|  |  |
| --- | --- |
| 学院标识组合（方） | **校名** |

**本科生毕业论文（设计）**

基于Python的房屋租赁

系统与实现

**Python-based House Lease System and Implementation**

专 业  **电气工程及其自动化**

姓 名 王举

学 号  **16031107**

指 导 教 师  **陈垚**

完 成 时 间

独创性声明

本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包括其他人已经发表或撰写过的研究成果，也不包含为获得商洛学院或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文作者签名： 日期：

关于论文使用授权的说明

本人完全了解商洛学院有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

□公开 □保密（\_\_\_\_年\_\_\_\_月） (保密的学位论文在解密后应遵守此协议)

签名： 导师签名： 日期：

基于Python的租赁房屋系统与实现

**摘要：**近几年来，我国的经济迅速发展，各大城市的流动人口正在逐步增加，导致了房屋租赁的需求也日益剧增。进入信息化时代，房产行业的发展和居民的租房需求的多元化带动了房产中介的发展，虽然存在各种房屋中介网站，但总体的信息化管理水平有限，对提高管理和促进房屋租赁行业的发展起到至关重­要的作用。而且我国房产中介仍仅限于传统的方式，信息间的传递都是孤立的。因此，未来房产租赁信息市场面临着网络化发展的趋势，对实现房产交易中共享信息的功能，增加交易的机会，缩短交易的时间【1】有着重要的研究意义。

本文首先对该系统的开发环境（PyCharm、HBuilder X和微信小程序开发者工具）进行了介绍，综合考虑各技术的优缺点和该系统的需求，决定本系统采的实现方式:该系统分前台和后台，后台管理功能的实现主要采用Python的Django框架，而Python的Flask框架又特别的小巧、灵活，因此前台API接口的实现主要依赖于Flask框架，值得一提的是由于DCloud公司推出的uni-app前端框架具有跨平台、可转换、小巧等优点，故前台的前端代码完全采用uni-app实现。需要指出，在该系统中数据的缓存采用的是Redis缓存技术，数据的持久性存储采用的是MySQL技术。同时也引入阿里云的短信验证、身份认证以及OSS存储等第三方技术。

该系统的后台主要有超级管理员、赞助管理员和审核管理员三大角色。其中超级管理员的权限有角色管理、用户管理、公告发布和删除、评论管理、投诉管理；审核管理员则可以进行房屋信息的审核和公告审核；赞助管理员就只能进行广告的管理和代金券的管理。该系统的前台针对的群体则主要是普通租户或房东。其中普通租户只能进行房屋信息查看以及评论、投诉等基本操作，如果想拥有发布房源的权限，便可以在升级为房东之后执行该操作。

**关键字:**房屋租赁；Python；Django；Flask；uni-app；MySQL；后台；前台

**Python-based House Lease System and Implementation**

**Abstract:**In recent years, China's economy has developed rapidly, and the floating population in major cities has gradually increased, resulting in a sharp increase in the demand for housing leasing. Entering the information age, the development of the real estate industry and the diversification of residents' rental needs have driven the development of real estate intermediaries. Although there are various housing intermediary websites, the overall level of information management is not rigorous, which is Development has played a crucial role. Moreover, China's real estate agency is still limited to traditional methods, and the transmission of information is isolated. Therefore, the future real estate leasing information market is facing the trend of network development. It has important research significance for realizing the function of sharing information in real estate transactions, increasing transaction opportunities, and shortening transaction time.

This article first outlines the development environment of the system (PyCharm, HBuilder X, and WeChat applet developer tools), comprehensively analyzes the advantages and disadvantages of various technologies, and combines the needs and characteristics of the system. Back-end implementation: The system is divided into front-end and back-end. The implementation of back-end management functions mainly uses Python's Django framework, and Python's Flask framework is particularly small and flexible. Therefore, the implementation of the front-end API interface mainly depends on the Flask framework, which is worth mentioning It is because the uni-app front-end framework launched by DCloud has the advantages of cross-platform, convertible, and small, so the front-end code of the front desk is completely implemented by uni-app. It should be pointed out that the data cache in this system uses Redis cache technology, and the persistent storage of data uses MySQL technology. At the same time, Alibaba Cloud also introduced third-party technologies such as SMS verification, identity authentication and OSS storage.

The background of the system mainly has three roles of super administrator, sponsor administrator and audit administrator. Among them, the super administrator's permissions include role management, user management, announcement release and deletion, comment management, and complaint management; the review administrator can conduct house information review and announcement review; the sponsor administrator can only manage advertising and cash Management of coupons. The group targeted by the front desk of the system is mainly ordinary tenants or landlords. Among them, ordinary tenants can only perform basic operations such as viewing house information, comments, and complaints. If you want to have the permission to publish a house, you can perform this operation after upgrading to a landlord.

**Keywords:** housing lease;Python;Django;Flask;uni-app;MySQL;background;front des

目录

[1 绪论 1](#_Toc41822762)

[1.1 课题研究的目的和意义 1](#_Toc41822763)

[1.1.1 研究目的 1](#_Toc41822764)

[1.1.2 研究意义 1](#_Toc41822765)

[1.2 课题研究的国内外现状 1](#_Toc41822766)

[1.2.1 国内现状 1](#_Toc41822767)

[1.2.2 国外现状 2](#_Toc41822768)

[1.3 论文的主要功能与结构 2](#_Toc41822769)

[1.3.1 论文的主要作用 2](#_Toc41822770)

[1.3.2 论文的主要结构 3](#_Toc41822771)

[2 开发环境及相关技术介绍 4](#_Toc41822772)

[2.1 开发环境介绍 4](#_Toc41822773)

[2.1.1 PyCharm 4](#_Toc41822774)

[2.1.2 HBuilder 4](#_Toc41822775)

[2.2 相关技术介绍 5](#_Toc41822776)

[2.2.1 Python 5](#_Toc41822777)

[2.2.2 Web框架 5](#_Toc41822778)

[2.2.3 数据库技术 5](#_Toc41822779)

[2.2.4 uni-app 6](#_Toc41822780)

[3 系统分析 7](#_Toc41822781)

[3.1 可行性分析 7](#_Toc41822782)

[3.1.1 技术可行性 7](#_Toc41822783)

[3.1.2 经济可行性 7](#_Toc41822784)

[3.1.3 操作可行性 7](#_Toc41822785)

[3.2 系统需求分析 7](#_Toc41822786)

[3.2.1 系统目标 8](#_Toc41822787)

[3.2.2 系统需求分析 8](#_Toc41822788)

[3.3 业务分析 9](#_Toc41822789)

[3.3.1 用户群体 9](#_Toc41822790)

[3.3.2 业务模型图和模型表建立 9](#_Toc41822791)

[4 系统的设计与实现 11](#_Toc41822792)

[4.1 后台管理系统 11](#_Toc41822793)

[4.1.1 登录界面 11](#_Toc41822794)

[4.1.2 超级管理员 12](#_Toc41822795)

[4.1.3 审核管理员 12](#_Toc41822796)

[4.1.4 赞助管理员 13](#_Toc41822797)

[4.2前台操作系统 14](#_Toc41822798)

[4.2.1 首页模块 14](#_Toc41822799)

[4.2.2 收藏模块 15](#_Toc41822800)

[4.2.3 聊天模块 16](#_Toc41822801)

[4.2.4 我的模块 16](#_Toc41822802)

[5 测试 18](#_Toc41822803)

[总结 19](#_Toc41822804)

[参考文献 20](#_Toc41822805)

[附录 21](#_Toc41822806)

[致谢 87](#_Toc41822807)

1 绪论

1.1 课题研究的目的和意义

1.1.1 研究目的

作为世界的第一人口大国、第二大经济体的中国，各大城市流动人口每年都在逐步增加，导致对房屋租赁的市场需求也日益剧增。房屋中介也大量地涌现了出来，出租、租赁的大量业务量产生了对高效合理的租赁业务信息管理系统的迫切需求。另一方面，因为计算机的崛起，设计一款信息管理系统，可以极大的提高人们工作效率。

1.1.2 研究意义

由于国外互联网技术的出现，在大型工业中的使用也很受欢迎。 在欧美等发达国家，50％的中低收入者依靠租金来满足他们的需求。 经过中国和城市多年的经济发展，国内农民在该国总人口中所占的比例稳步下降。越来越多的农民正逐渐从乡村转移到城市。同时，大量学生从一个国家转移到另一个城市。这些大批人在流动，这是租赁市场的重要力量。流动人口的快速增长推动了租赁市场。在许多房屋租赁网站上，对房屋信息的管理仍然不完善，因此在提高租赁效率和降低成本方面没有发挥重要作用。

1.2 课题研究的国内外现状

1.2.1 国内现状

随着中国城市移民的快速增长，许多人选择到城市学习和工作，从而增加了房地产平台买家的数量，也为租赁市场的复苏做出了贡献。在中国和其他各种地点。解放中国房地产市场并不是一夜之间发展起来的，而是随之而来的是各种计划。随着中国加入世贸组织，房地产业向其竞争对手和出口商发起了挑战。借助新兴的互联网技术和信息交换平台，外国公司对中国的租赁市场产生了重大影响。在国外竞争的情况下，中国的信息管理公司已开始开发自己的租赁平台。随着新世纪的到来，中国的互联网技术发展迅速。大多数在线公司已使用互联网技术，并通过与中国品牌会合而成为大生意。房地产市场被认为是第三方业务的一部分，并且正在迅速发展。但是，中国中小企业租赁系统的发展缓慢，它们之间没有数据，它们是独立工作的。该程序的设计和开发质量也很差，并且没有使用任何合适的文件供第三方考虑从“ Internet +”数据库中提取内容。现在，中国的房地产系统存在问题：

（1）无效的数据信息。 许多公司发布的信息并不涉及实际的房地产信息，有关房屋的过多信息给用户的选择带来了很多问题。

（2）传输速率低，状态不高。 租户管理系统不符合现代信息管理系统的要求。

（3）信息管理系统缺乏专业人员。

基于上述问题，中国许多科学家一直在研究中国房屋管理系统，一些科学家根据当前的推荐问题设计了一些简单的系统。2018年黄学波和梁俊卿基于BIM消防的房屋租赁系统的研究与实现【2】。2019年李瑞娥、沈文枫、杨宇华、周丽萍和王洋，基于区块链实现了房屋租赁分配系统设计与实现【3】。2019年张基提出基于Android平台的房屋租赁管理系统的设计【4】。2019年朱园园、邵雯娟、黄志远基于Scrapy实现了房屋租赁信息搜索系统设计与实现【5】。2019年济南大学李佳雯基于区块链技术设计与实现了房屋租赁平台【6】。2019年东南大学罗茜基于微服务实现了房屋租赁服务系统设计【7】。2019年武海龙,李国平基于SpringBoot实现了房屋租赁系统设计【8】。2018年江西财经大学胡世民完成了基于互联网+的房屋租赁管理系统的设计与实现【9】。2017年王飞研究了房屋租赁信息系统的设计与实现【10】。

1.2.2 国外现状

随着欧美信息技术的兴起，租赁系统的开发首先在欧美实施，并完成了有关租赁市场的信息。但是，外部租赁系统技术的发展已导致许多学者考虑使用行政理论来解决租赁过程中的问题，并开发包含实用模块的系统。Forgione GA使用决策方法来确定到军事租赁地点的距离。Park M.等人提出了用于管理家庭事务的新技术。她认为，租赁行业必须谨慎，并实施新的策略和技术来管理租赁信息和管理管理系统。可以改善经理人员管理的工资。技术和方法的进步是韦伯和罗伯特·阿等学者的研究。  
  成年的租户公司使用Internet来管理其房屋和客户的信息，并将房屋和客户的信息与MMS（多重上市服务）相结合，以了解这两个数据库的一般管理。另外，一些学者强调了租金管理系统的重要性。

1.3 论文的主要功能与结构

1.3.1 论文的主要作用

本文首先认真研究了当下各优秀的Web框架及数据库相关技术的理论知识以及各自的优缺点，然后从该项目的业务流程、功能性需求以及使用场景等多个方面综合考虑，基于Python开发出了一套适用于国内现状的房屋租赁系统。本论文所做的主要研究工作如下：

(1)对国内房屋租赁市场进行了一定的需求分析和研究，同时结合市面上已经成熟的租赁系统，搭建出了一套适用于国内的房屋租赁系统的业务流程图。

(2)分析管理系统的结构的特点以及各Web框架的优缺点，选择了合适的Web进行来针对性实现功能。

(3)针对数据的存储的目的以及使用场景进行分析，采用MySQL数据库进行数据的持久性存储，同时选择Redis进行了数据的缓存。

(4)采用相关技术，完成基于Python的房屋租赁系统的整体结构设计和功能模块实现。

1.3.2 论文的主要结构

本文共分为以下五章：

第1章 本章主要介绍了该房屋管理系统的研究意义和背景、国内外的研究现状以及本论文的主要功能与结构。

第2章 本章对该系统采用的开发环境以及主要技术展开介绍。

第3章 本章对该系统进行可行性分析和业务分析。

第4章 本章主要针对实现该系统的数据库设计、前后台的模块化功能实现以及测试调优的介绍。

第5章 总结。

2 开发环境及相关技术介绍

2.1 开发环境介绍

2.1.1 PyCharm

PyCharm是JetBrains为Python打造的IDE，为Python开发者提高了非常齐全的开发工具，比如智能提示、语法高亮、调试、代码跳转、项目管理、自动导入、版本控制、单元测试、全局搜索、正则替换，如图2-1：

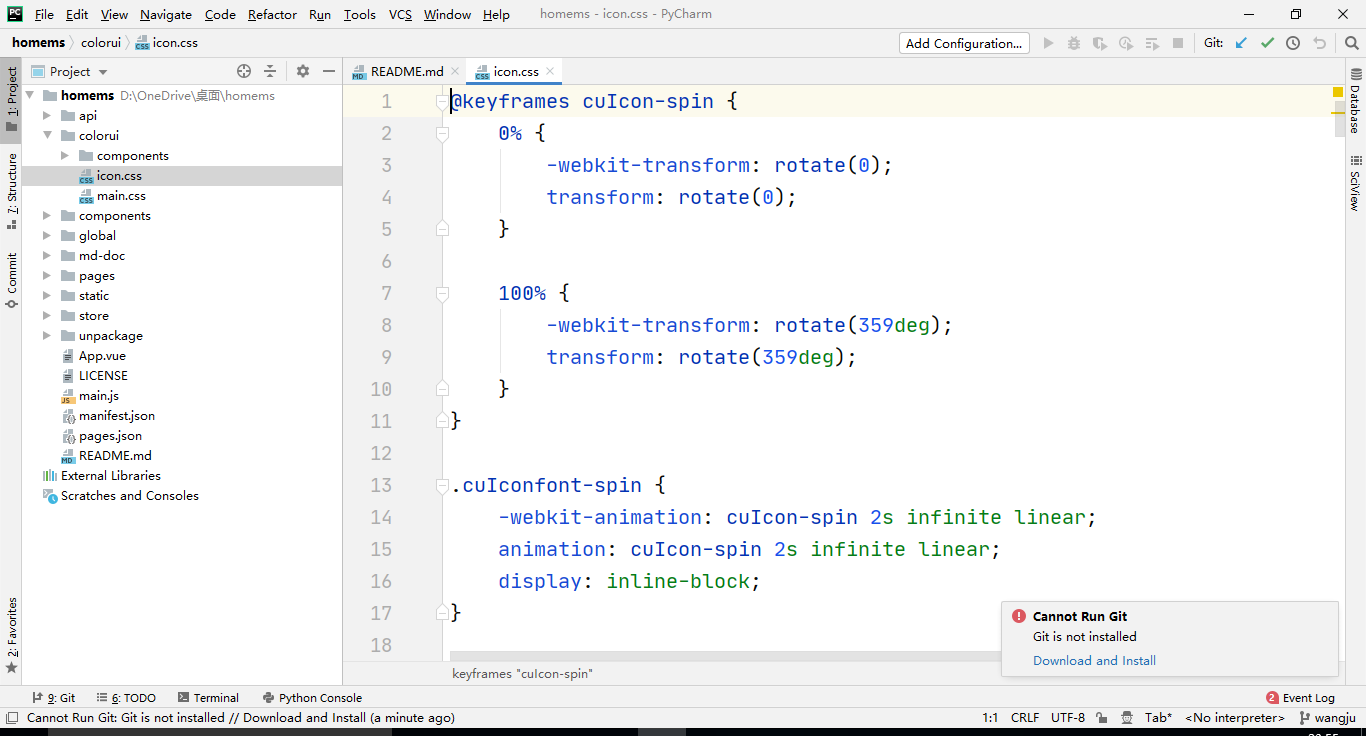


图 2-1

2.1.2 HBuilder

XHBuilder X是数字天堂（DCloud）专为vue打造的一款Web开发IDE。HBuilder X同时不但轻巧、急速，而且有着相当强大的语法提示功能。HBuilder X的界面方式比较简单，如图2-2；HBuilder X内置着各种环境，无需配置nodejs。HBuilder X是通用的前端开发工具，但为uni-app做了特别强化。

