****

**本 科 毕 业 论 文（设计）**

|  |  |
| --- | --- |
| 课题名称 | 基于Web交互电子书的内容生成系统设计与实现 |
| 学 院 | 计算机科学与网络工程学院 |
| 专 业 | 计算机科学与技术 |
| 班级名称 | 计科172班 |
| 学生姓名 | 崔玉勤 |
| 学 号 | 1706100090 |
| 指导教师 | 李传中 |
| 完成日期 | 2021年5月1日 |

教 务 处 制

[基于Web交互电子书的内容生成系统设计与实现](javascript:f_viewtopic('4a409232674baeba016771a4081e145c');void(0);)

计算机科学与技术172班 崔玉勤

指导教师:李传中

摘要 本系统的设计主要是为了对接交互电子书的阅读系统，阅读系统所展现的各种内容都可以通过本系统来生成。本系统实质是一个编辑创作平台，不仅可以用markdown语法进行写作编辑，还可以对创作的文章进行整理归纳，这让用户可以按自己的喜好自由快速地创作出丰富好看的电子书，可以大大激发用户的创造热情。为了满足电子书内容生成系统的要求，降低耦合，系统架构为前后端分离，前端使用Vue3框架开发页面，后端使用Golang编写程序，使用Mysql 持久化业务数据，使用Docker进行项目部署，并使用Gitee进行项目代码版本的管控。

本系统重点实现了文件的管理，基于markdown语法的交互电子书编辑创作内容生成等功能。

关键词 内容生成；编辑；markdown；电子书；文件管理；

**ABSTRACT** The design of this system is mainly for docking the reading system of interactive e-books, and all kinds of content displayed by the reading system can be generated by this system. This system is essentially an editing and creation platform. It can not only use markdown grammar for writing and editing, but also organize and summarize the created articles. This allows users to freely and quickly create rich and beautiful e-books according to their own preferences, which can greatly stimulate User's enthusiasm for creation. In order to meet the requirements of the e-book content generation system and reduce coupling, the system architecture is separated from the front and back ends. The front-end uses the Vue3 framework to develop pages, the back-end uses Golang to write programs, uses Mysql to persist business data, uses Docker for project deployment, and uses Gitee Carry out the management and control of the project code version.

This system focuses on file management and interactive e-book editing and creation content generation based on markdown syntax.

**KEY WORDS** Content generation; editing; markdown; e-books; file management;

目录

[1. 前 言 5](#_Toc21013)

[1.1 课题设计开发的背景和意义 5](#_Toc17924)

[1.2 文章组织结构 5](#_Toc7505)

[2. 系统开发环境和主要技术背景 6](#_Toc21452)

[2.1 节开发环境 6](#_Toc14883)

[2.2 Vue.js框架 6](#_Toc27898)

[2.2.1 Vue.js简介 6](#_Toc22868)

[2.2.2 Vue.js全家桶 7](#_Toc8790)

[2.3 Golang服务端开发语言 8](#_Toc1803)

[2.4 Docker容器技术 8](#_Toc7150)

[2.5 Mysql数据库 9](#_Toc19893)

[3. 需求分析 10](#_Toc31419)

[3.1 系统设计目标 10](#_Toc12676)

[3.2 系统可行性分析 10](#_Toc1225)

[3.2.1 技术可行性分析 10](#_Toc20619)

[3.2.2 经济可行性分析 11](#_Toc22344)

[3.2.3 操作可行性分析 11](#_Toc5365)

[3.3 系统功能的分析 11](#_Toc30316)

[4. 系统的总体设计 13](#_Toc4609)

[4.1 系统总体架构设计 13](#_Toc4359)

[4.2 系统核心业务模块设计 13](#_Toc10643)

[5. 系统的详细设计和具体实现 15](#_Toc15940)

[5.1 详细设计简要说明 15](#_Toc18337)

[5.2 代码结构 15](#_Toc28142)

[5.3 模块功能的设计和实现 17](#_Toc10698)

[5.3.1 用户管理模块 17](#_Toc18462)

[5.3.2 文件管理模块 23](#_Toc7379)

[5.3.3 编辑创作模块 27](#_Toc24405)

[参考文献 30](#_Toc9951)

[致谢 31](#_Toc28733)

1. 前 言
   1. 课题设计开发的背景和意义

现如今的21世纪20年代，互联网技术有了很大的进步和发展。用户对于电子书也提出了更高的要求，不只是期待有基本的文字、图片内容，还希望有如网页般更丰富的其他的内容呈现。那么这些那么多的内容，就需要有专门用来进行编辑创作的内容生成工具，本系统就是这个工具角色。

内容生成系统本质是一个编辑创作平台，每个注册了的用户都有属于自己的文件空间，用户可以创建文档和文件夹，并对它们进行整理归纳，删除的文档或文件夹保留在回收站，这给了用户有悔改的机会，而且还有基于时间线的最近编辑功能，方便记录创作的每个时刻。在编辑创作电子书时，采用的是基于markdown语法来进行编辑创作， 除了常见的如标题、图片、粗体、引用等常见语义定义，还有数学公式、布局、流程图、时序图、echarts、geogebra等的有特色的语义定义，可以边编辑边预览，而且会实时保存文章到后台数据库中，确保数据不被丢失，下次进来继续接着编辑创作。这么方便丰富的内容生成功能，相当于给用户搭建了一个自由发挥的无比宽敞的创作舞台，不仅可以挖掘出用户的创作灵感，还大大激发了用户的创作热情，这在一定程度上可以丰富电子书平台的内容。

* 1. 文章组织结构

根据课题要求和具体内容，本论文主要分为五章节：

第一章节，前言，首先介绍现如今用户对电子书的要求越来越高，接着引出就要有一个功能完备的电子书内容生成创作平台，最后对本文的文章组织结构等做简要介绍。

第二章节，系统开发环境和主要技术背景，首先简要介绍项目开发以及部署环境，接着重点介绍系统开发所采用的主要技术以及其依赖，让读者对本系统的技术背景有个大概的了解。

第三章节，需求分析，主要介绍系统的设计目标以及可行性分析，让读者了解本系统的需求

第四章节，系统的总体设计，主要介绍系统的架构是如何设计和实现，主线业务流程是怎样，让读者对本系统的架构有所了解。

第五章节，系统的详细设计和具体实现，主要介绍系统的核心功能的设计、描述和实现，让读者知晓本系统的实现。

1. 系统开发环境和主要技术背景
   1. 节开发环境

硬件环境：固态硬盘容量250G，内存容量8G，2台主机

开发环境：windows10

部署环境：Ubuntu18

代码编辑工具：VsCode

浏览器：Chrome90.0.4430.212（正式版本） （64 位）

API调试、HTTP请求/响应工具：PostMan

后端开发语言：GoLang1.16

服务端的数据库：MySql

前端开发语言：Html，JS，Css

前端框架：Vue.js框架

容器化部署：DocKer

* 1. Vue.js框架
     1. Vue.js简介

Vue (读 /vjuː/，就如view) 是一套用于构建用户界面的前端渐进式框架，渐进式，顾名思义就是可以自底向上，由简单到复杂地去使用Vue。它在2014年开始发布以来，就已经成为目前世界上最流行的三大前端框架之一，特别是由于它的创作者是一位中国人，这让它成为中国开发者目前最热的一款前端框架，热情的中国开发者也为Vue社区生态贡献了许多有用的第三方库。

