



中山大學  
SUN YAT-SEN UNIVERSITY

# 本科生毕业论文（设计）

## Undergraduate Graduation Thesis（Design）

题目 Title: 基于 Node.js 的个人博客  
系统的设计与实现

院系  
School (Department): 数据科学与计算机学院

专业  
Major: 软件工程（移动信息工程）

学生姓名  
Student Name: 刘鹏飞

学号  
Student No.: 15352234

指导教师(职称)  
Supervisor (Title): 温武少（教授）

时间: 2019 年 5 月 7 日

Date: May 7, 2019

## 说 明

1. 毕业论文（设计）的写作格式要求请参照《中山大学本科生毕业论文的有关规定》和《中山大学本科生毕业论文（设计）写作与印制规范》。
2. 除完成毕业论文（设计）外，还须填写三份表格：
  - （1）表一 毕业论文（设计）开题报告；
  - （2）表二 毕业论文（设计）过程检查情况记录表；
  - （3）表三 毕业论文（设计）答辩情况登记表。
3. 上述表格均可从教务部主页的“下载中心”处下载，如表格篇幅不够，可另附纸。每份毕业论文（设计）定稿装订时应随同附上这三份表格。
4. 封三是毕业论文（设计）成绩评定的主要依据，请认真填写。

## Instruction

1. Please refer to '*The Guidelines to Undergraduate Graduation Thesis (Design) at Sun Yat-sen University*' and '*The Writing and Printing Format of Undergraduate Graduation Thesis(Design) at Sun Yat-sen University*' for anything about the thesis format.
2. Three forms should be filled up before the submission of the thesis (design):
  - （1）Form 1: Research Proposal of Graduation Thesis.
  - （2）Form 2: Process Check-up Form.
  - （3）Form 3: Thesis Defense Performance Form.
3. All the above forms could be downloaded on the website of the Office of Education Administration. If there is not enough space in the form, please add extra sheets. Each thesis (design) should be submitted together with the three forms.
4. The form on the inside back cover is the grading sheet. Please fill it up before submission.

毕业论文（设计）成绩评定记录  
Grading Sheet of the Graduation Thesis (Design)

<p>指导教师评语 Comments of Supervisor:</p> <p>本论文基于 postgresSQL 数据库,以 Node.js 搭建后台,前端基于 React 技术栈,设计并实现了一个个人博客网站,可展示在其他博客平台所收藏的文章的功能,有一定的参考意义。本论文虽然选题不算新颖,但作者能够系统化地利用所学新知识研究这个问题,体现其具备利用专业计算机知识分析解决实际工程问题的能力。</p> <p>论文总体内容结构完整,写作基本规范,体现该生可解决计算机工程技术的实际问题,已经达到了中山大学本科学士学位论文要求水平。</p>	
<p>成绩评定 Grade:</p>	
<p>指导教师签名 Supervisor Signature :</p>	<p>Date:</p>
<p>答辩小组意见 Comments of the Defense Committee:</p>	
<p>成绩评定 Grade:</p>	
<p>签名: Signatures of Committee Members</p>	<p>Date:</p>
<p>院系负责人意见 Comments of the Academic Chief of School:</p>	
<p>成绩评定 Grade:</p>	

签名

Signature:

院系盖章

Stamp:

Date:

**表一：毕业论文（设计）开题报告**  
**Form 1: Research Proposal of Graduation Thesis (Design)**

论文（设计）题目：基于 Node.js 的个人博客系统的设计与实现 Thesis (Design) Title:	
<p>（简述选题的目的、思路、方法、相关支持条件及进度安排等） （Please briefly state the research objective, research methodology, research procedure and research schedule in this part.）</p> <p>选题目的：实一个既能个性化定制网页效果，又能展示其他博客平台所收藏的文章的个人博客。在实现个性化的同时帮助博主整合学习资源，方便回顾收藏的文章。</p> <p>思路：准备实现的系统是一个 B/S 架构的 Web 应用，后端主要涉及数据库读写操作，是一个 I/O 密集型应用,因此考虑用 Node.js 作为开发语言。前端准备用 React 全家桶实现一个单页应用，数据库准备选用一个关系型数据库，例如 MySQL 或 PostgreSQL。</p> <p>方法：首先，进行需求分析，根据需求设计好数据库结构。然后，分析哪些数据需要服务端和客户端通信，设计好前后端交互的 API 及交互的数据的格式。接着，设计好前端大体页面样式，并设计好前端路由的 URL。最后根据上面的设计思考项目架构与模块划分，然后再进行开发实现。</p> <p>相关支持条件：要实现该项目，需要熟练掌握数据库、HTTP 协议相关知识，熟练使用 JavaScript 语言，并会使用一个 Node.js 的 Http 服务器框架，掌握 React 相关技术栈。</p> <p>进度安排：预估整个毕业设计于 2018 年 12 月~2019 年 5 月完成，具体安排如下：</p> <p>2018 年 12 月~2019 年 1 月：进行需求分析，选定并学习相关技术栈；</p> <p>2019 年 1 月~2019 年 2 月：进行数据库设计、前后端 API 交互设计，项目整体架构设计；</p> <p>2019 年 2 月~2019 年 3 月：搭建好数据库，完成后端功能；</p> <p>2019 年 3 月~2019 年 4 月：完成前端功能；</p> <p>2019 年 4 月~2019 年 5 月：项目测试与修改，论文撰写与修改。</p>	
Student Signature:	Date:

指导教师意见

Comments from Supervisor:

- 1.同意开题
- 2.修改后开题
- 3.重新开题
- 1.Approved(    )
2. Approved after Revision (    )
3. Disapproved(    )

Supervisor Signature:

Date:

**表二：毕业论文（设计）过程检查情况记录表**  
**Form 2: Process Check-up Form**

指导教师分阶段检查论文的进展情况（要求过程检查记录不少于 3 次）  
The supervisor should check up the working process for the thesis（design）and fill up the following check-up log. At least three times of the check-up should be done and kept on the log.

**第 1 次检查（First Check-up）：**

- 学生总结  
Student Self-summary:
- 1. 完成初稿；
  - 2. 参考文献可能太少，需要后续补充

- 指导教师意见  
Comments of Supervisor:
- 1. 内容稍显单薄；
  - 2. 参考文献太少了；

**第 2 次检查（Second Check-up）：**

- 学生总结  
Student Self-summary:
- 1. 增加了几篇参考文献；
  - 2. 在前端实现部分，增加了原创文章编辑模块的实现论述；

- 指导教师意见  
Comments of Supervisor:
- 1. 有些部分表述略显口语化，需要进行修改；
  - 2. 论文的立项依据再展开一些讲；

**第 3 次检查（Third Check-up）：**

- 学生总结  
Student Self-summary:
- 1. 对语言表述进行了修改，用了更书面的表达方式；

- 指导教师意见  
Comments of Supervisor:
- 1. 背景部分谈一下与通用博客系统相比，有哪些创新点；
  - 2. 再增加一点参考文献；

### Fourth Check-up

**Student Self-summary:**

- 指导教师意见 (Comments of Supervisor):

- 学生签名 (Student Signature):

指导教师签名 (Supervisor Signature):

日期 (Date) :

指导教师意见 Comments of Supervisor:

- 指导教师签名 (Supervisor Signature):

日期 (Date) :



表三：毕业论文（设计）答辩情况登记表  
Form 3: Thesis Defense Performance Form

答辩人 Student Name		专 业 Major	
论文（设计）题目 Thesis (Design) Title			
答辩小组成员 Committee Members			
<div>答辩记录 Records of Defense Performance:</div> <div></div> <div>记录人签名 (Clerk Signature):</div> <div>日期 (Date):</div>			

## 学术诚信声明

本人所提交的毕业论文，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、图片资料均真实可靠。除文中已经注明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的作品或成果。对本论文的研究作出重要贡献的个人和集体，均已在文中以明确的方式标明。本毕业论文的知识产权归属于培养单位。本人完全意识到本声明的法律结果由本人承担。

本人签名：

日期：

## Statement of Academic Integrity

I hereby acknowledge that the thesis submitted is a product of my own independent research under the supervision of my supervisor, and that all the data, statistics, pictures and materials are reliable and trustworthy, and that all the previous research and sources are appropriately marked in the thesis, and that the intellectual property of the thesis belongs to the school. I am fully aware of the legal effect of this statement.

Student Signature:

Date:

## 【摘 要】

~~(\* 中文摘要不少于 300 字。语言力求精练、准确，以 300—500 字为宜。)~~

本文介绍了一个基于 `postgreSQL` 数据库，以 `Node.js` 搭建后台，前端基于 `React` 技术栈的个人博客网站的设计与实现过程。该网站后台系统定时爬取博主在思否、简书两大主流博客平台收藏的文章并展示，支持博主撰写、修改和删除原创文章，同时支持博客浏览者浏览博主收藏和原创的文章，注册并登录该网站后还可以对博主的原创文章点赞、评论，以及回复他人的评论。

▲  
【关键词】：关键词 1 博客；`postgreSQL` 数据库；`Node.js` 关键词 2；`React` 技术栈关键词 3；÷  
关键词 4  
▲

带格式的：缩进：首行缩进： 0.76 厘米

设置了格式：字体颜色：自动设置

设置了格式：字体：（默认）楷体，（中文）楷体，非加粗



[ABSTRACT]

(※ 英文摘要以 250—400 个实词为宜，严格使用英文标点符号。中、英文摘要意思要基本相同。在撰写完英文摘要后，请执行一次自动拼写检查，以减少英文拼写错误的可能性。※)

This paper introduces the design and implementation process of a personal blog website based on the PostgreSQL database, with Node.js built in the back-end system and front-end based on React technology stack. The back-end system of the website regularly crawls and displays the articles collected by the bloggers on the two major blog platforms, such as Segment Fault and Jianshu. At the same time, it supports bloggers to compose, modify and delete original articles, and supports visitors to browse bloggers' collections and original articles. After registering and logging in to the site, vistor can also like and comment on blogger's original articles, and respond to the comments from others.

