《Python程序设计》

实验作品

作品题目： Django博客系统

院 系： 计算机与信息科学学院

专 业： 软件工程

学 号： 021321752215

姓 名： 舒洪凡

二零二一年十二月

**实验作品评分参考标准表**

|  |  |
| --- | --- |
| **评分标准及分值** | **得分** |
| **选题符合课程内容范围要求（10分）** |  |
| **设计思路和内容具有独特性（10分）** |  |
| **作品设计结构合理、功能完整（40分）** |  |
| **具有一定的实用研究价值（15分）** |  |
| **内容上无明显抄袭现象（15分）** |  |
| **图表格式正确，书写格式规范（10分）** |  |
| **合 计（100分）** |  |

**Django博客系统**

**摘要：**本作品主要实现了一个Django博客系统，采用前后端不分离的项目结构。使用 Django，只要很少的代码，Python 的程序开发人员就可以轻松地完成一个正式网站所需要的大部分内容，并进一步开发出全功能的 Web 服务 Django 本身基于 MVC 模型，即 Model（模型）+ View（视图）+ Controller（控制器）设计模式，MVC 模式使后续对程序的修改和扩展简化，并且使程序某一部分的重复利用成为可能。Python 加 Django 是快速开发、设计、部署网站的最佳组合。前端采用了JavaScript、BootStrap、Vuejs，axios，后端采用Python、Django（开放源码的重Web应用框架），数据存储采用MySQL数据库、Redis、Session技术，系统能够实现用户注册、用户登录退出、用户个人中心、用户修改个人信息、用户重置密码、用户写博客、博客的分页展示、每篇博客的博客详情、用户发表对博客的评论、每篇博客的评论分页展示、用户删除自己的博客，以及用户的权限管理。

**目 录**

[第一章 作品概述 1](#_Toc90541759)

[第二章 作品设计 2](#_Toc90541760)

[2.1作品的构思 2](#_Toc90541761)

[2.1.1 起源 2](#_Toc90541762)

[2.1.2 总体架构 2](#_Toc90541763)

[2.2作品使用的第三方库 3](#_Toc90541764)

[第三章 作品的实现 4](#_Toc90541765)

[2.1作品框架 4](#_Toc90541766)

[2.1.1 流程图 4](#_Toc90541767)

[2.2作品实现要点 15](#_Toc90541768)

[2.2.1 技术 15](#_Toc90541769)

[2.2.2 核心代码 15](#_Toc90541770)

[第四章 总结 32](#_Toc90541771)

[参考文献 33](#_Toc90541772)

# 第一章 作品概述

本作品为一个Django博客系统，开发采用Pycharm编辑器，采用Vue作为前端框架，采用Django作为后端框架，采用Django模板引擎，采用Session和Cookie技术，采用MySQL数据库，采用Redis对验证码进行存储。

本博客系统能够实现：

1. 用户注册
2. 用户登录退出
3. 记住我
4. 用户个人中心
5. 用户修改个人信息
6. 用户重置密码
7. 用户写博客
8. 博客的分页展示
9. 每篇博客的博客详情
10. 用户发表对博客的评论
11. 每篇博客的评论分页展示
12. 用户删除自己的博客，以及用户的权限管理等功能。

# 第二章 作品设计

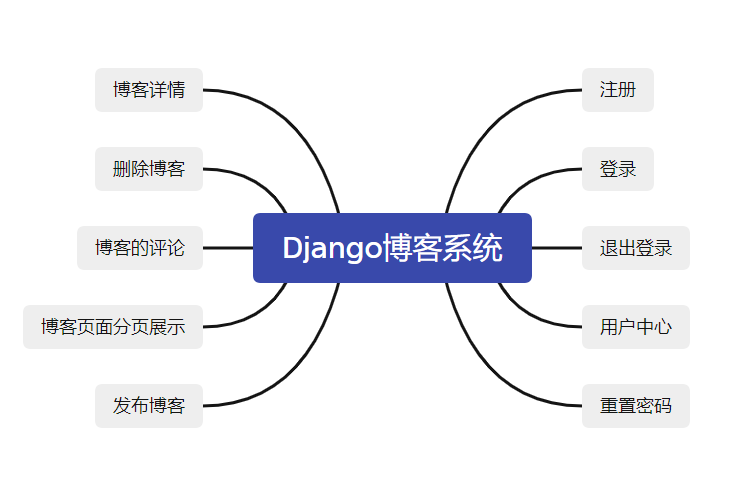
## 2.1作品的构思

### 2.1.1 起源

本作品对当下最新的技术进行分享。博客是现在网络上主要的网络交流方式，是个人的‘读者文摘’，一个典型的博客结合了文字、图像、其他博客或网站的链接及其它与主题相关的媒体，能够让读者以互动的方式留下意见，并且进行讨论。

### 2.1.2 总体架构

本作品主要模块如图1.1所示



**图1.1 程序功能**

1. 注册（图形验证码验证）
2. 登录（状态保持，Cookie，Session）
3. 个人中心（图片上传，更新数据）
4. 发布博客（数据入库）
5. 博客授课（数据分页）
6. 博客详情（博客详情数据展示，评论功能）