Vue只关注视图层，不仅十分方便地与第三方库或已有项目整合，还对新手特别友好，十分易于上手。它的性能高，占用内存很小，只关心视图和数据，在底层的实现上是基于虚拟DOM,结合响应系统，计算机能够通过各种看不到的预先javascript计算，把最终的DOM操作计算出来并做优化，这个DOM操作是预处理操作，没有真实地去操作，这把DOM 操作次数减到了最少，完全能够为复杂的单页应用提供驱动。

* + 1. Vue.js全家桶

Vue.js有著名的全家桶系列，按字面意思来说，其实就是开发一个完整的中或大型单页面应用项目的核心插件构成。

Vue-cli，也称脚手架，Vue.js的官方定义开发标准工具，用于快速构建单页面应用项目，它相较与Script便签引入，有功能丰富、容易扩展、可配置、面向未来等特点。

Vue-Router,是Vue.js的路由管理器，这插件是官方用来补充由于之前的Vue开发时对路由支持的不足，在如今最新版本中是核心深度集成，越来越重要，它让构建单页面应用变成了一件十分容易的事。功能包括嵌套的路由或视图表、基于组件路由配置、模块化、通配符、路由参数、视图过度效果、细粒度的导航控制、历史模式或Hash模式等。

Vuex,是Vue.js 的应用程序开发状态管理模式,它所有组件的状态是采用集中式存储管理应用的，而且根据相应的规则做到保证状态以一种可预测的方式发生变化。它也集成了官方devtools extension的调试工具，便于开发人员的调试以及提高开发效率。当然，如果我们不打算开发大型单页面应用，使用Vuex就会显得繁琐冗余，那么这时侯就最好不要使用Vuex，而是使用简单的store模式来满足开发。在本系统中，由于组件间的关系复杂，数据不只是局限于在父子组件间传递，所以为了方便不同组件间的数据共享和组件通信，采用了Vuex,这大大提高了我的开发效率，最重要的是愉悦了在整个开发过程中的心情。

* 1. Golang服务端开发语言

Golang（也称Go）是美国谷歌Google开发的开源编程语言，它是具有自动垃圾回收、强静态类型、并发型、编译型等功能的编程语言，是非常年轻的一门语言。它起源于2006年，开源于2009年，稳定版发布于2012年，Google对其寄予了很大的厚望。Go语言拥有原生的多核并发设计优势，从底层原生支持并发，无须第三方库、开发者过多开发经验和编程技巧，它主要的目标是有接近于C/C++等编译型语言的性能与安全性和Python等动态语言的开发速度。目前中国有很多互联网巨头即将或者已经在使用 Go语言开发或重构旧系统，如字节跳动就主要使用Go语言以及腾讯、美团都开始向社会招聘Go语言方面的工程师。

Go语言是居于goroutine实现并发的，采用了CSP并发模型，goroutine类似线程而又不是线程，可以把它理解为一种虚拟线程。Go语言执行程序时会参与调度goroutine，根据CPU个数和每个CPU的核数来合理地分配goroutine，尽可能地使用CPU性能。开启一个goroutine的代价是很小的，大约2KB的内存，这样你能轻松地创建数百万个goroutine。

Go语言目前有很多优秀的标准库，可以用很少的代码量就完成项目的开发，如只需几十行代码就可以开启一个http服务。它的内置数据结构slice和map都是十分常用而且又好用的数据结构，它们都可以自动合理地扩容。而它的chan管道，使得不要通过共享来通信而是通过通信来共享的理念得以很好的实现。它语法简洁，进行了大量简化，去掉不必要的表达式括号，十分易学，学习曲线平缓，比其他语言有少的学习成本，更高的开发效率。

* 1. Docker容器技术

Docker是一个DotCloud开源的应用容器引擎，用[Go语言](https://www.runoob.com/go/go-tutorial.html" \t "https://www.runoob.com/docker/_blank)来开发，且遵从 Apache2.0协议。开发者可以将它们的任何应用包装运行在Docker容器上面，由于Docker容器在每个系统都是一样的，所以可以运行在Docker容器的应用就实现了跨服务器、跨平台、一次配置和一键部署，而且由于沙箱环境实现了轻型隔离，多个容器间是相互不会影响到的，这可以用于搭建开发测试环境或者用来部署可扩展web应用，解决由于程序执行所需要的环境发生变化导致的各种BUG,大大简化操作，实现虚拟化。

Docker跟传统虚拟机最大的区别是，传统虚拟机技术是首先虚拟出一套硬件，接着在其上面运行完整的操作系统，最后在该系统上接着运行所需应用进程。而这个操作系统非常耗资源，可能其硬盘就使用了几十个G，内存也使用了几个G,在启动系统时还得乖乖地检测或加载一遍各类硬件资源，整个过程可以说十分缓慢而不必要。而Docker容器是直接运行于宿主机的内核中，但是DOcker容器自己没有内核，不必做硬件虚拟操作，所以，Docker容器就比虚拟机更加轻便。它有自己的文件作用域，容器间不受影响。

* 1. Mysql数据库

Mysql是目前最火热的一款关系型数据库，特别是在Web应用方面用户有很高的选择性。它作为一种关系型数据库管理系统，就是关系数据库不是将所有相关数据都放在一个统一的大仓库中，而是将其保存在各不相同的表中，这会提高灵活性和增加速度。Mysql开源，成本低，使用标准的SQL数据语言语法，还支持可以在一定时间内处理成千上万条记录的大型数据库，同时在多个系统都可以运行，安全性高，最重要的是，它支持多种开发语言，十分方便开发者调用操作Mysql。

Mysql的默认存储引擎为innodb,这是一种事务型的存储引擎，采用MVCC来支持高并发，而且在四个标准隔离级别(未提交读、提交读、可重复读、可串行化)中，它的默认级别为可重复读，在这个级别下，Mysql是通过MVCC加 Next-Key Locking来防止幻读的，在并发事务时，确保了执行性能。Mysql拥有多种形式的索引，最常的就是B+Tree索引，这种数据结构的索引应用范围广，查找速度非常快，在实际项目开发过程中，通过合理的设计并使用索引，可以大大提高数据库的查询速度，提高系统的执行速度。Mysql具备数据备份恢复功能，还可以通过配置来做到读写分离，保证系统的高可用，高并发，做到尽可能的减少数据方面的影响。

