

**本 科 毕 业 论 文**

**­­­­­­­­­**

仿简书网站的设计与实现

**Design and implementation of a JianShu website**

学 院 名 称： 计算机科学与信息工程学院

专 业 班 级： 网络工程14-1

学 生 姓 名： 谢天昌

学 号： 14031110102

指导教师姓名： 李爱玲

指导教师职称： 教 授

2018年 5月

**毕业设计（论文）原创性声明和使用授权说明**

**原创性声明**

本人郑重承诺：所呈交的毕业设计（论文），是我个人在指导教师的指导下进行的研究工作及取得的成果。尽我所知，除文中特别加以标注和致谢的地方外，不包含其他人或组织已经发表或公布过的研究成果，也不包含我为获得安阳工学院及其它教育机构的学位或学历而使用过的材料。对本研究提供过帮助和做出过贡献的个人或集体，均已在文中作了明确的说明并表示了谢意。

作 者 签 名：　　 　　 日　 期：

指导教师签名：　 日　 期：

**使用授权说明**

本人完全了解安阳工学院关于收集、保存、使用毕业设计（论文）的规定，即：按照学校要求提交毕业设计（论文）的印刷本和电子版本；学校有权保存毕业设计（论文）的印刷本和电子版，并提供目录检索与阅览服务；学校可以采用影印、缩印、数字化或其它复制手段保存论文；在不以赢利为目的前提下，学校可以公布论文的部分或全部内容。

作者签名：　　 　 　 日　 期：

目 录

[引言 1](#_Toc513042931)

[第1章 绪论 2](#_Toc513042932)

[1.1 课题背景 2](#_Toc513042933)

[1.2 课题目的和意义 3](#_Toc513042934)

[1.3 设计思想 3](#_Toc513042935)

[1.4 开发技术简介 4](#_Toc513042936)

[1.5 环境简介 4](#_Toc513042937)

[第2章 系统需求分析 6](#_Toc513042938)

[2.1 系统需求分析 6](#_Toc513042939)

[2.2 可靠性和可行性需求分析 6](#_Toc513042940)

[第3章 系统设计 8](#_Toc513042941)

[3.1 功能模块设计 8](#_Toc513042942)

[3.2 系统流程设计 8](#_Toc513042943)

[3.3 系统数据结构设计 9](#_Toc513042944)

[3.4 前端设计 10](#_Toc513042945)

[3.5 网站后台设计 11](#_Toc513042946)

[第4章 系统实现 12](#_Toc513042947)

[4.1 前端界面结构的实现 12](#_Toc513042948)

[4.2 前端界面样式的实现 14](#_Toc513042949)

[4.3 前端HTTP请求接口的实现 16](#_Toc513042950)

[4.4 网站数据库的实现 21](#_Toc513042951)

[4.5 网站服务端的实现 23](#_Toc513042952)

[第5章 系统测试 29](#_Toc513042953)

[5.1 测试方法 29](#_Toc513042954)

[5.2 网站首页按钮功能测试 29](#_Toc513042955)

[5.3 网站编写文章功能测试 30](#_Toc513042956)

[5.4 网站文章详情展示功能测试 31](#_Toc513042957)

[5.5 用户评论功能测试 32](#_Toc513042958)

[总 结 33](#_Toc513042959)

[致 谢 34](#_Toc513042960)

[参考文献 35](#_Toc513042961)

仿简书网站的设计与实现

摘要：本课题研究的是简书网站的开发与实现，主要由前端部分，网站后台功能组成，基本上实现了简书官方网站的基本功能。譬如用户注册、用户登录、用户发表文章、用户修改文章、用户删除文章用户阅读文章、用户对文章进行评论、用户对其他用户的关注、用户对文章的收藏、站内私信等基本功能。前端开发主要是仿造简书网站页面功能进行模仿开发，让用户通过前端界面操作与服务器进行数据交换，对应的前端按钮使用AJAX调取对应服务器的HTTP接口进行数据获取，跟服务器后台进行数据交互，将获取到的用户信息、文章信息、评论信息、喜欢的文章信息、文章点赞数等数据渲染到HTML中在浏览器显示。后台开发接受前端页面返回来的参数进行对数据库的增加，删除，修改，查询等操作，将结果通过HTTP进行服务器响应，实现网站的前后端对接。

系统网站采用NODE环境进行开发，前端MVVM框架VUE进行前端界面的构建，后台部分使用node.js的EXPRESS轻量级web框架进行与MYSQL数据库的操作。并用谷歌浏览器作为网站系统的展示平台，设计并实现了简书网站的基本功能开发。Node.js是一种通过JavaScript语言开发web服务端的开发语言，同时也是一个运行环境。

关键字：NODE；简书网站；VUE；MYSQL数据库；网站前端；

**Design and implementation of a JianShu website**

**Abstract:** The research of this topic is the development and Realization of the website of Jane book, which is mainly composed of the front end part and the background function of the website, which basically realizes the basic function of the official website of Jane book. For example, user registration, user logged in, users publish articles, users modify articles, users delete article users to read articles, users comment on articles, users' attention to other users, users' collection of articles, station private letters and other basic functions. The front-end development is to imitate and develop the web page function of the simple book website, and let the user operate the data exchange through the front end interface and the server. The corresponding front end button uses AJAX to retrieve the HTTP interface of the corresponding server, and interacts with the server background, and will obtain the user information and articles. Data, such as information, commentary, favorite articles, and articles, are rendered to HTML in browser display. The backstage development accepts the parameters of the front-end page to increase, delete, modify, query and so on. The results will respond to the server through HTTP to achieve the front and back docking of the web site.

The system website is developed with the NODE environment, the front-end MVVM framework VUE is built for the front end interface, and the backstage part uses the EXPRESS lightweight web framework of node.js to operate with the MYSQL database. Using Google browser as the display platform of the website system, the basic function of the website is designed and implemented. Node.js is a development language for developing web server side through JavaScript language, and also a running environment.

**Key words:** NODE; Jane book website; VUE; MYSQL database; website front-end

# 引言

在互联网、地球村的状态下，以前的笔友通过写信来往，对出版设进行纸质文章投稿的形式渐渐减少。同时，创作社区的出现使得人们读书交友的范围变得更加广泛，人们可以通过在社区网站注册账号进行文章的发表与阅读他人优秀文章。在发现好文章的同时可以进行简单的评论进行交流，同时留下对方的社区id进行下次发表新文章的查看。

本系统仿照简书网站进行开发，实现了简书官网的基本功能，简书首页是用户进入简书网站后的默认页面，实时为用户推送所以作者的最新的创作作品，以供所有进入首页的用户进行文章阅读，提高大家的阅读兴趣，也可以点击文章标题进入文章的详情页进行文章的评论跟其他用户交互。

基于node.js环境开发网站社区系统，使用MySql数据库对网站信息进行存储处理，开发前端交互友好的界面，使用AJAX异步通信技术实现与服务器之间的通信，对用户的操作进行响应返回。本软件界面友好，功能齐全，与后台对接时响应快速体验良好，是一个发表原创文章的好社区。