[Keywords]: KeywordBlog4; postgresqlkeyword2; Node.jskeyword3; React

设置了格式: 字体颜色: 自动设置

设置了格式: 字体颜色: 自动设置

设置了格式: 字体颜色: 自动设置

设置了格式: 字体颜色: 自动设置

设置了格式: 字体颜色: 自动设置

设置了格式: 字体颜色: 自动设置

设置了格式: 字体颜色: 自动设置

设置了格式: 字体颜色: 自动设置

设置了格式: 字体颜色: 自动设置

设置了格式: 字体颜色: 自动设置

## 目 录

(\* 目录放在正文之前，中英文摘要之后。目录至少有二级目录，即包含正文的章和节及其页码，最好有三级目录，即包含正文的章、节和小节及其页码。目录还要包含参考文献及其页码。中英文摘要及目录本身及其页码可不出现在目录中。请使用由 Word 等排版软件自动生成目录，不要自己手工做目录。\*)

<b>第一章</b>	<b>绪论</b>	<b>2</b>
1.1	背景	2
1.2	论文结构简介	3
<b>第二章</b>	<b>系统相关技术概述</b>	<b>4</b>
2.1	POSTGRESQL	4
2.2	网络爬虫	5
2.3	AJAX 技术	6
2.4	NODE.JS	7
2.5	RESTFUL API	8
2.6	REACT.JS	8
2.7	REDUX	9
<b>第三章</b>	<b>系统分析和设计</b>	<b>11</b>
3.1	系统可行性分析	11
3.1.1	技术可行性	11
3.1.2	经济可行性	11
3.1.3	操作可行性	12
3.1.4	法律可行性	12
3.2	需求分析	12
3.2.1	功能需求分析	12
3.2.2	用例分析	13
3.3	系统概要设计	13
3.4	系统数据库设计	16
3.5	后端 API 设计	21
3.6	前端路由设计	23
3.7	前端 REDUX STORE 设计	24
<b>第四章</b>	<b>详细设计</b>	<b>26</b>

4.1	爬虫模块设计.....	26
4.2	后端功能模块设计.....	27
4.2.1	数据库操作模块设计.....	27
4.2.2	路由转发模块设计 .....	29
4.3	前端功能模块设计.....	29
4.3.1	登录模块组件设计 .....	29
4.3.2	文章列表与内容模块组件设计 .....	30
4.3.3	点赞模块组件设计 .....	32
4.3.4	评论模块组件设计 .....	32
4.3.5	回复模块组件设计 .....	33
4.3.6	原创文章编辑器模块组件设计 .....	35
4.3.7	消息通知模块组件设计.....	35
<b>第五章</b>	<b>系统测试 .....</b>	<b>37</b>
5.1	测试目的.....	37
5.2	测试方法.....	37
5.2.1	单元测试 .....	37
5.2.2	黑盒测试 .....	38
5.3	测试样例.....	38
5.3.1	单元测试 .....	38
5.3.2	黑盒测试 .....	39
<b>第六章</b>	<b>总结与展望 .....</b>	<b>41</b>
6.1	总结.....	41
6.2	展望.....	41
参考文献.....		<b>43</b>
致谢 .....		<b>45</b>

## 第一章 绪论概述/引言

- (\* 正文不少于 10 千字(学校要求 5 千字以上,数据科学与计
- 算机学院要求 10 千字以上);或使用小四字体、1.5 倍行距、
- A4 纸版式排版时不少于 10 页纸。正文须有页码,从第 1 页
- 开始编页码。正文采用章、节、小节组织。章的标题使用“第
- 一章”等字样开头,节的标题采用“1.1”等字样开头,表示
- 第一章的第一节,小节的标题采用“1.1.1”等字样开头,表
- 示第一章的第 1.1 小节。正文章、节、小节标题与正文段落
- 使用不同的字体,并且之间有适当的间距。正文段落要统一
- 缩进两个汉字。
- 行文时注意语句通顺,条理清晰;每章节开头部分需要有承
- 上启下描述,先简要介绍本章节内容,再展开详细描述。
- 第一章作为概述,也是完整的短文,体现全文的内容。

—\*)

### 1.1XXX 问题的背景和意义

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

带格式的: 标题 1, 缩进: 首行缩进: 0 字符, 行距:  
多倍行距 3 字行

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

设置了格式: 字体: (默认) + 中文正文 (宋体), (中文)  
+ 中文正文 (宋体), 字体颜色: 红色

带格式的: 标题 1, 左, 行距: 单倍行距



~~(\*) 阐明问题的来源、研究的动机、意义等。\*)~~

设置了格式: 字体颜色: 自动设置

带格式的: 标题 1, 缩进: 首行缩进: 0 字符, 行距: 单倍行距

#### 4-21. 1X背景 XX问题的描述

~~(\*) 用简明语言描述所研究的问题, 说清楚要解决什么问题、难点和挑战性。\*)~~

互联网和计算机行业发展迅猛, 新技术层出不穷, 作为计算机行业从业者, 为了不被行业淘汰, 我们需要不断学习各种新知识, 新技术, 在学习过程中我们会收藏别人写的优秀文章, 也会撰写自己的原创技术博客, 因此, 许多人都有自己的博客以记录和展示自己的学习成果。

设置了格式: 字体颜色: 自动设置

通常开设博客有两种形式, 一种是在现有的博客平台注册账号撰写博文, 这样的好处是没有搭建博客平台的时间和技术成本, 同时依赖现有成熟博客平台的大量用户群, 我们撰写的文章也能有较高的访问流量, 被更多人看到, 但是通过这种方式我们就没有办法个性化定制自己博客的页面, 每个人的博客页面都是千篇一律, 不够个性化。另一种开设博客的方式就是由开发者自己设计个人博客并开发建站, 这样我们就可以完全个性化定制我们自己的博客页面, 然而这样的不好处是我们的博客系统是完全独立的, 只能展示自己原创文章, 而不能展示我们在其他博客平台收藏的文章, 使我们不能方便地查看收藏的优秀文章。

设置了格式: 字体颜色: 自动设置

设置了格式: 字体颜色: 自动设置

设置了格式: 字体颜色: 自动设置

设置了格式: 字体颜色: 自动设置

设置了格式: 字体颜色: 自动设置

设置了格式: 字体颜色: 自动设置

设置了格式: 字体颜色: 自动设置

那么能不能开发这样一个博客系统, 让我们在既能个性化定制博客页面的同时, 又能与其他博客平台连接, 展示我们在其他博客平台收藏的文章呢? 要实现这一功能, 一些博客平台会提供 API 接口供开发者使用, 但是经过调查, 我发现我平时最常逛的简书和思否 (SegmentFault) 两大博客平台都没有提供公开的 API。因此, 我们只能通过网络爬虫的方式, 爬取我在简书和思否的收藏夹以及收藏夹内文章的内容, 然后展示在我们自己开发的博客网站上。

因此, 我决定实现的就是这样一个个人博客系统: 该系统前端页面由我自行设计, 支持普通个人博客系统的文章撰写、发布、修改、删除功能, 也支持博客访问者点赞和评论文章。同时还通过网络爬虫的方式定时爬取我在思否和简书这两大博文平台上收藏的文章, 当我在这两个网站收藏了新文章时, 该博客系统也能更新显示我添加的新收藏。这样就整合了我的学习资源, 方便我回顾学习过的内容。

## 1.4.1.2 论文结构简介

~~(简单介绍论文后面章节的安排和主要内容。\*)~~本论文共分为七章，首先介绍了项目相关的应用背景和技术背景，然后论文就系统的业务需求以及整体架构设计进行了分析，紧接着介绍了具体功能模块的设计与实现。最后介绍了项目测试、部署和运行效果，并对项目进行了总结和展望。

论文具体安排如下：

第一章：介绍了个人博客项目的应用背景，以及实现该系统的意义。

第二章：介绍了项目应用到的数据库、后端技术栈、前端技术栈以及其他相关的技术背景，同时介绍了这些技术的发展现状以及本项目选用这些技术的原因。

第三章：分析了系统功能需求，详细介绍了该个人博客系统的整体架构设计，包括数据库设计，后端模块与接口设计，前端路由和数据状态设计。

第四章：详细介绍了各个功能模块的具体设计和实现。

第五章：介绍了该个人博客系统的系统测试样例和运行结果。

第六章：最后论文总结了毕业设计所做的工作，并提出目前项目存在的不足之处，展望改进方向。

带格式的：正文文本缩进 2，缩进：首行缩进： 2 字符，行距：1.5 倍行距

## 第二章 ~~X~~系统相关技术概述 ~~XX~~ 综述

~~特定含义的名词术语或新名词、以及使用外文缩写代替某一名词术语时，首次出现时应在括号内注明其含义，如：OECD（Organisation for Economic Co-operation and Development）代替经济合作发展组织。~~

~~（\* 正文中有图、表和公式时必须对图、表和公式按章进行编号。例如，下图编号“图 2-1”表示第二章第 1 个图；或者，在整篇正文中，图和表按出现的顺序依次编号，如“图 1”、“图 2”、“表 1”、“表 2”等。~~

设置了格式：字体颜色：自动设置

PostgreSQL 使用的是客户端/服务器的架构模式。一次会话由一个客户端进程和一个服务器进程构成。客户端进程是指需要执行数据库操作的用户应用，它的形式可能是一个字符界面工具，一个图形界面应用，也可能是一个 web 服务器，或者是一个数据库管理工具。服务器进程称为 postmaster，它管理数据库文件，接受并响应客户端应用与数据库的连接。客户端进程与服务端进程可能在不同主机上，它们通过 TCP/IP 进行通讯。postgresql 服务器可以通过启动多个进程的方式实现并发处理多个客户端请求户端请求。[1]

PostgreSQL 具有优越的可编程性，可以使用类似于 Oracal PL/SQL 的 PL/pgSQL 语言进行存储过程、触发器的开发，还可以使用 PL/python、PL/Ruby 等内置脚本语言进行开发，甚至可以使用 C、C++或者 Java 进行开发。[2]

PostgreSQL 支持复杂的 SQL 功能，包括复杂 SQL 查询、子选择、外键、触发器、视图、事务、多进程并发控制、流式控制等，同时还支持增加新的数据类型、函数、操作符、聚集函数、索引方法、过程语言等特性。

与业界非常流行的 MySQL 数据库相比，PostgreSQL 虽然处理数据速度更慢，但是对数据完整性和严肃性要求较高，逻辑封装性较好，因此更适合数据量大且无法预测的场景。[3]

### 2.42.2 网络爬虫

网络爬虫也叫网络蜘蛛，即 Web Spider，是一种按照一定规则，通过网页链接获

取网页内容的脚本或程序。网络爬虫程序从 Web 页面的链接地址中查找 Web 页面，从一个特定页面（一般是主页）开始，读取页面内容并查找其他链接地址，通过这些链接地址进入下一个页面爬取网页内容，如此往复循环，直到整个网站中的所有页面都被抓取完为止。[4]

一个相对完整的网络爬虫的步骤和注意事项如下：首先，分析待抓取网站的 URL 路径和网页数据请求和返回格式，然后分析如何解析目标字段。当正式进行爬取的时候，需要创建合适的 HTTP 请求，发起的请求需要使用人类浏览网页时常用的 User-Agent 字段，从而迷惑服务器，让服务器以为是人类在浏览网页而不是程序在抓取数据。如果需要高频次大批量地抓取数据，则需要进行 IP 代理，每次轮换不同的 IP 去请求数据，以防 IP 被封。有的网页内容需要用户登录后才能查看，抓取这种类型的网页则需要在请求中添加合适的 Cookie。[5]

