****

软件工程课程设计

“曲辰智慧点餐系统”概要设计报告

**队长：丁开锋**

**团队成员：季文昊、刘诗妮、刘卓霖**

目 录

[1.引言 1](#_Toc197462881)

[1.1项目背景 1](#_Toc197462882)

[1.2文档目标 1](#_Toc197462883)

[1.3术语定义 1](#_Toc197462884)

[2总体设计 2](#_Toc197462886)

[2.1系统架构 2](#_Toc197462887)

[2.1.1系统结构图 2](#_Toc197462888)

[2.1.3技术架构 4](#_Toc197462889)

[2.1.4设计构思 4](#_Toc197462890)

[2.2模块划分 5](#_Toc197462891)

[3.模块设计 5](#_Toc197462892)

[3.1核心模块说明 5](#_Toc197462893)

[3.1.1 用户管理模块 5](#_Toc197462894)

[3.1.2 菜品管理模块 6](#_Toc197462895)

[3.1.3 点餐模块 6](#_Toc197462896)

[3.1.4 订单管理模块 6](#_Toc197462897)

[3.1.5 优惠券系统模块 6](#_Toc197462898)

[3.1.6管理员后台模块 6](#_Toc197462899)

[3.1.7实时经营模块 7](#_Toc197462900)

[3.1.8数据统计模块 7](#_Toc197462901)

[3.2模块关系图 7](#_Toc197462902)

[4.接口设计 7](#_Toc197462903)

[4.1内部接口 8](#_Toc197462904)

[4.1.1 用户接口（顾客） 8](#_Toc197462905)

[4.1.2 普通管理员接口（菜品管理、订单监控） 9](#_Toc197462906)

[4.1.3 超级管理员接口 10](#_Toc197462907)

[4.2外部接口 10](#_Toc197462908)

[5.数据结构设计 11](#_Toc197462909)

[5.1数据库设计 11](#_Toc197462910)

[5.2缓存设计 15](#_Toc197462911)

[5.3文件存储 15](#_Toc197462912)

[6.性能设计 15](#_Toc197462913)

[6.1性能要求 15](#_Toc197462914)

[6.2安全性 16](#_Toc197462915)

[7.设计约束 17](#_Toc197462916)

# 1.引言

## 1.1项目背景

随着餐饮行业的数字化转型，越来越多的餐厅开始采用信息化管理手段来提高服务质量和运营效率。传统的点餐方式存在诸多弊端，如手工记录、订单容易遗漏、等待时间长等问题。

为解决传统点餐中的痛点，提高顾客的就餐体验，帮助餐厅提高效率，降低运营成本，本团队计划基于微信小程序构建一款**智能餐厅点餐系统**，进一步促进餐饮行业的智能化和信息化发展。

## 1.2文档目标

本《概要设计报告》面向产品经理、系统架构师、研发工程师与质量保证团队，目标是：

1.梳理系统范围——明确功能边界与主要业务流程；

2.给出总体架构与关键技术选型——为详细设计与编码提供依据；

3.定义性能、安全、兼容性指标——作为后续性能测试与安全评估的验收基线；

4.列出设计约束与风险点——指导资源投入与实施排期；

5.提供统一的术语和用——避免跨团队沟通歧义。

## 1.3术语定义

|  |  |
| --- | --- |
| 缩写 / 术语 | 说明 |
| 微信小程序 | 运行于微信客户端内的轻量级应用，无需安装即可使用。 |
| WXML / WXSS | 微信小程序的视图层标记语言与样式语言，对标 HTML / CSS。 |
| Django REST Framework | 基于 Django 的 RESTful API 框架，用于构建后端接口。 |
| JWT | 无状态身份认证令牌，包含用户声明与签名。 |
| CRUD | Create / Read / Update / Delete——数据的基本增删查改操作。 |

