**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**

[1 绪论 6](#_Toc1153405523)

[1.1博客的背景 6](#_Toc2099227239)

[1.2系统设计的意义 6](#_Toc703369310)

[2 系统相关技术介绍 6](#_Toc1778000082)

[2.1 SPA（Single Page App） 6](#_Toc612430169)

[2.1.1 单页应用 6](#_Toc224730312)

[2.1.2 与传统网页的比较 7](#_Toc1766102358)

[2.1.3 单页应用的好与坏 8](#_Toc363362072)

[2.2 NPM 9](#_Toc1730335683)

[2.3 vue.js 9](#_Toc528276507)

[2.3.1 介绍 9](#_Toc1045856451)

[2.3.2 组件化应用构建 10](#_Toc555721262)

[2.3.3 vue响应式原理 11](#_Toc600869631)

[2.3.4 Vue-cli 11](#_Toc1347780023)

[2.4状态管理 12](#_Toc481338005)

[2.5 node.js 13](#_Toc276951786)

[2.5.1 简介 13](#_Toc1131604253)

[2.5.2 Node.js模块系统 14](#_Toc757502339)

[2.5.3 Node.js Express 框架 15](#_Toc1058752157)

[2.5.4 Node.js 连接 MySQL 15](#_Toc398003657)

[3系统分析 16](#_Toc1983386441)

[3.1需求分析 16](#_Toc1534745153)

[3.2 可行性分析 16](#_Toc1035702354)

[4系统概要设计 17](#_Toc1694504743)

[4.1前台功能设计 17](#_Toc1760572734)

[4.2后台功能设计 17](#_Toc1916251972)

[4.3数据库设计 18](#_Toc634639345)

[4.3.1 概念结构设计 18](#_Toc1979680413)

[4.3.2逻辑结构设计 19](#_Toc1524558320)

[5 系统详细设计 20](#_Toc1624291883)

[5.1注册功能的实现 20](#_Toc661556917)

[5.2 登录功能的具体实现 20](#_Toc1264263500)

[5.3 前台文章列表的实现 21](#_Toc1273441082)

[5.4 后台管理系统的实现 21](#_Toc902239172)

[5.5 组件及路由设计 22](#_Toc551732337)

[5.6 axios封装 23](#_Toc131000213)

[5.7 登录拦截 25](#_Toc549841716)

[6 系统测试 25](#_Toc567587771)

7 [结论 27](#_Toc325307223)

[参考文献 28](#_Toc2092615346)

[致谢 29](#_Toc1246433303)

# 

# **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。访客和其他博主写我们的文章的留言或评论，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，我们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：

