河南科技学院
2019 届本科毕业论文（设计）

基于 vue 的个人博客的设计与实现

学生姓名：周会艳

所在院系：信息工程学院

所学专业：计算机科学与技术

导师姓名：郑颖

完成时间：2019-00-00

基于 vue 的个人博客的设计与实现

摘要

互联网成为了当今世界人们对外交流，快速获取、发布和传递信息的最常用也是最便捷的渠道，在人们的日常生活和工作学习的各方面互联网技术都在发挥着不可或缺的作用。所以顺应时代出现了博客。博客网站正好满足了互联网中个人的共享需求，在博客中互相交流技术发展，共同学习，共同进步。本文对博客的功能与需求进行了完整分析，设计出了一个简单、易用的个人博客系统。 为了提高开发效率和代码复用率，本系统使用采用 SPA（单页面应用）思想的 vue.js 进行系统的开发。Vue.js 是一个轻量级的基于 MVVM 模式的渐进式框架。 vue 的组件化与数据绑定的思想，在很大程度上简化了前端开发的复杂度。 后端将 node.js 技术与 express 结合使用，以创建符合 RESTful API 设计规范的接口。

关键词:博客,vue.js,,node.js ,API,vuex

DESIGN AND IMPLEMENTATION OF PERSONAL BLOG BASED ON VUE

Abstract

The Internet has become the most common and convenient channel for people in the world to communicate, quickly acquire, publish and transmit information. Internet technology plays an indispensable role in people's daily life and work and study. So in the era of the emergence of a blog. The blog website just meets the sharing needs of individuals in the Internet, and exchanges technology development, learning together, and making progress together in the blog. This article provides a complete analysis of the functions and needs of blogs and designs a simple, easy-to-use personal blogging system. In order to improve development efficiency and code reuse rate, the system uses vue.js using the SPA (single page application) idea for system development. Vue.js is a lightweight, progressive framework based on the MVVM pattern. Vue's idea of componentization and data binding greatly simplifies the complexity of front-end development.The backend uses node.js technology in conjunction with express to create interfaces that conform to the RESTful API design specification.

Keywords:Blog,Vue.js,Node.js,API,Vuex

目录

第一章 绪论

1.1 博客的背景

互联网对人们的生活、工作和学习方式进行了新的定义。Blog 是互联网 web2.0 的产物，是跟随邮箱（Email），论坛（BBS）和 ICQ 之后出现的第四种网络通信方式，是以互联网作为基础，使得人们可以快速，方便的分享个人想法并与他人建立沟通的综合的可以展示自己个人特性的的平台。本系统就是按用户划分的个人博客。博客有以下特点：

（1）简单的操作是博客开发的动力。这是博客受到如此众多网友欢迎的最大特点。许多博客托管网站使用的口号就是“在一分钟内轻松拥有一个博客”。

（2）不断更新是博客壮大的催化剂。现代社会，信息传递的速度非常快速，博客的更新就如逆水行舟，在时代的洪流中没有原地不动，只用前进与退步，不及时更新的博客很快就会被淹没在技术前进的道路上。

（3）开放互动是博客传播的推动剂。互联网为博客提供了开放性，因此博客的空间界定变得模糊，不在只是一个简单的私有空间。访客和其他博主写给文章的留言或评论，如果我们回复并通过访客和其他博主留下的地址去返回给他们访问的结果，这就可以实现互动效果。因此，只要善于利用博客开放互动的特点，就可以将博客用于交流和推广，使之最终形成一个固定的圈子。

（4）展示个性是博客出彩的原动力。作为博主可以在博客的界面设计，功能搭配与文章喜好上展现出自己的独有的个人性格特点，吸引同类或者欣赏此类性格的访问者。

1.2 系统设计的意义

在科技快速进步的今天，互联网成为了当今世界人们对外交流，快速获取、发布和传递信息的最常用也是最便捷的渠道，互联网如今已经渗入人们的日常生活中方方面面。当今时代，技术日新月异，作为互联网技术从业者，需要不断的汲取新知识，走在技术的前沿，但是大部分新技术文档与书籍都是英文，对于英语能力有限的互联网技术从业者，不可避免的需要借助搜素引擎或者其他人的帮助，在搜索引擎的搜索结果中大部分的技术问题解决方案的贡献者都来自博客，所以此次毕业设计自己也想要做一个记录自己学习与工作过程的心得体会的博客网站，让自己对社会和他人可以有所贡献。

第二章 系统相关技术介绍

2.1 SPA（Single Page App）

2.1.1 单页应用

单页应用（英语：Single Page Application， SPA）是通过动态重写当前页面与用户交互的 Web 应用程序或网站的模型。避免了传统网页模型的打断用户体验的页面切换方式，也避免了不断的向服务器发送请求去加载整个新页面，减轻了服务器的负担。在单页应用中，通过加载该 html 时来检索所有必需的代码（HTML，JavaScript 和 CSS）。 或者根据需要动态加载适当的资源并通过路由程序将它们添加到页面中（通常是响应用户操作）。网站所有的页面内容都包含在这个主页面中。但是在实际开发的时候，还是会分开写（页面片段）。SPA 支持多种多样的客户端功能，很少需要重新加载整个页面。因此当用户执行操作或在网站的不同页面之间跳转时，不必重新加载整个页面。另外单页应用它在后台提取数据，对单个用户操作的响应更快。

2.1.2 与传统网页的比较

为了更好的理解什么是单页应用，我们先来了解传统的网页应用。传统网页应用的工作模式如下图：

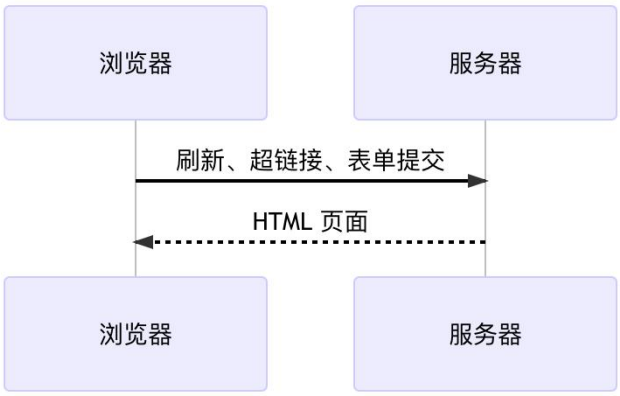

图 2-1 传统网页应用的工作模式

此种方式作为网页应用的传统形式长久不衰，很多流行的开发框架都以之作为范式设计的。比如 Ruby on Rails，Spring MVC，Express 等等。在传统的网页应用中页面跳转控制与数据计算等都放在服务器端，而作为前端展示的用户界面则通过网络发送到浏览器端，作为与用户交互的入口。

