

**本科毕业论文(设计)**

题目： 基于python的博客后台系统设计与实现

学院： 数学与计算机科学学院

班级： 2014级计算机本科二班

学号： 140809011223

姓名： 乔伟杰

指导教师： 王莉 职称： 讲师

完成日期： 2018 年 5 月 25 日

**基于python的博客后台系统设计与实现**

**摘 要**：因为互联网的普及，人们可以很方便的记录下自己平时的所思所想。所以我的想法是设计一个简单的工具来提供这样的一个功能。基于这样的一个目的，可这个博客系统就诞生了，它可以在手机或者电脑的浏览器上随时随地的登录，记载下平时的想法、创意或者日志，我们的博客就是这样一个简答的思想载体。

这个系统的设计以Linux为平台，并且以B/S为基础，在Vscode上开发，前台主要开发技术为CSS框架Bootstrap，搭建的服务器为Nginx。利用Python的Flask框架对系统页面进行设计、开发，其设计主要包括用户对信息的浏览、用户编写自己的博客、用户信息的验证、用户的关注功能，主页面可以显示所有人的博客或者自己关注的人的博客。后台用Mysql数据库保存客户信息，方便系统对信息的获取，达到查询的目的。

**关键词：**B/S；Python；数据库；博客

**The Design and Implementation of Blog Back-end System**

**Based on Python**

**Abstract:** Because of the popularity of the Internet, people can easily record their usual thoughts. So my idea is to design a simple tool to provide such a feature. For this purpose, the blog system was born. It can be logged on the mobile phone or computer browser anytime and anywhere to record the usual ideas, ideas or logs. Our blog is such a short-answer thought carrier.

The design of this system is based on Linux, and is based on B/S. It is developed on vscode. The foreground development technology is the CSS framework bootstrap, and the server is Nginx. The Python flask framework is used to design and develop the system pages. The design mainly includes users' browsing of information, users' writing their own blogs, verification of user information, and user's attention. The main page can display everyone's blog or pay attention to it. People's blog. Mysql database is used to save customer information in the background to facilitate the system to obtain information and achieve the purpose of the query.

**Key words**: B/S; Python; Database; Blog

**目 录**

**[1. 绪论 1](#_Toc9635)**

**[1.1 互联网的发展 1](#_Toc19775)**

**[1.2 博客的发展背景 1](#_Toc23990)**

**[2. 系统分析与开发环境 2](#_Toc32034)**

**[2.1 开发环境 2](#_Toc14422)**

**[2.1.1硬件平台 2](#_Toc22143)**

**[2.1.2软件平台 2](#_Toc30759)**

**[2.1.3开发工具介绍 2](#_Toc7286)**

**[2.2 需求分析 3](#_Toc23209)**

**[2.2.1编写博客 3](#_Toc2525)**

**[2.2.2 关注功能 3](#_Toc15690)**

**[2.2.3 粉丝人数 3](#_Toc9366)**

**[2.2.4 博客的再编辑 4](#_Toc23529)**

**[2.2.5 权限管理功能 4](#_Toc22657)**

**[2.2.6 登录、注册和注销 4](#_Toc31858)**

**[2.3 可行性分析 4](#_Toc32544)**

**[2.3.1技术可行性 4](#_Toc14475)**

**[2.3.2经济可行性 4](#_Toc24860)**

**[2.3.3 操作可行性 4](#_Toc23905)**

**[3. 系统设计 5](#_Toc26968)**

**[3.1设计目标 5](#_Toc23314)**

**[3.2 系统功能结构图 5](#_Toc15667)**

**[3.3 系统的总体设计 5](#_Toc18514)**

**[3.4系统的详细设计 6](#_Toc19486)**

**[3.5系统的数据流图 7](#_Toc25651)**

**[3.6 数据路设计 8](#_Toc14523)**

**[3.6.1系统概念模型 8](#_Toc3901)**

**[3.6.2系统数据库表 9](#_Toc20509)**

**[4. 系统实现 11](#_Toc15927)**

**[4.1博客系统的主页面 12](#_Toc10213)**

**[4.2用户信息管理模块设计 13](#_Toc15446)**

**[4.2.1注册模块 13](#_Toc582)**

**[4.2.2用户登录模块 15](#_Toc32674)**

**[4.2.3个人信息编辑模块 17](#_Toc20283)**

**[4.3博客模块 19](#_Toc10710)**

**[4.3.1 博客编辑模块 19](#_Toc7293)**

**[4.3.2博客的再编辑模块： 21](#_Toc24850)**

**[4.4关注模块 23](#_Toc4588)**

**[5. 测试 25](#_Toc18175)**

**[5.1界面测试 25](#_Toc18065)**

**[5.2功能测试 26](#_Toc19049)**

**[5.3未完成的功能 27](#_Toc23140)**

**[6. 小结 27](#_Toc11866)**

**[参考文献 28](#_Toc25458)**

# **基于python的博客后台系统设计与实现**

# 绪论

## 1.1 互联网的发展

最近这几年互联网快速发展，更被人成为第三次技术革命，人工智能、物联网、VR技术都跟随互联网的发展如雨后春笋般快速的发展起来，在“互联网+”这一号召下，传统行业最大限度的与互联网结合，创造出更多的商业和人文价值。互联网在个个方面都发挥着自己价值，我们在生活中，每一个地方都会受到互联网便捷、快速的影响，它与我们的生活相结合，创造了更多具有人文价值的产品。并且互联网在其他的领域不断的延伸，它现在已经延伸到我们生活的方方面面了[1]。它的衍生物已经与我们的生活每一个地方都挂钩了。而且现在进行一波又一波的技术革命都与互联网有关，人工智能、物联网、大数据等，他们全部都是互联网在新的时代、新的历史背景下的发展。它已经成为这个世界中最具有力量、最影响我们生活的一个技术。

