

湖南科技大学计算机科学与工程学院

综合实训 课程设计报告

专业班级: 计算机科学与技术七班

姓 名: 陈琪琪

学 号: 2102010629

指导教师: 林剑

时 间: 2023.05.22-2023.06.03

地 点: 逸夫楼 416

指导教师评语:

成绩:

等级:

签名: _____
年 月 日

网上订餐系统

一. 实验题目

基于 SpringBoot 框架开发的网上订餐系统

二. 实验要求

课程设计目的：

WEB 应用技术课程设计是计算机专业的一个综合性实践教学环节，是学习完《Internet 与 Web 编程》课程后进行的一次全面的综合练习。其目的在于促进学生复习和巩固计算机软件设计知识，加深对软件设计方法、软件设计技术和设计思想的理解，并能运用所学软件设计知识和 WEB 工程技术进行综合软件设计，通过本课程设计让学生进行简单 WEB 应用软件系统开发，掌握软件设计的方法和面向对象程序设计的基本技术，提高学生在软件开发方面综合应用能力。

功能要求：

- 门店管理
- 菜品管理
- 用户注册
- 用户点餐、查看修改菜单
- 模拟支付
- 订单管理
- 配送管理

基本要求：

1. 熟悉 Tomcat、Weblogic、Websphere、JBoss、Nginx 等实际工作中常用的 Web 应用服务器。
2. 熟悉一种开发工具的 Web 编程模式。
3. 设计并实现一个具体的 Web 应用系统。
 - (1) 选定的项目进行需求分析，写出需求说明书，并搜集、整理所需素材。
 - (2) 根据项目需求，进行数据库的概要设计与详细设计。
 - (3) 根据需求说明进行项目的功能设计，画出每个界面的原型。
 - (4) 进行详细设计，实现每个模块的功能。
 - (5) 对项目进行部署与测试。
 - (6) 分析总结项目的创新点和存在的不足，提出优化思路。

三. 总体设计

1. 背景知识

| | |
|-----|---|
| 用户层 | 本项目中在构建系统管理后台的前端页面，会用到H5、Vue.js、ElementUI等技术。 |
| 网管层 | 项目部署使用的是Tomcat，作用①管理servlet应用程序的生命周期。②将客户端请求的url映射到相应的servlet③配合Servlet程序处理HTTP请求 |
| 应用层 | SpringBoot：快速构建Spring项目，采用 "约定优于配置" 的思想，简化Spring项目的配置开发。 |
| | Spring：统一管理项目中的各种资源(bean)，在web开发的各层中都会用到。 |
| | SpringMVC：SpringMVC是spring框架的一个模块，springmvc和spring无需通过中间整合层进行整合，可以无缝集成。 |
| | SpringSession：主要解决在集群环境下的Session共享问题。 |
| | lombok：能以简单的注解形式来简化java代码，提高开发人员的开发效率。 |
| | Swagger：可以自动的帮助开发人员生成接口文档，并对接口进行测试。 |
| 数据层 | MySQL：关系型数据库，本项目的核心业务数据都会采用MySQL进行存储。 |
| | MybatisPlus：本项目持久层将会使用MybatisPlus来简化开发，基本的单表增删改查直接调用框架提供的方法即可。 |
| 工具 | maven：项目构建工具。 |
| | junit：单元测试工具，开发人员功能实现完毕后，需要通过junit对功能进行单元测试。 |

2. 模块介绍

(1) 后台管理：

①管理员或员工信息管理模块

这一模块负责管理员工或者管理员的信息以及操作权限，管理员的权限是最大的，员工有权限限制，如不能修改他人信息。同时管理员具有增加员工、编辑员工信息操作以及启停用员工操作。

②分类管理模块

该模块涉及对菜品的分类，例如：本次课程设计的背景为售卖汉堡，从而售卖产品的分类可以是单个菜系，也可以是套餐，套餐里可以有多个菜品，实现菜品组合优惠售卖，与现实订餐系统类似。同时具有新增菜品分类，新增套餐分类，修改，删除菜品或套餐操作。

③菜品管理模块

此模块涉及对菜品的批量或单个删除、启用以及停售菜品操作，修改菜品信

息，根据输入信息查询菜品信息，新增菜品信息操作。同时新增与修改操作里涉及口味选择，上传图片等操作。

④套餐管理模块

此模块操作同菜品管理模块相似，不同点在于修改和新增操作里可以选择菜品的数量。

⑤订单明细模块

此模块主要记录客户的订单信息，里面可以根据订单时间或是订单号来查询某个或某些订单，同时可以具有查看订单信息以及选择派发或完成订单指令的操作。

⑥管理员或者员工登录退出模块

该部分涉及登录，退出基本操作

(2) 客户端：

①购物车管理模块

此模块涉及管理用户的购物车信息，包含对菜品的增添，删除，查看，结算操作。

②用户地址管理模块

此模块里可以进行新增地址，删除地址，修改地址以及设置默认地址操作。

③订单信息管理模块

此模块主要负责给予用户历史订单信息的呈现，以及最新订单信息的呈现。

④菜品浏览模块

此模块根据在后台中进行的菜品分类呈现给用户，让用户能够快速定位其需要的菜品，同时这里也进行了用户选择口味的操作，对菜品的增加或减少操作。

⑤用户管理模块

此模块涉及模拟用户使用验证码登录，以及注销操作。

(3) 统一管理：

①拦截器

为防止用户或者员工非法操作，设置了拦截器，仅对一些访问路径进行了放行操作，其他的须在登录状态下才可访问链接。

②文件管理模块

由于涉及到图片的上传，以及浏览器图片的展示功能，且是用户端和管理端












均需的操作，从而放置一起统一管理。这里面负责将员工或管理员上传的图片存储在本地磁盘上，以及每次用户或者管理员，员工查看菜品图片时，浏览器通过图片存储在数据库的名字来下载相应的图片。

③其他资源管理模块

此模块包含静态资源映射，配置 MybatisPlus 的分页插件，基于 ThreadLocal 封装工具类，自定义业务异常，全局异常处理，JSON 反序列化 java 对象或 java 对象序列化 JSON 操作，公共字段统一管理（如创建时间，更新时间，创建人，更新人的自动插入或修改），返回的数据统一封装成 R 实体，随机生成验证码以及短信发送工具类等模块。


3. 数据库

(1) 项目涉及的表：

| | |
|---|---------------|
|  | address_book |
|  | category |
|  | dish |
|  | dish_flavor |
|  | employee |
|  | order_detail |
|  | orders |
|  | setmeal |
|  | setmeal_dish |
|  | shopping_cart |
|  | user |

(2) 各表结构展示：

①address_book 表(地址簿表)：

| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|---------------|----------|-----|-----|-------------------------------------|--------------------------|--|------------|
| id | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> |  1 | 主键 |
| user_id | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 用户id |
| consignee | varchar | 50 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 收货人 |
| sex | tinyint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 性别 0 女 1 男 |
| phone | varchar | 11 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 手机号 |
| province_code | varchar | 12 | | <input type="checkbox"/> | <input type="checkbox"/> | | 省级区划编号 |
| province_name | varchar | 32 | | <input type="checkbox"/> | <input type="checkbox"/> | | 省级名称 |
| city_code | varchar | 12 | | <input type="checkbox"/> | <input type="checkbox"/> | | 市级区划编号 |
| city_name | varchar | 32 | | <input type="checkbox"/> | <input type="checkbox"/> | | 市级名称 |
| district_code | varchar | 12 | | <input type="checkbox"/> | <input type="checkbox"/> | | 区级区划编号 |
| district_name | varchar | 32 | | <input type="checkbox"/> | <input type="checkbox"/> | | 区级名称 |
| detail | varchar | 200 | | <input type="checkbox"/> | <input type="checkbox"/> | | 详细地址 |
| label | varchar | 100 | | <input type="checkbox"/> | <input type="checkbox"/> | | 标签 |
| is_default | tinyint | 1 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 默认 0 否 1 是 |
| create_time | datetime | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 创建时间 |
| update_time | datetime | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 更新时间 |

