山西工商学院

Shanxi Technology And Business College

本科毕业设计

题 目： 基于 Spring Boot 的个人博客管理系统设计与实现

学　　院: 计算机信息工程学院

专　　业: 软件工程

姓　　名: 梁渲

学　　号: 2019090640114

指导教师: 尹少平（副教授）

2023 年 5 月 16 日

毕业论文（设计）学术承诺

本人郑重承诺：所呈交的毕业论文（设计）是我个人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不存在抄袭情况，论文中不包含其他人已经发表的研究成果，也不包含他人或其他教学机构取得研究成果。

作者签名：_____ 日　　期：_____

关于毕业论文（设计）使用授权的声明

本人在指导老师指导下所完成的论文（设计）及相关资料（包括图纸、试验记录、原始数据、实物照片、图片、摄像录像、设计手稿等），知识产权归属山西工商学院。本人授权山西工商学院可以将本毕业论文（设计）的全部或部分内容编入有关数据库进行检索，可以采用任何复制手段保存和汇编本毕业论文（设计）。如果发表相关成果，一定征得指导教师同意，且第一署名单位为山西工商学院。本人离校后使用毕业论文（设计）或与该论文（设计）直接相关的学术论文或成果时，第一署名单位仍然为山西工商学院。

作者签名：_____ 指导教师签名：_____

日　　期：_____ 日　　期：_____

摘　要

个人博客是一种常见的 Web 应用程序，它可以帮助个人或团队管理博客内容，并提供一种方便的方式来分享这些内容。本毕业设计旨在设计和实现一个基于 Spring Boot 的个人博客系统，以满足用户日常博客管理的需求。

本系统的主要功能包括博客文章的创建、编辑、删除、浏览、评论以及用户注册和登录等。系统后端采用了 Spring Boot 作为主要框架，前端使用 Vue 框架来构建，使用了 Spring Security 来实现用户认证和授权，前后端交互主要借用于 Thymeleaf 模板引擎，使用了 MySQL 数据库来存储数据。系统还实现了 RESTful API 接口，方便其他应用程序与本系统进行交互。

在本毕业设计中，我们将会学习到如何设计和实现一个基于 Spring Boot 的 Web 应用程序，包括前端页面的设计和实现，后端业务逻辑的实现，以及数据库的设计和管理。同时，我们还将学习到如何使用 Spring Boot 框架来构建一个可扩展、可维护、高性能的 Web 应用程序。

通过本毕业设计的实现，我们可以更好地理解和掌握 Spring Boot 框架的使用，同时也可以学习到如何设计和实现一个完整的 Web 应用程序。

关键词：Spring Boot；个人博客；前后端分离

Abstract

A personal blog is a common web application that can help individuals or teams manage blog content and provide a convenient way to share that content. The aim of this graduation project is to design and implement a personal blog system based on Spring Boot to meet the daily blog management needs of users. The main functions of this system include user registration, login, creation, editing, deletion, browsing, and commenting of blog articles. The system uses Spring Boot as the main framework, Spring Security for user authentication and authorization, Thymeleaf template engine and Vue framework to build the frontend, and MySQL database to store data. The system also implements RESTful API interfaces to facilitate interaction with other applications. In this graduation project, we will learn how to design and implement a web application based on Spring Boot, including the design and implementation of frontend pages, the implementation of backend business logic, and the design and management of databases. At the same time, we will learn how to use the Spring Boot framework to build a scalable, maintainable, and high-performance web application. Through the implementation of this graduation project, we can better understand and master the use of the Spring Boot framework, as well as learn how to design and implement a complete web application.

Key words: Spring Boot; Personal blog; Separation of front and back ends

目 录

1 绪论

1.1 课题研究背景

随着时代的进步和科技的发展，以互联网为核心的互联网和通信手段已经得到了广泛发展，各种互联网交流软件也慢慢成为生活中不可缺少的东西。其中就有QQ、论坛、贴吧、微博等较受大家喜爱，也是现在发展的比较成熟的社区形式的信息交流工具。同时，个人博客作为一种流行的网络形式，越来越多人使用它分享经验、知识，记录生活和思考。

随着科技的进步和互联网的发展，现代网络和通信技术得到。各种网络通信工具如 QQ、论坛、贴吧、微博等已成为受欢迎的社区形式的信息交流工具。

博客网站主要是为用户提供一个社交平台，可以让那些兴趣爱好相同、工作方向相关、学习内容相近的人有一个共同的社交圈子，博主们可以互相交流、相互评论，网络上提供博客平台的网站也有不少，如 CSDN、博客园等。

在过去，人们使用个人博客的方式比较简单，通常是通过一些开源的博客平台来搭建自己的博客。但是，这种方式存在一些问题，比如平台的限制、安全性等等。因此，越来越多的人开始寻求一种更加自由、安全、高效的个人博客管理系统。

1.2 开发意义

分享知识和经验：个人博客是一个分享知识和经验的好地方。通过博客，你可以写下自己的想法、经验和知识，与他人分享。这不仅可以帮助他人，也可以帮助自己更好地理解和巩固所学的知识。

建立个人品牌：个人博客可以帮助你建立个人品牌。通过博客，你可以展示自己的专业知识和技能，吸引潜在的招聘者或客户。此外，你还可以在博客中分享自己的兴趣爱好和个人经历，让读者更好地了解你的个性和价值观。

提升写作能力：个人博客可以帮助你提升写作能力。通过不断地写作，你可以锻炼自己的表达能力和逻辑思维能力，提高自己的写作水平。

学习新技能：个人博客的开发过程中，你需要学习很多新的技能，比如网站开发、SEO 优化、内容创作等。通过学习这些新技能，你可以扩展自己的技能树，提高自己的职业竞争力。

记录成长历程：个人博客可以帮助你记录自己的成长历程。通过博客，你可以记录自己的学习、工作和生活经历，回顾自己的成长历程，发现自己的不足和成长点，从而更好地提高自己。

综上所述，开发一个高效的个人博客管理系统是非常有意义的。它可以满足人们对于自由、安全、高效的博客管理需求，促进互联网信息的传播和共享，以及帮助个人建立和推广自己的品牌形象。

1.3 国内研究现状

目前，国内已经有很多关于个人博客管理系统的研究。这些研究主要集中在博客系统的设计和实现上，但是大多数研究都没有考虑到博客管理的效率和用户友好性。同时，这些研究中使用的技术栈也比较单一，缺乏综合性和实用性。因此，本研究将采用 Spring Boot 作为开发框架，结合多种技术，开发一个高效、实用、易用的个人博客管理系统。

在国内，已经有不少基于 Node.js 的博客系统开发案例，如 Hexo、Jekyll、Hugo 等。这些案例都采用了 Node.js 框架，并结合了其他技术，如 Markdown、Git 等，它们有着各式各样的主题，并实现了博客系统的快速搭建和管理。但是，这些系统大多数是静态页面，没有统一的数据库管理，而且主题配置和头文件也是没有统一格式的，面向要求不高的技术人员或愿意折腾的用户是个不错的选择，但对于一般用户来说不够友好。因此，本研究将结合前端技术，设计并实现一个更加易用、实用的个人博客管理系统[1]。

1.4 研究问题和目标

本研究的主要问题是如何设计和实现一个基于 Spring Boot 的个人博客管理系统，以提高博客管理的效率和用户友好性。具体的研究问题和目标如下：

设计并实现一个高效、实用、易用的个人博客管理系统，包括博客的创建、编辑、发布、分类、搜索等功能。

探究 Spring Boot 在博客管理系统中的应用，结合多种技术实现系统的综合性和实用性。Spring Boot 作为一款轻量级、快速开发的 Java 框架，可以帮助开发者快速搭建应用程序，提高开发效率。本研究将使用 Spring Boot 作为开发框架，结合其他技术，如 Thymeleaf、MyBatis、Redis 等，实现系统的综合性和实用性[2]。

研究博客管理系统的设计和实现，提高博客管理的效率和用户友好性。在博客管理方面，需要考虑到博客的管理流程、操作方式、界面设计等方面，以提高博客管理的效率和用户友好性。同时，需要考虑到博客的安全性、可靠性等方面，以保证博客系统的稳定运行。

在系统开发完成后，需要对系统进行全面评估，以验证其性能和稳定性。测试和评估的内容包括系统的功能测试、性能测试、安全测试等方面，以保证系统的质量和稳定性。

通过以上研究，本研究旨在开发出一个高效、实用、易用的个人博客管理系统，为用户提供一个方便快捷的博客管理平台，同时也为我提供一个学习和实践的机会。

2 相关技术介绍

2.1 开发工具

2.1.1 IDEA

IntelliJ IDEA 是一款由 JetBrains 开发的 Java 集成开发环境，是目前 Java 开发中最流行的 IDE 之一。它提供了丰富的功能和插件，支持多种编程语言和框架，例如 Java、Kotlin、Spring 等，可以大大提高开发效率。

2.1.2 VSCode

Visual Studio Code 是一款由微软开发的轻量级代码编辑器，支持多种编程语言和框架，例如 Java、JavaScript、Vue 等，可以通过插件扩展其功能。它具有代码提示、调试、版本控制等功能，是一款非常实用的开发工具。

2.1.3 Navicat

Navicat 是一款数据库管理工具，支持多种数据库，例如 MySQL、Oracle、SQL Server 等，可以方便地进行数据库的管理和维护。它提供了可视化的界面和强大的功能，例如数据导入导出、备份恢复、数据同步等。

2.1.4 X-shell 和 Xftp

X-shell 和 Xftp 是一款远程连接工具，可以通过 SSH 协议连接远程服务器进行操作。X-shell 提供了强大的终端模拟器和 SSH 客户端，可以方便地进行命令行操作。Xftp 提供了可视化的文件传输界面，可以方便地进行文件传输和管理。

2.2 数据库

2.2.1 MySQL

MySQL 是一款开源的关系型数据库管理系统，是目前最流行的数据库之一。它具有高性能、可靠性和可扩展性等优点，支持多种数据类型和存储引擎，例如 InnoDB、MyISAM 等。

2.2.2 Redis

Redis 是一款开源的内存数据库，支持多种数据结构和操作，例如字符串、哈希表、列表等。它具有高性能、可靠性和可扩展性等优点，可以用于缓存、消息队列、分布式锁等场景。

2.3 服务器

2.3.1 Nginx

Nginx 是一款高性能的 Web 服务器和反向代理服务器，可以用于负载均衡、静态文件服务、反向代理等场景。它具有高性能、可靠性和可扩展性等优点，是目前最流行的 Web 服务器之一。

2.3.2 Docker

Docker 是一款开源的容器化平台，可以方便地进行应用程序的打包、发布和管理。它具有高效、轻量、可移植等优点，可以提高应用程序的可靠性和可移植性。

2.4 框架

2.4.1 Spring Boot

Spring Boot 是一款轻量级的 Java 开发框架，可以快速地构建基于 Spring 的应用程序。它提供了自动化配置、快速开发、微服务等特性，可以提高开发效率[3]。

2.4.2 MyBatisPlus

MyBatisPlus 是一款基于 MyBatis 的增强工具，可以方便地进行 SQL 操作和代码生成。它提供了多种功能和插件，例如分页插件、逻辑删除插件、代码生成插件等，可以大大提高开发效率。

