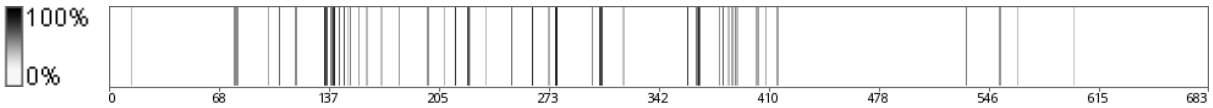
PaperPass旗舰版检测报告
简明打印版

比对结果(相似度):

总　体：10% (总体相似度是指本地库、互联网的综合对比结果)
本地库： 5% (本地库相似度是指论文与学术期刊、学位论文、会议论文、图书数据库的对比结果)
期刊库： 3% (期刊库相似度是指论文与学术期刊库的比对结果)
学位库： 3% (学位库相似度是指论文与学位论文库的比对结果)
会议库： 1% (会议库相似度是指论文与会议论文库的比对结果)
图书库： 2% (图书库相似度是指论文与图书库的比对结果)
互联网： 5% (互联网相似度是指论文与互联网资源的比对结果)

报告编号：5CB3B5D52BDD4F29U
检测版本：旗舰版
论文题目：[降重]基于vue的个人博客的设计与实现1
论文作者：周会艳
论文字数：14824字符(不计空格)
段落个数：456
句子个数：683 句
提交时间：2019-4-15 6:36:05
比对范围：学术期刊、学位论文、会议论文、书籍数据 、互联网资源
查询真伪：http://www.paperpass.com/check

句子相似度分布图:



本地库相似资源列表(学术期刊、学位论文、会议论文、书籍数据):
暂无本地库相似资源

互联网相似资源列表：
1.相似度： 2% 标题：《单页面应用 - yangyang的专栏 - CS...》
https://blog.csdn.net/u014465934/article/details/82948700
2.相似度： 1% 标题：《RESTful 架构详解 菜鸟教程》
https://www.runoob.com/w3cnote/restful-architecture.html
3.相似度： 1% 标题：《Vue状态管理vuex - 小火柴的蓝色理想 -...》
https://www.cnblogs.com/xiaohuochai/p/7554127.html
4.相似度： 1% 标题：《对于restful的简单理解 - ...》
https://blog.csdn.net/Poison__biting/article/details/77072695
5.相似度： 1% 标题：《 Vue 状态管理》
http://www.mamicode.com/info-detail-1977539.html
6.相似度： 1% 标题：《Vue 32 内部 - 深入响应式原理 - ...》
https://blog.csdn.net/wudizhanshen/article/details/86653946

全文简明报告:

基于vue的个人博客的设计与实现

姓名： 周会艳

院系： 信息工程学院

专业： 计算机科学与技术

导师： 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， {43%：互联网技术在人们的生活，工作和学习的各个方面都发挥着重要作用。}而博客网站正适应这种人际与技术交流方式的改变，满足个人技术与信息共享的需求。 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 to communicate with others in today’ s world. Internet technology plays an important role in all aspects of people’ s life, work and study. Blog sites are adapting to this change in interpersonal and technical communication, to meet the needs of personal technology and information sharing. This article has carried on the complete analysis to the blog function and the demand, has designed a simple, easy to use personal blog system. In order to improve the development efficiency and code reuse rate, the system USES the idea of SPA(single page application) for the development of vue. Js system. Vue. Js is a lightweight progressive framework based on the MVVM pattern. The idea of vue componentization and data binding greatly simplifies the complexity of front- end development. The back end USES node. js technology in conjunction with express to create interfaces that conform to RESTful API design specifications .

keywords: blog, vue.js , node.js , API, vuex

绪论

1.1博客的背景

Blog是继Email，BBS和ICQ之后的第四种网络通信方式。 {59%：它也代表了一种新的生活方式，一种新的工作方式和一种新的学习方式。} {58%：简而言之，博客是以网络为载体，快速，轻松地发布您自己的想法，并及时有效地与他人沟通。} {58%：这是一个丰富多彩的综合性个性化平台。}

博客的分类有很多。 比如，按功能分： 基本博客、微型博客； 按用户分： 个人博客、企业博客； 按存在方式分： 附属博客，托管博客，独立博客。 本系统就是按用户划分的个人博客。

博客有以下特点：

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。 {49%：互联网为博客提供了开放性，因此博客不再是一个简单的私人空间。} 访客和其他博主写我们的文章评论或评论，如果我们回复并通过链接地址返回给他们，我们可以实现互动结果。 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， {69%：互联网在人们的日常生活中起着举足轻重的作用。}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， 去适应这种人际与技术交流方式的改变，同时也实现个人技术与信息共享的需求。

