上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

数据库系统课程设计

DATABASE SYSTEM PROJECT

“饿了吗”在线订餐平台设计

ELEMA ONLINE FOOD DELIVERY SYSTEM

课题报告

PROJECT REPORT

NAME: 秦格华 罗 晶

Student ID: 5140219335 5140219349

Instructor: 李 芳

Contents

一、项目简介

1. 课题介绍

本系统采用关系型数据库模式，基于 Apache、PHP 和 MySQL 的开发环境设计了“饿了吗”在线订餐平台。整个平台的设计思路主要借鉴了现有的网络订餐平台“饿了么”的流程设计和外观，为用户在该平台上的注册登录、查询餐厅和订餐提供服务，同时为餐厅经营者提供管理菜单、接受订单和管理订单状态等服务。数据库系统大致上由用户、餐厅和订单三部分组成，每部分都尽量涵盖了完成操作所需要的详细信息。同时我们借助已有服务提供商的 API 接口，实现了基于高德地图的餐饮查询系统，并利用已有 twitter 设计模板美化了用户界面，通过比较合理的操作逻辑完成了网站页面的设计。

2. 相关文档

[1]《A First Course in Database Systems》 机械工业出版社

[2] php 在线教程 W3SCHOOL

[3] CSS 在线教程 W3SCHOOL

[4] jQuery 在线教程 W3SCHOOL

[5] bootstrap 官方文档 GetBootstrap

二、系统架构

1. 基本结构

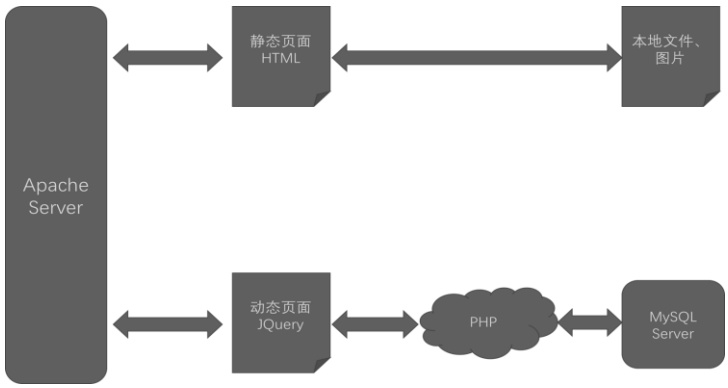


图 1. 平台基本结构

我们平台系统的基本结构如上图所示。

服务器上部署的 MySQL 服务器通过封装好的用户界面操作平台与 PHP 脚本进行连接，我们这里直接利用了一个 AMPPS 集成环境下的现有平台：PHPMyAdmin。作为一个可视化工具，它可以更好地进行对数据库的基本操作（如修改用户权限，监控数据库状态等），同时进行访问和控制。