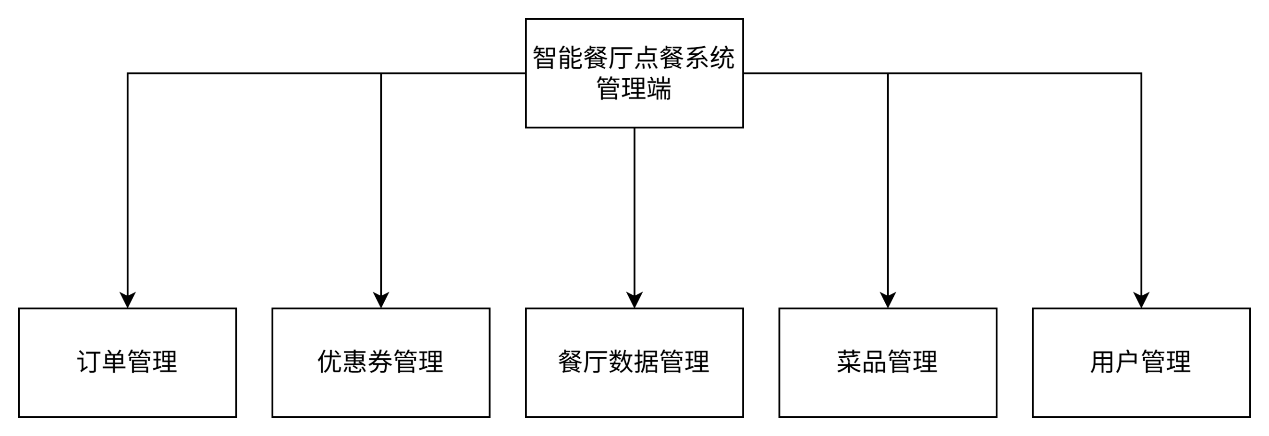
# 2总体设计

## 2.1系统架构

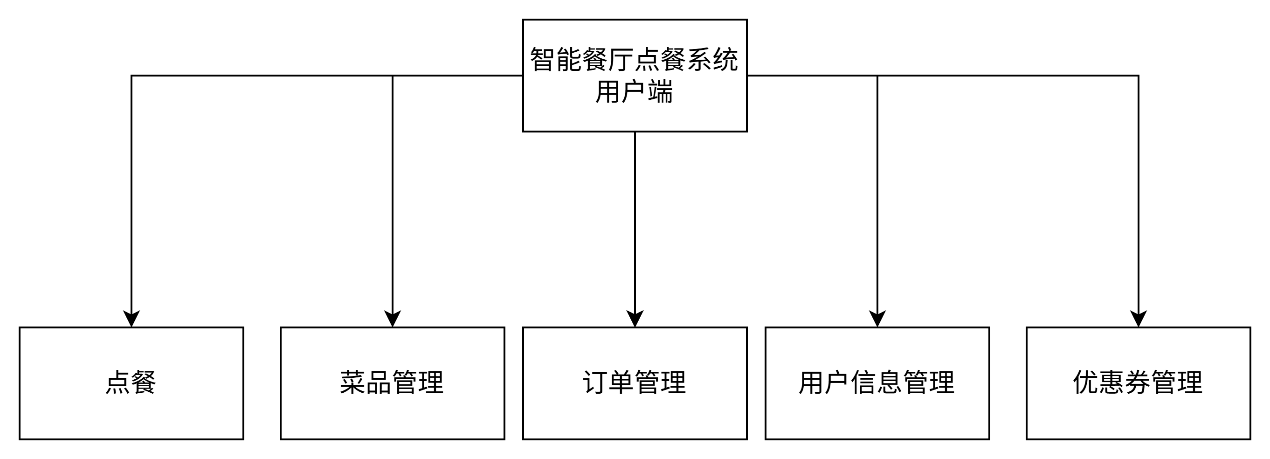
### 2.1.1系统结构图

本系统分为管理端和用户端两端。

管理端可实现对菜品、订单、用户数据等多类信息的管理

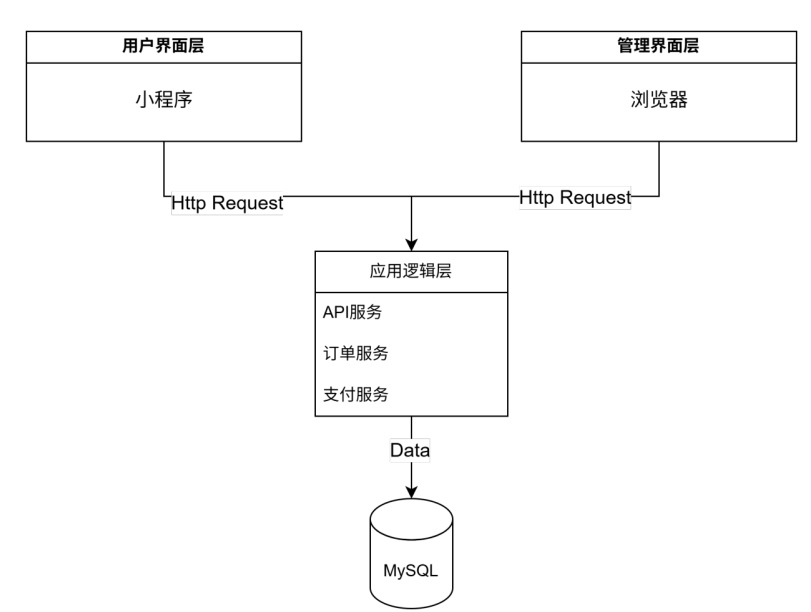


智能餐厅点餐系统管理端结构图

用户端可实现菜品选取订单生成、优惠券使用等功能的实现

智能餐厅点餐系统用户端结构图

#### 2.1.2系统架构图



### 2.1.3技术架构

1、开发语言：选择Python进行后端开发，通过微信小程序开发工具使用JavaScript编写逻辑，WXML和WXSS用于UI布局和样式设计。

2、Web框架：使用VUE框架构建后台前端服务，使用Django框架构建后端服务，使用Django的内置功能快速实现模块功能。

3、数据库：使用MySQL关系型数据库系统，确保系统的数据管理高效稳定。

4、版本控制工具：使用Git进行代码版本控制，确保多人协作开发时的代码管理和协作。

### 2.1.4设计构思

1、项目需求

传统餐饮往往伴随着效率低下、信息混乱、优惠匮乏等缺点。为了满足顾客和餐厅管理的需求，本智能餐厅管理系统目标提高餐厅自动化管理程度，提供一个用户便捷使用的点餐平台，支持点餐、使用优惠券、订单管理等功能来为顾客和管理者提供便捷高效的服务。

2、核心功能设计

|  |  |
| --- | --- |
| 功能模块 | 主要功能 |
| 智能点餐识别 | 扫描桌面二维码自动关联该桌的所有信息  实现不同桌的信息独立获取 |
| 优惠券使用 | 用户可使用不同金额的优惠券  管理员管理优惠券的发放 |
| 订单管理 | 实时订单状态追踪 |
| 后台管理 | 经营数据分析看板 |
| 用户管理 | 支持账户通过小程序的注册与登录  实现微信账户绑定系统 |
| 菜品管理 | 管理员新增、修改、上下架菜品；查询菜品价格历史，并管理套餐 |

