****

**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用户注册、登录、博客文章的创建、编辑、删除、浏览、评论等。系统采用了 Spring Boot 作为主要框架，使用了 Spring Security 来实现用户认证和授权，使用了 Thymeleaf 模板引擎和 Vue 框架来构建前端页面，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](#_Toc130233862)

[1.1 课题研究背景 1](#_Toc130233863)

[1.2 开发意义 1](#_Toc130233864)

[1.3 国内研究现状 2](#_Toc130233865)

[1.4 研究问题和目标 2](#_Toc130233866)

[2 相关技术介绍 4](#_Toc130233867)

[2.1 开发工具 4](#_Toc130233868)

[2.1.1 IDEA 4](#_Toc130233869)

[2.1.2 VSCode 4](#_Toc130233870)

[2.1.3 Navicat 4](#_Toc130233871)

[2.1.4 X-shell和Xftp 4](#_Toc130233872)

[2.2 数据库 4](#_Toc130233873)

[2.2.1 MySQL 4](#_Toc130233874)

[2.2.2 Redis 4](#_Toc130233875)

[2.3 服务器 5](#_Toc130233876)

[2.3.1 Nginx 5](#_Toc130233877)

[2.3.2 Docker 5](#_Toc130233878)

[2.4 框架 5](#_Toc130233879)

[2.4.1 Spring Boot 5](#_Toc130233880)

[2.4.2 MyBatisPlus 5](#_Toc130233881)

[2.4.3 Vue 5](#_Toc130233882)

[3. 需求分析 6](#_Toc130233883)

[3.1 功能需求 6](#_Toc130233884)

[3.2 性能需求 6](#_Toc130233885)

[3.3 用户需求 6](#_Toc130233886)

[3.4 界面需求 7](#_Toc130233887)

[3.5 用例 7](#_Toc130233888)

[3.6 数据需求 7](#_Toc130233889)

[3.7 可维护性需求 7](#_Toc130233890)

[3.8 兼容性需求 7](#_Toc130233891)

[4 系统设计 8](#_Toc130233892)

[4.1 系统功能设计 8](#_Toc130233893)

[4.1.1 功能总体设计 8](#_Toc130233894)

[4.1.2 前台功能设计 9](#_Toc130233895)

[4.1.3 后台功能设计 10](#_Toc130233896)

[在这个示例中，我们可以看到有十二个实体：用户信息、用户角色、角色、角色功能、文章、文章标签、分类、标签、评论、回复、日志和友链。它们之间的关系如下： 12](#_Toc130233897)

[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，因此用户信息和用户角色之间的关系是一对多。 12](#_Toc130233898)

[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，因此用户角色和用户信息之间的关系也是一对多。 12](#_Toc130233899)

[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。 12](#_Toc130233900)

[每个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，因此角色功能和角色之间的关系也是一对多。 12](#_Toc130233901)

[每篇文章可以有多个标签，即一篇文章对应多个文章标签，因此文章和文章标签之间的关系是多对多。 12](#_Toc130233902)

[每个标签可以被多篇文章使用，即一个标签对应多篇文章，因此标签和文章之间的关系也是多对多。 12](#_Toc130233903)

[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。 12](#_Toc130233904)

[每篇文章可以被归属到一个分类，即一篇文章对应一个分类，因此文章和分类之间的关系也是一对多。 12](#_Toc130233905)

[每个评论可以有多个回复，即一个评论对应多个回复，因此评论和回复之间的关系是一对多。 12](#_Toc130233906)

[每个回复都必须对应一个评论，因此回复和评论之间的关系是多对一。 12](#_Toc130233907)

[每个日志可以对应多个用户信息，即一个日志对应多个用户信息，因此日志和用户信息之间的关系是一对多。 12](#_Toc130233908)

[每个友链都必须对应一个用户信息，因此友链和用户信息之间的关系是一对多。 12](#_Toc130233909)

[除此之外，我们还可以看到每个实体都有其自身的属性，例如用户信息实体具有用户名、密码和邮箱等属性，角色实体具有角色名称等属性，文章实体具有标题、内容和发布时间等属性，评论实体具有评论内容和评论时间等属性。这些属性可以通过 E-R 图中的小矩形来表示。 13](#_Toc130233910)

[...... 13](#_Toc130233911)

[4.2.2 数据库表设计 13](#_Toc130233912)