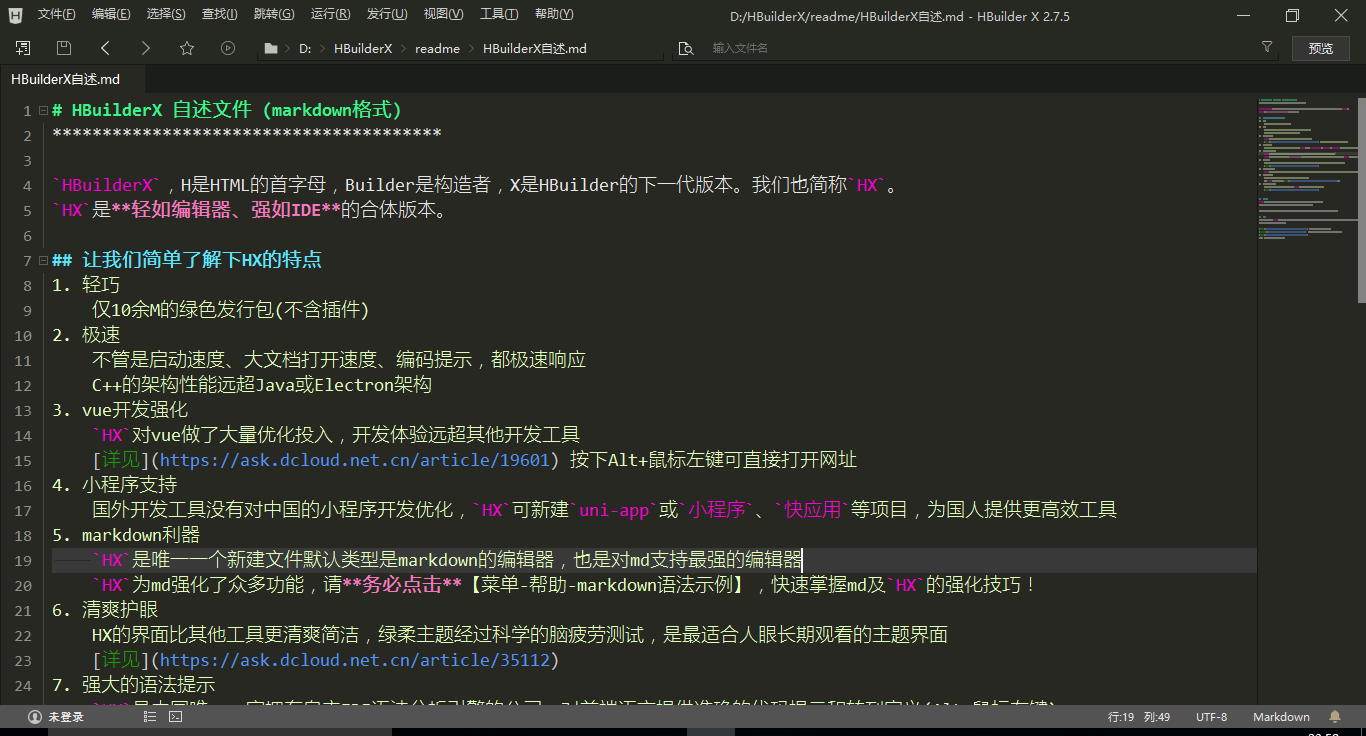


图 2-2

2.2 相关技术介绍

2.2.1 Python

Python是一种可跨平台的解释型程序设计语言。Python不但在爬虫、数据分析方面有着得天独厚的优势，而且随着Python语言的不断完善，越来越多地被用于独立的、大型项目的开发，随着人工智能的崛起，Python语言的使用范畴也正在迅速扩大，并且，Python也拥有者相当多的第三方库，也可以直接和各种数据库进行交互，极大地缩减了程序员的开发成本。

2.2.2 Web框架

Python有许多不同的 Web 框架，在该系统中主要用到了Django和Flask框架。可以这样理解，如果把Flask比喻成一个毛坯房，内部家具基本没有，然而Django就相当于是一栋别墅，内部家具齐全。市面上许多网站和APP都是使用的Django框架，Django是一个开源的Web框架，是基于Flask用Python写成的，相比于Django框架，Flask框架就显得比较轻量级。

2.2.3 数据库技术

数据库，其实就是数据的仓库，称为DataBase，简称DB，指的是将指定的数据按照一定的规则组织起来，长期保存在计算机的存储设备上并能够被各种应用使用的数据集合，其本质上就是数据存储的容器，和普通的数据容器不同的是：数据库是依据数据结构来进行数据存储的，正是因为有着数据结构的存在，数据库中的数据的是比较条理化或者格式化的。市面上的常见的数据库有：Oracle、SQL Server、MySQL、Redis、MongoDB等。

在本系统中主要用到了MySQL和Redis数据库。值得一提的是，这两种数据的差别决定了其不同的用途，首先在类型上MySQL数据库属于关系型数据库，然而Redis属于NoSQL，其次在数据存储位置上有非常大的差别，MySQL数据库将数据存储在硬盘上，然而Redis一般将数据存储于内存中，因此，MySQL一般用于持久化数据，偏向于存数据，然而Redis一般用于对数据进行缓存，偏向于取数据。

2.2.4 uni-app

uni-app是DCloud（数字天堂）公司开发的一套可以使用Vue.js语法的前端框架，开发者只需要编写一套代码，便可以运行和发布到市面上的各种平台：Android、iOS、各种小程序（微信/支付宝/百度/头条/QQ/钉钉）以及H5，因此，对于uni-app框架的产生，极大地降低了开发者的学习成本，一套代码可以多平台转换运行。同时uni-app也引入了众多的第三方接口，如：地图、支付、导航等，极大地增强了开发效率。

3 系统分析

3.1 可行性分析

该房屋租赁系统的可行性分析是在对该业务进行了初步分析之后，同时结合主要采用的技术优缺点、经济成本等因素，最终确定该系统的开发方案的方法。它是更深层次的思考该系统采用技术的适配性和经济成本等，从而确定其技术、经济最优的方案。可行性分析是项目开发过程中必须进行的步骤，决定了整个系统的大体方向和产品质量。本节从技术、经济、操作是哪个方面对房屋租赁系统进行了全面分析。

3.1.1 技术可行性

在对该系统进行了全面分析之后，用Django做后台管理，Flask前台的想法是完全合适和可行的。Django强大后台管理模块可以非常完美的解决后台管理设计繁琐的问题，且其自带的ORM完美的实现了关系型数据库和Django服务器之间的交互；其次，Flask的灵活多变性也相当方便接口的实现，而且，Flask中的sqlAlchemy插件也完美的解决了Flask服务器和数据库之间的交互问题。

3.1.2 经济可行性

经济上考虑的是系统相应的开发成本、维护成本与系统使用中租户和房东之间的关系。该系统开发成本较低，只需要一台云服务器以及短信服务便可以完成项目的开发和部署。另外，该系统的前台前端采用uni-app开发，可以适应更多的平台，极大地降低了开发成本。此外，系统的使用能大大增加房屋成交的机会，极大地提高房东和租客的劳动成本。

3.1.3 操作可行性

由于该系统无论前台还是后台服务，都是具有前端界面和后端服务器的，因此，本系统只需要将该后端Django和Flask服务器成功部署到云服务器中，系统管理员便可以直接通过浏览器访问指定URL就可以访问该后台管理服务，而用户便可以便可以直接安装有uni-app转换来的Apk文件便可以，从而非常方便进行各种前台操作。

3.2 系统需求分析

分析系统需求是一个重要的网络，它基于与系统设计和实现的成功或失败以及软件操作质量相关的分析可能性。分析需求的主要目的是确定系统必须满足的功能模块并反映用户的实际需求，并将此结果用作开发的前提。系统的实施，这对于设计和开发至关重要。连续系统。对系统需求的充分分析可以防止或消除先前的错误，优化程序开发，降低程序开发成本，并提高程序性能。结合对城市管理信息系统现状的基本理解和实践检验，存在以下各种要求。

3.2.1 系统目标

通过根据当地住房市场的研究需求引入诸如Python和数据库之类的技术，租户与各个房屋管理部门之间的关系可以是合理且可预测的，从而提高管理部门管理的效率。一个基于Python的管理信息管理平台，可以轻松提供租赁详细信息，降低运营成本，提高租赁率和这些关键指标的有效性。为了使市场租金变得越来越高效，在此项目中设计和开发了Python管理系统，主要是考虑到现有营销信息管理的后果，其次是管理系统的特点，租用当地房屋并满足操作，直觉，便利，实用性和安全性的要求。该系统的特点主要包括：舒适的界面，良好的后台管理，数据的安全性，快速且易于操作的功能。

3.2.2 系统需求分析

该主题根据完全集成的房屋租赁业务部门，租赁管理部门以及基于Python的租赁系统的设计和开发的需求，提供有关租赁市场的重要信息。它具有后台管理系统和前台显示系统。其中，基础结构管理具有几个主要模块，例如管理模块，角色管理模块，用户管理模块，声明模块，注释管理模块，语句管理模块和模块。模块管理，凭证管理模块等等；前台展示主要包含房源展示模块、快速检索模块、定位模块、收藏模块、聊天模块、用户注册模块、登录模块、订单查询模块、升级房东模块、实名认证模块等。下面简要说明各个功能模块的具体需求。

（1）后台管理系统

管理员登录模块的功能是检验登录该后台管理系统的管理员是否合法。每个管理员都有自己的角色，角色主要有：超级管理员、审核管理员和赞助管理员。当用户输入用户名和密码后，如果该用户的身份合法，则可以经过确认后直接进入房屋租赁系统的后台管理系统，并且不同角色的呈现的界面是不一样的。如果该登录的信息是非法的，则系统拒绝该用户的进入。对于超级管理员，可以进行角色的创建、编辑和删除，可以创建不同角色的管理员，同时，超级管理员也是可以进行公告的发布、查看和删除；审核管理员可以对房屋的信息进行审核，其中包括是否实名认证，如果审核通过，则可以在前台进行显示，也可以对公告信息进行审核，通过则任何管理员都可以直接看到该公告信息；此外，赞助管理员则可以直接对代金券和轮播图进行管理，添加的代金券和轮播图可以直接在前台展示。

（2）前台展示系统

在前台展示系统中，首页可以进行轮播图和房源信息的展示，同时在首页中可以通过关键字进行房源搜索，同时，首页中也可以进行定位操作，在首页中也可以进行房源信息的收藏，同时点击对应的房源信息，会跳转到详情页面，可以对该房源进行收藏和预定，其中预定可以采用支付宝和微信支付；在收藏模块中，可以查看当前账户收藏的房源信息，如果还未登录会跳转到登录页面，在登录页面中，输入用户名和密码，如果验证通过，则可以登录成功，返回到首页页面，同时在收藏页面中，也是可以进行预定的；在聊天页面中，可以列出最近30天的聊天记录，点击任意一个聊天记录，便可以打开一个临时会话页面；在我的页面中，除了可以进行基本地用户注册、登录，还可以进行当前用户的实名认证，之后便可以通过后台进行审核，当然，我的页面中，也可以查询订单，点击查询订单便可以将当前用户所有的订单信息列出来，同时，对用普通租客，在我的页面中也可以升级为房东。当然是建立在已经进行实名认证的前提下。

3.3 业务分析

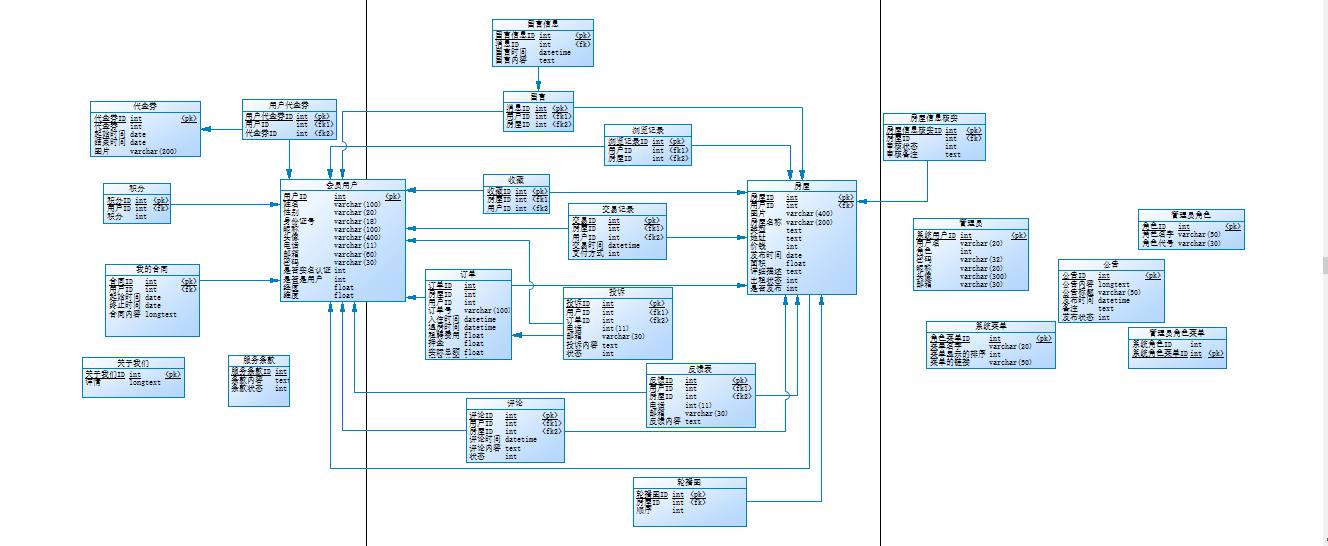
3.3.1 用户群体

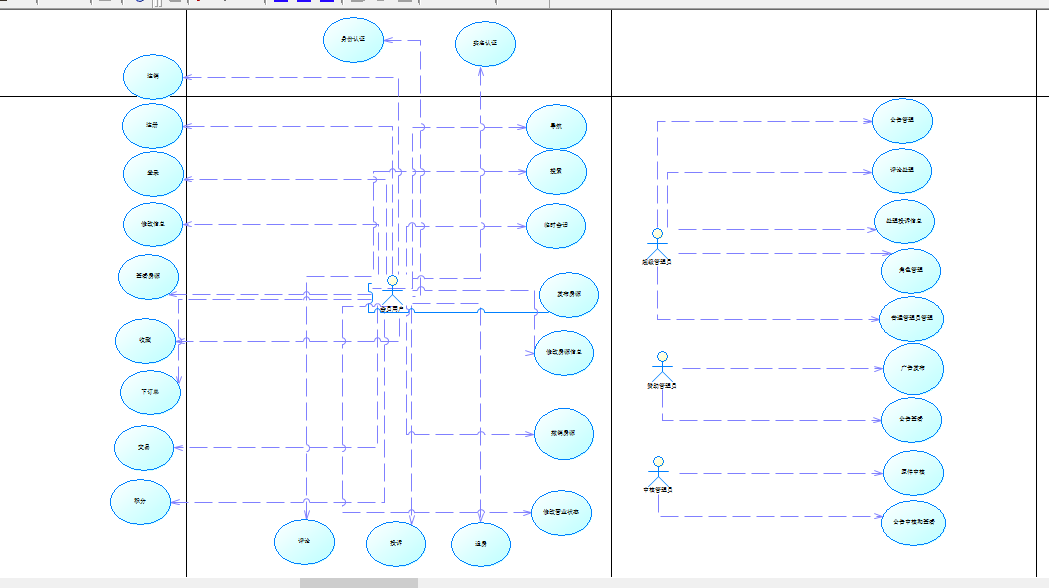
该系统的在功能上主要分前台和后台，前台针对的群体主要是普通租客和房东，而后台主要针对的群体则是超级管理员、审核管理员和赞助管理员。该系统前台将要展示的房源信息、广告等都是需要经过后台管理员的审核的，同时在前台发布的评论、投诉等，也都是需要经过后台管理员处理。

因此，前台和后台的关系是相当紧密的，后台管理员对数据的处理结果将会直接影响前台用户的体验。

3.3.2 业务模型图和模型表建立

对该系统进行了全面的业务分析之后，根据前台和后台所针对的群体为出发点，通过PowerDesigner对该分析结果进行具体化，首先画出了该系统的业务用例图（如图3-1所示），在该业务用例图中详细地描述了每个群体的所具有的属性和动作，并根据该业务分析图画出了该系统的物理模型图（如图3-2所示），在该物理模型图中，具备了该业务中所需要的所有表信息，同时，也可以非常直观的看到每个表之间联系，对该系统之后的开发起着决定性的作用。

