电子科技大学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

专业学位硕士学位论文

MASTER THESIS FOR PROFESSIONAL DEGREE

论文题目 基于 Django 的智慧园区平台系统设计与实现

专业学位类别	工 程 硕 士
学　　号	201552201030
作 者 姓 名	牛宁
指 导 教 师	佘　堃　教 授

分类号 _____ 密级 _____

UDC [注1] _____

学 位 论 文

基于 Django 的智慧园区平台系统设计与实现

（题名和副题名）

牛 宁

（作者姓名）

指导教师 佘 堃 教 授

电子科技大学 成 都

王祺 硕 士

入迷(成都)信息技术有限公司 成 都

（姓名、职称、单位名称）

申请学位级别 硕士 专业学位类别 工 程 硕 士

工程领域名称 软 件 工 程

提交论文日期 2018.9.6 论文答辩日期 2018.11.23

学位授予单位和日期 电子科技大学 2018 年 12 月

答辩委员会主席 _____

评阅人 _____

注 1：注明《国际十进分类法 UDC》的类号。

Design and implementation of smart park platform on Django

A Master Thesis Submitted to

University of Electronic Science and Technology of China

Discipline: **Master of Engineering**

Author: **Niu Ning**

Supervisor: **Prof.Shekun**

School: **School of Information and Software Engineering**

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得电子科技大学或其它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谢意。

作者签名：牛宁　　日期：2018 年 11 月 29 日

论文使用授权

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

作者签名：牛宁　　导师签名：

日期：2018 年 11 月 29 日

摘　要

近年来，随着移动互联网、大数据、物联网等技术的广泛运用，信息化、数字化渗透进人类社会的更多方面。智慧城市便是在这一大背景下催生出来的新兴概念。智慧城市建设是一个分阶段、渐进式的庞大系统工程，智慧园区作为其在园区管理方面的一个具体应用，旨在通过信息化手段转变传统园区管理模式，以达到园区方与企业方，企业方与企业方资源信息及时流通共享，降低企业运营成本，提升产能集聚效应，提高生产效率，增强园区方服务能力的目的。

本文以浩旺产业园智慧园区一期工程系统建设为论文研究对象，重点研究基于 Django 技术以前后端分离方式实现智慧园区平台。系统包含园区管理、孵化服务及系统功能管理三大模块，园区管理模块包含了公告管理、意见反馈管理、访客管理等子模块，覆盖了园区管理方日常管理工作中可以进行信息化管理的绝大部分，同时为入驻企业之间，企业与园区之间提供了互动、交流的窗口。孵化服务模块主要涉及代理注册、金融服务、人员招聘等，该模块主要起到整合园区内、企业间信息资源，对外提供便捷的一站式企业服务的目的。系统功能管理模块主要提供如菜单管理、推送管理等系统内部功能。

系统以 Django 技术为基础，服务层基于 RESTful 风格进行开发，主要使用基于 Django 实现的第三方框架 Django REST framework 对外提供 RESTful API 服务，后台内容管理系统(Content Management System)则采用传统的服务端渲染技术实现。系统使用 MySQL 作为数据库，采用 Redis 对实时性要求不高的 API 进行缓存，以提供更好的并发处理能力及更好的用户体验。所有数据接口可针对不同角色开放不同的访问权限，同时可针对不同角色设置接口访问频率，以确保接口的安全性、健壮性。系统采用前后端分离方式进行开发，能够以最低成本满足多客户端访问需求，同时，采用 Django 基于 APP 模式进行开发，使得系统具有较好的可扩展性，为后期工程中需要陆续建成的视频监控 3D 综合管理系统、巡更系统、周界防护系统等提供了基础。

园区方认为平台的建成将能很好的解决现阶段面临的问题，对园区产生服务销售的规模效应，一方面能提升企业形象，另一方面能通过提供现代化服务手段，整合园区内各企业间信息资源，避免信息孤岛的存在，彻底盘活园区资源，达到服务效能的最大化。

关键词：智慧城市，智慧园区，RESTful，移动互联网，多端分离

ABSTRACT

In recent years, as mobile Internet, big data, Internet of things and other technologies have been widely used, informatization and digitalization have permeated more and more aspects of human society. Smart city is a new concept in this context. The construction of smart city is a huge systematic project in stages and gradually. Smart park as one specific application in park management aims to reach the park with the enterprise, enterprise with enterprise, and enterprise resource sharing timely flow of information, reduce business operating costs, improve production agglomeration effect, improve the production efficiency, enhance the purpose of the park service ability via the information means.

This thesis takes the phase I engineering system construction of the Hao Wang smart park as the research subject and mainly focuses on how to implement this system based on Django technology with front and end separation. The system includes three modules: park management, incubation service and system function management. The park management module includes such sub-modules as announcement management, opinion feedback management and visitor management. It covers the vast majority of information management that can be carried out in the daily management work of park, and provides a window for interaction and communication between enterprises and the park. The incubation service module mainly involves agent registration, financial services, personnel recruitment, etc., which is mainly for the purpose of integrating information resources within the park and between enterprises and providing convenient one-stop enterprise services. System function management module mainly provides such as menu management, push management and other internal functions of the system.

The system is based on Django technology, and the service layer is developed based on RESTful style. Django REST framework, a third-party framework based on Django implementation, is mainly used to provide RESTful API services, while the background Content Management System adopts traditional server-side rendering technology. MySQL is used as the database in the system, and Redis is used to cache apis with low real-time requirements to provide better concurrent processing capability and better user experience. All data interfaces can open different access permissions for

different roles, and the access frequency can be set for different roles to ensure the security and robustness of the interface. The system is developed with front and end separation, which can meet the requirements of multi-client access at the lowest cost, at the same time, Django is adopted to develop the system based on APP mode, which makes the system have better scalability and provides the foundation for video monitoring 3D integrated management system, patrol system and perimeter protection system that need to be built successively in the later project.

The park side thinks the platform will be very good to solve the problem facing at the present stage. It will service sales scale effect, on the one hand, it can improve the enterprise image, on the other hand, it can integrate the information resources among enterprises in the park by providing modern service means, avoid the existence of information islands, and fully activate the resources of the park to maximize the service efficiency.

Keywords: smart city, smart park, RESTful, mobile Internet, multiterminal separation

目 录

第一章 绪 论

1.1 论文背景及研究意义

1.1.1 论文背景

近年来，全球城市化建设发展速度加剧，城市化建设所需的各种物质资源、信息资源和智力资源等如何最大化得到合理分配与利用[1]，成为目前影响城市化建设进程至关重要的因素。智慧城市这一概念便是为解决城市资源合理统筹问题而提出的。

智慧园区作为智慧城市的一个缩影，是其在园区管理方面的一个具体应用。智慧园区主要依托于当下最为流行的大数据、云计算、移动互联网、物联网等信息化技术手段，旨在将传统的园区管理模式、服务模式“智慧化”。

浩旺产业园区由产业园区主体和周边城市综合体构成，园区方希望通过投入建设智慧园区管理平台，使用信息化手段切实有效地将园区内信息、入驻企业信息、周边商家信息串联起来，达成人与人，物与物以及人与物之间的紧密连接，以达到产能集聚，资源及时共享，生产效能最大化等目的。

1.1.2 研究意义

浩旺产业园区作为川内智慧化园区实践的一个典型，一直致力于积极探索基于产业背景，专业孵化和风险投资[2]并存的园区发展模式，将建成基于产业背景的复合生态型孵化器作为长久发展的目标[3]。以达到提升园区管理水平，强化孵化服务水平的目的。

目前园区方面临的迫切问题是缺少一个一站式的服务窗口，该服务窗口需要能够对园区业主提供有效便捷的服务通道，对服务效能进行量化考核，对业主服务要求做到实时反馈，同时能就园区规模产生服务销售的规模效应，能接入第三方服务。平台的投入建成将满足园区方这一方面的需求。该项目拟分为多期投入建设，本文所实现项目为一期阶段，旨在主要完成园区管理、孵化服务这两项主要功能，后期随着对 IT 支撑投入力度的加大，将会建成视频监控 3D 综合管理系统、巡更系统、周界防护系统等。有了平台以后，园区方认为既提升了企业形象，也提供了现代化的服务手段。对于公司在积极探索实践基于产业背景的复合生态型孵化器建设方面踏出了坚实的一步。

因此，本文将围绕以上需求进行智慧园区平台项目的设计与开发，以解决企

业当前面临的实际需求。

1.2 国内外研究历史与现状

在全球范围内，智慧城市建设正在飞速发展。智慧园区作为践行智慧城市建设理念的先行者，作为其在园区管理方面的一个具体应用，其建设发展如何做到科技化、智慧化、合理化，显得尤为重要。

近年来我国在制造业、高新技术产业等方面发展势头强劲，对与之需求配套的各类产业园、工业园、科技园、物流园区建设要求也越来越高。传统的园区经营管理模式已不能满足当下园区发展建设要求。通过引入流行的物联网、大数据、云计算、移动互联网等信息化技术手段，能很好地解决传统管理模式下引发的连接效应薄弱、信息孤立封闭的问题，通过智慧化管理手段进行资源整合，统一监测布控、智能数据分析、智慧感知响应等方式，便能很好地改变政府与企业之间、企业与企业之间、企业与公众之间的交互模式，通过将园区中相对分散的信息基础设施、社会基础设施以及商业基础设施进行有效连接，园区资源将得到最大限度的利用，企业的生产效能也能得到极大地提高。产业园区智慧化建设一方面可以对园区内相关产业有所帮助，提高园区的管理水平与服务水平。另一方面，作为智慧城市的缩影，建设智慧园区也可以反过来给智慧城市的建设提供宝贵的经验[4]。

2014年3月，国务院印发《国家新型城镇化规划（2014-2020）》，将智慧城市建设纳入规划之中[5]。2015年3月，工信部表示将继续加大对智慧城市等领域的投资。

国家不断加大智慧城市建设的投入，使得作为智慧城市缩影的智慧园区建设也因此得到了长足的发展。

我国园区发展正在历经从传统的工业园区向经济开发区、高新技术开发区、智慧园区等新兴产业园区方向过渡转型的阶段，主体形式上呈现从低级到高级，从单一管理向综合发展的特点[6]。传统园区力图通过高科技手段对产业结构进行调整，对服务内容进行升级，逐步打造成为拥有丰富社会服务职能和多产业聚集的新兴产业园区。北京经开智慧园便是国内在园区智慧化管理方面成功实践的一个典型，通过统一的智慧云服务平台对园区进行管理，使得园区内各项垂直服务能够以插件形式进行定制化开发，最终形成相互支持的智慧化整体。除此之外，国内还有其他关于智慧园区系统的成功案例，例如海通安恒智慧园区系统，路通物联智慧社区系统，锐捷智慧园区系统，蜂鸟视图智慧园区系统等。

现阶段国内智慧园区发展从空间上呈现集中式的分布特点[7]。东部地区以环渤海、长江三角区、珠江三角地区为代表，是国内轻、重工业、制造业发展最早、

最集中、最成熟的区域，基础条件的先天优势促使这些地区智慧园区建设发展较为规模化、成熟化、体系化。中部地区则主要依托沿江城市群空间格局及联动效应[8]，大力致力于将已有传统园区升级为智慧园区。西部地区由于政府近年来对第三产业，特别是高新技术产业建设地持续投入、使得以成都天府软件园区、西安软件园、贵州大数据中心等为代表的智慧科技园区相继落成建设和发展扩张。未来中西部地区将会是智慧园区发展建设的热土[9]。

国外方面，智慧园区建设起步较早，欧洲有以Living Lab为代表的智慧园区。其中荷兰在智慧园区建设方面取得的成绩比较突出。埃因霍温高科技园被评为“世界上最智慧的园区”，其前身是飞利浦高科技园[10]。园区设计和经营者认为作为具有时代感的现代智慧园区，不仅需要为入驻企业及公众提供现代化的办公场所以及营造舒适温馨的办公环境，其最为重要的是需要围绕现代企业所需的“开放式创新”提供一个充满活力的商务生态系统[11]。园区内拥有来自不同领域的企业，园区通过智慧云系统引入大数据、云计算分析平台等高科技手段，经过科学分析，不断挖掘出企业与企业之间的共性与连接，使得这些看似毫不相关的企业之间找到了共同合作创新的赢利点。

在智慧园区建设工作中信息化建设是重中之重，智慧园区信息化建设是指通过IT信息技术将各类资源整合，将智慧化管理渗透到园区建设与运营的每个细节，加强园区业务、服务和管理能力，创新组织架构，以达到园区创新性发展与可持续性发展的目标[12]。智慧化园区信息化建设要求将整个园区看做由多个互联互通的子系统构成的统一整体，强调园区与企业、企业与人之间的互动性，园区服务与管理不再是独立封闭的个体，而是相互有机融合在了一起。

智慧园区信息化建设按其功能架构应包括园区基础设施层、园区平台层、园区软件服务层和客户端四个部分。按其内容可大体划分为智慧招商、智慧办公、智慧环境、智慧生活、智慧服务、智慧管理、智能安全、智能停车、智能楼宇和智能分析几个部分，内容涉及面需覆盖包含服务与业务、管理与组织、架构与协同、政策与规章、技术、发展与长期、竞争与氛围几个方面。其建设目标根据不同的针对人群而有所不同，从园区运营者角度来说，需要达到管理智慧化，需要借助信息化手段为企业、员工提供高效智能的管理与服务。从入驻企业角度来说，需要达到工作智慧化，需要及时便捷地获得各类企业服务资源，例如金融服务、财务代理、法律咨询、企业培训等。从企业员工角度来说，需要达到生活智慧化，需要有温馨舒适的办公环境与完善高质的生活服务。

国外在围绕智慧化园区建设方面的宗旨便是对资源的共享与利用。这里的资源包括物质资源、信息资源以及智力资源，智慧化园区可以统一整合这些资源，

使得这些设备资源、生产资源、人力资源等最大限度的得到利用，形成具有生命力的商业生态链。这些先进的理念和实践，都是值得国内的智慧园区设计者和运营者仔细研究与借鉴的。

总结来说，智慧园区未来发展方向将会主要集中在以下三个方面：

1. 智慧园区建设将强化与园区产业的互动发展，探究新的模式。

2. 智慧园区管理与城市化管理将进一步融合，产创融合的新局面将逐步形成。

3. 电子政务与信息基础设施依然是智慧园区建设的重点。

从未来发展趋势分析，无论是国内还是国外，智慧园区设计的核心理念将会一直围绕创新性、合作性、开发性和可持续性四个方面展开。在未来的知识经济中，跨学科、跨领域、跨系统、跨背景的人才交流与企业合作将会越来越多，合理成熟的智慧园区信息化解决方案将促使更多的商机与发展空间。

1.3 论文主要研究内容

1.3.1 论文研究目标

浩旺产业园为践行成为“中小企业产业链资源整合者”、“工业地产订单服务模式先行者”的发展目标，需要一个信息化平台为园区内各商家、各企业提供一站式企业服务。园区现阶段虽已存在一部分已建成的信息系统，但是由于这些信息系统规划时间较早，在前期的需求调研方面做的不是很完善，可提供的功能比较简单、单一，已不能满足当下智慧园区所需的一站式服务的业务闭环。平台的规划建设正是很好地解决了这一点。本文所研究与实现的是智慧园区平台项目的一期部分，主要致力于为园区方提供园区管理和孵化服务两方面的信息化支撑。

平台投入使用后能够很好地整合园区内多方优势资源，解决政府、入驻企业、投资商等各方面的需求，以达到助力企业发展，促进区域经济的发展的目标。

1.3.2 论文主要研究内容

本文主要研究基于 Django 框架实现智慧园区后台管理系统及基于 Django REST framework 框架为不同类型客户端提供基于 RESTful 风格的 API 服务。主要研究内容如下:

1. 研究国内外智慧园区发展模式、成功案例及相对应的解决方案。

2. 对智慧园区平台系统中所采用技术进行选型，设计出高效、易扩展的系统架构。

3. 整理分析客户需求，通过用例图方式描述确定业务及功能需求。

4. 结合浩旺产业园区智慧园区平台建设实际需求，分别针对园区管理、孵化服务、系统功能管理三大模块划分出其下子模块。梳理子模块中业务流程，确定系统中所需涉及实体并对实体进行数据模型层设计。

5. 划分出系统中所需涉及角色，确定不同角色对于各个子模块所拥有的操作权限。

6. 在接口层使用 JSON Web Token 做用户身份认证，使用基于用户、用户组、权限三权分离模式对服务层接口和后台系统接口实现基于角色的权限管理，保证接口的安全性和健壮性。

7. 设计与实现园区管理、孵化服务、系统功能管理三大模块，内容主要包含后台管理系统及接口层的设计与实现，内容涵盖路由器规划、视图层设计、认证和权限控制设计、序列化器设计、分页器设计、过滤器及搜索器设计与实现等。

8. 在系统中应用基于 Redis 的接口缓存技术，对实时性要求不高的接口采用 Redis 进行缓存以减少数据库查询，减轻服务器压力，提高响应效率。

9. 采用基于 Nginx+uWSGI 组合方式实施服务器端的部署，保证在高并发访问下系统业务流程的正常运行。

1.4 论文组织结构

本文的章节结构安排如下：

第一章绪论部分主要介绍论文背景与研究意义，详细阐述分析了国内外关于智慧园区建设方面的发展历史与现状。明确了论文的研究目标与主要研究内容，并对论文组织结构进行了详细的梳理与阐述。

第二章相关技术部分主要针对系统实现中所需涉及到的主流技术进行了详细的阐述与说明并分析了系统选用前后端分离模式进行开发的原因。

第三章系统需求分析部分从浩旺智慧园区管理平台需求分析入手，分别从功能性需求和非功能性需求两方面对实现目标进行抽象与分析。基于平台实际业务需求，对业务进行模块划分，并针对各模块实际功能需求进行阐述，同时梳理了非功能需求中需要达到的一些关键指标，为下一步系统的概要设计、详细设计、实现与测试奠定了基础。

第四章系统设计部分首先提出了平台系统的设计目标与原则，然后对系统整体架构设计进行了详细的阐述，对系统功能模块及进行了划分，对主要业务流程进行了设计。

第五章系统实现部分主要针对智慧园区平台系统中接口层和后台管理系统中

主要功能模块和业务流程实现进行了说明。详细阐述了代码实现思路与核心代码实现。

第六章系统测试部分主要就系统测试进行了说明，首先对测试计划进行了阐述，然后就界面测试和功能测试部分测试用例进行了说明，最后阐述了接口限频测试、性能测试和系统整体验收结果。

第七章全文总结与展望部分对论文研究内容进行了总结并对下一步工作计划进行了展望。

第二章 相关技术

2.1 Python

Python是一门以面向对象编程思想为核心，兼具面向过程思想的动态脚本语言。Python拥有庞大的的基础类库，其功能覆盖了网络、文件、GUI、数据库、文本等各个方面[13]，同时拥有基于PyPi托管的海量第三方类库可供使用，使得Python非常适合于爬虫、数据分析、运维、Web开发、人工智能和机器学习等开发场景。

作为一门解释性语言，Python有基于不同语言实现的解释器，例如有基于C语言实现的CPython，这是官方版本使用的解释器，也是使用范围最广泛的解释器，有基于Java实现的JPython，可以直接将Python源代码编译为Java字节码进行解释执行[14]，有基于.Net平台实现的IronPython，其可以将Python源代码编译为.Net平台下的字节码进行解释执行，还有基于即时编译技术JIT实现的PyPy解释器，该解释器很大程度上提高了Python代码的执行效率。

Python被人们称为胶水语言。因为Python可以很方便地调用不同语言写好的类库，例如在实际生产开发中，Python经常在上位机上被使用，对一些性能要求较高的场景可以使用C或者C++语言进行开发，然后Python可以直接调用这些语言写好的模块，以弥补性能上的劣势。

2.2 前后端分离

传统开发模式中，一般采用不分离或者半分离模式进行开发[15]，这种方式带来了一系列弊端。

1. 前后端开发职责及范围不清

无论是不分离还是半分离模式，都是采用服务器端渲染技术，即后端模板语言和前端HTML、CSS、JS代码嵌套在一起返回给客户端进行渲染[16]。这种模式下前端页面和后台模板语言耦合性十分严重。

这种开发模式要求前端人员至少需要熟悉后端的模板语法，同时也需要后端开发人员了解掌握前端布局的基本知识[17]。而且许多涉及到两端交互的地方很难划分出该由前端还是后端来实现。前后端人员不能做到高效的并行开发，同时需要跨领域学习对方知识，学习成本增大，效率低下。

2. 多元化客户端要求前后端做到彻底分离

Web1.0时代，企业应用一般基于PC端浏览器，随着近10年来移动互联网技术

和前端技术的发展，出现了大量面向移动设备的Web应用开发技术[18]。除了Android、iOS、Windows Phone等操作系统平台原生的开发技术之外，基于HTML5的开发技术也变得非常流行，现代的企业应用便是建立在基于这些多元化的客户端技术的基础之上的。如果针对如此多的客户端应用都需要开发各自对应的服务端程序的话，那么将使得开发成本增大。如果规定了前后端通信的数据格式，服务器端只向前端提供定义好格式的数据服务，那么只需要开发一套服务器端程序便可适配不同的客户端，前后端分离技术正是多端适配解决方案的最佳实践[19]，能很好的节约人力、时间成本，缩短开发周期，对后期扩展提供良好的伸缩性。这也是本系统选用前后端开发方案的最主要原因。

2.3 RESTful

RESTful架构既不是一种标准也不是一种规范，而是一种风格，一种约束，一种软件设计理念。简单来说，便是通过HTTP动词加资源路径的方式来操作资源[20]。

REST是由资源、表现层、对象转化三部分组成的。其中资源可以理解为互联网上由统一资源定位符指向的一个实体[21]。表现层是资源的状态，即资源的表现形式，通俗的说就是该资源是什么格式，浏览器应该如何解析渲染这个资源给用户，是HTML文件还是JSON数据还是文本文件等，在HTTP响应头里应该有相应字段标识这些信息。对象转化则指在客户端和服务器端交互过程中，客户端的一系列操作引起的服务器端数据的变化[22]。

如果满足以上三点特性对外提供服务的API接口就可被称为RESTful API，RESTful API的优点是：

1. 接口风格统一，语义明确

RESTful API都是通过HTTP动词搭配URL来操作资源的，不同动词对应对资源的不同操作，从而触发资源状态的改变。一旦比较熟悉这种设计理念后，接口的设计和定义将变得非常迅速，使得无论是前端还是后端开发人员都能很快的着手于各自的开发工作，提高了开发效率。

2. 数据以通用的 JSON 格式定义，十分高效轻量

传统的Webservice以SOAP协议为基础，数据采用XML进行描述。

REST相比SOAP来说十分轻量，数据格式采用JSON，直观，易读性好。

REST出现以前，前端JS想要访问基于SOAP协议的Webservice需要先访问后台代码，然后后台代码通过公共服务代理类再访问服务，而REST是基于HTTP协议的，JS可以非常方便的直接访问服务接口。

JSON对象的序列化与反序列化十分高效，JS和Python都内置了模块进行解析，

无需像解析XML格式数据一样需要借助其他的一些第三方类库来完成。

3. 面向资源开发，接口定义清晰

更加易于抽象实际业务场景，前后端联调更加平滑、分离性更好。

而RESTful API风格的设计接口也存在一些缺点，最明显的是在表达一些业务时灵活性较差，因为标准的RESTful API是纯面向资源的，为了抽象出一些用户操作，比如添加收藏、删除收藏等必须在用户与资源上再添加一个模型以便关联这两者。

2.4 Django 框架

Django是一个由Python语言编写的具备完整架站能力的开源Web框架[23]。使用Django框架，程序开发人员可以十分快速地完成一个正式网站所需要的大部分内容，通过集成具有不同功能的第三方扩展，可以进一步开发出全功能的Web服务。

相比于Flask这种轻量级框架，Django框架本身集成了一套功能强大完整的解决方案，从模型层的到视图层再到模板层，针对缓存、日志系统、发送邮件、资源聚合(RSS/Atom) [24]等各式各样常见的应用场景，通过简单的配置调用这些集成好的模块可以非常快速的完成功能实现。但是从另一方面来看，由于各模块之间结合得比较紧密，所以在功能强大的同时显得相对封闭。不过类似Flask这种轻量级框架由于框架本身只提供非常简单的核心功能，开发者需要自己去实现Web服务中一系列通用功能或通过集成第三方类库来解决，重复造轮子降低了开发效率，而且从最终效果来看，实际上变相的生成了另一个“Django”，所以Django的核心理念便是 DRY(Don't Repeat Yourself) [25]，鼓励开发者快速构建自己的应用，这一点也十分符合当下对应用开发快速交付上线提出的需求。同时，Django框架本身是基于插件式开发的[26]，只要符合Django插件开发规范，开发者本身可以自行开发出高度可复用的模块或者十分轻易的集成其他开发好的模块，开发效率将得到成倍提升[27]。Django框架基本架构如图2-1所示。

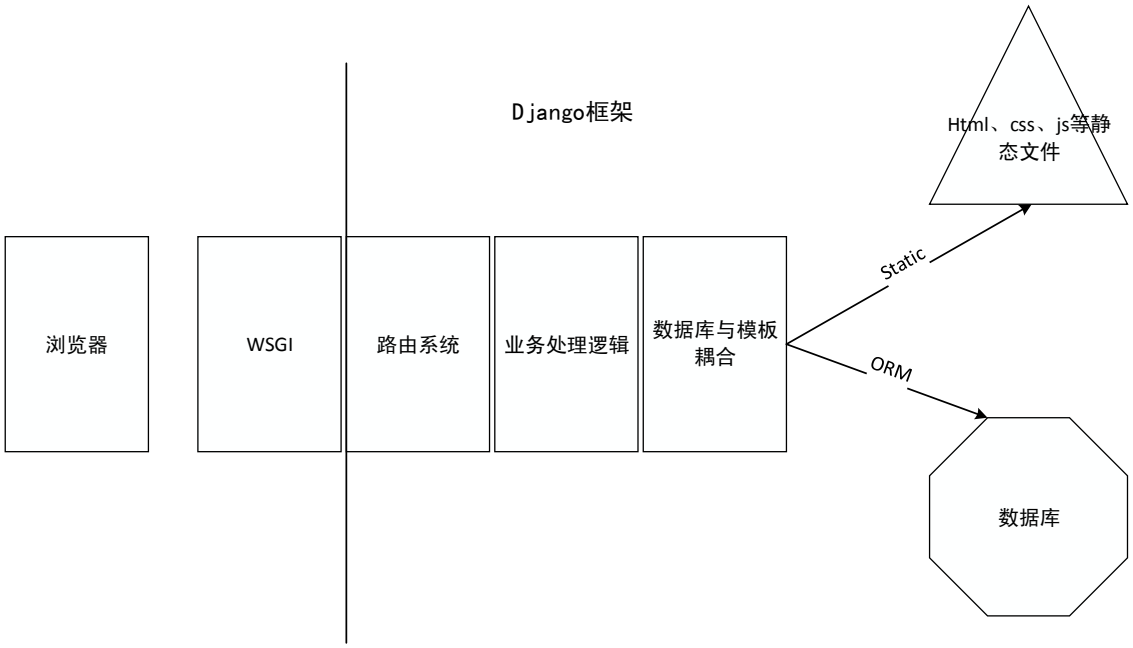

图 2-1 Djiang 框架基本架构

2.5 Django REST framework

Django REST框架(以下简称DRF)是基于Django进行开发的一个第三方辅助类框架，其主要目的是为构建基于RESTful风格Web API提供强大而灵活的工具包[28]。DRF现已发布到v3版本，框架本身提供了十分强大完备的功能，其本身提供的可浏览API接口使得可以通过浏览器即可对API进行各种测试，开发者依照DRF规范开发完成后可以直接生成API接口文档，而且该文档同时提供了可交互界面以及针对不同前端语言的测试接口。同时DRF框架能很好的支持基于OAuth1和OAuth2的身份验证策略，支持基于ORM或者非ORM的数据源序列化[29]，DRF框架实现并对外提供基于请求、响应、视图、通用视图、视图集、路由、解析、渲染、序列化、验证器、认证权限控制、限流、过滤、分页、版本控制、内容协商等一系列API[30]，同时DRF还拥有丰富的文档和良好的社区支持，可以说是基于Python语言开发RESTful API最为成熟的一套解决方案。

2.6 MySQL

MySQL 是最为著名且应用最为广泛的关系型数据库，其体积小、速度快、通过配合不同的存储引擎，可以满足不同的业务场景。

MySQL 逻辑架构整体分为三层，最上层为客户端层。针对连接处理、授权认证、安全[31]等功能均在这一层进行处理。

中间层集中了 MySQL 绝大部分核心服务，其主要由查询缓存，分析器和优化器组成[32]，处理包括查询解析、分析优化、缓存、内置函数等业务。所有的跨存储引擎的功能也在这一层实现，包括存储过程、触发器、视图等。

最下层的存储引擎层主要负责 MySQL 中的数据存储和提取[33]。和 Linux 下的文件系统类似，每种存储引擎都有其优势和劣势，比如 InnoDB 可提供基于 ACID 的事务支持，MyISAM 具有较高的查询插入速度等。中间的服务层通过 API 与存储引擎通信，这些 API 接口屏蔽了不同存储引擎之间的差异，向上提供一致性对外服务。MySQL 组件逻辑架构如图 2-2 所示。

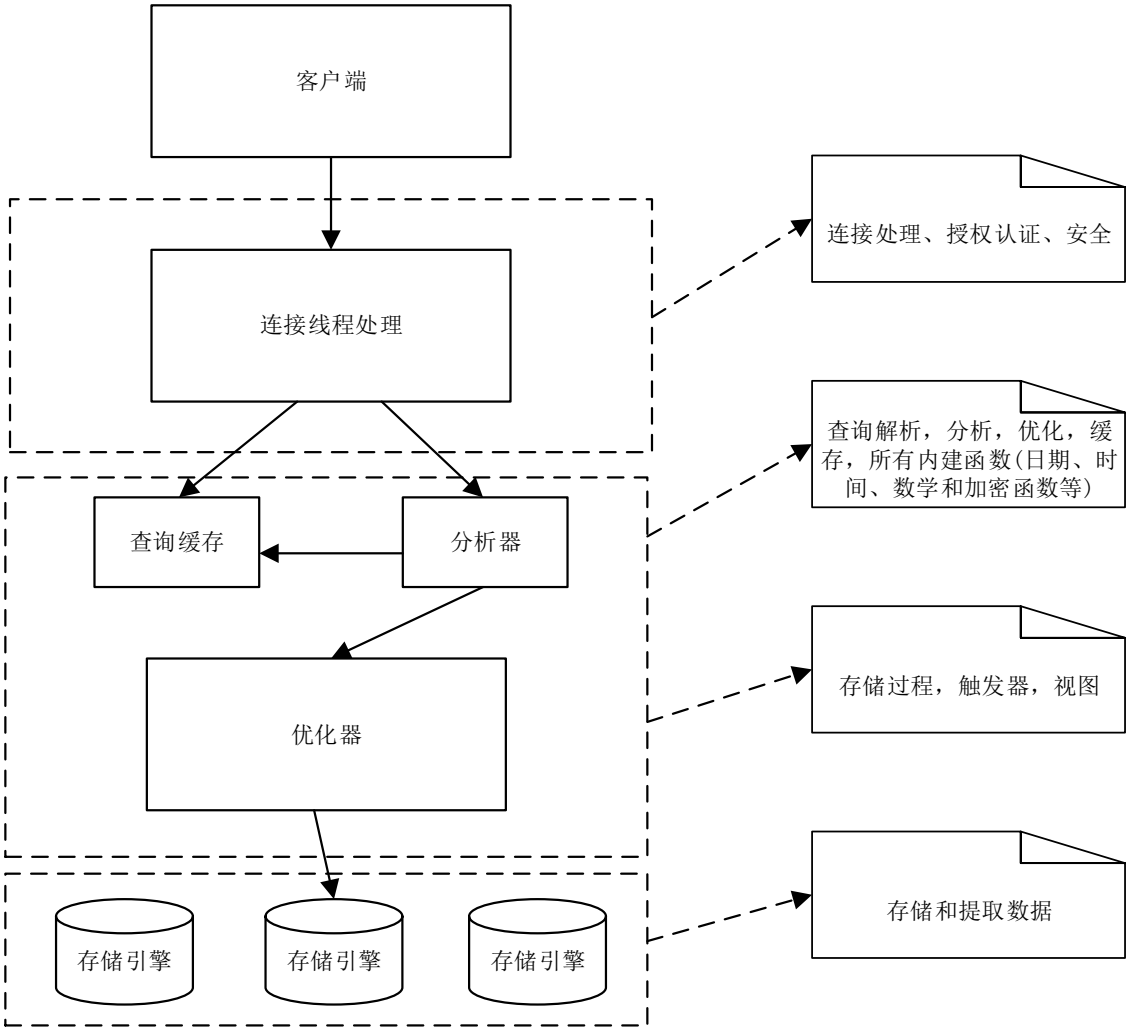

图 2-2 MySQL 组件逻辑架构图

2.7 Redis

Redis 是一个开源的，基于内存存储的非关系型数据结构存储系统，它可以用作数据库、缓存和消息中间件[34]。 Redis 支持多种类型的数据结构，如字符串

（strings），散列（hashes），列表（lists），集合（sets），有序集合（sorted sets）[35]与范围查询等。Redis 支持对内存数据的持久化，如果我们不使用此功能，可以将其作为高速的网络缓存使用，通常我们采用 Redis 缓存一些对实时性要求不高的数据以减少频繁的数据库读写造成的性能瓶颈。

2.8 本章小结

本章针对基于 Django 的智慧园区平台系统设计与实现中所需涉及到的主要技术进行了详细的阐述与说明。主要分析了系统选用前后端分离模式进行开发的原因。Djaong 和基于 Djaong 开发的 Django REST framework 框架封装了功能丰富的各种模块并对外提供基于插件形式的可插拔接口，开发者可非常灵活的自行开发模块或者组合其他第三方模块为自己所用。正是由于这些原因，选取 Django 和 Django REST framework 为主要技术栈的解决方案为后续系统的实现与持续交付提供了可靠的保证，同时为系统后期针对不同客户端的多端适应提供了基础和前提。

第三章 系统需求分析

3.1 需求调研

浩旺产业园区自建成以来，已从最初的只有几家入驻企业发展成为现在拥有上百家企业及商铺租户的中小型产业园区。服务对象的增加及生产规模的扩大对园区管理工作提出了更高的要求。同时，随着入驻企业业务的越来越多元化，企业之间实际存在很多合作共赢的潜在联系，企业之间希望寻求到更多的合作，一些生产企业希望能将自己过剩的产能通过代工等模式发挥到最大限度，这就要求园区方能提供一个平台及时收集、共享、发布这一些需求。作为以“成为全球中小型产业园领航者”为宗旨，致力打造中小型企业的优质发展环境，为投资者搭建良好的事业起步和扩张平台，创建西部中小企业腾飞基地为发展目标的现代化产业园区，就园区目前现状而言，应对这一些需求，对园区实现全面智慧化升级是下一步发展工作的重点，智慧化园区建设实际上是一个涉及面十分庞大，需要长期投入建设以及需要园区方与企业方通力合作才能逐步完成的项目。园区方希望通过投入多期工程逐步达到能成为服务于中小型企业比较成熟的智慧园区典范。园区在早期已投入建成一系列信息化系统，不过由于这些系统规划于园区建成之初，主要目的是为了招商引资而服务，偏重于企业展示、招商引资立项、租赁合同管理等需求。园区现在已完成招商引资、企业入驻的第一阶段。对于现阶段而言，园区信息化管理以及孵化服务是园区建设最为迫切的需求，园区目前还没有已有系统能完全满足这些需求，现投入建设智慧园区一期工程正是为解决园区方最为急需解决的这两点而服务的。