[4 系统实现 14](#_Toc130233913)

[5 系统测试与部署 15](#_Toc130233914)

[总 结 16](#_Toc130233915)

[参考文献 17](#_Toc130233916)

[致 谢 18](#_Toc130233917)

[附 录 19](#_Toc130233918)

1 绪论

1.1 课题研究背景

随着时代的进步和科技的发展，以互联网为核心的现代网络和通信技术已经得到了快速的发展和广泛的应用，各种网络通信工具也随时代而生。其中就有QQ、论坛、贴吧、微博等较受广大人民欢迎，也是现在发展的比较成熟的社区形式的信息交流工具。而随着网络技术的日渐成熟，个人博客逐渐成为了一个非常流行的网络形式。越来越多的人开始使用个人博客来分享自己的经验和知识，以及记录自己的生活和思考。

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。

## 1.4 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。

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。

对系统进行测试和评估，验证系统的性能和可靠性。在系统开发完成后，需要对系统进行全面的测试和评估，验证系统的性能和可靠性。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，可以大大提高开发效率。

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：包括用户注册、登录、修改个人信息等功能。

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：包括对角色管理、接口管理和菜单管理等功能。

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：系统需要保证用户数据的安全性，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 用例

用户注册：用户进入注册页面，输入个人信息，系统进行验证并保存信息。

用户登录：用户进入登录页面，输入用户名和密码，系统进行验证并登录。

发布文章：用户进入发布文章页面，输入文章内容和相关信息，系统保存文章信息并发布。

评论文章：用户在文章页面发表评论，系统保存评论信息并审核。

编辑文章：管理员登录后进入文章管理页面，选择要编辑的文章并进行编辑操作，系统保存修改后的文章信息。

管理用户：管理员登录后进入用户管理页面，选择要管理的用户并进行相应操作，系统保存管理结果。

3.6 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

3.7 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。

3.8 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 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。