**1.2 博客的发展背景**

博客作为新兴事物的发展，完全是互联网的产品，它是互联网在快速发展下必然的产物。他让人们可以接受更多的信息从平常的生活中，慢慢的报纸也变得不流行了，因为它可以说是报纸的完全替代品[2]。另外还拥有报纸所没有的功能。此外，它可以让我们随时随地的记录下的自己的东西，所闻所想啊，见解啊。我们可以跟别人交换见解。所以本质上互联网、博客，它是信息传递的载体。而且会有博客的替代品，可能在一方面替代，在一些方面它是不可替代的。由此可以推出，它的发展平台是很巨大的，它必然会渗透到我们生活的方方面面[3]。在不同的历史情形下，我们必须对博客这个产品进行不同的改造和升级，我们不能局限于刚开始创造博客是提供的一些简略的功能，它所提供的功能必须要与时俱进，但是我们的所有的功能都必须围绕它的核心。我们绝对不能脱离它的本质，它的初始的目地——使人们可以随时记录下自己的所思所想，这是它的本质。所以现在博客有很巨大的发展空间。

# 系统分析与开发环境

## 2.1 开发环境

### 2.1.1硬件平台

（1）CPU：Intel Core i7 3537或更高；

（2）内存：4G以上；

（3）硬件空间：500MB以上。

### 2.1.2软件平台

（1）操作系统：Deepin Linux操作系统；

（2）浏览器：Google chrom；

（3）软件开发工具：vscode、Python、flask、mysql、bootstrap。

### 2.1.3开发工具介绍

（1）Vscode：是微软编写一款代码编辑器。主要的方面就是进行代码的编写。它并不像一款代码IDE，它默认是不提供代码的编译功能的。这个编辑器可以安 装在各个平台，Windows、Linux、macOS都有相对应的安装包。它是由著名设计师设计的，非常的美观。并且在这个编辑器上提供了很多的插件。可以很方便的进行各个语言的编写，所以它不局限于一个语言的编辑器。并且默认有git插 件，可以直接提交自己的代码。

（2）Python：python是一个程序编写语言，但是它又与其他语言有显然不同的特点。它是一个动态解释性语言，它自带非常丰富的数据结构，很适合开发后台代码。

（3）Flask：它是一个由Python语言编写的，用于后端web开发的一个工具。它主要的特点就是非常的快捷和方便。体积非常的小，它提供一套可以直接用于web开发的组件，而其他的web开发工具并没有直接进行集成在这个工具里面。所以它非常的小巧，但是它的人气非常的高，在python web开发的环境里，它的名气是非常的大的。

（4）Mysql：是一个[关系型数据库系统](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E5%9E%8B%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F" \t "/home/deepln/Documents\\x/_blank)。它可以创建很多个数据库，并且每一个数据库里面可以创建多个表，MySQL里的表类似于Excel的表，分为列和行。它是一个非常方便使用和快捷的数据库系统。并且它的数据查询速度非常的快，并且可以在Linux系统上部署，在非常多的网站中都是用的这个数据库。并且它支持事务，可处理比较复杂的业务逻辑。在网站开发中，它是一个非常不错的数据库选择。

（5）Bootstrap：它不同于其他的后台的框架，它有本质的差别。它是一个前端的框架。它的出现非常便捷快速的提升了前台的开发速度，并且它的外观非常的好看，而且它可以根据不同设备、不同的屏幕大小来调整它的页面布局。这是一个很优秀的地方。而且flask框架有一个bootstarp的插件。对我们的开发非常有利。

## 2.2 需求分析

Python的博客系统可以让用户方便的编辑自己的所见所闻，并且快捷的发布到网上，可以实现信息共享，另外可以让用户再可以进一步的去编辑自己所发布的文章。这个系统使用Python实现的，所以在这里Python技术对于系统的实现起着至关重要的作用。在这个系统中提供的功能如图2-1：

博客系统

### 

编编写博客

编关注功能

编粉丝人数

编博客的再编辑

编权限管理

编登录注册注销

图2-1 系统分析图

### 2.2.1编写博客

这个系统必须让用户可以直接编写自己的博客，这是一个博客系统的基本功能。但是用户必须可以有一点markdown文档编写的基础。这个markdown是一个HTML编写博客的基础，它可以让用户写的文章有很强的可读性。至关重要的一点就是这个markdown语法是很简单的，用户只需要进行简单的学习就可以掌握markdown的语法，从而进行编写自己的博客。

### 2.2.2 关注功能

在主页面会显示所有用户的博客，所以在主页面显示的文章会非常的多。有时候一个用户可能会喜欢上某一个或几个用户的文章博客，这个时候这个关注功能就是非常重要的了[4]。它可以让用户可以很方便的进行筛选自己所感兴趣的博客。这个功能极大的节省了用户的时间。并且这个功能必须要实现在一个用户对某一个用户不感兴趣的实现进行取关的功能。这一点也是很重要的。

