哈爾濱工業大學（威海）
Harbin Institute of Technology, Weihai

软件综合课程设计报告

任务题目：基于 Springboot 和微信小程序的校园外卖平台

学　　号：2201110424

姓　　名：李彦廷

组　　号：2

任课教师：胡鑫

开课学期：2023 年春季学期

哈尔滨工业大学（威海）计算机科学与技术学院

前 言

《软件综合课程设计》是基于项目学习，锻炼学生动手实践能力的一个重要环节，是本科毕业设计的预演。需要综合应用《高级语言程序设计 I|、II》、《数据库系统原理》、《软件工程》等课程中讲授的知识与方法，培养学生构建系统与数据建模的能力。

具体运用《软件工程》中面向对象建模及结构化编程的思想，针对实际应用系统进行可行性及需求分析，将用户需求转换为软件需求，采用概要设计、详细设计、编码实现测试和运行维护等来实现工程化的软件生产过程，追求软件产品的高可靠与易维护，提升软件的全生命周期管理效率。

充分利用《数据库系统原理》中数据建模、关系分析、参照约束、并发管理、安全控制、范式理论等方面的知识与技术，深刻理解数据库应用的核心问题驱动与分阶段设计思想，能够熟练利用统一建模语言 UML、数据建模工具 E-R 图、结构化查询语言 SQL、数据完整性约束及安全性控制技术等来完成实际软件系统的数据库设计实现和应用业务支撑等。

本课程要求学生分组进行，通过一定的调研来结合实际应用问题自行选题。要求所设计开发的软件具有较高的实用性和较好的完整性，系统 UI 交互便捷，所设计实现的软件功能要符合用户实际需求。本课程主要教学环节包括：学生选题、开题指导、中期检查、软件验收、课程报告撰写和资料整理归集等。

教师评语：   报告成绩：

1. 选题背景与意义

（描述选题的背景、所针对的具体实际问题及任务所体现的实用性价值等）

选题背景：

数据显示，截至 2022 年，中国在线外卖用户规模达到 5.21 亿人，用户规模庞大，增长迅猛。外卖行业蓬勃发展，成为许多人在家中用餐的首选，大学生也不例外。在我们学校，学生用餐的方式主要有去食堂、去大学生服务中心或者点外卖。然而，去食堂和去大学生服务中心都需要步行一段距离。观察发现，在用餐高峰期，特别是下课时间，食堂和大学生服务中心人流量非常大，导致秩序混乱，排队时间过长。另外，出于食品安全考虑，包括我们学校在内的许多学校不允许校外外卖配送到校园内，学生通常需要去校园外围取外卖。这样一来，点外卖并没有体现出便捷性，而且当许多外卖聚集在一起时，找到自己的外卖变得困难，还增加了外卖被偷或者取错的风险。

具体实际问题：

①竞争对手和市场饱和：外卖市场竞争激烈，存在许多已经建立起来的外卖平台和小程序。在这个竞争激烈的环境下，如何在大学中因地制宜，联合学校与食堂，与竞争对手区分开来，吸引用户并建立自己的用户基础是一个主要问题。

②校园内外卖配送限制：学校存在校内外卖配送限制，可能需要与学校方面协商和解决相关问题，确保外卖能够顺利配送到宿舍楼。

③用户体验和界面设计：微信小程序的成功与否与用户体验和界面设计密切相关。提供简洁、直观且易于导航的用户界面，确保良好的用户体验和操作流程是关键问题。

④用户信任和数据安全：作为一项涉及用户个人信息和支付信息的服务，信任和数据安全对对于大学生来说是至关重要的。确保用户数据的安全性、隐私保护和支付安全，以及建立良好的用户评价和反馈机制是必要的。

实用性价值：

①方便快捷的用户体验：通过微信小程序的特性，用户无需额外下载应用程序，可以直接在微信中访问和使用外卖服务，节省了安装和更新应用的时间和流量消耗，提供了更加便捷的用户体验。

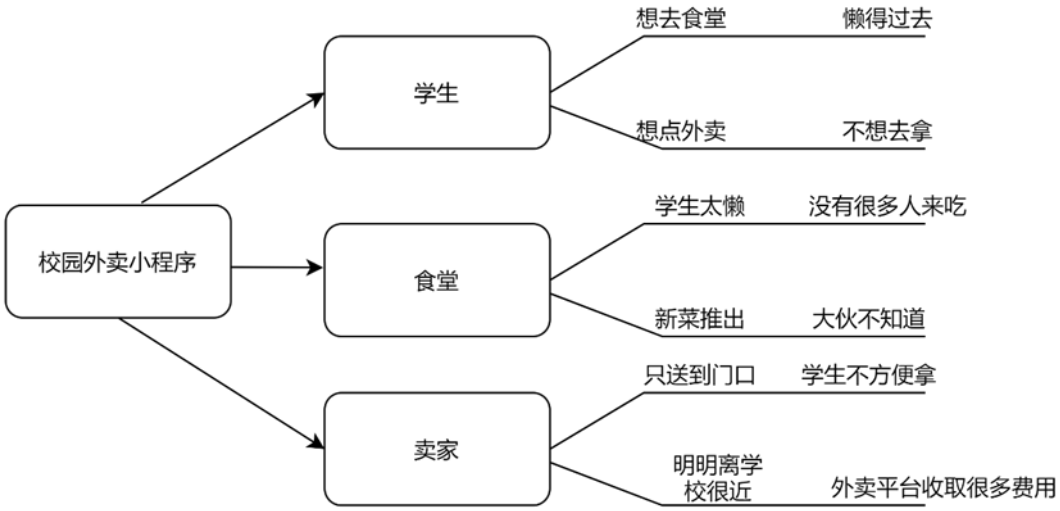
②满足即时需求：外卖系统基于对大学生的即时洞察，能够及时满足用户的食物需求。用户可以随时通过微信小程序浏览店铺的商品内容，并快速完成购买，实现了快速的点餐和配送至宿舍的服务。

③促进本地生活服务经济发展：外卖系统作为本地生活服务平台的一部分，通过线上和线下消费场景的打通，可以帮助本地餐饮行业吸引更多用户和订单，促进经济发展。同时，外卖系统也为学校食堂提供了新的营销渠道，增加食堂收入。

④小程序生态的推动：通过开发基于微信小程序的外卖系统，进一步推动微信小程序的发展和普及。微信小程序已经构建了庞大的开发者生态，为程序员提供了新的就业和创业机会，通过加入微信小程序开发，可以扩展技能和职业发展的可能性。