## 2.2模块划分

**用户管理模块**：管理用户账户数据

**菜品管理模块**：管理菜品数据

**点餐模块**：提供点餐功能

**订单管理模块**：管理订单数据，实现实时订单管理追踪

**优惠券系统模块**：管理优惠券信息和状态

**管理员后台模块**：提供管理餐厅日常运营的平台

**实时经营模块**：展示餐厅经营状态

**数据统计模块**：存储并分析餐厅数据

# 3.模块设计

## 3.1核心模块说明

### 3.1.1 用户管理模块

设计思路：提供用户身份验证和个人信息管理功能，确保用户能够安全地访问系统并管理自己的账户信息。

实现方式：

使用数据库存储用户信息，如用户名、密码（加密存储）、联系方式等。

提供注册和登录接口，支持邮箱验证和第三方授权登录（如微信）。

提供个人信息管理界面，允许用户更新个人资料。



### 3.1.2 菜品管理模块

设计思路：允许管理员管理系统中的菜品信息，包括添加新菜品、修改菜品信息、删除菜品等。

实现方式：

使用数据库存储菜品信息，如菜品名称、价格、描述、图片等。

提供后台管理界面，管理员可以通过该界面进行菜品信息的增删改查操作。

菜品信息通过API提供给点餐模块，供用户浏览。



### 3.1.3 点餐模块

设计思路：为用户提供在线点餐的界面，包括浏览菜品、加入购物车、提交订单等功能。

实现方式：

用户选择菜品后，可以添加到购物车，提交订单时调用订单管理模块的接口。

提供扫码点餐功能，用户可以通过扫描二维码快速下单。



### 3.1.4 订单管理模块

设计思路：处理用户的订单，包括订单状态查看、订单修改与取消、订单历史记录等。

实现方式：

使用数据库存储订单信息，包括订单编号、用户信息、订单内容、订单状态等。

提供订单管理界面，用户可以查看订单状态，管理员可以更新订单状态。

订单状态更新后，通过支付接口模块处理支付。



### 3.1.5 优惠券系统模块

设计思路：管理员可以发放优惠券，用户可以在点餐时使用优惠券享受折扣。

实现方式：

使用数据库存储优惠券信息，包括优惠券编号、金额、有效期、使用条件等。

提供后台管理界面，管理员可以创建和管理优惠券。

用户在结算时可以选择使用优惠券，系统自动计算折扣。

### 3.1.6管理员后台模块

设计思路：为餐厅管理员提供管理餐厅日常运营的工具，包括管理菜品、订单、用户数据和优惠券。

实现方式：

提供一个后台管理界面，管理员可以登录并访问所有管理功能。

管理员可以查看和管理系统中的所有数据，包括菜品、订单、用户信息和优惠券。

管理员可以通过后台界面生成报告和分析，以优化餐厅运营。



### 屏幕截图 2025-05-07 173735

### 3.1.7实时经营模块

设计思路：提供一个界面显示餐厅的即时运营数据，包括餐桌状态和当日营业额。

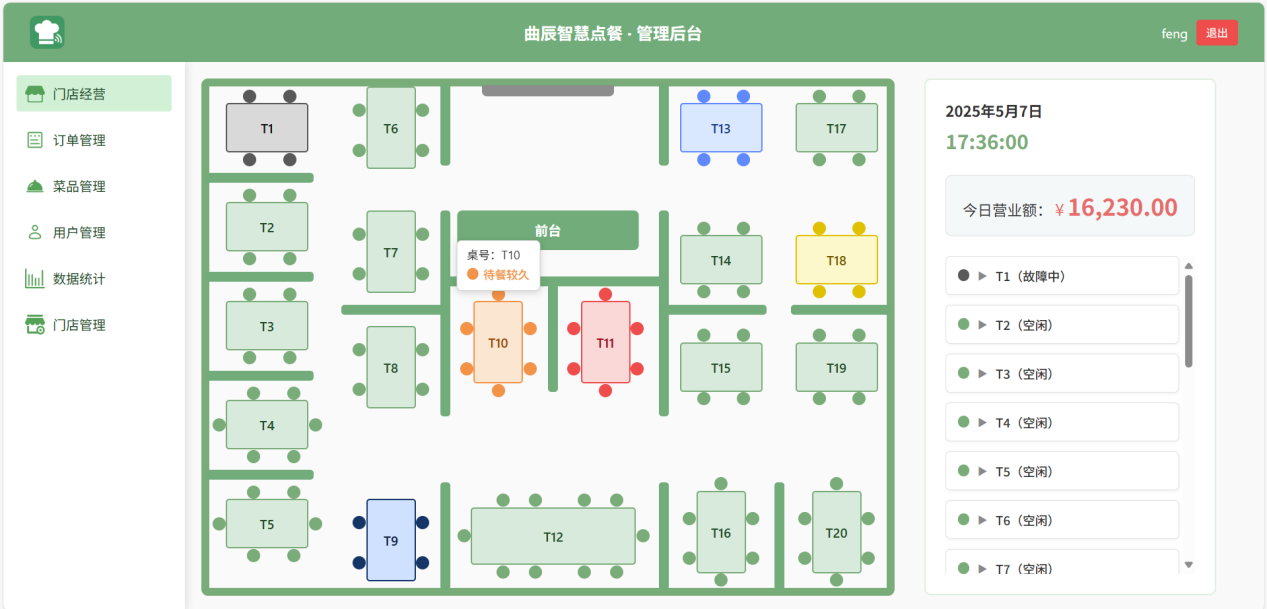
实现方式：

实现数据收集，自动收集餐桌使用情况和订单数据。

界面展示中用不同颜色或图标表示餐桌状态（空闲、占用、故障）。

