****

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](#_Toc22347)

[1.1项目背景 1](#_Toc10346)

[1.2编写目的 1](#_Toc8374)

[2总体设计 1](#_Toc1993)

[2.1系统结构图 1](#_Toc27098)

[2.2系统架构图 3](#_Toc10302)

[2.3技术架构 3](#_Toc32761)

[3.代码开发架构说明 3](#_Toc15874)

[3.1 用户前端架构设计（QuickMealFront） 3](#_Toc31332)

[3.2 管理员前端架构设计（QuickMealManagerFront） 4](#_Toc20280)

[3.3 后端架构设计（QuickMealBack） 5](#_Toc10877)

[3.4代码注释设计 6](#_Toc25690)

[3.4.1注释原则 6](#_Toc2290)

[3.4.2注释类型 6](#_Toc17683)

[3.4.3注释规范 7](#_Toc29096)

[4.一般用户设计说明 8](#_Toc27888)

[4.1功能（需要一张功能流程图） 8](#_Toc19506)

[4.1.1用户主体功能 8](#_Toc32397)

[4.1.2功能流程图 12](#_Toc27848)

[4.2输入项和输出项（程序的输入和输出） 13](#_Toc28965)

[4.2.1输入项 13](#_Toc6674)

[4.2.2输出项 13](#_Toc21426)

[4.3算法 14](#_Toc17996)

[4.4存储分配 14](#_Toc6655)

[4.4.1 前端存储 14](#_Toc16013)

[4.4.2 后端存储 14](#_Toc19292)

[4.5测试计划 15](#_Toc26932)

[4.5.1单元测试 15](#_Toc26576)

[4.5.2集成测试 15](#_Toc13509)

[4.5.3性能测试 15](#_Toc16379)

[5.管理员设计 15](#_Toc8902)

[5.1程序说明 15](#_Toc7442)

[5.2 功能 15](#_Toc30291)

[5.3 输入项和输出项 18](#_Toc16786)

[5.4 算法 20](#_Toc3203)

[5.5存储分配 20](#_Toc20630)

[5.6测试计划 21](#_Toc21723)

[6.超级管理员设计说明 21](#_Toc22786)

[6.1程序说明 21](#_Toc21081)

[6.2功能（需要一张功能流程图） 21](#_Toc27358)

[6.3输入项和输出项（程序的输入和输出） 23](#_Toc4579)

[6.4算法 24](#_Toc4235)

[6.5存储分配 24](#_Toc10522)

[6.6测试计划 24](#_Toc4807)

[7.页面设计 24](#_Toc10469)

[7.1用户端设计 24](#_Toc26000)

[7.1.1页面展示 24](#_Toc2349)

[7.1.2处理逻辑 25](#_Toc19818)

[7.2管理端设计 26](#_Toc32674)

[7.2.1页面设计 26](#_Toc1946)

[7.2.2处理逻辑 27](#_Toc388)

[8.性能设计 28](#_Toc21829)

# 1.引言

## 1.1项目背景

随着餐饮行业数字化与精细化运营需求的不断提升，传统人工点餐方式暴露出排队慢、出错率高、客户体验分散等痛点。尤其在COVID-19疫情后，“无接触服务”与堂食+外卖一体化的经营模式快速普及。微信生态凭借12亿月活与完善的支付、社交链路，成为餐饮门店部署智能点餐的首选平台。本项目基于微信小程序搭建“智能餐厅点餐系统”，旨在提供：

顾客端——扫码即用、快速下单、在线支付、实时催单&评价；

店员端——厨房分单、上菜提醒、库存/售价实时同步；

管理端——菜品与菜单配置、订单与财务统计、会员与营销活动管理。

通过目标检测+推荐算法（可在未来迭代入）实现智能配餐与销售数据驱动的运营优化，帮助中小餐饮商家降本增效、提升顾客满意度。

## 1.2编写目的

本《详细设计报告》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。

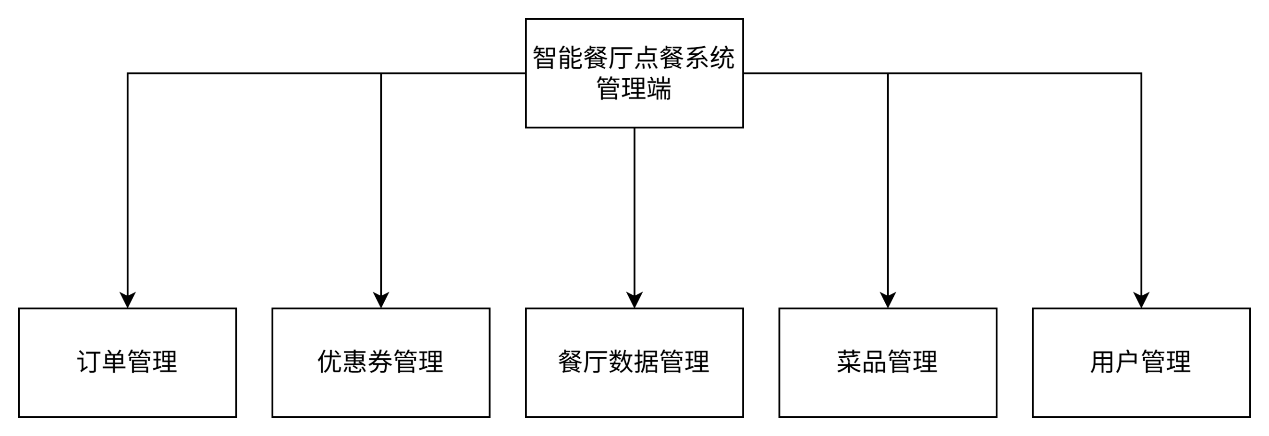
文档完成后，可作为立项评审、招投标、以及后续迭代版本的对照基准。

# 2总体设计

## 2.1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

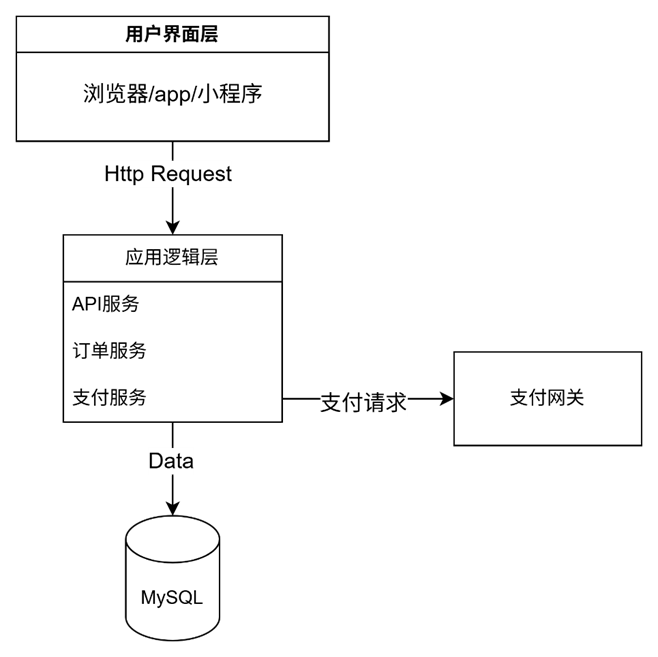


智能餐厅点餐系统管理端结构图

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

## 2.2系统架构图



## 2.3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或PostgreSQL等关系型数据库系统，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。