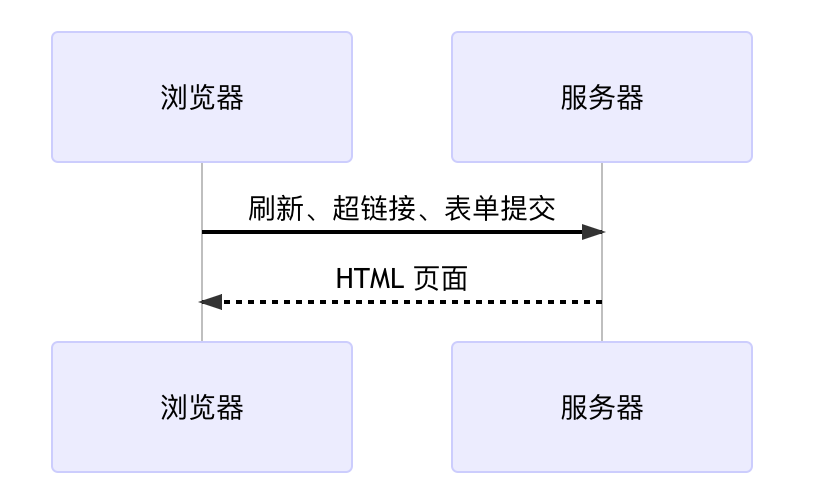
****

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：

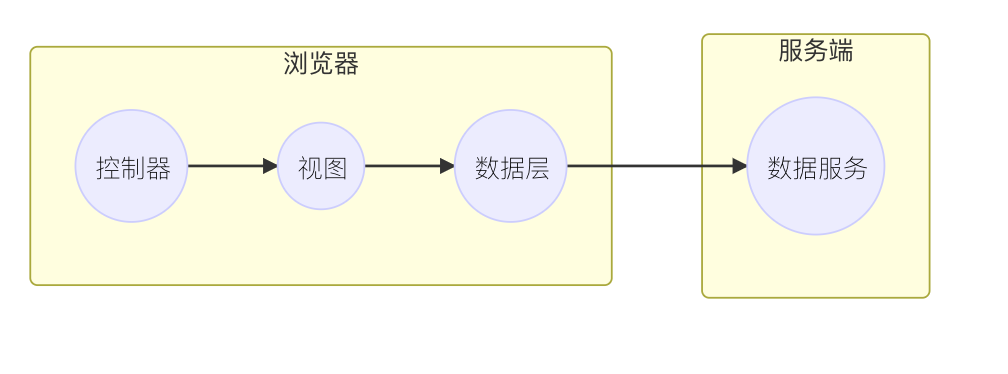
****

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。

## **2.2 NPM**

由于使用到的技术都是通过npm来安装的，所以先来介绍npm到底是什么：

NPM是一个包管理工具。为什么我们需要一个包管理工具呢? 因为我们在Node. js上开发时，很多时候都会用到很多别人写的JavaScript代码。如果我们要使用别人写的某个包，需要经历非常麻烦与复杂的过程，从搜索到这个代码，再到下载，解压，调试。于是一个集中管理的工具应运而生:开发者都把自己开发的模块进行打包（如webpack）后放到npm的官网上。这时如果要使用这个包，直接通过npm安装就可以直接得到，不需要知道代码存放在哪里，只要知道使用npm命令如何安装就行。

更重要的是，如果我们想要使用模块A，模块A依赖于模块B，模块B有与其它模块存在依赖关系。 Npm可以根据依赖性下载和管理所有依赖包。否则，靠我们自己手动管理，肯定既麻烦又容易出错。

NPM 即 Node 包管理器（Node Package Manager）。它是一个以 Node.js为默认环境 ，使用JavaScript 编写的软件包管理系统。在 Node.js 0.6.3 版本之前npm需要自行安装，但是node.js 0.6.3之后NPM 被自动附带在安装包中。

使用npm安装包的命令是：npm install 包名 [--save-dev]

--save与-dev都是可选的，--save表示自动修改package.js文件，并自动添加依赖项。

--save:将安装的包放在依赖项dependencies中

--save-dev:安装的包将出现在devDependencies中

## **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。

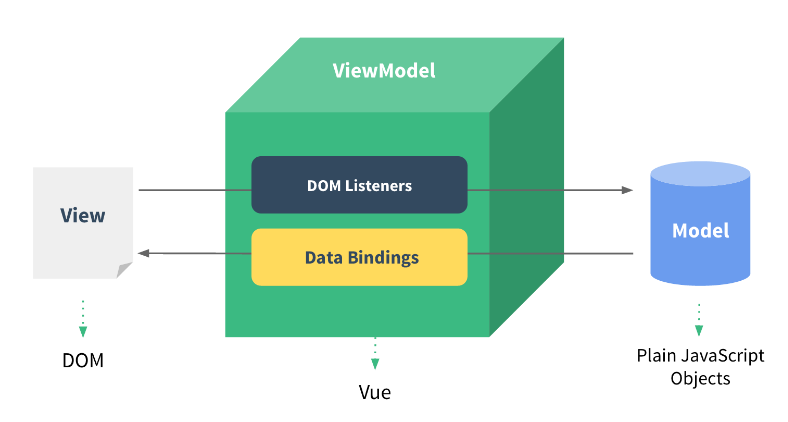


图 2-3 Vue.js中View与Model的交互原理

图2-3不仅描述了MVVM模式（Model-View\_ViewModel），还描述了在Vue.js中ViewModel是如何和View以及Model进行交互的。

ViewModel是Vue.js的核心，它是一个Vue实例。Vue的实例是绑定在HTML的body元素，或者是指定了id的某个HTML的元素上的。

当创建了ViewModel后，双向绑定是如何达成的呢？

DOM Listener和Data Bindings这两个工具是实现双向绑定的关键。

在View侧也就是页面上，DOM Listeners工具会监测页面上DOM结构与元素的变化，一旦发生变化，就会去更改Model中的数据。

在Model侧也就是传统意义上的服务器端上，当我们将Model中的数据进行更新时，Data Bindings工具会将数据变化同时到页面，从而更新页面中的DOM元素。