这样的范式有以下特点：

服务端负担过重，由于 MVC 存在于服务器上，因此开发资源的重点和此类应用程序的开发偏向于后端，通常这时会由后端工程师来领导整个项目的开发；

页面刷新频率过快，导致资源的浪费。当网页的功能发生变化时，页面需要根据变化进行刷新，而等待页面完成刷新则需要花费多出用户预计之外的时间，

致使用户对页面的体验非常不佳。

单页应用与传统网页应用的最大的不同就是将 MVC 前置到了浏览器端：

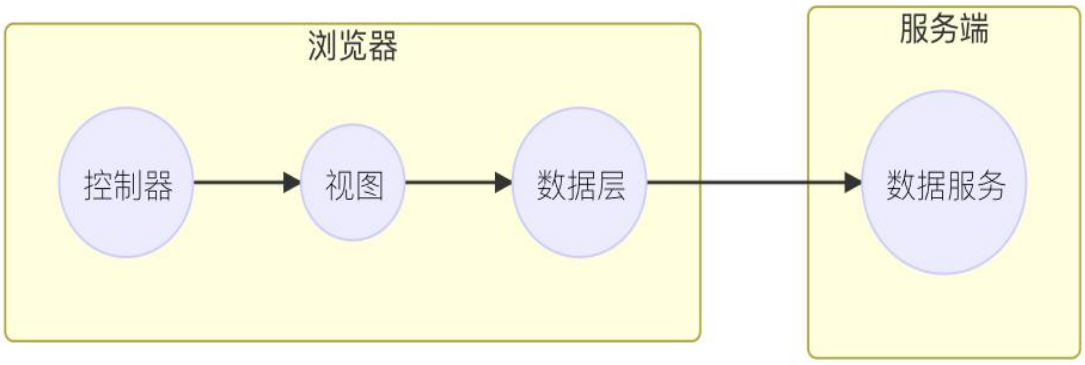

图 2-2 单页应用的工作分布

单页应用将实现页面跳转的路由功能的处理放在浏览器端，即直接响应浏览器端的浏览地址的变化。然后会将变化通知到与页面对应的路由，以此来向用户呈现对应的界面。小组件为功能元件。单页应用以小的组件作为功能元件，在路由变化时，不再刷新整个页面，而是将这些小组件进行组合以实现页面的改变。数据层前置。以 JSON 的形式发送应用程序数据的方式在 HTML 的 view 层和应用程序层之间创建了一个分隔。这便使得表示层与应用层相分离，浏览器端形成一层实实在在的数据层，而服务端则退化成了完全的数据 API，从而方便使用相同数据但要求实现不同功能的开发人员去独立地开发每一个页面。

2.1.3 单页应用的好与坏

每一种存在的技术都有自己的优点和缺点，单页应用也不例外。

单页面开发的优点：

（1）良好的用户体验：用户不再需要等待刷新页面，大大减少了 HTTP 请求造成的时间损失。

（2）前后端分离：前端负责界面显示，后端负责数据存储和逻辑计算，前后端的业务与数据逻辑更加清晰明了。

（3）减轻服务端压力： 在单叶应用中服务器只需要提供数据的 API 接口，不需要知道前端的代码实现，服务器不再承担页面逻辑和页面拼接，并且可以最大化服务器的性能。

（4）共享一组后端程序代码：同一组后端程序代码，可以在不经修改的情况下就可以适用于三端 Web、手机、平板。

（5）组件共享：对于使用到同样功能或者同样展示型组件的多端应用可以

在项目中开辟公共组件的存储文件，将所有的公共组件都放入其中，以供多端使用，简化了代码的重复性。

单页面开发的缺点：

（1）首屏加载过慢：当第一次加载单页面时，需要组合所有页面所依赖的 css 和 js 并统一加载它们。因此导致首页需要加载的 css 与 js 文件较大，会在一定程度上影响页面加载时间。

（2）SEO：由于页面数据都是前端异步加载出来的，浏览器看到的是所有结构加载出来之前的页面，找不到 meat 标签或者任何可用的文字，不利于搜索引擎的抓取。

2.3 vue.js

2.3.1 介绍

Vue.js 是尤玉溪编写的用来构建用户界面的渐进式 JavaScript 框架 。与其它大型框架不同的是，Vue 被设计为可以自下而上应用。Vue.js 的核心库只关注视图层，这种结构使其不仅易于上手，而且还便于与第三方库或现有项目整合。另一方面，当 Vue.js 与现代化工具链和各种支持的类库结合使用时可以为构建一个复杂的单页应用。

2.3.2 组件化应用构建

组件系统是 Vue 的重要概念，它是一种抽象的概念，此设计允许我们使用小型的、独立的和通常可复用的组件通过组合去构建大型应用。在 WEB 网页制作中几乎任意类型的应用界面都可以抽象为一个组件树：

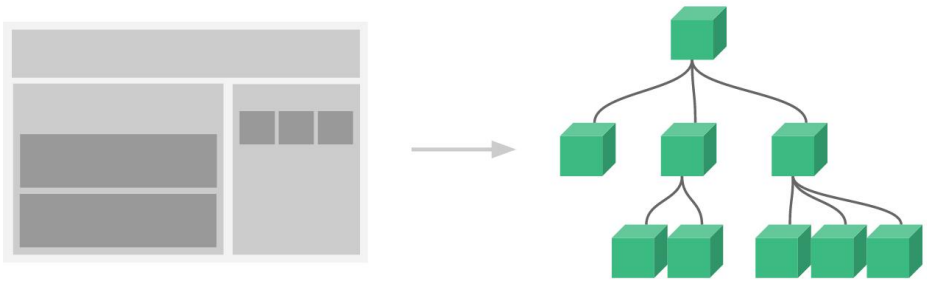

图 2-3 页面结构的树形示意

2.3.3 vue 响应式原理

Vue 最独特的功能之一是其非侵入性的响应式系统。数据模型仅仅是普通的 JavaScript 对象。而当你修改它们时，视图会进行更新。这使得状态管理非常简单直接。Vue 响应式系统的底层的细节如下图：