②category 表(分类表)：

| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|-------------|----------|----|-----|-------------------------------------|--------------------------|---|------------------|
| id | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> |  1 | 主键 |
| type | int | | | <input type="checkbox"/> | <input type="checkbox"/> | | 类型 1 菜品分类 2 套餐分类 |
| name | varchar | 64 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 分类名称 |
| sort | int | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 顺序 |
| create_time | datetime | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 创建时间 |
| update_time | datetime | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 更新时间 |
| create_user | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 创建人 |
| update_user | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 修改人 |
| is_deleted | varchar | 20 | | <input type="checkbox"/> | <input type="checkbox"/> | | 是否删除 |

③dish 表(菜品表):

| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|-------------|----------|-----|-----|-------------------------------------|--------------------------|---|-----------|
| id | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | 1 | 主键 |
| name | varchar | 64 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 菜品名称 |
| category_id | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 菜品分类id |
| price | decimal | 10 | 2 | <input type="checkbox"/> | <input type="checkbox"/> | | 菜品价格 |
| code | varchar | 64 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 商品码 |
| image | varchar | 200 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 图片 |
| description | varchar | 400 | | <input type="checkbox"/> | <input type="checkbox"/> | | 描述信息 |
| status | int | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 0 停售 1 起售 |
| sort | int | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 顺序 |
| create_time | datetime | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 创建时间 |
| update_time | datetime | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 更新时间 |
| create_user | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 创建人 |
| update_user | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 修改人 |
| is_deleted | int | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 是否删除 |

④dish_flavor 表(口味表):

| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|-------------|----------|-----|-----|-------------------------------------|--------------------------|---|----------|
| id | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | 1 | 主键 |
| dish_id | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 菜品 |
| name | varchar | 64 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 口味名称 |
| value | varchar | 500 | | <input type="checkbox"/> | <input type="checkbox"/> | | 口味数据list |
| create_time | datetime | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 创建时间 |
| update_time | datetime | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 更新时间 |
| create_user | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 创建人 |
| update_user | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 修改人 |
| is_deleted | int | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 是否删除 |

⑤employee 表(员工表):

| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|-------------|----------|----|-----|-------------------------------------|--------------------------|---|---------------|
| id | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | 1 | 主键 |
| name | varchar | 32 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 姓名 |
| username | varchar | 32 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 用户名 |
| password | varchar | 64 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 密码 |
| phone | varchar | 11 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 手机号 |
| sex | varchar | 2 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 性别 |
| id_number | varchar | 18 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 身份证号 |
| status | int | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 状态 0:禁用, 1:正常 |
| create_time | datetime | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 创建时间 |
| update_time | datetime | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 更新时间 |
| create_user | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 创建人 |
| update_user | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 修改人 |


⑥order_detail 表(订单明细表):

| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|-------------|----------|----|-----|-------------------------------------|--------------------------|---|---------------|
| id | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | 1 | 主键 |
| name | varchar | 32 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 姓名 |
| username | varchar | 32 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 用户名 |
| password | varchar | 64 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 密码 |
| phone | varchar | 11 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 手机号 |
| sex | varchar | 2 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 性别 |
| id_number | varchar | 18 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 身份证号 |
| status | int | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 状态 0:禁用, 1:正常 |
| create_time | datetime | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 创建时间 |
| update_time | datetime | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 更新时间 |
| create_user | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 创建人 |
| update_user | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 修改人 |

⑦orders 表(订单表):

| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|-----------------|----------|-----|-----|-------------------------------------|--------------------------|---|-----------------------------------|
| id | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> |  1 | 主键 |
| number | varchar | 50 | | <input type="checkbox"/> | <input type="checkbox"/> | | 订单号 |
| status | int | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 订单状态 1待付款, 2待派送, 3已派送, 4已完成, 5已取消 |
| user_id | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 下单用户 |
| address_book_id | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 地址id |
| order_time | datetime | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 下单时间 |
| checkout_time | datetime | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 结账时间 |
| pay_method | int | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 支付方式 1微信,2支付宝 |
| amount | decimal | 10 | 2 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 实收金额 |
| remark | varchar | 100 | | <input type="checkbox"/> | <input type="checkbox"/> | | 备注 |
| phone | varchar | 255 | | <input type="checkbox"/> | <input type="checkbox"/> | | |
| address | varchar | 255 | | <input type="checkbox"/> | <input type="checkbox"/> | | |
| user_name | varchar | 255 | | <input type="checkbox"/> | <input type="checkbox"/> | | |
| consignee | varchar | 255 | | <input type="checkbox"/> | <input type="checkbox"/> | | |

⑧setmeal 表(套餐表):

| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|-------------|----------|-----|-----|-------------------------------------|--------------------------|---|--------------|
| id | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> |  1 | 主键 |
| category_id | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 菜品分类id |
| name | varchar | 64 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 套餐名称 |
| price | decimal | 10 | 2 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 套餐价格 |
| status | int | | | <input type="checkbox"/> | <input type="checkbox"/> | | 状态 0:停用 1:启用 |
| code | varchar | 32 | | <input type="checkbox"/> | <input type="checkbox"/> | | 编码 |
| description | varchar | 512 | | <input type="checkbox"/> | <input type="checkbox"/> | | 描述信息 |
| image | varchar | 255 | | <input type="checkbox"/> | <input type="checkbox"/> | | 图片 |
| create_time | datetime | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 创建时间 |
| update_time | datetime | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 更新时间 |
| create_user | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 创建人 |
| update_user | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 修改人 |
| is_deleted | int | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 是否删除 |


⑨setmeal_dish 表(套餐菜品关系表):

| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|-------------|----------|----|-----|-------------------------------------|--------------------------|---|-------------|
| id | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> |  1 | 主键 |
| setmeal_id | varchar | 32 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 套餐id |
| dish_id | varchar | 32 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 菜品id |
| name | varchar | 32 | | <input type="checkbox"/> | <input type="checkbox"/> | | 菜品名称 (冗余字段) |
| price | decimal | 10 | 2 | <input type="checkbox"/> | <input type="checkbox"/> | | 菜品原价 (冗余字段) |
| copies | int | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 份数 |
| sort | int | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 排序 |
| create_time | datetime | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 创建时间 |
| update_time | datetime | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 更新时间 |
| create_user | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 创建人 |
| update_user | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 修改人 |
| is_deleted | int | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 是否删除 |

⑩shopping_cart 表(购物车表):

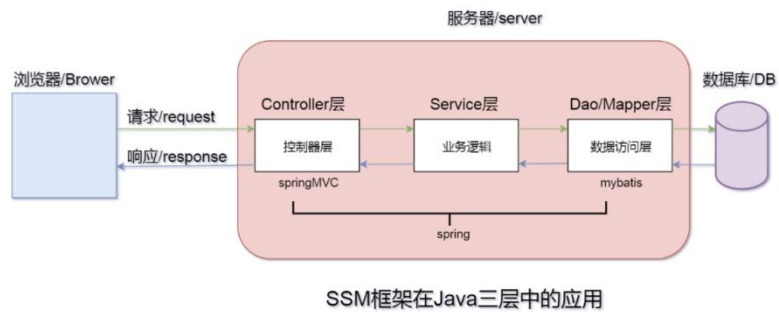
| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|-------------|----------|-----|-----|-------------------------------------|--------------------------|---|------|
| id | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> |  1 | 主键 |
| name | varchar | 50 | | <input type="checkbox"/> | <input type="checkbox"/> | | 名称 |
| image | varchar | 100 | | <input type="checkbox"/> | <input type="checkbox"/> | | 图片 |
| user_id | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 主键 |
| dish_id | bigint | | | <input type="checkbox"/> | <input type="checkbox"/> | | 菜品id |
| setmeal_id | bigint | | | <input type="checkbox"/> | <input type="checkbox"/> | | 套餐id |
| dish_flavor | varchar | 50 | | <input type="checkbox"/> | <input type="checkbox"/> | | 口味 |
| number | int | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 数量 |
| amount | decimal | 10 | 2 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 金额 |
| create_time | datetime | | | <input type="checkbox"/> | <input type="checkbox"/> | | 创建时间 |