2. 需求分析

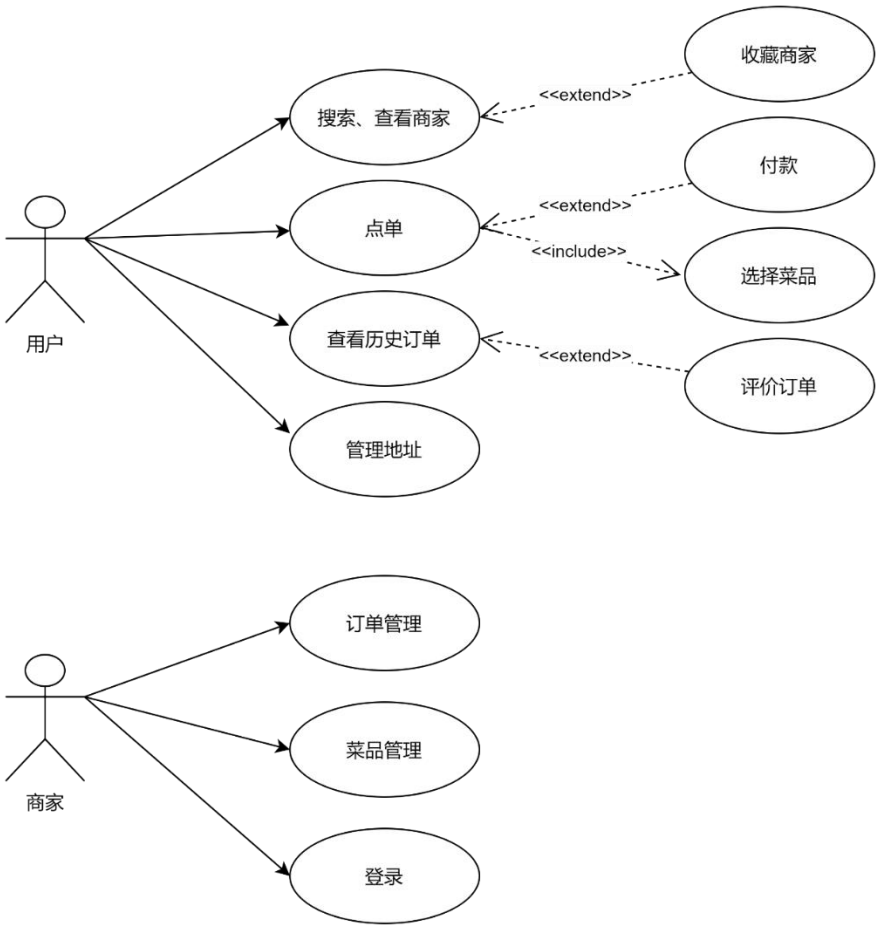
（根据任务选题的要求，充分地分析和理解问题，明确用户要求做什么？完整性约束条件是什么？运行环境要求、图形操作界面要求,画出用户用例/泳道图等）



分类	主要功能	特性描述
用户端	点单	用户可以在浏览商家时选择菜品，完成点单
	历史订单	用户可以查看历史订单
	浏览商家	用户可以从首页或搜索结果进入店家详情页，浏览菜品
	地址管理	用户可以选择添加/删除地址和设置默认地址
	搜索	用户可以输入关键词搜索店铺
商家端	管理订单	商家可以处理订单，决定是否接收/拒绝订单，或者取消订单
	管理菜品	商家可以增删改菜品
	登录	商家进行身份核实

运行环境要求：
用户端：微信
商家端：Web 浏览器

用例图：


收藏商家
搜索、查看商家
<<extend>>
付款
点单
<<extend>>
<<include>>
选择菜品
用户
查看历史订单
<<extend>>
评价订单
管理地址
订单管理
商家
菜品管理
登录

3. 系统概要设计

（在该部分中分析叙述清楚系统每个模块的功能要求，画出功能模块图，类图及对应文字说明，识别边界类，控制类和实体类，给出关键用例的顺序图）

（1）用户端，主要由以下三大模块组成：首页模块，展示店铺列表，用户点击进某一个店铺后，展示出此店铺的商品；订单模块，展示用户的当前订单以及历史已完成订单，展示如下单商家、订单总价、订单所包含商品、订单状态等基本信息，并支持点击订单进入订单详情页面，展示订单下单时间、订单完成时间以及本次配送地址等订单相关信息；个人中心模块，主要是对用户的相关信息进行设置，例如收货地址、联系电话、昵称等，另外还有联系客服、联系开发者等相关入口。

（2）商家端，主要由以下两大模块组成：订单管理模块，显示当前收到的订单，并显示订单状态；订单详情显示历史订单的相关信息；菜品管理模块，从菜品管理入口增加、删除或修改本店铺的菜品。