本论文基本框架结构如下：第一章绪论，主要阐明了仿简书网站的设计与实现的课题背景、研究目的以及实际意义，同时也简要介绍了本软件依托的设计思想和理论技术基础等；第二章分析了一个创作社区网站的系统需求，第三章根据第二章的分析，提出合理的系统设计，为后面的具体如何进行系统实现打下了良好的基础；第四章是本论文的核心内容，主要讲述了实现一个完成简书网站功能的具体过程；第五章主要针对网站设计出来的功能进行测试，并且将发现的一些需求，还有漏洞填补上；最后，就是对于本次设计的一些经验总结，对在软件设计过程中提供过帮助的人表示感谢，以及罗列设计过程中用到的参考文献。

# 第1章 绪论

现代阅读较少，简书是一个优质的创作社区，在这里，你可以进行大量优质原创文章的阅读，与作者进行评论交互，可以找到你的良好笔友。你可以任性地进行创作，一篇短文、一张照片、一首诗、一幅画…我们相信，每个人都是生活中的艺术家，有着无穷的创造力。

本软件界面友好，用户体验良好，用户只需要按照正常的操作习惯就可以使用本软件进行创作、阅读、用户管理。

## 1.1 课题背景

与国际网络发展相比，中国网络发展较晚。对于社交创作的网站更少，譬如美国的脸书（Facebook）是世界上排名领先的站点，我国类似的网站有知乎，微博，简书等。虽然国内社区网站起步较慢，但因为中国网民的基数得以快速发展，在互联网时代，网络通信社区将是现代人没时间阅读课本找到阅读资源的途径和方法。

与纯粹的社交社区相比，简书社区是一个支持原创跟分享的创作社区，简书社区创建的目标是：成为是一个优质的创作社区，在这里，你可以进行大量优质原创文章的阅读，你可以任性地进行创作，一篇短文、一张照片、一首诗、一段对事物的见解等…我们相信，每个人都是生活中的艺术家，有着无穷的创造力。

简书社区的出现是为了满足现代人们在时间上去不了图书馆、报亭、书店等地方进行阅读，但又有阅读的需求。在简书社区，你可以在首页观看他人发表的文章，对此文章说出自己的见解、看法、疑问等，可以通过评论功能进行交互。在人们看到优秀的文章时进行收藏文章，对作者的关注，可以快速找到该作者发布的新文章，这样人们之间在创作与阅读的同时还可以交友互动，体现了当今互联网时代的交友需求。

本课题仿简书网站的设计与实现是为了实现简书官方网站的基本功能，可以让用户在上边进行创作与优质阅读，更可以通过文章的发表记录中的点点滴滴，让记忆留在最初的感觉，在以后重新翻到的时候开怀一笑、或者感慨不已。本系统主要是仿照简书进行功能开发，为以后开发类型网站积攒经验。

## 1.2 课题目的和意义

在现代实体阅读较少，社区网站是一个很好的阅读替代品。在创作社区网站，你可以任性地进行创作，你也可以阅读大量优质原创文章。同时，在进行文学创作与阅读的同时，你可以与他人进行互动，有一种以前通信靠寄信的感觉，让每个人的交友变得更简单，不用担心电话不敢接的情况，在本系统中人们可以通过给对方留言就行交流，当年上线的时候就可以看到你收到的信息，这种感觉和90年代的书信来往一模一样。

本课题的目的是实现简书的基本功能应用：创作--遇到开心的时候，在简书上写作记录生活，简单而实用的功能；评论--对优秀文章进行点评讨论，让社交网站沟通远在他方的人，因为乐趣而相识，文笔交友；浏览--闲暇时间打开简书浏览，发现优秀文章进行阅读，还可以阅读他人金点评论，增加阅读趣味，提高文学修养。仿简书开发与实现系统实现了发表文章、评论、关注、赞同、收藏等功能，对于一个创作社区来说，比较全面了。简书满足用户创作的欲望，同时满足了个人树立威望的人性需求。

本课题研究的设计思想是实现网站前后端分离开发，同时进行前后端开发的目标，更好的掌握自身的技能，对思维逻辑有很好的提升，在开发的过程中，发现问题，解决问题，注意开发细节，同时写出比较不错的系统。

基于以上的原因与目的进行了仿简书开发与实现课题的研究，并实现简书官方网站的基本功能。

## 1.3 设计思想

本软件主要由前端功能开发和网站后台应用两大部分组成。其中前端功能包含网站的首页、注册页、登录页、个人中心页、写文章页、文章详情页等组成。其中功能按钮是使用AJAX进行异步HTTP请求，在不刷新页面的情况下获取数据，使得用户体验更高。在网站首页需要考虑用户是否登录，用户登录后隐藏掉注册和登录的按钮，展示用户头像。用户可以通过鼠标放在头像上选择操作，可以跳转到个人中心页，当然也可以选择注销账号。而在网站后台部分使用node.js进行HTTP请求的接口开发，使用MYSQL数据库存储用户的信息，文章的信息。设计思路如图1-1所示。

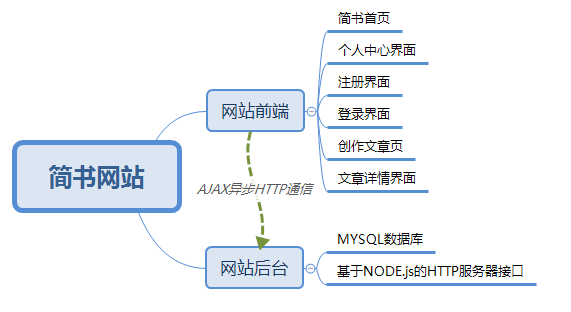


图1-1 设计思路图

## 1.4 开发技术简介

本课题采用NODE作为开发环境，进行网站的前后端开发，使用VUE的脚手架进行项目搭建，使用webpack 3.6版本进行代码的打包管理。前端使用.vue文件进行模版组件开发配合vue-router进行单页面路由的切换实现页面切换，vuex进行状态管理，在经过页面跳转的时候进行数据之间的传递。

网站后台开发，使用NODE.js连接操作MYSQL数据库进行存储数据，使用EXPRESS轻量级WEB框架进行服务器的HTTP接口进行开发，前端页面通过AJAX发送请求调用对应的HTTP接口返回前端需要的数据，实现简书网站系统的前后端对接。Node.js是一种通过JavaScript语言开发web服务端的开发语言，同时也是一个运行环境。在电脑中安装NODE的安装包即可使用node的环境进行网站开发，是一个可以跨平台，性能极高的后端开发语言。

## 1.5 环境简介

软件是基于node.js运行环境进行开发的后台，前端所用到的插件等也需要node软件自带的npm包进行依赖下载和管理，使用MYSQL数据库对网站数据的存取。本系统基本硬件、环境软件的开发环境如下：

1.系统的软硬件平台

内存：6GB ；

CPU：Intel(R) Celeron(R) CPU酷睿i3 3代系列；

硬盘：128GB固态；

操作系统： Windows 10(64位)；

2.开发语言和工具

开发语言：node.js；JavaScript；HTML；CSS

开发工具：Visual-Studio；NODE运行环境；WebPack打包管理工具

NODE运行环境是在电脑上可以通过命令行运行JavaScript代码的一种代码执行环境，Node.js是一个Javascript运行环境(runtime environment)，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对一些特殊用例进行优化，提供替代的API，使得V8在非浏览器环境下运行得更好。通过使用Node.js可以对JavaScript语言在非浏览器上执行，实现利用JavaScript语言进行网站后台服务器的开发。而Node支持的平台几乎包括了Windows、MAC、Linux等在内的所以主流平台。因此对于前端开发的人员开发完成的网站，使用Node.js作为后台语言是很好的选择。