3.2 功能需求分析

根据对智慧园区平台需求的调研与分析，系统可主要概述为以下三部分。

第一部分是客户端，客户端面向使用群体主要针对入驻企业、企业员工及园区方特定人群，例如园区方维修工人、保安等。企业方可以通过客户端及时获取园区方发布的各种信息，及时方便的获取园区方提供的各种服务，同时企业方也可以提交自己的一些用工信息，租赁需求，代工需求等给园区方审核，一旦信息审核通过，将展现在各自对应分类的子模块中。园区方维修工人可以利用客户端进行维修作业进度的上报查询等。

第二部分是服务层，主要针对不同的客户端提供具有统一数据接口的数据。

实现园区管理和孵化服务等功能。

第三部分是后台管理系统，主要供园区管理人员使用。所有信息的录入、查询、修改、增加均可通过该系统进行操作。主要功能包括公告管理、活动管理、周边商家管理，报修管理，缴费管理等。

综合考虑项目前期建设背景以及后期业务发展将本次浩旺园区项目的建设分为三个主要模块。分别是园区管理模块，孵化服务模块和系统功能管理模块。

3.2.1 园区管理

园区管理模块主要由公告管理、意见反馈管理、活动管理、周边商家管理、广告管理、访客管理、物流快递管理、服务管理子模块构成。

1. 公告管理

公告管理模块集中管理园区方发布的各项通知和公告。公告类型一共有四种，分别是通知、公告、制度和标准。园区管理方可以通过后台管理系统创建、修改、删除、发布公告。客户端将会在对应的版块对这些公告进行分类展示以便于企业方业主能快速阅读管理处发出的例如停水、停气、设备维护、网络故障通知等消息。公告管理用例如图 3-1 所示。

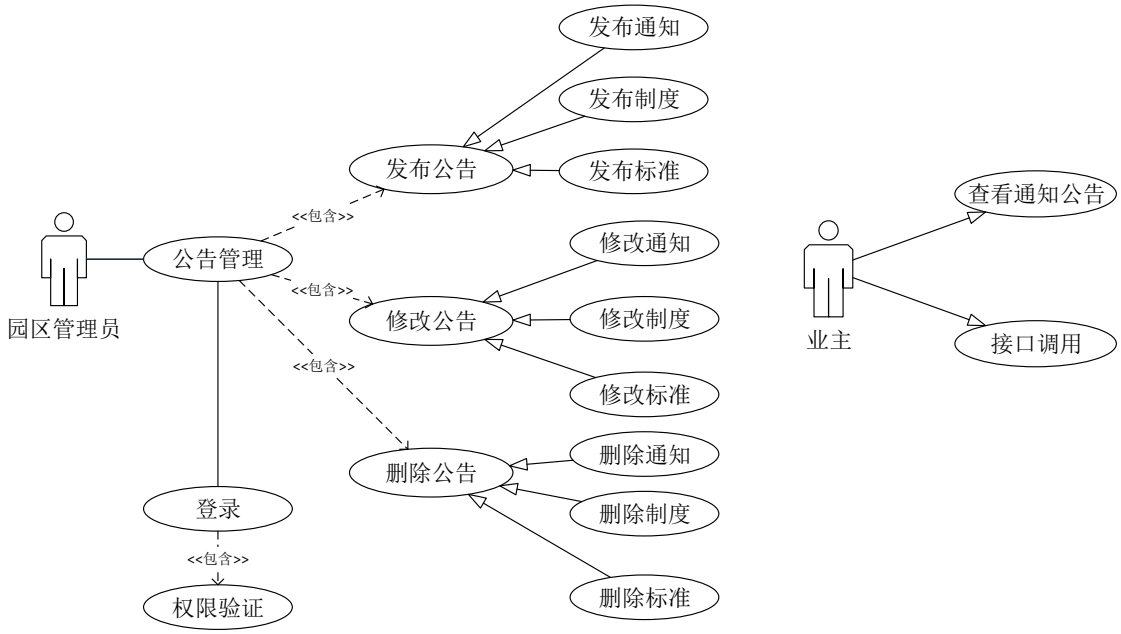

图 3-1 公告管理用例图

用例说明如下：

园区管理方拥有管理员权限的用户登录后台管理系统，针对可预计的或接到相关部门通知的如停水、停气、维修等情况通过后台系统发布通知，针对政府、

团体、法律法规等相关信息通过选取发布类型为公告选项进行发布，针对相关制度规定方面的信息通过选取发布类型为制度选项进行发布，针对一些生产、施工等行业最新标准的发布，旧标准的一些废除修订等通过发布类型为标准选项进行信息发布，以前企业方能在第一时间内获取相关的最新动态信息。

业主可在登录或者未登录状态下进入通知公告版块，及时查看获取相应的信息。此接口调用不对用户登录进行验证，为公共接口。

2. 意见反馈管理

业主可通过在 APP 端个人中心里意见反馈版块向园区管理者提交意见反馈信息等。管理者可以在后台系统查看相关意见反馈，并可指派相关处理人员后续跟进意见反馈并添加处理结果。

3. 活动管理

园区入驻企业或者商家可以向园区管理方申请举办联谊、篮球赛等活动，活动也可由园区管理方主动进行发布，园区业主可在客户端企业活动版块查看到对应活动信息，并且可通过点击进入活动详情进行活动报名，园区管理者可在后台查看到报名信息详细信息以及统计出活动报名总人数等。企业活动管理用例如图 3-2 所示。

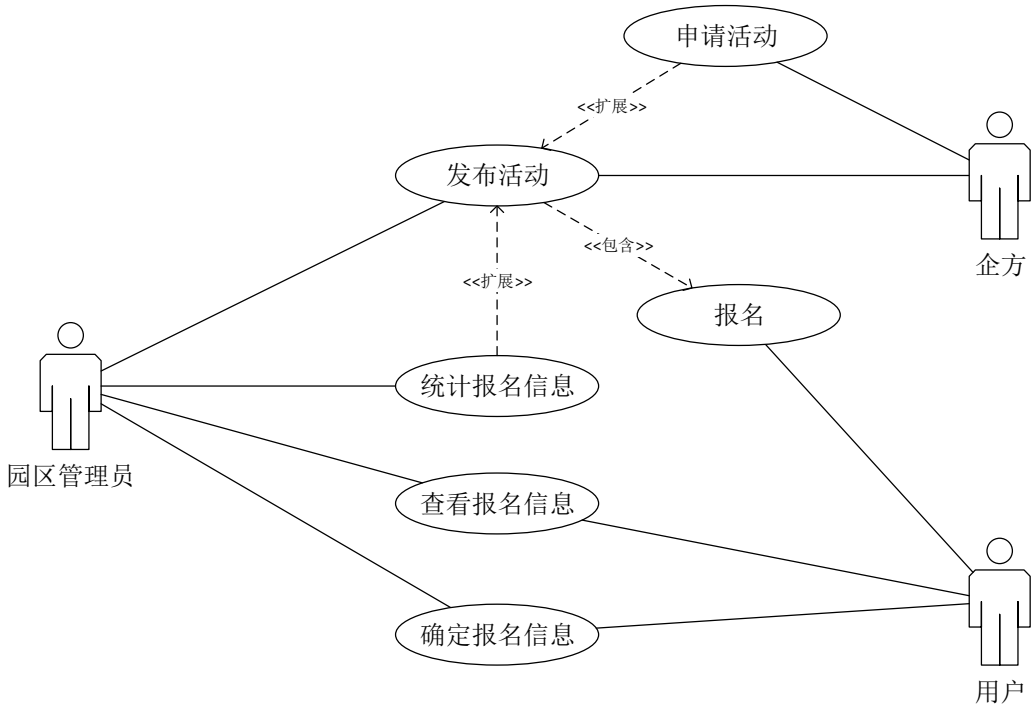

图 3-2 企业活动管理用例图

用例说明如下：

园区方可自己发布园区举办的活动，企业方如需发布自己举办或者与其他企

业合作举办的活动需提前联系园区方由园区方统一通过后台管理系统进行活动的发布，登录 APP 端的用户可以报名参加活动并留下联系方式。园区方管理员可以在后台查看报名信息以及获得活动人数统计等信息。

4. 周边商家管理

该模块集中管理园区范围内城市综合体内入驻商家，该模块主要针对经营餐饮类商家进行管理，周边商家属性包含：商家名字、商家简介、商家地址、商家类型、商家联系电话、商家联系人、商家评价星级、是否推荐、人均消费、添加时间、修改时间等。

5. 广告分组管理

广告分组管理可对 APP 端不同板块的广告进行分组管理，相关属性包括名称、描述、是否启用、是否可自动生成。

6. 广告管理

广告管理属性包括名称、类型、所属分组、权重、开始时间、结束时间。代码用于 APP 端对广告所在广告位进行标识，权重用于对同一个分组下广告先后顺序进行排序。开始时间和结束时间分别标识该条广告的整个生命周期。

7. 访客管理

访客管理模块包括访客详情和访客登记两个子功能，访客登记主要是针对园区保安来使用的。访客管理用例如图 3-3 所示。

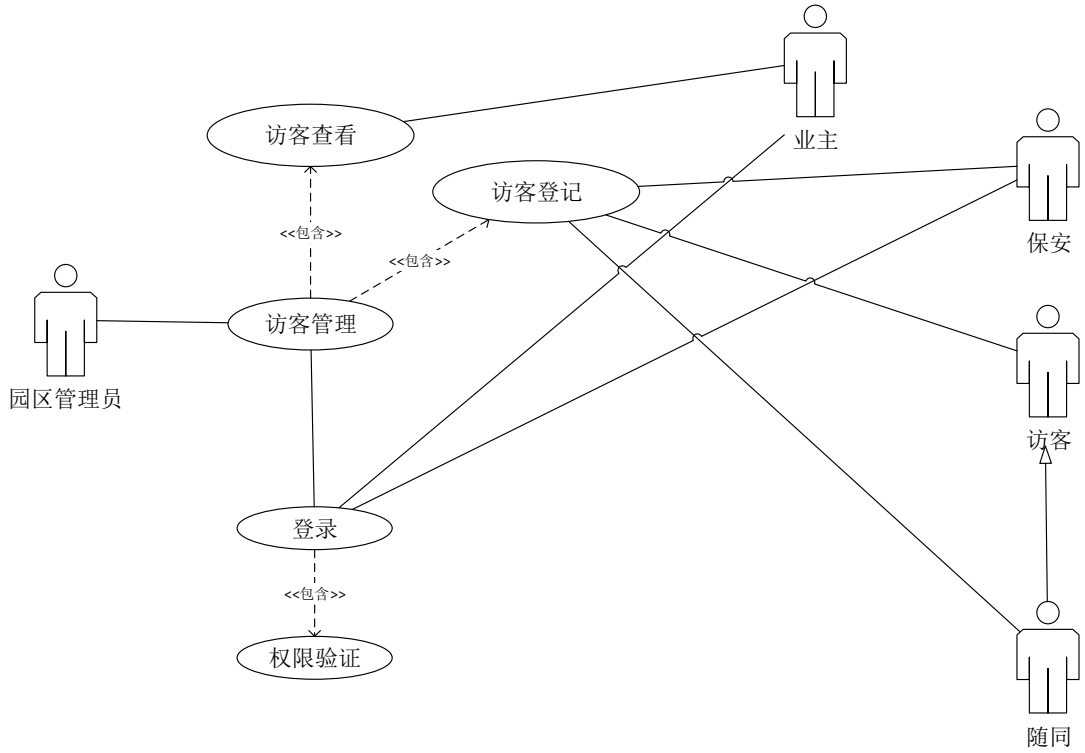

图 3-3 访客管理用例图

用例说明如下：

当有到访人员的时候，由楼宇保安使用 APP 录入到访人员的数据。录入数据包括：来访者姓名、性别、电话、来访人数、身份证、到访地点、负责人，身份证照片，身份证可填写多个，身份证和身份证照片非必填项，如未填写身份证和身份证照片，则需要填写备注。

APP 端到访人员列表由时间倒叙显示来访者信息，信息包括：来访者姓名、性别、电话、来访人数、身份证、到访地点、负责人。访客管理列表上方有到访楼栋单元和到访时间的筛选条件，可以根据需要缩小查询范围。

到访详情界面显示：访客姓名、性别、电话、来访人数、身份证、到访地点、负责人、身份证照片。身份证照片如果有多张，显示所有。如果没有填写如未填写身份证和身份证照片，则在相应区域显示备注信息。

8. 物流快递管理

后台发布各物流公司的信息，物流公司需与园区绑定。信息包括：物流公司名、联系电话、logo 图片、介绍。能够通过 APP 搜索到周边的物流公司的信息，如果有寄快递需求，通过在 APP 上的功能直接拨打物流公司的电话联系物流公司上门服务，实现足不出户寄包裹的需求。

快递列表实现置顶功能，如果物流公司与园区有进行合作的话，可以选择性地将有合作的物流公司电话置顶。

9. 服务管理

服务泛指物业服务窗口可以向业主提供的服务，业主可以自由发起各种类别的服务诉求（系统交付后，系统使用单位可以调整表单）。系统后台显示由 APP 发起的服务诉求列表。

业主如果有报修之类的服务需求，通过联系园区人员提出自己的需要服务的内容。园区人员此时在数据库中录入服务数据，录入数据包括：业主名、服务描述、业主电话、负责人、负责人电话。并在后续服务推进过程中录入服务进度，录入数据：进度信息。

服务管理列表界面显示业主申请服务的记录。园区管理人员在这里可以查看到所有业主申请服务的信息，列表显示的数据有：业主名、服务描述、业主电话、发布时间、服务状态。业主在此功能上只能看到自己的所发布的服务信息。

服务信息详情界面显示当前服务内容详细数据，包括：业主名、服务描述、业主电话、负责人、负责人电话、发布时间、服务状态。同时在服务信息下面显示服务的进度，进度显示数据：进度信息，时间。

3.2.2 孵化服务

孵化服务模块主要由联系方式管理、代理注册、办事指南、车辆物品租赁、云代工、保洁服务、金融服务、财务代理、法律咨询、会议室租赁、知识产权、企业培训、人员招聘、申请须知、视频观看子模块构成。

1. 联系方式管理

联系方式管理模块主要管理园区各部门联系负责人信息。业主可以通过 APP 端查看各部门负责人信息，如果联系电话等，可以通过 APP 端拨打电话联系相应负责人。

2. 代理注册

代理注册模块主要提供帮助园区企业完成公司注册等一系列流程化服务。

后台接收企业在 APP 提交的代理注册申请信息，管理人员对提交的信息进行初步审核，审核通过根据接收到的信息线下联系申请人进行后续注册。根据自身情况填写代理注册申请的表单，内容包括：选择注册类别、经营范围，填写公司名、法人姓名、联系方式。填写完毕等待审核，审核通过后园区工作人员会主动联系申请人，并能再后台对注册进度信息进行添加，显示在 APP 供业主查看。

业主填写完成注册信息后，能够在“我的”里面的“我的申报”中注册分类里面找到代理注册的列表和详细信息。列表显示注册名、时间；详情界面显示注册信息的注册类别、公司名称、经营范围、公司法人姓名、联系方式、负责人、负责人电话。基本信息下面显示代理注册流程的进度。进度显示数据：进度信息、时间。代理注册用例如图 3-4 所示。

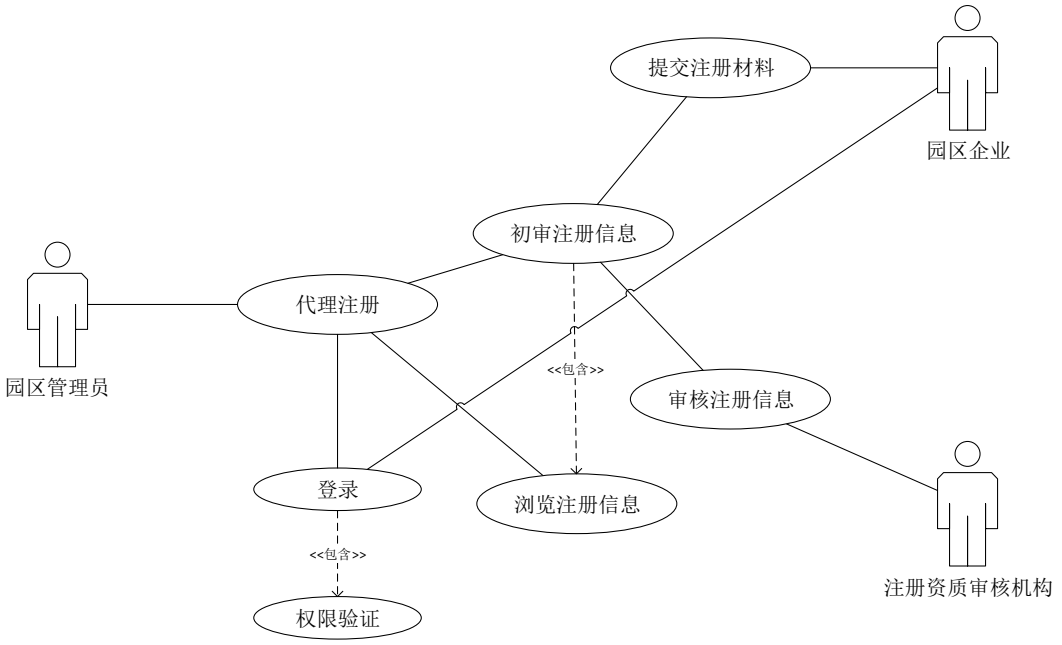

图 3-4 代理注册用例图

3. 办事指南

办事指南模块主要用于管理园区内各种问题处理及解决流程信息。

业主通过办事指南内容了解园区内各种问题处理及解决流程信息。办事指南列表显示由园区人员发布的办事指南信息。内容包括：标题，发布的时间。办事指南详情界面显示：标题、发布时间、发布人、图片、内容。

4. 车辆物品租赁

业主可以在 APP 端发布一些个人的租赁需求，园区管理方也可以在后台发布一些具体的租赁需求信息，为业主提供车辆物品等租赁服务。

点击 APP 端物品租赁界面显示园区车辆物品租赁列表。列表显示车辆物品的图片、物品名、价格、发布时间、部分介绍。业主可根据自身情况选择需要的物品租赁。租赁详情界面显示数据包括：物品名、图片、价格、发布时间、出租时间、基本信息、发布人名、联系电话、详细描述。业主也可以自行在 APP 端发布一些租赁需求，经过审核后租赁需求将在 APP 端租赁列表中显示。

租赁需求信息可以由 APP 端或者后台进行发布，发布数据包括：物品名、图片、价格、出租时间、基本信息、发布人名、联系电话、详细描述。业主如果需要租赁车辆或者物品，通过详情界面上的电话联系园区或者租赁信息发布者进行车辆物品的租赁。

5. 云代工

云代工主要是园区管理方为帮助某些产能未充分利用的企业出售剩余产能。

云代工列表显示由企业发布的代工信息。内容包括：代工的标题，发布的时间。代工信息的发布由业主联系园区人员主动提供相关的代工介绍信息，有后台录入数据库，录入数据包括：标题、内容、联系人、联系电话、发布企业。列表详情界面也是显示这些数据，如果有企业需要这些代工可以通过电话联系。

6. 保洁预约

保洁预约子模块主要为业主办公场所卫生清洁提供预约服务, 后台发布保洁需求, 业主在 APP 端可查看。

保洁预约列表显示的是提供保洁服务的公司的列表。这些公司是与园区进行合作，由后台录入进数据库中，录入数据包括：公司名、公司地址、服务范围、服务区域、联系人、联系电话、详细描述。

保洁公司信息列表显示：公司 logo、公司名、服务范围、地址、服务次数。每行列表最右边的图标提供能直接拨打保洁公司电话的功能。

保洁公司详情界面显示公司 logo、公司名、公司地址、服务范围、服务区域、联系人、联系电话、详细描述。同时提供业主能进行评价的功能，业主能够通过

其他业主评价的内容来择优选择保洁公司。当选定保洁公司后通过点击界面上的“电话联系”按钮拨打电话给保洁公司，也可以点击“立即预约”按钮，填写预约信息给后台，后台通知保洁公司联系业主，表单填写信息：姓名、服务时间、电话。保洁预约用例如图 3-5 所示。

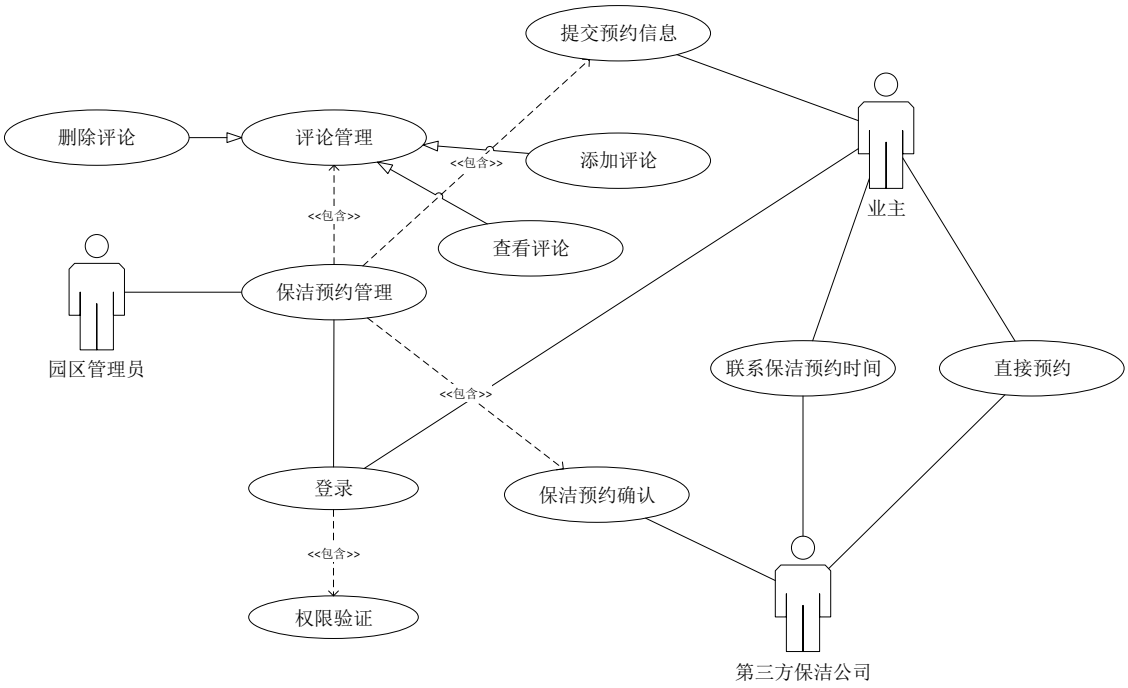

图 3-5 保洁预约用例图

用例说明如下：

园区方在后台录入与园区有合作关系的保洁公司信息，业主可以直接在 APP 端查看保洁公司详细信息，当有保洁需求时可直接在详情页通过点击电话和保洁公司直接联系。也可以在详情页面点击立即预约按钮进行预约信息的录入提交。园区方在收到预约请求后与保洁公司联系，将保洁请求信息和业主电话告知保洁公司由保洁公司与业主进行预约时间等信息的确认。业主可在详情页对相应保洁公司服务进行评价，该评价可以作为其他业主选择保洁公司的参考依据。

7. 金融服务

金服服务子模块主要是企业和园区管理员定期发布一些金融服务提供者，包括一些金融产品和联系方式，APP 端可查看。

金融服务列表显示的是提供金融服务的列表。提供这些服务的公司是与园区进行合作，由后台录入进数据库中，录入数据包括：服务标题、公司名、公司地址、服务类型、联系人、联系电话。

金融服务信息列表显示：公司 logo、服务标题、公司名称、服务类型、服务

次数。每行列表最右边的图标提供能直接拨打金融服务公司电话的功能。

金融服务详情界面显示公司 logo、服务标题、公司名、公司地址、服务范围、联系人、联系电话。同时提供业主能进行评价的功能，业主能够通过其他业主评价的内容来择优选择金融服务公司。如果选定金融服务公司后通过点击界面上的“电话联系”按钮拨打电话给金融公司，也可以点击“立即预约”按钮，填写预约信息给后台，后台通知金融服务公司联系业主，表单填写信息：姓名、电话。

8. 财务代理

财务代理子模块主要是园区方定期发布一些代理会计的服务提供信息，展示园区相关人员的联络方式。

财务列表显示的是代理会计的列表。代理会计以个人或者公司形式与园区进行合作，由后台录入进数据库中，录入数据包括：公司 logo 或者个人头像、代理会计名、公司名、公司地址、服务类型、联系人、联系电话。

财务信息列表显示：公司 logo 或者个人头像、代理会计名、公司名称、服务类型、服务次数。

财务信息详情界面显示公司 logo 或者个人头像、代理会计名、公司名称、服务类型、服务次数、联系人、联系电话。同时提供业主能进行评价的功能，业主能够通过其他业主评价的内容来择优选择代理会计。如果选定代理会计后通过点击界面上的“电话联系”按钮拨打电话给会计公司或者个人，也可以点击“立即预约”按钮，填写预约信息给后台，后台通知会计公司联系业主，表单填写信息：姓名、电话。

9. 法律咨询

法律咨询子模块主要是为园区入驻企业企业提供法律服务。

法律咨询列表显示的是提供法律咨询服务的公司列表。法律服务公司与园区进行合作，由后台录入进数据库中，录入数据包括：公司 logo、公司名、公司地址、服务次数、联系人、联系电话。

法律咨询服务信息列表显示：公司 logo、公司名称、服务类型、服务次数。每行列表最右边的图标提供能直接拨打法律咨询服务公司电话的功能。如果有咨询意向，通过拨打法律咨询公司的电话了解相关的法律服务。

10. 会议室租赁

会议室租赁子模块主要实现为园区入驻企业提供会议室租赁等增值服务。

业主可以通过在 APP 端点击进入会议室租赁界面显示会议室的列表，列表数据包括：会议图片、会议室名、适合人数、发布时间、地址、价格。会议室租赁列表上方有地点、时间、类型的删选条件，可以根据需要缩小查询范围。地点是

根据楼栋分类，类型分为：会议，办公。

会议室详情界面显示显示数据：会议图片、会议室名、适合人数、发布时间、地址、价格、联系人、联系电话、基本设备。业主可以通过点击电话来致电园区工作人员进行会议室租赁。

园区工作人员在业主租赁时将业主数据绑定到租赁的会议室上，业主可以在我的申报里面的会议室分类里面查看自己的会议室租赁信息。信息界面与会议室详情界面一致。

11. 知识产权

知识产权子模块是园区管理方为企业申报知识产权流程提供的增值服务。

知识产权 APP 列表里面的第一条为“知识产权申报须知”，这一条为置顶。点击进入详情页面介绍园区知识产权申请的具体流程和申请须知，这些流程的文字能够在后台做管理。在详情界面有申请知识产权的按钮，其功能为打电话给工作人员，业主通过点击联系园区工作人员进行知识产权的申请或了解具体情况，申报知识产权的信息由园区人员在此时录入进系统，录入数据为：专利种类、类型、申报公司、申报项目、申请人、申请人电话。录入之后，园区工作人员应对其进行审核，审核通过后显示在 APP 上面，并且在后续对申报的进度信息进行录入以供业主在 APP 上查看，录入信息为：进度信息。

在“知识产权申报须知”以下的数据是业主知识产权申报记录。业主在这里可以查看到其他公司在园区申报知识产权的申报情况，列表显示的数据有：专利种类、类型、申报公司、申报项目。在申报情况的详情界面显示：专利种类、类型、申报公司、申报项目、申报时间、申报负责人、负责人电话。同时显示专利申报的进度，进度显示：进度信息、时间。

12. 企业培训

企业培训子模块主要是园区管理员在后台发布为园区企业提供人员培训等服务，业主在 APP 端可查看和进行报名。

通过在 APP 端点击进入企业培训界面显示培训的列表，列表数据包括：培训图片、培训名、主办单位、发布时间、报名状态。状态分为：已报名、未报名。

进入详情界面，界面显示信息包括：培训名称、主讲人、培训时间、主办单位、主要内容、相关资料。业主在 APP 查看、搜索人员培训信息，通过此平台了解招聘信息，如果有培训的意向，点击“立即报名”按钮来参加企业相关培训，由主办单位电话联系业主进行培训具体流程的相关安排，报名表单填写数据为：姓名、性别、年龄、联系电话、地址、备注。

13. 人员招聘

人员招聘子模块是园区方提供的另一项增值服务。主要是为入驻企业提供合适人才招聘信息。

通过在 APP 端点击招聘服务进入招聘列表界面，列表显示数据：职位、薪资、公司名、地区、学历、经验、职位描述部分。

招聘详情界面显示：职位名、薪资、公司名、公司性质、公司规模、公司地址、发布时间、人数、经验、学历、地区、职位描述、薪资福利、职位描述。园区入职企业可以通过此模块在 APP 端发布招聘信息，业主通过此平台了解招聘信息，如果有求职意向，通过申请职位按钮填写相关的信息向园区管理员发起求职申请。园区管理员在后台收到求职申请后联系申请公司，由申请公司自行联系求职者进行面试。申请表单填写数据为：姓名、性别、年龄、联系电话、地址、备注。

14. 项目申报

项目申报子模块主要是为企业提供项目申报，增加企业价值。

项目申报页面介绍园区项目申报的具体流程和申请须知，这些流程的文字能够在后台做管理。在详情界面有申请知识产权的按钮，业主通过点击打电话联系园区工作人员进行知识产权的申请或了解具体情况，申报知识产权的信息由园区人员在此时录入进系统，录入信息为：项目名、公司名、联系人、联系电话、负责人、负责人。并且在后续对申报的进度信息进行录入以供业主在 APP 端进行查看，录入数据：进度名。

后台录入申报信息后，能够在“我的”里面的“我的申报”中项目分类里面找到代项目申报的列表和详细信息。列表显示项目名、发布时间；详情界面显示注册信息的项目名、公司名称、联系人、联系电话、负责人、负责人电话。基本信息下面显示代理注册流程的进度。进度显示数据：进度信息、时间。

15. 视频管理

视频管理子模块主要供园区方发布一些最新的短视频。

3.2.3 系统功能管理

系统功能管理模块主要由园区菜单管理、推送管理、认证和授权管理、业主管理、物业人员管理、组织机构管理、缴费管理、短信验证码管理八个子模块构成。

1. 园区菜单管理

园区方管理员可通过此模块达到对 APP 端显示菜单项的自由定制，可以调整一级菜单是否是默认显示，是否固化显示，编辑修改一级菜单下的二级菜单信息，

同时可以通过调整排序权重将某一菜单位置移动到最前面或者往后移动。同时可对链接类型进行分类，可分为功能类或者链接类，链接类的菜单点击后实际是请求到了第三方系统的接口，所以这里需要对其进行区分。园区菜单管理用例如图3-6 所示。

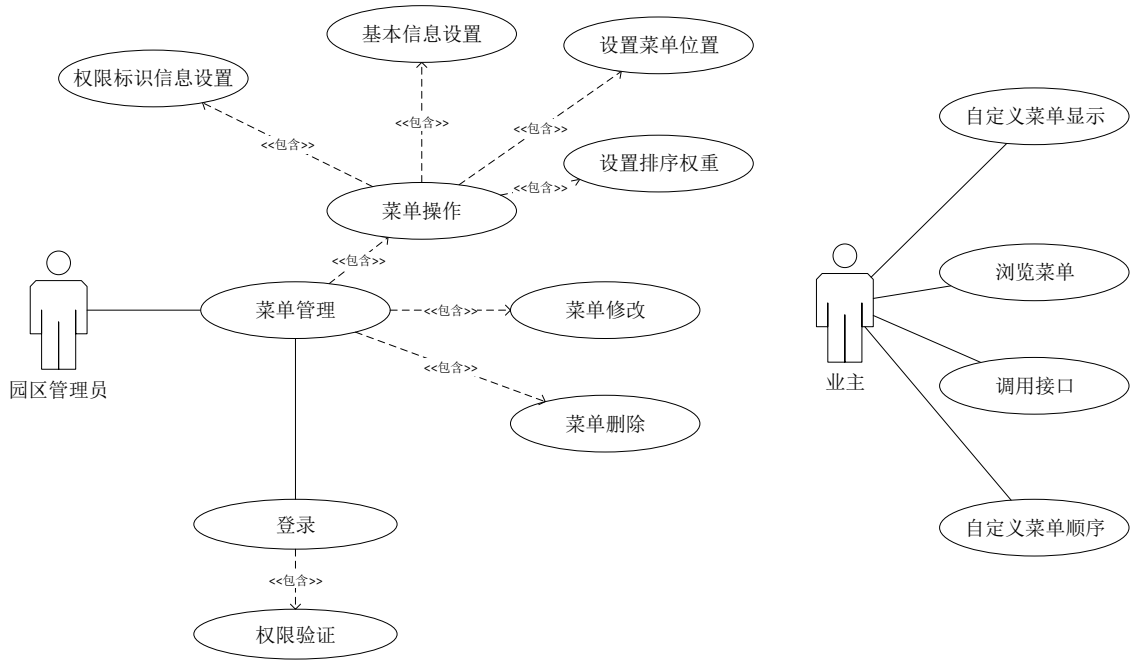

图 3-6 园区菜单管理用例图

用例说明如下：

园区方管理员对 APP 端菜单进行预先创建设置，界面菜单深度最多可设置为二级，在设置菜单时可指定上一级父菜单以及对应链接的 URL 地址。在添加菜单时必须选择上级菜单，或者直接在某个菜单下面选择添加下级菜单。添加成功后，在 APP 端可看见添加的菜单信息。管理员可以通过设置菜单位置指定该菜单是否在 APP 首页快捷操作栏置顶，业主也可以通过长按快捷操作栏菜单图标对快捷菜单进行删除操作，通过长按拖动可对同一级别菜单进行排序。

2. 推送管理

推送管理模块主要功能是园区管理方可以在该模块针对特定业主或者广播推送消息。被发送消息用户可在 APP 我的消息列表里面查看消息。针对园区广播消息所用用户都可以看到。

3. 认证和授权管理

针对用户、用户组、权限进行三权分离映射，对隶属于不同角色的用户进行权限管理。实现最终为系统平台内所有用户按业务需要划分权限，并赋予相关的功能权限，包括权限修改，撤销，删除等。

4. 业主管理

业主管理模块为业主身份信息管理模块，业主信息的录入主要由具有管理员权限的园区管理者在后台进行统一录入，并绑定业主所在楼宇信息等，同时业主也可以自己下载 APP，在 APP 上面可以通过手机号自行进行注册添加账户。用户添加成功，即可使用 APP 进行登录。

园区管理员可以在后台管理系统对业主信息按不同字段进行过滤、排序、查看，修改和删除等。业主也可以通过手机 APP 对例如头像，昵称等基本信息进行修改。业主管理用例如图 3-7 所示。

