

**本科毕业设计（论文）**

题目**：**基于vue2.x的前端开发者社区

学 院：计算机科学与工程学院

专业班级：计算机科学与技术2017-4班

姓 名：陈振靖

学 号：201701060603

指导教师：沙静

完成日期：2021年5月20日

**BACHELOR'S DEGREE THESIS OF SHANDONG UNIVERSITY**

**OF SCIENCE AND TECHNOLOGY**

**Front-end developer community based on Vue2.x**

College：Shandong University Of Science And Technology

Subject：Computer Science and Technology

Name：Chen Zhenjing

Directed by：Sha Jing Professor

**QINGDAO CHINA**

**郑 重 声 明**

本人呈交的毕业论文，是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本设计（论文）的研究成果不包含他人享有著作权的内容。对本设计（论文）所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确的方式标明。本设计（论文）的知识产权归属于培养单位。

本人签名： 日期：

摘 要

随着知识经济时代的到来，知识在为个人、企业和国家发展中所占的比重也越来越大。在互联网中，难免会产生许多繁琐冗余的知识碎片，让人感觉晕眩和焦虑，对缺乏对知识的筛选则是产生该现象的原因之一，人们缺乏一个帮助自己整理某领域实用的知识或经验的工具。另外，现阶段互联网商业发展着重于提升用户的互联网体验，Web技术的发展产生了一大批前端开发者，因此需要一个平台记录、分享学习或工作中的知识，社区应运而生。然而，国内外的许多社区体系却相当臃肿，充斥着许多不相关的因素。用户对功能和效率的追求经济逐渐发展提高，为用户创建一个简单高效的社区意义重大。

本文结合实际应用，设计并实现了一个基于Vue.js和Node.js的风格简约的社区系统。系统将在线代码编译器等开发工具以及网络论坛整合在一起，用户可以使用在线工具满足开发上的需求，也可以与他人分享开发中的心得，交流前沿技术。旨在为用户提供一个集咨询、交流、分享、操作为一体的在线平台。

**关键词：**社区；在线工具；Node.js；Vue.js

**ABSTRACT**

With the advent of the era of knowledge-based economy, knowledge plays an increasingly important role in the development of individuals, enterprises and countries. On the Internet, it is inevitable to produce a lot of cumbersome and redundant knowledge fragments, which makes people feel dizzy and anxious. One of the reasons for this phenomenon is the lack of knowledge screening. People lack a tool to help them organize practical knowledge or experience in a certain field. In addition, the development of Internet business at the present stage focuses on improving users' Internet experience. The development of Web technology has produced a large number of front-end developers, so a platform is needed to record and share knowledge in study or work, and community arises as The Times require. However, many community systems at home and abroad are rather bloated and full of many unrelated factors. As users' pursuit of function and efficiency grows economically, it is of great significance to create a simple and efficient community for users.

In this paper, a simple community system based on Vue.js and Node.js is designed and implemented based on practical application. The system integrates the development tools such as the online code compiler and the network forum, so that users can use the online tools to meet the needs of development, and also share the experience of development with others and exchange cutting-edge technologies. It aims to provide users with an online platform that integrates consultation, communication, sharing and operation.

**KEY WORDS:** community; online tools; Node.js; Vue.js

**目 录**

[**第1章 绪论** 4](#_Toc72610860)

[1.1 项目背景 4](#_Toc72610861)

[1.1.1 什么是社区 4](#_Toc72610862)

[1.1.2 社区的技术发展 4](#_Toc72610863)

[1.1.3 JavaScript的技术发展 4](#_Toc72610864)

[1.2 系统开发的意义 6](#_Toc72610865)

[1.2.1 提升Node.js实现社区引擎项目的比重 6](#_Toc72610866)

[1.2.2 实现网络社区、在线工具一体化 7](#_Toc72610867)

[1.3 系统开发目标 7](#_Toc72610868)

[1.3.1 可用性 7](#_Toc72610869)

[1.3.2 响应时间 7](#_Toc72610870)

[1.3.3 可操作性 8](#_Toc72610871)

[1.3.4 可扩展性 8](#_Toc72610872)

[1.4 论文组织结构 8](#_Toc72610873)

[**第2章 相关技术和工具综述** 10](#_Toc72610874)

[2.1 Vue.js 10](#_Toc72610875)

[2.2 Node.js 10](#_Toc72610876)

[2.3 Express.js 11](#_Toc72610877)

[2.4 MySQL 12](#_Toc72610878)

[2.5 Docker 13](#_Toc72610879)

[2.6 MVVM设计模式 13](#_Toc72610880)

[**第3章 系统分析** 15](#_Toc72610881)

[3.1 可行性分析 15](#_Toc72610882)

[3.2 需求分析 15](#_Toc72610883)

[3.2.1 登录模块 16](#_Toc72610884)

[3.2.2 在线工具模块 17](#_Toc72610885)

[3.2.3 文章模块 18](#_Toc72610886)

[3.2.4 用户模块 18](#_Toc72610887)

[3.2.5 管理员模块 19](#_Toc72610888)

[3.3 性能规定 20](#_Toc72610889)

[3.3.1 性能需求分析 20](#_Toc72610890)

[3.3.2 安全性需求分析 20](#_Toc72610891)

[3.3.3 服务器端响应时间 21](#_Toc72610892)

[3.3.4 客户端性能需求 21](#_Toc72610893)

[**第4章 系统设计** 22](#_Toc72610894)

[4.1 系统总体架构设计 22](#_Toc72610895)

[4.2 数据库设计 22](#_Toc72610896)

[4.2.1 逻辑结构设计 23](#_Toc72610897)

[4.2.2 物理设计 26](#_Toc72610898)

[4.2.3 触发器设计 29](#_Toc72610899)

[4.3 后台接口设计 30](#_Toc72610900)

[**第5章 系统实现** 32](#_Toc72610901)

[5.1 系统主要功能页面展示 32](#_Toc72610902)

[5.1.1 主页面展示 32](#_Toc72610903)

[5.1.2 开发工具页面展示 32](#_Toc72610904)

[5.1.3 登录模块展示 33](#_Toc72610905)

[5.1.4 社区页面展示 34](#_Toc72610906)

[5.1.5 文章发布页面 35](#_Toc72610907)

[5.1.6 文章详情页面 36](#_Toc72610908)

[5.1.7 用户个人中心页面 37](#_Toc72610909)

[5.1.8 用户信息修改模块 37](#_Toc72610910)

[5.2 系统部分功能实现 38](#_Toc72610911)

[5.2.1 MD5加密 38](#_Toc72610912)

[5.2.2 MarkDown语法解析 38](#_Toc72610913)

[5.2.3 前端数据持久化 38](#_Toc72610914)

[**第6章 系统测试** 39](#_Toc72610915)

[6.1 测试的原理和目的 39](#_Toc72610916)

[6.2 测试方法 39](#_Toc72610917)

[6.3 测试实例 40](#_Toc72610918)

[6.3.1 用户登录 40](#_Toc72610919)

[6.3.2 自动登录 40](#_Toc72610920)

[6.3.3 发布文章 40](#_Toc72610921)

[6.3.4 发表评论 41](#_Toc72610922)

[6.3.5 分类查看社区文章 41](#_Toc72610923)

[6.3.6 删除文章或评论 41](#_Toc72610924)

[6.3.7 查看个人中心 42](#_Toc72610925)

[6.3.8 修改个人信息 42](#_Toc72610926)

[**第7章 总结与展望 44**](#_Toc72610927)

[**参考文献 45**](#_Toc72610928)

[**致谢 46**](#_Toc72610929)

**第1章 绪论**

**1.1** **项目背景**

**1.1.1** **什么是社区**

“社区”，即community，是若干社会群体或社会组织聚集在某一个领域里所形成的一个生活上相互关联的大集体。在网络中，社区则特指网络社区，是一个以成熟社区为内容的大规模性局域网，在该网上交流空间中，包含有BBS/论坛、贴吧、公告栏、个人知识发布、群组讨论、个人空间、无线增值服务等形式。在同一主题的社区中，聚集着具有共同兴趣的社区成员。

在很多年前，人们把论坛说成社区；即使到了今天，人们仍然习惯叫一个论坛为某某社区。到了今天，我们对网上的社区有了更加深入全面的认识，论坛只是构成社区的一部分，是社区中的公共活动和议论的场所，是组织社区活动和体现社区文化特征的平台。

**1.1.2** **社区的技术发展**

调查发现，各大互联网社区系统所使用的语言不尽相同，语言也是各有优劣，技术的选型往往参考流行的技术以及具体的需求。国内一些偏业务的公司会用Java做后台开发比如淘宝，网易，美团等。Python则是一些体量偏小的公司使用，例如豆瓣，知乎等。

表1.1 各大互联网网站系统的实现语言

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 网站 | 成立时间 | 系统实现语言 | 网站 | 成立时间 | 系统实现语言 |
| Stackoverflow | 2008年 | .NET | 必应 | 2009年 | .NET |
| Foursquare | 2009年 | Scala | 新浪微博 | 2009年 | PHP |
| Facebook | 2004年 | PHP | 凤凰网 | 1998年 | PHP |
| 人人网 | 2005年 | Java | 百度 | 2000年 | PHP |
| 豆瓣 | 2005年 | Python | 腾讯 | 1998年 | C++ |
| Twitter | 2006年 | Ruby on Rails | 网易 | 1997年 | Java |

**1.1.3 JavaScript的技术发展**

JavaScript诞生于1995年，最初的主要目的是处理以前由服务器端需要负责的一些没有输入验证操作。随着时间的推移，JavaScript在开发人员的挖掘及传道下，它的一些高级模式也被大多数开发者所接受。

在二十世纪初期，谷歌邮件，即我们熟知的Gmail面世。它很可能是第一个将JavaScript提升到另一个高度的应用。Gmail用JavaScript来动态提取你在在浏览器中查看的内容。原本你需要在点击收信箱内的邮件后等待新的网页加载，但现在谷歌在背景使用JavaScript捕捉你想要浏览的内容然后呈现给你。再也不需要等待一整个新网页加载了，这也就是我们所熟知的Ajax，JavaScript也因此在Web2.0革命中起到了关键作用。一些JavaScript原本所不具有的功能也得到了扩充，如DOM的操作、模块管理、依赖管理、面向对象编程等。

JavaScript 高级程序设计（特别是对浏览器差异的复杂处理），通常很困难也很耗时，为了应对这些调整，许多的 JavaScript (helper) 库应运而生，这些 JavaScript 库常被称为 JavaScript 框架。目前，使用较多的框架有jQuery、Vue.js、Angular.js、React.js等，每个框架自身都拥有着成熟的社区和使用者。

为了将事情提升到一个新高度，瑞安·达尔（Ryan Dahl）在2010年向世界介绍了Node.js，一个可以在浏览器外部执行JavaScript代码的运行环境。现在开发者们可以使用JavaScript进行服务器端脚本编写。作为第一个服务器端的JavaScript环境，这实际上对网景LiveWire Pro Web的一种重新评估。

Node.js如今已是行业内的一股强势力量。Node.js如此成功的原因之一就是做出贡献的庞大的开发人员群体。安装Node.js，不需要编写服务器代码就可以呈现出复杂的web应用程序。

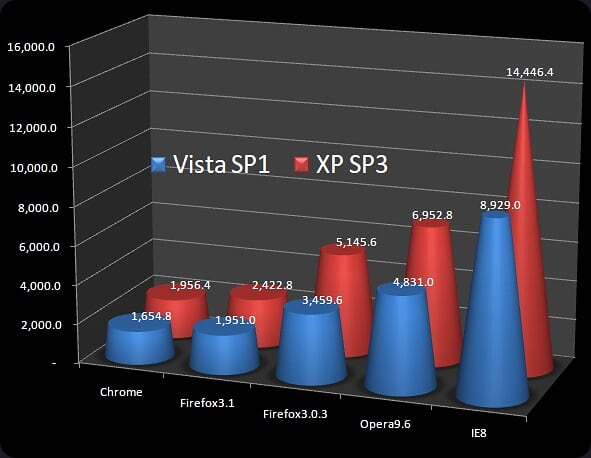
****

图1.1 五款浏览器JavaScript引擎表现（数值越小越快）

目前Node.js和Vue.js的开发已逐渐完善，各种工具的接口的模块也在开发中逐渐完善，但是基于Node和Vue的系统目前还是较少，其中一个就是网络社区。

**1.2** **系统开发的意义**

**1.2.1 提升Node.js实现社区引擎项目的比重**

每当有一种流行语言诞生，各种相关应用便会应运产生，同时，语言的流行程度也影响着应用的多样性。例如，使用PHP实现的社区系统远高于Perl，其社区力量相应的更强。

在近十余年的发展中，JavaScript的社区影响力也逐渐强大。在GitHub中Star数量最多的前50个项目中，基于JavaScript的项目多达14个，JavaScript的繁荣程度可见一斑。

在当今服务器语言百花齐放的背景下，虽然每种开发语言都拥有一大批使用者，但是在总体的占比中，PHP仍是处于垄断地位，占比近80%，基于JavaScript的Node.js在其中可谓是微乎其微。本项目旨在开发一个基于Vue.js和Node.js实现网络社区，完善Node.js的生态环境。

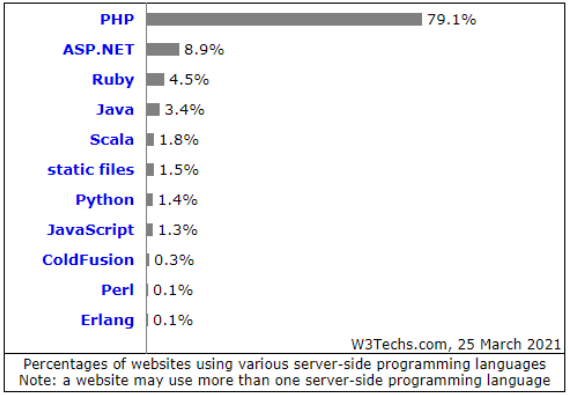


图1.2 2021年3月WEB服务器端语言排行榜

**1.2.2 实现网络社区、在线工具一体化**

在现阶段的互联网商业发展中，提升用户的互联网体验是当下的重要一环，Web技术的发展产生了大批前端开发者。在日常的开发过程中，必然会有各种开发中的心得、问题、经验同他人分享，网络社区便为这些开发人员提供了交换开发经验的场所，使他们能够跨越时空限制，建立有意义的人际关系，将分散在全国各地，对前端开发具有共同兴趣爱好的开发人员聚集在一起。

现如今的程序员社区中，大部分网站只是单一的论坛，为用户提供交流的平台，而在开发中如果需要使用某个开发工具，如JSON解析、进制转换、代码编译等功能，则需要单独使用搜索引起查找对应工具，使得时间都浪费在了无意义的搜索中。因此，在社区内为开发人员提供在线开发工具可以大大节省用户的时间，这也是当下大部分热门社区所没有提供的功能，而提供在线工具的开源中国（OSCHINA）则功能过于繁多，界面设计略显繁杂。本系统希望开发一个集在线工具和网络社区为一体的网站，为开发人员设计一个功能实用、页面简洁的网络社区。

