|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **西安电子科技大学网络教育 毕业论文（设计）任务书** | | | | |
| **学习中心** | **河南郑州轨道学习中心** | **专 业** | **计算机科学与技术** | |
| **学 号** | **713400210092062** | **姓 名** | **郜科端** | |
| **题 目** | **基于Web的博客系统设计与实现** | | | |
| **任务与要求** | | | | |
| 完全使用前端技术栈开发一个稳定性高、轻量化、配置化简单、易部署的博客系统。解决现如用户搭建博客配置复杂学习成本高的问题。 | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
| **开始日期** | **2023年1月16号** | **完成日期** | | **2023年3月27号** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **西安电子科技大学网络教育 毕业论文（设计）工作计划** | | | | |
| **学习中心** | **河南郑州轨道学习中心** | | **专 业** | **计算机科学与技术** |
| **学 号** | **713400210092062** | | **姓 名** | **郜科端** |
| **题 目** | **基于Web的博客系统设计与实现** | | | |
| **工作进程** | | | | |
| **起止时间** | | **工作内容** | | |
| 23.01.12-23.01.16 | | 确定选题方向，撰写设计方案、开题报告 | | |
| 23.01.16-23.01.31 | | 查阅资料、项目代码实现、论文撰写 | | |
| 23.01.31-23.02.16 | | 收集、整理资料，论文初稿完成，项目实现初步完成 | | |
| 23.02.16-23.03.06 | | 完善项目功能及论文内容 | | |
| 23.03.06-23.03.27 | | 完成论文查重及格式检测 | | |
|  | |  | | |
|  | |  | | |
|  | |  | | |
|  | |  | | |
| **主要参考书目：**  **《React设计原理》-卡颂**  **《Node.js权威指南》-陆凌牛**  **《JavaScript权威指南》-DAVID FLANAGAN**  **主要仪器设备：**  1. 处理器：Apple M1  2. 内存：16G  3. 操作系统：macOS  4. 开发框架：React 18.2.0、Next.js 12.2.5  5. 开发语言：HTML、CSS、JavaScript  6. 开发工具：Visual Studio Code  7. 开发环境：Node.js 16.12.0  8. 版本控制工具：Git 2.37.1  9. 数据库：MySQL 8.0.29 | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **西安电子科技大学网络教育** **毕业论文（设计）评审表** | | | | |
| **学习中心** | **河南郑州轨道学习中心** | | **专 业** | **计算机科学与技术** |
| **学 号** | **713400210092062** | | **姓 名** | **郜科端** |
| **题 目** | **基于Web的博客系统设计与实现** | | | |
| **论文评语及成绩** | **前期及开题**  **检查情况** | **评语：**  **指导教师签名:** | | |
| **中期检查情况** | **评语：**  **指导教师签名:** | | |
| **后期检查情况** | **评语：**  **指导教师签名:** | | |
| **总体评语及成绩：**  **指导教师签名: 年 月 日** | | | |
| 论文评语应包括：学生的工作态度和投入时间、精力情况；运用基本技能获取、利用信息的情况；学生对基础知识、专业知识的掌握程度；分析、解决问题和进行创新的能力；论文（设计）的质量情况；**须给出成绩并明确是否同意答辩。** | | | |
| **答辩小组意见** | **组长签名: 年 月 日** | | | |
| 答辩小组对评定的“优秀”和“不及格”成绩需给出明确和详细的意见！ | | | |
| **综合评审组意见** | **组长签名: 年 月 日** | | | |

**摘 要**

随着互联网的快速发展，现在博客已经是大家在网络冲浪时必不可少的一部分了。大家写博客的动机时多种多样的，尤其目前深处与信息化时代的浪潮中，大家博客的内容也是多种多样的。从网络日志到会议记录，从生活体验到技术分析。

博客作为一种广受欢迎工具，技术也一直在变化，从一开始的WordPress，到现在流行的静态博客VitePress，Hexo，再以后可能会出现更多的链（区块链）上博客。诚然博客系统已经成为社会网络媒介中重要的一部分。

本系统对当先最流行的Web技术进行了实践，通过Web前端工程化构建项目。采用了基于React渲染和Node.js的上层元框架Next.js，采用了开源且高性能的关系型数据库MySQL。采用了B/S架构以及从软件工程化的方法对系统进行架构。系统分为前台和后台两个部分。前台主要是给用户使用，可以进行文章浏览，编写文章，发布评论和查询文章等操作。系统的重点模包括用户管理、文章管理、评论管理和查询管理，然后描述了用于编写文章的Markdown编辑器。

经过对前沿技术对实践落地，本系统具有稳定性高、轻量化、易部署等特点。且系统界面简约大方，功能合理对于用户体验非常友好。

关键词：博客系统 React.js库 Node.js Next.js框架

**目 录**

第一章 绪论 6

1.1 研究项目背景 6

1.2 研究目的与意义 7

1.3 国内外研究现状 8

1.3.1 博客系统的研究概述 8

1.3.2 React库的研究概述 8

1.3.3 前端工程化的研究概述 9

1.4 主要研究内容 10

第二章 关键技术分析 11

2.1 React前端 11

2.1.1 虚拟DOM 11

2.1.2 Hooks 11

2.1.3 JSX 12

2.1.4 组件化 12

2.2 Node.js后端 12

2.3 MySQL数据库 13

2.4 本章小结 13

第三章 博客系统的需求分析与设计 15

3.1 博客系统需求分析 15

3.1.1 用户管理 15

3.1.2 文章管理 16

3.1.3 评论管理 17

3.1.4 查询管理 17

3.1.5 文章编辑器 17

3.2 博客系统总体设计 18

3.3 博客系统模块设计 20

3.3.1 用户管理 21

3.3.2 文章管理 21

3.3.3 评论管理 22

3.3.4 查询管理 23

3.4 博客系统数据库设计 23

3.5 本章小节 27

第四章 博客系统的功能实现 28

4.1 博客系统开发环境 28

4.2 博客系统功能模块实现 28

4.2.1 用户管理 29

4.2.2 文章管理 31

4.2.3 评论管理 33

4.2.4 查询管理 33

4.3 本章小结 34

第五章 结 论 35

5.1 论文小结 35

5.2 展望 35

致 谢 37

参考文献 39

# 第一章 绪论

随着互联网的广泛普及和快速发展，Web技术得到了广泛应用。作为互联网上最常见的信息发布和交流工具之一，博客系统在Web技术的推广与应用中发挥了重要作用。通过基于Web的博客系统，用户可以方便地随时随地发布和查看信息，并享受良好的交互性、易用性和方便性等诸多优点。

本文旨在对基于Web的博客系统的现状进行分析，在此基础上，对其设计与实现进行深入研究，探索出一种高效、实用且易用的博客系统。

## 1.1 研究项目背景

随着互联网技术爆炸式的发展，现如今人们的生活中已经与信息化技术紧密结合，从现在的短视频流和文字信息流等现象可以看出，我们已经进入到了Web时代的中后期，正在向着Web3.0的时代迈进。而当我们处于这个时代中，我发现越来多的人希望通过网络获取更多的信息和资源，同时希望能够将自己生活中的体验和感想与他人分享。这就为博客作为信息传递和交流的平台创造了巨大的机遇，也促使了它的飞速发展。

博客系统作为一种广受欢迎的沟通工具，通常具有多种功能，包括文章、图片和评论它们可能还会包含与主题相关的链接和其他内容。使用博客平台，用户可以阅读他人的博客，探索自己感兴趣的信息，也可以自行创建博客，分享自己的经验和见解。写博客是一种表达自我思想、感情、观点的重要途径，同时也是一个提升自身影响力、传播思想的有力工具。人们写博客的动机因人而异，可以是想要表达自己的个人感受和见解，也可以是想要传播自己的技能与知识，以及对于重要议题的评论与看法。此外，在团队、社区中写博客也是一种提供情感支持和合作交流的方式。通过写博客，人们不仅可以记录自己的生活和工作经历，也可以和他人分享自己的经验，并对世界上的重要事件和议题发表自己的观点。用户可以在博客平台上与作者进行互动，包括留言、评论和发表观点，从而实现直接的沟通和交流。这种交流方式不仅方便，而且比传统的电子邮件更加便捷，因此博客系统在家庭、公司和部门之间的信息交流中也越来越受欢迎。博客系统现如今不仅仅是一种信息传递工具，也已经成为社会网络媒介中重要的一部分。

