @return
 */
@PostMapping("/register")
public String registerUser(User user)
```

图 5-1 用户注册方法签名

用户在创建账号的时候，将会调用这个接口。

请求地址：/register

请求方式：POST

接口参数：该接口传入用户的基本信息，进行创建用户，如表 5-2 所示：

表 5-2 注册请求参数

字段	意义	是否可选	案例
username	用户名	必选	"username": "root"
password	密码	必选	"password": "123456"
email	邮箱	必选	"email": "zwz3366@gmail.com"
name	姓名	必选	"name": "zwz"

接口返回，采用 model 的形式进行返回，如表 5-3 所示：

表 5-3 注册方法返回参数

字段	意义	案例
userId	新增用户的编号	userId: 3

5.1.2 博客管理模块

博客管理模块是本课题的重点，主要有最新、最热、推荐博客接口，以及查看单个博客信息，创建博客和修改博客等接口。

1) 全文搜索博客

最新、最热、关键字搜索和推荐博客，是本课题的重中之重，它结合了 elasticsearch 来进行全文搜索，比关系型数据库有明显的优势。同时该方法除了返回博

客列表外还返回一些聚合信息，比如最活跃的用户，最热门的标签等信息，其中方法如图 5-2 所示：

```
/**
 * 博客列表
 *
 * @param boolean async
 * @param keyword
 * @param async
 * @param pageIndex
 * @param pageSize
 * @param model
 * @return
 */
@GetMapping
public String listEsBlogs(
    @RequestParam(value = "order", required = false, defaultValue = "new") String order,
    @RequestParam(value = "keyword", required = false, defaultValue = "") String keyword,
    @RequestParam(value = "async", required = false) boolean async,
    @RequestParam(value = "pageIndex", required = false, defaultValue = "0") int pageIndex,
    @RequestParam(value = "pageSize", required = false, defaultValue = "10") int pageSize,
    Model model) {
```

图 5-2 全文搜索博客方法签名

请求地址：/blogs

请求方式：GET

请求参数：该接口用来查看博客信息，用户可以输入关键字进行搜索，也可以直接点击最新和最热或者推荐链接进行操作，如表 5-4 所示

表 5-4 全文搜索博客方法请求参数

字段	意义	是否可选	案例
order	排序字段	可选	"order": "hot"
keyword	关键字	可选	"password":123456
async	是否异步请求	可选	"async": true
pageIndex	当前页	可选默认为 0	"pageIndex": 1
pageSize	一页显示多少条	可选，默认为 10	"pageSize": 10

请求结果：返回为携带博客信息的 model，渲染好页面后进行返回。Model 中主要包含以下信息，如表 5-5 所示：

表 5-5 全文搜索博客方法返回参数

字段	意义	案例
order	排序字段	"order": "hot"
keyword	关键字	"password":123456

续表 5-5

字段	意义	案例
page	分页信息	“page”: “...”
blogList	总条数	“blogList”: []

2) 查看单个博客

用户可以直接点击一个博客进行查看，后台会根据是否是自己的博客进行相应的返回，如图 5-3 所示：

```
/**
 * 获取博客展示界面
 *
 * @param id
 * @param model
 * @return
 */
@GetMapping("/u/{username}/blogs/{id}")
public String getBlogById(@PathVariable("username") String username, @PathVariable("id") Long id, Model model)
```

图 5-3 查看单个博客方法签名

请求地址：/u/{username}/blogs/{id}

请求方式：GET

请求参数：通过 RESTful url 传输用户名和博客 id，如表 5-6 所示：

表 5-6 查看单个博客方法请求参数

字段	意义	是否可选	案例
username	用户名	必选	/u/zww/blogs/1
id	博客编号	必选	/u/zww/blogs/1

接口返回：该博客的具体信息页面，具体采用 model 返回。如表 5-7 所示：

表 5-7 查看单个博客方法返回参数

字段	意义	案例
isBlogOwner	是否是博客所有者	“isBlogOwner”:true
blogModel	关键字	“blogModel”:“...”
currentVote	分页信息	“currentVote”: “...”

3) 创建/修改博客:

由于创建和修改博客的区别只有博客编号的有无，所以将二者放到一起来进行传输。方法签名如图 5-4 所示：

```
/**
 * 保存博客
 *
 * @param username
 * @param blog
 * @return
 */
@PostMapping("/{username}/blogs/edit")
@PreAuthorize("authentication.name.equals(#username)")
public ResponseEntity<Response> saveBlog(@PathVariable("username") String username, @RequestBody Blog blog)
```

图 5-4 创建/修改博客方法签名

请求地址：/u/{username}/blogs/edit

请求方式：POST

请求参数：第一个参数为用户名，采用@PathVariable 注解绑定，第二个为博客的整个对象，为一个JSON 结构。具体如表 5-8 所示：

表 5-8 创建或者修改博客方法请求参数

字段	意义	是否可选	案例
username	用户名	必选	/u/zwz/blogs/edit
blog	博客	必选	“blog”: “...”

接口返回，采用JSON 的格式进行返回，如表 5-9 所示：

表 5-9 创建或者修改博客方法返回参数

字段	意义	案例
id	博客编号	“id”:1
title	标题	“title”:“One title”
summary	摘要	“summary”: “one summary”
content	内容	“content”: “one content”
htmlContent	网页内容	“htmlContent”: “<div>...<div>”
user	用户信息	“user”: “...”
createTime	创建时间	“createTime”: “20180510”
readSize	阅读量	“readSize”: 10
commentSize	评论量	“commentSize”: 2
voteSize	点赞量	“voteSize”: 7
comments	评论列表	“comments”: []
votes	点赞列表	“votes”:[]

续表 5-9

字段	意义	案例
catalog	分类	“catalog”:[]
tags	标签	“tags”: “one,two”
success	成功与否	“success” :true
message	消息	“message” : “处理成功”

5.1.3 点赞管理模块

1) 点赞

点赞用户和博主的一个交流方式，表达了用户对博主的博客的赞赏。其方法签名如图 5-5 所示：

```
/**
 * 发表点赞
 * 指定角色权限才能操作方法
 *
 * @param blogId 博客id
 * @return 发表点赞
 */
@PostMapping
@PreAuthorize("hasAnyAuthority('ROLE_ADMIN','ROLE_USER')")
public ResponseEntity<Response> createVote(Long blogId) {
```

图 5-5 点赞方法签名

请求地址：/votes

请求方式：POST

请求参数：该接口是进行点赞，为一个固有的动作，所以只需要传一个博客编号。具体如表 5-10 所示：

表 5-10 点赞操作请求参数

字段	意义	是否可选	案例
blogId	博客编号	必选	“blogId”:2

接口返回，采用 JSON 的格式进行返回，如表 5-11 所示：

表 5-11 点赞操作返回参数

字段	意义	案例
redirectUrl	重定向 url	“redirectUrl”: “/u/zwz/blogs/1”
success	成功与否	“success”:true
message	消息	“message”: “处理成功”

2) 取消点赞

取消点赞表达了用户对该博客的不满意的心情，将会给博主敲一个警钟。其方法签名如图 5-6 所示：