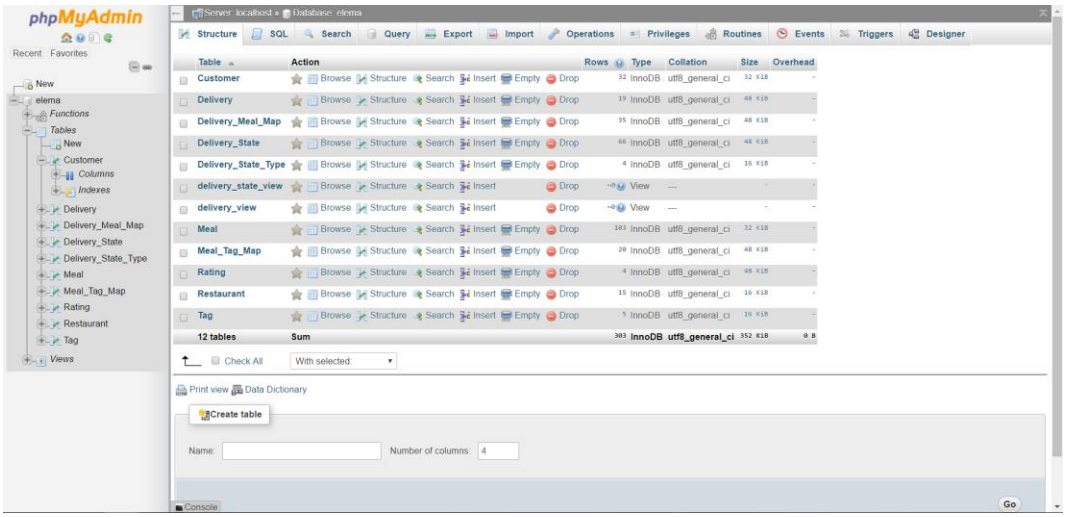


图 2. PHPMyAdmin 界面

为了建立更友好的交互界面，我们在静态 HTML 之外，利用 JQuery 与 PHP 连接完成了动态页面的设计。用户在动态页面中完成的动作（填写表单、点击按钮等）由 JQuery 执行后发送给 PHP 进行数据处理，最终将数据写回数据库。

所以总结来说我们的系统主要分为 3 部分：

a) 数据库层，存放了用户信息、餐厅信息、菜品信息、外卖信息和配送状态的具体数据；

b) 数据服务层，用于负责处理用户请求，与数据库进行连接，并接受上层发来的结果对数据库进行更新或修改；

c) 用户图形界面层，用户或餐厅管理员通过图形界面发起相应的指令请求，并接收来自下面一层返回的信息。

依据上面的基本架构，我们的编程工作主要包含了三部分内容：数据库的编写（SQL），数据服务层通信脚本语言代码的编写（PHP），用户图形界面层脚本语言的编写（HTML+CSS+JQuery）。

2. 特殊结构

本系统借鉴了“饿了么”平台的餐厅选择理念，通过调用开放的高德地图 API 对搜索地点周围一定距离内的餐厅进行显示，以保证搜索到的餐厅可以满足餐厅属性中有关配送距离的要求。搜索调用 API 的具体实现在目录`./js/index.js` 文件中。基本思路是首先利用高德地图返回的地点的经纬度坐标，将这一坐标交付给 PHP 脚本`./api/customer/delievery.php`，脚本调取数据库中满足配送距离的所有餐厅后发送给 JS 脚本，再通过另外一个 JS 脚本将所有的餐厅信息按照一定的顺序列举出来。

3. 系统 E/R 模型设计

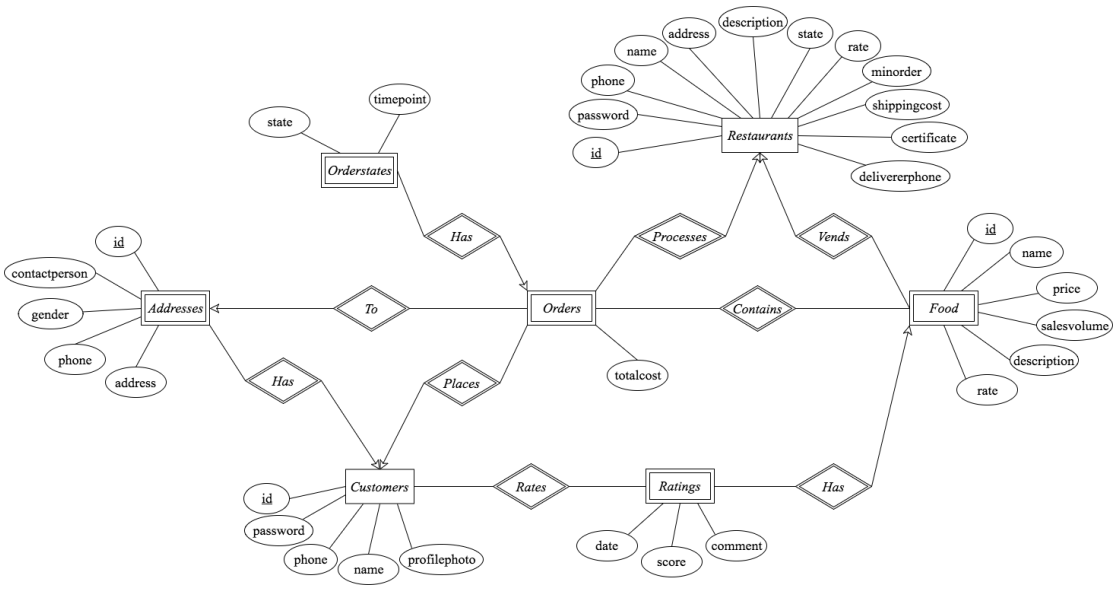


图 3. 系统 E/R 模型设计

三、数据库模式

1. 用户信息

Customer（用户信息表）的主键由唯一指定的用户 ID：usr_id 构成，同时用户拥有自己的密码，手机号码，用户昵称，另外用户的微信号和邮箱作为可空属性。

在与其他 schema 的关联上，Customer 与 Delivery 是一对多的关系，代表一个用户可以点多单外卖，之间的虚线表示并非强连接，即 Delivery 不一定要必须相应的 Customer 存在。另外一个 Customer 与 Rating（评分表）