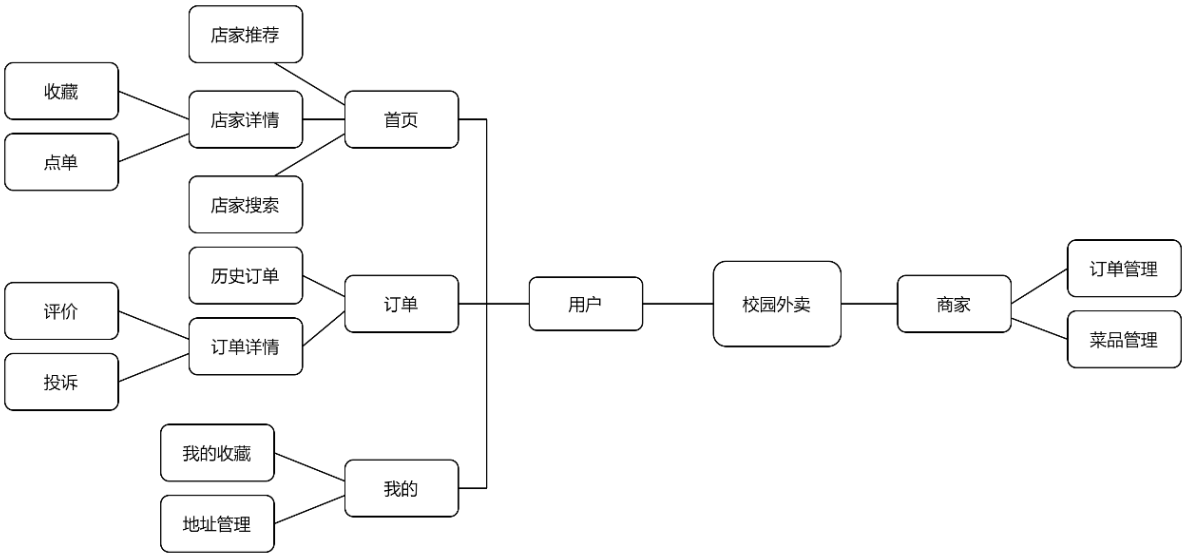


图 3-1：功能模块图

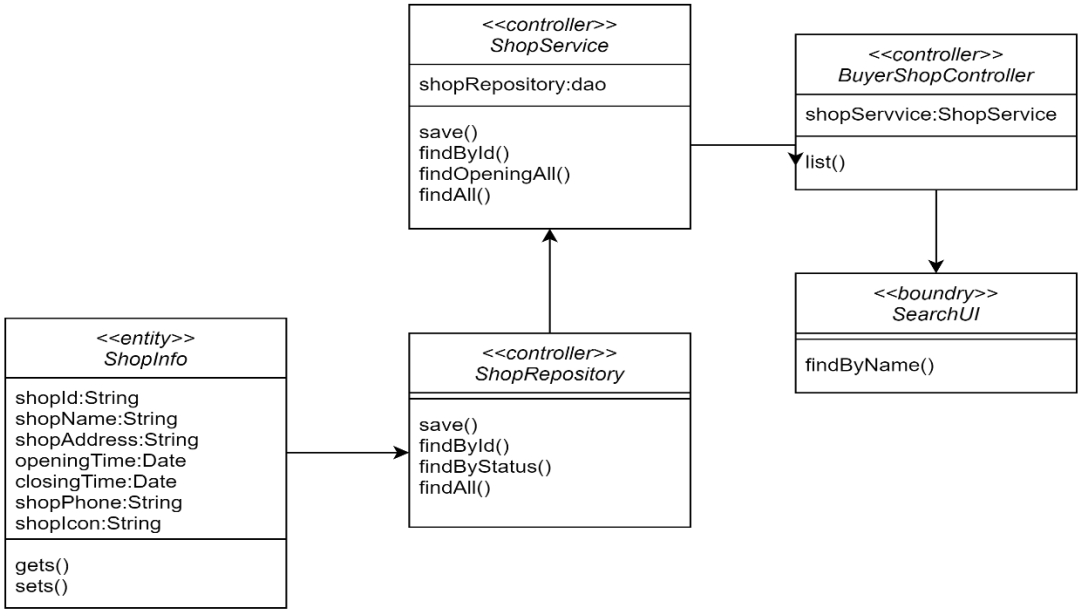


图 3-2：查询类图

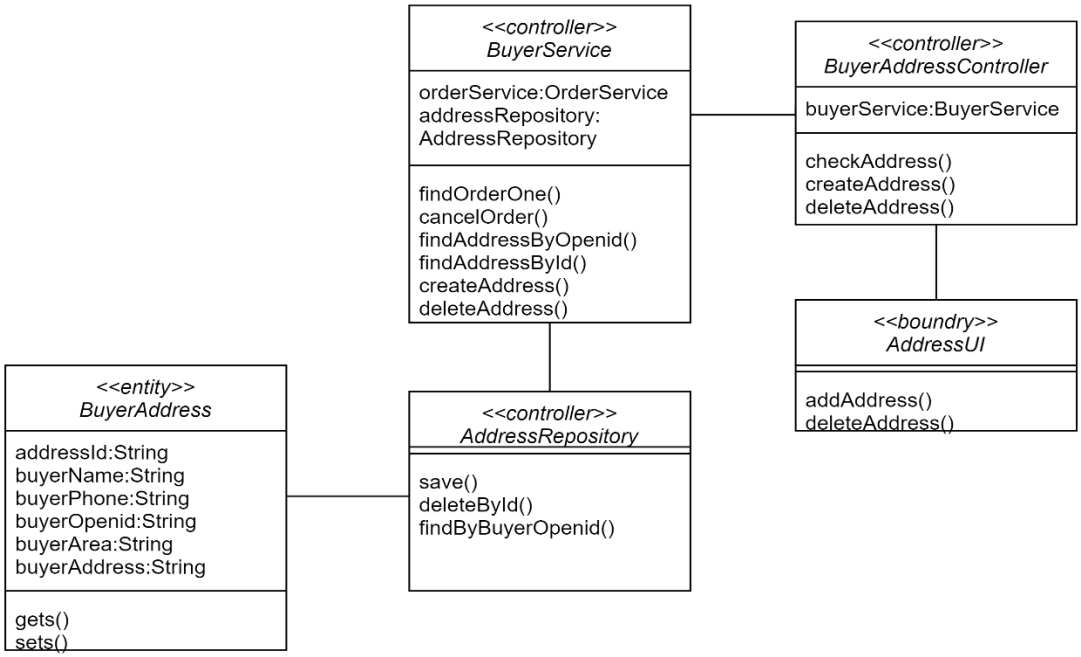


图 3-3 地址管理类图