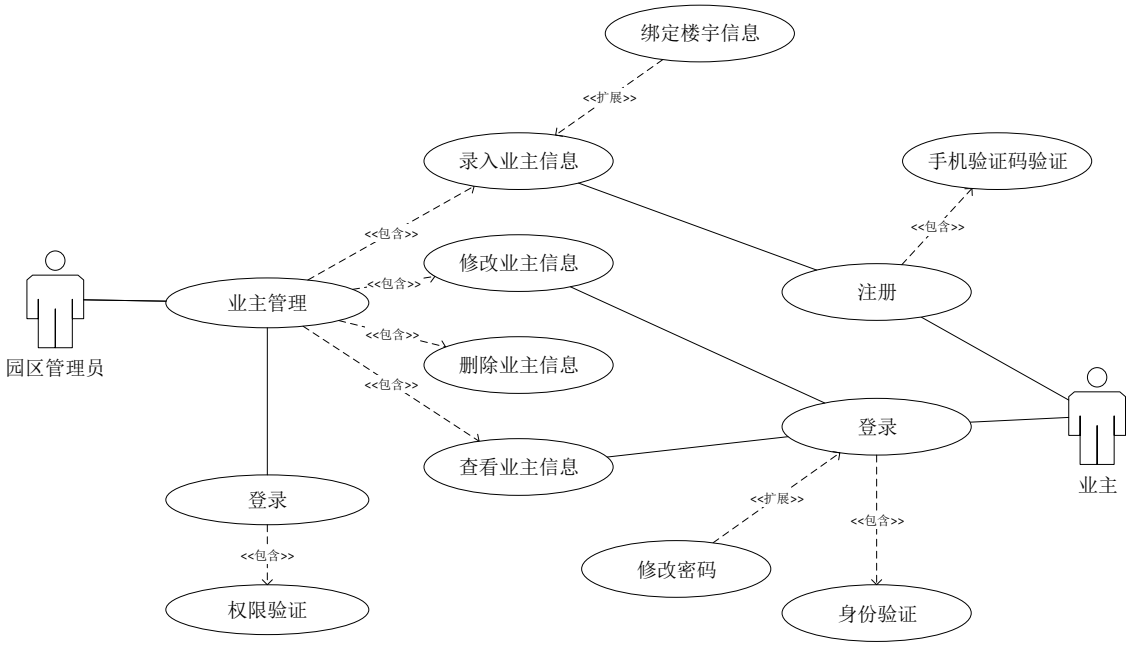

图 3-7 业主管理用例图

用例说明如下：

园区管理方拥有管理员权限的用户登录后台管理系统，通过后台系统对业主信息进行录入，业主信息包括业主姓名，业主昵称，业主密码，业主手机，业主邮箱，归属楼宇，头像，性别，认证图片等信息。业主也可以自行在 APP 端完成个人注册，注册时需要至少填写用户姓名，密码和手机验证码等信息，后续可在个人中心里完善个人其他信息。

5. 物业人员管理

物业人员管理基于用户，用户组，权限进行操作权限分配。通过该模块可以将园区管理人员分配进具有不同权限的用户组，分配进该用户组的人员可以获取该用户组具有的所有操作权限。同时也可以单独给该用户分配额外特定权限，使得不同角色的工作人员拥有操作不同模块的权限。

6. 组织机构管理

组织结构管理子模块针对园区内部人员的组织机构进行管理。

7. 缴费管理

缴费管理子模块提供管理、查看缴费欠费的记录，业主在 APP 端可选择使用支付宝或者微信进行缴费。

8. 短信验证码管理

提供用户登录流程中短信验证码管理功能。

3.3 非功能需求分析

3.3.1 界面需求

操作界面需求：

1. 合理利用界面空间，不浪费空间。
2. 布局合理、简洁、紧凑而不拥挤、没有多余线条。
3. 着色对比清晰，不繁杂。
4. 提示明确、完备、不含糊。
5. 操作方便、合理、详细介绍设置得当。
6. 风格统一。

3.3.2 性能需求

系统满足 600 个并发用户，系统响应时间不超过 6 秒，事务成功率整体不低于 93%。

3.4 本章小结

本章从浩旺智慧园区管理平台需求分析入手，分别从功能性需求和非功能性需求两方面对实现目标进行抽象与分析。基于平台实际业务需求，对业务进行模块划分，并针对各模块实际功能需求进行阐述，同时梳理了非功能需求中需要达到的一些关键指标，为下一步系统的概要设计、详细设计与功能的实现奠定了基础。

第四章 系统设计

4.1 设计目标与原则

4.1.1 设计目标

设计出一个满足园区方需求的智慧园区平台系统，以达到助力产业园区信息化管理和支撑起园区孵化服务的目的。

1. 系统功能模块设计覆盖到园区日常管理工作中可以进行信息化管理的事项。

2. 需要对园区提供的孵化服务项目提供信息化支撑，有效地支持孵化服务工作中定义的关键业务及流程。

3. 开发符合 RESTful 规范的系统接口，对待提交数据的错误提示做到明确、完备、不含糊，同时提供基于浏览器访问的在线接口文档，该文档可以随着后端代码的修改而自动更新，文档界面友好、灵活、接口字段描述清晰。前端开发人员可以直接在该在线文档上进行接口测试而不必依赖与如 postman、httprequester 等 HTTP 接口测试工具，减少了前后端沟通成本。

4. APP 端 UI 界面风格统一，布局合理、简洁、紧凑而不拥挤、没有多余线条。着色对比清晰，不繁杂。

5. 后台管理系统功能模块划分清晰、操作方便、友好、易用。

6. 系统整体架构及部署规划能提供很好的负载均衡，可以应对高并发访问。

4.1.2 设计原则

1. 易用性

根据确认后的最终需求对系统功能进行设计，抽象出数据库模型，将系统功能进行模块化划分及细化。结合园区方信息化管理建设现状对平日里管理工作事项以及需提供的孵化服务项目进行抽象，设计出满足园区方园区管理和孵化服务业务需求的系统。APP 端功能菜单划分清晰，布局合理，菜单及功能操作符合用户使用习惯。后台管理系统操作简便易用，提供基于字段过滤、排序及模糊查询，数据导出等操作。

2. 稳定性

在系统架构及部署方面，采用多 Nginx 实例进行静态资源代理和反向代理，服务器端同时启动多个 uWSGI 应用服务实例，通过负载均衡的方式保证系统在高

并发访问情况下依旧能够向外提供高效稳定的服务，保证各业务流程与功能的正常使用，通过 Redis 在服务器端缓存一些常用的非实时数据，能很好的减少 I/O 操作频率，提高了系统的响应速度与执行效率。

3. 灵活性

RESTful 服务层需保证接口调用灵活，通过在服务层配置不同类型过滤器，可以很轻易的对各层次数据进行排序、筛选、过滤等功能，同时提供的在线 API 接口文档具备自动生成更新功能，前端开发人员可直接在接口文档界面使用不同开发语言对接口进行测试。

4. 可扩展性

智慧园区平台系统规划分为多期进行开发实现，随着项目进度的推进，会有越来越多新的业务流程以及需求的提出，这就对系统的可扩展性提出了要求。

采用基于 APP 的松耦合方式进行系统功能拆解，每一个 APP 都是一个相对独立的功能模块。遵循 Django APP 设计规范的模块具有高内聚低耦合的特征，模块之间互相不具有侵入性，这就为频繁的功能变更提供了有力的基础。在系统平台上无论是进行新功能的开发，已有功能的修改与删除，还是集成第三方功能模块等都能做到互不影响，保证了系统的具有良好的可伸缩性与可扩展性。

5. 可维护性

通过 Python 自身的内建 logging 模块与 Django 框架进行结合，为系统提供了强大的日志监控系统，通过基于 dictConfig 的规则配置，可实现具有丰富组合功能的日志监控服务，同时系统通过集成 Sentry 这个基于 Django REST framework 实现的第三方线上日志监控平台，更大力度的保证了系统的可维护性，在系统出现异常告警的情况下，将第一时间通知系统管理员及运维人员等，可最快速度掌握系统故障信息，做出及时响应。

4.2 系统整体架构设计

系统平台整体架构可分为三个部分，分别是客户端、RESTful 服务层以及后台管理系统。系统整体架构如图 4-1 所示。

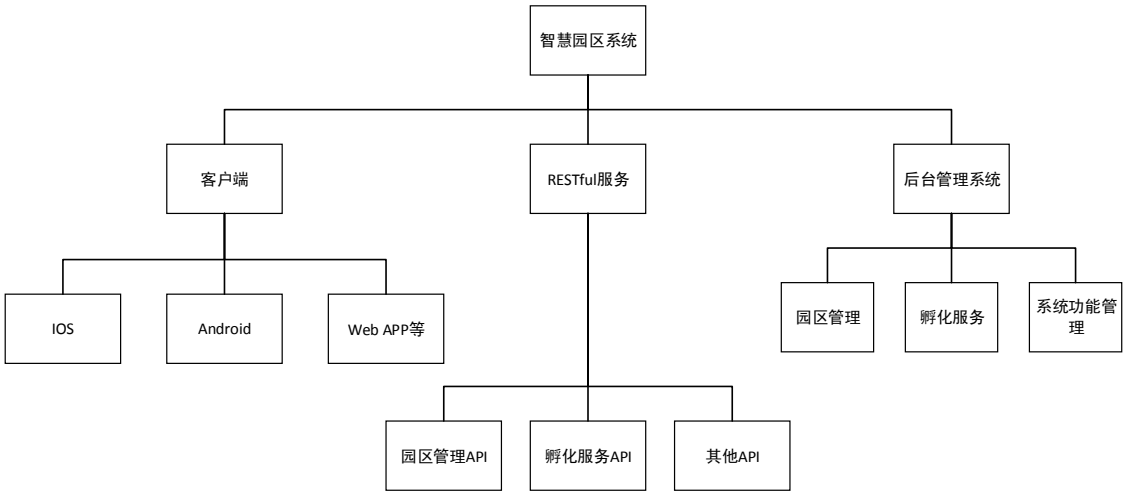

图 4-1 系统整体架构图

客户端现阶段只开发 Android 版本，后续将根据用户使用情况决定何时开发 iOS 和 Web APP 等形式的客户端，RESTful 服务层采用前后端完全分离模式进行开发，后台管理系统采用 B/S 模式进行开发。

系统平台架构组成第一部分为客户端，由于系统采用前后端完全分离模式进行开发，所以具有良好的多端适配性，客户端既可以采用基于 iOS 和 Android 平台的原生 APP 开发，也可以采用基于 HTML5 的 Web APP 或者同时基于原生和 HTML5 的混合式方式进行开发，项目一期只开发基于 Android 平台的原生 APP 客户端，客户端采用基于 HTTP 协议通过 OKHttp3 这个轻量级框架处理网络请求与响应，通过与 RESTful 服务层进行数据交互，以提交请求及获得视图层数据。

第二部分为 RESTful API 服务层。基于 Django RESTful framework 框架开发的服务层自身采用多层架构模式，在 Django 原生的 MTV 的基本三层架构基础上进行改造，由于服务层只提供基于 RESTful 风格的数据服务，所以传统 Django 框架中的 Template 也就是视图层被移除了，转变成了对应于 Django RESTful framework 中的 Serializers 序列化器层，Serializers 类及其子类类似 Django 框架中的 Form 类及 ModelForm 类，可以提供双向数据服务，接收请求时可以对请求数据进行完整性、有效性等验证，提供响应时允许将诸如查询集和模型实例之类的复杂数据转换为原生 Python 数据类型，然后可以将它们轻松地呈现为 JSON，XML 或其他内容类型，同时可非常灵活的自定义响应数据结构及数据字段。基于 Serializers 的 ModelSerializer 子类更是提供了一个快捷操作方式，可让你自动创建一个 Serializer 类，其中的字段与模型类字段一一对应，通过在 View 层配置对应的 ModelSerializer 可以轻松将验证后的数据持久化到数据库之中。Model 层和 Django 原生 Model 层一致，Django 的 Model 层采用 Code First 原则，对数据库里的各个实体封装成对

应的 Python 类对象，对数据库的所有操作都被封装成了对 Python 对象的操作，同时提供非常强大易用的数据库操作 API 接口。View 视图层是核心逻辑控制器，客户端请求通过 URLconf 进行路由分发到对应的视图层，视图层针对不同的请求动词对请求进行分类过滤，针对不同资源的不同动作都会被映射到对应的处理函数之中，最后通过序列化器组装响应数据，向客户端返回 JSON 格式的结果，客户端最终拿到数据后便可进行前端渲染。可配置的中间件队列则可针对 HTTP 请求和响应作全局拦截和验证。系统 RESTful API 服务层技术架构如图 4-2 所示。

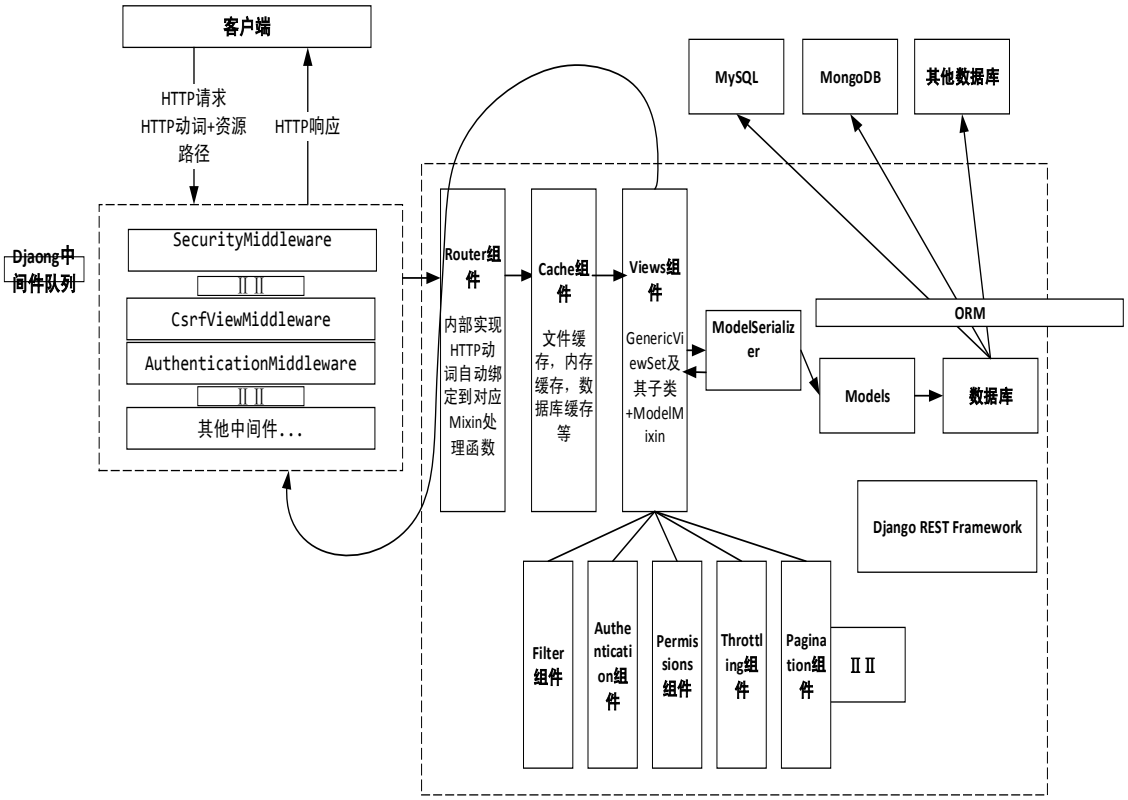

图 4-2 智慧园区系统 RESTful API 服务层技术架构

第三部分为后台管理系统。后台管理系统负责智慧园区平台系统内所有数据的管理与操作。整体采用 Django 内置的基于用户、用户组、权限的三权分离模式进行权限控制。只有具有园区管理员或者对应功能模块权限的人员才能登录进后台系统并可以对其拥有权限的模块进行操作。园区管理、孵化服务与系统管理数据均可通过后台管理系统进行录入、修改、查询及删除，APP 端用户也可以提交与更新其中部分数据。

4.3 系统整体功能设计

在确定系统架构设计之后，需要对系统所提供的业务功能进行详细的阐述，

继而为后面的功能模块划分提供相应的依据。

根据业务分析结果，本平台系统涉及的主要模块可以划分为：园区管理、孵化服务、系统功能管理，除此之外还包含一些辅助性的功能模块。园区管理板块主要将园区管理方日常工作中的一些事项进行可视化和信息化，例如园区管理方可以第一时间发布一些重要公告或通知，园区业主可以报名参加企业活动，业主可以申请服务需求，例如维修上报等，使得园区管理工作更为高效。在孵化服务方面通过与第三方服务机构进行合作，拓展了对园区内企业的服务范围和服务内容，例如在保洁服务、金融服务、财务代理中都是通过与第三方机构合作达成最终服务目标，该种模式使得园区可以在原有单一租金收益基础上增加创收，同时入驻企业也可以享受更加丰富的服务。

智慧园区平台系统总体功能设计如图 4-3 所示：

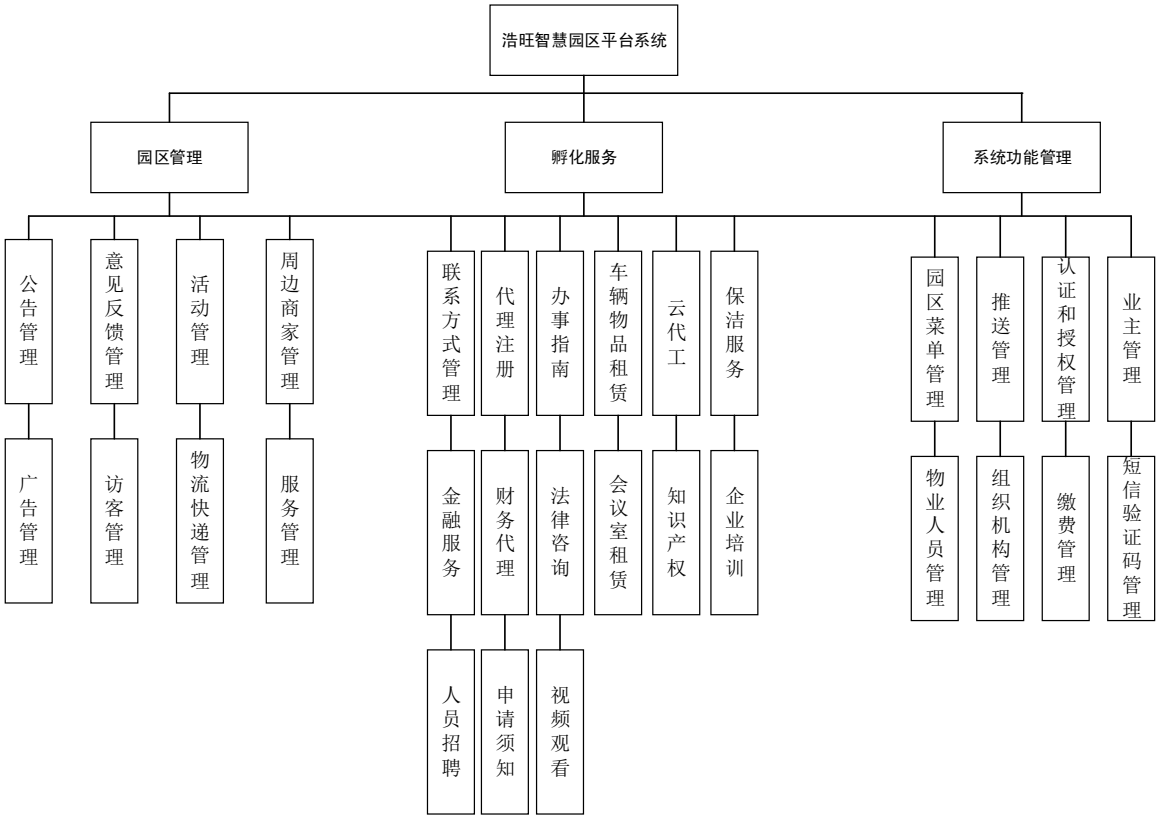

图 4-3 智慧园区系统功能设计

4.4 系统架构分层设计

Django 框架整体请求、响应流程如图 4-4 所示：

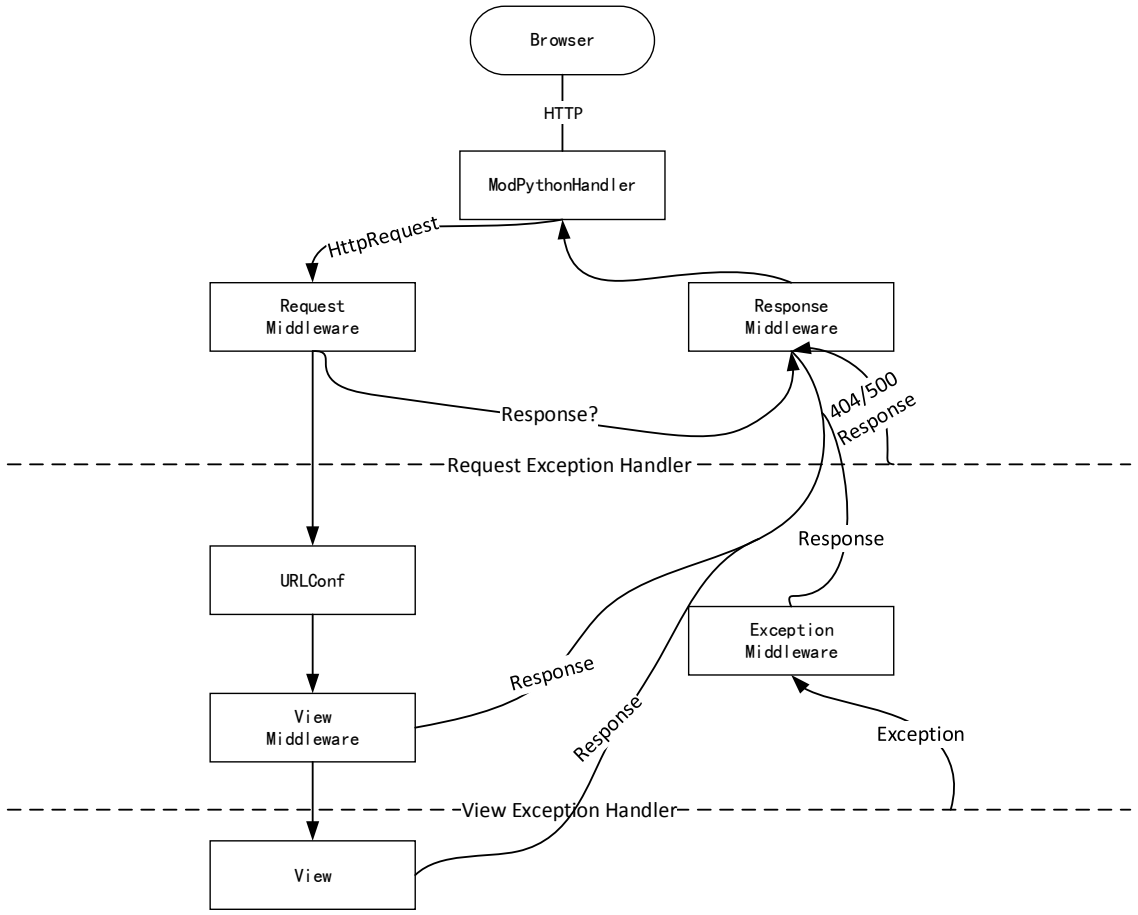

图 4-4 Django 请求响应流程图

4.4.1 视图层设计

这里所说的视图层类似于 MVC 三层架构中的控制器层。

Django REST framework 中提供了 APIView、GenericAPIView、ViewSet、GenericViewSet 等视图层基类及子类，另外还提供了 5 个扩展 Mixin：CreateModelMixin、ListModelMixin、RetrieveModelMixin、UpdateModelMixin 和 DestroyModelMixin。这 5 个扩展类实际上提供了 REST 中对资源的创建、查询所有数据、查询单条数据及更新、部分更新和删除的接口并给与了默认实现。通过在自定义视图类中实现对视图层基类和 Mixin 的多重继承的各种组合可以完成基于不同业务需求的实现。

考虑到系统在以后功能扩展和变更中可能面对的需求，这里选用最为灵活及功能性最完备的一种组合方式，基于 GenericViewSet 配合 Mixin 的方式来实现视图层。视图层采用 Django 官方推荐的 Class Based View 的方式而不采用传统的 Function Based View 的方式来组织代码结构，以达到最好的复用性。HTTP 动词在 REST 中语义如表 4-1 所示。

表 4-1 HTTP 动词在 REST 中语义

方法	语义
OPTIONS	用于获取资源支持的所有 HTTP 方法
HEAD	用于只获取请求某个资源返回的头信息
GET	用于从服务器获取某个资源的信息： 1. 完成请求后，返回状态码 200 OK 2. 完成请求后，需要返回被请求的资源详细信息
POST	用于创建新资源： 1. 创建完成后，返回状态码 201 Created 2. 完成请求后，需要返回被创建的资源详细信息
PUT	用于完整的替换资源或者创建指定身份的资源，比如创建 id 为 10086 的某个资源： 1. 如果是创建了资源，则返回 201 Created 2. 如果是替换了资源，则返回 200 OK
PATCH	用于局部更新资源： 1. 完成请求后，返回状态码 200 OK 2. 完成请求后，需要返回被修改的资源详细信息
DELETE	用于删除某个资源，完成请求后返回状态码 204 No Content

4.4.2 路由设计

系统中针对 RESTful API 的接口应该与后台系统和其他的功能接口进行分离，本系统在一级路由配置文件中进行路由汇总，将所有和 RESTful 接口相关的子接口配置到/api 路由下，后台管理系统接口配置到 xadmin/下。在使用 Djong REST framework 配置路由时一般有两种选择。

方式一可以通过在全局配置中，自定义绑定 HTTP 请求动词和视图函数的方式来实现。这种方式的优点是用户对路由的定制型更强。

方式二可以选用 ViewSet 搭配 Router 组件进行路由绑定。这种方式的优点是代码更加简洁。

这里系统选用了第二种方式进行路由绑定，因为考虑到后期功能的扩展，对园区管理和孵化服务中的各个资源均可能进行 GET、POST、PUT、PATCH、DELETE 等方式的操作，所以直接选用对所有动词均进行了绑定的方式二来实现。通过在一级路由配置文件中初始化 DefaultRouter 类实例，再通过调用实例的 register 方法来进行路由绑定。系统路由设计如图 4-5 所示。

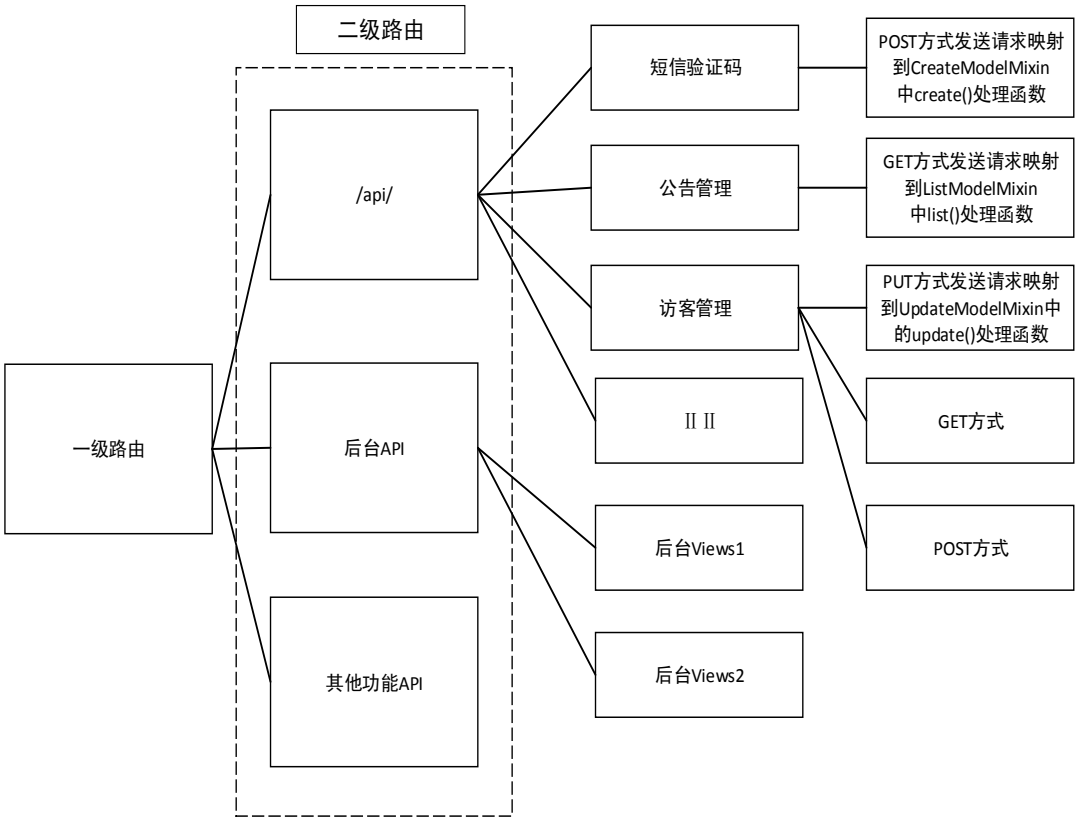

图 4-5 智慧园区系统路由设计

4.4.3 认证和权限控制设计

系统采用混合式认证模式，在 RESTful 服务层采用 JSON Web Token 的方式对用户身份进行认证，在后台管理系统端则采用 Cookie-Session 模式对登录用户进行身份认证。

权限控制方面通过集成第三方框架 django-guardian 提供基于对象级别的细粒度权限控制，Django 默认的权限控制只能细分到 Model 级别，对应到数据库也就是只能精确到数据表级别，而该粒度级别只能满足一些简单的场景。

在本系统中需要针对不同角色权限做出限制，某一角色有可能只拥有针对表中部分字段的读写权限，而 django-guardian 模块提供了精确到对象的 field 级别也就是数据库的字段级别的权限控制。

4.4.4 序列化器设计

对于类似公告管理、联系方式管理、办事指南等更多的是信息发布后前端进行数据查询展示的功能，直接在自定义序列化类中继承 ModelSerializer。然后通过声明式绑定方式在元数据子类中绑定数据模型和字段，然后可以正向进行数据的持久化，反向进行数据的序列化。

在缴费管理模块中，在进行费用缴纳时，需要对前端提交字段进行有效性验证，同时前端提交字段不能直接全部持久化到模型层中，比如用户字段是不需要客户端来提交的，因为用户已经登录，状态在服务器端已经存储了，这种情况下采取直接继承 ModelSerializer 的方式不太合适。直接选用继承底层 Serializer 重写 create()方法的方式可以做到自定制。所以在序列化器设计上需要根据不能功能点业务流程进行合理的技术选型。

4.4.5 分页器设计

通过 GET 方法查询到的一个接口的数据有可能有很多，客户端可能需要对数据进行分页懒加载，所以服务器端需要进行数据分页。

因为 REST 服务器端是无状态的，在进行分页的时候，服务器端并不知道你当前的访问位置及前一页和后一页的地址，这个状态需要客户端来维护，但是一旦知道了当前页，下一页及上一页资源是需要服务器端来维护的。

通过自定义分页器类继承 restframework 中提供的 PageNumberPagination 基类，对分页中每页记录条数，最大分页数，分页查询字符串和记录查询字符串等进行配置。然后在视图类中对分页器进行显示声明注入。

这样可得到形如

```
{
    "count": 52,
    "next": "http://localhost:8866/api/notices/?page=4",
    "previous": "http://localhost:8866/api/notices/?page=2",
    "results": [
        {}
    ]
}
```

风格良好的数据，返回数据中标明了记录总数及上一页下一页的接口 URL 地址，维护了当前资源状态，便于客户端使用。

4.4.6 过滤器及搜索器设计

系统中过滤功能主要依赖于 django-filter 这个第三方库，避免了我们在视图层中自己写 QuerySet 查询语句造成的代码量过大，只需进行显示声明式配置便可实现简单的过滤需求，遇到复杂需求时也可进行自定制规则。以下是系统中采用的四种主要过滤器。过滤器设计功能说明见表 4-2。

表 4-2 过滤器设计功能说明

过滤器名	功能
DjangoFilterBackend	提供基于字段的精确过滤，可自定制过滤规则
SearchFilter	提供数据表中模糊匹配的功能
OrderingFilter	提供字段级别排序的功能
DjangoObjectPermissionsFilter	仅返回用户具有适当查看权限的查询集对象

4.5 系统模块设计

本部分将分别对系统功能管理模块、园区管理模块和孵化服务模块中的主要功能部分及涉及到的主要业务流程进行描述。

4.5.1 功能设计

4.5.1.1 系统功能管理模块功能设计

1. 园区菜单管理子模块

针对园区管理员权限或拥有模块操作权限用户可对 APP 内所有一级、二级菜单添加、编辑、删除、排序功能。

针对普通用户权限可对 APP 首页自定义快捷菜单栏进行编辑、排序功能。

2. 推送管理子模块

针对园区管理员权限或拥有模块操作权限用户可对特定用户推送消息，也可对全体用户广播消息。

针对普通用户权限可查看系统推送过来的消息。

3. 认证和授权子模块

针对园区管理员权限或拥有模块操作权限用户可添加、编辑、删除用户、用户组以及权限，可对用户组进行权限赋予及权限删除，可将用户赋予给特定用户组或从中删除。

4. 业主管理、物业人员管理、组织机构管理子模块

针对园区管理员权限或拥有模块操作权限可对业主、物业人员、组织机构信息进行添加、编辑、删除。

针对业主角色可在 APP 端对自身信息进行完善补充，并绑定相应楼宇号。

5. 缴费管理子模块

针对园区管理员权限或拥有模块操作权限可对不同业主进行费用记录录入、修改、删除操作。

针对业主角色可在 APP 端进行欠费记录查看，在线缴费。

6. 短信验证码子模块

提供在业务流程中需要用到短信验证码功能的接口。

4.5.1.2 园区管理模块功能设计

1. 公告管理子模块

针对园区管理员权限或拥有模块操作权限用户可分别选择通知、公告、制度和标准四种不同类型进行消息的发布、编辑、删除。

针对普通用户权限可在 APP 端相应模块中浏览园区发布的最新公告信息。

2. 意见反馈管理子模块

针对园区管理员权限或拥有模块操作权限用户可查看用户意见反馈，对意见反馈进行编辑、删除，对于不同用户的意见反馈可指派相关处理人员后续跟进意见反馈并添加处理结果。

针对普通用户权限可在 APP 端进行意见反馈的提交并查看意见反馈处理进度。

3. 活动管理子模块

针对园区管理员权限或拥有模块操作权限用户可新增活动信息、编辑、删除活动信息。同时可对活动报名情况进行审批，统计活动报名人数等。

针对普通用户权限可在 APP 端查看浏览园区方发布活动信息，对感兴趣活动提交报名申请，并可查看活动报名处理进度。

4. 周边商家管理子模块

针对园区管理员权限或拥有模块操作权限用户可对园区内入驻商家信息进行录入、编辑、删除。

针对普通用户权限可在 APP 端查看浏览周边商家信息。

5. 广告管理子模块

针对园区管理员权限或拥有模块操作权限用户可对 APP 内广告位广告分组管理、编辑、排序、删除等，并可指定广告投放有效期。

6. 访客管理子模块

该模块主要供拥有保安角色用户使用，可对来访人员信息进行录入，信息包括来访人员及随同信息，到访时间，到访缘由，离开时间等，便于园区安全管理。

7. 物流快递管理子模块

针对园区管理员权限或拥有模块操作权限用户进行周边物流公司、快递代收点信息的录入、编辑、删除操作。信息包括物流公司地址，联系方式，距离等。

针对普通用户权限可在 APP 端查看浏览物流快递公司信息，如有寄件需求等，可直接进行电话联系。

8. 服务管理子模块

针对园区管理员权限或拥有模块操作权限用户可查看业主发起的不同类型的服务诉求，指派相关人员进行处理，更新服务诉求处理进度等。

针对普通用户权限可在 APP 端发起各种类别的服务诉求，并可在 APP 端查看诉求处理进度信息及处理人信息等。

4.5.1.3 孵化服务模块功能设计

1. 联系方式管理子模块

针对园区管理员权限或拥有模块操作权限用户可对园区各部门负责人信息进行录入、删除、修改。

针对普通用户权限可通过 APP 端查看各部门负责人信息，如果联系电话等，可以通过 APP 端拨打电话联系相应负责人。

2. 代理注册子模块

针对园区管理员权限或拥有模块操作权限用户可对企业在 APP 提交的代理注册申请信息进行初步审核，审核通过后可交由专业注册资质审核机构进行后续审核，每一个过程中都可以更新代理注册申请状态。