伴随着目前越来越多的人有写博客的需求，有一些人会直接通过购买一些大型博客系统的方式来创建博客平台，如WordPress，来满足写博客的需求；然而对于一些技术熟练的人才说，他们更倾向于通过开源免费的博客系统进行系统的搭建，如VitePress、Hugo、Storybook等。不过尽管这些大型博客系统虽然功能比较丰富，名气也颇高，但实际上，其中很多功能完全不必要，功能些许冗余。相比起轻量化的博客系统来说，这些系统就有些过于笨重了，且在搭建过程中需要复杂的前后端配置，搭建成本也比较高，并不能完全满足每个人的需求。另外，这些系统的主题设计往往单一，交互设计虽华丽炫酷但冗余，不能满足人们的个性化和简约化需求。因此，许多人实际上更希望拥有一个搭建成本低，交互设计简约实用，轻量级，稳定性高等特点等的博客系统。

## 1.2 研究目的与意义

旨在相关调研和项目创作过程中，提升实践能力和创新思维能力。通过对当前人们的需求和对传统博客系统的深入研究，我们有机会拓宽博客系统在互联网应用中的价值，提高博客系统的使用范围。同时，这也可以提高大众对博客的认识，使博客系统的部署变得更加简单方便，与人们的生活更加接轨，让博客系统对实际应用更有意义。

随着技术的进步，拥有自己的博客网站已经成为人们的普遍需求。然而，对于非软件行业的人员，搭建博客系统看起来非常困难。因此，设计一款简易的博客系统，使用户可以轻松搭建自己的博客，显得极具意义。

本文的研究将完全基于Web前端技术的实现，采用了当前最受欢迎的JavaScript库React，用于构建博客系统的用户界面。React的声明式和组件化的设计，使代码可读性更高，易于维护，同时增强了页面组件的复用性。

为了与前端的开发语言保持一致，本文的后端部分选择了Node.js，这是一种当前非常流行的服务器环境，具有活跃的社区和友好的开发环境[1]。

数据库选择了MySQL，这是一款免费开源的数据库，稳定可靠，并且支持Node.js作为服务器语言，非常适合用于存储本系统的数据。

选择Next.js作为整体框架，是因为它在底层整合了React和Node Server框架Express，并且提供了完整的服务端渲染(SSR-Server side Rendering)和路由方案。Next.js拥有一个完善的生态系统，包含丰富的文档、插件和社区支持，使得它非常适合作为该项目的框架。它的生态系统为开发者提供了更多的可能性和方便，从而能够提高项目落地的质量和项目的开发效率。

通过应用流行的技术，不仅可以学习实践先进技术，还能提高系统的技术水平、性能和可靠性。本系统设计了美观的用户界面，操作简单易用，方便用户发布和编辑文章。系统中的文章编辑器功能强大，支持使用简单明了的Markdown语法编辑美观大方的文章，还提供了实时预览的功能，使用者在编写文章时可以实时预览文章的效果。

## 1.3 国内外研究现状

## 1.3.1 博客系统的研究概述

博客一词的历史可以追溯到英语中Web和log两个单词的结合，形成了 Weblog。然而，随着时间的推移，一些人将 Weblog 拆开成 we 和 blog 两个单词，最终 blog 这个词被广泛接受和使用，取代了Weblog，可以翻译为中文的“网络日志” [2]。因此，博客这个词不仅是一种跨越语言的文化传承，更是一种关于网络时代个人表达的独特体现。博客是一种令人兴奋的交流方式，连接着来自不同地区的人们，为他们提供了在网络平台上相互交流和沟通的机会。在博客的诞生之前，主要的网络交流方式是 BBS 和电子邮件，但博客的出现为人们提供了一种全新的途径。人们认为博客具有社区群体的特点，它将网络上的所有网友汇聚在一起，形成了一个更大的整体。因此，博客不仅成为自由媒体中具有强大影响力和吸引力的平台，也是一个汇集多元化声音的舞台[3]。

博客作为一种新型的信息传播方式，在国内外的许多网站都以B/S模式的博客系统为基础，从而使用户能够在不需要下载安装任何软件的情况下，通过浏览器直接使用博客系统。此外，通过博客这一全新的信息表达和沟通途径，可以传递有价值的信息，包括最先进的学科知识、富有创意的思想。

不可否认的是，博客的诞生和繁荣发展，让网络知识的价值得到了真正的体现，这也标志着互联网信息技术的演进已经迈向了更高的阶段。博客，不仅仅是一种新的文化现象，更是标志着人类文明进程中的一次飞跃[4]。

## 1.3.2 React库的研究概述

React，这个为构建用户界面而设计的JavaScript库，在当前的Web开发领域内占有重要的地位。它通过将复杂的用户界面拆分成一个个独立、可重用的组件，使得构建和维护Web应用程序变得更加简单高效[5]。React是由Facebook工程师Jordan Walke于2013年首次推出的，并迅速得到了Web开发者的热爱。几年来，React已成为Web开发领域中最受欢迎的JavaScript库之一，并在各种领域中得到了广泛的应用。

React以模块化和组件化为特色的设计理念，使代码更加清晰易维护，其先进的技术栈兼容各种后端技术，如Node.js、Java、Rust等，拥有丰富的生态系统，可满足各种不同的需求。React的独特方法彻底改变了用户界面的构建方式，成为了Web开发领域的领军者。

而前端中相对React一样火的Vue号称是“构建用户界面的渐进式框架”，不管是React还是Vue，它们的核心都是“构建UI的库”，由基于状态的声明式渲染和组件化的层次架构组成，当应用需要进一步扩展，从简单的页面升级为SPA（Single Page Application，单页面应用）时，需要增加客户端路由方案，比如React中的React-Router，Vue中的Vue-Router等，初了上面之外，还有许多功能是React和Vue本身不包含的，比如构建支持数据流方案、文档工具等。React和Vue本身仅仅是库，而不是框架，我们可以称“包含库本身以及附加功能”的解决方案为框架，例如：Next.js-一款基于React，支持SSR、SSG（Static Site Generation，静态页面构建）的服务端框架[6]。

本文选择基于React的上层框架Next.js作为开发技术选型。React的主要特特性包括：

1. 声明式渲染：使用声明式方式渲染组件，使代码可读性更高，容易维护。
2. 虚拟DOM：使用虚拟DOM来优化页面渲染性能，使得更新页面的速度更快。
3. 组件化：支持组件化开发，使得代码可重用，项目结构更清晰。
4. 单向数据流：采用单向数据流，使得组件间通信更简单。

正是这些特性使得React成为了当前最受欢迎的前端开发框架之一，在许多知名公司和项目中得到了广泛应用，也是我们选择React的主要原因。

## 1.3.3 前端工程化的研究概述

软件工程是一门系统的、科学的方法，它旨在对软件的设计、开发、测试、维护和改进进行全面的管理。它关注于软件的可靠性、可维护性和可扩展性。前端工程化则是软件工程的一个重要分支，它专注于优化和自动化Web前端开发流程[7]。

前端工程化涵盖了软件工程中前端开发过程中的方方面面。Web前端工程化是通过采用工程化的方法来开发Web应用程序的前端部分，以提高开发效率、保证代码质量并方便维护[8]。随着Web前端技术的不断发展，前端工程化已经成为了解决复杂Web应用问题的关键因素。前端工程化的目的是提高Web前端开发的效率和质量，通过提高代码的可维护性和可重用性，以更快、更高效地开发和部署Web应用。

伴随着技术的发展和项目复杂度不断增加，以往前后端一体的开发模式已经不能适应当前的需求。因此，前后端分离成为Web开发中一种模式。

前端主要负责面向用户界面的布局、交互，同时接收后端的数据，然后对在界面上对数据做出相对应的界面变化。

后端主要负责系统的后台功能开发，包括为前端提供数据和对数据的存储和处理、服务的构建和管理等。

在前后端分离模式的初期，前端开发过程中经常出现开发效率低、代码重用性低、项目对低版本浏览器不兼容等问题，于是为了解决这一系列问题，前端工程化便应运而生，并且伴随着Node.js的出现走上了一个新的高度。

前端工程化的主要特性包括自动化构建、代码分离、代码优化、模块化开发、单元测试、代码风格管理等。这些特性的实现通常依赖于一系列工具和技术，如Node.js、Webpack、Gulp、Babel等。而我们在使用的上层框架Next.js中，用到了Webpack做项目打包工作，同时还帮我们在项目开发过程中启动dev server和hot reload功能，达到代码即编即看的效果，极大的提升了开发效率。我们在项目中还使用到了Babel去转换代码，为浏览器做低版本的兼容，让我们的项目支持旧版本浏览器，受众面积更广。

当今的前端世界里，新工具层出不穷，近期还有比较火热的使用rust、go等语言构建的工程化工具，如SWC、ESBuild、Turbopack等，新工具的出现极大的提升了项目打包和编译所需要的时间，极大的推进了Web前端工程化方向快速的发展。