项目的开发模式前后端不分离

## 2.2作品使用的第三方库

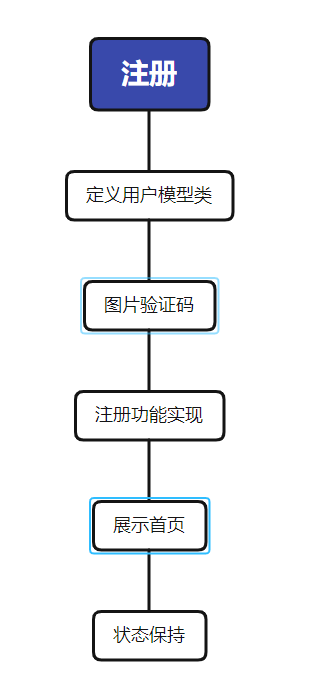
本作品使用的第三方库包括：

1. Captcha.py
2. Django
3. Axios
4. Bootstrap
5. Vue.js
6. jquery

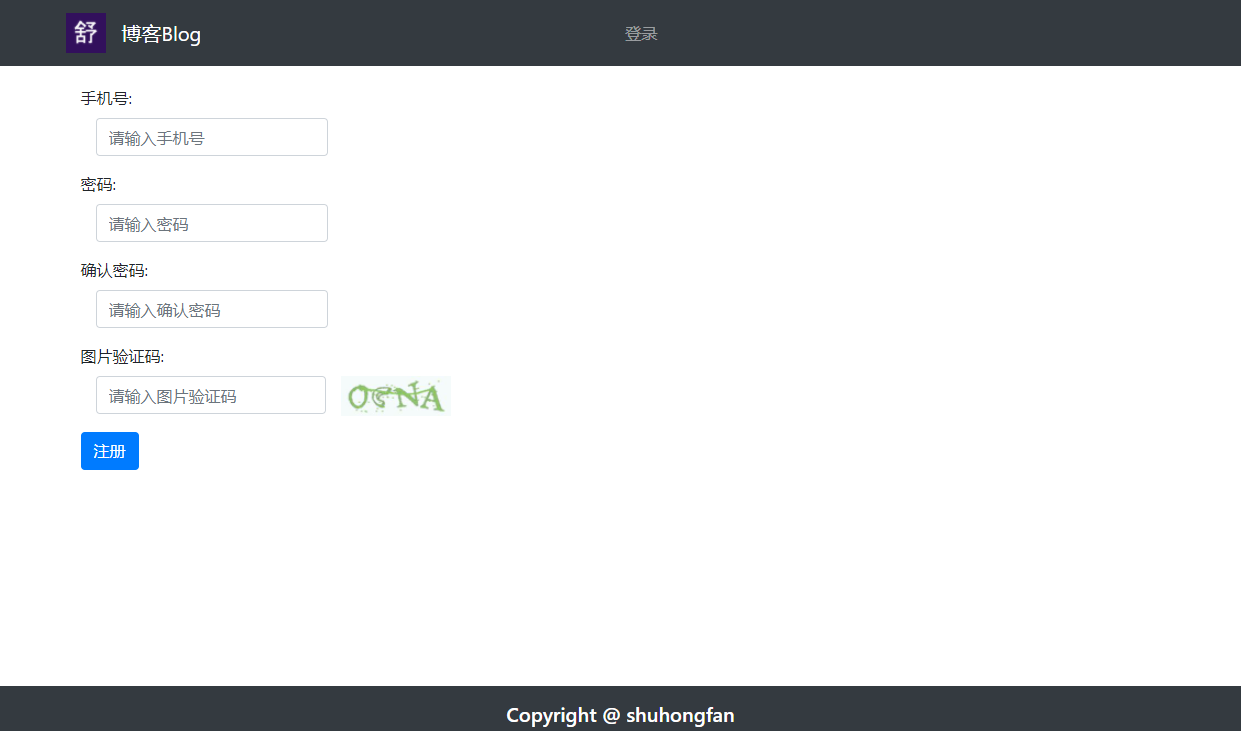
# 第三章 作品的实现

## 2.1作品框架

### 2.1.1 流程图



**图2.1 注册功能**



**图2.2 注册页面**

注册功能如图2.1所示。注册页面如图2.2所示。注册功能的界面展示的流程如下：

* 创建用户模块应用
* 注册用户模块应用
* 准备模板目录并设置模板路径
* 定义用户注册视图
* 定义用户注册路由
* 修改静态文件加载方式

定义用户模型类的流程如下：

* Django默认用户认证系统
* Django默认用户模型类
* 自定义用户模型类
* 指定用户模型类
* 迁移用户模型类

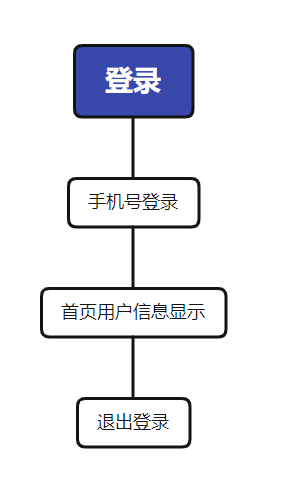
图形验证码接口设计和定义的流程如下：

* 从GitHub下载captcha包（该包用于生成图形验证码）
* 图形验证码后端接口设计
* 图形验证码后端实现
* 修改模板中图片验证码HTML代码

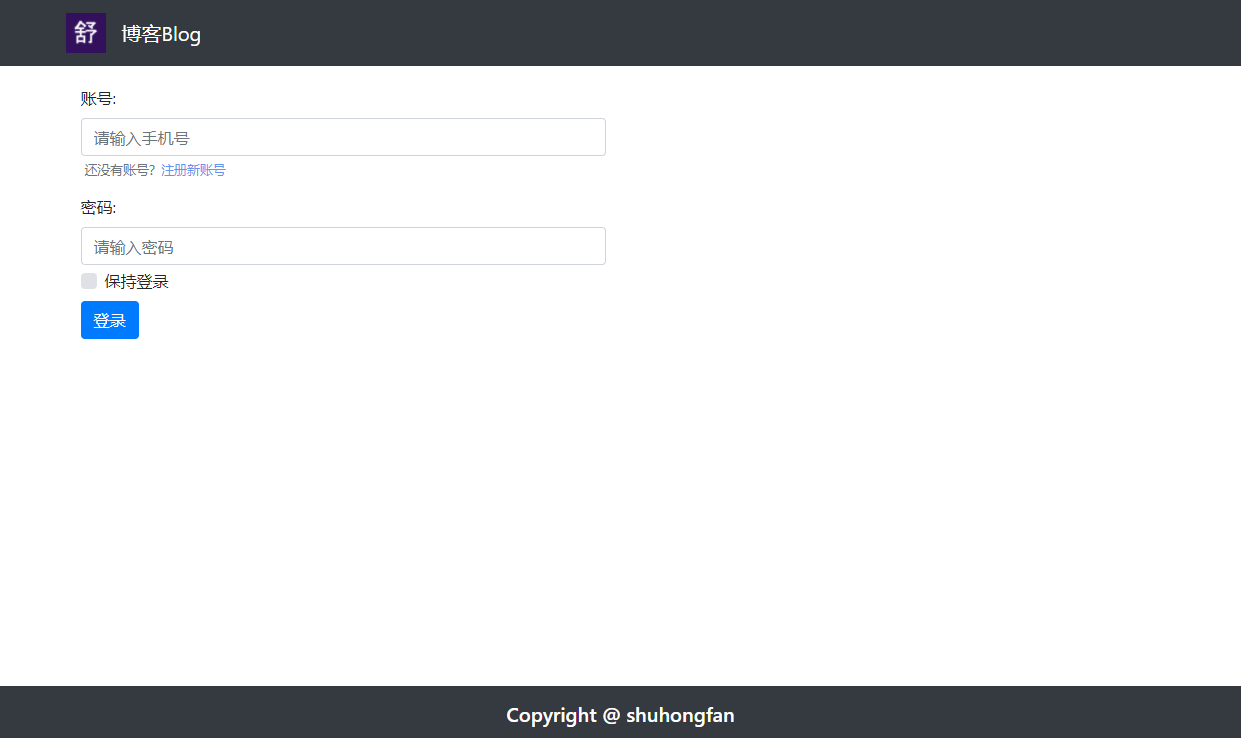
用户注册实现的流程如下：

* 用户注册接口设计
* 用户注册接口实现
* 解决页面提交时报403csrf问题

状态保持将通过认证的用户的唯一标识信息（比如：用户ID）写入到当前session会话中。login()方法：Django用户认证系统提供了login()方法封装了写入session的操作，帮助我们快速实现状态保持login(request, user)



