**毕业论文（设计）正文**

**题目：“口腔导航”移动应用开发**

# **摘 要**

口腔健康在现代生活中扮演着至关重要的角色，对个体的整体健康和生活质量有着深远的影响。现有市面上存在大量牙科诊所平台并且能够为用户提供较为丰富的服务。然而尽管这些平台在推动口腔医疗服务的数字化转型方面取得了一定成就，但在满足患者个性化需求、用户使用便捷性、用户交互体验等方面仍有改进空间。因此提出创建自己的牙科诊所平台。

平台主要为管理员，诊所用户，医生以及普通用户提供服务，其中管理员用户能够管理平台上一切资源，审批普通用户向管理员提出的建立诊所的申请。诊所用户能够管理自己所在诊所的资源，包括诊所的位置、提供的服务，添加医生，为医生安排排班等。医生用户通过向诊所申请得到医生权限，并管理自身的信息，包括姓名、年龄以及学历，并能在线填写病案，查看患者过往就诊信息等。普通用户则能够在平台上查询服务信息获取服务，查看自己的就诊记录，在线进行就诊预约，并对获得的服务进行评价等。

整个软件分为四个部分：Android系统前端、Windows系统前端以及运行在服务器的后端和数据库。Android前端和Windows前端分别运行在微信小程序以及浏览器上，用于读取用户端输入的信息，并向后端发送服务请求。后端运行在服务器端，用于接受并处理前端发出的服务请求，并与数据库进行交流，更新数据库中的内容。数据库运行在服务器端，用于存储并与后端交流系统中的数据，为后端提供需要处理的数据。

Android前端为普通用户提供服务，分为四个部分，分别为注册登录模块，预约模块，评价社交模块以及管理模块，四个模块相互交流共同构成了Android前端这一本项目的主体。同时，运行Windows的Windows前端分别为医生用户、诊所用户以及管理员用户提供服务。

**关键词：**微信小程序，vuejs，django，sqlite，牙科诊所平台

# **Abstract**

Oral health plays a crucial role in modern life and has a profound impact on an individual's overall health and quality of life. There are numerous dental clinic platforms available on the market that can provide users with a wealth of services. However, despite these platforms having achieved certain successes in promoting the digital transformation of oral medical services, there is still room for improvement in meeting patients' personalized needs, user convenience, and user interaction experience. Therefore, the proposal to create our own dental clinic platform is introduced.

The platform mainly serves administrators, clinic users, doctors, and ordinary users. Among them, administrator users can manage all resources on the platform and approve the applications from ordinary users to establish clinics. Clinic users can manage the resources of their own clinics, including the location, services provided, adding doctors, and scheduling for doctors. Doctor users obtain doctor privileges by applying to the clinic and manage their own information, including name, age, and education background. They can also fill out medical records online and view patients' past medical records. Ordinary users can query service information, obtain services, view their own medical records, make online appointments, and evaluate the services received on the platform.

The entire software is divided into four parts: the Android system front-end, the Windows system front-end, the server-side backend, and the database. The Android front-end and Windows front-end run on WeChat mini-programs and browsers, respectively, to read user input information and send service requests to the backend. The backend runs on the server side and is responsible for accepting and processing service requests from the frontend, as well as communicating with the database to update the content. The database runs on the server side and is used to store and communicate with the backend, providing the necessary data for processing.

The Android front-end serves ordinary users and is divided into four parts: the registration and login module, the appointment module, the evaluation and social interaction module, and the management module. These four modules interact and together form the main body of the Android front-end in this project. At the same time, the Windows front-end running on Windows provides services for doctor users, clinic users, and administrator users.

Keywords: WeChat Mini Program, Vue.js, Django, SQLite, Dental Clinic Platform

# **图目录**

[图 2.1 找到HBuilder中的运行设置 7](#_Toc19569)

[图 2.2 填入包路径 7](#_Toc31039)

[图 2.3 打开安全设置 8](#_Toc10642)

[图 2.4 打开服务端口 8](#_Toc18087)

[图 2.5 运行到微信开发者工具 8](#_Toc4184)

[图 2.6 运行成功截图 9](#_Toc22430)

[图 2.7 uni.request示例 9](#_Toc14906)

[图 2.8 uni.navigateTo示例 10](#_Toc20607)

[图 2.9 uni.redirectTo示例 11](#_Toc1685)

[图 2.10 uni.navigateBack示例 11](#_Toc25773)

[图 2.11 background类属性 12](#_Toc26110)

[图 2.12 screenHeightRpx的计算 12](#_Toc15492)

[图 2.13 path函数的应用 13](#_Toc4055)

[图 2.14 path函数的应用 13](#_Toc30608)

[图 2.15 视图函数对request的处理 14](#_Toc13856)

[图 2.16 返回JsonResponse 14](#_Toc7933)

[图 2.17 返回HttpResponse 15](#_Toc11152)

[图 2.18 数据表的注册 15](#_Toc7247)

[图 2.19 AccountTable的注册 16](#_Toc12260)

[图 2.20 AccountTable在admin站点的显示 16](#_Toc2132)

[图 2.21 sign\_in函数 17](#_Toc22136)

[图 2.22 get\_code函数 17](#_Toc26198)

[图 2.22 read函数 18](#_Toc15124)

[图 2.23 get函数 18](#_Toc8084)

[图 2.24 render函数使用示例 20](#_Toc30984)

[图 3.1 ER图 24](#_Toc21236)

[图 4.1 登录界面 30](#_Toc6267)

[图 4.2 send\_email函数展示 31](#_Toc19698)

[图 4.3 注册界面展示 31](#_Toc26112)

[图 4.4 个人首页 32](#_Toc15324)

[图 4.5 预约首页 32](#_Toc26730)

[图 4.6 诊所展示页面展示 33](#_Toc29761)

[图 4.7 诊所展示页面展示 34](#_Toc26850)

[图 4.8 预约展示页面展示 35](#_Toc22951)

[图 4.9 消息页面展示 35](#_Toc16694)

[图 4.10 个人预约页面展示 36](#_Toc16252)

[图 4.11 个人评价页面展示 37](#_Toc25716)

[图 4.12 评价页面展示 37](#_Toc22095)

[图 4.13 评价展示页面 38](#_Toc21944)

[图 4.14 评论区展示页面 39](#_Toc31040)

[图 4.15 个人主页 40](#_Toc322)

[图 4.16 收藏页面 41](#_Toc28597)

[图 4.17 管理主页 42](#_Toc11211)

[图 4.18 地点管理页面 43](#_Toc30543)

[图 4.19 账号管理页面 43](#_Toc27673)

[图 4.20 医生界面展示 44](#_Toc10781)

[图 4.21 诊所展示界面 44](#_Toc25298)

[图 4.22 诊所展示界面 45](#_Toc5391)

[图 4.23 诊所界面展示 45](#_Toc9358)

[图 4.24 诊所界面展示 46](#_Toc4781)

[图 4.25 消息处理界面 46](#_Toc2607)

[图 4.26 消息处理界面 47](#_Toc15880)

[图 4.27 评论查看界面 47](#_Toc23413)

[图 4.28 管理员界面展示 48](#_Toc4593)

[图 4.29 医生管理界面 48](#_Toc1997)

[图 4.30 评论管理界面 49](#_Toc2154)

[图 4.31 账户管理界面 49](#_Toc24938)

[图 4.32 地区管理界面 50](#_Toc20058)

[图 4.33 诊所管理界面 50](#_Toc3735)

# **表目录**

[表2.1 应用开发硬件环境 5](#_Toc20494)

[表2.2 移动端开发软件环境 5](#_Toc307)

[表2.3 web端开发软件环境 5](#_Toc19163)

[表2.4 后端开发软件环境 5](#_Toc1588)

[表 3.1 个人信息模块 22](#_Toc21276)

[表 3.2 管理模块 22](#_Toc22157)

[表 3.3 预约模块 22](#_Toc13767)

[表 3.4 评价模块 23](#_Toc31619)

[表 3.5 账号表 24](#_Toc16361)

[表 3.6 城市表 25](#_Toc28972)

[表 3.7 县表 25](#_Toc18923)

[表 3.8 验证码表 25](#_Toc11220)

[表 3.9 诊所表 25](#_Toc2068)

[表 3.10 医生表 25](#_Toc3231)

[表 3.11 服务表 26](#_Toc17488)

[表 3.12 诊所服务表 26](#_Toc2416)

[表 3.13 医生服务表 26](#_Toc25904)

[表 3.14 预约表 26](#_Toc3082)

[表 3.15 消息表 27](#_Toc26161)

[表 3.16 评价表 27](#_Toc19573)

[表 3.17 点赞表 27](#_Toc9537)

[表 3.18 收藏表 28](#_Toc10916)

[表 3.19 收藏表 28](#_Toc10623)

[表 3.20 回复点赞表 28](#_Toc19284)

[表 3.21 后台账户表 28](#_Toc18243)

[表 3.22 后台Cookie表 28](#_Toc31799)

[表 3.23 后台台医生申请表 29](#_Toc17758)

[表 3.24 诊所医生申请表 29](#_Toc9042)

# **目 录**

[摘 要 I](#_Toc10836)

[Abstract II](#_Toc12926)

[图目录 IV](#_Toc7168)

[表目录 VII](#_Toc8230)

[目 录 IV](#_Toc15780)

[第1章 概述 1](#_Toc25478)

[1.1 课题背景 1](#_Toc16958)

[1.2 课题意义 1](#_Toc4786)

[1.3 口腔医疗平台现状 2](#_Toc622)

[1.4 论文基本内容与结构 3](#_Toc23328)

[1.4.1 研发内容 3](#_Toc14283)

[1.4.2 论文结构 4](#_Toc4661)

[第2章 课题相关技术 5](#_Toc932)

[2.1 运行平台 微信小程序介绍 5](#_Toc30592)

[2.2 前端框架 uni-app 介绍 6](#_Toc9441)

[2.2.1 将uni-app运行到微信小程序 7](#_Toc7572)

[2.2.2 uni-app API介绍 9](#_Toc7654)

[2.3 服务端框架Django介绍 12](#_Toc22236)

[2.3.1 Django接收请求 13](#_Toc21162)

[2.3.2 ORM系统 15](#_Toc11741)

[2.3.3 模板系统 19](#_Toc10333)

[第3章 系统设计 21](#_Toc27844)

[3.1 系统整体架构 21](#_Toc15179)

[3.2 后端接口 21](#_Toc5685)

[3.2.1 接口规范和管理 21](#_Toc9447)

[3.2.2 用户端接口罗列 21](#_Toc434)

[3.3 数据库设计 24](#_Toc23329)

[3.3.1 数据库架构 24](#_Toc8210)

[3.3.2 数据库设计 24](#_Toc15220)

[第4章 功能展示与实现 30](#_Toc20706)

[4.1 用户界面 30](#_Toc10058)

[4.1.1 登录注册模块 30](#_Toc14250)

[4.1.2 预约模块 32](#_Toc1200)

[4.1.3 评价社交模块 36](#_Toc14435)

[4.1.4 管理模板 41](#_Toc19816)

[4.2 医生界面 43](#_Toc15531)

[4.2.1 诊所展示 44](#_Toc26594)

[4.2.2 挂号信息 44](#_Toc12143)

[4.3 诊所界面 45](#_Toc8353)

[4.3.1 诊所医师 45](#_Toc19244)

[4.3.2 消息处理 46](#_Toc4115)

[4.3.3 出诊设置 47](#_Toc30456)

[4.3.4 评论查看 47](#_Toc22507)

[4.4 管理员界面 47](#_Toc4323)

[4.4.1 医生管理 48](#_Toc15476)

[4.4.2 评论管理 49](#_Toc14467)

[4.4.3 账户管理 49](#_Toc390)

[4.4.4 地区管理 50](#_Toc665)

[4.4.5 诊所管理 50](#_Toc28666)

[第5章 总结与展望 51](#_Toc1849)

[5.1 总结 51](#_Toc20425)

[5.2 展望 51](#_Toc1500)

[参考文献 52](#_Toc17483)

[致 谢 54](#_Toc30523)

# 概述

## 1.1 课题背景

近年来，随着经济的发展和人们生活水平的提高，口腔健康逐渐成为社会关注的重点。良好的口腔健康不仅关乎个人形象，还与全身健康密切相关。研究表明，口腔疾病与多种全身性疾病（如心血管疾病、糖尿病等）存在关联。因此，越来越多的人开始重视口腔护理，定期进行口腔检查和治疗。