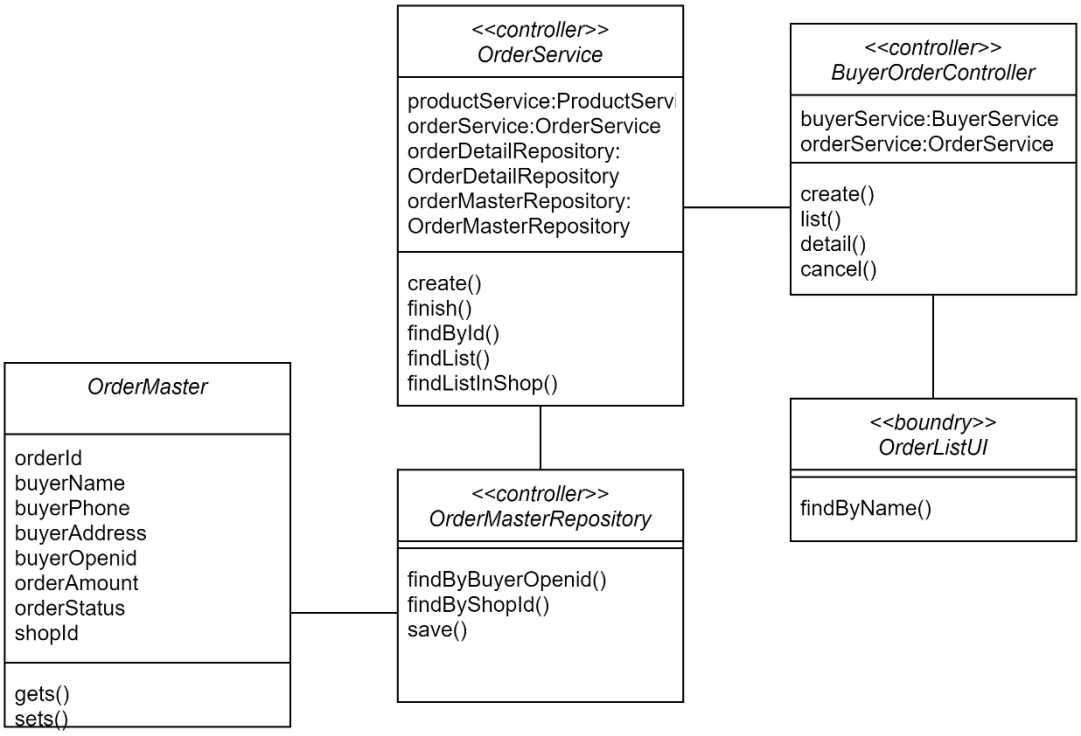


图 3-4 历史订单类图

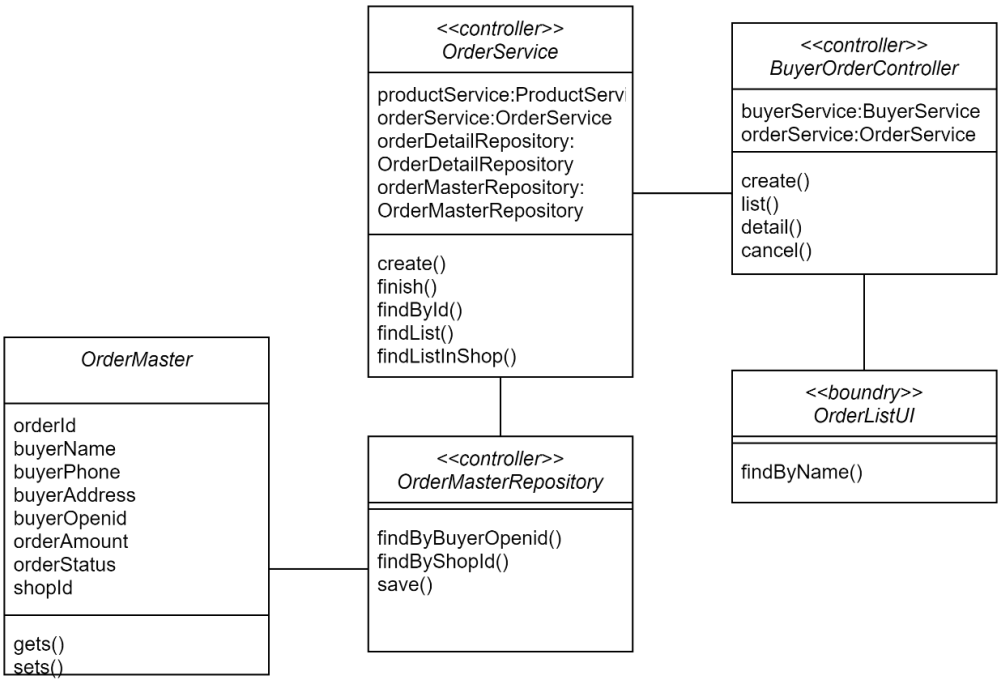


图 3-5 历史订单类图

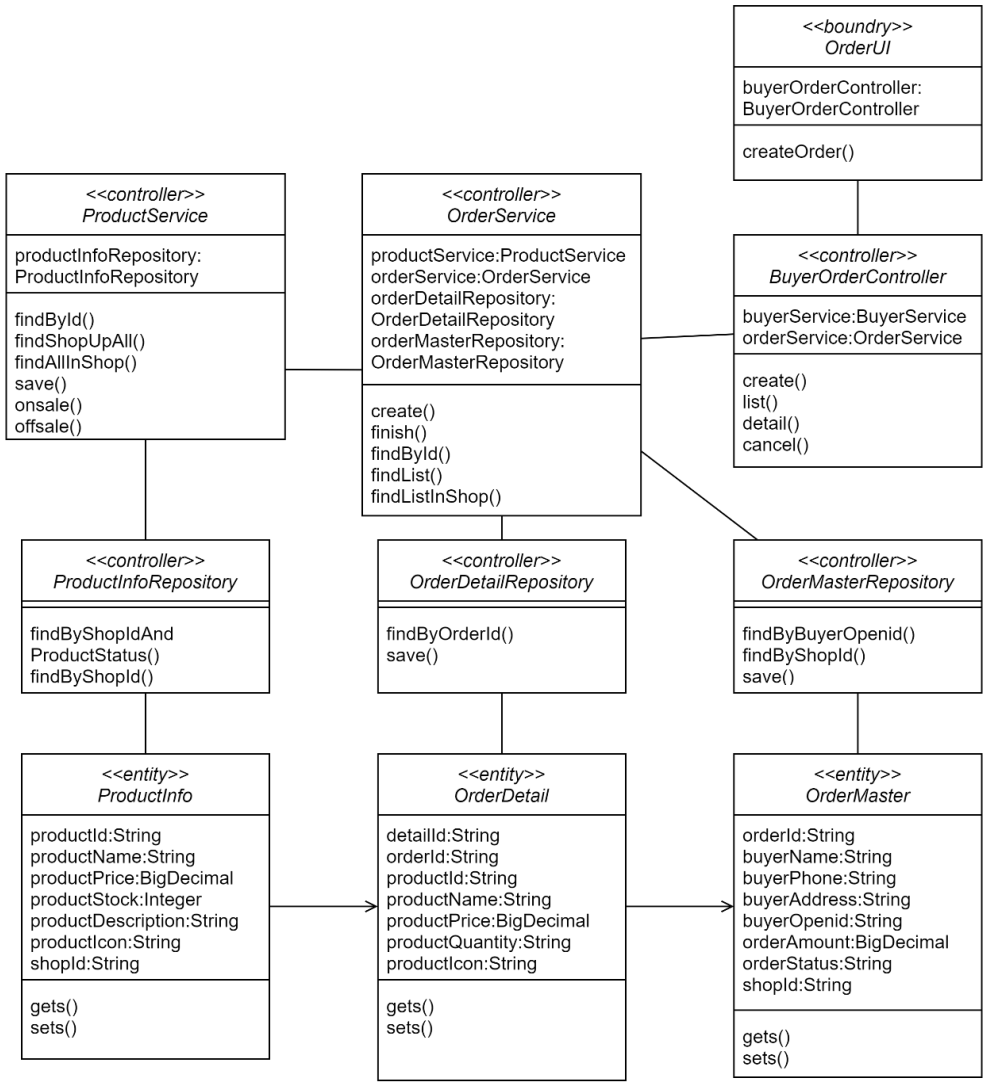


图 3-6 点单类图

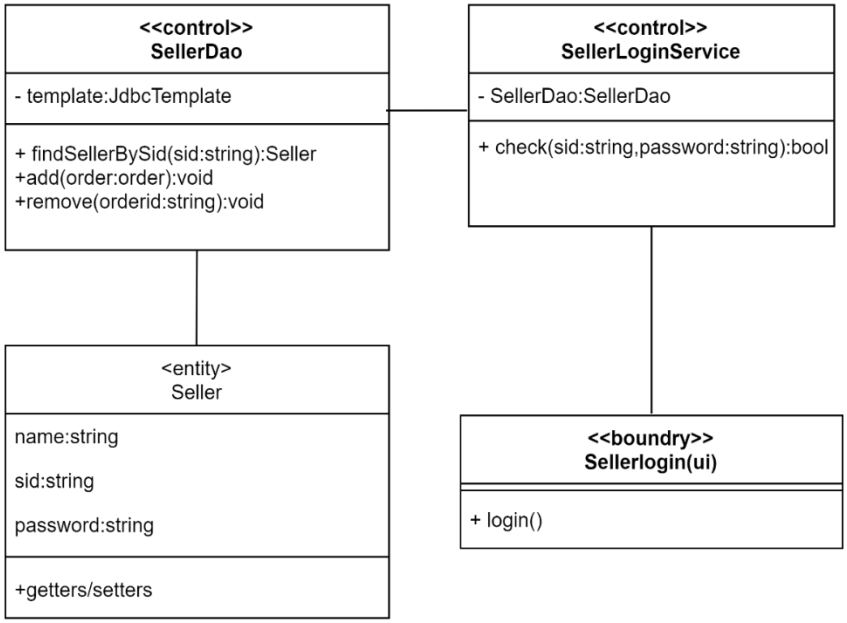