```
/**
 * 删除点赞
 * 指定角色权限才能操作方法
 *
 * @return
 */
@DeleteMapping("/{id}")
@PreAuthorize("hasAnyAuthority('ROLE_ADMIN','ROLE_USER')")
public ResponseEntity<Response> delete(@PathVariable("id") Long id, Long blogId) {
```

图 5-6 取消点赞方法签名

请求地址：/votes

请求方式：DELETE

请求参数：该接口是取消点赞，需要传一个博客编号和点赞编号。具体如表 5-12 所示：

表 5-12 取消点赞方法请求参数

字段	意义	是否可选	案例
blogId	博客编号	必选	“blogId”:2

接口返回，返回类型为 JONS，如表 5-13 所示：

表 5-13 取消点赞法返回参数

字段	意义	案例
success	成功与否	“success”:true
message	消息	“message”: “处理成功”

5.1.4 评论管理模块

1) 获取评论列表

当用户查看博客单个博客的时候，会异步加载博客的评论信息，后者在用户点赞或者取消点赞的时候，也会去异步加载改博客的评论信息。其方法签名如图 5-7 所示：

```
/**
 * 获取评论列表
 *
 * @param blogId
 * @param model
 * @return
 */
@GetMapping
public String listComments(@RequestParam(value = "blogId") Long blogId, Model model)
```

图 5-7 获得博客评论列表方法签名

请求地址：/comments

请求方式：GET

请求参数：该接口为获得某一个博客下的评论列表，所以只需要传输一个博客编号就行。具体如表 5-14 所示：

表 5-14 获得评论列表方法请求参数

字段	意义	是否可选	案例
blogId	博客编号	必选	“blogId”:2

接口返回，采用 model 返回，其中 model 具体携带的信息如表 5-15 所示：

表 5-15 获得评论列表方法返回参数

字段	意义	案例
commentOwner	是否是评论所有者	“commentOwner”:true
comments	评论列表	“comments”:[...]

2) 发表评论

用户对于自己喜欢的博客可以进行发表评论的操作。其方法签名如图 5-8 所示：

```
/**
 * 发表评论
 *
 * @param blogId
 * @param commentContent
 * @return
 */
@PostMapping
@PreAuthorize("hasAnyAuthority('ROLE_ADMIN','ROLE_USER')")
public ResponseEntity<Response> createComment(Long blogId, String commentContent)
```

图 5-8 创建评论方法签名

请求地址：/comments

请求方式：POST

请求参数：该接口为创建评论，所以需要传入博客编号和评论内容就可以。具体如表 5-16 所示：

表 5-16 创建评论请求参数

字段	意义	是否可选	案例
blogId	博客编号	必选	“blogId”:2
commentContent	评论内容	必选	“commentContent”: “...”

接口返回，采用 JSON 返回，其中 JSON 具体携带的信息如表 5-17 所示：

表 5-17 创建评论法返回参数

字段	意义	案例
message	返回信息	“message”: “成功”
success	是否成功	“success”: true

3) 删除评论

用户可以自己所创建的博客评论进行删除，方法签名如图 5-9 所示：

```
/**
 * 删除评论
 *
 * @return
 */
@DeleteMapping("/{id}")
@PreAuthorize("hasAnyAuthority('ROLE_ADMIN','ROLE_USER')")
public ResponseEntity<Response> delete(@PathVariable("id") Long id, Long blogId)
```

图 5-9 删除评论方法签名

请求地址：/comments

请求方式：DELETE

请求参数：该接口为删除评论，所以需要传入博客编号和评论编号。具体如表 5-18 所示：

表 5-18 删除评论方法请求参数

字段	意义	是否可选	案例
blogId	博客编号	必选	“blogId”:2
id	评论编号	必选	“id”: 3

接口返回，采用 JSON 返回，其中 JSON 具体携带的信息如表 5-19 所示：

表 5-19 删除评论方法返回参数

字段	意义	案例
message	返回信息	“message”: “成功”
success	是否成功	“success”: true

5.1.5 分类管理模块

1) 获取分类列表

该接口为了获得博客的分类列表，方法签名如图 5-10 所示：


```
/**
 * 获取分类列表
 *
 * @param username
 * @param model
 * @return
 */
@GetMapping
public String listComments(@RequestParam(value = "username") String username, Model model)
```

图 5-10 获得分类列表方法签名

请求地址：/catalogs

请求方式：GET

请求参数：该接口为获得分类列表，所以需要传入用户名。具体如表 5-20 所示：

表 5-20 获得分类列表方法请求参数

字段	意义	是否可选	案例
username	用户名	必选	“username”：“zwz”

接口返回，采用 Model 返回，其中 Model 具体携带的信息如表 5-21 所示：

表 5-21 获得分类列表方法返回参数

字段	意义	案例
isCatalogsOwner	是否是分类的所有者	“isCatalogsOwner”：true
catalogs	分类	“catalogs”：[]

2)创建分类:

用户可以创建分类，丰富博客的管理模块。方法签名如图 5-11 所示：

```
/**
 * 发表分类
 * 指定用户才能操作方法
 *
 * @param catalogV0
 * @return
 */
@PostMapping
@PreAuthorize("authentication.name.equals(#catalogV0.username)")
public ResponseEntity<Response> create(@RequestBody CatalogV0 catalogV0)
```

图 5-11 创建分类方法签名

请求地址：/catalogs

请求方式：POST

请求参数：该接口为创建分类，需要传入用户名和分类信息。具体如表 5-22 所示：

表 5-22 创建分类方法请求参数

字段	意义	是否可选	案例
username	用户名	必选	“username”：“zwz”
catalog	分类信息	必选	“catalogs”：“...”

接口返回，采用 JSON 返回，其中 JSON 具体携带的信息如表 5-23 所示：

表 5-23 创建分类方法返回参数

字段	意义	案例
message	提示信息	“message”：“成功”

续表 5-23

success	成功与否	“success”：true
---------	------	----------------

3) 删除分类

用户可以删除自己所定义的博客类型标签，具体方法签名如图 5-12 所示：

```
/**
 * 删除分类
 *
 * @param username
 * @param id
 * @return
 */
@DeleteMapping("/{id}")
@PreAuthorize("authentication.name.equals(#username)")
public ResponseEntity<Response> delete(@RequestParam("username") String username, @PathVariable("id") Long id)
{
    ...
}
```

图 5-12 删除分类方法签名

请求地址：/catalogs

请求方式：DELETE

请求参数：该接口为删除分类，需要传入分类编号和用户名信息。具体如表 5-24 所示：

表 5-24 删除分类方法请求参数

字段	意义	是否可选	案例
username	用户名	必选	“username”：“zwz”
id	分类编号	必选	“id”:1

接口返回，采用 JSON 返回，其中 JSON 具体携带的信息如表 5-25 所示：

表 5-25 删除分类方法返回参数

字段	意义	案例
message	提示信息	“message”：“成功”
success	成功与否	“success”：true

5.2 数据构建层

这一节，将具体讲述各个模块的核心代码，将数据拼装的一个过程细化描述。

5.2.1 用户管理模块

用户管理主要包括登录、注册以及管理员下查看用户列表等操作。下面是各个方法的核心实现逻辑：

1)登录：

登录采用的是 spring security 中自己的 login 实现，阅读源码发现，它会优先去自己的用户实现类中查找该用户的信息，然后和前端传输的进行比对，吻合则放行，否则出现相应的错误提示。

2)注册：

注册主要是通过将用户输入的信息进行保存，默认用户角色编号为 2L,采用常量表示。