### 2.2.3 粉丝人数

在用户资料界面也必须可以显示，有多少人关注自己。这个功能是关注功能的衍生。但是这个功能也是必须要实现的。

### 2.2.4 博客的再编辑

有些时候用户可能文章只编辑了一半或者文章后来需要进一步的更改。这个时候博客的再编辑就显得尤为重要了。在这个功能里用户必须可以显示以往的博客文章，点击博客的编辑按钮的时候，显示的页面和上次自己编写的页面是一模一样的。这个功能提供用户很大的便捷性。

### 2.2.5 权限管理功能

有时候某一些用户可能发了一些不适当的文章或者水文。这个时候就需要管理员进行删除后者编辑文章了。这个功能实现了在博客系统中不同的角色的功能，普通注册用户是有默认权限的。默认权限就是可以发表文章和评论，还有就是关注其他人。此外还有管理员，和超级管理员。管理员可以进行其他用户博客的删除和更改操作。而超级管理员则可以进行全站的管理工作[5]。

### 2.2.6 登录、注册和注销

这三个功能是一个博客系统最基本的功能。但是这三个功能也是最容易受到攻击的地方。如果这三个模块的安全校验没有做好的话，会使所有注册这个博客的用户受到信息的流失甚至让服务器受到黑客的攻击。

## 2.3 可行性分析

### 2.3.1技术可行性

本系统是B/S模式的，在前段我们再用HTML，CSS，Jquery技术作为我们的页面编写技术，并且使用bootstrap作为一个整体的前端框架。Bootstrap是一个以简便、简洁为主题的前端框架。并且上手容易，很方便的就可以写出不错的前端页面。在后端我们以Linux操作系统为基础，以Python为编写语言，用Python的flask的框架为博客系统的基本实现框架。这些技术也是简单易学、灵活多变的，所以从技术角度来看是可行的。

### 2.3.2经济可行性

系统的实现不需要复杂昂贵的设备，只需要具备Linux的服务器和熟练Python设计的程序人员，以及前端设计人员就可以实现这个博客系统。所以从经济的角度实现这个系统的设计是可行的。

### 2.3.3 操作可行性

这个博客的设计基础观点就是以便捷以上手为基础。博客的编辑和设计是极其简单的，并且前端页面的可读性很强，所以在操作可行性反面也是可以的。

# 系统设计

## 3.1设计目标

本系统设计目标为让用户可以随时随地的编辑自己的文章，享受便利的互联网功能，将自己的博客发布到网上，实现信息共享。这个系统的主要功能为一下几个方面[6]：

（1）用户可以便捷的登录，注销；

（2）用户可以很方便的编写博客，并且进行发布到网上；

（3）用户可以对自己已经发布的博客进行进一步的编辑；

（4）用户可以关注自己喜欢的其他用户，以及取关功能；

（5）博客系统的权限管理功能。

## 3.2 系统功能结构图

本系统的功能结构图如下：



图3-1 系统结构图

## 3.3 系统的总体设计

这里主要的是后端的设计，后台服务和数据库都属于后台的部分，因为我们采用的是B/S框架，所以在前端的设计也在后端服务器中，前端页面主要有数据展示和请求的作用。所以后端的东西主要有三个部分，前端的设计、逻辑处理部分、数据库部分，所以采用的这个flask的这个MVC的框架，非常贴合这个系统的设计[7]。

博客系统的总体架构框架如下：

用户操作后台请求

前端页面

数据库

后台服务

系统数据数据返回

图3-2 架构图

## 3.4系统的详细设计

博客系统的主要模块有登录模块，信息管理模块，关注模块和博客模块。每个模块的功能表如表3-1：

表3-1 详细设计表

|  |  |  |
| --- | --- | --- |
| 博客系统 | 登录模块 | 1、客户根据已经注册过的用户名和密码进行注册、登录。  2、管理员可以使用自己的用户名登录。 |
| 信息管理模块 | 1、记录用户的个人信息，比如地址、联系方式、性别等。  2、用户可以修改个人的信息。 |
| 关注模块 | 1. 用户可以关注自己喜欢用户的博客。 2. 用户同时可以查看关注自己的人。 |
| 博客模块 | 1. 用户可以编写和发布自己的博客文章。 2. 同时用户可以再对自己上传的博客文章进行再一次的编辑。 |
| 权限管理模块 | 用户分为不同的三个角色，管理员可以管理网站。 |

本系统的设计可以抽象出来的实体：用户、用户信息、博客和评论。各个实体的定义及功能如下：

1. 用户：代表一个用户实体：功能如下：
2. 用户利用用户名和密码在登录页面进行登录；可以编辑自己的信息和编辑发布自己的博客可以对博客进行评论
3. 用户信息：代表一个用户信息实体，功能如下：
4. 用户可以在编辑界面对自己的个人信息进行查看和编辑。任何用户都可以查看其他用户的个人信息。
5. 博客：代表一个博客文章的实体，功能如下：
6. 博客是用户发布的文章。任何用户发布的文章都可以给其他人观看。用户可以在发布以后进一步的编辑自己的博客。
7. 评论：代表一个评论实体，功能如下：在每一个博客都下面任何用户都可以发布自己对这个博客评论。任何用户在打开这个博客的时候都可以查看到这个博客的所有评论信息.