图 3-7 商家登录类图

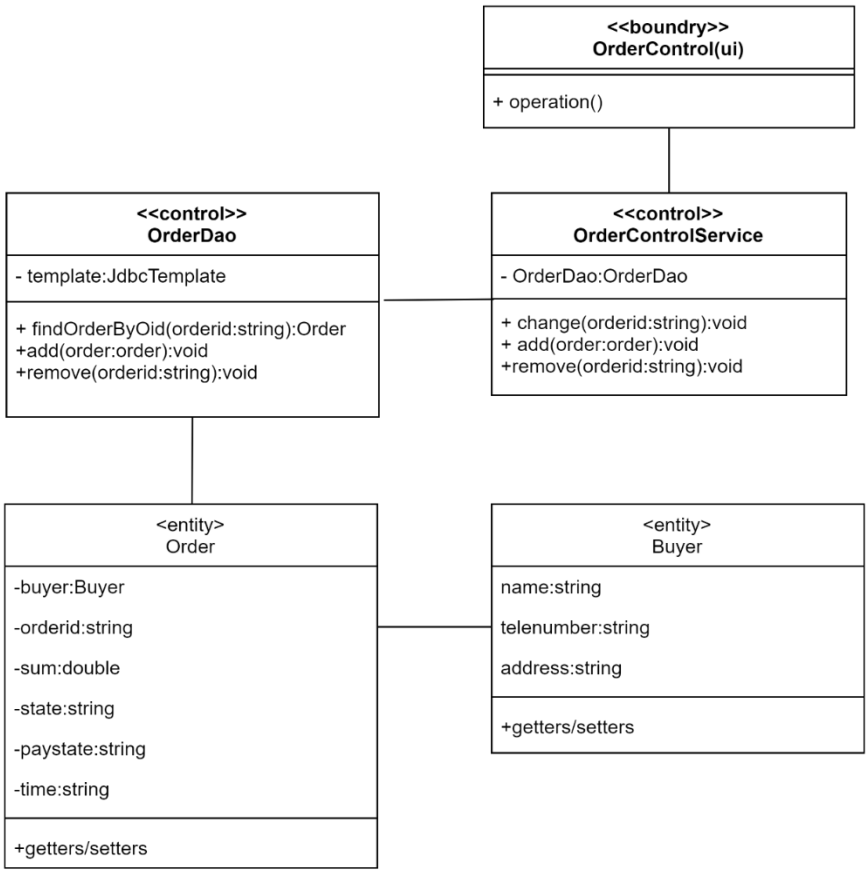


图 3-8 订单管理类图

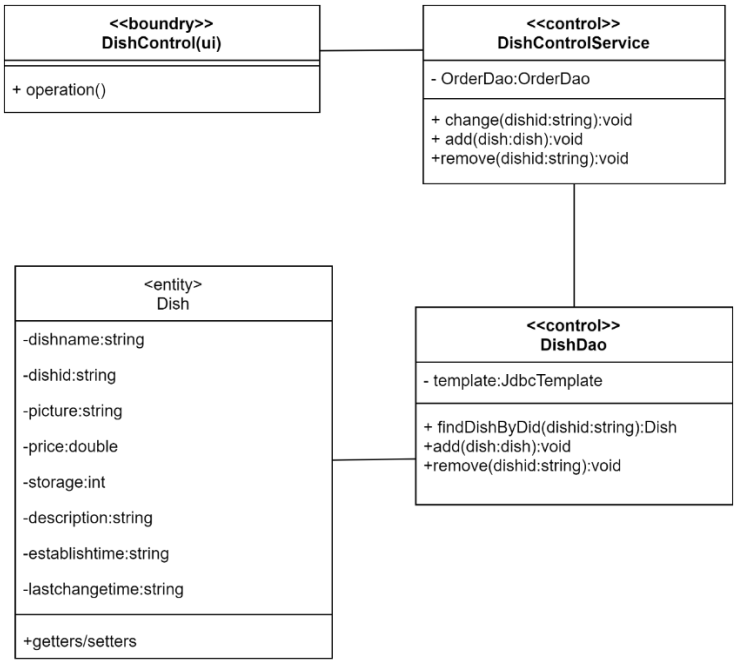


图 3-9 菜品管理类图

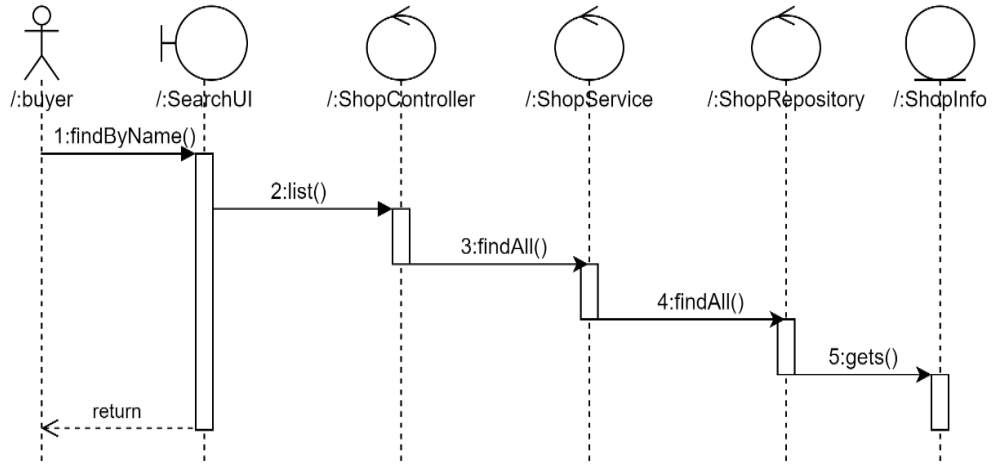


图 3-10 查询时序图