图 3-1

图 3-2

4 系统的设计与实现

在系统设计中，究其本质，任何的操作其实都是建立在对数据库的增加、删除、修改和查询的基础上的，本文分后台管理系统和前台操作系统来进行详细阐述。

4.1 后台管理系统

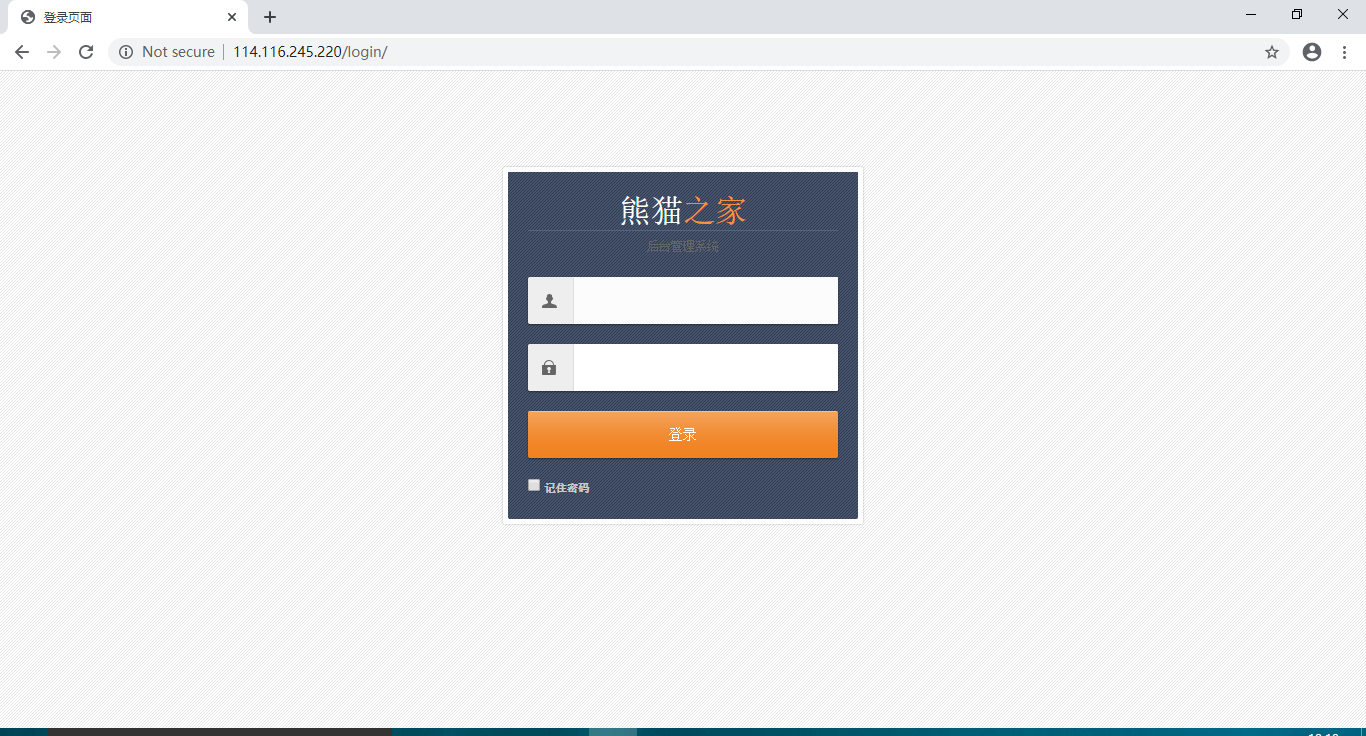
在后台管理系统中整体采用了Django的表单验证功能，从前端向后端传入相应的表单数据，并对该表单数据进行验证，最终进行相应的操作或跳转。

4.1.1 登录界面

在前端通过表单向后端传入用户名和密码，通过后端进行数据的获取和验证，如果传入数据和数据库匹配的话，则登录成功，否则，登陆失败，通过Ajax弹出提醒信息。

同时，如果后端验证登录成功，那么后端会通过Session将登录用户的信息进行保存，方便后续前后端全局进行调用，值得一提的是，在后续的超级管理员、审核管理员、赞助管理员的头像、用户名等信息都是可以通过Session来获取，极大地增强了系统的开发的灵活性。

打开浏览器，输入该系统后台管理登录地址，即出现如图4-1所示的登录界面：

图 4-1

如果己经在本系统的后台系统中已经注册为用户，只需输入用户名与密码点击“登录”即可进入后台管理系统。

如果不是本系统的用户，请联系系统超级管理员进行系统用户注册，然后才能登录本后台系统。详情请见4.1.2节超级管理员详细介绍。

4.1.2 超级管理员

超级管理员的权限有角色管理、系统用户管理、评论管理、投诉管理以及公告的发布和删除，但却并不能进行公告的审核。其中的角色管理、系统用户管理、评论管理、投诉管理主要就是对数据库中的相应表进行增删改查，然而，对于超级管理员，只能对公告表进行增加和删除的操作。

在浏览器中通过登录超级管理员之后，可以看到如图4-2所示界面：

图 4-2

对于超级管理员的角色管理和系统用户管理员都采用相同的思想，向角色管理接口或者系统管理员接口发送GET请求，可以获取到角色或者用户的详细信息；发送POST请求时，可以通过Django自带的form表单或者携带相应的参数，在前端通过判断携带的参数，从而判断该页面都到底是新增页面或者是编辑页面，同时对于新增、编辑公告页面，也引入富文本插件，从而达到多功能输入的目的，如图4-3所示。

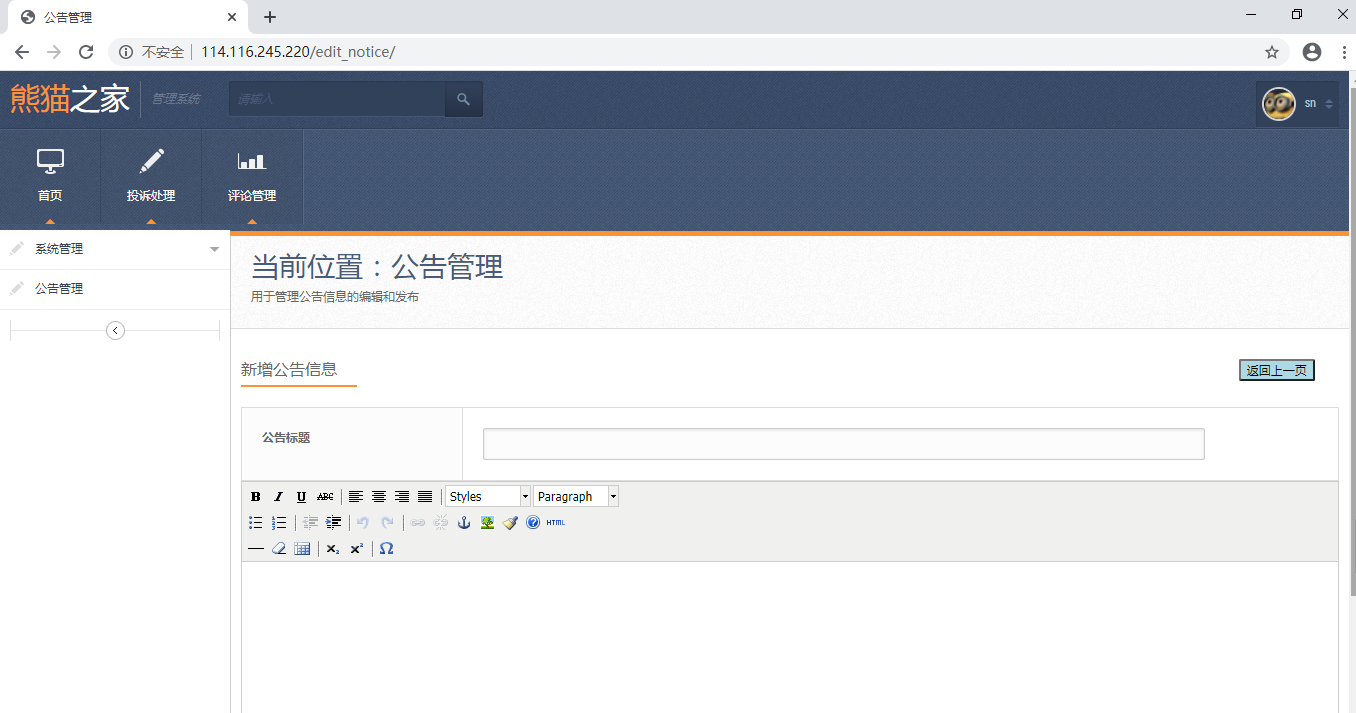


图4-3

4.1.3 审核管理员

对于审核管理员，其权限完全是和各种信息审核相关了，有房屋审核、公告审核，同时也可以查看审核已经通过的公告信息。房屋审核和公告审核其实就是更改数据库中房屋表和公告表的某一字段值。

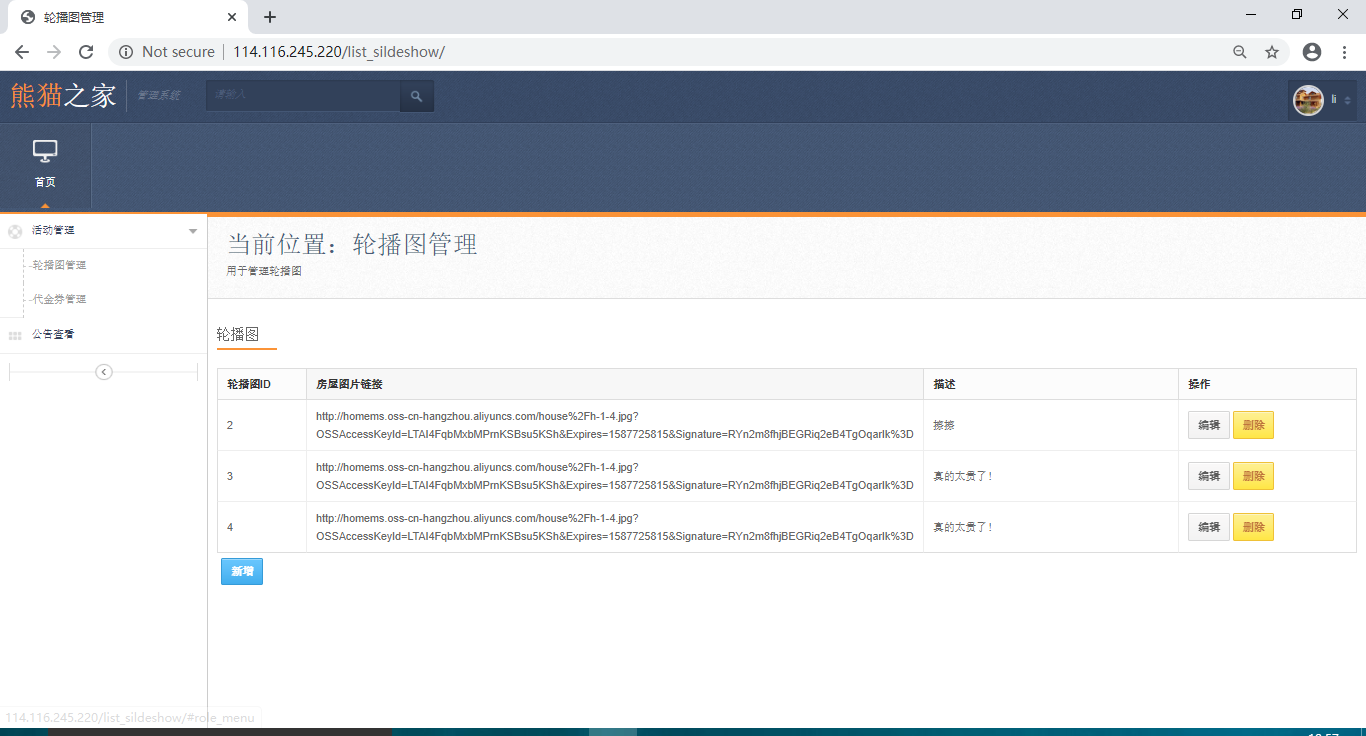
如果通过登录界面登录的是审核管理员，成功之后，便会到达如图4-4所示界面：

图 4-4

在房屋审核的过程中，如果点击审核不通过，那么采用前端ajax技术，界面会弹出一个对话窗，可以填写未通过的理由，同时将数据库中房屋标的审核字段的值设置为0；如果审核通过的话，可该数据库中房屋表的审核状态字段值为1。

4.1.4 赞助管理员

赞助管理员，主要负责的便是代金券和广告的管理，商家可以通过线下的方式与赞助管理员管理员联系，最终将商品以轮播图的形式展示给用户，达到广告的效应，同时，也可以商家也可以发布一些代金券，如图4-5所示：

图 4-5

由上图可以非常清楚的看到，赞助管理员的可以非常方便的对广告信息进行管理，极大地保障了系统整个的良好环境和商业效应。考虑到系统中随着使用量的增大，日后广告的数量也将更大，因此对于广告的图片管理将会是一个非常巨大的问题。在本系统中，采用了阿里云服务的OSS服务，在数据库中存储的只是一个广告图片的键，通过这个键并进行相应的OSS验证，便可以获取到相应广告图片的具体链接，为后续前台的图片的处理也减少了很多麻烦。

4.2前台操作系统

4.2.1 首页模块

对于一个系统，首页的信息展示是必不可少的元素，在该系统中，当广告信息通过后台管理员审核通过之后，通过轮播图的形式展示了出来，从而达到推销的目的，如图；同时在首页中，用户可以清楚的看到每个已通过后台审核的房源的大致信息，比如地段、价格、面积等，如图4-6：

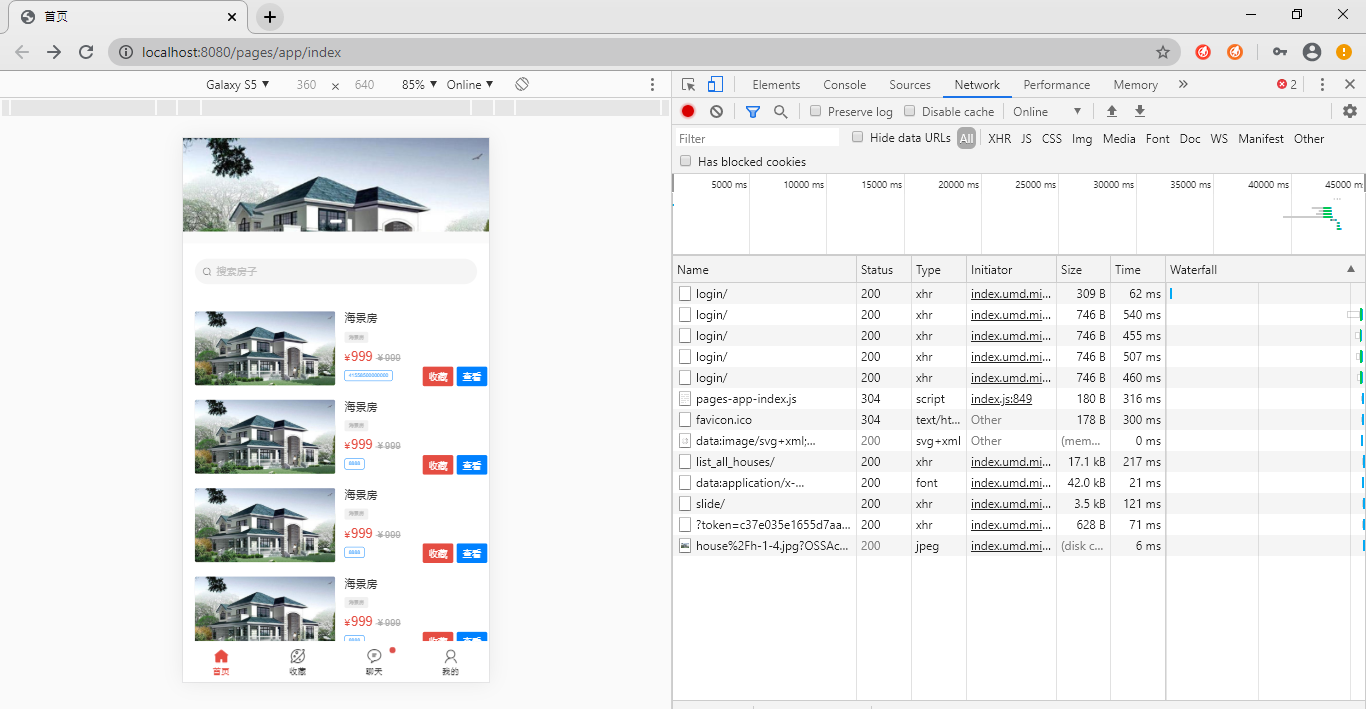


图 4-6

当然，首页的广告和房源信息都是可以点击，从而进入到房源的详情页面，这里主要是将房屋的house\_id作为请求参数，在详情页面变通过接收到house\_id将房屋的详细信息查取出来。详情页面如图4-7：