针对普通用户权限可通过 APP 端进行代理注册申请，注册信息填报，注册申请进度查询等操作。

3. 云代工子模块

想要出售过剩产能的企业可向园区方提交代工信息，针对园区管理员权限或拥有模块操作权限用户可录入代工信息，并对代工信息进行编辑、删除等。

针对普通用户权限可通过 APP 端进行企业代工信息查询，如果有需要这些代工可以通过电话联系。

4. 保洁服务子模块

针对园区管理员权限或拥有模块操作权限用户发布保洁信息，对用户在 APP 端发起的保洁预约申请进行二次确认，联系响应保洁公司进行服务预约。

针对普通用户权限可通过 APP 端进行保洁信息查询，可直接在 APP 端通过电话联系保洁公司进行预约或填报提交预约申请，由园区管理者帮助预约。

其他子模块设计方式相似，在此不再赘述。

4.5.2 流程设计

4.5.2.1 系统功能管理模块流程设计

1. 用户注册流程

用户注册流程如图 4-6 所示。

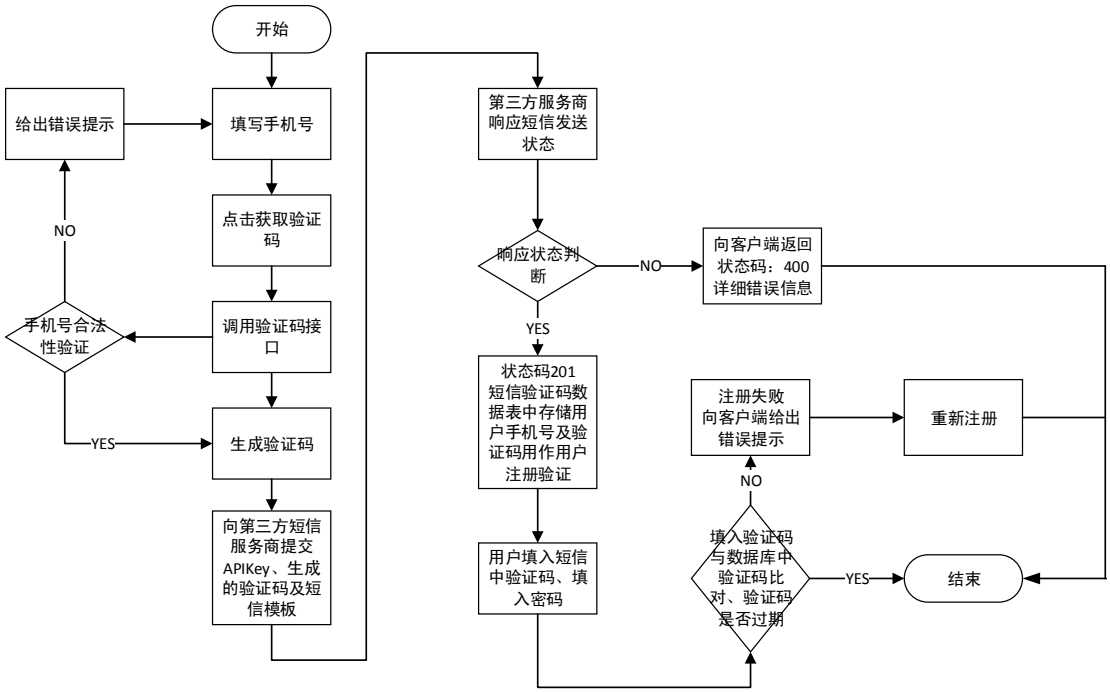

图 4-6 用户注册流程图

流程描述：

用户在 APP 端填写注册手机号，再点击获取验证码按钮，此时 APP 端会调用 RESTful 接口端验证码接口，服务层在处理程序中随机生成 4 位数字验证码，然后组装短信模板，将预先在第三方短信服务商处申请的 APIKEY、验证码、短信模板、用户手机号都提交到第三方短信服务商提供的短信发送接口，此时服务商将向指

定手机号发送短信验证码，在发送短信前，服务层处理程序将对用户填写的手机号合法性进行验证，如果手机号格式无效将向客户端注册页面发送错误提示，提示用户重新注册，如果验证通过，会检查第三方短信服务商响应数据状态，如果发送失败，向客户端返回状态码 400 及详细错误信息，如果成功，返回状态码 201 及在事先设计好的短信验证码表里存储生成的短息验证码及手机号，用于注册提交时进行验证码的验证，用户继续填入验证码及密码，点击注册，将数据提交到服务层注册接口，如果填写验证码与数据库中一致且验证码未过期，则用户注册成功，用户注册信息将存入用户表里，默认角色为业主。

2. 用户认证流程

JSON Web Token 工作流程如图 4-7 所示，用户登录认证流程如图 4-8 所示。

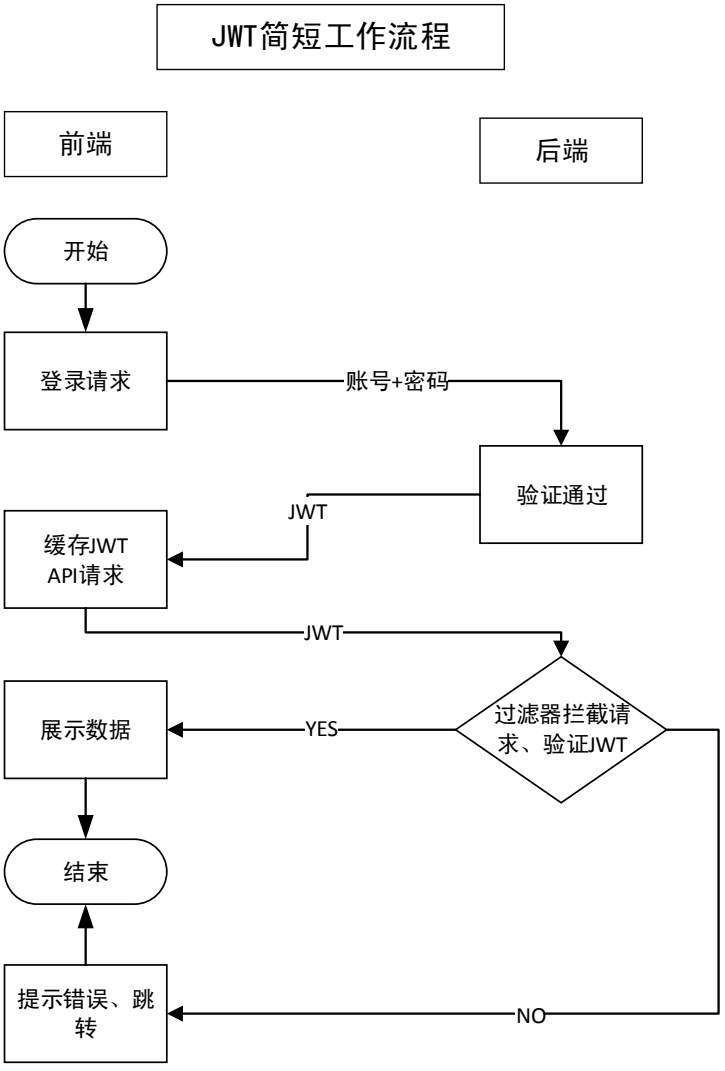

图 4-7 JSON Web Token 工作流程图

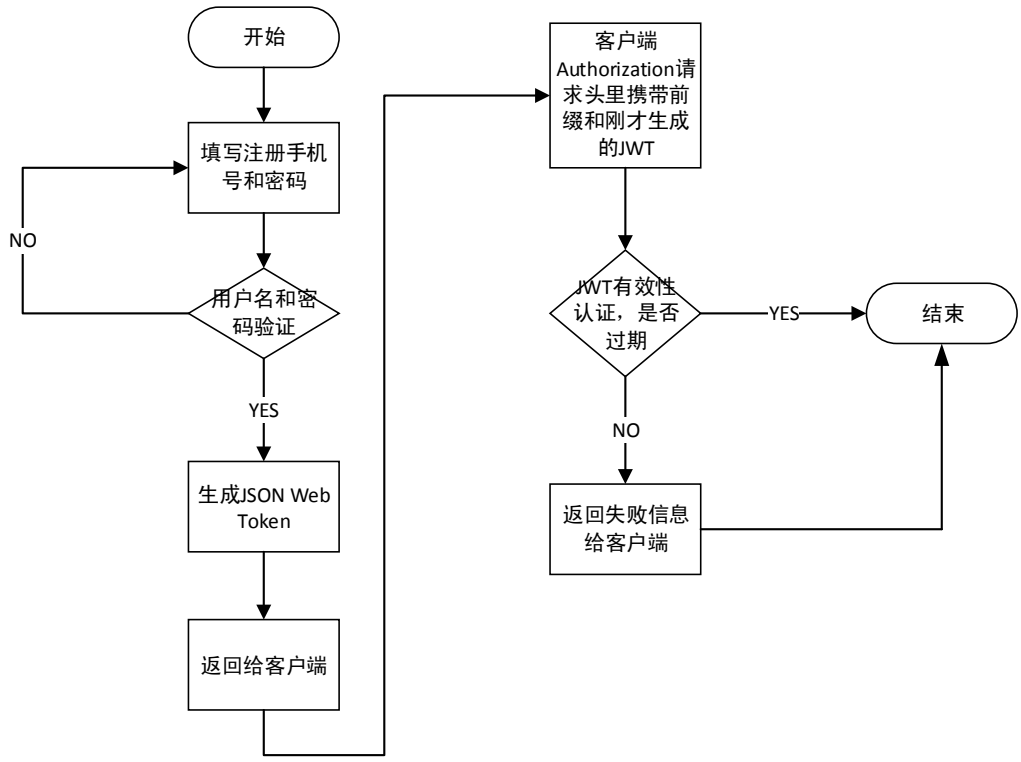

图 4-8 用户登录认证流程图

服务层采用 JSON Web Token 认证模式，采用非对称加密算法将用户的非敏感信息通过加密后存储在客户端，减轻了服务器端和数据库的压力。

JSON Web Token 由头部、负载和签名构成。头部包含了采用的加密算法和 Token 类型信息，使用 Base64 格式进行编码。负载部分包含了签发者、过期时间、面向用户对象、接收方和签发时间等，用户自身的非敏感信息一般也存放在这部分。同样也使用 Base64 格式进行编码。签名部分使用头部里指定算法将编码后头部、负载和密钥进行签名。JSON Web Token 构成部分如图 4-9 所示。

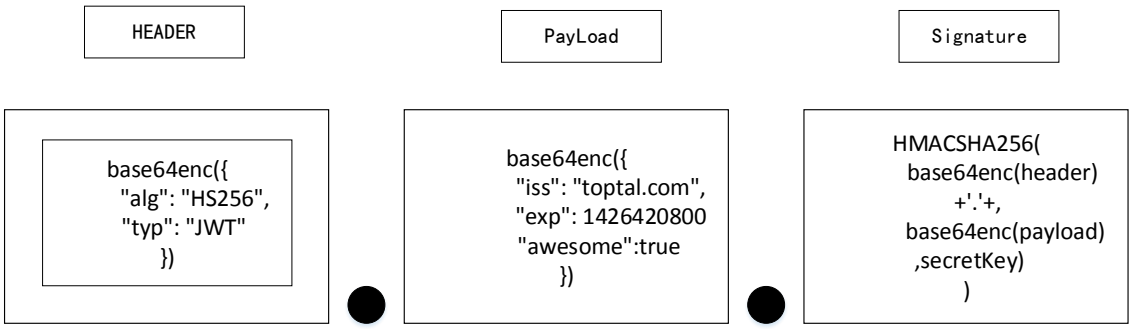

图 4-9 JSON Web Token 构成部分示意图

3. 消息推送流程

流程描述：

拥有模块操作权限的后台管理员可在后台进行消息的推送。如果 user_id 置为 0 则是针对所有用户的群发推送，否则 user_id 应该填写具体用户的 id 进行特性用户消息推送。消息推送流程如图 4-10 所示。

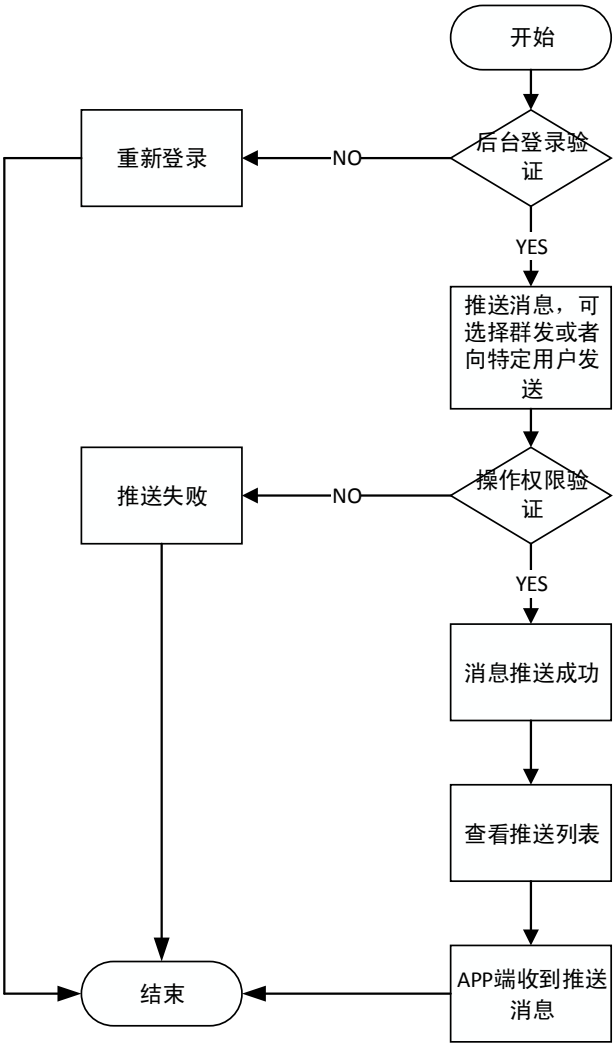

图 4-10 消息推送流程图

4. 在线缴费流程

流程描述：

业主通过在 APP 端可查看到当月和历史欠费账单。用户可在 APP 端发起在线缴费流程。在线缴费分为同步流程和异步流程两类，同步流程是指用户从 APP 端发起在线缴费流程开始，一次性正常完成整个缴费流程，中途没有遇到如断网、误操作退出缴费等其他因素干扰，整个缴费流程是同步进行的。而异步缴费流程是指用户在缴费过程中，中途退出了缴费操作，但是缴费流水号和订单已经生成，此时用户想重新返回支付。系统针对同步和异步流程均有相应的处理。在线缴费流程时序图如图 4-11 所示。

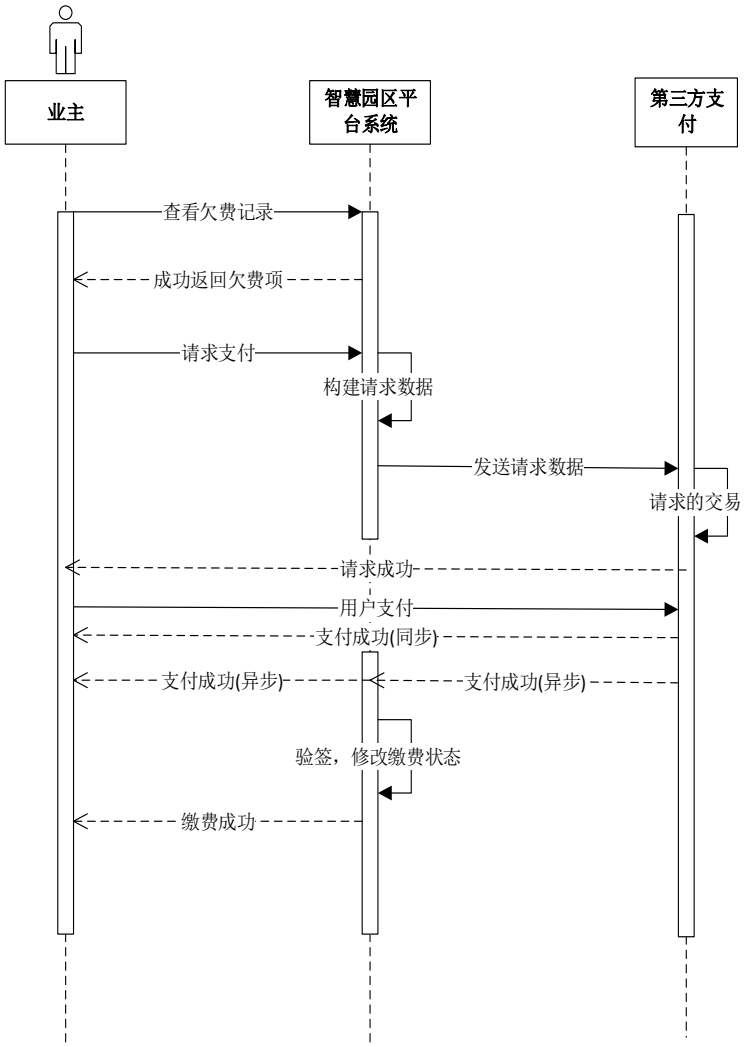

图 4-11 在线缴费流程时序图

4.5.2.2 园区管理模块流程设计

1. 公告发布流程

流程描述：

拥有模块操作权限的后台管理员可在后台分别选择通知、公告、制度和标准四种不同类型进行消息的发布、编辑、删除。

在发布公告之前，对应视图会首先判断该操作用户是否具有相应操作权限，如果没有响应模块的操作权限或者不是超级管理员则发布、编辑、删除操作均会失败。公告发布流程如图 4-12 所示。

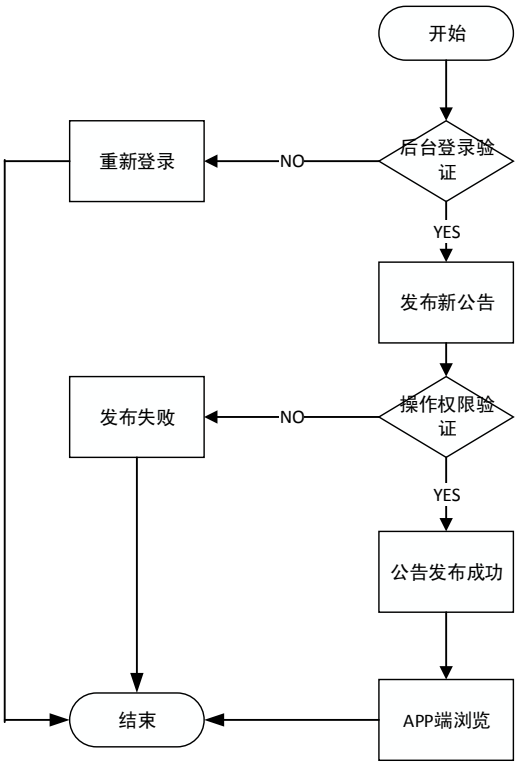

图 4-12 公告发布流程图

2. 意见反馈提交流程

意见反馈提交流程如图 4-13 所示。

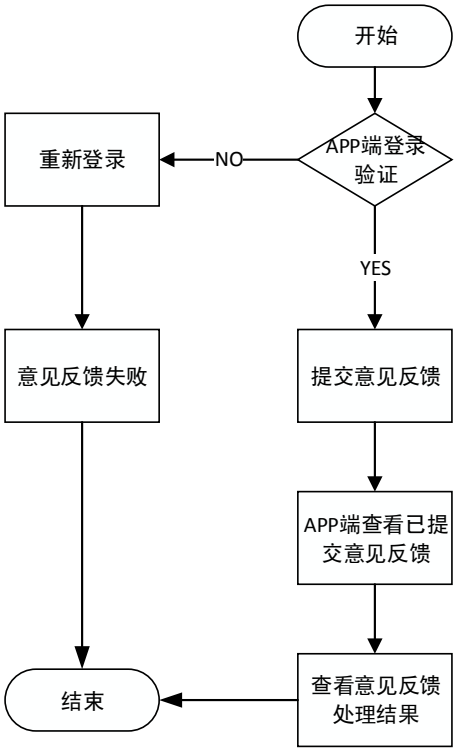

图 4-13 意见反馈提交流程图

3. 意见反馈处理流程

意见反馈处理流程如图 4-14 所示。

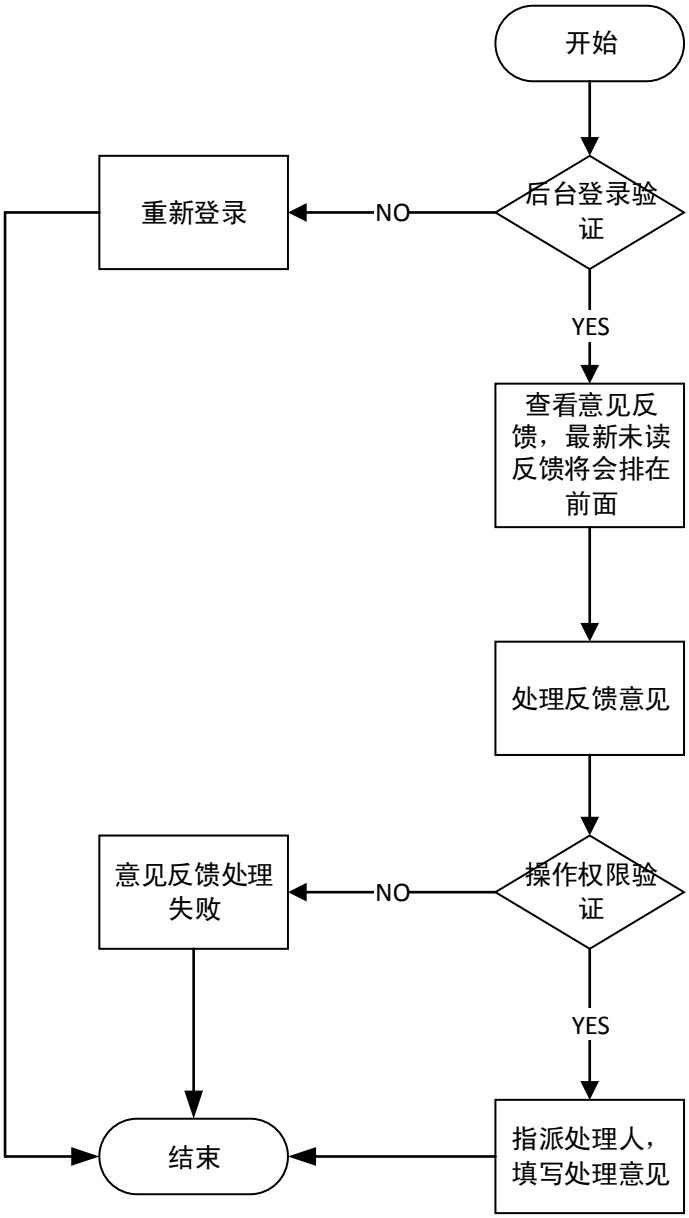

图 4-14 意见反馈处理流程图

4. 活动报名流程

流程描述：

用户可在 APP 端浏览不同的活动信息，可点选具体活动进行报名申请。园区管理员可在后台浏览活动的报名申请信息，对报名申请进行审核，审核通过的报名申请状态将被改成已成功报名，用户可在 APP 端查看到报名申请状态。未审核通过的报名申请可备注审核未通过原因，同时，报名审核通过后，报名信息表中

对应记录中报名人数将随之增加。活动报名流程如图 4-15 所示。

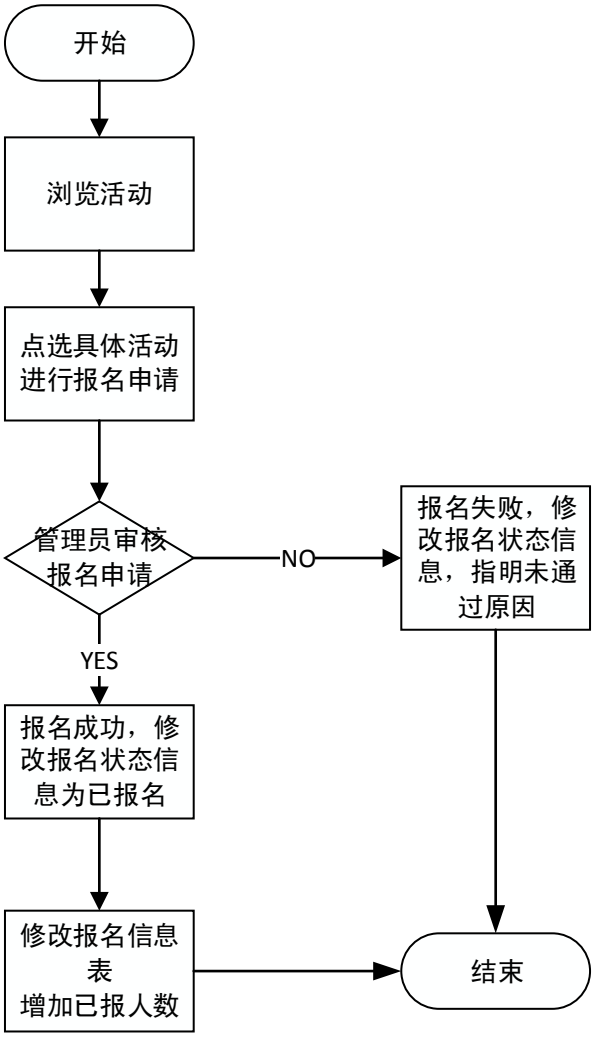

图 4-15 活动报名流程图

5. 访客登记流程

流程描述：

访客登记功能一般由楼宇保安使用，首先需要具备相应权限的用户登录 APP 才能看到此级界面，具有相应权限的用户可以对来访人员进行访客登记。同时可以随时更新来访记录。访客登记流程如图 4-16 所示。

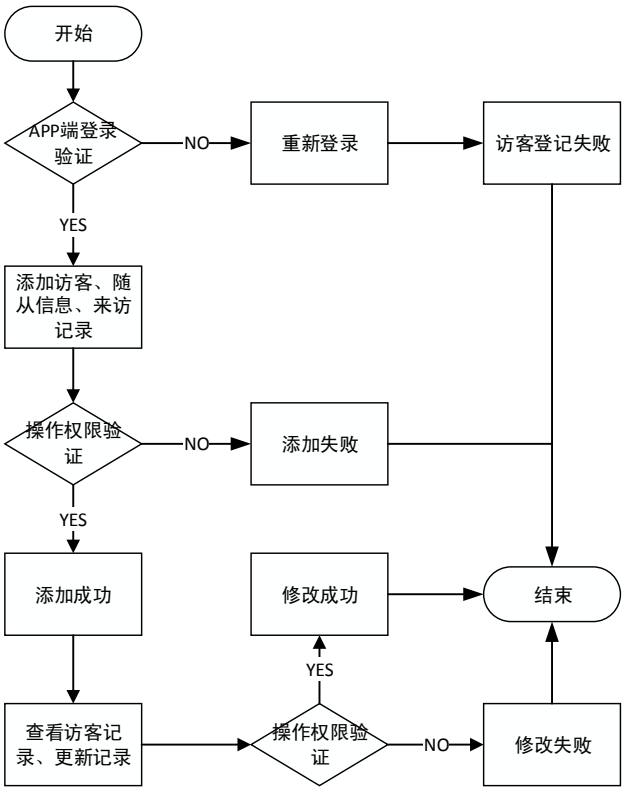

图 4-16 访客登记流程图

6. 服务管理流程

服务管理流程如图 4-17 所示。

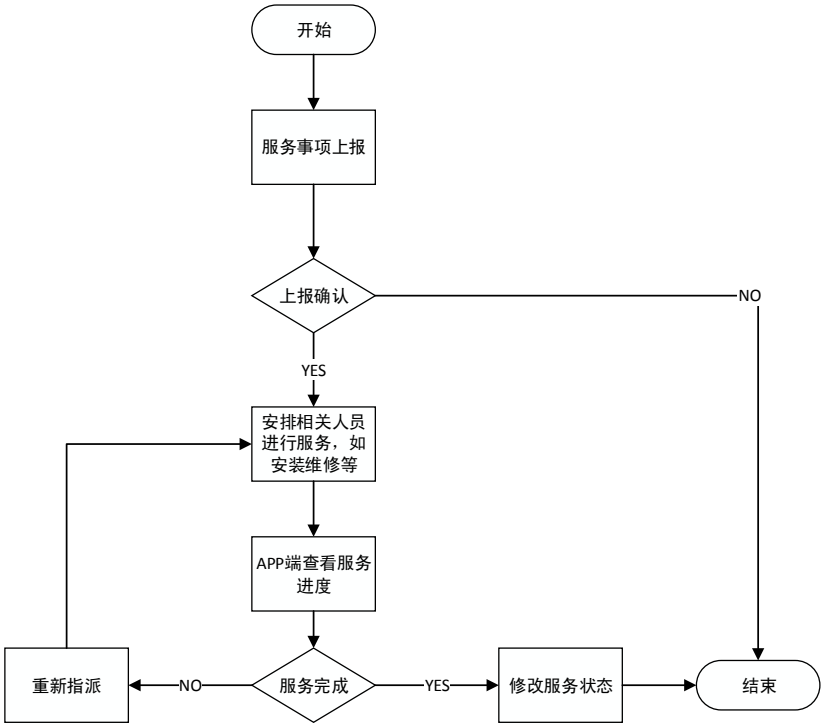

图 4-17 服务管理流程图

4.5.2.3 孵化服务模块流程设计

1. 代理注册流程

代理注册流程如图 4-18 所示。

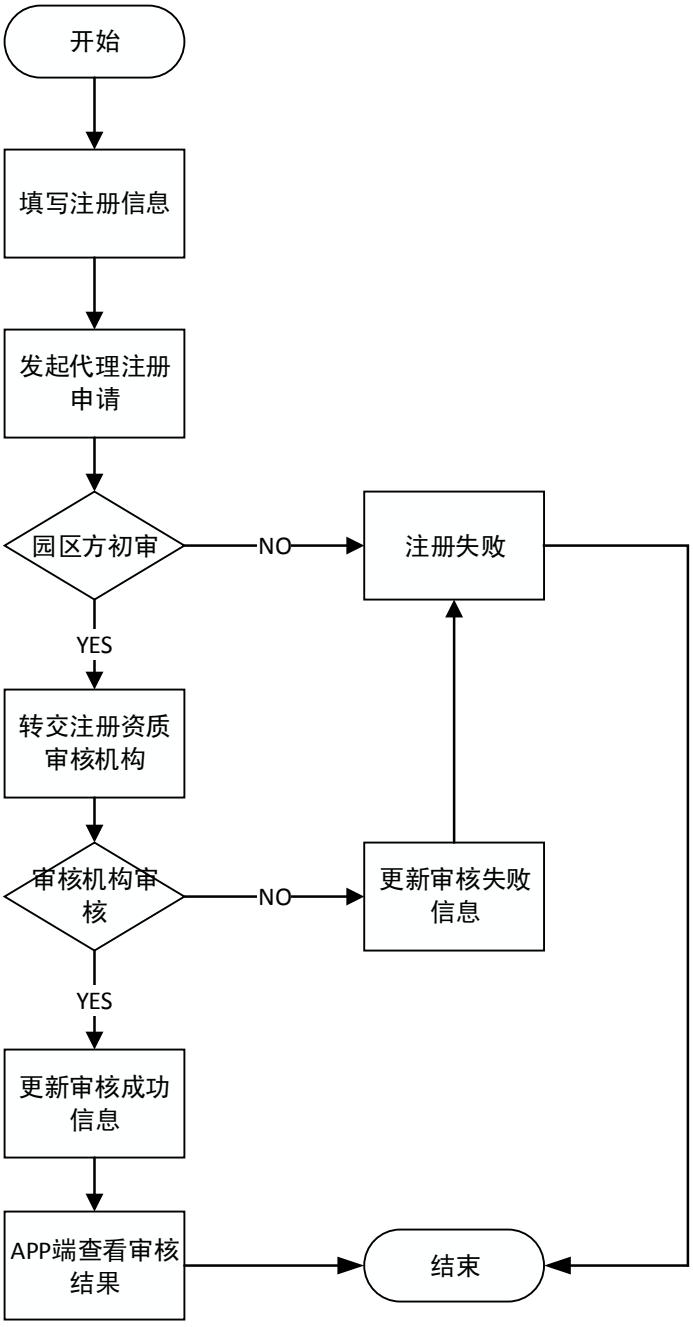

图 4-18 代理注册流程图

2. 车辆物品租赁流程

车辆物品租赁流程如图 4-19 所示。

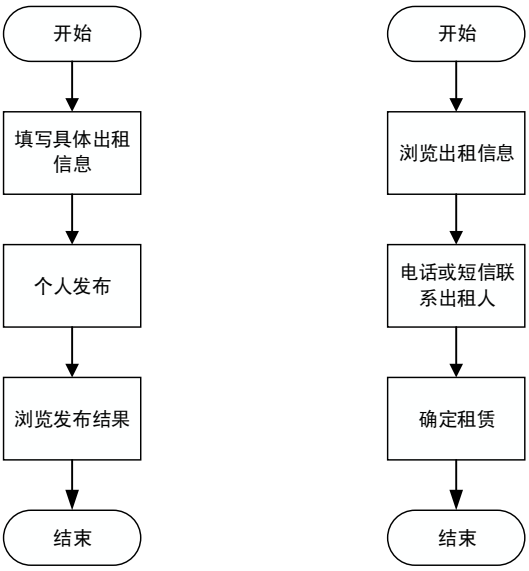

图 4-19 车辆物品租赁流程图

3. 云代工发布流程

云代工发布流程图如图 4-20 所示。

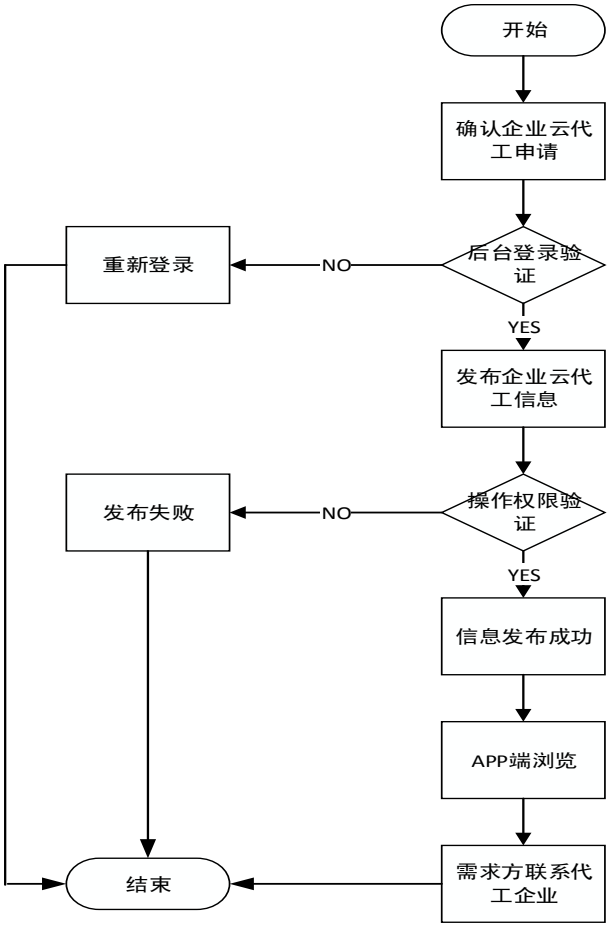

图 4-20 云代工发布流程图

4. 保洁服务流程

流程描述：

用户可以直接在 APP 端进行保洁预约，此时需要填写保洁预约表单并进行提交，物管方收到用户提交表单后会统一为用户进行第三方公司保洁服务预约。用户也可以直接通过 APP 端查看保洁公司信息，包括评价信息，服务评分等，直接通过 APP 端向保洁公司拨打电话进行预约。保洁服务流程如图 4-21 所示。