各个实体模型之间关系：

用户与用户信息: 一对一。

用户与博客：一对多。

用户与评论：一对多。

用户与用户的关注关系：多对多。

博客与评论：一对多[8]。

## 3.5系统的数据流图

数据流图用图形展现系统整体逻辑功能、系统内部数据的逻辑流向、逻辑变换过程。本系统的数据流图如图3-3：

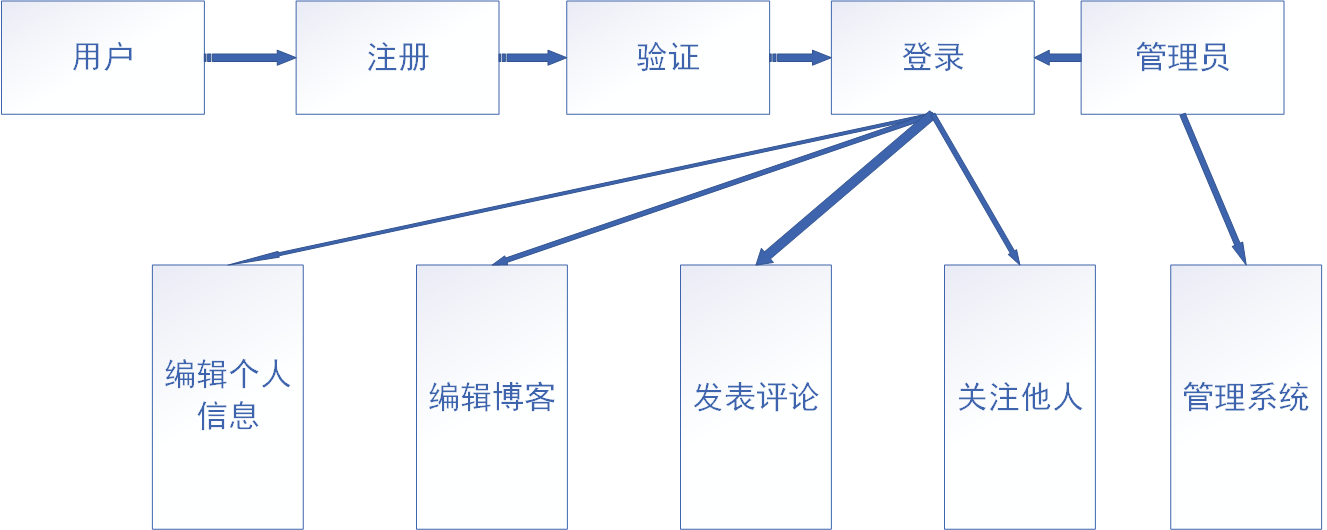


图3-3 系统流程图

## 3.6 数据路设计

### 3.6.1系统概念模型

数据库设计在本系统中占有重要地位，合理的数据设计与安排能使数据存储效率提高，而且数据的统一和完整性能够得到保证。数据库设计一般包括数据库概念结构设计、数据库需求分析、和数据库逻辑结构设计。

关注

管理员

管理

用户

博客

发布

用户

编辑

发表

用户信息

评论

图3-4 实体联系图



图3-5 用户图



图3-6 用户信息



图3-7 博客图



图 3-8 评论图

### 3.6.2系统数据库表

本系统需要用到一个数据库，fblog作为库名，有以下四个主要的表：用户表表、博客表、评论表和权限表。另外，还有一个客房查询表，方便客户查询客房的入住情况。如表3-2、表3-3、表3-4、表3-5、表3-6所示：

表3-2 用户表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 说明 |
| Username | Integer | 用户名 |
| Password | Varchar(60) | 密码 |
| Role\_id | Integer | 权限ID |
| Email | Varchar(100) | 邮箱 |
| Name | Varchar(100) | 昵称 |
| Location | Varchar(500) | 位置 |
| About\_me | Text | 个人介绍 |
| Member\_since | Datetime | 注册时间 |
| Last\_seen | Datetime | 上次登录时间 |

表3-3 权限表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 说明 |
| Name | Char(30) | 全选角色 |
| Default | Boolean | 默认权限 |
| Permissions | Interger | 权限 |

表3-4 关注表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 说明 |
| Follower\_id | Interger | 关注者ID |
| Followed\_id | Interger | 被关注者ID |
| Timestamp | Datetime | 时间 |

表3-5 博客表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 说明 |
| Id | Interger | 博客ID |
| Title | Text | 标题 |
| Body | Text | 内容 |
| Body\_html | Text | 内容的HTML格式 |
| Timestamp | Datetime | 发布时间 |
| Author\_id | Interger | 作者ID |

表3-6评论表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 说明 |
| Id | Interger | 评论ID |
| Body | Text | 内容 |
| Body\_html | Text | 内容的HTML格式 |
| Timestamp | Timedate | 时间 |
| Post\_id | Interger | 博客ID |
| Author\_id | Interger | 作者ID |