**1.3** **系统开发目标**

**1.3.1 可用性**

确保系统可用性的根基在于基础功能的可用，即：

（1）访客能够：使用工具，注册账号，查看文章、评论和用户信息。

（2）用户能够：使用工具，登陆账号，查看文章、评论和用户信息，创建文章，评论文章，删除自己的评论和文章，修改个人信息，找回密码。

（3）管理员能够：删除评论、删除文章。

在界面中，要保证用户能够找到自己想要的东西，为此，系统提供了多种视角供用户检索，如社区首页、工具列表页、用户信息页等等。

在设计页面时，需保证导航一致性，为此系统将导航与主体内容区分隔，在页面跳转时只重新渲染主体区域。另外还要保证页面风格一致，使用户不会在网站中迷失[[[1]](#endnote-1)]。

UI方面，尽量保证着重显示主要内容，在样式设计方面，尽量维持简洁大方的特点[[[2]](#endnote-2)]。

**1.3.2 响应时间**

响应时间是Web系统性能的一个重要指标，也是应用程序服务质量众多属性中最重要的属性之一。服务质量是一个综合指标，用于衡量使用一个服务的满意程度，描述关于一个服务的某些性能特点。可以看到，服务质量的实质是用户的满意程度。专门就响应时间而言，要使Web应用程序的服务质量满足用户的要求就要使用户的等待时间尽量控制在用户可以接受的范围之内。而用户在使用一个Web系统时，其等待时间通常取决于多种因素。如何准确的描述这些影响因素和响应时间的关系，以及如何有效的控制这些因素并对其进行优化，对于提升系统性能及服务质量至关重要。

因此，我们要将网络请求响应时间控制在1s内，用户在前端让数据填好，通过网络的传输发送给服务器，在服务器处理之后将结果存入数据库，响应时间可以控制在100ms内。

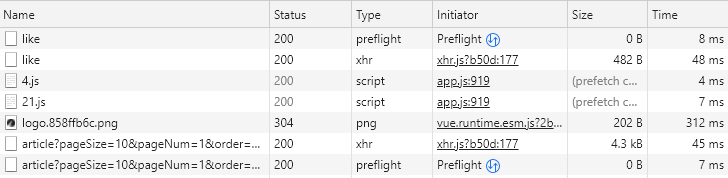


图1.3 部分网络请求的响应时间

**1.3.3 可操作性**

Web的优势在于其可操作性，在开发过程中，应考虑页面操作尽量便捷，使用户在访问中能够容易理解操作步骤，保证用户能够短时间内上手。

**1.3.4 可扩展性**

在开发时，如果要添加新功能或者新模块，应该保证该扩展行为是便捷的、即添即用的。在设计系统时，前端部分采用MVVM设计模式，将视图UI和业务逻辑进行分离，对数据处理、页面展示和逻辑控制进行分割，在后续开发过程中，若果要添加新的功能，只需要嵌入对应代码，无需进行全篇改动。

**1.4** **论文组织结构**

论文组织安排如下：

第一章为绪论，主要介绍网络社区背景，JavaScript技术的发展和系统开发的意义和目标。

第二章介绍了项目中用到的相关技术，包括Vue.js，Node.js，Express.js，Docker，MySQL和MVVM设计模式。

第三章为系统分析，包括可行性分析、需求分析和性能规定。

第四章为系统设计，包含了系统总体架构的设计，数据库的逻辑和物理结构的设计以及服务器端接口的设计。

第五章为系统实现，展示了项目的部分前端页面，同时介绍了开发中一些功能的实现方法。

第六章为系统测试，对系统性能进行了综合测试。

最后则是对系统的总结，以及参考文献的引用和致谢。

**第2章 相关技术和工具综述**

**2.1** **Vue.js**

Vue.js，简称vue，是一套可以专注于构建一个用户管理界面的渐进式Javascript框架，具有一种高性能、灵活性强的特点。Vue遵循渐进递增的设计原则，便于整合到现有项目中。这一特性使vue能够快速构建小型原型和复杂的大规模网络应用程序[[[3]](#endnote-3)]。

Vue.js并不是一个入侵式前端框架库，在与其他库进行配合的方面提供了更大的空间。例如，Vue.js核心默认不包含路由和Ajax功能，另外在应用中一般假设使用了一个模块构建系统。而Angular.js和React.js都过于庞大和全面，两者涉及了一个页面应用的所有阶段和所有功能，一旦使用，就需要抛弃现有的成熟框架，否则基础功能均导致重复冗余，降低客户的体验效果。而更换基础框架的学习和维护成本巨大。

响应数据系统作为VUE的核心功能之一，通过轻量级的虚拟DOM引擎可以自动高效地更新用户界面，避免了以往DOM命令式操作带来的弊端，简化了前端开发过程和开发效率，大大提高了系统后期的可维护性。麦冬、陈涛和梁宗湾在轻量级响应式框架Vue.js应用分析[[[[4]](#endnote-4)]]也说到，大数据驱动时代取代了传统繁琐的dom驱动程序，同时也避免了大量的选择器复杂程序，改革后只需要对数据的源头信息进行关注，能够有效的缩短开发的用时，降低开发的难度，提升用户的使用效果。

**2.2** **Node.js**

Node.js是一个基于Chrome V8引擎的JavaScript运行时环境，旨在构建可扩展的 Web 应用程序。凭借其事件驱动、非阻塞I/O模型的特性，Node.js成为了一个轻量和高效的语言，其成为在分布式设备上运行数据密集型实时应用程序的理想选择。npm（Node package manager）是Node.js的包管理器，解决了Node.Js平台代码部署问题，使用npm命令，开发人员能够轻松下载、安装、卸载、升级、查询第三方软件包。在Node.js平台上进行开发Web应用，还能通过统一前后端系统开发的编程语言，减少开发者的学习成本。

例如，在登录模块设计时，参考杨小彦的基于TOKEN令牌的身份校验方法及相关设备[[[5]](#endnote-5)]文章，服务器要对用户登录生成一个有时效的TOKEN令牌，进行授权操作，此时便可以使用npm导入jsonwebtoken模块来生成TOKEN。同时在数据库存储密码时，出于安全的考虑，需要对密码进行加密，此时后端采用了The MD5 Message-Digest Algorithm[[[6]](#endnote-6)]里提到的MD5加密算法，便可以使用npm引入md5模块或crypto模块进行密码加密。

虽然让JavaScript运行于服务器端不是Node的独特之处，但却是一个强大的功能。诚然，浏览器环境限制了我们选择编程语言的自由。任何一个服务器与日益发展复杂的浏览器客户端技术应用系统程序间共享代码的愿望必须通过JavaScript来实现。虽然还有其他平台支持服务器端运行JavaScript，但是Node由于上述的特点，发展日益迅速。

近年来，Node.js不断发展，已经成为了一个比较成熟的开发平台，吸引着大量的开发者。有许多大型网站也都开始选择Node.Js用于开发。此外，开发人员还可以用它来开发一些快速移动的Web框架。

下图是普通Web服务器与Node的工作方式区别。

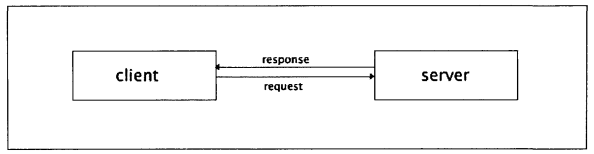


图2.1 普通服务器的工作方式

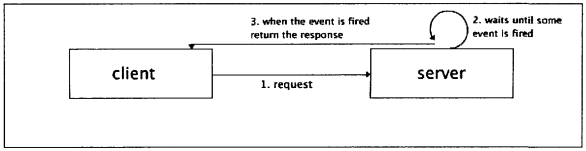


图2.2 Node的工作方式

**2.3** **Express.js**

后台框架选用Express，它是一个简洁而灵活的Node.js Web应用框架, 它具有一系列强大特性，基于此，用户可以建各种Web应用。Express有着丰富的HTTP工具以及随取随用的来自Connect框架的中间件，可以快速、简洁地创建强健、友好的API。另外，Express不对Node.js现有的特性进行二次抽象，只是在开发过程中，对Web应用所需的功能进行扩展。

NodeJS上的ExpressJS框架可以方便实现众多功能，新建Server配置，设置路由，中间件，错误处理，参数预处理，渲染，Partials，View查找，模板引擎合作等众多功能[[[7]](#endnote-7)]。

**2.4** **MySQL**

MySQL是当下最热门的关系型数据库，在WEB应用方面有很多RDBMS（Relational Database Management System：关系数据库管理系统）应用软件，MySQL便是其中之一。MySQL是由瑞典MySQL AB公司所开发，目前隶属Oracle公司。MySQL不是将所有数据放在同一仓库内，相反而是将数据保存在不同的表中，使用这种存储方式，MySQL的速度和灵活性得以提升。

MySQL所谓的"关系型"可以理解为"表格"的概念, 一个关系型数据库由一个或数个表格组成，如下图所示：

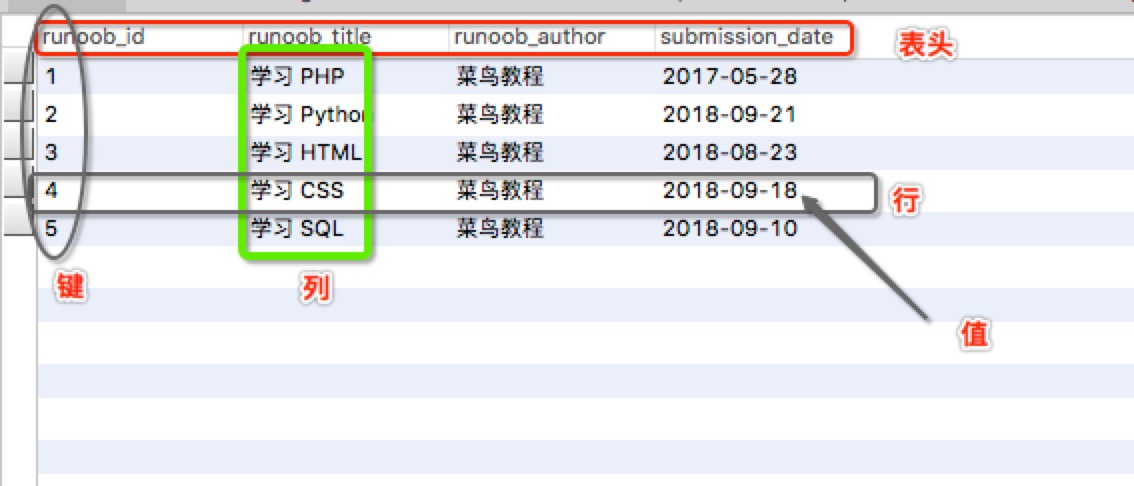


图2.3 MySQL表格中的定义

下面是MySQL所具有的特点：

（1）MySQL 是开源的，目前隶属于 Oracle 旗下产品。

（2）MySQL 支持大型的数据库，可以处理拥有上千万条记录的大型数据库。

（3）MySQL 使用标准的 SQL 数据语言形式。

（4）MySQL 可以运行于多个系统上，并且支持多种语言。这些编程语言包括 C、C++、Python、Java、Perl、PHP、Eiffel、Ruby 和 Tcl 等。

（5）MySQL 支持大型数据库，支持 5000 万条记录的数据仓库，32 位系统表文件最大可支持 4GB，64 位系统支持最大的表文件为8TB。

（6）MySQL 是可以定制的，采用了 GPL 协议，你可以修改源码来开发自己的MySQL 系统。

**2.5** **Docker**

Docker是一个开源的应用容器引擎，基于Go语言 并遵从Apache2.0协议开源。Docker可以让开发者打包他们的应用以及依赖包到一个轻量级、可移植的容器中，然后发布到任何流行的Linux机器上，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口（类似iPhone的app），更重要的是容器性能开销极低。

Docker是一个用于开发，交付和运行应用程序的开放平台。Docker能够将应用程序与基础架构分开，从而可以快速交付软件。借助 Docker，开发中可以与管理应用程序相同的方式来管理基础架构。通过利用 Docker的方法来快速交付，测试和部署代码，可以大大减少编写代码和在生产环境中运行代码之间的延迟。

Docker具有下述优点：

（1）快速，一致地交付应用程序：

Docker允许开发人员使用您提供的应用程序或服务的本地容器在标准化环境中工作，从而简化了开发的生命周期。

（2）响应式部署和扩展：

Docker是基于容器的平台，允许高度可移植的工作负载。Docker容器可以在开发人员的本机上，数据中心的物理或虚拟机上，云服务上或混合环境中运行。

Docker的可移植性和轻量级的特性，还可以使您轻松地完成动态管理的工作负担，并根据业务需求指示，实时扩展或拆除应用程序和服务。

（3）在同一硬件上运行更多工作负载

Docker轻巧快速。它为基于虚拟机管理程序的虚拟机提供了可行、经济、高效的替代方案，因此可以利用更多的计算能力来实现业务目标。Docker非常适合于高密度环境以及中小型部署，从而可以用更少的资源做更多的事情。

**2.6** **MVVM设计模式**

随着前端页面越来越复杂，用户对于交互性要求也越来越高，在开发中jQuery已经不能满足正常需求。MVVM模型应运而生。MVVM最早由微软提出来，它借鉴了桌面应用程序的MVC思想，在前端页面中，把Model用纯JavaScript对象表示，View负责显示，两者做到了最大限度的分离。把Model和View关联起来的就是ViewModel。ViewModel负责把Model的数据同步到View显示出来，还负责把View的修改同步回Model。

数据驱动是MVVM的核心[[[8]](#endnote-8)]，即ViewModel，它是View和Model的关系映射。类似中转站(Value Converter)，ViewModel负责转换Model中的数据对象，使得数据变得更加易于管理和使用。本质上，MVVM就是基于操作数据来操作视图进而操作DOM，借助于MVVM无需直接操作DOM，开发者只需完成包含声明绑定的视图模板，编写ViewModel中有业务，使得View完全实现自动化。

在vue中，ViewModel做了如下两件事：

（1）数据绑定：把js里面的数据通过插值操作mustachae语法绑定到真实dom里面 而且是响应式的 一旦数据发生改变，自动监听到数据的改变，把最新的数据再重新绑定到view上面，view里面显示得永远是model里面最新的数据。

（2）DOM监听：当view上面有一些事件或者用户操作的时候，通过指令，把事件响应相关的东西，绑定到model里面，在model里面可以写一些js代码（比如回调methods里面定义的一些函数）。