1. 需求分析

## 系统设计目标

本系统的主要设计目标是实现交互电子书的内容生成，用户通过本系统可以快速方便地编辑创作出在交互电子书阅读页面所展现的各类多姿多彩的内容，并且可以拥有文件管理增删改查操作，通过整理归纳文档和文件夹可以类似生成一本电子书，实现各类数据实时持久保存。

## 系统可行性分析

* + 1. 技术可行性分析

本系统的开发模式采用了前后端分离的模式，这是目前最流行的项目开发模式，有许多的建议和经验分享资源，前后端通过接口来进行业务数据的请求和响应。前端使用了Vue.js框架,后端使用了Golang语言,数据用了Mysql,部署用了Docker容器技术。

前端Vue.js框架是目前市场最火热的三大前端框架之一，目前该框架更新维护密集，第三方库生态资源丰富，而且对于要求掌握的Js、html、css程度较低，易于新人上手。在前端页面中基于markdown语法解析需求，在github上有相关的第三方库，以及其他各种个性化的markdown语法拓展都很多，而且这些第三方库资源采用的技术栈也是Vue,对应的README.md说明详细又专业。

后端Golang语言的语法跟动态语言Python一样具有更高接近人类大脑思考的可读性，而且其启动http服务，log日志，io文件操作都集成在了标准库中，不必再到别的网上仓库去拉取，至于mysql数据库操作，也提供了数据库驱动接口来调用数据库。它还能在较少的硬件资源中获取到相当高的I/O吞吐和并发表现，不必纠结于硬件资源。

数据库Mysql,docker容器部署都没有特别的需求。mysql这类关系型数据在大学专业课程中有了深入的学习了解，docker容器技术在大学操作系统课中对于虚拟机也有一定的了解，只需区别它们的不同，况且docker容器是用golang开发的，经常使用的操作命令也不多，也不难。

以上的各类技术，经过仔细分析，不存在特别难跨过的技术门槛。基本都只需稍稍学习较基础的部分，加上查阅相关的资料和查找开源库就可以进行本系统的开发工作。

* + 1. 经济可行性分析

本系统所使用到的所有技术都在github上开源或者有免费的官方指导文档，相关的开发工具也是可以无限免费下载和安装到最新版本，由于本系统是毕业设计而不涉及到商业用途。所以，在软硬件上的成本可以说是免费，从经济上来说，属于经济非常友好型。

* + 1. 操作可行性分析

本系统的操作界面采了Element-ui框架，这个ui框架十分匹配前端Vue框架，也被官方重力推荐。所以其操作和样式是十分友好于用户的，不仅样式美观大气，其操作也是十分符合各类群体的操作习惯。本系统每个功能模块操作区域有合理的分隔，而且有明确温馨的中文提示语或交互提示磨板，方便用户进行各类页面操作。

## 系统功能的分析

账号登录：用户第一次来到本系统，可以在登录界面点击登录模板的账号注册文字区域来进入账号注册页面。输入一个还未注册过的账号和密码，可以点击注册按钮来完成注册。如果已有该账号，则提示其用密码登录。如为新账号，则直接进入主页面。

密码登录：用户在登录模板的账号输入框上可以输入账号，在密码输入框输入密码，通过点击登录按钮来完成登录。如果账号和密码有未输入的，则提醒其先完整输入。输入错误的，则提醒其重新输入。账号不存在的，则提醒他先注册。账号和密码都正确，则进入主页面。

文件新建：用户可以在任何文件夹的任何位置来新增文档或文件夹，以扩充文件树。

文件删除：用户不想要的文档或文件夹可以点击删除按钮，删除后的文档或文件夹会保存在回收站内，以提供用户一个悔过或误删的机会。

文件重命名：用户可以重命名文档或文件夹的标题文字。

文件浏览：用户在左侧文件区域点击某个文件夹，在右侧界面就会显示出该文件夹的祖辈文件夹链和该文件夹下的下一级所有文档或文件夹，对右侧文档或文件夹的操作，跟在左侧文件区域的操作一模一样。

文件整理归纳：用户可以通过拖拽某个文档或文件夹到另外一个文档或文件夹前面或里面或后面的位置来重新整理归纳该文件数。

文件恢复：用户发现误删或者想要悔改已删文档或文件夹，在回收站页面能够找到该文档或文件夹。点击恢复按钮，可以恢复该文档或文件夹。

文件彻底删除：用户在回收站页面可以通过点击彻底删除，让该文档或文件夹彻底消失在本系统，无法再恢复。

文档记录：用户在最近编辑页面可以清晰看到以时间线从最近到最远的形式展现的文档排序列表，点击某个时刻的文档，可以进入接着在该文档继续编辑创作。

文档编辑：用户在编辑区内可以使用基于markdown语法的编辑创作，除了有常见的markdown语义外，还得有如latex/katex格式的数学公式、流程图、时序图、echarts图、geogebra动态几何图等有个性的语义拓展。

文档实时预览：用户在编辑创作过程中，可以实时编辑实时预览。可以在左侧文件区域看到文章的大纲，并且点击某个大纲，就会在编辑区和预览区定位到该大纲的大体位置。

文档保存：系统可以做到及时地将编辑创作数据保存起来，确保不丢失不残缺，下次进来，可以接着继续编辑创作。

1. 系统的总体设计
   1. 系统总体架构设计

本系统主要分为三层，它们分别为前端视图层，后端业务逻辑层，数据库持久化层。

前端视图层：提供一个操作友好且美观的前端web界面，用来展现用户的信息，包括其拥有的各类文件信息，以及文档编辑和预览效果信息。同时用户可以在该web界面上进行各种编辑创作和文件整理归纳操作。用户在界面操作以后，前端通过Http请求同后端业务逻辑层以Json格式将数据打包后进行数据传递，接着等到后端业务逻辑层响应后，将Json数据包解析处理，并把最终结果渲染呈现在web界面上。

后端业务逻辑层：主要给前端视图层提供登录注册，文件树创建、查询、更改，电子书创建、查询、更改，Geogebra动态几何图静态文件查询、获取等的基于Http协议的Api接口。业务逻辑层通过监听Http请求,对于前端视图层发起的请求被动地进行路由分发到相关业务逻辑处理函数，该函数会处理相应业务，期间涉及数据交互就会调用数据库持久化层进行操作。处理后会将响应信息封装好以json格式响应给前端视图层。