**图2.3 登录功能**



**图2.4 登录页面**

登录功能如图2.3所示。登录页面由图2.4所示。手机号登录的主要流程如下：

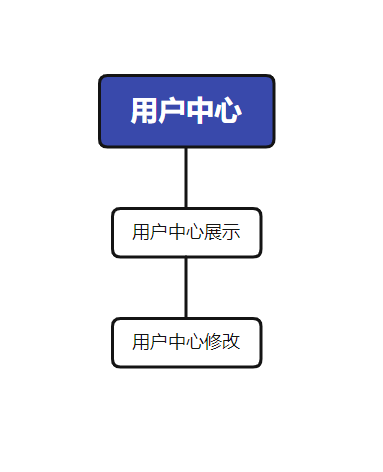
* 登录页面展示
* 登录接口设计
* 登录接口实现

首页用户名展示的主要流程如下：

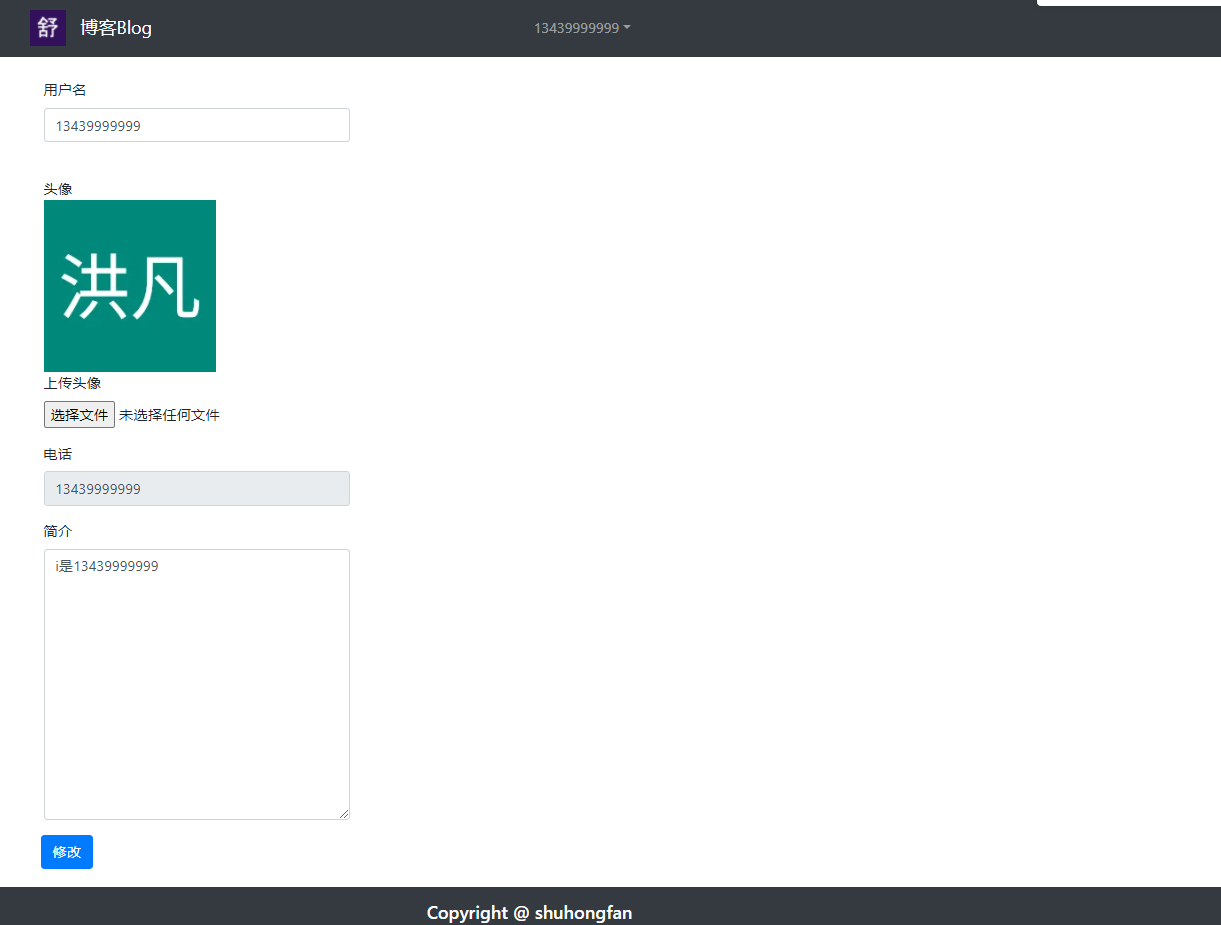
* 用户名写入到cookie
* Vue渲染首页用户名

退出登录的主要流程如下：

* 退出登录：正好和登录相反（清理session会话信息）
* logout()方法：Django用户认证系统提供了logout()方法，封装了清理session的操作，帮助我们快速实现登出一个用户