随着口腔健康意识的提高，公众对口腔医疗服务的需求迅速增长。口腔医疗服务是指针对口腔健康问题提供的预防、诊断、治疗和康复等一系列医疗服务。根据艾瑞咨询的数据显示，2023年中国口腔医疗服务的市场规模已经接近2000亿人民币，预计未来将持续扩张，有望在2026年突破3000亿，年复合增长率超过15%。同时，我国口腔诊疗人次逐年增加，特别是牙齿美容、正畸、种植牙等高端口腔服务需求更是呈现出爆发式增长。老龄化社会的到来也进一步推动了口腔护理需求的增长，老年人口的口腔健康问题日益严重，亟需专业的口腔护理服务。

然而，口腔医疗服务需求的持续增加，暴露出了我国口腔医疗资源的种种问题。首先我国口腔医生数量严重短缺。世界卫生组织认为，每万人口应该有不低于3-4名口腔医生，但是我国目前口腔医生密度不到1人每万人口。更重要的是，我国口腔医疗资源分布严重不均。从人均口腔医院数量来看，东部地区为0.0009家/万人，而中西部地区分别为0.0006家/万人和0.0005家/万人。从人均口腔执业医师数量来看，东部地区为0.23人/万人，而中西部地区分别为0.15人/万人和0.13人/万人。资源总量的紧张，加上地区间资源的不平衡与信息的不对称，严重制约我国口腔医疗服务健康发展。

## 1.2 课题意义

为了缓解上述问题，本文开发了一个“口腔导航”移动应用，将各种公立和私立口腔医院资源进行线上整合，结合图像处理和定位技术，构建起患者与医生以及诊所之间的桥梁，使不同区域的患者能够共享医疗资源，提高医疗服务的可及性和便捷性。本项目的具体意义可以概括为以下三点：

提高口腔医疗资源利用效率。本项目通过整合公立和私立口腔医院的资源，构建一个在线口腔导航移动应用，能够有效提高口腔医疗资源的利用效率。在传统的口腔医疗服务模式下，资源分散、信息不对称等问题导致了资源浪费和患者就诊不便。通过应用图像处理和定位技术，患者可以更方便地找到合适的医疗资源，合理安排就诊时间，减少候诊时间，提高医疗资源的利用率，从而优化口腔医疗服务的整体效率。

缓解不同地区医疗资源分布不均。我国口腔医疗资源分布不均，尤其是城乡之间存在较大差距。城市地区医疗资源丰富，设备先进，专业医生数量多，而农村和偏远地区则存在医疗资源匮乏的问题。本项目通过信息化手段，将各地的口腔医疗资源进行整合并在线展示，使偏远地区的患者能够通过移动应用获取更为便捷的医疗服务信息，享受到与城市居民同等的医疗资源，从而在一定程度上缓解城乡医疗资源分布不均的问题。

提升患者就医体验。随着人们生活水平的提高，对医疗服务的需求不仅仅局限于治疗本身，更加关注就医体验。本项目通过开发口腔导航移动应用，为患者提供一站式的口腔健康管理服务，包括预约挂号、在线咨询、导航导诊等功能，使患者能够更加便捷地获取医疗服务。患者可以通过手机轻松完成预约、导航到达诊所，并进行在线咨询和健康管理，极大提升了患者的就医体验，增加了医疗服务的满意度。

## 1.3 口腔医疗平台现状

近年来，随着数字化技术的迅猛发展，口腔医疗服务领域也迎来了新的变革。一系列数字化平台被纷纷推出，旨在提升口腔医疗服务的便捷性和效率。这些平台通过整合在线咨询、预约挂号、健康管理等功能，打破了传统医疗服务的时间和空间限制。然而，这些平台的效果尚需进一步验证，且存在许多提升空间。以下将介绍几款具有代表性的口腔医疗服务应用，展示国内外在这一领域的研究和应用成果。

好牙医在线平台[1]。好牙医在线使用“产业互联网+数字化口腔技术服务”,面向全国诊所与医生，提供数字化口腔的全面支撑性服务。服务覆盖全国30多个省市约40000家诊所，是目前国内最大的口腔数字化设计中心。但是该app并没有面向患者，患者只能被动的接受而无法主动提出需求。

牙e家app[2]。牙e家是以口腔医生为核心的学习、交流、分享平台。平台内有涵盖各个口腔学科的课程视频，专业牙医的公开讲座、聚集广大牙医的学术社区、实时的咨询分享和病例讨论等功能。但牙e家平台的使用具有一定专业知识门槛，一般患者很难从中获得有用的信息。

微笑口腔健康app。微笑口腔健康联合各大医院科室，为患者整理资料、联系医生确认病情，为符合手术要求的患者快速预约全国知名专家手术。让患者免挂号、免门诊、直接入院，避免反复奔波、长时间院外待床，减少不必要的花费。虽然微笑口腔健康平台在患者个性化服务上有所尝试，但在数据隐私保护和用户互动性上还有待改进[3]。

DentaLink牙医管家[4]。DentaLink是国内口腔医疗行业上时间最早、服务最久及活跃度最高的SaaS软件，致力于为口腔机构提供更先进的数字化管理技术，为牙医提供更安全的数字化诊疗方案。它为患者提供了口腔全生命周期健康管理，同时为门诊提供了智能化的运营管理。不足的是，DentaLink需要付费，同时也不够便捷与轻量。

Dental Monitoring[5]。Dental Monitoring是一家法国牙科及正畸领域人工智能解决方案提供商，创建了一个高度精密的人工智能平台，允许牙科专业人员在诊所或远程评估和监测包括正畸在内的牙科病况，旨在通过运用精密评估工具来提高治疗水平，通过简化沟通来增进患者参与度，并同时提高患者便利和执业效率。与好牙医平台的定位类似，该平台并不能解决患者在初期根据自身情况选择合适就医的需求。

总体而言，尽管这些平台在推动口腔医疗服务的数字化转型方面取得了一定成就，但在满足患者个性化需求、用户使用便捷性、用户交互体验等方面仍有改进空间。

## 1.4 论文基本内容与结构

### 1.4.1 研发内容

1. 开发一个能在多平台运行的牙科诊所软件，为患者和牙科诊所之间建立一个有效的沟通和交流平台。该平台提供用户友好的界面，方便患者浏览、比较和选择符合需求的牙科诊所，同时具备搜索、筛选和预约功能，帮助患者根据特定条件选定合适的诊所。
2. 实现患者对诊所服务的评价系统，合理反映医疗专业性、服务态度、等候时间等方面的信息。这些评价有助于其他患者全面了解诊所的实际情况，为患者选择合适的牙科诊所提供了参考依据，从而提升诊所服务质量和患者满意度。
3. 分别为管理员、诊所用户、医生和患者提供专属服务。管理员可以管理平台上的一切资源并审批诊所申请；诊所用户可以管理诊所资源和医生排班；医生可以管理自身信息并在线填写病案；普通用户可以查询服务信息、进行就诊预约并评价服务。通过这些功能，平台实现了灵活的管理和高效的沟通，提升了各方的使用体验。
4. 为医生和患者提供实时获取与共享医疗信息的途径。包括：在线查看病历和检查报告、健康档案管理、在线沟通和咨询、电子处方等功能。

### 1.4.2 论文结构

本文将分为五个章节介绍项目：

第一章概述课题背景，分析口腔医疗行业及其移动应用的现状与发展前景，讨论其面临的问题，并制定本课题项目的研究内容。

第二章讨论课题相关知识，介绍项目所用的平台，前后端框架。

第三章介绍具体的系统架构，给出系统核心功能开发的具体细节。

第四章为功能展示，将以文字配合图片的形式，对系统各项界面进行展示。

第五章总结项目成果，指出优势与不足，提出改进思路及展望未来发展。

# 课题相关技术

本章将概述本课题口腔护理移动应用的相关技术和框架，本系统分为前端后端两个模式，并通过网络协议的方式进行通信。下面是所使用的开发环境。

表2.1 应用开发硬件环境

|  |  |
| --- | --- |
| CPU | 12th Gen Intel(R) Core(TM) i9-12900KF 3.19 GHz |
| GPU | NVIDIA GeForce RTX 3080 |
| 内存空间 | 32.0 GB (31.8 GB 可用) |
| 外存空间 | 954GB |

表2.2 移动端开发软件环境

|  |  |
| --- | --- |
| 基础框架 | Vue3 |
| 框架 | uniapp |
| IDE | HBuilder |
| 运行平台 | 微信开发者工具 |

表2.3 web端开发软件环境

|  |  |
| --- | --- |
| 语言 | HTML |
| IDE | Pycharm 2023.3.4 |

表2.4 后端开发软件环境

|  |  |
| --- | --- |
| 语言 | Python 3.8.19 |
| 框架 | django 4.1.0 |
| 数据库 | SQLite |
| IDE | Pycharm 2023.3.4 |

## 运行平台 微信小程序介绍

微信小程序（Wechat Mini Program），小程序的一种，是一种运行在微信端而不需要下载安装即可使用的应用[6]。相较于传统的移动应用（APP），选择微信小程序作为运行平台有着许多优势：

（1）快速访问与高用户留存：用户可通过微信搜索或好友分享直接访问小程序，不需下载安装，这降低了用户接触服务的障碍。用户完成服务后不必卸载应用，保持在微信内，便于再次访问。化性能的方向。

（2）社交功能的整合：用户可以非常方便地将预约信息、健康资讯或服务分享到微信群或朋友圈，增加应用的曝光和用户引流，可以利用微信群、公众号等进行用户教育和持续营销。

（3）便捷的支付解决方案：用微信支付的集成使得支付过程无缝且安全，用户可以轻松支付咨询费、预约定金等。支付后可以直接在小程序中获取支付结果，方便用户和业务流程管理。

（4）优化的用户体验：用微信小程序提供的界面和交互设计符合用户习惯，使得应用易于操作，界面友好。可以利用微信的登录机制，无需独立的账户系统，减少用户操作步骤。

（5）易于维护和更新：用微信小程序平台允许快速迭代和发布更新，无需经过繁琐的应用商店审核。开发者可以实时修复bug和更新内容，用户访问时总是最新版本。

（6）跨平台性：微信小程序基于微信，可在iOS和Android系统上无缝运行，无需为不同平台分别开发。

微信小程序为口腔移动护理应用提供了一个方便、快捷、成本低和用户基础广的平台，非常适合用来提供预约、咨询、用户管理等服务。这些优势使得微信小程序成为开发口腔护理相关应用的理想选择。

## 前端框架 uni-app 介绍

uni-app是一个使用 vue.js 开发所有前端应用的框架，非常适合开发跨平台的移动应用和网页应用。这个框架允许开发者使用一套代码，可以编译到iOS、Android、Web（包括PC网站和移动网站）、以及各种小程序（微信、支付宝、百度、字节跳动等）和快应用等多个平台。该框架的核心技术是Vue.js,这是一个流行的前端框架，专注于视图层，易于学习且轻量级[10]。核心是一个允许开发者以数据驱动的方式进行视图开发的系统，通过使用声明式的数据绑定和组件系统，开发者可以以直观和结构化的方式构建。跨平台能力使得uni-app非常适合快速开发和推广本系统，特别是当需要覆盖广泛用户时。