网络爬虫的系统框架包括调度器，解析器，资源库三部分。调度器主要负责管理待抓取和已抓取的 URL 集合，通过多线程分配抓取任务，防止重复抓取和循环抓取。解析器的主要工作是下载网页内容到本地，进行页面的处理，准确的说就是提取 HTML 信息，主要是将一些 JS 脚本标签、CSS 代码内容、空格字符、HTML 标签等内容处理掉[6]，提取对我们有价值的内容或者下一步需要爬取的 URL 链接。资源库用于存储提取到的数据，一般都采用大型的数据库存储，如 Oracle 数据库，实现数据持久化。[7]

## 2-52. 3Ajax 技术

Ajax 是 Asynchronous JavaScript and XML 的首字母缩写，翻译过来就是异步 JavaScript 与 XML 技术。其根本理念是按需获取数据，只和服务器交换有用的数据，而不是整个页面。使用 Ajax 与服务器通信时，数据是通过 JavaScript 脚本异步交互的，虽然名字中包含 XML 的成分，但 Ajax 通信与数据格式无关，现在服务端与客户端通信一般采用 JSON 格式的数据。总之，这种技术就是无须刷新页面即可从服务器取得数据，但不一定是 XML 数据。[8]

Ajax 技术的核心是由微软首先引入的 XMLHttpRequest 对象（简称 XHR），现在几乎所有浏览器提供商都提供了该对象。在 XHR 出现之前，Ajax 式的通信必须借助

隐藏的框架或内嵌框架等 **hack** 手段来实现。**XHR** 为向服务器发送请求和解析服务器响应提供了流畅的接口。能够以异步方式从服务器获得更多信息,意味着用户单击后,可以不必刷新页面也能取得新数据。也就是说,可以使用 **XHR** 对象取得新数据,然后再通过 **DOM** 将新数据插入到页面中。

实际上,在 **Ajax** 这个名字被正式提出之前,这种技术已经存在很长时间了,人们过去通常将这种技术叫做远程脚本 (**remoting scripting**),而且早在 1998 年就有人采用不同的手段实现了这种浏览器与服务端的通信。再往前推,**JavaScript** 需要通过 **java applet** 或 **flash** 电影等中间层向服务器发送请求。而 **XHR** 则将浏览器原生的通信能力提供给了开发人员,简化了实现同样操作的任务。[9]

在被命名为 **Ajax** 后,这种浏览器与服务器的通信技术可谓红极一时。人们对 **JavaScript** 和 **web** 的全新认识,催生了很多使用原有特性的新技术和新模式。就目前来说,熟练使用 **XHR** 对象已经成为所有 **Web** 开发人员必须掌握的一种技能。也正因为 **Ajax** 技术,与客户交互的复杂逻辑才从后端转移到了前端,催生了后来的 **Angular**、**React**、**Vue** 等前端框架。

## 2-62.4Node.js

**JavaScript** 最为广泛的用途是运行在浏览器中,但是事实上,**JavaScript** 在诞生之初,就可以运行在服务端,只是在与 **Java**、**PHP**、**.NET** 等风靡的服务端技术的竞争中,**JavaScript** 逐渐式微,而它在浏览器端的盛行,让人们几乎忘记了 **JavaScript** 是可以在服务端运行的这回事。[10]

但是 **Node.js** 的出现改变了一切。**Node.js** 是一个基于 **Google** 的 **V8** 引擎[11],用于构建快速且可扩展的 **Web** 应用的平台,凭借 **V8** 的高性能和异步 **I/O** 模型,**Node.js** 将 **JavaScript** 推向了一个新高潮,现在,**JavaScript** 不仅可以同时运行在前后端,而且性能十分出众,与其他同类脚本语言相比甚至可以说有过之而不及。

**Node.js** 具有以下几个特点: [12]

- 1) 采用异步 **I/O**: 在 **Node** 中,从文件读取到网络请求在内的许多底层 **I/O** 的 **API** 都是以异步的方式进行调用的。其意义在于,每个调用之间无须等待之前的 **I/O** 调用结束,在编程模型上可以极大地提高效率。

- 2) 基于事件和回调函数: Node 将前端中应用广泛且成熟的事件与回调函数机制引入后端, 配合异步 I/O, 将事件点暴露给业务逻辑, 具有轻量级, 松耦合的优势。
- 3) 单线程: Node 保持了浏览器端 JavaScript 单线程的特点, 不用像多线程编程那样在意状态同步、死锁等问题, 也没有线程上下文切换所带来的性能开销。
- 4) 跨平台: Node 在架构层面, 在操作系统与 Node 上层模块之间构建了一个基于 libuv 的中间层, 通过良好的架构, Node 的第三方 C++ 模块也可以借助 libuv 实现跨平台。

总之, Node.js 将前端中广泛运用的异步编程和事件驱动的思想迁移到了服务端, 采用单线程, 异步 I/O, 搭配事件循环, 使得 Node.js 非常适合处理大量的并发 I/O 请求。而本个人博客系统在后端的业务逻辑, 主要就是对数据库的增删查改, 是一个 I/O 密集型应用, 因此非常适合采用 Node.js 作为后端开发语言。

## 2.72. 5 RESTful API

REST 的全称是 Representational State Transfer, 中文含义即表现层状态转化。MVC 模式大行其道许多年, 一直到 RESTful 的流行, 大家才意识到 URL 也可以设计得很规范, 请求方法也能作为逻辑分发的单元。符合 REST 规范的设计, 我们称为 RESTful 设计。他的设计哲学主要将服务器端提供的内容实体看作一个资源, 并表现在 URL 上, 对资源的操作体现在请求方法上, 例如, 一个用户资源对应的 url 可以被设计为 /user/jack, 而查看, 修改, 删除, 添加该资源则分别是通过以 GET、PUT、DELETE、POST 方法请求该 url 来实现的, 对于这个资源的表现形态, 不再像以前一样表现在 url 的文件后缀上, 而是在请求报头的 Accept 字段和服务端的支持情况来决定的。所以 REST 的设计就是, 通过 URL 设计资源、请求方法定义资源的操作、通过 Accept 决定资源的表现形式。

相比传统的 RPC (Remote Procedure Call) 模式[13]的 Web 服务, REST 风格提出了相互协作的架构约束, 其限制了架构元素的功能和角色, 以及在任何一个遵循该风格的架构中允许存在的元素之间的关系, 这些约束给 REST 架构带来了一些架构属性,

而这些架构属性恰恰是 REST 较 RPC 模式更有优势的体现。[14]

## 2.82.6 React.js

React.js 是 Facebook 开发的一个开源渐进式前端框架。基于 HTML 的前端界面开发正变得越来越复杂，其本质问题基本归结于如何将用户输入的或者从服务器获取的数据与页面元素进行高效地实时映射。而 React 正是 Facebook 团队针对此问题的一个解决方案，正如其官网所述，React 用于开发数据不断变化的大型前端应用（Building large applications with data that changes over time），当页面数据发生改变时，React 可以高效地对页面进行重新渲染。相比传统型的前端开发，React 具有组件化的特点，前端页面以组件的形式被构建起来，每个组件在内部维护自己的状态。组件式开发使前端项目可以不再使用模板代码，方便代码复用，同时也使数据管理和传递变得更加规范和方便。[15]

React 是十分高效的，这得益于它采用了虚拟 DOM 技术[16]。在 Web 开发中，要改变页面 UI（User Interface），我们就需要对 DOM 进行操作，这会触发浏览器对 HTML 的重绘动作，频繁的重绘是造成前端性能瓶颈的主要原因。虚拟 DOM 技术就是将整个 HTML 文档的结构和内容以 JavaScript 对象的形式在内存中存储起来，形成一个虚拟的 DOM 树，当页面结构发生变化时，React 不是直接更新页面的 DOM 元素，而是更新内存中的虚拟 DOM，然后比对前后两次虚拟 DOM 的区别，然后在实际 DOM 结构中仅仅更新变化了的部分。这就使得页面只对部分 HTML 节点进行重新渲染，其余大部分不变的内容无须进行重绘操作，从而大大提高了性能。同时，由于所有的 DOM 操作都是由 React 完成的，因此开发者可以将注意力集中在数据的管理上来，而不必手动更新 DOM。

## 2.92.7 Redux

React 只是 DOM 的一个抽象层，在 MVC 模式中，只是一个 View 层框架，不是 Web 应用的完整解决方案。随着前端工程规模越来越大，开发的重点也越来越转移到数据状态的管理上，单独用 React 已经很难承担大型前端应用的开发。Redux 就

是针对数据状态管理的一个框架，用于辅助 React 开发。它于 2015 年由 Dan Abramov 和 Andrew Clark 共同创建，受函数式编程思想的启发，在 Flux 框架的基础上进行了优化和改造。

Redux 由 actions、reducers 和 store 三部分组成。Store 是管理和存储所有数据状态的对象，reducers 是 JavaScript 纯函数，用于接收 action 的处理逻辑来修改 store 中的数据状态，actions 则用于在用户交互、内部事件或 API 调用之后发送给 reducer 告知要修改数据状态。[17]

要使用 Redux 管理数据，需要遵循以下原则：

- 1) 数据源是单一的：即应用的所有状态都存储在同一棵状态树中，应用中的每个组件都可以访问这个数据源。
- 2) 数据是只读的：即不可以直接修改数据对象本身，而是用先深拷贝再修改的方式返回最新的数据。
- 3) 必须用纯函数来执行数据的变更：这与数据是只读的这条规则相似，即执行数据变更的函数必须是纯函数，不能直接变更数据。

如果前端应用数据状态并不复杂，那就没有必要使用 Redux，但是如果前端项目规模较大，数据状态繁杂，那么就很有必要使用 Redux 管理数据状态，从组件角度看，如果 Web 应用有以下场景，可以考虑使用 Redux。

- 1) 某个组件的状态需要共享。
- 2) 某个状态需要在任何地方都可以拿到。
- 3) 一个组件需要改变全局状态。
- 4) 一个组件需要改变另一个组件的状态。

发生上面情况时，如果不使用 Redux 或者其他状态管理工具，不按照一定规律处理状态的读写，代码很快就会变成一团乱麻。Redux 提供了一种机制，可以在同一个地方查询状态、改变状态、传播状态的变化。



## 第三章 系统分析和设计

### 3.1 系统可行性分析

可行性分析，又称可行性研究，简单地概括，就是指在项目开发前对项目进行的考察和鉴定。信息系统开发是一项耗资大，耗时长，且具有一定风险的工程项目，因此有必要对信息系统开发的有益性、可能性和必要性进行充分的论证

它的目的不是解决问题，而是确定问题是否值得去解决。[18]

可行性分析从以下四个方面来考虑：[19]