⑪user 表(用户表):

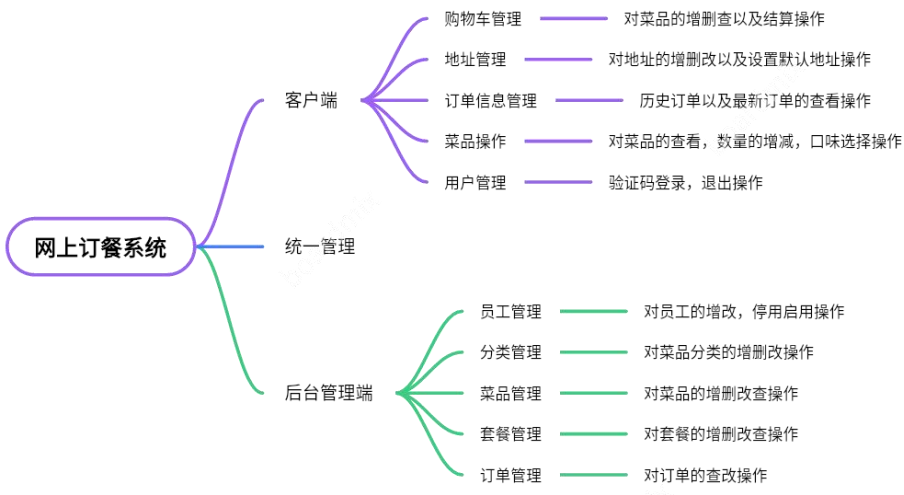
| 名 | 类型 | 长度 | 小数点 | 不是 null | 虚拟 | 键 | 注释 |
|-----------|---------|-----|-----|-------------------------------------|--------------------------|---|---------------|
| id | bigint | | | <input checked="" type="checkbox"/> | <input type="checkbox"/> |  1 | 主键 |
| name | varchar | 50 | | <input type="checkbox"/> | <input type="checkbox"/> | | 姓名 |
| phone | varchar | 100 | | <input checked="" type="checkbox"/> | <input type="checkbox"/> | | 手机号 |
| sex | varchar | 2 | | <input type="checkbox"/> | <input type="checkbox"/> | | 性别 |
| id_number | varchar | 18 | | <input type="checkbox"/> | <input type="checkbox"/> | | 身份证号 |
| avatar | varchar | 500 | | <input type="checkbox"/> | <input type="checkbox"/> | | 头像 |
| status | int | | | <input type="checkbox"/> | <input type="checkbox"/> | | 状态 0:禁用, 1:正常 |

四. 详细设计

1. SSM 框架图展示，即整体程序运行流程结构图



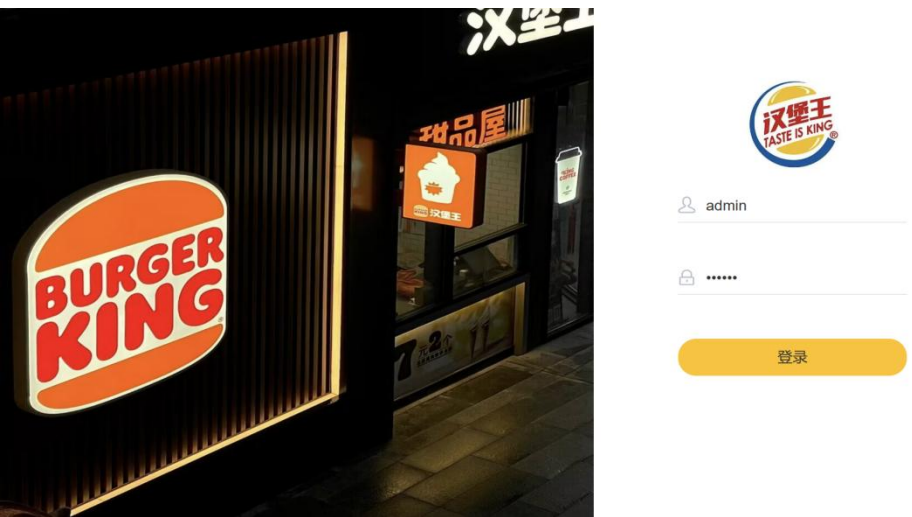
2. 程序流程图



3. 主要功能介绍

(1) 后台管理端

① 登录界面



这里涉及到拦截器，主要代码为：

```
//1. 获取本次请求的uri
String requestURI = request.getRequestURI();

log.info("拦截到请求: {}", requestURI);

//定义不需要拦截的请求路径
String[] urls = new String[]{
    "/employee/login",
    "/employee/logout",
    "/backend/**",
    "/front/**",
    "/user/sendMsg", //移动端发送短信
    "/user/login" //移动端登录
};

//2. 判断本次请求是否需要处理
boolean check = check(urls, requestURI);

//3. 如果不需要处理，则直接放行
if(check){
    log.info("本次请求{}不需要处理", requestURI);
    filterChain.doFilter(request, response);
    return;
}

//4-1. 判断登录状态，如果已登录，则直接放行
if(request.getSession().getAttribute("employee") != null){
    log.info("员工用户已登录，员工用户id为: {}", request.getSession().getAttribute("employee"));
```

```
Long empId = (Long) request.getSession().getAttribute("employee");
BaseContext.setCurrentId(empId);

filterChain.doFilter(request, response);
return;
}

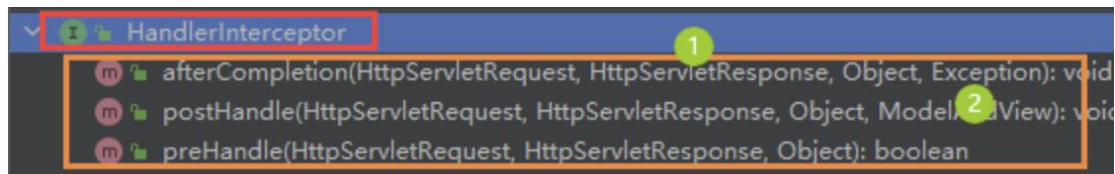
//4-2. 判断移动端用户登录状态，如果已登录，则直接放行
if(request.getSession().getAttribute("user") != null){
    log.info("用户已登录，用户id为: {}", request.getSession().getAttribute("user"));

    Long userId = (Long) request.getSession().getAttribute("user");
    BaseContext.setCurrentId(userId);

    filterChain.doFilter(request, response);
    return;
}

//5. 如果未登录则返回未登录结果，通过输出流方式向客户端页面响应数据
log.info("用户未登录");
response.getWriter().write(JSON.toJSONString(R.error("NOTLOGIN")));
```