2.4.3 Vue

Vue 是一款流行的前端框架，可以用于构建单页面应用程序和组件化开发。它具有简单、灵活、高效等特点，可以提高前端开发效率。

3 需求分析

3.1 功能需求

用户管理：包括查看用户个人信息、用户强制下线、禁用用户等功能。

文章管理：包括发布、编辑、删除文章，文章分类和标签管理等功能。

消息管理：包括对文章的评论和回复，评论审核和删除等功能。

权限管理：包括对菜单管理、接口管理和角色管理等功能。

系统管理：包括网站主页信息管理、页面管理、友链管理等功能。

相册管理：包括上传图片和创建相册等功能。

说说管理：包括对说说的发表、修改和删除一系列操作等功能。

日志管理：包括查看和删除管理员的一系列后台操作记录。

个人中心：包括修改用户信息、管理员信息和管理员登录密码。

查看文章：包括管理员已发布的所有文章内的图片、文字等信息。

查看评论：包括管理员审核过后的所有评论。

发表和回复评论：包括已登录用户发表一级评论或对其回复二级评论。

查看和发表留言：包括用户、游客等所有人在留言板上留下的留言。

查看相册：包括所有相册和所有管理员已上传的图片。

查看说说：包括管理员已发布的所有说说。

3.2 性能需求

响应时间：系统需要能够在短时间内响应用户的请求，以保证用户的体验。

并发处理能力：系统需要能够同时处理多个用户的请求。

数据安全：需要保证数据的安全，包括用户信息、文章内容、评论信息等。

3.3 用户需求

管理员：能够对系统进行管理，包括文章管理、评论管理、用户管理等。

普通用户：能够查看文章、发表评论、回复评论、修改个人信息等。

游客：能够查看文章、查看评论、查看留言等。

3.4 界面需求

界面风格：简洁明了，易于操作。

布局设计：合理布局，易于用户浏览。

颜色搭配：配色搭配清晰明了，易于用户辨认。

3.5 数据需求

数据存储：需要存储用户信息、文章内容、评论信息等数据。

数据库设计：需要设计合理的数据库结构，以方便数据存储和查询

3.6 可维护性需求

系统可维护性：系统需要设计为易于维护和修改，以满足后续的需求变化。

代码可读性：系统的代码需要易于阅读和理解，以便后续的维护和修改。

3.7 兼容性需求

浏览器兼容性：系统需要在主流的浏览器中正常运行。

设备兼容性：系统需要在不同设备上正常运行，包括桌面端和移动端[4]。

4 系统设计

4.1 系统功能设计

4.1.1 功能总体设计

此系统的使用用户分为系统管理员、普通用户和游客。

系统管理员功能模块如图 4-1 所示。

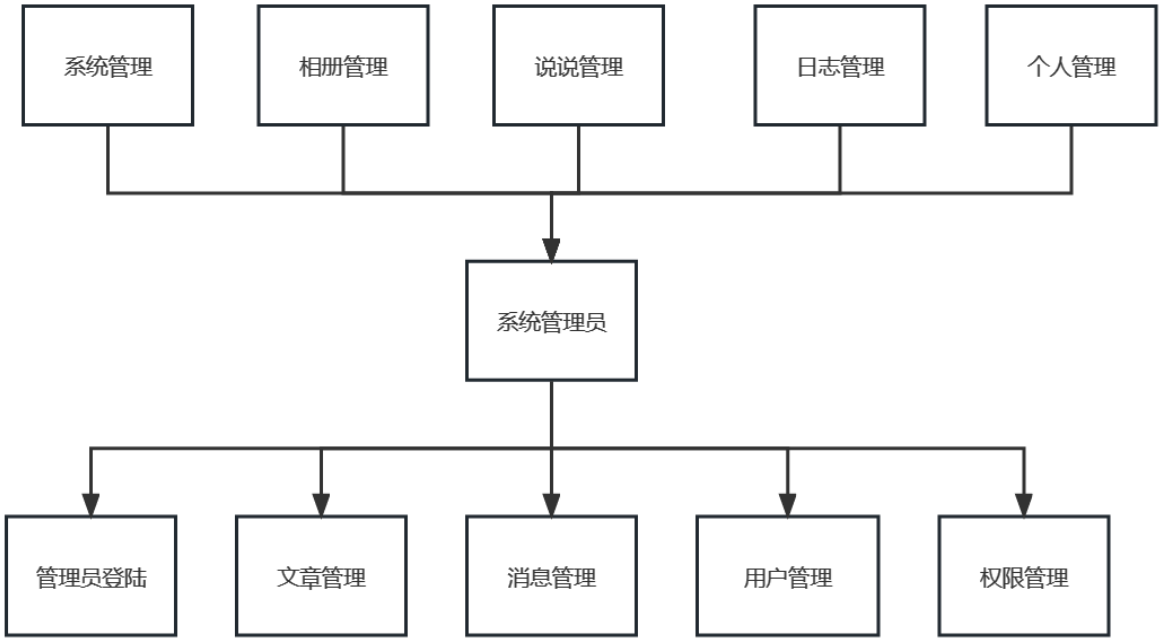

图 4-1 系统管理员功能模块图

普通用户功能模块如图 4-2 所示。

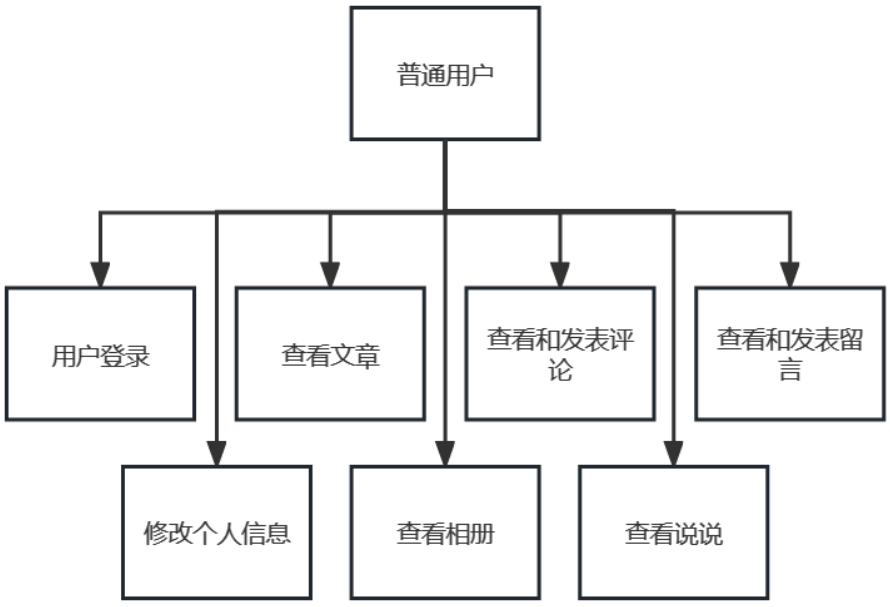

图 4-2 普通用户功能模块图

游客功能模块如图 4-3 所示。

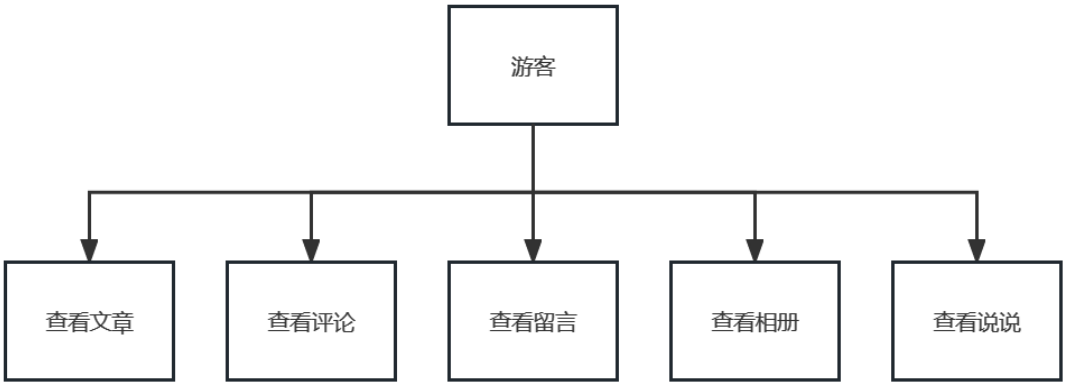

图 4-3 游客功能模块图

4.1.2 前台功能设计

前台功能是面向普通用户的，包括用户登录、用户注册、查看文章、评论、留言等。其中，用户登录和注册功能是基本的身份验证功能，查看文章、评论、留言等功能则是博客系统的核心功能。如图 4-4 前台功能模块图[5]。

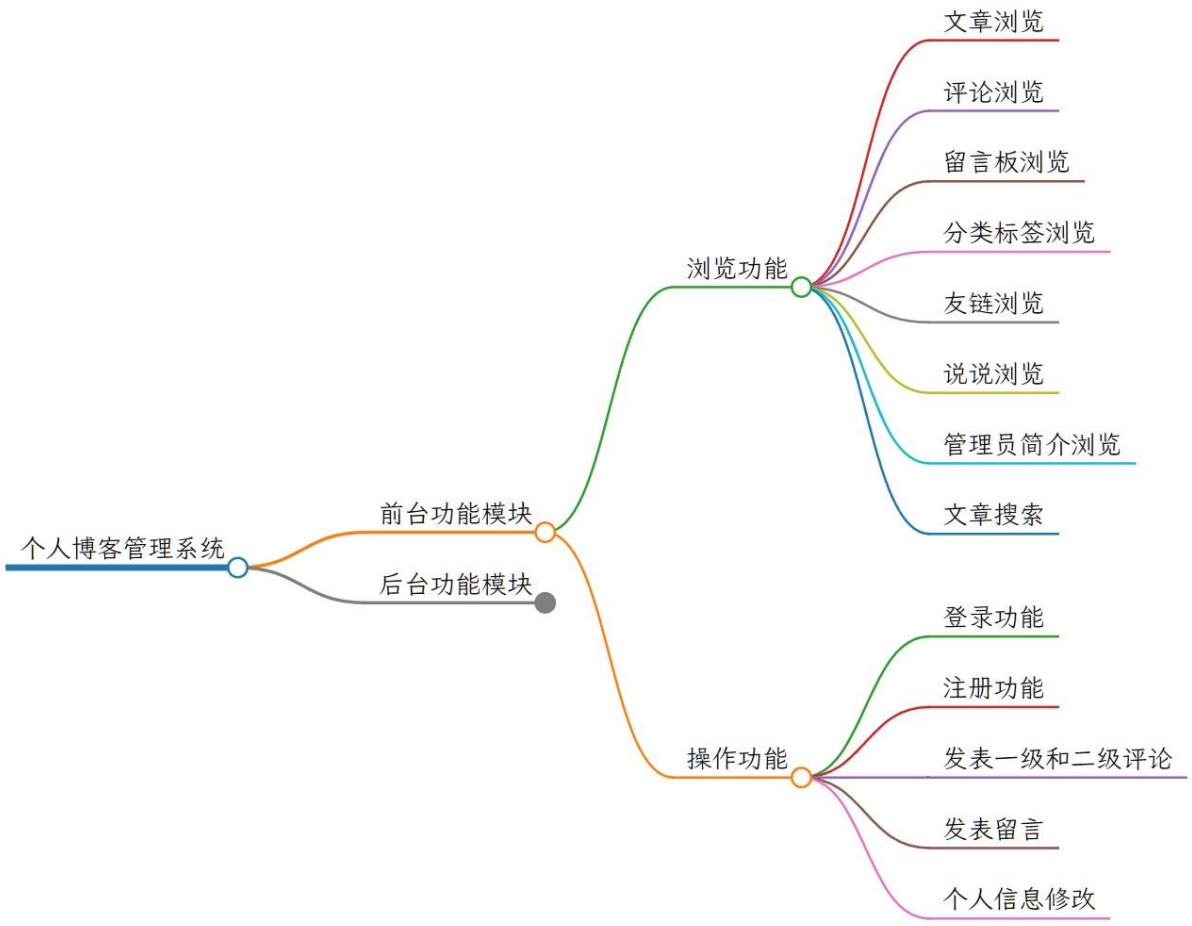

图 4-4 前台功能模块图

4.1.3 后台功能设计

后台功能是面向管理员的，包括管理员登录、文章管理、评论管理、系统管理、管理员信息修改等。其中，管理员登录功能是基本的身份验证功能，文章管理、评论管理、系统管理等功能则是博客系统的管理功能。如图 4-5 后台功能模块图。