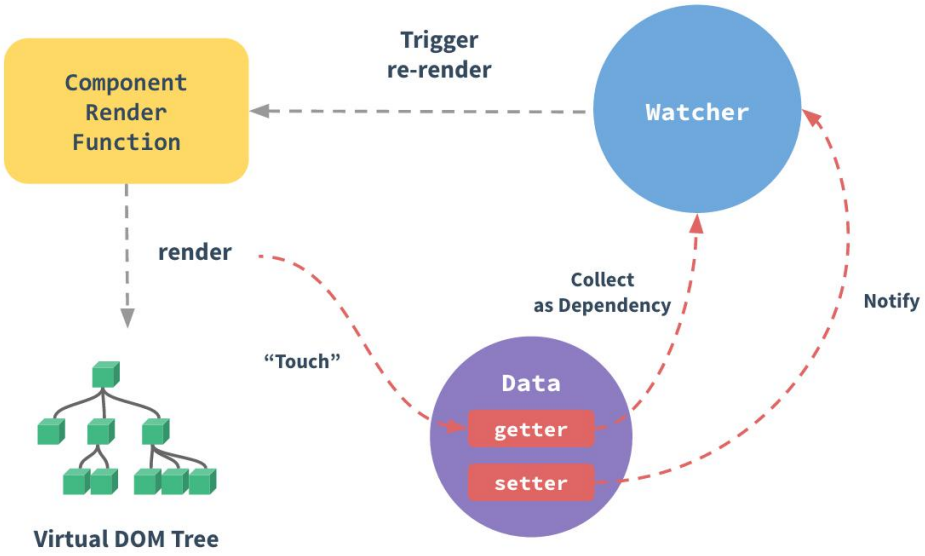

图 2-4 vue 响应式原理

vue 如何追踪变化呢？将普通的 JavaScript 对象传给 Vue 实例的 data 选项时，Vue 将迭代此对象所有的属性，并使用 Object.defineProperty 将所有这些属性转为 getter/setter。Vue.js 不支持 IE8 及更低版本的浏览器，这是由于 Object.defineProperty 是 ES5 中无法填充的功能，无法用已有的方式组合去实现。这些 getter/setter 对用户是不可见的，但是内部 Vue 通过它们跟踪依赖关系，在属性被访问和修改时通知变化。每个组件实例都有一个相应的 watcher 实例对象，该对象在组件呈现的过程将该属性记录为依赖项。稍后，当调用依赖项的 setter 时，会通知 watcher 重新计算，从而更新其关联的组件。这就是 vue 响应式更新组件的原理。

2.3.4 Vue-cli

Vue- cli 是 vue 官方基于 webpack 构建开发的脚手架工具。拥有一个非常完善的开发生态，开发者只需要关注业务代码本身，而不必操心复杂的 webpack 配置，可以说是对开发人员尤其是新手是相当友好的。

Vue CLI 提供了以下功能：

（1）通过 @vue/cli 搭建交互式的项目脚手架。

（2）通过 @vue/cli + @vue/cli-service-global 快速开始零配置原型开发。

（3）一个运行时依赖 (@vue/cli-service)，该依赖：

①可升级；

②基于 webpack 构建，并带有合理的默认配置；

③可以通过项目内存在的配置文件进行自主配置；

④可以通过插件进行扩展。

⑤丰富的官方插件集，集成了前端生态中的最佳工具。

⑥用于创建和管理 Vue.js 项目的完全图形用户界面。

2.4 状态管理

Vuex 是一个专为 Vue.js 设计的状态管理库，可利用 Vue.js 的细粒度数据响应机制实现高效的状态更新。本系统使用 vuex 来进行系统组件之前公共数据的管理。

Vuex 是一种集中的状态管理模型。在 vue 的模块化开发过程中使用了组件作为模块单元。 这确保了我们模块之间的变量函数名称不会发生冲突，但有时我们需要在组件之间共享一些数据或状态，通常使用参数。 但是传参的做法至少有两个弊端，一个是麻烦（特别是当需要传递很多参数时）， 其次，管理和冗余并不容易（将参数传递给多个组件需要多个参数列表。 而且容易出错）。vuex 提供的集中管理通过集中要共享的数据或状态来解决此问题。其他组件根据需要访问更改，大大提高了系统的可维护性和开发效率。