# 3.代码开发架构说明

### 3.1 用户前端架构设计（QuickMealFront）

用户前端是基于微信小程序框架开发的，主要目录结构如下：

QuickMealFront/

├── components/ # 公共组件，例如菜品卡片、评分组件等

├── icons/ # 图标资源文件夹

├── images/ # 图片资源文件夹

├── miniprogram\_npm/ # 第三方小程序依赖（如使用了 npm 构建）

├── node\_modules/ # Node.js 依赖模块（本地构建依赖）

├── pages/ # 页面目录（首页、菜单、下单、个人中心等）

│ ├── index/ # 首页页面

│ ├── menu/ # 菜单页面

│ ├── order/ # 下单页面

│ ├── profile/ # 用户中心页面

│ └── ... # 可能还有其他功能页面

├── app.js # 小程序逻辑入口文件

├── app.json # 小程序全局配置，定义页面路径和窗口样式

├── app.wxss # 小程序全局样式文件

├── project.config.json # 微信开发者工具的项目配置文件

├── project.private.config.json# 私密配置，仅在本地生效

├── package.json # npm 项目的配置文件

├── package-lock.json # npm 锁定文件，记录依赖的确切版本

├── .eslintrc.js # ESLint 配置文件，用于代码风格检查

├── sitemap.json # 小程序页面结构地图（用于微信爬虫优化）

└── README.md # 项目说明文档

该结构遵循微信小程序的标准开发模式，页面和组件分离，便于维护和扩展。

### 3.2 管理员前端架构设计（QuickMealManagerFront）

管理员前端是使用 Vue.js 3 开发的单页面应用，主要用于餐厅管理人员进行菜品管理、订单处理等操作。目录结构如下：

QuickMealManagerFront/

├── public/ # 公共资源文件夹，如 favicon 等

├── src/ # 核心源码目录

│ ├── api/ # 封装的后端接口请求模块

│ ├── assets/ # 静态资源，如图片、字体等

│ ├── components/ # 公共组件，如表单组件、导航栏等

│ ├── views/ # 页面视图组件

│ │ ├── Dashboard.vue # 首页

│ │ ├── Dishes.vue # 菜品管理页

│ │ ├── Orders.vue # 订单管理页

│ │ └── Tables.vue # 餐桌管理页

│ ├── router/ # 前端路由配置（Vue Router）

│ ├── store/ # 状态管理目录（Pinia）

│ ├── styles/ # 样式文件（如全局 CSS、主题等）

│ ├── utils/ # 工具函数，如时间格式化、本地存储等

│ ├── App.vue # 根组件

│ ├── main.js # 应用入口文件（初始化 Vue 应用）

│ └── style.css # 全局样式文件（可用于重置样式）

├── index.html # 项目入口 HTML 文件

├── package.json # 项目依赖配置

├── package-lock.json # 锁定依赖版本

├── vite.config.js # 构建工具 Vite 的配置文件

├── .gitignore # Git 忽略文件配置

└── README.md # 项目说明文档

该结构清晰地分离了组件、视图、路由和状态管理，便于团队协作和功能扩展。

### 3.3 后端架构设计（QuickMealBack）

后端采用 Django 框架开发，提供 RESTful API 接口，供前端调用。目录结构如下：

QuickMealBack/

├── dish/ # 菜品管理模块（如 models、views、serializers）

│

├── manager/ # 管理员账户相关模块

│ ├── \_\_init\_\_.py # Python 包初始化文件

│ ├── admin.py # Django 后台管理配置（如注册模型）

│ ├── apps.py # 应用配置类

│ ├── models.py # 数据模型定义（如管理员表、权限字段等）

│ ├── serializers.py # DRF 序列化类（将模型转换为 JSON）

│ ├── urls.py # 本模块路由定义（如 /manager/login/）

│ ├── views.py # 视图处理逻辑（支持 RESTful 请求）

│ ├── tests.py # 单元测试文件

│ ├── migrations/ # 数据库迁移文件目录（自动生成）

│ └── \_\_pycache\_\_/ # 编译缓存（自动生成）

│

├── media/ # 媒体资源目录（用于图片、文件上传）

│

├── quickmeal\_backend/ # 主 Django 项目目录

│ ├── \_\_init\_\_.py

│ ├── asgi.py # 异步服务器网关接口

│ ├── settings.py # 全局配置文件（包含数据库、应用注册等）

│ ├── urls.py # 根 URL 路由入口，连接各子应用

│ └── wsgi.py # Web 服务器网关接口（部署时使用）

│

├── utils/ # 公共工具模块（如响应封装、权限中间件等）

├── manage.py # Django 管理命令入口（如 runserver、migrate 等）

└── README.md # 项目说明文档

每个应用（如 dishes、orders）都包含 models.py、views.py、serializers.py、urls.py 等文件，遵循 Django 的应用分离原则，便于模块化开发和维护。

## 3.4代码注释设计

### 3.4.1注释原则

必要性：注释用于解释代码的核心逻辑，避免对显而易见的语句赘述；

准确性：注释必须准确描述当前代码功能，随代码更新同步维护；

统一性：遵循统一的注释格式，包括缩进、语言、标记等；

可读性：使用通俗易懂的中文，清晰表达业务和逻辑意图。

### 3.4.2注释类型

文件头注释:每个源文件顶部应包含该文件的名称、功能描述、作者、创建时间等基本信息。用于帮助开发者快速了解该文件的作用。

模块注释:模块注释用于标识代码中的逻辑块或功能区段，通常作为分割线使用，便于开发者在浏览代码时快速定位某个功能区域。

函数注释:所有关键函数必须附带注释，说明该函数的主要功能、输入参数、返回值以及必要的使用说明。采用统一格式书写，提升代码自说明能力。

变量注释:对全局变量、结构复杂的局部变量或关键状态变量进行说明，便于理解其用途和在程序中的作用，尤其适用于对象、数组等结构体类型的数据。

关键逻辑注释:对程序中涉及判断、计算、流程控制等关键业务逻辑部分进行详细注释。特别是在涉及动画处理、状态同步或算法流程等复杂逻辑时，注释可帮助维护人员快速理解原始设计思路。