实现实时更新，使用推送技术保证数据的即时性。

设置警报通知，如餐桌故障或订单异常。



### 3.1.8数据统计模块

设计思路：收集并分析餐厅运营数据，以辅助管理决策。

实现方式：

从订单和用户行为中收集数据，计算关键指标，如销售额和顾客流量，用图表展示数据分析结果，提供业务洞察。

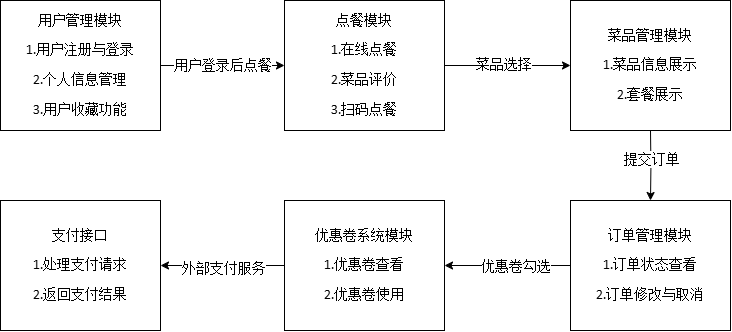
## 3.2模块关系图

用户管理模块依赖于点餐模块来处理用户的点餐活动。

点餐模块调用菜品管理模块以获取菜品信息供用户选择。

用户在点餐模块中选择菜品后，通过订单管理模块提交订单。

订单管理模块会将支付完成后的结果返回给订单管理模块，以便更新订单状态。

优惠券系统模块与订单管理模块交互，以便在用户支付时应用优惠券。

客户前台模块关系图

管理员后台模块作为系统的管理中心，管理员可以通过它访问和操作菜品管理模块、用户管理模块、优惠券系统模块、订单管理模块、门店经营模块、数据统计模块。



管理员后台模块关系图

# 4.接口设计

所有接口均采用 RESTful 风格，身份识别依赖 JWT Token，通过 Authorization: Bearer <token> 进行权限判断。

## 4.1内部接口

### 4.1.1 用户接口（顾客）

|  |  |  |  |
| --- | --- | --- | --- |
| **功能** | **请求方式** | **接口路径** | **说明** |
| 登录/注册 | POST | /user/login/ | 微信授权或表单登录，返回 token |
| 获取用户信息 | GET | /user/profile/ | 获取当前用户信息（需携带 token） |
| 更新用户信息 | PUT | /user/update/ | 修改昵称、头像等信息 |
| 获取菜品列表 | GET | /dishes/list/ | 支持分类、关键词等筛选 |
| 获取菜品详情 | GET | /dishes/<id>/ | 查看某道菜的图文介绍、评分、评论 |
| 点餐下单 | POST | /order/create/ | 提交购物车生成订单 |
| 查询订单列表 | GET | /order/list/ | 查看历史订单和当前订单（含待支付） |
| 查看订单详情 | GET | /order/<id>/ | 查看某个订单的详细信息和状态 |
| 评价菜品 | POST | /review/create/ | 提交菜品评价，需包含 dish\_id、rating 等 |
| 使用优惠券 | POST | /coupon/use/ | 提交订单时绑定可用优惠券 |
| 评价门店 | POST | /review/shop/ | 针对整体门店服务进行评论打分 |
| 删除评价 | DELETE | /review/<review\_id>/delete/ | 删除自己发布的菜品或门店评价 |
| 扫码点餐 | GET | /table/scan/<code>/ | 扫码后进入点餐页面，code 表示餐桌编号 |

