饕餮食客微信小程序项目文档

2020年6月15日

目录

[1.项目背景 3](#_Toc42543158)

[2.项目需求 4](#_Toc42543159)

[2.1.项目目标 4](#_Toc42543160)

[2.2.项目适用对象 4](#_Toc42543161)

[2.3.解决的实际问题 4](#_Toc42543162)

[2.4.项目特色 4](#_Toc42543163)

[2.4.1.个性化 4](#_Toc42543164)

[2.4.2.易推广 4](#_Toc42543165)

[2.4.3.轻量级 4](#_Toc42543166)

[3.项目功能介绍 5](#_Toc42543167)

[3.1.总体功能模块介绍 5](#_Toc42543168)

[3.1.1.登录模块 5](#_Toc42543169)

[3.1.2.打卡模块 6](#_Toc42543170)

[3.1.3.个人信息模块 7](#_Toc42543171)

[3.1.4.餐厅列表模块 7](#_Toc42543172)

[3.2.详细功能介绍 7](#_Toc42543173)

[4.技术开发方案 9](#_Toc42543174)

[4.1.项目总体开发方案 9](#_Toc42543175)

[4.2.项目前端开发方案 9](#_Toc42543176)

[4.3.项目后端开发方案 10](#_Toc42543177)

[4.3.1.自定义后台 10](#_Toc42543178)

[4.3.2.官方后台 10](#_Toc42543179)

[4.3.3.Redis缓存服务器 10](#_Toc42543180)

[4.3.4.MySQL数据库服务器 11](#_Toc42543181)

[4.4.项目服务器开发方案 11](#_Toc42543182)

[4.5.项目数据库开发方案 12](#_Toc42543183)

[4.6.遇到的问题和优化方案 13](#_Toc42543184)

[5.团队组成和分工 14](#_Toc42543185)

1.项目背景

越来越多的小吃店琳琅满目，越来越多的大餐厅鳞次栉比，这让许多人都面对丰富多样的选择不知所措。所以也就希望有这样一款app能够根据其他用户的介绍、用户自己选择的口味、用户自己所处的位置进行推荐，帮助用户的个性化选择。虽然不一定是最好的，但一定需要是最适合用户的。

目前在餐饮行业的app不在少数，美团、饿了么、云狗宅配、大众点评等等应有尽有，互相之间有功能大量重复的，也有完全不同的。这些餐饮app已经使得餐饮行业日渐成熟。但是以大众点评为代表的众多平台，从运营方式的角度看，主要以餐厅本身为主题，罗列了餐厅的许多特点和菜品。虽然整体突出了客户对餐厅的选择，但是参差不齐的餐厅依然让用户不知所措。数百万的餐厅，上亿的菜品和评论，无论是谁都会有选择的困难，即使有评论作为参考也无法逐一查看；而从技术实现细节的角度看，这些app仅仅只是通过用户自己所处的位置和其他用户的评价进行简单筛选和排列，对于每一个用户依然没有个性化，在用户面对众多选择时依然还是会遇到困难。故在针对每一位用户的个性化定制服务上依然还是有一定的局限性。

2.项目需求

2.1.项目目标

本项目旨在使用推荐系统针对每一个用户的口味选择、点赞情况和地理位置进行推荐，帮助用户进行选择。

2.2.项目适用对象

2.1.1.有选择困难症的用户

2.1.2.没有选择困难症，只是看到诱人的就希望尝试的用户

2.1.3.没有太多时间根据自己的喜好选择的用户

2.1.4.希望经常了解目前最热门的餐饮，紧跟潮流的用户

2.1.5.在遇到中意的餐厅或者菜品后迫切的希望能够告诉所有人的用户

2.1.6.喜欢探索并大胆尝试冷门美食的用户

2.1.7.职业美食评论家

2.1.8.业余美食爱好者

2.3.解决的实际问题

本项目在运营上没有太大的改变，依然是突出了客户对餐厅的选择。而在技术实现上，本项目重点突出了用户的个性化，即用户除了可以根据自己的所在位置之外，还能够通过自己设定自己喜欢的口味以便系统根据口味推荐；同时系统还会收集用户对其他用户打卡的点赞情况来推荐。