### 将uni-app运行到微信小程序

uni-app的官方指定IDE为HBuilder，需要如下步骤将HBuilder中的代码运行到微信开发者工具中。

1. 打开HBuilder，打开菜单栏下的运行，在运行到小程序模拟器一栏中找到运行设置选项并且打开

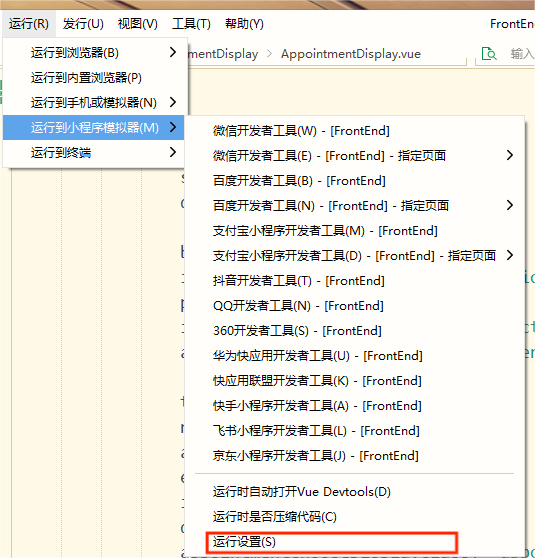


图 2.1 找到HBuilder中的运行设置

1. 在运行配置一栏中找到微信开发者工具路径一栏，填入微信开发者工具后台路径



图 2.2 填入包路径

1. 在微信开发者工具中打开任意项目，在菜单栏中找到设置选项打开安全设置

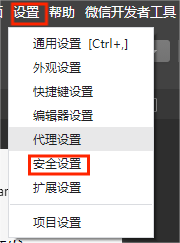


图 2.3 打开安全设置

1. 打开服务端口，待端口号出现则代表服务端口成功打开

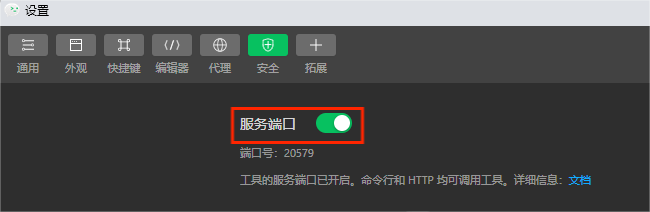


图 2.4 打开服务端口

1. 返回HBuilder，打开运行框，在“运行到小程序模拟器”一栏中找到“微信开发者工具”，点击即可运行到微信开发者工具中

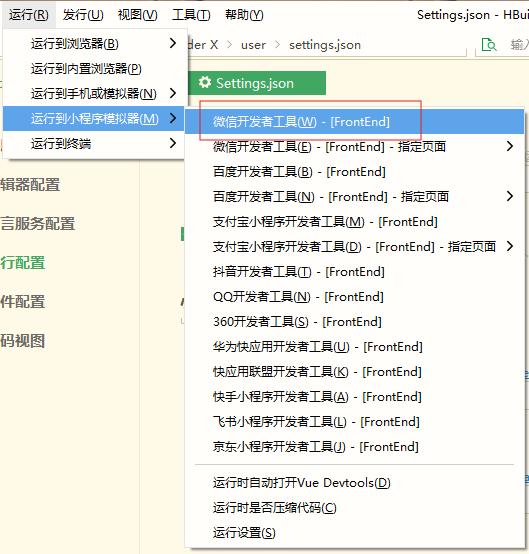


图 2.5 运行到微信开发者工具

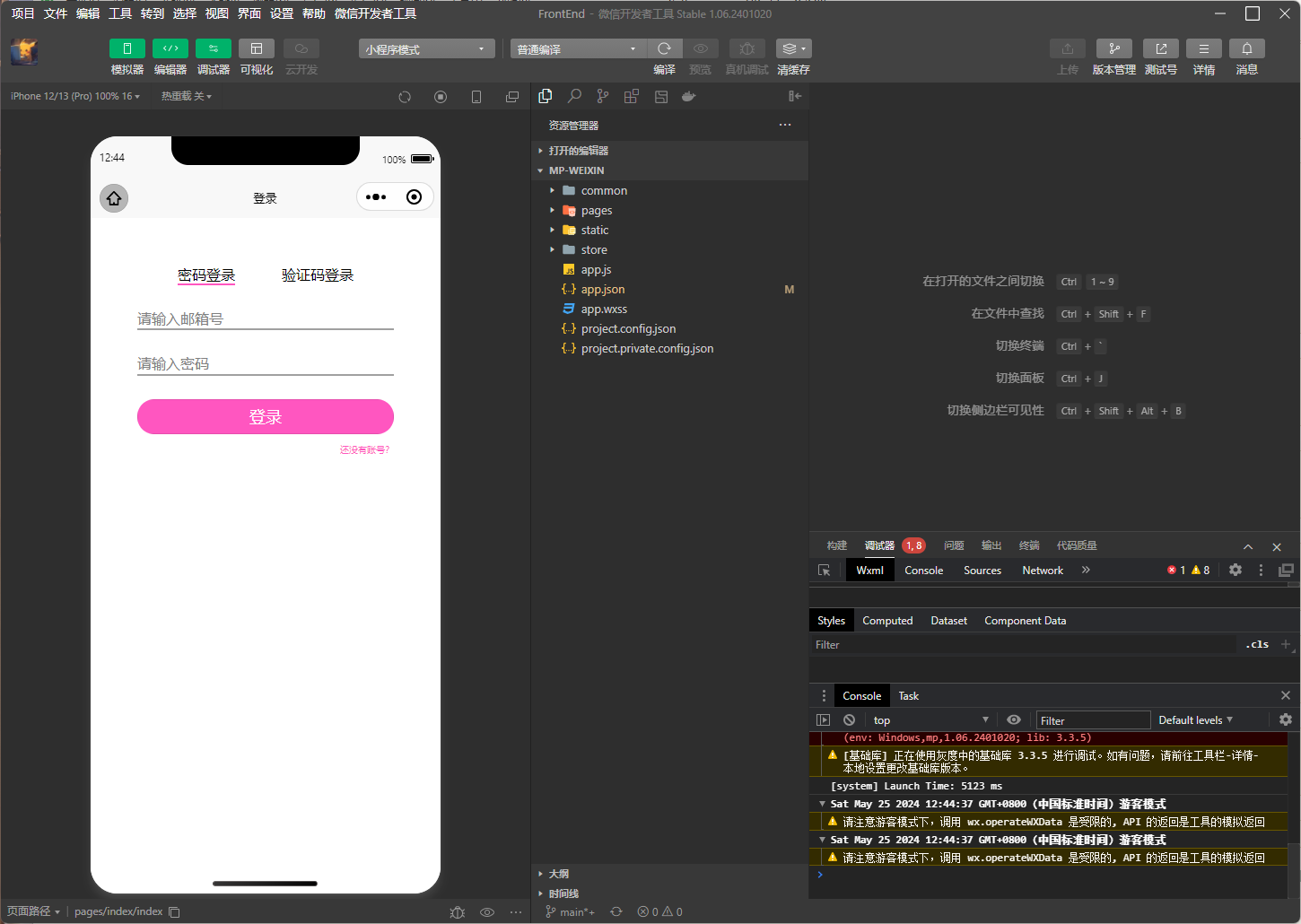


图 2.6 运行成功截图

### uni-app API介绍

uni-app的 js API 由标准 ECMAScript 的 js API 和 uni 扩展 API 这两部分组成。标准 ECMAScript 的 js 仅是最基础的 js。浏览器基于它扩展了 window、document、navigator 等对象。小程序也基于标准 js 扩展了各种 wx.xx、my.xx、swan.xx 的 API。node 也扩展了 fs 等模块。uni-app 基于 ECMAScript 扩展了 uni 对象，并且 API 命名与小程序保持兼容。

（1）uni.request。前端资源请求函数，通过设置url、请求参数以及请求方法向服务器发送请求，请求成功后在success函数中处理服务器返回的数据。



图 2.7 uni.request示例

如果所示为项目中appointment.vue文件中的一个uni.request函数，其实现了通过城市以及县的名称来获取指定地点的诊所信息的功能。其中url为访问地址，后端在接收到请求后将前往“self.BaseURL + ‘appointment/clinic/get/’”地址获取信息，method定义了HTTP请求方式为GET，data中提供了城市以及县级行政区的信息，后端在接收请求后能够通过data中的数据来获取指定的内容。success函数定义了获取成功后的行为，示例中的函数将返回的列表中的信息存入前端本地的数组中，方便后期调用。

1. 页面切换API。在普通的vuejs中涉及到页面切换功能常常要使用Vue Router，即Vue.js官方路由管理器。但uni-app提供了自己的API来实现页面切换。这里涉及如下几个API：

* uni.navigateTo，保存当前页面而后跳转至新的页面，如下示例为appointment.vue文件中的getClinic函数，通过调用该函数能够打开选中的诊所展示页面，图中的uni.navigateTo函数在保存当前页面的情况下能够打开clinicDisplay.vue页面

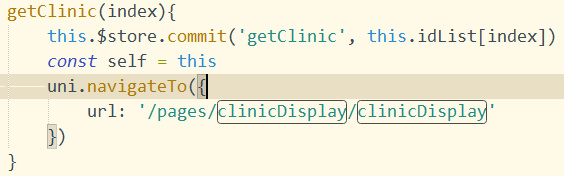


图 2.8 uni.navigateTo示例

* uni.redirectTo，如图所示为appointment.vue文件的mounted生命周期钩子函数，即在页面挂载阶段调用的函数，该函数主要具有两个功能：判断地点初始化以及页面初始化，地点若判断为未初始化将跳转至地点选择页面。图中的uni.redirectTo函数即允许用户能够在关闭当前页面的情况下跳转至location.vue页面



图 2.9 uni.redirectTo示例

* uni.navigateBack，如图所示为comment.vue文件的commentDeleteClick函数，该函数允许用户向服务器端发送信息删除评价，删除后显示“删除成功”的提示而后返回上一页面。图中的uni.navigateBack函数允许用户在成功删除评价之后返回上一页面。



图 2.10 uni.navigateBack示例

1. uni.getSystemInfo。通过该API能够获取屏幕尺寸，在设定背景颜色时通过获取屏幕尺寸减去标题高度得到整个背景的尺寸，通过position属性为fixed的css属性设置整个背景的颜色。如图 2.12所示，在mounted生命周期钩子函数分为两个部分：诊所评价获取以及页面尺寸计算。页面尺寸计算调用了uni.getSystemInfo函数来获取页面尺寸并计算出页面背景的rpx尺寸。而后如图 2.11所示，使用v-bind将screenHrightRpx绑定到height属性上，并将position属性设置为fixed即可创建出background的css属性，通过使用这个css类可以改变页面的背景颜色。

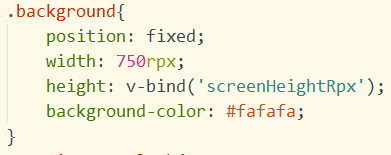


图 2.11 background类属性



图 2.12 screenHeightRpx的计算

## 服务端框架Django介绍

Django 是一个高级的 Python Web 框架，由国外的程序员Adrian Holovaty和Simon Willison在2005年开发。Django 是为了快速开发复杂、数据库驱动的网站而创建的，它采用了MVC（模型-视图-控制器）的设计模式，但进行了改进，提出了MTV（模型-模板-视图）架构。

### Django接收请求

Django接收前端请求主要设计路由分发以及视图函数两个部分。

#### 路由分发

Django的路由分发书写在url.py中，涉及path以及include两个函数。

（1）path函数。path函数有两个参数，分别为路由路径以及视图函数，根据路由路径找到视图函数，即可调用对应的函数。如下图，如果一个请求到达此url.py，根据如果对应地址符合“follow/dislike”即可调用follow.dislike视图函数。



图 2.13 path函数的应用

（2）include函数。在一些情况下如果将所有path函数书写在同一个url.py中会显得比较臃肿，因此提出使用多个url.py分别书写，这就需要使用include函数。使用一个include函数即可将与地址对应的请求重定向到include函数指定的url.py，扣除匹配的地址后再由第二个url.py做进一步处理。如图所示，所有访问此py文件的请求都将继续访问client.urls.py文件。



图 2.14 path函数的应用

#### 视图函数

视图函数与普通函数的不同在于HTTP的请求以及返回。普通函数的输入参数与返回类型为用户手动传入，而视图函数的输入参数为request，即用户端发回的请求，输出内容为HttpResponse以及JsonResponse类型，能够作为HTTP回复返回给用户端。

（1）request参数。request为用户端请求在Django项目中的表达，通过request.method以及request.GET或者request.POST方法获取请求的参数。如下图所示为服务端appointmentDisplay.py文件中的initial1函数，通过提供诊所编号以及医生账户的邮箱返回医生的信息。request作为函数中的固定参数被传入initial1函数内部，首先查看request.method是否为GET，即请求方法是否正确，其次通过request.GET获取请求的具体参数。request.GET[‘clinic’]获取请求中json格式clinic属性的参数，即诊所编号，request.GET[‘doctor’]获取请求中json格式doctor属性的参数，即医生账户邮箱。