系统相关技术介绍

SPA（Single Page App）

单页应用

单页应用（英语： {58%：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，也避免了不断的加载整个新页面，减轻了服务器的负担。 在单页面应用程序中，通过加载单个页面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支持丰富的客户端功能，很少需要重新加载整个页面，因此当用户执行操作或在应用程序的各个区域之间导航时，无需重新加载整个页面。 因此，它加载速度更快，在后台提取数据，并对单个用户操作响应更快。

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

/

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

/

传统网页应用的工作分布

在传统的WEB应用程序中，浏览器作为显示层视图，模式和控制器以及诸如路由跳转，数据请求等的计算和调度由服务器执行。

{100%：在传统的网页应用中，浏览器更多的是充当一个展示层，路由处理、服务调用、页面跳转流程都由服务端来处理。} {77%：也就是说，MVC被放置在服务器端，并且V作为用

户界面通过网络被发送到浏览器以作为UI与用户交互。}

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，后端工程师通常会领导整个项目的开发;

{62%：页面经常刷新，当页面的功能发生变化时，页面就需要进行刷新，导致资源的浪费。} {81%：而且导致用户需要花费额外的时间等待页面刷新，用户体验不佳。}

{100%：相较于传统网页应用，单页应用将 MVC 前置到了浏览器端：}

/

单页应用的工作分布

{71%：1控制器前置，单页应用将对页面路由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，} 将更改通知给相应的路由以向用户呈现相应的接口。

2小组件为功能元件。 {77%：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。}

3数据层前置。 {41%：JSON发送应用程序数据的方式在HTML视图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，{62%：而服务端则退化成了完全的数据 API，方便不同的开发人员独立开发每个级别。}

单页应用的好与坏

每一种技术都有优缺点，单页应用也不例外。 单页面开发的优点：

1良好的用户体验： {53%：用户无需刷新页面，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计算，使得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， {44%：服务器不再负责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。 {42%：因此，css和js文件会更大，这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。

如果您的应用程序要求不仅包括典型HTML可以提供的丰富功能，那么您应该选择SPA样式的应用程序。

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：

为什么我们需要一个包管理工具呢? 因为我们在Node.在js上开发时，它将使用其他人编写的大量JavaScript代码。 如果我们想使用其他人编写的软件包，那么按名称搜索官方网站，下载代码，提取代码并再次使用它是非常麻烦的。 于是一个集中管理的工具应运而生: 每个人都打包了他们开发的模块并将它们放在npm官方网站上。

{54%：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C和模块D是模块B的依赖性。} 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

2.1vue.js

2.1.1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的核心库仅侧重于视图层，使其不仅易于使用，而且易于与第三方库或现有项目集成。 {56%：另一方面，当与现代工具链和各种支持库结合使用时，Vue.js也完全能够为复杂的单页应用程序提供驱动程序。}

/

Vue.js中View与Model的交互原理

此图不仅描述了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的一个实例绑定到HTML的一个元素，它可以是HTML的body元素或指定id的元素。

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（即页面）上，DOM Listeners工具将监视DOM结构和页面元素的更改。

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，{44%：数据绑定工具通过将数据更改为页面来更新页面中的 DOM元素。}

2.2.1 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：

/

页面结构的树形示意

2.2.2 vue响应式原理

{84%：Vue最独特的功能之一是其非侵入式响应系统。} 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：

/

vue响应式原理

vue如何追踪变化呢？ {77%：将普通JavaScript对象传递给Vue实例的数据选项时，Vue将遍历此对象的所有属性，并使用Object.defineProperty将所有这些属性转换为getter / setter。} {41%：由于Object.defineProperty是ES5中无法填充的功能，因此Vue.js不支持IE8及更低版本的浏览器。} 这些getter / setter对用户是不可见的，但内部Vue通过它们跟踪依赖关系，通知在访问和修改属性时的更改。 每个组件实例都有一个相应的观察器实例对象，该对象在组件呈现期间将该属性记录为依赖项。 稍后，当调用依赖项的setter时，会通知观察者重新计算以更新其关联的组件。 这就是vue响应式更新组件的原理。

2.2.5 与其他框架相比优点

1易用。 只需要已学习了HTML、CSS、JavaScript，就可以阅读vue开发指南并开始构建应用程序。 不像其他前端框架一样有比较高的使用门槛。

2.灵活。 {44%：一个繁荣的生态系统，可以在图书馆和完整的框架之间进行扩展。}

Vue-cli

Vue- cli是 vue官方开发的脚手架工具，基于 webpack构建，很好地规范了其开发生态，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：