数据库持久化层：本系统通过采用关系型数据库Mysql来实现数据的持久化。对于用户的账号密码等个人信息、用户的文件信息、电子书信息等数据会存储持久化到服务器的硬盘中，保证系统数据的完整性。

* 1. 系统核心业务模块设计

本交互电子书内容生成系统根据实际业务功能来划分，可以分为用户管理模块、文件管理模块和编辑创作模块。以下对这几个模块的设计划分做一下说明：

用户管理模块：

对于用户管理模块，主要实现用户账户注册和密码登录功能。在实际实现这些功能时，为了保证用户数据的安全性，前端视图层密码区域不能明文显示，在数据持久化存储时也不能明文密码存储，而是采用MD5加密算法加密后再存储。

用户注册：用户通过输入账号和密码来注册，其中账号必须是本系统唯一的，不能出现账号名冲突，出现冲突是无法完成注册操作。而且对账号和密码有长度限制。

用户登录：用户成功注册一个账号后，可以用已注册账号和密码来进行登录操作，同样会有对账号和密码有长度限制。如果能够登录成功，就会在浏览器中生成一个token,系统根据这个token来维持在线状态，只有token存在且正确才能在主页面进行各种操作。

文件管理模块：

文件整理模块主要实现的是对用户文件空间下文档和文件夹的增删改查操作功能，为了方便实现这些功能，保持维护两份映射于文件书的对象数组，一份为我的文件树数组，一份为删除文件树数组。每个对象映射一个文件节点，这个对象节点的属性有节点ID、创建时间、编辑时间戳、节点名称、节点类型、节点对应电子书的ID和孩子节点数组。

用户在任何文件夹的任何位置可以新增文档或文件夹，那么就在操作的节点孩子节点数组里生成一个文件节点来实现。用户删除不想要的文档或文件夹，删除后的文档或文件夹保存在回收站内，那么就在我的文件树中删除该节点，并在删除树中插入该节点。用户重命名文档或文件夹的标题文字，只需更改节点的名称。用户在左侧文件区域点击某个文件夹，在右侧界面显示出该文件夹的祖辈文件夹链和该文件夹下的下一级全部文档或文件夹，这通过把这个节点作为入口，往前查找该节点的父级节点来生成文件夹链，通过遍历该节点的孩子节点来显示下一级全部节点。用户通过拖拽某个文档或文件夹到另外一个文档或文件夹前面或里面或后面的位置来重新整理归纳该文件数，只需删除该节点接着插入它最后放置所在我的文件树的位置。

用户想要恢复已删文档或文件夹，就要遍历删除文件树数组所有节点来在回收站页面显示所以已删除文档或文件夹。用户选择某个节点进行恢复，就在删除文件树数组删除该节点并加入我的文件数数组的尾部。点击彻底删除，就在删除文件树数组删除该节点。

用户在最近编辑页面看到以时间线展现的文档排序列表，只需通过层级遍历我的文件树数组，筛选出类型为文档的节点，接着按照时间戳的大小从大到小重新排序，最终展现在界面上。

编辑创作模块：

编辑创作模块主要实现的是对文档编辑、浏览和保存功能，其中对文档的编辑，使用的是基于markdown语法的编辑创作，除了常见的markdown语义外，还包括如latex/katex格式的数学公式、流程图、时序图、echarts图、geogebra动态几何图等有个性的语义拓展，这就需要添加相对应的markdown-it插件，并对有关插件进行修改添加代码，以符合本系统的需求。对于geogebra动态几何图这种第三方库没有的语义拓展需要自己编写新的解析渲染规则。用户在编辑创作过程中，为了实现实时预览，需要每次修改都重新解析渲染源编辑数据。用户能够定位文档大纲位置，就获取文档解析后的Html数据，找出是标题的DOM节点，计算其距离上层控件的距离，并使编辑和预览区滚动这个距离的长度。系统能将编辑创作数据保存起来，就需在每次修改重新修改后，在一定时间内触发保存函数，将目前最新的数据持久到数据库。

1. 系统的详细设计和具体实现

## 详细设计简要说明

在这章，我会介绍系统的多个核心功能模块的详细设计和实现。期间会通过附带相关核心代码并作一定的实现原理解析，有必要可能会展示有关效果。介绍的主要模块有：文件管理模块，编辑创作模块，用户管理模块。