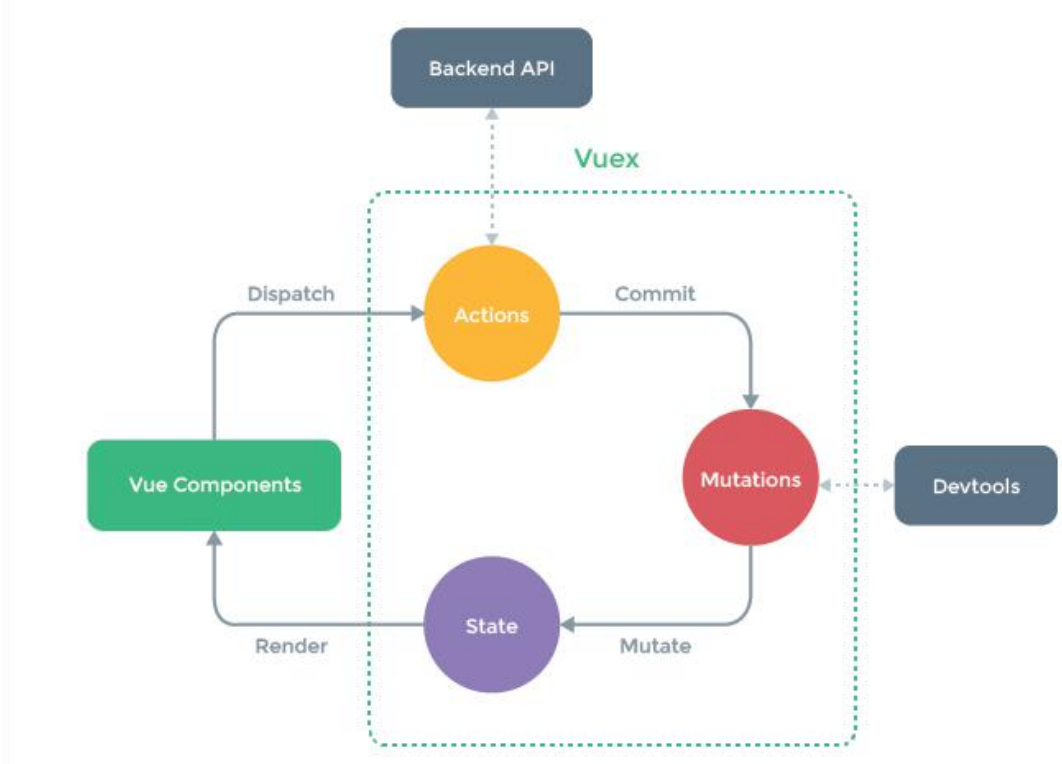

图 2-5 vuex 状态管理流程

所有 store 中 state 的改变，都放置在 store 自身的 action 中去管理。这种集中式状态管理能够被更容易地理解哪种类型的 mutation 将会发生，以及它们是如何被触发。当错误出现时，我们现在也会有一个 log 记录 bug 之前发

生了什么。

2.5 node.js

2.5.1 简介

在 node.js 出现之前，应用的服务器端实现往往比较复杂。创建服务器的技术门槛比较高，需要对多线程、伸缩性以及服务器部署有专业的技术知识才可以完成服务器的搭建。并且由于前后端开发语言的差异，使得开发者不得不使用多种编程语言，增加了开发的难度。

Google Chrome 浏览器的 v8 引擎是一个开源的项目，通过简单的 API 就可以将其集成进浏览器中，v8 解决了使用 javascript 作为服务端语言的性能与内存管理混乱两大问题。

Node.js 的作者 Ryan Dahl 发现了这样一个机会，将 V8 引擎内嵌到了操作系统的集成层，让 JavaScript 可以实现底层操作系统的异步接口，实现了将其带到服务器的目的。使得开发者可以使用同样的编程语言就可以完成客户端和服务器端的功能。

特点：Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.(node.js 使用事件驱动的非阻塞 I/O 模型，使其轻量级和高效).

2.5.3 Node.js Express 框架

Express 是一个简洁而灵活的 node.js Web 应用框架，提供了一系列强大特性帮助你创建各种 Web 应用，和丰富的 HTTP 工具。使用 Express 可以快速地搭建一个完整功能的网站。Express 框架是后台的 Node 框架，所以和 jQuery、element-ui、bootstrap 都不是一个东西。

Express 和 js 框架或者后台框架一样，把我们经常使用的东西比如方法放到一起，就形成了自己的一套东西，其实就是一个完善的库文件，里面写了大量的函数。Express 的思想是在于在工程师的想法和服务器之间充当很薄的一层。这并不意味着 Express 不够健壮，或者没有足够的有用的特性，而是尽量少干预你，让你充分表达自己的思想，同时提供一些有用的东西。Express 框架核心特性如下：

（1）强大的路由能力。

（2）强大的静态文件渲染能力。

（3）对模版引擎支持。可以通过向模板传递参数来动态渲染 HTML 页面。

（4）丰富的HTTP 工具以及来自Connect 框架的中间件随取随用，创建强健、友好的 API 变得快速又简单。

（5）性能更好。Express 在 node.js 上扩展了 Web 应用所需的功能

安装 Express 并将其保存到依赖列表中：$npm install express --save

第三章 系统分析

3.1 需求分析

想要完成一个系统，需求分析是第一步，明确系统是什么方向，要干什么，要完成那些功能。

本系统需求如下：

博客的游客用户可以在网站上对文章进行常规访问。以及在通过注册登陆后，可以进行文章的发布与发表评论。博主可以通过后台对用户进行管理以及文章与文章分类添加、删除、修改。

针对博客系统的以上需求，总结出如下信息：

（1）用户分为游客、普通用户和超级管理用户。

（2）超级管理用户员可以用户进行管理和设置权限。

（3）博客的超级管理员涉及对博客的文章类型管理、文章管理、评论管理、和用户管理。

（4）用户可以阅读文章、发表评论，游客只能进行文章的阅读。

（5）文章类型与文章之间形成一对多的关系，文章与评论之间也是一对多的关系。

3.2 可行性分析

可行性分析(Feasibility Analysis)的目的是以最小的代价在尽可能短的时间内确定问题是否能够解决。为了确定开发具有可行性，对本系统主要进行了以下几个方面的分析。