### 3.4.3注释规范

|  |  |  |
| --- | --- | --- |
| 注释类型 | 位置/适用场景 | 内容要求 |
| 文件头注释 | 所有源文件顶部 | 文件名、核心功能、作者、创建时间、最后修改记录（版本+日期） |
| 模块注释 | 功能模块分界处 | 模块名称+核心职责，可选接口说明 |
| 函数注释 | 方法/函数定义前 | 功能说明、参数（类型+单位+约束）、返回值、异常、示例（复杂函数必需） |
| 变量注释 | 全局/复杂变量定义处 | 用途说明、数据结构、有效范围 |
| 关键逻辑 | 复杂算法/状态机处 | 业务背景、算法原理、注意事项 |
| 维护规范 | 代码变更时 | 1. 修改代码必须同步更新注释 |
| 2. 删除代码需清除关联注释 |
| 3. 禁止提交TODO/FIXME到生产环境 |

# 4.一般用户设计说明

## 4.1功能（需要一张功能流程图）

### 4.1.1用户主体功能

**用户管理模块：**

注册、登录、修改个人资料（如头像）。

查看和管理个人账户信息（如订单历史、优惠券、菜品收藏）。

**菜品管理模块：**

浏览餐厅菜单（菜品名称、价格、图片、描述）。

根据分类（如热销、折扣、菜系）筛选菜品。

**点餐模块：**

可通过扫描餐桌二维码进行点餐（不同餐桌推荐菜品有所不同，用户可按需选择）。

添加菜品到购物车，调整数量或备注需求。

在线支付，生成订单。

**优惠券系统模块：**

领取平台发放的优惠券（如满减券、折扣券）。

下单时选择可用优惠券抵扣金额。

**订单管理模块：**

查看订单状态（待支付、制作中、已完成）及订单具体信息。

取消商家暂未处理的订单或申请售后服务联系服务员。

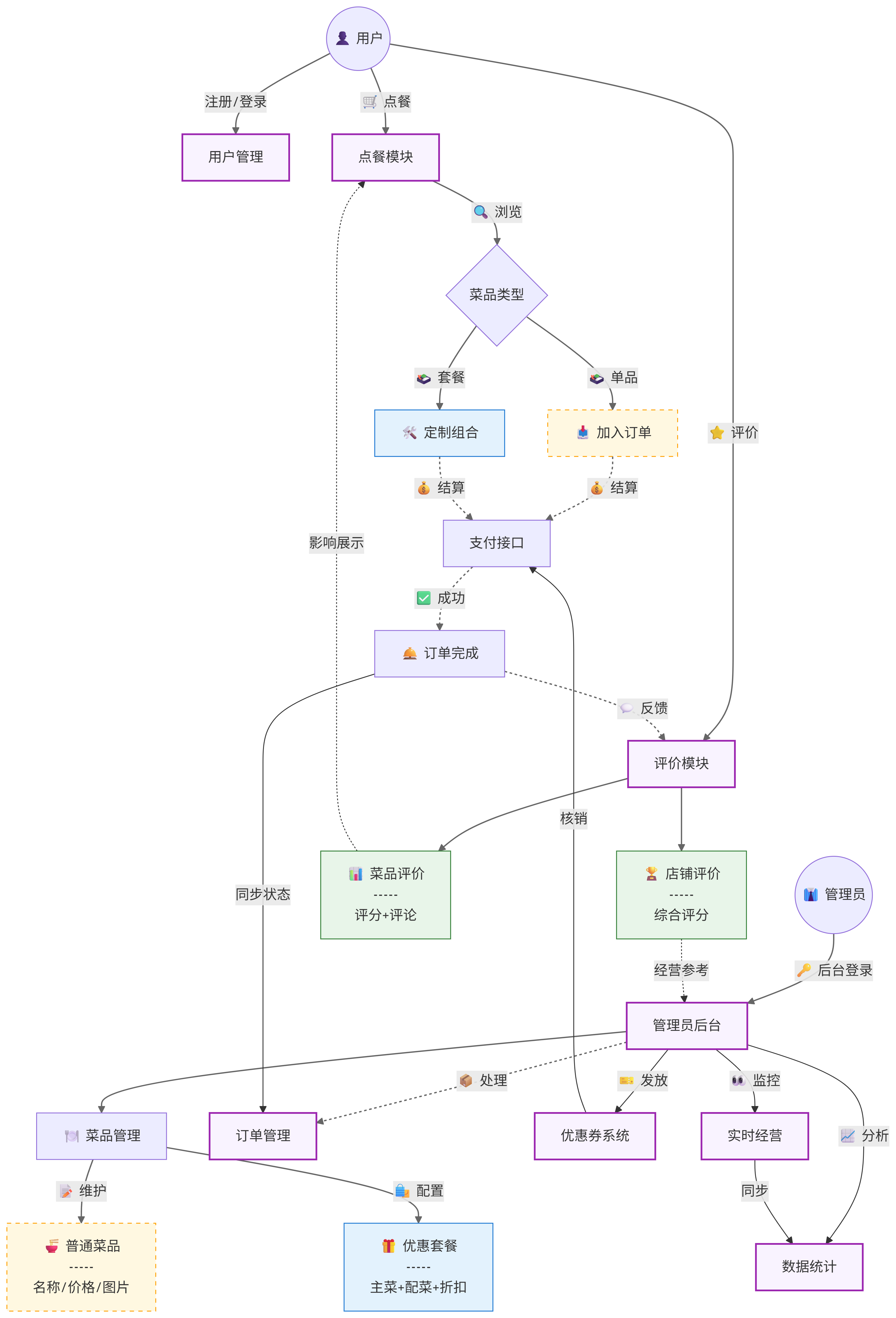
**菜品评价模块：**

查看历史菜品评价。

发表当前点餐菜品评价。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模块 | 功能 | 程序流程 | 接口 | 涉及数据表 | 关键算法/机制 |
| 用户管理模块 | 注册/登录 | 1. 用户选择微信授权或表单登录。 | POST /user/login/ | users | - BCrypt加密密码 |
| 2. 后端验证身份（微信OpenID或密码哈希匹配）。 | - JWT生成与验证 |
| 3. 生成并返回JWT Token，存储至Redis。 | - Redis缓存Token |
|  | 修改个人资料 | 1. 用户提交修改请求（头像、联系方式）。 | PUT /user/update/ | users | - 参数格式校验 |
| 2. 后端验证Token权限。 | - 数据库事务更新 |
| 3. 更新用户表字段并同步缓存。 |  |
|  | 查看账户信息 | 1. 用户请求个人资料。 | GET /user/profile/ | users、orders、user\_coupons | - 多表关联查询 |
| 2. 联查用户表、订单表、用户优惠券表、菜品收藏表。 | - Redis缓存用户基础信息 |
| 3. 返回订单历史、优惠券列表、收藏列表。 |  |
| 菜品管理模块 | 浏览菜单 | 1. 用户按分类或关键词筛选菜品。 | GET /dishes/list/ | dishes、dish\_categories | - 分页算法（offset计算） |
| 2. 后端联查菜品表和分类表，返回分页结果。 | - SQL动态条件拼接 |
| 3. 缓存热门分类数据。 |  |
| 点餐模块 | 扫码点餐 | 1. 用户扫描二维码，后端验证餐桌状态是否空闲，不同餐桌推荐不同菜品风味。 | GET /table/scan/{code}/ | tables、table\_status、order\_items | - 协同过滤推荐算法（基于历史订单） |
| 2. 根据历史订单或热门销量推荐菜品，提供优惠套餐供用户选择。 | - Redis ZSET维护实时热门菜品 |
| 3. 更新餐桌状态为“点餐中”。 |  |
|  | 购物车管理 | 1. 用户添加菜品至购物车，调整数量和备注。 | - | - |  |
| 2. 购物车数据临时存储于Redis（按用户和餐桌区分）。 | - 乐观锁控制并发 |
| 3. 使用事务保证并发操作安全。 |  |
|  | 在线支付与订单生成 | 1. 提交购物车生成订单。 | POST /order/create/ | orders、order\_items、dishes | - 乐观锁扣减库存 |
| 2. 模拟支付，若支付成功则更新订单状态；若支付失败则有界面提示。 | - 异步消息队列处理支付回调 |
| 优惠券系统模块 | 领取优惠券 | 1. 用户领取优惠券，后端校验有效期和剩余数量。 | POST /coupon/use/ | coupons、user\_coupons | - 唯一性校验（防重复领取） |
| 2. 插入用户-优惠券关联表，并更新优惠券剩余数量。 | - 原子操作更新剩余数量 |
|  | 使用优惠券 | 1. 用户下单时选择优惠券，后端校验门槛金额和有效期。 | POST /coupon/use/ | user\_coupons、coupons | - 规则引擎解析优惠条件（如满减、折扣） |