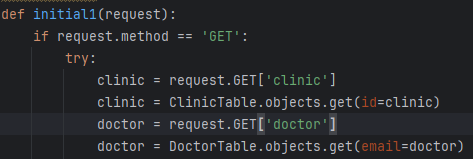


图 2.15 视图函数对request的处理

1. 返回数据类型。使用HttpResponse以及JsonResponse来返回数据。两者都能返回数据且都能返回HTTP状态，不同之处在于HttpResponse只能返回字符串数据，JsonResponse却能返回JSON格式数据，在python中JSON格式数据由字典表示。如图 2.16为appointmentDisplay.py文件initial1函数正常运行的返回内容，data使用字典格式表示json格式数据并罗列医生姓名，年龄，学历等信息而后使用JsonResponse返回消息；如图 2.17所示为initial1函数运行错误的返回内容，仅返回状态码400。



图 2.16 返回JsonResponse



图 2.17 返回HttpResponse

### ORM系统

Django的ORM（对象关系映射）系统是其核心特性之一，它允许开发者以Python类的形式定义模型（models），这些模型类会自动映射到数据库表。通过ORM，Django提供了高级的抽象，使得数据库操作变得简单且直观。

#### 项目注册

Django中的所有应用都是通过项目来实现的。通过django-admin startproject <name>来创建项目，项目目录中找到apps.py，将其中的Config注册到settings.py的INSTALLED\_APPS列表中即可。

#### 数据表注册

Django的ORM系统的优越性之一在于允许用户通过python语句定义和操作数据表。在models.py文件中使用继承自models.Model的python类来表示数据表。与大部分数据库应用不同的是，Django中不存在联合主键的概念；在没有显式定义主键的情况下Django将自动在数据表中注册一个AutoField类型的id变量作为主键。另一方面，Django数据表利用Django库中的类型定义数据类型，利用数据类型的接口定义数据类型的数据，例如，通过设置CharField类型的null接口的Boolean类型来设置这个属性能否为空。

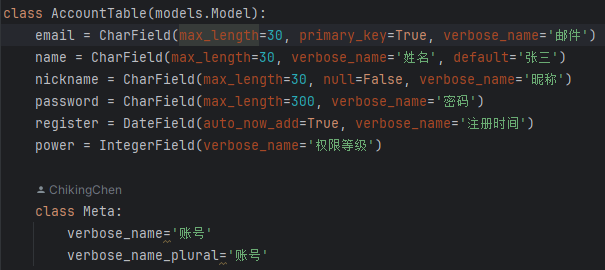


图 2.18 数据表的注册

如上图所示定义了一个继承自models.Model的AccountTable类，通过定义email，name，nickname，password等6个变量的类型以及类型的接口属性来确定数据表的属性。

编写完成后输入python manage.py makemigrations创建migraiton文件，标记数据库数据表发生的变化，而后输入python manage.py migrate将migration文件中的变化更新到数据库中。

#### 数据库管理

Django通过提供admin站点来管理ORM系统中的数据库。在admin.register装饰器中指明注册的数据表并装饰继承自admin.ModelAdmin的类即可将类注册为显示指定数据表的管理类，在类中通过设定list\_display来设置显示的属性。

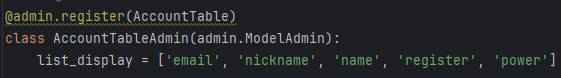


图 2.19 AccountTable的注册

图 2.19为admin.py文件中将AccountTable注册至AccountTableAdmin类并且在admin站点中显示email，nickname，name，register，power等属性。如图 2.20显示，为Django admin站点中AccountTable数据表的详细信息。

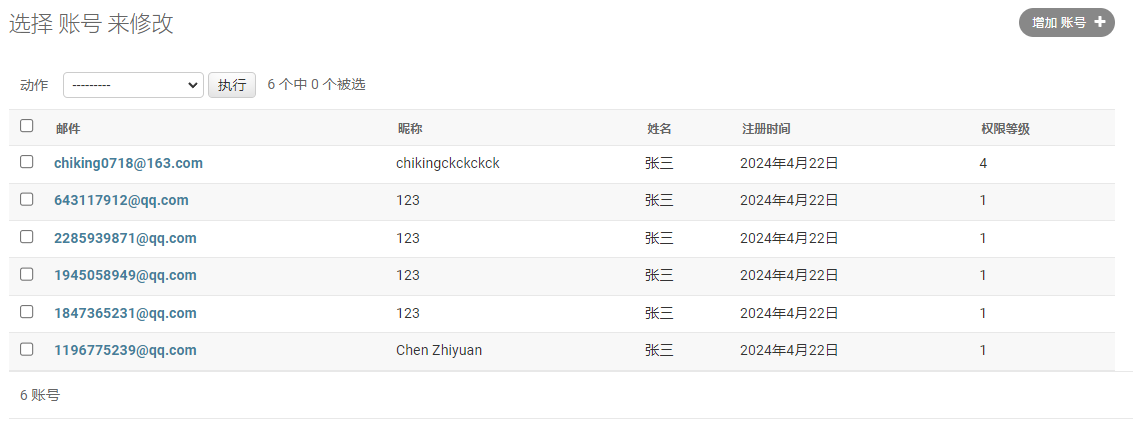


图 2.20 AccountTable在admin站点的显示

通过admin站点开发者能够手动对站点数据库增删改查。

#### 数据库增删改查

Django框架的一大优势就在于能够通过python语言直接对数据库增删改查。从models.py文件中将数据表对象导入，使用数据表类名直接引用数据表例如通过引用AccountTable来作为AccountTable表的引用，AccountTable.objects即为整个数据表的数据及类型。

1. 增。如图所示为signin.py的sign\_in函数，用户能够通过调用该函数来完成账号注册的功能，函数的主要部分为try中的内容，分为三个部分：信息读取、注册合法性检查以及数据库保存，其中数据库保存使用了.save()接口。在前端发来的验证码与数据库中的验证码相同的情况下创建AccountTable对象，而后使用.save接口保存。

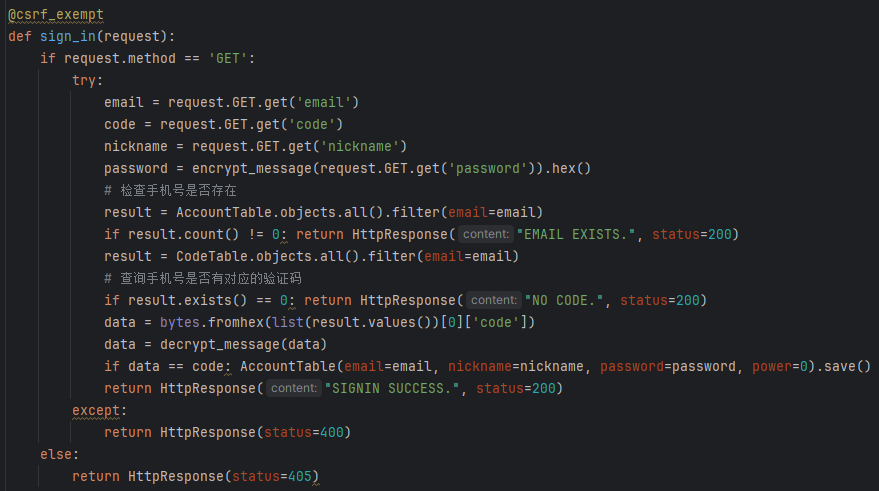


图 2.21 sign\_in函数

1. 删。如图所示为signin.py的get\_code函数，用户通过调用该函数能够向指定邮箱发送指定内容的邮件。函数分为四个部分：信息读取、发送合法性检查、验证码发送以验证码保存，在合法性检查部分如果验证码表中已经存在该邮箱则对CodeTable的QuerySet对象调用.delete接口来删除邮箱的对应项。

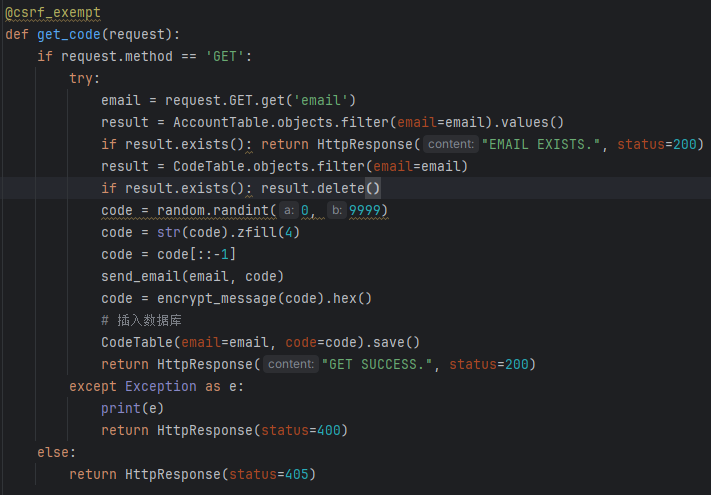


图 2.22 get\_code函数

1. 改。改的对象同样为QuerySet类型，对QuerySet类型的对象调用.update操作即可更新QuerySet中的数值。如图所示为read函数，通过调用该函数用户能够修改消息的已读状态。函数分为两个部分：信息读取以及状态修改。其中，在状态修改部分用户通过MessageTable.objects获取数据表的全部对象，而后使用filter来筛选id最后调用update来修改被筛选出来的QuerySet的read状态。



图 2.23 read函数

1. 查。查的步骤分为两步——筛选和显示。筛选使用.filter接口，在其中输入查找的条件即可筛选出符合条件的记录，而后使用.values接口，在values中输入需要查看的属性，即可返回包含具体数据的QuerySet。如图所示，为star.py文件的get函数，用户通过该函数能够获取平台用户的收藏内容。其分为两个部分：信息获取以及内容构建，内容构建中用户通过筛选StarTable.objects中的QuerySet对象而后使用.values()接口获取详细内容，使用for循环来遍历QuerySet的内容进而构建starList。



图 2.24 get函数

#### SQLite

Django ORM系统中默认数据库为SQLite。SQLite，是一款轻型的数据库，是遵守ACID的关系型数据库管理系统，它包含在一个相对小的C库中。它是D.RichardHipp建立的公有领域项目。它的设计目标是嵌入式的，而且已经在很多嵌入式产品中使用了它，它占用资源非常的低，在嵌入式设备中，可能只需要几百K的内存就够了。它能够支持Windows/Linux/Unix等等主流的操作系统，同时能够跟很多程序语言相结合，比如 Tcl、C#、PHP、Java等，还有ODBC接口，同样比起Mysql、PostgreSQL这两款开源的世界著名数据库管理系统来讲，它的处理速度比他们都快。正是其快速、便捷的特点允许其嵌入ORM系统成为其一部分。

### 模板系统

Django的模板系统是一个强大的工具，它允许开发者使用HTML、CSS、JavaScript以及Django特有的模板语言来创建动态的Web页面。模板系统的主要作用是将Web页面的布局、设计和内容分离，使得Web应用的维护变得更加简单。在项目文件夹下方创建一个templates文件夹，其中编写的html文件能够作为模板系统中的文件进行调用。

#### render函数

可以理解的是，Django项目运行在127.0.0.1:8000端口上，我们通过url来访问不同的页面本质上与用户端访问服务器来获取资源在具体操作上具有同质性，不同的地方在于用户端访问服务器是为了获取JSON格式数据或者字符串类型数据，而模板系统中通过url访问是为了获取由HTML编写的页面，这就需要引入新的返回函数。

我们这边引入render函数。render函数有三个参数，第一个为视图函数的request参数，第二个为template文件夹中html文件的路径，第三个为一字典变量，作为Django模板系统的特性可在html文件中调用。

如图 2.24为doctor.py文件的index函数，用户通过调用这个函数能够获取医生主页，设置dict字典对象作为render的字典调用对象而后调用render函数作为医生主页内容。



图 2.25 render函数使用示例

#### HTML

Django模板系统使用HTML语言来进行用户端开发。HTML是一种用于创建网页的标准标记语言。它描述了一个网站的结构骨架，使得浏览器能够展示具有特定格式的文本、链接、图片和其他内容。HTML 文档由一系列的元素构成，这些元素通过标签来定义，并且可以嵌套使用，以表示页面内容的不同部分和结构。

# 系统设计

## 系统整体架构

系统分为移动端、web端以及服务端三个部分，采用前后端分离的结构。移动端为用户服务，运行在用户设备上，负责获取用户的需求、向服务端服务器发送请求获取资源并在移动端上显示；web端为医生、诊所以及管理员用户服务向服务端发送数据并提供数据库管理服务；服务端是唯一能够访问数据库的设备，接受移动端以及web端的请求、访问数据库进行处理并且返回处理的结果。