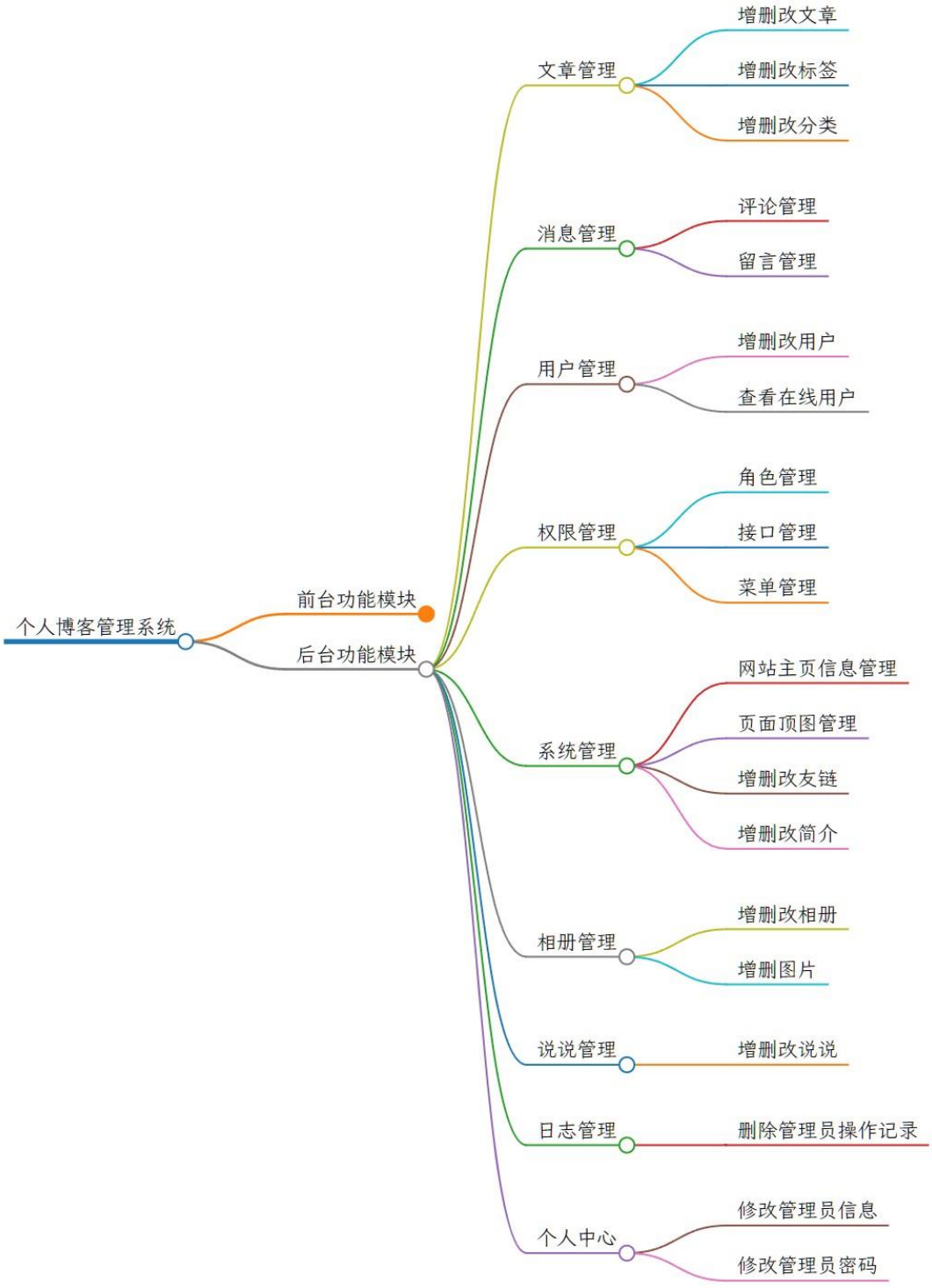

图 4-5 后台功能模块图

4.2 数据库设计

4.2.1 需求分析

文章表：保存博客文章的基本信息，包括标题、内容、发布时间、作者等。

标签表：记录标签名等信息。

文章标签表：记录文章和标签之间的关系。

分类表：保存文章的分类信息，包括分类名称等。

页面表：保存前端页面的标题和封面图。

照片表：保存管理员上传的图片。

相册表：保存管理员创建的相册。

说说表：保存管理员发布的说说信息，包括说说内容等。

评论表：保存用户对文章的评论信息，包括评论内容、评论时间、评论者等。

留言表：记录用户和游客在留言板上的留言，包括头像、留言内容等。

友链表：保存其他网站的链接信息，包括链接名称、链接地址等。

角色表：记录角色名和角色描述。

菜单表：记录前端和后端的各个页面路径、组件和图标等。

资源表：记录网站的所有请求路径和方式。

角色菜单表：记录各个角色和菜单之间的关系。

角色资源表：记录各个角色和权限之间的关系。

用户角色表：记录已知用户和角色之间的关系。

用户注册登录表：保存用户注册和登录信息，包括邮箱、密码、登陆方式、IP 地址、注册时间和登录时间等。

用户个人信息表：保存登录的用户信息，包括邮箱、用户名、头像、简介和个人站点链接等。

每日浏览表：记录每天的网页访问次数。

日志表：对管理员在后台的一系列操作进行记录。

网站配置表：保存网站的配置信息，包括网站名称、图标、管理员头像、简介、创站时间等[6]。

4.2.2 概念设计

根据需求分析，设计出该系统简单的 E-R 图。如下 4-6 E-R 图所示：

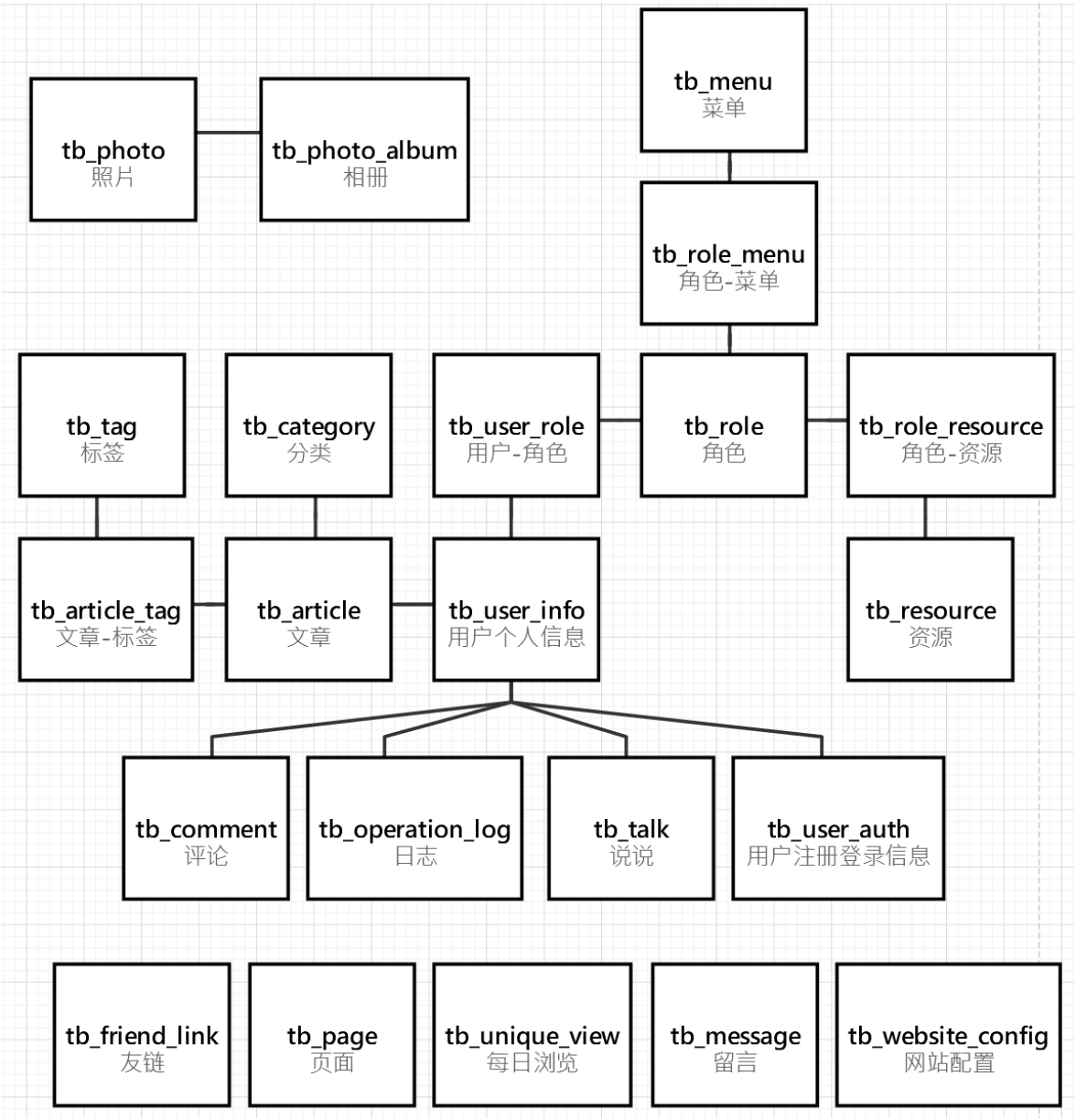

图 4-6 E-R 图

其中使用了“-”符号的实体表示两边实体的关系。

在这个示例中，我们可以看到有二十二个实体：照片、相册、菜单、角色-菜单、角色、角色-资源、资源、用户-角色、用户个人信息、文章、分类、文章-标签、标签、评论、日志、说说、用户注册登录信息、友链、页面、每日浏览、留言、网站配置。它们之间的关系如下：

每篇文章可以有多个标签，即一篇文章对应多个标签，每个标签可以被多篇文章使用，即一个标签对应多篇文章，因此文章和标签之间的关系是多对多。

每个用户可以拥有多个用户角色，即一个用户对应多个用户角色，每个用户角色可以被多个用户拥有，即一个用户角色对应多个用户，因此用户个人信息和用户角色之间的关系是多对多。

每个角色可以拥有多个角色功能，即一个角色对应多个角色功能，因此角色和角色功能之间的关系是一对多。

不是所有角色功能可以被多个角色拥有，即一个角色功能对应多个角色、也可能对应一个角色，因此角色功能和角色之间的关系既是多对多也是一对一。

每个角色可以拥有多个资源，即一个角色对应多个资源，每个资源可以拥有多个角色，即一个资源对应多个角色，因此角色和资源之间的关系是多对多。

每个角色可以拥有多个菜单，即一个角色对应多个菜单，每个菜单可以拥有多个角色，即一个菜单对应多个角色，因此角色和资源之间的关系是多对多。

每个分类可以拥有多篇文章，即一个分类对应多篇文章，因此分类和文章之间的关系是一对多。

每篇文章只能被归属到一个分类，即多篇文章对应一个分类，因此文章和分类之间的关系是多对一。

每个相册可以拥有多个照片，即一个相册对应多个照片，因此相册和照片之间的关系是一对多。

每个照片只能放在一个相册，即多个照片对应一个相册，因此照片和相册之间的关系是多对一[7]。

4.2.3 逻辑设计

在概念设计的基础上，进行逻辑设计，将简单的 E-R 图赋予属性，并转换为关系模式图。接下来会将 E-R 图转换的关系模式图分成多部分讲解。

（1）用户信息、角色、菜单和资源关系模式图

4 个实体转换为关系如下：

用户个人信息表（id，email，nickname，avatar，intro，web_site，is_disable，create_time，update_time）

角色表（id，role_name，role_label，is_disable，create_time，update_time）

菜单表（id，name，path，component，icon，create_time，update_time，orfer_num，parent_id，is_hidden）

资源表（id，resource_name，url，request_method，paremt_id，is_anonymous，create_time，update_time）