# 第2章 系统需求分析

需求分析是软件开发中计划阶段的重要活动，也是软件生存、生产周期中的一个重要环节，在需求分析阶段分析系统在功能上需要“实现什么”，而不是去考虑如何“实现”具体的功能。系统需求分析的目标和目的是要把用户对待开发软件提出的 “需要”或“要求”进行分析、整理、确认后形成文档确定一个描述完整、清晰与规范的文档，把软件需要实现的功能定下来，写入到文档中去，还有需要去完成的工作也要明确。此外，软件中一些非硬性需求(如软件性能、可靠性、响应时间、可扩展性等)，软件设计的约束条件，明确运行过程中与其他软件的各种关系依赖等也是软件需求分析的目标。

## 2.1 系统需求分析

系统网站需求分析包括对网站应该提供的服务、如何对用户操作做出响应，以及网站系统在一些情况下的做出的行为的描述。在某些情况下，功能需求可能还需要声明系统不应该去做什么。仿简书网站的设计与实现是一个仿照简书网站进行开发的创作社区，首要实现的功能需求就是，在网站中创作文章，阅读优质文章。在此首页需求下，需要有不同的用户进行社区互动，同时为了保证数据的持久性，网站对应的服务器也不可以减少，还要有相对应的网站后台支持才可以配得上是一个网站系统。

经过对仿简书网站的设计与实现的调查分析，简书网站主要应该满足以下功能：

1.要求有友好网站页面，操作简单，易于用户使用社区进行创作与阅读。

2.要求有很好并且不错的系统兼容性，可以在不同的平台进行开发使用。

4.网站系统拥有用户注册、用户登录等基本功能。

4.网站应该有自己的服务器数据库，对不同用户的操作响应不一样的操作结果。

5.用户可以实现文章创作，对已经发表的文章进行修改，删除等。

6.对用户发表的文章可以及时传到服务器中，可以让其他用户实时查看到文章。

## 2.2 可靠性和可行性需求分析

简书是一个为了讲写作和阅读整合放在一起的网络网站系统。旨在为写作者打造最优秀的写作、阅读平台。简书目标用户群是几大类用户，他们可能属于不同群体，却有着共同的动机和特征。需要将自己的故事分享出来，让阅读者与之情感共鸣，是人们孤独在外时寻找爱和归属感的需求。另一类是将自己的想法分享，通过阅读进行学习，使自己不断进步，总结成功或者失败的原因等等。对用户应该分为两类，一是热爱写作，热衷于表达分享所思所想，输出干货的作者。二是热爱生活，聆听故事的文艺青年，或者热爱学习，不断找寻优质文章的读者。网站系统应该有如下功能：

1.进入网站的用户可以查看所有的文章，但不能对文章进行评论、收藏等操作

2.未注册账号的用户可以进行账号注册，未登录的用户可以进行用户登录

3.用户登录后可以进行写作，对他人的文章进行评论，收藏等

4.文章发表后应该通过网络传输及时的传回后台进行数据存储

5.网站首页刷新可以及时的拉取服务器中最新的文章显示，供读者进行阅读

6.作者应该有对已经发表过的文章进行修改、删除的功能，实现对作品的可控性

本软件主要由前端功能开发和网站后台应用两大部分组成。其中前端功能包含网站的首页、注册页、登录页、个人中心页、写文章页、文章详情页等组成。其中功能按钮是使用AJAX进行异步HTTP请求，在不刷新页面的情况下获取数据，使得用户体验更高。在网站首页需要考虑用户是否登录，用户登录后隐藏掉注册和登录的按钮，展示用户头像。用户可以通过鼠标放在头像上选择操作，可以跳转到个人中心页，当然也可以选择注销账号。而在网站后台部分使用node.js进行HTTP请求的接口开发，使用MYSQL数据库存储用户的信息，文章的信息等。

在对仿简书网站的设计与实现进行功能需求分析，可靠性及可行性进行分析后，我们就有了合理的设计策略，可以着手开始开发设计类似于简书的网站了。

# 第3章 系统设计

所谓设计是将一个实际的需求转换成相应的功能的主动过程，也就是对一种实现需求的描述。而系统设计目的是为了告诉用户系统具体将要实现什么，系统设计在描述系统时，不会采用计算机方面的专业术语，其描述重点在描述系统的功能上，使得用户能够完全理解。根据上一章对仿简书网站的设计与实现的系统需求分析，我们可以将该简书网站的功能分为以下几个模块。

## 3.1 功能模块设计

系统的功能模块简要明了概括了一个系统软件将要实现的功能以及各部分功能之间的联系。简书网站的设计与实现的整体功能模块结构如图3-1所示。

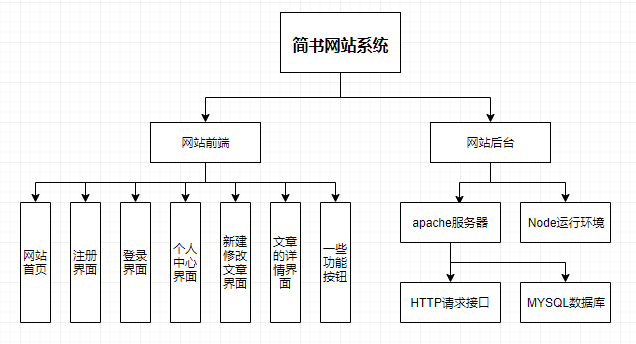


图3-1 功能模块图

## 3.2 系统流程设计

系统流程图是作为描绘系统物理模型的传统方法工具。系统流程图的基本思想是系统理念的将各个功能利用图形符号并以导航连接的形式描绘，以此表示信息在各个功能块之间流动的情况，并不是表示对信息进行如何具体实现处理的控制过程。简书网站的设计与实现的系统流程图简述了简书网站的逻辑过程，如图3-2所示。

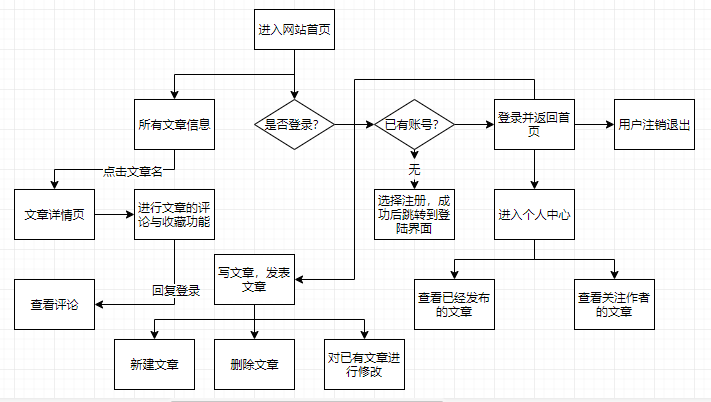


图3-2 系统流程图

## 3.3 系统数据结构设计

数据结构是[计算机](http://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA" \t "_blank)存储、组织[数据](http://baike.baidu.com/item/%E6%95%B0%E6%8D%AE)的方式。数据结构一般是指数据彼此之间存在一种或多种特定关联的[数据元素](http://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%85%83%E7%B4%A0)的集合。一般，想要得到高效的系统运行或者提高存储[效率](http://baike.baidu.com/item/%E6%95%88%E7%8E%87)，就必须进行精心设计。网站存储的用户信息，以前用户所发表的文章等信息，用户评论与文章的关联关系等都需要精心进行数据结构设计。