* 1. 代码结构

如下为本系统项目包主要结构一览表：

├── backend

│ ├── common

│ │ └── common.go

│ ├── controllers

│ │ ├── ebook.go

│ │ ├── ggb.go

│ │ ├── login.go

│ │ └── tree.go

│ ├── doc

│ │ └── schema.sql

│ ├── ggb

│ │ └── GeoGebra

│ ├── go.mod

│ ├── go.sum

│ ├── main.go

│ ├── sql

│ │ ├── ebook.go

│ │ ├── initdb.go

│ │ └── tree.go

│ └── vendor

├── package.json

├── src

│ ├── App.vue

│ ├── main.js

│ ├── router

│ │ └── index.js

│ ├── store

│ │ └── index.js

│ └── views

│ ├── layout

│ │ ├── aside

│ │ │ ├── aside.vue

│ │ │ └── FM.vue

│ │ ├── layout.vue

│ │ ├── main

│ │ │ └── main.vue

│ │ └── page

│ │ └── page.vue

│ └── login

│ └── login.vue

└── vue.config.js

* 1. 模块功能的设计和实现

* + 1. 用户管理模块

账号注册：

注册页面如图5-1所示：

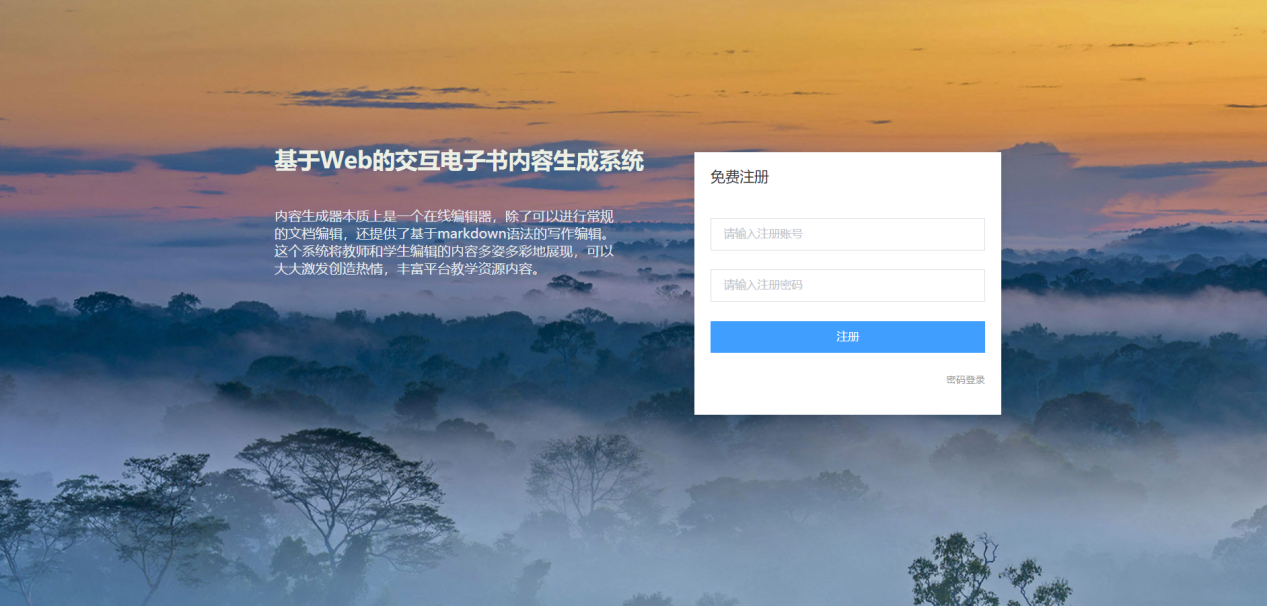


图5-1 注册页面

页面的背景图片链接于微软的搜索引擎Bing,由于它每天都会提供不同的有意义的图片，给人一种有故事的感觉，因此很适合作为注册页面的背景图片。CSS代码如下：

#register\_boby {

width: 100%;

height: 100%;

background-image:url("https://api.dujin.org/bing/1920.php");

position: fixed;

}

用户来到注册页面，输入账号和密码注册，点击注册，向后端API提交表单，成功就进入主页面，不成功，就会跳出对应的错误提示信息。

前端主要代码如下：

//注册

axios.post('/api/inserttree', {

userName: that.loginForm.userName,

   PassWord: that.loginForm.PassWord,

    types: 1,

   tree: [{

id: new Date().getTime(),

time: this.CurentTime(),

editorTime: new Date().getTime(),

label: "文件夹",

type: true,

     url: '# 文件夹',

children: []

}],

    deleteTree: []

}).then(res => {

    console.log("/api/inserttree:", res.data.status);

    if (res.data.status == 200) {

    that.$store.commit("setToken", that.loginForm.userName)

    that.$store.commit("setUserName", that.loginForm.userName)

that.$router.push({ path: "/editor" })

    } else {

     that.$message({

     showClose: true,

     message: "已有本账号，请用密码登录！",

     type: "warn"

    })

    }

}).catch((err) => {

    console.log(err)

})

以上代码，在点击注册时，如果账号和密码都有正确输入，就会封装一个结构对象，并填入初始化数据，请求后端接口。成功收到响应信息体，如果判断到注册成功，就设置token值为本账号名，跳到主页面。如果是注册过了，就跳出提示信息。如果接口响应出错，就打印出错误信息。

后端会监听注册接口，收到请求，就提取重要信息，查找插入数据库来完成注册，并封装好响应信息传到前端。后端业务处理代码如下：

//请求的数据

requestData := common.GetData(r)

requestDataTree, err := json.Marshal(requestData["tree"])

common.CheckError(err)

requestData["tree"] = requestDataTree

requestDataDeleteTree, err := json.Marshal(requestData["deleteTree"])

common.CheckError(err)

requestData["deleteTree"] = requestDataDeleteTree

  //返回的数据体

 responseData := common.ResponseData{}

  //数据库操作

 sql.InsertTree(requestData, &responseData)

  res, \_ := json.Marshal(responseData)

 w.Write(res)

以上代码，是获取到请求的数据map字典requestData，然后对对应的键值做字符串类型的转化，传入数据库操作函数，操作完后，返回转化为Json格式的数据包到前端。

数据库操作代码如下：

GetTree(requestData,responseData)

if responseData.Object.(User).Id != "" {

responseData.Message = "已有账号！"

     responseData.Status = 300

     return

  }

res, err := db.Exec("insert into user (username, password, types, tree, delete\_tree,create\_time)"+"values(?,?,?,?,?,now())", requestData["userName"], requestData["PassWord"],requestData["types"],requestData["tree"], requestData["deleteTree"])

common.CheckError(err)

GetTree(requestData, responseData)

responseData.Message = "成功生成"

responseData.Status = 200

密码登录：

用户来到密码页面，输入账号和密码登录，点击登录按钮，向后端API提交登录数据表单，成功就进入主页面，不成功，就会跳出对应的错误提示信息。

前端主要代码如下：

//登录

axios.post('/api/login', {

userName: that.loginForm.userName,

PassWord: that.loginForm.PassWord

  }).then(res => {

      console.log("/api/login:", res.data.status);

      if (res.data.status == 200) {

that.$store.commit("setToken", that.loginForm.userName)

       that.$store.commit("setUserName", that.loginForm.userName)

       that.$router.push({ path: "/editor" })

    } else if (res.data.status == 300) {

            that.$message({

            showClose: true,

             message: "密码错误，请重新登录！",

            type: "error"

           })

           this.PassWord = "";

      } else {

             that.$message({

                 showClose: true,

                  message: "不存在这个账号，请先注册！",

                  type: "warn"

              })

        }

     }).catch((err) => {

          console.log(err)

     })

以上代码，在点击登录时，如果账号和密码都有正确输入，就会封装好账号和密码数据到表单中，请求后端接口。成功收到响应信息体，如果判断到登录成功，就设置token值为本账号名，跳到主页面。如果是登录密码错误或者不存在该账号，就跳出提示信息。如果接口响应出差，就打印出错误信息。

后端会监听登录接口，收到请求，就提取重要信息，操作数据库来完成登录，并封装好响应信息传到前端。

数据库操作代码如下：

rows, err := db.Query("Select id , username, password, tree, delete\_tree, create\_time from user "+" where username = ?", requestData["userName"])

common.CheckError(err)

defer rows.Close()

var user User

for rows.Next() {

    err = rows.Scan(&user.Id, &user.UserName, &user.PassWord, &user.Tree,

&user.DeleteTree, &user.CreateTime)

    common.CheckError(err)

}

responseData.Object = user

if user.Id != "" {

     if user.PassWord == requestData["PassWord"] {

         responseData.Message = "成功登录"

          responseData.Status = 200

          return

     }

responseData.Message = "密码错误，请重新登录！"

        responseData.Status = 300

        return

  }

  responseData.Message = "不存在这个账号，请先注册！"

  responseData.Status = 400

以上代码中user变量是用来作为用户信息的数据结构。首先先查询该账号存不存在，用user来接收用户信息，存在用户信息，就比较数据库查出的密码和前端传来的密码是否同样，如果同样，就封装登录成功信息，如果不同样，就封装密码错误信息，如果不存在用户信息，就封装请先注册信息，返回。