移动端使用uniapp框架以及vuejs，使用Hbuilder X作为IDE进行编程[13]，并将微信小程序作为运行平台。web端使用html语言进行页面设计，并使用xHttpRequest对象向后端发送请求。服务端采用python语言以及django框架接受前端的请求。

## 后端接口

### 接口规范和管理

本系统接口采用 RESTful 架构风格，用户端使用 GET、POST、PUT、 DELETE 四个操作类型对服务端资源进行操作：GET 用来获取资源，POST 用来新建资源和更新资源，PUT用来更新资源，DELETE用来删除资源。

前后端交互采用JSON格式。JSON是一种轻量级的数据交换格式，它基于JavaScript编程语言的一个子集，但是由于其文本格式清晰且易于解析，因此它被许多编程语言广泛支持。JSON格式易于人阅读和编写，同时也易于机器解析和生成，因此在Web开发中被广泛应用于数据的格式化和传输。

### 用户端接口罗列

用户端接口主要分为以下几个模块：个人信息模块，管理模块，预约模块以及评价模块。

（1）个人信息模块

表 3.1 个人信息模块

|  |  |  |
| --- | --- | --- |
| **接口名称** | **请求方式** | **请求路径** |
| 密码登录 | GET | login/psw/ |
| 邮件登录 | GET | login/email/ |
| 登录验证码 | GET | login/get/ |
| 注册验证码 | GET | signin/get/ |
| 注册提交 | GET | signin/signin/ |
| 主页姓名 | GET | personality/name/ |
| 主页初始化 | GET | personality/initial/ |

个人信息模块主要负责登录注册以及主页的信息传递。通过访问AccountTable，CodeTable等数据表来获取对应的数据。

（2）管理模块

表 3.2 管理模块

|  |  |  |
| --- | --- | --- |
| **接口名称** | **请求方式** | **请求路径** |
| 城市获取 | GET | location/city/get/ |
| 城市删除 | GET | location/city/delete/ |
| 城市添加 | GET | location/city/add/ |
| 城市修改 | GET | location/city/modify/ |
| 县获取 | GET | location/county/get/ |
| 县删除 | GET | location/county/delete/ |
| 县添加 | GET | location/county/add/ |
| 县修改 | GET | location/county/modify/ |
| 账号获取 | GET | account/get/ |
| 账号信息获取 | GET | account/information/ |
| 账号昵称修改 | GET | account/modify/nickname/ |
| 账号密码修改 | GET | account/modify/psword/ |
| 账号权限修改 | GET | account/modify/power/ |

管理模块主要负责平台高级管理员管理功能的信息传递，将完成城市、县以及账号信息增删改查的功能，允许管理员在用户端也能实现一部分功能。

（3）预约模块

表 3.3 预约模块

|  |  |  |
| --- | --- | --- |
| **接口名称** | **请求方式** | **请求路径** |
| 获取诊所 | GET | appointment/clinic/get/ |
| 预约初始化 | GET | appointment/get/ |
| **接口名称** | **请求方式** | **请求路径** |
| 预约展示 | GET | appointment/initial/ |
| 普通号预约展示 | GET | appointment/display/initial1/ |
| 专家号预约展示 | GET | appointment/display/initial2/ |
| 普通号挂号 | GET | appointment/make1/ |
| 专家号挂号 | GET | appointmen/make2/ |
| 诊所初始化 | GET | fclinic/initial/ |
| 诊所医生 | GET | fclinic/getdoctor/ |
| 获取信息 | GET | message/get/ |
| 信息读取 | GET | message/read/ |

预约模块主要负责实现预约以及信息读取功能的前后端信息交互，能够完成诊所选择、诊所展示、诊所预约等功能，最终构成预约板块。

（4）评价模块

表 3.4 评价模块

|  |  |  |
| --- | --- | --- |
| **接口名称** | **请求方式** | **请求路径** |
| 评价名称获取 | GET | name/get/ |
| 获取待评价 | GET | comment/appointment/ |
| 评价提交 | GET | comment/submmit/ |
| 评价删除 | GET | comment/delete/ |
| 评价获取 | GET | comment/get/ |
| 评价信息获取 | GET | comment/information/ |
| 评价诊所获取 | GET | comment/clinic/ |
| 评价点赞 | GET | comment/makelike/ |
| 评价取消点赞 | GET | comment/dislike/ |
| 评价收藏 | GET | comment/makestar/ |
| 评价取消收藏 | GET | comment/disstar/ |
| 评价页面初始化 | GET | comment/initial/ |
| 评价修改信息获取 | GET | comment/modify/information/ |
| 评价修改提交 | GET | comment/modify/submmit/ |
| 回复取消点赞 | GET | follow/dislike/ |
| 回复点赞 | GET | follow/like/ |
| 回复提交 | GET | follow/add/ |
| 回复删除 | GET | follow/delete/ |
| 收藏获取 | GET | star/get/ |

评价模块主要负责评价以及评论区信息的前后端交互。评价模块为推荐系统的主要内容，用户通过访问评价模块的信息能够选取更加合适自己的服务。通过访问CommentTable，LikeTable，StarTable，FollowTable等数据表，能够完成对应的任务。

## 数据库设计

### 3.3.1 数据库架构

本系统采用django自带的SQLite数据库。

SQLite负责完成关系型数据库的管理，通过关系型数据库的⾼效查询能⼒，满⾜关系查询能⼒。

### 3.3.2 数据库设计

数据库主要涉及20张表，其中的依赖关系如ER图所示

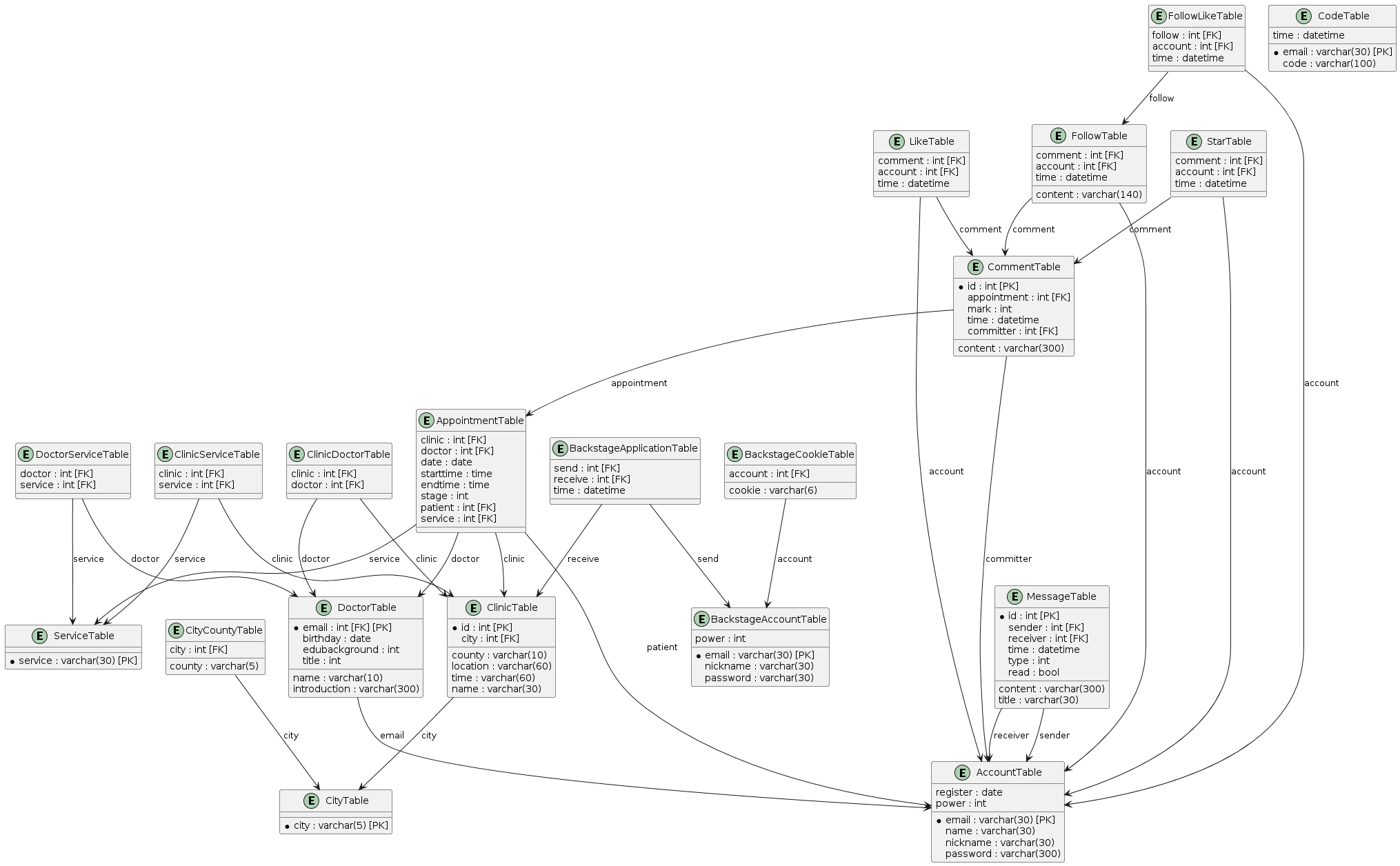


图 3.1 ER图

1. 账号表，存储用户端所有账号。数据表以邮箱为主键，意味着当前平台通过邮箱来标记用户的唯一性，用户的账号即为注册邮箱。

表 3.5 账号表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| email | CharField | 30 | 是 | 邮件 |
| name | CharField | 30 | 否 | 姓名 |
| nickname | CharField | 30 | 否 | 昵称 |
| password | CharField | 300 | 否 | 密码 |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| register | DateField | - | 否 | 注册时间 |
| power | IntegerField | - | 否 | 权限等级 |

1. 城市表，存储平台上的城市。方便服务端获取平台上已经注册的城市。

表 3.6 城市表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| city | CharField | 5 | 是 | 城市 |

1. 县表，记录了平台上注册的县级行政区。

表 3.7 县表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| id | CharField | - | 是 | - |
| city | ForeignKey (CityTable) | - | 否 | 城市 |
| county | CharField | 5 | 否 | 县 |

1. 验证码表，存储注册登录功能模块中发送的验证码。

表 3.8 验证码表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| email | CharField | 30 | 是 | 用户邮箱 |
| code | CharField | 100 | 否 | 验证码 |
| time | DateTimeField | - | 否 | 输入时间 |

1. 诊所表，存储平台上的诊所详细信息。

表 3.9 诊所表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| id | AutoField | - | 是 | - |
| city | ForeignKey (CityTable) | - | 否 | 城市 |
| county | CharField | 10 | 否 | 县 |
| location | CharField | 60 | 否 | 地点 |
| time | CharField | 60 | 否 | 营业时间 |
| name | CharField | 30 | 否 | 诊所名称 |

1. 医生表，存储平台上的医生详细信息。

表 3.10 医生表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| email | ForeignKey (AccountTable) | - | 是 | 账户 |
| name | CharField | 10 | 否 | 姓名 |
| birthday | DateField | - | 否 | 生日 |
| edubackground | IntegerField | - | 否 | 学历 |
| title | IntegerField | - | 否 | 职称 |
| introduction | CharField | 300 | 否 | 介绍 |

1. 服务表，存储平台上能够提供的服务，诊所选择服务表中的服务来选择自己诊所能够提供的服务。

表 3.11服务表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| service | CharField | 30 | 是 | 服务 |

1. 诊所服务表，存储诊所能够提供的服务。

表 3.12 诊所服务表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| id | AutoField | - | 是 | - |
| clinic | ForeignKey (ClinicTable) | - | 否 | 诊所 |
| service | ForeignKey (ServiceTable) | - | 否 | 服务 |

1. 医生服务表，存储医生能够提供的服务。

表 3.13 医生服务表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| id | AutoField | - | 是 | - |
| doctor | ForeignKey (DoctorTable) | - | 否 | 医生 |
| service | ForeignKey (ServiceTable) | - | 否 | 服务 |