技术可行性分析：本人对于 vue.js 与 node.js 在实习中有所接触，对这方面有所了解，所以从技术方面看做个人博客是可行的。

时间可行性：现在博客技术已经非常成熟，多方面的问题都有成熟的解决方案，结合技术可行性，在预定的时间里可以完成此次设计。

第四章 系统概要设计

个人博客的总体规划图如下：

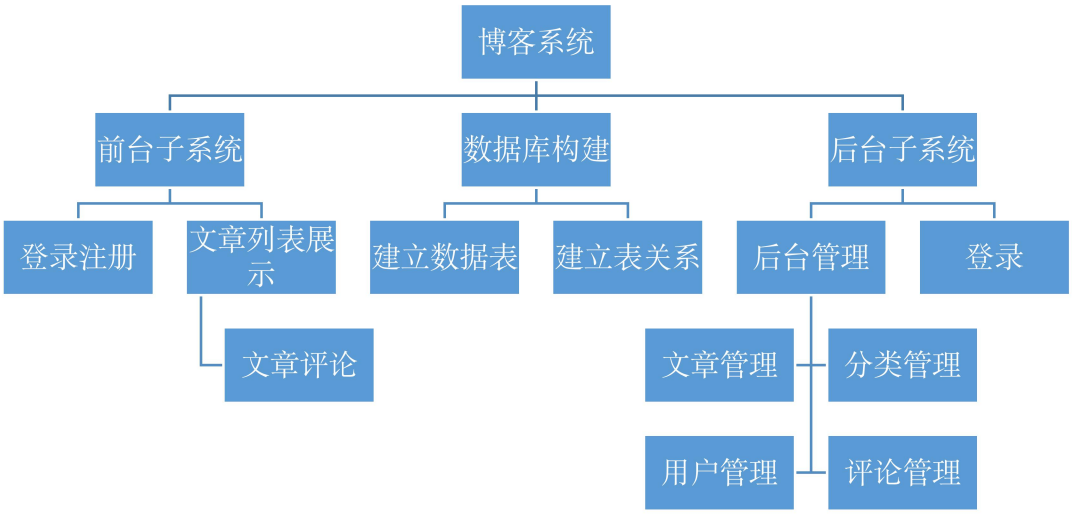

图 4-1 总体规划图

4.1 前台功能设计

在前台，游客浏览首页，文章详情页，文章列表与文章分类，注册用户在此基础上可以对文章进行评论。

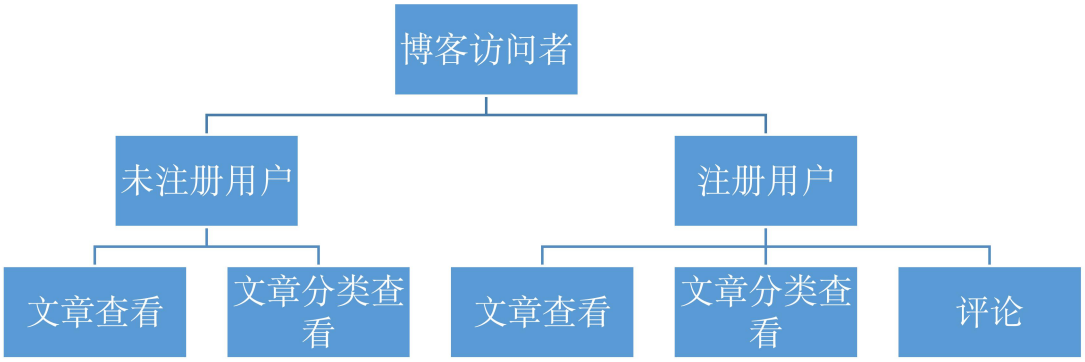

图 4-2 前台功能结构图

4.2 后台功能设计

系统的超级管理员通过预设的账号密码登陆后台。可以有所有的权限，对评论进行查看和删除；对注册用户进行管理，可以进行的操作有查看和封禁，对文章进行增删查改，对文章分类进行增删查改。

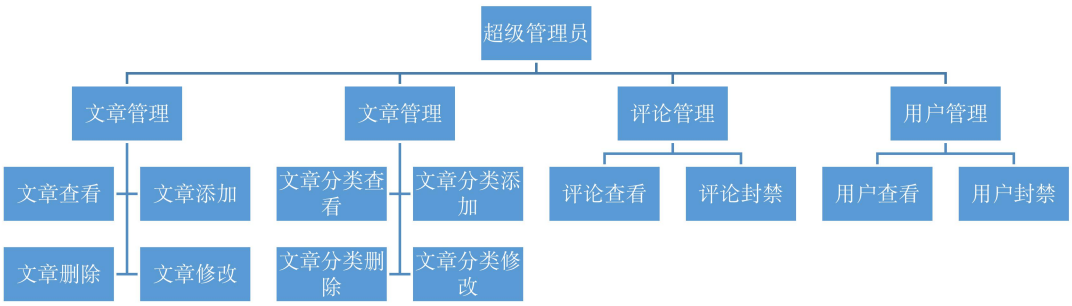

图 4-3 后台功能结构图

4.3 数据库设计

数据库是个人博客设计的主要部分，本系统选用了 mysql 数据库，使用 navicat premium 可

视化数据库管理工具进行数据库的设计与管理。

4.3.1 概念结构设计

对于属性比较多的实体，在 E_R 图中只列出了部分属性以做示意，具体的详细设计将体现在数据库表的结构中。如下图所示：

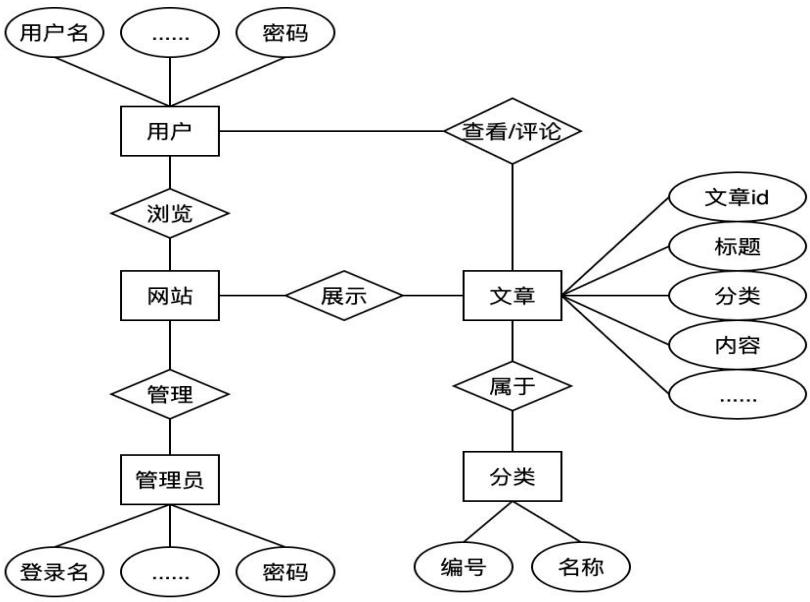

图 4-4 数据库概念结构 e-r 图

4.3.2 逻辑结构设计

个人博客系统的表结构设计如下所示。

（1）用户表管理用户的信息。表结构设计如下：

表 4-1 用户表

字段名称	数据类型	字段长度	说明
id	int	5	用户 ID(主键)
account	varchar	50	登陆名
password	varchar	50	密码
pwd_salt	int	5	密码加盐随机数
nickname	varchar	50	用户名字
birthday	date	20	出生日期
gender	tinyint	1	性别
introduce	varchar	200	个人介绍
status	tinyint	1	用户状态
role_id	int	5	角色 id