仿简书网站系统利用MySQL存储网站数据，需要建立一个独立的网站数据库，数据库包含用户表，文章表，评论表，收藏表，以及他们之间的一些关联表等等。

网站系统一般除了从服务器获取数据进行渲染以外，还有可能需要用到cookie和session的来提高网站的响应速度，在前端网站需要对服务端返回的数据进行JSON数据结构的转换等方便前端存储与使用。

对于用户表，需要用到唯一标识来知道是哪一位用户，使用id作为主键得以解决唯一标识的问题。还需要通过一个状态字段is\_actie进行判断该用户是否在线，还有存储用户的邮箱信息，手机号码，用户名，密码，头像，关注作者数，关注文章数等信息等等。用户表（users）的字段包括有以下内容。

id; //用户的唯一标识

name; //用户名

email; //用户邮箱

phone; //用户电话

password; //用户密码

avatar; //用户头像图片地址

is\_active; //用户在线信息

likes\_count; //收藏文章数等等

comments\_count；//文章评论数等等

利用MySQL数据库进行建立简书网站数据库，通过Node,js进行数据库的操作实现数据的存取功能，当然除了用户表（users）以外还有一些网站的关键表需要建立设计才能保证整体网站的稳定运行。

## 3.4 前端设计

在对简书网站的需求以及系统流程进行分析之后，需要对前端开发进行设计。前端项目利用Vue2.0版本的官方脚手架进行项目搭建，利用Webpack进行前端代码的打包管理，拥有npm包管理工具进行项目依赖的下载。在对进行系统界面开发的时候使用vue组件开发方式进行系统界面开发，包含视图组件Index.vue网站首页，Register.vue用户注册界面，Login.vue用户登录组件，Personal.vue个人中心界面组件，Writer.vue文章发布修改界面，Article.vue文章详情界面。还包含一个公共的组件如：Header.vue网站顶部组件，List.vue文章列表组件，Content.vue首页内容组件。前端开发就是通过开发组件，然后进行组件之间的调用传参进行实现。

对于各种界面之间的关系，使用vue-router进行页面之间的路由跳转以及传递参数，实现功能需求。和一些网站加载自调用或者用户操作调用的接口等等。

getArticleAll (); //通过 AJAX服务器发送HTTP请求，将获取到所有文章接口

readArticle (id, article); //请求查看文章详情接口

submitForm(); //登录请求接口

registerUser (); //注册请求接口

reviseArticle (); //发布修改文章请求接口

getArticles (users\_id); //获取指定用户请求接口

deleteArticle (); //删除文章请求接口

newArticle();// 新建文章请求接口

而函数的具体实现，我们可以在下一章做进一步了解，在此不做过多敷述。系统网站前端开发就是对界面进行样式开发，以及对按钮请求服务器数据接口的编写，并把获取到的数据通过数据绑定渲染到浏览器中供用户查看。

## 3.5 网站后台设计

网站系统后台设计包括对前端发送过来的数据进行存储，所有需要建立一个数据存放的地方。本课题使用了MySQL数据库进行数据存取。需要建立一个bibased数据库，以及用途不一样的表，用户表（users），文章表（articles），评论表（comments）等等。还包括服务器对前端请求进行的响应等等，需要编写服务器响应接口对前端请求进行响应，所以还需要编写后台服务器HTTP响应请求接口，其接口与前端请求接口相互映照。

# 第4章 系统实现

软件系统的设计和实现是软件工程过程中一个重要的阶段，在这一部分我们编码需要开发出可执行的软件系统。经过前几章的分析和设计，我们已经大致简书网站系统开发的设计思路，现在需要着手编码实现各个模块的具体功能了。

## 4.1 前端界面结构的实现

网站界面的实现分为.vue组件的开发和在跟组件APP.vue文件中挂载渲染。组件开发我们用到html编写网站界面结构，利用less编写网站样式，使用JavaScript脚本语言进行AJAX异步HTTP请求。系统中需要单独渲染的界面已经存放在src里面的views文件夹中。网站首页完成图如图4-1所示。



图4-1 网站面首页

### 4.1.1网站首页的开发

网站首页包含顶部导航栏还有一个用来展示推荐文章的内容块，我们利用contents将他包裹起来。所有网站首页需要引入一个自行开发的顶部组件Header.vue以及内容组件Content.vue进行HTML结构渲染。组成一个完整的简书网站系统首页。

使用Vue框架进行开发的项目，网站首页需要引入公共组件注册并在结构中使用，利用vue组件开发的功能实现代码模块化，便于管理和开发。首页中引入注册组件的代码如下。

*// 引入组件*

import Header from '@/components/Header.vue'

import Content from '@/components/Content.vue'

// 组件注册

export default {

name: 'Index',

components: {

Header,

Content

},

**data** () {

return {

}

},

// 在HTNL中使用组件

<template>

<div *class*="index body">

<Header></Header>

<Content></Content>

</div>

</template>

上述代码中的import就是通过Webpack配置引入其他资源的方法，而components:{}将引入组件进行注册的方法然后可以在HTML中像标签那样的方法使用它。

### 4.1.2 网站其他界面的开发

网站其他界面的开发也是类似于网站首页的开发，在引入组件之后进行注册，然后在HTMl中直接使用已经注册过的标签即可。本网站一共开发了如下几个需要独立展示的vue组件。如图4-2所示。



图4-2 所有界面组件

## 4.2 前端界面样式的实现

在网站界面需要达到预期要求的时候还需要进行CSS样式的开发，本课题使用LESS预编译编写网站的CSS样式，利用Webpack打包管理babel-core和less-loader插件进行less编译成为CSS加载在网站界面中。

### 4.2.1 CSS样式的处理

在编写使用CSS样式的时候，使用到了预编译语言less来对界面样式的编写，仿简书网站的实现与设计中使用Webpack进行项目的打包。所以在其中我们使用了以下代码进行打包的配置，让Webpack知道如何对CSS样式代码如何处理。

*// generate loader string to be used with extract text plugin*

function **generateLoaders** (loader, loaderOptions) {

const loaders = options.usePostCSS ? [cssLoader, postcssLoader] : [cssLoader]

if (loader) {

loaders.**push**({

loader: loader + '-loader',

options: **Object**.**assign**({}, loaderOptions, {

sourceMap: options.sourceMap

})

})

}

*// Extract CSS when that option is specified*

*// (which is the case during production build)*

if (options.extract) {

return ExtractTextPlugin.**extract**({

use: loaders,

fallback: 'vue-style-loader'

})

} else {

return ['vue-style-loader'].**concat**(loaders)

}

}

*// https://vue-loader.vuejs.org/en/configurations/extract-css.html*

return {

css: **generateLoaders**(),

postcss: **generateLoaders**(),

less: **generateLoaders**('less'),

sass: **generateLoaders**('sass', { indentedSyntax: true }),

scss: **generateLoaders**('sass'),

stylus: **generateLoaders**('stylus'),

styl: **generateLoaders**('stylus')