也是一对多的关系，代表一个用户可做出多项评分。

2. 餐厅信息

Restaurant（餐厅信息表）类似地以餐馆 ID：restaurant_id 作为唯一主键，同时一个餐厅拥有名称、配送范围、城市、街道、街道号、订单起送价格和用于管理自身数据库的密码，还有一个管理用户管理的 rate 属性作为餐馆的评分。

在与其他 schema 的关联上，一个餐馆可以对应多份 Delivery，一个餐馆可以拥有很多种类 Food。

3. 菜品信息

Food（食物信息表）在拥有 food_id 作为主键的同时，还有关联的餐馆的 restaurant_id 作为外键。Food 自身有食物名称、价格、卖出份数、评分和可选的描述信息属性。

在与其他 schema 的关联上，一个 Food 可以显示在很多个。订单的的列表 Delivery_Food_List 中，同时也唯一属于一个 Restaurant，在评分 Rating 中一种食物可以接受来自多个用户的评分，所以也是一对多的关系。

4. 外卖订单信息

Delivery（外卖信息表）代表了一份外卖，通常是由一个用户向一个餐馆发起的含有多份食物的订单，所以其中包含了用户 ID 和餐馆 ID 作为外键，同时每一份外卖有一个自动生成的 delivery_id 作为主键。注意到我们并没有在用户的信息中加入地理信息，而是将其作为属性放在了外卖信息表中，这代表了一个用户不一定只能在同一个地方点外卖。同时一份外卖会包含一个 total_price 作为订单价格。特别地，我们加入了配送员的概念，每一份外卖都有一个唯一的配送员，所以也将其作为了外键。

在与其他 schema 的联系上，首先一个用户或一个餐馆都可以有多个 delivery，另外每一个配送员会负责多份订单的配送。

同时我们需要记录外卖的配送状态，因为配送状态分为“已接单”，“制作中”，“配送中”，“已接单”和“订单完成”五个状态，如果都在 Delivery 中更新会导致冗余，所以我们将配送状态作为了一个单独的 schema，将对应的 delivery_id 作为外键，同时有 state 表示状态，time_point 用于记录时间

戳。为了使顾客可见这一状态，我们创建了一个 View：Customer2Delivery 来为顾客查看订单状态提供视图。

更为重要的是，一份 Delivery 中拥有多份食物，同样地为了减小冗余，我们加入了新的 schema：Delivery_Food_List 用于标识一个订单内的食物类型，所以也是一个一对多的关系。

5. 评分信息

这是我们系统额外的一部分，用于记录用户对每一种食物做出的评价。由于一个用户可以做出多份评价且一种食物可以接受多份评价，所以都是多对一的关系。同时 Rating 有时间戳和 rate 作为必需属性，而 comment 作为文字内容是可选属性。