## 1.4 主要研究内容

本文旨在为那些发现其他网站提供的个人博客系统配置繁琐、搭建麻烦、功能复杂的用户提供一种简单易用的解决方案。基于Web前端技术的博客系统完全使用Web前端技术设计和实现了发布、浏览和评论博客的功能，并提供了用户个人信息管理和文章搜索功能。在这个系统中，用户可以使用Markdown语法更高效快捷地书写文章，同时还可以通过标签分类和对文章标题的搜索快速定位博客。管理员用户可以对系统后台数据进行查看和删除，实现对系统整体数据统计展示，并对文章内容进行质量把关和管理。总的来说，本系统让用户能够方便地部署系统，快捷地发布美观大方的文章，同时为浏览者留下实用美观的使用体验。

# 第二章 关键技术分析

本系统采用灵活的B/S架构，即浏览器（Browser）/服务器（Server）。B/S 架构代表着三层客户机/服务器体系结构的现代实现方式，主要包括Web浏览器、Web服务器、数据库服务器。B/S架构的核心是Web 服务器，它不仅负责数据请求、网页生成、数据库访问，还承担了应用程序等执行，使得应用程序以网页的形式展现在Web服务器上，使用户可以通过浏览器无需下载和安装任何软件即可访问到本系统。

本系统前台交互界面使用React进行构建，采用了组件化的设计思想，将可重复使用的部分封装成可重用的组件，以提高代码复用性。同时，样式的渲染使用了CSS，保证界面的美观和统一性。此外，接口的请求使用了Ajax技术，保证用户的操作能够及时的响应。

系统的后端方面使用了高效的Node.js，它支持使用JavaScript语言开发服务器代码。并且使用了基于React的Next.js框架，使用了服务端（Server-side Rendering）渲染技术，即由服务器将用户请求的页面DOM在后端层面组装好后，再返回给浏览器渲染。这种技术特别适合用来做内容资讯类的系统，与本系统非常契合，此外，Next.js还提供了接口的创建能力，方便与MySQL数据库进行连接，提高数据的存取效率。

## 2.1 React前端

React是一种用于构建用户界面的JavaScript库，用实际践行了UI=f(state)的渲染公式，即“框架内部运行机制根据当前状态渲染视图”其在前端开发领域中非常流行。React的核心技术是JavaScript，它采用声明式编程风格，使得开发者能够编写清晰、可维护的代码。React具有以下重要的特点：

## 2.1.1 虚拟DOM

React的一个关键技术是虚拟DOM，它是React的核心机制之一。虚拟DOM使得React可以高效地渲染页面，因为它避免了对浏览器的实际DOM的不必要的操作。此外，虚拟DOM还使得React可以轻松地处理复杂的交互，因为它可以通过Diff算法比较出两个虚拟DOM树前后变化之间的差异来判断该如何更新页[9]。React中的Diff算法是前端框架中的领军代表，现如今的前端框架也都有属于自己框架特性的Diff比较过程，例如Vue，Solid.js等。

## 2.1.2 Hooks

React还包含了一个强大的状态管理机制，即React Hooks，它允许开发者简化状态管理的复杂性，同时保证了代码的可维护性。React Hooks是React库在16.8版本中引入的一种新的编码方式，它以函数式编程的方式代替了以往的面向对象编程方式，彻底践行UI=f(state)的渲染公式。React Hooks的发展过程始于React团队对于面向对象编程在处理组件状态和生命周期时带来的一些限制，因此React团队开始思考如何使用函数式编程来解决这些问题。最终，React Hooks的概念应运而生。React Hooks以函数的方式编写代码，这种方式更容易开发者理解和学习，React Hooks还可以非常轻松与第三方库结合使用，从而使开发者能够轻松地扩展React的功能。

## 2.1.3 JSX

JSX是Meta（原Facebook）提出的一种“类似XML语法”的ECMAScript语法糖，在框架运行时，使用Babel等工具，将其编译成React.createElement代码语句执行。JSX可以非常轻松的描述出“复杂的UI”，并抽离逻辑耦合的代码。这使得高度灵活的JSX语法在React社区早期帮助用户快速实现各种复杂的基础库，丰富社区生态，为现如今的繁荣打下了牢不可破的基础，因为一旦确定技术选型后，中途再切换其他技术栈会付出高昂的成本，正是这一步，推动了更多的开发者参与社区贡献，最终形成源源不断的正反馈，促使React长期占领各大“工程师最愿意使用的前端框架”榜单前列。

## 2.1.4 组件化

React 组件化是一种开发模式，其中把用户界面分解成独立的、可复用的组件。React 组件化开发帮助人们在不同项目中复用组件，还可以帮助开发人员协同开发，因为他们可以共享组件。这种方法有许多优点，其中最主要的是它可以提高开发效率并增加代码的可维护性。

总体而言，React是一种高效、灵活、易于使用的前端开发框架，其关键技术包括虚拟DOM、React Hooks、JSX、组件化等。React的流行程度和其对开发者的支持说明了它对于前端开发领域的重要性。

## 2.2 Node.js后端

Node.js是一个基于Chrome JavaScript Runtime的开源平台，它的发展起源于2009年，由Ryan Dahl开发。它的目的是为了解决服务端程序员的痛点，它将前后端开发的语言统一使用JavaScript，结合了JavaScript的快速、高效的特点，使得Web开发降低了学习成本，开发也更加轻松高效[10]。

Node.js是单线程的，基于事件驱动和异步I/O来处理并发请求，这意味着Node.js不仅可以处理高并发，同时可以保证可靠性[11]。另外Node.js还有着丰富的生态系统，提供了各种各样的包和框架，可以满足各种不同的开发需求。其中最流行的基于Node.js的Web框架非Express.js莫属，Express.js采用洋葱模型的中间件架构，可以快速开发出一个高性能的Web Server，而我们使用的上层框架Next.js中，其API层面也是基于Express.js进行封装的。而Node.js也不仅限于此，Node.js在爬虫和脚本编写领域也取得了非凡的成就，可以说Node.js推动的整个前端的发展。

总的来说，Node.js以其快速、高效、灵活的特点已经成为了Web开发的重要技术，广泛应用于各种Web应用程序的开发。它为开发者提供了一种全新的思路和开发方法，使得Web开发变得更加容易和有趣。

## 2.3 MySQL数据库

MySQL是一个开源的关系型数据库，MySQL的核心技术是基于SQL语言的关系数据库技术，它可以将大量的数据存储在结构化的表格中，并通过索引技术快速查询、更新和维护数据。MySQL也支持多种数据类型，包括字符串、数字、日期和时间等。

MySQL还支持高级数据库管理功能，包括数据备份和恢复、数据完整性检查、自动恢复和约束等。它还支持与其他编程语言和开发平台的集成，使用者可以使用不同的编程语言和开发平台来开发基于MySQL的应用。

MySQL支持多种语言编程接口，如C、C++、Java和Python等，并提供了丰富的数据类型和存储引擎选项，可以满足不同的数据存储需求。它的安全性也得到了保障，可以通过密码保护、权限管理等机制保证数据的安全。

在企业数据管理领域中，MySQL通常作为应用程序的后端数据存储系统使用，它提供了良好的数据查询、数据分析和数据报告等功能，为企业信息管理提供了有力的支持。

我们使用基于Node.js的开源数据持久层框架TypeORM作为操作MySQL数据库的中间层，TypeORM采用了类似于Object-Relational Mapping (ORM)的思想，可以将数据库表映射为JavaScript类，并提供了方便的API来简化对数据库的操作，提高开发效率。

## 2.4 本章小结

本章节旨在对博客系统的核心技术进行简要介绍，包括Web前端库React、服务器开发语言Node.js、以及数据存储方案MySQL。此外，在绪论部分还对Web前端工程化的一些技术和工具进行了简要概述。

# 第三章 博客系统的需求分析与设计

首先，在这一部分中，我们将深入讨论这个博客的需求并进行分析。依据软件工程开发的角度出发，系统被应该被划分为前台功能和后台功能两个部分，以满足不同的使用逻辑需求。系统的前台功能主要是给访客和普通用户使用的，其中包括博客列表、博客文章和博客评论的展示，以及用户信息的修改、文章分类展示、文章搜索、文章的编辑和修改。而后台功能主要负责前台展示内容的管理，主要包括用户管理、文章管理和评论管理三大模块，以及用户数量、文章被浏览数量、评论数量的统计，方便管理人员分析数据。因此，本章将系统划分为用户管理、文章管理、评论管理、查询管理四大模块，分别进行分析与设计。

## 3.1 博客系统需求分析

## 3.1.1 用户管理

本系统一共分为三种用户角色，访客、普通用户和系统管理员。访客即没有进行注册的用户，普通用户拥有对文章的控制权，而每个访客进入系统可以自由的进行文章的浏览和搜索，但访客不能改动文章内容，访客与普通用户功能上最大的差别是无法发布文章，以及无法对文章进行评论。访客可以通过登录进行注册，成为一个普通用户，从而得到文章的控制权，解锁相应的功能。这样做可以防止非法用户修改文章内容，危害文章数据安全[1]。