}

}

利用generateLoaders()；函数通过传参的不同使用不同的CSS样式处理方法，对编写的CSS各种格式的文件实现转化成前端界面需要的CSS样式。

### 4.2.2 CSS样式的编写

完成了对CSS不同文件的处理之后，开始着手编写界面样式，本课题中我们使用LESS进行样式编写。通过仿照简书官网的界面解析书写LESS，使得仿简书网站界面和简书官网样式看起来差不多。在样式实现的过程中，我们需要对标签样式进行统一处理，避免在各个不同浏览器之间使用导致样式错乱。以下是对样式初始化和根据不同组件进行编写的.less文件。如图4-3所示。



图4-3 所有外联的less样式文件

为了在界面中加载已经编写好的样式，需要在项目的入口文件main.js中引入，或者在单独组件内部进行组件的单独调用。这样就可以在使用组件的时候加载到相关联的样式了。

import 'element-ui/lib/theme-chalk/index.css';

import '@/style/common/font\_icon/iconfont.css'

*// 自己写的样式*

import './style/base.less';

import './style/common/sign.less';

import './style/common/list.less';

import './style/common/writer.less';

## 4.3 前端HTTP请求接口的实现

上面前端界面和前端样式的实现已经完成了前端需要展示的界面了，但一个网站系统只有静态界面是万万不行的。完整的简书网站应该可以与自己的服务器进行通信，在此小章中主要描述前端通过AJAX向服务器发送HTTP请求建立通信的过程。

### 4.3.1 用户相关接口的开发

一个社区创作网站基本需求需要有用户，所以需要能够进行用户注册，用户登录的基本功能。前端用户注册接口，需要将用户填写的用户名、密码、手机号等发送到服务器，根据服务器响应的状态进行判断是否注册成功，如果用户名已被注册或者用户信息输入违法等则提示用户重新注册账号。用户注册账号成功之后，在界面中提示用户你已经注册成功请选择登录，并通过路由插入跳转到用户登录界面。如下代码是用户注册的AJAX请求实现。

this.axios.**post**('/api/users/register', {

name: this.user.name,

phone: this.user.phone,

password: this.user.pass,

avatar: this.user.avatar

})

.**then**(function (response) {

**console**.**log**(response.data.code);

let data = response.data;

if (data.code === 'success') {

**console**.**log**(data.msg);

\_this.**$message**({

message: data.msg,

type: data.code

});

**setTimeout**(() => {

\_this.$router.**push**('/login')

}, 2000);

} else if(data.code === 'error') {

**console**.**log**(data.msg);

\_this.**$message**({

message: data.msg,

type: data.code

});

}

})

.**catch**(function (error) {

**console**.**log**(error);

});

网站有了注册功能之后，还需要对用户开发登录请求接口，用户通过输入用户名和密码，点击登录进行AJAX异步HTTP请求服务器。登录接口需要将用户的用户名、密码发送给后台服务器，经过后台服务器接口的信息对比返回登录是否成功，将后端返回的数据写入到浏览器sessionstorage中。然后跳转到网站首页，进行改变网站顶部的用户信息。用户登录接口实现代码如下。

this.axios.**post**('/api/users/login', {

name: this.user.name,

password: this.user.pass

})

.**then**(function (response) {

let data = response.data;

let user = data.user;

user.is\_active = true;

*// console.log(data);*

if(data.code === 'success') {

\_this.**$message**({

message: data.msg,

type: data.code

});

**setTimeout**(() => {

\_this.$router.**push**({

name: 'Index',

params: {

user

}

})

},1000)

window.localStorage.**setItem**('user',JSON.**stringify**(user));

\_this.$store.**commit**( 'getUser', user );

} else if (data.code === 'error') {

\_this.**$message**({

message: data.msg,

type: data.code

})

}

else { *//返回的不是这两种状态*

**console**.**log**('发送未知的错误');

}

})

这样我们实现了前端用户相关的接口开发。

### 4.3.2 文章相关接口的开发

在网站加载的时候，首页需要自调用获取推荐文章接口获取数据由前端通过数据渲染展示文章。在用户发布文章的时候也需要与后台服务器进行通信，将文章的最新信息传递到数据库中进行存储。修改文章也是需要一些特定的文章修改接口，阅读也需要阅读文章接口，还需要增加文章删除的接口，获取一些特定文章的接口：如关注作者的文章，收藏的文章等等。因此需要对文章接口进行开发，如下一些重要的文章节接口。

getArticleAll()； //获取所有文章

*// 获取所有文章*

**getArticleAll**: function(num) {

let \_this = this;

\_this.axios.**post**('/api/article/getArticleAll', {

num: num,

}).**then**(function (response) {

let data = response.data.result;

**console**.**log**(data);

\_this.$store.**commit**( 'loadArticles', data );

\_this.article = \_this.**copyArr**(data);

}).**catch**(function (error) {

**console**.**log**(error);

})

},

getArticle ()； //获取单个文章

*// 获取单个文章*

**getArticle**(id) {

let \_this = this;

\_this.axios.**post**('api/article/getArticle',{

id: id

})

.**then**(function (response) {

let data = response.data.result[0];

\_this.article = data;

})

.**catch**(function (error) {

**console**.**log**(error);

});

},

newArticle ()； //新建文章

*// 新建文章函数*

**newArticle**: function () {

let \_this = this;

\_this.axios.**post**('api/article/newArticle',{

title: this.**currentTime**(),

body: '',

user\_id: \_this.$store.state.user.id

})

.**then**(function (response) {

let data = response.data;

**console**.**log**(data);

\_this.**getArticles**();

})

.**catch**(function (error) {

**console**.**log**(error);

});

},

deleteArticle ()； //删除文章

*// 删除文章*

**deleteArticle**: function (id) {

let \_this = this;

\_this.axios.**post**('api/article/deleteArticle',{

user\_id: \_this.$store.state.user.id,

id: id

})

.**then**(function (response) {

let data = response.data;

*// 重新获取数据库的数据*

\_this.**getArticles**();

**console**.**log**(data);

})

.**catch**(function (error) {

**console**.**log**(error);

});

},

reviseArticle ()； //修改文章

*// 修改文章*

**reviseArticle**() {

let \_this = this;

\_this.axios.**post**('api/article/reviseArticle',{

title: \_this.title,

body: \_this.Editor.txt.**html**(),

text: \_this.Editor.txt.**text**().**slice**(0, 80) + '...',

id: \_this.id

})

.**then**(function (response) {

let data = response.data;

\_this.**getArticles**();

})

.**catch**(function (error) {

**console**.**log**(error);

});

},

getArticles ()； //获取特定用户的所有文章

*// 获取用户所有文章*

**getArticles**() {

let \_this = this;

\_this.axios.**post**('api/article/getArticles',{

user\_id: \_this.$store.state.user.id

})

.**then**(function (response) {

let data = response.data.result;

\_this.articles = \_this.**copyArr**(data);

})

.**catch**(function (error) {

**console**.**log**(error);

});