6. 各类图关系表

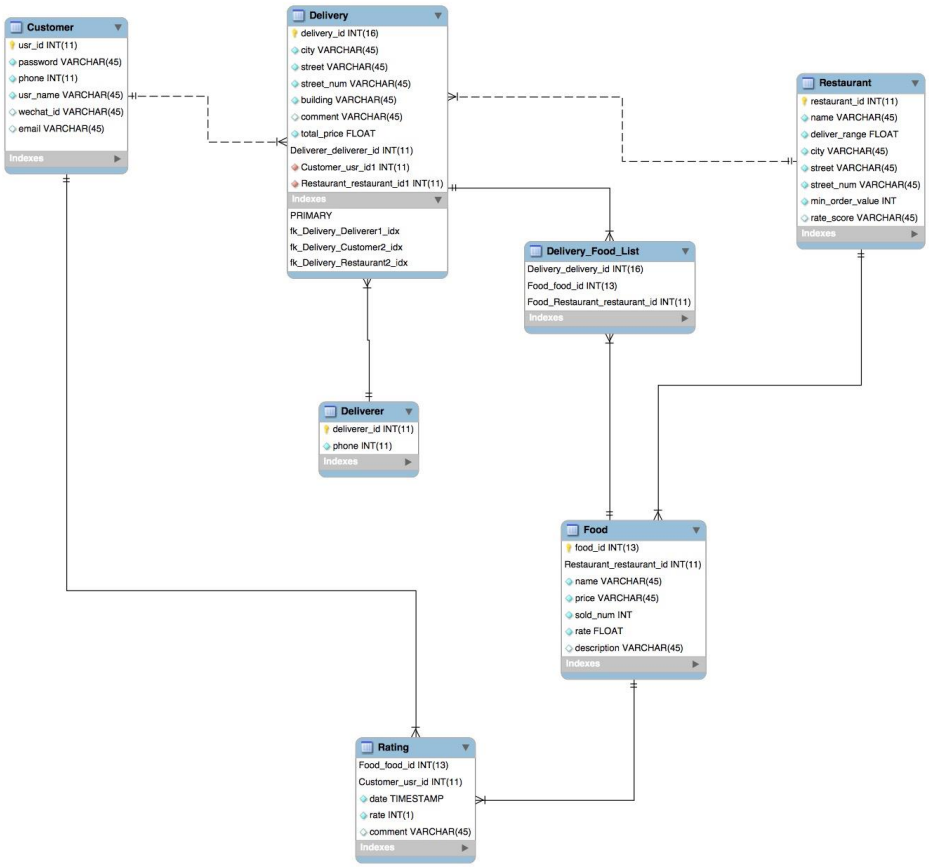


图 4. 数据库的 EER 模型

7. 限制条件（Constraints）

a) 订单里的某一个种类菜品的数量一定是大于零的；

b) 每一个菜品的价格一定是大于等于零的；

c) 一个菜品的 rate 是在 0~5 之间的数字；

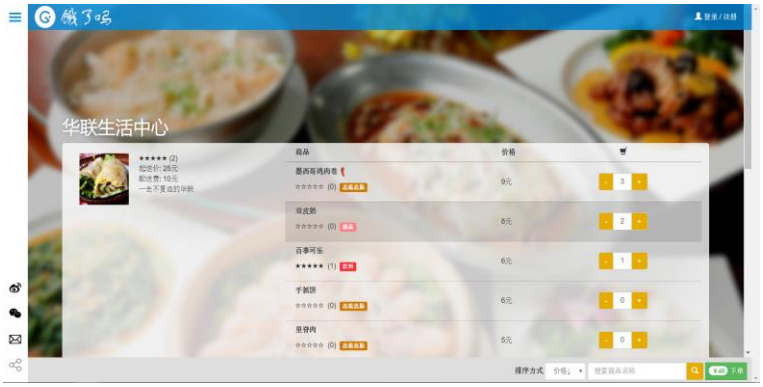
d) 一个餐馆自身的配送费价格大于等于零、最小订单价格大于等于零、最大配送范围大于等于零、经纬度大于等于零。