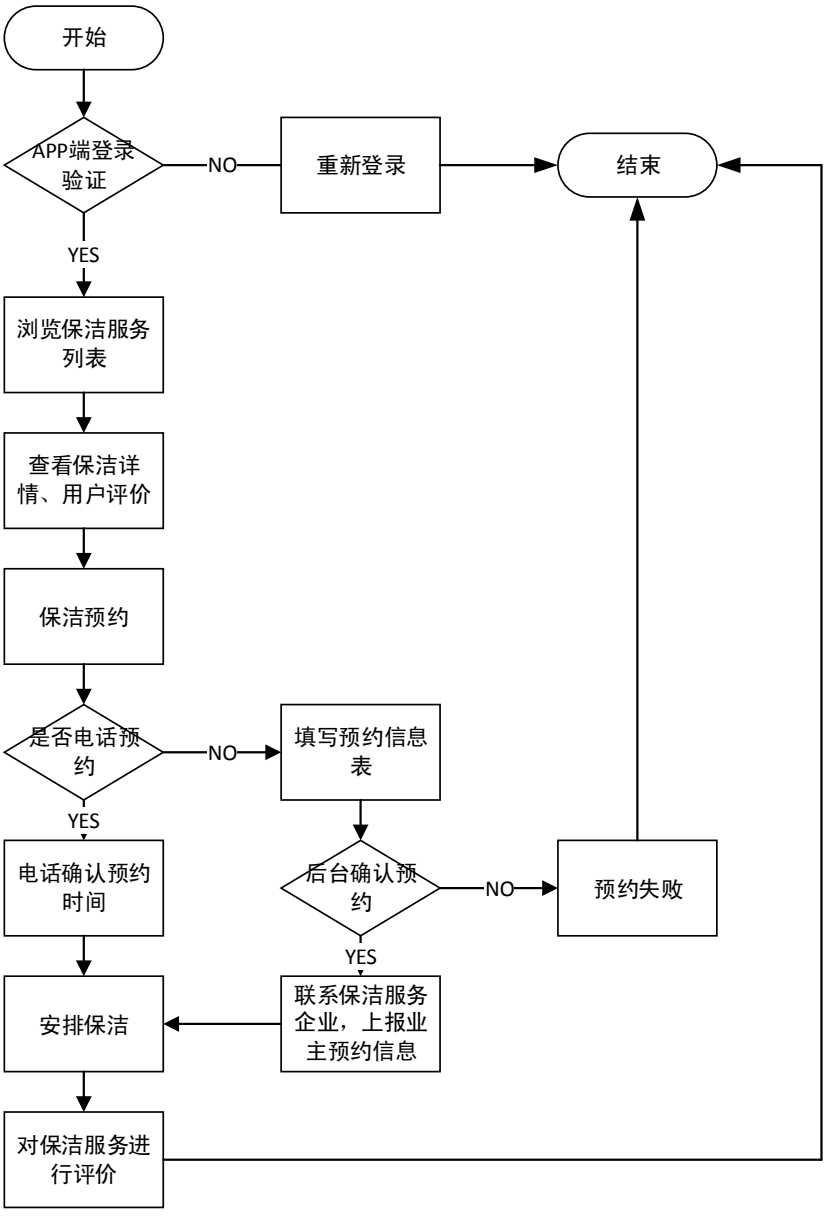

图 4-21 保洁服务流程图

5. 财务代理流程

流程描述：

财务代理子模块主要是园区方定期发布一些代理会计的服务提供信息，展示

园区相关人员的联络方式。用户可以通过 APP 端直接通过电话联系代理公司进行财务代理预约，完成代理事项。也可以在 APP 端填写代理信息表进行预约。财务代理流程如图 4-22 所示。

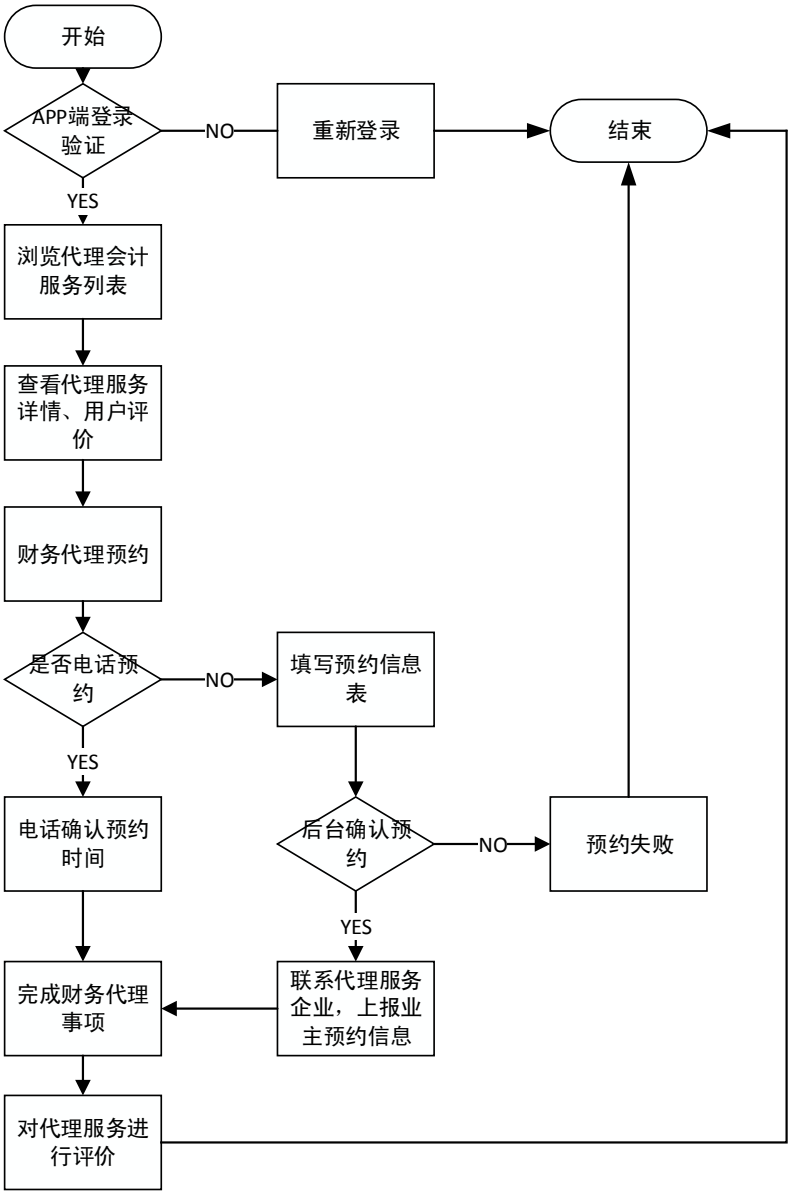

图 4-22 财务代理流程图

6. 招聘信息发布流程

流程描述：

园区企业可以在 APP 端发布用工信息或者联系物管方由物管方在后台帮助发布招聘信息。APP 端发布的招聘信息需要由园区方与用工企业沟通后进行审核。信息确认无误且审核通过后，招聘信息将在 APP 端进行发布。用户可以在 APP 端浏览到企业发布的招聘信息并进行后续职位申请。招聘信息发布流程见图 4-23。

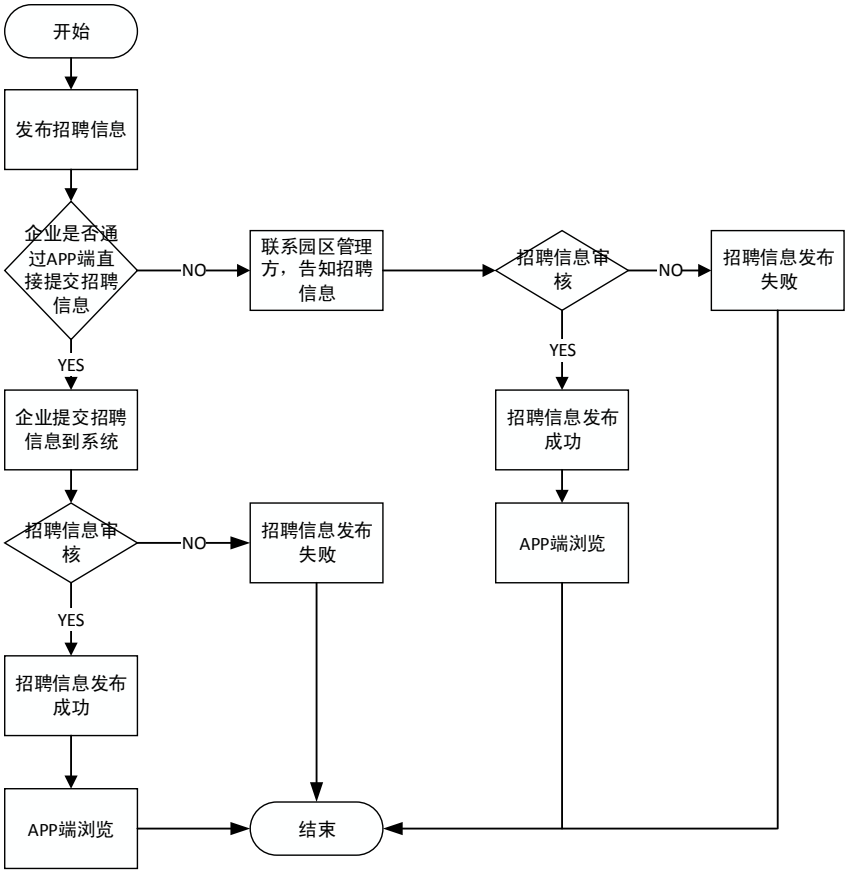

图 4-23 招聘信息发布流程图

7. 职位申请流程

职位申请流程如图 4-24 所示。

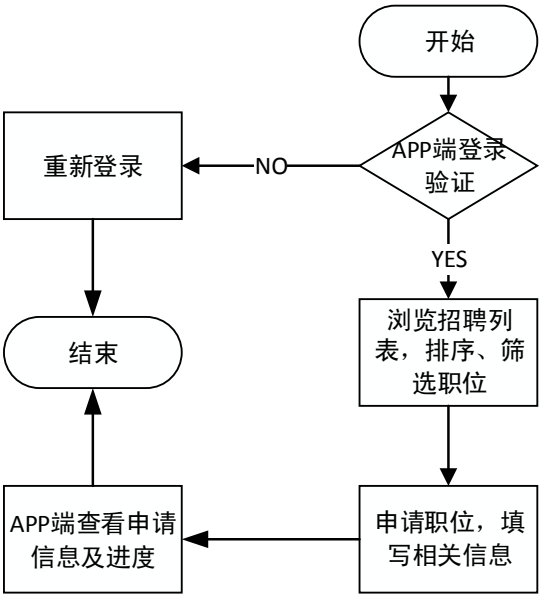

图 4-24 职位申请流程图

4.6 数据库设计

系统采用基于 Code First 原则的 ORM 数据模型设计方案，即先在代码中将系统所要使用到的数据表通过模型类和字段的形式抽象出来。

然后通过在全局配置文件中配置相应数据库的驱动及连接配置项，使用 manage.py 命令管理器中的 makemigration 命令将定义好的模型类转换生成中间数据对象，最后通过 migrate 命令将中间对象映射成数据库中的真实数据表。

Django 本身提供了一套完备的基于 QuerySet 接口的 ORM API，使得可以非常方便灵活地对数据进行增、删、改、查及其他各种操作，同时采用 ORM 方式配合 Django 的内置命令管理器可随时对表结构进行新增或者修改而不会影响已存记录。

系统数据模型设计整体上可分为系统功能管理模型、园区管理模型、孵化服务模型三部分。

4.6.1 数据库 E-R 关系图

4.6.1.1 系统功能管理部分模块 E-R 关系图

1. 用户、用户组、权限、组织机构、菜单管理 E-R 关系图见图 4-25

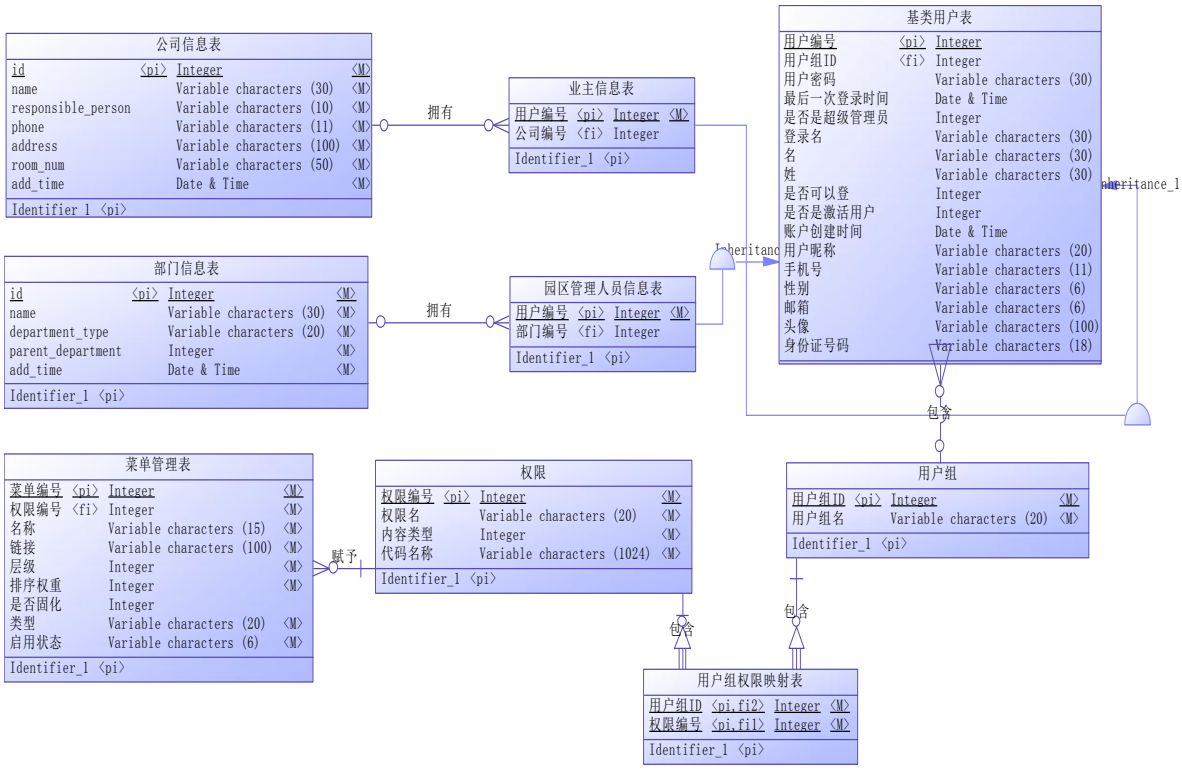

图 4-25 用户、用户组、权限、组织机构、菜单管理 E-R 关系图

2. 在线缴费 E-R 关系图见图 4-26

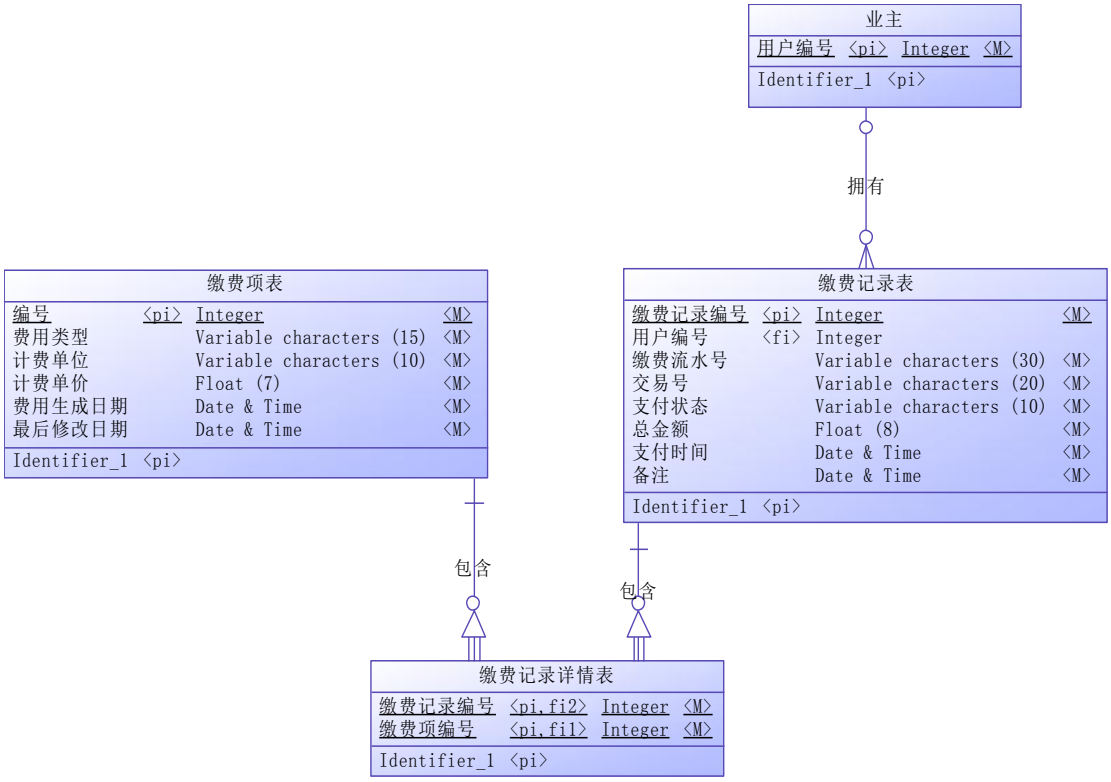

图 4-26 在线缴费 E-R 关系图

4.6.1.2 园区管理部分模块 E-R 关系图

1. 意见反馈 E-R 关系图见图 4-27

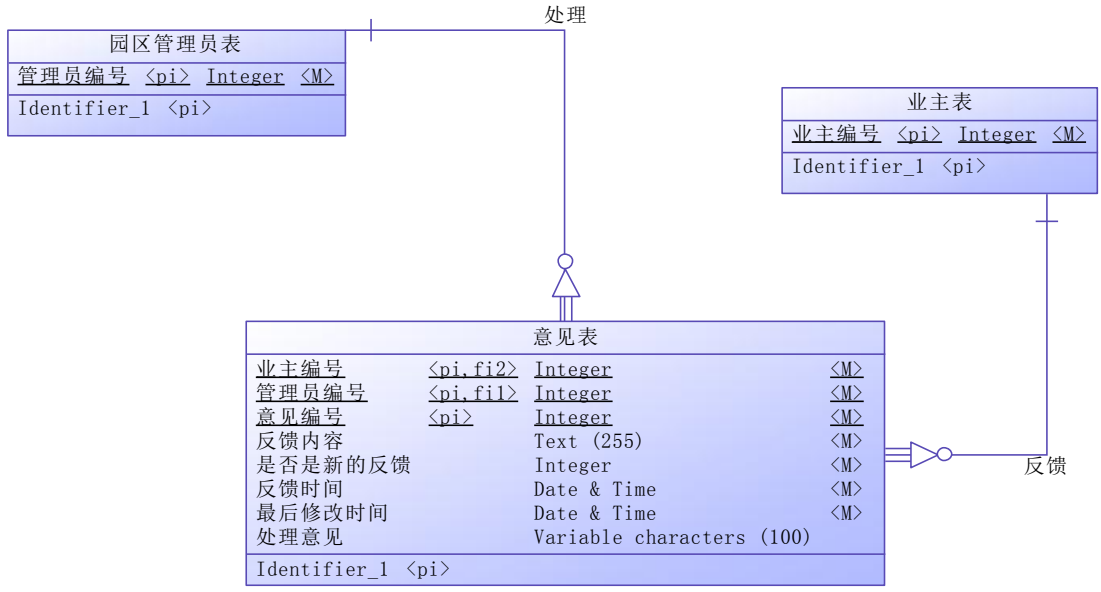

图 4-27 意见反馈 E-R 关系图

2. 广告管理 E-R 关系图见图 4-28

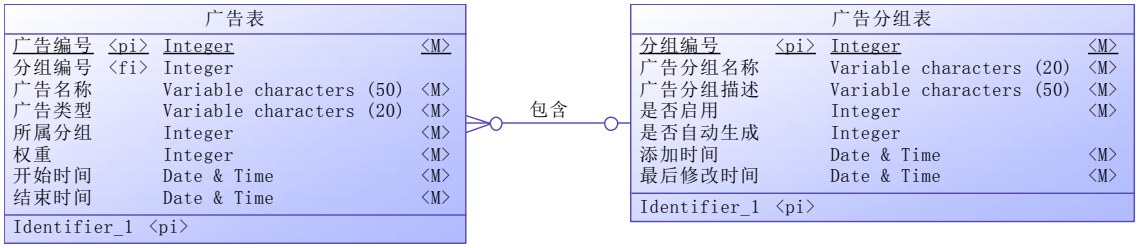

图 4-28 广告管理 E-R 关系图

4.6.1.3 孵化服务部分模块 E-R 关系图

1. 代理注册 E-R 关系图见图 4-29

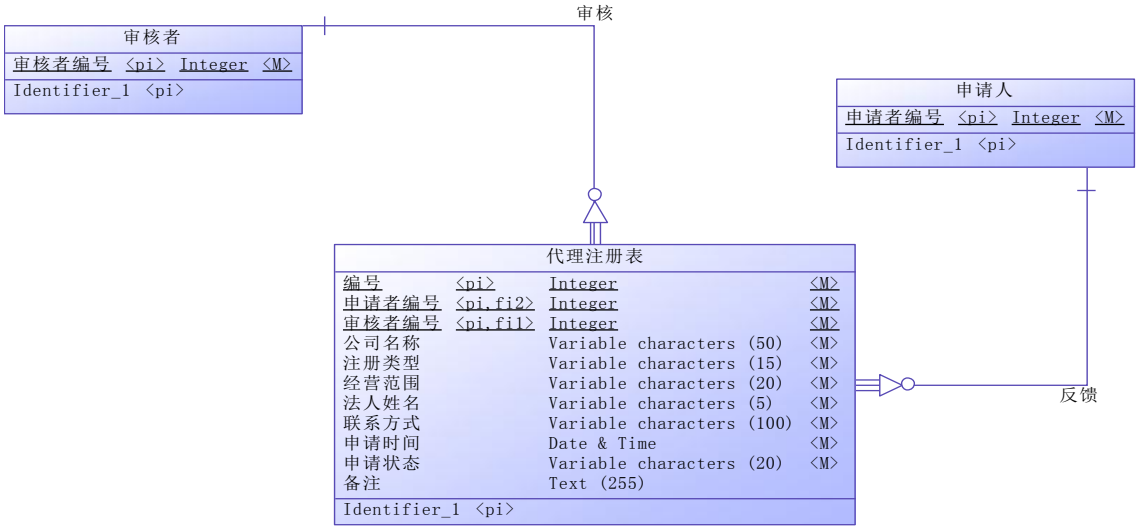

图 4-29 代理注册 E-R 关系图

2. 保洁服务 E-R 关系图见图 4-30

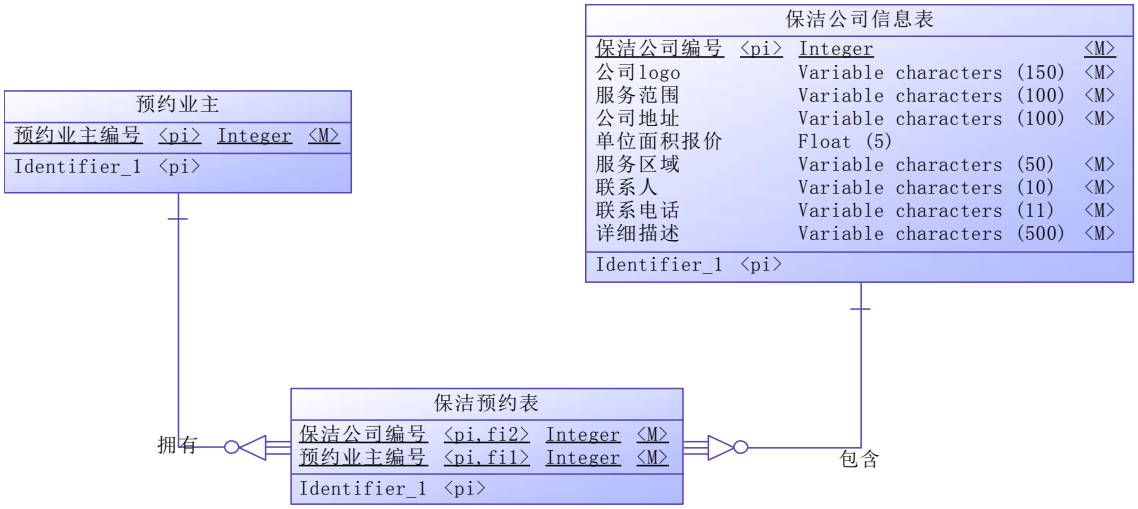

图 4-30 保洁服务 E-R 关系图

4.6.2 数据模型层设计及表结构定义

4.6.2.1 系统功能管理模型

1. 用户模型设计

Django 内置了一套 auth 用户认证模块，提供了用户认证方面的各个功能闭环，本系统基于 auth 认证模块进行二次开发，但是 auth 认证模块中提供的 User 模型只有 id、password、last_login、is_superuser、is_staff 等字段，无法满足平台系统中对用户信息的需求。

通过继承 Django 内置 auth 模块的 AbstractUser 类来扩展用户模型，因为系统中用户角色可以分为业主和园区管理人员，所以需要在继承后的用户基类基础上再扩展出两个新的子类模型，两个子类采用代理模型继承的方式来各自角色功能的扩展。

1）用户基类模型

在原有 auth 模块的 AbstractUser 模型下进行扩展，扩展后模型字段及其说明如表 4-3 所示，基类用户信息表字段见表 4-4。

表 4-3 用户基类模型

字段名	字段类型	字段描述
id	IntegerField	用户 id，主键， AbstractUser 内建字段
password	CharField	用户密码，AbstractUser 内建字段
last_login	DateTimeField	最后一次登录时间，AbstractUser 内建字段
is_superuser	SmallIntegerField	是否是系统管理员，AbstractUser 内建字段
login_name	CharField	登录名(只能是英文数字)，AbstractUser 内建字段
first_name	CharField	名，AbstractUser 内建字段
last_name	CharField	姓，AbstractUser 内建字段
is_staff	SmallIntegerField	是否可以登录后台，AbstractUser 内建字段
is_active	SmallIntegerField	是否是激活用户，AbstractUser 内建字段
date_joined	DateTimeField	账户创建时间，AbstractUser 内建字段
nickname	CharField	用户昵称，扩展字段

mobile	CharField	手机号，扩展字段
gender	CharField	性别，扩展字段
email	EmailField	邮箱，扩展字段
avatar	ImageField	用户头像，扩展字段
id_card	CharField	身份证号，扩展字段

表 4-4 基类用户信息表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	用户编号	int	11		否	主键
password	用户密码	varchar	30		否	AbstractUser 内建字段
last_login	最后一次登录时间	datetime	6		是	AbstractUser 内建字段
is_superuser	是否是超级管理员	tinyint	1		否	AbstractUser 内建字段
login_name	登录名	varchar	30		否	AbstractUser 内建字段,只能是英文数字组合
first_name	名	varchar	30		否	AbstractUser 内建字段
last_name	姓	varchar	30		否	AbstractUser 内建字段
is_staff	是否可以登录后台	tinyint	1		否	AbstractUser 内建字段
is_active	是否是激活用户	tinyint	1		否	AbstractUser 内建字段
date_joined	账户创建时间	datetime	6		否	AbstractUser 内建字段

nickname	用户昵称	varchar	20		是	
mobile	手机号	varchar	11		是	
gender	性别	varchar	6	male	否	
email	邮箱	varchar	30		是	
avatar	头像	varchar	100	default.png	是	
id_card	身份证号码	varchar	18		是	

2）业主模型

业主模型见表 4-5，业主信息表字段见表 4-6。

表 4-5 业主模型

字段名	字段类型	字段描述
id	IntegerField	业主 id，主键
company	ForeignKey	所属公司外键

表 4-6 业主信息表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	业主编号	int	11		否	主键
company_id	公司外键	int	11		否	所属公司外键

3）园区管理人员模型

园区管理人员模型见表 4-7，园区管理人员信息表字段见表 4-8。

表 4-7 园区管理人员模型

字段名	字段类型	字段描述
id	IntegerField	管理人员 id，主键
department	ForeignKey	所属部门外键

表 4-8 园区管理人员信息表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	用户编号	int	11		否	主键
department_id	部门外键	int	11		否	所属部门外键

2. 认证和授权模型设计

认证和授权功能共需要四个数据模型提供支撑，分别是用户组模型，权限模型，用户组权限映射模型和用户用户组模型。用户组和权限模型之间是多对多关系，一个用户组可以拥有多种权限，一个权限可以属于多个用户组，两者通过用户组权限映射模型进行关联，用户和用户组模型之间也是多对多关系，一个用户可以属于多个用户组，一个用户组可以拥有多个用户。

1）用户组模型

用户组模型见表 4-9，用户组信息表字段见表 4-10。

表 4-9 用户组模型

字段名	字段类型	字段描述
id	IntegerField	用户组 id(主键)
name	CharField	用户组名

表 4-10 用户组信息表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	用户编号	int	11		否	主键
name	用户组名	varchar	50		否	用户组名

2）权限模型

权限模型见表 4-11，权限信息表字段见表 4-12。

表 4-11 权限模型

字段名	字段类型	字段描述
id	IntegerField	权限 id(主键)
name	CharField	权限名

content_type	ForeignKey	内容类型
codename	CharField	代码名称

表 4-12 权限信息表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	权限编号	int	11		否	主键
name	权限名	varchar	20		否	权限名
content_type_id	内容类型	int	11		否	内容类型
codename	代码名称	varchar	20		否	代码名称

3）用户组权限映射模型

用户组权限映射模型见表 4-13，用户组权限映射表字段见表 4-14。

表 4-13 用户组权限映射模型

字段名	字段类型	字段描述
id	IntegerField	主键
group	ForeignKey	用户组外键
permission	ForeignKey	权限外键

表 4-14 用户组权限映射表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	编号	int	11		否	主键
group_id	组外键	int	11		否	组外键
permission_id	权限外键	int	11		否	权限外键

4）用户用户组模型

用户用户组模型见表 4-15，用户用户组映射表字段见表 4-16。

表 4-15 用户用户组模型

字段名	字段类型	字段描述
id	IntegerField	主键
user	ForeignKey	用户外键
group	ForeignKey	用户组外键

表 4-16 用户组用户组映射表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	编号	int	11		否	主键
user_id	用户外键	int	11		否	用户外键
group_id	用户组外键	int	11		否	用户组外键

3. 短信验证码模型设计

短信验证码模型是用户注册流程中依赖的一个数据模型，用户注册时需要填写用户基本信息并对第三方短信服务商发回的验证码进行验证，验证通过后才能成功注册，验证码验证机制能有效防止爬虫程序等恶意注册带来的系统资源耗损。短信验证码模型见表 4-17，短信验证码表字段见表 4-18。

表 4-17 短信验证码模型

字段名	字段类型	字段描述
id	IntegerField	主键
verify_code	CharField	短信验证码
mobile	CharField	注册手机号
add_time	DateTimeField	验证码生成时间

表 4-18 短信验证码表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	编号	int	11		否	主键
verify_code	短信验证码	varchar	6		否	短信验证码

mobile	注册手机号	varchar	11		否	注册手机号
add_time	验证码生成时间	datetime	6		否	验证码生成时间

4. 评论模型设计

因为考虑到系统中存在多个模块中都会用到用户评论这个功能，例如在周边商家管理中用户可以对商家服务等进行评价，在孵化服务中的保洁预约、金融服务、财务代理等子模块中也会有评论功能。如果在每个功能模块中都单独创建一个用户评价模型，那么数据将变得十分冗余。所以考虑设计一个上层抽象模型，只需在该模型中关联评论用户、被评论对象和被评论对象 id 即可实现使用同一个评价管理模型对不同评论进行集中式管理，从而可以有效防止结构相似的数据表过多而造成的数据冗余。在技术实现方面 Django 框架中集成了一个基于模型的高级通用接口 contenttypes，通过在自定义评论模型中引入 contenttypes 接口模型中的 content_type、object_id 和 content_object 即可实现多实体关联。评论模型中未涉及到回复功能。评论模型见表 4-19，评论表字段见表 4-20。

表 4-19 评论模型

字段名	字段类型	字段描述
id	IntegerField	主键
content_type	ForeignKey	内容模型类型
object_id	PositiveIntegerField	内容模型记录外键
content_object	GenericForeignKey	内容模型对象
comment_content	CharField	评论内容
comment_time	DateTimeField	评论时间

表 4-20 评论表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	编号	int	11		否	主键
content_type_id	内容模型类	int	11		否	内容模型类

	型					型
object_id	内容模型记录外键	int	11		否	内容模型记录外键
content_object_id	内容模型对象	int	11		否	内容模型对象
comment_content	评论内容	varchar	500		是	评论内容
comment_time	评论时间	datetime	6		否	评论时间

5. 菜单管理模型设计

菜单管理模型见表 4-21，菜单管理表字段见表 4-22。

表 4-21 菜单管理模型

字段名	字段类型	字段描述
id	IntegerField	主键
permission	ForeignKey	权限外键
menu_name	CharField	名称
url	CharField	链接
level	IntegerField	层级
order_weight	IntegerField	排序权重
is_fixed	BooleanField	是否固化
menu_type	CharField	类型
status	CharField	启用状态

表 4-22 菜单管理表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	主键	int	11		否	主键
permission_id	权限外键	int	11		否	权限外键

menu_name	名称	varchar	15		否	名称
url	链接	varchar	100		否	链接
level	层级	int	3	1	否	层级
order_weight	排序权重	int	3	1	否	排序权重
is_fixed	是否固化	tinyint	1		是	是否固化
menu_type	类型	varchar	20	main	否	类型
status	启用状态	varchar	6		否	启用状态

6. 推送管理模型设计

推送管理模型见表 4-23，推送管理表字段见表 4-24。

表 4-23 推送管理模型

字段名	字段类型	字段描述
id	IntegerField	主键
user_id	IntegerField	接收用户,0 表示对所有用户进行推送
content	CharField	推送内容
is_read	BooleanField	是否已读
add_time	DateTimeField	推送时间

表 4-24 推送管理表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	主键	int	11		否	主键
user_id	接收用户	int	5	0	否	接收用户
content	推送内容	varchar	300		否	推送内容
is_read	是否已读	tinyint	1	否	否	是否已读
add_time	推送时间	datetime	6		否	推送时间

7. 组织机构模型设计

组织机构根据业主和管理方不同角色可分为公司和部门两个模型。

1）公司模型

公司模型见表 4-25，公司信息表字段见表 4-26。

表 4-25 公司模型

字段名	字段类型	字段描述
id	IntegerField	主键
name	CharField	公司名
responsible_person	CharField	负责人
phone	CharField	联系电话
address	CharField	地址
room_num	CharField	房号
add_time	DateTimeField	添加时间

表 4-26 公司信息表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	主键	int	11		否	主键
name	公司名	varchar	30		否	公司名
responsible_person	负责人	varchar	10		否	负责人
phone	联系电话	varchar	11		否	联系电话
address	地址	varchar	100		否	地址
room_num	房号	varchar	50		否	房号
add_time	添加时间	datetime	6		否	添加时间

2）部门模型

部门模型见表 4-27，部门信息表字段见表 4-28。

表 4-27 部门模型

字段名	字段类型	字段描述
id	IntegerField	主键
name	CharField	部门名称
department_type	CharField	部门级别
parent_department	ForeignKey	上级部门外键
add_time	DateTimeField	添加时间