},

还有一些接口可能并没有加载到本论文中，这些文章接口配合界面结构HTML调用对应的前端文章HTTP接口对服务器发送请求接收到服务器响应之后进行页面渲染。

## 4.4 网站数据库的实现

本网站使用MySQL数据库进行网站数据的存取，使用Node.js进行连接操作数据库，使用sql数据来对数据库增删改查。

### 4.4.1 数据库用户表的设计

数据库用户表用来存储网站用户的信息，包括用户唯一标识id字段，name用户名字段，phone手机号字段，password密码字段，is\_active判断用户是否在线字段等等。需要创建不同的字段来存储数据，每一个字段都具备不同的功能。具体用户表创建SQL代码如下。

CREATE TABLE `users` (

`id` int(10) UNSIGNED NOT NULL,

`name` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`email` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`phone` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`password` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`avatar` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`confirmation\_token` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`is\_active` smallint(6) NOT NULL DEFAULT '0',

`question\_count` int(11) NOT NULL DEFAULT '0',

`answers\_count` int(11) NOT NULL DEFAULT '0',

`comments\_count` int(11) NOT NULL DEFAULT '0',

`favorites\_count` int(11) NOT NULL DEFAULT '0',

`likes\_count` int(11) NOT NULL DEFAULT '0',

`followers\_count` int(11) NOT NULL DEFAULT '0',

`followings\_count` int(11) NOT NULL DEFAULT '0',

`settings` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`remember\_token` varchar(100) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`created\_at` timestamp NULL DEFAULT NULL,

`updated\_at` timestamp NULL DEFAULT NULL,

`api\_token` varchar(64) COLLATE utf8mb4\_unicode\_ci NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

### 4.4.2 数据库文章表的设计

数据库文章表用来存储网站用户发表的文章信息，包括文章唯一标识id字段，title文章字段，body字段用来获取前端编写文章引入的富文本编辑器中的html代码，user\_id字段用户标识该文章的作者，text截取富文本编辑器中前80字段的纯文本内容，还有一些因为功能需要创建的表。需要创建不同的字段来存储数据，每一个字段都具备不同的功能。具体文章表创建SQL代码如下。

CREATE TABLE `articles` (

`id` int(10) UNSIGNED NOT NULL,

`title` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`body` text COLLATE utf8mb4\_unicode\_ci NOT NULL,

`user\_id` int(10) UNSIGNED NOT NULL,

`comments\_count` int(11) NOT NULL DEFAULT '0',

`followers\_count` int(11) NOT NULL DEFAULT '1',

`close\_comment` varchar(8) COLLATE utf8mb4\_unicode\_ci NOT NULL DEFAULT 'F',

`is\_hidden` varchar(8) COLLATE utf8mb4\_unicode\_ci NOT NULL DEFAULT 'F',

`created\_at` timestamp NULL DEFAULT NULL,

`updated\_at` timestamp NULL DEFAULT NULL,

`text` text COLLATE utf8mb4\_unicode\_ci NOT NULL,

`reads\_count` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

### 4.4.3 数据库表的整体设计

由于网站系统的信息较多，需要各种不同的表进行关联，所以需要为以后网站扩建优化做出可扩展性需要创建一些之后的可能用到的表。整个数据库的表结构如图4-4所示。

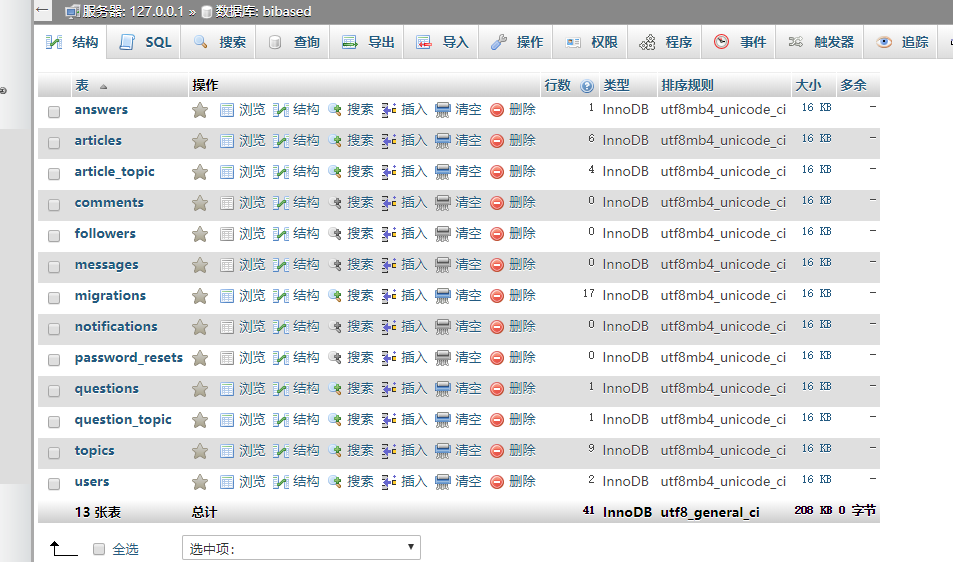


图4-4 数据库整体表

## 4.5 网站服务端的实现

一个网站后台服务器需要对数据库进行操作，还需要对前端发送的请求进行响应。该小章主要讲述如何进行本课题服务后台的开发实现。

### 4.5.1 服务器后台项目搭建

仿简书网站后台使用基于Node的Express框架进行快速搭建。在服务器搭建中使用到的插件还有库等如图4-5所示。

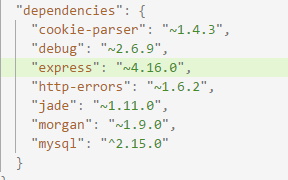


图4-5 服务器依赖

使用Express框架自带的路由方法进行后台响应接口的编写实现。

### 4.5.2 服务器对数据库的操作

使用Node.js编写JavaScript代码进行连接数据的选项的配置，配置代码如下。

*// 数据库连接配置*

module.exports = {

mysql: {

host: 'localhost',

user: 'root',

password: '',

database: 'bibased',

port: '3306'

}

}

使用Node.js编写JavaScript代码进行sql语句的配置，方便服务端接口文件调用，配置代码如下。

*// sql语句*

var sqlMap = {

*// 用户*

user: {

add: 'insert into users(name, phone, password, avatar) values (?, ?, ?, ?)',

select: 'SELECT \* FROM users WHERE find\_in\_set(?, name)',

login: 'SELECT \* FROM users WHERE find\_in\_set(?, name) && find\_in\_set(?, password)'

},

*// 文章语句*

article: {

add: 'insert into articles(title, body, user\_id) values (?, ?, ?)',

selects: 'SELECT \* FROM articles WHERE find\_in\_set(?, user\_id)',

selectAll: 'SELECT \* FROM articles',

select: 'SELECT \* FROM articles WHERE find\_in\_set(?, id)',

deleteArticle: 'DELETE FROM articles WHERE user\_id = ? and id = ?',

reviseArticle: 'UPDATE articles SET title = ? , body = ?, text = ? WHERE id = ?'