### 4.1.2 普通管理员接口（菜品管理、订单监控）

|  |  |  |  |
| --- | --- | --- | --- |
| **功能** | **请求方式** | **接口路径** | **说明** |
| 管理员登录 | POST | /manager/login/ | 返回 token（管理员权限） |
| 菜品新增 | POST | /dish/create/ | 上传菜品图片、名称、价格等 |
| 菜品修改 | PUT | /dish/update/ | 修改描述、价格、图片等 |
| 菜品上下架 | POST | /dish/status/ | 设置菜品是否启用/禁用 |
| 查询订单列表 | GET | /order/manage/ | 查看所有用户订单，支持按状态、时间筛选 |
| 修改订单状态 | PUT | /order/update/ | 更新订单状态为“配送中”、“已完成”等 |
| 查看菜品评价 | GET | /review/dish/<id>/ | 查看指定菜品的全部评论 |
| 查看门店评价 | GET | /review/shop/ | 按时间段统计全店评分均值 |
| 查看用户列表 | GET | /user/list/ | 所有注册用户的信息，支持模糊搜索和状态筛选 |
| 创建优惠券 | POST | /coupon/create/ | 设置面值、满减限制、有效期 |
| 发放优惠券 | POST | /coupon/assign/ | 向指定用户ID分发优惠券 |
| 创建套餐 | POST | /package/create/ | 将多个菜品组合为一个套餐 |
| 编辑套餐 | PUT | /package/update/ | 修改套餐的内容、价格等 |
| 删除套餐 | DELETE | /package/delete/ | 逻辑删除套餐（标记为不可用） |
| 获取套餐列表 | GET | /package/list/ | 返回所有可用套餐信息 |
| 系统数据统计 | GET | /report/summary/ | 查询订单量、总销售额、菜品数量等经营指标 |
| 查看餐桌状态 | GET | /table/<code>/status/ | 查看指定餐桌编号的当前使用状态和订单信息 |

### 4.1.3 超级管理员接口

|  |  |  |  |
| --- | --- | --- | --- |
| **功能** | **请求方式** | **接口路径** | **说明** |
| 创建管理员账号 | POST | manager/create\_admin/ | 创建普通管理员账号，设置初始密码与角色 |
| 删除管理员账号 | DELETE | manager/delete\_admin/ | 删除管理员账号或设为逻辑删除 |
| 启/禁用管理员 | POST | manager/toggle\_admin/ | 启用或冻结指定管理员账号 |
| 管理员列表查询 | GET | manager/admin\_list/ | 获取当前系统中所有管理员账户信息 |
| 查看所有餐桌二维码 | GET | manager/table/qrcodes/ | 返回所有桌号与对应二维码 URL（用于打印/展示） |
| 批量生成餐桌二维码 | POST | manager/table/generate/ | 输入桌号范围，批量生成二维码 |

## 4.2外部接口

|  |  |  |
| --- | --- | --- |
| **第三方平台** | **接口类型** | **说明** |
| 微信开放平台 | 登录与授权接口 | 实现用户微信一键登录功能 |
| 微信支付接口 | 支付确认与回调 | 提交支付请求后由微信回调订单支付状态 |
| 微信小程序 SDK | 客户端页面跳转等 | 实现扫码跳转点餐页面、识别餐桌编号 |

# 5.数据结构设计

## 5.1数据库设计

系统数据库采用 MySQL 关系型数据库，主要表大纲如下：

1.用户表 `users`

2.管理员表 `admin\_infos`

3.餐桌表 `tables`

4.餐桌状态表 `table\_status`

5.菜品表 `dishes`

6.菜品分类表 `dish\_categories`

7.菜品价格历史表 `dish\_price\_history`

8.套餐表 `packages`

9.套餐-菜品关联表 `package\_items`

10.订单表 `orders`

11.订单详情表 `order\_items`

12.优惠券表 `coupons`

13.用户-优惠券表 `user\_coupons`

14.菜品评价表 `dish\_reviews`

15.门店评价表 `shop\_reviews`

## 5.2缓存设计

系统采用 Redis 进行缓存设计，针对高频访问的数据和临时性数据进行缓存，确保数据访问的效率和系统性能优化。缓存的主要应用场景如下：

**缓存结构设计：**

|  |  |  |  |
| --- | --- | --- | --- |
| **缓存对象** | **缓存键格式** | **数据类型** | **说明** |
| 用户登录 Token | token:user:<user\_id> | String | 用户登录后的 JWT Token |
| 菜品列表缓存 | dishes:category:<cat\_id> | Hash | 菜品分类下的所有菜品信息 |
| 菜品详情缓存 | dish:<dish\_id> | Hash | 单个菜品详细信息（价格、描述） |
| 门店评价缓存 | shop\_reviews | List | 最新 50 条门店评价 |
| 菜品评价缓存 | dish\_reviews:<dish\_id> | List | 某菜品的最新 20 条评价 |
| 套餐缓存 | packages:list | Hash | 所有套餐信息 |