- 1) 技术可行性：衡量开发新系统是否具备必要的硬件、软件技术以及从事这些技术开发工作的人员数量及水平是否达到要求。
  - 2) 经济可行性：验证信息系统开发所需的投资是否可以承受，开发出系统所带来的经济效益是否足够。
  - 3) 操作可行性：也称运行可行性，评价系统开发完成后是否便于使用者使用以及是否能引起各方面的变化(如组织机构、管理方式、工作环境等)。
  - 4) 法律可行性：验证系统开发的过程和使用该系统时是否会违反相关法律。
- 本个人博客系统的可行性分析如下：

#### 3.1.1 技术可行性

本个人博客项目是典型的基于 B/S 架构的 Web 应用，后端业务逻辑主要是对数据库中的数据进行增删查改，实现相对简单。同时前端展示的是博文列表和博文内容，涉及到的用户交互主要是登录，注册，点赞，博主撰写文章，评论和回复的提交，也不是很复杂。再加上已经有较成熟的框架，因此开发这个系统的技术难题不多，具备技术可行性。

#### 3.1.2 经济可行性

本项目是一个 Web 应用，部署好之后只需一个浏览器就可以访问，后期如果给网站申请域名也仅需很小的经济成本，可以忽略不计，同时该系统是个人独立

开发的系统，后期维护也不用投入成本，因此经济上基本上零成本的。

### 3.1.3 操作可行性

本系统参照其他博客系统，对于博客访问者来说，涉及到的用户交互都很简单，没有复杂操作，作为设计者与开发者，我也没有加入让用户难以理解和操作的交互方式。

### 3.1.4 法律可行性

本系统不会违反国家法律，爬取和展示的文章是我本人在简书和思否两大网站收藏的免费公开文章，同时展示时也注明了原文作者和出处，不涉及侵权问题

综上所述，该系统在技术上、经济上、可操作性上、法律上都是可行的，而且要求不高，所以该系统的开发是可行的。

## 3.2 需求分析

### 3.2.1 功能需求分析

本系统涉及的用户角色有两个，即访客和博主。

对于访客，又分为已注册访客和未注册访客两种，未注册访客仅可以查看博主收藏和原创文章。对于已注册访客，应实现以下功能：

- 1) 能查看博主收藏和原创文章。
- 2) 能点赞、评论博主的原创文章。
- 3) 对于其他人在博主原创文章下的评论，可以进行回复讨论。
- 4) 可以查看博主原创文章下与自己相关的回复(包括其他人对自己的回复，以及自己的评论下的所有回复)。对于自己还没有查看过的相关回复，系统有未读提示。

对于博主，即本系统管理者，除了能使用访客的所有功能外，还应实现如下功能：

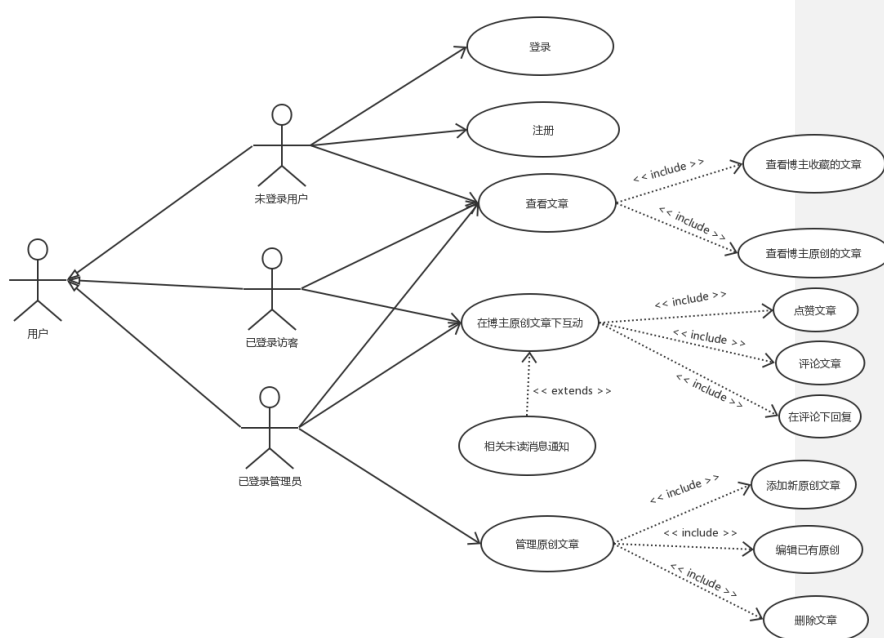
- 1) 在思否、简书两大博客平台收藏了优秀文章后，系统会在某个时间对这两个网站中博主的收藏夹页面进行爬虫，获取博主新收藏的文章后更新

本系统中博主的收藏夹数据库。

- 2) 能用 markdown 语法撰写并发布原创文章，同时对于已发布的文章，可以修改和删除。
- 3) 能查看访客对博主原创文章下的评论，点赞，回复，对于还未查看过的相关消息，系统有未读提示。

### 3.2.2 用例分析

图 3-1：系统用例图



### 3.3 系统概要设计

要实现该系统，我们需要从三个方面进行设计，即数据库设计，后端设计和前端设计。

本系统使用的数据库是 PostgreSQL，是关系型数据库，需要存储用户信息、博主收藏的文章、原创的文章，原创文章下的点赞、评论、回复，以及与用户相

关的通知。与后端通过 Node.js 与 PostgreSQL 数据库的程序连接接口进行连接。

后端是实现一个响应 RESTful API 的 Web 服务，关键主要是实现三大模块，一是爬虫模块，实现每隔 24 小时爬取一次我在简书和思否两大博客平台收藏的文章，然后将新收藏的文章存储到数据库中；二是数据库操作模块，通过 Node.js 与 PostgreSQL 数据库的程序连接接口，执行不同的 SQL 语句，对数据库里的数据进行查询、删除、新增和修改。三是路由模块，解析前端通过 AJAX 发送的 HTTP 请求报文，根据不同的路由请求，运行数据库操作模块对应的处理程序，然后将数据库操作返回的数据通过 HTTP 响应报文返回给前端处理。

前端需要实现的是一个单页应用，用 React-router 管理前端路由。页面分为页头，页面主体和页脚三部分，页头展示导航栏和登录/注册组件，页脚展示博主的联系方式等简要信息，页面主体根据前端路由，分别展示以下内容：

- 1) 个人主页：显示博主的简历信息。
- 2) 注册页：供用户填写注册信息，注册为系统用户。
- 3) 收藏夹文章列表页：展示博主收藏的文章列表，每个列表项包括文章的标题和发布日期，列表内容由收藏夹标签<sup>1</sup>和文章标签<sup>2</sup>决定。
- 4) 收藏夹文章内容页：展示博主收藏的文章的内容，包含文章标题、作者、内容、发布日期。
- 5) 博主原创文章列表页：展示博主原创文章列表，每个列表项包括文章的标题、发布日期、点赞数、评论数。
- 6) 博主原创文章内容页：展示博主原创文章的内容，包含文章标题、作者、内容、点赞数、回复列表、每条回复下的评论列表。同时包含点赞按钮，评论框和回复框供用户交互。

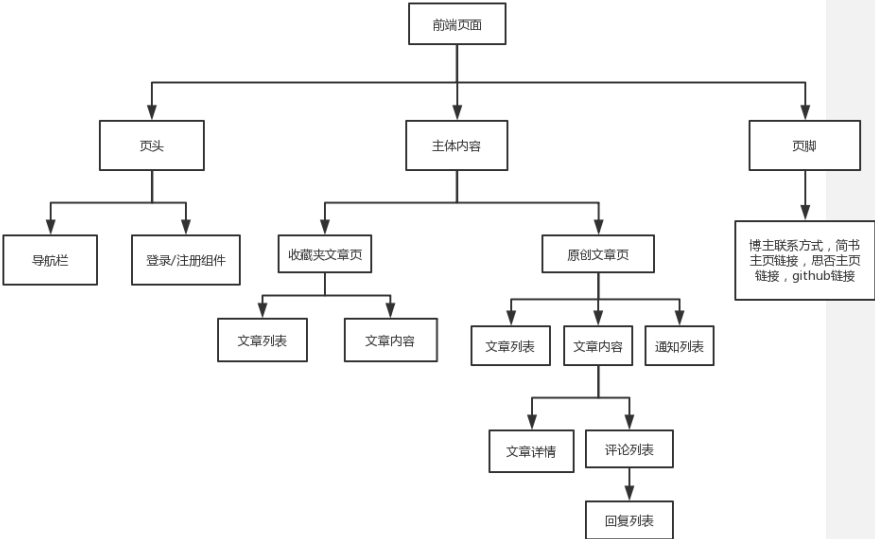
前端页面以 React 组件的形式进行组织，通过 Ajax 技术向后端动态请求数据，对于多组件公用的数据，通过 Redux 进行状态管理。

---

<sup>1</sup> 收藏夹标签：前端、服务端、数据库、其他

<sup>2</sup> 文章标签：前端收藏夹下的文章标签有 JavaScript、Node、React、Vue、Webpack、CSS；后端收藏夹的文章标签有 java、cpp、python、linux；数据库收藏夹下的文章标签有 mysql、postgresql、redis、mongodb；其他收藏夹下的文章标签有：算法、数据结构、人工智能

图 3-2：前端页面结构图

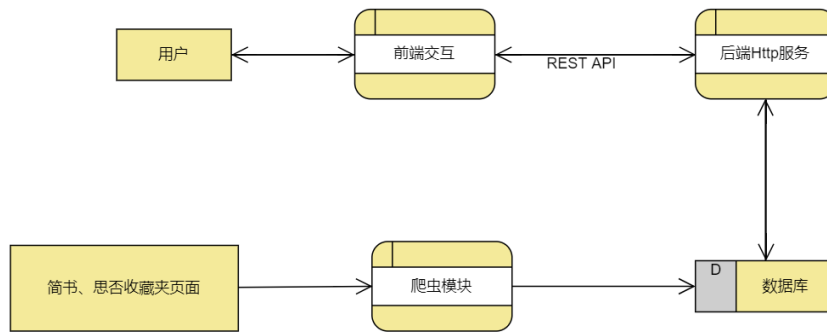


系统数据流图如下所示：

表 3-1 数据流图图例

图例	含义
<div>Entity</div>	实体
<div>Process</div>	数据加工
<div>D DataStore</div>	数据存储
<div>→</div>	数据流