**图2.5 用户中心功能**



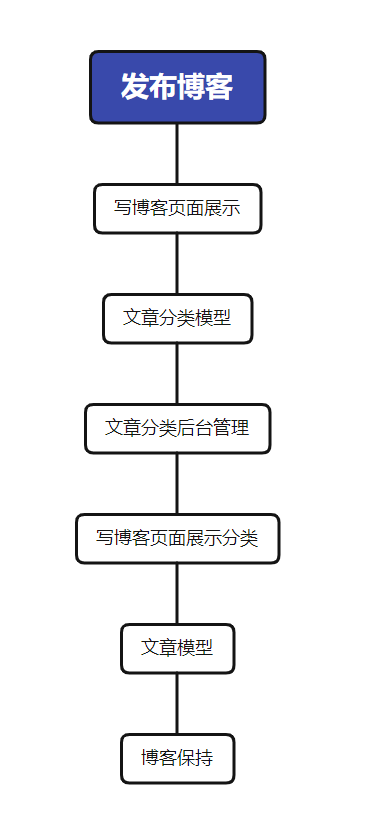
**图2.6 用户中心页面**

用户中心如图2.5所示。用户中心页面由图2.6所示。用户中心展示的主要流程如下：

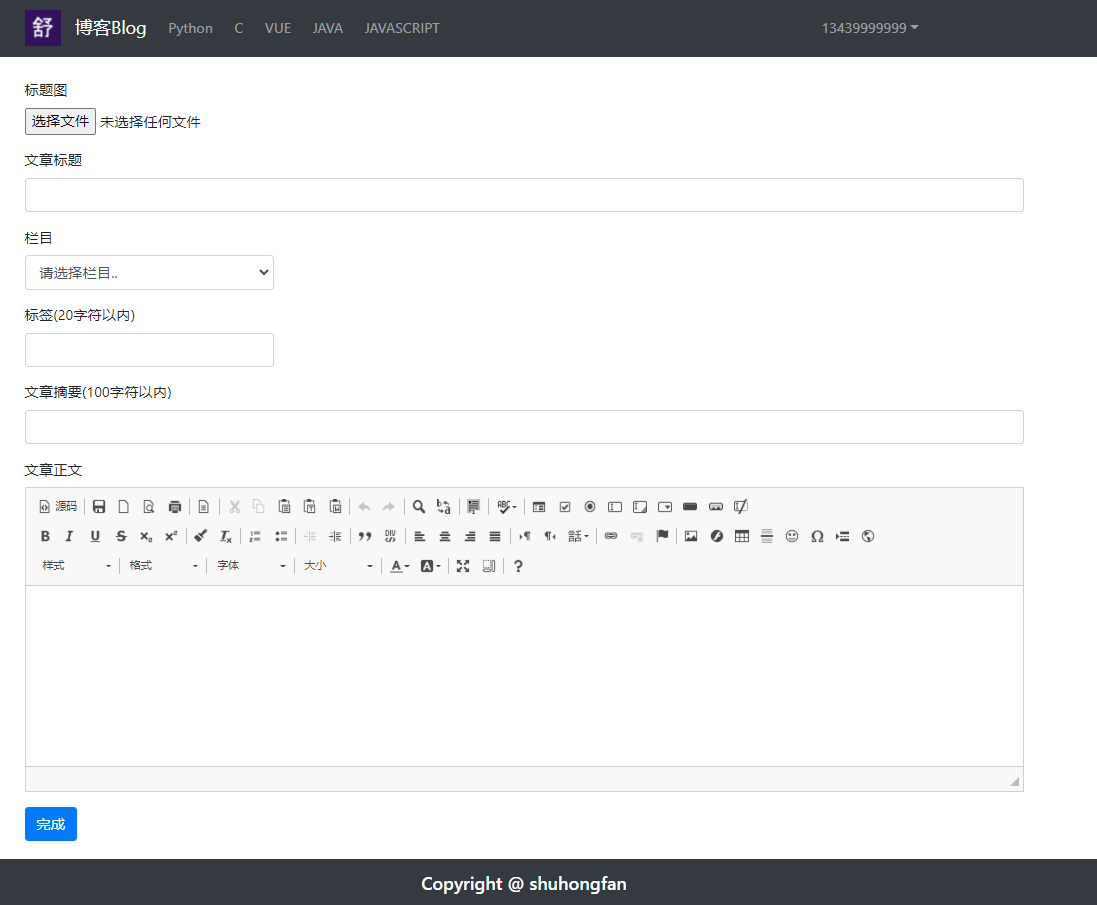
* 页面展示
* 判断用户是否登录
* 获取用户信息，模板渲染数据

用户中心修改的主要流程如下：

* 用户中心接口设计
* 用户中心修改接口实现
* 用户中心头像的上传和展示



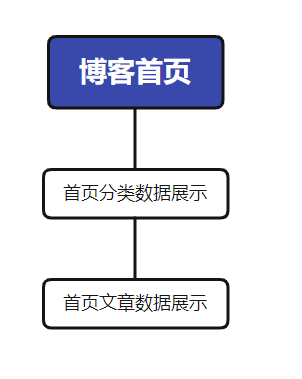
**图2.7 发布博客功能**



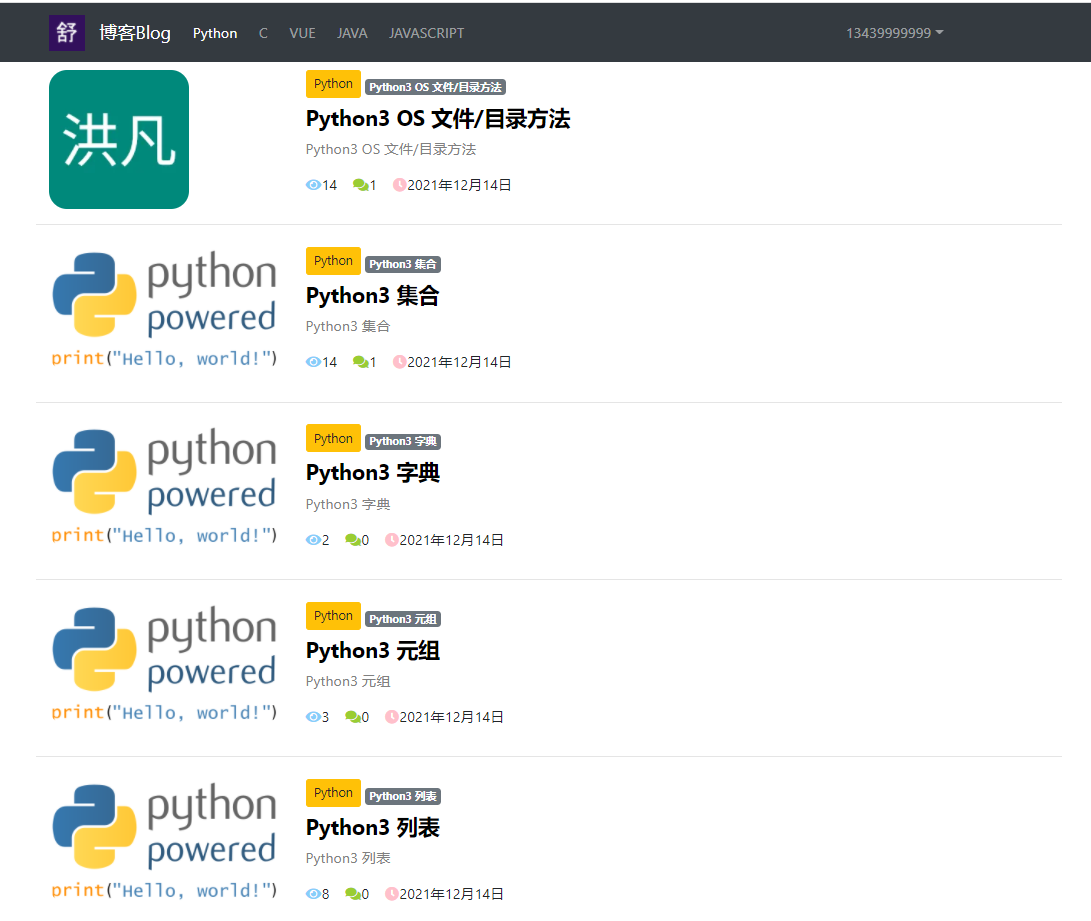
**图2.8 发布博客页面**

发布博客功能如图2.7所示。发布博客页面由图2.8所示。文章模型和文章分类模型的主要流程如下：

* 定义模型类并且通过迁移文件写入数据库
* 展示写博客页面
* 通过axios发送异步POST请求，携带文章数据以及用户信息请求服务器



**图2.9 博客首页功能**



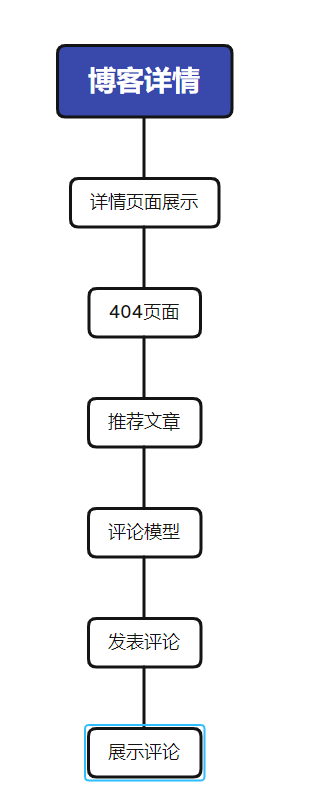
**图2.10 博客首页页面1**



**图2.11 博客首页页面2**

博客首页的主要流程如图2.5所示。博客首页页面如图2.10和2.11所示。

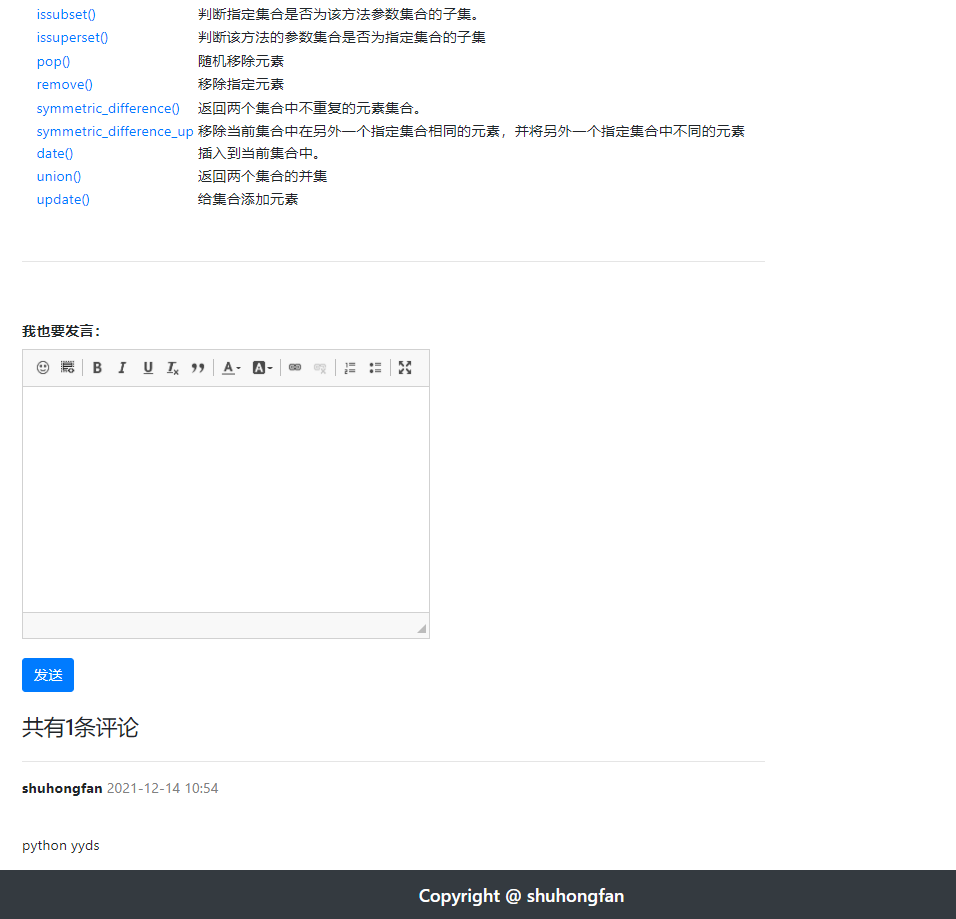
* 首页分类数据展示，可以通过页面顶部的导航栏切换你需要看的博客类型，对于同一类型数据超过10条进行分页展示
* 首页文章数据展示，每一项可以查看文章标题、文章图片、栏目、标签、文章摘要以及观看人数、评论总数和发布时间