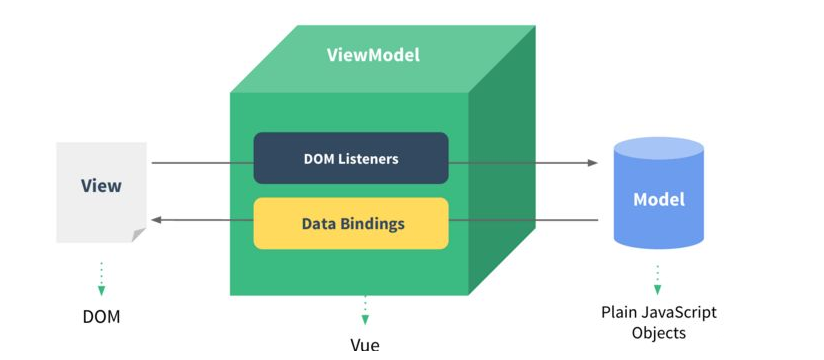


图2.4 vue中的MVVM设计模式

**第3章 系统分析**

**3.1** **可行性分析**

所谓可行性分析，便是对项目进行调查研究和分析比较，从技术、经济、工程等角度进行兼顾对比，同时需要科学的预测项目建成以后的结果，包含财务、经济效益和社会环境影响等方面，为项目决策提供一些公正、可靠、科学的咨询意见。需要对经济、技术、社会环境等方面进行分析得出结果，然后分析解决方案是否可行，在最终分析解决方案后，如果项目可行，并且有一定的经济效益和社会效益，便可以开始计算机的系统的开发。

本文主要讨论技术可行性，主要以项目开发的技术角度，设计合理的技术方案，进行分析、比较和评价，是指决策的技术和决策方案的技术不能突破组织所拥有的或有关人员所掌握的技术资源条件的边界。

前端部分，我们使用vue框架，在前端发展迅猛的今天，vue作为GitHub上stars数量排名第四的项目，有着越来越多的基于vue封装的组件供项目开发使用，例如本项目所使用的富文本编辑器、代码编辑器、JSON解析等组件，均能找到兼容vue的项目通过npm导入，大大缩短了开发周期。

后端方面，Node.js提供了诸多库，项目可以使用Express进行新建Server服务、配置路由、中间件、参数预处理、错误处理等功能；使用orm2（Object Relational Mapping）进行服务端与数据库的连接操作，它可以创建模型，同步，删除，批量创建，获取，查找，移除，计数，聚合函数，还可以创建模型的关联，查找，检查，创建和移除。

基于此，本系统的开发是完全可行的。

**3.2** **需求分析**

需求分析是软件计划阶段的重要活动，也是软件生存周期中的一个重要环节需求分析的目标是把用户对待开发软件提出的“要求”或“需要”进行分析与整理，确认后形成描述完整、清晰与规范的文档，确定软件需要实现哪些功能，完成哪些工作。

在3.1中，对项目的可行性分析的过程中，已经简单地分析了项目的基本需求，也讨论了开发平台和软件的可行性，但为了清楚系统设计目标，现对子系统进行进一步分析。

系统功能结构图如图3.1所示，其中包括登录功能界面、在线工具界面、用户相关界面、文章相关界面和管理员模块。

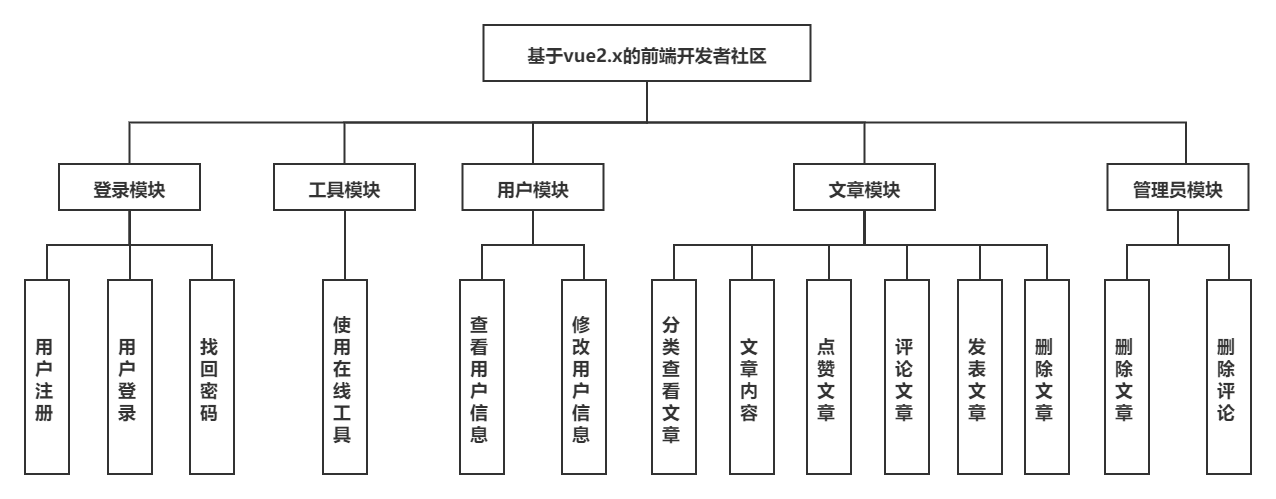


图3.1 系统功能结构图

**3.2.1** **登录模块**

本系统一共分为三种用户角色，分别是用户、游客和管理员。用户具有对文章的操作功能，如点赞、评论和删除，而访客仅可以查看文章，因此系统加入了登录模块，允许游客输入相关信息进行账号的注册和登录，登录成功便可以进行相关操作。使用账号来对访客进行功能限制，可以防止非法用户对数据进行篡改，进而保证了系统安全。

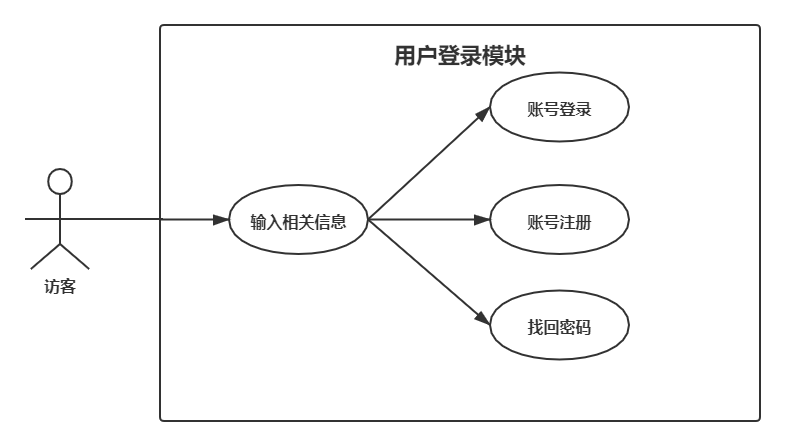


图3.2 用户登录模块用例图

图3.3是用户登录模块详细设计流程图：

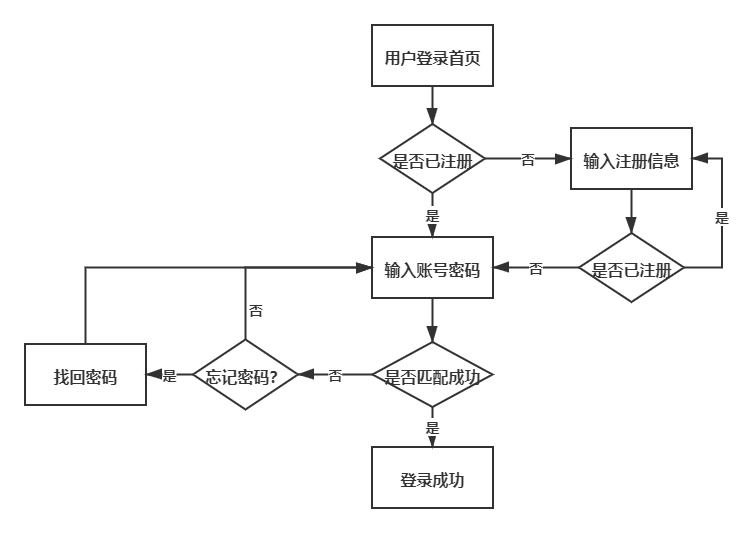


图3.3 用户登录模块流程图

如图3.3所示，访客进入登录模块时，进入登录主页，如果用访客拥有已注册的账号，则用户可以输入账号密码进行登录，否则需要前往注册页面进行注册，而注册时如果账号或昵称重复，则需要重新输入注册信息。在输入账号密码进行登录时，如果账号或密码验证错误，则需要重新输入。考虑到用户可能会忘记自己的账号密码，系统提供了找回密码的功能，用户只需输入账号和绑定邮箱，系统便会向该邮箱发送6位数的验证码，在匹配成功后，便可以修改密码进行登录。登录成功后，用户便可以进行一系列授权操作。

**3.2.2** **在线工具模块**

本系统的主要功能之一就是在线工具，每一个进入网站的访客，都可以进行在线工具的使用，无需登录。由于技术原因，在线编译器部分系统借用了菜鸟教程的现有接口。

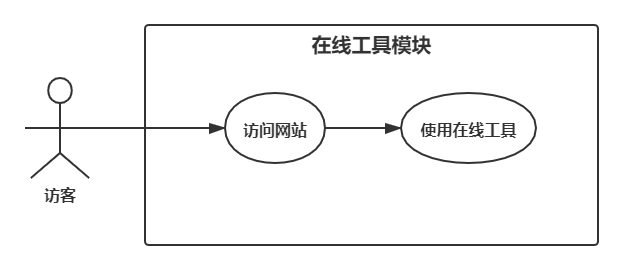


图3.4 在线工具模块用例图

**3.2.3** **文章模块**

系统为用户提供了文章的增删改查功能（CRUD）。

（1）分类查看文章：通过选定的过滤、排序等条件，分类查看社区内的文章。

（2）发表文章：在选择文章类型后，发表新文章。

（3）显示文章内容：显示文章的信息，包括文章内容、文章评论、点赞数、评论数和作者信息等，文章内容和评论以MarkDown格式显示。

（4）点赞文章、评论：用户可以对非本人发布的文章和评论进行点赞。

（5）回复文章、评论：用户可以在文章下方进行评论，同时也可以对已有的评论进行回复。

（6）删除文章、评论：用户可以删除本人的文章和评论。如果删除文章，则文章下的评论也会删除；如果删除评论，其子评论也会被删除。

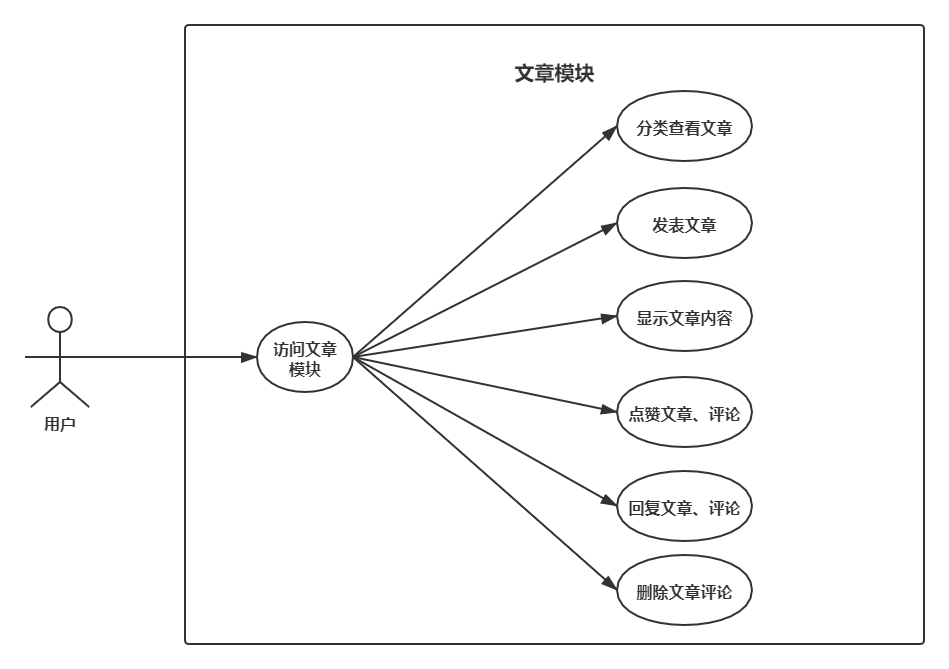


图3.5 文章模块用例图

**3.2.4** **用户模块**

用户在登录后，可以访问个人主页，在个人主页中，可以查看用户的一系列信息，包含发表过的文章、注册时间、最后活跃时间、个性签名等内容。另外，在设置页面，用户可以修改自己的头像、个人信息、登录密码和绑定手机。

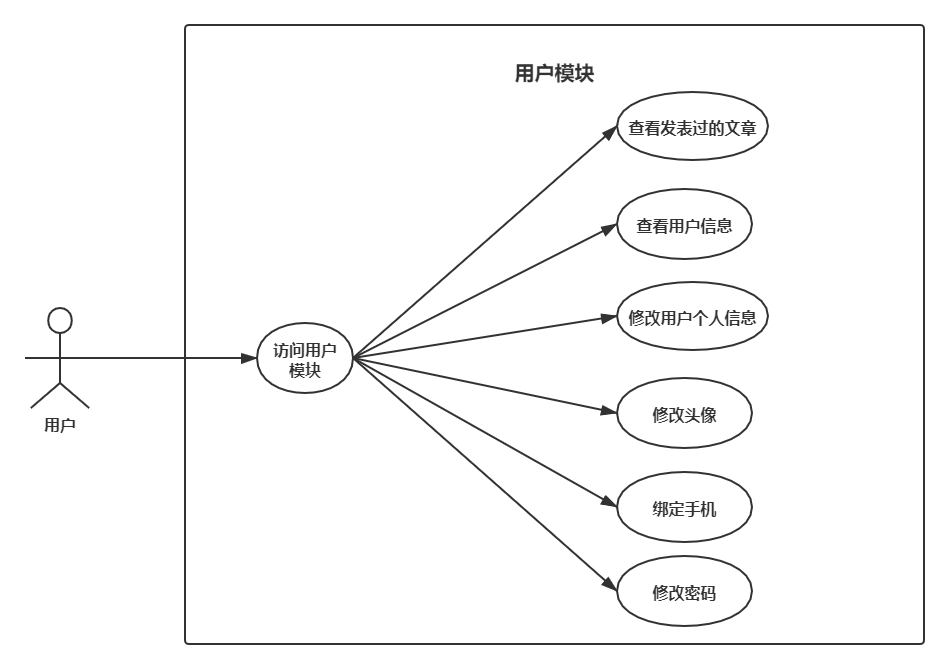


图3.6 用户模块用例图

**3.2.5** **管理员模块**

社区提供了管理员角色，在浏览社区内容时，如果发现了不良文章或评论，管理员可以对其进行删除操作。如果删除文章，则文章下的评论也会被删除；如果删除评论，其子评论也会被删除。