}

}

module.exports = sqlMap;

上面代码就是根据数据库的用户名等进行连接配置，以及为了响应前端接口发送的请求可能需要进行的数据库操作语句。这样设计的目标是为了将操作数据库以及后台接口分装成不同模块进行开发，方便系统的实现。

### 4.5.3 服务器HTTP响应接口的设计

作为一个网站的后台服务器，需要对前端发送过来的请求进行不同的响应，有些接口只需要进行返回一个状态，表示成功或者失败。但也有一些需要进行数据库的操作，从数据库中查找数据并进行数据处理，将处理过的数据发送到前端界面。

主要分为用户服务器接口和文章接口。用户请求响应接口代码如下。

*//增加用户接口*

router.**post**('/register', (req, res) => {

let params = req.body;

*// 查询昵称*

conn.**query**(sqlUser.select, [params.name], (err, result) => {

if(err) {

**console**.**log**(err);

}

*// 数据库返回数组大于1，代表已有*

if (result.length >= 1){

res.**json**({

code: 'error',

msg: '该昵称已被注册，请换一个'

});

} else { *//未被注册执行插入操作*

conn.**query**(sqlUser.add, [params.name, params.phone, params.password, params.avatar], (err, result) => {

if(err) {

**console**.**log**(err)

}

if(result) {

**console**.**log**('注册成功');

res.**json**({

code: 'success',

msg: '注册成功'

});

}

})

}

})

});

*//用户登录验证借口*

*//增加用户接口*

router.**post**('/login', (req, res) => {

let params = req.body;

*// 查询昵称*

conn.**query**(sqlUser.login, [params.name, params.password], (err, result) => {

if(err) {

**console**.**log**(err);

} else {

*// 数据库返回数组大于1，代表已有*

if (result.length >= 1){

**console**.**log**('登录成功');

res.**json**({

code: 'success',

msg: '登录成功',

user: **copyObject**(result[0])

});

} else { *// 用户名和密码不匹配*

**console**.**log**('用户名或密码不正确');

res.**json**({

code: 'error',

msg: '用户名或密码不正确'

})

}

}

})

});

文章后台响应接口代码如下。

*//发布文章接口*

router.**post**('/newArticle', (req, res) => {

let params = req.body;

**console**.**log**(params);

*// 查询昵称*

conn.**query**(sqlAll.add, [params.title, params.body, params.user\_id], (err, result) => {

if(err) {

**console**.**log**(err);

**console**.**log**('新增错误');

}

*// 数据库返回符合要求的文章数组*

if (result) {

*// console.log(result);*

res.**json**('新增成功');

}

})

});

*//修改文章reviseArticle*

router.**post**('/reviseArticle', (req, res) => {

let params = req.body;

**console**.**log**(params);

*// 查询昵称*

conn.**query**(sqlAll.reviseArticle, [params.title, params.body, params.text, params.id], (err, result) => {

if(err) {

**console**.**log**(err);

**console**.**log**('修改错误');

}

*// 数据库返回符合要求的文章数组*

if (result) {

**console**.**log**('修改成功');

res.**json**({

result

})

}

})

});

*//获取用户所有文章接口*

router.**post**('/getArticles', (req, res) => {

let params = req.body;

*// 查询昵称*

conn.**query**(sqlAll.selects, [params.user\_id], (err, result) => {

if(err) {

**console**.**log**(err);

}

if (result) {

*// console.log(result);*

res.**json**({

result

});

}

})

});

*//获取用户所有文章接口*

router.**post**('/getArticleAll', (req, res) => {

let params = req.body;

*// 查询昵称*

conn.**query**(sqlAll.selectAll, (err, result) => {

if(err) {

**console**.**log**(err);

}

if (result) {

*// console.log(result);*

res.**json**({

result

});

}

})

});

*// 获取单个文章*

router.**post**('/getArticle', (req, res) => {

let params = req.body;

*// 查询昵称*

conn.**query**(sqlAll.select, [params.id], (err, result) => {

if(err) {

**console**.**log**(err);

}

if (result) {

res.**json**({

result

});

}

})

});

*// 删除文章*

router.**post**('/deleteArticle', (req, res) => {

let params = req.body;

**console**.**log**(params);

*// 查询昵称*

conn.**query**(sqlAll.deleteArticle, [params.user\_id, params.id], (err, result) => {

if(err) {

**console**.**log**(err);

}

if (result) {

**console**.**log**(result);

res.**send**('删除成功');

}

})

});

这就基本实现了网站服务器HTTP响应接口的开发实现。同时页完成了仿简书网站的基本功能开发的工作。

# 第5章 系统测试

在完成整个网站的实现之后，我们应必要的对整个网站前后端进行全方位的测试。系统测试可以将我们所预期的系统的功能的样子展示出来，于此同时还可以发现一些在正式投入使用之前的漏洞和网站缺陷。只有系统通过了所有的测试，并对发现的缺陷和漏洞采取相应的维护修复，这样才能完成一个合格的完整功能的网站。

## 5.1 测试方法

系统的测试方法多种多样，通常我们采用黑盒测试或白盒测试。黑盒测试也可以称作功能测试，它是通过测试系统功能功能是否都能正常使用的系统测试。在测试中，把程序看作一个整体的不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，在程序接口进行系统测试，它只负责检查程序功能是否能够实现需求规格说明书的规定预期效果，程序是否可以适当地接收输入数据或者操作而产生正确的输出信息。黑盒测试是在程序外部结构，不会考虑内部逻辑结构，主要针对软件界面效果和软件功能进行测试。

黑盒测试是主要切入点在软件的宏观表现，而我们现在主要测试所设计的系统功能是不是符合要求，并不是关注具体内部实现；白盒测试则恰恰相反，它更加重视系统内部操作是否合乎要求，要对每步的实现进行仔细测试。鉴于白盒测试的测试工作步骤相对复杂，这里我们采用黑盒测试的方法检测系统的完成度，使用白盒代码的方法测试系统接口。另外进行一些HTTP接口的参数测试，查看在请求的是否是否返回我们预期想要的结果。

## 5.2 网站首页按钮功能测试

由于本课题实现的是一个网站系统，需要进行测试的是否能够正确在浏览器中打开网站相对应的网站界面，打开正确的对网站界面预示者网站界面功能完整。网站界面包含网站首页，用户中心页，注册、登录界面，还有用户进行文章编写界面，文章详情页。系统首页如图5-1所示。测试网站系统界面部分功能用例分析如表5-1所示。



图5-1 系统首页

|  |  |  |  |
| --- | --- | --- | --- |
| 测试编号 | 测试目标 | 测试结果 | 错误原因 |
| 1 | 进入网站系统首页 | 仿简书首页被打开 | 无 |
| 2 | 点击“注册”打开用户注册界面 | 用户注册界面被打开 | 无 |
| 3 | 点击“登录”打开用户登录界面 | 用户登录界面被打开 | 无 |
| 4 | 点击“网络对战（客户端）”打开客户端界面 | 网络对战客户端界面被打开 | 无 |
| 5 | 点击“写文章”打开用户文章编写界面 | 打开之后跳转到登录界面 | 用户未登录不能进行文章发表 |
| 6 | 点击“用户头像，选择个人中心”打开用户个人中心界面 | 个人中心界面被打开 | 无 |

表5-1 对弈选项测试用例

## 5.3 网站编写文章功能测试

编写文章功能测试主要进行，测试用户能否进行文学创作，在创作成功之后能否正确传递到网站后台服务器进行数据存储。在新建文章成功之后是否可以正确的预期进行已发表进行修改，或者进行删除。用户文章创作界面如图5-2所示。