图 3-3：系统数据流图

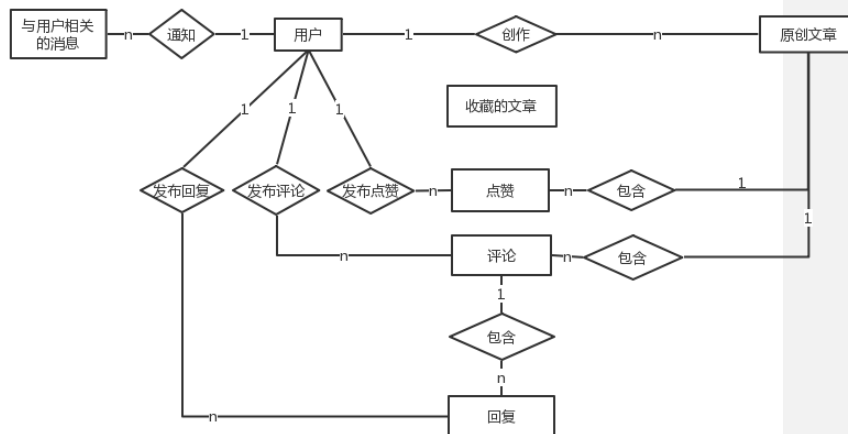


### 3.4 系统数据库设计

本系统数据库采用的是关系型数据库 PostgreSQL，在设计数据库时，通过之前的需求分析，确定了该系统数据库主要包括以下几个实体：用户、收藏的文章、原创文章、评论、回复、点赞、与用户相关的消息。

这些实体之间的关系用 E-R 图表示如下所示：

图 3-3 个人博客数据库 E-R 图



收藏的文章是通过爬虫模块添加到数据库中的，与其他实体之间没有联系。当用户是博主的时候，一个用户可以写多篇原创文章，一篇原创文章的作者只能是一个用户，因此用户与原创文章之间存在一对多（1：n）的联系；一个用户可以发出多个点赞，一个点赞只能属于一个用户，因此用户和点赞之间存在一对多（1：n）的联系；一个用户可以发布多个评论，一个评论的作者只可能是一个用户，因此用户和评论之间存在一对多（1：n）的联系；一个用户可以发布多条回复，一条回复的作者只能是一个用户，因此用户和回复存在一对多（1：n）的联系；一个用户可以收到多条与他相关的消息，一条消息只能发给一个用户，因此用户和消息之间存在一对多（1：n）的联系；一篇原创文章中可以包含多个点赞，而一个点赞只属于一篇原创文章，因此原创文章和点赞存在一对多（1：n）的联系；一篇原创文章可以包含多条评论，一条只能发布在篇原创文章下；因此原创文章和评论之间存在一对多（1：n）的联系；一条评论下可以有多条回复，一条回复只能发布在一条评论下，因此评论和回复存在一对多（1：n）的联系。在创建数据表时，我们需要先创建两个枚举类型，一个是文章种类 ARTICLE\_TAG，它标记了收藏的文章是属于什么类型的文章，该类型的取值有：'JS', 'NODE', 'REACT', 'VUE', 'WEBPACK', 'CSS', 'JAVA', 'CPP', 'PYTHON', 'LINUX', 'MYSQL', 'POSTGRESQL', 'REDIS', 'MONGODB', 'ALGORITHM', 'DATASTRUCTURE', 'AI'。需要创建的另一个枚举类型是用户角色 USER\_ROLE，它标记了用户是博主还是访客，该类型的取值有：'ADMIN', 'VISITOR'。

经过数据库需求分析和设计后，我们设计出各个实体的数据表如下：

表 3-2：用户（users）数据表设计

字段名	数据类型	是否可以 空	默认值	说明
uid	SERIAL	NO	序列递增值	用户 id，主键
nickname	VARCHAR(20)	NO		用户名
email	VARCHAR(20)	NO		邮箱
password	VARCHAR(20)	NO		密码
role	USER_ROLE	NO	'VISITOR'	用户角色

表 3-3: 收藏的文章 (collected\_articles) 数据表设计

字段名	数据类型	是否可以 空	默认值	说明
caid	SERIAL	NO	序列递增值	收藏的文章 id, 主键
tag	ARTICLE_TAG	YES		文章类别
title	VARCHAR(20)	YES		文章标题
author	VARCHAR(20)	YES		文章作者
content	TEXT	YES		文章内容
original_link	VARCHAR(50)	YES		原文出处链 接
date	DATE	YES	CURRENT_DATE	收藏的日期

表 3-4: 原创文章 (original\_articles) 数据表设计

字段名	数据类型	是否可以 空	默认值	说明
oaid	SERIAL	NO	序列递增值	原创文章 id, 主键
title	VARCHAR(20)	YES		文章标题
author	INTEGER	YES		文章作者 id, 外键
content	TEXT	YES		文章内容
date	DATE	YES	CURRENT_DATE	发布的日期



表 3-5: 评论(comments)数据表设计

字段名	数据类型	是否可以 为空	默认值	说明
cid	SERIAL	NO	序列递增值	评论 id, 主键
oaid	VARCHAR(20)	NO		原创文章 id, 外键
owner	INTEGER	NO		评论作者 id, 外键
content	TEXT	YES		文章内容
date_time	TIMESTAMP	YES	CURRENT_TIMESTAMP	评论发布 的时间戳

表 3-6 回复 (replies) 数据表设计

字段名	数据类型	是否可以 为空	默认值	说明
rid	SERIAL	NO	序列递增值	回复 id, 主键
cid	INTEGER	NO		评论 id, 外键
owner	INTEGER	NO		回复作者 id, 外键
responder	INTEGER	NO		被回复者 id, 外键
content	TEXT	YES		回复内容
date_time	TIMESTAMP	YES	CURRENT_TIMESTAMP	回复发布 的时间戳

表 3-7 点赞 (likes) 数据表设计

字段名	数据类型	是否可以 为空	默认值	说明
lid	SERIAL	NO	序列递增值	点赞 id, 主键
oaid	INTEGER	NO		原创文章 id, 外键
owner	INTEGER	NO		回复作者 id, 外键
date_time	TIMESTAMP	YES	CURRENT_TIMESTAMP	回复发布 的时间戳

表 3-8: 通知消息(messages)数据表设计

字段名	数据类型	是否可以 为空	默认值	说明
mid	SERIAL	NO	序列递增值	消息 id, 主键
uid	INTEGER	NO		要通知的 用户 id, 外键
oaid	INTEGER	NO		消息所属 的原创文 章 id, 外 键
title	VARCHAR(20)	YES		消息标题
content	TEXT	YES		消息详情
unread	BOOLEAN	NO	TRUE	是否已读
date_time	TIMESTAMP	YES	CURRENT_TIMESTAMP	消息的时 间戳

### 3.5 后端 API 设计

因为前端 MVVM 框架的出现，包括前端页面路由，页面渲染在内的大量的工作可以交给前端完成。在本系统中，后端服务只负责响应 RESTful API，前端和后端通过 AJAX 进行数据交互，因此 API 的设计是非常重要的。

本系统的 API，传递的数据统一为 JSON[21]格式，因此前端发送的 HTTP 请求需要添加 Content-Type: application/json 字段。后端返回的数据 JSON 格式为：

```
{
    succeed: true,
    errorCode: -1,
    message: null,
    data: {
        // 响应得到的数据，各个接口各不相同
    }
}
```

API 设计满足 REST 规范，即 url 代表资源，请求方法代表对资源的操作，其中资源种类包括 collected\_articles、original\_articles、users 三种，原创文章下评论、点赞、回复资源包含在 original\_articles 下，与用户相关的消息包含在 users 下。具体 API 设计如下：

表 3-9：后端 RESTful API 设计

请求方法	请求 URL	查询参数	请求内容所需字段	说明
GET	/collected_articles/{tag}	start,num		获取收藏的文章列表
GET	/collected_articles/a/{caid}			获取收藏的文章内容

续表 3-9

GET	/original_articles	start,num		获取原创文章列表
POST			title, author, content	发布原创文章
GET	/original_articles/{oaid}			获取原创文章内容
PUT			title, content	修改原创文章内容
DELETE				删除原创文章
GET	/original_articles/{oaid}/comments	start,num		获取评论列表
POST			owner, content	发布评论
GET	/original_articles/comments/{cid}/replies	start,num		获取回复列表
POST			responder, owner, content	发表回复
GET	/original_articles/{oaid}/like			获取点赞数
POST			owner	给文章点赞
DELETE			owner	给文章取消点赞
POST	/user/login		user, password	登录接口
POST	/user/register		nickname, password, email	注册接口

续表 3-9

GET	/users/{uid}/messages	start,num		获取用户 相关消息
POST			uid, oaid, title, content,	添加新消 息
PUT	/users/messages/:mid			修改消息 已读状态
GET	/users/{uid}/like/{oaid}			获取用户 是否给文 章点过赞

### 3.6 前端路由设计

根据不同 URL 地址来显示不同的内容或页面称之为路由。过去，这一任务交由后端来实现，每次 URL 变化，后端都要发送一次完整的 HTML 给前端进行展示。有了 AJAX 之后，前后端数据交互可以不再刷新页面，路由的映射通常是进行一些 DOM 的显示和隐藏操作，当访问不同路径的时候，会显示不同的页面组件。这样一来，服务端只用给前端发送一次 HTML 文件，这样的应用我们称之为 SPA(Single Page Application)。

前端路由有两种实现方式，一种是 Hash 路由。URL 中 ‘#’ 符号及其后面的部分为 Hash，Hash 仅仅是客户端的一个状态，也就是说，当向服务器发请求时，Hash 部分并不会发过去。通过监听 window 对象的 hashChange 事件，就可以实现简单的 Hash 路由。另一种实现前端路由的方式是利用 HTML5 的 History API，利用它，我们可以在不刷新页面的情况下，直接改变当前 URL。[20]

在用 React 搭建前端页面时，我们可以用官方维护的 React-router 库实现前端路由，本系统中，我们使用的是基于 History API 的路由。

具体设计如下：

表 3-10: 前端路由设计

URL	说明
/	首页
/register	注册页
/front-end/{tag <sup>3</sup> }	前端收藏夹文章列表页
/front-end/{tag}/{caid}	前端收藏夹文章内容页
/server/{tag}	服务端收藏夹文章列表页
/server/{tag}/{caid}	服务端收藏夹文章内容页
/database/{tag}	数据库收藏夹文章列表页
/database/{tag}/{caid}	数据库收藏夹文章内容页
/original/all-articles	原创文章列表页
/original/all-articles/{oaid}	原创文章内容页
/original/write-article	博主编辑原创文章页

### 3.7 前端 Redux store 设计

React 组件之间的数据传递是单向数据流, 即数据只能由上层组件通过 props 或 context 属性传递给下层组件。因此要实现两个组件之间的通信, 只能通过把公共状态放在这两个组件的同一个祖先组件上实现。结合 React 的这一特点, React-redux 库把各个组件之间需要通信的数据放在一个顶层组件(Provider 组件)的 context 中, 而对这些数据的操作统一由 Redux 来进行管理。