四、用户界面

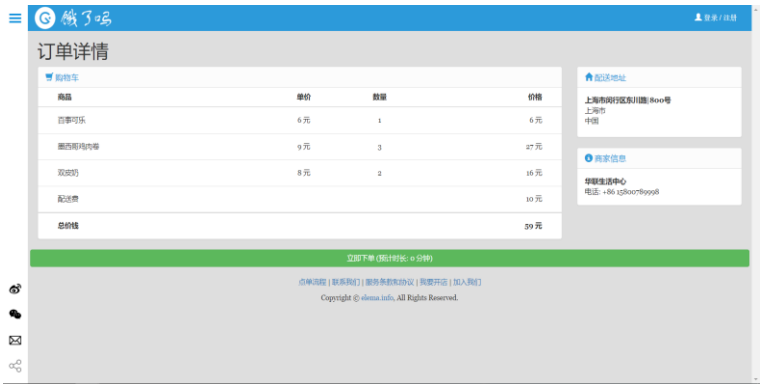
1. 网站首页

2. 餐馆选择页面

3. 菜品选择页面



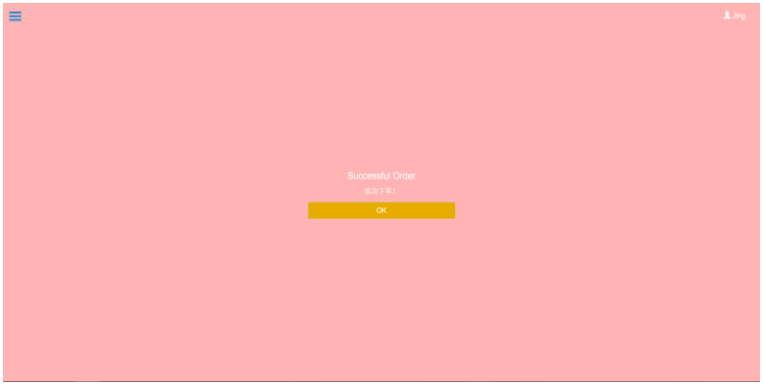
4. 订单确认页面



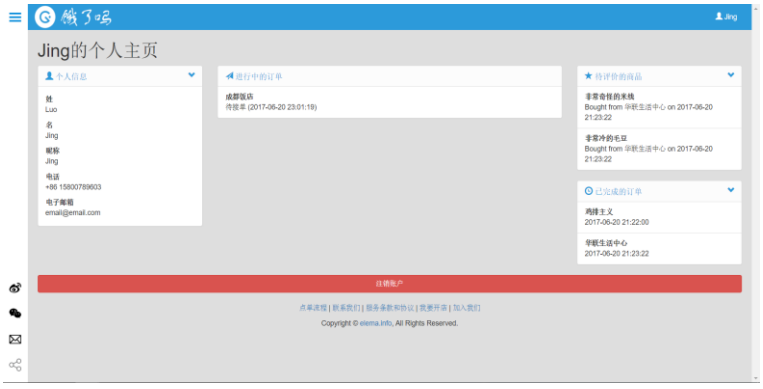
5. 用户注册页面

6. 用户登录页面

7. 下单成功页面

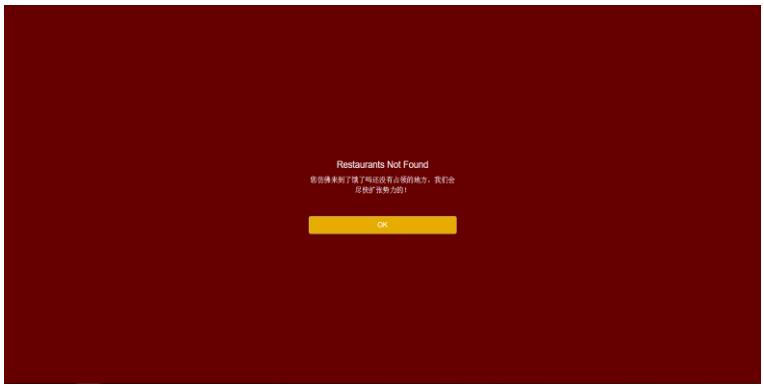


8. 个人信息页面



9. 未输入具体地址时报错页面

10. 未查询到具体餐馆时报错页面



11. 帮助页面

12. 流程页面

五、事务流程

1. 用户订餐流程

a) 用户访问首页，输入自己当前所在位置，或者选择快速定位栏中推荐位置；

b) 调用高德地图 API 对输入的位置进行查找，获得该位置的经纬度信息，再调取数据库中满足距离范围要求的餐馆，在餐馆选择页进行挑选餐馆；

c) 用户进入餐馆菜单目录，进行菜品挑选，此时订单总价需要满足最低起送价格；

d) 页面跳转至确认页面，确认后若用户未登陆则提示进行登陆；

e) 用户输入用户名和密码后登陆；

f) 登陆成功后页面返回确认页面，确认后页面提示提交成功，并返回到用户个人信息页面显示订单信息。

2. 用户注册流程

a) 用户访问首页，搜索位置后进入餐馆页面，点击右上角登陆/注册按钮进行注册；

b) 输入用户的个人信息，当个人各项信息满足后页面提示创建账户成功。

3. 用户评价流程

a) 用户进入个人信息页，页面右侧显示已评价订单和待评价订单；

b) 用户选择任一待评价订单，在跳出的订单评价表中选择评分，并依情况加入一定的 comment 内容。

4. 餐馆处理订单流程

a) 餐馆登陆管理页面；

b) 餐馆在“待接收的订单”中查看用户提交的订单，选择“接收”则代表餐馆已接单；

c) 餐馆在“制作中的订单中”查看已经在制作的订单，制作完成后选择已完成则代表进入配送状态；

d) 餐馆在“配送中的订单”中查看正在配送的订单，选择“完成”则代表用户已接单；该订单完成；

e) 餐馆在“已完成的订单”中查看已经完成的订单信息。

5. 餐馆管理菜品

a) 餐馆登陆管理页面；

b) 餐馆在“菜单管理”中查看已有的菜品信息，点开菜品名以查看具体的菜品信息或修改；

c) 餐馆在“菜单管理”中选择添加菜品，输入新加入的菜品信息后自动返回菜单管理页面，可以查看新加入的菜品信息。

6. 餐馆管理信息

a) 餐馆登陆管理页面；

b) 餐馆在“餐馆信息”中查看已有的餐馆信息，并对需要修改的具体信息进行修改；

c) 修改完成后点击保存，则页面返回管理页面，并可再次查看新的餐馆信息。

六、项目总结

1. 收获与心得