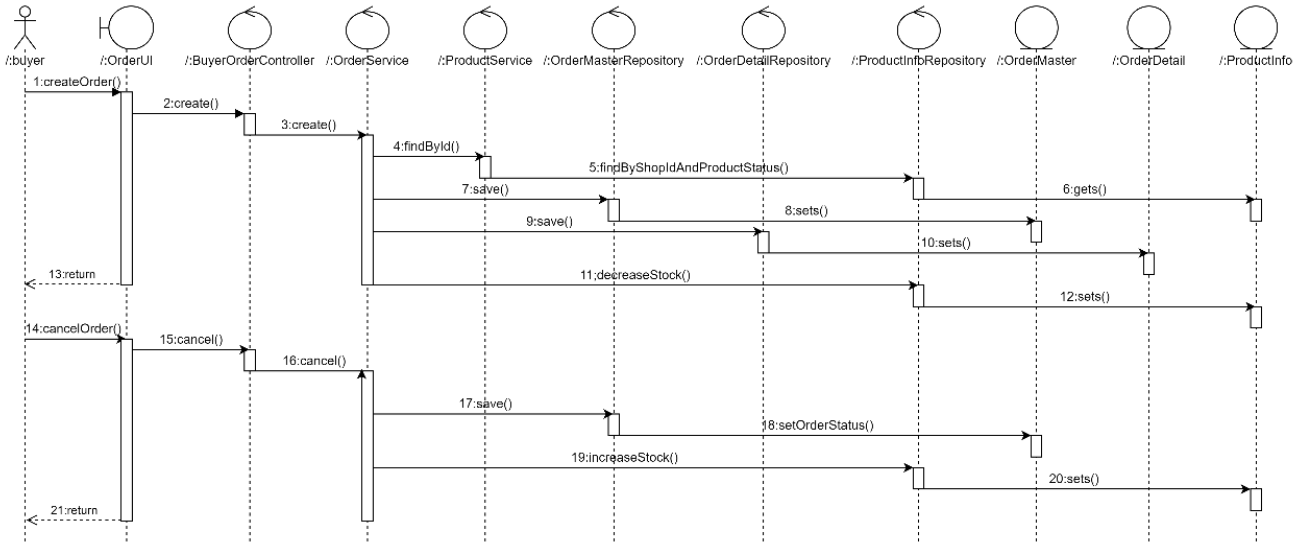


图 3-11 点单时序图

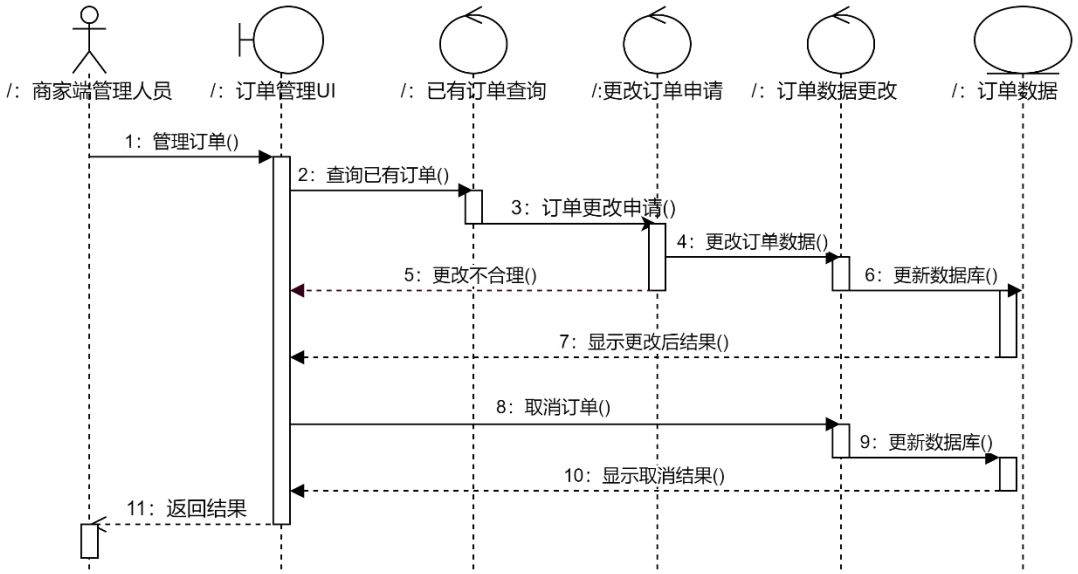
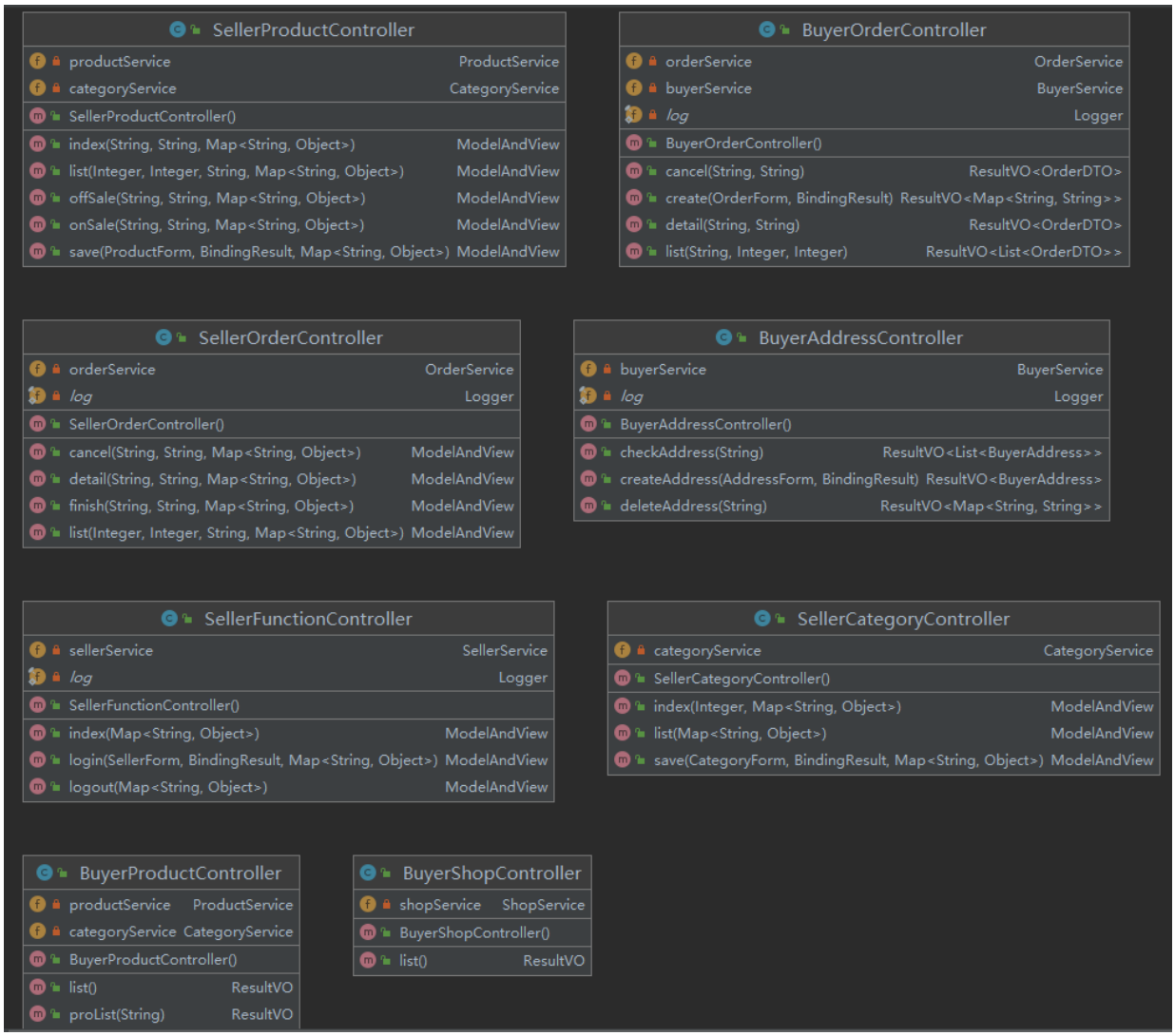
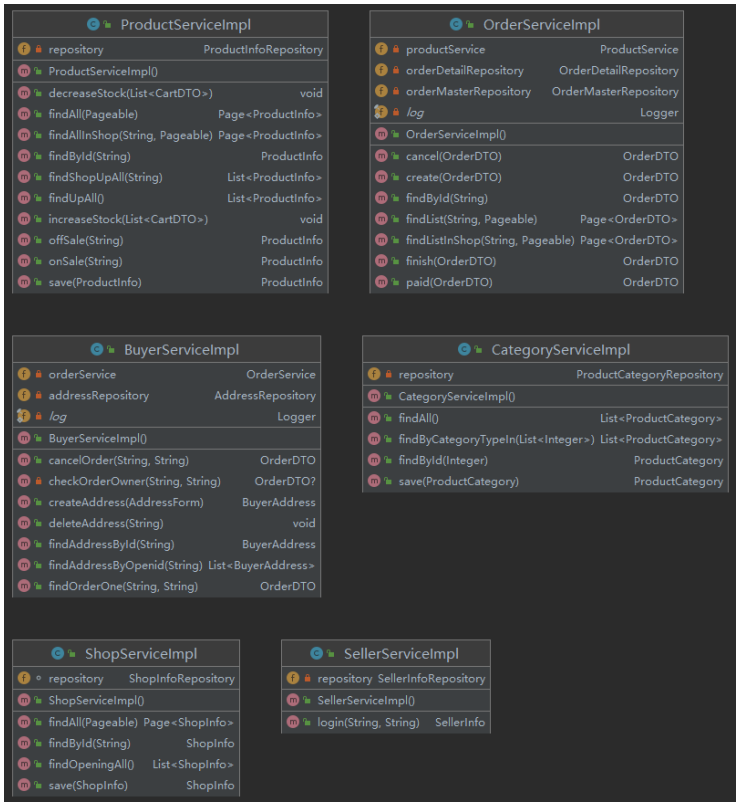