```
List<Authority> authorities = new ArrayList<>();
authorities.add(authorityService.getAuthorityById(ROLE_USER_AUTHORITY_ID));
user.setAuthorities(authorities);
userService.saveUser(user);
```

3)查看用户列表：

主要分页显示用户列表，供管理员调用

```

Pageable pageable = PageRequest.of(pageIndex, pageSize);
Page<User> page = userService.listUsersByNameLike(name, pageable);
// 当前所在页面数据列表
List<User> list = page.getContent();

```

5.2.2 博客管理模块

博客管理模块主要包括，用户博客的全文搜索模块、博客的创建和修改模块。

1)全文搜索模块：

主要包括输入关键字和点击最新和最热排序，以及右边的聚和排序等模块。

```

// 最热查询
if (HOT.equals(order)) {
    page = esBlogService.listHotTestEsBlogs(keyword, pageIndex, pageSize);
} else if (NEW.equals(order)) {
    // 最新查询
    page = esBlogService.listNewestEsBlogs(keyword, pageIndex, pageSize);
} else if (RECOMMEND.equals(order)) {
    esBlogService.listRecommendEsBlogs(pageIndex, pageSize);
}
isEmpty = false;

```

2)博客的创建/修改模块：

```

// 判断是修改还是新增
if (blog.getId() != null) {
    Blog originalBlog = blogService.getBlogById(blog.getId());
    originalBlog.setTitle(blog.getTitle());
    originalBlog.setContent(blog.getContent());
    originalBlog.setSummary(blog.getSummary());
    originalBlog.setCatalog(blog.getCatalog());
    originalBlog.setTags(blog.getTags());
    blogService.saveBlog(originalBlog);
} else {
    User user = (User) userDetailsService.loadUserByUsername(username);
    blog.setUser(user);
    blogService.saveBlog(blog);
}

```

5.2.3 分类管理模块

分类管理模块主要包括创建分类和删除分类。

1) 创建分类

```
String username = catalogV0.getUsername();
Catalog catalog = catalogV0.getCatalog();

User user = (User) userService.loadUserByUsername(username);

try {
    catalog.setUser(user);
    catalogService.saveCatalog(catalog);
} catch (ConstraintViolationException e) {
```

2) 删除分类

```
try {
    catalogService.removeCatalog(id);
} catch (ConstraintViolationException e) {
    return ResponseEntity.ok().body(new Response(success: false, Constr
```

5.2.4 点赞管理模块

点赞管理主要包括点赞和取消点赞。

1) 取消点赞

```
try {
    blogService.createVote(blogId);
} catch (ConstraintViolationException e) {
    return ResponseEntity.ok().body(new Response(success: false, Constr
```

2) 取消点赞

```
User user = voteService.getVoteById(id).getUser();
// 判断操作用户是否是点赞的所有者
boolean isOwner = AuthenticationUtil.isOwner(user.getUsername());
if (!isOwner) {
    return ResponseEntity.ok().body(new Response(success: false, message: "没有操作权限"));
}

try {
    blogService.removeVote(blogId, id);
```

5.2.5 评论管理模块

评论管理模块主要包括添加评论和删除评论。

1) 创建评论

```
try {
    blogService.createComment(blogId, commentContent);
```

2) 删除评论

```

boolean isOwner = false;
User user = commentService.getCommentById(id).getUser();

// 判断操作用户是否是评论的所有者
Authentication authentication = AuthenticationUtil.authentication();
if (AuthenticationUtil.isAuthenticated(authentication)) {
    User principal = AuthenticationUtil.getUser(authentication);
    if (principal != null && user.getUsername().equals(principal.getUsername())) {
        isOwner = true;
    }
}

if (!isOwner) {
    return ResponseEntity.ok().body(new Response("success: false, message: 没有操作权限"));
}

try {
    blogService.removeComment(blogId, id);
    commentService.removeComment(id);
}

```

5.2.6 权限管理模块

权限模块贯穿全文，主要通过一个 config 配置类来注入 spring security 的配置信息，具体如下所示：

```

@Override
protected void configure(HttpSecurity http) throws Exception {
    http.authorizeRequests().antMatchers(
        ...antPatterns: "/css/**", "/js/**", "/fonts/**", "/index").permitAll()
        .antMatchers(
            ...antPatterns: "/h2-console/**").permitAll() // 都可以访问
        .antMatchers(
            ...antPatterns: "/admins/**").hasRole("ADMIN") // 需要相应的角色才能访问
        .and()
        .formLogin() // 基于 Form 表单登录验证
        .loginPage("/login").failureUrl("/login-error") // 自定义登录界面
        .and().logout().logoutUrl("/logout").logoutSuccessUrl("/login") // 指定注销后重定向的页面
        .and().rememberMe().key(KEY) // 启用 remember me、
        .and().exceptionHandling()
        .accessDeniedPage("/403"); // 处理异常，拒绝访问就重定向到 403 页面

    http.csrf().ignoringAntMatchers("/h2-console/**"); // 禁用 H2 控制台的 CSRF 防护
    http.headers().frameOptions().sameOrigin(); // 允许来自同一来源的 H2 控制台的请求
}

```

同时开启了方法级别的注解驱动。

```

@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)

```

使得用户可以在方法中添加注解进行拦截。