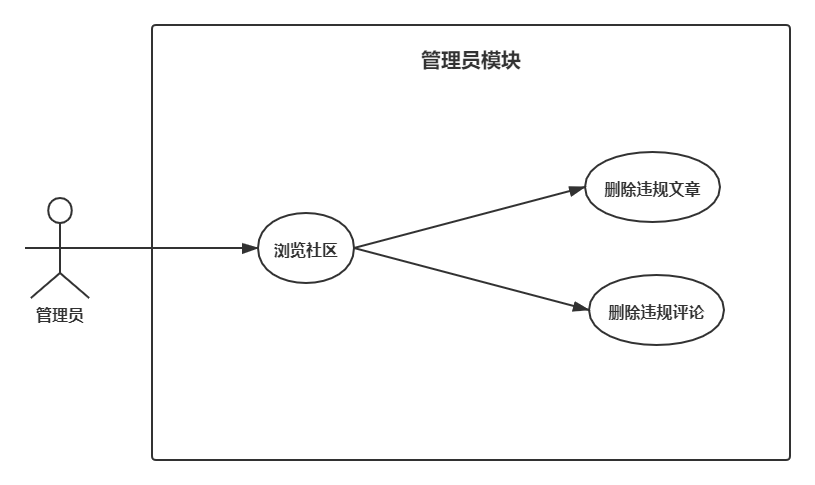


图3.7 管理员模块用例图

**3.3** **性能规定**

**3.3.1** **性能需求分析**

本系统性能要求可以满足用户数量10000，同时在线人数300人，平均的服务响应时间不超过4秒，在高峰期的响应时间不能超过8秒。

对于项目的并发需求，由于系统采用Node服务器，Node.js通过事件循环机制（初始化和回调）的方式运行 JavaScript 代码，并且提供了一个线程池处理诸如文件 I/O 等高成本的任务。Node的伸缩性非常好，某些场景下它甚至比类似 Apache等更重量级的解决方案表现更优异。Node可伸缩性的秘诀在于它仅使用了极少数的线程就可以处理大量客户端连接。如果 Node.js只需占用很少的线程，那么它就可以将更多的系统 CPU 时间和内存花费在客户端任务而不是线程的空间和时间消耗上（内存，上下文切换）。

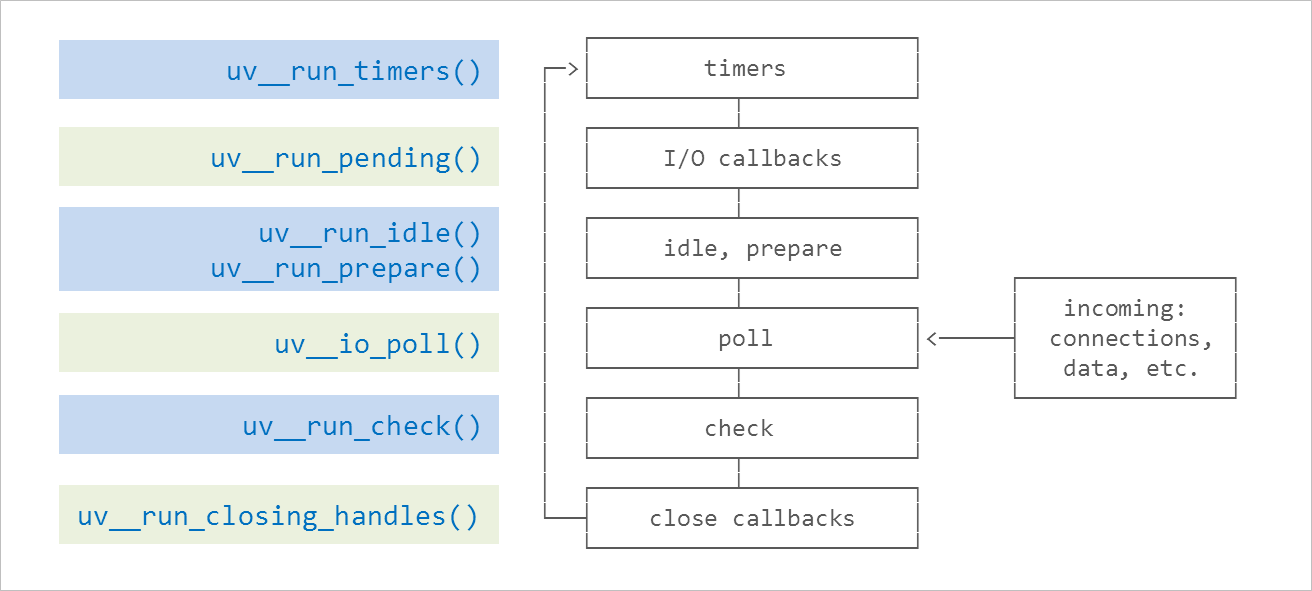


图3.8 Node.js的事件循环机制

**3.3.2** **安全性需求分析**

在安全性需求方面，在用户操作时要进行验证和授权。在服务端，系统使用了jwt（jsonwebtoken），使用户在登录后，生成一个具有时效存储着用户id和身份的特定token，发送给前端，前端在接收后将其保存在浏览器的cookie中。在用后续的操作中，每一次对服务端的请求，都将该token放在request的Authorization中，后端服务器对请求头中的token进行解析，如果解析成功，则进行后续的授权操作，否则没有授权，无法进行后面的操作。通过Express.js的中间件，可以实现用户授权，对用户进行功能限制，可以防止非法用户对数据进行篡改，进而保证了系统安全。

在数据安全方面，系统对用户密码进行了md5加密处理，服务器将加密后的密码存入数据库，在处理后，即使是数据库管理者也不能知道用户的密码，保证了用户信息的安全性。另外也要防止SQL注入，因为系统使用了orm框架进行服务器与数据库的交互，该框架对接口进行了封装，不对外暴露，因此在理论上有效地对SQL注入攻击进行了防范。

**3.3.3** **服务器端响应时间**

如1.3.2响应时间中所提，如果要提升Web应用程序的服务质量，就需要使用户的等待时间尽量控制在用户可以接受的范围之内，因此，在开发中保证应保证请求响应时间少于1秒。

**3.3.4** **客户端性能需求**

对Web服务来说，前端体验也是评判性能的重要一环，而不仅仅是后端。因此在前端设计时，应将网页尽可能设计为响应式布局，使得网页在浏览器不同的分辨率下，都能有很好的显示效果。另外在系统构建时，也应考虑参照闫志英[[[9]](#endnote-9)]提出的前端性能优化的解决方案：

（1）减少HTTP的请求

（2）对样式表进行改善

（3）压缩文件

（4）将script放在底部

（5）减少DNS查询

（6）避免重定向，杜绝http 404错误

另外，Goel U和Steiner M[[[10]](#endnote-10)]在2020年发表的一篇文章中也提出，系统可以识别和删除多余的JavaScript代码，从而更快地加载网页。

**第4章 系统设计**

**4.1** **系统总体架构设计**

本系统采用三层架构模式。分为前台页面、业务逻辑和数据结构，其分布在各个层次，由于各层次间耦合性很低，所以维护成本很低。

此模型与MVVM设计模式吻合，View层负责视图逻辑，Model层负责处理数据逻辑，ViewModel层负责处理业务逻辑，为View层提供数据源。MVVM设计模式的具体特点在本文相关技术选型的2.5部分已做介绍。

本系统的整体架构如图4.1所示：

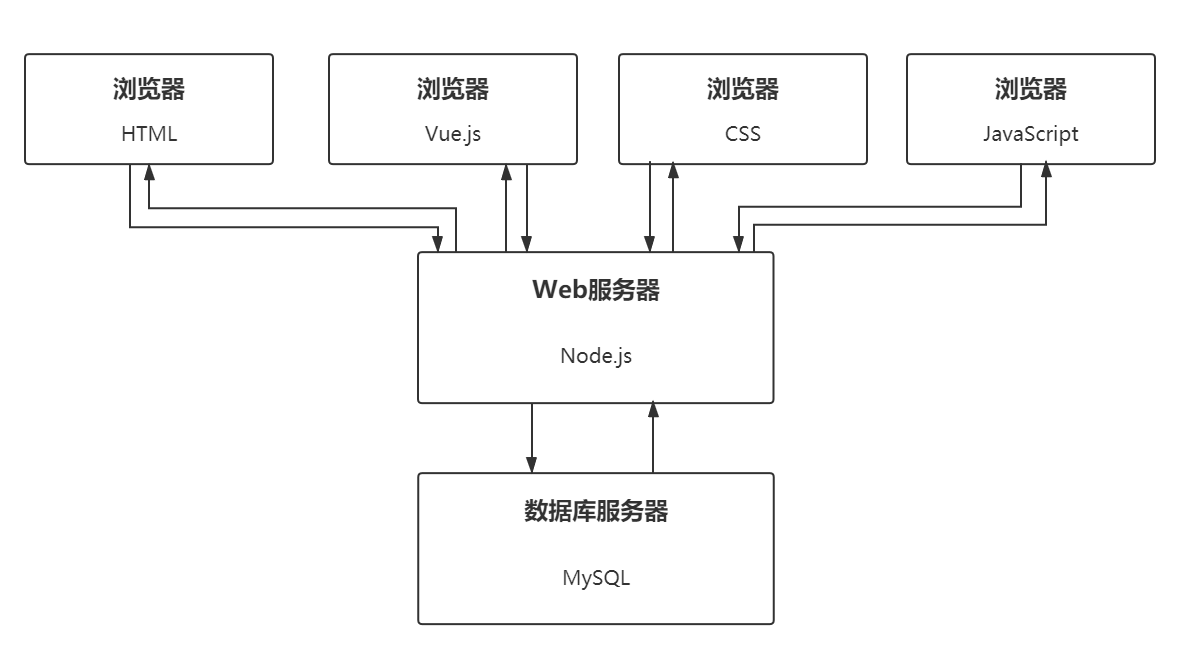


图4.1 系统架构设计图

系统的前端技术栈使用了HTML+CSS+JavaScript+Vue.js+Axios来实现基本的逻辑和页面设计。其中，Vue.js用来构建数据驱动的Web界面，HTML+CSS用来完成页面的排版和美化，JavaScript用来处理前端的业务逻辑，采用Axios库来进行浏览器对服务器接口的请求。系统后端使用了Node.js框架，数据库的选取则选择了MySQL。

**4.2** **数据库设计**

数据库设计(Database Design)是指根据用户需求，设计数据库结构，在特定的数据库管理系统上建立数据库的过程。数据库系统需要操作系统的支持。

数据库进行设计是建立一个数据库及其应用系统的技术，是信息网络系统开发和建设中的核心企业技术。由于数据库应用系统的复杂性，为了支持相关程序的运行，数据库设计变得极其复杂，因此最好的设计不在一朝一夕，而只能是一个"反复探索、逐步细化"的过程，即规划和构建数据库数据对象与这些数据对象之间的关系的过程。

**4.2.1** **逻辑结构设计**

逻辑结构是独立于任何一种数据模型的，在实际应用中，一般所用的数据库环境已经给定（如SQL Server或Oracle或MySQL)。由于目前使用的数据库基本上都是关系数据库，因此首先需要将实体-关系图转换为关系模型，然后根据具体数据库管理系统的特点和限制转换为指定数据库管理系统支持下数据模型，最后进行优化。1、用户实体，具体示意图如图4.2所示：