| 2. 计算抵扣金额（满减或折扣）。 |
| 3. 标记优惠券为已使用。 |
| 订单管理模块 | 查看订单详情 | 1. 用户请求订单详情，后端联查订单表和订单详情表。 | GET /order/{id}/ | orders、order\_items | - 多表关联查询 |
| 2. 返回菜品列表、支付状态和金额信息。 | - 结果聚合与格式化 |
|  | 取消订单 | 1. 用户发起取消请求，仅允许取消未处理的订单。 | DELETE /order/{id}/ | orders、order\_items、dishes | - 乐观锁恢复库存 |
| 2. 更新订单状态为“已取消”。 | - 消息队列解耦业务逻辑 |
| 3. 触发消息队列通知厨房系统。 |  |
| 菜品评价模块 | 查看菜品评价 | 1. 用户进入菜品详情页，请求评价列表。 | GET /dishes/{id}/reviews | dish\_reviews、users | - 分页算法（offset计算） |
| 2. 后端根据菜品ID查询评价表，按时间或评分排序。 | - 多表关联查询（联查用户表获取昵称） |
| 3. 分页返回评价数据（评分、评论内容、用户昵称）。 |  |
|  | 提交菜品评价 | 1. 用户填写评分和评论，提交评价。 | POST /reviews/create | dish\_reviews、dishes | - 评分更新算法（动态计算菜品平均分） |
| 2. 后端校验用户登录状态及数据合法性（评分范围、评论长度）。 |
| 3. 校验用户是否购买过该菜品（可选）。 |
| 4. 插入评价表并更新菜品平均分。 |
|  | 删除菜品评价 | 1. 用户请求删除评价。 | DELETE /reviews/{id} | dish\_reviews、dishes | - 权限校验（用户ID匹配） |
| 2. 后端验证用户权限。 | - 事务操作（删除评价+更新统计） |
| 3. 删除评价表记录，同步更新菜品平均分。 |  |

### 4.1.2功能流程图



## 4.2输入项和输出项（程序的输入和输出）

### 4.2.1输入项

|  |  |  |  |
| --- | --- | --- | --- |
| 输入项 | 来源 | 描述 | 数据格式/约束 |
| 用户扫码信息 | 用户扫描餐厅二维码 | 包含餐桌ID数据 | JSON: {tableId: string, shopId: string, scanTime: timestamp} |
| 菜品选择 | 用户操作 | 所选菜品ID、数量、特殊要求（如忌口） | Array: [{dishId: string, qty: number, remark?: string}] |
| 用户注册信息 | 用户填写表单 | 姓名、账号、密码 | 密码强度: 最少8位含大小写+数字 |
| 账户偏好 | 系统自动收集 | 历史订单偏好（如常点菜品）、手动设置的饮食禁忌 | 结构: {favTags: string[], restrictions: string[]} |
| 支付确认 | 用户支付行为 | 微信支付、生物识别验证结果（可选） | 枚举: paymentMethod: "WECHAT"｜"ALIPAY" |
| 生物识别: {faceAuth?: boolean} |
| 评价内容 | 用户提交评价 | 评分（1-5星）、文字评价、图片（最多3张） | 评分必填，文字最少10字，图片格式: jpg/png（单张≤2MB） |

### 4.2.2输出项

|  |  |  |  |
| --- | --- | --- | --- |
| 输出项 | 展示位置 | 描述 | 数据示例 |
| 菜品信息 | 菜单页/详情页 | 带分类的菜品列表，含实时库存状态 | {category: "招牌菜", dishes: [{id: "D001", name: "红烧肉", price: 38, stock: 15}]} |
| 购物车内容 | 购物车弹窗 | 已选菜品小计、总价（区分原价与优惠价） | {items: [{name: "红烧肉", qty: 2, price: 38}], total: 76, discount: 10} |
| 订单确认信息 | 订单提交页 | 订单号、预计备餐时间、可用的优惠券 | {orderNo: "NO202311011234", eta: "30分钟", coupons: ["满50减10"]} |
| 支付结果 | 支付完成页 | 支付状态、实际支付金额等 | {status: "SUCCESS", paidAmount: 66, pointsEarned: 20} |
| 实时订单状态 | 订单跟踪页 | 当前状态（待接单/制作中）、进度条百分比 | {status: "COOKING", progress: 60%, chef: "王师傅"} |
| 历史订单 | "我的订单"页 | 可筛选的订单列表，含订单状态、时间、菜品快照 | [{orderNo: "NO202310015678", status: "DONE", dishes: ["红烧肉 x1"], time: "2023-10-01 12:30"}] |