3 个联系转换为关系：

用户角色表（id，user_id，role_id）

用户菜单表（id，role_id，menu_id）

角色资源表（id，role_id，resource_id）

根据以上示例画出它们的关系模式图，如图 4-7。

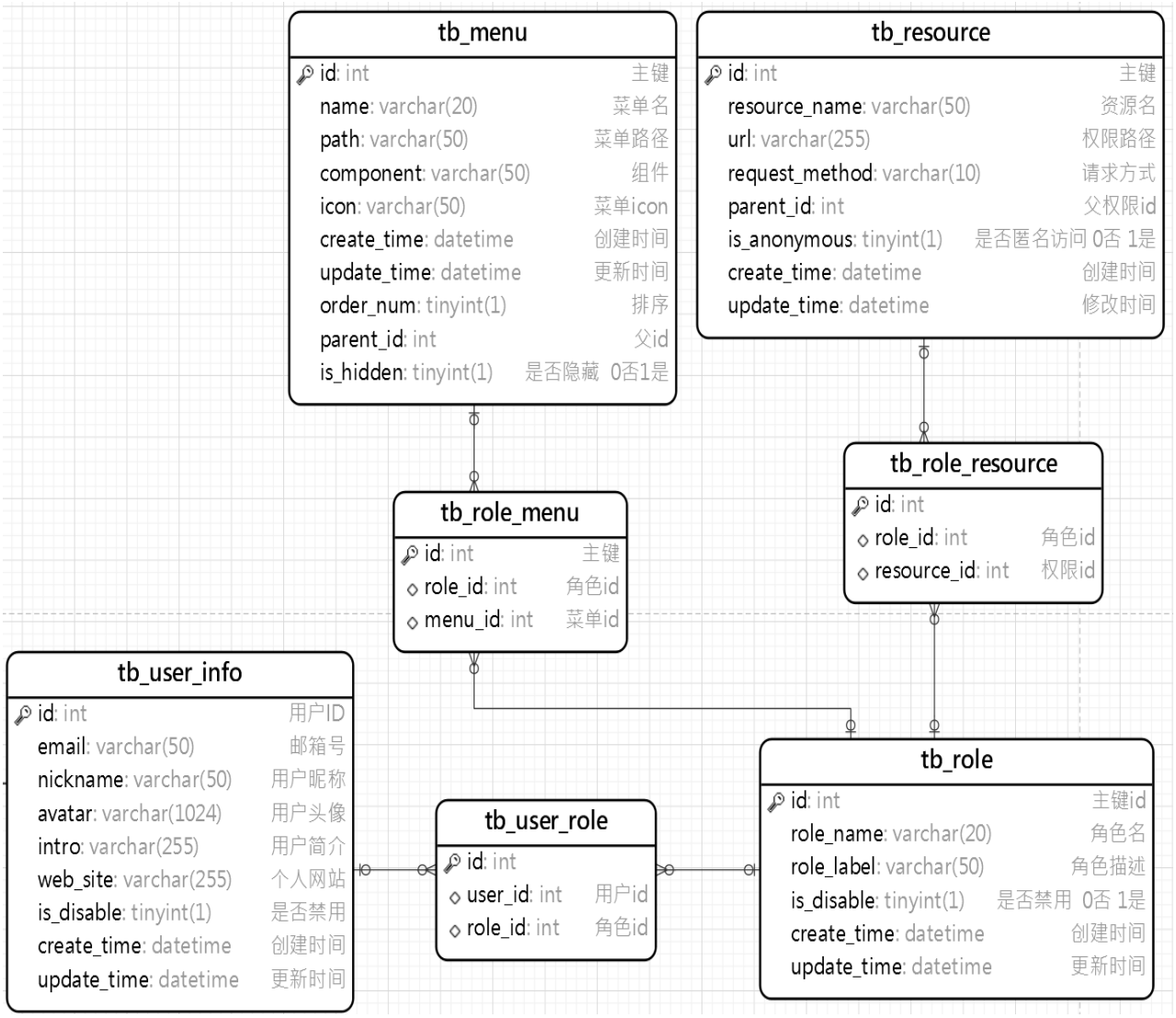

图 4-7 用户个人信息、角色、菜单和资源关系模式图

（2）用户个人信息、文章、标签和分类关系模式图

4 个实体转换为关系如下：

用户个人信息表（id，email，nickname，avatar，intro，web_site，is_disable，create_time，update_time）

文章表（id，user_id，category_id，article_cover，article_title，article_content，type，original_url，is_top，is_delete，status，create_time，update_time）

标签表（id，tag_name，create_time，update_time）

分类表（id，category_name，create_time，update_time）

1 个联系转换为关系：

文章标签表（id，article_id，tag_id）

根据以上示例画出它们的关系模式图，如图 4-8。

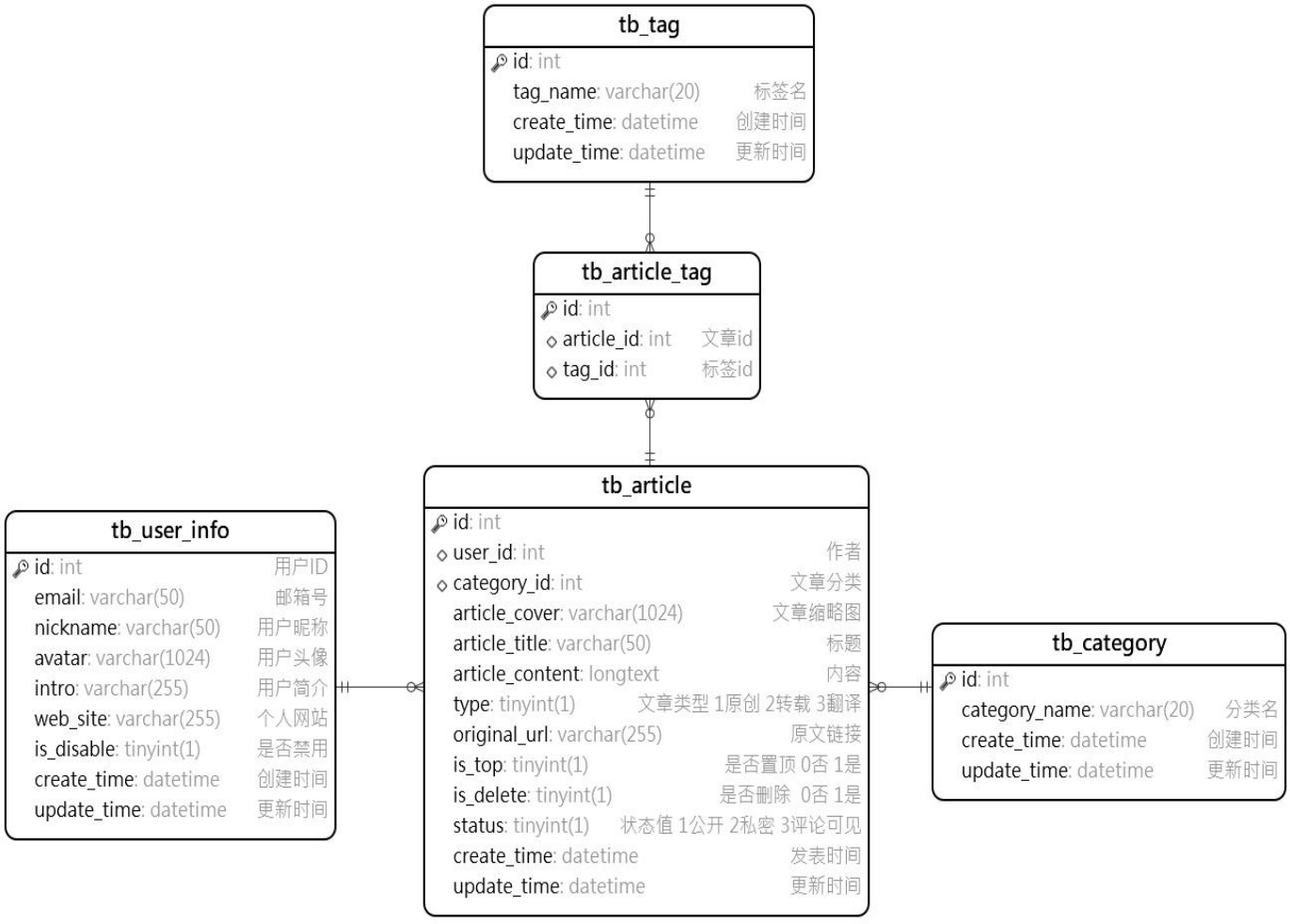

图 4-7 用户个人信息、文章、分类和标签关系模式图

（3）用户个人信息、用户登录信息、评论、说说和日志关系模式图

5 个实体转换为关系如下：

用户个人信息表（id，email，nickname，avatar，intro，web_site，is_disable，create_time，update_time）

用户登陆注册信息表（id，user_info_id，username，password，login_type，ip_address，ip_source，create_time，update_time，last_login_time）

说说表（id，user_id，content，images，is_top，status，create_time，update_time）

评论表（id，user_id，topic_id，comment_content，reply_user_id，parent_id，type，is_delete，is_review，create_time，update_time）

日志表（id，user_id，opt_module，opt_type，opt_url，opt_method，opt_desc，request_param，request_method，request_data，nickname，ip_address，ip_source，create_time，update_time）

根据以上示例画出它们的关系模式图，如图 4-8。

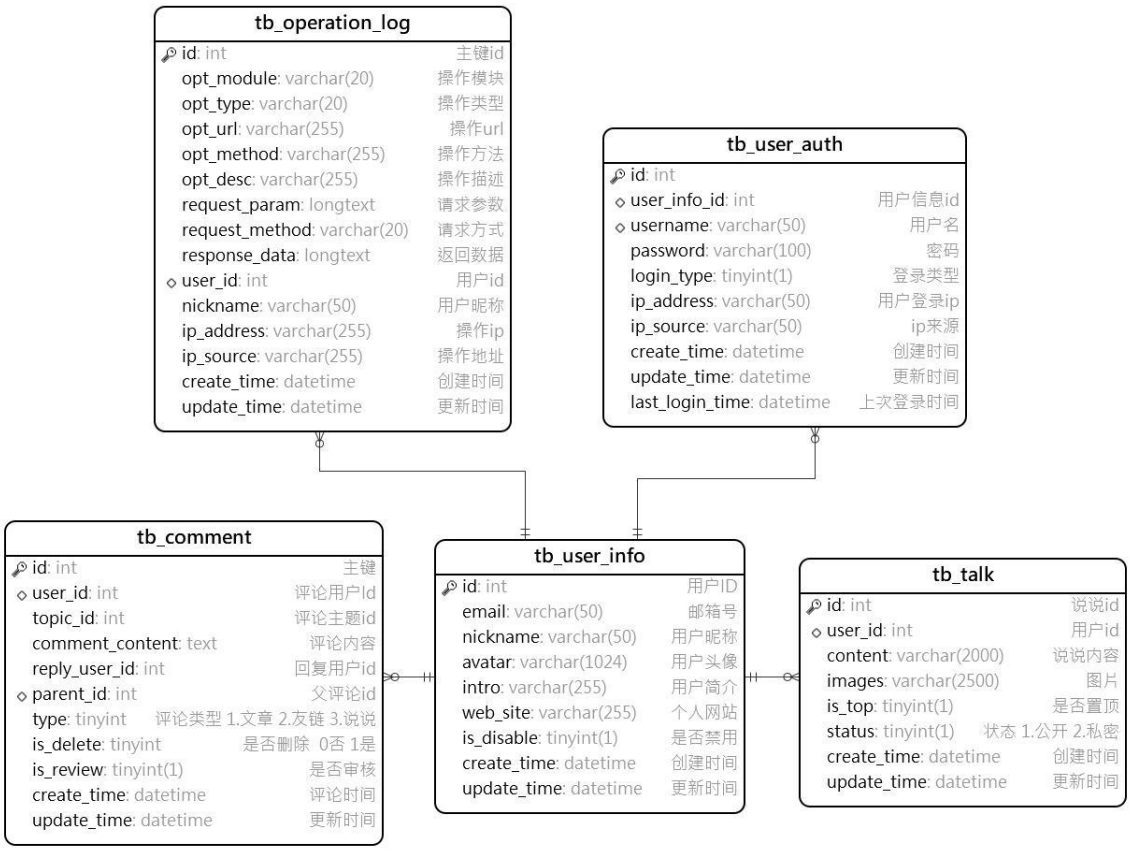

图 4-8 用户个人信息、用户登录信息、评论、说说和日志关系模式图

（4）照片、相册关系模式图

2 个实体转换为关系如下：

照片（id，album_id，photo_name，photo_desc，photo_src，is_delete，create_time，update_time）

相册（id，album_name，album_desc，album_cover，is_delete，status，create_time，update_time）