拦截器的实现机制：



我们可以发现，拦截器是一个接口，一共有三个方法，我们可以重写这三个方法，对其进行自定义拦截。

preHandle ()：是执行目标方法前，执行该方法（通过方法的参数值也可以发现 -->request、response、handle）

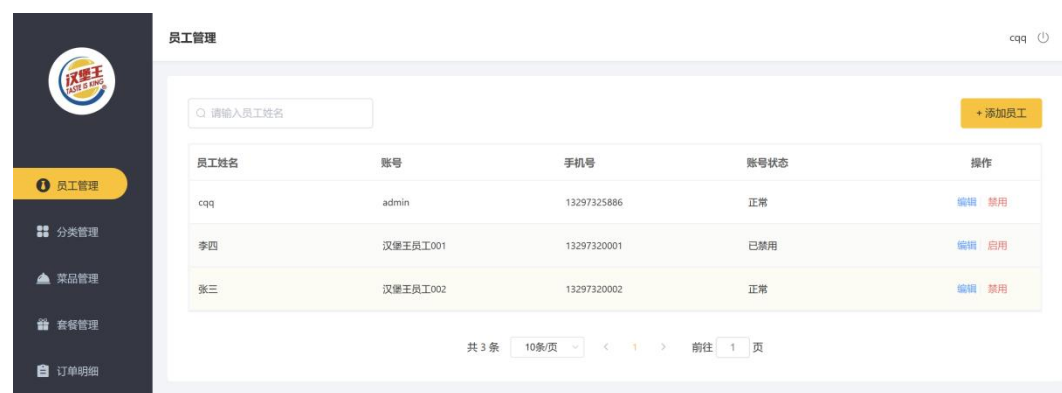
`postHandle()`：执行完成目标方法后，页面渲染前，有了返回的 `ModelAndView` 后，执行该方法。（参 --> `request`、`response`、`handle`、`modelAndView`）
`afterCompletion()`：页面渲染完后，执行该方法，其实主要是为了显示异常。（参： `request`、`response`、`handler`、`ex`）

拦截器的执行顺序为：

- 1) `preHandle()`
- 2) `postHandle()`
- 3) `afterCompletion()`

②主界面

提示：由于这里涉及到多个界面，这里就统一拿菜品管理和套餐管理进行代码解释。



菜单界面的实现：主要将每个操作界面进行编号，当点击时则跳转到相应的界面。

```
<div v-for="item in menuList" :key="item.id">
  <el-submenu :index="item.id" v-if="item.children && item.children.length>0">
    <template slot="title">
      <i class="iconfont" :class="item.icon"></i>
      <span>{{item.name}}</span>
    </template>
    <el-menu-item
      v-for="sub in item.children"
      :index="sub.id"
      :key="sub.id"
      @click="menuHandle(sub, false)"
    >
      <i :class="iconfont" :class="sub.icon"></i>
      <span slot="title">{{sub.name}}</span>
    </el-menu-item>
  </el-submenu>
  <el-menu-item v-else :index="item.id" @click="menuHandle(item, false)">
    <i class="iconfont" :class="item.icon"></i>
    <span slot="title">{{item.name}}</span>
  </el-menu-item>
</div>
```

点击，响应相应的函数 `menuHandle(sub, false)`

这里 `sub` 这个参数是前面根据遍历 (`v-for`) 得到的单行对象，里面有多个属性，如 `menuActiveId` 就是用于指定下次跳转的界面对应的编号，`iframeUrl` 则是下次

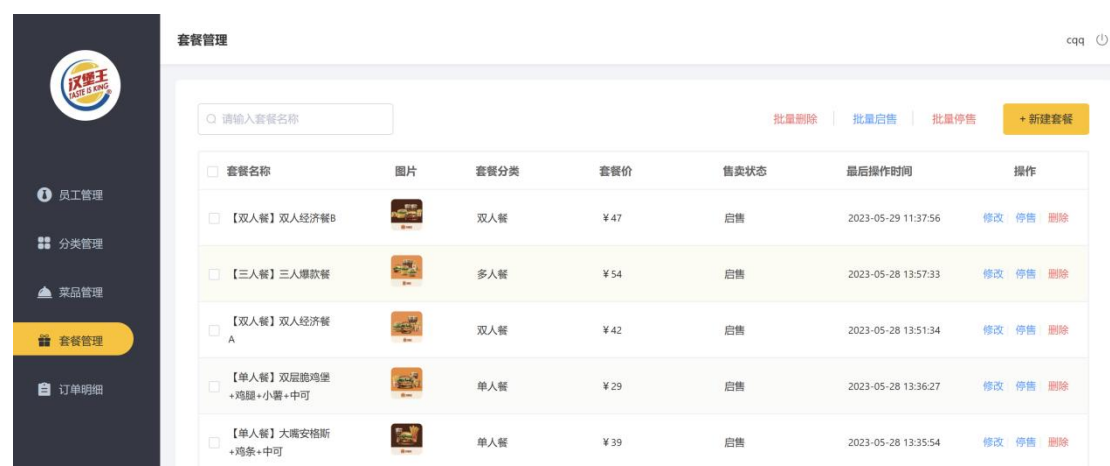
跳转界面的路径，以实现正确的页面跳转，对于 headTitle 这个属性则是对应于每次跳转界面上面的操作菜单名称。

```
goBack() {
  // window.location.href = 'javascript:history.go(-1)'
  const menu = this.menuList.find(item=>item.id===this.menuActivated)
  // this.goBackFlag = false
  // this.headTitle = menu.name
  this.menuHandle(menu,false)
},
menuHandle(item, goBackFlag) {
  this.loading = true
  this.menuActivated = item.id
  this.iframeUrl = item.url
  this.headTitle = item.name
  this.goBackFlag = goBackFlag
  this.closeLoading()
},
```

当这些参数的值得到了相应的赋值后，才可以跳转到相应的界面，其中 icon 对应的是展示在页面上相应的图标

```
{
  id: '4',
  name: '菜品管理',
  url: 'page/food/list.html',
  icon: 'icon-food'
},
```

③套餐管理界面(此部分为代码原理以及主要解释部分)