# 

# 系统实现

## 4.1博客系统的主页面

主页面是博客系统中信息量最大的一个页面，对用户没有限制，只要点击查询系统的页面，就会自动跳转到该页面，方面用户查看信息。系统效果如图4-1：

图4-1 主页面图

此页面后端中的主要代码（1）：

@main.route('/', methods=['GET','POST'])

def index():

show\_followed = False if current\_user.is\_authenticated:

show\_followed = bool(request.cookies.get('show\_followed'))

if show\_followed:

query = current\_user.followed\_posts

else:query = Post.query

page = request.args.get('page',1, type=int)

pagination = query.order\_by(Post.timestamp.desc()).paginate(

page, per\_page=10,error\_out=False)[9]

posts = pagination.items

return render\_template('index.html', show\_followed=show\_followed,

posts=posts, pagination=pagination

此页面前端中的主要代码：

{% extends 'base.html' %} {% import 'bootstrap/wtf.html' as wtf %} {% import 'include/\_macros.html' as macros %} {% block title %}首页{% endblock %}

{% block content %}{{ super() }}<div class="container">

<div class="page-header"><h1>

{% if current\_user.is\_authenticated %}

欢迎回来, {{ current\_user.username }}{% else %}

文章列表{% endif %}

</h1></div></div><div class="container"><div class="row">

<div class="col-md-12 well bs-component">

<div class="post-tabs">

<ul class="nav nav-tabs">

<li {% if not show\_followed %}class="active" {% endif %}>

<a href="{{ url\_for('main.show\_all') }}">所有文章</a>

</li>{% if current\_user.is\_authenticated %}

<li {% if show\_followed %}class="active" {% endif %}>

<a href="{{ url\_for('main.show\_followed') }}">我的关注</a>

</li>{% endif %}</ul></div>

{% include 'include/\_post.html' %}

<nav class="text-center">

<ul class="pagination">

{{ macros.pagination\_widget(pagination, 'main.index') }}

</ul></nav></div></div></div>

{% endblock %}

{% block scripts %}

{{ super() }}

{{ pagedown.include\_pagedown() }}

{% endblock %}

## 4.2用户信息管理模块设计

### 4.2.1注册模块

用户要在该系统中进行博客编辑工作时，需要首先进行登录，方便网站统一进行管理博客信息，下面是关于博客注册模块的设计，它的主页面效果如图4-2：

图4-2 注册页面

此页面的后端代码为：

@auth.route('/register', methods=['GET', 'POST'])

def register():

from app.auth.forms import RegisterForm

form = RegisterForm()

if form.validate\_on\_submit():

user = User(username=form.username.data,

password=form.password.data, email=form.email.data)

db.session.add(user)

db.session.commit()

User.add\_self\_follows()

return redirect(url\_for('auth.login'))

else:return render\_template('register.html', title=u'注册', form=form)

此页面的前端代码为：

{% extends 'base.html' %}

{% import 'bootstrap/wtf.html' as wtf %}

{% block head %}

{{ super() }}<style type="text/css">

#registerlength {width: 35%;

}</style>{% endblock %}

{% block content %}

<div class="page-header">

<div class="container">

<h1>{{ title }}</h1>

</div></div><div class="container">

{% for message in get\_flashed\_messages() %}

<div class="alert alert-warning">

<button type="button" class="close" data-dismiss="alert">&times;</button>

{{ message }}</div>{% endfor %}

<form method="post">

{{ wtf.quick\_form(form) }}

</form></div>{% endblock %}

### 4.2.2用户登录模块

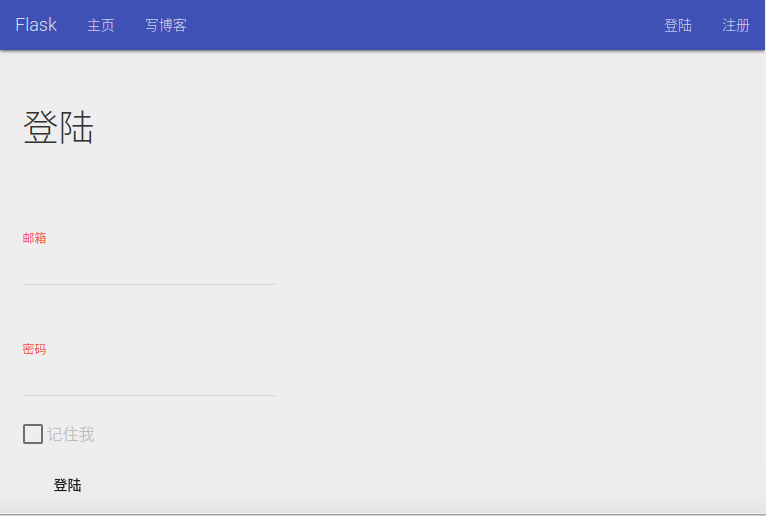
用户想要进行编辑自己的博客，或者关注他人或者查看自己的信息， 编辑自己的信息等等其他一系列操作，效果如图4-3：

图4-3 登录页面图

此页面的后端代码为：

@auth.route('/login', methods=['GET', 'POST'])

def login():

from app.auth.forms import LoginForm

form = LoginForm()

print(form.email.data)

print("sdfsdf")

if form.validate\_on\_submit():

print("sdfsdwerwerf")

user = User.query.filter\_by(email=form.email.data).first()

if user is not None and user.verify\_password(form.password.data):

login\_user(user, form.remember\_me.data)[5]

return redirect(url\_for('main.index'))

flash(u'帐号或者密码错误')

return render\_template('login.html', title=u'登陆', form=form)