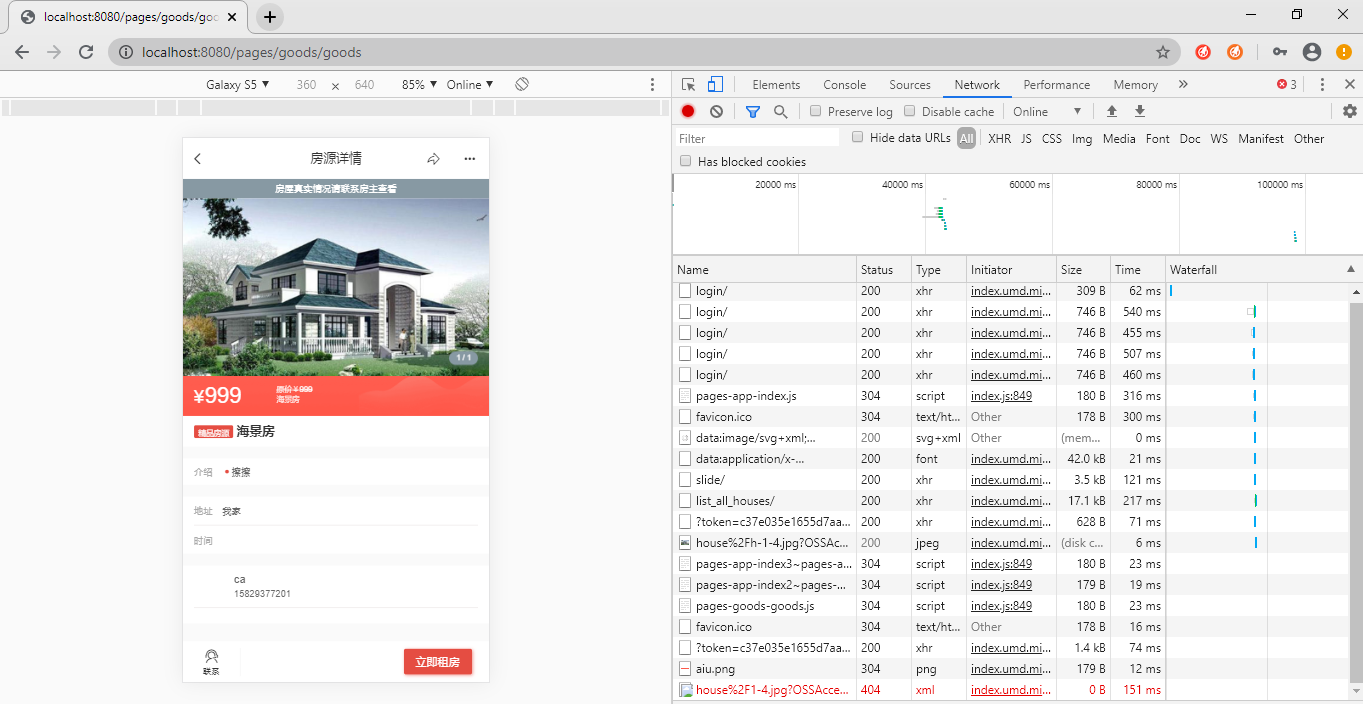


图 4-7

在详情页面中，用户可以对该房屋进行预订，当点击预订按钮时，可以跳转到支付宝支付的页面，并完成相应的支付功能。

4.2.2 收藏模块

每个用户都可收藏页面，可将自己喜欢的房源信息收藏起来，对于每个用户所收藏的房源都可以通过该系统的收藏模块房屋信息展示出来，如图4-7所示：

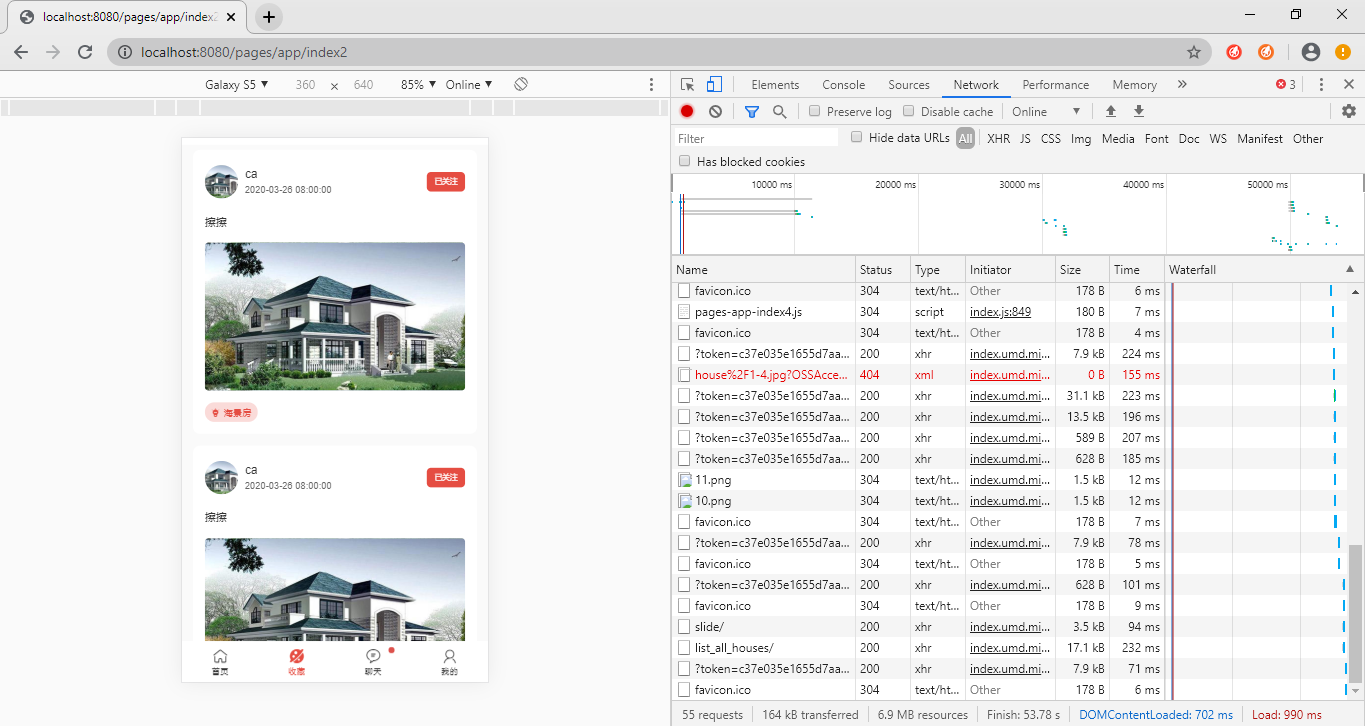


图 4-8

在该页面中，用户可以通过点击具体的收藏房屋信息，进而跳转到租赁详情页面并完成相应的下单操作。

4.2.3 聊天模块

通过聊天模块，用户可与房东进行线上交流，在本系统中采用的是UDP协议，聊天服务是建立在非连接的基础上完成，无论房东或者租客是否在线，都可以向对方发送消息，通过Socket编程完成服务端、客户端的创建，便可完成在非连接的基础上，实现消息通信的目的，聊天界面如图4-9所示：

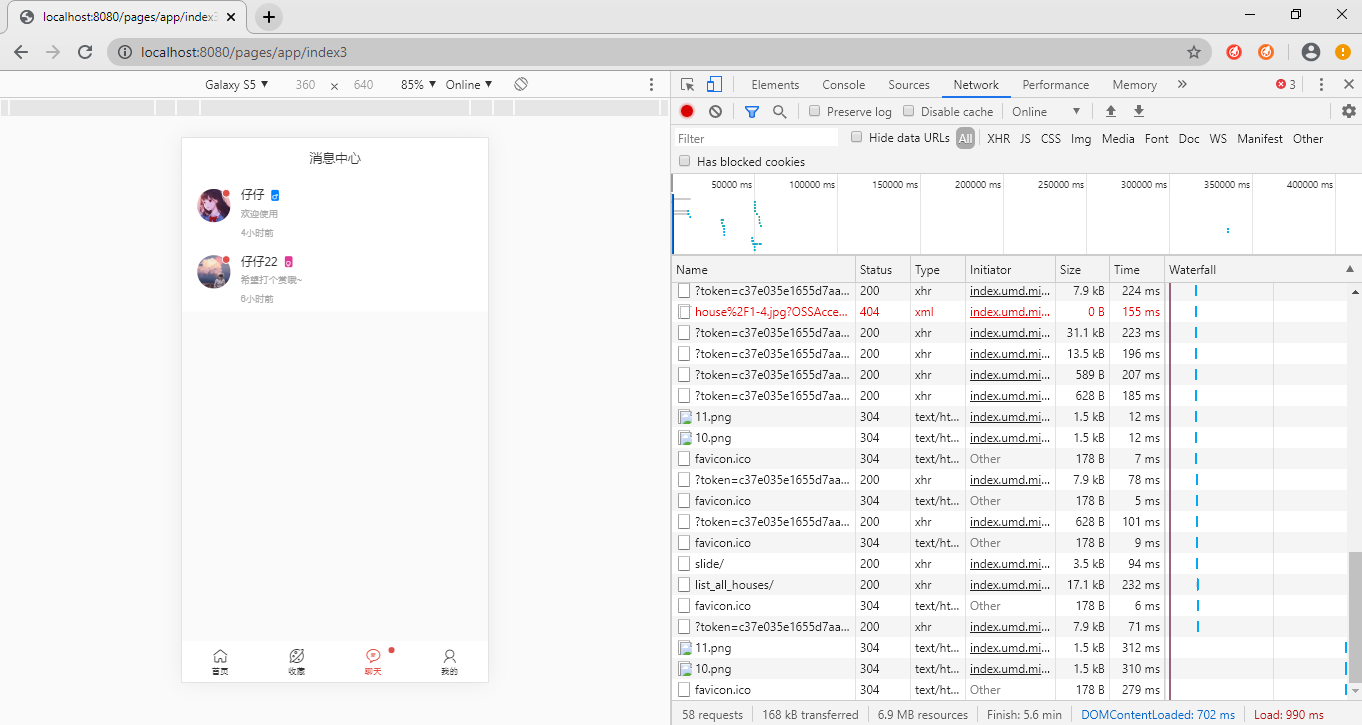


图 4-9

在该系统中，服务端绑定了指定的IP和端口，多个客户端可向该服务端发送数据，在客户端和服务端都可以进行数据的接收和发送，同时，在该服务端中，可通过recvfrom

方法获取到从客户端发送过来的数据库和对应的IP，拿到了客户端的IP，便可以向客户端发送消息。

4.2.4 我的模块

在我的模块里面，用户可以进行登录、注册、实名认证、订单查询、以及优惠券查询等。这些操作本质上都是对数据表的增加、删除、修改、查询操作，其中实名认证引入了阿里云的身份认证服务，只需要向该接口提供相应身份证号和姓名，如果身份证号合法且可以和姓名对应，将可以返回该用户的详细信息；订单查询和代金券查询需要携带token，利用Flask的flask-sqlalchemy插件对数据库进行查询。实现界面如图4-10：

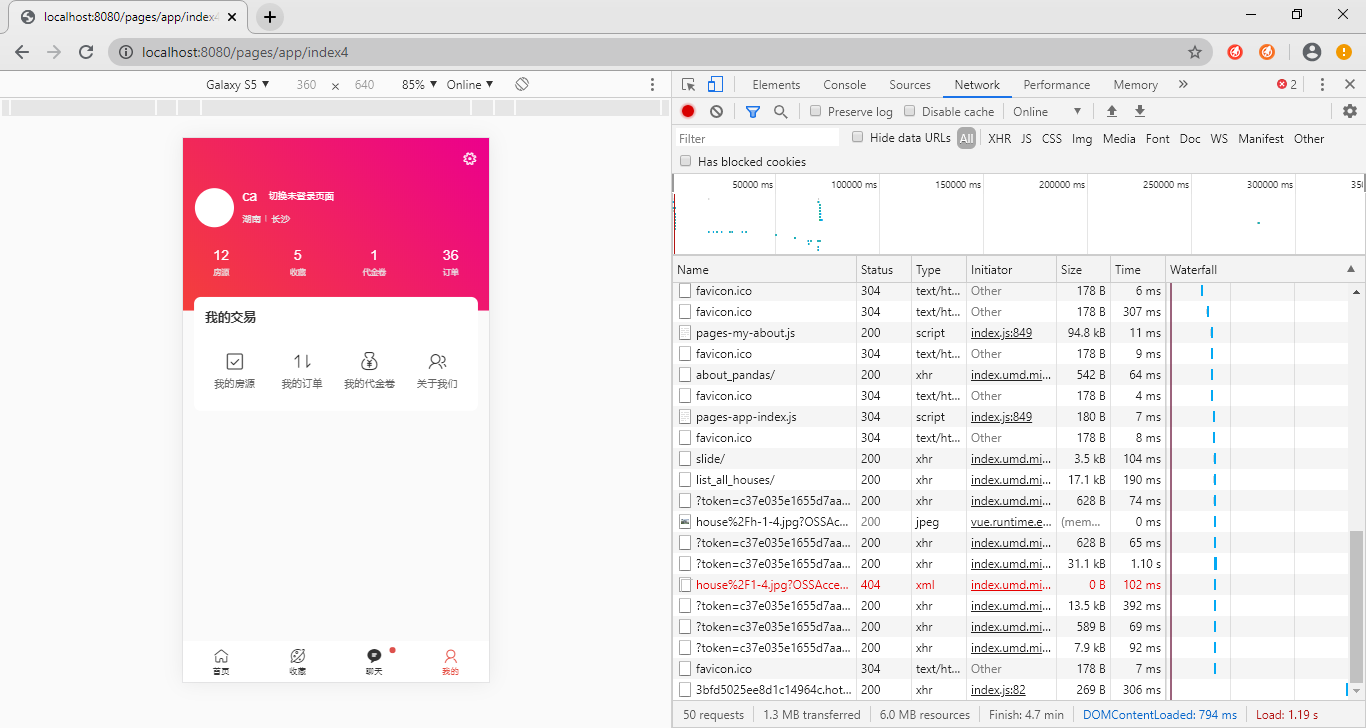


图 4-10

5 测试

在开发一套系统时，选择合理的测试方式是相当重要的，一套合理的测试方式，在API接口开发的过程，将更加方便、直接地判断接口开发中是否存在各种问题，并及时定位BUG出现的位置。一般系统测试分为黑盒测试和白盒测试，其中黑盒测试主要是针对功能和性能直接进行测试，不需要了解系统内部设计情况，然而白盒测试需对该系统内部结构有较深入的了解，会对该系统进行更加全面的测试。在该系统的前台中主要采用了白盒测试，测试的接口有31个，如：登录、注册、修改密码、发送验证码、身份验证、房源发布、房源展示、房源详情、收藏房屋、取消收藏和订单查询等。

在该系统中，主要使用requests库和unittest库进行测试，通过requests库可以向接口发送GET、POST、PUT、DELETE等各种请求，同时携带接口请求所需要的请求参数或表单数据，而unittest库则是Python中相当专业的一个测试库，可以对各个接口进行分模块测试，并且最终得到测试详细信息。例如登录接口测试，接口请求路由为/api/login/，请求方式为POST请求，可通过requests.post()方法来模拟浏览器进行接口测试。

总结

本文在进行了认真的市场需求调研和选用最合适的开发技术的基础上，分别从系统整体设计目标、模块、功能、业务流程等几个方面考虑，设计出了一套开发成本最低的设计框架。设计并开发了一套基于Python的房屋租赁管理系统。本文所做的主要工作如下：

分析和研究了我国房屋租赁市场的需求和研究现状，结合目前市面上已经较为成熟的房屋租赁平台和失败的案例，制定出了最为合适的开发思路。

更加深入研究和分析了Python各个Web框架的优缺点以及适用场景，选择了最合适的技术处理相应的问题。

本系统在大的方向上，主要分为2个设计方向：前台系统和后台系统，同时，无论是前台和后台系统都包括有客户端和服务端。在此基础上再结合数据库技术开发出来的房屋租赁系统在功能上将更加的完善，并且，在用户使用上，使用感受也将更加的轻松，最重要的是，在房屋信息的管理上，也更加的安全和方便。

该系统一大特色就是，前台完全采用uni-app小程序开发，可以适配各种移动端和PC端，实用性得到了极大的提高，而且也达到了“秒开”的效果。并且极大地节省了开发的成本。

参考文献

[1]魏志军. 高校实验室管理平台设计与实现[J]. 中国优秀硕士学位论文全文数据库, 2016(02):66-66.

[2]黄学波 梁俊卿. 基于BIM消防的房屋租赁系统的研究与实现[J]. 计算机科学与应用, 2018(5):10-10.