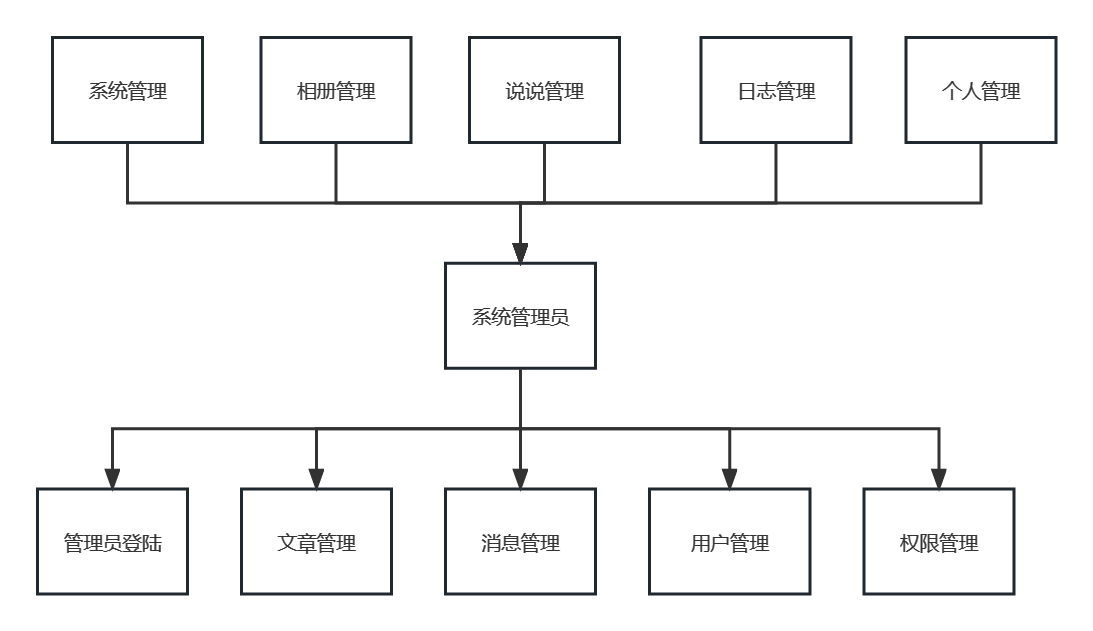


图4-1系统管理员功能模块图

普通用户功能模块如图4-2所示。

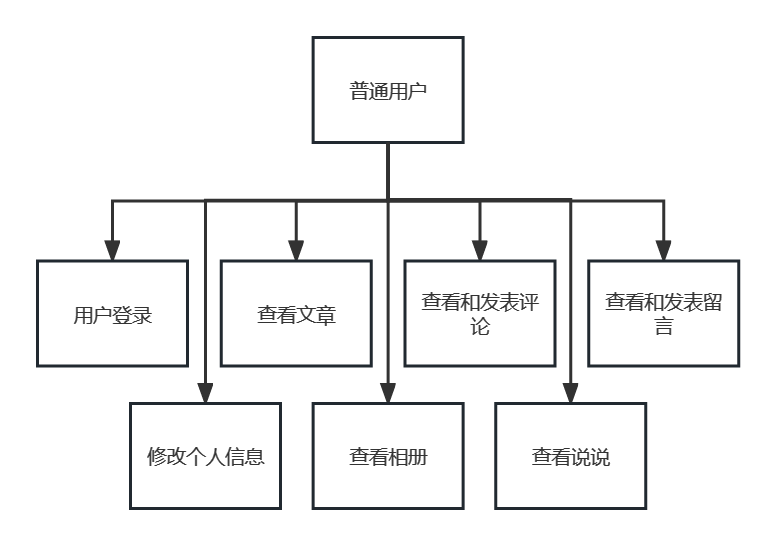


图4-2普通用户功能模块图

游客功能模块如图4-3所示。

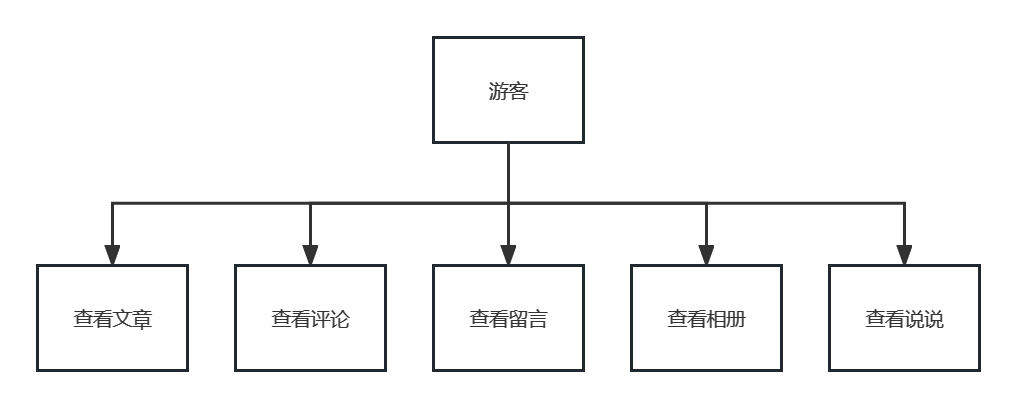


图4-3游客功能模块图

### 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。

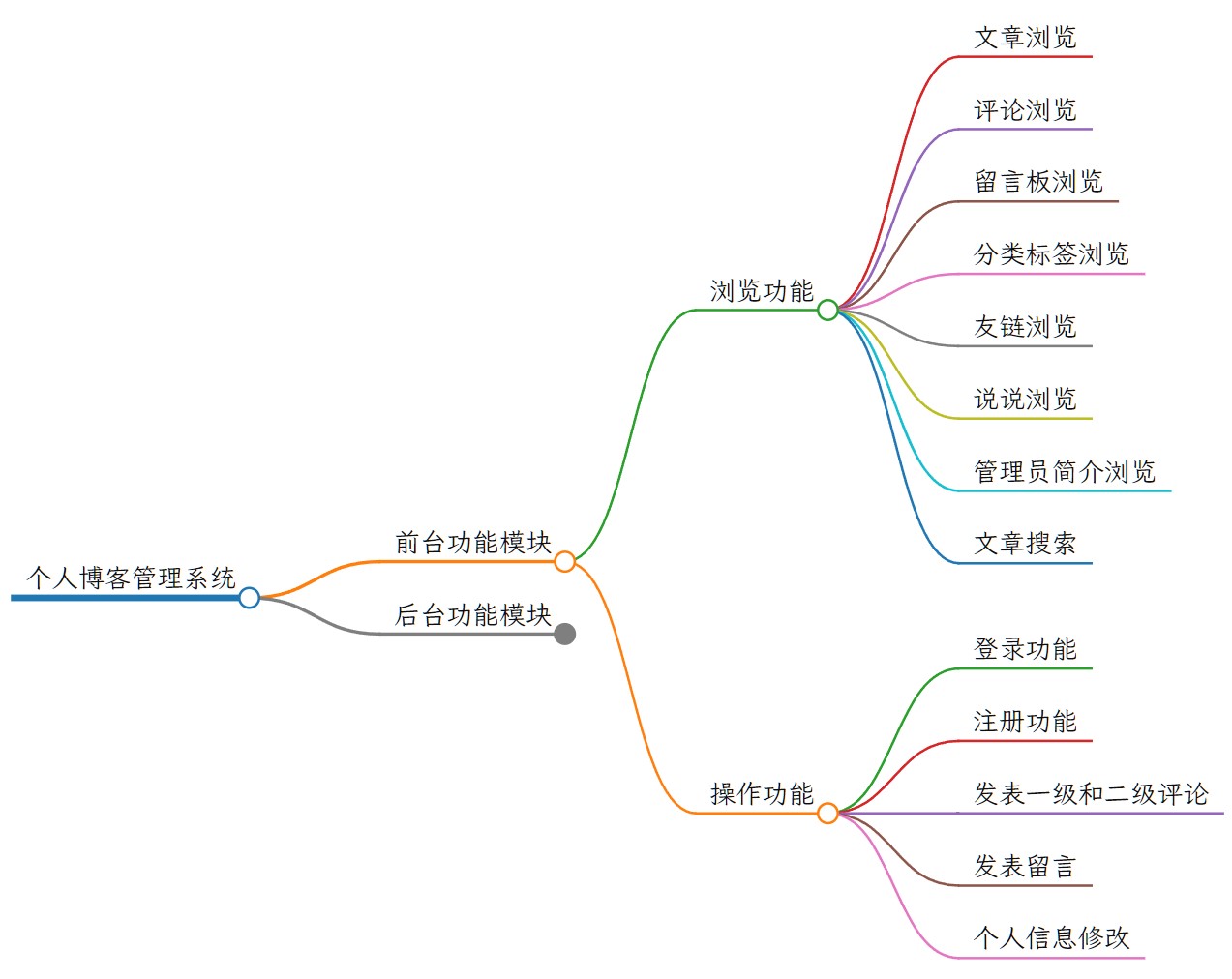


图4-4前台功能模块图

### 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。

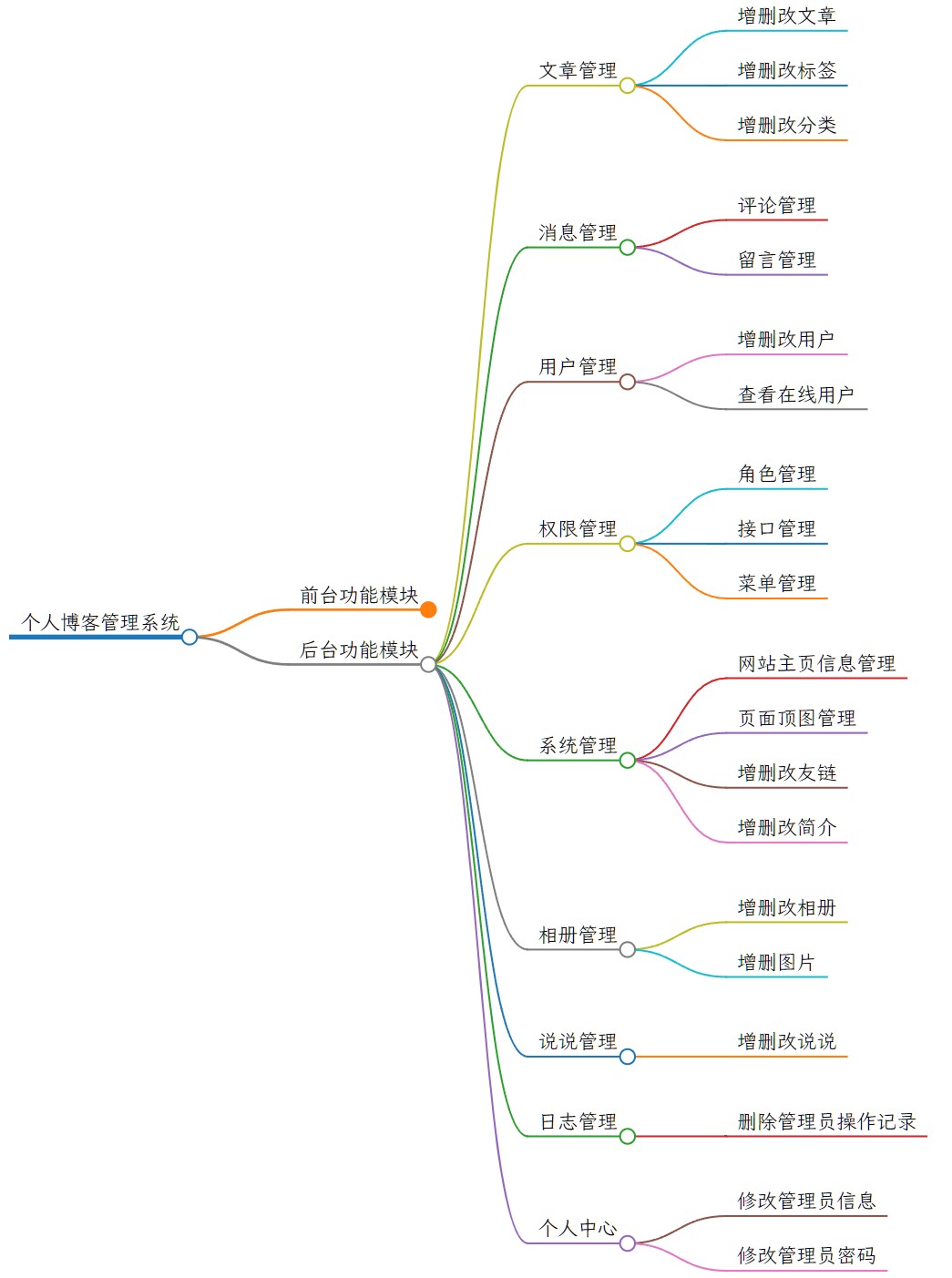


图4-5后台功能模块图

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。

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：

图示

描述已自动生成

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。

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。

* 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）

图示

描述已自动生成根据以上示例画出它们的关系模式图，如图4-7。

图4-7 用户信息、角色、菜单和资源关系模式图

* 用户信息、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）

图示

描述已自动生成根据以上示例画出它们的关系模式图，如图4-8。

图4-7 用户信息、文章、分类和标签关系模式图

文章（文章id、分类id）、文章标签（文章id、标签id）、标签（标签id）、分类（分类id）、

评论、友链、留言、日志、首页置顶图、相册、图片、