此页面的前端代码为：

{% extends 'base.html' %}

{% import 'bootstrap/wtf.html' as wtf %}

{% block head %}

{{ super() }}<style type="text/css">

#loginlength {width: 35%;}</style>

{% endblock %}{% block content %}

<div class="page-header"><div class="container">

<h1>{{ title }}</h1>

</div></div><div class="container">

{% for message in get\_flashed\_messages() %}

<div class="alert alert-warning">

<button type="button" class="close" data-dismiss="alert">&times;</button>

{{ message }}</div>{% endfor %}

<form method="post">

{{ wtf.quick\_form(form) }}</form></div>

{% endblock %}

### 4.2.3个人信息编辑模块

在用户登录或者注册以后，用户可以在自己的信息界面进行编辑关于自己的个人信息，关于个人信息模块，效果如图4-4：

图4-4 编辑个人信息图

此页面的主要后端代码为：

@main.route('/edit-profile',methods=['GET','POST'])

@login\_required

def edit\_profile():

form = EditProfileForm()

if form.validate\_on\_submit():

current\_user.name = form.name.data

current\_user.location = form.location.data[10]

current\_user.about\_me = form.about\_me.data

flash(u'你的个人信息已经被更改')

db.session.add(current\_user)

db.session.commit()

returnredirect(url\_for('main.user', username=current\_user.username))

form.name.data = current\_user.name

form.location.data = current\_user.location

form.about\_me.data = current\_user.about\_me

return render\_template('edit\_profile.html',form=form)

@main.route('/edit-profile/<int:id>',methods=['GET','POST'])

@login\_required

@admin\_required

def edit\_profile\_admin(id):

user = User.query.get\_or\_404(id)

form = EditProfileAdministratorForm(user=user)

if form.validate\_on\_submit():

user.email = form.email.data

user.username = form.username.data

user.confirmed = form.confirmed.data

user.itsrole = Role.query.get(form.role.data)

user.name = form.name.data

user.location = form.location.data

user.about\_me = form.about\_me.data

db.session.add(user)

db.session.commit()

flash(u'该用户的信息已经更新了')

return redirect(url\_for('main.user',username=user.username))

form.email.data = user.email

form.username.data = user.username

form.confirmed.data = user.confirmed

form.role.data = user.role\_id # role\_id 与 itsrole是关联的

form.name.data = user.name

form.location.data = user.location

form.about\_me.data = user.about\_me

return render\_template('edit\_profile.html',form=form,user=user)[6]

此页面的主要前端代码为：

{% extends 'base.html' %}

{% import 'bootstrap/wtf.html' as wtf %}

{% block title %}编辑个人信息{% endblock %}

{% block content %}

<div class="page-header">

<h1>编辑您的信息</h1></div>

<div class="container">

<form method="post">

{{ wtf.quick\_form(form) }}

</form></div>{% endblock %}

## 4.3博客模块

### 4.3.1 博客编辑模块

此系统是一个博客系统所以最主要的功能就是在用户登录后可以编辑自己的博客，这个系统中编写博客都是用markdown语法进行编辑的，分为编辑区和预览区，此页面的效果如图4-5：

 图4-5 博客编辑

此页面的主要后端代码如下：

@main.route('/blog',methods=['GET','POST'])

@login\_required

def blog():

form = PostForm()

if current\_user.can(Permission.WRITE\_ARTICLES) and \

form.validate\_on\_submit():

post = Post(title=form.title.data,body=form.body.data,

author=current\_user.\_get\_current\_object())

db.session.add(post)

db.session.commit()

return redirect(url\_for('main.index'))

return render\_template('blog.html',form=form)

此页面的前端代码如下：

{% extends 'base.html' %}

{% import 'bootstrap/wtf.html' as wtf %}

{% block content %}

{{ super() }}

<div class="container">

<div class="row well">

<form method="post" class="form " role="form">

{{ form.hidden\_tag() }}

<div class="form-group">

{{ form.title.label }}

{{ form.title(id="title", class="form-control" )}}</div>

<div class="form-group">

{{ form.body.label }}

<div class="row">

<div class="col-md-6 " >

<div class="panel panel-default", style="height: 600px;">

<div class="panel-heading", style="height: 50px;">编辑区</div>

<div class="panel-body", style="height: 550px;">

{{form.body(only\_input=True,style="height: 100%;width:100%;") }}

</div></div></div>

<div class="col-md-6" >

<div class="panel panel-default", style="height: 600px;">

<div class="panel-heading", style="height: 50px;">预览区</div>

<div class="panel-body", style="height: 550px;">

{{ form.body(only\_preview=True) }}

</div></div></div></div></div>

<div class="form-group pull-right">

{{ form.submit(class="btn btn-raised") }}

</div></form></div></div><style>

#flask-pagedown-body{

border-style: none;

resize: none;

}

.flask-pagedown-preview{

overflow-x:hidden;

overflow-y: auto;

height: 100%;}

.flask-pagedown{

height: 100%;}

.panel-body img{ max-width: 100%; display: block; }

</style>

{% endblock %}

{% block scripts %}

{{ super() }}

<script src="https://cdnjs.cloudflare.com/ajax/libs/showdown/1.6.4/showdmi

n.js"> < /script><script type="text/javascript">

var converter = new showdown.Converter();

function flask\_pagedown\_converter(text){

return converter.makeHtml(text);}</script>

{% endblock %}

### 4.3.2博客的再编辑模块：

在用户已经提交了自己的博客以后，还可以在以后对自己的博客在原来的基础之上进行编辑，这个页面的效果如图4-6：

图4-6 博客的再编辑

此页面的后端代码为：

@main.route('/edit/<int:id>', methods=['GET','POST'])