1. 预约表，存储医生在指定时间能够提供的服务。

表 3.14 预约表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| id | AutoField | - | 是 | - |
| clinic | ForeignKey (ClinicTable) | - | 否 | 诊所 |
| doctor | ForeignKey (DoctorTable) | - | 否 | 医生 |
| date | DateField | - | 否 | 日期 |
| starttime | TimeField | - | 否 | 开始时间 |
| endtime | TimeField | - | 否 | 结束时间 |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| stage | IntegerField | - | 否 | 阶段 |
| patient | ForeignKey (AccountTable) | - | 否 | 病人 |
| service | ForeignKey (ServiceTable) | - | 否 | 服务 |

1. 消息表，存储平台上传递的消息。

表 3.15 消息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| id | AutoField | - | 是 | - |
| sender | ForeignKey (AccountTable) | - | 否 | 发送者 |
| receiver | ForeignKey (AccountTable) | - | 否 | 接收者 |
| time | DateTimeField | - | 否 | 发送时间 |
| content | CharField | 300 | 否 | 内容 |
| title | CharField | 30 | 否 | 标题 |
| type | IntegerField | - | 否 | 类型 |
| read | BooleanField | - | 否 | 已读 |

1. 评价表，存储用户对完成的预约的评价。

表 3.16 评价表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| id | AutoField | - | 是 | - |
| appointment | ForeignKey (AppointmentTable) | - | 否 | 预约 |
| mark | IntegerField | - | 否 | 分数 |
| time | DateTimeField | - | 否 | 评价时间 |
| committer | ForeignKey (AccountTable) | - | 否 | 提交者 |
| content | CharField | 300 | 否 | 内容 |

1. 点赞表，存储用户对评价的点赞。

表 3.17 点赞表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| id | AutoField | - | 是 | - |
| comment | ForeignKey (CommentTable) | - | 否 | 评价 |
| account | ForeignKey (AccountTable) | - | 否 | 点赞人 |
| time | DateTimeField | - | 否 | - |

1. 收藏表，存储用户对评价的收藏。

表 3.18 收藏表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| id | AutoField | - | 是 | - |
| comment | ForeignKey (CommentTable) | - | 否 | 评价 |
| account | ForeignKey (AccountTable) | - | 否 | 收藏人 |
| time | DateTimeField | - | 否 | - |

1. 回复表，存储用户对评价的回复。

表 3.19 收藏表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| id | AutoField | - | 是 | - |
| comment | ForeignKey (CommentTable) | - | 否 | 评价 |
| account | ForeignKey (AccountTable) | - | 否 | 评价人 |
| content | CharField | 140 | 否 | - |
| time | DateTimeField | - | 否 | - |

1. 回复点赞表，存储用户对回复的点赞。

表 3.20 回复点赞表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| id | AutoField | - | 是 | - |
| follow | ForeignKey (FollowTable) | - | 否 | 回复 |
| account | ForeignKey (AccountTable) | - | 否 | 评价人 |
| time | DateTimeField | - | 否 | 时间 |

1. 后台账户表，存储web端的账户。

表 3.21 后台账户表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| email | EmailField | - | 是 | 邮件 |
| nickname | CharField | 30 | 否 | 昵称 |
| power | IntegerField | - | 否 | 权限 |
| password | CharField | 30 | 否 | 密码 |

1. 后台Cookie表，存储web端浏览器的Cookie。

表 3.22 后台Cookie表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| id | AutoField | - | 是 | - |
| account | ForeignKey (BackstageAccountTable) | - | 否 | 账号 |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| cookie | CharField | 6 | 否 | cookie |

1. 后台医生申请表，存储web端医生的申请。

表 3.23 后台台医生申请表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| id | AutoField | - | 是 | - |
| send | ForeignKey (BckstgeAcountTabe) | - | 否 | 发送者 |
| receive | ForeignKey (ClinicTable) | - | 否 | 诊所 |
| time | DateTimeField | - | 否 | 发送时间 |

1. 诊所医生表，存储哪些诊所有哪些医生。

表 3.24 诊所医生申请表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **最大长度** | **是否主键** | **显示名称** |
| id | AutoField | - | 是 | - |
| clinic | ForeignKey (ClinicTable) | - | 否 | 诊所 |
| doctor | ForeignKey (DoctorTable) | - | 否 | 医生 |

# 功能展示与实现

## 用户界面

### 登录注册模块

登录注册模块主要分为登录以及注册两个页面，更具体来说登录又分为密码登录以及验证码登录两个模式，以下详细介绍。

(a)密码登录 (b)验证码登录

图 4.1 登录界面

如上为两种登录模式。密码登录中使用邮箱以及对应的密码即可登录，验证码登录则使用邮箱以及实时发送的验证码作为登录凭证。两种登陆模式分别依赖login/psw/以及login/email/端口。前者查询账号表数据对比发送的密码以及数据表中对应的密码，如若相同则登录成功；后者则查询验证码表对比发送的验证码以及数据表中对应的验证码，如若相同则登录成功。

验证码的发送依靠依赖smtplib库的send\_email函数。



图 4.2 send\_email函数

如图所示为服务端利用smtp库发送邮件的代码，通过MIMEText设置邮件内容最后调用smtp库使用发送邮件。

(a)空白页面 (b)填写完成

图 4.3 注册界面

如图所示为注册界面，用户通过输入点击邮箱、对应的验证码、昵称以及密码即可向服务端发送注册请求。注册请求依赖服务端signin/signin/的端口，注册成功后则在账号表数据中增加当前账号的信息。

### 预约模块

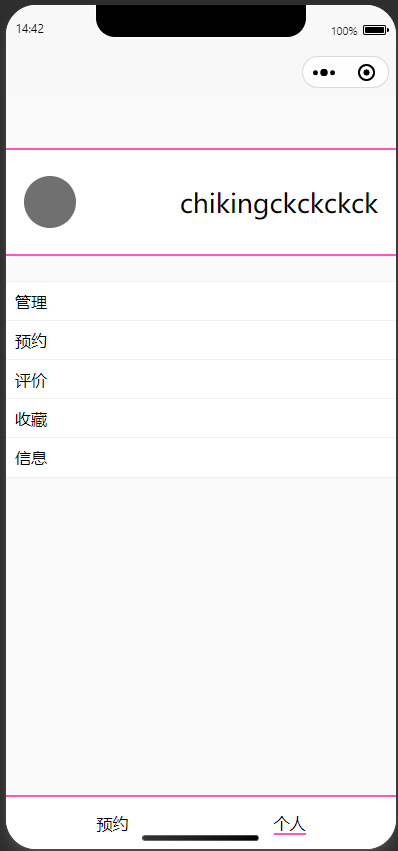
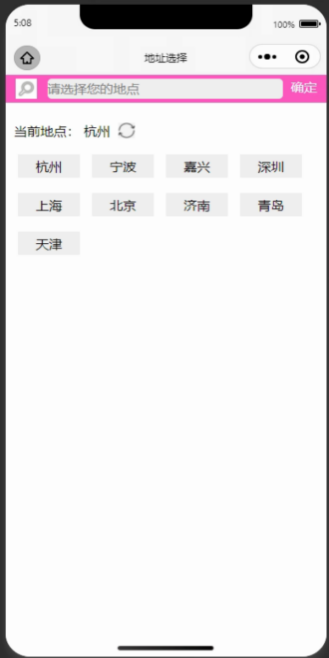


图 4.4 个人首页

登录后进入主页，内容分为导航栏以及页面两个部分，用户能够通过导航栏选取显示的页面。个人首页中将展示用户的昵称以及可以选择的个人选项。在mounted阶段访问服务端的personality/name/接口通过查询账号表数据来获取用户的用户名以及权限。

点击预约即可进入预约界面，接下来介绍预约模块的页面以及功能。

(a)地址选择 (b)诊所展示

图 4.5 预约首页

点击预约后，用户端将检测是否已经选择城市，若无则直接跳转至城市选择界面，用户需要在城市选择界面中选择所在城市进而进一步获取口腔服务。城市选择界面使用三种方式获取所在城市：搜索框、当前地点以及城市选择。搜索框支持精确查找，在搜索框中输入查找城市的名称，即可跳转至诊所查看界面查看被查找城市的口腔诊所；当前地点调用腾讯定位服务的API获取当前所在的市级行政区划，点击后也可以跳转查看；下方城市选择将展示平台上所有注册的城市，点击其中的任意一者即可查看当前地点的牙科诊所。总体而言，城市选择界面的主要功能为获取用户所在地。地点选择完成之后使用vuex的store来存储被选择的城市，在跳转至诊所展示界面之后，程序通过读取vuex中的store来获取被选择的城市，并展示被选择城市的县级行政区以及所在的牙科诊所。

(a)上城区 (b)钱塘区

图 4.6 诊所展示页面

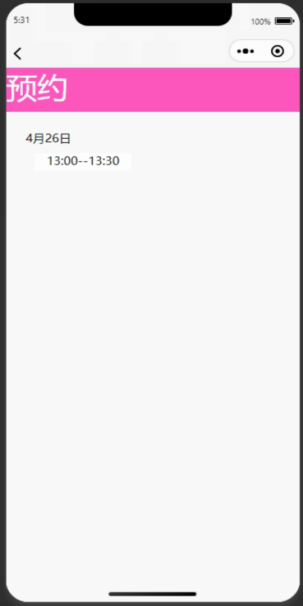
诊所展示页面分为两部分：县级行政区选择以及牙科诊所展示。点击不同的县级行政区可以查看当前县级行政区之下的牙科诊所及其名称、地点以及服务时间信息，选择牙科诊所，进入诊所界面查看详细的医生与服务信息。县级行政区的切换依赖服务端appointment/clinic/get/端口，通过向服务端发送选择的城市以及县级行政区，服务端能够在诊所表中查询符合条件的城市以及县级行政区的诊所，并返回地址、时间、名称以及诊所编号列表。

(a)口腔预防保健 (b)补牙

图 4.7 诊所展示页面

诊所展示界面分为三个部分：标题栏、服务选择以及医生选择，分别展示诊所的信息、诊所能够提供的服务以及能够提供被选中服务的专家号以及普通号。选择普通号或者具体的医生名称能够进入预约展示界面，查看能够预约的预约号，考虑到普通号以及专家号的区别，预约展示界面亦分为两种模式。

(a)普通号 (b)专家号

图 4.8 预约展示页面

如图所示分别为普通号（左）与专家号（右）的预约展示界面。

专家号界面包含专家的包括姓名、年龄、学历、职称、简介等详细信息，并在下方展示其在特定日期特定时间段的预约，只有黑色字体的选项代表当前预约没有被选择，依然能够被预约；灰色字体代表被先前的用户预约。普通号的界面相对简单只包含特定日期特定时间段的预约，其背后的逻辑为能够提供该项服务的医生所有没有被预约的集合。患者选择后将会随机选择一名医生作为自己的服务者。

专家号预约将访问服务端地址appointment/make1/，调用appointmentDisplay.py的makeAppointment1函数，创建预约并发送信息。

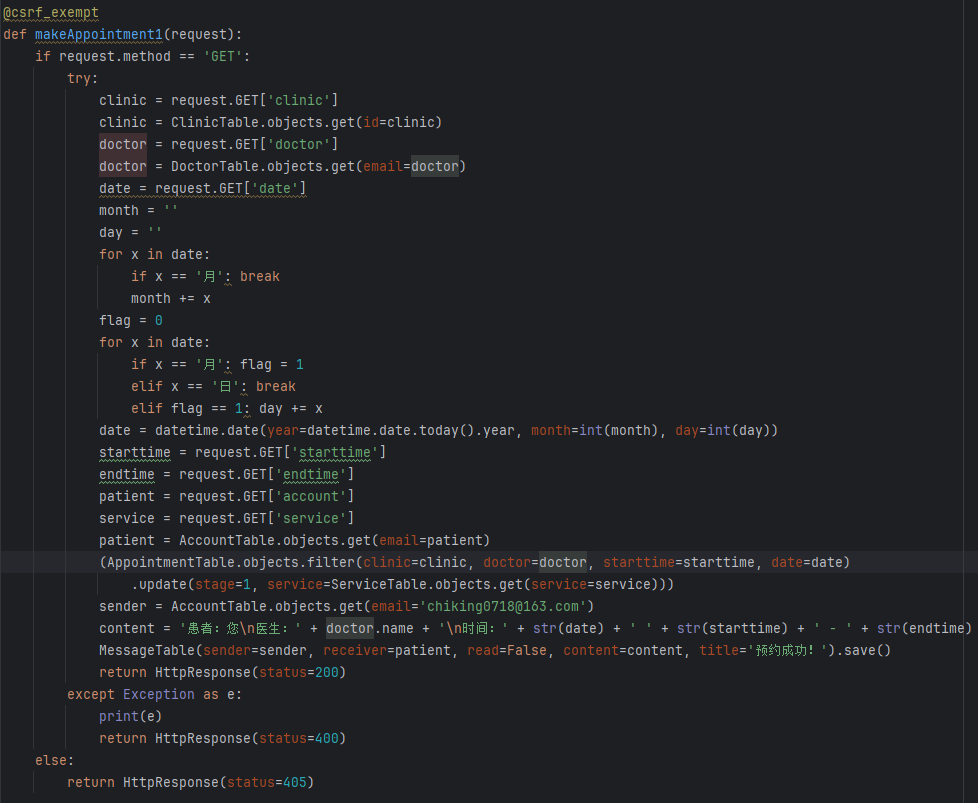


图 4.9 makeAppointment1函数

如图 4.9为appointmentDisplay.py的makeAppointment1函数，分为三个部分：信息获取，预约修改以及信息发送。信息获取部分函数将用户端发送的信息通过request.GET赋值给本地的变量并通过简单的处理成为能够赋值给filter接口的变量；预约修改部分函数查找到预约表中符合条件的记录并标记已被预约；信息发送部分函数设置发送者以及发送内容存储为信息表中的对象。