根据以上示例画出它们的关系模式图，如图 4-9。

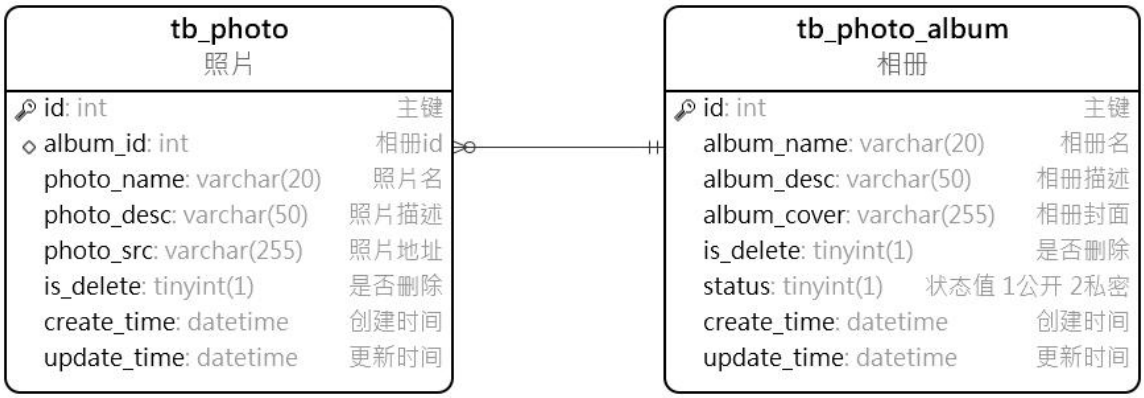

图 4-9 照片和相册关系模式图

4.2.4 数据表设计

根据上一节中的 E-R 图、关系模式图的设计，继续在物理层面上创建数据表。

（1）文章表（tb_article）

表 4-1 文章表

序号	数据名称	字段名	数据类型
1	文章 id	id	int
2	用户 id	user_id	int
3	分类 id	category_id	int
4	文章封面	article_cover	varchar
5	文章标题	article_title	varchar
6	文章内容	article_content	longtext

7	文章类型 1 原创 2 转载 3 翻译	type	tinyint
8	原文链接	original_url	varchar
9	是否置顶 0 否 1 是	is_top	tinyint
10	是否删除 0 否 1 是	is_delete	tinyint
11	状态值 1 公开 2 私密	status	tinyint
12	发表时间	create_time	datetime
13	更新时间	update_time	datetime

（2）标签表（tb_tag）

表 4-2 标签表

序号	数据名称	字段名	数据类型
1	标签 id	id	int
2	标签名	tag_name	varchar
3	创建时间	create_time	datetime
4	更新时间	update_time	datetime

（3）文章标签表（tb_article_tag）

表 4-3 文章标签表

序号	数据名称	字段名	数据类型
1	id	id	int
2	文章 id	article_id	int
3	标签 id	tag_id	int

（4）分类表（tb_category）

表 4-4 分类表

序号	数据名称	字段名	数据类型
1	分类 id	id	int

2	分类名	category_name	varchar
3	创建时间	create_time	datetime
4	更新时间	update_time	datetime

（5）评论表（tb_comment）

表 4-5 评论表

序号	数据名称	字段名	数据类型
1	评论 id	id	int
2	评论用户 id	user_id	int
3	评论主题 id	topic_id	int
4	评论内容	comment_content	text
5	回复用户 id	reply_user_id	int
6	父评论 id	parent_id	int
7	评论类型 1.文章 2.友链 3.说说	type	tinyint
8	是否删除 0 否 1 是	is_delete	tinyint
9	是否审核	is_review	tinyint
10	评论时间	create_time	datetime
11	更新时间	update_time	datetime

（6）留言表（tb_message）

表 4-6 留言表

序号	数据名称	字段名	数据类型
1	留言 id	id	int
2	昵称	nickname	varchar
3	头像	avatar	varchar
4	留言内容	message_content	varchar

5	用户 ip	ip_address	varchar
6	用户地址	ip_source	varchar
7	弹幕速度	time	tinyint
9	是否审核	is_review	tinyint
10	发布时间	create_time	datetime
11	修改时间	update_time	datetime

（7）说说表（tb_talk）

表 4-7 说说表

序号	数据名称	字段名	数据类型
1	说说 id	id	int
2	用户 id	user_id	int
3	说说内容	content	varchar
4	图片	images	varchar
5	是否置顶	is_top	tinyint
6	状态 1.公开 2.私密	status	tinyint
7	发布时间	create_time	datetime
8	修改时间	update_time	datetime

（8）友链表（tb_ftiend_link）

表 4-8 友链表

序号	数据名称	字段名	数据类型
1	友链 id	id	int
2	链接名	link_name	varchar
3	链接头像	link_avatar	varchar
4	链接地址	link_address	varchar

5	链接介绍	link_intro	varchar
6	创建时间	create_time	datetime
7	更新时间	update_time	datetime

（9）照片表（tb_photo）

表 4-9 照片表

序号	数据名称	字段名	数据类型
1	照片 id	id	int
2	相册 id	album_id	int
3	照片名	photo_name	varchar
4	照片描述	photo_desc	varchar
5	照片地址	photo_src	varchar
6	是否删除	is_delete	tinyint
7	创建时间	create_time	datetime
8	更新时间	update_time	datetime

（10）相册表（tb_photo_album）

表 4-10 相册表

序号	数据名称	字段名	数据类型
1	照片 id	id	int
2	相册名	album_name	varchar
3	相册描述	album_desc	varchar
4	相册封面	album_cover	varchar
5	是否删除	is_delete	tinyint
6	状态值 1 公开 2 私密	status	tinyint
7	创建时间	create_time	datetime

8	更新时间	update_time	datetime

（11）用户个人信息表（tb_user_info）

表 4-11 用户个人信息表

序号	数据名称	字段名	数据类型
1	用户 id	id	int
2	邮箱号	email	varchar
3	用户昵称	nickname	varchar
4	用户头像	avatar	varchar
5	用户简介	intro	varchar
6	个人网站	web_site	varchar
7	是否禁用	is_disable	tinyint
8	创建时间	create_time	datetime
9	更新时间	update_time	datetime

（12）用户注册登录信息表（tb_user_auth）

表 4-12 用户注册登录信息表

序号	数据名称	字段名	数据类型
1	用户注册登录信息 id	id	int
2	用户个人信息 id	user_info_id	int
3	用户名	username	varchar
4	密码	password	varchar
5	登录类型	login_type	tinyint
6	用户登录 ip	ip_address	varchar
7	ip 来源	ip_source	varchar
8	创建时间	create_time	datetime

9	更新时间	update_time	datetime
10	上次登录时间	last_login_time	datetime

（13）角色表（tb_role）

表 4-13 角色表

序号	数据名称	字段名	数据类型
1	角色 id	id	int
2	角色名	role_name	varchar
3	角色描述	role_label	varchar
4	是否禁用　0 否 1 是	is_disable	tinyint
5	创建时间	create_time	datetime
6	更新时间	update_time	datetime

（14）用户角色表（tb_user_role）

表 4-14 用户角色表

序号	数据名称	字段名	数据类型
1	id	id	int
2	用户 id	user_id	int
3	角色 id	role_id	int

（15）资源表（tb_resource）

表 4-15 资源表

序号	数据名称	字段名	数据类型
1	资源 id	id	int
2	资源名	resource_name	varchar
3	权限路径	url	varchar
4	请求方式	request_method	varchar

5	父权限 id	parent_id	int
6	是否匿名访问 0 否 1 是	is_anonymous	tinyint
7	创建时间	create_time	datetime
8	修改时间	update_time	datetime

（16）角色资源表（tb_role_resource）

表 4-16 角色资源表

序号	数据名称	字段名	数据类型
1	id	id	int
2	角色 id	role_id	int
3	权限 id	resource_id	int

（17）菜单表（tb_menu）

表 4-17 菜单表

序号	数据名称	字段名	数据类型
1	菜单 id	id	int
2	菜单名	name	varchar
3	菜单路径	path	varchar
4	组件	component	varchar
5	菜单 icon	create_time	datetime
6	创建时间	update_time	datetime
7	更新时间	order_num	tinyint
8	排序	parent_id	int
9	是否隐藏　0 否 1 是	is_hidden	tinyint

（18）角色菜单表（tb_role_menu）

表 4-18 角色菜单表

序号	数据名称	字段名	数据类型
1	id	id	int
2	角色 id	role_id	int
3	菜单 id	menu_id	int

（19）页面表（tb_page）

表 4-19 页面表

序号	数据名称	字段名	数据类型
1	页面 id	id	int
2	页面名	page_name	varchar
3	页面标签	page_label	varchar
4	页面封面	page_cover	varchar
5	创建时间	create_time	datetime
6	更新时间	update_time	datetime

（20）日志表（tb_operation_log）

表 4-20 日志表

序号	数据名称	字段名	数据类型
1	日志 id	id	int
2	操作模块	opt_module	varchar
3	操作类型	opt_type	varchar
4	操作 url	opt_url	varchar
5	操作方法	opt_method	varchar
6	操作描述	opt_desc	varchar
7	请求参数	request_param	longtext
8	请求方式	request_method	varchar

9	返回数据	response_data	longtext
10	用户 id	user_id	int
11	用户昵称	nickname	varchar
12	操作 ip	ip_address	varchar
13	操作地址	ip_source	varchar

（21）每日浏览表（tb_unique_view）

表 4-21 每日浏览表

序号	数据名称	字段名	数据类型
1	每日 id	id	int
2	访问量	views_count	int
3	创建时间	create_time	datetime
4	更新时间	update_time	datetime

（22）网站配置（tb_website_config）

表 4-22 网站配置表

序号	数据名称	字段名	数据类型
1	id	id	int
2	配置信息	config	varchar
3	创建时间	create_time	datetime
4	更新时间	update_time	datetime

4.2.5 物理设计

（1）数据表的存储引擎

常见的存储引擎有以下几种：InnoDB、MyISAM、Memory、TokuDB、RocksDB、WiredTiger。

在该数据库中使用的存储引擎是 InnoDB，因为 InnoDB 支持事务、ACID 特性以及多版本并发控制等功能，符合 Java 程序开发中对于数据一致性和可靠性的要

求。此外，在 JDBC 驱动程序中，InnoDB 是默认的存储引擎。因此，使用 Java 程序开发时，InnoDB 是一个很常用的存储引擎选择，可以满足大多数应用的需求[8]。

（2）数据表的索引优化

数据表中所有主键和外键都默认添加了索引，除此之外的索引还有如下。

评论表（tb_comment）的父评论 id（parent_id）：博客中的评论具有一级和二级之分，二级评论属于一级评论的“回复”，因此需要父评论 id（parent_id）该字段来快速定位二级评论的坐标。

用户注册登录信息表（tb_user_auth）的用户名（username）：每个注册用户都会有不唯一的用户名，方便管理员快速检索。

友链表（tb_friend_link）友链名（link_name）：被管理员加入友链的朋友都会有不唯一的友链名，方便管理员在后台快速检索。

5 系统实现

5.1 系统架构实现

根据需求分析和系统设计，包括但不仅限于以下系统步骤：用户登录、发表文章、搜索文章、评论等。整体系统大致分为视图层、服务层和持久层。其中，用户和管理员将访问前端部分，处理程序通过 JSON 同服务器通信，进而调用后端方法以完成数据的传输和渲染。依次传输请求由 RESTful API 完成， Spring Boot 微服务将处理所有逻辑以完成对数据库的操作。我将从以下几个能够体现系统技术功能的实现进行描写[9]。

5.1.1 用户登录和注册功能实现

（1）前端中导入 axios 请求库。

```
import axios from "axios";
import VueAxios from "vue-axios";
```

创建一个登陆实例。

```
const routes = [
  {
    path: "/login",
    name: "登录",
    hidden: true,
    component: () => import("../views/login/Login.vue")
  }
];
```

向后端发送登录请求，后端处理后返回数据，如图 5-1、5-2。

```
let param = new URLSearchParams();
param.append("username", that.loginForm.username);
param.append("password", that.loginForm.password);
that.axios.post("/api/login", param).then(({ data }) => {
  if (data.flag) {
```

```
            // 登录后保存用户信息
            that.$store.commit("login", data.data);
            // 加载用户菜单
            generaMenu();
            that.$message.success("登录成功");
            that.$router.push({ path: "/" });
          } else {
            that.$message.error(data.message);
          }
        });
```