* + 1. 文件管理模块

文件整理归纳：

文件整理归纳页面如图5-2所示:

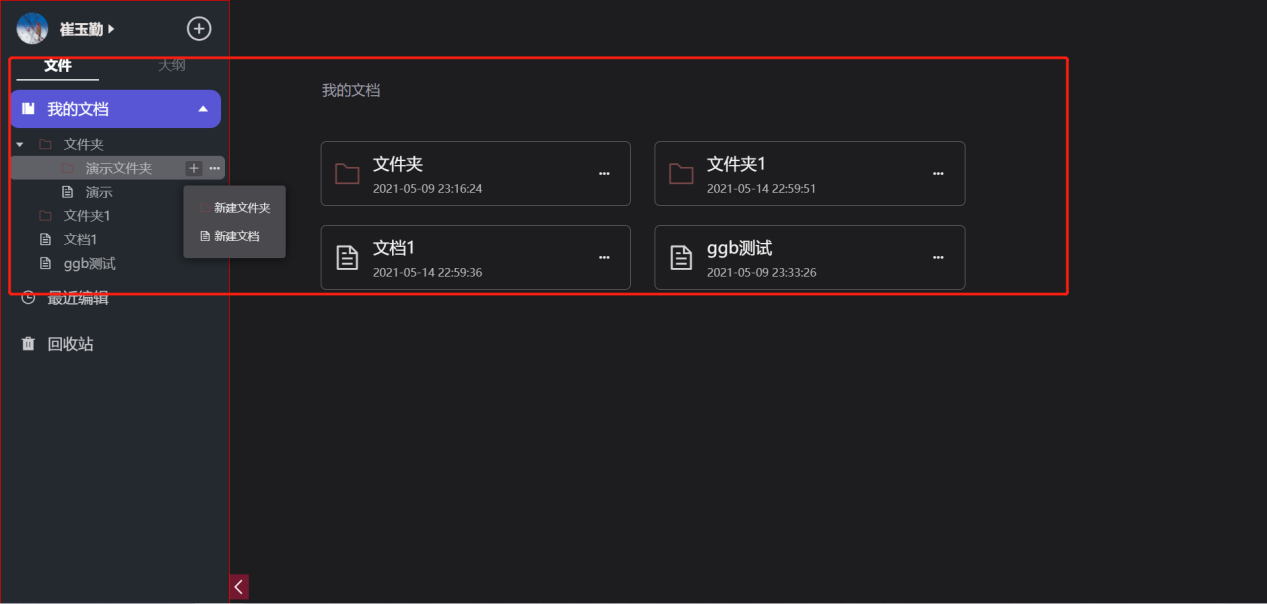


图5-2 文件整理归纳页面

左侧显示的文件树区域可以拖拽，左右两侧的节点都可以进行新建文档或文件夹，重命名，删除这些操作，点击某个节点，如果为文档，就进入编辑创作页面，如果为文件夹，就显示该文件夹的祖文件夹链以及该文件夹下一级所有的节点。

其中新增文件夹节点代码：

//new 一个孩子

const newChild = {

    id: new Date().getTime(),

time: this.CurentTime(),

     editorTime:new Date().getTime(),

     label: this.dialog.dialogInput,

     type: true,

     url: '# ' + this.dialog.dialogInput,

     children: []

    };

   //如果本节点数据对象没有children这个属性，就要加入vue数据监听，要不不会响应式

    if (!this.treeData.children) {

   this.$set(this.treeData, 'children', []);

    }

     //把孩子节点加入孩子属性

      if (this.treeData === this.data) {

          this.treeData.push(newChild);

      } else {

          this.treeData.children.push(newChild);

       }

其中删除节点代码：

this.$refs.tree.remove(this.treeData);

//把该节点加入删除delereTree的头部

this.object.deleteTree.unshift(this.treeData);

其中重命名节点代码：

  this.treeData.time = this.CurentTime();

     this.treeData.editorTime = new Date().getTime();

      his.dialog.dialogRename = false;

其中拖拽代码：

allowDrop(draggingNode, dropNode, type) {

         if (dropNode.data.type) {

           return true;

         } else {

           if (type == 'inner') {

             return false;

          }

          return true;

        }

     },

     allowDrag() { return true; },

以上新建文件夹节点代码跟新建文档代码十分类似，只在对象属性的type设置为false以及在新建前先要完成电子书文档创建的操作。其他的重名名就更改lable的值为用户输入值。删除就去掉tree中该节点，并加入该节点到deletetree数组头部中。以上操作都需要请求后端，把最新的数据发送到后端来做持久化实现。

回收站：

回收站页面如图5-3所示：

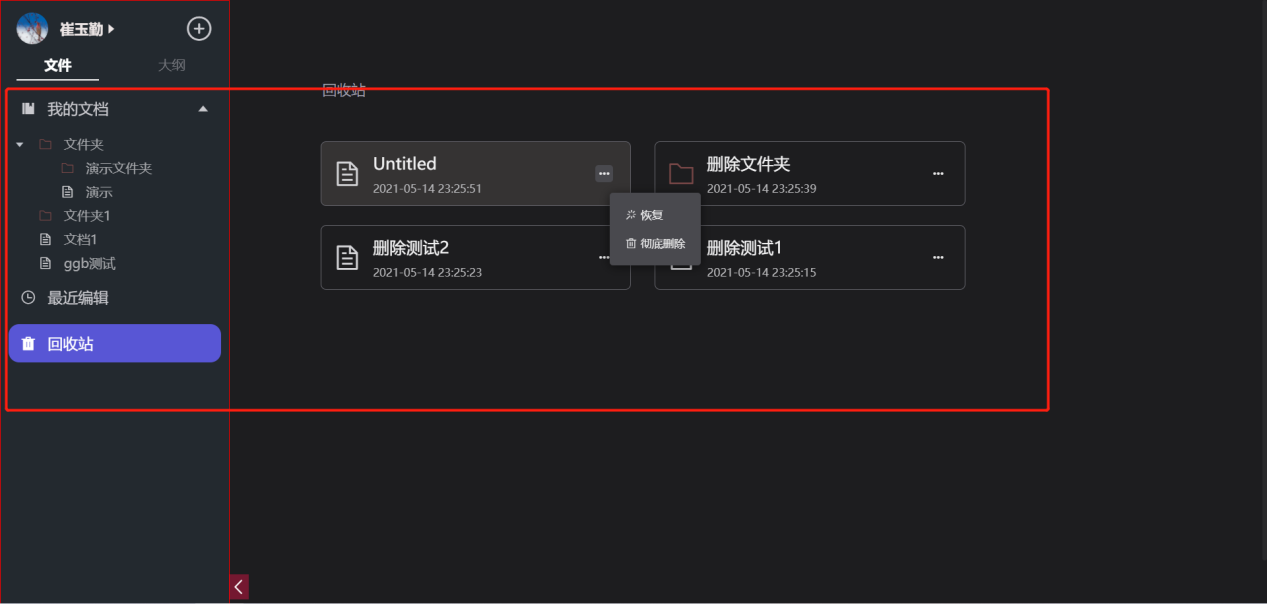


图5-3 回收站页面

点击左侧的回收站按钮，在右侧就会显示出已删除的数据，并且是按时间线的形式以最近到最远来排序的。想要恢复数据，就点击恢复按钮，确定不在需要的数据，可以点击彻底删除来让它消失。