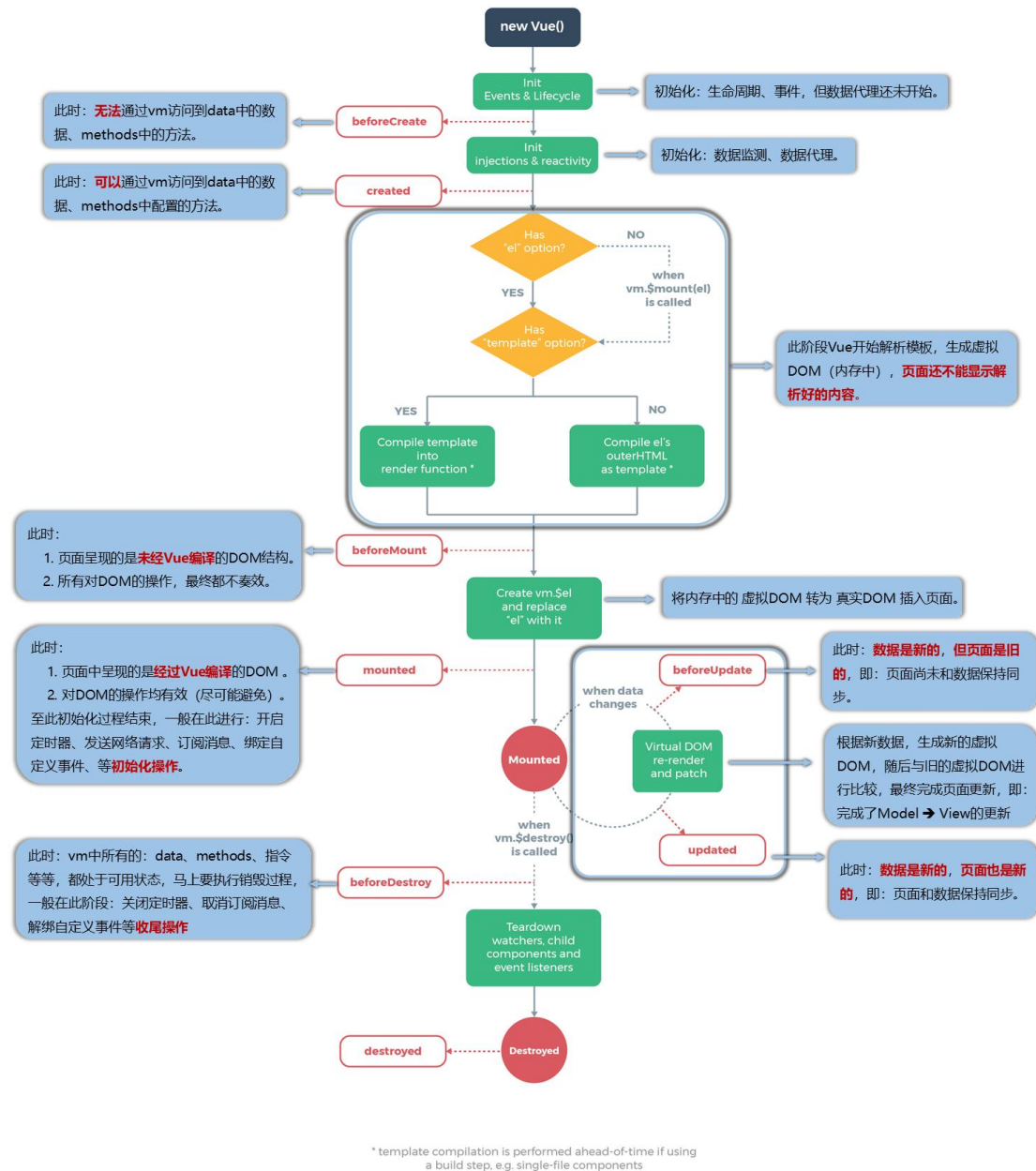


首先通过在主界面点击了相应的菜单选项，跳转至相应的页面，即套餐页面，由于本项目采用的是前端框架 vue 来实现的，所以进入时，调用 vue 框架的钩子函数 mount()。这里就涉及到 vue 的生命周期，以下就是介绍相关内容：

生命周期：

1. 又名：生命周期回调函数、生命周期函数、生命周期钩子。
2. 是什么：Vue 在关键时刻帮我们调用的一些特殊名称的函数。
3. 生命周期函数的名字不可更改，但函数的具体内容是程序员根据需求编写的。
4. 生命周期函数中的 `this` 指向是 `vm` 或 组件实例对象。

而其生命周期也可以用以下一张图来进行分析概括：



常用的生命周期钩子：

1. **mounted**：发送 ajax 请求、启动定时器、绑定自定义事件、订阅消息等【初始化操作】。
2. **beforeDestroy**：清除定时器、解绑自定义事件、取消订阅消息等【收尾工作】。

关于销毁 Vue 实例

1. 销毁后借助 Vue 开发者工具看不到任何信息。
2. 销毁后自定义事件会失效，但原生 DOM 事件依然有效。
3. 一般不会在 **beforeDestroy** 操作数据，因为即便操作数据，也不会再触发更新流程了。

由图以及以下代码可知，在 **create()** 函数时进行了初始化工作，调用了 **init()** 自定义方法。

async 介绍：

async 的用法，它作为一个关键字放到函数前面，用于表示函数是一个异步函数，因为 **async** 就是异步的意思，异步函数也就意味着该函数的执行不会阻塞后面代码的执行。

await 介绍：

async 的用法，它作为一个关键字放到函数前面，用于表示函数是一个异步函数，因为 **async** 就是异步的意思，异步函数也就意味着该函数的执行不会阻塞后面代码的执行。

注意 **await** 关键字只能放到 **async** 函数里面。

其中 **params** 表示参数，数据是以一种为 json 的格式进行传递，由于项目内涉及分页，所以将每页对应的页数，以及对应每页装载数据的数量的大小，以及搜索框内放置的检索名打包成一个对象通过 **getSetmealPage** 函数进行传递，**getSetmealPage** 函数后跟的 **then** 表示执行完后端的操作时前端所需要进行的操作，由图中给出的代码可知，这里先是做一个判断，判断后台的操作是否有误，无误则将查询出上传给前端的数据进行相应变量的赋值，其中 **tableData** 表示的是查询出的 **page** 对象(里面包含了查询的多条数据)，**counts** 表示查询数据的数量；否则则在界面显示出错的信息以方便提示用户。

```
created() {
  this.init()
},
mounted() {
},
methods: {
  async init () {
    const params = {
      page: this.page,
      pageSize: this.pageSize,
      name: this.input ? this.input : undefined
    }
    await getSetmealPage(params).then(res => {
      if (String(res.code) === '1') {
        this.tableData = res.data.records || []
        this.counts = res.data.total
      }
    }).catch(err => {
      this.$message.error('请求出错了: ' + err)
    })
  },
}
```


接下来介绍一下 `getSetmealPage` 函数：

```
const getSetmealPage = (params) => {  
  return $axios({  
    url: '/setmeal/page',  
    method: 'get',  
    params  
  })  
}
```

由此段代码可知，上传的路径为“`/setmeal/page`”，上传方式为 `get`。

这里介绍一下请求方式的类型：

常见的请求方式包括：

GET： 用于请求访问已经被 URI（统一资源标识符）识别的资源，可以通过 URL 传参给服务器；

POST： 用于传输信息给服务器，主要功能与 GET 方法类似，但一般推荐使用 POST 方式；

PUT： 传输文件，报文主体中包含文件内容，保存到对应 URI 位置；

HEAD： 获得报文首部，与 GET 方法类似，只是不返回报文主体，一般用于验证 URI 是否有效；

DELETE： 删除文件，与 PUT 方法相反，删除对应 URI 位置的文件；

OPTIONS： 查询相应 URI 支持的 HTTP 方法。

在这里我们也知道，发送的是 ajax 请求，因为代码里面有 `$axios`。

axios 介绍：

1. 什么是 axios：

axios 是一个基于 promise 用于浏览器和 nodejs 的 HTTP 客户端。简单的理解就是 ajax 的封装。

2. axios 的特征：

从浏览器中创建 XMLHttpRequest

从 node.js 发出 http 请求

支持 Promise API

拦截请求和响应

转换请求和响应数据

取消请求

自动转换 JSON 数据

客户端支持防止 CSRF/XSRF

3. axios 在使用时需要注意的细节：

引用 axios 时 `Vue.prototype.axios = axios` `Vue.prototype.$axios = axios`
`Vue.prototype.$http = axios` 其实是都一个东西，只是 vue 的原型链上加个变量（且变量不同），值是 axios 对象。

只是 一个是 jquery 封装过的异步调用方法，一个是 vue 推荐的第三方异步封装方法，他们都是调用的 axios 对象。只是调用的时候 `axios.post({...})`
`this.$axios.post({...})` `this.$http.post({...})`

通过路径我们就可以访问到后端了，以下就是 `controller` 层中相关的代码：
由于需要上传给前端是一个 `page` 对象，所以先进行一个 `Page` 对象 (`mybatisPlus` 中封装) 的创建，并且通过使用泛型限制了类型的赋值。且 `page`, `pageSize` 为创建 `Page` 时输入的参数，分别对应页数，以及每页数据的数量。

`Page` 对象的介绍，它有以下 10 个参数：

| 参数名 | 参数类型 | 默认值 | 描述 |
|------------------|-----------------|-------|---------------------------------|
| records | List<T> | | 用来存放查询出来的数据 |
| total | long | | 返回记录的总数 |
| size | long | 10 | 每页显示条数 |
| current | long | 1 | 当前页 |
| orders | List<OrderItem> | | 排序字段信息 |
| optimizeCountSql | boolean | true | 自动优化 COUNT SQL |
| isSearchCount | boolean | true | 是否进行 count 查询，设置false后不会返回total |
| hitCount | boolean | false | 是否命中count缓存 |
| countId | String | | 暂时未知 |
| maxLimit | Long | null | 单页分页条数限制 |