### 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 2-4 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 2-5 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。

上边提到了webpack,那么什么是webpack呢？请看以下介绍**：**

webpack就是前端资源模块化管理和打包工具，它可以将很多松散的模块按照依赖和规则打包成符合生产环境部署的前端资源，还可以将按需加载的模块进行代码分割，等到实际需要的时候再异步加载，而要它自动实现这些功能，你得提前编辑好配置文件。

用vue项目来举例：浏览器它是只认识js，不认识vue的。我们编写的代码后缀主要是.vue的，而html、js、css甚至是图片资源可能是在每个.vue文件中；并且由于组件化，每一个.vue文件之间还有错综复杂的关系。所以项目要被浏览器识别，我们就要使用webpack将它们打包成js文件以及相应的资源文件。它做的事情是，分析你的项目结构，找到 JavaScript模块以及其它的一些浏览器不能直接运行的拓展语言（ less，TypeScript等），并将其打包为合适的格式以供浏览器使用。

## **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 2-6 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/0 model that makes it lightweight and efficient.(node.js使用事件驱动的非阻塞I/0模型，使其轻量级和高效).

### **2.5.2 Node.js模块系统**

什么是模块?在Node.js中，将很多的功能，划分成了一个个的module,有些书将其翻译为模块，有些书将其翻译为模组。由于每一个程序的功能不一样，用到的所以为了效率，需要什么模块就使用require引入什么模块。模块是Node.js 应用程序的基本组成部分，文件和模块是一一对应的。换言之，一个 Node.js 文件就是一个模块，这个文件可能是JavaScript 代码、JSON 或者编译过的C/C++ 扩展。每个模块都有自己的作用域，当我们使用 var 来申明的一个变量，他并不是全局的，而是属于当前模块下。

模块是构建应用程序的基础，也使得函数和变量私有化，不直接对外暴露出来。JavaScript起初并没有内置的模块系统，CommonJS社区为了使JavaScript可以提供一个类似Python、Ruby等的标准库，自己实现了一套API填补了JavaScript没有内置模块的空白。CommonJS规范本身涵盖了模块、二进制、Buffer、文件系统、包管理等内容，而NodeJS正是借鉴了CommonJS规范的模块系统，自身实现了一套非常易用的模块系统。CommonJS对模块的定义可分为三部分：模块引用（require）、模块定义（exports、module）、模块标识。

在Node.js中，以模块为单位划分所有功能，并且提供了一个完整的模块加载机制，这时的我们可以将应用程序划分为各个不同的部分。我们不可能用一个js文件去写全部的业务。狭义的说，每一个JavaScript文件都是一个模块:而多个JavaScript文件之间可以相互require,他们共同实现了一个功能，他们整体对外，又称为一个广义上的模块。Node. js中，一个JavaScript文件中定义的变量函数，都只在这个文件内部有效。当需要从此JS文件外部引用这些变量、函数时，必须使用exports对象进行舉露。使用者要用require命令引用这个JS文件。例子如下：