另外，近期对于推荐系统有着很大的争议，即在推荐的过程中将会逐渐缩小用户的圈子，从而使得用户能够看到的餐厅越来越单一。这方面本项目也进行了考虑，即用户可以选择性地清空以前所有的推荐数据，从而让推荐系统从新开始推荐。这样使得用户可以快速跳出原有圈子的束缚，看到全新的餐厅和菜品，选择也更为广阔。

2.4.项目特色

2.4.1.个性化

使用推荐算法，收集用户根据自己的喜好选择的口味标签、用户对其他用户的打卡的点赞情况和用户所在的地理位置进行推荐。针对每个人，不同的口味选择能够生成完全不同的餐厅列表，是完全个性化的选择。

2.4.2.易推广

微信早已深入我们的日常生活，交流、支付等等全部离不开微信。而微信小程序依赖微信平台，这使得本项目在推广上不需要什么成本，非常容易就能分享到世界的每一个角落。

2.4.3.轻量级

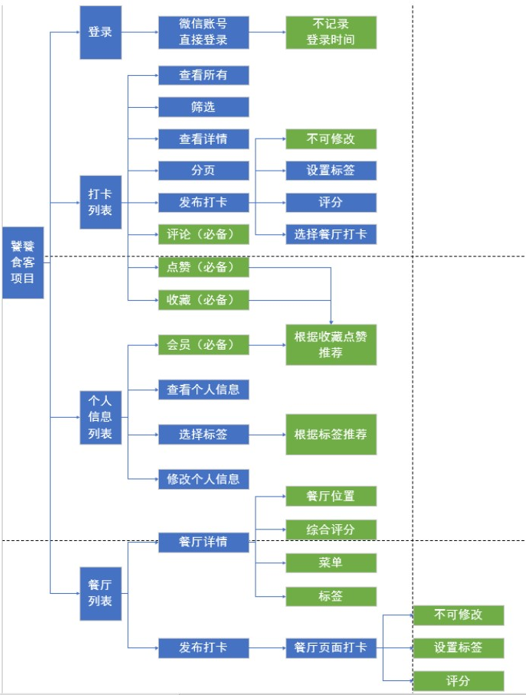
比起手机应用，小程序最大的特点便是轻量级。整个应用加起来也在2M以内，就算全部下载也非常迅速，保证了启动速度和响应时间。

3.项目功能介绍

3.1.总体功能模块介绍

本项目一共分为登录、打卡、个人信息、餐厅四大模块，不同的模块对应不同的业务。其中登录模块直接使用微信账号进行登录，免去了用户注册的烦恼；其次打卡能够让用户更好地将自己的经验展示给所有有可能看到该餐厅信息的其他用户参考，包含文字、图片、视频信息，并且其他用户可以通过评论吐槽、询问，也可以点赞种草等等；个人信息模块也作“用户模块”，不仅记录了微信账号中所有的信息，也包含了会员等级和三个口味喜好的标签，并且还会记录用户的点赞情况和打卡情况；最后的餐厅列表中会详细列出所有的餐厅，并且在餐厅中显示所有对于用户来说必要的信息；同时，为了方便用户打卡，餐厅内就设立了打卡按钮。详细情况如图3.1所示。

图3.1 项目模块功能图

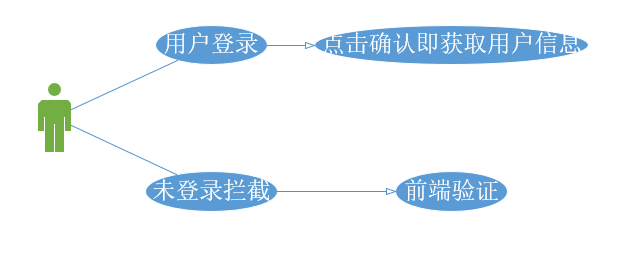


3.1.1.登录模块

登录模块主要功能是验证用户的微信账号，并使用微信账号记录用户产生的数据，以供推荐系统参考，再根据这些参考计算并推送用户最可能喜欢的餐厅。如果没有用户信息，所有的信息将无法确定保存位置。所以当用户没有登录时，由前端优先确认用户信息。如果用户信息不存在，则无法进行下一步的任何操作；如果用户在线并且信息完整，系统就能很好的保存用户数据并进行计算，推荐用户最可能喜欢的餐厅。