## 4.3算法

**推荐算法**：基于用户历史订单和热门菜品构建推荐列表

**购物车计算逻辑**：实时合计菜品总价，支持加减数量自动更新。

**输入验证算法**：用于注册与登录信息的校验，如用户名正则判断、密码长度检测等。

**本地缓存处理**：使用微信小程序 API wx.setStorageSync 和 wx.getStorageSync 进行本地收藏、购物车等数据的存储和读取，提高响应速度。

**条件逻辑判断**：下单模块中对库存量、菜品状态、是否重复收藏等进行条件控制与提示反馈

## 4.4存储分配

### 4.4.1 前端存储

|  |  |  |
| --- | --- | --- |
| 数据类型 | 存储方式 | 说明 |
| 登录Token | wx.setStorageSync | 加密存储 |
| 购物车数据 | 内存+本地缓存 | 数据存储 |

### 4.4.2 后端存储

|  |  |  |
| --- | --- | --- |
| 数据表 | 关键字段 | 优化措施 |
| 用户表 | user\_id, last\_login | 密码加密 |
| 菜品表 | dish\_id, price, real\_time\_stock | Redis缓存热点数据 |
| 订单表 | order\_id, user\_id, status, amount | 按月分表 |

## 4.5测试计划

### 4.5.1单元测试

用户登录：测试正确/错误密码的响应

优惠券计算：测试不同订单金额下的最优优惠券匹配

### 4.5.2集成测试

完整点餐流程：从登录 → 选菜 → 使用优惠券 → 支付 → 订单状态更新

支付回调：模拟支付成功/失败，检查订单状态是否同步

### 4.5.3性能测试

高并发点餐：模拟多用户同时下单，检查系统响应时间

数据库压力测试：检查订单表在较大数据量下的查询效率

# 5.管理员设计

## 5.1程序说明

管理员模块主要是对系统信息的管理，包括菜品管理、优惠券管理、订单管理、用户管理、对餐厅数据和经营状态的查看。

## 5.2 功能

**菜品管理模块**

管理菜品信息（菜品名称、价格、图片、描述、类别）

**套餐管理模块**

管理套餐信息（菜品，价格）

**订单管理模块**

可以更新订单状态，查看订单的具体信息

**优惠券系统模块**

管理优惠券信息（折扣、有效日期）

创建新优惠券以及发放优惠券

**实时经营模块**

通过界面查看餐厅的即时运营数据（餐桌状态、当日营业额）

**数据统计模块**

通过收集并分析餐厅运营数据，生成运营报告辅助餐厅经营

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模块 | 功能 | 程序流程 | 接口 | 涉及数据表 | 关键算法/机制 |
| 菜品管理模块 | 管理菜品信息 | 1. 管理员对菜品的信息进行修改 | PUT /dish/update/ | dishes、dish\_categories |  |
| 2. 后端联查菜品表和分类表，对表内数据进行修改。 |
|  | 增加菜品 | 1. 管理员对菜品的信息进行修改 | POST /dishes/create/ | dishes、dish\_categories | - |
| 2. 后端联查菜品表和分类表，对表内数据进行修改。 |
|  | 菜品上下架 | 1. 管理员对菜品的状态进行修改 | POST | dishes | - |
| 2. 后端联查菜品表，对表内数据进行修改。 | /dish/status/ |
| 套餐管理模块 | 编辑套餐 | 1. 管理员对套餐的信息进行修改 | PUT | package | - |
| 2. 后端联查套餐表，对表内数据进行修改。 | /package/update/ |
|  | 创建套餐 | 1. 管理员创建套餐 | POST | package | - |
| 2. 后端查询套餐表，对表内数据进行修改 | /package/create/ |
|  | 上下架套餐 | 1.管理员上下架套餐 | POST | package | - |
| 2.后端查询套餐表，对表内数据进行修改 | /package/status/ |
|  | 查看套餐列表 | 1. 管理员请求套餐相关数据 | GET | package | - |
| 2. 后端查询套餐表 | /package/list/ |
| 3. 返回套餐相关数据 |  |
| 实时经营模块 | 查看系统数据 | 1. 管理员请求系统数据 | GET /report/summary/ | Orders/dishes |  |
| 2. 联查系统数据相关表单 |
| 3. 返回具体表单信息 |
|  | 查看餐桌状态 | 1. 管理员请求餐桌状态相关数据 | - | - |  |
| 2. 联查餐桌状态表单 |
| 3. 返回餐桌状态 |
| 优惠券系统模块 | 创建优惠券 | 1. 管理员新建优惠券 | POST /coupon/create/ | coupons、user\_coupons |  |
| 2. 后端查询优惠券表并更新表内内容 |
|  | 发送优惠券 | 1. 管理员发送优惠券 | POST /coupon/assign/ | user\_coupons、coupons |  |
| 2. 后端将优惠券与用户关联，并更新用户-优惠券关联表 |
| 订单管理模块 | 查看订单列表 | 1. 管理员请求订单详情，后端联查订单表和订单详情表。 | GET /order/manage/ | orders、order\_items |  |
| 2. 返回菜品列表、支付状态和金额信息。 |
|  | 修改订单状态 | 1. 管理员对订单状态进行修改，仅允许修改未结束的订单。 | PUT /order/update/ | orders、order\_items |  |
| 2. 修改菜品库存，更新订单状态。 |
| 3. 触发消息队列通知厨房系统。 |