这里用户注册使用手机号+验证码/第三方OAuth的方式，只要登录系统就表示用户同意注册，保存用户数据到数据库。普通用户包括的主要功能有：登录系统、修改用户个人信息、对个人文章的添加修改和删除、在文章下发布评论、关注文章分类标签、退出登录等功能。根据分析，普通用户的用例图如图3.1所示。

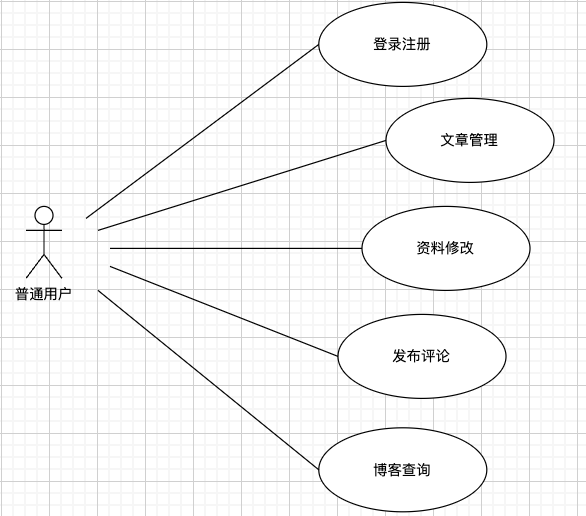


图3.1 普通用户的用例图

系统管理员主要是针对系统所有资料进行管理，所以不需要通过登录注册生成，而是直接通过操作后台数据直接生成管理员账号。管理员在本系统主要负责对文章、用户、评论的管理。系统管理员用户不需要发布文章，修改资料和评论功能，主要是针对系统中所有用户所发布文章进行审核查阅，防止文章内出现不合规的内容。主要操作是删除用户文章和删除用户账号信息。根据分析，管理员用户的用例图如3.2所示。

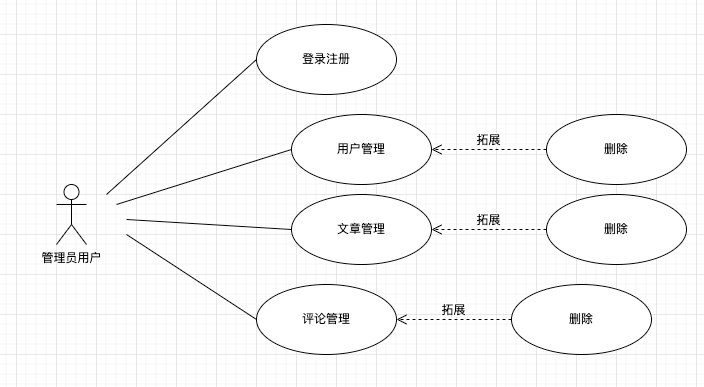


图3.2 管理员用户的用例图

## 3.1.2 文章管理

文章管理可以说是一个博客系统的核心模块了。毕竟每一篇文章都是由用户精心编写的，是用户的心血也是系统的心脏部分。站在用户的角度出发，文章管理模块应该包含四个部分：写文章、编辑文章、删除文章和查看文章。

用户在登录后，首先会进入的是系统的首页，而首页由上方的导航栏和中间的文章列表组成，文章列表相当于一个精美的画廊，用户一走进来，就能预览到由系统中所有用户所发布的文章。而这个精美的画廊在系统中实现的是一个精心设计过的文章列表，列表中的每一项都是一篇文章，可以看到文章的大标题，和部分文章内容，还可以看到文章浏览数量和文章评论数量，方便用户可以快速的了解每一篇文章的信息，排版没关，交互友好，用户可以便捷的上下滑动浏览。当用户想要详细的查看某一篇文章，可以点击列表项进入文章详情页面。

进入文章详情页面后，文章的大标题和文章内容会引入眼帘，文章内容使用Markdown语法编写，然后再呈现到Web端，这样可以让用户的文章不用太担心排版内容乱和文章的视觉效果不好等问题。还能展示代码块和图片等富文本内容。滑动到文章的下面，会看到文章评论区域，可以对文章内容发布自己的见解。

写文章也是博客系统的核心模块，设计出良好的交互体验可以给用户写作时带来不一样的灵感。用户可以登录系统后点击写文章按钮进入文章编辑区域，进行文章书写，文章编辑区域分为左右两栏，左边可以使用Markdown语法进行书写，右边可以对内容进行实时预览，用户体验良好。写作完成后选择文章分类标签再点击发布按钮，文章便发布完成，并且会展示在文章列表内容里。

编辑文章也是针对用户的功能设计，只有用户自己发布过的文章，用户才可以进行修改。在用户登录系统后，进入个人中心里会展示用户发布过的所有文章，然后点击修改，就会进入文章编辑页面。当编辑完成后点击发布便会自动更新。

删除文章和修改文章功能类似，也是针对用户的功能设计，进入到用户的个人中心后，点击文章列表项的删除按钮，便会弹出一个提示框，询问用户是否确认进行删除操作。这样做的目的是为了防止用户进行误操作，意外删除文章。如果点击确定删除，那么该文章便会在数据库里被删除，文章列表项中的文章也会随则消失，不会再展示。

## 3.1.3 评论管理

首先在文章列表中可以看到，该文章共有多少人对此文章进行了评论，然后进入文章详情页面，在详情页的最下方可以看到用户对文章评论的内容。用户可以在评论框内输入内容，然后点击发表评论按钮进行评论。评论内容可以在后台进行删除，访客用户可以看到评论，但是不能进行评论，这是为了保证评论内容的可溯源性，访客若想对文章发表评论需要先登录系统才可以解锁该功能。

## 3.1.4 查询管理

博客查询管理主要包括了标签分类查询以及标题搜索两部分查询。

1. 标签分类查询

在系统的导航栏会有一项关于分类的菜单，通过点击分类菜单可以显示在当前标签分类下的所有文章列表。标签是用户对文章的一种分类方式。在用户发布文章的时候会先选择一个符合文章内容的标签，在首页列表项中也会有文章所属的标签显示，可以根据标签快速分类查询。当然如果这种文章过多，这种分类方式不一样的能帮助用户第一时间准确定位文章，所以还准备了搜索的查询方式。

1. 标题搜索查询

为了方便访客和普通用户更加方便的找到想需要的文章，导航栏也增加了一个搜索框。通过用户在搜索框输入的标题内容，系统可以模糊搜索到标题内容，为了更好的交互，还需要对搜索结果的标题进行高亮色提醒，这样用户会觉得非常醒目，可以帮助用户快速检索文章，提供一个更加便捷的查询文章的方式。

## 3.1.5 文章编辑器

在文章管理模块中，我们对写文章和修改文章都需要对文章内容进行编辑和排版操作，为了更好的用户体验和博客文章的排版便利性以及美观性，所以需要设计和实现一个支持非常流行和易于学习的Markdown语法的文章编辑器。

使用Markdown语法作为文章编辑器，用户只需要用简单的语法便可以专注的书写文章，并且可以把文章渲染成排版良好的HTML页面。其语法的简洁可以替换传统的Word编辑器，由于Markdown输出的是存文本格式内容，因此可以在不同的平台上运行，包括各种主流的操作系统和移动设备上，为博客系统提供更广的受众群体。

参考Word编辑器的设计发现，编辑器在上方会为用户提供一个工具栏，里面包含了一些用户经常使用的功能，例如：字体加粗、删除线、斜体字、大标题、超链接等。所以此系统在设计的时候也为用户提供一个更加方便的操作栏目。用户在进入文本编辑器后，按照文本的布局方式，第一行的左边是的输入框为用户的文章标题，右边分别是发布按钮和文章分类标签的选择框。第二行的左边是文章编辑器的操作工具栏，右边是方便用户切换不同的编辑排版模式，它服务于具有不同编辑习惯的用户，例如用户可以从左编辑右预览模式切换为，无预览模式，或者全屏模式等等。根据分析，Markdown文章编辑器的原型图如图3.3所示。