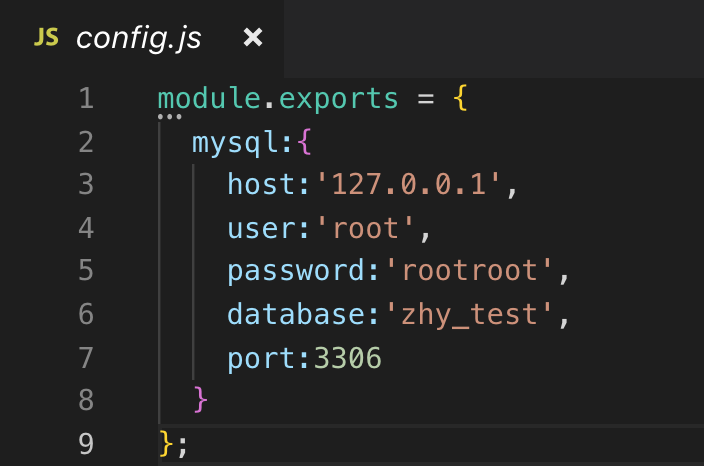


图 2-7 创建模块

当我们需要在其他地方使用 config.js方法时，我们只需要调用 require('./config) 就能实现对模块的引入。比如：

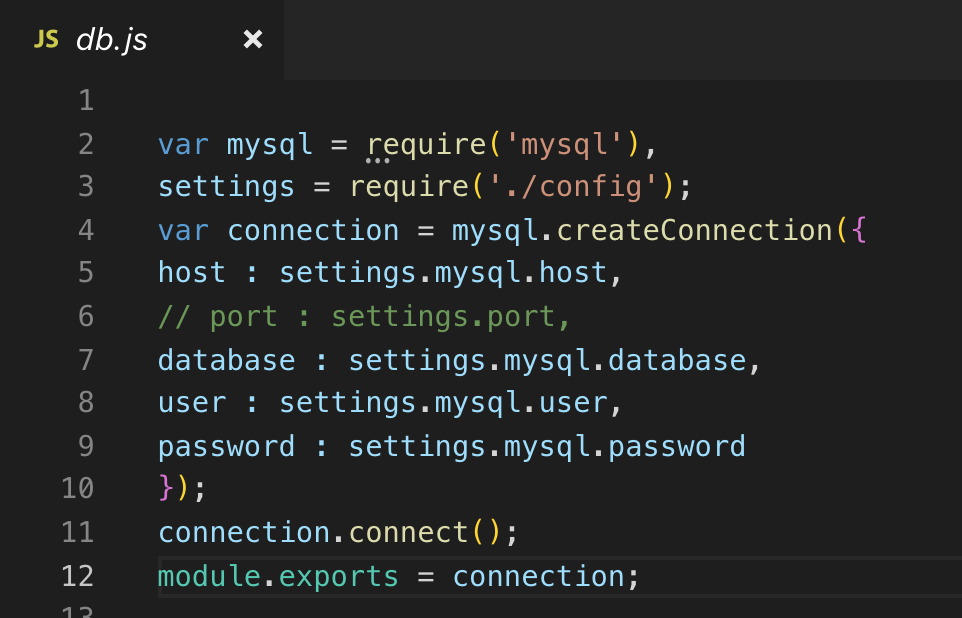


图 2-8 创建模块

### **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

### **2.5.4 Node.js 连接 MySQL**

连接数据库代码如图所示，其中connect()方法用来创建连接，end()方法用来关闭连接，query()方法用来向mysql传递参数。

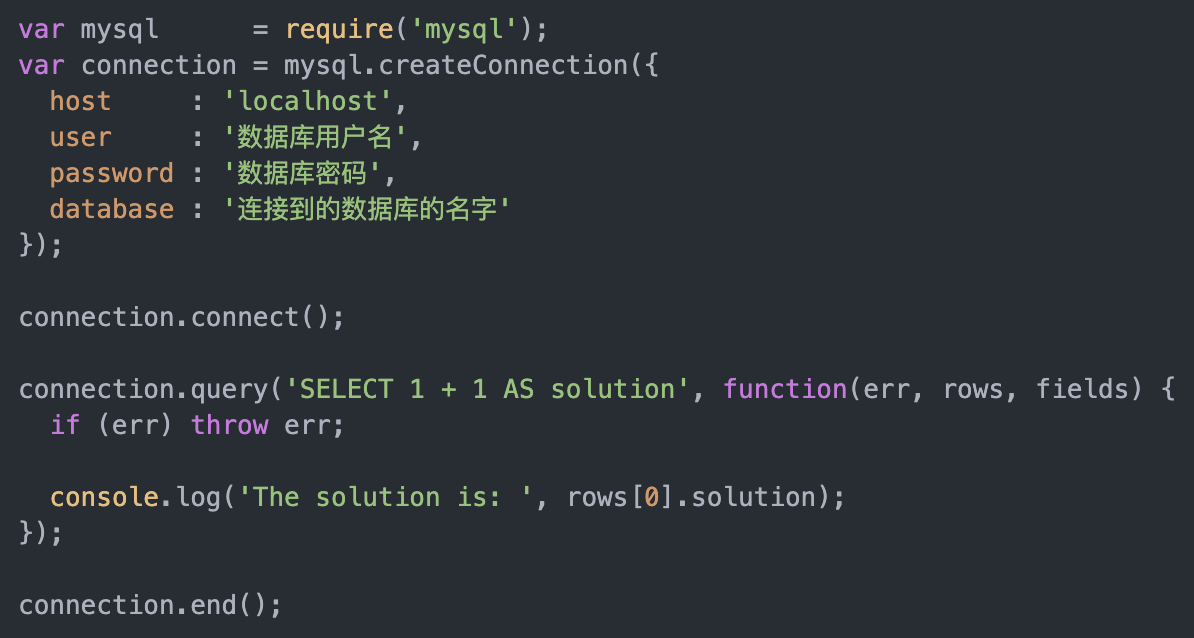


图 2-9 连接mysql数据库

# **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。以及在通过注册登录后，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录后台。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：