**图2.12 博客详情功能**



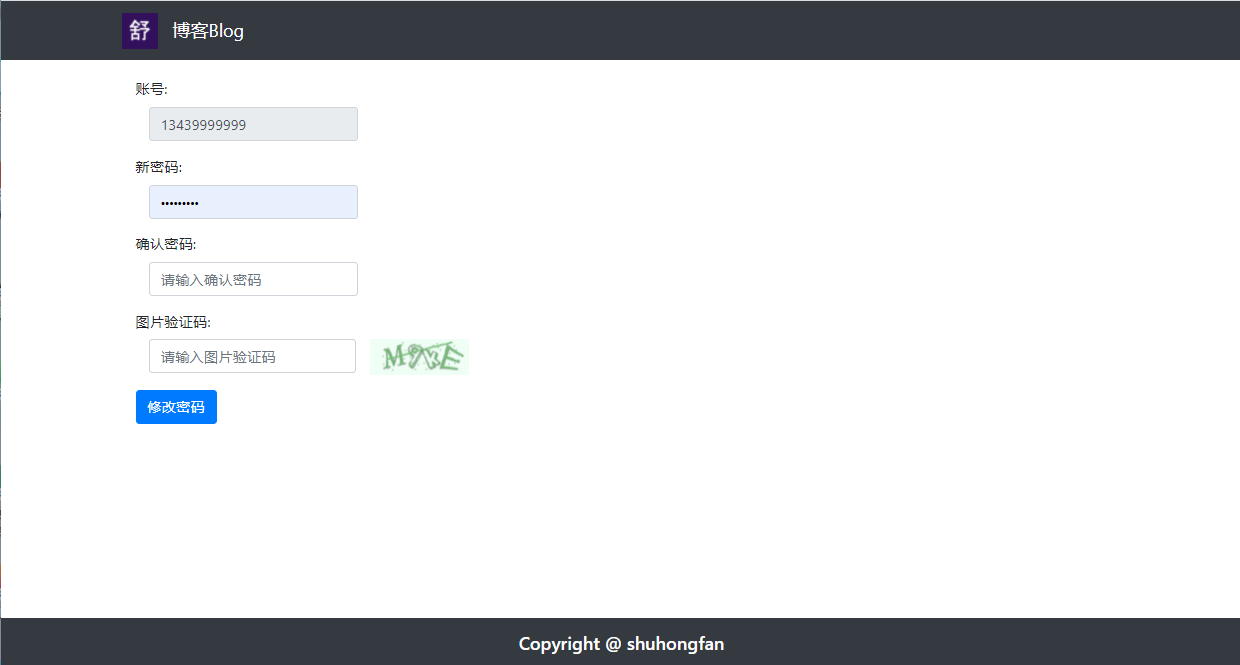
**图2.13 博客详情页面1**



**图2.14 博客详情页面2**

博客详情流程图如图2.1所示。博客详情页面如图2.13和2.14所示。博客详情主要处理了

* 详情页面的展示
* 没有此详情页面自动跳转到404页面
* 通过文章的浏览数量自动筛选出浏览次数前10条的文章展示到推荐文章面板
* 建立评论模型并迁移到数据库
* 实现发表评论存到数据，未登录用户需要登录才能提交评论
* 读取数据库展示评论的基本功能
* 删除自己发布的文章



**图2.15 用户重置密码**

用户重置密码页面如图2.15所示。博客详情主要处理了对登录用户的密码进行修改。

## 2.2作品实现要点

### 2.2.1 技术

前端采用JavaScript、BootStrap、Vuejs，后端采用Python、Django（开放源码的重Web应用框架），数据存储采用MySQL数据库、Redis、Session、Cookie技术。

### 2.2.2 核心代码

配置MySQL数据库

DATABASES = ***{* 'default'**: ***{* 'ENGINE'**: **'django.db.backends.mysql'**,  
 **'HOST'**: **'127.0.0.1'**,  
 **'PORT'**: 3306,  
 **'USER'**: **'root'**,  
 **'PASSWORD'**: **'root'**,  
 **'NAME'**: **'python\_blog'**,  
 ***}  
}***

配置Redis数据库

CACHES = ***{* "default"**: ***{*** *# 默认* **"BACKEND"**: **"django\_redis.cache.RedisCache"**,  
 **"LOCATION"**: **"redis://127.0.0.1:6379/0"**,  
 **"OPTIONS"**: ***{* "CLIENT\_CLASS"**: **"django\_redis.client.DefaultClient"**,  
 ***}  
 }***,  
 **"session"**: ***{*** *# session* **"BACKEND"**: **"django\_redis.cache.RedisCache"**,  
 **"LOCATION"**: **"redis://127.0.0.1:6379/1"**,  
 **"OPTIONS"**: ***{* "CLIENT\_CLASS"**: **"django\_redis.client.DefaultClient"**,  
 ***}  
 }***,  
***}***

Blog/Setting.py 博客配置类

*# 替换 系统的User 来使用我们自定义的User  
# 配置信息为 '子应用名。模型类型'*AUTH\_USER\_MODEL=**'users.User'***# 修改系统的未登录跳转链接*LOGIN\_URL=**'/login/'***# 设置上传的图片 保存到media目录下*MEDIA\_ROOT = os.path.join***(***BASE\_DIR,**"media/"*)****# 设置图片访问的统一路由*MEDIA\_URL = **'/media/'**

blog/urls.py 包含其他子应用路由

**def** log***(***request***)***:  
 **return** HttpResponse***(*'test'*)***urlpatterns = ***[*** path***(*'admin/'**, admin.site.urls***)***,  
 *#include的参数中,我们首先来设置一个元组 urlconf\_module,app\_name  
 # urlconf\_module 子应用的路由  
 # app\_name 子应用的函数名  
 # namespace 命名空间* path***(*''**,include***((*'users.urls'**,**'users'*)***,namespace=**'users'*))***,  
 *# path('',log),* path***(*''**,include***((*'home.urls'**,**'home'*)***,namespace=**'home'*))***,  
***]****# 图片访问的路由*urlpatterns += static***(***settings.MEDIA\_URL,document\_root=settings.MEDIA\_ROOT***)***

home/models.py 子应用模型类

**# 文章分类模型**

**class** ArticleCategory***(***models.Model***)***:  
 *# 文章分类  
 # 分类标题* title = models.CharField***(***max\_length=100,blank=**True*)*** *# 分类的创建时间* created = models.DateTimeField***(***default=timezone.now***)*** *# admin站点信息 调试查看对象* **def** \_\_str\_\_***(***self***)***:  
 **return** self.title  
  
 **class** Meta:  
 db\_table = **'tb\_category'** *#修改表名* verbose\_name=**'类别管理'**,  
 verbose\_name\_plural=verbose\_name