## 5.3 输入项和输出项

输入项

|  |  |  |  |
| --- | --- | --- | --- |
| 输入项 | 来源 | 描述 | 数据格式/约束 |
| 菜品新增信息 | 管理员后台操作 | 新增菜品基础信息 | JSON: {  name: string(2-20字),  price: number(≥0.01),  category: string(预定义分类),  image: base64/jpeg/png(≤5MB)  } |
| 订单状态更新 | 管理员后台操作 | 修改订单处理状态 | 枚举: {  status: "PREPARING"｜"SERVING"｜"COMPLETED"｜"CANCELLED",  reason?: string(可选取消原因)  } |
| 优惠券配置参数 | 管理员后台操作 | 创建/编辑优惠券规则 | JSON: {  type: "FIXED"｜"PERCENT",  value: number(0.1-1000),  validDays: number(1-30),  threshold: number(≥0)  } |
| 员工管理数据 | 系统同步 | 新增/编辑员工账户 | {  name: string(2-10字),  role: "CHEF"｜"WAITER"｜"MANAGER",  mobile: /^1[3-9]\d{9}$/  } |
| 经营报表参数 | 管理员筛选操作 | 生成数据分析报表 | {  startTime: ISO8601,  endTime: ISO8601(≥startTime),  granularity: "HOURLY"｜"DAILY"｜"WEEKLY"  } |
| 套餐基本信息 | 管理员后台操作 | 套餐名称、描述、主图 | JSON: {  name: string(3-20字, 唯一),  description: string(10-200字),  coverImage: base64/jpeg/png(≤5MB)  } |

输出项

|  |  |  |  |
| --- | --- | --- | --- |
| 输出项 | 展示位置 | 描述 | 数据示例 |
| 菜品信息 | 菜单页/详情页 | 带分类的菜品列表，含实时库存状态 | {category: "招牌菜", dishes: [{id: "D001", name: "红烧肉", price: 38, stock: 15}]} |
| 实时订单状态 | 订单跟踪页 | 当前状态（待接单/制作中）、进度条百分比 | {status: "COOKING", progress: 60%, chef: "王师傅"} |
| 优惠券信息 | 优惠券详情页 | 优惠券编号，优惠码，最低优惠金额，截止时间 | {  "couponId": "C202310011234", // 优惠券唯一标识  "amount": "10.00", // 面值  "miniamount": 50.00, // 最低使用门槛金额  "expire\_at": {  "end": "2023-10-31T23:59:59+08:00", // 过期时间  "code":25663556//优惠码} |
| 套餐信息 | 套餐详情页 | ID，套餐名，简介，价格，状态，创建时间 | {package\_id：1  name：家庭欢乐套餐description：适合4-5人，包含3荤2素1汤，附送水果拼盘。Price：198.00  is\_available：1  created\_at：2023-10-01 10:00:0} |

## 5.4 算法

**优惠券使用算法**：对优惠券的使用情况进行逻辑判断，并做出相应响应

## 5.5存储分配

前端存储：

|  |  |  |
| --- | --- | --- |
| 数据类型 | 存储方式 | 说明 |
| 登录Token | localstorage | 加密存储 |
| 菜品/用户头像图片 | localstorage | 网页前端缓存 |

后端存储：

|  |  |  |
| --- | --- | --- |
| 数据表 | 关键字段 | 优化措施 |
| 用户表 | user\_id， last\_login | 密码加密 |
| 菜品表 | dish\_id, price, real\_time\_stock | Redis缓存热点数据 |
| 订单表 | order\_id, user\_id, status, amount | 按月分表 |
| 套餐表 | Package\_id, name, price | 与订单同分片 |
| 餐桌表 | Table\_id,status | 缓存顺序改变 |

## 5.6测试计划

**功能测试**：

优惠券发放：测试优惠券是否能正常发放并与用户利息起来

实时经营：测试实时经营页面能否正常展示餐厅情况

**性能测试**：

高峰时期并发操作，多人多进程的实现

**安全测试**：

越权访问的测试

# 6.超级管理员设计说明

## 6.1程序说明

超级管理员模块的基础功能与管理员类似，都可对基础的模块进行内容的管理。除此之外，超级管理员能够生成每张桌上的二维码，实现桌与桌之间的独立点餐。

## 6.2功能（需要一张功能流程图）

超级管理员功能与管理员大致相同

额外功能

**餐桌二维码生成功能**

可为每个餐桌生成唯一的二维码

**管理管理员信息功能**

管理管理员账户的基础信息，增加管理员账户，删除管理员账户

查看管理员列表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模块 | 功能 | 程序流程 | 接口 | 涉及数据表 | 关键算法/机制 |
| 管理管理员信息 | 添加管理员账号 | 1. 高级管理员添加管理员账号 | POST | admin\_infos |  |
| 2. 后端联查管理员信息表，对表内数据进行修改。 | manager/create\_admin/ |
|  | 删除管理员账号 | 1. 高级管理员删除管理员账号 | DELETE | admin\_infos | - |
| 2. 后端联查管理员信息表，对表内数据进行修改。 | manager/delete\_admin/ |
|  | 启/禁用管理员 | 1. 高级管理员更改管理员账号状态 | POST | admin\_infos |  |
| 2. 后端联查管理员信息表，对表内数据进行修改 | manager/toggle\_admin/ |
|  |  |
|  | 管理员列表查询 | 1. 高级管理员请求管理员详情，后端联查管理员信息。 | GET | admin\_infos |  |
| 2. 返回管理员信息。 | manager/admin\_list/ |
| 餐桌二维码 | 二维码生成 | 1. 高级管理员请求二维码的生成 | POST | tables |  |
| 2. 后端根据逻辑生成二维码 | manager/table/generate/ |
|  | 二维码查看 | 1. 高级管理员请求二维码详情，后端联查餐桌表。 | GET | Tables |  |
| 2. 返回二维码信息。 | manager/table/qrcodes/ |

## 6.3输入项和输出项（程序的输入和输出）

输入项：

|  |  |  |  |
| --- | --- | --- | --- |
| 输入项 | 来源 | 描述 | 数据格式/约束 |
| 管理员管理数据 | 系统同步 | 新增/编辑管理员账户 | {  name: string(2-10字),  role: "CHEF"｜"WAITER"｜"MANAGER",  mobile: /^1[3-9]\d{9}$/  } |

输出项：

|  |  |  |  |
| --- | --- | --- | --- |
| 输出项 | 展示位置 | 描述 | 数据示例 |
| 餐桌信息 | 二维码获取页面 | 获得餐桌唯一二维码 | {  qrcode\_url：char  }  {  Name : ...  status:..  } |
| 管理员信息 | 管理员展示界面 | 获取所有管理员信息 |

## 6.4算法

**二维码生成算法**：每张餐桌都有自己唯一的二维码，二维码必须与餐桌对应且互不相同

## 6.5存储分配

前端存储：

|  |  |  |
| --- | --- | --- |
| 数据类型 | 存储方式 | 说明 |
| 登录Token | localstorage | 加密存储 |