[3]李瑞娥,沈文枫,杨宇华,周丽萍,王洋.基于区块链的房屋租赁分配系统设计与实现[J].软件导刊,2019,18(10):111-116.

[4]张基.基于Android平台的房屋租赁管理系统的设计[J].老字号品牌营销,2019(07):70-71.

[5]朱园园,邵雯娟,黄志远.基于Scrapy的房屋租赁信息搜索系统设计与实现[J].电脑与电信,2019(06):14-19.

[6]李佳雯. 基于区块链技术的房屋租赁平台设计与实现[D].济南大学,2019.

[7]罗茜. 基于微服务的房屋租赁服务系统的设计与实现[D].东南大学,2019.

[8]武海龙,李国平.基于SpringBoot的房屋租赁系统设计[J].电脑与信息技术,2019,27(03):76-78.

[9]胡世民. 基于互联网+的房屋租赁管理系统的设计与实现[D].江西财经大学,2018.

[10]王飞.论房屋租赁信息系统的设计与实现[J].电脑迷,2017(07):194-195.

附录

**后台系统代码**

**models.py**

*# This is an auto-generated Django model module.  
# You'll have to do the following manually to clean this up:  
# \* Rearrange models' order  
# \* Make sure each model has one field with primary\_key=True  
# \* Make sure each ForeignKey has `on\_delete` set to the desired behavior.  
# \* Remove `managed = False` lines if you wish to allow Django to create, modify, and delete the table  
# Feel free to rename the models, but don't rename db\_table values or field names.***from** datetime **import** datetime  
  
**from** django.db **import** models  
**from** tinymce.models **import** HTMLField  
  
**from** common **import** md5\_  
  
**class** TUser(models.Model):  
 user\_id = models.AutoField(primary\_key=**True**)  
 name = models.CharField(max\_length=100, blank=**True**, null=**True**)  
 sex = models.CharField(max\_length=20, blank=**True**, null=**True**)  
 identity\_number = models.CharField(max\_length=18, blank=**True**, null=**True**)  
 nickname = models.CharField(max\_length=100, blank=**True**, null=**True**)  
 img = models.CharField(max\_length=200, blank=**True**, null=**True**)  
 phone = models.CharField(max\_length=11, blank=**True**, null=**True**)  
 email = models.CharField(max\_length=60, blank=**True**, null=**True**)  
 password = models.CharField(max\_length=30, blank=**True**, null=**True**)  
 has\_real\_name = models.IntegerField(blank=**True**, null=**True**)  
 is\_member = models.IntegerField(blank=**True**, null=**True**)  
 longitude = models.FloatField(blank=**True**, null=**True**)  
 dimension = models.FloatField(blank=**True**, null=**True**)  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_user'  
  
class** Contract(models.Model): *# 合同模型* t\_contract\_id = models.AutoField(primary\_key=**True**)  
 user = models.ForeignKey(**'TUser'**, models.DO\_NOTHING, blank=**True**, null=**True**)  
 start\_time = models.DateField(blank=**True**, null=**True**)  
 stop\_time = models.DateField(blank=**True**, null=**True**)  
 content = models.TextField(blank=**True**, null=**True**)  
  
 **class** Meta:  
 managed = **False** db\_table = **'contract'  
  
  
class** TBrowsingHistory(models.Model):  
 browsing\_history\_id = models.AutoField(primary\_key=**True**)  
 usert = models.ForeignKey(**'TUser'**, models.DO\_NOTHING, blank=**True**, null=**True**)  
 house = models.ForeignKey(**'THouse'**, models.DO\_NOTHING, blank=**True**, null=**True**)  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_browsing\_history'  
  
  
class** TChatmsg(models.Model): *# 留言* chatmsg\_id = models.AutoField(primary\_key=**True**)  
 msg = models.ForeignKey(**'TMessage'**, models.DO\_NOTHING, blank=**True**, null=**True**)  
 chatmsg\_time = models.DateTimeField(blank=**True**, null=**True**)  
 content = models.TextField(blank=**True**, null=**True**)  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_chatmsg'  
  
  
class** TComment(models.Model): *#* comment\_id = models.AutoField(primary\_key=**True**)  
 user = models.ForeignKey(**'TUser'**, models.DO\_NOTHING, blank=**True**, null=**True**)  
 house = models.ForeignKey(**'THouse'**, models.DO\_NOTHING, blank=**True**, null=**True**)  
 create\_time = models.DateTimeField(blank=**True**, null=**True**)  
 content = models.TextField(blank=**True**, null=**True**)  
  
 states = (  
 (0, **'审核中'**),  
 (1, **'已通过'**),  
 (2, **'未通过'**),  
 )  
 state = models.IntegerField(blank=**True**, default=0)  
  
 @property  
 **def** state\_label(self):  
 **return** self.states[self.state][-1]  
  
 **def** save(self, force\_insert=**False**, force\_update=**False**, using=**None**,  
 update\_fields=**None**):  
 **if** self.create\_time **is None**:  
 self.create\_time = datetime.now()  
  
 super(TComment, self).save()  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_comment'  
  
class** TComplaint(models.Model):  
 fadeback\_id = models.AutoField(primary\_key=**True**)  
 user = models.ForeignKey(**'TUser'**, models.DO\_NOTHING, blank=**True**, null=**True**)  
 order = models.ForeignKey(**'TOrder'**, models.DO\_NOTHING, blank=**True**, null=**True**)  
 phone = models.IntegerField(blank=**True**, null=**True**)  
 email = models.CharField(max\_length=30, blank=**True**, null=**True**)  
 content = models.TextField(blank=**True**, null=**True**)  
  
 states = (  
 (0, **'审核中'**),  
 (1, **'已通过'**),  
 (2, **'未通过'**),  
 )  
 state = models.IntegerField(blank=**True**, default=0)  
  
 @property  
 **def** state\_label(self):  
 **return** self.states[self.state][-1]  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_complaint'  
  
  
class** TFavorite(models.Model):  
 favorite\_id = models.AutoField(primary\_key=**True**)  
 house = models.ForeignKey(**'THouse'**, models.DO\_NOTHING, blank=**True**, null=**True**)  
 user = models.ForeignKey(**'TUser'**, models.DO\_NOTHING, blank=**True**, null=**True**)  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_favorite'  
  
  
class** TFeedback(models.Model):  
 fadeback\_id = models.AutoField(primary\_key=**True**)  
 user = models.ForeignKey(**'TUser'**, models.DO\_NOTHING, blank=**True**, null=**True**)  
 house = models.ForeignKey(**'THouse'**, models.DO\_NOTHING, blank=**True**, null=**True**)  
 phone = models.IntegerField(blank=**True**, null=**True**)  
 email = models.CharField(max\_length=30, blank=**True**, null=**True**)  
 content = models.TextField(blank=**True**, null=**True**)  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_feedback'  
  
  
  
class** THouse(models.Model):  
 house\_id = models.AutoField(primary\_key=**True**)  
 user = models.ForeignKey(TUser, models.DO\_NOTHING, blank=**True**, null=**True**)  
 image = models.CharField(max\_length=200, blank=**True**, null=**True**)  
 *# image = models.ImageField()* name = models.CharField(max\_length=200, blank=**True**, null=**True**)  
 type = models.TextField(blank=**True**, null=**True**)  
 address = models.TextField(blank=**True**, null=**True**)  
 price = models.IntegerField(blank=**True**, null=**True**)  
 publish\_time = models.DateField(blank=**True**, null=**True**)  
 area = models.FloatField(blank=**True**, null=**True**)  
 description = models.TextField(blank=**True**, null=**True**)  
 sale\_status = models.IntegerField(blank=**True**, null=**True**)  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_house'  
  
  
  
class** THouseVerify(models.Model): *# 房屋审核模型* verify\_id = models.AutoField(primary\_key=**True**)  
 house = models.ForeignKey(THouse, models.DO\_NOTHING, blank=**True**, null=**True**)  
 *# verify\_status = models.IntegerField(blank=True, null=True)* remarks = models.TextField(blank=**True**, null=**True**)  
 states = (  
 (0, **'审核中'**),  
 (1, **'已通过'**),  
 (2, **'未通过'**)  
 )  
 verify\_status = models.IntegerField(choices=states, default=0)  
  
 @property  
 **def** state\_label(self):  
 **return** self.states[self.verify\_status][-1]  
  
 **def** save(self, force\_insert=**False**, force\_update=**False**, using=**None**,  
 update\_fields=**None**):  
 *# if self.create\_time is None:  
 # self.create\_time = datetime.now()* super(THouseVerify, self).save()  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_house\_verify'  
  
  
class** TLuckyTicket(models.Model):  
 lucky\_ticket\_id = models.AutoField(primary\_key=**True**)  
 money = models.IntegerField(blank=**True**, null=**True**)  
 begin\_time = models.DateField(blank=**True**, null=**True**)  
 end\_time = models.DateField(blank=**True**, null=**True**)  
 image = models.CharField(max\_length=20, blank=**True**, null=**True**)  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_lucky\_ticket'  
  
class** TMessage(models.Model):  
 msg\_id = models.AutoField(primary\_key=**True**)  
 user = models.ForeignKey(**'TUser'**, models.DO\_NOTHING, blank=**True**, null=**True**)  
 house = models.ForeignKey(THouse, models.DO\_NOTHING, blank=**True**, null=**True**)  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_message'  
  
  
class** TOrder(models.Model):  
 order\_id = models.AutoField(primary\_key=**True**)  
 house = models.ForeignKey(THouse, models.DO\_NOTHING, blank=**True**, null=**True**)  
 user = models.ForeignKey(**'TUser'**, models.DO\_NOTHING, blank=**True**, null=**True**)  
 order\_number = models.CharField(max\_length=16, blank=**True**, null=**True**)  
 enter\_time = models.TimeField(blank=**True**, null=**True**)  
 exit\_time = models.TimeField(blank=**True**, null=**True**)  
 hire\_price = models.FloatField(blank=**True**, null=**True**)  
 cash\_price = models.FloatField(blank=**True**, null=**True**)  
 total = models.FloatField(blank=**True**, null=**True**)  
 order\_status = models.IntegerField(blank=**True**, null=**True**)  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_order'  
  
  
class** TPanda(models.Model):  
 panda\_id = models.AutoField(primary\_key=**True**)  
 detail\_content = models.TextField(blank=**True**, null=**True**)  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_panda'  
  
  
class** TPublicNotice(models.Model):  
 public\_notice\_id = models.AutoField(primary\_key=**True**)  
 content = HTMLField(null=**True**)  
 public\_title = models.CharField(max\_length=50, null=**True**)  
 public\_time = models.DateTimeField(auto\_created=**True**, blank=**True**)  
 public\_remarks = models.TextField(null=**True**, blank=**True**)  
  
 states = (  
 (0, **'待审核'**),  
 (1, **'已通过'**),  
 (2, **'未通过'**)  
 )  
 public\_status = models.IntegerField(choices=states, default=0)  
  
 @property  
 **def** state\_label(self):  
 **return** self.states[self.public\_status][-1]  
  
 **def** save(self, force\_insert=**False**, force\_update=**False**, using=**None**,  
 update\_fields=**None**):  
  
 **if** self.public\_time **is None**:  
 self.public\_time = datetime.now()  
  
 super(TPublicNotice, self).save()  
  
 **class** Meta:  
 *# managed = False* db\_table = **'t\_public\_notice'** ordering = [**'-public\_time'**]  
  
  
**class** TScore(models.Model):  
 score\_id = models.AutoField(primary\_key=**True**)  
 user = models.ForeignKey(**'TUser'**, models.DO\_NOTHING, blank=**True**, null=**True**)  
 score = models.IntegerField(blank=**True**, null=**True**)  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_score'  
  
  
class** TService(models.Model):  
 service\_id = models.AutoField(primary\_key=**True**)  
 content = models.TextField(blank=**True**, null=**True**)  
 service\_status = models.IntegerField(blank=**True**, null=**True**)  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_service'  
  
  
class** TSlidesshow(models.Model):  
 slidesshow\_id = models.AutoField(primary\_key=**True**)  
 house = models.ForeignKey(THouse, models.DO\_NOTHING, blank=**True**, null=**True**)  
 ord = models.IntegerField()  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_slidesshow'  
  
  
class** TSysMenu(models.Model):  
 menu\_id = models.AutoField(primary\_key=**True**)  
 menu\_name = models.CharField(max\_length=20, blank=**True**, null=**True**)  
 ord = models.IntegerField(blank=**True**, null=**True**)  
 url = models.CharField(max\_length=50, blank=**True**, null=**True**)  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_sys\_menu'  
  
  
class** TSysRole(models.Model):  
 role\_id = models.AutoField(primary\_key=**True**)  
 role\_name = models.CharField(max\_length=20)  
 role\_code = models.CharField(max\_length=10)  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_sys\_role'  
  
  
class** TSysRoleMenu(models.Model):  
 role\_id = models.IntegerField(blank=**True**, null=**True**)  
 sys\_menu\_id = models.AutoField(primary\_key=**True**)  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_sys\_role\_menu'  
  
  
class** TSysUser(models.Model):  
 user\_id = models.AutoField(primary\_key=**True**)  
 username = models.CharField(max\_length=20, blank=**True**, null=**True**)  
 role\_id = models.IntegerField(blank=**True**, null=**True**)  
 password = models.CharField(max\_length=32, blank=**True**, null=**True**)  
 nick\_name = models.CharField(max\_length=20, blank=**True**, null=**True**)  
 head = models.ImageField(upload\_to=**'upload\_head/%Y/%m'**, null=**True**, blank=**True**)  
 *# head = models.CharField(max\_length=300, null=True, blank=True)* email = models.CharField(max\_length=30, blank=**True**, null=**True**)  
  
 @property  
 **def** role(self):  
 **return** TSysRole.objects.get(role\_id=self.role\_id)  
  
 **def** save(self, force\_insert=**False**, force\_update=**False**, using=**None**,  
 update\_fields=**None**):  
  
 **if** len(self.password) != 32:  
 self.password = md5\_.hash\_encode(self.password)  
  
 super(TSysUser, self).save()  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_sys\_user'  
  
  
class** TTradingrecord(models.Model):  
 tradingrecord\_id = models.AutoField(primary\_key=**True**)  
 house = models.ForeignKey(THouse, models.DO\_NOTHING, blank=**True**, null=**True**)  
 user = models.ForeignKey(**'TUser'**, models.DO\_NOTHING, blank=**True**, null=**True**)  
 payment\_date = models.DateTimeField(blank=**True**, null=**True**)  
 payment\_type = models.IntegerField(blank=**True**, null=**True**)  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_tradingrecord'  
  
  
class** TULuckyTicket(models.Model):  
 u\_lucky\_ticketid = models.AutoField(primary\_key=**True**)  
 user = models.ForeignKey(**'TUser'**, models.DO\_NOTHING, blank=**True**, null=**True**)  
 lucky\_ticket = models.ForeignKey(TLuckyTicket, models.DO\_NOTHING, blank=**True**, null=**True**)  
  
 **class** Meta:  
 managed = **False** db\_table = **'t\_u\_lucky\_ticket'**

**views.py**

**import** json  
  
**from** django.core.serializers **import** serialize  
**from** django.db.models **import** Q  
**from** django.http **import** HttpResponse  
**from** django.shortcuts **import** render, redirect  
**from** django.urls **import** reverse  
**from** django.views **import** View  
  
**from** common **import** md5\_  
**from** . **import** get\_house\_image, get\_user\_image  
**from** .models **import** TSysRole, TSysUser, TPublicNotice, TSlidesshow, TLuckyTicket, THouseVerify  
  
  
*# Create your views here.***def** login(request):  
 *# 分两种用户，一个是会员，一个管理员（系统）  
 # print('--->', request.method)* **if** request.method == **'POST'**:  
 *# print(request.POST)* error = **None** username = request.POST[**'username'**].strip()  
 password = request.POST[**'password'**].strip()  
 remeber = request.POST.get(**'remeber'**, **''**) *# checkbox* password\_ = md5\_.hash\_encode(password) *# 转成md5后的密文* print(password\_)  
  
 *# 验证用户名和口令是否为空* **if not** all((username, password)):  
 error = **f'用户名或口令不能为空！'  
  
 else**:  
 login\_user = TSysUser.objects.filter(username=username, password=password\_).first()  
 **if** login\_user:  
 *# 超级管理员* role\_ = login\_user.role  
  
 login\_info = {  
 **'user\_id'**: login\_user.user\_id,  
 **'head'**: str(login\_user.head),  
 **'email'**: login\_user.email,  
 **'nick\_name'**: login\_user.nick\_name,  
 **'role\_name'**: role\_.role\_name,  
 **'role\_code'**: role\_.role\_code,  
 }  
  
 **else**:  
 error = **f'{**username**} 用户名或口令错误！'  
  
 if not** error:  
 request.session[**'login\_user'**] = login\_info  
 **return** redirect(reverse(**'main:dash'**))  
 **return** render(request, **'login.html'**, locals())  
  