表 4-28 部门信息表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	主键	int	11		否	主键
name	部门名称	varchar	30		否	部门名称
department_type	部门级别	varchar	20		否	部门级别
parent_department	上级部门外键	int	11		否	上级部门外键
add_time	添加时间	datetime	6		否	添加时间

8. 缴费管理模型设计

缴费管理模型可分为下面三个子数据模型，分别是缴费项管理模型、缴费纪录模型和缴费详情模型。

1）缴费项管理模型

缴费项模型见表 4-29，缴费项管理表字段见表 4-30。

表 4-29 缴费项模型

字段名	字段类型	字段描述
id	IntegerField	主键
fees_type	CharField	费用类型
fees_unit	CharField	计费单位

fees_price	FloatField	计费单价
add_time	DateTimeField	费用生成日期
updated_time	DateTimeField	最后修改日期

表 4-30 缴费项管理表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	主键	int	11		否	主键
fees_type	费用类型	varchar	15		否	费用类型
fees_unit	计费单位	varchar	10		否	计费单位
fees_price	计费单价	float	7		否	计费单价
add_time	费用生成日期	datetime	6		否	费用生成日期
updated_time	最后修改日期	datetime	6		否	最后修改日期

2）缴费记录模型

缴费记录模型见表 4-31，缴费记录信息表字段见表 4-32。

表 4-31 缴费记录模型

字段名	字段类型	字段描述
id	IntegerField	主键
user	ForeignKey	业主外键
fees_sn	CharField	缴费流水号
trade_sn	CharField	交易号
pay_status	CharField	支付状态
pay_amount	FloatField	总金额
pay_time	DateTimeField	支付时间
remarks	TextField	备注

表 4-32 缴费记录信息表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	主键	int	11		否	主键
user_id	业主外键	int	5		否	业主外键
fees_sn	缴费流水号	varchar	30		否	缴费流水号
trade_sn	交易号	varchar	20		否	交易号
pay_status	支付状态	varchar	10		否	支付状态
pay_amount	总金额	float	8		否	总金额
pay_time	支付时间	datetime	6		否	支付时间
remarks	备注	text	255		是	备注

3）缴费记录详情模型

缴费记录详情模型见表 4-33，缴费记录详情表字段见表 4-34。

表 4-33 缴费记录详情模型

字段名	字段类型	字段描述
id	IntegerField	主键
fees_order_info	ForeignKey	缴费记录外键
fees_item	ForeignKey	缴费项外键
nums	IntegerField	缴费月数

表 4-34 缴费记录详情表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	主键	int	11		否	主键
fees_order_info_id	缴费记录外键	int	11		否	缴费记录外键
fees_item_id	缴费项外键	int	11		否	缴费项外键

nums	缴费月数	int	5	0	否	缴费月数

4.6.2.2 园区管理模型

1. 公告管理模型设计

公告类型可分为通知、公告、制度和标准四种，园区方选择相应类型进行最新公告的发布，公告内容由于可能存在图片、文字等混编模式，故该字段采用UeditorField 类型，该类型为集成第三方富文本模块 DjangoUeditor 后提供的一种基于 HTML 代码转义的字段类型。公告管理模型见表 4-35，公告管理信息表字段见表 4-36。

表 4-35 公告管理模型

字段名	字段类型	字段描述
id	IntegerField	公告 id，主键
notice_title	CharField	公告标题
poster	ImageField	封面海报
publisher	CharField	发布机构
notice_type	IntegerField	公告类型
notice_content	UEditorField	公告内容，富文本字段
click_nums	IntegerField	点击数
order_weight	IntegerField	排序权重
remarks	TextField	备注
add_time	DateTimeField	发布时间
updated_time	DateTimeField	最后修改时间

表 4-36 公告管理信息表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	公告编号	int	11		否	主键

notice_title	公告标题	varchar	50		否	
poster	封面海报	varchar	100		是	
publisher	发布机构	varchar	30	浩旺管理有限公司	是	
notice_type	公告类型	int	11	1	否	
notice_content	公告内容	longtext	0		否	
click_nums	点击数	int	11		否	
order_weight	排序权重	int	3	1	否	数字越大，权重越重
remarks	备注	longtext	0		是	
add_time	发布时间	datetime	6		否	
updated_time	最后修改时间	datetime	6		否	

2. 意见反馈模型设计

意见反馈模型见表 4-37，意见反馈表字段见表 4-38。

表 4-37 意见反馈模型

字段名	字段类型	字段描述
id	IntegerField	意见反馈 id，主键
user	ForeignKey	用户外键，用于标识提交意见反馈用户
feedback	TextField	反馈内容
is_new	BooleanField	是否是新的反馈
add_time	DateTimeField	反馈时间
updated_time	DateTimeField	最后修改时间
responsible_person	ForeignKey	处理人外键，用于标识处理人
result	CharField	处理意见

表 4-38 意见反馈表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	主键	int	11		否	主键
user_id	提交意见反馈用户	int	5		否	提交意见反馈用户
feedback	反馈内容	text	255		是	反馈内容
is_new	是否是新的反馈	tinyint	1	0	否	是否是新的反馈
add_time	反馈时间	datetime	6		否	反馈时间
updated_time	最后修改时间	datetime	6		否	最后修改时间
responsible_person	处理人外键，用于标识处理人	int	5		否	处理人外键，用于标识处理人
result	处理意见	varchar	100		否	处理意见

3. 活动管理模型设计

根据需求分析活动管理模型可分为两个个模型，分别是活动模型，和活动报名模型。活动模型包含活动相关的基本信息，活动报名模型则包含用户报名活动时填报的相关信息。报名模型中的状态字段标明了当前活动报名的审核状态。

1）活动模型

活动模型见表 4-39，活动信息表字段见表 4-40。

表 4-39 活动模型

字段名	字段类型	字段描述
id	IntegerField	活动 id，主键
activity_title	CharField	活动标题
activity_type	CharField	活动类型

reg_nums	IntegerField	报名人数
begin_time	DateTimeField	报名开始时间
end_time	DateTimeField	报名截止时间
remarks	TextField	备注
add_time	DateTimeField	添加时间
updated_time	DateTimeField	最后修改时间

表 4-40 活动信息表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	活动 id，主键	int	11		否	活动 id，主键
activity_title	活动标题	varchar	30		否	活动标题
activity_type	活动类型	varchar	30	园区活动	否	活动类型
reg_nums	报名人数	int	5	0	否	报名人数
begin_time	报名开始时间	datetime	6		否	报名开始时间
end_time	报名截止时间	datetime	6		否	报名截止时间
remarks	备注	text	255		是	备注
add_time	添加时间	datetime	6		否	添加时间
updated_time	最后修改时间	datetime	6		否	最后修改时间

2）活动报名模型

活动报名模型见表 4-41，活动报名表字段见表 4-42。

表 4-41 活动报名模型

字段名	字段类型	字段描述

id	IntegerField	主键
user	ForeignKey	报名用户外键
activity	ForeignKey	报名活动外键
status	CharField	报名状态
remarks	TextField	备注
add_time	DateTimeField	添加时间
updated_time	DateTimeField	最后修改时间

表 4-42 活动报名表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	主键	int	11		否	主键
user	报名用户外键	int	11		否	报名用户外键
activity	报名活动外键	int	11		否	报名活动外键
status	报名状态	varchar	6	未报名	否	报名状态
remarks	备注	text	255		是	备注
add_time	添加时间	datetime	6		否	添加时间
updated_time	最后修改时间	datetime	6		否	最后修改时间

4. 周边商家管理模型设计

周边商家管理模型主要包含以下字段：商家名字、商家简介、商家地址、商家类型、商家联系电话、商家联系人、商家评价星级、是否推荐、人均消费、添加时间、修改时间等。是否推荐字段为布尔型类型，通过后台设置该字段为 true，会将该商家在 APP 端推荐商户位置置顶显示，同为推荐商户的则按最后修改时间倒叙显示。周边商家管理模型见表 4-43，周边商家管理表字段见表 4-44。

表 4-43 周边商家管理模型

字段名	字段类型	字段描述

id	IntegerField	主键
merchant_name	CharField	商家名字
brief_intro	UEditorField	商家简介
address	CharField	商家地址
merchant_type	CharField	商家类型
phone	CharField	商家联系电话
contact_person	CharField	商家联系人
rates	CharField	商家评价星级
is_recomm	BooleanField	是否推荐
avg_money	FloadField	人均消费
add_time	DateTimeField	添加时间
updated_time	DateTimeField	最后修改时间

表 4-44 周边商家管理表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	主键	int	11		否	主键
merchant_name	商家名字	varchar	20		否	商家名字
brief_intro	商家简介	longtext	0		否	商家简介
address	商家地址	varchar	30		否	商家地址
merchant_type	商家类型	varchar	10		否	商家类型
phone	商家联系电话	varchar	11		否	商家联系电话
contact_person	商家联系人	varchar	5		否	商家联系人
rates	商家评价星级	varchar	5		否	商家评价星级
is_recomm	是否推荐	tinyint	1	0	否	是否推荐

avg_money	人均消费	float	7		否	人均消费
add_time	添加时间	datetime	6		否	添加时间
updated_time	最后修改时间	datetime	6		否	最后修改时间

5. 广告分组管理模型设计

广告分组管理模型见表 4-45，广告分组管理表字段见表 4-46。

表 4-45 广告分组管理模型

字段名	字段类型	字段描述
id	IntegerField	广告分组 id，主键
ad_group_name	CharField	广告分组名称
ad_group_desc	CharField	广告分组描述
is_active	BooleanField	是否启用
is_auto_gen	BooleanField	是否自动生成
add_time	DateTimeField	添加时间
updated_time	DateTimeField	最后修改时间

表 4-46 广告分组管理表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	广告分组 id，主键	int	11		否	广告分组 id，主键
ad_group_name	广告分组名称	varchar	20		否	广告分组名称
ad_group_desc	广告分组描述	varchar	20		否	广告分组描述
is_active	是否启用	tinyint	1	0	否	是否启用
is_auto_gen	是否自动生成	tinyint	1	0	否	是否自动生成
add_time	添加时间	datetime	6		否	添加时间
updated_time	最后修改时间	datetime	6		否	最后修改时间

6. 广告管理模型设计

广告管理模型见表 4-47，广告管理表字段见表 4-48。

表 4-47 广告管理模型

字段名	字段类型	字段描述
id	IntegerField	广告 id，主键
ad_name	CharField	广告名称
ad_type	CharField	广告类型
ad_group	ForeignKey	所属分组
order_weight	IntegerField	权重
begin_time	DateTimeField	开始时间
end_time	DateTimeField	结束时间

表 4-48 广告管理表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	广告 id，主键	int	11		否	广告 id，主键
ad_name	广告名称	varchar	20		否	广告名称
ad_type	广告类型	varchar	20		否	广告类型
ad_group_id	所属分组	int	11		否	所属分组
order_weight	权重	int	3	1	否	权重
begin_time	开始时间	datetime	6		否	开始时间
end_time	结束时间	datetime	6		否	结束时间

7. 访客管理模型设计

访客管理模型见表 4-49，访客管理表字段见表 4-50。

表 4-49 访客管理模型

字段名	字段类型	字段描述

id	IntegerField	主键
visitor_name	CharField	访客姓名
visitor_type	CharField	访客类型，访客或随同
gender	CharField	访客性别
visit_place	CharField	到访地点
visit_time	DateTimeField	到访时间
leave_time	DateTimeField	离开时间
contacts	CharField	联系方式
id_card	CharField	身份证号
id_card_image	ImageField	身份证照片
reason	CharField	到访原因

表 4-50 访客管理表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	主键	int	11		否	主键
visitor_name	访客姓名	varchar	10		否	访客姓名
visitor_type	访客类型，访客或随同	varchar	9		否	访客类型，访客或随同
gender	访客性别	varchar	6		否	访客性别
visit_place	到访地点	varchar	200		否	到访地点
visit_time	到访时间	datetime	6		否	到访时间
leave_time	离开时间	datetime	6		否	离开时间
contacts	联系方式	varchar	11		否	联系方式
id_card	身份证号	varchar	18		否	身份证号
id_card_image	身份证照片	varchar	255		是	身份证照片

reason	到访原因	varchar	500		是	到访原因

8. 物流快递管理模型设计

物流快递管理模型见表 4-51，物流快递管理表字段见表 4-52。

表 4-51 物流快递管理模型

字段名	字段类型	字段描述
id	IntegerField	主键
name	CharField	物流公司名
logo	ImageField	logo
phone	CharField	联系电话
add_time	DateTimeField	添加时间
updated_time	DateTimeField	最后修改时间

表 4-52 物流快递管理表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	主键	int	11		否	主键
name	物流公司名	varchar	15		否	物流公司名
logo	logo	varchar	200		否	logo
phone	联系电话	varchar	11		否	联系电话
add_time	到访地点	datetime	6		否	添加时间
updated_time	到访时间	datetime	6		否	最后修改时间

4.6.2.3 孵化服务模型

1. 代理注册模型设计

代理注册模型见表 4-53，代理注册管理表字段见表 4-54。

表 4-53 代理注册模型

字段名	字段类型	字段描述
id	IntegerField	主键
applicant	ForeignKey	申请人
company_name	CharField	公司名称
reg_type	CharField	注册类型
business_scope	CharField	经营范围
legal_person	CharField	法人姓名
contacts	CharField	联系方式
apply_time	DateTimeField	申请时间
approver	ForeignKey	审核者
status	CharField	申请状态
remarks	TextField	备注

表 4-54 代理注册管理表

字段名	字段描述	类型	长度	默认值	是否允许为空	说明
id	主键	int	11		否	主键
applicant_id	申请人	int	11		否	申请人
company_name	公司名称	varchar	50		否	公司名称
reg_type	注册类型	varchar	15		否	注册类型
business_scope	经营范围	varchar	20		否	经营范围
legal_person	法人姓名	varchar	5		否	法人姓名
contacts	联系方式	varchar	11		否	联系方式
apply_time	申请时间	datetime	6		否	申请时间
approver	审核者	int	11		否	审核者

status	申请状态	varchar	20	已提交	否	申请状态
remarks	备注	text	255		是	备注

其他功能数据模型和上述模型类似，在此处不再赘述。

4.7 本章小结

本章首先提出了平台系统的设计目标与原则，然后对系统整体架构设计进行了详细的阐述，对系统功能模块进行了划分与设计。在系统软件架构设计中，分层阐述了系统实现过程中将涉及到的数据模型层、视图层、路由、认证和权限控制设计方案与技术选型。最后对系统中部分关键业务流程进行了详细设计，并对生成的部分数据表做了说明，为下一步系统的实现做好了准备。

第五章 系统实现

5.1 系统开发环境

主要开发环境如下：

计算机硬件：Intel(R) Core(TM) i5-7300HQ CPU@2.50GHz 2.50 GHz，内存8.00GB。

操作系统：Windows 10 专业版。

Python 版本：Python 3.6.3。

开发 IDE：Pycharm professional 2017.3。

Python 虚拟环境：Virtualenv 15.1.0。

数据库：MySQL 5.7.18。

缓存数据库：Redis 3.2。

5.2 系统主要功能实现

5.2.1 项目整体规划

项目总体目录结构如下图所示，其中 apis 目录下分别针对系统所需要实现的系统管理、园区管理和孵化服务进行分目录管理，其中 filters.py 文件主要用来实现 RESTful API 中的过滤器，serializers.py 文件是用来实现序列化器，signals.py 文件实现一些自定义信号量，作为全局钩子使用，views.py 实现 RESTful API 的视图层。

apps 目录下根据功能划分出不同的 app，主要用于后台管理系统，其中 models.py 为数据模型层，可供后台管理系统和 RESTful API 服务层共同使用。HaoWangPark 目录下 settings.py 为全局基础配置文件，staging.py 为测试服务器配置文件，prod.py 为生产服务器配置文件，urls.py 为一级路由配置文件，wsgi.py 则是 uWSGI 服务器配置文件。

vendor_apps 目录下是项目中所需要依赖的一些第三方库及框架，项目目录结构规划如图 5-1 所示。

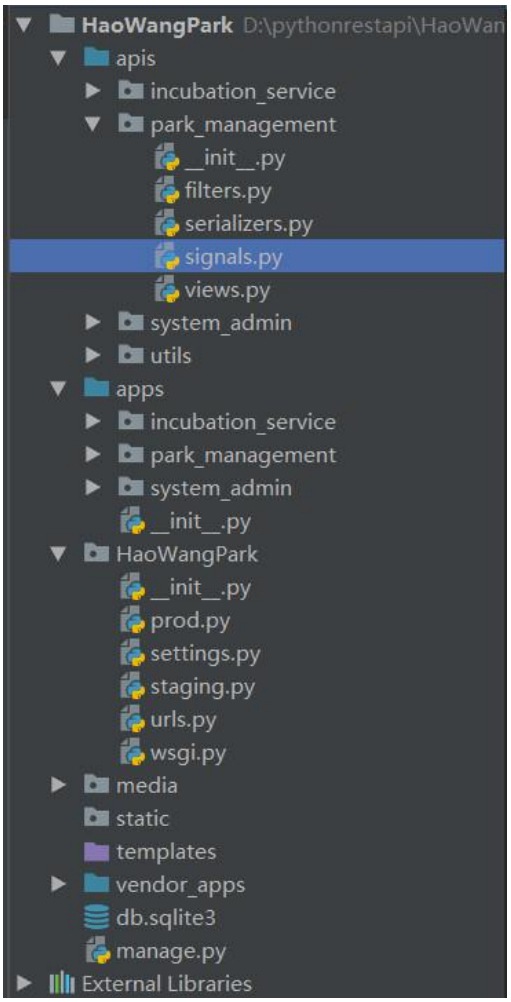

图 5-1 项目目录结构规划

5.2.2 系统功能管理实现

5.2.2.1 用户登录实现

1. 代码设计思路

用户在 APP 端使用手机号和密码进行登录，如果手机号和密码和之前注册时存入数据库中的是一样的，则登录成功，RESTful API 层路由到登录视图，登录视图里调用 JSON Web Token 生成代码，将生成的凭证颁发给客户端，凭证中的 payload 段将记录用户的 user_id、username、email 和凭证过期时间等信息并采用 Base64 格式进行编码，在凭证过期之前，客户端只需要在 HTTP header 里的带上 Authorization 字段，设置其值为后台里配置的 JSON Web Token 前缀+凭证，客户端带上来的凭证和之前在服务器端存储的凭证进行比对，如果一致则身份认证成功。

如果登录验证未通过，如果是因为用户未提交账号或者密码则提示响应字段

不能为空，如果是账号或者密码错误则使用统一键:non_field_erros 返回错误提示，HTTP 状态码返回 400。

2. 核心代码

1）路由代码片段：

```
urlpatterns = [
    ......
    url(r'^login_jwt_auth/', obtain_jwt_token),
    # 过期后重新获取 JWT 接口
    url(r'^refresh_jwt_token/', refresh_jwt_token),
    # JWT 验证接口
    url(r'^verify_jwt_token/', verify_jwt_token),
    ......
]
```

2）视图层代码片段：

```
class JSONWebTokenAPIView(APIView):
    #处理客户端登录信息验证，生成并颁发 JWT 凭证
    def post(self, request, *args, **kwargs):
        serializer = self.get_serializer(data=request.data)

        if serializer.is_valid():
            user = serializer.object.get('user') or request.user
            token = serializer.object.get('token')
            response_data = jwt_response_payload_handler(token, user, request)
            response = Response(response_data)
            if api_settings.JWT_AUTH_COOKIE:
                expiration = (datetime.utcnow() +
                              api_settings.JWT_EXPIRATION_DELTA)
                response.set_cookie(api_settings.JWT_AUTH_COOKIE,
                                    token,
                                    expires=expiration,
                                    httponly=True)
            return response

        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

#生成 JWT
class ObtainJSONWebToken(JSONWebTokenAPIView):
    serializer_class = JSONWebTokenSerializer

#验证 JWT
class VerifyJSONWebToken(JSONWebTokenAPIView):
    serializer_class = VerifyJSONWebTokenSerializer

#刷新 JWT
class RefreshJSONWebToken(JSONWebTokenAPIView):
    serializer_class = RefreshJSONWebTokenSerializer
```

3）配置 JWT 过期时间和前缀：

```
JWT_AUTH = {
    'JWT_EXPIRATION_DELTA': datetime.timedelta(days=3),
    'JWT_AUTH_HEADER_PREFIX': 'HaoWang-JWT'
```

```
}
```

3. 实现截图

图 5-2 APP 端登录界面

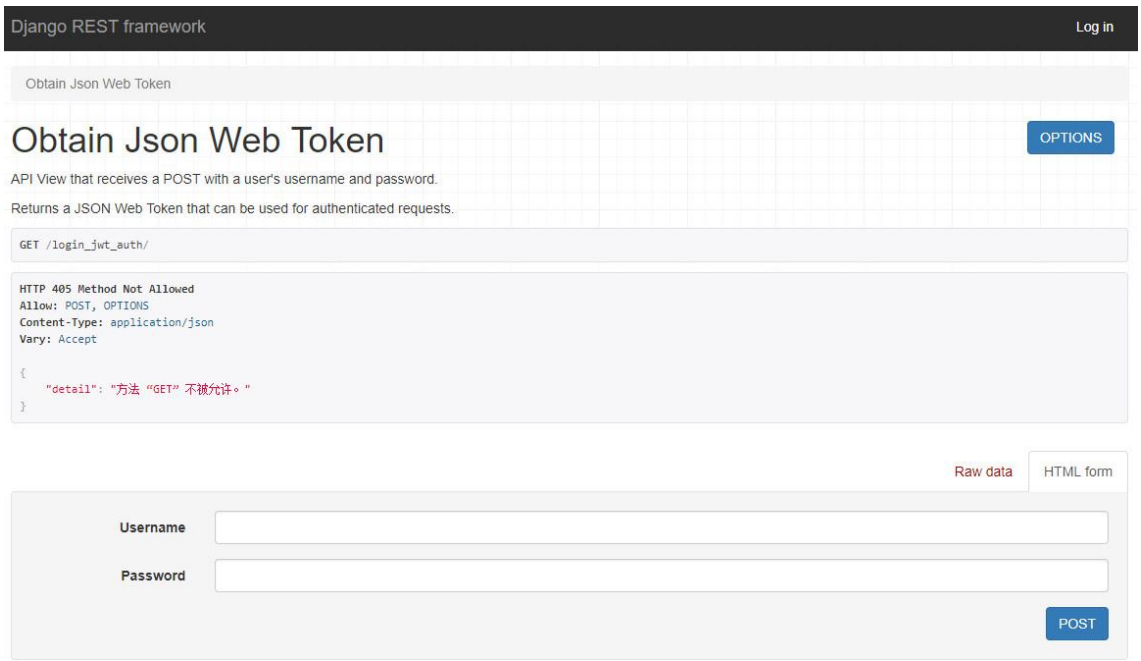

图 5-3 用户认证接口

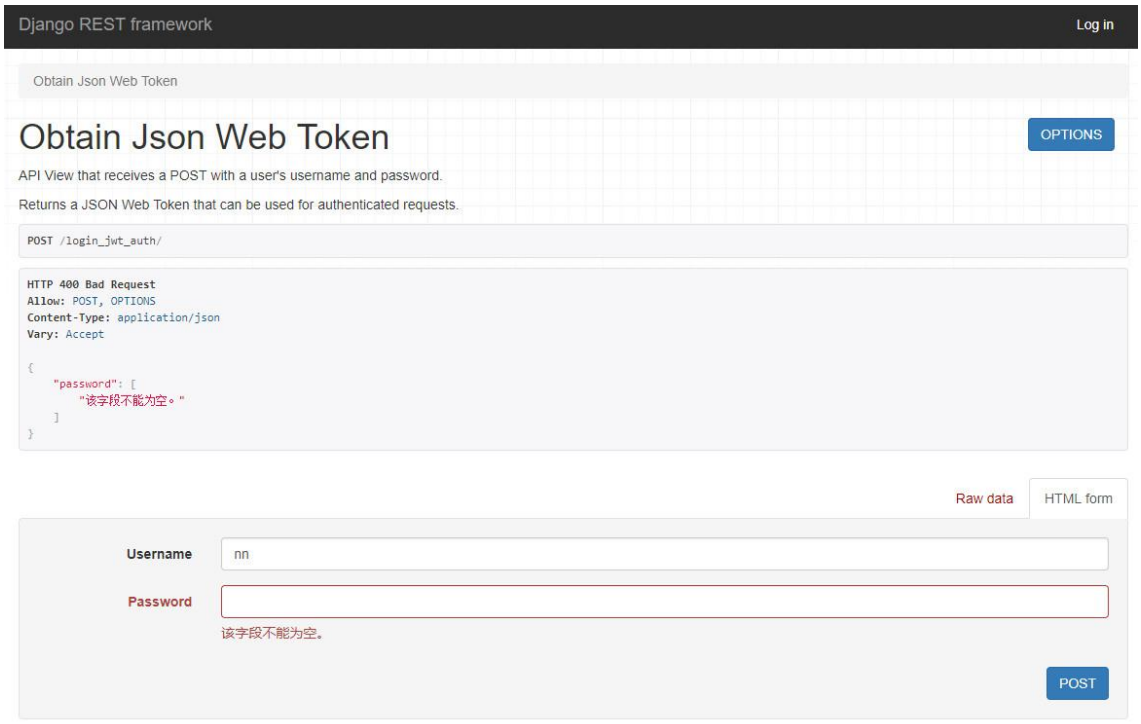

图 5-4 用户登录失败处理场景 1

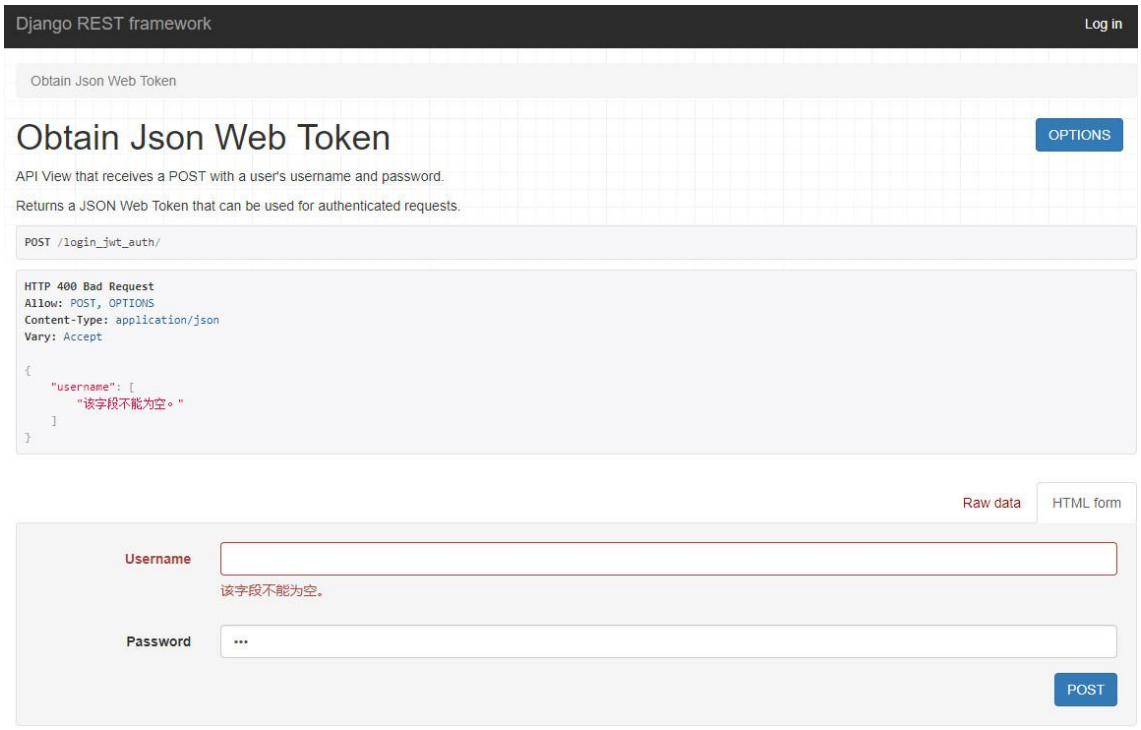

图 5-5 用户登录失败处理场景 2

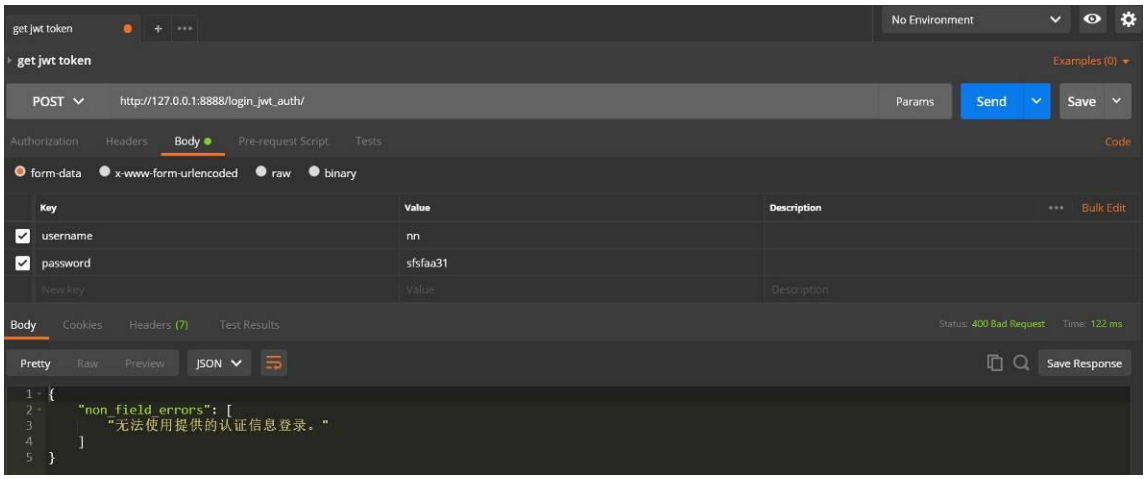

图 5-6 用户登录失败处理场景 3

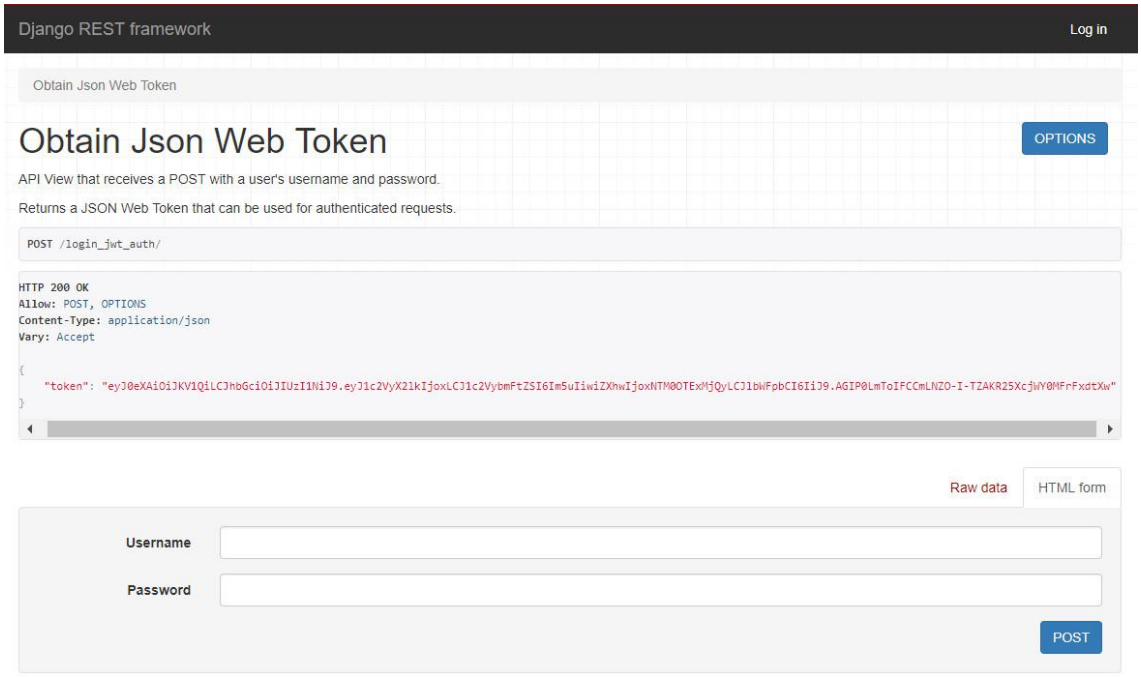

图 5-7 用户登录成功获得 JWT

5.2.3 园区管理实现

5.2.3.1 公告管理实现

1. 代码设计思路

首先使用 Djaong 自带的命令管理器命令 python startapp park_management 命令生成园区管理 app，在 settings.py 中 INSTALLED_APPS 配置项中注册该生成的 app，依据第四章中有关公告管理的模型层设计进行模型层代码编写，检查无误后，在项目根目录下执行 python manage.py makemigrations 生成数据库中间对象再执行

python manage.py migrate 进行数据库迁移。

因为公告管理只需要客户端 GET 请求后得到数据后展示，没有提交操作，所以序列化器只需要继承 ModelSerializer，在元数据中配置模型和字段，这里没有敏感字段和需要验证的数据，所以直接可以使用“__all__”进行一一映射。

2. 核心代码

1）模型层主要代码片段：

```
from Django.contrib.auth.models import AbstractUser
from Django.db import models
from DjangoUeditor.models import UeditorField

# Create your models here.
Class Notice(models.Model):
“””
公告管理
“””
NOTICE_TYPE = (
        (1, “通知”),
        (2, “公告”),
        (3, “制度”),
        (4, “标准”),
    )

    notice_title = models.CharField(max_length=50, verbose_name=”公告标题”, help_text=”公告标题”)

    poster = models.ImageField(upload_to=”notices/posters/”, verbose_name=”封面海报”, null=True, blank=True,help_text=”封面海报”)

    publisher = models.CharField(max_length=30, verbose_name=”发布机构”, null=True, blank=True, help_text=”发布机构”,default=”浩旺管理有限公司”)

    notice_type = models.IntegerField(choices=NOTICE_TYPE, default=1, verbose_name=”公告类型”,help_text=”公告类型”)

    notice_content = UeditorField(verbose_name=u”公告内容”, imagePath=”notices/images/”, width=1000, height=200,filePath=”notices/files/”, default=’’)

    click_nums = models.IntegerField(default=0, verbose_name=”点击数”, help_text=”点击数”)

    order_weight = models.IntegerField(default=1, verbose_name=”权重”, help_text=”权重”)

    remarks = models.TextField(verbose_name=”备注”, null=True, blank=True, help_text=”备注”)

    add_time = models.DateTimeField(auto_now_add=True, verbose_name=”添加时间”)

    updated_time = models.DateTimeField(auto_now=True, verbose_name=”最后修改时间”)

    # 元数据
    class Meta:
```

```
        verbose_name = "公告管理"
        verbose_name_plural = verbose_name
        # 数据按权重和创建时间倒序
        ordering = ['-order_weight', '-add_time']

    # 重写对象 toString()方法
    def __str__(self):
        return self.notice_title
```

2）序列化器主要代码片段：

```
from rest_framework import serializers
from apps.park_management.models import Notice

class NoticeSerializer(serializers.ModelSerializer):
    # 元数据
    class Meta:
        # 关联模型是 Notice
        model = Notice
        # 关联字段是 Notice 中所有字段
        fields = "__all__"
```