# 文章模型  
**class** Article***(***models.Model***)***:  
 *"""  
 文章  
 """  
 # 定义文章作者。 author 通过 models.ForeignKey 外键与内建的 User 模型关联在一起  
 # 参数 on\_delete 用于指定数据删除的方式，避免两个关联表的数据不一致。* author = models.ForeignKey***(***User, on\_delete=models.CASCADE***)*** *# 文章标题图* avatar = models.ImageField***(***upload\_to=**'article/%Y%m%d/'**, blank=**True*)*** *# 文章栏目的 “一对多” 外键* category = models.ForeignKey***(*** ArticleCategory,  
 null=**True**,  
 blank=**True**,  
 on\_delete=models.CASCADE,  
 related\_name=**'article'  
 *)*** *# 文章标签* tags = models.CharField***(***max\_length=20,blank=**True*)*** *# 文章标题。* title = models.CharField***(***max\_length=100,null=**False**,blank=**False*)*** *# 概要* sumary = models.CharField***(***max\_length=200,null=**False**,blank=**False*)*** *# 文章正文。* content = models.TextField***()*** *# 浏览量* total\_views = models.PositiveIntegerField***(***default=0***)*** *# 文章评论数* comments\_count = models.PositiveIntegerField***(***default=0***)*** *# 文章创建时间。  
 # 参数 default=timezone.now 指定其在创建数据时将默认写入当前的时间* created = models.DateTimeField***(***default=timezone.now***)*** *# 文章更新时间。  
 # 参数 auto\_now=True 指定每次数据更新时自动写入当前时间* updated = models.DateTimeField***(***auto\_now=**True*)*** *# 内部类 class Meta 用于给 model 定义元数据* **class** Meta:  
 *# ordering 指定模型返回的数据的排列顺序  
 # '-created' 表明数据应该以倒序排列* ordering = ***(*'-created'**,***)*** db\_table=**'tb\_article'** verbose\_name=**'文章管理'** verbose\_name\_plural=verbose\_name  
 *# 函数 \_\_str\_\_ 定义当调用对象的 str() 方法时的返回值内容  
 # 它最常见的就是在Django管理后台中做为对象的显示值。因此应该总是为 \_\_str\_\_ 返回一个友好易读的字符串* **def** \_\_str\_\_***(***self***)***:  
 *# 将文章标题返回* **return** self.title  
  
# 评论模型  
**class** Comment***(***models.Model***)***:  
 *#评论内容* content=models.TextField***()*** *#评论的文章* article=models.ForeignKey***(***Article,  
 on\_delete=models.SET\_NULL,  
 null=**True*)*** *#发表评论的用户* user=models.ForeignKey***(*'users.User'**,  
 on\_delete=models.SET\_NULL,  
 null=**True*)*** *#评论发布时间* created=models.DateTimeField***(***auto\_now\_add=**True*)* def** \_\_str\_\_***(***self***)***:  
 **return** self.article.title  
  
 **class** Meta:  
 db\_table=**'tb\_comment'** verbose\_name = **'评论管理'** verbose\_name\_plural = verbose\_name

迁移模型类

1.创建迁移文件

* python manage.py makemigrations

2.执行迁移文件

* python manage.py migrate

Home/urls.py 子应用路由类

urlpatterns = ***[*** *# 首页的路由* path***(*""**,IndexView.as\_view***()***,name=**'index'*)***,  
 *# 详情视图的路由* path***(*'detail/'**,DetailView.as\_view***()***,name=**'detail'*)  
]***

home/views.py 子应用控制器类

**class** IndexView***(***View***)***:  
 **def** get***(***self,request***)***:  
 *# 1. 获取所有分类信息* categories = ArticleCategory.objects.all***()*** *# 2.接收用户点击的分类id* cat\_id = request.GET.get***(*'cat\_id'**,1***)*** *# 3.根据分类id进行分类的查询* **try**:  
 category = ArticleCategory.objects.get***(***id=cat\_id***)* except** ArticleCategory.DoesNotExist:  
 **return** HttpResponseNotFound***(*"没有此分类"*)*** *# 4.获取分页参数* page\_num = request.GET.get***(*"page\_num"**,1***)*** page\_size = request.GET.get***(*"page\_size"**,10***)*** *# 5.根据分类信息查询文章数据* articles = Article.objects.filter***(***category=category***)*** *# 6.创建分页器* paginator = Paginator***(***articles,page\_size***)*** *# 7.进行分页处理* **try**:  
 page\_articles = paginator.page***(***page\_num***)* except** EmptyPage:  
 **return** HttpResponseNotFound***(*"empty page"*)*** *# 8. 获取总页数* total\_page = paginator.num\_pages  
 *# 9.组织数据传递给模板* context = ***{* 'categories'**: categories,  
 **'category'**: category,  
 **'articles'**: articles,  
 **'page\_articles'**: page\_articles,  
 **'total\_page'**: total\_page,  
 **'page\_size'**: page\_size,  
 **'page\_num'**: page\_num  
 ***}* return** render***(***request, **'index.html'**,context=context***)*class** DetailView***(***View***)***:  
 **def** get***(***self,request***)***:  
 *# 获取登录用户的信息* user = request.user  
 *# 1.接收文章id信息* id = request.GET.get***(*'id'*)*** *# 2.根据文章id进行文章数据的查询* **try**:  
 article = Article.objects.get***(***id=id***)* except** Article.DoesNotExist:  
 **return** render***(***request, **'404.html'*)* else**:  
 *# 浏览量+1* article.total\_views += 1  
 article.save***()*** *# 3.查询分类数据* categories = ArticleCategory.objects.all***()*** *# 4.查询浏览量前10的文章数据* hot\_articles = Article.objects.order\_by***(*'-total\_views'*)[***:9***]*** *# 5.获取分页请求参数* page\_size=request.GET.get***(*'page\_size'**,10***)*** page\_num=request.GET.get***(*'page\_num'**,1***)*** *# 6.根据文章信息查询评论数据* comments = Comment.objects.filter***(***article=article***)***.order\_by***(*'-created'*)*** *# 7.获取评论总数* total\_count = comments.count***()*** *# 8.创建分页器* paginator = Paginator***(***comments, page\_size***)*** *# 9.进行分页处理* **try**:  
 page\_comments = paginator.page***(***page\_num***)* except** EmptyPage:  
 **return** HttpResponseNotFound***(*"empty page"*)*** *# 10.总页数* total\_page = paginator.num\_pages  
  
 *# 11.组织模板数据* context = ***{* 'id'**:id,  
 **'categories'**:categories,  
 **'category'**:article.category,  
 **'article'**: article,  
 **'hot\_articles'**: hot\_articles,  
 **'total\_count'**:total\_count,  
 **'page\_comments'**:page\_comments,  
 **'total\_page'**:total\_page,  
 **'comments'**:comments,  
 **'page\_size'**:page\_size,  
 **'page\_num'**:page\_num,  
 **'username'**: user.username,  
 **'mobile'**: user.mobile,  
 **'avatar'**: user.avatar.url **if** user.avatar **else None**,  
 **'user\_desc'**: user.user\_desc  
 ***}* return** render***(***request,**"detail.html"**,context=context***)* def** post***(***self,request***)***:  
 *# 1. 先接收用户信息* user=request.user  
 *# 2.判断用户是否登录* **if** user **and** user.is\_authenticated:  
 *# 3.登录用户则可以接收 form数据  
 # 3.1接收评论数据* id = request.POST.get***(*'id'*)*** content=request.POST.get***(*'content'*)*** *# 3.2 判断文章是否存在* **try**:  
 article = Article.objects.get***(***id=id***)* except** Article.DoesNotExist:  
 **return** HttpResponseNotFound***(*"没有此文章"*)*** *# 3.2 保存评论数据* Comment.objects.create***(*** content=content,  
 article=article,  
 user=user  
 ***)*** *# 3.3修改文章的评论数量* article.comments\_count+=1  
 *# 3.4修改文章的评论数量* article.save***()*** *# 刷新当前页面 页面重定向* path = reverse***(*"home:detail"*)***+**'?id={}'**.format***(***article.id***)* return** redirect***(***path***)* else**:  
 *# 4. 未登录用户则跳转到登录页面* **return** redirect***(***reverse***(*"users:login"*))***

users/models.py 用户子应用模型类

**class** User***(***AbstractUser***)***:  
 *# 手机号* mobile = models.CharField***(***max\_length=11,unique=**True**,blank=**False*)*** *# 头像信息* avatar=models.ImageField***(***upload\_to=**'avatar//%Y%m%d'**,blank=**True*)*** *# 简介信息* user\_desc = models.CharField***(***max\_length=500,blank=**True*)*** *# 修改认证的字段为mobile* USERNAME\_FIELD = **'mobile'** *# 创建超级管理员必须输入的字段(不包括 手机号和密码)* REQUIRED\_FIELDS = ***[*'username'**,**'email'*]* class** Meta:  
 db\_table = **'tb\_users'** *#修改表名* verbose\_name = **'用户管理'** *#admin 后台显示* verbose\_name\_plural = verbose\_name *#admin后台显示* **def** \_\_str\_\_***(***self***)***:  
 **return** self.mobile

users/urls.py 用户子应用路由类

urlpatterns = ***[*** *# path的路由  
 # 第一个参数: 路由  
 # path的第二个参数: 视图函数名* path***(*'register/'**,RegisterView.as\_view***()***,name=**'register'*)***,  
  