该模块的具体活动图如图3.2所示。

图3.2 登录模块活动图



在这个模块中，用户在小程序端中发起登录，经由官方封装的login方法获得code，并发送到自定义的后台中，再由后台发起请求，查询官方数据库中的用户信息，最终向小程序端返回自定义的登录状态码。数据流图如图3.3所示。

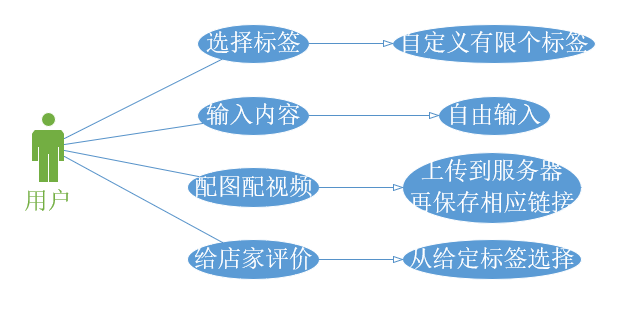
图3.3 登录模块数据流图



3.1.2.打卡模块

用户在本次用餐结束后任意时间可以对该餐厅进行评论，这些评论可以附上用户自己选择的照片、视频等。以后某个用户看到该餐厅时就能够根据现有的评论判断这个餐厅是否适合自己。具体活动图如图3.4所示。

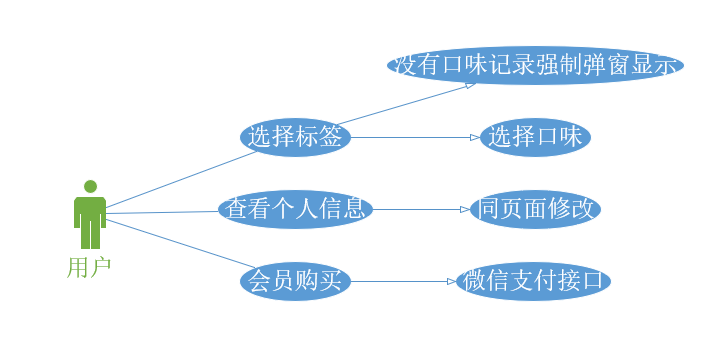
图3.4 打卡模块活动图



3.1.3.个人信息模块

也可以称作“用户模块”。个人信息中不仅包含了用户的微信账号，还有用户的点赞、自定义口味、餐厅收藏和会员信息。如果是会员则可以享受推荐算法帮助选择的服务，而普通用户只能看到类似大众点评一样简单罗列所有选项的界面。具体活动图如图3.5所示。

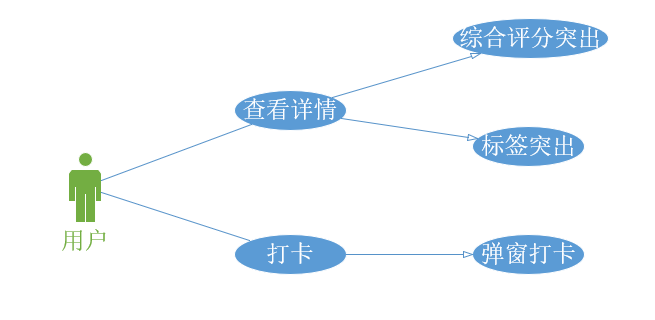
图3.5 个人信息模块活动图



3.1.4.餐厅列表模块

餐厅列表能够显示所有用户可以选择的餐厅。而如果是购买了会员的用户则会显示经过推荐算法个性化定制后的所有餐厅，即最符合用户口味的部分餐厅，其余的不相关餐厅都被筛选掉不再显示了，在一定程度上帮助用户进行了选择。具体活动图如图3.6所示。

图3.6 餐厅列表模块活动图



3.2.详细功能介绍

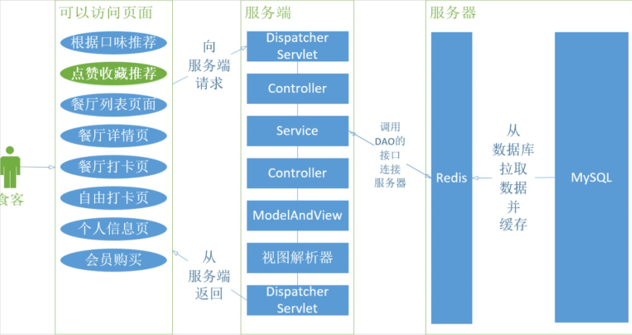
提示：从进入主页开始一步一步介绍那个按钮会实现什么功能，主要说明干什么，而不是技术点，需要截图，并配以文字说明。群内的pdf文档中有一个版本，但是太少了，需要丰富一下。