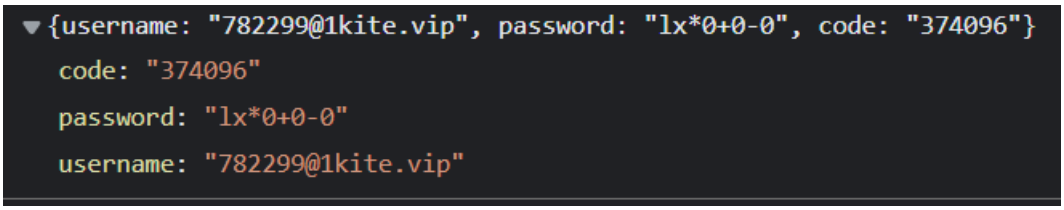

图 5-1 发送数据

图 5-2 后端返回数据数据

（2）UserAuthController.java 控制类。用于处理用户登陆或者注册的请求。使用 userAuthService 对象从数据库中找寻数据，然后调用 UserAuthServiceImpl 中的

register 方法对数据进行处理，也就是判断邮箱是否被注册，如果没有则新建用户信息、绑定用户角色、新增用户账号。

UserAuthServiceImpl.java 数据服务实现类。在此处判断邮箱是否被注册，进而创建用户。

代码详情请见附录 1 用户登录和注册功能实现。

（3）第三方登录

第三方网站主要通过使用“QQ 登录”接入 QQ 互联开放平台。

接入 QQ 第三方登录的好处主要有以下几点：

（1）降低用户注册门槛：用户可以使用自己已经拥有的 QQ 账号直接登录网站，无需再填写一遍注册信息，提高了用户使用网站的便捷性。

（2）提升用户留存率：使用 QQ 登录的用户可以快速完成登录操作，减少用户的耐心等待时间，增加用户在网站上停留的时间，提升用户的留存率。

（3）提高网站访问量和用户数：借助 QQ 空间庞大的用户群，网站的信息可以通过好友关系得到进一步的传播，提高网站的访问量和用户数。

（4）增强网站安全性：QQ 第三方登录使用了 OAuth2.0 协议，可以有效避免网站用户信息的泄露和被盗用，增强了网站的安全性。

以下是接入 QQ 登录的具体步骤：

（1）前往 QQ 互联官网 https://connect.qq.com/，创建应用并等待通过。

图 5-3QQ 互联管理中心

（2）在 QQ 开放平台的应用管理页面，配置网站的回调地址和授权作用域。回调地址是用户在 QQ 登录成功后返回的网站页面，授权作用域则决定了你需要获取

哪些用户信息。

图 5-4 平台信息

（3）在网站登录页面或者其他需要接入 QQ 登录的页面上，添加 QQ 登录按钮。这个按钮可以是一个图片链接或者自定义的按钮样式。当用户点击 QQ 登录按钮后，网站会跳转到 QQ 登录页面，引导用户进行 QQ 账号的登录和授权操作。登录成功后，QQ 会将授权码返回到配置的回调地址。

（4）根据获取到的用户信息，可以在系统中验证用户身份，或者创建新的用户账号。在这里我将邮箱注册的用户用一个字段与其区分开。

5.1.2 验证码功能实现

（1）在 pom.xml 中引入 mail 依赖。

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
</dependency>
```

（2）在 application.yml 文件中配置邮箱邮箱主机名和邮箱授权码。

```
# 邮箱配置
mail:
  host: smtp.qq.com
  username: 1274872144@qq.com #邮箱用户名
  password: szvvqocdfmwebabh #邮箱授权码
```

default-encoding: UTF-8

port: 587

（3）使用 EmailDTO 类发送包含验证码的邮件。

校验账号是否合法后生成六位随机数字作为验证码，如图 5-5。

代码详情请见附录 2 验证码功能实现。

图 5-5 注册验证码

5.1.3 文章添加功能实现

（1）前端创建文章实例。

代码详情请见附录 3 前端文章添加功能实现。

（2）调用 ArticleDao.xml 在对数据库执行 SQL 语句，添加文章内容。

代码附录 4 后端文章添加功能实现。

5.1.4 文章搜索功能实现

（1）前端发送 search? current=1&keywords=<搜索内容>的请求。

（2）后端 ArticleServiceImpl.java 类收到关键词执行相对应的 SQL 语句。

代码详情请见附录 5 文章搜索功能实现。

（3）后端将查找后的内容返回给前端，此次使用‘java’作为关键词搜索，如图 5-6、5-7、5-8。

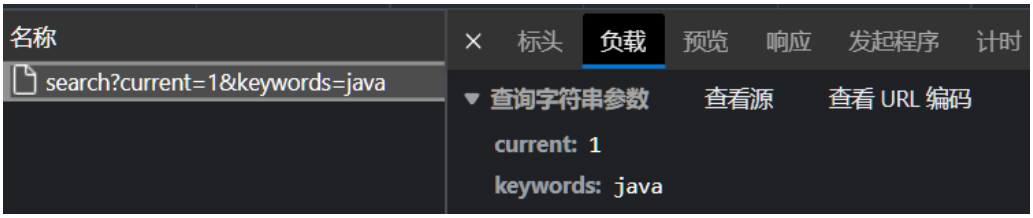

图 5-6 前端发送查询请求

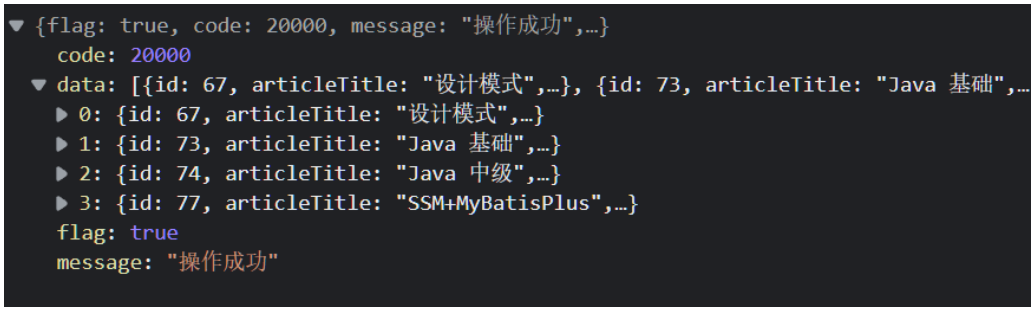

图 5-7 后端返回数据

图 5-8 前端收到 json 数据后展示

5.1.5 评论功能实现

（1）若输入内容包括表情包，则前端只显示表情包的名字，如“[开心]”。

（2）向后端发送 comments 请求，请求内容包含评论内容、文章 id、评论类型。

（3）后端判断是否需要审核，修改评论表的 is_review 属性。

（4）过滤敏感词，此处需要导入所需依赖，而后将内容中的敏感词替换为“*”。

```
<dependency>
    <groupId>com.github.houbb</groupId>
    <artifactId>sensitive-word</artifactId>
     <version>0.2.0</version>
</dependency>
source = WORD_BS.replace(source);
public String replace(String target) {
        return this.replace(target, '*');}
```

（5）过滤图片等标签元素，并删除转义字符、script 标签、style 标签，用于

保留表情包图片。

```
source = source.replaceAll("(?!<(img).*?>)<.*?>", "")
                .replaceAll("(onload(.*?)=)", "")
                .replaceAll("(onerror(.*?)=)", "");
source = source.replaceAll("&.{2,6}?;", "");
source=source.replaceAll("<[\\s]*?script[^>]*?>[\\s\\S]*?<[\\s]*?\\/[\\s]*?script[\\s]*?>", "");
source=source.replaceAll("<[\\s]*?style[^>]*?>[\\s\\S]*?<[\\s]*?\\/[\\s]*?style[\\s]*?>", "");
```

（6）查询回复用户邮箱号，并发送邮件，如图 5-9 所示。

代码详情请见附录 6 评论功能实现。

图 5-9 邮件提醒

5.1.6 照片上传功能实现

图片上传是现代 Web 应用中常见的功能之一。通常，我们会使用文件上传组件将图片文件上传到服务器端（一般是存储在本地服务器或云存储系统中），并将图片信息存储到数据库中。

（1）腾讯云 COS 对象存储

首先在 COS 控制台创建一个 Bucket（存储桶），用于存放上传的图片。

图 5-10 创建存储桶

存储桶的存储量和流量使用情况可以在控制台一览，如下图 5-11。

图 5-6 腾讯云 COS 控制台

（2）引入 COS Java SDK 依赖

```
<dependency>
    <groupId>com.qcloud</groupId>
    <artifactId>cos_api</artifactId>
    <version>5.6.75</version>
</dependency>
```

（3）在后端代码中实例化 COS 客户端对象，并配置相关参数

```
cos:
    url: https://blog-1311123634249.cos.ap-.myqcloud.com/
    secretId: AKIDtiWARYDKUGKSVKXx5O0RhjX6
    secretKey: TiiUP7nmKGCKHJVhjvkhMHGCNYWFjz2rRm
    region: ap-
    bucketName: blog-1311532452378
@Configuration
@ConfigurationProperties(prefix = "upload.cos")
public class CosConfigProperties {
    private String url;
```