@login\_required

def edit(id):

post = Post.query.get\_or\_404(id)

if current\_user != post.author and \

not current\_user.can(Permission.ADMINISTRATOR):

abort(403)

form = PostForm()

if form.validate\_on\_submit():

post.title = form.title.data

post.body = form.body.data

db.session.add(post)

db.session.commit()

return redirect(url\_for('main.post',id=post.id))

form = PostForm()

form.title.data = post.title

form.body.data = post.body

return render\_template('edit\_post.html', form=form)

此页面的前端代码为：

{% extends 'md\_editor.html' %}

{% block title %}编辑博客{% endblock %}

{% block content %}

<div class="container">

<div class="page-header ">

<h1>编辑博客</h1>

</div>

</div>

{{ super() }}

{%endblock%}

## 4.4关注模块

用户可以关注自己喜欢的用户，并且在主界面可以进行筛选自己只喜欢的用户的博客，而且可以在这个用户不喜欢之后进行取消关注，另外可以查看自己的关注的所有用户或者是查看自己的粉丝数目，此页面的效果如图4-7：

图4-7 个人介绍图

此页面的后端代码为：

@main.route('/follow/<username>')

def follow(username):

u = User.query.filter\_by(username=username).first()

if u is None:

flash(u'没有该用户')

return redirect(url\_for('main.index'))

if current\_user.is\_following(u):

flash(u'已经关注了该用户')

return redirect(url\_for('main.user', username=username))

current\_user.follow(u)

flash(u'关注了 %s' % username)

return redirect(url\_for('main.user', username=username))

@main.route('/unfollow/<username>')

def unfollow(username):

u = User.query.filter\_by(username=username).first()

if u is None:

flash(u'没有该用户')

return redirect(url\_for('main.index'))

if u.is\_followed\_by(current\_user):

current\_user.unfollow(u)

flash(u'取消对 %s 的关注' % username)

return redirect(url\_for('main.user', username=username))

@main.route('/followers/<username>')

def followers(username):

u = User.query.filter\_by(username=username).first()

page = request.args.get('page', 1, type=int)

pagination = u.followers.paginate(page, per\_page=10, error\_out=False)

follows = [{'user':item.follower,'timestamp':item.timestamp}

for item in pagination.items] # 转换成一个字典列表

return render\_template('followers.html', title=u'的粉丝',user=u, pagination=pagination, follows=follows)

# 列出所有我的关注

@main.route('/followed-by/<username>')

def followed\_by(username):

u = User.query.filter\_by(username=username).first()

page = request.args.get('page', 1, type=int)

pagination = u.followed.paginate(page, per\_page=10, error\_out=False)

follows = [{'user':item.followed,'timestamp':item.timestamp}

for item in pagination.items] # 转换成一个字典列表

return render\_template('followers.html', title=u'的关注',user=u, pagination=pagination, follows=follows)

此页面的前端代码为：

{% extends 'base.html' %}

{% import 'include/\_macros.html' as macros %}

{% block title %}

{{ user.username }}{{ title }}

{% endblock %}

{% block content %}

<div class="page-header">

<h2>{{ user.username }}{{ title }}</h2>

</div>

<table class="table table-hover followers">

<thead><tr><th>用户名</th><th>时间</th></tr></thead>

{% for follow in follows %}

{% if follow.user != user %}<tr><td><a href="{{ url\_for('main.user',username=follow.user.username) }}">

<img class="img-rounded" src="{{ follow.user.gravatar(size=32) }}">

{{ follow.user.username }}</a></td><td>

{{ moment(follow.timestamp).format('L') }}</td></tr>

{% endif %}

{% endfor %}</table><div class="pagination">

{{macros.pagination\_widget(pagination,'main.followers',username= user.userna me) }}</div>{% endblock %}

# 测试

在系统开发时期过程中，比较复杂繁琐就是软件测试，测试作为系统正常运行的保障。测试的目的是发现系统中存在的错误，使系统中的一些错误或者漏洞在计算机上执行的过程中显露出来。