  
**def** dashboard(request):  
 *"""  
 登陆之后主页界面* **:param** *request:* **:return***:  
 """* notices = TPublicNotice.objects.filter(public\_status=1)  
 **return** render(request, **'dashboard.html'**, locals())  
  
  
**def** role(request):  
 *"""  
 查看角色页面和删除角色* **:param** *request:* **:return***:  
 """* action = request.GET.get(**'action'**, **''**)  
 **if** action == **'del'**:  
 role\_id = request.GET.get(**'role\_id'**)  
 TSysRole.objects.get(role\_id=role\_id).delete()  
 user\_list = TSysUser.objects.filter(role\_id=role\_id)  
 **for** obj **in** user\_list:  
 obj.delete()  
  
 roles = TSysRole.objects.all()  
 **return** render(request, **'role/list.html'**, locals()) *#***def** list\_sys\_user(request):  
 *"""  
 查看所有管理员和删除管理员* **:param** *request:* **:return***:  
 """* action = request.GET.get(**'action'**, **''**)  
 **if** action == **'del'**:  
 TSysUser.objects.get(user\_id=request.GET.get(**'user\_id'**)).delete()  
  
 *# 查询系统时，除去超级管理员的用户* users = TSysUser.objects.filter(~Q(user\_id=request.session[**'login\_user'**][**'user\_id'**])).all()  
 **return** render(request, **'sys\_user/list.html'**, locals())  
  
  
**class** EditRoleView(View): *# 编辑角色信息  
 """  
 编辑角色信息和增加角色信息  
 """* **def** get(self, request):  
 role\_id = request.GET.get(**'role\_id'**, **''**)  
 **if** role\_id:  
 role = TSysRole.objects.get(role\_id=role\_id)  
 **return** render(request, **'role/edit.html'**, locals())  
  
 **def** post(self, request):  
 **from** .forms **import** RoleForm  
 role\_id = request.POST.get(**'role\_id'**, **''**)  
 **if** role\_id:  
 form = RoleForm(request.POST, instance=TSysRole.objects.get(role\_id=role\_id))  
 **else**:  
 form = RoleForm(request.POST)  
 **if** form.is\_valid():  
 form.save()  
 **return** redirect(reverse(**'main:role'**))  
  
 errors = json.loads(form.errors.as\_json())  
 **return** render(request, **'role/edit.html'**, locals())  
  
  
**class** EditSysUserView(View):  
 *"""  
 编辑管理员信息和增加管理员  
 """* **def** get(self, request):  
 user\_id = request.GET.get(**'user\_id'**, **''**)  
 **if** user\_id:  
 obj = TSysUser.objects.get(user\_id=user\_id)  
  
 roles = TSysRole.objects.filter(~Q(role\_code=**'admin'**))  
 **return** render(request, **'sys\_user/edit.html'**, locals())  
  
 **def** post(self, request):  
 **from** .forms **import** SysUserForm  
 *# print(request.POST,"+++++++++++++++++++++")* user\_id = request.POST.get(**'user\_id'**, **''**)  
 **if** user\_id:  
 form = SysUserForm(request.POST, request.FILES, instance=TSysUser.objects.get(user\_id=user\_id))  
 **else**:  
 form = SysUserForm(request.POST, request.FILES)  
 **if** form.is\_valid():  
 form.save()  
 **return** redirect(reverse(**'main:list\_sysuser'**))  
  
 errors = json.loads(form.errors.as\_json())  
  
 roles = TSysRole.objects.filter(~Q(code=**'admin'**))  
  
 **return** render(request, **'sys\_user/edit.html'**, locals())  
  
  
**def** logout(request):  
 *"""  
 注销* **:param** *request:* **:return***:  
 """* **del** request.session[**'login\_user'**]  
 **return** redirect(reverse(**"main:login"**))  
  
  
**def** notice(request):  
 *"""  
 公告展示* **:param** *request:* **:return***:  
 """* notices = TPublicNotice.objects.all()  
 action = request.GET.get(**'action'**, **''**)  
 **if** action:  
 TPublicNotice.objects.get(public\_notice\_id=request.GET.get(**'public\_notice\_id'**)).delete()  
 **return** render(request, **'notice/list.html'**, locals())  
  
  
**class** EditPublicNotice(View):  
 *# 编辑公告* **def** get(self, request):  
 public\_notice\_id = request.GET.get(**'public\_notice\_id'**, **''**)  
 **if** public\_notice\_id:  
 notice = TPublicNotice.objects.get(public\_notice\_id=public\_notice\_id)  
 **return** render(request, **'notice/edit.html'**, locals())  
  
 **def** post(self, request):  
 **from** .forms **import** NoticeForm  
  
 public\_notice\_id = request.POST.get(**'public\_notice\_id'**, **''**)  
 **if** public\_notice\_id:  
 notice = TPublicNotice.objects.get(public\_notice\_id=public\_notice\_id)  
 form = NoticeForm(request.POST, instance=notice)  
  
 **else**:  
 form = NoticeForm(request.POST)  
  
 **if** form.is\_valid():  
 form.save()  
 **return** redirect(reverse(**'main:notice'**))  
  
 errors = json.loads(form.errors.as\_json())  
 **return** render(request, **'notice/edit.html'**, locals())  
  
  
**def** list\_slide\_show(request):  
 *"""  
 轮播图展示* **:param** *request:* **:return***:  
 """* action = request.GET.get(**'action'**, **''**)  
 **if** action == **'del'**:  
 TSlidesshow.objects.get(pk=request.GET.get(**'id\_'**)).delete()  
 slides = TSlidesshow.objects.all()  
 **for** obj **in** slides:  
 h\_key = obj.house.image  
 h\_image\_url = get\_house\_image(h\_key)  
 obj.house.image = h\_image\_url  
  
 **return** render(request, **'sys\_user/list\_sildeshow.html'**, locals())  
  
  
**class** EditSlideWhowView(View):  
 *# 编辑轮播图* **def** get(self, request):  
 id\_ = request.GET.get(**'id\_'**, **''**)  
 **if** id\_:  
 obj = TSlidesshow.objects.get(pk=id\_)  
 h\_key = obj.house.image  
 h\_image\_url = get\_house\_image(h\_key)  
 **return** render(request, **'sys\_user/edit\_slide.html'**, locals())  
  
 **def** post(self, request):  
 **from** .forms **import** SlideForm  
 id\_ = request.POST.get(**'id'**, **''**)  
 **if** id\_:  
 form = SlideForm(request.POST, instance=TSlidesshow.objects.get(pk=id\_))  
 **else**:  
 form = SlideForm(request.POST)  
  
 **if** form.is\_valid():  
 form.save()  
 **return** redirect(**'/list\_sildeshow/'**)  
  
 errors = json.loads(form.errors.as\_json())  
  
 **return** render(request, **'sys\_user/edit\_slide.html'**, locals())  
  
  
**def** list\_lucky(request):  
 *"""  
 代金券展示* **:param** *request:* **:return***:  
 """* action = request.GET.get(**'action'**, **''**)  
 **if** action == **'del'**:  
 TLuckyTicket.objects.get(pk=request.GET.get(**'id\_'**)).delete()  
 tickets = TLuckyTicket.objects.all()  
 **return** render(request, **'sys\_user/list\_lucky.html'**, locals())  
  
  
**class** EditLuckyView(View):  
 *"""代金券管理"""* **def** get(self, request):  
 id\_ = request.GET.get(**'id\_'**, **''**)  
 **if** id\_:  
 obj = TLuckyTicket.objects.get(pk=id\_)  
  
 **return** render(request, **'sys\_user/edit\_lucky.html'**, locals())  
  
 **def** post(self, request):  
 **from** .forms **import** LuckyForm  
 id\_ = request.POST.get(**'id'**, **''**)  
 begin = request.POST.get(**'begin\_time'**)  
 print(**f'------->>>{**begin**}---{**type(begin)**}'**)  
 **if** id\_:  
 form = LuckyForm(request.POST, instance=TLuckyTicket.objects.get(pk=id\_))  
 **else**:  
 form = LuckyForm(request.POST)  
  
 **if** form.is\_valid():  
 form.save()  
 **return** redirect(**'/lucky\_ticket/'**)  
  
 errors = json.loads(form.errors.as\_json())  
  
 **return** render(request, **'sys\_user/edit\_lucky.html'**, locals())  
  
  
**class** AuditMessage(View):  
 *# 房屋审核* **def** get(self, request):  
 action = request.GET.get(**'action'**, **''**)  
 **if** action:  
 obj = THouseVerify.objects.get(pk=request.GET.get(**'id\_'**))  
  
 **if** action == **'yes'**:  
 obj.verify\_status = 1  
 **elif** action == **'no'**:  
 obj.verify\_status = 2  
 obj.remarks = request.GET.get(**'remarks'**, **''**)  
 obj.save()  
 obj.full\_clean()  
  
  
 objs = THouseVerify.objects.filter(verify\_status=0).all()  
 **for** obj **in** objs:  
 h\_key = obj.house.image  
 h\_image\_url = get\_house\_image(h\_key)  
 obj.house.image = h\_image\_url  
 u\_key = obj.house.user.img  
 u\_image\_url = get\_user\_image(u\_key)  
 obj.house.user.img = u\_image\_url  
 **return** render(request, **'message/list\_audit.html'**, locals())  
  
  
**class** TPublic(View):  
 *"""公告管理"""* **def** get(self, request):  
 action = request.GET.get(**'action'**, **''**)  
 public\_status = request.GET.get(**'public\_status'**,**''**)  
 **if** action:  
 obj = TPublicNotice.objects.get(pk=request.GET.get(**'id\_'**))  
 **if** action == **'yes'**:  
 obj.public\_status = 1  
 **elif** action == **'no'**:  
 obj.public\_status = 2  
 obj.public\_remarks = request.GET.get(**'public\_remarks'**, **''**)  
 obj.save()  
 obj.full\_clean()  
  
 **if** public\_status:  
 objs = TPublicNotice.objects.filter(public\_status=1).all()  
 **else**:  
 objs = TPublicNotice.objects.filter(public\_status=0).all()  
 **return** render(request, **'message/list\_public.html'**, locals())  
  
  
**from** mainapp.models **import** TComplaint  
**def** get\_complain(request):  
 *# get请求参数：action 有三个可选值（yes, del, query)  
 # yes和del 都有一个额外参数：id\_(投诉的ID）  
 # query 有一个参数wd:可以按照订单号或者用户账号、手机号搜索都行* action = request.GET.get(**'action'**, **''**)  
 **if** action==**'yes'**:  
 obj = TComplaint.objects.get(pk=request.GET.get(**'id\_'**))  
 obj.state = 1  
 obj.save()  
 obj.full\_clean()  
 **if** action==**'del'**:  
 TComplaint.objects.get(pk=request.GET.get(**'id\_'**)).delete()  
 *# 查看评论主页* complains = TComplaint.objects.all()  
 **return** render(request, **'complaint/list.html'**, locals())  
  
  
**from** mainapp.models **import** TComment  
**def** get\_comment(request):  
 *# get请求参数：action 有三个可选值（yes, del, query)  
 # yes和del 都有一个额外参数：id\_(投诉的ID）  
 # query 有一个参数wd:可以按照订单号或者用户账号、手机号搜索都行* action = request.GET.get(**'action'**, **''**)  
 **if** action == **'yes'**:  
 obj = TComment.objects.get(pk=request.GET.get(**'id\_'**))  
 obj.state = 1  
 obj.save()  
 obj.full\_clean()  
 **if** action == **'del'**:  
 TComment.objects.get(pk=request.GET.get(**'id\_'**)).delete()  
 comments = TComment.objects.all()  
 **return** render(request, **'comment/list.html'**, locals())

**forms.py**

*#!/usr/bin/python3  
# coding: utf-8  
  
"""  
用来对数据进行检验  
  
"""***from** django **import** forms  
**from** .models **import** TSysRole, TSysUser, TPublicNotice, TLuckyTicket, TSlidesshow  
  
  
**class** RoleForm(forms.ModelForm):  
  
 **class** Meta:  
 model = TSysRole  
 fields = [**'role\_name'**, **'role\_code'**] *# '\_\_all\_\_'* error\_messages = {  
 **'role\_name'**: {  
 **'required'**: **'角色名不能为空'** },  
 **'role\_code'**: {  
 **'required'**: **'角色代码不能为空'** }  
 }  
  
**class** SysUserForm(forms.ModelForm):  
  
 **class** Meta:  
 model = TSysUser  
 fields = [**'username'**, **'password'**, **'role\_id'**, **'nick\_name'**, **'head'**]  
 error\_messages = {  
 **'username'**: {  
 **'required'**: **'账号不能为空'** },  
 **'nick\_name'**: {  
 **'required'**: **'昵称不能为空'** },  
 **'password'**: {  
 **'required'**: **'口令不能为空'** },  
 **'role\_id'**: {  
 **'required'**: **'系统用户角色不能为空'** }  
 }  
  
  
**class** NoticeForm(forms.ModelForm):  
  
 **class** Meta:  
 model = TPublicNotice  
 fields = [**'content'**, **'public\_title'**, **'public\_time'**, **'public\_remarks'**, **'public\_notice\_id'**]  
 error\_messages = {  
 **'public\_title'**: {  
 **'required'**: **'标题不能为空'** },  
  
 **'content'**:{  
 **'required'**: **'内容不能为空'** }  
 }  
  
**class** SlideForm(forms.ModelForm): *# 轮播图* **class** Meta:  
 model = TSlidesshow  
 fields = [**'house'**, **'ord'**] *# '\_\_all\_\_'* error\_messages = {  
 **'house'**: {  
 **'required'**: **'房屋ID不能为空'** },  
 **'ord'**: {  
 **'required'**: **'顺序不能为空'** }  
 }  
  
  
**class** LuckyForm(forms.ModelForm): *# 代金券* **class** Meta:  
 model = TLuckyTicket  
 fields = [**'money'**, **'begin\_time'**, **'end\_time'**, **'image'**] *# '\_\_all\_\_'* error\_messages = {  
 **'money'**: {  
 **'required'**: **'面值不能为空'** },  
 **'begin\_time'**: {  
 **'required'**: **'起始时间不能为空'** },  
 **'end\_time'**: {  
 **'required'**: **'结束时间不能为空'** },  
 **'image'**: {  
 **'required'**: **'图片路径不能为空'** }  
 }

**oss\_.py**

*#!/usr/bin/python3  
# coding: utf-8***import** oss2  
  
small\_style = **'image/interlace,1/resize,m\_lfit,w\_100/quality,q\_90/bright,3/contrast,-21'  
  
  
def** get\_bucket():  
 *# 阿里云主账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM账号进行API访问或日常运维，请登录 https://ram.console.aliyun.com 创建RAM账号。* auth = oss2.Auth(**'LTAI4FqbMxbMPrnKSBsu5KSh'**, **' copeGy6ElAJ0fdleAkYxcOME5TR4CJ'**)  
 *# Endpoint以杭州为例，其它Region请按实际情况填写。* bucket = oss2.Bucket(auth, **'http://oss-cn-hangzhou.aliyuncs.com'**, **'homems'**)  
  
 **return** bucket  
  
  
**def** upload\_head(user\_id, file\_name, file\_path):  
 bucket = get\_bucket()  
  
 *# <yourObjectName>上传文件到OSS时需要指定包含文件后缀在内的完整路径，例如abc/efg/123.jpg。  
 # <yourLocalFile>由本地文件路径加文件名包括后缀组成，例如/users/local/myfile.txt。* key = **f'head/{**user\_id**}-{**file\_name**}'** ret = bucket.put\_object\_from\_file(key, file\_path)  
  
 **if** ret.status == 200:  
 **return** bucket.sign\_url(**'GET'**, key, 3600 \* 24 \* 7, params={**'x-oss-process'**: small\_style})  
  
  
**def** get\_oss\_img\_url(key):  
 bucket = get\_bucket()  
  
 **return** bucket.sign\_url(**'GET'**, **f'head/{**key**}'**, 3600 \* 24 \* 7, params={**'x-oss-process'**: small\_style})  
  
  
**def** upload\_house\_image(user\_id, file\_name, file\_path):  
 bucket = get\_bucket()  
  
 *# <yourObjectName>上传文件到OSS时需要指定包含文件后缀在内的完整路径，例如abc/efg/123.jpg。  
 # <yourLocalFile>由本地文件路径加文件名包括后缀组成，例如/users/local/myfile.txt。* key = **f'house/h-{**user\_id**}-{**file\_name**}'** ret = bucket.put\_object\_from\_file(key, file\_path)  
  