{70%：Webpack是一种前端资源模块化管理和打包工具，可以将许多松散模块打包成依赖关系和规则，以满足生产环境中部署的前端资源。} 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（如图）， 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，而html，js，css甚至图像资源都可能在每个.vue文件中; 而且由于组件化，每个.vue文件之间存在复杂的关系。 因此项目需要被浏览器识别，我们将使用webpack将它们打包成js文件和相应的资源文件。

WebPack可以看做是模块打包机： 它的作用是分析你的项目结构，找到 JavaScript 模块和浏览器无法直接运行的其他扩展语言（更少， TypeScript等）并将其打包成适合浏览器使用的格式。

2.3状态管理

{85%：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。 nbsp;

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。{47%：其他组件根据需要访问更改，大大提高了系统可维护性和开发效率。}

/

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。 {100%：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。} {97%：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。}

2.4node.js

4.1.1简介

Node.js是一个Javascript运行环境(runtime)，发布于2009年5月，由Ryan Dahl开发，实质是对Chrome V8引擎进行了封装。 Node，js优化了一些特殊用例，并提供了一个替代API，使V8在非浏览器环境中更好地工作。

V8引擎本身是Chrome浏览器JS解释的一部分，但Ryan Dahl将V8移至服务器并将其用作服务器软件。

Node.Js是一位专注于优化高性能Web服务器的专家。

Node.Js是一个允许JavaScript在服务器端运行的开发平台。

但Node似乎有点不同:

Node.Js不是一种独立的语言，不像PHP和JSP Python“是一种语言和一种平台”，Node。js使用JavaScript进行编程，运行在JavaScript引擎上(V8)。

与PHP、JSP等相比(PHP、JSP、 . net都需要运行在服务器程序上，Apache、 Nginx、IIS.） ， Node.Js跳过Apache，Naginx，IIS等HTTP服务器。 它不必构建在任何服务器软件之上。 Node.许多js的设计概念与经典架构（LANP = Linux + Apache + MySQL + PHP）完全不同，后者提供了强大的可扩展性。 一会儿我们就将看到，Node.js没有web容器。

Node.Js自己的理念是花费最低的硬件成本，追求更高的并发性和更高的处理性能。

官网: https: //nodejs.org en/

特点: Node.js uses an event- driven, non- blocking I/O model that makes itlightweight and efficient.(node. {60%: js使用事件驱动的非阻塞 I/O模型，使其轻量级和高效).}

2.4.1Node.js模块系统

什么是模块?

在Node.js中，许多功能分为模块，有些书将它们翻译成模块，有些书将它们翻译成模块。由于每个程序具有不同的功能，因此使用哪些模块来提高效率，需要哪些模块来提高效率。

{81%：模块是Node.js应用程序的基础部分，文件和模块是一对一的。} {78%：换句话说，Node.js文件是一个模块，可以是JavaScript代码，JSON或编译的C / C ++扩展。}每个模块都有自己的范围。

模块是构建应用程序的基础，它们还使函数和变量私有化，而不是直接暴露。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，所有功能都分为模块，并提供完整的模块加载机制。 我们不可能用一个js文件去写全部的业务。 肯定要有WVC.

从狭义上讲，每个JavaScript文件都是一个模块： 可以相互需要多个JavaScript文件，它们共同实现一个功能。

{41%：Node.在js中，JavaScript文件中定义的变量函数仅在此文件中有效。} {43%：当您需要从此JS文件外部引用这些变量和函数时，必须使用oxports对象来公开它。} 使用者要用require0命令引用这个JS文件。 例子如下：

在Node.js 中创建模块

/

当我们需要在其他地方使用 config.js方法时，比如：

/

我们只需要调用 require('./config) 就能实现对模块的引入。

Node.js Express 框架

Express是一个简单而灵活的node.js Web应用程序框架，它提供了一系列强大的功能，可帮助您创建各种Web应用程序以及丰富的HTTP工具。 使用Express快速构建功能齐全的网站。 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的想法是充当工程师的想法和服务器之间的薄层。 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。 您可以通过将参数传递给模板来动态呈现HTML页面。

Connect框架提供了4个丰富的HTTP工具和中间件，可以快速轻松地创建健壮，友好的API

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

2.4Node.js GET/POST请求

GET的请求方式实际就是通过地址栏参数的形式发送请求的，下面是GET获取地址栏参数的方法：

/

获取get参数

POST请求的内容都在请求正文中。 例如，上传文件，但大多数时候我们不需要注意请求体的内容，因为恶意的POST请求会大大消耗服务器的资源。 所以 node. js默认不解析请求体的，所以当需要解析 post请求体时，需要自己来进行参数的解析工作。