这里用到的就是 `current` 和 `size` 这两个参数。

由于前端还传来了一个查询字段，所以这里需要创建一个 `LambdaQueryWrapper` 对象。

`LambdaQueryWrapper` 介绍：

1. `LambdaQueryWrapper` 是一个基于 `MyBatis-Plus` 的增强查询构造器，并支持 `Lambda` 表达式，可以更加方便地进行对象的筛选和查询。`MyBatis-Plus` 是 `MyBatis` 的增强工具，可以方便快捷地进行数据读写操作。
2. `LambdaQueryWrapper` 支持链式调用，通过添加各种条件来实现动态查询。`LambdaQueryWrapper` 在底层通过 `Java` 动态代理实现。在使用时，需要将实体类作为泛型传入 `LambdaQueryWrapper` 类中，以此来进行操作。
3. `LambdaQueryWrapper` 返回的查询结果通常是由 `MyBatis-Plus` 提供的 `IBaseMapper` 中的方法返回的类型。（`IBaseMapper` 是一个基础的 `Mapper` 接口，提供了很多基础的 `CRUD` 操作，如 `insert`、`update`、`delete`、`select` 等。）查询结果的类型通常是一个 `Java Bean` 对象或一组 `Java Bean` 对象的集合，它们由

IBaseMapper 中的方法返回并由 MyBatis 进行映射。例如，当使用 selectList 方法查询时，返回的是一个泛型为 Java Bean 的 List 集合；当使用 selectOne 查询时，返回一个 Java Bean 对象。

由以下代码也可知 queryWrapper 是作为一个条件，一个参数的放置在 page 方法里的，这里的 page 方法也是 mybatis-plus 中封装的方法。

page 方法介绍：

page () 方法有俩个参数，一个是 page，一个是 queryWrapper

后者可以给他设置一些条件，属于非必要参数。

page 参数是必要的，这个 page 类是 java8 之后的，是由 mybatis-plus 自带的，

它的包如下：

```
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
```

它由几个重要参数：

records 用来存放查询出来的数据

total 返回记录的总数

size 每页显示条数，默认 10

current 当前页，默认 1

orders 排序字段信息

optimizeCountSql 自动优化 COUNT SQL，默认 true

isSearchCount 是否进行 count 查询，默认 true

hitCount 是否命中 count 缓存，默认 false

```
//分页构造器
Page<Setmeal> pageInfo = new Page<>(page,pageSize);
Page<SetmealDto> dtoPage = new Page<>();

LambdaQueryWrapper<Setmeal> queryWrapper = new LambdaQueryWrapper<>();
//添加查询条件，根据name进行like查询
queryWrapper.like(name != null,Setmeal::getName,name);
//添加排序条件，根据更新时间降序排序
queryWrapper.orderByDesc(Setmeal::getUpdateTime);

setmealService.page(pageInfo,queryWrapper);
```

这里调用的 page 方法后，得出的数据会保存在 pageInfo 这个变量里，所以在传递数据给前端时，可以直接 return page。

修改套餐界面（修改与增加操作类似，不同点在于数据回显与路径）

* 套餐名称: * 套餐分类:


* 套餐价格:

* 套餐菜品:

+ 添加菜品

| 名称 | 原价 | 份数 | 操作 |
|-------|----|--|----|
| 霸王鸡盒 | 29 | <input type="text" value="-"/> <input type="text" value="1"/> <input type="text" value="+"/> | 删除 |
| 可乐(中) | 8 | <input type="text" value="-"/> <input type="text" value="1"/> <input type="text" value="+"/> | 删除 |

* 套餐图片:



这里先获得该套餐的 id，然后根据 id，进行数据回显

```
/**
 * 根据id查询套餐信息和对应的分类信息
 * @param id
 * @return
 */
@GetMapping("/{id}")
public R<SetmealDto> get(@PathVariable Long id){
    SetmealDto setmealDto = setmealService.getByDish(id);
    return R.success(setmealDto);
}
```

由于采用的是 MybatisPlus 技术，且其进行的操作有限，以及修改界面需要添加套餐类别，从而创建了一个 SetmealDto 的中间实体负责将数据更完整的呈现的同时也减少了表的创建，从而需要在 SetmealService 中新增函数，代码如下：

```
/**
 * 根据id查询菜品信息及口味
 * @param id
 * @return
 */
@Override
public SetmealDto getByDish(Long id) {
    //查询套餐基本信息，从setmeal表查询
    Setmeal setmeal = this.getById(id);

    SetmealDto setmealDto = new SetmealDto();
    BeanUtils.copyProperties(setmeal, setmealDto);

    //查询当前套餐对应的菜品信息，从setmeal_dish表查询
    LambdaQueryWrapper<SetmealDish> queryWrapper = new LambdaQueryWrapper<>();
    queryWrapper.eq(SetmealDish::getSetmealId, setmeal.getId());
    List<SetmealDish> dishes = setmealDishService.list(queryWrapper);
    setmealDto.setSetmealDishes(dishes);

    return setmealDto;
}
```

同理，update 以及 delete 也需要额外的创建函数，而非使用 MybatisPlus 中现成的函数，这里就不一一展示相关代码了。

添加套餐菜品界面

添加菜品

汉堡

小食

饮品

甜品

☐ 奇奇黑脆鸡堡

在售

21

☐ 麦辣鸡腿堡

在售

16

☐ 3层芝士牛堡

在售

25

☐ 大嘴安格斯

在售

28

☐ 皇堡

在售

19

☐ 果木风味鸡腿堡

在售

19

☐ 狼霸王牛堡

在售

23

☐ 炫辣鸡腿堡

在售

16

☐ 双层脆鸡堡

在售

22

已选菜品(2)

¥ 29

×

¥ 8

×

取消

确定

这里涉及到从数据库查找套餐菜品关系表，具体代码如下：

```
/**
 * 移动端点击套餐图片查看套餐具体内容
 * 这里返回的是dto 对象，因为前端需要copies这个属性
 * 前端主要要展示的信息是:套餐中菜品的基本信息，图片，菜品描述，以及菜品的份数
 * @param SetmealId
 * @return
 */
@GetMapping("/dish/{id}")
public R<List<DishDto>> dish(@PathVariable("id") Long SetmealId){
    LambdaQueryWrapper<SetmealDish> queryWrapper = new LambdaQueryWrapper<>();
    queryWrapper.eq(SetmealDish::getSetmealId, SetmealId);
    //获取套餐里面的所有菜品 这个就是SetmealDish表里面的数据
    List<SetmealDish> list = setmealDishService.list(queryWrapper);

    List<DishDto> dishDtos = list.stream().map((setmealDish) -> {
        DishDto dishDto = new DishDto();
        //其实这个BeanUtils的拷贝是浅拷贝，这里要注意一下
        BeanUtils.copyProperties(setmealDish, dishDto);
        //这里是为了把套餐中的菜品的基本信息填充到dto中，比如菜品描述，菜品图片等菜品的基本信息
        Long dishId = setmealDish.getDishId();
        Dish dish = dishService.getById(dishId);
        BeanUtils.copyProperties(dish, dishDto);

        return dishDto;
    }).collect(Collectors.toList());

    return R.success(dishDtos);
}
```

注意：移动端与后台管理端可共用同一代码，减少代码的冗余

批量删除或修改套餐状态操作
这里添加了，防止误删操作，删除的前提是菜品或是套餐停售

套餐正在售卖中，删除失败

请输入套餐名称

批量删除

批量启售

批量停售

+ 新建套餐

| <input checked="" type="checkbox"/> | 套餐名称 | 图片 | 套餐分类 | 套餐价 | 售卖状态 | 最后操作时间 | 操作 |
|-------------------------------------|-------------|---|------|------|------|---------------------|--|
| <input checked="" type="checkbox"/> | 【双人餐】双人经济餐B |  | 双人餐 | ¥ 47 | 启售 | 2023-05-29 11:37:56 | 修改 停售 删除 |
| <input checked="" type="checkbox"/> | 【三人餐】三人爆款餐 |  | 多人餐 | ¥ 54 | 启售 | 2023-05-28 13:57:33 | 修改 停售 删除 |

当将套餐进行停售处理，界面为，操作字样也相应的更改为需要启售的状态

套餐状态已经更改成功!