 *# 图片验证码的路由* path***(*'imagecode/'**,ImageCodeView.as\_view***()***,name=**'imagecode'*)***,  
  
 *# 登录路由* path***(*"login/"**,LoginView.as\_view***()***,name=**"login"*)***,  
  
 *# 退出登录* path***(*"logout/"**,LogoutView.as\_view***()***,name=**"logout"*)***,  
  
 *# 个人中心* path***(*"center/"**,UserCenterView.as\_view***()***,name=**"center"*)***,  
  
 *# 写博客的路由* path***(*"writeblog/"**,WriteBlogView.as\_view***()***,name=**"writeblog"*)***,  
  
 *# 删除博客* path***(*"deleteblog/"**,DeleteBlogView.as\_view***()***,name=**"deleteblog"*)***,  
  
 *# 重置密码* path***(*"resetpassword/"**,ResetPasswordView.as\_view***()***,name=**"resetpassword"*)  
]***

users/views.py 用户子应用控制器类

**class** RegisterView***(***View***)***:  
 **def** get***(***self, request***)***:  
 **return** render***(***request, **"register.html"*)* def** post***(***self, request***)***:  
 *# 1.接收数据* mobile = request.POST.get***(*"mobile"*)*** password = request.POST.get***(*"password"*)*** password2 = request.POST.get***(*"password2"*)*** *# 2.验证数据  
 # 2.1 参数是否安全* **if not** all***([***mobile, password, password2***])***:  
 **return** HttpResponseBadRequest***(*"缺少必要的参数！"*)*** *# 2.2 手机号的格式是否正确* **if not** re.match***(*r'^1[3-9]\d{9}$'**, mobile***)***:  
 **return** HttpResponseBadRequest***(*"手机号不符合规则！"*)*** *# 2.3 密码是否符合格式* **if not** re.match***(*r"^[0-9A-Za-z]{8,20}$"**, password***)***:  
 **return** HttpResponseBadRequest***(*"请输入8-20为密码，密码是数字，字母！"*)*** *# 2.4 密码和确认密码是否要一致* **if** password != password2:  
 **return** HttpResponseBadRequest***(*"两次输入密码不一致！"*)*** *# 3.保存注册信息  
 # create\_user 可以使用系统的方法来对密码进行加密* **try**:  
 user = User.objects.create\_user***(***username=mobile,  
 mobile=mobile,  
 password=password***)* except** DatabaseError **as** e:  
 logger.error***(***e***)* return** HttpResponseBadRequest***(*"注册失败"*)*** *# 4.返回相应跳转到指定页面  
 # redirect 进行重定向  
 # reverse 可以通过 namespace:namespace 来获取到视图所对应的路由* response = redirect***(***reverse***(*"home:index"*))*** *# login() 登录状态保持* login***(***request, user***)*** *# 设置cookie信息，方便 首页中 用户信息的展示和判断* response.set\_cookie***(*"is\_login"**, **True*)*** response.set\_cookie***(*"username"**, user.username, max\_age=7 \* 24 \* 3600***)* return** response  
  
  
**class** ImageCodeView***(***View***)***:  
 **def** get***(***self, request***)***:  
 *# 1.接收前端传递过来的uuid* uuid = request.GET.get***(*'uuid'*)*** *# 2.判断uuid是否获取到* **if** uuid **is None**:  
 **return** HttpResponseBadRequest***(*'没有传递UUID'*)*** *# 3.通过调用captcha来乘车图片验证码* text, image = captcha.generate\_captcha***()*** *# 4.将图片内容保存到redis中* redis\_connection = get\_redis\_connection***()*** *# uuid作为一个key，图片内容作为一个value，同时我们还需要设置一个时效  
 # key设置为uuid  
 # seconds 过期秒数 300秒 5 分钟过期时间  
 # value text* redis\_connection.setex***(*"img:%s"** % uuid, 300, text***)***;  
 *# 5. 返回图片二进制* **return** HttpResponse***(***image, content\_type=**"image/jpeg"*)*class** LoginView***(***View***)***:  
 **def** get***(***self, request***)***:  
 **return** render***(***request, **"login.html"*)* def** post***(***self, request***)***:  
 *# 1.接收参数* mobile = request.POST.get***(*"mobile"*)*** password = request.POST.get***(*"password"*)*** remember = request.POST.get***(*"remember"*)*** *# 2.参数的验证  
 # 2.1 验证手机号* **if not** re.match***(*r'^[a-zA-Z0-9]{8,20}$'**, mobile***)***:  
 **return** HttpResponseBadRequest***(*"手机号不符合规则"*)*** *# 2.2 验证密码是否符合规则* **if not** re.match***(*r'^[a-zA-Z0-9]{8,20}$'**, password***)***:  
 **return** HttpResponseBadRequest***(*"密码不符合规则"*)*** *# 3.用户认证登录  
 # 采用系统自带的认证方法进行认证  
 # 如果我们的用户名和密码正确，会返回user  
 # 如果我们的用户名或密码不正确,会返回None  
 # 默认的认证的方法 针对于username字段进行用户名的判断  
 # 当前的判断信息是 手机号,索引我们需要修改一下认证字段* user = authenticate***(***mobile=mobile, password=password***)*** *# 4.状态的保持* login***(***request, user***)*** *# 5.根据用户选择的是否记住登录状态来进行判断  
  
 # 根据next参数来进行页面的跳转* next\_page = request.GET.get***(*"next"*)* if** next\_page:  
 response = redirect***(***next\_page***)* else**:  
 response = redirect***(***reverse***(*'home:index'*))* if** remember != **'on'**: *# 没有记住用户信息  
 # 浏览器关闭后* request.session.set\_expiry***(***0***)*** *# 6.为了首页显示我们需要设置一些cookie信息* response.set\_cookie***(*"is\_login"**, **True*)*** response.set\_cookie***(*"username"**, user.username, max\_age=14 \* 24 \* 3600***)* else**: *# 记住用户信息 过期时间2周* request.session.set\_expiry***(*None*)*** response.set\_cookie***(*'is\_login'**, **True**, max\_age=14 \* 24 \* 3600***)*** response.set\_cookie***(*"username"**, user.username, max\_age=14 \* 24 \* 3600***)*** *# 7. 返回相应* **return** response  
  
  
**class** LogoutView***(***View***)***:  
 **def** get***(***self,request***)***:  
 *# 1. session数据清除* logout***(***request***)*** *# 2. 删除部分cookie数据* response=redirect***(***reverse***(*'home:index'*))*** response.delete\_cookie***(*"is\_login"*)*** *# 3.跳转到首页* **return** response  
  