3）视图层主要代码片段：

```
from rest_framework import filters
from Django_filters.rest_framework import DjangoFilterBackend
from rest_framework import mixins, viewsets
from rest_framework.pagination import PageNumberPagination

from apps.park_management.models import Notice
from .serializers import NoticeSerializer

# 自定义局部分页器类
class NoticePagination(PageNumberPagination):
    # 每页 6 条记录
    page_size = 6
    # 客户端查询字符串中自定义每页记录数字段名称
    page_size_query_param = 'page_size'
    # 客户端查询字符串中自定义第几页字段名称
    page_query_param = "page"
    # 允许的最大页面大小
    max_page_size = 100


class NoticeViewSet(CacheResponseMixin, mixins.ListModelMixin, mixins.RetrieveModelMixin,
viewsets.GenericViewSet):
    """
    公告管理 API
    list:
        返回通知公告列表，可根据不同字段进行结果筛选
    retrieve:
        返回具体某一个公告列表详情
    """
```

```
    # 配置查询集
    queryset = Notice.objects.all()
    # 配置序列化器
    serializer_class = NoticeSerializer
    # 配置局部分页器
    pagination_class = NoticePagination
    # 配置搜索和过滤器
    filter_backends = (DjangoFilterBackend, filters.SearchFilter, filters.OrderingFilter)
    # 配置可供精确过滤字段元组
    filter_fields = ()
    # 配置可供模糊搜索的字段元组
    search_fields = ('notice_title', 'publisher', 'notice_content')
    # 配置可供排序的字段元组
    ordering_fields = ('click_nums', 'order_weight', 'add_time')
```

4）路由代码片段：

```
from Django.conf.urls import include, url
from Django.views.static import serve
from rest_framework.documentation import include_docs_urls
from rest_framework.routers import DefaultRouter

import xadmin
from HaoWangPark.settings import MEDIA_ROOT
from apis.park_management.views import NoticeViewSet

router = DefaultRouter()

# 配置 notice 的 url
# 园区管理-公告管理
router.register(r'notices', NoticeViewSet, base_name='notices')
urlpatterns = [
    # 配置 Django REST framework 一级路由
    url(r'^api/', include(router.urls)),
    ......
]
```

3. 实现截图

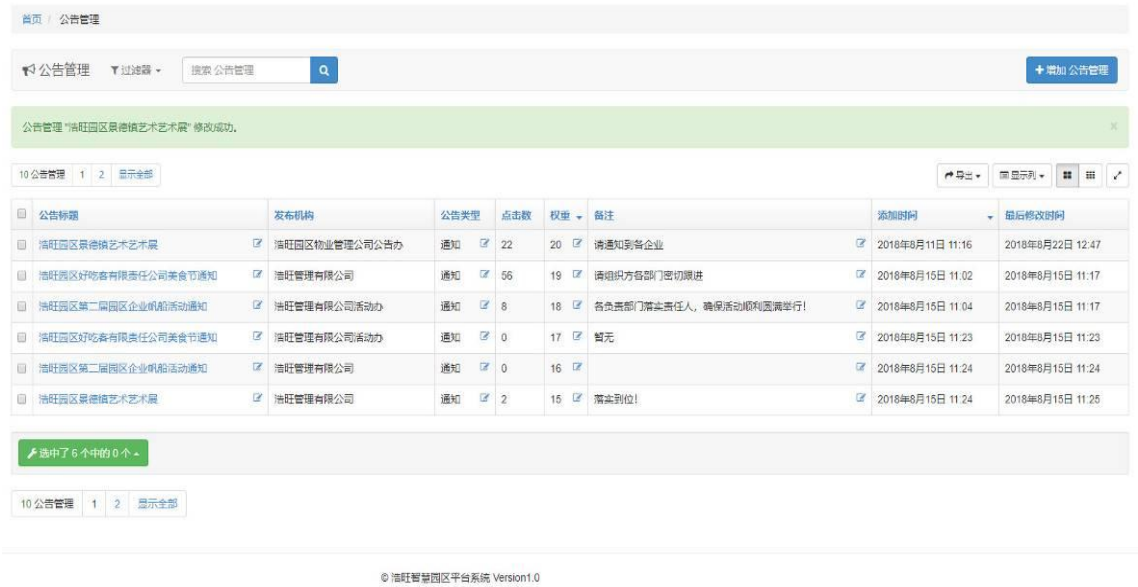

图 5-8 系统后台公告管理列表页

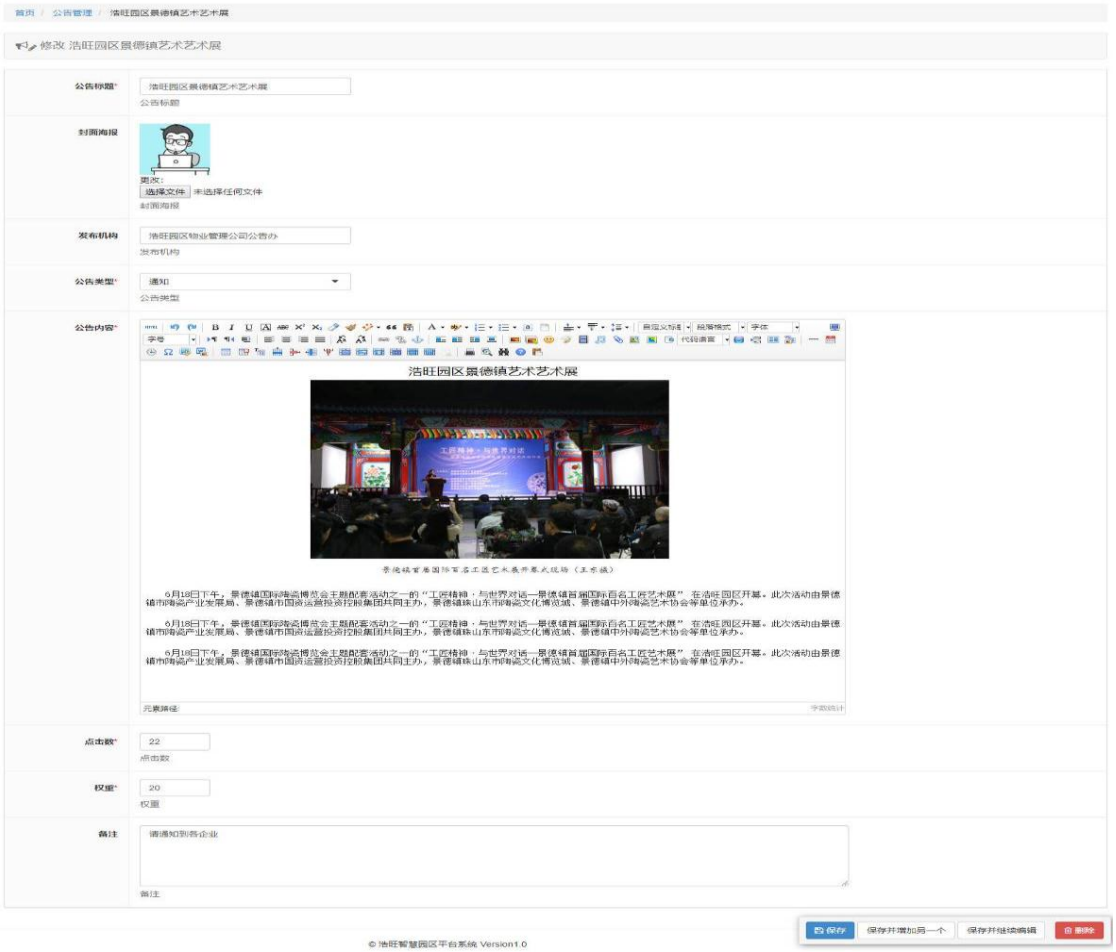

图 5-9 系统后台公告管理详情页

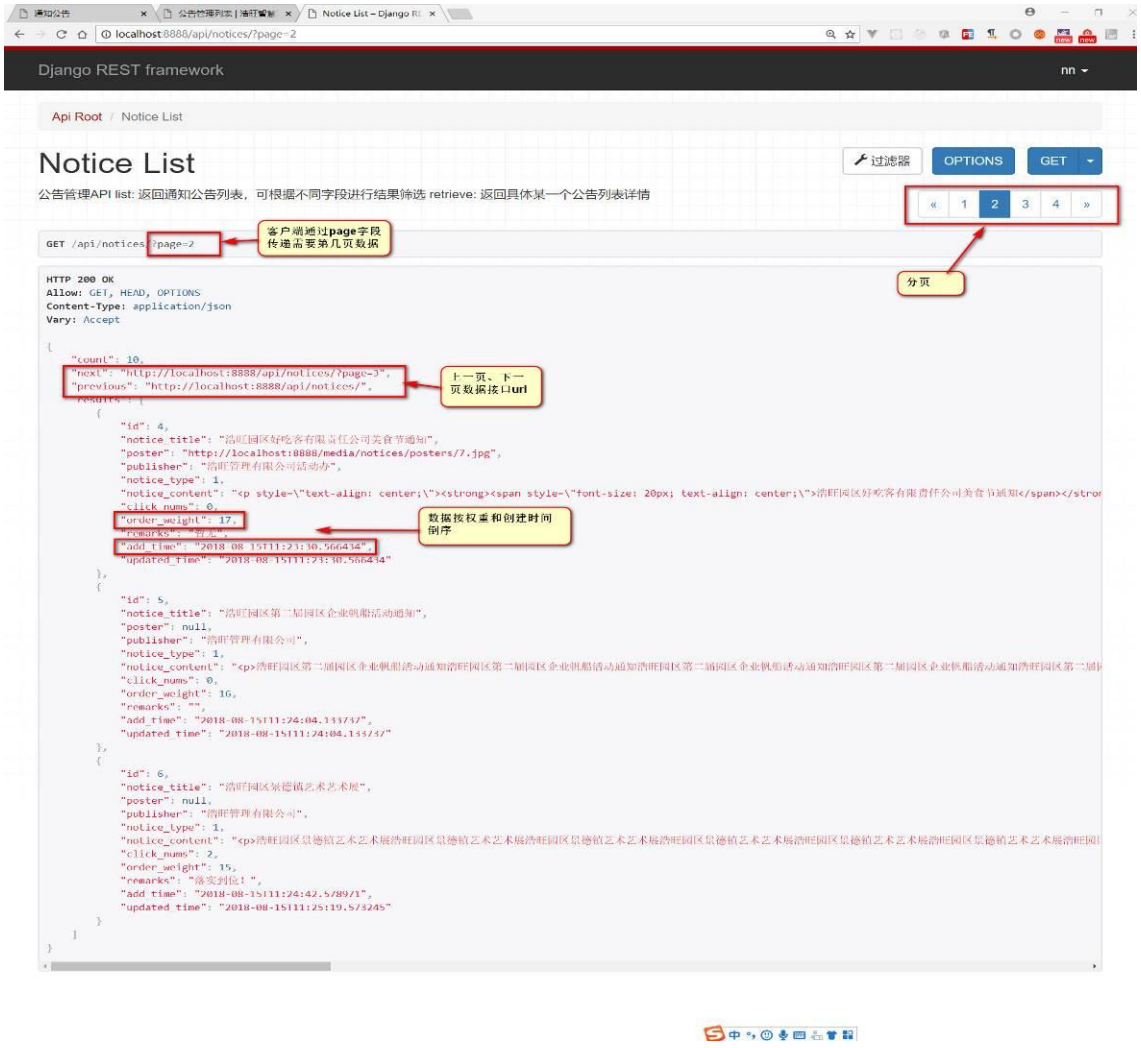

图 5-10 公告管理列表页在线 API

图 5-11 公告管理详情页在线 API

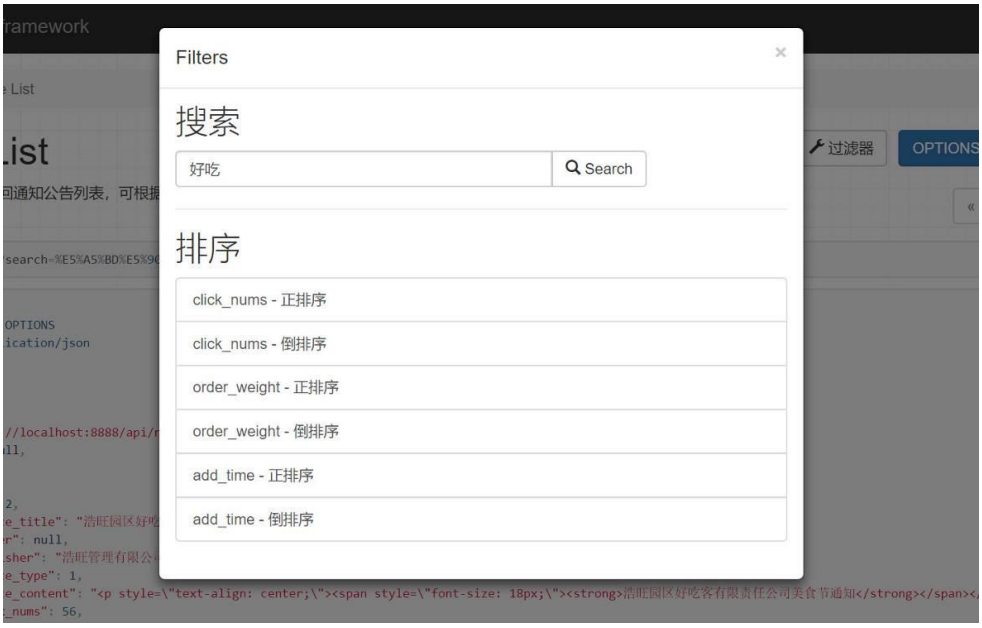

图 5-12 公告管理搜索排序

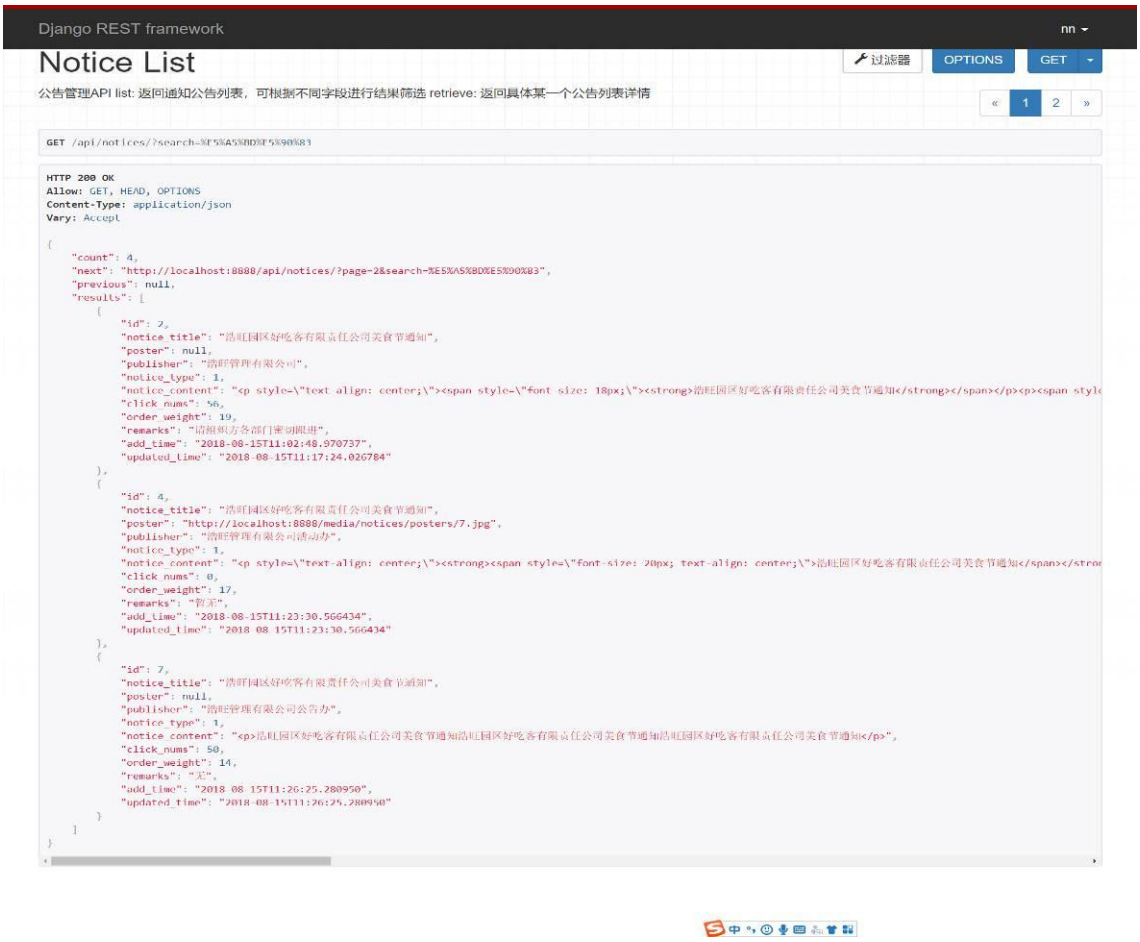

图 5-13 公告管理搜索排序结果

图 5-14 公告管理列表页客户端界面

5.2.3.2 访客管理实现

1. 代码设计思路

具有相应角色权限的保安或者园区管理员可以在 APP 端登记访客信息，RESTful API 层先是根据路由跳转到对应的视图函数，视图函数里将先对用户权限和身份进行验证，通过后中继到序列化器中进行字段有效性验证，然后视图层针对不同 HTTP 动词进行处理。

2. 核心代码

1）模型层主要代码片段：

```
class Visitor(models.Model):
```

```
    “””
    访客管理
    “””
    VISITOR_TYPE = (
        (‘visitor’, “访客”),
        (‘companion’, “随同”)
    )
    visitor_name = models.CharField(max_length=10, verbose_name=”访客姓名”, help_text=”访客姓名”)
    visitor_type = models.CharField(max_length=9, choices=VISITOR_TYPE, default=1, verbose_name=”访客类型”,
                                    help_text=”访客类型”)
    gender = models.CharField(max_length=6, choices=((“male”, “男”), (“female”, “女”)), default=”male”,
                              verbose_name=”性别”, help_text=”性别”)
    visit_place = models.CharField(max_length=200, verbose_name=”到访地点”, help_text=”到访地点”)
    visit_time = models.DateTimeField(auto_now_add=True, verbose_name=”到访时间”)
    leave_time = models.DateTimeField(verbose_name=”离开时间”)
    contacts = models.CharField(max_length=11, null=True, blank=True, verbose_name=”联系方式”, help_text=”联系方式”)
    id_card = models.CharField(max_length=18, null=True, blank=True, verbose_name=”身份证号码”, help_text=”身份证号码”)
    id_card_image = models.ImageField(upload_to=”visitors/id_cards/”, verbose_name=”身份证照片”, null=True, blank=True,
                                      help_text=”身份证照片”)
    reason = models.CharField(max_length=500, verbose_name=”到访原因”, help_text=”到访原因”)

    # 元数据
    class Meta:
        verbose_name = “访客管理”
        verbose_name_plural = verbose_name
        # 数据按到访时间倒序
        ordering = [‘-visit_time’]

    # 重写对象 toString()方法
    def __str__(self):
        return self.visitor_name
```

2）序列化器主要代码片段：

```
class VisitorSerializer(serializers.ModelSerializer):
    def validate_contacts(self, contacts):
        “””
        验证录入的访客手机号码合法性
        “””
        if not re.match(REGEX_MOBILE, contacts):
            raise serializers.ValidationError(“手机号码无效”)

        return contacts
```

```
    # 元数据
    class Meta:
        # 关联模型是 Visitor
        model = Visitor
        # 关联字段是 Notice 中所有字段
        fields = “__all__”
```

3）权限文件主要代码片段：

```
class IsOwnerOrReadOnly(permissions.BasePermission):
    #重写 BasePermission 中的 has_object_permission 方法，以便操作者只能操作自己创建的记录，不能修改他人提交的记录
    def has_object_permission(self, request, view, obj):
        if request.method in permissions.SAFE_METHODS:
            return True

        # Instance must have an attribute named `owner`.
        Return obj.user == request.user
……
```

3. 实现截图

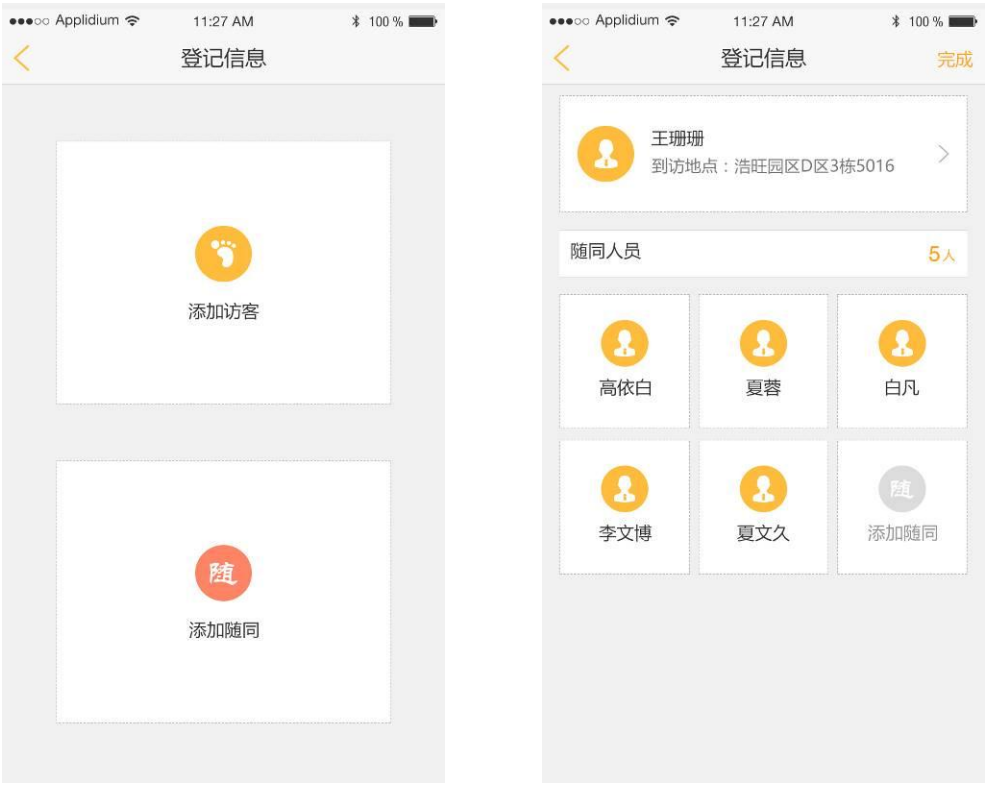

图 5-15 访客管理登记信息界面

图 5-16 访客管理访客详情界面

图 5-17 访客管理添加访客随同界面

5.2.4 孵化服务实现

5.2.4.1 缴费管理实现

1. 代码设计思路

首先，业主在 APP 端查看当月费用详情，选择对应支付方式后，点击在线支付。

此时会请求到 RESTful 服务层缴费管理 URL, 支付对应的是 POST 请求，在缴费管理对应的视图函数里，通过重写 get_serializer_class 方法使得缴费管理视图函数可以动态对应了多个序列化器，POST 请求对应的 action 是”create”，此时返回对应的在线缴费的序列化器对象。

在线缴费序列化器对象中最关键的需要向客户端暴露组装好的支付宝请求 URL。该 URL 根据具体的缴费清单生成。支付宝请求 URL 需包含公共请求参数，有 app_id、method、format、return_url、charset、sign_type、sign、timestamp、version、notify_url、biz_content。其中，biz_content 是承载业务参数的容器，按照蚂蚁金服开放平台文档进行应用公钥、私钥和开放平台公钥和私钥的配置以后，按要求将业务数据排序、编码组装到 biz_content 字典里。最后将生成的 URL 字符串当做请求参数和支付宝支付网关进行拼接生成支付接口 URL。然后 RESTful 服务层将带有缴费流水号、支付接口 URL 等信息的数据以 JSON 格式返回给客户端，客户端请求对应支付接口 URL 即可拉起支付宝支付界面。

支付宝支付界面保持长连接状态，一旦用户用手机扫码成功支付后，支付宝后端将会根据支付状态使用 GET 方式请求之前公共参数里配置的同步回调接口 return_url 或者使用 POST 方式请求异步回调接口 notify_url。服务层将针对两种不同的请求方式在对应视图函数里进行数据验签，以保证数据安全性和完整性，验签通过后会修改缴费项管理里对应字段状态，保存到数据库之中，从而完成整个业务闭环。

2. 核心代码

1）支付宝支付接口配置项主要代码片段：

```
# 支付宝支付接口基础配置项
ZHIFUBAO_PAY_CONFIG = {
    # 是否启用沙箱环境
    'SANDBOX_ON': False,
    'SIGN_TYPE': 'RSA2',
    # 开发者应用私钥存放位置
    'APP_PRIVATE_KEY_URI': os.path.join(BASE_DIR, 'secret_keys/app_private_key.py'),
    # 蚂蚁金服开放平台公钥存放位置
    'ZHIFUBAO_PUBLIC_KEY_URI': os.path.join(BASE_DIR,
'secret_keys/zhifubao_public_key.py'),
```

```
    # 线上支付网关
    'REAL_GATEWAY':'https://openapi.alipay.com/gateway.do',
    #沙箱环境支付网关
    'SANDBOX_GATEWAY':'https://openapi.alipaydev.com/gateway.do'
```

2）支付工具类主要代码片段

```
import json
from datetime import datetime
from base64 import decodebytes, encodebytes
from Crypto.Hash import SHA256
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
from urllib.parse import quote_plus

from HaoWangPark import settings

zhifubao_pay_config = settings.ZHIFUBAO_PAY_CONFIG


class ZhifubaoPay(object):
    """
    支付宝 APP 端支付类,用于对接蚂蚁金服开放平台 SDK,完成支付功能
    """

    def __init__(self, appid, zhifubao_public_key_uri, app_private_key_uri
                 , async_notify_url, sync_return_url):
        """
        初始化参数
        :param appid:开发者应用 id
        :param async_notify_url:异步通知回调 url
        :param zhifubao_public_key_uri:蚂蚁金服开放平台公钥存放位置
        :param app_private_key_uri:开发者应用私钥存放位置
        :param sync_return_url:同步通知回调 url
        """
        self.appid = appid
        self.app_private_key = None
        self.async_notify_url = async_notify_url
        self.sync_return_url = sync_return_url

        # 根据配置文件中沙箱环境开关状态选用对应支付网关
        if zhifubao_pay_config['SANDBOX_ON'] is True:
            self._gateway = zhifubao_pay_config['SANDBOX_GATEWAY']
        else:
            self._gateway = zhifubao_pay_config['REAL_GATEWAY']
        # 设置应用私钥
        self.app_private_key_uri = app_private_key_uri
        with open(self.app_private_key_uri) as f:
            self.app_private_key = RSA.importKey(f.read())
        # 设置支付宝公钥
        self.alipay_public_key_path = zhifubao_public_key_uri
        with open(self.zhifubao_public_key_uri) as f:
            self.alipay_public_key = RSA.import_key(f.read())

    def raw_qs_generator(self, method, biz_content, return_url=None):
```

```
        """
        公共请求参数生成器
        :param method: 接口名称
        :param biz_content:业务请求参数容器对象
        :param return_url:同步回调 url
        :return:原始请求参数字符串
        """
        raw_data = {
            "app_id": self.appid,
            "method": method,
            "charset": "utf-8",
            "sign_type": "RSA2",
            "timestamp": datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
            "version": "1.0",
            "biz_content": biz_content
        }

        if return_url is not None:
            raw_data["notify_url"] = self.app_notify_url
            raw_data["return_url"] = self.return_url

        return raw_data

    def _order_handler(self, raw_data):
        """
        查询字符串排序方法，按 ASCII 码顺序排序
        :param raw_data: 未排序查询字符串
        :return: 排序后查询字符串
        """
        output_keys = []
        for key, value in raw_data.items():
            if isinstance(value, dict):
                output_keys.append(key)

        for key in output_keys:
            raw_data[key] = json.dumps(raw_data[key], separators=(',', ':'))

        return sorted([(k, v) for k, v in raw_data.items()])

    def _sign_handler(self, unsigned_string):
        app_private_key = self.app_private_key
        signer = PKCS1_v1_5.new(app_private_key)
        signature = signer.sign(SHA256.new(unsigned_string))
        sign = encodebytes(signature).decode("utf8").replace("\n", "")
        return sign

    def signature(self, raw_data):
        raw_data.pop("sign", None)
        unsigned_items = self._order_handler(raw_data)
        unsigned_string = "&".join("{0}={1}".format(k, v) for k, v in unsigned_items)
        sign = self._sign_handler(unsigned_string.encode("utf-8"))
        quoted_string = "&".join("{0}={1}".format(k, quote_plus(v)) for k, v in unsigned_items)
        signed_string = quoted_string + "&sign=" + quote_plus(sign)
        return signed_string
```

```
def pay_executor(self, subject, out_trade_no, total_amount, **kwargs):
    """
    支付方法
    :param subject:  订单交易标题
    :param out_trade_no:  商户网站唯一订单号
    :param total_amount:订单总金额
    :param kwargs:其他字段
    :return:签名后支付 url 字符串
    """
    # 业务请求参数容器
    biz_content = {
        "subject": subject,
        "out_trade_no": out_trade_no,
        "total_amount": total_amount,
        "product_code": "FAST_INSTANT_TRADE_PAY",
    }
    biz_content.update(kwargs)
    raw_data = self.raw_qs_generator("alipay.trade.app.pay", biz_content, self.return_url)
    return self.signature(raw_data)

def verify_signature(self, response_data, signature):
    """
    解析请求返回结果并做验签
    :param response_data:  返回结果
    :param signature:  支付宝端返回签名
    :return:验签是否通过
    """
    if "sign_type" in response_data:
        response_data.pop("sign_type")
    unsigned_items = self._order_handler(response_data)
    message = "&".join(u"{}={}".format(k, v) for k, v in unsigned_items)
    key = self.alipay_public_key
    signer = PKCS1_v1_5.new(key)
    digest = SHA256.new()
    digest.update(message.encode("utf8"))
    if signer.verify(digest, decodebytes(signature.encode("utf8"))):
        return True
    return False
```

3. 实现截图

图 5-18 历史缴费清单界面

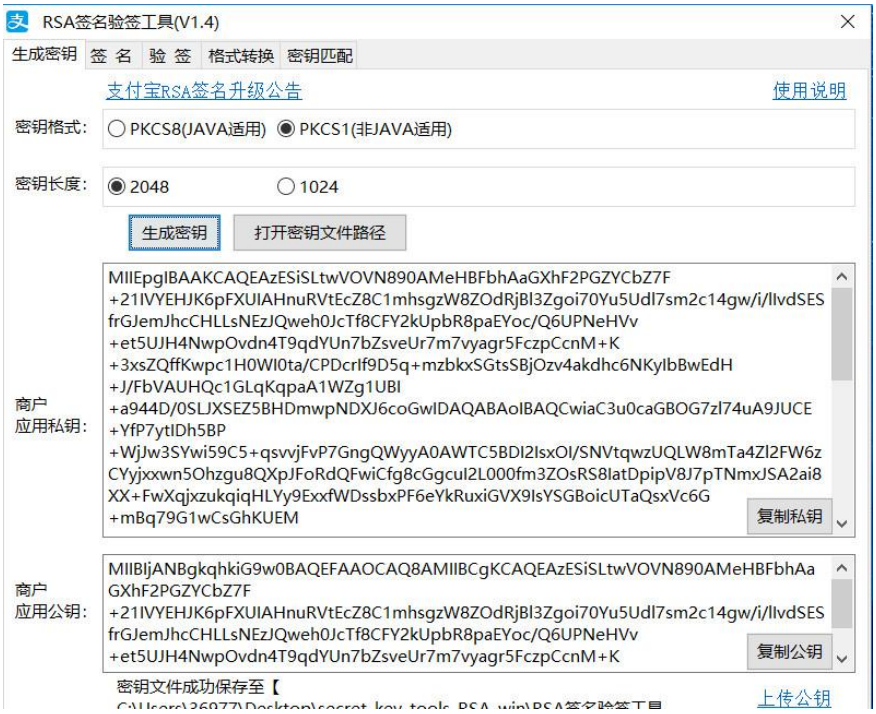

图 5-19 生成 RSA 密钥

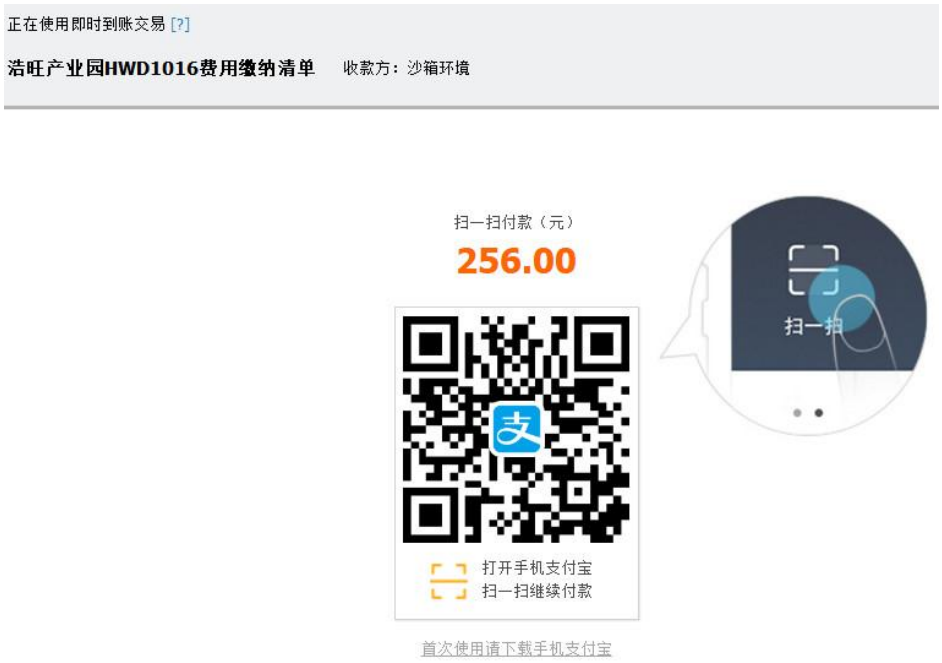

图 5-20 缴费支付界面

5.2.5 其他部分实现

5.2.5.1 跨域策略

因为涉及到分布式系统调用，所以会产生跨域问题，解决方案在后台配置 cors 跨域策略，技术实现基于 Django 的第三方扩展 django-cors-headers，在全局基础配置文件里可以配置跨域策略，系统采用针对特定域名设置白名单的方式。

主要实现片段：

```
INSTALLED_APPS = [
    ......
    'corsheaders', #  扩展注册
     ......
]

MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware', #配置中间件
     ......
]

#  后台跨域白名单配置
CORS_ORIGIN_WHITELIST = (
    'api.haowangpark.xxx',
    'localhost:8888',
    '127.0.0.1:8888'
)
```

5.2.5.2 API 缓存

将常用的不涉及到数据提交和对实时性要求较低的接口进行缓存，缓存策略可以自由配置，缓存数据库采用较为主流的 Redis，实现主要依赖于两个第三方扩展：drf-extensions 和 django-redis。在全局基础配置里配置 Redis 服务端信息和缓存策略，在视图层里注入 CacheResponseMixin 进行缓存。

主要实现片段：

```
CACHES = {
    "default": {
        "BACKEND": "django_redis.cache.RedisCache",
        "LOCATION": "redis://xxx.xxx.xxx.xxx:6379",
        "OPTIONS": {
            "CLIENT_CLASS": "django_redis.client.DefaultClient",
        }
    }
}
```

视图层注入：

```
class NoticeViewSet(CacheResponseMixin, mixins.ListModelMixin, mixins.RetrieveModelMixin,
viewsets.GenericViewSet):
    ......
```

5.2.5.3 API 访问频率限制

Django RESTful framework 里对 API 的访问频率限制提供了很好的支持。从大类来分 API 接口可分为公共接口和私有接口，公共接口允许任何用户请求访问，不需要身份认证，而私有接口需要对特定身份权限的用户开放。