请输入套餐名称

批量删除

批量启售


批量停售

+ 新建套餐

| <input type="checkbox"/> | 套餐名称 | 图片 | 套餐分类 | 套餐价 | 售卖状态 | 最后操作时间 | 操作 |
|--------------------------|-------------|---|------|------|------|---------------------|--|
| <input type="checkbox"/> | 【双人餐】双人经济餐B |  | 双人餐 | ¥ 47 | 停售 | 2023-06-01 19:25:33 | 修改 启售 删除 |
| <input type="checkbox"/> | 【三人餐】三人爆款餐 |  | 多人餐 | ¥ 54 | 停售 | 2023-06-01 19:25:33 | 修改 启售 删除 |
| <input type="checkbox"/> | 【双人餐】双人经济餐A |  | 双人餐 | ¥ 42 | 启售 | 2023-05-28 13:51:34 | 修改 停售 删除 |

同时菜品的停售，则会导致所有含有此菜品的套餐都处于停售状态，属于一个连锁反应

④订单管理界面（由于操作类似，这里仅展示页面效果）



员工管理

分类管理

菜品管理

套餐管理

订单明细

订单明细

cqq

请输入订单号

开始日期 至 结束日期

查询

| 订单号 | 订单状态 | 用户 | 手机号 | 地址 | 下单时间 | 实收金额 | 操作 |
|---------------------|------|--------|-------------|----------|---------------------|------|---|
| 1662705256322056194 | 已完成 | wangwu | 13297325886 | hnust | 2023-05-28 14:19:59 | ¥ 39 | 查看 |
| 1662706246026805250 | 已派送 | cqq | 13297325886 | 湖南科技大学八区 | 2023-05-28 14:23:55 | ¥ 99 | 查看 完成 |
| 1664062885086699522 | 正在派送 | cqq | 13297325886 | 湖南科技大学八区 | 2023-06-01 08:14:43 | ¥ 84 | 查看 派送 |

共 3 条 10条/页 < 1 > 前往 1 页

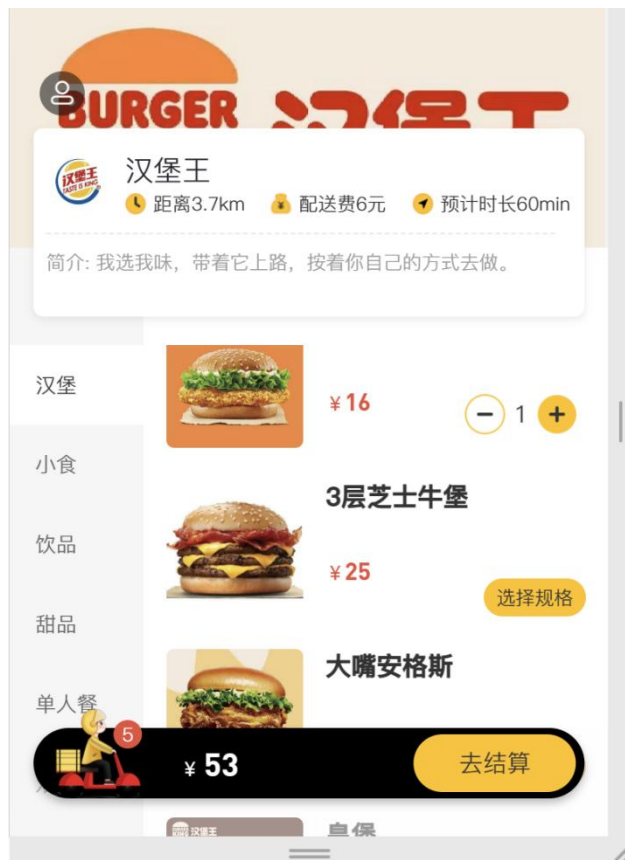
(2) 客户端（模仿移动端的界面）

①登录界面（模拟手机获取验证码）

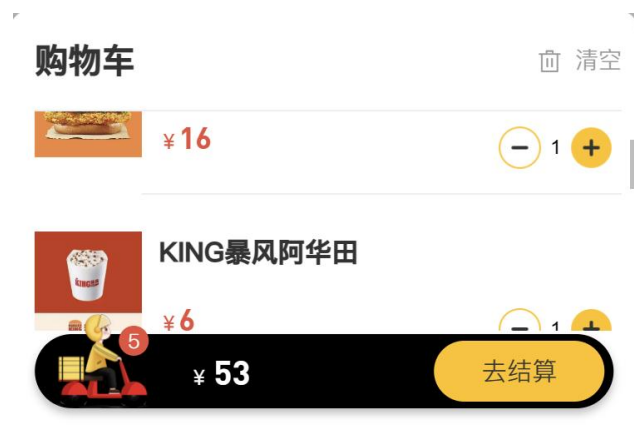


A mobile app login screen for a burger restaurant. At the top, a green banner shows a success message: '手机验证码为: 5593'. Below this is a white input area containing a phone number '13297325886' and a yellow button labeled '获取验证码'. Underneath is a text input field with the placeholder '请输入验证码'. At the bottom is a large yellow button labeled '登录'.

②主界面展示:



购物车界面会从数据库获取用户购物车的商品数据，代码与上述相似，不展示了



点击选择规格，可以展示相应的口味选项，这里应用的代码与管理端相同



③用户的个人界面（包含订单信息）



④历史订单界面：



⑤地址管理界面



上述界面的增删改操作均与后台管理端的操作类似，不一一展示

(3) 统一管理（这里就说明主要部分，其余技术不一一讲述）

全局异常处理，用于处理加入重复菜品或套餐亦或是其他异常，对于手机号码，信息输入等信息的校验均在前端做了处理，全局异常处理代码如下：

```
@Slf4j
@ResponseBody
@ControllerAdvice(annotations = {RestController.class, Controller.class})
public class GlobalExceptionHandler {

    /**
     * 异常处理方法
     * @param ex
     * @return
     */
    @ExceptionHandler(SQLIntegrityConstraintViolationException.class)
    public R<String> exceptionHandler(SQLIntegrityConstraintViolationException ex){
        log.error(ex.getMessage());
        if(ex.getMessage().contains("Duplicate entry")){
            String[] split = ex.getMessage().split(" ");
            String msg = split[2] + "已存在";
            return R.error(msg);
        }
        return R.error("未知错误");
    }

    @ExceptionHandler(CustomException.class)
    public R<String> exceptionHandler(CustomException ex){
        return R.error(ex.getMessage());
    }
}
```

通过 ThreadLocal 封装工具类来获取或保存用户或员工 id

```
/**
 * 基于ThreadLocal封装工具类，用户保存和获取当前登录用户id
 */
public class BaseContext {

    private static ThreadLocal<Long> threadLocal = new ThreadLocal<>();

    /**
     * 设置值
     * @param id
     */
    public static void setCurrentId(Long id){
        threadLocal.set(id);
    }

    /**
     * 获取值
     * @return
     */
    public static Long getCurrentId(){
        return threadLocal.get();
    }
}
```


公共字段填充部分，由于无论是管理员还是员工对菜品还是套餐进行操作时，需要记录下其修改时间，亦或是创建根据修改时间进行菜品或者套餐的排序，都需要对时间进行修改，而频繁手动操作很繁琐且使得代码很冗余，所以采用公共字段填充。

```
/**
 * 插入操作自动填充
 * @param metaObject
 */
@Override
public void insertFill(MetaObject metaObject) {
    log.info("公共字段自动填充[insert]...");
    log.info(metaObject.toString());
    metaObject.setValue("createTime", LocalDateTime.now());
    metaObject.setValue("updateTime", LocalDateTime.now());
    metaObject.setValue("createUser", BaseContext.getCurrentId());
    metaObject.setValue("updateUser", BaseContext.getCurrentId());
}

/**
 * 修改操作自动填充
 * @param metaObject
 */
@Override
public void updateFill(MetaObject metaObject) {
    log.info("公共字段自动填充[update]...");
    log.info(metaObject.toString());
    metaObject.setValue("updateTime", LocalDateTime.now());
    metaObject.setValue("updateUser", BaseContext.getCurrentId());
}
```