4.技术开发方案

4.1.项目总体开发方案

项目主要使用SpringBoot、MyBatisPlus、Redis、Nginx、微信小程序进行开发。整体架构如图4.1所示。

图4.1 项目架构图

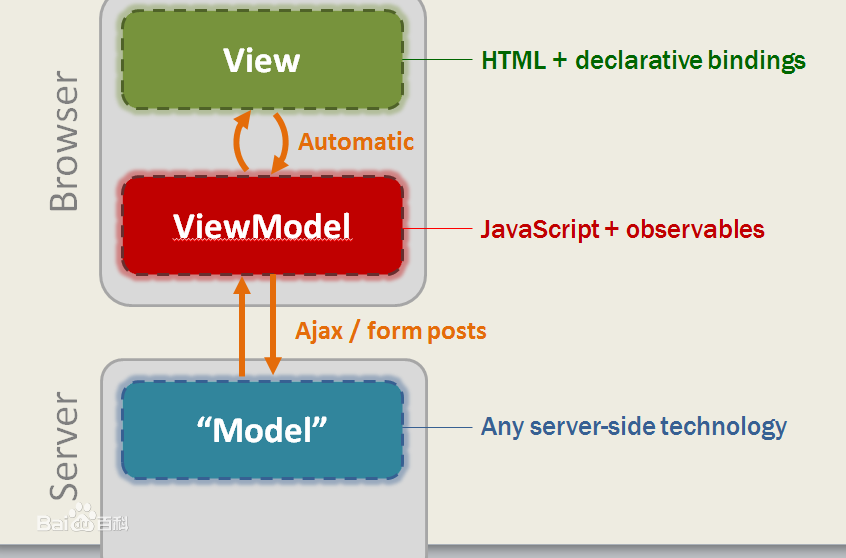


如图4.1所示，微信小程序主要作为前端页面，是独立于后台的前端项目。而后台、Redis和MySQL数据库则合为一个整个后台项目，是独立于前端的项目。两者之间通过HTTP请求交互，并且后台项目使用Nginx进行动静分离的部署，减小服务器的压力；同时Redis作为缓存服务器，与MySQL数据库服务器相独立，也分担了数据库的一部分压力。服务端和服务器由MyBatisPlus框架和Jedis进行交互，封装在DAO层中。

4.2.项目前端开发方案

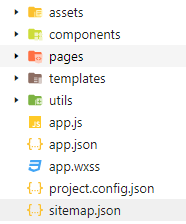
微信小程序是使用了MVVM架构的Web框架，即Model-View-ViewModel三层架构。其中ViewModel层通过数据的双向绑定，监听Model层对应数据的变化，实现了框架的响应式布局和数据的实时渲染。MVVM架构如图4.2所示。

图4.2 MVVM架构



由于MVVM框架的特点和模块化的思想，一整个微信小程序能够划分为不同的模块，每个页面模块对应一个page，每个page中包含独立的wxml网页文件、js脚本文件、json配置文件、wxss样式文件，分别进行相对独立的开发，既方便了多人多任务的开发，也方便了工程的解耦和独立测试。同样的，模块也可以有component和template，实现了H5无法实现的模板页，增加了复用性，减少了开发上时间和精力的浪费。项目文件结构如图4.3所示。

图4.3 项目文件结构



除了框架，还有微信官方封装的一些小工具，比如tabbar、navigator等构成边界的页面元素，还有createAnimation实现元素的动画效果等等。

4.3.项目后端开发方案

后端分为四个部分，一个是根据需求自定义开发并部署在云服务器上的后台，另一个是微信官方后台，还有一个是Redis缓存服务器，最后一个是MySQL数据库服务器。

4.3.1.自定义后台

自定义后台主要使用了SpringBoot和MyBatisPlus两个框架，并引入RESTful规范。

SpringBoot框架作为快速开发框架能够帮助快速搭建并开发项目，其中内置了非常多的默认配置且集成了Tomcat服务器，相较于SSM等框架方便了许多。同时，该框架沿用了Spring对依赖和变量的管理、SpringMVC强大的事务管理能力和面向切面的编程思想，进一步简化了一些特殊业务的编写，在方便的同时依然保持了非常高的自由度。