全局配置文件中配置全局策略：

```
REST_FRAMEWORK = {
    ......
    'DEFAULT_THROTTLE_CLASSES': (
        'rest_framework.throttling.AnonRateThrottle',
        'rest_framework.throttling.UserRateThrottle'
    ),
    'DEFAULT_THROTTLE_RATES': {
        'anon': '4/minute',    #公共接口 每分钟限制最大访问次数为 4 次
        'user': '6/minute'     #私有接口 每分钟限制最大访问次数为 6 次
    }
    ......
}
```

也可在具体视图函数里配置针对特定 API 的策略，此处略去不表。

5.3 部署方案

部署方案如下图所示，Nginx 主要承担静态资源代理和负载均衡的作用，

uWSGI 充当 Web 服务器容器，uWSGI 中部署多个 Django 应用实例。系统部署架构如图 5-21 所示。

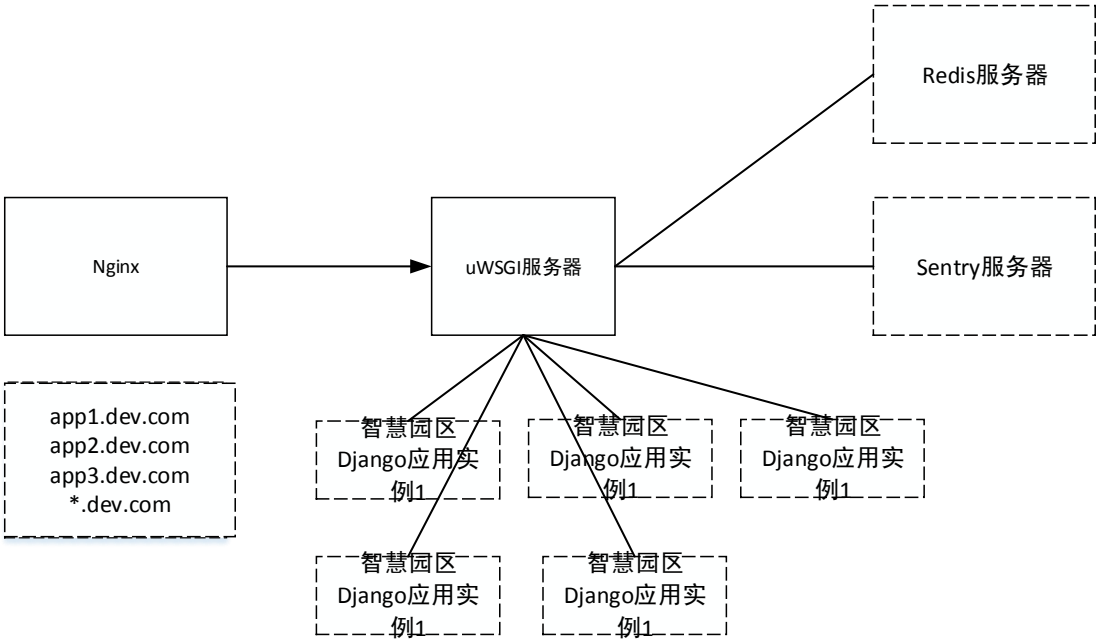

图 5-21 系统部署架构

5.4 本章小结

本章主要针对智慧园区平台系统中接口层和后台管理系统中主要功能模块和业务流程实现进行了说明。首先阐述了项目工程结构整体规划，然后就用户登录、公告管理、访客管理等主要功能的实现进行了详细阐述，列出了主要核心部分代码，最后对系统接口层中运用到的访问控制相关的解决方案进行了阐述，并对系统部署方案进行了说明。

第六章 系统测试

6.1 测试计划

根据智慧园区平台系统验收标准确定系统测试计划，如表 6-1 所示。

表 6-1 测试计划

被测试对象(SUT)	智慧园区平台系统
测试范围	APP 端、系统接口、后台管理系统
测试目的	确保所有功能模块运行正常，高负载下不宕机，持续提供服务
测试环境	服务器：CentOS 7.4 Docker：18.06.1-ce Nginx：1.12.2 uWSGI：2.0.17 Python 版本：Python 3.6.3 Python 虚拟环境：Virtualenv 15.1.0 数据库：MySQL 5.7.18 缓存数据库：Redis 3.2 Sentry：8.9.3

6.2 功能测试

6.2.1 部分界面测试用例

1. APP 首页测试用例

APP 首页测试用例如表 6-2 所示。

表 6-2 APP 首页测试用例

功能模块	用例名称	测试步骤	期望结果	实际结果
APP 首页	首页导航栏展示	1.进入 APP	导航栏数据显示正	通过，与期望结

		2.查看首页导航栏显示	确，界面美观与设计图保持一致	果一致
APP 首页	首页轮播图展示	1.进入 APP 2.查看首页轮播图显示	轮播图显示正常，自动播放与切换正常	通过，与期望结果一致
APP 首页	首页功能菜单展示、编辑	1.进入 APP(分别测试未登录登录状态) 2.查看首页功能菜单显示 3.长按菜单项可以编辑菜单位置、删除菜单	未登录时能挑选出游客可以浏览的功能展示出来(至少 5 个)，登录用户的首页界面，默认时需要显示 10 个功能，第十个为“全部”功能，编辑删除功能呢正常	通过，与期望结果一致
APP 首页	首页列表展示	1.进入 APP 2.查看首页列表显示 3.点击具体列表进入详情	列表页显示正常，点击后能进入详情页	通过，与期望结果一致


Applidium
11:27 AM
100 %
浩旺园区
温馨提醒：业主，你好！由于电路改造，将在今天...
通知公告
办事指南
园区托管
物流快递
企业选址
贴心服务
人员招聘
缴费管理
周边服务
全部
最新活动
往期活动
浩旺园区景德艺术有限公司第一届艺术展
王珊珊
10:26
浩旺园区好吃客有限责任公司美食节
王珊珊
08:52
浩旺园区第二届园区企业帆船活动
王珊珊
2018年6月12号
浩旺园区2018年企业新年年会活动通知
王珊珊
2018年6月10号
查看更多
首页
园区
服务
我的

图 6-1 APP 首页界面

2. 后台登录测试用例

后台登录测试用例如表 6-3 所示，后台登录界面如图 6-2 所示，后台登录错误场景 1 如图 6-3 所示，后台登录错误场景 1 如图 6-4 所示。

表 6-3 后台登录测试用例

功能模块	用例名称	测试步骤	期望结果	实际结果
后台登录	后台登录页展示	1.访问后台系统 url 2.未登录直接跳转到登录页 2.查看后台登录页显示	能正常跳转到登录页，登录页界面美观与设计图保持一致	通过，与期望结果一致
后台登录	登录验证场景 1	1.访问后台系统 url 2.未登录直接跳转到登录页 3.输入已注册用户名，不输入密码 4.点击登录	登录不成功，显示密码项不能为空	通过，与期望结果一致
后台登录	登录验证场景 2	1.访问后台系统 url 2.未登录直接跳转到登录页 3.不输入用户名，输入密码 4.点击登录	登录不成功，显示用户名项不能为空	通过，与期望结果一致
后台登录	登录验证场景 3	1.访问后台系统 url	登录不成功，显示请输入正确用户名和密	通过，与期望结果一致

		2.未登录直接跳转到登录页 3.输入已注册用户名，错误密码 4.点击登录	码	
后台登录	登录验证场景 4	1.访问后台系统 url 2.未登录直接跳转到登录页 3.输入未注册用户名，输入密码 4.点击登录	登录不成功，显示请输入正确用户名和密码	通过，与期望结果一致
后台登录	登录验证场景 5	1.访问后台系统 url 2.未登录直接跳转到登录页 3.输入正确用户名，正确密码 4.点击登录	登录成功，跳转到后台首页，后台首页界面美观与设计图保持一致	通过，与期望结果一致

图 6-2 后台登录界面

图 6-3 后台登录错误场景 1

图 6-4 后台登录错误场景 2

6.2.2 部分功能测试用例

1. 系统功能管理测试用例

1）用户注册

用户注册测试用例见表 6-4。

表 6-4 用户注册测试用例

功能模块	用例名称	测试步骤	期望结果	实际结果
用户注册	使用非法手机号获取验证码	1.进入 APP 我的界面，点击注册按钮 2.填写非法手机号	弹出该手机号非法 toast 提示，注册失败	通过，与期望结果一致

		3.点击获取验证码按钮		
用户注册	使用已注册手机号注册	1.进入 APP 我的界面，点击注册按钮 2.填写已注册过手机号 3.点击获取验证码按钮	弹出该手机号已注册 toast 提示，注册失败	通过，与期望结果一致
用户注册	验证码倒计时超时	1.进入 APP 我的界面，点击注册按钮 2.填写未注册过合法手机号 3.点击获取验证码按钮 4. 填写长度 6-14 位合法密码 5.1 分钟后正确填入短信收到验证码 6.点击注册按钮	弹出验证码超时 toast 提示，注册失败	通过，与期望结果一致
用户注册	密码不符合限制	1.进入 APP 我的界面，点击注册按钮 2.填写未注册过合法手机号	弹出密码不符合验证，请填入 6-14 位数组字母，符号组合，不包括空格 toast 提示，注册失败	通过，与期望结果一致

		3.点击获取验证码按钮 4.填写长度6-14位合法密码 5.倒计时内填入短信收到正确验证码 6.点击注册按钮		
用户注册	正常注册	1.进入APP我的界面，点击注册按钮 2.填写未注册过合法手机号 3.点击获取验证码按钮 4.填写符合限制密码 5.倒计时内填入短信收到正确验证码 6.点击注册按钮	弹出恭喜你，注册成功toast提示，随后我的界面顶部显示用户登录状态信息	通过，与期望结果一致

2）消息推送

消息推送测试用例见表6-5。

表6-5 消息推送测试用例

功能模块	用例名称	测试步骤	期望结果	实际结果

消息推送	特定用户消息推送	1.使用管理员账号登录后台系统 2.点击系统功能菜单下的推送管理链接，进入推送管理主页 3.点击右上角增加推送 4.推送对象填入数字 0，填写推送内容 5.点击保存按钮 6.用任意 2 个手机使用已注册不同账号登录 APP 7.点击我的标签下消息列表 8.查看推送记录	两个账号均收到该条推送，且推送内容正确	通过，与期望结果一致
消息推送	群发消息推送	1.使用管理员账号登录后台系统 2.点击系统功能菜单下的推送管理链接，进入推送管理	A 账号收到推送，且推送内容正确，B 账号没有收到推送	通过，与期望结果一致

		主页 3.点击右上角增加推送 4.推送对象选择 A 用户，填写推送内容 5.点击保存按钮 6.使用 A,B 两个已注册账号登录两台手机的 APP 7.点击我的标签下消息列表 8.查看推送记录		

3）在线缴费

在线缴费测试用例见表 6-6。

表 6-6 在线缴费测试用例

功能模块	用例名称	测试步骤	期望结果	实际结果
在线缴费	立即支付	1.进入 APP 园区界面，点击账单缴费 2.查看当月账单详情 3.点击立即缴费 4.选择支付宝	账单缴费列表显示该月账单状态为已支付	通过，与期望结果一致

		支付 5.完成支付		
在线缴费	下单后支付	1.进入 APP 园区界面，点击账单缴费 2.查看当月账单详情 3.点击立即缴费 4.选择支付宝支付 5.弹出支付界面点击又上角关闭 6.再次进入当月账单详情界面，点击完成付款按钮 7.完成支付	账单缴费列表显示该月账单状态为已支付	通过，与期望结果一致

2. 园区管理测试用例

1）公告发布

公告发布测试用例见表 6-7。

表 6-7 公告发布测试用例

功能模块	用例名称	测试步骤	期望结果	实际结果
公告发布	公告发布	1.使用管理员权限登录后台系统 2.进入园区界面下公告管理	通知发布成功，且按发布时间倒序排列，详情页内容显示正常	通过，与期望结果一致

		界面 3.选择公告为通知，填写必填字段，多录入几条数据 4.进入 APP 端通知公告界面查看，点击详情查看		
公告发布	无发布权限账号发布公告	1.使用没有添加公告权限账号登录后台系统 2.发布公告	增加公告选项为不可用状态，发布失败	通过，与期望结果一致

2）意见反馈

意见反馈测试用例见表 6-8。

表 6-8 意见反馈测试用例

功能模块	用例名称	测试步骤	期望结果	实际结果
意见反馈	意见反馈提交	1.登录，进入 APP 我的界面，点击意见反馈 2.填写表单 3.点击右上角提交按钮	系统后台意见反馈管理板块显示最新提交意见反馈，状态为未读，点击详情内容显示正确	通过，与期望结果一致
意见反馈	意见反馈处理	1.点击进入园区界面下意见反馈管理界面 2.点击一条记	APP 端查看意见反馈列表，状态显示已处理，点击详情可查看处理意见及处理人	通过，与期望结果一致

		录详情，指派处理人，填写处理意见 3.点击保存		

3）活动报名

活动报名测试用例见表 6-9。

表 6-9 活动报名测试用例

功能模块	用例名称	测试步骤	期望结果	实际结果
活动报名	报名成功	1.登录，进入 APP 园区界面，点击企业活动 2.浏览活动详情，选择想要报名活动 3.点击下方我要报名按钮 4.填写报名信息，点击提交	系统后台活动报名理板块下显示最新提交报名信息。修改活动报名状态为审核通过，活动管理界面该活动已报人数加 1， APP 端活动报名里显示已报名活动状态为已成功	通过，与期望结果一致
活动报名	报名失败	1.登录，进入 APP 园区界面，点击企业活动 2.浏览活动详情，选择想要报名活动 3.点击下方我要报名按钮 4.填写报名信	系统后台活动报名理板块下显示最新提交报名信息。修改活动报名状态为审核未通过。 APP 端活动报名里显示已报名活动状态为已成功	通过，与期望结果一致

		息，点击提交		

4）访客登记

访客登记测试用例见表 6-10。

表 6-10 访客登记测试用例

功能模块	用例名称	测试步骤	期望结果	实际结果
访客登记	添加访客信息	1.保安角色登录，进入 APP 园区下访客管理界面，点击右上角登记按钮 2.录入访客、随从信息，点击保存	APP 端访客管理列表显示最新录入信息，详情显示正确	通过，与期望结果一致
访客登记	修改来访信息	1.点击一条访客记录详情， 2.添加离开时间 3.点击保存	查看系统后台访客管理里来访记录，该修改记录正确记录离开时间	通过，与期望结果一致

5）服务管理

服务管理测试用例见表 6-11。

表 6-11 服务管理测试用例

功能模块	用例名称	测试步骤	期望结果	实际结果
服务管理	服务上报	1.登录，进入 APP 园区界面，点击服务管理，点击右上角我要上报	系统后台服务管理板块显示最新服务上报信息，点击详情内容显示正确	通过，与期望结果一致

		2.填写表单 3.点击右上角提交按钮		
服务管理	上报处理	1.点击进入园区下服务管理界面 2.点击一条记录详情，指派处理人，修改处理进度 3.点击保存 4.再次重复2-3，每次处理进度修改为不同状态	APP端查看上报服务详情，显示处理责任人与处理进度	通过，与期望结果一致

3. 孵化服务测试用例

1）招聘信息发布

招聘信息发布测试用例见表6-12。

表6-12 招聘信息发布测试用例

功能模块	用例名称	测试步骤	期望结果	实际结果
招聘信息发布	招聘信息发布	1.登录，进入APP孵化界面，点击招聘服务，点击右上角我要发布 2.填写表单 3.点击右上角提交按钮 4.后台修改审	发布信息正确，详情里有申请职位按钮	通过，与期望结果一致

		核状态为通过 5.APP 端查看发布信息		

2）职位申请

职位申请测试用例见表 6-13。

表 6-13 职位申请测试用例

功能模块	用例名称	测试步骤	期望结果	实际结果
职位申请	职位申请	1.登录，进入 APP 孵化界面，点击招聘服务。 2.浏览招聘信息，选择一个，点击最下面申请职位按钮 3.填写申请表单 4.点击提交按钮	APP 端查看申请进度	通过，与期望结果一致

6.3 接口限频测试

分别针对公共接口和私有接口进行访问频率测试，测试配置策略为公共接口每小时限制最大访问次数为 120 次，私有接口为 160 次。公共接口限频结果如图 6-5 所示，私有接口限频结果如图 6-6 所示。

图 6-5 公共接口限频结果

图 6-6 私有接口限频结果

不同策略下，超过各自接口访问频率，接口层能进行正确阻止访问，并返回 429 Too Many Requests 状态码给予提示。

6.4 性能测试

采用比较流行的测试软件 LoadRunner9.5 版本对系统进行性能测试，使用 Virtual User Generator 模拟 600 个虚拟并发用户，测试场景选用园区管理模块下意见反馈提交流程和活动报名流程，因为这 2 个流程相对独立，无需园区管理方参与即可完成整个流程的业务，所以用来进行性能测试比较容易。测试完成后记录并分析测试结果，比对非功能需求中性能需求验收标准，得出最终结论。

6.4.1 意见反馈提交流程测试

1. 测试方法：

模拟一组虚拟并发用户数，从 40 个并发用户开始，最大到 600 个用户停止，中间以每 40 个用户数递增。

模拟完成用户意见反馈提交流程，记录事务最小响应时间、事务最大响应时间、事务平均响应时间、事务通过率等性能指标，发现系统性能瓶颈，得出系统性能测试结果。

2. 测试结果：

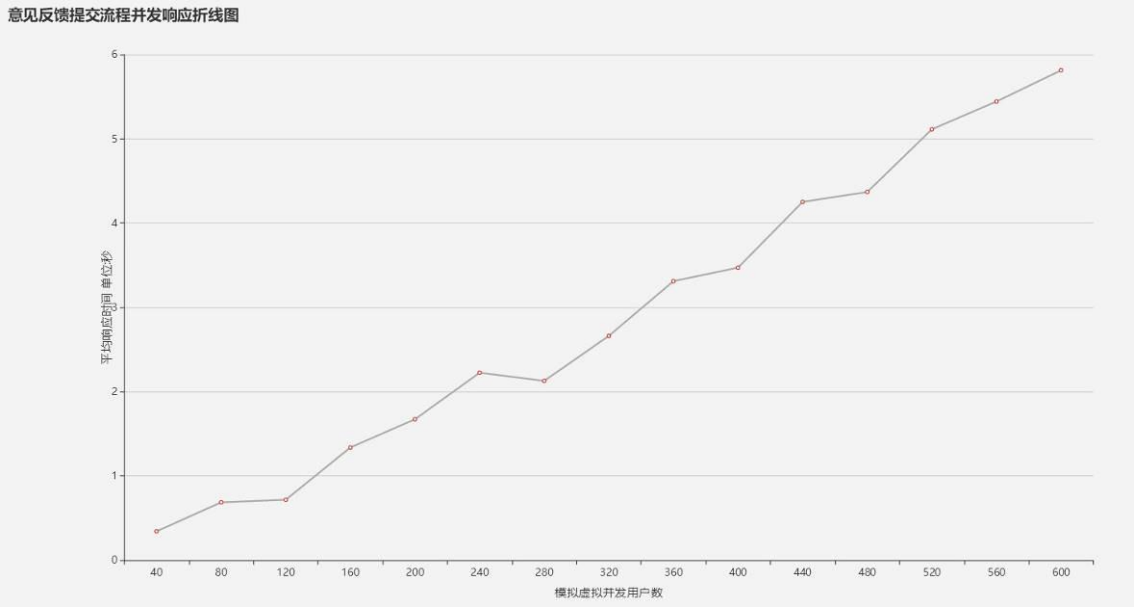

图 6-7 意见反馈提交流程并发响应折线图

表 6-14 意见反馈提交流程并发性能测试结果

测试用例	虚拟用户并发量	事务最小响应时间（单位：秒）	事务最大响应时间（单位：秒）	事务平均响应时间（单位：秒）	事务通过率
意见反馈提交	40	0.221	0.465	0.343	100%
	80	0.380	0.994	0.687	100%
	120	0.662	0.774	0.718	100%
	160	1.212	1.464	1.338	99%
	200	1.456	1.890	1.673	100%
	240	2.101	2.349	2.225	100%
	280	1.986	2.270	2.128	99%
	320	2.211	3.113	2.662	98%

	360	3.121	3.501	3.311	100%
	400	3.021	3.923	3.472	99%
	440	3.988	4.516	4.252	97%
	480	4.121	4.617	4.369	96%
	520	4.991	5.235	5.113	95%
	560	5.013	5.873	5.443	97%
	600	5.725	5.901	5.813	96%

6.4.2 活动报名流程测试

1. 测试流程：

模拟一组虚拟并发用户数，从 40 个并发用户开始，最大到 600 个用户停止，中间以每 40 个用户数递增。

模拟完成用户活动报名流程，记录事务最小响应时间、事务最大响应时间、事务平均响应时间、事务通过率等性能指标，发现系统性能瓶颈，得出系统性能测试结果。

2. 测试结果：

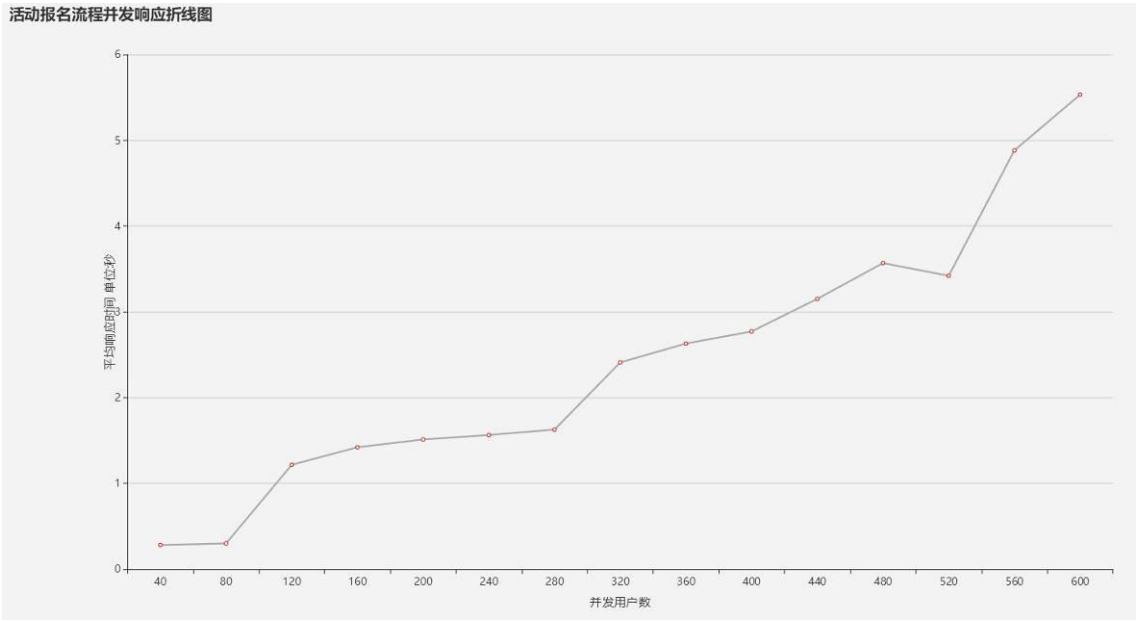

图 6-8 活动报名流程并发响应折线图

表 6-15 活动报名流程并发性能测试结果

测试用例	虚拟用户并发量	事务最小响应时间（单位：秒）	事务最大响应时间（单位：秒）	事务平均响应时间（单位：秒）	事务通过率
活动报名	40	0.198	0.366	0.282	100%
	80	0.258	0.344	0.301	99%
	120	0.998	1.438	1.218	99%
	160	1.112	1.732	1.422	100%
	200	1.323	1.703	1.513	100%
	240	1.402	1.728	1.565	98%
	280	1.511	1.745	1.628	98%
	320	2.212	2.610	2.411	100%
	360	2.358	2.904	2.631	100%
	400	2.668	2.876	2.772	99%
	440	2.987	3.317	3.152	98%
	480	3.325	3.813	3.569	98%
	520	3.411	3.431	3.421	95%
	560	4.252	5.514	4.883	94%
	600	5.191	5.873	5.532	95%

6.4.3 性能测试结果分析

从测试结果可以看出，无论是在对意见反馈提交流程还是测试活动报名流程的测试结果中，虚拟用户数按规律递增，对服务器发起请求数和服务器处理事务数也随之增加，事务最小响应时间、最大响应时间和平均响应时间随着用户数增加总体呈现增大趋势，但总体来说未超过非功能需求中要求的 6 秒阈值，衡量性能标准的另一个指标便是事务通过率。

从测试结果可以看到，事务通过率整体围绕 95%上下波动，整体未低于 93%的阈值，系统整体性能指标满足预期要求。

6.5 验收结果

表 6-16 验收结果表

序号	测试验收结果
HWZHYQ-20180707001	所有功能完成情况与合同约定相符
HWZHYQ-20180707002	所有额外约定及要求按照使用单位意见执行
HWZHYQ-20180707003	全部设备正常安装，施工符合相关要求，线路安装符合标准
HWZHYQ-20180707004	项目涉及到系统后台，app 完成相关业务测试，试运行，均通过
HWZHYQ-20180707005	该节点完工后形成验收文件经用户签字确认
HWZHYQ-20180707006	目前所有设备/系统运行正常
HWZHYQ-20180707007	项目建设达到设计要求，相关功能齐备
HWZHYQ-20180707008	目前系统运行正常同意通过验收

6.6 本章小结

本章主要就系统测试部分进行了说明，首先对系统整体测试计划进行了阐述，然后分别就功能测试和性能测试用例进行了设计，并对结果进行了分析，最后阐述了接口限频测试和系统整体验收结果。

第七章 全文总结与展望

7.1 总结

本文主要围绕项目一期工程中客户对园区管理和孵化服务功能的信息化要求进行详细阐述。解决方案采用基于 Django 的第三方框架的 Django REST framework 进行前后端分离开发。为后续系统在多客户端上上线提供了基础。基于 Django 的解决方案具有高内聚，低耦合的特性，也使得系统后续增加新功能进行迭代更为灵活、方便。

本文中承担的主要工作如下：

1. 研究 Django REST framework 框架中各种功能组件及 API, 并结合业务需求选取适合的部分。

2. 复用 Django 框架中例如认证鉴权、缓存等模块，基于业务进行二次定制改写。

3. 研究后台管理系统的架构与设计，研究 Django 框架的 MTV 模式，包括 Models 层的设计， Views 层设计以及基于 URLconf 的路由设计及规划。

4. 研究如何高效结合 Django 和 Django REST framework 框架，设计出分层合理，符合 REST 接口规范的项目结构。

5. 研究 API 的限频、跨域策略，研究 Redis 数据结构，对实时性要求不高的 API 进行采用 Redis 缓存。

6. 针对耗时任务通过引入 Celery 框架进行异步任务分发，以提供更好的并发处理能力及更好的用户体验。

7. 对智慧园区平台系统进行细致的需求分析，划分出主要模块部分，分别基于园区管理、孵化服务、系统功能管理模块进行数据库模型层设计，E-R 关系设计，抽象出各部分主要业务流程。

8. 编码和测试。

客户对目前服务的归类和基础服务的交付模式比较满意，有了平台以后，园区认为既提升了企业形象，也提供了现代化的服务手段。已完成建设的功能得到客户充分认可，对于公司在积极探索实践基于产业背景的生态型孵化器建设方面踏出了坚实的一步，同时客户表示希望在后续服务运营及整合方面继续深化交流。

7.2 展望

智慧园区系统是一个庞大复杂的项目，现阶段其实只实现了其中很小的一部

分，对于产业园区智慧化管理中需要涉及到的例如园区智慧泊车、园区一卡通、电子围栏等功能均未涉及。下一步将针对现有系统中存在的缺陷进行功能修改，迭代，重构，同时着力解决以下几个问题：项目相关运营责任落实、系统外部链接完成、运营化部署落实、iOS 客户端上线、运营问题支撑。后期随着对 IT 支撑投入力度的加大，将会建成视频监控 3D 综合管理系统、巡更系统、周界防护系统等。

项目下一步工作计划主要是实现视频监控 3D 综合管理模块。计划功能如表 7-1 所示。

表 7-1 视频监控 3D 综合管理模块功能描述

子模块	主要功能	功能描述
园区 3D 模型管理	模型绘制	模型 3D 建模
	模型贴图	3D 模型纹理，贴图，光影
	模型变更	模型后续根据具体情况的调整
	点位管理	监控点位的添加
	基本 GIS 信息	模型的基本 gis 信息关联
监控信息输出	列表信息	监控信息列表
	视频展示信息	视频播放，放大等功能
	历史回放	视频历史回放
告警信息管理	告警信息展示	告警信息在 3D 模型中的展示
	告警事件触发	告警事件可关联其他联动系统的接口
	告警轮播接口	提供告警的轮播及查询功能
app 对接接口	对接鉴权	与 app 对接的认证管理
	app 场景展示对接	app 场景展示对接
	控制管理	视频播放及回放的 app 管理

致　谢

值此论文完成之际，千言万语汇于心头。回首三年研究生生涯，有辛勤的汗水，拼搏的付出，但更多的是学有所获，获有所得的幸福与快乐。

论文能够最终的顺利完成，我最要感谢的是我的指导老师佘堃老师，佘堃老师学识渊博，在信息安全、云计算和大数据等方面都有很深的造诣，从我论文确定研究方向到开题，再到中期最后到整个论文成文，佘堃老师都给予了我细致耐心的指导与悉心的帮助，对于论文中不足与描述不清之处，佘堃老师都第一时间给我指出并给予了详尽的修改意见。能得到佘堃老师在软件工程和解决方案方面的意见与指导，使我感觉到研究生生涯中的所有努力与付出都是值得的，也使我在软件工程与技术能力方面开阔了视野，得到了很大的提高。

我还要感谢我的企业指导老师王祺老师，他在实际工作中的丰富经验给予了我很大的帮助。

我还要感谢专家评审组的各位老师，你们不辞辛劳的付出，给予我们最中肯的意见，都是为了我们能在软件工程领域方向走的更远、走的更坚定。

其次，我要感谢在攻读研究生阶段教授我的所有老师与研究生院的老师，正是因为你们，使得我在三年时间里，在职业规划方面有了明确的方向与目标，在职业素养，解决问题能力方面有了很大的提高，并让我知道了要想在软件工程这条路上走的远，对于计算机基础理论原理的掌握程度是多么的重要。

同时，我要感谢 2015 级软件工程专业所有的同学，不仅是我的同班同学，还有计算机学院和航空航天学院等其他学院的同学，正是由于你们对我在学习上、论文上的帮助，才使得我能顺利完成研究生阶段的学习。

最后，我要感谢一直陪伴在我左右的家人，正是因为你们的包容与理解，正是因为你们无微不至的关怀与鼓励，我才能最终坚持顺利完成学业！

由于我的软件工程知识和技术能力有限，论文陈述观点与解决方案难免有不足之处，恳请阅读此文的各位老师专家给予指正！

参考文献

[1] 徐大成. 基于物联网和云计算的智慧园区信息系统的研究与实现[D].西安电子科技大学,2015.

[2] 张春光,李树泉,蔡永涛,吴文炤.面向智慧园区的云计算平台设计[J].供用电,2015,32(12):21-25.

[3] 张凯书,张怡,严杰. 智慧园区信息化建设解决方案[J]. 信息通信,2012,(06):118-119.

[4] 谢俊. 信息化视角下的智慧园区运营管理系统构建和模式研究[D].华东理工大学,2016.

[5] 陈冲,吴越.基于能源互联和“互联网+”理念的智慧园区 2.0 的研究[J].电力信息与通信技术,2016,14(04):22-26.

[6] 庞元乐. 智慧城市应用之智慧园区[A]. 云南省科学技术协会、红河州人民政府、中国通信学会.第六届云南省科协学术年会暨红河流域发展论坛论文集——专题二：滇南中心智慧城市建设[C].云南省科学技术协会、红河州人民政府、中国通信学会:,2016:4.

[7] 朱敏,杨会华.智慧园区解决方案探讨及建议[J].移动通信,2013,37(05):56-58.

[8] 刘鸿雁.建设智慧园区　助推传统园区升级[J].经济研究导刊,2016(04):113-114.

[9] 曹茂春,洪劲飞.智慧园区建设探讨[J].智能建筑,2014(09):34-39.

[10] 陈燕.三维地理信息融入智慧园区建设的应用研究[J].测绘通报,2015(S1):192-195.

[11] 马俊.智慧园区解决方案[J].中国新通信,2015,17(21):61-62.

[12] 赵东辉. 基于物联网的智慧园区信息平台的设计与实现[D].河北科技大学,2018.

[13] 董伟明. Python Web 开发实战[M]. 北京:电子工业出版社, 2016.

[14] 刘长龙. Python 高效开发实战[M]. 北京:电子工业出版社, 2016.

[15] 林嘉婷.试谈前后端分离及基于前端 MVC 框架的开发[J].电脑编程技巧与维护,2016(23):5-8.

[16] 叶泉. 基于前后端分离的虚拟现实实验教学平台的设计与实现[D].武汉大学,2018.

[17] 温馨. 基于 Node.js 的 Web 前端框架的研究与实现[D].东南大学,2017.

[18] 戈家龙,吴红亚,杨保华.基于 SSM 的前后端分离电商网站的设计与实践[J].电脑知识与技术,2018,14(13):276-277.

[19] 杜艳美,黄晓芳.面向企业级 web 应用的前后端分离开发模式及实践[J].西南科技大学学报,2018,33(02):83-87.

[20] 赵启升,李存华,吴庚午,刘小明.一种基于 REST 架构的高校移动教务数据开放平台设计方法[J].江苏科技大学学报(自然科学版),2014,28(05):500-504.

[21] 刘兴邦. 基于 RESTful 和 Android 的途家网房源管理系统的设计与实现[D].北京交通大

学,2015.

[22] 冯伟. 基于 REST 风格的 Android 系统 Web 服务的研究[J].淮北职业技术学院学报. 2012(03).

[23] 于心愿. 基于 Django 框架的存储资源管理系统的设计与实现[D].南京大学,2013.

[24] 朱道雨. 基于 Django 的旅游管理信息系统的建设[D].中国地质大学(北京), 2011.

[25] 郑成刚. 基于 Django 的日程协作系统的设计与实现[D].大连理工大学,2014.

[26] Marty Alchin. Pro Django [M]. Apress, 2008.

[27] Amber Spackman Jones,Jeffery S. Horsburgh,Douglas Jackson-Smith,Maurier Ram ír̃ez,Courtney G. Flint,Juan Caraballo. A web-based, interactive visualization tool for social environmental survey data[J]. Environmental Modelling and Software,2016,84.

[28] James Bennett. Practical Django Projects[M].Apress:2008-06-15.

[29] A Django-Powered Weblog[M].Apress:2008-06-15.

[30] Aziz Khan,Anthony Mathelier. JASPAR RESTful API: accessing JASPAR data from any programming language[J]. Bioinformatics,2018,34(9).

[31] Kartika Firdausy,Samadri,Anton Yudhana. Web based Library Information System Using PHP and MYSQL[J]. TELKOMNIKA,2008,06(2).

[32] Michael Kruckenberg,Jay Pipes. Pro MySQL[M].Apress:2005-06-15.

[33] Wenrong Jiang,Shiwei Lin. Software Engineering and Knowledge Engineering: Theory and Practice[M].Springer Berlin Heidelberg:2012-01-15.

[34] Xiaobo Gao,Xianmei Fang. High-Performance Distributed Cache Architecture Based on Redis[M].Springer Berlin Heidelberg:2014-06-15.

[35] Bo Yu,Limin Jia,Guoqiang Cai,Honghui Dong. The Implementation of the HTTP-Based Network Storage Queue Service[M].Springer Berlin Heidelberg:2013-06-15.