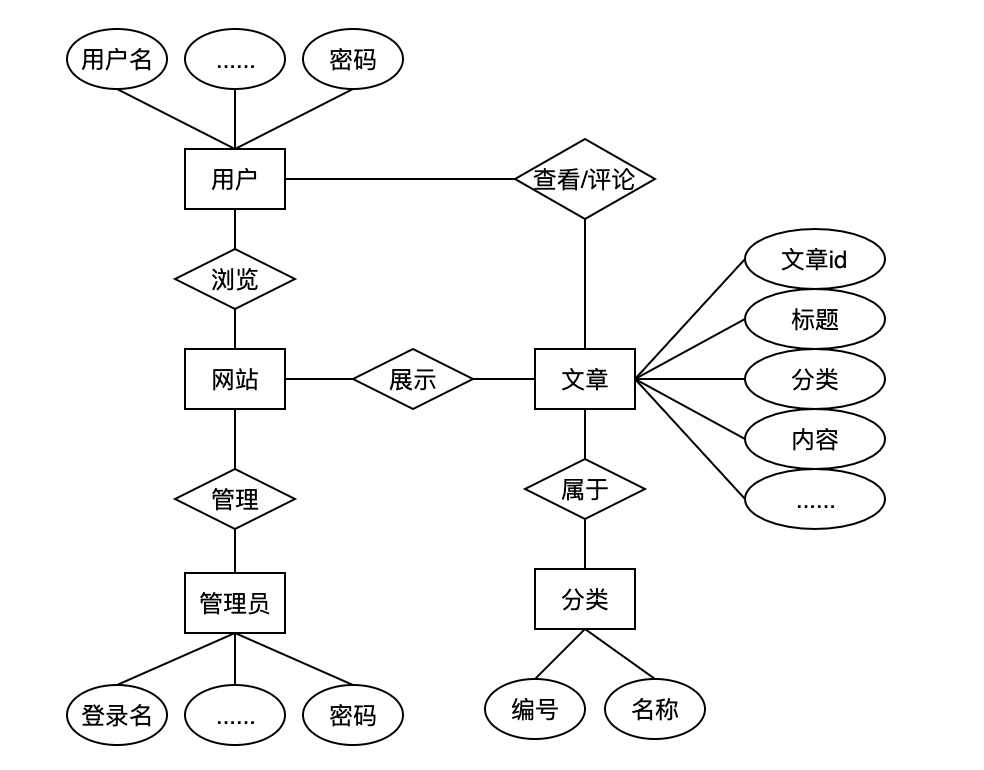


图 4-4 数据库概念结构e-r图

### 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：

|  |  |  |  |
| --- | --- | --- | --- |
| **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 | 登录名 |
| 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 |

表 4-1 用户表

（2）评论表存储文章的评论，表结构设计如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |

表 4-2 评论表

（3）文章表管理博客发表的文章，表结构设计如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |
| 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 |

表 4-3 文章表

（4）文章分类表管理文章分类的具体名称，表结构设计如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |

表 4-4 文章分类表

# **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录密码即可完成注册。

主要代码：

## 5.2 登录功能的具体实现

登录分为前台的登录与后台的登录，两个登录是公用一个组件，通过传入参数的不同去执行不同的登录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，若存在则登录成功，跳转到博客前台首页，显示登录状态，可以对文章进行评论。前台登录如下图：