图 3-12 订单管理时序图

Controller 层：



Service 层：



Entity 类：

4. 系统详细设计

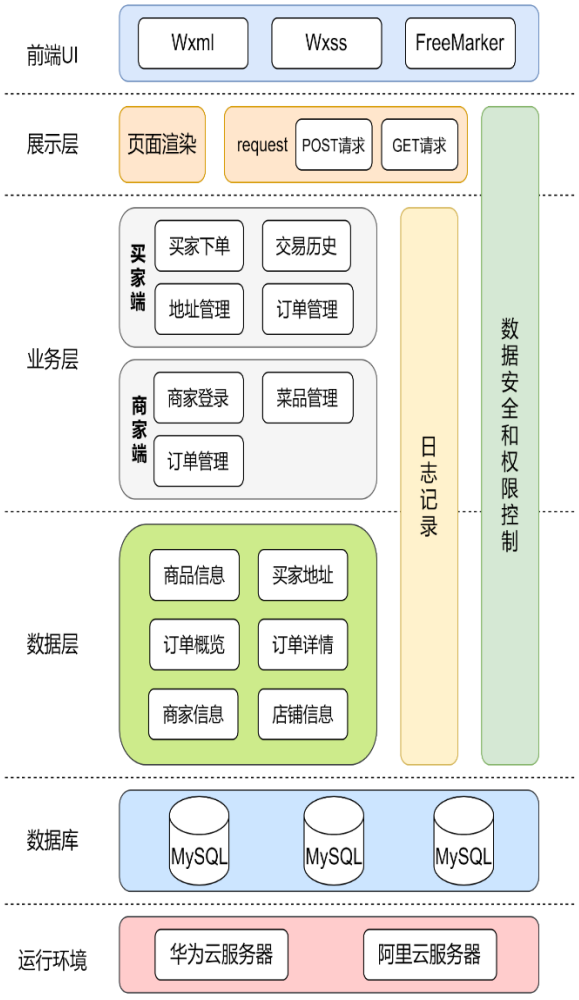


图 4-1 软件体系结构图

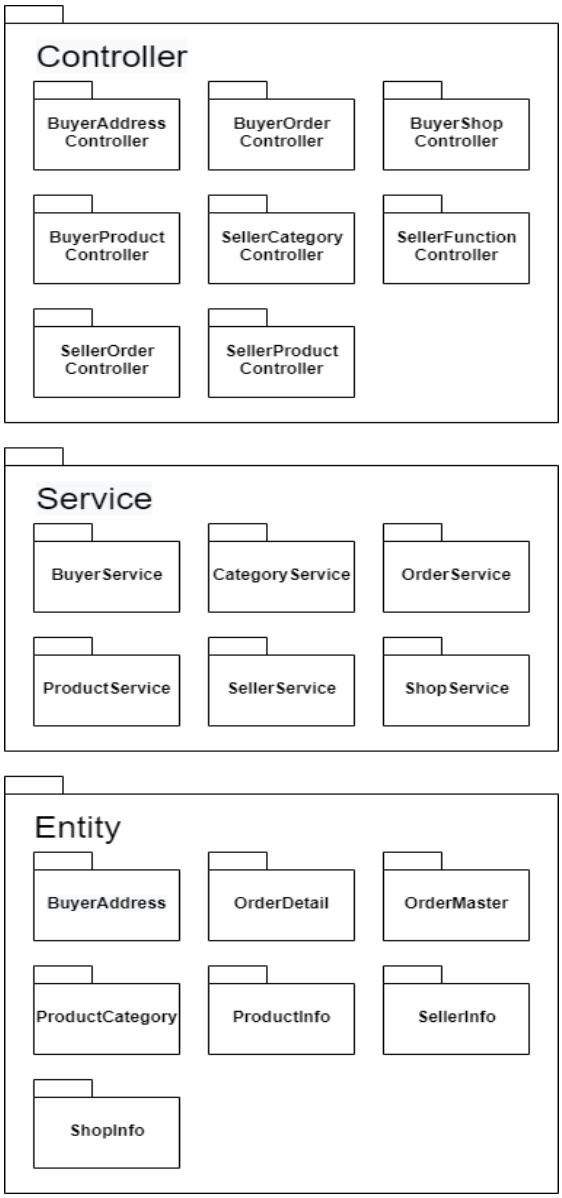


图 4-2 包图

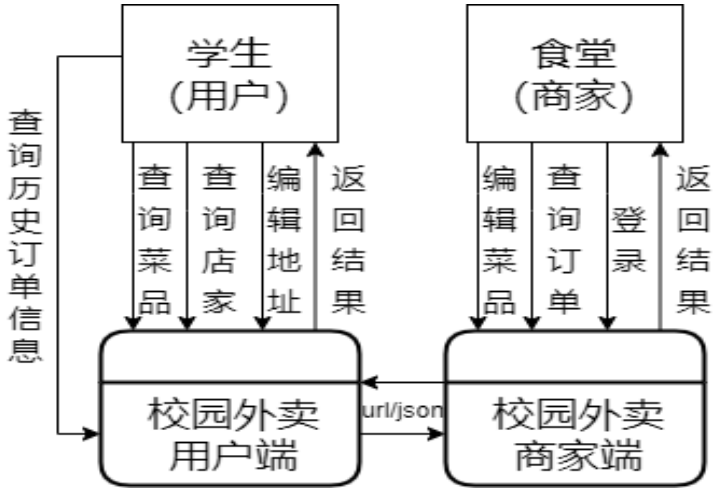


图 4-3 数据流图

数据字典（粗体为主键）：

①商品信息 product_info：

包含**商品 id**，名称，单价，库存，描述，配图，上架状态，所属店铺等

②商品类别 product_category：

包含**类别 id**，类别名，类别编号，时间

②订单概览 order_master：

包含**订单 id**，买家姓名，电话，地址，买家 id，订单状态，支付状态，时间，店铺 id

③订单条目详情 order_detail：

包含**详情 id**，所属订单 id，商品 id，名称，价格，数量，配图，时间

④店家信息：shop_info：

包含**店家 id**，名字，店铺地址，营业时间，营业状态，电话，图标

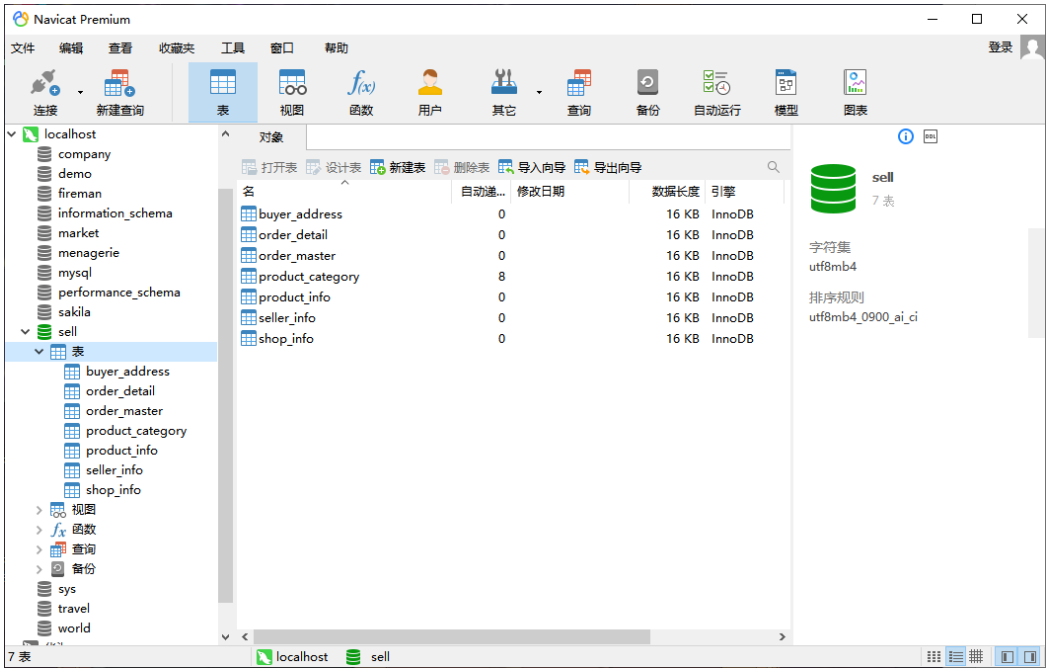
⑤买家地址：buyer_address：

包含**地址 id**，买家姓名，电话，买家 id，所在区域，详细地址

⑥卖家信息 seller_info：

包含**卖家 id**，用户名，密码，微信 id，创建时间

数据库设计：



数据库连接配置文件 appliction.yml：

```yaml
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    username: root
    password: LIyt060912
    url: jdbc:mysql://localhost:3306/sell?allowPublicKeyRetrieval=true&characterEncoding=utf-8&useSSL=false
  jpa:
    show-sql: true
  jackson:
    default-property-inclusion: non_null
server:
  servlet:
    context-path: /sell
```