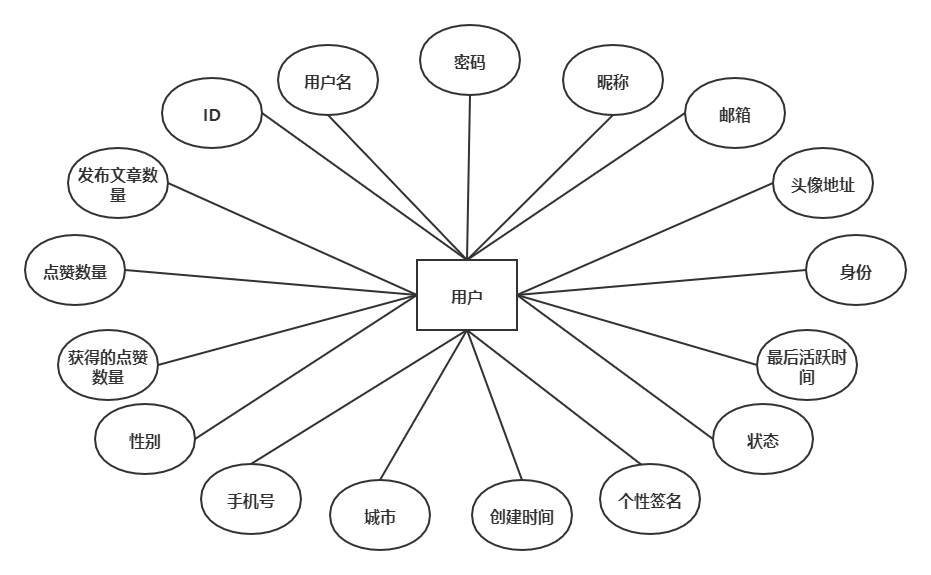


图4.2 用户实体图

通过图4.2，系统的用户实体包括的属性有id、用户名、昵称、密码、邮箱、头像地址、创建时间等。

2、文章实体，具体示意图如图4.3所示：

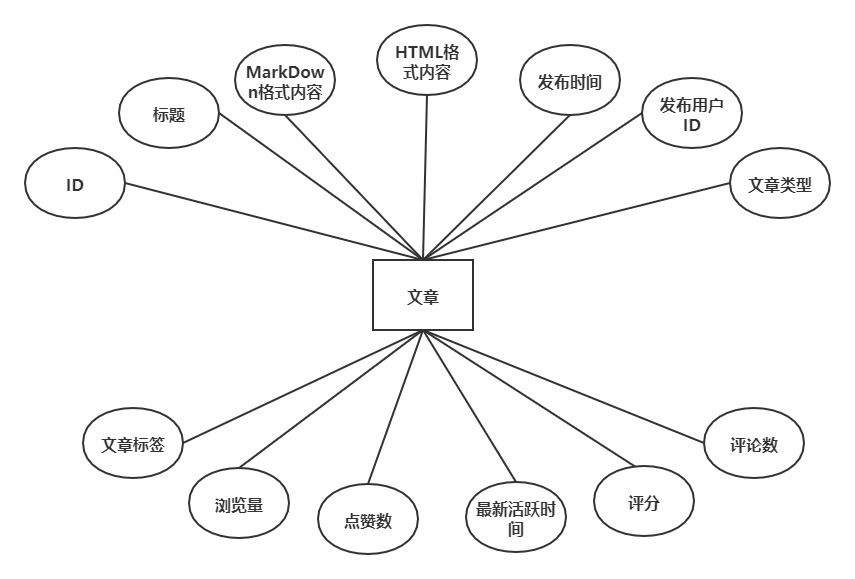


图4.3 文章实体图

通过图4.3，系统的文章实体包括的属性有id、标题、MarkDown格式的内容、HTML格式的内容、发布时间、文章类型等。

3、评论实体，具体示意图如图4.4所示：

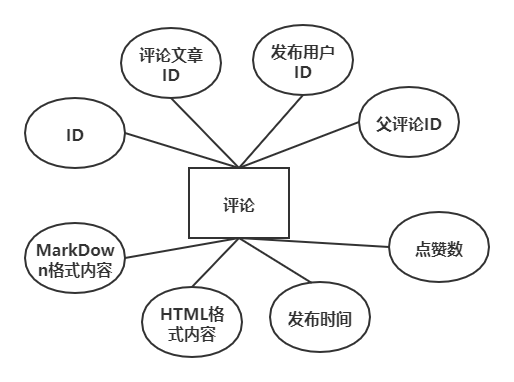


图4.4 评论实体图

通过图4.4，系统的评论实体包括的属性有id、评论文章的id，发表用户的id、MarkDown格式的内容、HTML格式的内容、发布时间等。

4、点赞实体，具体示意图如图4.5所示：

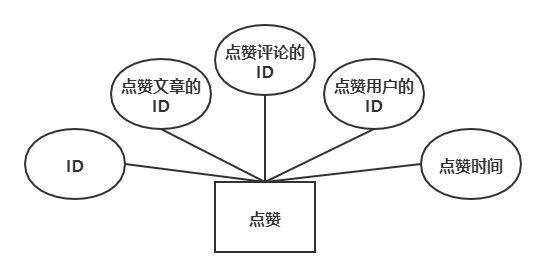


图4.5 点赞实体图

通过图4.5，系统的点赞实体包括的属性有id、点赞文章的id，点赞用户的id、点赞评论的id和点赞时间。

5、验证码实体，具体示意图如图4.6所示：

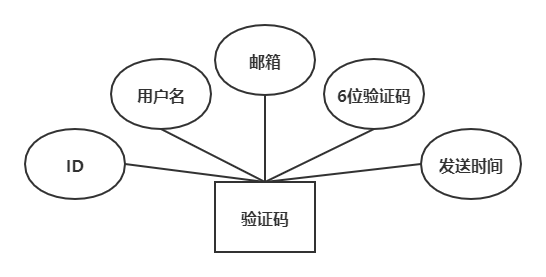


图4.6 验证码实体图

通过图4.6，系统的验证码实体包括的属性有id、接收验证码的用户名、邮箱、6位验证码和发送时间。

依照上述实体类的设计，得出社区系统的相关实体有用户、文章、评论、点赞和验证码实体，实体与实体之间的关系图如下所示：

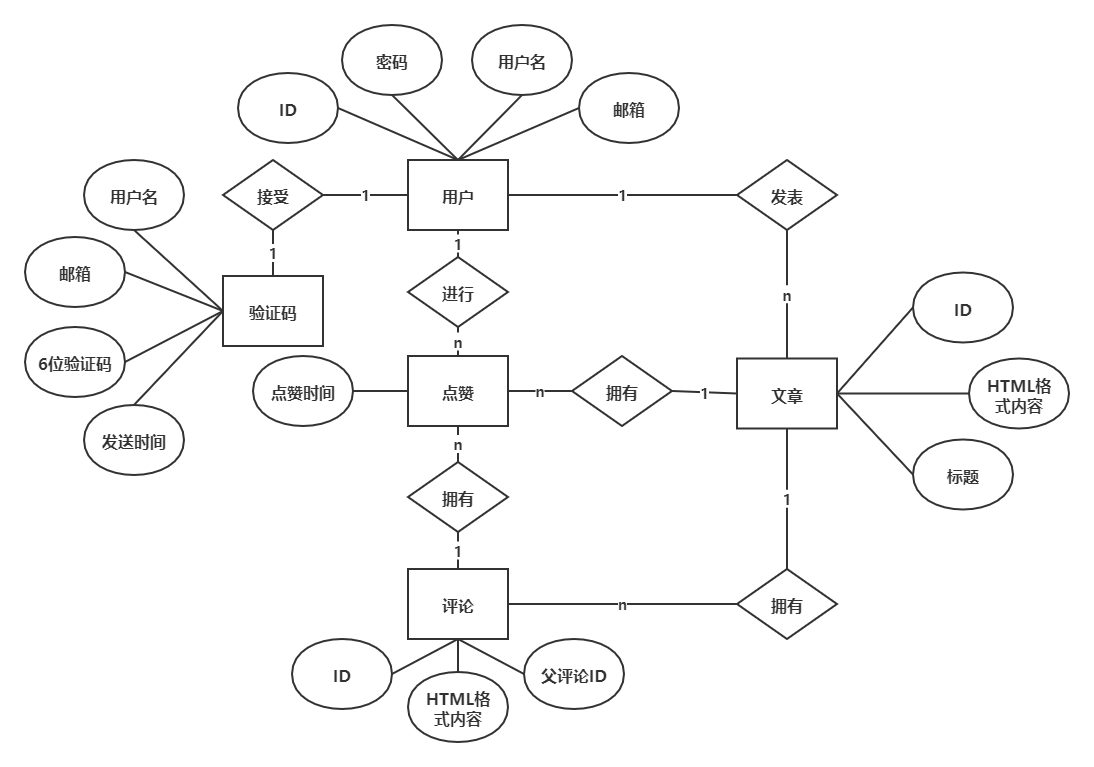


图4.7 系统总E-R图

**4.2.2** **物理设计**

数据库物理设计是后半段。将一个给定逻辑结构实施到具体的环境中时，逻辑数据模型要选取一个具体的工作环境，这个工作环境提供了数据存储结构与存取方法，这个过程就是数据库的物理设计。

1、用户表（user）：该表用来存放用户的各种信息，比如账号、密码、头像地址、最后活跃时间、文章数量等。

表4.1 用户信息表（user）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据结构 | 主键 | 备注 |
| id | int(11) | Y | 用户id，主键 |
| username | varchar(50) | N | 用户名 |
| password | varchar(50) | N | 用户密码 |
| nickname | varchar(255) | N | 昵称 |
| email | varchar(50) | N | 邮箱 |
| gender | tinyint(2) | N | 性别 |
| phone | varchar(255) | N | 手机号 |
| city | varchar(255) | N | 城市 |
| description | varchar(255) | N | 个性签名 |
| state | tinyint(2) | N | 状态 |
| identify | varchar(255) | N | 身份 |
| create\_time | timestamp | N | 创建时间 |
| avatar\_path | varchar(255) | N | 头像路径 |
| article\_count | int(11) | N | 发表文章数量 |
| like\_count | int(11) | N | 点赞数量 |
| acquired\_like\_count | int(11) | N | 收到的点赞数 |
| active\_time | timestamp | N | 最后活跃时间 |

2、文章表（article）：该表用来存放文章的各种信息，比如标题、内容、发表用户的ID、发布时间、文章类型等。

表4.2 文章信息表（article）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据结构 | 主键 | 备注 |
| article\_id | int(11) | Y | 文章id，主键 |
| article\_title | text | N | 文章标题 |
| article\_md\_content | longtext | N | MarkDown格式的文章内容 |
| article\_html\_content | longtext | N | HTML格式的文章内容 |
| article\_create\_time | timestamp | N | 文章发布时间 |
| user\_id | int(11) | N | 发表文章的用户的ID |
| article\_type | enum | N | 文章类型 |
| article\_label | varchar(255) | N | 文章标签 |
| article\_views | int(11) | N | 文章浏览量 |
| article\_comment\_count | int(11) | N | 文章评论数 |
| article\_update\_time | timestamp | N | 最新更新时间 |
| article\_likes | int(11) | N | 点赞量 |
| article\_score | int(11) | N | 文章得分 |

3、评论表（comment）：该表用来存放评论的各种信息，比如内容、发表用户的ID、父评论的ID、评论的文章的ID等。

表4.3 评论信息表（comment）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据结构 | 主键 | 备注 |
| comment\_id | int(11) | Y | 评论id，主键 |
| user\_id | int(11) | N | 发表评论用户的ID |
| article\_id | int(11) | N | 评论的文章的ID |
| comment\_parent\_id | int(11) | N | 父评论的ID |
| comment\_md\_content | text | N | MarkDown格式的评论内容 |
| comment\_html\_content | text | N | HTML格式的评论内容 |
| comment\_create\_time | timestamp | N | 评论的发表时间 |
| comment\_likes | int(11) | N | 点赞量 |

4、点赞表（like）：该表用来存放点赞的各种信息，比如点赞文章的ID、点赞用户的ID、点赞评论的ID、点赞时间等。

表4.4 点赞信息表（like）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据结构 | 主键 | 备注 |
| like\_id | int(11) | Y | 点赞id，主键 |
| article\_id | int(11) | N | 点赞文章的ID |
| comment\_id | int(11) | N | 点赞评论的ID |
| user\_id | int(11) | N | 点赞用户的ID |
| like\_create\_time | timestamp | N | 点赞时间 |

5、验证码表（email）：该表用来存放发送的验证码的各种信息。

表4.5 验证码信息表（email）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据结构 | 主键 | 备注 |
| id | int(11) | Y | 验证码id，主键 |
| username | varchar(255) | N | 接收验证码的用户名 |
| email | varchar(255) | N | 接收验证码的邮箱 |
| code | varchar(6) | N | 6位验证码 |
| create\_time | bigint(20) | N | 验证码发送时间（13位时间戳） |

6、编译信息表（compile）：该表用来存放编译相关信息，如编译后的文件后缀、默认显示的代码、是否存在输入等。

表4.6 编译信息表（compile）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据结构 | 主键 | 备注 |
| id | int(11) | Y | 编译信息id，主键 |
| name | varchar(50) | N | 编译的语言名 |
| title | varchar(255) | N | 显示的标题 |
| description | varchar(255) | N | 详细描述 |
| runcode | int(11) | N | 编译时的运行代码（均不同） |
| path | varchar(50) | N | 图片名 |
| mode | varchar(255) | N | 用来匹配代码输入框的CSS |
| fileext | varchar(255) | N | 编译文件的后缀名 |
| code | varchar(512) | N | 初始代码 |
| hasStdin | int(11) | N | 是否需要输入 |

**4.2.3** **触发器设计**

触发器（trigger）是数据库提供的一种方法，用来保证数据完整性，它是与表事件相关的特殊的存储过程，它的执行不是由程序调用，也不是手工启动，而是由事件来触发，比如当对一个表进行操作（insert，delete，update）时就会激活它执行。触发器经常用于加强数据的完整性约束和业务规则等。触发器可以从 DBA\_TRIGGERS，USER\_TRIGGERS数据字典中查到。

触发器功能强大，轻松可靠地实现许多复杂的功能，但同时也要慎用触发器。触发器本身没有过错，但滥用可能会造成数据库及应用程序的维护困难。在数据库操作中，系统可以通过关系、触发器、存储过程、应用程序等来实现数据操作…… 同时规则、约束、缺省值也是保证数据完整性的重要保障。如果对触发器过分的依赖，势必影响数据库的结构，同时增加了维护的复杂程度。

在本项目中，多表之间存在一定程度的耦合，在项目中一些复杂的业务规则或要求（如当给用户点赞，本质上是向like表插入一条新的记录，此时首先需要判断是给文章还是评论点赞，判断之后需要给对应的文章或评论表里的对应记录中的点赞量加1，最后还要将user表的点赞量加1），这些需求如果在后端使用一条条的数据库请求操作进行更新，会让Service层过于臃肿，针对这些牵一发而动全身的数据库操作，系统选用触发器更为方便。



图4.8 在Navicat中创建触发器

**4.3** **后台接口设计**

本系统是使用vue+node前后端分离的开发模式，在前后端分离的开发模式中，后端仅返回前端所需的数据，前端负责渲染html页面，后端不再控制前端的效果，用户看到什么样的效果，从后端请求的数据如何加载到前端中，都由前端自己决定，后端仅仅需要提供一套逻辑对外提供数据即可，并且前端与后端的耦合度相对较低，在这种模式中，开发人员通常将后端开发的每个视图都成为一个接口，或者API，前端通过访问接口来对数据进行增删改查。综上所述，后台负责提供数据，前端负责数据展示，职责分离，分工明确。

后台接口的设计如表4.7所示：

表4.7 后台接口设计表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 接口名称 | 方法 | 参数描述 | 接口功能 |
| 1 | /api/private/v1/article/create | post | 新建文章的一些信息，存放在body中 | 发表新文章 |
| 2 | /api/private/v1/article/:id | get | id：文章的id | 获取文章信息 |
| 3 | /api/private/v1/article | get | 查询文章的条件，在params中 | 根据条件查找文章列表 |
| 4 | /api/private/v1/article/:id | delete | id：文章的id | 删除文章 |
| 5 | /api/private/v1/comment/article/:id | get | id：文章的id | 获取某个文章下的所有评论 |
| 6 | /api/private/v1/comment | post | 新建评论的一些信息，存放在body中 | 发表评论 |
| 7 | /api/private/v1/comment/:id | delete | id：评论的id | 删除评论 |
| 8 | /api/private/v1/compile | get |  | 所有编译信息 |
| 9 | /api/private/v1/compile/:language | get | language：要编译的语言类型 | 获取某个编译语言的信息 |
| 10 | /api/private/v1/email | post | 接收邮件用户的一些信息，存放在body中 | 发送验证码到邮箱 |
| 11 | /api/private/v1/like | post | 点赞对象的一些信息，存放在body中 | 文章或评论点赞/取消赞 |
| 12 | /api/private/v1/login | post | 登录账号的一些信息，存放在body中 | 用户登录 |
| 13 | /api/private/v1/login/auto | post |  | 自动登录 |
| 14 | /api/private/v1/regist | post | 注册账号的一些信息，存放在body中 | 用户注册 |
| 15 | /api/private/v1/upload/avatar | post | 新头像文件等信息，存放在body中 | 修改头像 |
| 16 | /api/private/v1/upload/image | post | 上传图片的文件信息，存放在body中 | 上传图片 |
| 17 | /api/private/v1/user/:id | get | id：用户的id | 用户首页信息 |
| 18 | /api/private/v1/user/forgetPassword | post | 新密码、验证码等信息，存放在body中 | 忘记密码 |
| 19 | /api/private/v1/user/modifyPassword | post | 新密码，存放在body | 修改密码 |
| 20 | /api/private/v1/user/dynamic/:id | get | id：用户的id | 获取用户动态 |
| 21 | /api/private/v1/user/editPhone | post | 新手机号，放在body | 绑定手机 |
| 22 | /api/private/v1/user/editInfo | post | 修改后的用户信息，存放在body中 | 修改个人信息 |

在上表中的post请求中，前端的网络请求携带有一些参数，这些参数存放在request请求的body中，以x-www-form-urlencoded格式传递，在本文中不做详细列举。

**第5章 系统实现**

**5.1** **系统主要功能页面展示**

**5.1.1** **主页面展示**

图5.1为网站的首页，如果访客之前登录过社区，会在cookie中存有一个token，在后面访问时会自动登录账号。网站的首页显示的是社区为所有访客提供的开发工具，访客点击即可打开新的页面使用在线工具。点击左侧的导航区域即可滚动到对应的工具区域。

其中，编译工具部分为访客提供了各种语言的在线编译器；前端工具部分为访客提供了一些开发工具，如JSON在线解析、RGB与16进制颜色格式转化、在线字数统计等；学习网站位访客列举了一些国内外的优秀论坛；开发工具部分为访客列举了一些实用的网站。