图5-2 文章创作界面

## 5.4 网站文章详情展示功能测试

在网站首页可以获取到文章的信息，但因为文章内容过长的原因不能将文章的全部内容全部展示在网站首页中。因此，我们需要编写一个专门的详情页用户进行文章的阅读，在首页点击文章名可以打开一个新的文章详情页。在文章详情界面，用户可以进行文章的阅读或者在底部进行文章的评论。文章详情展示界面如图5-3所示。



图5-3 文章详情展示界面

|  |  |  |  |
| --- | --- | --- | --- |
| 测试编号 | 测试内容 | 测试结果 | 错误原因 |
| 1 | 点击新建文章 | 在文章栏中增加一个文章名 | 无 |
| 2 | 点击文章后边的删除图标 | 文章被进行删除 | 无 |
| 3 | 点击文章 | 右边编辑器展示文章的完整信息 | “无 |
| 4 | 点击文章修改 | 将重新修改过的文章进行发布 | 无 |

表5-2 文章编写功能用例

## 5.5 用户评论功能测试

在网站社区，交流算是一个很重要的环节，在这一小章中，我们进行对评论功能的测试。评论测试主要是测试在用户输入评论并点击发表之后是否能够及时的在界面中展示，在此我们需要对文章进行评论查看行为。系统评论功能测试如图5-4。



图5-4 用户评论功能测试

# 总 结

最初开始毕业课程设计是在2018年的3月，一直到现在将课程设计完成经过了两个多月了。回想起来，在最开始的有过迷茫不知道应该选择什么样的课题作为毕业设计，后来经过老师提醒用自己擅长的方向来进行毕业设计的实现完成，我选择了以前端方面进行发掘，确定了以网站类型的课题进行实现完成。之后找过很多网站，发现它们商业性太强，最后确定了对简书网站的发掘，感觉简书网站的界面简洁明了，广告几乎没有，因此放弃了毕业设计更为常见的个人博客搭建题目而转为仿照简书开发一个创作社区类型的网站。这样感觉其质量价值更加服务为人民服务的理念，也可以更好的进行个人开发的锻炼。

在确定题目之后，开始构思毕业设计的实现，考虑到只是个人开发，把简书全部的功能进行实现是不现实的。经过老师的谆谆教导和同学们的无私帮助我选择进行简书网站的基本功能开发，在实现最基本的创作、阅读功能之后进行开发，评论、收藏、关注作者等一些比较需要的功能。互联网出现到现在，把许多人的阅读习惯从课本转化到了网络。未来需要更多的、单纯的阅读网站社区，因此在18年毕设我们着手进行创作社区的开发也是可行的。

在具体实现的时候因为学习了前端Vue框架，因此选用其官方文档的脚手架工具进行前端项目的搭建开发，使用Node内嵌的npm包进行项目依赖的下载。在开发工具上选用了Visual-Studio代码编辑器，选择了代码管理工具github进行项目开源。在实现本设计之前，由于后台开发了解不深，仿简书网站使用了更适合前端人员的Node.js进行服务器开发，使用Express进行网站后台的搭建。在数据存储方面使用了MySQL数据库进行网站数据存取。

通过深入的分析，我逐步抓住明确实现一个完整网站所需的要点。前端项目与后台进行结构分离，在界面之间使用.vue文章开发界面组件得以高效率的复用，使用路由跳转传参实现目标界面的展示。在本项目中几乎所有的前端请求都使用了AJAX异步通信技术进行对服务器的HTTP请求，保证了在页面不刷新的情况下完成用户操作需要实现的功能。一个好的网站不仅需要在体验上效果良好，还应该具备特有的、优秀的社区特效才可更好的发展。

# 致 谢

时间如白驹过隙，是时候踏出校园出去看看了。在临近毕业之际，我非常感谢学校给了大学生活的体验和学习，让我可以独立完成毕业设计。我很感谢我的导师李爱玲老师，她不仅是我的毕业设计老师，在本科阶段也是我的数据结构课程的授课教师。在外出实习工作中，数据结构的熟悉使得我的工作更加顺利。由衷的感谢，李老师不仅仅是因为在课程设计的过程中给了我许多建设性的指导意见，使我顺利完成毕业设计的任务。还有在课程学习中，李老师同时是我的授业恩师，在课程授课中讲课令我们好学易懂，学到了数据结构相关的知识，在课余时间也给我们分享一些社会、人生经验，在此我非常感激李爱玲老师！当然，对于本科阶段教授过我课程的其他老师，我也由衷的表示感谢！还要感谢的是与我朝夕相处四个年头的同学们。在课程设计的过程中，对于一些我不懂的方面对我进行帮助。若得不大家的帮助，我想许多困惑会难以解决。所以，在以后的学习和生活中，我会铭记大家给予我的帮助，努力战胜新的挑战。大学就要结束了，虽然不舍，可终究人还是应该往前奋勇前进，挥一挥道别，诉说不舍之情。最后再次对李爱玲老师和同学们表示诚挚的感谢！

# 参考文献

[1] Jeremy Keith (作者) JavaScript DOM编程艺术：杨涛 , 王建桥 , 杨晓云 (译者) 2011年人民邮电出版社出版图书

[2] 关涛 CSS从入门到精通 化学工业出版社 2009.08

[]刘贵国 Html+JavaScript网页制作与开发完全学习手册

[4]耿国华：数据结构—用C语言描述（第2版）高等教育出版社 北京 2015.7

[5]王珊 萨师煊，数据库系统概论 高等教育出版社 北京 2014.9

[6]严蔚敏，吴伟民.数据结构：C语言版[M].北京：清华大学出版社，2014.11

[7]谢希仁 计算机网络（第6版） 电子工业出版社 北京 2013.6

[8] (美)(Nicholas C.Zakas)扎卡斯 JavaScript高级程序设计（第3版） 人民邮电出版社 2017

[9]弗拉纳李东博根（David Flanagan，美）Javascript权威指南 械工业出版社 2007.8

[10]李东博 HTML5+CSS3从入门到精通 清华大学出版社 2013.6

[11]陆凌牛 HTML 5与CSS 3权威指南 机械工业出版社 2011.4

[12]余果 Web全栈工程师的自我修养 人民邮电出版社 2015.9

[13]张耀春，黄轶，王静，苏伟，王瑾，殷献勇； Vue.js权威指南 电子工业出版社 2016.9

[14]梁睿坤 Vue2实践揭秘 电子工业出版社 2017.4

[15]吴浩麟 深入浅出Webpack 电子工业出版社2018

[16]（美）BYVoid, (作者), (译者) 人民邮电出版社 2012.7

[17]（美）深入浅出Node.js人民邮电出版社 2013.12

[18]vue官方文档 https://cn.vuejs.org/

[19]Webpack 中文文档 https://www.webpackjs.com/

[20]Node中文网 http://nodejs.cn/

[21] Express 4.x API 中文手册 http://www.expressjs.com.cn/4x/api.html

[22] Axios 中文说明 https://www.kancloud.cn/yunye/axios/234845

[23]Vuex官方文档 https://vuex.vuejs.org/zh-cn/api.html

[24]百度百科 系统测试说明