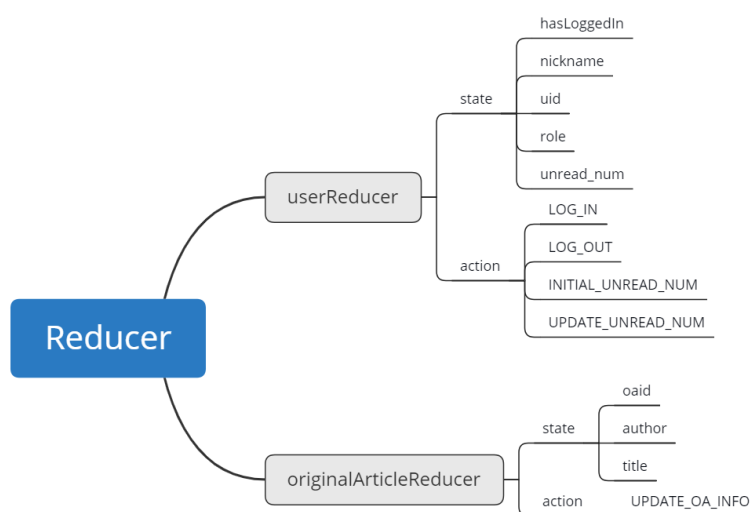
因此在设计我们的前端工程时, 对 Redux 状态的设计是至关重要的。设计 Redux 状态时, 主要是设计两项, 一是存储的数据内容, 即 state, 另一个是对数据可以进行哪些操作, 即 action。

结合前文的需求分析、前端页面架构设计和 API 设计, 我们可以分析出: 首先, 用户登录后显示用户信息, 点赞、发表评论、回复时给后台发送请求以及根据用户是否登录决定是否显示撰写、编辑、删除原创文章的相关组件, 实现这三项功能的相关组件需要共享用户 id, 用户名, 用户角色, 用户是否登录这些状态。其次, 点赞、发表评论、回复时需要向后台发送添加新通知的请求, 请求内容中

<sup>3</sup> 这里的 tag 取值即数据库设计中提及的文章类别标签

包含文章标题、文章 id，文章作者 id，因此点赞、评论、回复组件需要与原创文章组件共享原创文章的这些状态。由此，我们把这些状态放在 `redux` 的 `store` 中进行管理，设计出如下 `reducer`<sup>4</sup>：

图 3-4：前端 `redux store` 设计



如图所示，`userReducer` 中包含的状态从上到下依次是：用户的登录状态、用户名、用户 id、用户角色、用户的未读消息数；对这些状态的操作有：`LOG_IN`(将 `hasLoggedIn` 更改为 `true`，同时将 `nickname`、`uid`、`role` 更改为传入的值，用于登录)、`LOG_OUT`(将 `hasLoggedIn` 更改为 `false`，用于退出登录)、`INITIAL_UNREAD_NUM`(将 `unread_num` 更改为传入的值，用于在页面初次挂载时初始化用户未读消息数)、`UPDATE_UNREAD_NUM`(将 `unread_num` 减一，用于在用户点击了未读消息后更新未读消息数)。`OriginalArticleReducer` 包含的状态从上到下依次是：原创文章 id、原创文章作者 id、原创文章标题；对这些状态的操作有：`UPDATE_OA_INFO`（将这三个状态都修改为传入的新值）。

<sup>4</sup> `reducer` 是 `redux` 库的一个函数，用于定义数据和对数据的操作，具体用法参考 `redux` 相关资料

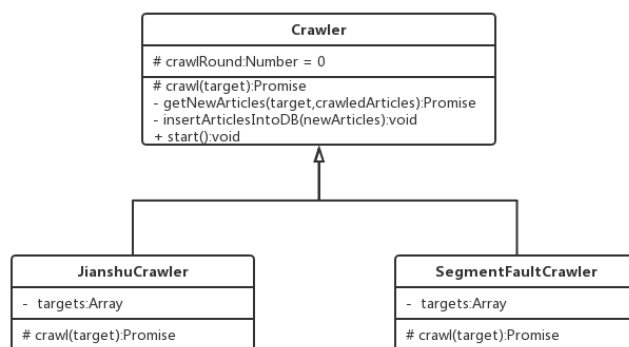
## 第四章 详细设计

### 4.1 爬虫模块设计

爬虫模块需要分别爬取我的简书收藏夹和思否收藏夹，然后将爬取到的文章存入数据库中。首先我们需要做的准备工作是，分别在简书和思否两个网站上，都创建如下的收藏夹：JS、NODE、REACT、VUE、WEBPACK、CSS、JAVA、CPP、PYTHON、LINUX、MYSQL、POSTGRESQL、REDIS、MONGODB、ALGORITHM、DATASTRUCTURE、AI。爬虫模块下分三个子模块，分别是简书爬虫模块，思否爬虫模块和日志模块，前两个模块采用 `request` 库发起 HTTP 请求获取页面 HTML，然后用 `cheerio` 库，用类 `jQuery` 的 API 非常方便地提取页面信息。在爬虫过程中，如果出现错误，日志模块会记录错误信息到日志文件中，有新文章入库成功也会有对应的日志记录，方便日后查看。

简书爬虫模块和思否爬虫模块只有爬取具体页面的时候有所不同，其他流程有重复，因此我采用了面向对象编程，编写了 `Crawler` 类，然后 `JianshuCrawler` 和 `SegmentFaultCrawler` 继承 `Crawler` 类并实现相关接口。

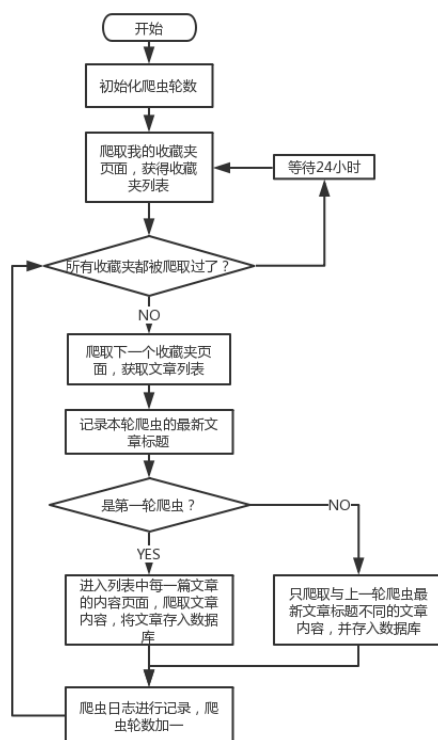
图 4-1：爬虫模块 UML 图



爬虫程序执行流程如下：



图 4-2: 爬虫模块流程图



这样，该模块就实现了每隔 24 小时爬取一次我的简书和思否中的所有收藏夹，并将新收藏的文章存入数据库的功能

## 4.2 后端功能模块设计

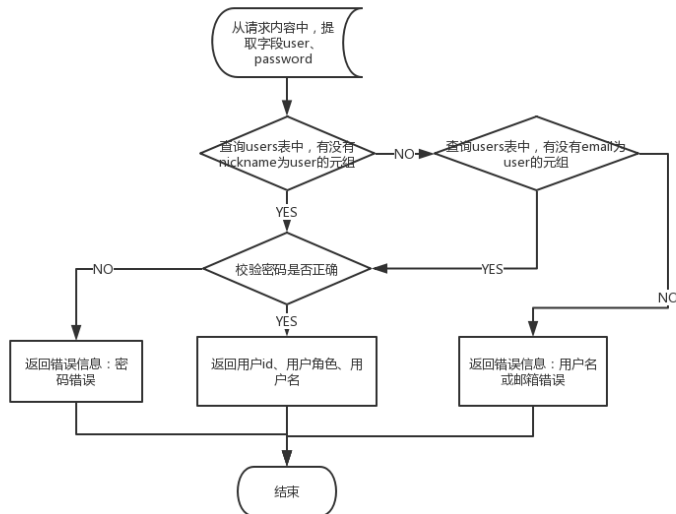
后端需要完成的是数据库操作和路由转发两大功能模块。

### 4.2.1 数据库操作模块设计

本系统后端使用开源的 pg-promise 库连接 PostgreSQL，要返回正确的数据，并不需要我们处理太多的逻辑，大多都是直接执行 SQL 语句然后返回数据即可。只有登录和注册功能需要简单的校验处理。

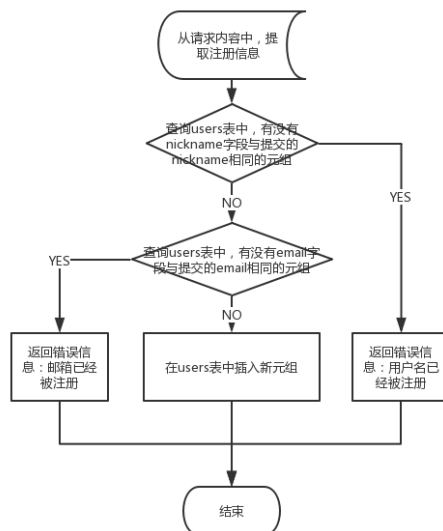
登录校验处理流程：

图 4-3: 登录模块流程图



注册校验处理流程:

图 4-4: 注册校验功能

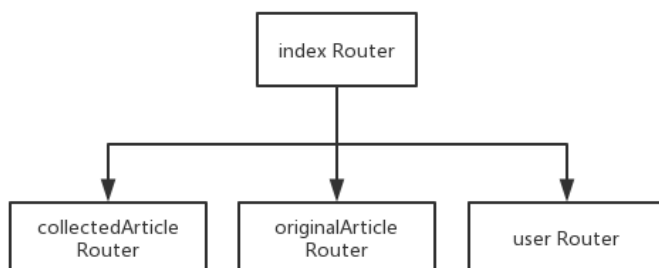


我将一系列对数据库的操作写成类中的静态方法，即完成了后端数据库操作模块，供路由转发模块调用。

#### 4.2.2 路由转发模块设计

后端路由转发就是根据 HTTP 请求报文的 URL，调用相应的处理模块。我们的后端是用 Node.js 搭建的 HTTP 服务，Node.js 内置了 http 模块，可以对 http 报文进行收和发。但是 http 模块相对底层，因此我采用了 Node.js 中最流行的 express 错误!未找到引用源。 框架，利用 express 的 router 模块进行路由转发。根据 API 设计，路由转发主要转发到三个模块，即 collected\_articles，original\_articles，users，因此我写了三个 router，根据设计好的 API 和数据库操作模块，将 URL 与对应的数据库操作一一映射即可。

图 4-5：路由转发模块划分图



### 4.3 前端功能模块设计

#### 4.3.1 登录模块组件设计

登录模块总组件下分两个组件，一个是登录信息输入框组件，一个是登录成功后，展示用户信息的组件，显示哪个组件由 store 中的 state.user.hasLoggedIn 决定，登录模块总组件挂载时，会从 localStorage 中查看是否有用户信息，如果有，则直接更新登录状态为已登录，并显示用户信息，同时向后端发起请求获取最新的未读消息数。用户未登录时，显示登录信息输入框组件。用户在登录信息提示框中输入用户名和密码，点击登录按钮后，向后端发送登录请求，如果登录