MyBatisPlus则是MyBatis框架的升级版，却保留了完整的MyBatis框架的功能，内置一般性的增删改查；为了进一步简化业务的处理，也可以使用MyBatis框架的注解查询视图，将繁琐的联合查询交由数据库强大的事务处理能力，并且保证了查询的便捷性。

RESTful规范本身不是框架，只是一种约定俗成的规范，帮助我们定义简单易懂的接口，并且能够让前后端依照一种通用规范开发，简化了项目前期对接口的规定、设计与统一。

4.3.2.官方后台

官方后台保存了完整的用户信息，为了安全，只能使用code进行验证，也就是官方规定的token。由小程序提交code到自定义后台，自定义后台向官方后台发起请求，拿取token验证后换取自定义的登录状态。

Token作为一个字符串，封装了设备信息、IP信息和一些用户的自定义字段。而为每一个用户的每一台设备维护一个Session是非常消耗资源的，所以每次请求时在请求头中加入token，每次时请求验证身份和权限，也极大地减小了单位时间内服务器内存的消耗，从而能够让服务器做更多的事。

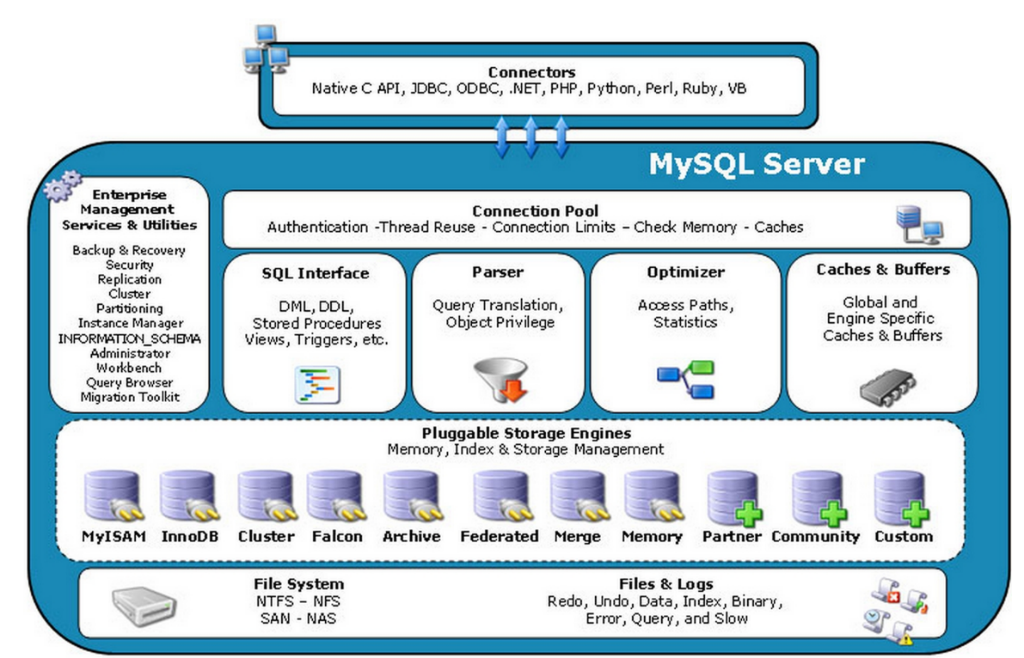
4.3.3.Redis缓存服务器

Redis本身作为缓存，可以像MongoDB一样存储数据，也能够像cache一样缓解请求发送的速度和数据库查询的速度差。同时，完全基于内存的Redis也能够让缓存表现远高于数据库查询时的IO速度；另外，多路IO复用也加大了服务器CPU的使用效率，也使用多线程技术提高了并发处理的能力；与此同时，Redis集群也能够帮助负载均衡，进一步提高并发处理的能力。