```

*/
@DeleteMapping("/{id}")
@PreAuthorize("hasAnyAuthority('ROLE_ADMIN', 'ROLE_USER')")

```

5.3 数据展示层

最终界面如下所示：

5.5.1 登录界面

登录界面需要用户填写账号信息和密码，同时还可以点击记住我选择框。输入完毕后点击登录按钮即可。下面是登录界面的演示，此处用适配移动端形式展示，如图 5-13 所示：



图 5-13 用户登录实现图

5.5.2 注册界面

用户注册界面需要用户填写账号和、邮箱信息、真实姓名和密码输入无误后点击提交按钮，即可注册成为本博客系统中的用户。下面是注册页面的展示，此处用适配移动端形式展示，如图 5-14 所示：

Blog

注册成为博主

账号

请输入账号，至少3个字符，至多20

邮箱

请输入邮箱

姓名

请输入姓名，至少2个字符，至多20

密码

请输入密码，字母或特殊符号和数字

提交

© 2018 朱文赵

图 5-14 用户登录实现图

5.5.3 博客主页界面

本课题博客主页面可以分为用户登录和未登录两种状态，其中区别主要是在右上角的按钮显示不同。

已经登录后的主页面如图 5-15 所示：

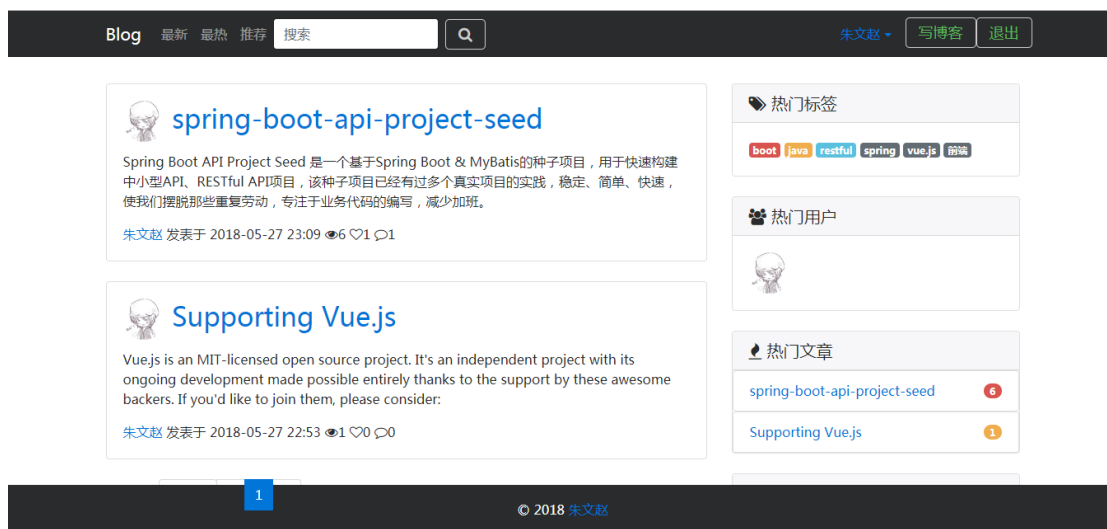


图 5-15 用户已经登录主页面实现图

未登录的主页面如图 5-16 所示：

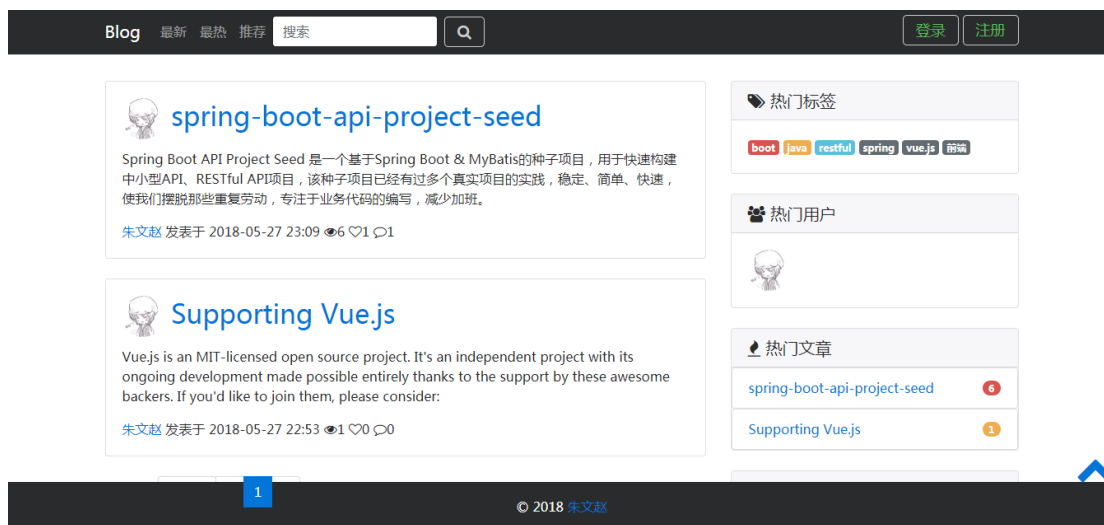


图 5-16 用户未登录主页面实现图

其中头部主要内容是本系统名称、最新排序按钮、最热排序按钮、还有推荐排序按钮和搜索框。用户可以根据自己的需要添加相应的按钮进行查看相应的博客。

左下半部分为博客列表，显示当前操作下的博客排序列表。右下半部分分为热门标签、热门用户和热门文章三个部分，主要是为了统计相应的信息，使博客看起来更加的完整。

5.4.4 创建博客界面

用户登录后可以点击写博客按钮，然后进入到写博客页面，输入相应的博客标题、摘要、内容，然后输入相应的标签和选择博客的分类，点击创建按钮即可，界面如图 5-17 所示：

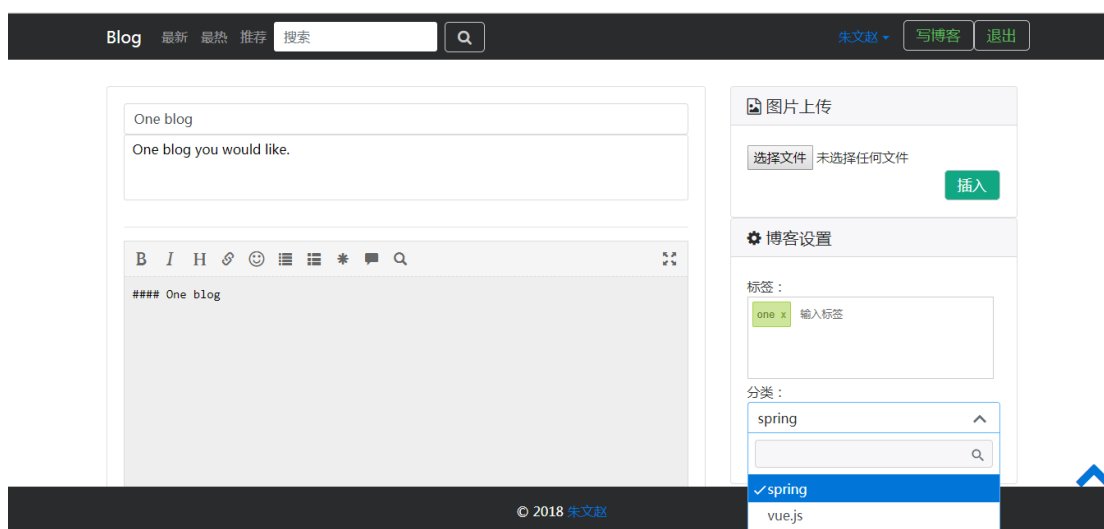


图 5-17 创建博客页面实现图

5.5.5 查看博客界面

用户点击博客列表中任意一条，即可进入到查看页面，这里又需要判断该博客的创建者是否为当前用户，若是则显示编辑按钮，若不是则不显示。下面是这两中情况的界面设计图。如图 5-18 所示：

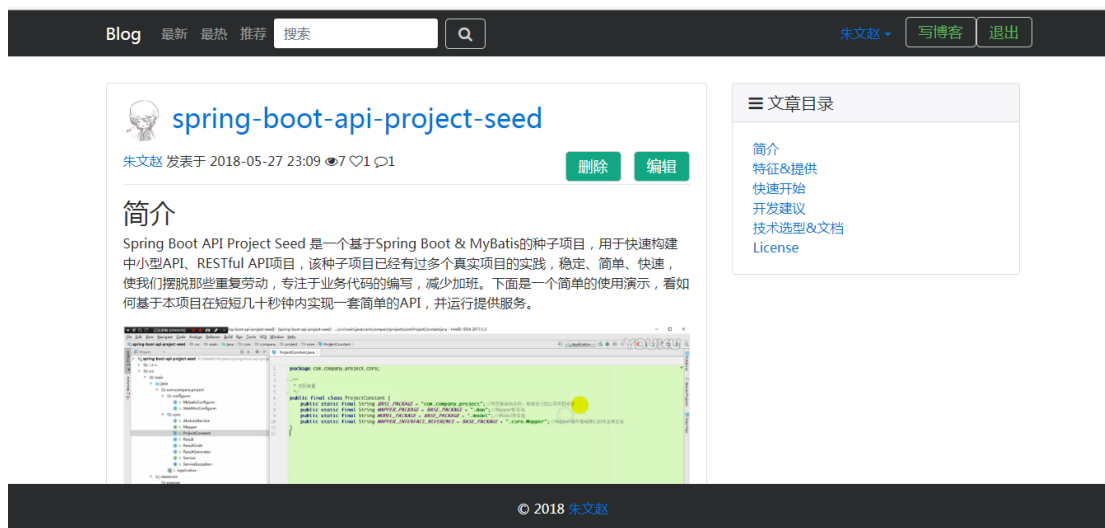


图 5-18 查看自己的博客页面实现图

如图 5-19 所示

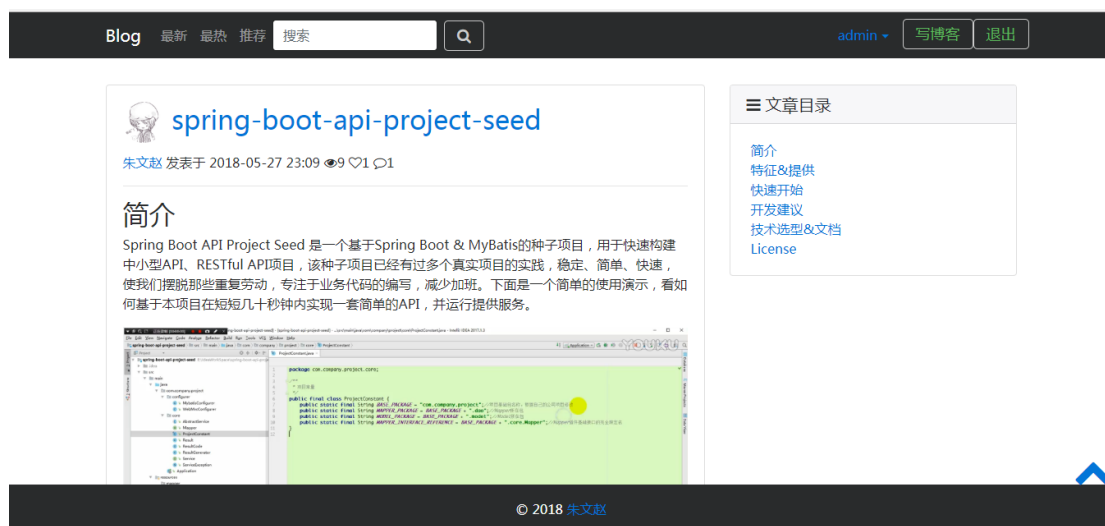


图 5-19 查看非自己的博客页面实现图

5.5.6 点赞界面

用户在点击查看一个博客列表的时候，可以 进行点赞操作。如图 5-20 所示：

分类：[spring](#)

标签：[java](#) [spring](#) [boot](#) [RESTful](#)

评论：

看帖需留言~

点赞

发表评论

图 5-20 博客点赞实现图

5.5.7 评论界面

用户同样也可以对博客进行评论操作，如图 5-21 所示：

评论：

One comment!

点赞

发表评论


 [朱文赵](#) 1楼 2018-05-29 02:35
欢迎大家互相交流

图 5-21 博客评论实现图

5.5.8 用户主页界面

用户点击个人主页按钮即可跳转到该页面，这里显示了用户的头像信息昵称邮箱地址还有分类各个分类信息，其次右半部分显示的则是自己所创建的博客列表，和主页相同可以进行最热、最新和关键词搜索。如图 5-22 所示：