失败，则显示错误信息，如果登录成功，则将用户信息存入 localStorage 中，方便用户下一次访问时实现自动登录，同时将 store 中的用户信息更新为后端返回的用户信息，并更新 store 中的用户登录状态为已登录，这时就会隐藏登录信息输入框组件，显示用户信息组件，同时向后端发起请求获取用户的最新未读消息数，并更新到 store 中。

图 4-6：登录框组件



图 4-7：用户信息组件



#### 4.3.2 文章列表与内容模块组件设计

文章列表组件在挂载阶段向后台发送请求获取起始序号为 1，数量为 10 的一组文章标题列表展示出来，当点击第  $n$  页的时候（点击下一页或上一页时，组件内部计算好是第几页），向后台发送请求获取起始序号为  $n*10+1$ ，数量为 10 的一组文章标题列表，更新展示出来。当点击文章标题时，跳转到文章内容页面，如果是收藏的文章，则直接在文章内容组件挂载阶段向后台发起请求获取文章内容并展示，如果是原创文章，则还涉及点赞、评论、回复等组件，将在下文中详细介绍。

图 4-8：文章列表组件



图 4-9：文章内容组件



### 4.3.3 点赞模块组件设计

当用户未登录时（从 `store` 中获取用户是否登录），点赞组件只显示文章点赞数，点击点赞按钮，弹出信息：“登录后才可以点赞”。当用户已经登录时，原创文章内容组件在挂载阶段向后端发起请求，获取用户是否给文章点过赞，如果点过，则点赞按钮为实心，点击该按钮时，向后台发送取消点赞请求，后台返回取消点赞成功的响应后，点赞数减一，点赞按钮变空心。如果用户没有点过赞，则点赞按钮为空心，再点击按钮时，向后台发送点赞请求，后台返回点赞成功的响应后，点赞数加一，点赞按钮变实心。

图 4-10：点赞组件（点赞状态和取消点赞状态）



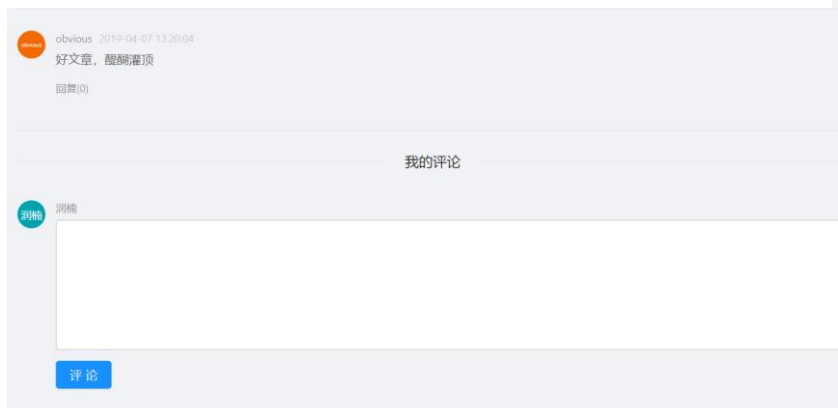
### 4.3.4 评论模块组件设计

评论模块包含评论列表组件和评论输入框组件，二者均挂载在原创文章内容组件下。评论列表同文章列表组件逻辑相似，只是向后台请求数据的是文章下的评论。评论输入框组件显示已登录用户的用户名和一个输入框，监听输入框的 `onChange` 事件，当输入的评论内容超过 200 字时，显示提示信息：评论内容不能超过两百字。当点击发布评论按钮时，向后台发送两个 `POST` 请求，一个是插入新评论的请求，请求报文中所需的原创文章 `id` 从 `store` 的 `originalArticle` 模块获取，评论作者的 `id`，也就是已登录用户的 `id`，从 `store` 的 `user` 模块获取；另一个是给文章作者添加新消息通知的请求，请求内容中的原创文章标题、文章 `id`、作者 `id` 从 `store` 的 `originalArticle` 模块获取，评论者 `id` 从 `store` 的 `user` 模块获取。从后台返回评论成功的响应后，显示提示信息：“评论成功，刷新页面后 visible”。

图 4-11：评论列表组件



图 4-12：评论输入框组件



#### 4.3.5 回复模块组件设计

回复模块包含回复列表组件和回复输入框组件，在评论列表的每个评论项中有一个回复按钮，当用户未登录时，单击该按钮，只显示评论列表，不显示评论输入框。评论列表组件的实现逻辑同文章列表和评论列表组件类似，只是向后台

请求的是评论下的回复，请求回复时发送的报文需要评论的 `id`，这一数据是评论项组件直接通过 `props` 传递给回复列表组件的。当用户已经登录时，单击评论项下的回复按钮，会在评论列表下显示回复输入框组件，同时把评论的作者 `id` 通过 `props` 传给该组件。当用户直接在回复框内输入回复内容并点击发布回复按钮时，向后台发送插入新回复的请求，请求内容中的 `responder` 字段，即被回复者，就是上层组件传入的评论者 `id`。如果在回复列表中点击了列表项的回复按钮，则评论输入框内会自动添加 “@” 后跟被回复者的用户名，同时在点击发布回复按钮时，向后台发送插入新回复的请求，请求内容中的 `responder` 字段，会变成刚才点击的回复项的作者 `id`。发送了插入新回复的请求后，还要分别给文章作者、评论作者、被回复者、发送插入新消息通知的请求，请求报文内容中所需的字段也是从 `store` 中获取。当后台返回回复成功的响应后，显示提示信息：回复成功，刷新页面后可见。

图 4-13：回复列表和回复输入框组件



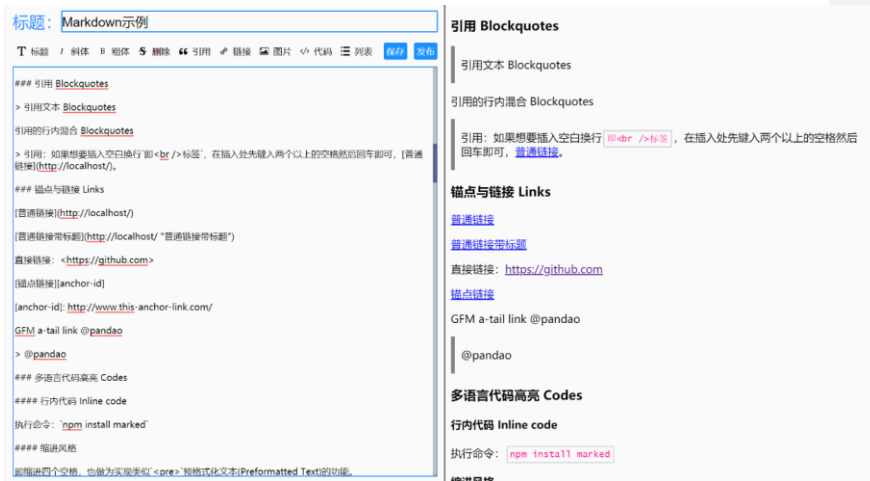


### 4.3.6 原创文章编辑器模块组件设计

原创文章编辑模块实现了一个可解析 markdown 语法并实时预览的组件，该组件下分编辑器组件和预览组件，编辑器组件是一个简单的 Textarea 框，监听该 Textarea 的 onChange 事件，当内容变化时，将用户输入的内容通过 marked.js 库实时解析为 HTML 标签，然后将解析出的 HTML 内容放在预览组件的 \_\_dangerouslyHTML 属性中，然后我们再写一个美观的 markdown 语法 CSS 样式即可。

当点击保存按钮时，将用户输入的内容存入 localStorage 中，当用户下一次进入该页面时，组件在挂载阶段从 localStorage 中读取上一次编辑的内容，当用户点击发布按钮时，则向服务端发送新增文章的 HTTP 请求。

图 4-14：原创文章编辑器组件



### 4.3.7 消息通知模块组件设计

消息通知列表组件主要是显示与用户相关的消息，对访客来说，包括自己评论下的回复和@自己的回复；对于博主，还包括博主所写的原创文章下的评论、回复、点赞。消息通知组件挂载阶段向后台请求消息列表，对于用户未读的消息，会有红点提醒，当点击查看消息的链接后，向后台发送将消息设置为已读的请求，同时将 store 的 user 模块中，用户的未读消息数减一。

图 4-15：消息列表组件



## 第五章 系统测试

### 5.1 测试目的

软件测试和软件开发是同时出现的。软件测试的目的在不同时期也各有不同，50 年代，软件测试的目的是证明软件系统能正常工作；70 年代，软件测试是为了发现系统中的错误和问题，即检错功能；80 年代，软件测试的目的是测试系统质量，即预测功能；而到了 90 年代，软件测试的目的则是为了控制质量，即预防功能；时至今日，软件测试的目的，按顺序包含了预防、检错、预测以及演示等功能，并且向着集成化、自动化、智能化的方向演进。[23]

### 5.2 测试方法

#### 5.2.1 单元测试

单元测试是指对软件中有独立功能的代码片段进行的检查和验证。对于单元测试中单元的含义，一般来说，要根据实际情况去判定其具体含义，对单元测试而言，测试的代码片段的逻辑功能应该是独立的，即一个代码片段能够完整地表达一个逻辑功能，只有对这种独立的代码片段，才有进行单元测试的必要。无论是面向过程的单元测试还是面向对象的单元测试，逻辑功能越独立越容易被测试。[24]

单元测试的方法其实主要只有两种：

- 1) 代码复查：人为地对代码片段进行审查。这种方法比较原始，而且很容易受人为错误的影响，比较依赖检查者的技术水平。
- 2) 编写测试程序测试：是目前比较主流的单元测试方法，编写测试程序需要测试对象具有很好的内敛性和可测试性，其次开发的测试驱动程序、稳定桩要保证测试用例的全面完整。目前已经有了非常多成熟的测试框架，如 CppUnit、JUnit、Jest 等。

本系统中，我们主要对后端数据库操作模块的每一个函数进行单元测试，确

保查询出的数据是正确的。

### 5.2.2 黑盒测试

黑盒测试也称功能测试、数据驱动测试或者基于规格说明书的测试。它是通过测试来检测每个功能是否都能正常使用。在测试中，把程序看作一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，在程序接口进行测试，它只检查程序功能是否按照需求规格说明书的规定正常使用，程序是否能适当地接收输入数据而产生正确的输出信息。黑盒测试着眼于程序外部结构，不考虑内部逻辑结构，主要针对软件界面和软件功能进行测试。[25]

黑盒测试是以用户的角度，从输入数据与输出数据的对应关系出发进行测试的。很明显，如果外部特性本身设计有问题或规格说明的规定有误，用黑盒测试方法是发现不了的。[26]