图 5-1 前台登录

后台管理系统的登录同前台登录逻辑相同只是最后跳转的页面不同。后台登录如下图：



图 5-2 后台登录

## 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。

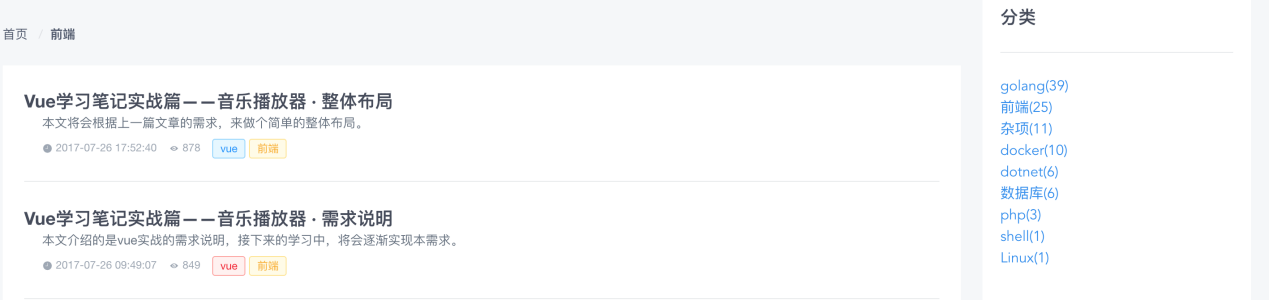


图 5-3 文章列表

## 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。

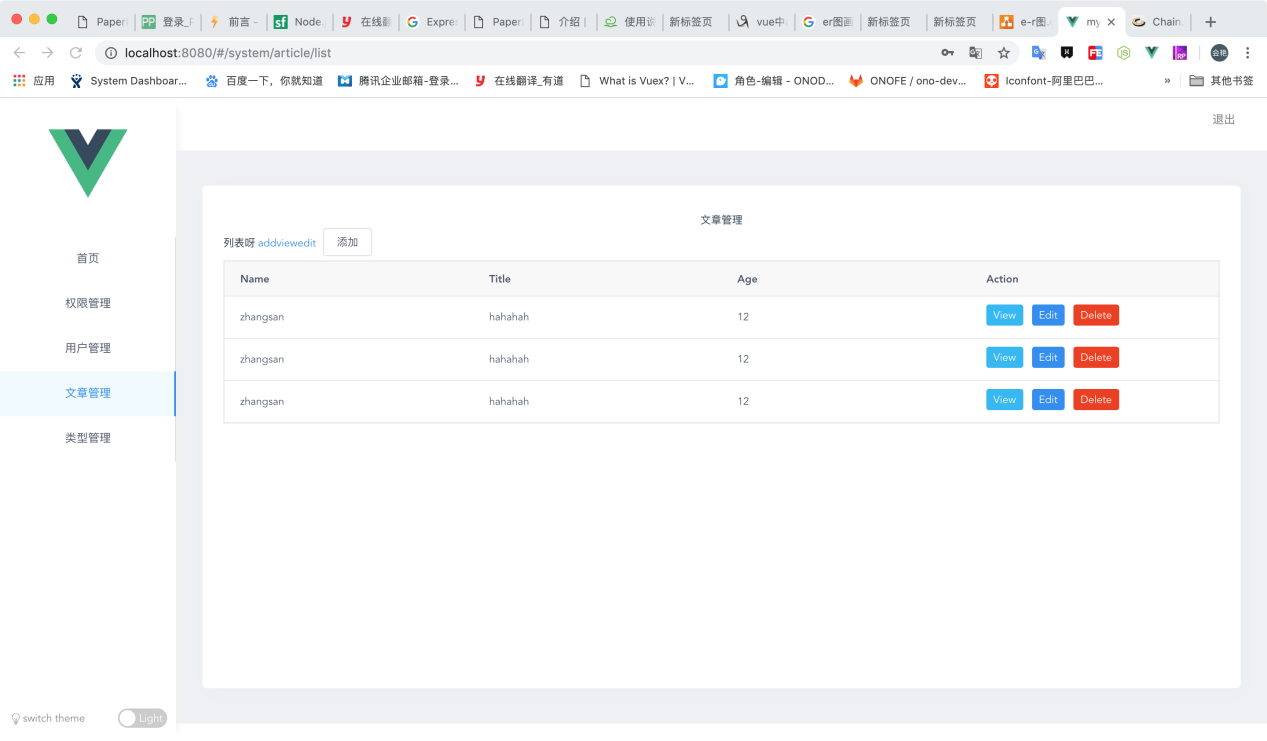


图 5-4 后台管理系统

## **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。

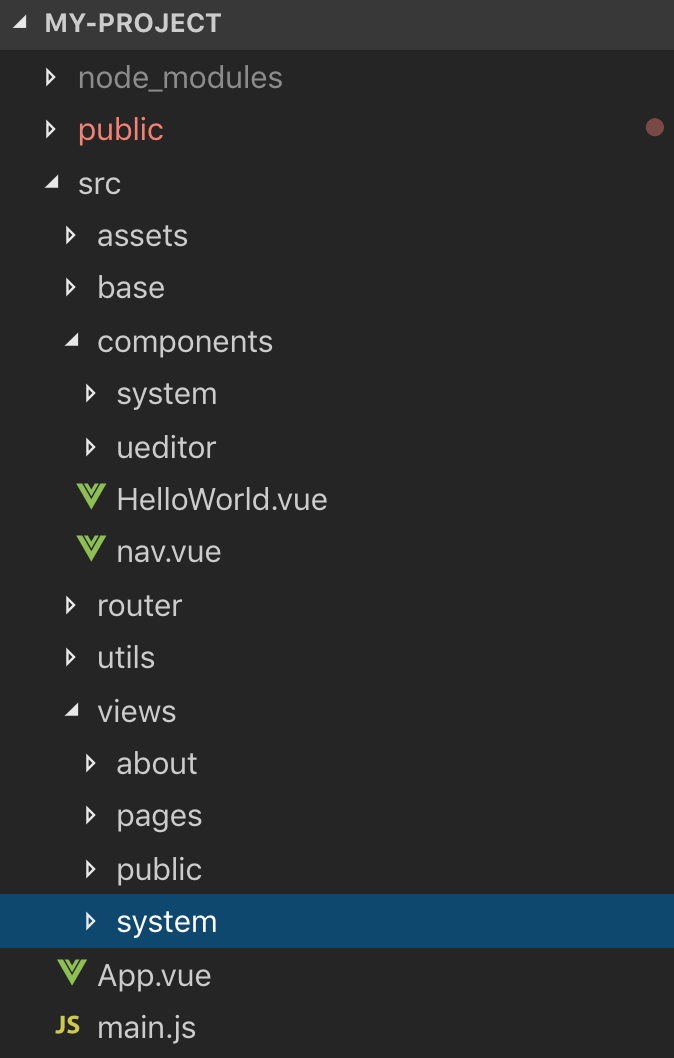


图 5-5 项目结构图

本系统的路由实现使用vue-router进行路由管理**。**Vue Router 是 [Vue.js](http://cn.vuejs.org/" \t "/Users/fisher_998/Documents\\x/_blank) 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。

## 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。

主要代码：

## 5.7 登录拦截

使用vue-router的router.beforeEach可以创建一个全局前置守卫。对于需要登录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录才可以访问，在vuex里创建isLogined作为已登录标志。然后在这个钩子里判断当前页面是否需要登录，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录的页面，若是不存在则无需登录。然后判断登录状态即isLogined为true还是为false,若是false则跳转到登录页面login，若是true代表已登录则不会执行去往登录页面的代码。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) {