**缓存更新策略：**

**(1)用户 Token：**每次登录或 Token 刷新时，更新缓存；

**(2)菜品列表缓存：**菜品新增、修改、上下架时，刷新对应分类缓存；

**(3)订单缓存：**用户下单、支付成功、订单完成后更新；

**(4)门店/菜品评价缓存：**每次新增评价时，更新并维持最大数量条目。

## 5.3文件存储

系统中的文件存储涉及**用户头像、**菜品**图片、餐桌二维码等**，根据不同文件类型分别存储到不同路径，以保证文件管理的规范性和有效性。

**文件路径及命名规则:**

|  |  |  |
| --- | --- | --- |
| **文件类型** | **路径格式** | **命名规则** |
| 用户头像 | /media/users/<user\_id>/avatar.jpg | 用户ID + 文件扩展名 |
| 菜品图片 | /media/dishes/<dish\_name>/ | 菜品名 + 文件扩展名 |
| 套餐图片 | /media/packages/<package\_id>/ | 套餐ID + 文件扩展名 |
| 餐桌二维码 | /media/qrcodes/<table\_id>.png | 餐桌ID + .png |

**文件访问控制:**

**用户头像：**通过 Token 鉴权访问 /user/profile/ 获取用户头像 URL；

**菜品图片：**公开访问，所有菜品图片可直接加载；

**二维码文件：**超级管理员访问，餐桌二维码用于扫码点餐；

# 6.性能设计

## 6.1性能要求

|  |  |  |
| --- | --- | --- |
| 指标 | 目标值 | 说明 |
| 页面首屏加载时间 | ≤ 2 s | 完成 app.js 初始化、WXML/WXSS 解析与首批接口请求 |
| 交互响应时间 | ≤ 200 ms | 包括点餐、加购、切换 Tab 栏等前端本地交互 |
| 接口延迟 | 平均 ≤ 300 ms，峰值 ≤ 500 ms | 超时降级兜底返回缓存数据 |
| 并发承载 | ≥ 100 并发在线用户，≥ 50 笔订单/分钟 | 使用 Gunicorn + Uvicorn 的多 worker、数据库连接池、Redis 缓存 |
| 资源占用 | 前端包体 ≤ 2 MB；接口峰值 CPU < 60 % | 小程序包体符合微信 2 MB 限制；后端按 8 C 16 G 预估 |

## 6.2安全性

1.通信安全

小程序端统一走微信域名白名单。

2.身份认证与授权

小程序端：`code→session`换取`openid`+`session\_key`，生成自定义`access\_token`（JWT）。

管理端：DjangoDRFToken（JWT）认证，密码采用PBKDF2+。

RBAC：游客/普通管理员/超级管理员三级角色，接口粒度权限控制。

3.数据保护

客户隐私数据最小化存储；敏感字段（手机号、交易号）AES‑256CTR加密。

4.应用安全

ORM+参数化查询防SQL注入；统一校验层防止XSS、SSRF、路径遍历。

# 7.设计约束

|  |  |
| --- | --- |
| 维度 | 内容 |
| 前端 | 微信小程序原生技术栈：WXML／WXSS／JavaScript（可选 TypeScript）组件库：Vant Weapp（二次封装保持主题一致）包体大小 ≤ 2 MB，使用分包、图片 CDN、Tree-Shaking 优化 |
| 后台 | Python 3.11；Web 框架：Django 4.2 + Django REST Framework并发：Gunicorn（sync 或 uvicorn workers） |
| 数据库 | MySQL 8.0（InnoDB），字符集 utf8mb4；读写分离、连接池（300） |
| 缓存 / 队列 | Redis 7（缓存、分布式锁、订单异步超时关闭） |
| 管理后台 | Vue 3 + Vite + Element Plus；运行时 Node.js 18 LTS |
| 部署（可选） | Docker Compose/Kubernetes，镜像基于 python:3.11-slim统一 CI/CD（GitHub Actions → 阿里云容器镜像服务） |
| 网络 | 启用 GZIP、WebP、HTTP keep-alive |
| 支付 | 仅作为演示界面，并不实际接入微信支付 |
| 第三方约束 | 微信接口域名白名单限制单接口超时 60 s／报文体积 1 MB（小程序限制） |
| 编码规范 | PEP 8 / ESLint；Git 提交遵循 Conventional Commits |