4.3.4.MySQL数据库服务器

MySQL服务器主要使用B+树作为存储结构，其中非叶子节点并不占用实际的内存，从而减少了查询时的IO次数，也就减少了查询过程中时间的消耗；索引结构使用Hash查找，通过直接取值并计算出该节点的物理地址，进一步减少了轮询的时间，最好的情况下可以将时间复杂度将为O(1)；数据库引擎使用InnoDB，其中行锁的机制能够让数据库在单位时间内处理更多的事务，对主外键的支持也更好的实现了各种业务的需求。MySQL内部构造详情如图4.4所示。

图4.4 MySQL内部构造详解



4.4.项目服务器开发方案

由于项目需求中不仅有基本数据的交互，也有媒体资源的交互。而媒体资源往往比基本数据资源要消耗更多的代价，这对于本来就能力有限的Tomcat服务器来说会带来非常大的压力。所以本项目将使用Nginx的动静结合来进行控制，将本项目所涉及的接口全部设置为需要通过Nginx转发，而其他的直接放行，让请求更快速的发送和返回。

与此同时，Redis服务器作为缓存服务器起着至关重要的作用。餐厅信息和打卡信息的数量是非常庞大的，为了能够应对多个并发请求中对大量列表的请求，Redis的单线程操作能够保证查询业务的安全性，从无需加锁这个方向上减少了请求时延；同时多路IO复用也能够在单位时间内尽可能多的处理并发请求；完全基于内存的操作也减少了单次请求处理的时间，加快了平均处理速度。

编码时间同样是项目的一个考量。在项目走向市场、企业化管理的路上，每个实体对象的属性只会越来越多，对于一个实体类的构造器也就越来越复杂。针对每一个实体类定义构造器是非常耗时的过程，重复性的操作增加了编码时间。而lombok注解则针对每个类都能自动生成构造器，从而减少了编码时间。Lombok的使用如图4.5所示。

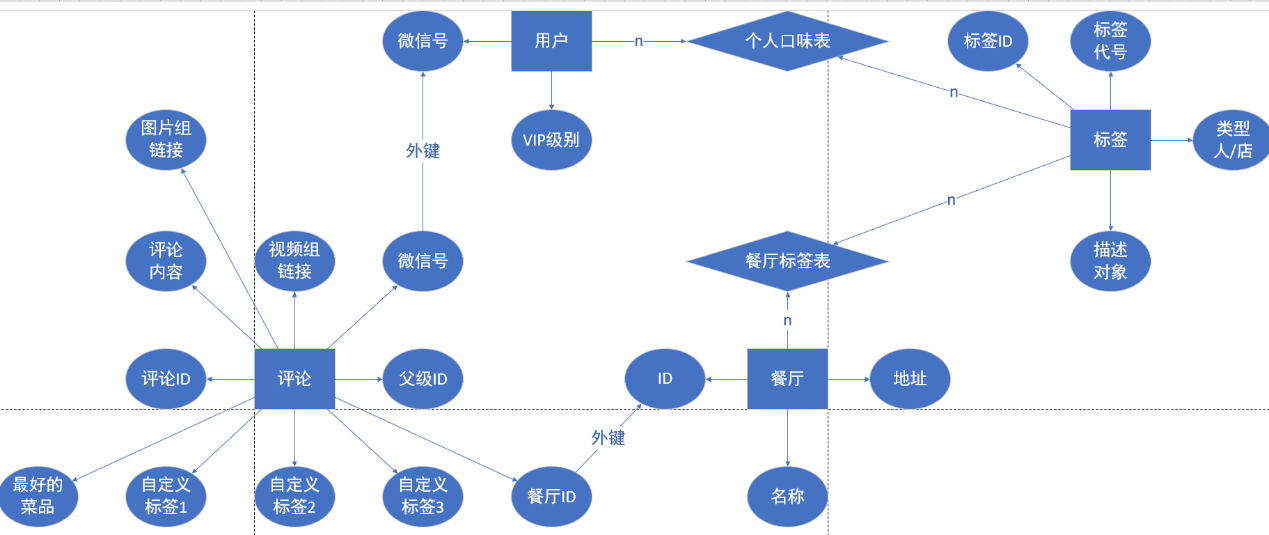
图4.5 lombok注解的使用



4.5.项目数据库开发方案

数据库属于自定义后台中的内容，能够根据需要设置字段并存储在数据库中。对于本项目而言，一共包含：餐厅、用户、打卡、评论、标签这五个基本实体。而用户和标签联立可以生成用户口味表，餐厅和标签能够生成餐厅口味属性表，餐厅和用户联立能够产生餐厅收藏表，打卡和用户联立能够生成用户打卡表，评论和用户联立能够生成用户评论表，用户评论表和用户打卡表联立才是完整的打卡模块。综上所述，能够设计数据库的E-R图如表4.6所示。