 **if** ret.status == 200:  
 **return** bucket.sign\_url(**'GET'**, key, 3600 \* 24 \* 7)  
  
  
**def** get\_oss\_house\_image(key):  
 bucket = get\_bucket()  
  
 **return** bucket.sign\_url(**'GET'**, **f'house/{**key**}'**, 3600 \* 24 \* 7)

前台系统代码

**\_\_init\_\_.py**

**from** flask **import** Flask  
**from** flask\_cors **import** CORS  
  
**import** settings  
**from** apiapp.views **import** user\_api, house\_api, slide\_api  
  
  
app = Flask(\_\_name\_\_,  
 static\_folder=settings.STATIC\_DIR,  
 static\_url\_path=**'/s/'**)  
  
app.config[**'ENV'**] = **'development'** *# production*app.config[**'DEBUG'**] = **True**app.register\_blueprint(user\_api.blue, url\_prefix=**'/api/'**)  
app.register\_blueprint(house\_api.blue, url\_prefix=**'/api/'**)  
app.register\_blueprint(slide\_api.blue, url\_prefix=**'/api/'**)  
CORS(app) *# 全局方式支持跨域请求（前端服务器和后端API服务器分开部署）*

**house\_api.py**

**import** datetime  
**import** os  
  
**from** flask **import** Blueprint, request  
**from** flask.json **import** jsonify  
**from** sqlalchemy **import** and\_  
  
**import** settings  
**from** apiapp.views **import** validate\_json, validate\_params, get\_house\_image  
**from** common **import** cache\_, oss\_  
**from** common.serializer **import** to\_json  
**from** db\_ **import** session  
**from** apiapp.models **import** THouse, THouseVerify  
  
blue = Blueprint(**'house\_api'**, \_\_name\_\_)  
  
  
  
*# 房源发布*@blue.route(**'/add/'**, methods=[**'POST'**])  
**def** add\_house():  
 *"""  
 房源发布* **:return***:  
 """  
 # resp = validate\_json()  
 # if resp: return resp  
 # resp = validate\_params(['token', 'image', 'name', 'type', 'address', 'price', 'area', 'discription'])  
 # if resp: return resp* upload\_file = request.files.get(**'image'**)  
 file\_name = upload\_file.filename  
 *# print(file\_name)* save\_file\_path = os.path.join(settings.TEMP\_HOUSE\_DIR, file\_name)  
 *# 保存上传的文件到临时的目录中* upload\_file.save(save\_file\_path)  
  
 user\_id = cache\_.get\_user\_id(request.form.get(**'token'**))  
 **if** user\_id:  
 **try**:  
 house = THouse()  
 house.user\_id = user\_id  
 *# house.image = 'http://localhost:5000/s/' + file\_name* key = **f'h-{**user\_id**}-{**file\_name**}'** *# 存储oss上的key对象* house.image = key  
 house.name = request.form.get(**'name'**)  
 house.type = request.form.get(**'type'**)  
 house.address = request.form.get(**'address'**)  
 house.price = request.form.get(**'price'**)  
 house.area = request.form.get(**'area'**)  
 house.description = request.form.get(**'discription'**)  
 house.publish\_time = datetime.datetime.now()  
 house.sale\_status = 0  
 house.is\_public = 1  
 session.add(house)  
 session.commit()  
  
  
 house\_image\_url = oss\_.upload\_house\_image(user\_id, file\_name, save\_file\_path)  
  
 *# 将房屋图片的url存到redis中* cache\_.save\_house\_url(key, house\_image\_url)  
  
 verify = THouseVerify()  
 verify.house\_id = house.house\_id  
 verify.verify\_status = 0  
 session.add(verify)  
 session.commit()  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'发布房源成功'** })  
 **except**:  
 **return** jsonify({  
 **'state'**: 2,  
 **'msg'**: **'网络延迟，请等会再尝试！'** })  
 **return** jsonify({  
 **"state"**: 1,  
 **'msg'**: **'未登录，请先登录！'** })  
  
  
@blue.route(**'/del\_house/'**, methods=[**'POST'**])  
**def** del\_house():  
 *"""  
 删除房源信息的房源信息* **:return***:  
 """  
 # 验证参数完整性  
 # resp = validate\_json()  
 # if resp: return resp  
 # resp = validate\_params('token', 'house\_id')  
 # if resp: return resp* data = request.get\_json()  
 house\_id = data[**'house\_id'**]  
 token = data[**'token'**]  
 user\_id = cache\_.get\_user\_id(token)  
 house = session.query(THouse).filter(house\_id == house\_id, user\_id == user\_id)  
 print(house)  
 **if** house:  
 obj = house.first()  
 obj.is\_public = 0  
 session.add(obj)  
 session.commit()  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'撤销房源成功'** })  
 **else**:  
 **return** jsonify({  
 **'state'**: 1,  
 **'msg'**: **'登录已过期或房屋信息已删除'** })  
  
  
@blue.route(**'/change\_house\_info/'**, methods=[**'POST'**])  
**def** change\_house\_info():  
 *"""  
 改变房屋信息  
 :return:  
 """  
 # resp = validate\_json()  
 # if resp: return resp  
 # resp = validate\_params('image', 'name', 'type', 'address', 'price', 'area', 'description', 'token', 'house\_id')  
 # if resp: return resp* upload\_file = request.files.get(**'image'**)  
 file\_name = upload\_file.filename  
 *# print(file\_name)* save\_file\_path = os.path.join(settings.TEMP\_HOUSE\_DIR, file\_name)  
 *# 保存上传的文件到临时的目录中* upload\_file.save(save\_file\_path)  
  
 data = request.form  
 house\_id = data[**'house\_id'**]  
 token = data[**'token'**]  
 user\_id = cache\_.get\_user\_id(token)  
 **if** user\_id:  
 **try**:  
 house = session.query(THouse).filter(THouse.house\_id == house\_id, THouse.user\_id == user\_id).first()  
 key = **f'h-{**user\_id**}-{**file\_name**}'** *# 存储oss上的key对象* house.image = key  
 house.name = data[**'name'**]  
 house.type = data[**'type'**]  
 house.address = data[**'address'**]  
 house.price = data[**'price'**]  
 house.area = data[**'area'**]  
 house.description = data[**'description'**]  
 house.publish\_time = datetime.datetime.now()  
 session.add(house)  
 session.commit()  
  
 house\_image\_url = oss\_.upload\_house\_image(user\_id, file\_name, save\_file\_path)  
  
 *# 将房屋图片的url存到redis中* cache\_.save\_house\_url(key, house\_image\_url)  
  
 verify = THouseVerify()  
 verify.house\_id = house.house\_id  
 verify.verify\_status = 0  
 session.add(verify)  
 session.commit()  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'房源信息修改成功！等待审核！'** })  
 **except**:  
 **return** jsonify({  
 **'state'**: 2,  
 **'msg'**: **'未发布房源，请先发布房源！'** })  
 **else**:  
 **return** jsonify({  
 **'state'**: 1,  
 **'msg'**: **'还未登陆，请先登录！'** })  
  
  
@blue.route(**'/list\_houses/'**, methods=[**'GET'**])  
**def** list\_user\_houses():  
 *"""  
 查看现登录房东的已通过审核的的房源信息  
 :return:  
 """* data = request.args.get(**'token'**)  
 user\_id = cache\_.get\_user\_id(data)  
 print(user\_id)  
 **if** user\_id:  
 ret = session.query(THouse).filter(THouse.user\_id == user\_id).all()  
 **try**:  
 data = to\_json(ret)  
  
 **for** house\_info **in** data:  
 h\_key = house\_info.get(**'image'**)  
 h\_image\_url = get\_house\_image(h\_key)  
 *# print("+++++++++++++++", image\_url)* house\_info[**'image'**] = h\_image\_url  
 u\_key = house\_info[**'user'**][**'img'**]  
 u\_image\_url = get\_house\_image(u\_key)  
 house\_info[**'user'**][**'img'**] = u\_image\_url  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'查询成功'**,  
 **'data'**: data  
 })  
 **except**:  
 **return** jsonify({  
 **'state'**: 2,  
 **'msg'**: **'暂未发布房源'**,  
 })  
 **else**:  
 **return** jsonify({  
 **'state'**: 1,  
 **'msg'**: **'未登录'** })  
  
  
@blue.route(**'/list\_all\_houses/'**, methods=[**'GET'**])  
**def** list\_all\_houses():  
 *"""  
 查看所有房源详细信息  
 :return:  
 """* **try**:  
 ret = session.query(THouse).all()  
 data = to\_json(ret)  
 **for** (house\_info, verify\_obj) **in** zip(data, ret):  
 h\_key = house\_info.get(**'image'**)  
 h\_image\_url = get\_house\_image(h\_key)  
 *# print("+++++++++++++++", image\_url)* house\_info[**'image'**] = h\_image\_url  
 u\_key = house\_info[**'user'**][**'img'**]  
 u\_image\_url = get\_house\_image(u\_key)  
 house\_info[**'user'**][**'img'**] = u\_image\_url  
 house\_info[**"verify\_status"**] = verify\_obj.t\_house\_verifies[0].verify\_status  
 print(**"verify\_status"**, house\_info[**"verify\_status"**])  
  
 **return** jsonify({  
 **'state'**: 0,  
 **'data'**: data  
 })  
 **except** Exception **as** e:  
 session.rollback()  
 print(e)  
 **return** jsonify({  
 **'state'**: 1,  
 **'msg'**: **'获取房屋信息失败！'** })  
  
@blue.route(**'/detail/'**, methods=[**'GET'**])  
**def** detail\_info():  
 *"""  
 查看房屋的详情  
 :return:  
 """* token = request.args.get(**'token'**)  
 user\_id = cache\_.get\_user\_id(token)  
 **if** user\_id:  
 house\_id = request.args.get(**'house\_id'**)  
  
 verify\_info = session.query(THouseVerify).filter(THouseVerify.house\_id == house\_id)  
 **if** verify\_info:  
 obj = verify\_info.first()  
 house\_info = to\_json(obj.house)  
 *# print("+++++++++++++++", type(house\_info), house\_info)* h\_key = house\_info.get(**'image'**)  
 h\_image\_url = get\_house\_image(h\_key)  
 *# print("+++++++++++++++", image\_url)* house\_info[**'image'**] = h\_image\_url  
 u\_key = house\_info[**'user'**][**'img'**]  
 u\_image\_url = get\_house\_image(u\_key)  
 house\_info[**'user'**][**'img'**] = u\_image\_url  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'查询成功'**,  
 **'data'**: {  
 **'verify\_state'**: obj.verify\_status,  
 **'houses'**: house\_info  
 }  
 })  
 **else**:  
 **return** jsonify({  
 **'state'**: 2,  
 **'msg'**: **'查询失败，发生了未知错误，请重新尝试！'** })  
  
 **return** jsonify({  
 **'state'**: 1,  
 **'msg'**: **'未登录，请先登录！'** })

**user\_api.py**

**import** os  
  
**from** flask **import** Blueprint, Response  
**from** flask **import** request, jsonify  
**from** sqlalchemy **import** or\_  
  
**import** settings  
**from** apiapp.models **import** TUser, TScore, TPanda, Contract  
**from** apiapp.views **import** validate\_json, validate\_params  
**from** common **import** code\_, token\_, cache\_, oss\_  
**from** common.serializer **import** to\_json  
**from** common.simple\_check **import** simple\_check  
**from** db\_ **import** session  
*# from db.models import TUser***from** db\_.raw **import** query  
  
blue = Blueprint(**'user\_api'**, \_\_name\_\_)  
  
  
*# 发送短信验证码*@blue.route(**'/code/'**, methods=[**'get'**])  
**def** get\_code():  
 *"""  
 发送短信验证码* **:return***:  
 """* phone = request.args.get(**'phone'**)  
 **if** phone:  
 **try**:  
 code\_.send\_code(phone)  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'验证码已发送'** })  
 **except**:  
 **return** jsonify({  
 **'state'**: 2,  
 **'msg'**: **'获取验证码失败'** })  
 **return** jsonify({  
 **'state'**: 1,  
 **'msg'**: **'手机号不能为空'** })  
  
  
*# 用户注册*@blue.route(**'/regist/'**, methods=[**'GET'**, **'POST'**])  
**def** regist():  
 *# 获取前端传来的JSON格式数据* valid\_fields = {**"phone"**, **"code"**, **"password"**}  
 data = request.get\_json()  
 **if** data **is None**:  
 **return** jsonify({  
 **'state'**: 4,  
 **'msg'**: **'必须提供json格式的参数'** })  
  
 *# 验证参数的完整性* **if** set(data.keys()) == valid\_fields:  
 *# 验证输入的验证码是否正确* **if not** code\_.valid\_code(data[**'phone'**], data[**'code'**]):  
 **return** jsonify({  
 **'state'**: 2,  
 **'msg'**: **'验证码输入错误，请确认输入的验证码'** })  
 user = TUser()  
 *# user.name = data.get('name')* user.phone = data.get(**'phone'**)  
 user.password = data.get(**'password'**)  
  
 session.add(user)  
 session.commit()  
 *# 向前端返回信息中，包含一个与用户匹配的token(有效时间为一周)  
 # 1. 基于uuid+user\_id生成token  
 # 2. 将token和user\_id保存到缓存（cache\_.save\_token(token, user\_id)）  
 # JWT 单点授权登录* token = token\_.gen\_token(user.user\_id)  
 cache\_.add\_token(token, user.user\_id)  
 **else**:  
 **return** jsonify({  
 **'state'**: 1,  
 **'msg'**: **'参数不完整，详情查看接口文档'** })  
  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'注册成功'** })  
  
  
*# 用户登陆*@blue.route(**'/login/'**, methods=[**'POST'**])  
**def** login():  
 data = request.get\_json()  
 print(**"data="**, data)  
 **try**:  
 user = session.query(TUser).filter(or\_(TUser.phone == data[**'phone'**],  
 TUser.nickname == data[**'nickname'**]),  
 TUser.password == data[**'password'**]).one()  
  
  
 **if** user:  
 token = token\_.gen\_token(user.user\_id)  
 cache\_.add\_token(token, user.user\_id)  
 head\_url = **''  
 if** user.img:  
 head\_url = cache\_.get\_head\_url(user.img)  
 *# print(head\_url)* **if not** head\_url:  
 head\_url = oss\_.get\_oss\_img\_url(user.img)  
 cache\_.save\_head\_url(user.img, head\_url)  
 resp: Response = jsonify({  
 **'state'**: 0,  
 **'msg'**: **'登录成功'**,  
 **'token'**: token,  
 **'head'**: head\_url  
 })  
 *# 设置响应对象的cookie，向客户端响应cookie* resp.set\_cookie(**'token'**, token)  
 resp.set\_cookie(**'head'**, head\_url)  
 **return** resp  
 **except**:  
 **return** jsonify({  
 **'state'**: 1,  
 **'msg'**: **'用户名或口令输入错误'**,  
 })  
  
  
*# 普通用户升级为会员用户*@blue.route(**'/become\_superuser/'**, methods=[**'POST'**])  
**def** become\_superuser():  
 *"""  
 普通用户升级为会员用户,并进行实名认证  
 :return:  
 """  
 # 验证用户是否输入json格式数据* resp = validate\_json()  
 **if** resp: **return** resp  
 *# 验证用户输入参数是否齐全* resp = validate\_params(**'token'**, **'nickname'**, **'sex'**, **'phone'**, **'email'**, **'name'**, **'identity\_number'**)  
 **if** resp: **return** resp  
 data = request.get\_json()  
 print(**"data="**, data)  
 token = data.get(**'token'**)  
 user\_id = cache\_.get\_user\_id(token)  
 print(user\_id)  
 **if** user\_id:  
 user = session.query(TUser).get(user\_id)  
 **if** simple\_check(data[**'identity\_number'**], data[**'name'**]):  
 user.name = data[**'name'**]  
 user.nickname = data[**'nickname'**]  
 user.sex = data[**'sex'**]  
 user.phone = data[**'phone'**]  
 user.identity\_number = data[**'identity\_number'**]  
 user.is\_member = 1  
 user.has\_real\_name = 1  
 session.add(user)  
 session.commit()  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'已成功成为会员用户，您可以开始发布自己的房源了！'**,  
 })  
 **return** jsonify({  
 **'state'**: 2,  
 **'msg'**: **'请填写正确的身份信息！'** })  
 **else**:  
 **return** jsonify({  
 **'state'**: 1,  
 **'msg'**: **'未登录， 请先登录！'** })  
  
  
*# 用户修改密码*@blue.route(**'/modify\_password/'**, methods=[**'POST'**])  
**def** modify\_password():  
 *# 验证用户是否输入json格式数据* resp = validate\_json()  
 **if** resp: **return** resp  
  