```
    private String secretId;
    private String secretKey;
    private String region;
    private String bucketName;
}
```

5.2 数据库配置及实现

基于之前的需求分析，我们决定使用 MySQL 作为数据库，采用 MyBatis-plus 做 ORM。

5.2.1 数据库连接

（1）在 application.xml 文件的 spring 下配置 mysql 数据库。

代码详情请见附录 7 数据库连接。

5.2.2 SQL 语句实现

MyBatisX 这款插件可以轻松创建 SQL 语句。随着系统功能的实现逐步实现对应的 SQl 语句的完善，最终完成如下图 5-5。

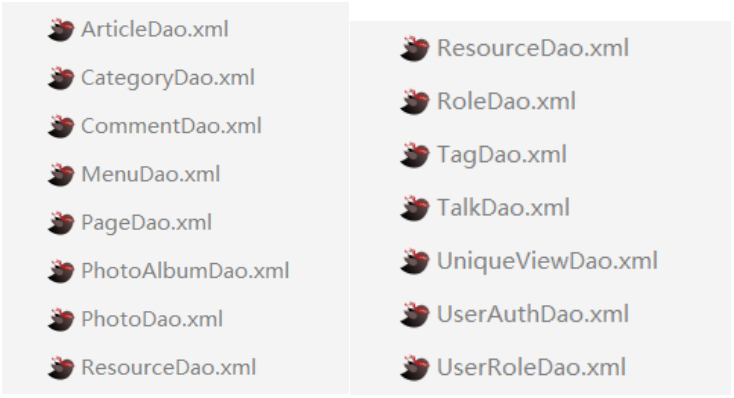

图 5-5SQL 代码文件

5.2.3 缓存技术配置和实现

这里使用 Redis 的配置，应用 Redis 提高系统的缓存速度，并将每个客户请求与多个服务器节点整合，增加负载均衡并减少延迟。举例来说：我们可以 Redis 缓存上最新的文章标题列表，连续请求时不需要重复查询数据库，这样就能达到快速响应的效果[10]。

代码详情请见附录 8 缓存技术配置和实现。

6 系统测试与部署

6.1 功能测试

功能测试是对整个系统进行测试，通常需要模拟真实的用户场景来测试系统的功能、性能和稳定性。

6.1.1 测试环境搭建

在进行功能测试之前，需要先搭建测试环境。测试环境应该尽可能接近实际生产环境，包括操作系统、数据库、网络等。测试环境应该与开发环境和生产环境隔离，以避免对生产环境造成影响。这里使用 My VMware 虚拟机在 Windows 系统上模拟 Linux 服务器，系统是 Ubuntu 22.04 - x86_64 版本，配置为 2 vCPUs、2 GB RAM、80 GB Disk。

6.1.2 测试用例设计

在进行功能测试之前，需要设计测试用例。测试用例应该尽可能覆盖系统的各个功能，并考虑各种可能的输入和情况。测试用例应该清晰、可执行，并有明确的预期结果[11]。

表 6-1 测试用例表

序号	用例名	行为角色	操作过程	测试结果
1	查看首页等前台页面	访客	浏览器输入[虚拟机 ip:81]	页面加载成功
2	查看首页等前台页面	用户、管理员	用户登陆后执行上述操作	页面加载成功
3	文章评论	访客	进入文章页面在底部评论	弹出登陆页面
4	文章评论	用户、管理员	进入文章页面在底部评论	评论加入审核
5	留言板留言	访客、用户、管理员	进入留言板页面并留言	留言成功，头像加载成功
6	用户注册	用户	注册页面输入邮箱、验证码及密码	注册成功并跳转登陆页面
7	用户注册	用户	输入错误邮箱	注册失败并提示邮箱格式不正确

8	用户邮箱登录	用户	用户输入正确的用户名和密码进行登陆	登陆成功
9	用户邮箱登录	用户	用户输入错误的用户名或密码进行登	登陆失败
10	用户第三方登录	用户	登陆页面使用 QQ 或微博登录	登陆成功
11	管理员登陆	管理员	浏览器输入[虚拟机 ip:81]	登录成功
12	发布文章	管理员	输入不能为空的内容后发布	成功添加文章并显示在文章列表中
13	删除文章	管理员	管理员删除一篇文章	文章被删除并从文章列表中移除
14	修改文章	管理员	替换新标题、新内容、新标签、新分类等信息	文章被成功修改并显示更新后的信息
15	分类管理	管理员	添加、删除分类	操作成功
16	标签管理	管理员	添加、删除标签	操作成功
17	添加评论	管理员	评论 ID、审核状态（通过）	评论被成功保存并显示在文章评论列表中
18	删除评论	管理员	删除一条评论	评论被从文章评论列表中删除
19	留言管理	管理员	审核、删除留言	操作成功
20	添加角色	管理员	添加角色名称、角色描述、角色权限等信息	添加成功并显示在角色列表
21	修改用户角色	管理员	修改除管理员以外的用户角色	角色权限被成功更新
22	用户强制下线	管理员	将登录用户踢下线	操作成功
23	添加、删除角色	管理员	添加角色并赋予资源权限、删除角色	操作成功
24	网站信息管理	管理员	添加、删除或修改网站名称、简介、公告等信息	操作成功
25	添加、删除照片	管理员	上传、删除照片	上传成功

26	添加错误照片	管理员	上传错误格式的文件	仍然上传成功但是不保存信息
27	说说管理	管理员	发表、修改、删除说说	操作成功
28	个人中心	管理员	修改管理员账户和密码	操作成功

测试结果如下图所示。

首页功能无误，如图 6-1。

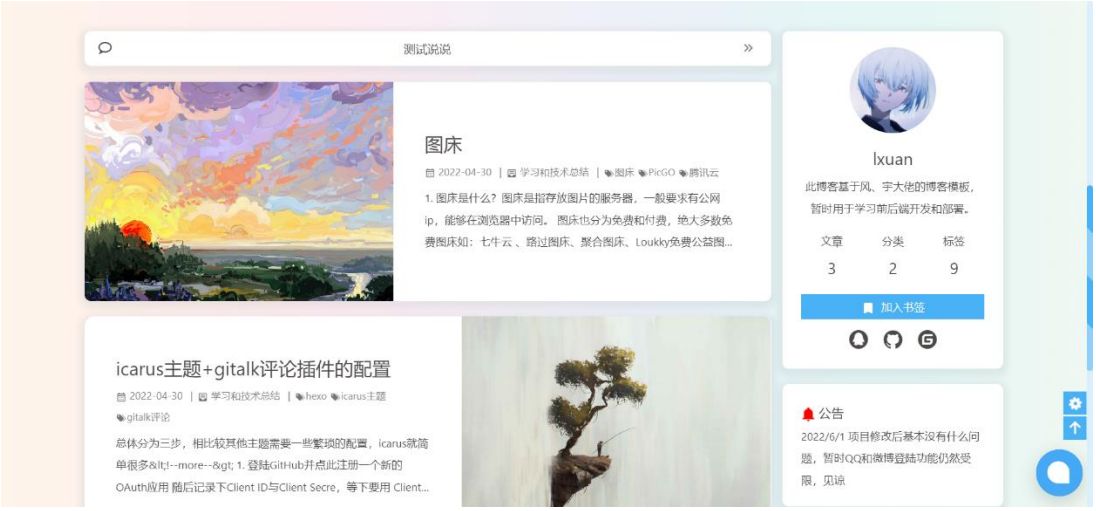

图 6-1 首页测试

文章搜索功能无误，如图 6-2。

图 6-2 文章搜索测试

文章显示功能无误，如图 6-3。

图 6-3 文章显示测试

留言功能无误，如图 6-4。

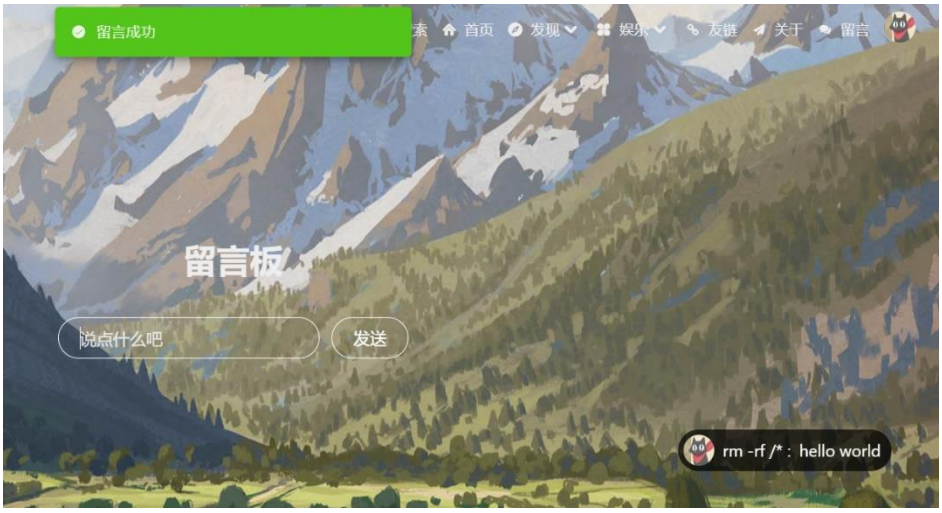

图 6-4 留言板测试

后台登陆功能无误，如图 6-5。

图 6-5 后台登陆测试

文章添加功能无误，如图 6-6。

图 6-6 文章添加功能测试

评论管理功能无误，如图 6-7。

图 6-7 评论管理测试

用户管理功能无误，如图 6-8。

图 6-8 用户管理测试

密码修改功能无误，如图 4-9。

图 6-9 管理员密码修改测试

6.1.3 测试报告及总结

以上是一个简单的测试用例列表，在测试用例时，还需考虑如何设计测试数据、测试环境和测试工具等方面，以便进行有效的测试和分析。

测试报告是对测试过程和结果的总结和分析，是测试工作的重要成果之一。

本次测试旨在对基于 SpringBoot 的个人博客管理系统进行功能测试，包括用户的登陆注册测试、文章的添加删除和修改测试、评论的发表和审核测试、图片上传测试、系统页面管理测试、角色权限管理测试等。测试覆盖了系统的主要功能和模块，采用了手动测试的方式。

测试结果基本全部通过，但是还存在不少问题：

（1）文章添加时，缺少对文章 markdown 格式的适配

（2）用户注册时，密码要求过于严格，不易记忆；

（3）图片上传功能测试中，上传大图片时，页面反应较慢，甚至一度卡死，建议后续设置单文件大小上线；

（4）系统页面管理功能测试中，添加页面时，没有对页面名称和内容进行长度限制。

为了提高系统的质量和用户体验，进行以下改进：

（1）调整密码要求，提高用户的注册和登陆体验；

（2）对文章内容格式进行限制，避免用户输入非法字符；

（3）修复评论审核通过后没有显示在文章详情页面的问题；

（4）对图片上传功能进行限制，提高上传速度；

（5）添加对页面名称和内容长度的限制，避免用户输入过长的内容。

本次测试对基于 SpringBoot 的个人博客管理系统进行了功能测试，覆盖了系统的主要功能和模块。测试结果显示，系统的功能正常，用户体验较好。在测试过程中发现了一些问题和改进点，需要进行优化和调整。测试报告为项目的开发和改进提供了重要的参考和指导，对项目的后续工作有很大帮助[11]。

6.2 安全测试

安全测试是保证系统安全性的重要手段之一，主要是通过模拟攻击和漏洞检测来发现系统的安全问题，从而提供针对性的解决方案。针对基于 SpringBoot 的个人博客管理系统，可以进行以下安全测试：

（1）SQL 注入测试：通过在表单中输入 SQL 语句，测试系统是否能够正确地过滤并防止 SQL 注入攻击；

（2）XSS 攻击测试：通过在表单中输入脚本代码，测试系统是否能够正确地过滤并防止 XSS 攻击；

（3）CSRF 攻击测试：通过伪造请求，测试系统是否能够正确地验证请求来源并防止 CSRF 攻击；

（4）文件上传测试：测试系统是否能够正确地限制文件类型和大小，并防止恶意文件上传；

（5）访问控制测试：测试系统是否能够正确地限制用户访问权限，避免未授权访问；

（6）加密测试：测试系统是否能够正确地对用户密码等敏感信息进行加密处理，保证用户信息安全；

（7）弱口令测试：测试系统是否能够正确地限制用户密码强度，避免用户使用弱口令[12]。

通过以上安全测试，可以发现系统中存在的安全漏洞和问题，并分别在前端和后端提供相应的解决方案，从而保障系统的安全性和用户信息的安全。

6.3 项目部署及调试

在完成单元测试、功能测试和安全测试之后，我们需要将基于 SpringBoot 的个人博客管理系统部署到服务器上，以供用户访问和使用。以下是项目部署的步骤：

6.3.1 运行环境准备

服务器：cloudcone 2 vCPUs、2 GB RAM

操作系统：Ubuntu 22.04 - x86_64

域名：lxuan.fun

CDN：cloudflare 全站加速

对象存储：腾讯云 COS

6.3.2 部署并上线

（1）首先在 application.yml 文件中将本地配置替换为远程配置，并启动远程软件：mysql、radis、docker 等。

（2）其次在 IDEA 中使用 Maven 的 package 命令对 Java 项目打包、使用 npm 的 build 命令对前端代码 admin 和 blog 分别打包。

（3）然后将上述三个打包文件上传至服务器，并将本地数据库代码上传到远程数据库。

（4）最后编写 Dockerfile 文件在 Docker 上运行 Java 程序，而前端代码则在 Nginx 中代理，并将前台页面和后台页面分别反向代理到不同的三级域名：www.lxuan.fun、admin.lxuan.fun。

6.3.3 项目调试

依次点击各个页面，用不同角色来模拟用户操作，确保项目中已上线的所有功能一切正常后，自此宣布项目部署完毕。

总　结

在本次毕业设计中，我基于 SpringBoot 开发了一个个人博客管理系统。该系统旨在为用户提供一个方便管理和分享博客文章的平台。通过系统的设计和实现，我深入理解了 SpringBoot 框架的应用和数据库设计的重要性。

在需求分析阶段，我明确了系统的功能需求和用户需求，包括用户登录、注册、文章管理、评论管理等。这些需求直接影响了系统的设计和开发过程。

在概念设计阶段，我通过绘制 E-R 图和定义实体与关系，明确了系统中各个实体之间的关系和属性。这为后续的逻辑设计提供了基础。

在逻辑设计阶段，我进一步细化了系统的功能模块，并设计了相应的数据库表结构。通过合理的表设计和关系建立，实现了用户、文章、标签、分类等实体的存储和关联。

在物理设计阶段，我选择了适合的数据库管理系统（MySQL）和存储设备（磁盘阵列），并设计了合适的索引以提高数据的检索效率。同时，我进行了存储空间分配，将数据表和索引分别存储在不同的磁盘分区上，以提高数据访问速度和可靠性。

在数据库的实施与维护阶段，我实现了系统的数据库功能，并进行了测试和优化。我采取了备份和恢复策略，确保数据的安全性和可靠性。同时，我也进行了数据库性能监控和调优，提高了系统的响应速度和并发能力。

通过本次毕业设计，我不仅加深了对 SpringBoot 框架和数据库设计的理解，还提升了系统设计与实现的能力。我深入学习了需求分析、概念设计、逻辑设计、物理设计以及数据库实施与维护等环节，掌握了系统开发的全过程。

总而言之，本次毕业设计不仅是对我在学校学到知识的整合和应用，也是对我职业发展的重要积累。通过这个项目，我不仅提高了编程和数据库技术，还培养了团队合作、项目管理和问题解决的能力。我相信这些经验和技能将对我的未来职业发展产生积极影响。

参考文献

[1] 罗路腾，王贵鑫 . 基于 Springboot 的博客网站的设计与 实现 [J]. 科学技术创新，2019（33）：64-66.

[2] 李孟津，杨丹 . 基于 SpringBoot 的在线招聘网站的设计 与实现 [J]. 科学技术创新，2020（26）：98-99.

[3] 熊永平 . 基于 SpringBoot 框架应用开发技术的分析与研 究 [J]. 电脑知识与技术，2019，15（36）：76-77.

[4] 杨伟凡 . 基于 Java 技术平台的在线考试系统的设计与实 现 [J]. 卫星电视与宽带多媒体，2020（3）：99-100.

[5] 王丹，孙晓宇，杨路斌，等 . 基于 SpringBoot 的软件统 计分析系统设计与实现 [J]. 软件工程，2019，22（3）：40-42.

[6] 余思源，张伟 . 基于 JAVA 的个人博客系统的设计与实现 [J]. 电脑知识与技术，2018，14（17）：135-137.

[7] 邓笑 . 基于 Spring Boot 的校园轻博客系统的设计与实现 [D]. 武汉：华中科技大学，2018.

[8] 唐炜 .Spring Data、MongoDB、Thymeleaf 的数据持久 化方案及分页技术实现 [J]. 陇东学院学报，2017，28（5）：9-13.

[9] 刘磊. 基于 SSH 框架的博客用户分享平台的设计与实现[D]. 河北工业大学, 2017.

[10] 陈玲，夏汛 . 利用 Mybatis 的动态 SQL 实现物理分页 [J].数字技术与应用，2011（11）：227.

[11] Regmi A, Alsadoon A, Withana C, et al. Impact of privacy invasion in social network sites IEEE,Computing and Communication Workshop and Conference. IEEE,2018,31(4):457-462

[12] Patil M M, Hanni A, Tejeshwar C H, et al. A qualitative analysis of the performance of MongoDB vs MySQL database based on insertion and retriewal operations using a web/android application to explore load balancing-Sharding in MongoDB and its advantages International Conference on I-Smac.2017,45(9):325-330

致　谢

在完成毕业设计的过程中，我要感谢许多人和机构对我的支持和帮助。他们的鼓励和支持使得我能够顺利完成这个项目。

首先，我要衷心感谢我的指导老师，对我的毕业设计给予了悉心的指导和建议。您的专业知识和经验对我产生了深远的影响，帮助我理清思路，解决问题，并在项目开发过程中提供了宝贵的指导。

其次，我要感谢我的家人和朋友。感谢他们在我整个学习过程中的支持和理解，鼓励我坚持下去，克服困难。他们的支持给了我无限的动力，使我能够克服挑战，坚持不懈地追求我的目标。

我还要感谢我的同学们，特别是那些和我一起合作和讨论的同学们。感谢你们的合作和贡献，我们共同面对挑战，一起努力解决问题，相互支持和帮助。你们的友谊和团队合作精神是我成功完成毕业设计的重要支持。

最后，我要感谢所有为我提供信息和资源的机构和个人。感谢开源社区的贡献者们，他们的开源项目为我的毕业设计提供了宝贵的参考和工具。感谢所有提供指导和建议的专家和学者们，他们的研究成果为我的毕业设计提供了理论基础。

在此，我再次向所有支持和帮助过我的人表示由衷的感谢。没有你们的支持和鼓励，我将无法完成这个毕业设计。感谢你们的付出和支持，我将倍加珍惜，并努力将所学应用于实践，为社会做出更大的贡献。谢谢大家！

附　录

附录 1 用户登录和注册功能实现