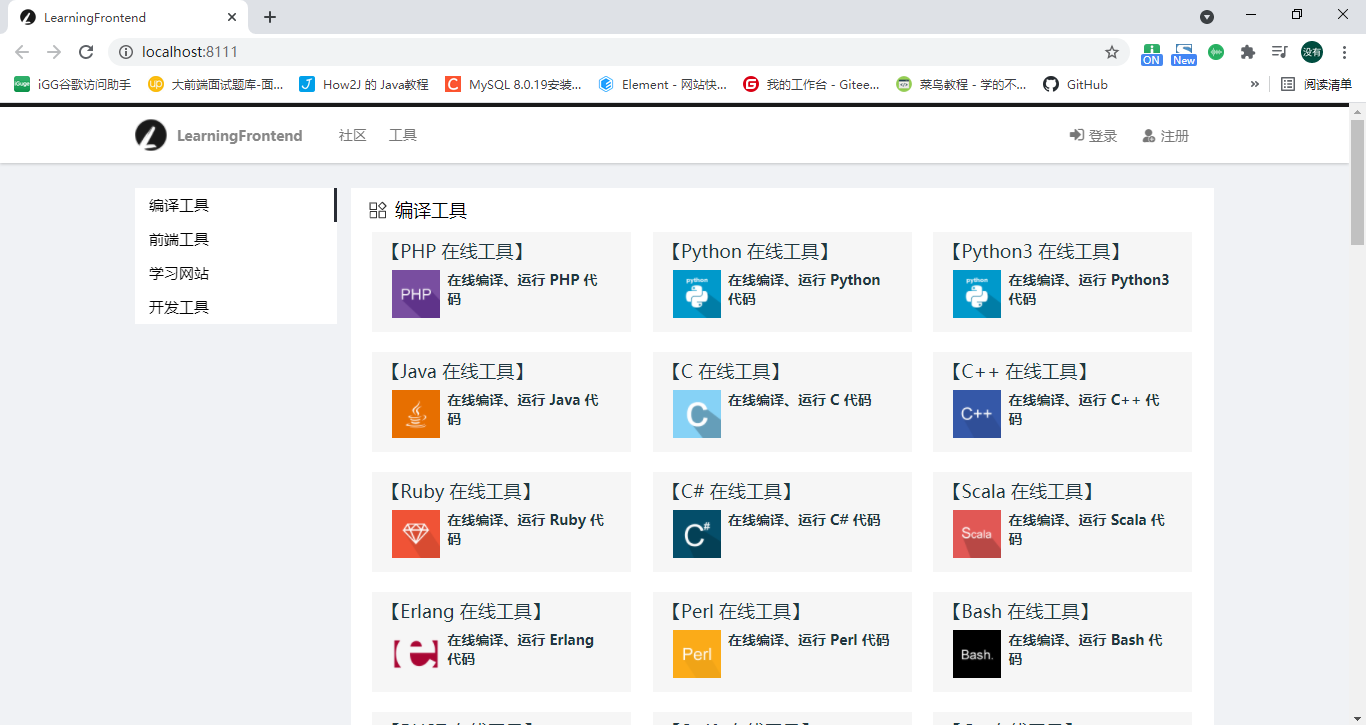


图5.1 网站首页

**5.1.2** **开发工具页面展示**

在图5.1的页面中点击即可进入对应在线工具页面，下面选取两个展示。

图5.2为前端工具中在线取色器，访客首先在右侧长方形选取某个颜色，左侧正方形区域便会出现黑色到该颜色之间的所有颜色，用户点击任一颜色，网页右侧便会显示改颜色的RGB和16进制颜色值。

图5.3为编译工具中的Java在线工具，用户在左侧输入框填写Java代码，点击运行后，会对代码进行编译运行，最后将结果显示在右侧。

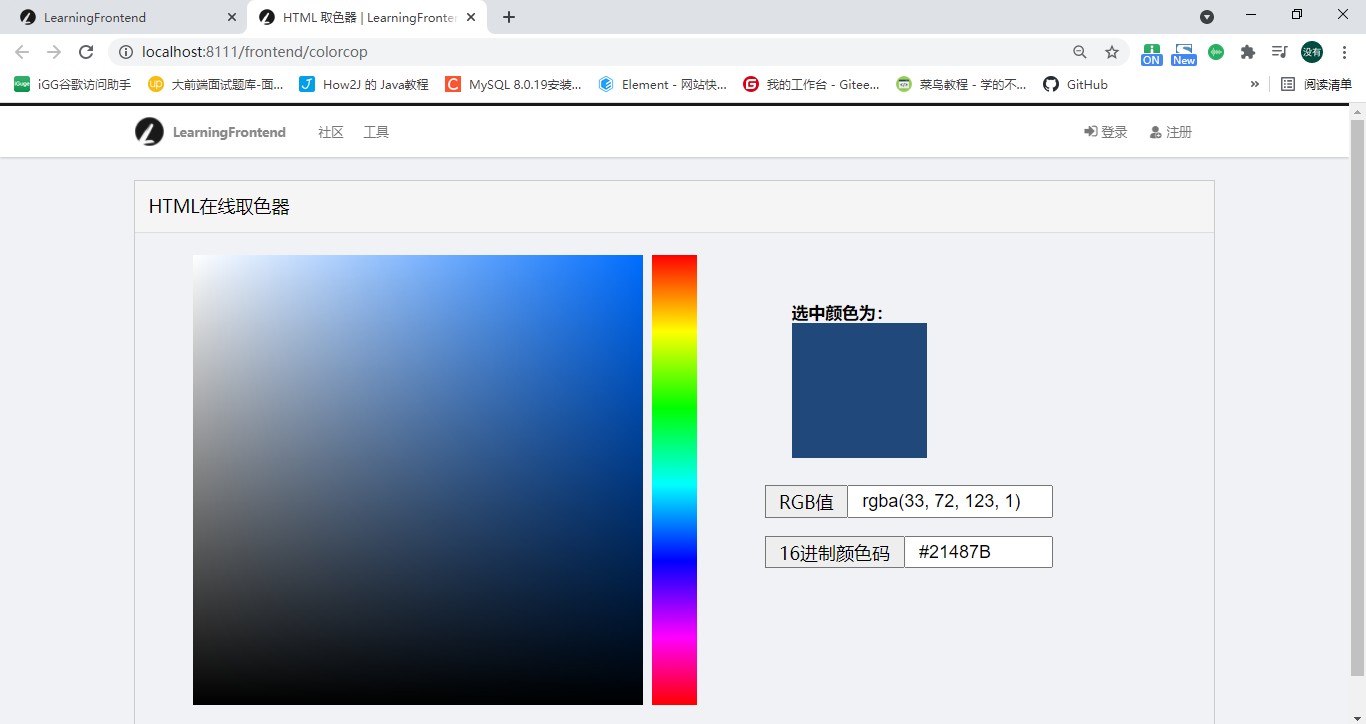


图5.2 HTML在线取色器页面

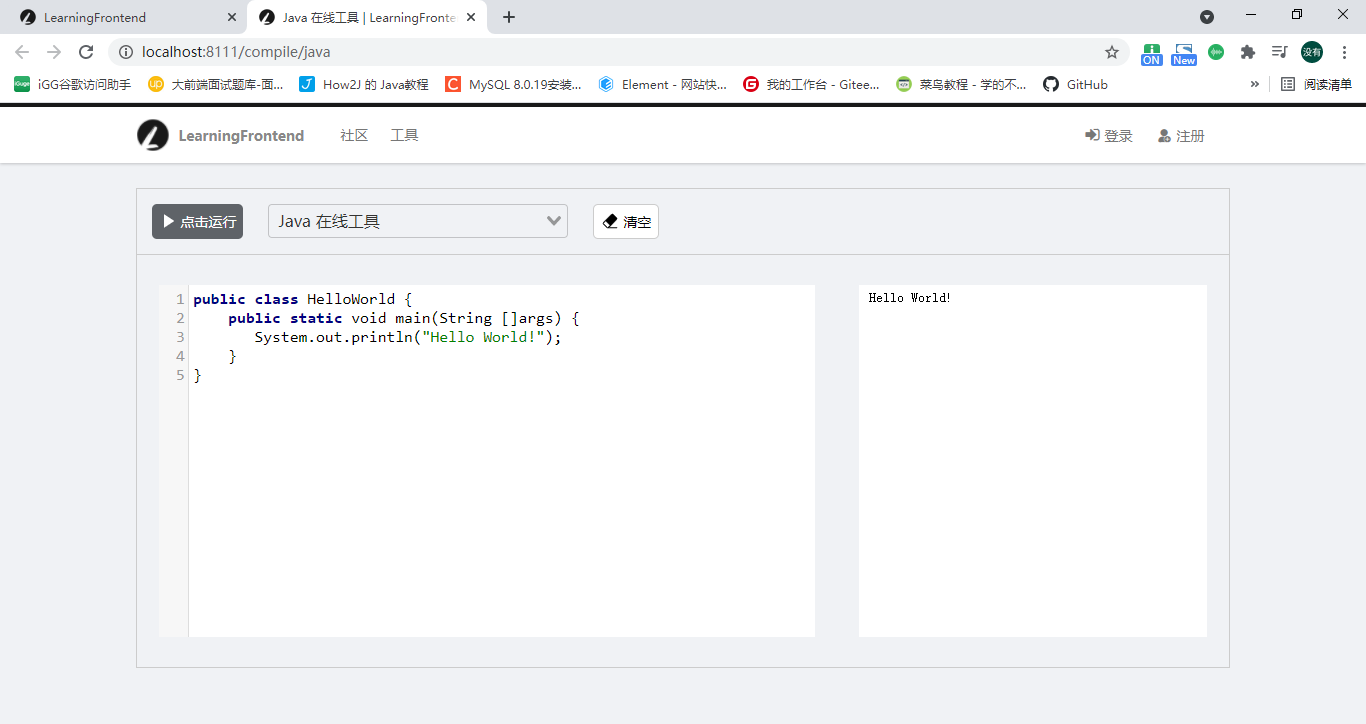


图5.3 Java在线编译工具页面

**5.1.3** **登录模块展示**

访客如果想要登录，点击网站右上角的登录，即可跳到登录页面（图5.4），输入正确的用户名、密码和验证码即可登录。如果没有账号，则可以进入注册页面，输入对应的信息后，即可注册账号。而如果用户忘记了自己的密码，只需点击忘记密码，输入用户名和绑定的箱，点击发送验证码，该邮箱即可收到一个6位数的验证码（10分钟内有效），输入验证码和新密码，即可修改密码。

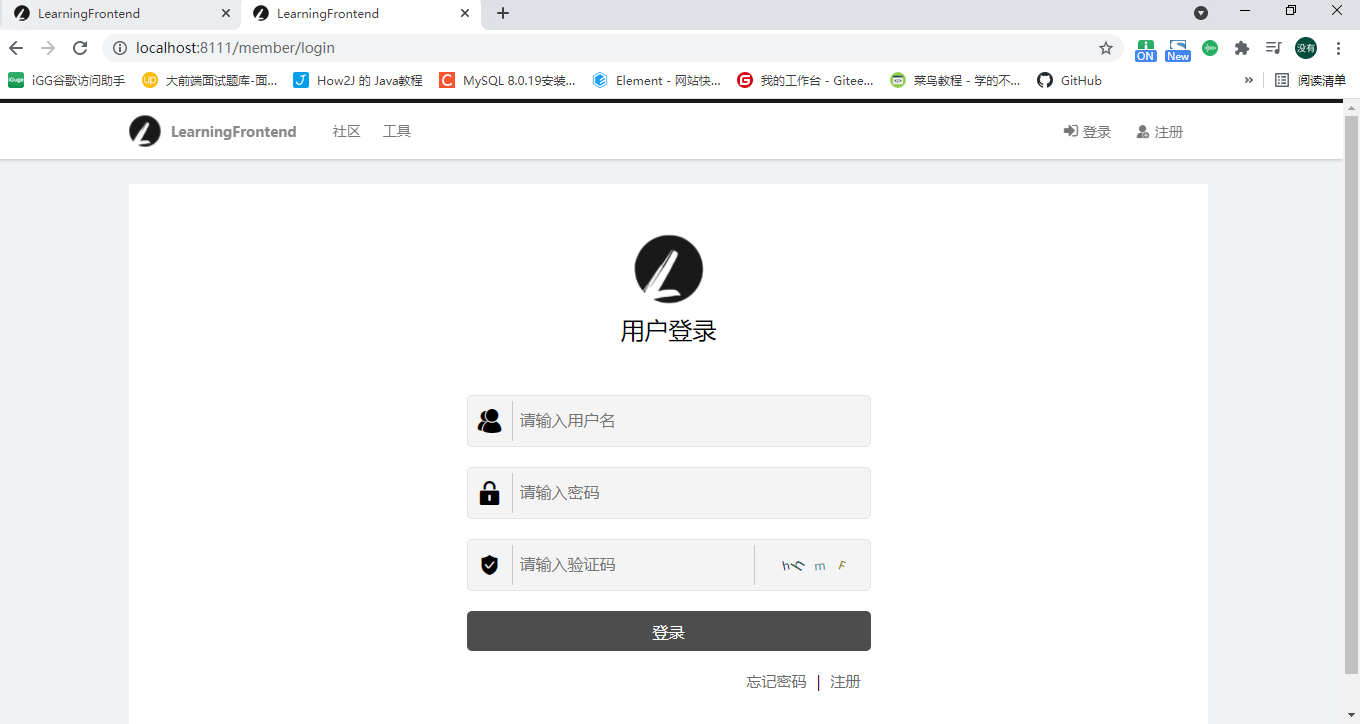


图5.4 登录页面

**5.1.4** **社区页面展示**

访客点击顶部导航栏的“社区”，便可进入社区页面。在该页面访客可以看到社区的文章，通过文章列表顶部的操作栏，可以对文章进行排序和筛选，执行操作后，页面会显示加入排序/筛选条件后的文章，排序条件包含按点赞数、评论数、发布时间、最后活跃时间等排序规则，筛选条件包含按文章类型筛选和查看规定时间段内的文章。点击任一文章行，即可进入文章的详情页；点击右侧的发布内容，即可跳转到文章发布页面。