普通号预约将访问服务端地址appointment/make2/，调用appointmentDisplay.py的makeAppointment2函数，创建预约并发送信息。

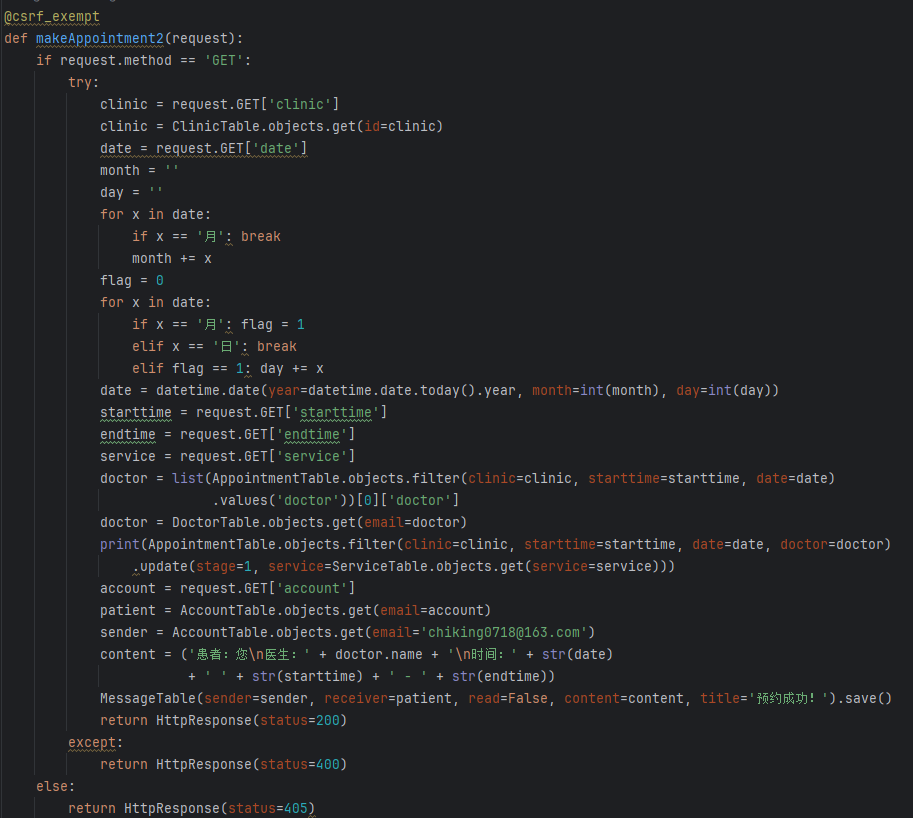


图 4.10 makeAppointment2函数

makeAppointment2函数与makeAppointment1函数不同在于医生对象的获取。makeAppointment1函数中医生对象由用户端直接获取，makeAppointment2函数中医生对象由预约表筛选符合条件的医生中随机选择。

预约成功后用户将收到“预约成功”的消息，页面也将跳转至消息界面。

(a)未读 (b)已读

图 4.11 消息页面

消息展示界面将展示接收到的消息。“预约成功”消息包含患者、医生以及服务时间的信息，用户能够通过查询消息界面中的消息来查找预约的详细信息以及预约是否已经成功。用户通过点击消息即可将消息的状态设置为“已读”，此功能需要访问服务端地址message/read/，调用message.py的read函数，通过用户端发送的消息id修改消息表记录的已读状态。



图 4.12 read函数

与消息界面同步更新的是个人预约界面。

(a)未评价 (b)已评价

图 4.13 个人预约页面

预约分为如下几个阶段：未被预约、已经预约但还未就诊、已经就诊还未评价、已评价。预约界面将显示后面三者的详细信息，包括就诊人、就诊诊所、医生、服务、时间以及预约状态，关于已经就诊但还未评价的预约将会出现评价按钮点击后将使用uni.navigateTo进入评价界面进行评价。

### 评价社交模块

用户在完成预约之后即可评价，评价的内容将成为后续使用此平台的用户的参考，在此构想基础上开发了评价社交模块。

用户进行评价的主要方式是通过个人预约界面以及个人评价界面的“修改”、“评价”按钮进入评价界面评价。

(a)未评价 (b)已评价

图 4.14 个人评价页面

个人评价界面中展示了用户已经完成的预约的诊所、医生、服务、时间等信息。点击右上角的三角按钮即可查看预约的评价情况，根据评价的完成与否将展示“评价”以及“修改”按钮。点击“评价”或“修改”按钮能够进入评价界面新增或者修改评价。

(a)未填写 (b)已填写

图 4.15 评价页面

如上所示为评价页面。评价界面分为三部分：被评价对象信息，评价的分数以及具体内容，评价按钮。通过输入或修改评价的分数及具体内容完成评价。评价完成后将在诊所的评价界面中展示已经完成的评价。评价页面的提交需要访问服务端的comment/submmit/调用comment.py的submmit函数在评价表中保存数据并更新预约表中的阶段。

(a)展示界面 (b)医生搜索 (c)服务搜索

图 4.16 评价展示页面

评价展示界面分为三部分：标题，搜索框以及用户评价。搜索框支持服务和医生名字的精确查找，在搜索框中输入医生或服务即可搜索对应的评价；用户评价包括用户的昵称，评价日期，评价内容以及点赞和收藏数量。用户能够通过点赞收藏数量对评价的质量进行估计选择合适的评价了解更加真实的诊所情况，点击右三角能够查看评价的评论区。点赞收藏的实现需要访问服务端的comment/makelike/，comment/dislike/，comment/makestar/，comment/disstar/地址来对收藏表以及点赞表进行增删。

(a)chiking评价 (b)chikingckckckck评价

图 4.17 评论区页面

评论区界面分为两个部分：用户评价以及回复。用户评价包括评价人，评分，被评价医生，服务以及评价内容和评价诊所等等信息，基于评价人是否为当前登录用户的条件将会允许用户对评价进行删除和修改。点击修改后点击诊所将进入被评价诊所的主页。评价的删除需要访问服务端comment/delete/地址，调用comment.py文件中的delete函数删除评价表中的数据以此删除评价。

回复区包含多条回复，用户能够根据自己的喜好为评价点赞，为后续的用户提供参考。点击评论区以及回复区的个人头像能够进入用户的个人主页。

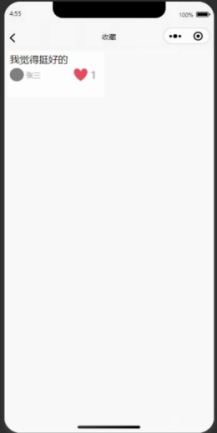
 

(a)主页 (b)评价内容

图 4.18 个人主页

个人主页包括个人名片以及评价两个部分。个人名片显示用户的昵称等信息；诊所评价中显示评价的对象、日期、星级以及具体内容，点击后进入评价的具体内容。用户通过个人主页能够进一步了解评价的用户，大大拓展了平台的社交属性。在mounted生命周期钩子函数中，用户端会访问服务端personality/initial/地址来获取用户对于不同预约的评价，通过调用服务端personality.py文件中的initial函数即可获取用户的评价。

评论收藏后能够在用户的评论界面中被查到，如下为收藏页面：

(a)收藏页面 (b)收藏内容

图 4.19 收藏页面

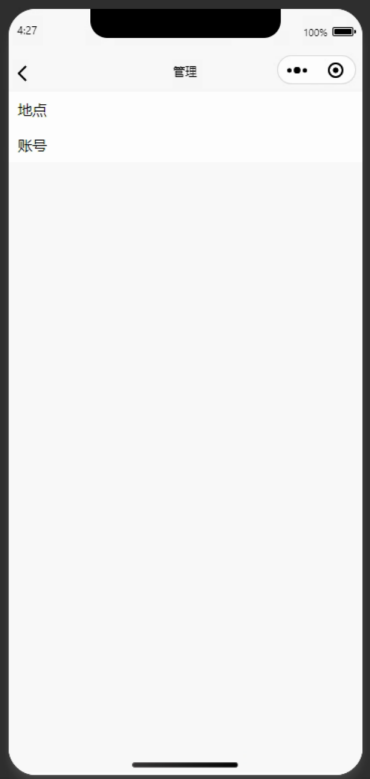
收藏页面将展示用户收藏的他人评价的内容，简略图中展示评价的内容、作者以及点赞的数量，点击进入后将展示评价的具体内容。用户能够将对自己有价值的评价放入收藏夹，方便日后查看。用户端将访问服务端star/get/地址来获取个人的收藏内容，通过调用start.py文件的get函数来访问收藏表中的数据来获取收藏的内容。

### 管理模板

高级管理员在前端能够实现部分平台管理的功能，将实现这部分功能的页面归入管理模板。

在个人页面中找到管理选项，即可进入管理页面。

如下显示管理页面：

(a)管理主页 (b)地区管理 (c)用户管理

图 4.20 管理主页

管理页面包含地点和账号两个选项，点击后分别进入地点管理页面以及账号管理页面。

地点管理页面如下：

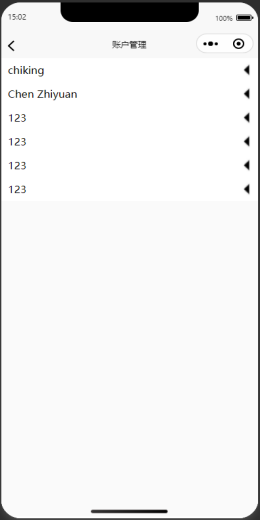
 

(a)县展示 (b)城市展示

图 4.21 地点管理页面

地点管理页面中罗列了平台中已经注册的城市，城市旁的左三角点击后下拉的输入框中展示了当前城市中已经被注册的县级行政区。在下方输入框中输入县级行政区的名字即可注册新的县级行政区，点击已经注册的县级行政区可以修改其名称，如果将县级行政区的名称清空则当前县级行政区将被删除。城市的增删改查需要访问服务端的location/city/add/，location/city/delete/, locaiton/city/modify/以及location/city/get/地址，调用city.py文件的add，delete，modify以及get函数修改城市表来完成任务；县的增删改查需要访问服务端的location/county/add/，location/county/delete/, locaiton/county/modify/以及location/county/get/地址，调用county.py文件的get，delete，add以及modify函数修改县表来完成任务。

账号管理页面如下：

(a)账号展示 (b)信息展示 (c)信息修改

图 4.22 账号管理页面

账号管理页面中罗列了平台中现有的账号，显示昵称作为账号的缩略信息，点击右侧的右三角能够展示账号的昵称、邮箱、密码、权限以及所属诊所、管辖诊所、预约记录等等，点击方框能够修改用户的昵称、密码以及管理员与普通用户的权限。修改昵称需要访问服务端的account/modify/nickname/地址，服务端程序将调用accountManage.py文件的modify\_nickname函数，修改账号表中的nickname字段来完成请求；修改密码需要访问服务端的account/modify/psword/地址，服务端程序将调用accountManage.py文件的modify\_psword函数，修改账号表中的psword字段来完成请求；修改权限需要访问服务端的account/modify/power/地址，服务端程序将调用accountManage.py文件的modify\_power函数，修改账号表中的power字段来完成请求。