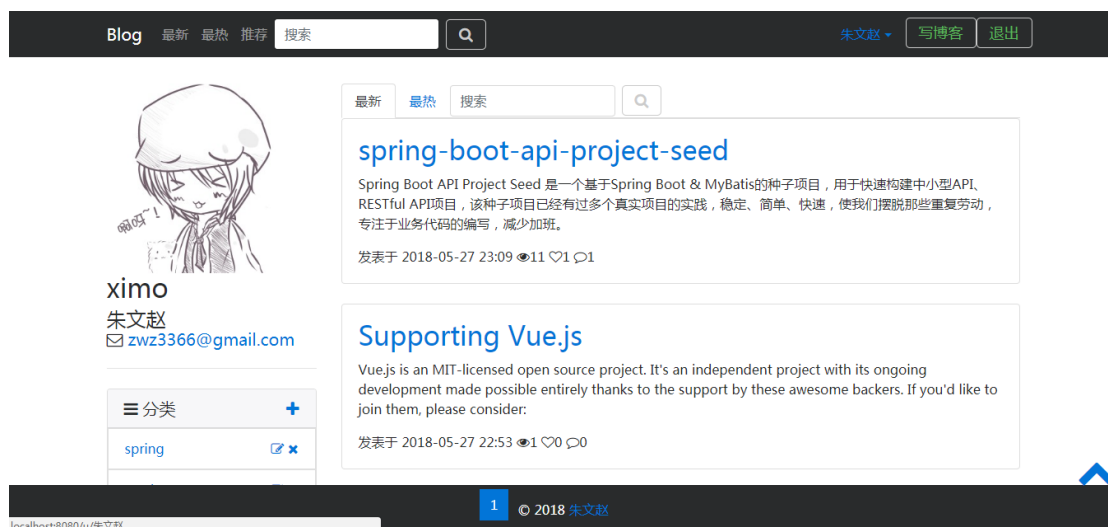


图 5-22 个人主页实现图

5.5.9 个人设置界面

用户点击个人设置按钮，即可跳转到该页面，该页面主要可以编辑自己的头像、邮箱、姓名和密码等信息。如图 5-23 所示：

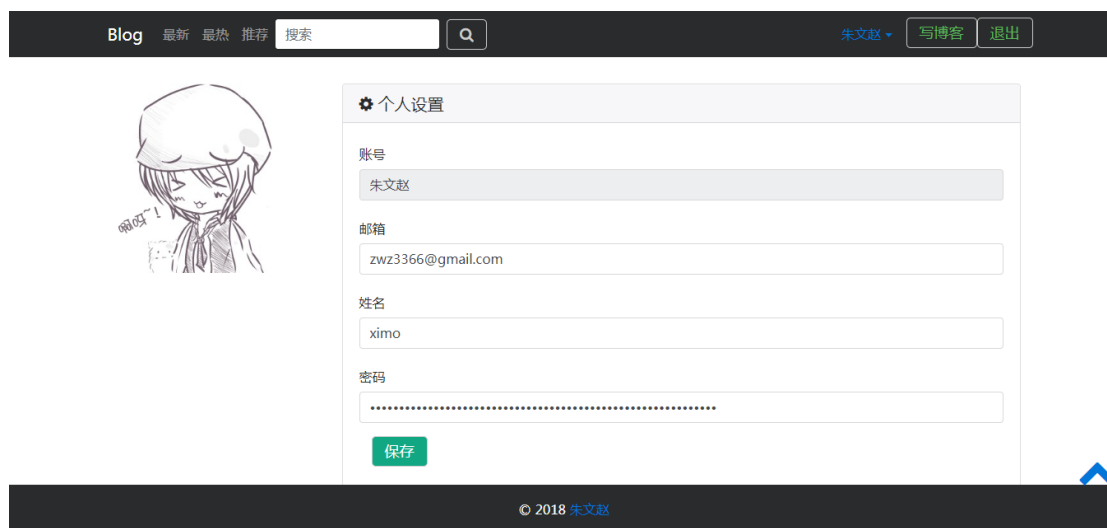


图 5-23 个人设置实现图

5.5.10 头像变更切图界面

用户点击自己的头像即可对自己的头像进行编辑，可进行左右移动。如图 5-24 所示：

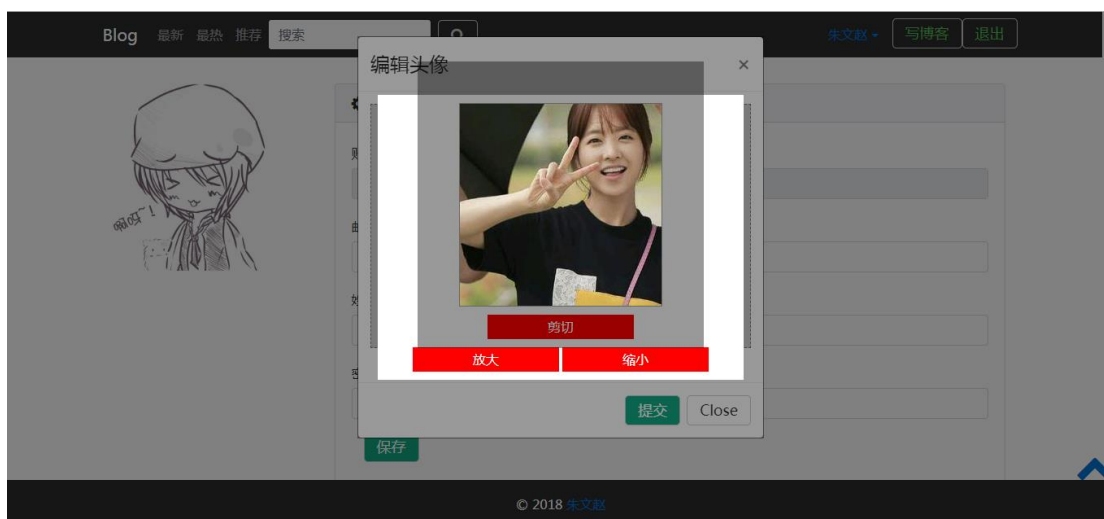


图 5-24 头像变更实现图

5.5.11 分类管理界面

用户点击个人主页，然后在下方即可看到对应的分类信息，点击添加即可添加分类，也可以选择编辑和删除分类的操作。如图 5-25 所示：

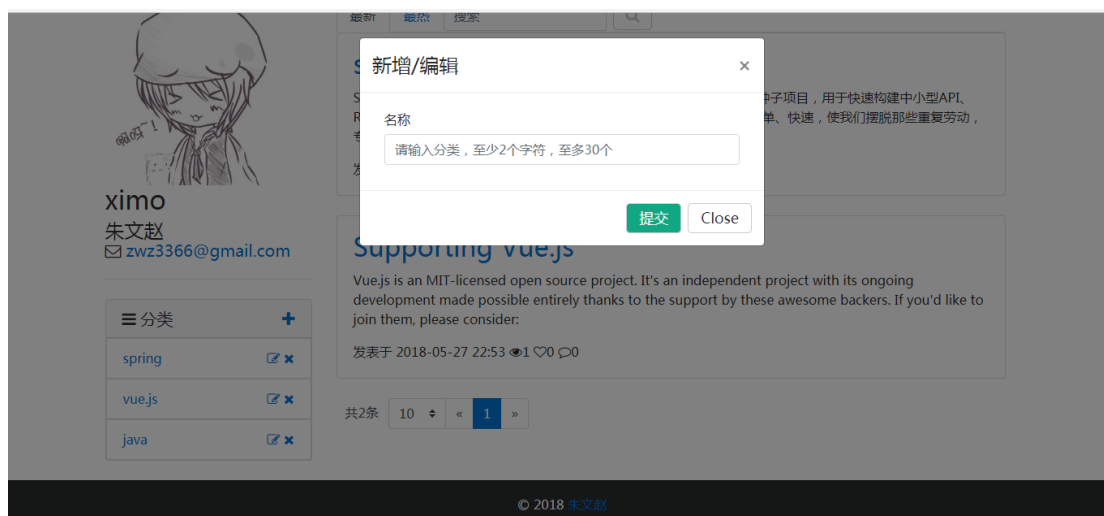


图 5-25 分类管理实现图

6. 软件测试

为了保证本课题最后给用户使用的时候，不会出现系统瘫痪和紊乱的问题，比如页面切换出现错误，跳转出现错乱，后台程序出现 500 错误，以及和其他框架或者数据库集成出现问题。所以测试是在一个非常重要的环节，能够帮助人们在上线前，有效地规避刚刚所提到的错误，保证程序能够正常地运作在服务器中，剔除一些潜在的 bug，不会因为错误而导致该系统不可用，从而发生用户体验下降的情况。

6.1 测试环境

本课题是一个 web 应用，主要测能否适配移动端的界面变化和后台逻辑是否满足要求。

- 1) 浏览器：Chrome 66.0.3359.139（正式版本）（64 位）
- 2) 编译器：IDEA Ultimate Edition 2018.1
- 3) Http 工具：Postman

6.2 测试方法

本课题将会分为三种测试方法，第一种为后台进行 JUNIT 单元测试验证服务层和数据库操作层是否能够正常调用；第二种用 Postman 进行接口测试，测试接口是否返回正确；第三种使用 Chrome 进行浏览器的界面渲染结果测试和适配移动端程度测试。以下是各个测试方法的截图：

- 1) postman 接口测试截图，如图 6-1 所示：

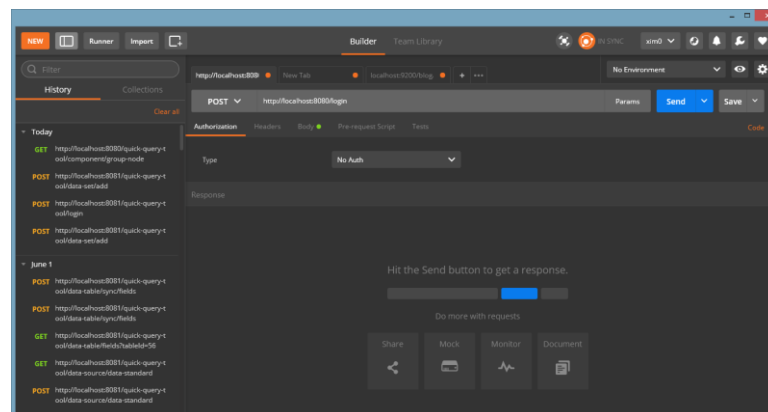


图 6-1 postman 接口测试截图

2) Junit 单元测试截图，如图 6-2 所示：

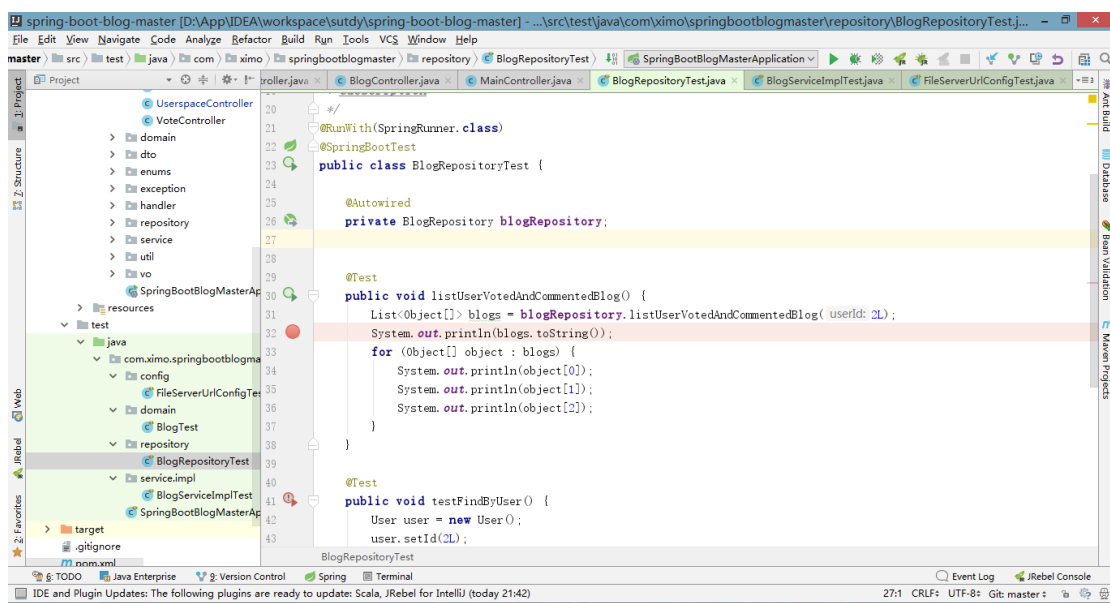


图 6-2 junit 单元测试截图

3) 移动端适配测试截图，如图 6-3 所示：



图 6-3 移动端适配测试截图

6.3 测试用例设计

由于正式上线后的系统会面临着不同的用户，且每个用户的操作习惯是不一样的，如果没有一些过滤掉一些极端的输入，可能会导致系统不可用。所以人们需要对系统进行各种各样的临界点测试，并设计好相应的测试用例。故根据本系统的需要，指定出以下几个测试用例

测试用例一：用户登录测试

表 6-1 用户登录测试用例表

用例编号		V001	
用例描述		用户登录	
用例目的		验证用户输入账号和密码，是否能正常登录	
步骤	操作	期望的结果	是否通过
1	输入账号密码	账号和密码能够正常输入，非正常的时候会有相应的错误提示	通过
2	点击登录按钮	判定用户名和密码是否匹配	通过
3	登录成功	跳转到用户未登录前所在的页面	通过

测试用例二：用户注册测试

表 6-2 用户注册测试用例表

用例编号		V002	
用例描述		用户注册	
用例目的		验证注册的时候，能否正常创建用户	
步骤	操作	期望的结果	是否通过
1	输入账号、密码、邮箱等个人信息	所以信息能够正常输入，非正常的时候会有相应的错误提示	通过