后端存储：

|  |  |  |
| --- | --- | --- |
| 数据表 | 关键字段 | 优化措施 |
| 管理员表 | manage\_id, last\_login | 密码加密 |
| 餐桌表 | Table\_id,status | 缓存顺序改变 |

## 6.6测试计划

**功能测试**：

测试二维码能否顺利生成以及二维码唯一性是否满足

**性能测试**：

高峰时期并发操作，多人多进程的实现

**安全测试**：

越权访问的测试

# 7.页面设计

## 7.1用户端设计

### 7.1.1页面展示

用户端主要包括以下页面：

1.菜单浏览页面  
 展示菜品分类，支持关键词搜索与扫码点餐入口。

2.菜品详情页面  
 展示菜品图片、描述、评分、价格、月销量等信息；可选择规格、份数并添 加至购物车。

3.扫码点餐页面  
 扫描餐桌二维码后，自动绑定桌号，进入点餐模式。

4.订单管理页面  
 展示用户所有历史订单，支持根据状态（已下单、待支付、已完成、已取消 等）进行筛选。

5.订单详情页面  
 展示某笔订单的详细信息，包括菜品列表、价格明细、支付状态、配送信息 等。

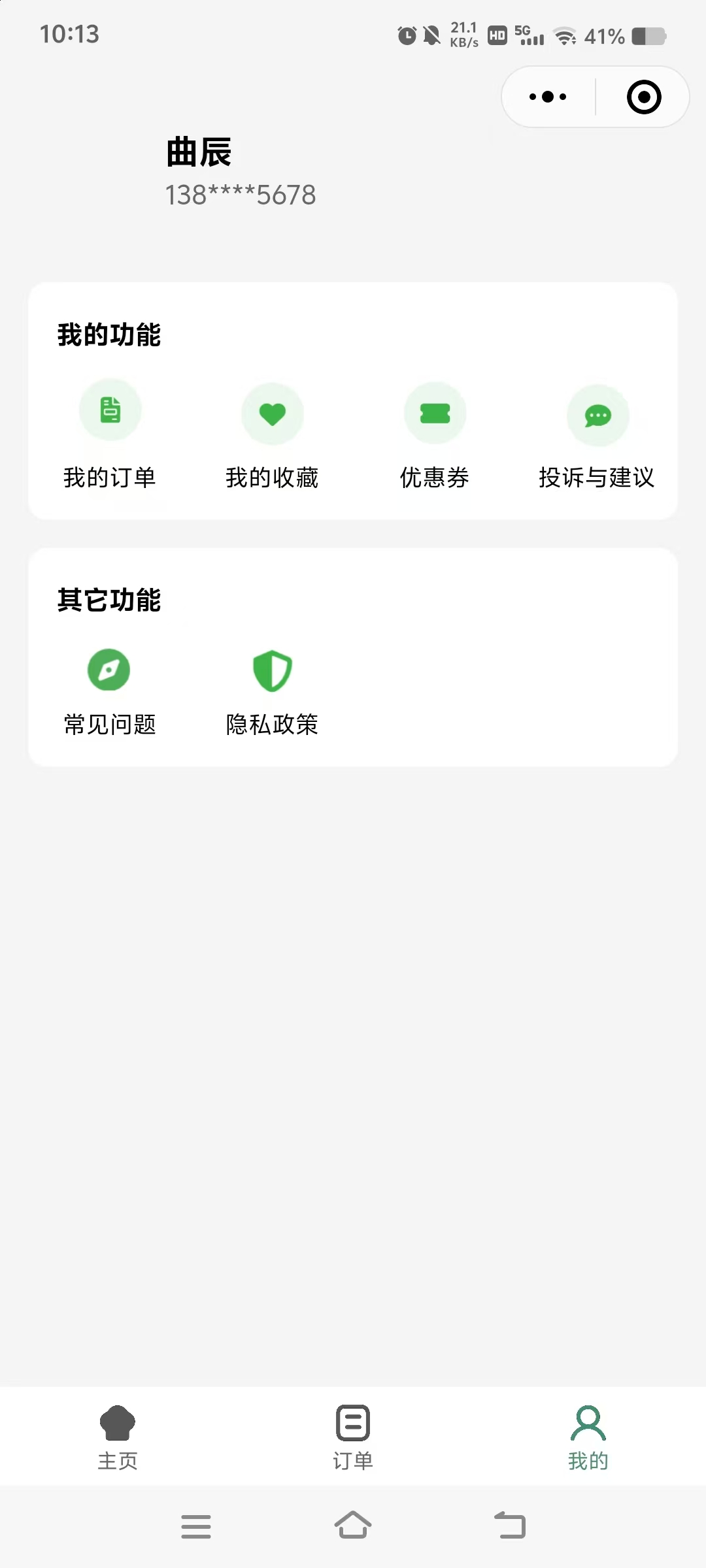
6.用户个人中心页面  
 展示用户基本信息，如头像、昵称；包含常用服务与其他服务入口。