```
/**
* 用户注册
*
* @param user 用户信息
* @return {@link Result<>}
*/
@ApiOperation(value = "用户注册")
@PostMapping("/register")
public Result<?> register(@Valid @RequestBody UserVO user) {
userAuthService.register(user);
return Result.ok();
}
@Transactional(rollbackFor = Exception.class)
    @Override
    public void register(UserVO user) {
        // 校验账号是否合法
        if (checkUser(user)) {
            throw new BizException("邮箱已被注册！");
        }
        // 新增用户信息
        UserInfo userInfo = UserInfo.builder()
                .email(user.getUsername())
                .nickname(CommonConst.DEFAULT_NICKNAME +
IdWorker.getId())
                .avatar(blogInfoService.getWebsiteConfig().getUserAvatar())
```

```
                .build();
        userInfoDao.insert(userInfo);
        // 绑定用户角色
        UserRole userRole = UserRole.builder()
                .userId(userInfo.getId())
                .roleId(RoleEnum.USER.getRoleId())
                .build();
        userRoleDao.insert(userRole);
        // 新增用户账号
        UserAuth userAuth = UserAuth.builder()
                .userInfoId(userInfo.getId())
                .username(user.getUsername())
                .password(BCrypt.hashpw(user.getPassword(), BCrypt.gensalt()))
                .loginType(LoginTypeEnum.EMAIL.getType())
                .build();
        userAuthDao.insert(userAuth);
    }
```

附录 2 验证码功能实现

```
@Override
    public void sendCode(String username) {
        // 校验账号是否合法
        if (!checkEmail(username)) {
            throw new BizException("请输入正确邮箱");
        }
        // 生成六位随机验证码发送
        String code = getRandomCode();
        // 发送验证码
        EmailDTO emailDTO = EmailDTO.builder()
                .email(username)
                .subject("验证码")
                .content("您的验证码为 " + code + " 有效期 15 分钟，请不要告诉他人哦！")
                .build();
        rabbitTemplate.convertAndSend(EMAIL_EXCHANGE, "*", new Message(JSON.toJSONBytes(emailDTO), new MessageProperties()));
        // 将验证码存入 redis，设置过期时间为 15 分钟
        redisService.set(USER_CODE_KEY + username, code, CODE_EXPIRE_TIME);
    }
```

附录 3 前端文章添加功能实现

```
import * as imageConversion from "image-conversion";
export default {
  created() {
    const path = this.$route.path;
    const arr = path.split("/");
    const articleId = arr[2];
    if (articleId) {
      this.axios.get("/api/admin/articles/" + articleId).then(({ data }) => {
        this.article = data.data;
      });
    } else {
      const article = sessionStorage.getItem("article");
      if (article) {
        this.article = JSON.parse(article);
      }
    }
  },
    openModel() {
      if (this.article.articleTitle.trim() == "") {
        this.$message.error("文章标题不能为空");
        return false;
      }
      if (this.article.articleContent.trim() == "") {
        this.$message.error("文章内容不能为空");
        return false;
```

附录 4 后端文章添加功能实现

```
<select id="listArticlesByCondition" resultMap="articlePreviewResultMap">
        SELECT
        a.id,
        article_cover,
        article_title,
        a.create_time,
        a.category_id,
        category_name,
        t.id AS tag_id,
        t.tag_name
        FROM
        (
        SELECT
        id,
        article_cover,
        article_title,
        article_content,
        create_time,
        category_id
        FROM
        tb_article
        <where>
            <if test="condition.categoryId != null">
                category_id = #{condition.categoryId}
            </if>
            <if test="condition.tagId != null">
```

```
                id IN (
                SELECT
                article_id
                FROM
                tb_article_tag
                WHERE
                tag_id = #{condition.tagId})
            </if>
        </where>
        AND is_delete = 0
        AND status = 1
        ORDER BY id DESC
        LIMIT #{current},#{size}
        ) a
        JOIN tb_category c ON a.category_id = c.id
        JOIN tb_article_tag atg ON a.id = atg.article_id
        JOIN tb_tag t ON t.id = atg.tag_id
</select>
```

附录 5 文章搜索功能实现

```
    @Override
        PublicArticlePreviewListDTO listArticlesByCondition(ConditionVO condition)
{
            // 查询文章
            List<ArticlePreviewDTO> articlePreviewDTOList =
articleDao.listArticlesByCondition(PageUtils.getLimitCurrent(), PageUtils.getSize(),
condition);
            // 搜索条件对应名(标签或分类名)
            String name;
            if (Objects.nonNull(condition.getCategoryId())) {
                name = categoryDao.selectOne(new
LambdaQueryWrapper<Category>()
                                .select(Category::getCategoryName)
                                .eq(Category::getId, condition.getCategoryId()))
                        .getCategoryName();
            } else {
                name = tagService.getOne(new LambdaQueryWrapper<Tag>()
                                .select(Tag::getTagName)
                                .eq(Tag::getId, condition.getTagId()))
                        .getTagName();
            }
            return ArticlePreviewListDTO.builder()
                    .articlePreviewDTOList(articlePreviewDTOList)
                    .name(name)
                    .build();
        }
```

附录 6 评论功能实现

```
                Integer userId = BLOGGER_ID;
                String id = Objects.nonNull(comment.getTopicId()) ?
comment.getTopicId().toString() : "";
                if (Objects.nonNull(comment.getReplyUserId())) {
                    userId = comment.getReplyUserId();
                } else {
                    switch
(Objects.requireNonNull(getCommentEnum(comment.getType()))) {
                            case ARTICLE:
                                userId =
articleDao.selectById(comment.getTopicId()).getUserId();
                                break;
                            case TALK:
                                userId =
talkDao.selectById(comment.getTopicId()).getUserId();
                                break;
                            default:
                                break;
                    }
                }
        String email = userInfoDao.selectById(userId).getEmail();
                if (StringUtils.isNotBlank(email)) {
                    EmailDTO emailDTO = new EmailDTO();
                    if (comment.getIsReview().equals(TRUE)) {
                        // 评论提醒
                        emailDTO.setEmail(email);
```

```
                emailDTO.setSubject("评论提醒");
                // 获取评论路径
                String url = websiteUrl + getCommentPath(comment.getType())
+ id;
                emailDTO.setContent("您收到了一条新的回复，请前往" + url
+ "\n 页面查看");
            } else {
                // 管理员审核提醒
                String adminEmail =
userInfoDao.selectById(BLOGGER_ID).getEmail();
                emailDTO.setEmail(adminEmail);
                emailDTO.setSubject("审核提醒");
                emailDTO.setContent("您收到了一条新的回复，请前往后台管
理页面审核");
            }
            rabbitTemplate.convertAndSend(EMAIL_EXCHANGE, "*", new
Message(JSON.toJSONBytes(emailDTO), new MessageProperties()));
        }
```

附录 7 数据库连接

```
# 配置 mysql 数据库
spring:
  datasource:
    type: com.zaxxer.hikari.HikariDataSource
    driver-class-name: com.mysql.cj.jdbc.Driver
    url:
jdbc:mysql:/127.0.0.1:3306/blog?serverTimezone=Asia/Shanghai&allowMultiQueries=tru
e
    username: root
    password: 123456789
    hikari:
      minimum-idle: 5
      # 空闲连接存活最大时间，默认 600000（10 分钟）
      idle-timeout: 180000
      # 连接池最大连接数，默认是 10
      maximum-pool-size: 10
      # 此属性控制从池返回的连接的默认自动提交行为,默认值：true
      auto-commit: true
      # 连接池名称
      pool-name: MyHikariCP
      # 此属性控制池中连接的最长生命周期，值 0 表示无限生命周期，默认
1800000 即 30 分钟
      max-lifetime: 1800000
      # 数据库连接超时时间,默认 30 秒，即 30000
      connection-timeout: 30000
      connection-test-query: SELECT 1
```

附录 8 缓存技术配置和实现

```
# redis 配置
redis:
  host: 127.0.0.1
  port: 6379
  password: 123456789
@Configuration
public class RedisConfig {

  @Bean
  public RedisTemplate<String, Object> redisTemplate(RedisConnectionFactory
factory) {
    RedisTemplate<String, Object> redisTemplate = new RedisTemplate<>();
    redisTemplate.setConnectionFactory(factory);
    Jackson2JsonRedisSerializer<Object> jackson2JsonRedisSerializer = new
Jackson2JsonRedisSerializer<>(Object.class);
    ObjectMapper mapper = new ObjectMapper();
    mapper.setVisibility(PropertyAccessor.ALL,
JsonAutoDetect.Visibility.ANY);
    //
mapper.enableDefaultTyping(ObjectMapper.DefaultTyping.NON_FINAL);
    mapper.activateDefaultTyping(LaissezFaireSubTypeValidator.instance,
        ObjectMapper.DefaultTyping.NON_FINAL,
JsonTypeInfo.As.PROPERTY);
    jackson2JsonRedisSerializer.setObjectMapper(mapper);
    StringRedisSerializer stringRedisSerializer = new StringRedisSerializer();
    // key 采用 String 的序列化方式
```

```
        redisTemplate.setKeySerializer(stringRedisSerializer);
        // hash 的 key 也采用 String 的序列化方式
        redisTemplate.setHashKeySerializer(stringRedisSerializer);
        // value 序列化方式采用 jackson
        redisTemplate.setValueSerializer(jackson2JsonRedisSerializer);
        // hash 的 value 序列化方式采用 jackson
        redisTemplate.setHashValueSerializer(jackson2JsonRedisSerializer);
        redisTemplate.afterPropertiesSet();
        return redisTemplate;
    }
}
```