 *# 验证用户输入参数是否齐全* resp = validate\_params(**'new\_password\_str'**, **'password\_str'**, **'token'**)  
 **if** resp: **return** resp  
  
 data = request.get\_json()  
 print(data[**'token'**])  
 **try**:  
 user\_id = cache\_.get\_user\_id(data[**"token"**])  
 print(user\_id)  
 **if not** user\_id:  
 **return** jsonify({  
 **'state'**: 3,  
 **'msg'**: **'登录已期，需要重新登录并获取新的token'**,  
 })  
 user = session.query(TUser).get(int(user\_id))  
 **if** user.password == data[**'password\_str'**]:  
 user.password = data[**'new\_password\_str'**]  
 session.add(user)  
 session.commit()  
  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'修改成功'** })  
 **return** jsonify({  
 **'state'**: 4,  
 **'msg'**: **'原口令不正确'** })  
 **except**:  
 **pass  
  
 return** jsonify({  
 **'state'**: 3,  
 **'msg'**: **'token已无效，尝试重新登录'**,  
 })  
  
  
*# 用户注销*@blue.route(**'/login\_out/'**)  
**def** login\_out():  
 token = request.args.get(**'token'**)  
 cache\_.del\_token(token)  
  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'退出登陆成功'** })  
  
  
*# 用户上传头像*@blue.route(**'/upload\_head/'**, methods=[**"POST"**,**"GET"**])  
**def** upload\_head():  
 *# 前端上传图片的两种方式(文件上传，base64字符串上传)* upload\_file = request.files.get(**'head'**)  
 token = request.cookies.get(**'token'**)  
 *# print(upload\_file.filename)  
 # print(token)* user\_id = cache\_.get\_user\_id(token)  
 filename = upload\_file.filename  
 save\_file\_path = os.path.join(settings.TEMP\_USER\_DIR, filename)  
 *# 保存文件到临时的目录中* upload\_file.save(save\_file\_path)  
  
 *# 将头像保存到用户表中* user = session.query(TUser).get(user\_id)  
 **if** user:  
 **try**:  
 user.img = **f"{**user\_id**}-{**filename**}"** *# 存储oss上的key对象* session.add(user)  
 session.commit()  
  
 *# 将保存的文件上传到oss服务中，并获取到缩小后的图片的url* head\_url = oss\_.upload\_head(user\_id, filename, save\_file\_path)  
  
 *# 将头像的url存到redis中* cache\_.save\_head\_url(user.img, head\_url)  
  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'上传成功'** })  
 **except**:  
 **return** jsonify({  
 **'state'**: 2,  
 **'msg'**: **'上传失败'** })  
 **return** jsonify({  
 **'state'**: 1,  
 **'msg'**: **'登录已超时，请先登录！'** })  
  
  
*# 获取用户头像*@blue.route(**'/head/'**, methods=[**"GET"**])  
**def** get\_head():  
 token = request.args.get(**"token"**)  
 user\_id = cache\_.get\_user\_id(token)  
 **if not** user\_id:  
 **return** jsonify({  
 **'state'**: 3,  
 **'msg'**: **'登录已期，需要重新登录并获取新的token'**,  
 })  
  
 *# sql = 'select nickname,img,is\_member from t\_user where user\_id=%s'  
 # data = query(sql, user\_id)* user = session.query(TUser).get(user\_id)  
 nickname = user.nickname  
 img = user.img  
 is\_member = user.is\_member  
 data = {  
 **'nickname'**: nickname,  
 **'img'**: img,  
 **'is\_member'**: is\_member  
 }  
 **return** jsonify({  
 **'state'**: 0,  
 **'data'**: data  
 })  
  
  
*# 用户更新信息*@blue.route(**'/detail\_resource/'**, methods=[**"POST"**])  
**def** change\_user\_info():  
 *# 验证用户是否输入json格式数据* resp = validate\_json()  
 **if** resp: **return** resp  
  
 data = request.get\_json()  
  
 token = request.args.get(**'token'**)  
 user\_id = cache\_.get\_user\_id(data[**"token"**])  
 **if not** user\_id:  
 **return** jsonify({  
 **'state'**: 3,  
 **'msg'**: **'登录已期，需要重新登录并获取新的token'**,  
 })  
  
 *# 根据id查询用户* user = session.query(TUser).get(user\_id)  
 user.nickname = data[**"nickname"**]  
 user.email = data[**"email"**]  
 session.add(user)  
 session.commit()  
  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'更新信息成功'**,  
 **'data'**: data  
 })  
  
  
*# 用户积分查询*@blue.route(**'/integral/'**, methods=[**"GET"**])  
**def** interal():  
 token = request.args.get(**'token'**)  
 user\_id = cache\_.get\_user\_id(token)  
 **if not** user\_id:  
 **return** jsonify({  
 **'state'**: 3,  
 **'msg'**: **'登录已期，需要重新登录并获取新的token'**,  
 })  
 sql = **'select score from t\_score join t\_user on t\_score.user\_id=t\_user.user\_id where t\_score.user\_id=%s'** score = query(sql, user\_id)  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'ok'**,  
 **'data'**: score  
 })  
  
  
*# 关于我们*@blue.route(**'/about\_pandas/'**, methods=[**"GET"**])  
**def** about\_pandas():  
 sql = **'select \* from t\_panda where t\_panda.panda\_id=%s'** detail = query(sql, 1)  
  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'ok'**,  
 **'interal'**: detail  
 })  
  
  
*# 查看合同*@blue.route(**'/get\_contract/'**, methods=[**"GET"**])  
**def** get\_contract():  
 token = request.args.get(**'token'**)  
 user\_id = cache\_.get\_user\_id(token)  
 **if not** user\_id:  
 **return** jsonify({  
 **'state'**: 3,  
 **'msg'**: **'登录已期，需要重新登录并获取新的token'**,  
 })  
 sql = **'select start\_time,stop\_time,content from t\_user join contract c2 on t\_user.user\_id = c2.user\_id where c2.user\_id=%s'** cont\_details = query(sql, user\_id)  
  
 **if** cont\_details:  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'已查询到合同信息'**,  
 **'data'**: cont\_details  
 })  
 **return** jsonify({  
 **'state'**: 1,  
 **'msg'**: **'您的合同信息不存在'** })  
  
  
*# 查看房屋交易记录*@blue.route(**'/traderecord/'**, methods=[**"GET"**])  
**def** traderecord():  
 token = request.args.get(**'token'**)  
 user\_id = cache\_.get\_user\_id(token)  
 **if not** user\_id:  
 **return** jsonify({  
 **'state'**: 3,  
 **'msg'**: **'登录已期，需要重新登录并获取新的token'**,  
 })  
 sql = **'select th.\*,td.\* from t\_house th,t\_tradingrecord td,t\_user tu where th.user\_id=td.user\_id and td.user\_id=tu.user\_id and tu.user\_id=%s'** house\_details = query(sql, user\_id)  
 **if** house\_details:  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'已查询到房屋交记录'**,  
 **"data"**: house\_details  
 })  
 **return** jsonify({  
 **'state'**: 1,  
 **'msg'**: **'您暂时没有交易记录'** })  
  
  
*# 查看订单*@blue.route(**'/get\_order/'**, methods=[**"GET"**])  
**def** get\_order():  
 token = request.args.get(**'token'**)  
 user\_id = cache\_.get\_user\_id(token)  
 **if not** user\_id:  
 **return** jsonify({  
 **'state'**: 3,  
 **'msg'**: **'登录已期，需要重新登录并获取新的token'**,  
 })  
 status = int(request.args.get(**"status"**))  
 print(status, type(status))  
 **if** status == 2:  
 *# 1查看全部订单* sql = **'select th.\*,td.\* from t\_user tu,t\_order td,t\_house th where tu.user\_id=td.user\_id and tu.user\_id=th.user\_id and tu.user\_id=%s'** all\_orders = query(sql, user\_id)  
 print(all\_orders)  
 **if** all\_orders:  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'全部订单信息'**,  
 **'order'**: all\_orders  
 })  
 **elif** status == 0:  
 *# 2.查看已支付订单* sql = **'select th.\*,td.\* from t\_user tu,t\_order td,t\_house th where tu.user\_id=td.user\_id and tu.user\_id=th.user\_id and td.order\_status=%s'** pay\_order = query(sql, 0)  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'有效订单信息'**,  
 **'order'**: pay\_order  
 })  
  
 **elif** status == 1:  
 *# 3.查看待支付订单* sql = **'select th.\*,td.\* from t\_user tu,t\_order td,t\_house th where tu.user\_id=td.user\_id and tu.user\_id=th.user\_id and td.order\_status=%s'** nopay\_order = query(sql, 1)  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'待支付订单信息'**,  
 **'order'**: nopay\_order  
 })  
  
 **return** jsonify({  
 **'state'**: 1,  
 **'msg'**: **'暂无订单'** })  
  
  
*# 查看代金券*@blue.route(**'/cash/'**, methods=[**"GET"**])  
**def** cash():  
 token = request.args.get(**'token'**)  
 user\_id = cache\_.get\_user\_id(token)  
 **if not** user\_id:  
 **return** jsonify({  
 **'state'**: 3,  
 **'msg'**: **'登录已期，需要重新登录并获取新的token'**,  
 })  
 sql = **'select tl.\* from t\_u\_lucky\_ticket tul,t\_lucky\_ticket tl,t\_user tu where tu.user\_id=tul.user\_id and tul.lucky\_ticket\_id=tl.lucky\_ticket\_id and tu.user\_id=%s'** cash\_ = query(sql, user\_id)  
 print(cash\_)  
  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'代金券查询成功'**,  
 **'data'**: cash\_  
 })

**slide.py**

**import** uuid  
  
**from** flask **import** Blueprint, request, g, redirect, url\_for  
**from** flask.json **import** jsonify  
  
**from** apiapp.views **import** validate\_json, validate\_params, get\_house\_image  
**from** common **import** cache\_  
**from** common.serializer **import** to\_json  
**from** db\_ **import** session  
**from** apiapp.models **import** TUser, THouse, THouseVerify, TSlidesshow, TFavorite, TOrder  
  
blue = Blueprint(**'slide\_api'**, \_\_name\_\_)  
  
  
@blue.route(**'/slide/'**, methods=[**'GET'**])  
**def** slide\_view(): *# 获取首页轮播图* **try**:  
 slides\_list = session.query(TSlidesshow).all()  
 *# house\_info = session.query(THouse).filter(THouse.house\_id == s.house\_id)* res = to\_json(slides\_list)  
 **for** ad **in** res:  
 key = ad[**'house'**][**"image"**]  
 ad[**'house'**][**"image"**] = get\_house\_image(key)  
  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'请求成功'**,  
 **'data'**: res  
 })  
  
 **except** Exception **as** e:  
 session.rollback()  
 print(**"获取轮播图数据错误"**, e)  
 **return** jsonify({  
 **'state'**: 1,  
 **'msg'**: **'服务器错误'** })  
  
  
*# 首页房屋展示*@blue.route(**'/house\_display/'**, methods=[**"GET"**])  
**def** house\_display():  
 houses = session.query(THouse).all()  
 **if** houses:  
 **return** jsonify({  
 **"state"**: 0,  
 **'data'**: to\_json(houses)  
 })  
 **return** jsonify({  
 **'state'**: 1,  
 **'msg'**: **'房屋信息不存在'** })  
  
  
*# 房屋详细信息页*@blue.route(**'/house\_detail/'**, methods=[**'GET'**])  
**def** detail\_info():  
 *"""  
 查看房屋的详情* **:return***:  
 """* token = request.args.get(**'token'**)  
 user\_id = cache\_.get\_user\_id(token)  
 **if** user\_id:  
 house\_id = request.args.get(**'house\_id'**)  
  
 verify\_info = session.query(THouseVerify).filter(THouseVerify.house\_id == house\_id)  
 **if** verify\_info:  
 obj = verify\_info.first()  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'查询成功'**,  
 **'data'**: {  
 **'verify\_state'**: obj.verify\_status,  
 **'houses'**: to\_json(obj.house)  
 }  
 })  
 **else**:  
 **return** jsonify({  
 **'state'**: 2,  
 **'msg'**: **'查询失败，发生了未知错误，请重新尝试！'** })  
  
 **return** jsonify({  
 **'state'**: 1,  
 **'msg'**: **'未登录，请先登录！'** })  
  
  
*# 收藏房屋*@blue.route(**'/insert\_collection/'**, methods=[**"POST"**])  
**def** collection():  
 data = request.get\_json()  
 user\_id = cache\_.get\_user\_id(data[**"token"**])  
 **if** user\_id:  
 house\_id = data[**"house\_id"**]  
 fav = TFavorite()  
 fav.house\_id = house\_id  
 fav.user\_id = user\_id  
 session.add(fav)  
 session.commit()  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'房屋收藏成功'**,  
 })  
 **return** jsonify({  
 **'state'**: 1,  
 **'msg'**: **'房屋添加不成功'** })  
  
  
*# 查询收藏房屋的信息*@blue.route(**'/get\_collection/'**, methods=[**"GET"**])  
**def** get\_collection():  
 data = request.args.get(**'token'**)  
 user\_id = cache\_.get\_user\_id(data)  
 **if** user\_id:  
 ret = session.query(TFavorite).filter(TFavorite.user\_id == user\_id).all()  
 *# if house\_detail:* **return** jsonify({  
 **'state'**: 0,  
 **'data'**: to\_json(ret)  
 })  
 **return** ({  
 **'state'**: 1,  
 **'msg'**: **'暂无收藏的房屋'** })  
  
  
*# 下订单*ORDER\_PAY = 0 *# 已支付订单*ORDER\_NOT\_PAY = 1 *# 待支付订单  
  
# 添加订单*@blue.route(**'/order/'**, methods=[**"POST"**])  
**def** order():  
 *"""  
 添加订单* **:return***:  
 """* resp = validate\_json()  
 **if** resp: **return** resp  
 resp = validate\_params(**'token'**, **'house\_id'**, **'hire\_price'**, **'cash\_price'**, **'enter\_time'**, **'exit\_time'**)  
 **if** resp: **return** resp  
  
 order\_num = uuid.uuid4().hex  
 data = request.get\_json()  
 user\_id = cache\_.get\_user\_id(data[**"token"**])  
 *# total\_price = (data["exit\_time"] - data["enter\_time"]) \* float(data["hire\_price"]) + float(data["cash\_price"])* order = TOrder()  
 order.order\_number = order\_num  
 order.user\_id = user\_id  
 order.house\_id = data[**"house\_id"**]  
 order.hire\_price = data[**"hire\_price"**]  
 order.cash\_price = data[**"cash\_price"**]  
 order.enter\_time = data[**"enter\_time"**]  
 order.exit\_time = data[**"exit\_time"**]  
 *# order.total = total\_price* order.order\_status = ORDER\_NOT\_PAY  
 session.add(order)  
 session.commit()  
 order\_info = session.query(TOrder).filter(TOrder.user\_id==user\_id, TOrder.order\_number==order\_num)  
 *# print(order\_info)* **if** order\_info:  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'下单成功'**,  
 **'data'**: to\_json(order\_info.all())  
 })  
  
  
@blue.route(**'/del\_order/'**, methods=[**'POST'**])  
**def** del\_order():  
 resp = validate\_json()  
 **if** resp: **return** resp  
 resp = validate\_params(**'token'**, **'order\_number'**)  
 **if** resp: **return** resp  
  
 data = request.get\_json()  
 user\_id = cache\_.get\_user\_id(data[**'token'**])  
 **if** user\_id:  
 order\_number = data[**'order\_number'**]  
 order = session.query(TOrder).filter(TOrder.order\_number==order\_number, TOrder.user\_id==user\_id).first()  
 **if** order:  
 session.delete(order)  
 session.commit()  
 **return** jsonify({  
 **'state'**: 0,  
 **'msg'**: **'该订单已删除！'** })  
 **else**:  
 **return** jsonify({  
 **'state'**: 1,  
 **'msg'**: **'订单不存在'** })  
 **else**:  
 **return** jsonify({  
 **'state'**: 1,  
 **'msg'**: **'登录已超时，请重新登录！'** })

致谢

从我的毕业系统设计的开题提交到程序和论文的完成，经历了一个非常漫长的过程，在这个漫长过程中，首先最应该感谢我的指导老师陈垚老师，陈垚老师总是特别认真和耐心，在陈垚老师的细心指导之下，非常大地促进了我的毕业设计进度，同时我也学到了特别多的新东西。陈垚老师也经常询问我们的毕业设计进度情况，而且每当我把已经完成的论文发给陈垚老师后，老师也会特别及时的进行检查和给予指导意见。

同时也特别地感谢我的同学和身边的朋友，在毕业设计期间，无论是我的学习、生活上遇到任何问题向他们请教时，他们也都总会力所能及的对我进行帮助;当我的毕设遇到难以解决的问题时，他们也一直也都一直在尽力的帮助我。我真的特别感谢他们，正是在他们的无私帮助之下，我的毕设才能圆满完成。

当然我也非常的感谢我的学校一一商洛学院，它为我们提供了一个特别好的学习平台，也给我们配备了最优秀的专业导师，让我们接触并且学习到了非常多的专业知识知识。“不忘初心，继续前进”的校训我将铭记在心!