7.我的收藏页面  
 展示用户收藏的菜品，可点击进入详情或取消收藏。

8.优惠券页面  
 展示用户所持有的优惠券，包括可用、已用、过期分类标签。

9.登录页面  
 提供账号密码登录、微信授权登录等方式，绑定用户身份。







### 7.1.2处理逻辑

菜单浏览：

onTabChange()：切换菜品分类；

onSearch()：根据关键词过滤菜品；

goToDishDetail(dishId)：跳转至菜品详情页；

onScan()：扫码绑定桌号并跳转点餐页；

onRefresh()：下拉刷新菜单数据。

菜品详情：

selectSpecification()：选择菜品规格；

addToCart()：添加菜品至购物车；

showRating()：查看用户评价列表。

扫码点餐页面：

bindTableId()：自动绑定二维码中的桌号；

navigateToMenu()：跳转至绑定桌号后的菜单浏览页面。

订单管理：

switchTab(status)：切换状态标签；

updateFilteredOrders()：更新过滤结果；

goToDetail(orderId)：查看订单详情。

订单详情页面：

loadOrderDetail(orderId)：加载订单详细信息；

reorder()：再次下单；

requestRefund()：申请退款等操作。

用户个人中心：

直接读取登录态信息展示头像、用户名；

点击“我的订单”等服务项，使用 wx.navigateTo() 跳转到对应页面。

我的收藏：

getFavorites()：获取收藏菜品列表；

removeFavorite(dishId)：取消收藏；

goToDishDetail()：跳转至详情。

优惠券页面：

getCoupons()：获取优惠券信息；

分类标签切换展示“可用”、“已用”、“已过期”优惠券；

在下单时自动匹配最优优惠券。

登录页面：

onWeChatLogin()：微信一键授权；

bindUserProfile()：首次登录时绑定用户信息。

## 7.2管理端设计

### 7.2.1页面设计

管理端主要包括以下页面：

1.管理员管理界面

仅超级管理员可见，用于查看和管理所有管理员账户（创建、修改、启用/ 禁用、删除）。

2.用户管理界面

展示所有注册用户基本信息，例如注册时间；

可进行冻结账户、查看用户订单、重置密码等操作。

3.餐厅平面图可视化界面

展示餐厅布局，包括餐桌、隔断墙，实时反映餐桌状态；

支持布局编辑和状态监控。

4.菜品管理界面

支持创建新菜品、编辑已有菜品、上架/下架、价格修改、查看评论与历史 价格；

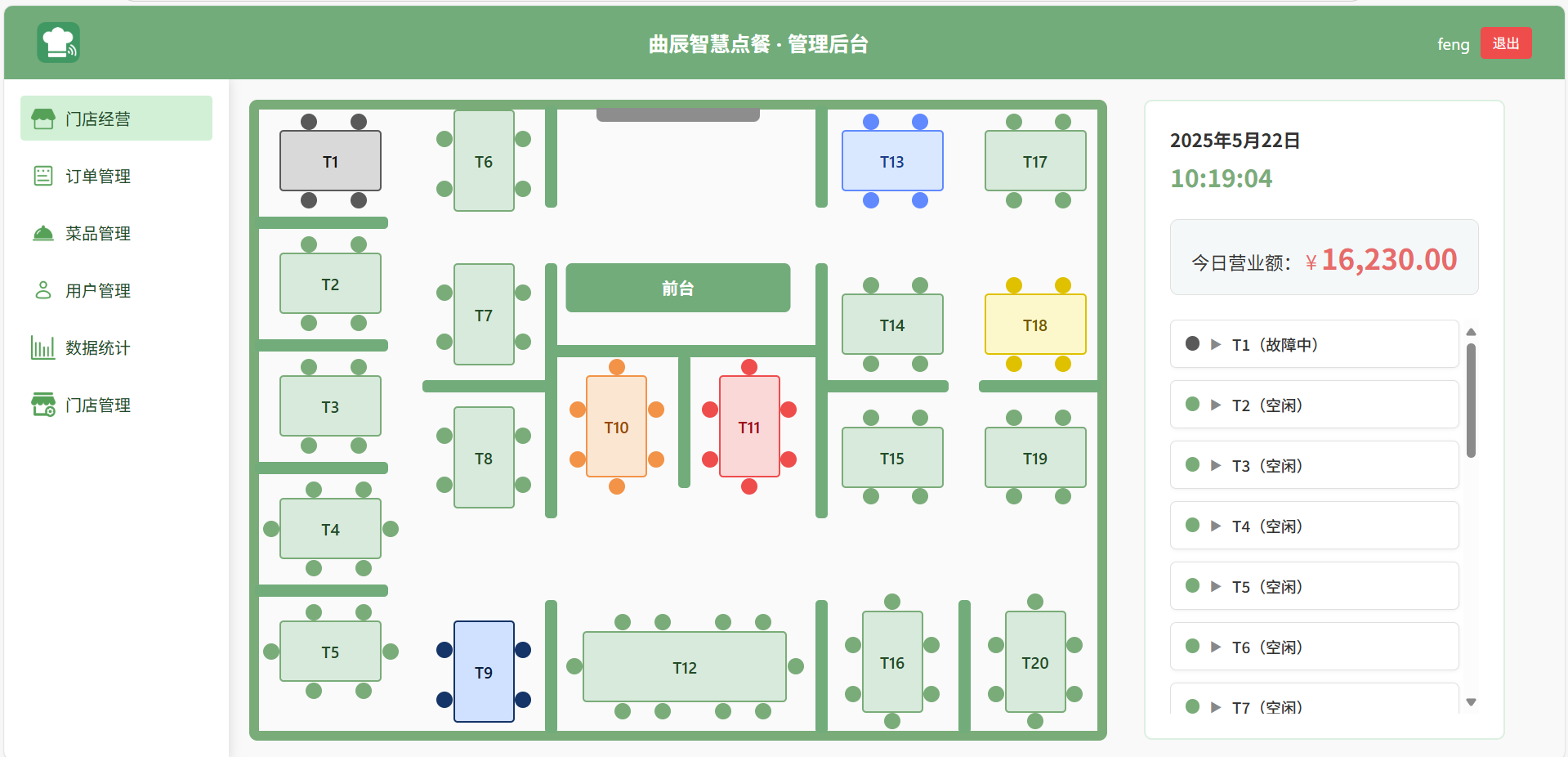
可按分类、状态进行筛选。

5.数据统计页面

展示关键运营数据指标，如日订单数、销售额、菜品热度、活跃用户数等；

图表展示采用 ECharts 实现，支持按时间维度筛选。









### 7.2.2处理逻辑

管理员管理：

fetchAdminList()：获取管理员列表；

createAdmin()：新增管理员；

changePassword()：修改密码；

confirmStatusChange()：启用/禁用管理员；

confirmDelete()：删除管理员账户。

用户管理（新增）：

fetchUserList()：拉取所有用户信息；

searchUser(keyword)：按关键词过滤用户；

freezeAccount(userId)：冻结用户；

resetPassword(userId)：重置密码；

viewUserOrders(userId)：跳转至用户订单查看页。

餐厅平面图可视化：

RestaurantTable：渲染餐桌，设置状态、方向、容量；

PartitionWall：绘制隔断墙，支持方向和长度设置；

setTableTheme(tableId, status)：设置餐桌状态颜色编码。

菜品管理：

fetchDishes()：加载所有菜品；

filterDishes()：分类和状态筛选；

updateDishPrice()：修改价格并记录价格历史；

toggleDishStatus()：菜品上下架；

viewPriceHistory()：ECharts 图表显示价格变动。

数据统计页面：

getChartDataByRange(startDate, endDate)：按日期区间查询趋势；

renderECharts()：渲染订单、销售、用户等相关图表。

# 8.性能设计

|  |  |
| --- | --- |
| 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）+ Nginx 反向代理 |
| 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）；可选 RQ 任务队列 |
| 管理后台 | Vue 3 + Vite + Element Plus；运行时 Node.js 18 LTS |
| 部署 | • 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 | Nginx 1.24 终端 TLS；启用 GZIP、WebP、HTTP keep-alive |
| 支付 | 预留 微信支付 v3；需使用平台证书加密与回调验签 |
| 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 + Prettier；Git 提交遵循 Conventional Commits |