## 5.1界面测试

（1）全部界面的设计和安排是否合理，样式设计是否看好，是否满足大多数用户的要求。

（2）在用户发布新的博客是，在网站的主页面是否会更新用户的博客信息。

（3）在注册时候，两次输入的密码不相同的时候是否会提示错误信息。

（4）登录的时候，如果用户输入不正确，网站后台时候会有提示信息。

表5-1 测试流程图

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| NO. | 测试对象 | 测试步骤 | | 预期结果 | | 测试结果 | |
| 主界面 | | | | | | | | |
| 1 | 主页面 | 开始调试执行 | | 跳转至主页 | | 和预想一致 | |
| 登录模块 | | | | | | | | |
| 2 | 登录页面 | 点击登录按钮 | | | 跳个人信息界面 | | 正常跳转 |
| 3 | 登录功能 | 输入正确的用户名和密码 | | | 能登录 | | 未正常登录 |
| 4 | 登录功能 | 输入不存在用户名密码 | | | 提示该用户不存在 | | 用户不存在 |
| 注册模块 | | | | | | | | |
| 5 | 注册界面 | | 单击注册按钮 | | 跳转注册界面 | | 和预想一致 |
| 6 | 注册功能 | | 输入还没有注册过用户名、密码 | | 提示注册成功 | | 提示注册错误 |
| 7 | 注册功能 | | 输入密码不是字母和数字的组合 | | 提示密码不符合 | | 密码符合，未达到预期效果 |
| 8 | 注册功能 | | 输入已注册过的用户名 | | 提示用户名已存在，直接登录 | | 未提示用户存在 |
| 用户个人资料模块 | | | | | | | | |
| 9 | 用户资料界面 | | 运行该模块 | | 资料列表 | | 和预想一致 |
| 博客编辑 | | | | | | | | |
| 10 | 博客编辑界面 | | 点击博客编辑 | | 编辑后，可以提交 | | 完成 |
| 博客信息查询模块 | | | | | | | | |
| 11 | 查询主页面 | | 点击标题 | | 进入主页面模块 | | 未进入主页面 |

测试结束后，经调试系统正常运行。

## 5.2功能测试

（1）客户在输入正确的用户名、密码后能否跳转到登录系统。若忘记密码时，可根据相应提示找回密码。

（2）进入博客管理系统主界面后，当用户点击要进行操作的功能模块后能否成功链接到相应的页面。

（3）用户权限不同，可以进行不同的操作，若以客户的身份进入系统，即可查询自己的一些预定信息；若以管理员的身份进入系统，即可对用户的注册信息以及客房预定信息进行增、删、改、查等操作。

## 5.3未完成的功能

这个系统可以实现用户注册、登录、编辑博客、关注等等，但是在用户量大了以后还没有一个功能可以显示本站的热门博客，这个就没有办法把优秀的博客选出来，实现的是最基本的博客设计。还需要学习更多的网站设计知识才可以更好的实现这个功能。

# 小结

在这个系统的开发过程中，我学习到了很多的知识点。在前端知识中，学习到了HTML，CSS，并且也很好的学习了CSS的一个框架Bootstrap，这对我提升前端的知识有很大的作用。但是这个系统的主要方面还是在后端，所以在后端中涉及的知识面更多、更广。在后端编写中用到了Python语言，以及它的Flask网站开发框架，还有flask里面的许多的小插件。还有Mysql的设计，数据的读取。在这个网站的设计中，很大的提升我的计算机能力。

# 参考文献

[1] 王珊. 数据库系统概论[M]. 北京：高等教育出版社，2007：30-36.

[2] 杨明华，谭励，于重重. LAMP 网站开发黄金组合:Linux+Apache+MySQL+PHP[M]. 北京:电子工业出版社，2008：102-116.

[3] 陈晖. [中小型物流企业车辆管理信息系统的设计与实现](http://kreader.cnki.net/Kreader/CatalogViewPage.aspx?tablename=CMFD2012&filename=1012295412.nh&dbCode=cdmd" \o "中小型物流企业车辆管理信息系统的设计与实现" \t "/home/deepln/Documents\\x/_self)[D]. 四川：电子科技大学，2011.

[4] 魏亚楠，宋义秋. oracle数据库应用系统的性能优化[J]. 唐山职业技术学院，2009(1)：40-66.

[5] 朱丽君. [基于mahout的高校图书馆个性化图书推荐系统设计与实现](http://kns.cnki.net/kcms/detail/detail.aspx?filename=1018055507.nh&dbcode=CMFD&dbname=CMFDTEMP&v=" \t "/home/deepln/Documents\\x/kcmstarget)[D]. 江西：南昌大学， 2018.

[6] 陈香. [浅谈高职院校Web在线考试系统的应用](http://kns.cnki.net/kcms/detail/detail.aspx?filename=XXXT201611099&dbcode=CJFD&dbname=CJFD2016&v=" \t "/home/deepln/Documents\\x/kcmstarget)[J]. 信息系统工程，2016(11)：54-70.

[7] 武新芳. [基于PHP+Mysql的图书馆公告系统前台开发](http://kns.cnki.net/kcms/detail/detail.aspx?filename=KJSJ201603153&dbcode=CJFQ&dbname=CJFD2016&v=" \t "/home/deepln/Documents\\x/kcmstarget)[J]. 科技视界， 2016(03)：13-25.

[8] 魏斌. [高性能MySQL集群部署](http://kns.cnki.net/kcms/detail/detail.aspx?filename=HNKJ201414004&dbcode=CJFQ&dbname=CJFD2014&v=" \t "/home/deepln/Documents\\x/kcmstarget)[J]. 河南科技，2014(14)：50-64.

[9] Jannach D， Zanker M， Felfernig A，et al. Recommender Systems: An Introduction[J]. Int.J.Hum Comput Interaction，2010，46(2)：55-90.

[10] Wiwit Suryanto，Theodosius Marwan Irnaka. [Web-based application for inverting one-dimensional magnetotelluric data using Python](http://kns.cnki.net/kcms/detail/detail.aspx?dbname=SJES_U&filename=SJES1DF4514F72804BB37798CDDC7761578C&dbcode=WWJD&v=" \t "/home/deepln/Documents\\x/kcmstarget)[J].  Computers and Geosciences，2015(2)：20-31.