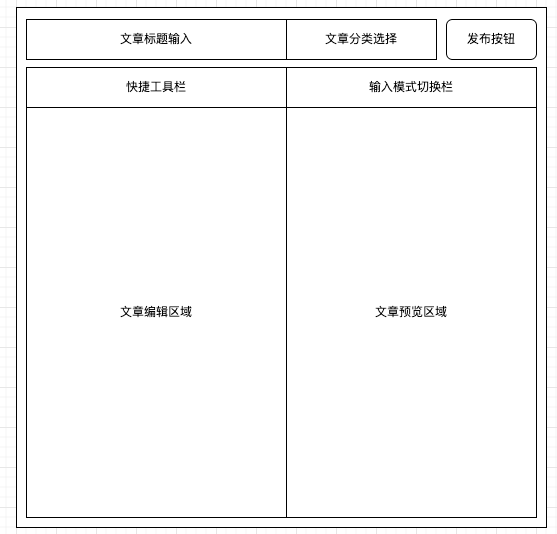


图3.3 Markdown文章编辑器的原型图

## 3.2 博客系统总体设计

从技术分析章节的描述，本博客系统使用Next.js为上层框架，React为渲染层框架，基于Node.js的Express.js为服务器框架，使用MySQL做为数据库，并采用Web前端工程化技术对项目进行构建和实现。

在架构体系上，采用B/S架构，B/S架构是指浏览器/服务器架构，它是一种基于Web的应用程序架构，也是一种现代化的客户端/服务器体系结构。由于浏览器的跨平台性，所以系统可以在任何包含浏览器的客户端运行，提高博客系统的灵活性和可访问性。B/S架构的核心是服务器，整个应用程序都部署在服务器上，采用SSR的渲染方式，数据的请求，网页的生成都在服务器上完成，浏览器接收到HTML字符串后便可以直接使用浏览器渲染。当然Next.js框架不仅支持SSR的渲染方式，还支持CSR的渲染方式，即使用客户端渲染。我们优先使用SSR渲染，因为这样用户的首次渲染速度比CSR方式更加快速，让用户更快的看到内容，带来更好的用户体验。

在使用SSR 渲染过后，针对用户的一下操作，使用ajax发送http请求给服务器，服务器在收到请求后执对应的逻辑然后通过TypeORM执行对应的SQL语句给数据库，然后将数据库返回的数据通过JSON的方式返回给前端发出的请求，前端界面收到相应后将数据绑定到React的hooks内，每次这样状态的改动都会触发React进行重新渲染，React会生成新旧两颗Fiber数，然后Diff算法会计算出两颗树中变化的地方，比如更新和删除操作，分别给Fiber打上不一样的Tag，计算完毕后再根据Tag进行统一处理，更换DOM并渲染到浏览器上。这样整个用户操作到浏览器渲染的流程就执行完毕了。这样通过后端API获取数据，并根据数据触发渲染的模式使得前后端完全分离。根据分析，本系统的架构图如图3.4所示。

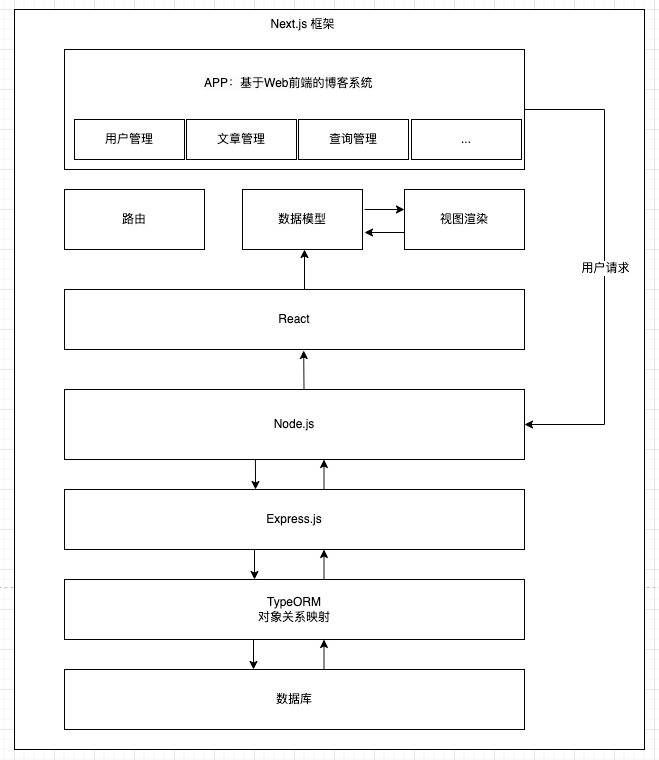


图3.4 系统架构图

我们按照前后端分离的开发模式，采用Web前端工程化的技术对项目进行划分。前端部分主要是对界面的展示和交互操作，后端部分主要是对业务和数据库的处理。这里我们分别讲解一下项目主要的文件结构和功能。

我们的前端和后端共用一个文件夹，但是在pages/api文件下属于Node.js后端部分，这样划分逻辑清晰，开发更加便捷。

1. package.json文件：记录项目所需要的npm包和一些基本信息和一些脚本。
2. pages目录：目录下每个文件就对应着一个页面，包括api目录下面的接口设计保持一致，例如pages/user下的文件是跟用户相关页面，而pages/api/user下的接口是与用户相关的接口。我们保持这样的规范Next.js框架就会保证路由跟页面互相匹配，不需要开发中再写一份单独的路由配置文件。
3. components目录：里面放着一些可以被抽离复用的React组件。
4. store目录：这个目录里存放一些全局可用的数据，一个大型应用往往会有数据需要能够在全局各个地方被访问到，store就是一个这样的作用，而且里面的数据若被修改，也是会让React重新渲染的。
5. db目录：这个里面存放一些跟数据库连接的方法和使用TypeORM与数据库向映射的entity。
6. .env.development文件存放了一些开发时所需要的环境变量，比如MySQL的配置信息等。

## 3.3 博客系统模块设计

针对博客系统实现的功能，如写文章、评论、浏览等进行分析和设计，便形成里系统等需求分析，根据对系统等需求分析以及系统的架构设计，可以对系统进行功能模块划分。主要包括用户管理、文章管理、评论管理和查询管理四个功能模块。下图展示了基于Web前端的博客系统的详细功能结构图，如图3.5所示。

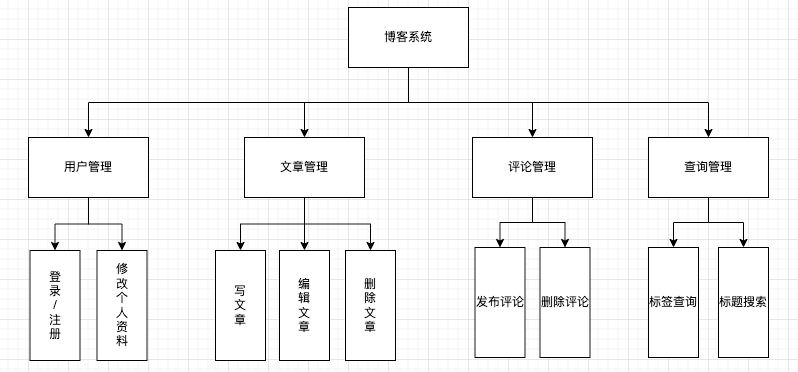


图3.5 系统功能结构图

## 3.3.1 用户管理

用户管理主要是针对所有用户信息的管理，主要是对普通用户信息的管理。使用者若没有进行登录就是访客身份，只能使用最基本的功能，例如查看和查询管理模块的功能。当访客登录后，若是第一次登录，便会自动注册成一个用户，可以使用修改个人资料、评论功能，可以发布编辑自己的博客文章。登录后系统管理员可以对所有用户的信息进行管理。用户管理的功能设计如图3.6所示。

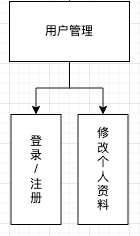


图3.6 用户管理的功能设计图

分析后得出用户管理主要包括登录/注册、修改个人资料和退出系统三个主要功能。登录/注册功能，我们为了符合现代人们的生活方式，不采用传统的账号密码登录，因为现在人们使用各个网站都需要独立的账号和密码，记忆起来十分繁琐和苦恼，而手机号+验证码的方式则更加现代化，同时考虑到手机手机不在身边的时刻，我们也提供了OAuth的登录方式，本系统选用GitHub平台的OAuth进行接入。采用这两种方式已经满足了绝大使用场景，当访客第一次登录系统，便代表了注册，会默认分配一个头像和昵称，若使用OAuth的方式，则继续使用之前平台的账号信息。在登录的时候会查询数据库是否在用户列表中，若不在会保留用户信息到数据库，相当于自动注册成立用户，以后每次登录便会直接使用原来注册用户的信息进行登录。

## 3.3.2 文章管理

文章管理模块主要包括文章列表的展示，写文章，编辑文章和删除文章。其中文章列表展示无论是访客还是用户都可以进行查看。只有登录用户才可以写文章、编辑文章、删除文章，当然仅限于属于自己的文章。浏览其他用户时，并不能对其有任何修改操作。

根据单一原则，我们可以把文章列表项抽离成一个具有单独职责的组件，这样无论是在哪里展示，都可以直接复用，包括写文章和编辑文章也可以进行组件化的封装。文章管理的功能设计如图3.7所示。