（2）评论表存储文章的评论，表结构设计如下：

表 4-2 评论表

字段名称	数据类型	字段长度	说明
id	int	5	评论 ID(主键)
art_id	int	5	文章 ID
content	varchar	255	评论内容
author	varchar	20	评论人

（3）文章表管理博客发表的文章，表结构设计如下：

表 4-3 文章表

字段名称	数据类型	字段长度	说明
id	int	5	文章 ID(主键)
type	int	20	文章类型
title	varchar	20	文章标题
content	mediumtext	255	文章内容
author_id	int	5	文章作者 id
status	tinyint	1	文章状态
views	int	20	文章浏览量
Comments	int	20	文章评论数

续表 4-3 文章表

字段名称	数据类型	字段长度	说明
createtime	int	20	文章创建时间
updatetime	int	20	文章更新时间
dest	varchar	100	对文章的简要描述

（4）文章分类表管理文章分类的具体名称，表结构设计如下：

表 4-4 文章分类表

字段名称	数据类型	字段长度	说明
id	int	5	角色 ID(主键)
name	varchar	20	角色名称
dest	varchar	20	角色描述
permission_list	varchar	255	权限列表

第五章 系统详细设计

5.1 注册功能的实现

将注册作为一个公共的组件独立出来，成为一个 vue 文件，用户通过填写简单的信息以及设置登陆密码即可完成注册。

主要代码：

```
reg(){
    var params={
      account:this.account,
      password:this.password,
    }
    //注册接口
    toregister({...params}).then((res)=>{
      if(res.data.status.code === 0){
        this.$message.success("注册成功");
      }else{
        this.$message.error('请求失败');
      }
    })
  }
```

5.2 登陆功能的具体实现

登陆分为前台的登陆与后台的登陆，两个登陆是公用一个组件，通过传入参数的不同去执行不同的登陆方法跳转不同的页面。首先用户在前台的表单中输入账号和密码，然后提交请求，查找用户的方法在数据库的用户表里的数据与提交的数据进行对比，若存在则登陆成功，跳转到博客前台首页，显示登陆状态，可以对文章进行评论。前台登陆如下图：

图 5-1 前台登陆

后台管理系统的登陆同前台登陆逻辑相同只是最后跳转的页面不同。后台登陆如下图：

图 5-2 后台登陆

5.3 前台文章列表的实现

通过不同的参数传递从 api 获得对应参数的文章数据，在前台进行展示。用户浏览文章与文章列表。通过对不同分类的选择改变 api 请求的参数，这样就可以获取不同分类下的文章。

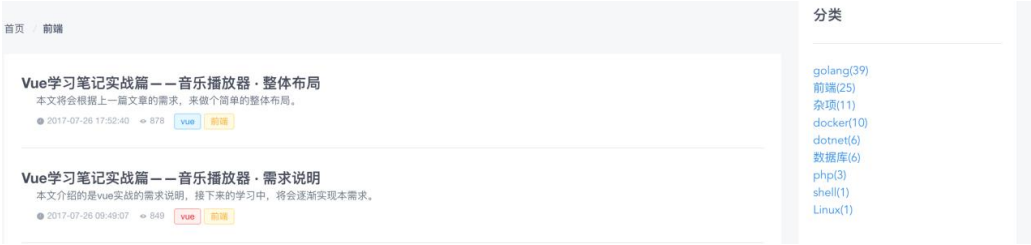

图 5-3 文章列表

5.4 后台管理系统的实现

后台实现对文章的增删查改，对文章列表的增删查改，对用户的查看和封禁以及对评论的查看和封禁。文章管理，分类管理，用户管理，评论管理都通过 axios 获取数据，然后通过对状态的修改来表示是可用还是不可用，是存在还是删除。

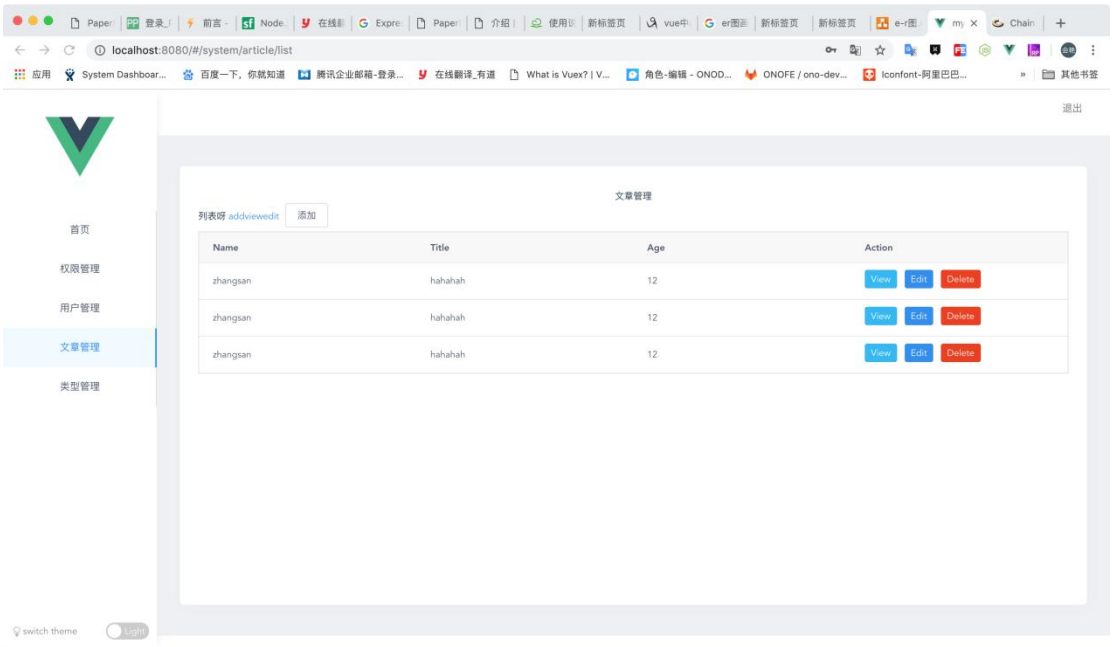

图 5-4 后台管理系统

5.5 组件及路由设计

本系统的项目结构如下图所示。所有的页面级的组件都放置在 views 文件夹下，一个文件夹或文件对应一个功能页面，公共组件或者每个页面所需要的组件放置在 components 文件夹下。