2	点击注册按钮	判定用户名是否已经存在是，然后能否正常保存用户信息	通过
3	注册成功	跳转到登录页面	通过

测试用例三：全文搜索测试

表 6-3 全文搜索测试用例表

用例编号		V003	
用例描述		全文搜索	
用例目的		验证用户在进入界面的时候，输入关键字能否进行搜索。	
步骤	操作	期望的结果	是否通过
1	输入关键字，点击搜索	满足关键字的结果在前，按照匹配度排序	通过

测试用例四：最新、最热排序

表 6-4 最新、最热排序测试用例表

用例编号		V004	
用例描述		最新和最热排序	
用例目的		测试最新和最热能否正常返回列表。	
步骤	操作	期望的结果	是否通过
1	点击最新，点击搜索或者敲击回车	结果按照现在系统中的最新创建排序	通过
2	点击最热，点击搜索或者敲击回车	结果按照现在系统中的最热创建排序，即按照评论量、阅读量和点赞量等因素进行排序	通过

测试用例五：博客推荐

表 6-5 博客推荐测试用例表

用例编号		V005	
用例描述		博客推荐	
用例目的		测试博客推荐能够正常显示列表。	
步骤	操作	期望的结果	是否通过
1	未登录，看不到推荐链接	没有推荐链接	通过
2	登录后	显示推荐链接	通过
3	点击推荐按钮	出现用户的土建博客列表	通过

测试用例六：创建博客

表 6-6 创建博客测试用例表

用例编号		V006	
用例描述		创建博客	
用例目的		测试博客能够正常创建。	
步骤	操作	期望的结果	是否通过
1	点击写博客按钮	跳转到创建博客列表	通过
2	输入博客相应的信息，点击保存	显示应当选择分类和填写标签的信息提示	通过
3	选择分类，填写标签，点击保存	出现保存成功的提示	通过

测试用例七：修改博客

表 6-7 修改博客测试用例表

用例编号		V007	
用例描述		修改博客	
用例目的		测试博客能够正常修改。	
步骤	操作	期望的结果	是否通过
1	点击一个博客列表	如果是自己创建的显示修改按钮，否则不显示	通过
2	点击修改按钮	出现修改界面	通过
3	填写内容，点击保存	出现保存成功的提示	通过

测试用例八：创建分类

表 6-8 创建分类测试用例表

用例编号		V008	
用例描述		创建分类	
用例目的		测试分类能够正常创建	
步骤	操作	期望的结果	是否通过
1	点击个人主页	跳转到个人主页	通过
2	点击左下方的创建分类加号按钮	出现弹窗，提示填写分类信息	通过
3	填写分类名字，点击保存	出现保存成功的提示	通过

测试用例九：修改分类

表 6-9 创建分类测试用例表

用例编号		V009	
用例描述		修改分类	
用例目的		测试分类能够正常修改	
步骤	操作	期望的结果	是否通过
1	点击个人主页	跳转到个人主页	通过
2	点击左下方的修改编辑按钮	出现弹窗，显示原来分类的信息，可以进行编辑	通过
3	修改信息，点击保存	出现保存成功的提示	通过

测试用例十：个人信息设置

表 6-10 个人信息设置测试用例表

用例编号		V0010	
用例描述		修改个人信息	
用例目的		测试个人信息能否正常修改	
步骤	操作	期望的结果	是否通过
1	点击个人设置	跳转到个人设置页面	通过
2	点击左侧的头像	出现弹窗，可以对图片进行更改和拖动	通过
3	点解保存	出现头像保存成功的提示	通过
4.	修改右侧的信息	除了用户名不能修改外，其他的都可以修改	通过

6.4 测试结果

经过上述的测试，确保改程序不会出现奔溃的情况，能够正常部署和上线，保证用户能够正常的使用，给用户带来一个良好的体验感。对这个系统可以看出本课题的程序设计思路良好，有着良好的代码风格，懂得测试驱动开发，有着良好的测试习惯，bug 很少。

7. 总结和展望

本章节主要是对整个过程进行一个总结，并对本文做了一个展望。

7.1 毕业设计工作总结