经过数周围绕系统设计的讨论和两周时间的集中开发，我们的系统终于在答辩验收的前一天完成。整个过程虽然辛苦，并在期末考试周继续熬夜开发的压力下也有过不少苦衷，但是经过努力和不断地尝试，总体来说我们已经尽最大能力完善了这一个系统。

对一个问题的分析，到对应数据库的建模分析和用例场景分析我们都有了更为贴近现实的斟酌。在前期大量的时间中我们反复讨论数据库本身的设计对系统整体功能的影响，而且在此过程中我们多次询问老师意见，收获了很多有用的建议。在数据库编写过程中，我们对 SQL 语言有了更为深入的体会与了解，同时通过实践也加深了我们对数据库这门课的知识掌握；

我们利用自身已有的前端和后端代码编写经验对系统的用户界面和控制逻辑进行了优化，使得系统尽可能用户友好，同时也使我们对前后端语言有了更为熟练的掌握；

收获了对工程编码规范的认识。我们在前期代码编写中，因为是两个人分开进行，所以在代码合并时出现了很多代码一致性上面的麻烦。经过讨论后我们决定先进行更为合理的分工，同时对整体工程的环境、编码方式和组织结构进行统一规划；

在整个项目进行过程中我们进行了大量的学习，通过网络资源对解决问题的办法进行搜索，加强了独立解决问题的能力

2. 问题与不足

由于开发时间不足，后期我们没有办法实现我们在一开始规划的很多功能。如允许每个用户保留多个地址，因为这样做的话地址既与用户多对一关联，同时要与订单关联，会造成逻辑上的复杂性。我们退而求其次，通过把地址信息从用户属性中剔除，将其只与订单本身一对一关联。再比如我们也曾想过为每个配送员再添加一个控制界面，这样每个配送员可以对配送中的订单进行处理，与现实中的场景更为接近。同样因为开发时间原因我们没有能完成这一部分的工作。

配置环境的过程中我们发现 MySQL 对中文的支持性不好，虽然表面上支持 gbk 和 utf8 的编码格式，但是在配置过程中我们屡次出现了中文在

页面上显示为乱码的情形。我们花了大量的时间在解决这一问题上，并且网络上因为个人配置的环境变量都有差异的原因解决的方法众说纷纭。最终我们的系统在 sql 每次重启时仍会出现乱码的问题（当然英文编码没有问题）。而这一问题我们还有待解决。

由于数据来源匮乏且获取方法较为繁琐，我们只在某些地点插入了餐厅的数据，也就是说我们系统目前支持的搜索范围还很小，还没有在更大地理范围上进行过搜索餐厅的测试。并且由于搜索是调用高德地图 API 的原因，在返回多个地址时我们的系统会出现比较奇怪的错误（如明明搜索位置附近预设了餐馆，但是系统无法显示，因为搜索返回的是另一处位置的经纬度坐标）。

系统设计的一开始我们希望加入短信验证注册的功能，并且也获取了阿里大于的短信验证码 API 许可。但是后来发现这一 API 无法在本地主机上调用，也就是说只有在拥有域名的情况下该 API 才可以被正常调用。我们为了解决这一问题特地去租借了云服务器和域名，但是因为备案时间过长最终无法实现这一功能。