本系统的路由实现使用 vue-router 进行路由管理。Vue Router 是 Vue.js 官方的路由管理器。路由设计使用了 vue-router 的 history 模式与路由嵌套，在 router.js 中需要将组件（components）与路由（routes）进行映射，然后告诉在页面中子路由通过<router-view></router-view>标签呈现内容和实现路由变化使 Vue Router 知道在哪里渲染它们。

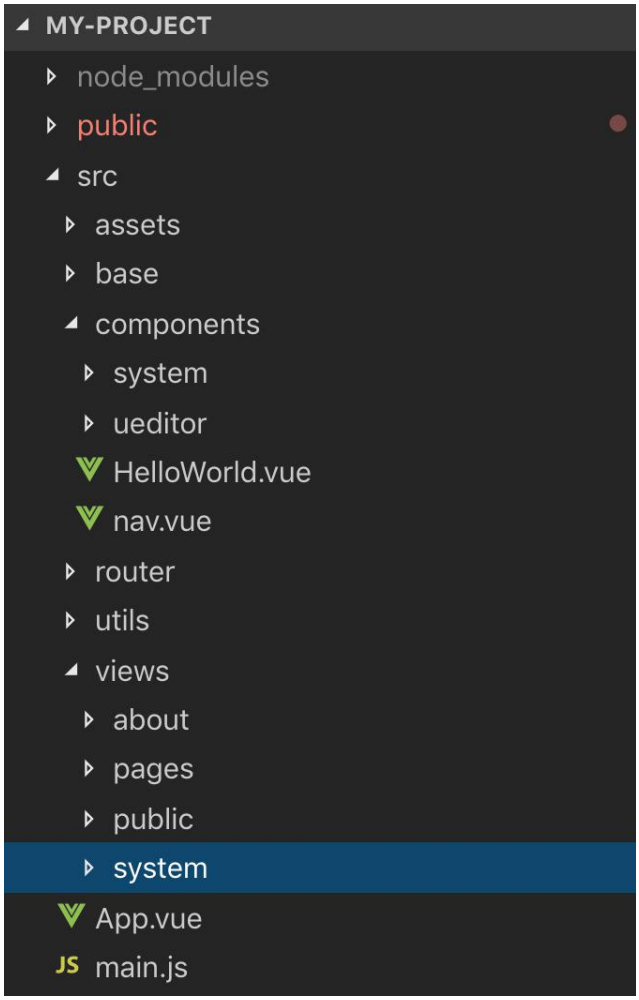

图 5-5 项目结构图

5.6 axios 封装

在 JavaScript 中发出 HTTP 请求的方法有很多，比如：Ajax，jQuery 中的方法（$.ajax，$.get，$.post 等），axios,Fetch 等。本系统选用 axios 进行数据请求。

在vue项目中,和后台交互获取数据这块,我们通常使用的是axios库。axios是可运行在浏览器端和node.js中的，它是一个基于promise的http库。拥有很多优秀的特性，例如拦截请求和响应、取消请求、转换json、客户端防御XSRF等。

本次使用对axios进行了封装，封装为request.js本质上返回了一个Promise。Promise 是一种对异步操作的封装，是一个保存着未来将要结束的事件的对象，可以通过独立的接口添加在异步操作执行成功、失败时执行的方法。当数据有返回并且返回的状态码是成功时，进入Promise的then中执行相应的操作，或是不是成功或者没有数据返回时，在catch中处理错误的情况。

主要代码:

```
import axios from 'axios';
import qs from 'qs';
import { getCurrentEnv } from './index';
const baseURL = getCurrentEnv();

// create an axios instance
const service = axios.create({
  baseURL, // api 的 base_url
  timeout: TIME_OUT, // request timeout
  headers: {
    'Content-Type': 'application/x-www-form-urlencoded',
  },
});

const request = (options) => {
  if (/post/i.test(options.method)) {
    options.data = options.params;
    delete options.params;
  }

  const version = options.customBaseURL;
  // eslint-disable-next-line
  options.baseURL = getCurrentEnv(version);
  return service(options);
```

```
};
export default request;
```

在系统中经常做的一件事就是对请求或响应进行拦截,希望在 then 或 catch 处理之前进行一些通用的操作。axios 提供了请求拦截(interceptors.request)和响应拦(interceptors.response)。

当求请发送时在请求拦截中判断当前接口是否需要 token 的,若需要就添加上 token。并在拦截里对 get、post 请求参数传递的方式进行区分。对于响应拦截主要是对于约定好的错误状态码或者常见的错误码进行统一处理,避免因为抛出异常导致的程序执行错误。

5.7 登陆的路由拦截

使用 vue-router 的 router.beforeEach 可以创建一个全局前置守卫。对于需要登陆的页面在路由定义时添加 meta: {requiresAuth: true}项, 来标识当前页面需要登陆才可以访问, 在 vuex 里创建 isLogined 作为已登陆标志。然后在这个钩子里判断当前页面是否需要登陆,即判断是否存在 meta 的 requireAuth 标签, 若 requireAuth 标签存在表明当前也是一个需要登陆的页面, 若是不存在则无需登陆。然后判断登陆状态即 isLogined 为 true 还是为 false,若是 false 则跳转到登陆页面 login, 若是 true 代表已登陆则不会执行去往登陆页面的代码。可以通过路由守卫实现简单的权限控制。