为了使前后端数据格式(JSON)的统一，采用R实体来装载数据

```
public static <T> R<T> success(T object) {
    R<T> r = new R<T>();
    r.data = object;
    r.code = 1;
    return r;
}

public static <T> R<T> error(String msg) {
    R r = new R();
    r.msg = msg;
    r.code = 0;
    return r;
}
```



五. 实验结果于分析

注意：由于详细设计部分已给出项目大部分界面图片，在此就展示上述未展示的界面图片。

1. 其余界面展示

(1) 分类管理界面

①分类列表



1 员工管理

2 分类管理

3 菜品管理

4 套餐管理

5 订单明细

分类管理

+ 新增菜品分类

+ 新增套餐分类

| 分类名称 | 分类类型 | 操作时间 | 排序 | 操作 |
|------|------|---------------------|----|---------------------------------------|
| 汉堡 | 菜品分类 | 2023-05-28 10:56:58 | 1 | 修改 删除 |
| 小食 | 菜品分类 | 2023-05-28 10:57:16 | 2 | 修改 删除 |
| 饮品 | 菜品分类 | 2023-05-28 10:57:23 | 3 | 修改 删除 |
| 甜品 | 菜品分类 | 2023-05-28 11:13:56 | 4 | 修改 删除 |
| 单人餐 | 套餐分类 | 2023-05-28 10:57:45 | 11 | 修改 删除 |
| 双人餐 | 套餐分类 | 2023-05-28 13:47:40 | 12 | 修改 删除 |
| 多人餐 | 套餐分类 | 2023-05-28 13:47:51 | 13 | 修改 删除 |

共 7 条

10条/页

< 1 >

前往

1 页

②新增分类页面

新增套餐分类

×

分类名称:

请输入分类名称

排序:

请输入排序

取消

确定

保存并继续添加


③删除操作：如果该分类下还有菜品，则删除失败，仅在分类下无菜品的情况下才能删除成功

×

当前分类下关联了菜品，删除失败

(2) 菜品管理界面

①菜品列表



员工管理

分类管理

菜品管理

套餐管理

订单明细

菜品管理







ccq

批量删除

批量启售

批量停售

+ 新建菜品

| <input type="checkbox"/> | 菜品名称 | 图片 | 菜品分类 | 售价 | 售卖状态 | 最后操作时间 | 操作 |
|--------------------------|---------|---|------|------|------|---------------------|--|
| <input type="checkbox"/> | 奇奇黑脆鸡堡 |  | 汉堡 | ¥ 21 | 启售 | 2023-05-28 14:01:40 | 修改 停售 删除 |
| <input type="checkbox"/> | 带劲香辣片片鸡 |  | 小食 | ¥ 9 | 启售 | 2023-05-28 14:00:15 | 修改 停售 删除 |
| <input type="checkbox"/> | 麦辣鸡腿堡 |  | 汉堡 | ¥ 16 | 启售 | 2023-05-28 13:47:12 | 修改 停售 删除 |
| <input type="checkbox"/> | 3层芝士牛堡 |  | 汉堡 | ¥ 25 | 启售 | 2023-05-28 13:29:51 | 修改 停售 删除 |
| <input type="checkbox"/> | 霸王鸡盒 |  | 小食 | ¥ 29 | 启售 | 2023-05-28 13:28:52 | 修改 停售 删除 |
| <input type="checkbox"/> | 霸王鸡条 |  | 小食 | ¥ 9 | 启售 | 2023-05-28 13:28:31 | 修改 停售 删除 |

里面如果图片未查询到或损坏，有默认图片进行替换，如图所示

②查询功能

批量删除

批量启售

批量停售

+ 新建菜品

| <input type="checkbox"/> | 菜品名称 | 图片 | 菜品分类 | 售价 | 售卖状态 | 最后操作时间 | 操作 |
|--------------------------|-------|---|------|------|------|---------------------|--|
| <input type="checkbox"/> | 可乐(小) |  | 饮品 | ¥ 6 | 启售 | 2023-05-28 12:00:51 | 修改 停售 删除 |
| <input type="checkbox"/> | 可乐(中) |  | 饮品 | ¥ 8 | 启售 | 2023-05-28 12:00:22 | 修改 停售 删除 |
| <input type="checkbox"/> | 可乐(大) |  | 饮品 | ¥ 10 | 启售 | 2023-05-28 12:00:09 | 修改 停售 删除 |

共 3 条

10条/页

< 1 >

前往 1 页

③修改菜单界面

* 菜品名称:

* 菜品分类:

汉堡

* 菜品价格:

口味做法配置:

口味名 (3个字内) 口味标签 (输入标签回车添加)

忌口

不要生菜 X

不要沙拉酱 X

删除

辣度

微辣 X


中辣 X

重辣 X

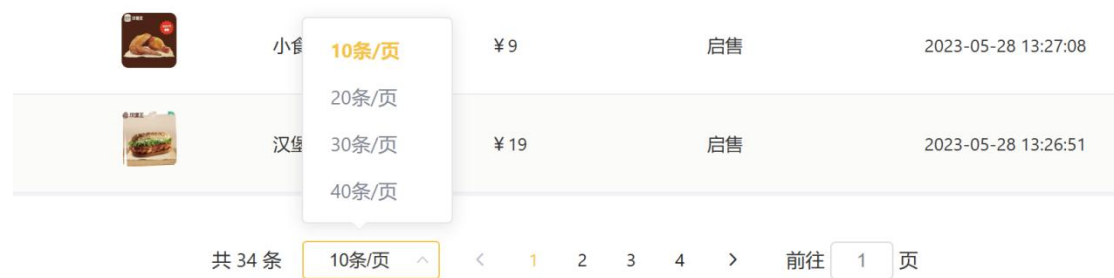
删除

添加口味

* 菜品图片:



④分页细节展示



⑤支付成功界面



下单成功

预计21:38到达

后厨正在加紧制作中，请耐心等待~

查看订单

2. 实验分析

该实验在基于课程设计给定的要求上，增添了许多额外的功能，如多组分页，员工管理，订单状态，菜品口味，套餐分类，地址管理等。

六. 小结与心得体会

小结：项目虽然实现了课设给定的功能以及拓展了额外的功能，但仍然有许多需要改进的地方以及增添的功能，如后续可以将该项目发展成微信小程序，也可以添加用户个人信息模块(修改个人信息，上传照片等)，可以增加月销展示功能，支付宝、微信支付，派送时间预估，查看外卖员的派送路径，客服交流，菜品评论等功能。

心得体会：通过对整个项目的开发后，进一步掌握了 SSM 框架的知识，以及进一步理解了 SSM 框架的相关内容及使用，为后续学习相关开发知识提供了良好的基础。接下来说说整个做项目的过程，在做本次项目前，不知如何下手，但是在一步步接触后，有了清晰的认知，同时可以根据自己的想法去添加一些额外的功能，总的来说，很享受，可以说是乐在其中，但是对于自己来说，这些还算是开发中最基础的部分，由于我们用的都是封装后的代码，相对于源码来说简单了许多，打个比方来说，也不过是在提供材料的前提下，搭建积木罢了，而且目前学的不过是冰山一角。当然有了这次经验后，激励我去追求更多的知识，去开发更多有用亦或是有难度的项目吧，总的来说是，非常棒！