在deleteTree数组删除该节点的代码：

spliceNodeList(state,index) {

  state.nodeList.splice(index,1)

 },

在tree中加入该节点代码：

case '恢复':

   this.data.push(this.treeData);

      this.collapse.collapseTreeIcon = "el-icon-caret-top";

      this.collapse.collapseTree = false;

      break;

以上代码，恢复功能就先在deleteTree数组删除该节点的，接着在tree

中加入该节点代码，而彻底删除功能，只需在deleteTree数组删除该节点的就可以了，以上操作都需要请求后端，把最新的数据发送到后端来做持久化实现。

最近编辑：

最近编辑页面如图5-4所示：

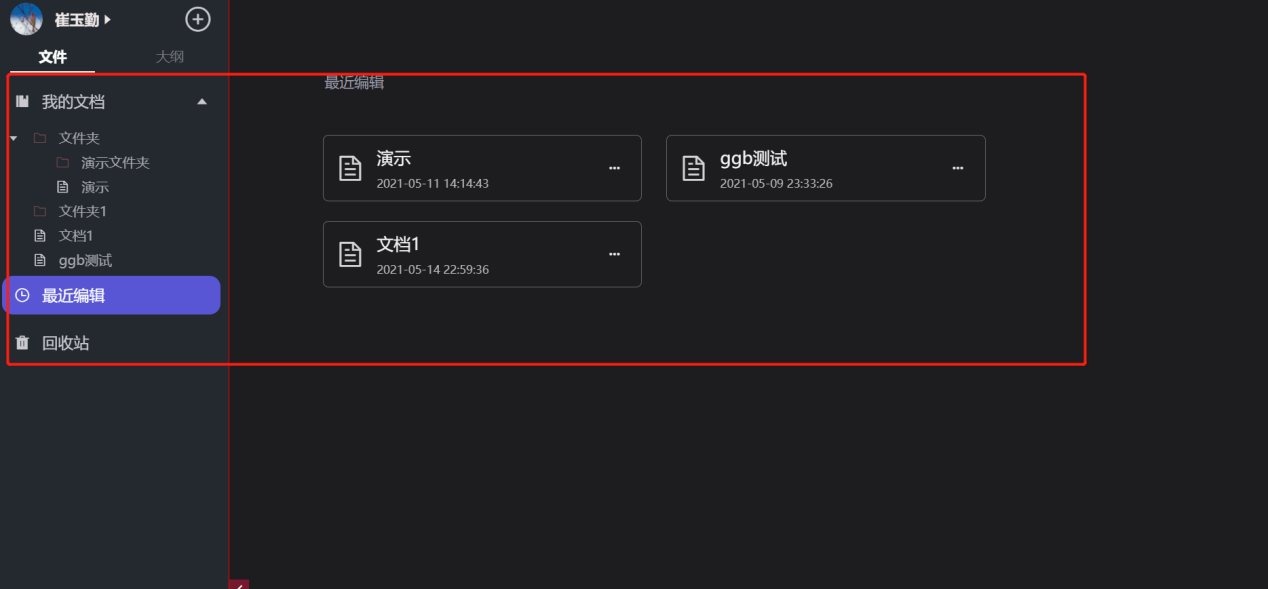


图5-4 最近编辑页面

点击左侧的最近编辑按钮，在右侧就会显示出最近编辑的文档，并且是按时间线的形式以最近到最远来排序的。想要接着编辑某个时刻的文档，就点击该节点，进入编辑页面。

生成最近编辑数组的代码：

    let data = this.$store.state.data;

    let queue = [];

    let res = [];

    queue = queue.concat(data);

    //获取文件夹

    while (queue.length > 0) {

         let node = queue[0];

         queue.shift();

         if (!node.type) {

             res.push(node);

          } else {

               queue = queue.concat(node.children)

               }

      }

      //排序

      if (res.length > 1) {

          res.sort(function (a, b) {

           return b.editorTime - a.editorTime;

        })

      }

      eturn res;

以上代码，先new两个数组queue和res,分别用来接收层级遍历中间缓存数组和返回的最近编辑文档节点数组。接着拼接queue数组和data我的文件树数组，再接着做层级遍历帅选出文档节点，最后按照节点的最近编辑时间戳从大到小排序返回数组，最终返回。以上操作只需在前端就可以实现。

* + 1. 编辑创作模块

编辑创作模块主要实现的是对文档编辑、浏览和保存功能，其中对文档的编辑，使用的是基于markdown语法的编辑创作，除了常见的markdown语义外，还包括如latex/katex格式的数学公式、流程图、时序图、echarts图、geogebra动态几何图等有个性的语义拓展，这就需要添加相对应的markdown-it插件，并对有关插件进行修改添加代码，以符合本系统的需求。对于geogebra动态几何图这种第三方库没有的语义拓展需要自己编写新的解析渲染规则。用户在编辑创作过程中，为了实现实时预览，需要每次修改都重新解析渲染源编辑数据。用户能够定位文档大纲位置，就获取文档解析后的Html数据，找出是标题的DOM节点，计算其距离上层控件的距离，并使编辑和预览区滚动这个距离的长度。系统能将编辑创作数据保存起来，就需在每次修改重新修改后，在一定时间内触发保存函数，将目前最新的数据持久到数据库。

文档编辑、浏览、大纲页面如图5-5所示：



图5-5 文档编辑、浏览、大纲页面

左侧为文档大纲区域，会显示出渲染为<h>的DOM节点文本，用户点击文档大纲，可以同时大约定位编辑和预览区到大纲的位置。中间编辑区是用户输入编辑的文字的区域，还可以在工具栏选择辅助的想要的语义。右侧预览区可以时预览编辑内容最终渲染的效果。以上数据是会在更改后持久化到数据库中。