主要代码:

```
router.beforeEach((to, from, next) => {
const { name, meta } = to;
const { requiresAuth } = meta;
if (!store.state.isLogined) {
// 如果是需要登陆的页面
const needLogin = requiresAuth && !localStorage.getItem('info'); //从 localStorage 中读取是否获取了已登陆的信息
if (needLogin) {
next( 'login' );//跳转到登陆页面
} else {
next();
}
});
```

第六章 系统测试

测试是对系统可靠性的检测，让系统的运行更加稳定，在自测过程中发现系统的问题并及时解决，不影响正常使用。

(1) 测试功能：前台用户登陆功能。

测试用例：进入博客主页，若用户没有登陆，在进入需要登陆的界面或者使用需要登陆才能使用的功能是登陆拦截会起作用，将用户被重定向到博客首页。

① 填写正确的用户名及密码,用户名：zhy，密码：123456。

② 填写错误的用户名或密码,用户名：zhy，密码：1234567。

测试结果：① 成功登陆。

② 弹窗提示用户名或密码错误。

(2) 测试功能：后台用户登陆功能。

测试用例：点击后台登陆入口或者地址栏输入后台地址“http://localhost/#/system”。

① 填写正确的用户名及密码,用户名：admin，密码：123456a。

② 填写错误的用户名或密码,用户名：admin，密码：123456。

测试结果：① 成功登陆。

② 弹窗提示用户名或密码错误。

(3) 测试功能：前台文章展示功能。

测试用例：进入博客首页点击各个分类下的文章均能功能展示。

测试结果：文章详情展示无误。

(4) 测试功能：后台文章管理的删除功能。

测试用例：点击要删除的文章所在行的删除按扭。

测试结果：成功删除文章。

(5) 测试功能：后台文章管理的添加功能。

测试用例：① 填写正确的内容，不能为空的项全部填写。

② 填写正确的内容，为空的项全部不填写或部分不填写。

测试结果：① 成功添加文章。

② 弹窗提示未填字段为空请重新填写。

(6) 测试功能：后台文章管理的修改功能。

测试用例：① 填写正确的内容，不能为空的项全部填写。

② 填写正确的内容，为空的项全部不填写或部分不填写。

测试结果：① 成功修改文章。

② 弹窗提示未填字段为空请重新填写。

结论

虽然系统设计已经完成，基本功能已经可以使用，但还是存在着不足，具体为一下几个方面：

（1）由于使用了 vue 框架，不利于 eso 即搜索引擎对网站内容的抓取。但是 vue 对与这个问题已经有了解决方法，会在后期对系统进行此方面的优化。

（2）博客首页的可操作功能不够丰富。

（3）未加入详细的权限规划与分配，导致用户之间的界限不是很明显，对于用户是否拥有此功能，每次都需要前端页面进行复杂的判断，还有可能存在漏判的情况。

设计这个系统是对大学四年以来学习结果与能力的检验，系统的每一个小细节都需要考虑到，从系统的初始设计到确定设计方案再到确定所用技术与页面结构与分布等等。这些都用到了平时所学，是对的学到的知识的一个综合。本次系统的设计也让我明白细心是很主要的特质，同时也要善于听取他人的意见。

参考文献

[1]梁灏.Vue.js 实战[M].清华大学出版社.2017.

[2]程桂花,沈炜,何松林,张珂杰.Node.js 中 Express 框架路由机制的研究[J].工业控制计算机.2016,29(8):101-102.

[3]方晖, 蔡昭权.基于.NET 的博客系统的设计与实现[J]. 惠州学院学报, 2007, 27(3):66-71.

[4]吉晓香, 张国华.基于 B/S 模式的博客系统[J]. 电脑知识与技术, 2010, 6(11):2561-2562.

[5]刘磊. 基于Web框架的博客管理系统设计与实现[J]. 计算机时代,2017(5).

[6]奥尔波傅强, 陈宗斌. Node.js 入门经典[M]. 人民邮电出版社, 2013.

[7]朱二华.基于 Vue.js 的 Web 前端应用研究[J]. 科技与创新, 2017(20):119-121.

[8]麦冬, 陈涛, 梁宗湾.轻量级响应式框架 Vue.js 应用分析[J]. 信息与电脑(理论版), 2017(7):58-59.

[9]王伶俐,张传国.基于 NodeJS+Express 框架的轻应用定制平台的设计与实现[J].计算机科学.2017,44(z2):596-599.

[10]聂鑫.前端编程与数据库设计的合理运用[J].信息与电脑(理论版).2011,(2):100.

[11]陈帅,关玉蓉.基于 Java Web 的奖助学金系统设计与实现[J].科技广场.2017,(3):190-192.

[12]李玉.Vue 框架的前端交互性能优化解决方案的研究[D].华中科技大学.2017

[13]邹竞莹.Node.JS 博客系统的设计与实现[D].黑龙江大学.2016.

[14]旷志光,纪婷婷,吴小丽.基于 Vue.js 的后台单页应用管理系统的研究与实现[J].现代计算机.2017,(30):51-55.

[15]邓雯婷.基于 Vue.js 构建单页面 GIS 应用的方法研究[J].科技创新与应用.2018, (14): 5-7, 10.

[16]王志任.基于 Vue.js 的开发平台的设计与实现[D].广东工业大学.2018.

致谢

最后要感谢在整个论文写作过程中帮忙过我的每一个人。大学期间的课程知识在本次毕业设计中都有体现，本设计能够顺利的完成，也归功于各位任课老师的认真负责，使我能够很好的掌握和运用专业知识，是我可以在用到这门技术时可以用的出来。正是有了他们的悉心帮忙和支持，才使我的毕业论文工作顺利完成，在此向我的授课老师表示由衷的谢意。其中主要感谢的是我的指导老师，郑颖老师。在选题、系统设计与论文修改时都给了我很大帮助，帮助我确立了题目，在后续的时间里对于我的问题都给我了很多指导。最后也非常感谢在系统设计过程中给予我帮助的朋友与同学，你们是构成我大学生活的不可或缺的重要的部分。