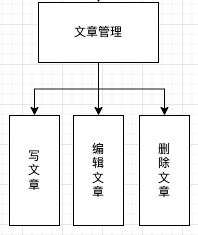


图3.7 系统功能结构图

文章展示的内容是对Markdown语法解析之后渲染成HTML的样式，对写文章和编辑文章进行组件化封装后也是共用同一个组件。因为React本身并没有封装好的React编辑器组件，所以这里单独封装了一个简易的编辑器组件，在写文章和编辑文章的时候重复使用。其中无论是写文章还是编辑文章，它们的发布流程都是一样的，都需要填写标题、选择分类标签、填写文章内容后才可以发布，用户发布文章的流程图如3.8所示。

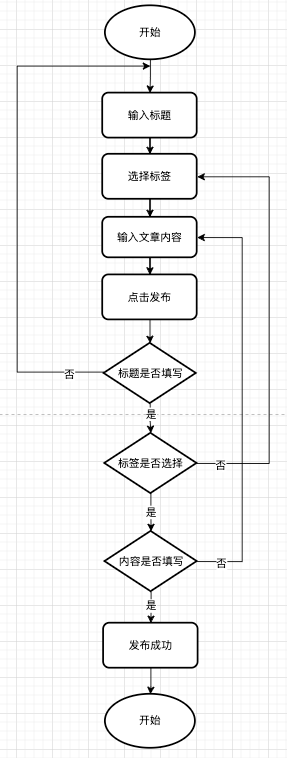


图3.8 发布文章流程图

## 3.3.3 评论管理

评论也是一个相对来说重要的模块。经过分析发现，评论跟文章的展示其实是深度绑定的，用户需要到文章下面才能进行评论以及对评论的查看，用户在首页进行文章浏览的时候也可以通过文章列表卡片上看到评论的数量。访客在首页进行浏览的时候是没有权限对文章进行评论的，对于评论模块来说，也就划分了两种查看状态。评论模块主要是对用户头像和评论内容的展示，为了防止用户发表一些违法乱纪的评论，系统需要在用户发布评论的时候对评论内容做一些关键字的过滤。为了防止用户通过评论做一些脚本注入之类的用户，系统也需要对内容中的一些字符进行转义，例如将`<`转义成"&lt;"等等，这样可以保证系统的安全性和稳定性，同时也保证了系统内容的质量。根据上面的分析，我们将评论的功能进行组件化封装，然后与文章展示功能相结合，使得文章和评论合二为一，API数据层面也可以直接将文章内容和对应的评论合在一起返回，使得两者相互关联对应。

## 3.3.4 查询管理

查询管理主要分为两个子模块分别是分类查询和标题查询，主要是为了帮助用户对系统已有文章的检索，当系统有了大量的文章时，若用户想回顾查看某一篇文章，可以通过分类查询和标题查询去快速定位文章。查询管理模块不设计权限，访客和用户都可以正常使用。

分类查询和标题查询都在系统的导航栏上。每篇文章在编写和编辑的时候都必须要选择一个或多个相对应的标签，相当于对文章的一个分类，分类查询就是通过标签帮助用户快速筛选出要查找的文章。用户点击分类下的一个标签后，文章列表展示的都是这个分类标签下的所有文章。

而文章查询是根据文章的标题进行模糊查询，因为文章的标题也是用户在编写时必须填写的内容，所以可以用标题来做查询依据。为了更好的用户体验，系统对通过模糊匹配搜索出的文章结果，把文章列表的标题做关键字高亮，这样可以让用户快速看到所需要的文章。

## 3.4 博客系统数据库设计

根据以上分析可以了解到本系统主要包括用户管理、文章管理、查询管理、评论管理四大模块，根据功能划分可以得出数据库主要存储用户信息、文章信息、标签信息、评论信息。根据这些存储信息设计数据库中的表信息。系统数据库的E-R图如图3.9所示。

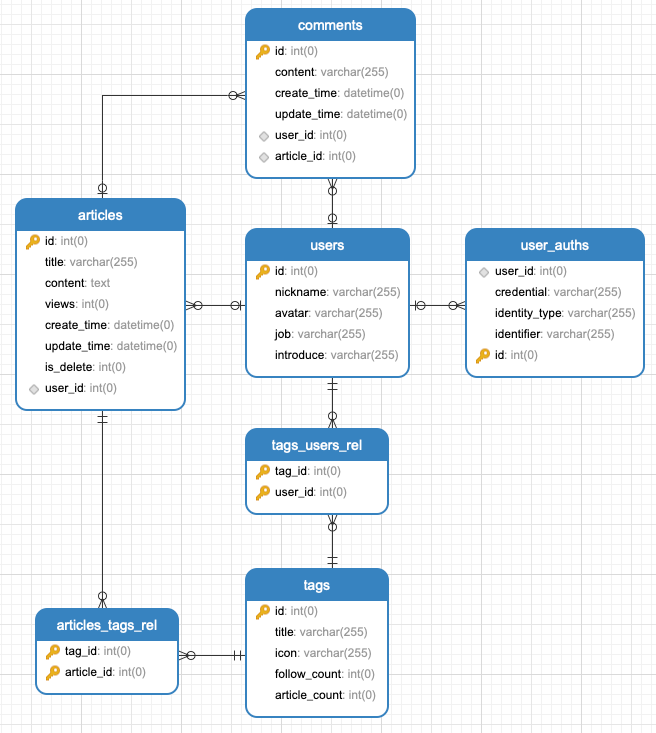


图3.9 博客系统数据库E-R图

系统中每个表的详细说明如表3.1所示。

表3.1 数据库表设计说明

|  |  |  |
| --- | --- | --- |
| 数据库 | 中文名 | 说明 |
| users | 用户表 | 存储用户的基本信息，昵称、头像、职业、个人介绍 |
| user-auths | 用户授权表 | 存储用户的授权信息，身份类型、身份标识、登录凭据、关联的用户id |
| articles | 文章表 | 存储文章信息，文章标题、文章内容、浏览次数、创建时间、更新时间、删除标记、关联的用户id |
| comments | 评论表 | 存储评论信息，评论内容、创建时间、更新时间、关联的用户id、关联的文章id |
| tags | 标签表 | 存储标签信息，标签标题、标签图标、关注统计、文章统计 |
| tags\_users\_rel | 标签用户中间表 | 存储标签和用户对应的中间表 |
| articles\_tags\_rel | 文章标签中间表 | 存储文章和标签的中间表 |

系统中每个数据表的详细说明如下

1. users表

用户表在db/entity文件夹下的user.ts文件中进行配置，详细的字段配置如表3.2所示。

表3.2 users表

|  |  |  |
| --- | --- | --- |
| 字段名称 | 数据类型 | 说明 |
| id | int | 用户id |
| nickname | varchar | 用户昵称 |
| avatar | varchar | 用户头像 |
| job | varchar | 用户职业 |
| introduce | varchar | 个人介绍 |

1. user-auths表

用户授权表在db/entity文件夹下的userAuth.ts文件中进行配置，详细的字段配置如表3.3所示。

表3.3 user-auths表

|  |  |  |
| --- | --- | --- |
| 字段名称 | 数据类型 | 说明 |
| id | int | 用户授权id |
| identity\_type | varchar | 身份类型 |
| identifier | varchar | 身份标识 |
| credential | varchar | 登录凭据 |
| user\_id | int | 关联的用户id |

1. articles表

文章表在db/entity文件夹下的article.ts文件中进行配置，详细的字段配置如表3.4所示。

表3.4 articles表

|  |  |  |
| --- | --- | --- |
| 字段名称 | 数据类型 | 说明 |
| id | int | 文章id |
| title | varchar | 文章标题 |
| content | text | 文章内容 |
| views | int | 浏览次数 |
| create\_time | datetime | 创建时间 |
| update\_time | datetime | 更新时间 |
| is\_delete | int | 删除标记 |
| user\_id | int | 关联的用户id |

1. comments 表

评论表在db/entity文件夹下的comment.ts文件中进行配置，详细的字段配置如表3.5所示。

表3.5 comments表

|  |  |  |
| --- | --- | --- |
| 字段名称 | 数据类型 | 说明 |
| id | int | 评论id |
| content | varchar | 评论内容 |
| create\_time | datetime | 创建时间 |
| update\_time | datetime | 更新时间 |
| user\_id | int | 关联的用户id |
| article\_id | int | 关联的文章id |

1. tags表

标签表在db/entity文件夹下的tag.ts文件中进行配置，详细的字段配置如表3.6所示。

表3.6 tags表

|  |  |  |
| --- | --- | --- |
| 字段名称 | 数据类型 | 说明 |
| id | int | 标签id |
| title | varchar | 标签标题 |
| icon | varchar | 标签图标 |
| follow\_count | int | 关注统计 |
| article\_count | int | 文章统计 |