本课题从项目的背景、国内外研究现状等方面展开论述，说明选题的意义。然后对开发技术如 `spring boot`、`spring security` 等进行描述。而后进行系统分析，主要从可行性分析、用户需求分析等角度进行深入描述。接着开始系统的初步设计，从架构模型和界面等方面进行概述，而后便是系统的真正实现，整个系统实现完毕后对该系统进行了全面的一个测试。

经过这么多步骤，笔者学习到了很多，比如说加深了对 `java8` 的熟练度，以及深层次掌握对 `spring boot`、`spring security`、`spring data` 等框架的相互使用方法和熟练对 `elasticsearch`、`mongodb` 和 `mysql` 等数据库的操作方法。将自己在学习和工作中学到的知识，不断的重复，加深记忆。

通过本课题的开发笔者渐渐的意识到完成一个系统，从无到有，是一个非常漫长的过程，也是一个充满挑战的过程，只有自己亲自经历过才知道这其中的艰辛。不过有苦就有甜，笔者在这个过程不但学会了一整套的软件开发流程，同时也学习到了良好的设计风格是多么的重要。能够将各个模块有机地结合在一起，才是一个完整而优雅的系统。

7.2 展望

未来的几个时间内，笔者将会将本课题持续自测，然后优化其中的部分功能，增加用户体验。做到最优后，将其部署到服务器中。同时在上线的同时，添加日志模块，将 `info` 和 `error` 日志隔开，每天重复生成指定位置，保证将来能够在遇到问题的时候，准确定位。同时也需要集成 `redis` 缓存，把一些常用的信息缓存起来，避免重复去连接数据库降低 `io` 操作。将本系统打造为一个优秀的，能够对技术人员的交流和使用带来便捷博客系统。

致谢

通过这次的毕业设计历程，笔者十分感谢马虹老师，感谢她对笔者的一些关心，无论是在工作还是论文学习中，都给笔者很大帮助。马虹老师在自己十分繁忙的同时，还不忘抽时间对人们的毕业论文进行查阅，为人们把关。她的这种辛勤劳作的品质，值得笔者去学习和借鉴。

系统开发中，马虹老师给笔者提出了很多宝贵的建议，也指出了笔者的不足之处。同时她的细心程度，实在是令笔者感到惊讶和敬佩。正是她的这种求细节的品质感染了笔者，让笔者知道做好每一件小事，是多么的重要。

在感谢老师帮助的同时，也要感谢张栋在前端方面对笔者的帮助，给笔者调整一些小的前端问题。

最后也要由衷的感谢论文评审的专家老师们，谢谢你们们的辛苦付出。

参考文献

- [1] 李昕.BBS 与 Blog 比较分析[J]. 忻州师范学院学报, 2008(04): 127-129.
- [2] 马迎.我是一只小小鸟[J]. 中国保安, 2005(06): 46-48.
- [3] 马雄.基于微服务架构的系统设计与开发[D]. 南京: 南京邮电大学, 2017.
- [4] 陈涛, 叶荣华.基于 Spring Boot 和 MongoDB 的数据持久化框架研究[J]. 电脑与电信, 2016(Z1): 71-74.
- [5] 程化梅.基于 React Native 的即时通讯应用的设计与实现[D]. 武汉: 武汉邮电科学研究院, 2017.
- [6] 张峰.应用 SpringBoot 改变 web 应用开发模式[J]. 科技创新与应用, 2017(23): 193-194.
- [7] Pivotal 团队. Spring Boot Reference Guide1.5.3.RELEASE[OL]. 2017.
- [8] 王永和, 张劲松, 邓安明, 周智勋.Spring Boot 研究和应用[J]. 信息通信, 2016(10): 91-94.
- [9] 郑彬彬.基于微服务的 OJ 系统重构与优化[D]. 东华: 东华大学, 2017.
- [10] 朱荣鑫.基于微服务架构的游戏商城服务端的设计与实现[D]. 南京: 南京大学, 2017.