  
*# LoginRequiredMixin  
# 如果用户未登录的话,则会进行默认的跳转  
# 默认的跳转连接是: accounts/login/?next/center***class** UserCenterView***(***LoginRequiredMixin,View***)***:  
 **def** get***(***self,request***)***:  
 *# 获取登录用户的信息* user = request.user  
 *# 组织获取用户的信息* context = ***{* 'username'**: user.username,  
 **'mobile'**: user.mobile,  
 **'avatar'**: user.avatar.url **if** user.avatar **else None**,  
 **'user\_desc'**:user.user\_desc  
 ***}* return** render***(***request,**"center.html"**,context=context***)* def** post***(***self, request***)***:  
 user = request.user  
 *# 1.接收参数* username = request.POST.get***(*"username"**,user.username***)*** user\_desc = request.POST.get***(*'desc'**,user.user\_desc***)*** avatar = request.FILES.get***(*"avatar"*)*** *# 2.将参数保存起来* **try**:  
 user.username = username  
 user.user\_desc = user\_desc  
 **if** avatar:  
 user.avatar = avatar  
 user.save***()* except** Exception **as** e:  
 logger.error***(***e***)* return** HttpResponseBadRequest***(*"修改失败,请稍后再试"*)*** *# 3.更新cookie中的username信息  
  
 # 4.刷新当前页面(重定向操作)* response = redirect***(***reverse***(*"users:center"*))*** response.set\_cookie***(*"username"**,user.username,max\_age=14\*3600\*24***)*** *# 5. 返回相应* **return** response  
  
  
**class** WriteBlogView***(***LoginRequiredMixin,View***)***:  
  
 **def** get***(***self,request***)***:  
 *#查询所有分类模型* categories=ArticleCategory.objects.all***()*** context = ***{* 'categories'**:categories  
 ***}* return** render***(***request,**'write\_blog.html'**,context=context***)* def** post***(***self,request***)***:  
 *"""  
 # 1.接收数据  
 # 2.验证数据  
 # 3.数据入库  
 # 4.跳转到指定页面（暂时首页）  
 :param request:  
 :return:  
 """  
 # 1.接收数据* avatar=request.FILES.get***(*'avatar'*)*** title=request.POST.get***(*'title'*)*** category\_id=request.POST.get***(*'category'*)*** tags=request.POST.get***(*'tags'*)*** sumary=request.POST.get***(*'sumary'*)*** content=request.POST.get***(*'content'*)*** user=request.user  
  
 *# 2.验证数据  
 # 2.1 验证参数是否齐全* **if** category\_id == **'none'**:  
 **return** HttpResponseBadRequest***(*'参数不全'*)* if not** all***([***avatar,title,category\_id,sumary,content***])***:  
 **return** HttpResponseBadRequest***(*'参数不全'*)*** *# 2.2 判断分类id* **try**:  
 category=ArticleCategory.objects.get***(***id=category\_id***)* except** ArticleCategory.DoesNotExist:  
 **return** HttpResponseBadRequest***(*'没有此分类'*)*** *# 3.数据入库* **try**:  
 article=Article.objects.create***(*** author=user,  
 avatar=avatar,  
 title=title,  
 category=category,  
 tags=tags,  
 sumary=sumary,  
 content=content  
 ***)* except** Exception **as** e:  
 logger.error***(***e***)* return** HttpResponseBadRequest***(*'发布失败，请稍后再试'*)*** *# 4.跳转到指定页面（暂时首页）* **return** redirect***(***reverse***(*'home:index'*))****# 删除博客***class** DeleteBlogView***(***LoginRequiredMixin,View***)***:  
 **def** get***(***self,request***)***:  
 *# 获取登录用户的信息* user = request.user  
 *# 获取删除博客id* articleID = request.GET.get***(*'articleID'*)*** *# 数据入库* **try**:  
 Article.objects.get***(***id=articleID***)***.delete***()* except** Exception **as** e:  
 logger.error***(***e***)* return** HttpResponseBadRequest***(*'删除博客失败，请稍后再试'*)*** *# 4.跳转到指定页面（暂时首页）* **return** redirect***(***reverse***(*'home:index'*))****# 重置密码***class** ResetPasswordView***(***LoginRequiredMixin,View***)***:  
 **def** get***(***self,request***)***:  
 user = request.user  
 *# 组织获取用户的信息* context = ***{* 'username'**: user.username,  
 **'mobile'**: user.mobile,  
 **'avatar'**: user.avatar.url **if** user.avatar **else None**,  
 **'user\_desc'**: user.user\_desc  
 ***}* return** render***(***request,**"reset\_password.html"**, context=context***)* def** post***(***self,request***)***:  
 *# 1.读取数据* mobile = request.POST.get***(*"mobile"*)*** password = request.POST.get***(*"password"*)*** password2 = request.POST.get***(*"password2"*)*** *# 2判断参数是否齐全* **if not** all***([***mobile, password, password2***])***:  
 **return** HttpResponseBadRequest***(*'缺少必传参数'*)*** *# 判断手机号是否合法* **if not** re.match***(*r'^1[3-9]\d{9}$'**, mobile***)***:  
 **return** HttpResponseBadRequest***(*'请输入正确的手机号码'*)*** *# 判断密码是否是8-20个数字* **if not** re.match***(*r'^[0-9A-Za-z]{8,20}$'**, password***)***:  
 **return** HttpResponseBadRequest***(*'请输入8-20位的密码'*)*** *# 判断两次密码是否一致* **if** password != password2:  
 **return** HttpResponseBadRequest***(*'两次输入的密码不一致'*)*** *# 3.根据手机号查询数据* **try**:  
 user = User.objects.get***(***mobile=mobile***)* except** User.DoesNotExist:  
 **return** HttpResponseBadRequest***(*'修改失败，请稍后再试'*)* else**:  
 *# 修改用户密码* user.set\_password***(***password***)*** user.save***()*** *# 跳转到登录页面* response = redirect***(***reverse***(*'users:login'*))* return** response

# 第四章 总结

通过实现本作品，我学会使用Python的web框架Django开发一个在线博客系统，在制作博客项目的过程中我进一步掌握了Python的基础语法，基本掌握Django的基础语法，熟悉MySQL的基础语法，进一步学习前端HMTL+CSS+JavaScript的基础知识，基本掌握Vue.js和bootstrap的基本使用，更进一步学习HTTP基础网络编程知识，能够使用Axios进行异步页面请求，从Python后台获取数据，并在页面上进行分页展示。

能够使用Django用户自定义模型类生成迁移文件，并通过执行迁移文件在MySQL数据库中生成对应字段的表。学会使用Django默认用户认证系统，它处理用户账号、组、权限以及基于cookie的用户会话。Django认证系统同时处理认证（验证一个用户是否它声称的那个人，可用于账号登录）和授权（授权决定一个通过了认证的用户被允许做什么），通过Django自带的权限管理大大减少我的工作量，使用user = User.objects.create\_user(username, password, \*\*extra\_fields)就可以创建一个用户，使用user = authenticate(username=username, password=password, \*\*kwargs)就可以进行用户认证，并可以设置密码set\_password(raw\_password)和校验密码是否正确check\_password(raw\_password)。

学会使用GitHub上的第三方python包，比如图形验证码captcha，直接在Django进行导入调用‘text, image = captcha.generate\_captcha()’即可生成验证码和获取验证码的内容。

# 参考文献

[1] Django documentation. https://docs.djangoproject.com/en/4.0/

[2] 菜鸟教程-Django 教程. https://www.runoob.com/django/django-tutorial.html

[3] Vue.js文档. <https://cn.vuejs.org/v2/guide/>

[4] bootstrap文档. <https://www.bootcss.com/>

[5] jQuery API文档. https://jquery.cuishifeng.cn/

[6] axios文档. https://www.axios-http.cn/

[7] bootstrap文档. https://www.bootcss.com/