1. tags\_users\_rel表

标签用户中间表关联着和用户多对多的关系，由TypeORM自动创建生成，详细的字段配置如表3.7所示。

表3.7 tags\_users\_rel表

|  |  |  |
| --- | --- | --- |
| 字段名称 | 数据类型 | 说明 |
| tag\_id | int | 标签id |
| user\_id | int | 用户id |

1. articles\_tags\_rel表

文章标签中间关联着文章与标签之间多对多的关系，由TypeORM自动创建生成，详细的字段配置如表3.8所示。

表3.8 articles\_tags\_rel表

|  |  |  |
| --- | --- | --- |
| 字段名称 | 数据类型 | 说明 |
| tag\_id | int | 标签id |
| article\_id | int | 文章id |

## 3.5 本章小节

本章节主要介绍了博客系统的需求分析与设计。首先对本系统从软件工程的角度出发针对需求进行简要的分析，并对系统的总体设计做出了分析为接下来的模块设计奠定了基础。在模块分析部分主要针对用户管理、文章管理、评论管理和查询管理四个模块进行了详细的说明。最后对本系统的数据库设计进行了简要的分析说明。

# 第四章 博客系统的功能实现

## 4.1 博客系统开发环境

系统基于Next.js框架进行开发，Next.js主要是对React和Express.js的上层封装，并且提供了完整的Web工程化项目，提供了非常丰富的渲染方式。项目的版本管理使用了Git，具体的开发环境和工具如下：

处理器：Apple M1

内存：16G

操作系统：macOS

开发框架：React 18.2.0、Next.js 12.2.5

开发语言：HTML、CSS、JavaScript

开发工具：Visual Studio Code

开发环境：Node.js 16.12.0

版本控制工具：Git 2.37.1

数据库：MySQL 8.0.29

## 4.2 博客系统功能模块实现

本系统采用了B/S架构，并使用了前后端分离的开发模式，前端主要负责界面的渲染，后端主要负责数据的逻辑，双方通过API接口进行数据传递。前端通过接口获取到数据后使用React中MVVM的分层设计，用数据驱动页面，提升开发效率[12]。本系统通过Next.js提供的CLI工具创建集成了工程化相关的配置，做到了开箱即用。项目目录结构如图4.1所示。

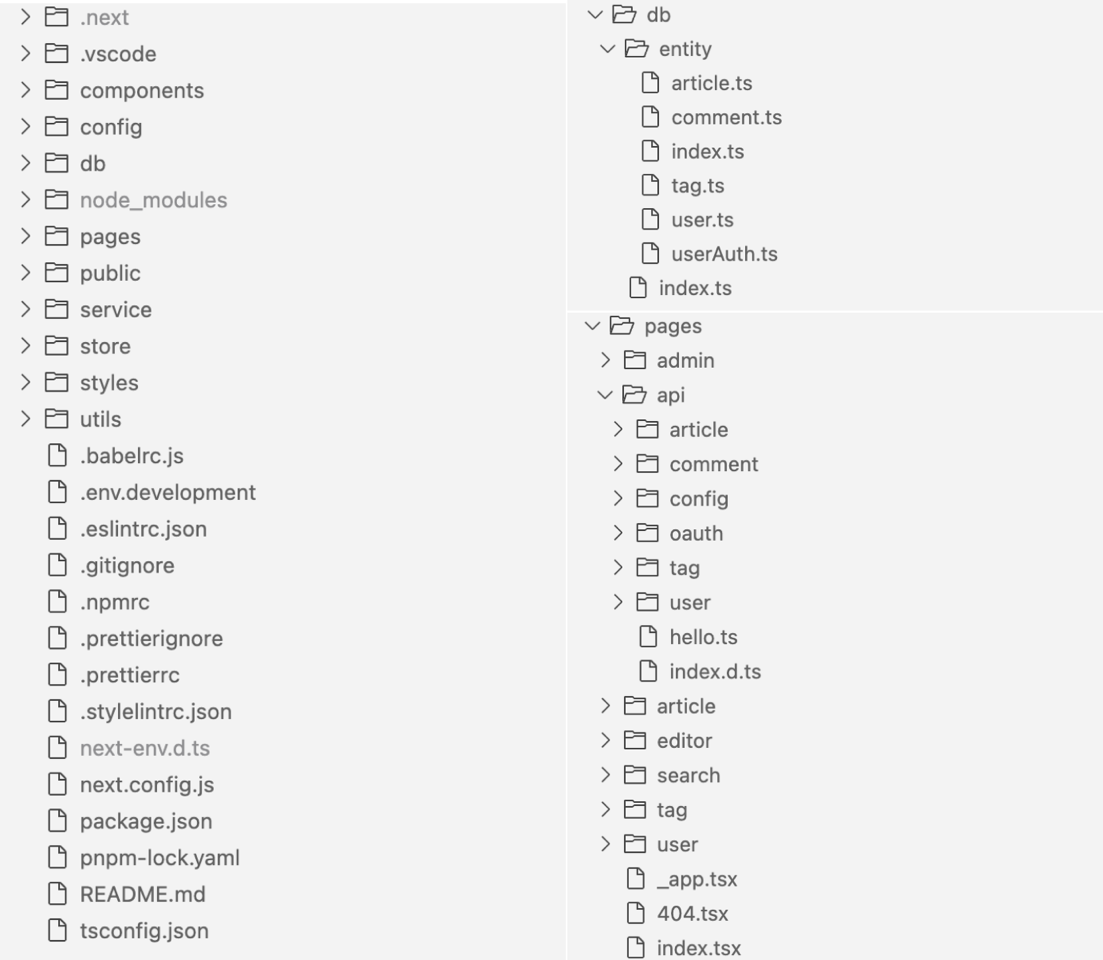


图4.1 项目目录结构图

项目整体目录结构如左侧图所示，其中定义数据库模型在db文件夹下，前端页面相关的逻辑都在pages下，后端所有相关API接口都在pages/api文件夹下。前后端的路由都是用Next.js框架根据目录自动生成，无需再单独配置前后端路由。

## 4.2.1 用户管理

用户管理在前台主要包括登录注册和修改个人资料，后台部分主要是管理员用户对用户的删除操作。项目封装的组件都放在了components文件下，其中包括登录注册和修改个人资料。

登录注册组件主要是用与提交用户手机号和验证码和接入GitHub的OAuth认证。用户若选择手机号登录，在输入手机号后，点击获取验证码按钮，然后会进入倒计时阶段，确保验证码的实效性。这个阶段会触发前端界面的handleGetVerifyCode方法，这个方法首先会检查手机号的有效性，防止向服务端提供无效的信息，造成短信服务的资源浪费。手机号检验通过过，便会调用通过API把手机号提交给后端，后端在拿到手机号后会去调用短信服务运营商提供的API向用户发送短信验证码，在调用成功后，运营商的API会返回验证码给后端，后端在收到后，会把验证码存入到Session中，作为后续用户的登录做准备，然后用户在这个时候手机会收到验证码。

待用户填写验证码后点击登录再次提交信息，后端接受到信息后首先对Session中的验证码做校验，查看是否匹配，若匹配失败则提示用户验证码错误。若匹配成功，则根据用户的手机号去查找用户授权表中是否已经存在了，若不存在则把用户的手机号和验证方式保持进数据库，为用户注册数据。若已经存在则用户表中所关联的用户信息，并给浏览器发生一个cookies作为登录凭证，并提示登录成功，到这里手机验证的方式已经整体走通了。而使用OAuth的方式也类似，需要先去对应的平台申请接入，然后再去调用API，在第三方平台对用的校验通过后会将该平台对用户信息返回给你，这个时候就可以把用户信息保存进数据库，创建一个用户，若该用户已经注册过，则无需注册直接返用户信息，提示登录成功，跟手机号+验证码的登录注册逻辑相似。而在用户点击头像下面的退出系统时，会清楚浏览器的cookie方便，恢复系统到未登录状态方便重新登录注册用户。登录界面如图4.2所示。



图4.2 登录界面图

修改个人资料组件，则是在用户登录成功后，进入个人中心点击修改资料按钮进行信息数据的提交，然后发送请求给后端，后端再接收到修改请求后会直接从Cookie中读取用户的信息，根据此信息先去从数据库中查找用户信息，然后对用户信息进行数据的修改，修改完成后再保存进数据库，然后返回修改成功，返回修改后的数据。

## 4.2.2 文章管理

文章管理主要包括文章列表的展示、写文章、编辑文章和删除文章功能。在后台管理中，系统管理员拥有所有文章的删除权限，在前台系统中，用户只拥有删除自身创建的文件的权限。文章列表中的每一个卡片项包含了文章的基础信息，文章标题、最近修改时间、所属标签、文章部分内容、作者头像、浏览次数以及评论数量。文章列表按照创建日期进行倒叙排列。文章列表如图4.3所示。