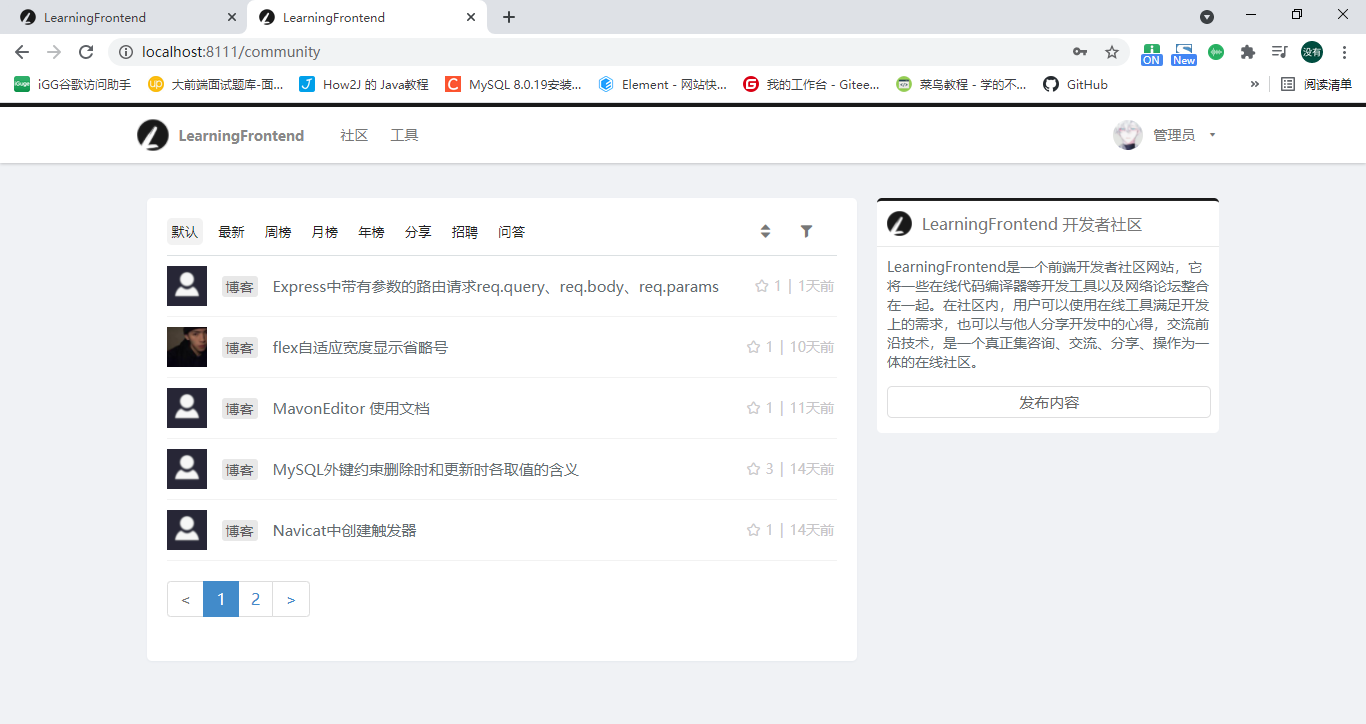


图5.5 社区页面

**5.1.5** **文章发布页面**

进入文章发布页面后，首先需要选择发布文章的类型，如图5.6，在选择文章类型后，页面会显示文章内容的编辑区域，用户可以输入文章标题和文章的内容，文章内容在左侧的区域进行输入，以支持markdown语法，在右侧会实施显示markdown语言解析后的样式，即最终文章会显示的样式。点击“保存草稿”或者按ctrl+s，可以将内容保存到localstorage，用户下次进入会自动加载之前保存的草稿。在发表成功后，会自动跳转到该文章的详情页面，没有登录的访客无法发表文章。

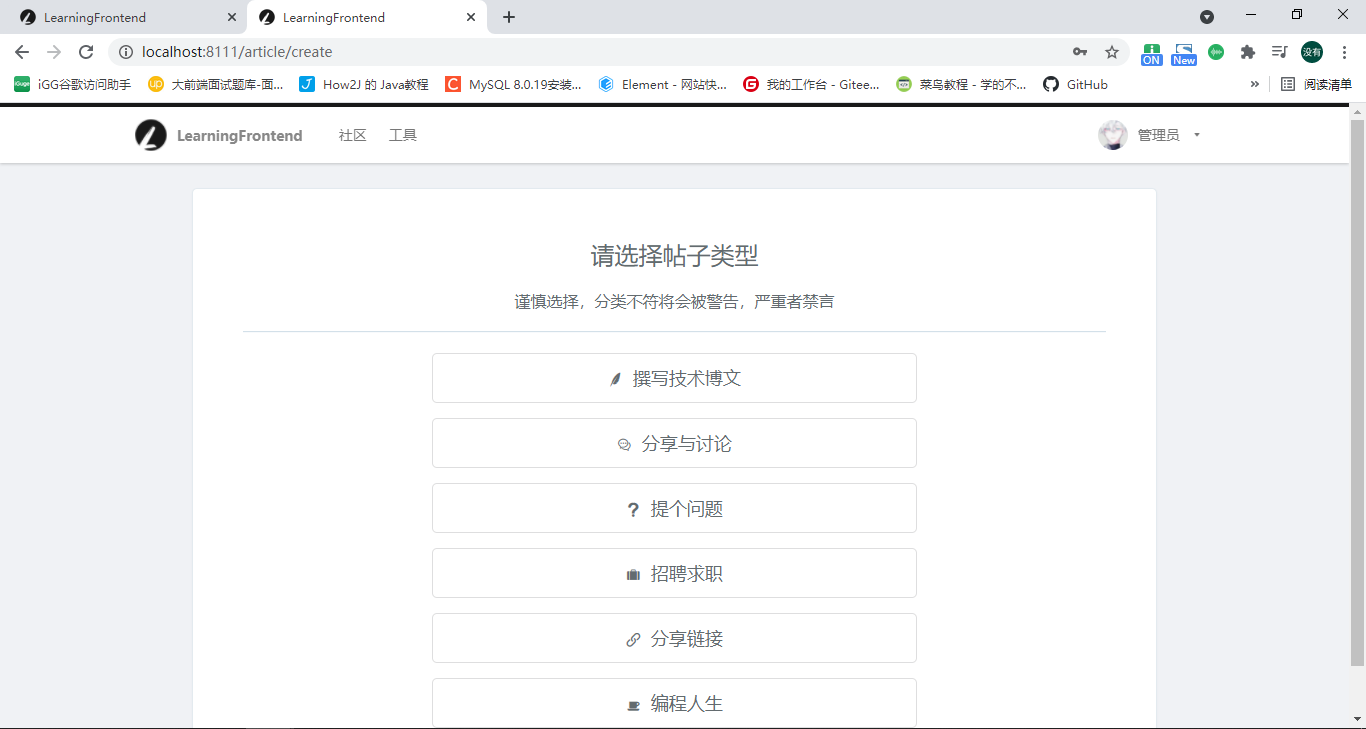


图5.6 文章发布页面 – 类型选择

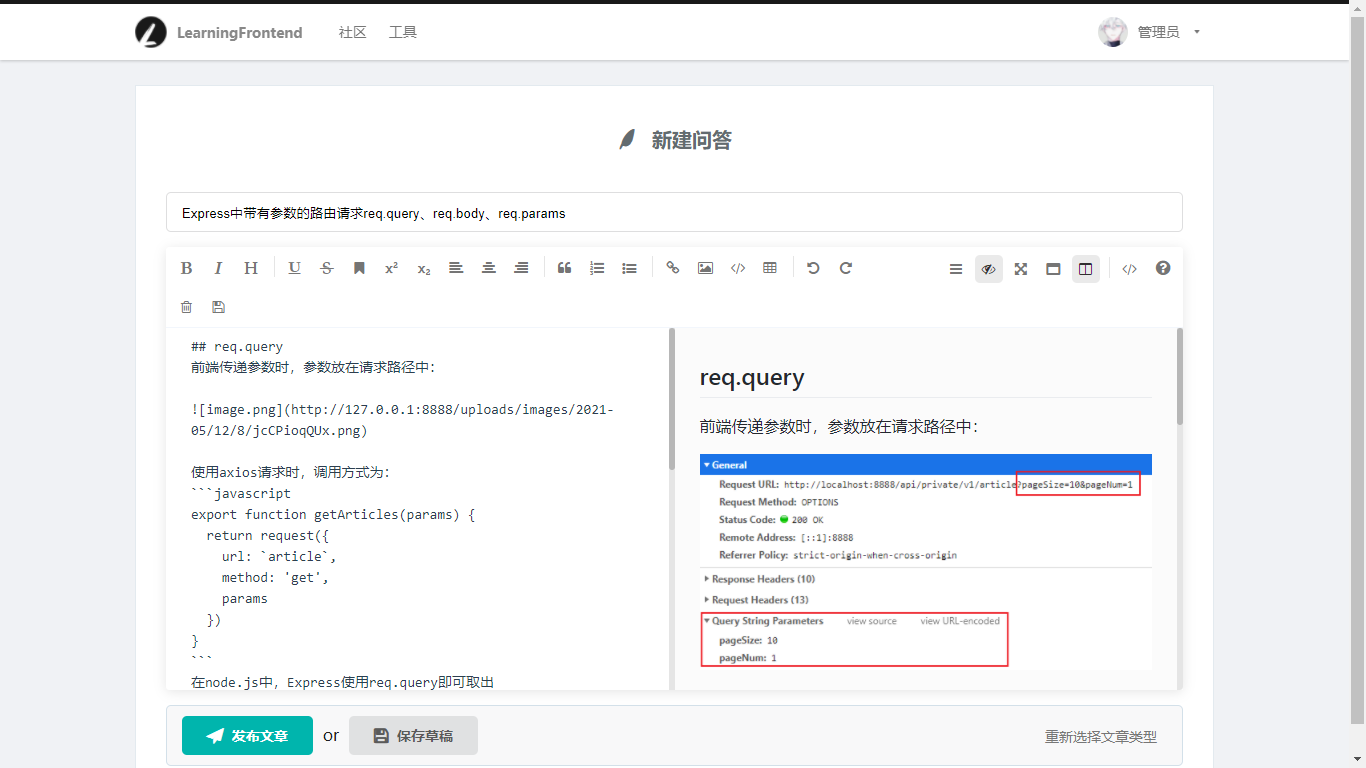


图5.7 文章发布页面 – 内容编辑

**5.1.6** **文章详情页面**

在文章详情页面，访客可以看到文章的一些信息，在页面左侧是文章作者的部分信息，包括昵称、头像、发表过的文章数量、收到的点赞数量等。在中间是文章的具体信息，包括标题、内容、发布时间、浏览量等，右侧的操作区域，可以进行文章点赞和删除，如图5.8所示。

在最下方，是评论区域，访客可以看到该文章下的所有评论，并且可以发表自己的评论（需要登录），文章评论支持markdown语法，用户可以边输入边查看解析后的样式。另外，用户也可以对评论进行回复，类似楼中楼的功能。

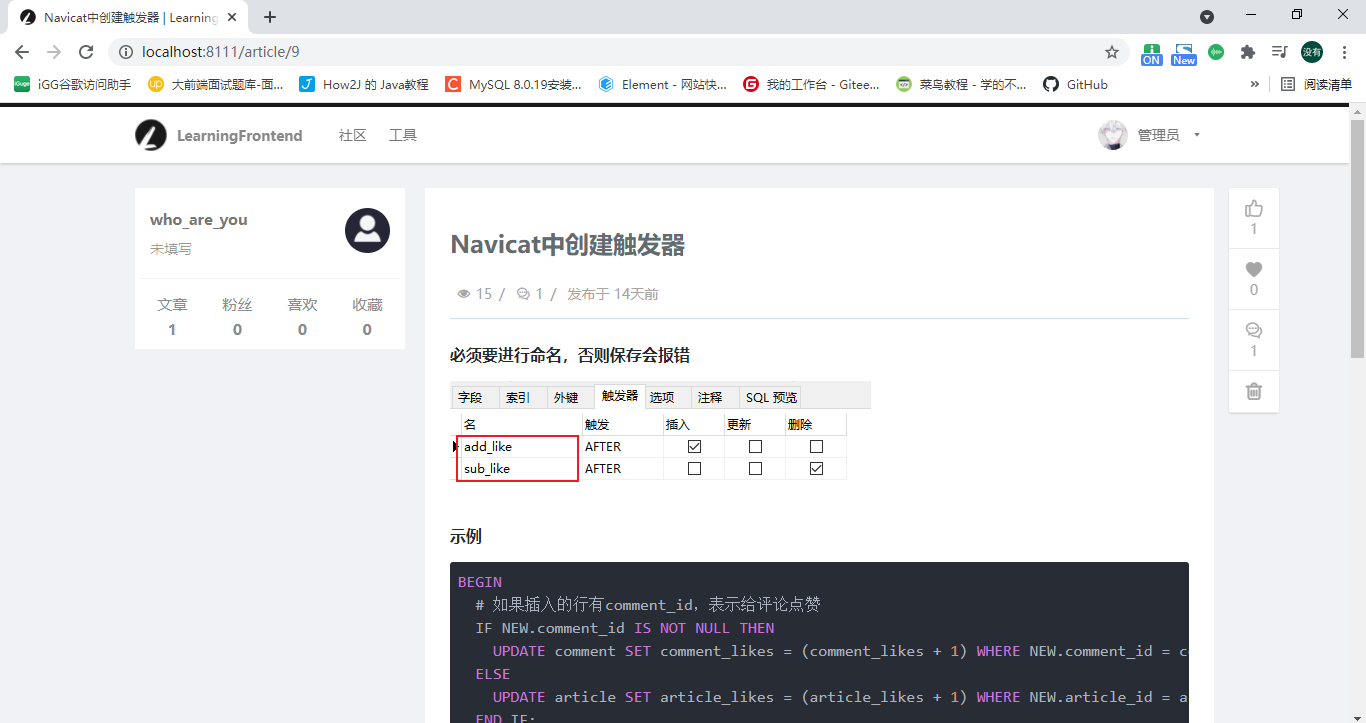


图5.8 文章详情页面 – 文章信息



图5.9 文章详情页面 – 评论区域

**5.1.7** **用户个人中心页面**

当访问用户的个人中心时，会显示该用户的部分信息，如图5.10所示，点击左侧的导航菜单，可以查看用户发布过的文章，在基本资料部分，可以看到该用户最后活跃时间、加入社区时间、个性签名、发布过的所有文章、最近的动态（包含点赞、评论、发布文章）。右侧区域显示该用户的头像、收到的点赞等信息，如果访问的是自己的个人中心，点击右侧的头像，可以进入修改头像页面。