- [11] 汪云飞.JavaEE 开发的颠覆者: Spring Boot 实战[M]. 北京: 电子工业出版社, 2016: 1-15.
- [12] 杨家炜.基于 Spring Boot 的 web 设计与实现[J]. 轻工科技, 2016, 32(07): 86-89.
- [13] Bhimani J, Yang Z, Mi N, et al. Docker Container Scheduler for I/O Intensive Applications running on NVMe SSDs[J]. 2018, 2(99): 1-1.
- [14] Reddy K S P. Testing Spring Boot Applications[M]. Beginning Spring Boot 2. Apress, Berkeley, CA, 2017: 221-246.
- [15] Reddy K S P. Getting Started with Spring Boot[M]. Beginning Spring Boot 2. Apress, Berkeley, CA, 2017: 21-33.
- [16] Walls C. Spring Boot in action[M]. Manning Publications Co., 2016:21-34
- [17] Felipe Gutierrez. Spring Boot, Simplifying Everything[M].Apress:2014-06-15:02-06
- [18] Miller R A, Heitkamp E M. Spring boot: U.S. Patent Application 12/435,432[P]. 2010-11-11.
- [19] Balalaie A, Heydarnoori A, Jamshidi P. Microservices architecture enables devops:

Migration to a cloud-native architecture[J]. IEEE Software, 2016, 33(3): 42-52.

[20] Balalaie A, Heydarnoori A, Jamshidi P. Migrating to cloud-native architectures using microservices: an experience report[C].European Conference on Service-Oriented and Cloud Computing. Springer, Cham, 2015: 201-215.

[21] Webb P, Syer D, Long J, et al. Spring boot reference guide[J]. Part IV. Spring Boot features, 2013, 10(3): 24-25.

[22] Prieto Barreiro I, Varela F, Mandilara E. Monitoring of CERN's Data Interchange Protocol (DIP) System[J]. 2018, 3(33): 3-5.

[23] Varela F, Gonzalez Corral M, Podgorski S, et al. MARS: Easing Maintenance and Interventions for CERN Controls[J]. 2018, 3(11): 3-4.

[24] Ebner S G, Brands H, Zellweger C, et al. SwissFEL-Beam Synchronous Data Acquisition-The First Year[J]. 2018, 2(16): 3-5.