// 如果是需要登录的页面

const needLogin = requiresAuth && !localStorage.getItem('info'); // 从localStorage中读取是否获取了已登录的信息

if (needLogin) {

next(‘login’);//跳转到登录页面

} 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：前台用户登录功能。

测试用例：进入博客主页，若用户没有登录，在进入需要登录的界面或者使用需要登录才能使用的功能是登录拦截会起作用，将用户被重定向到博客首页。输入用户名：zhy，密码：1234567，会提示用户名或密码错误，如图7-1所示。如果输入用户名：zhy，密码：123456，点击登录按钮，则登录成功。

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 提示用户名或密码错误。

(2) 测试功能：后台用户登录功能。

测试用例：点击后台登录入口或者地址栏输入后台地址“http://localhost/#/system”。输入用户名：admin，密码：123456，会提示用户名或密码错误，如图7-2所示。如果输入用户名：admin，密码：123456a，点击登录按钮，则登录成功。

(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，如图7-3所示。文章详情展示无误，如图7-4所示。

(4) 测试功能：后台文章管理的删除功能。

测试用例：

(5) 测试功能：后台文章管理的添加功能。

测试用例：

(3) 测试功能：后台文章管理的添加功能。

测试用例：

1. 用户测试权限

测试内容：填写用户名及密码，验证能否成功登陆。

操作：① 填写正确的用户名及密码。

② 填写错误的用户名或密码。

结果：① 成功登陆。

② 提示用户名或密码错误。

# **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**