图5.10 用户个人中心页面

**5.1.8** **用户信息修改模块**

在该页面，用户可以修改自己的个人信息，如修改头像、密码等。

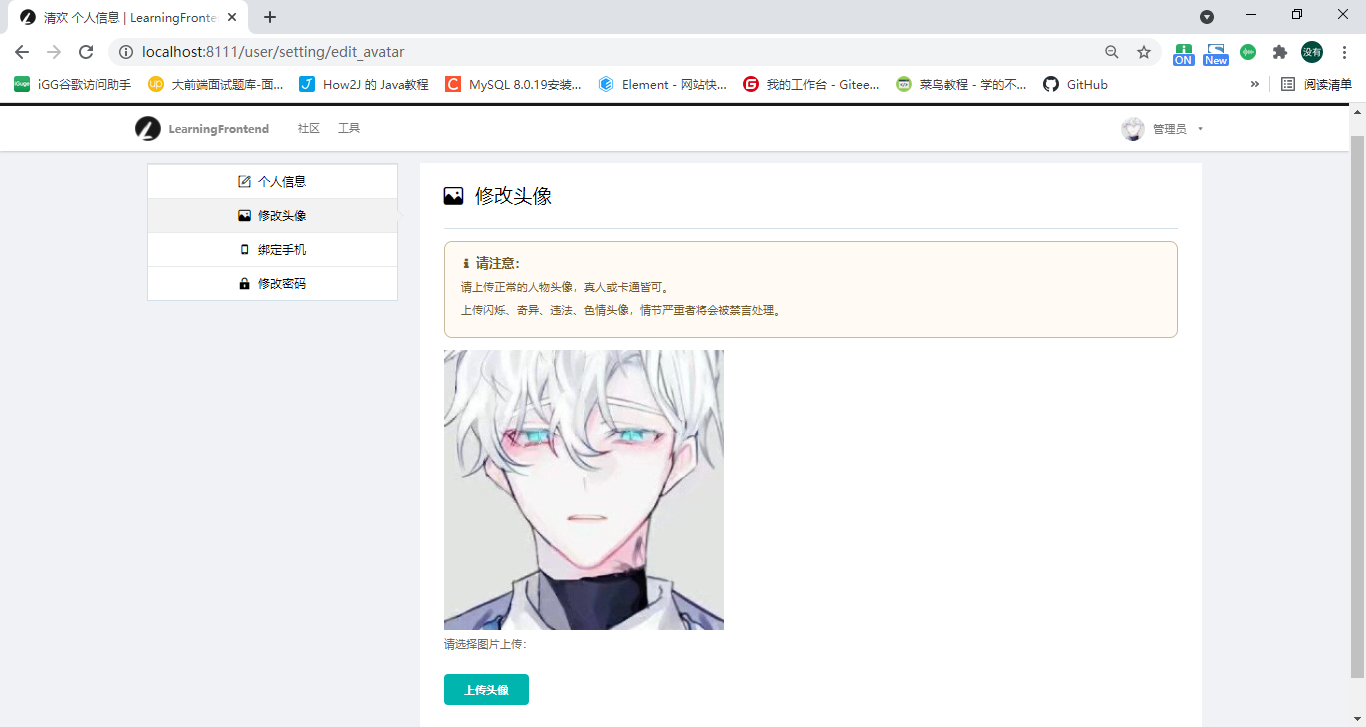


图5.11 用户头像修改页面

**5.2** **系统部分功能实现**

**5.2.1** **MD5加密**

在登录和注册时，服务器对前端传来的密码进行MD5加密，然后再同服务器中存储的密码进行对比或存储，这是一种被广泛使用的密码散列函数，可以产生出一个128位（16字节）的散列值（hash value），用于确保信息传输完整一致。在命令行键入npm install md5或者npm install crypto，引入加密模块，即可实现密码的加密功能。

**5.2.2** **MarkDown语法解析**

前端文章发布页面，我们使用了mavon-editor组件，这是一款基于vue的markdown编辑器，比较适合本系统，只需在前端项目中输入npm install mavon-editor –save进行模块安装，再参考官网文档进行使用即可。

而在文章评论时，本系统的需求是禁用H1-H6的markdown语法，这是mavon-editor无法满足的需求，查阅资料，系统选取了markdown-it，这是一个用来解析markdown的库，它能够将markdown代码编译为html代码，在引用之后，通过自带的disable函数，将H1-H6进行禁用，前端输入框在输入内容后，系统会将内容进行自动编译，需求得以解决。

**5.2.3** **前端数据持久化**

在用户访问浏览器时，一些数据需要在前端保存，如登录产生的token，登录状态，发布文章时保存的草稿等。Web应用的发展，使得客户端存储使用得也越来越多，而实现客户端存储的方式则是多种多样。最简单而且兼容性最佳的方案是Cookie，后面的发展中，出现了包含sessionStorage与localStorage两部分的Web Storage。

在本项目中，将token和登录状态的存储放在了cookie中，如果不在浏览器中设置过期时间，cookie被保存在内存中，生命周期随浏览器的关闭而结束，这种cookie简称会话cookie。如果在浏览器中设置了cookie的过期时间，cookie被保存在硬盘中，关闭浏览器后，cookie数据仍然存在，直到过期时间结束才消失。

发布文章时保存的草稿则存放在localstorage中，localStorage是持久化的本地存储，除非是通过js删除，或者清除浏览器缓存，否则数据是永远不会过期的，且localstorage可存储的内容更多。

**第6章 系统测试**

**6.1** **测试的原理和目的**

系统测试是将必须被测试的软件作为整个计算机系统的一个要素，结合硬件、外围设备、部分支持软件、数据和人员等系统和环境要素进行测试。它是一个提高软件的正确性、完整性、安全性和质量的过程，通过对可计算性理论（计算的一个分支）的一个简单的数学检验，得出如下结论：不可能完全解决所谓的“崩溃”，它是指一个计算机程序是否会进入死循环或命中并产生输出，换句话说，系统测试是一个修正或比较实际生产与预期生产的过程。

系统测试的方法有很多种，但对复杂产品的有效测试不仅是一个研究过程，更是一个创设并严格遵循一些刚性步骤的大事，测试的定义之一是对被评估产品提出质疑的过程；这里的“问题”是测试者试图对产品做什么，产品对测试者脚本的行为做出反应。虽然大多数测试的智力过程不过是一次检查和一次检验，但“测试”一词意味着对产品的动态分析，程序的质量可以，而且通常，因系统而异；但一些公认的特征是共同的：可靠性、稳定性、便携性、易维护性和实用性。

本系统在上线后直接面向用户，不仅对用户的体验要求较高，也对性能有着很高的要求，所以在本章节将着重对系统交互部分和性能进行系统测试。

**6.2** **测试方法**

软件测试一般分为黑盒测试和白盒测试。

黑盒测试：又称黑箱测试，是一种软件测试方法，用来测试应用程序的功能，而不是其内部结构或功能。测试人员不需要对应用程序代码有专门的知识，内部结构和编程语言。测试人员只需要知道系统应该做什么，即当输入特定的输入时，测试用例是根据应用系统应该执行的功能，按照规范或要求设计的，测试人员选择有效输入和无效输入来验证正确的输出，这种测试方法适用于大多数软件测试，作为集成测试和系统测试。

白盒测试：又称透明盒测试、玻璃盒测试、结构测试等，是一种软件测试方法、内部结构测试或应用操作测试，在白盒测试中，测试用例是从编程语言的角度设计的，测试者提供数据来验证程序中数据流的流动路径，并确定适当的输出，空盒测试可以应用于单元测试、集成测试和系统软件测试过程。它可以测试集成过程中各个单元之间的路径，也可以测试主系统和子系统，虽然这种测试方法可能会发现很多错误或问题，但可能无法检测到本专业中未使用的部分。

**6.3** **测试实例**

**6.3.1** **用户登录**

测试1：

测试内容：输入正确的用户名、密码，是否登录成功。

操作：访客输入正确的用户名和密码。

结果：顶部导航区域显示昵称和头像。

结论：测试成功。

测试2：

测试内容：输入错误的用户名、密码，是否登录成功。

操作：访客输入错误的用户名和密码。

结果：提示用户名或密码错误。

结论：测试成功。

**6.3.2** **自动登录**

测试1：

测试内容：用户已经登录网站，下次进入是否会自动登录。

操作：用户退出浏览器，重新进入网站。

结果：无需输入密码，自动登录。

结论：测试成功。

测试2：

测试内容：用户在登录后退出账号，下次进入是否会自动登录。

操作：用户退出账号，再关闭浏览器，重新进入网站。

结果：没有自动登录。

结论：测试成功。

**6.3.3** **发布文章**

测试1：

测试内容：测试访客是否可以进入文章发布页面。

操作：未登录的访客点击社区页面的“发布内容”。

结果：弹出未登录提示，并自动跳往登录页面。

结论：测试成功。

测试2：

测试内容：测试用户是否可以发布文章。

操作：用户选择文章类型，输入标题和内容并发布。

结果：提示发布成功，并跳往文章详情页面。

结论：测试成功。

**6.3.4** **发表评论**

测试1：

测试内容：测试访客是否可以评论文章。

操作：未登录的访客在文章评论区进行评论。

结果：弹出未登录提示。

结论：测试成功。

测试2：

测试内容：测试用户是否可以评论文章。

操作：用户在文章评论区进行评论。

结果：提示发布成功，评论区显示刚发表的评论。

结论：测试成功。

**6.3.5** **分类查看社区文章**

测试内容：测试根据查询条件是否可以查看对应文章。

操作：查看一个月内按评分排序的文章列表。

结果：文章列举成功。

结论：测试成功。

**6.3.6** **删除文章或评论**

测试1：

测试内容：测试访客是否可以删除文章或评论。

操作：未登录的访客在进入文章详情页。

结果：不显示删除按钮。

结论：测试成功。

测试2：

测试内容：测试非文章或评论的发布人是否可以进行删除。

操作：非文章或评论的发布人进入文章详情页。

结果：不显示删除按钮。

结论：测试成功。

测试3：

测试内容：测试文章或评论的发布人是否可以进行删除。

操作：文章或评论的发布人进入文章详情页进行删除。

结果：文章的发布人可以删除文章以及文章下的所有评论，评论的发布人可以删除自己的评论以及自己评论下的子评论。

结论：测试成功。

测试4：

测试内容：测试管理员是否可以删除文章或评论。

操作：管理员进入文章详情页进行删除。

结果：管理员可以删除所有文章和评论。

结论：测试成功。

**6.3.7** **查看个人中心**

测试1：

测试内容：测试访客是否可以查看某一用户的个人中心。

操作：访客在文章页面点击某一用户的头像或昵称。

结果：进入该用户的个人中心。

结论：测试成功。

**6.3.8** **修改个人信息**

测试1：

测试内容：测试用户是否可以进入个人信息修改页面。

操作：用户将鼠标移至顶部导航栏头像，并点击弹出的“编辑资料”。

结果：进入个人信息修改页面。

结论：测试成功。

测试2：

测试内容：测试用户是否可以修改个人信息。

操作：用户在修改资料页面填写信息并提交。

结果：弹出修改成功提示。

结论：测试成功。

测试3：

测试内容：上传非图片格式文件，测试用户是否可以修改头像。

操作：用户在上传图片修改头像时，选择上传非图片格式文件。

结果：提示“仅限图片格式”。

结论：测试成功。

测试4：

测试内容：上传符合要求的图片，测试用户是否可以修改头像。

操作：用户在上传图片修改头像。

结果：提示修改成功，顶部导航栏头像改变。

结论：测试成功。

**第7章 总结与展望**

本系统是一个社区系统，面向的主要用户群体是前端开发者，系统将开发中用到的一些开发工具放在了社区中，为用户提供了一个集咨询、交流、分享、操作为一体的在线平台。

本系统是一个小型的网站项目，前端使用了HTMLL+CSS+JavaScript+Vue的技术栈，后端部分使用了Node.js技术进行开发，同时使用了Express.js框架来构建服务器。Vue.js是近年来比较流行了框架，是当今三大框架之一，Vue.js凭借虚拟DOM减少了操作DOM时引起的页面回流和重绘，提升了性能，本身又是基于数据驱动和组件化思想开发，提供了非常简洁，易于理解的API。在后端语言的选取时，经过多方语言的对比，最终选择了运行在服务端的JavaScript—Node.js，前后端统一的开发语言，大大降低了开发成本。

本系统的特点在于：

（1）系统的后端基于Node.js开发，Node.js通过事件循环机制（初始化和回调）的方式运行 JavaScript 代码，并且提供了一个线程池处理高成本的任务，极大提高了服务器的并发性。

（2）系统的前端使用了Vue.js，基于MVVM的设计模式，使得在开发中，侧重点主要放在了用事件驱动UI的开发，View发生变化时，ViewModel也自动更新，而ViewModel的变化也会自动同步到View上显示，实现了“高内聚，低耦合”。

（3）系统重视数据的安全性，对用户密码进行了加密处理，使得在处理后即使是数据库管理者也不能知道用户真正的密码；同时使用了token进行授权，避免了非法用户对数据的操作。

目前系统己经通过了一系列的测试, 运行表现稳定 , 但是由于开发时间的原因，系统仍存在有不足之处，系统如今只实现了社区的基本功能，在后续还需要继续添加新功能，如消息推送、文章收藏、用户关注等。

由于是第一次进行vue+node的全栈式开发，很多功能都是在边学习边实践中实现的，在这个过程中学到了很多编程知识和思想，在以后的完善过程中，相信肯定可以学到更多新的知识。

**参考文献**

**致谢**

至此论文完成之际，谨向我的导师沙静导师致以衷心的感谢和崇高的敬意！本论文倾注了导师大量的心血，无论是在论文的选题、项目的研究，还是论文的撰写，均离不开沙老师的悉心指导。沙老师严谨的治学态度和高度的责任心是我学习的榜样，并将是我终生献身科学和献身事业的动力。

高山下，泉水旁，海滨好风光。感谢学校四年的培养。

感谢陪伴我的舍友和同学，四年朝夕相处的陪伴，让我领略了形形色色的个性，见证了无数的精彩与感动，让我的大学生活不在孤单乏味。

还要向我的父母和家人表达真挚的感谢，自始至终我都不是一个顺从的孩子，在每一个人生的关键阶段，我的选择都同父母发生了分歧，感谢父母对我的支持，感谢对我这个性格过于倔强的孩子的无限包容。

未来的生活可能会有不尽人意之时，但所有的路都是自己做的选择，虽然过程会很苦，但是我不会后悔。

感谢遇见，感谢经历。

感谢这一路坚持走下来的自己。

故事还长，请别失望。

1. [] Chun-Cheng, Hsu, Jo, et al. Exploring the Design Principles Necessary for Building Blog Website Interface[C]// 第十七届国际工效学大会暨中国人类工效学会成立20周年庆祝大会. 0. [↑](#endnote-ref-1)
2. [] 吴飞燕. 基于用户体验的网页UI设计[J]. 电子世界, 2014, 000(010):107-107. [↑](#endnote-ref-2)
3. [] 徐頔, 朱广华, & 贾瑶. (2017). 基于vuejs的web前端开发研究. 科技风, 000(014), 69-69. [↑](#endnote-ref-3)
4. [[]] 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, 377(07):58-59. [↑](#endnote-ref-4)
5. [] 杨小彦. 基于TOKEN令牌的身份校验方法及相关设备:, CN110493258A[P]. 2019. [↑](#endnote-ref-5)
6. [] Rfc B . 1321, The MD5 Message-Digest Algorithm[J]. Network Working Group Ietf, 1992. [↑](#endnote-ref-6)
7. [] 彭娜. 基于Node.JS博客系统的设计与实现. (Doctoral dissertation, 大连理工大学). [↑](#endnote-ref-7)
8. [] 刘立. MVVM模式分析与应用[J]. 微型电脑应用, 2012(12):65-68. [↑](#endnote-ref-8)
9. [] 闫志英. 浅析Web前端开发技术[J]. 无线互联科技, 2016, 000(002):47-48. [↑](#endnote-ref-9)
10. [] Goel U , Steiner M . System to Identify and Elide Superfluous JavaScript Code for Faster Webpage Loads[J]. 2020. [↑](#endnote-ref-10)