图4.6 项目数据库E-R图



对于基本表的联立查找，MyBatisPlus虽然可以在后台嵌入查询语句，但是使用后台联立查找依然非常消耗资源，并不是Java框架的长处所在。所以，本项目添加了一些视图简化查询，使用数据库非常强大的数据管理功能进行查找语句的处理。创建的视图如图4.6所示。

图4.6 视图的设置



4.6.遇到的问题和优化方案

在编辑“附近餐厅”这个功能的时候，本项目考虑到实际情况，设置了5千米的搜索范围。在进行这个业务的编写前，必须得到相应的餐厅的数据，于是使用爬虫将腾讯地图提供的API服务的数据爬取了一部分，爬到了湖北，四川，重庆等省份的大部分餐厅信息以及图片，并存储入数据库。 紧接着考虑到每次查询都要对大量数据进行筛选，于是设计了一个初始化方法接口将餐厅数据全部拷贝一份进入Redis服务器存储，在试运行阶段便完成餐厅信息的写入。但是后来发现即使这样查询接口所需要的时间依然在秒级别，用户在使用时依然能够通过肉眼观测到明显的白屏等待。

针对这个问题，我们也做了非常多的分析。首先考虑硬件问题，带宽仅有1M的服务器在请求发送的速度上有极限，单核处理器在请求处理过程中也有极限；其次在业务编辑中，完全不加优化地处理大量的餐厅信息，一次传输过程中需要加载过量的数据，造成了时延。

于是将所有数据以省份的经纬度进行分区划分存储，在每次筛选前都会进行分区的判定，这样在初始化一次后，便能达到每次查询接口完成的时间500ms以内，达到了相对较高的效率。

5.团队组成和分工

本团队一共四个人，分别是柳景兴、许祁、刘斯昊和杨伯益。职责表如表5.1所示。

表5.1 小组成员职责表

|  |  |  |
| --- | --- | --- |
| 姓名 | 身份 | 职责 |
| 许祁 | 项目小组长  后台工程师 | 1. 组员的调动 2. 数据库建立 3. 添加打卡信息接口 4. 上传用户收藏的餐厅（以及取消收藏）接口 5. 添加评论接口 6. 上传用户对打卡以及评论的点赞接口 7. 删除评论及打卡接口 8. 修改个人信息接口 9. 获取推荐打卡接口 10. 获取附近餐厅及打卡接口 11. 初始化Redis的存储 12. 完成项目的前后端对接与测试 |
| 刘斯昊 | 全栈工程师 | 1. 小程序打卡页面的评论查看和回复 2. 小程序餐厅的地图功能 3. 小程序的餐厅收藏功能 4. 小程序和后台服务器的接口交互编辑和测试 |
| 杨伯益 | 后台工程师 | 1. 使用MyBatisPlus自定义数据库增删改查接口 2. 后台服务器和Redis的交互 3. Redis服务器搭建与管理 4. 获取所有打卡信息接口 5. 获取打卡详情接口 6. 获取所有标签接口 7. 查找个人信息接口 8. 获取餐厅详情接口 9. 域名的备案 10. 项目实施Nginx动静结合、前后端项目分别部署的配置 11. 全局异常处理 12. 部署七牛云作为图片服务器 13. Nginx部署与管理，实现项目动静结合 |
| 柳景兴 | 前端工程师  架构师 | 1. 系统功能、界面的设计、模块功能、数据流的设计 2. 系统架构的设计 3. 数据库、视图的设计 4. 小程序主页面的设计和编码 5. 小程序餐厅页面的设计和编码 6. 小程序个人页面的设计和编码 7. 表单提交的请求头和请求参数的设置 8. 页面的美化 |