## 医生界面

如图所示为医生界面首页，其包含两个板块：诊所展示以及挂号信息。

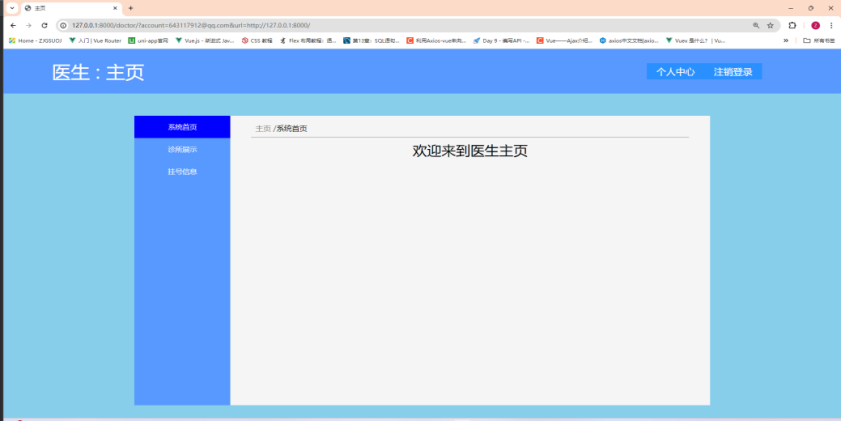


图 4.23 医生界面展示

### 诊所展示

诊所展示将展示现有的诊所并展示包括名称、城市、区级行政区以及位置信息的所有信息，医生能够在其中选择合适的诊所进行申请。



图 4.24 诊所展示界面

诊所申请需要访问服务端的doctor/application/，调用doctor.py文件的application函数在后台申请表中添加当前申请的信息。

### 挂号信息

挂号信息将展示病人的姓名、服务日期、时间段、服务内容。医生在完成服务点击完成预约列中的勾选项，预约表中对应的预约进度将向前推进，用户能够进入评价页面评价医生的服务。



图 4.25 诊所展示界面

预约完成需要访问服务端的doctor/finish，调用doctor.py文件的finish函数，修改预约表中对应项的阶段值。

## 诊所界面

诊所界面展示：



图 4.26 诊所界面展示

诊所界面主要包含四个板块：诊所医师、消息处理、出诊设置以及评论查看。

### 诊所医师



图 4.27 诊所界面展示

诊所医师将展示当前诊所中所属的医生名单并包含医生姓名、生日、学历等信息。诊所用户可以在该界面中直接修改医生的姓名、生日、学历、介绍等信息。点击右侧的删除列即可将该医生从诊所中删除。医生删除需要访问服务端的clinic/delete地址，调用clinic.py文件中的delete函数，访问诊所医生表中的对应选项并且删除该项同时查询预约表中该医生在该诊所的所有预约而后删除。

### 消息处理



图 4.28 消息处理界面

消息处理将展示医生对当前诊所的申请以及医生的姓名、生日、学历以及职称等信息，同意后申请的医生将成为诊所所属的医生。

诊所的同意将会访问服务端的clinic/agree/地址，调用clinic.py文件中的agree函数，访问后台信息表将对应的数据删除并访问诊所医生表，将申请医生加入被申请诊所。

### 出诊设置



图 4.29 消息处理界面

出诊设置将展示并设置医生在规定的时间段中出诊，通过设置出诊的起始时间与结束时间以及出诊时间间隔来设置出诊间隔。在下方的格子中填入出诊医生、出诊日期、出诊时间段来设定出诊信息，诊所用户可以点点击的删除按钮删除医生的出诊信息。诊所预约的添加以及删除需要访问服务端 的clinic/add/，clinic/delete/，clinic.py文件的add以及delete函数，需要访问预约表向其中增删数据。

### 评论查看



图 4.30 评论查看界面

评论查看界面能够查看普通用户对于诊所的评价，包含评价用户、被评价医生、被评价服务以及评价的星级和评价内容等属性。该界面在onload阶段会访问服务端的clinic/comment/地址，即clinic.py文件的comment函数，将会查询评价表中该诊所的所有评价。

## 管理员界面

管理员界面：



图 4.31 管理员界面展示

管理员界面中包括五个板块：诊所管理、医生管理、账户管理、地区挂历以及评论管理。

### 医生管理



图 4.32 医生管理界面

医生管理将展示平台上所有的医生，并且完成包括对医生名字、学历、职称以及简历等属性修改，以及对平台上现有医生增删的功能。点击删除框中的打勾符号即可删除对应的医生，该医生对应的后台以及应用程序中的账号也将删除。如果在下方方框中填入医生信息即可新建医生账户。点击界面中的方框即可编辑表格中的信息，点击页面中的空白区域即可保存。

医生的增删将访问服务端的administer/doctor/add/以及administer/doctor/delete/地址，调用administer.py文件中的doctor\_add以及doctor\_delete函数，访问医生表进行增删操作。

### 评论管理



图 4.33 评论管理界面

评论管理将显示平台上所有的评论的评论者、评论内容以及分数，点击评论内容和分数的方框将能够修改对应的内容，点击右侧删除列的勾将能够删除对应的评论。

评价删除、修改将访问服务端administer/comment/delete/，administer/comment/modify/地址，调用administer.py文件中的comment\_delete，comment\_modify函数，访问评价表删除或修改记录。

### 账户管理



图 4.34 账户管理界面

账户管理将展示平台上所有的账户以及账户的邮箱、昵称、姓名以及权限等级等信息。点击右侧删除列的勾选项即可删除对应的账号。点击昵称、姓名以及权限等级的具体内容即可修改，需要注意的是权限等级若为1或者2（医生和诊所用户）无法修改，只有0、3或者4才能修改（后台管理员用户能够向普通用户赋予管理员权力）。

账户信息的修改需要访问服务端administer/account/modify/地址，调用administer.py文件中的account\_modify函数修改账户表信息。

### 地区管理



图 4.35 地区管理界面

地区管理界面中将显示现有的县级行政区以及其所属的城市。点击右侧的删除按钮将删除数据库中的对应项。在下方的城市以及县一栏中输入地区名称可在数据库中创建对应的县，与此前没有出现过的城市将在城市表中创建对应的城市。

地区的增删改查将会访问服务端county/add/，county/delete/，county/moify/，city/add/地址，对城市表以及县表做出修改。

### 诊所管理



图 4.36 诊所管理界面

诊所管理包括对诊所地点、名称等属性的修改，对平台上诊所的增删同时也会展示平台上所有的诊所。

诊所的删除需要访问服务端administer/clinic/delete地址，调用administer.py文件中的clinic\_delete函数删除诊所表中对应的诊所。

# 总结与展望

## 总结

项目设立之初是为了填补市面上缺少专业牙科诊所推荐平台的空白，现有产品抑或是过于泛化抑或是缺少推荐功能，都不担当到这一重任。本本系统以预约功能为基础，完成了普通用户通过互联网获取牙科诊所服务的基本功能，同时通过医生查看、诊所查看等功能辅之以评价功能构成了基本的牙科诊所服务推荐功能为基础的预约功能提供赋能。

本项目采用的Django、SQLite以及uniapp等技术符合实际开发需要。其中Django自带的ORM系统允许开发者使用python语句操作数据库大大提高了编程开发的效率同时使用django自带的后台管理界面可以高效的管理数据；SQLite作为一种轻量级的数据库管理系统提供了完整的SQL功能，通过使用简单的命令和接口较好地嵌入了django系统之中；uniapp允许开发者编写一次代码而后能够发布到多个平台，这也给项目未来的发展提供想象的可能。

但在另一方面来看当前系统仍然存在一些问题：

1. 医生用户不能比较方便地查看自己的排班、评价等信息；
2. 前端界面相对比较简陋，需要后期不断更新；
3. 功能推荐功能较为单一，相较于更加专业的推荐系统仍有一段距离要走。

如此种种也给项目的未来发展留下想象空间。

## 展望

前文提到，项目依然存在部分信息查看不便、界面较为简陋以及推荐功能单一的问题在这方面的升级中我们仍然需要进一步的努力。

uniapp作为一个跨平台运行的框架允许用户将项目部署到不同的平台，未来也可持续推进项目将其部署到Android应用、浏览器等平台上以此拓宽牙科诊所平台的使用面，使得平台能够惠及更多的用户。

另一方面，在评价功能之上可以开发本地生活类服务应用，通过分享个人服务经历以及服务结果进一步拓展项目的功能以及应用场景，为普通大众提供更好的服务。同时为了方便网上问诊，亦可开发网上交流等功能，在其中加入语音、图片发送的内容将大大拓宽应用的功能。

# 参考文献

[1] 好牙医-牙医助理APP[EB/OL]. https://www.51haoyayi.com/.

[2] 牙e家口腔医生[EB/OL]. http://www.kqcn.com/.

[3] 李莉. 大数据环境下个人数字档案的分类管理与加密防护研究[J/OL]. 档案管理, 2021(6): 57-58. DOI:10.15950/j.cnki.1005-9458.2021.06.021.

[4] DentaLink牙医管家[EB/OL]. https://dental360.cn/.

[5] DentalMonitoring[EB/OL]. https://dentalmonitoring.com/.

[6] 张蒂, 刘春江, 谢靖, 等. 移动智能知识服务研究与实践——以慧科研微信小程序为例[J/OL]. 图书情报工作, 2023, 67(24): 44-53. DOI:10.13266/j.issn.0252-3116.2023.24.005.

[7] HUANG T, LIN H, YIN X, 等. Research on Architecture Design of E-commerce Mini-program and its application in WeChat[C/OL]//2023 International Conference on Intelligent Computing and Next Generation Networks（ICNGN). 2023: 1-6[2024-05-15]. https://ieeexplore.ieee.org/abstract/document/10396774. DOI:10.1109/ICNGN59831.2023.10396774.

[8] VAMSI C, PRIYA M L, SAI C H, 等. Design Your Career: Build Future with Job Portal Website Developed using Django Framework[J]. 27(4): 132-136.

[9] LINDSTROM D K. Microsoft Access to SQLite Database Migration for Existing Forest Management Software: Methods and Considerations[J/OL]. 2024[2024-05-15]. https://pdxscholar.library.pdx.edu/geog\_master\_GIS\_reports/5/.

[10] FILIPOVA O. Learning Vue.js 2[M]. Packt Publishing Ltd, 2016.

[11] CHANG Y, OU S, JIA Q, 等. Design and Implementation of Django-based Quality Traceability System for Medicinal Dendrobium[M/OL] https://www.atlantis-press.com/doi/10.2991/978-94-6463-304-7\_65. DOI:10.2991/978-94-6463-304-7\_65.

[12] 胡亨汶, 孟祥印, 李丹, 等. 基于RESTful Web Services的云边数据交换设计与实现[J/OL]. 现代制造工程, 2022(8): 25-32. DOI:10.16731/j.cnki.1671-3133.2022.08.004.

[13] 陈敏灵, 路子盈, 薛静. 基于微信小程序开发的高校校车订座系统的设计与实现[J]. 智能计算机与应用, 2023, 13(7): 155-162.

# 致 谢

本文的结束随即意味着本人大学生涯最后的终点，虽然过程多少有些艰辛曲折但最后还是顺利完成——大结局的反派头子往往让主角抓耳挠腮。

在我大一刚进入浙工商时候并没有多少对大学生涯的想象，这或许是另一场高中的起点。幸而加入了ACM集训队，参加了ACM竞赛，在这里我找到了自己所爱的以及愿意付出的东西。

首先感谢汤英杰同学，作为亦师亦友的角色，三年多来在与汤英杰同学的来往中我学习到许多。其次是所有ACM历届的学长学姐以及指导老师们，从无到有成立一个伟大的集训队并非一件简单的事情，是十多年来一届又一届的历届学长学姐们锲而不舍的接力才使得20届的同学们能够加入一个伟大的集训队。

感谢朱云芳教授在毕业设计制作过程中的悉心教导，在制作毕业设计的初期，我并不擅长系统开发，中间出了许多问题，但朱云芳教授总是帮我出谋划策，耐心地作出指导，我深受感动。

最后还是感谢我自己，在我刚来到大学的时候并不是一个优秀的人，但我并不放弃，四年以来我不断坚持探索，用自己认为正确的方式生活工作，在实践中不断探索自己，有了巨大的进步。毕业后我将继续自己考研的任务，希望一帆风顺。