/

获取post请求

2.3Node.js的RESTful API

REST全称是Representational State Transfer，中文意思是表述（编者注： 通常译为表征）性状态转移。 {82%：它首次出现在2000年Roy Fielding的博士论文中，而Roy Fielding是HTTP规范的主要作者之一。} 他在论文中提到： “我写作的目的是理解和评估基于网络的应用程序的架构设计，符合架构原则。 获得功能强大，良好且适合通信的结构。 REST指的是一组架构约束条件和原则。 如果架构符合REST约束和原则，我们称之为RESTful架构。

{63%：REST本身不会创建新技术，组件或服务，RESTful背后的想法是使用Web的现有特性和功能。} {100%：更好地使用现有Web标准中的一些准则和约束。} {68%：尽管REST本身深受Web技术的影响，但理论上REST架构风格并未绑定到HTTP，但HTTP是当前唯一与REST相关的实例。} 所以我们这里描述的REST也是通过HTTP实现的REST。

2.4Node.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。

/

连接mysql数据库

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，要干什么，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。 {54%：博主可以在后台管理用户并添加，删除和修改文章和文章。}

针对博客系统的以上需求，总结出如下信息：

{73%：用户分为游客，普通用户和超级管理用户。}

2.超级管理用户可以管理和设置用户的权限。

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。

{41%：4.用户可以阅读文章和发表评论，访问者只能阅读文章。}

{45%：5.文章类型和文章之间存在一对多的关系，文章和评论之间存在一对多的关系。}

3.23.2 可行性分析

{50%：可行性分析(Feasibility Analysis)是在系统调查的基础上，无论是否有必要开发新系统，从技术，} 经济和社会角度分析和研究新系统的开发，避免投资错误并确保新系统的开发成功。 {60%：可行性研究的目的是确定问题是否可以在最短的时间内以最小的努力解决。} {44%：为了确定开发的可行性，主要分析了系统的以下几个方面。}

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。

系统设计

个人博客的总体规划图如下：

/

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。

/

后台功能设计

{50%：超级管理员通过预设帐户登录后台。} {51%：超级管理员可以在后台添加，删除和修改文章，添加，删除和修改文章，以及查看和删除评论。} 管理注册用户，查看和禁止可执行的操作。

数据库设计

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

{50%：用于数据库设计和管理的可视化数据库管理工}

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。 如下图所以： /

数据库概念结构e-r图

逻辑结构设计

{52%：根据实际需要，基于数据库的概念结构设计，设计了下表结构。}

（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

（2）角色表存储用户角色，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

(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

{64%：（4）商品分类表存储商品分类的具体名称。}

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

{55%：表之间的关系显示在以下数据库模型中：}

数据库模型

接口

http: //www.shirdon.com/? p=426 参考网址

系统实现

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。

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。 {43%：首先，前台用户输入帐号和密码，并将其与数据库用户表中的数据进行比较。} 跳转到博客首页，显示登录状态，您可以对文章发表评论。前台登录如下图：

/

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：

/

后台登录

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。 通过选择分类更改api的参数来获取不同类别的文章。

/

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。

/

后台管理系统

组件及路由设计

本系统使用vue-router进行系统的路由管理。 Vue Router 是 Vue.js 官方的路由管理器。 我们需要将组件映射到路由并告诉Vue路由器在哪里呈现它们。

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。 本系统的路由设计，使用了 vue- router的 history模式与路由嵌套，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。

项目结构图

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，与后台交互以获取数据，我们通常使用axios库，它基于http库的承诺，它在浏览器端和node.js中运行。 他有许多出色的功能，例如拦截请求和响应，取消请求，转换json，客户端防御XSRF等等。

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。{40%：Promise是异步操作的包装器，可以添加到异步操作成功或失败时执行的方法。} 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）和响应拦截（interceptors.response）。

在请求拦截中，判断它是否是需要令牌的接口，并且如果需要，添加令牌，并且区分传递get和post请求参数的方式。

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，存在此标签，代表需要登录。 如果当前已登录，则会跳转到登录页面登录。 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();
}
});
```

文献：

[1]李宇.前后端分离框架在软件设计中的应用[J].无线互联科技.2018, 15(17): 41-42.

[2]汪彤.基于Node.js的图书共享平台的设计与实现[D].北京邮电大学.2018.

[3]梁灏.Vue.js实战[M].清华大学出版社.2017.

[4]麦冬.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.

[5]刘红卫.利用Node.js开发前后端分离的系统——以图书馆地方文献系统为例[J].天津科技.2018, 45(7): 67-70.

[6]茆玉庭.基于Node.js和WebSocket的即时通信系统的设计与实现[D].南京邮电大学.2018.

[7]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.

[8]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.

[9]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.

[10]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.

[11]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

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

[13]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.

[14]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.

[15]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.

[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.