## 5.3 测试样例

### 5.3.1 单元测试

本系统单元测试采用了业界比较流行的 `javaScript` 单元测试框架 `Jest.js`，测试的是后端数据库操作模块 `BlogDB`，该模块包含的是一系列操作数据库的静态方法，我们在单元测试中，依次将这些方法放入测试框架中运行即可。

例如，测试函数 `getUserInfoByNickname`，代码如下：

```
describe('测试函数 getUserInfoByNickname ',()=>{
  test('根据用户名查找一个不存在的用户',()=>{
    expect.assertions(1); // 确保至少有一个断言被调用，否则测试失败
    return db.getUserInfoByNickname('闰南').then(data=>{
      expect(data).toBe(null);
    })
  });
  test('根据用户名查找一个存在的用户',()=>{
    expect.assertions(1); // 确保至少有一个断言被调用，否则测试失败
```

```

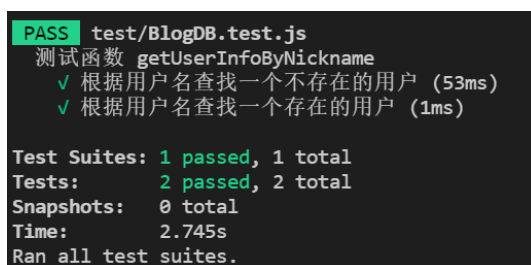
const expectedData = {
  uid: 1,
  nickname: '润楠',
  password: 'smielpf1204.',
  role: 'ADMIN',
  email: '1608272694@qq.com'
};

return db.getUserInfoByNickname('润楠').then(data=>{
  expect(JSON.stringify(data)).toBe(JSON.stringify(expectedData));
})
})
});

```

如果测试样例通过，则控制台显示如下：

图 5-1：单元测试控制台显示结果图



```

PASS test/BlogDB.test.js
  测试函数 getUserInfoByNickname
    ✓ 根据用户名查找一个不存在的用户 (53ms)
    ✓ 根据用户名查找一个存在的用户 (1ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:  0 total
Time:        2.745s
Ran all test suites.

```

### 5.3.2 黑盒测试

本系统黑盒测试过程就是运行并模仿用户操作使用系统，看是否符合预期效果，例如，登录模块的测试样例设计如下表：

表 5-1：登录模块测试用例

测试模块	用例描述	操作过程	预期结果	实际结果	偏差
登录模块	实现用户登录	输入一个已注册用户的用户名/邮箱和正确的密码	登录输入框消失，显示用户头像和用户未读消息数，进入原创文章页面，未出现“写文章”菜单项	如预期显示	无
	实现管理员登录	输入一个已注册管理员的用户名/邮箱和正确的密码	登录输入框消失，显示用户头像和用户未读消息数，进入原创文章页面，出现“写文章”菜单项	如预期显示	无
	处理登录信息异常	输入一个未注册用户的用户名和密码	显示“用户名或密码错误”的气泡提示信息	如预期显示	无
		输入一个已注册用户的用户名/邮箱和错误密码	显示“用户名或密码错误”的气泡提示信息	如预期显示	无

## 第六章 总结与展望

### 6.1 总结

本论文针对我想要定制个人博客页面的同时能查看我在简书和思否上的收藏夹的需求，介绍了一个基于 Node.js 和 React 技术栈的个人博客系统的设计与实现。

具体工作内容总结如下：

- 1) 介绍了当前大多数个人博客系统存在的问题与我的需求之间的冲突，提出了开发该系统的意义。
- 2) 阐述了项目开发用到的 Node.js、React.js、PostgreSQL 相关技术栈以及本系统采用这些技术栈的原因。
- 3) 分析了系统的功能需求，阐述了系统总体架构设计。
- 4) 详细介绍了各个功能模块的具体设计与实现方法。
- 5) 阐述了系统测试的目的，方法，并详细介绍了几个典型的测试样例。

### 6.2 展望

本系统已经满足了我的基本需求，但是仍然有许多需要改进的地方：

- 1) 跨浏览器兼容改进：本系统前端部分仅仅实现了在较新版本的 PC 端 Chrome、Firefox 等浏览器上的正确显示，还需要对其他多版本浏览器以及移动端浏览器设备进行适配。
- 2) 安全改进：本系统登录注册功能实现得比较简单，数据库中保存的以及通过 HTTP 传输的都是用户的明文密码，安全性较低。同时还需要再检查系统是否存在 SQL 注入，XSS 攻击等问题，并加以改进。
- 3) 部署改进：本系统服务端目前只是运行单个 Node.js 进程，可靠性较低，能应对的访问量也较少，未来可以考虑通过 K8S 部署在集群上。
- 4) SEO 改进：撰写了个人博客也需要被更多人看到，但是目前该系统在 SEO

方面基本没有优化，因此未来应该考虑针对 SEO 进行优化，让该系统上的文章在搜索引擎上能有相对较好的排名。



#### 参考文献:

- [1]. 陈文星,付继宗. Linux 下数据库 PostgreSQL 分析与应用[J]. 电脑开发与应用,2006(11):56-57.
- [2]. 纪洪波,崔立业. 使用 VC 开发 Windows 平台 PostgreSQL 数据库扩展[J]. 通化师范学院学报,2010,31(12):33-34+86.
- [3]. 刘鑫. MySQL 和 PostgreSQL 的对比选择[J]. 沈阳工程学院学报(自然科学版),2011,7(02):171-173+177.
- [4]. 王艳红,周军. 基于 Hadoop 的网络爬虫技术研究[J]. 吉林工程技术师范学院学报,2014,30(08):87-89.
- [5]. 陈利婷. 大数据时代的反爬虫技术[J]. 电脑与信息技术,2016,24(06):60-61.
- [6]. 邹科文,李达,邓婷敏,李嘉振,陈义明. 网络爬虫针对“反爬”网站的爬取策略研究[J]. 电脑知识与技术,2016,12(07):61-63.
- [7]. 魏程程. 基于 Python 的数据信息爬虫技术[J]. 电子世界,2018(11):208-209.
- [8]. 张薇,蒋胜山,吴汶芪. 基于 ASP.NET 中 AJAX 的 Web 新应用模型的设计[J]. 商场现代化,2008(23):13-14.
- [9]. Nicholas C. Zakas. JavaScript 高级程序设计[M]. 第三版. 北京:人民邮电出版社. 2012.
- [10]. 朴灵. 深入浅出 Node.js [M]. 北京:人民邮电出版社. 2013.
- [11]. Daniele Bonetta, Luca Salucci, Stefan Marr, Walter Binder. GEMs: shared-memory parallel programming for Node.js[J]. ACM SIGPLAN Notices, 2016, 51(10).
- [12]. Node.js: Using JavaScript to Build High-Performance Network Programs[J]. Tilkov, Stefan, Vinoski, Steve. IEEE Internet Computing . 2010 (6).
- [13]. 冯新扬,沈建京. REST 和 RPC: 两种 Web 服务架构风格比较分析[J]. 小型微型计算机系统, 2010, 31(07):1393-1395.
- [14]. 黄扬子. 基于 NodeJS 平台搭建 REST 风格 Web 服务[J]. 无线互联科技, 2015(16):57-59.
- [15]. 邬文怀. 基于 React 的超级账号教学支持系统前端的设计与实现[D]. 南京大学, 2018.
- [16]. 基于 Dom Diff 算法分析 React 刷新机制[J]. 严新巧, 白俊峰. 电脑知识与技术. 2017(18).
- [17]. 深入 React 技术栈[M]. 人民邮电出版社, 陈屹, 2016.
- [18]. 惠韶文. 信息系统开发的可行性研究[J]. 中文信息, 1994(06):24-25.
- [19]. 蒋跃星. 论信息管理系统的可行性研究[J]. 企业技术开发, 1998(07):17-18.
- [20]. 陈辰,王萌,程旭. 基于路由模式的前端框架设计与改进[J]. 电脑知识与技术, 2018, 14(12):53-54+58.
- [21]. 屈展,李婵. JSON 在 Ajax 数据交换中的应用研究[J]. 西安石油大学学报(自然科学版), 2011, 26(01):95-98+122.
- [22]. 崔莹,刘兵. Node.js 与 Express 技术在计算机课程教学中的应用[J]. 软件导刊, 2016, 15(09):190-192.
- [23]. 黄清清. 软件测试与测试方法浅析[J]. 舰船电子工程, 2004(01):32-35.
- [24]. 王雪男. 给程序员补上单元测试这一课[J]. 程序员, 2005(02):50-53.
- [25]. 什么是黑盒测试?[J]. 电子质量. 2019(02).

- [26]. Dusica Marijan, Nikola Teslic, Miodrag Temerinac, Vukota Pekovic. On the Effectiveness of the System Validation Based on the Black Box Testing Methodology[J]. Journal of Electronic Science and Technology of China, 2009, 7 (04) :385-389.

## 致谢

大学本科四年以这次毕业设计画上了圆满的句号，在此，我要感谢大学四年间给予过我帮助与关怀的每一位同学和老师，特别是感谢我的论文指导老师温武少教授，感谢他对我毕业设计给予的悉心指导，也感谢他对我生活和未来前途提出的建议，让我受益匪浅。

同时也要感谢我实习过的华为、Bigo 公司，是在这两家公司实习期间让我学到了很多，拓展了视野，让我在技术上，与人交流的技巧上都有了长足的进步。

最后，感谢我的父母为我的学业提供的无条件的支持，是他们在迷茫无助的时候给予我鼓励和陪伴，他们永远是我亲最爱的人。

衷心祝愿每一个帮助、支持我的人身体健康，生活顺利。

**毕业论文（设计）成绩评定记录**  
**Grading Sheet of the Graduation Thesis (Design)**

<p>指导教师评语 <b>Comments of Supervisor:</b> 本论文基于 postgresSQL 数据库，以 Node.js 搭建后台，前端基于 React 技术栈，设计并实现了一个个人博客网站，可展示在其他博客平台所收藏的文章的功能，有一定的参考意义。本论文虽然选题不算新颖，但作者能够系统化地利用所学新知识研究这个问题，体现其具备利用专业计算机知识分析解决实际工程问题的能力。 论文总体内容结构完整，写作基本规范，体现该生可解决计算机工程技术的实际问题，已经达到了中山大学本科学士学位论文要求水平。</p> <p>成绩评定 <b>Grade:</b></p> <p>指导教师签名 <b>Supervisor Signature :</b> _____ <b>Date:</b> _____</p>	
<p>答辩小组意见 <b>Comments of the Defense Committee:</b></p> <p>成绩评定 <b>Grade:</b></p> <p>签名: _____ <b>Date:</b> _____ <b>Signatures of Committee Members</b></p>	
<p>院系负责人意见 <b>Comments of the Academic Chief of School:</b></p> <p>成绩评定 <b>Grade:</b></p>	

签名 Signature:	院系盖章 Stamp:	Date:
------------------	----------------	-------