渲染Geogebra的代码：

  //renser geogebra

  document.querySelectorAll(".ggbid").forEach((element) => {

   try {

      let code = element.textContent;

      let fileName = element.textContent.slice(-7);

      $vm.$emit('ggb', code, fileName);

      element.textContent = "";

     element.outerHTML = `<iframe src="http://127.0.0.1:7000/ggb/` + fileName + `.html" style='width:750px; height:600px; border:0'></iframe>`;

       } catch (e) {

           element.outerHTML = `<pre>geogebra complains: ${e}</pre>`;

        }

     });

在实时渲染预览的代码：

     d\_value: function (val, oldVal) {

        clearTimeout(this.timeout);

        this.timeout = setTimeout(() => {

         this.iRender();

         }, 600);

     },

在计算定位大纲所在编辑预览位置的代码：

// 双栏

if ($vm.s\_preview\_switch) {

// 编辑预览

vNoteEdit.scrollTop = (vShowContent.children[i].offsetTop \*

vNoteEdit.scrollHeight) /

                     vShowContent.scrollHeight;

}

在实时保存编辑数据的代码：

//保存更改

change: function (d\_value, d\_render) {

this.d\_value = d\_value;

this.d\_render = d\_render;

clearTimeout(this.timeout);

this.timeout = setTimeout(() => {

this.object.content = d\_value;

 this.object.contentHtml = d\_render;

axios.post('/api/updateebook', this.object)

     .then(response => {

         console.log("/api/updateebook:", response.data.status);

         this.$emit("setEditorTime")

       }).catch(err => {

         console.log(err);

        })

}, 5000);

以上代码，渲染Geogebra是找到类选择器.ggbid所在的div,获取div里面的Geogebra的base64数据，传到后端，后端根据这个base64数据来生成一个Geogebra动态几何网页，然后替换类选择器.ggbid所在的Html内容为<iframe>标签引入Geogebra动态几何网页。本系统已下载官方的Geogebra资源放置到服务器的ggb文件夹中。实时渲染预览设置了一个600毫米的定时器，减少用户计算机的计算消耗。实时保存编辑数据也设置了一个5秒的定时器，用来减少服务器的消耗。

参考文献

[1] Tenen D, Wythoff G. Sustainable Authorship in Plain Text using Pandoc and Markdown[J]. The Programming Historian, 2018.

[2] Jann B. Creating HTML or Markdown documents from within Stata using webdoc[J]. The Stata Journal, 2017, 17(1): 3-38.

[3] Kalk K, Luik P, Taimalu M. The characteristics of students, blog groups and blogging that predict reflection in blogs during teaching practice and induction year[J]. Teaching and Teacher Education, 2019, 86: 102900.

[4] Jose B, Abraham S. Performance analysis of NoSQL and relational databases with MongoDB and MySQL[J]. Materials Today: Proceedings, 2020, 24: 2036-2043.

[5] 赵轲,易丹丹.Markdown代码生成程序流程图技巧[J].电子制作,2020(08):58-59+65.

[6] 曾广海. 基于Web前端组件化的个人博客系统的设计与实现[D].华中科技大学,2016.

[7] 王辰,刘晓鑫,曹晓燕,王佳楠.基于Vue.js平台的Markdown标记语言插件的研究与实现[J].科技风,2018(35):82+85.

[8] 胡亚明.基于标记语言的论文写作辅助系统[J].广东化工,2017,44(04):81+100.

[9] 吴力荣.基于MathJax的在线公式编辑器的设计与实现[J].廊坊师范学院学报(自然科学版),2014,14(05):40-42.

[10] 张滨.实现Web下公式符号在线操作的可视化编辑器系统的设计[J].计算机科学,2007(06):286-288.

[11] 郭威,刘三(女牙),杨宗凯,孙建文,刘林.基于MathML的在线公式编辑器的实现与应用[J].计算机应用研究,2008(10):3177-3180.

[12] 吴海,彭雪昶.HTML在线编辑器的选择和使用经验探讨[J].信息通信,2019(03):170-171.

[13] 古永红,廖选.在IE浏览器上模拟WORD编辑功能的方法[J].西南师范大学学报(自然科学版),2004(02):320-324.

[14] 于梅英,姜波,张珂.基于Java Web的HTML在线文本编辑器解决方案——以FckEditor在线编辑器为例[J].软件导刊,2011,10(02):36-37.

[15] 王浩.一种基于Markdown语言的静态博客生成程序[J].电脑编程技巧与维护,2019(02):39-51.

[16] 孙黎月. 面向博客管理的全文检索技术研究与实现[D].天津理工大学,2018.

致谢

时间流逝，大学学习生涯这么快就来到了尾声，现在在书写毕业设计时，心中思绪万千。我现在突然发现我是如此的怀念大学这接近四年的时光呀！这一路来，玩得开心的时刻，期末焦急复习备考的时刻，实验室绞尽脑汁思考问题的时刻......就像一段段小电影一样涌现在我的脑海里，这些记录我成长光阴的每一瞬间，我想往后余生都不会忘记。现如今，我作为快要踏入社会走进职场的新人，虽然内心还是有着丝丝的对于未知的忐忑，但更多的还是对未来的向往，毕竟我有家人，同学，朋友和老师的支持，大学期间也学到一点点有用的知识，我相信我可以开始勇敢地起航了。

首先，我要感谢我的母校广州大学对我悉心的培养！我的母校广州大学承载了我整大学生涯，我人生美好的一段旅程之一在这里度过，我在这里有拼搏过，有思考过，有开心过，对于我来说，母校广州大学就是我的人生奖牌。从今往后，我会秉承着“博学笃行，与时俱进”的校训，不断前进，不断拼搏，不断学习。

其次，我要感谢我的指导老师李传中老师对我的指导和培养。他会督促我赶快完成毕设工作，提前发送毕设指导文件给我让我了解，以及我有什么毕设的问题，通过请教会耐心了解并给出建议思路以及经验，还会检查我的各项毕设工作。这种关心促进了我投入了更多精力到毕设中。

接着，我要感谢这一路来陪伴在我周围的可爱的宿舍小伙伴们，是你们给了我友谊。我们共建了一个良好的学习氛围，我们一起玩耍，一起进步，互相推着对方前进。

再有，我要感谢我的父母，无论什么时刻，他们都是给我家的温暖，时时联系关心我的生活学习各方面，给了我一个美好的学习成长空间，让我无后顾之忧，有足够勇气往前冲。

最后，我要感谢我自己，谢谢我这一路来的不颓废，不放弃，能吃苦，热爱生活，永远充满热情充满信心，希望我以后任能做一个追梦的人。

往后余生，祝愿大家一帆风顺，我们都能遇到更好的自己！