页面、权限、角色、角色页面、角色权限、

说说、

每日浏览、

用户注册信息、用户第三方登录信息、用户角色

4.2.4 物理设计

### ......

### 4.2.2 数据库表设计

根据上一节中的E-R图的设计，......

（1）用户信息表（uni-id-users）

表4-1 用户信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 数据名称 | 字段名 | 数据类型 |
| 1 | 用户名 | username | string |
| 2 | 密码 | password | password |
| 3 | 角色 | role | array |
| 4 | 微信用户信息 | wx\_openid | object |
| 5 | 登录信息 | token | array |

4 系统实现

这个章节应该详细介绍系统的实现细节，包括如何使用 Spring Boot 框架来实现业务逻辑、如何使用 Mybatis 框架来操作数据库、如何实现用户界面等。

5 系统测试与部署

这个章节应该讨论如何进行系统测试，包括单元测试、集成测试、系统测试等。同时，你应该介绍如何将系统部署到服务器上，并提供相关的操作说明。

# 总 结

...................................................................... .................................................................

..............................................................................................................................................

...........................................................................................................................................

# 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] 陈秋玲 . 基于 JavaWeb 的图书购物网站的设计与实现 [J]. 电脑知识与技术，2019，15（16）：36-38.

[15]Ravi Tomar,Sarishma Dangi. JavaScript:Syntax and Practices[M].CRC Press:2021-07-18.

# 致 谢

............................................................................................................................................................................................................................................................................。

.............................................................................................................................................................................................................................................................。

# 附 录

........................................................................................................................................................................................................................................