图4.3 文章列表图

文章详情的查看使用的最上方是标题，下方紧接着就是头像然后就是文章内容。文章的正文内容正是通过编辑器进行编辑和排版的展示。因为Markdown语法普通的文本标记语言，所以并不能直接转换成对应的HTML标签，需要使用Markdown的语法解析器，把Markdown语法进行语法解析翻译成对应的HTML文章再交给浏览器进行渲染，还引入了代码高亮的CSS插件美化相应的代码块样式和标签样式[13]。高亮插件可以针对不同的编程语言渲染出不同样式的高亮显示，对使用者更加友好。文章详情展示如图4.4所示。



图4.4 文章详情展示图

当用户访问自己的文章时，会出现如图4.4所示的编辑按钮，点击后会进入编辑器，可以对文章进行修改更新。布局可以通过右侧的工具栏进行合适的编写方式，文章可以在左侧使用Markdown语法进行书写，生成的文章内容可以在右侧实时预览到，用户体验十分友好。用户在首次写文章的时候也是使用这个编辑器，若为第一次写内容，会默认给用户生成部分默认内容，这样不至于让用户一打开就看到空白页面，交互更加友好。用户在个人中心界面可以对自己的文章进行删除，与管理员的删除相似，前端提供文章id到后端，后端先根据文章id对数据库中的内容进行删除更新操作，然后提示用户删除功能，并删除界面中的文章列表。Markdown编辑器如图4.5所示。



图4.5 Markdown编辑器图

## 4.2.3 评论管理

评论管理主要是在给文章下方展示用户的评论信息。编写评论模块和评论展示列表分别时两个功能单一的组件，然后将其组合到一起，封装成一个功能更完整的评论组件，在用户发布评论后为了保证评论信息的时效性会向后端发起请求以此来获取用户最新的评论数据。其中某一篇文章下方的评论如图4.6所示。

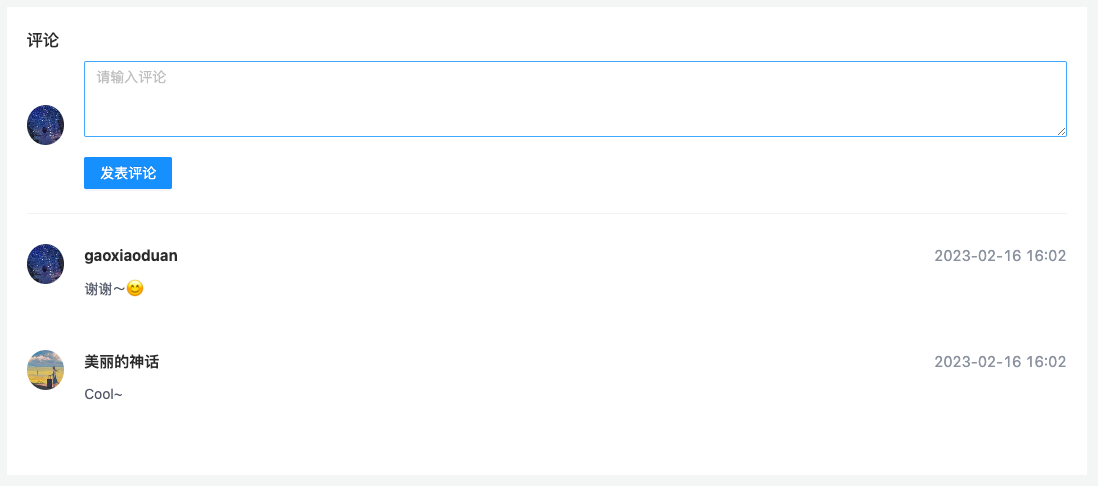


图4.6 评论模块视图

## 4.2.4 查询管理

查询管理主要是为了帮助用户快速定位文章内容，方便用户查找回顾文章。本系统分别提供了分类查询和标题搜索查询两种方式。分类查询是根据文章的分类标签进行文章筛选，然后将筛选结果返回给分类展示列表，分类查询内容如图4.7所示。



图4.7 分类查询视图

而标题搜索查询是根据用户输入的关键字对文章表中的标题进行模糊查找，然后返回查找的数据给文章列表组件进行渲染，这里对结果的关键字标题做实现了一个高亮组件，可以帮助用户快速定位，提升用户的使用体验。

## 4.3 本章小结

本章节阐述了本系统功能的实现，介绍了本系统开发所需要的环境，重点介绍了用户管理、文章管理、评论管理、查询管理四个模块的实现，通过不断的编写和调试代码最终使博客系统的需求和设计完整落地。

# 第五章 结 论

## 5.1 论文小结

本论主要讲述了基于Web的博客系统的设计与实现过程。至此，本系统全部基于Web前端技术实现。系统前端部分使用React去渲染界面，服务端使用Node.js开发，业务逻辑按照模块化划分，页面基于组件化搭建。本文首先对系统的背景情况和实现意义进行调研，并对比国内外现状进行分析，然后对关键的实现技术进行项目可行性分析与介绍。紧接着本文着手对需求进行分析，做出系统整体架构设计，并为功能模块进行划分，接着就经过不断的开发和调试初步实现了此系统。

在实现过程中后端使用了REST风格的接口设计，并对数据库表进行了实体抽象，这样可以极大的方便开发人员去操作数据库，通过模型建设封装，减少直接使用SQL语句操作数据库，可以提高开发效率。而前端部分首先设计出一个简约风格的布局和样式，在文章编写模块使用简单易用的Markdown语法做到一边写一边预览的用户体验。

综上所述，本文按照软件工程的开发要求，实现了包含用户管理、文章管理、评论管理、查询管理四个功能模块，让理论与实践互相结合。

## 5.2 展望

本系统应用了当前最流行的Web前端技术栈实现，以此将技术进行实践提升系统的性能和稳定性，但是如今的技术发展突飞猛进，以后还有涌现出更加迅速的技术，希望该博客系统可以跟技术的发展，方便技术的升级更新。由于时间问题以及种种因素，本系统还存在一些需要优化的地方，希望以后继续投入时间与实践来完善该系统，让用户体验越来越好。

# 致 谢

时间如梭，自从我高中起就对计算机技术产生了浓厚的兴趣。大专毕业后，我便一心投入计算机行业，想要一直深耕这个领域。在作为程序员的这些日子里，我深感自己的理论基础薄弱。因此，我选择来到西安电子科技大学网络学院选修此专业。通过这次论文的编写，我意识到作为一名毕业生，我们不仅需要敲代码写程序，还需要熟练掌握基础知识，并将其应用于项目实践中。在以后的工作和学习中，我会不断要求自己，终身学习，为社会做出自己的贡献。我希望能够为母校争光，展示母校的优秀品质。

最后，我要感谢所有的答辩老师。您们辛苦了！感谢您们抽出时间评阅论文，并给予耐心指导。在此，我向您们表示由衷的感谢！

# 参考文献

1. 闵亮,薛格格,张玉欣,赵彩.基于Node.JS博客系统的设计与实现[J].电子设计工程,2022,30(07):37-41.DOI:10.14022/j.issn1674-6236.2022.07.008.
2. 韦耿,钟亮.基于JSP的个人博客系统[J].电脑知识与技术,2013,9(21):4865-4869+4876.
3. 谭晶晶,谭丹丹.基于ASP的个人博客系统设计与实现[J].软件导刊,2013,12(05):82-84.
4. 战毅. 从UEditor谈Web编辑器技术[J]. 程序员, 2013(5):4.
5. 张丰麒, 王飞. ReactJS的新特性在Web开发中的应用[J]. 移动信息, 2015(10):2.
6. 卡颂. React设计原理[M]. 2022年12月第1版. 电子工业出版社, 2022.
7. 王宇翔.Web前端开发技术以及优化方向分析[J].电子技术与软件工程,2022(12):42-45.
8. 张志飞.前端工程化的研究与实践[J].电脑知识与技术,2016,12(25):224-226.DOI:10.14004/j.cnki.ckt.2016.3269.
9. 曹帅. 基于Diff算法的Web前端性能优化及评估[D].北京邮电大学,2019.
10. 陆凌牛. Node.js权威指南[M]. 机械工业出版社, 2014.
11. 钟强.Node.JS平台下Web前端架构的研究[J].无线互联科技,2013(12):124+196.
12. Hall G . Pro WPF and Silverlight MVVM : effective application development with Model-View-ViewModel[M]. Apress, 2010.
13. DAVID FLANAGAN. JavaScript权威指南: 第5版[M]. 机械工业出版社, 2007.