5. 系统核心模块实现

（给出编程语言、开发环境及支撑软件、软件系统架构；给出（1）系统每个模块的功能描述和工作流程；（2）系统主要功能模块代码以及核心算法对应的关键函数定义代码，包括输入参数的含义、函数输出结果等；核心函数的实现代码段及注解等；主要功能界面截图及对应文字概述。）
后端：
主要语言：Java17，FreeMarker
开发环境及框架：IDEA2021.2.2，Springboot2.3

此处使用 API 表来描述功能和函数相关:
①查看商家列表：

```
请求方式：GET
请求路径：/sell/buyer/shop/list
```

请求参数

```
无
```

响应参数

参数名	参数说明	备注
shopId	商家id	
shopName	商家名称	
shopAddress	商家地址	
openingTime	开业时间	
closingTime	关门时间	
shopStatus	营业状态	
shopPhone	商家电话号码	
shopIcon	商家图标路径	

返回

```
{
    "code":0,
    "msg":"成功",
    "data":[
        {
            "shopId":"1",
            "shopName":"佛笑园",
            "shopAddress":"学苑餐厅一楼",
            "openingTime":"8:00AM",
            "closingTime":"8:00PM",
            "shopStatus":0,
```

②用户端查看商品列表：

```
请求方式：GET
请求路径：/sell/buyer/product/list
```

请求参数

```
无
```

响应参数

参数名	参数说明	备注
name	分类名称	分类根目录下name
type	分类类别	
foods	餐品列表	
id	餐品id	
name	餐品名称	foods子目录下name
price	餐品价格	
description	餐品描述	
icon	餐品图标路径	

返回

```
{
    "code": 0,
    "msg": "成功",
    "data": [
        {
            "name": "热榜",
            "type": 1,
            "foods": [
                {
                    "id": "123456",
                    "name": "皮蛋粥",
                    "price": 1.2,
                    "description": "好吃的皮蛋粥",
                    "icon": "http://xxx.com",
                }
            ]
        },
        {
            "name": "好吃的",
            "type": 2,
            "foods": [
                {
                    "id": "123457",
                    "name": "慕斯蛋糕",
                    "price": 10.9,
                    "description": "美味爽口",
                    "icon": "http://xxx.com",
                }
            ]
        }
    ]
}
```

③用户端创建订单：

##1.3创建订单

请求方式：POST
请求路径：/sell/buyer/order/create

请求参数

参数名	参数说明	备注
name	用户姓名	
phone	用户电话	
address	送餐地址	
openid	用户微信id	登录小程序自动获取
productId	餐品id	
productQuantity	对应餐品数量	

```
name: "张三"
phone: "18868822111"
address: "慕课网总部"
openid: "ew3euwhd7sjw9diwkq" //用户的微信openid
items: [{
    productId: "1423113435324",
    productQuantity: 2 //购买数量
}]
```

响应参数

参数名	参数说明	备注
orderId	订单id	

返回

```
{
  "code": 0,
  "msg": "成功",
  "data": {
      "orderId": "147283992738221"
  }
}
```

④用户端查询订单

##1.4订单列表

```
请求方式：GET
请求路径：/sell/buyer/order/list
```

请求参数

参数名	参数说明	备注
openid	用户微信id	微信自动获取
page	餐品分页	
size	页规格	

响应参数

参数名	参数说明	备注
orderId	订单id	
buyerName	用户姓名	
buyerPhone	用户手机号	
buyerAddress	用户地址	
buyerOpenid	用户微信id	
orderAmount	订单总价	
orderStatus	订单状态	完成、取消、待支付
create/updateTime	创建/更新时间	
orderDetaiList	订单详情	具体餐品

传参实例

```
page: 0 //从第0页开始
size: 10
```

返回

```
{
  "code": 0,
  "msg": "成功",
  "data": [
    {
      "orderId": "1618733711711280 75",
      "buyerName": "张三",
      "buyerPhone": "18868877111",
      "buyerAddress": "慕课网总部",
      "buyerOpenid": "18eu2jwk2kse3r42e2e",
      "orderAmount": 0,
      "orderStatus": 0,
      "payStatus": 0,
      "createTime": 1490171219,
      "updateTime": 1490171219,
      "orderDetailList": null
    },
    {
      "orderId": "1618733711711280 76",
      "buyerName": "张三",
      "buyerPhone": "18868877111",
      "buyerAddress": "慕课网总部",
      "buyerOpenid": "18eu2jwk2kse3r42e2e",
      "orderAmount": 0,
      "orderStatus": 0,
      "payStatus": 0,
      "createTime": 1490171219,
      "updateTime": 1490171219,
```

⑤商家端查看商品列表：

```
请求方式：GET
请求路径：/sell/seller/product/list
```

请求参数

参数名	参数说明	备注
shopId	商家id	

响应参数

参数名	参数说明	备注
id	餐品id	
name	餐品名称	
price	餐品价格	
save	库存	
description	餐品描述	
icon	餐品图标路径	

返回

```
{
    "code":0,
    "msg":"成功",
    "data":[
        {
            "id": "123456",
            "name": "皮蛋粥",
            "price": 1.2,
            "save": 128,
            "description": "好吃的皮蛋粥",
            "icon": "http://xxx.com",
        }]
}
```

⑥商品创建/修改菜品信息：

请求方式：POST
请求路径：/sell/seller/product/index

请求参数

参数名	参数说明	备注
id	餐品id	
name	餐品名称	
price	餐品价格	
save	库存	
description	餐品描述	
icon	餐品图标路径	

响应参数

无

返回

```
{
    "code": 0,
    "msg": "成功",
    "data": null
}
```

核心代码：

①用户端订单控制类：

```
@RestController
@RequestMapping("/buyer/order")
@Slf4j
public class BuyerOrderController {

    @Autowired
    private OrderService orderService;

    @Autowired
    private BuyerService buyerService;

    //创建订单
    //采用 HashMap 是因为将前端的"orderId"当成一个字符串变量，而非一个变量名
    @PostMapping("/create")
    public ResultVO<Map<String, String>> create(@Valid OrderForm orderForm,
                                                BindingResult bindingResult)
{
        log.error("【创建订单】参数不正确, orderForm={}", orderForm);
```

```
        if (bindingResult.hasErrors()) {
            log.error("【创建订单】参数不正确, orderForm={}", orderForm);
            throw new SellException(ResultEnum.PARAM_ERROR.getCode(),
                    bindingResult.getFieldError().getDefaultMessage());
        }

        OrderDTO orderDTO = OrderForm2OrderDTOConverter.convert(orderForm);
        if (CollectionUtils.isEmpty(orderDTO.getOrderDetailList())) {
            log.error("【创建订单】购物车不能为空"); //难保购物车不为空
            throw new SellException(ResultEnum.CART_EMPTY);
        }

        OrderDTO createResult = orderService.create(orderDTO);

        Map<String, String> map = new HashMap<>();
        map.put("orderId", createResult.getOrderId());

        return ResultVOUtil.success(map);
    }

    //订单列表
    @GetMapping("/list")
    public   ResultVO<List<OrderDTO>>   list(@RequestParam("openid")   String
openid,
                                             @RequestParam(value    =    "page",
defaultValue = "1") Integer page,
                                             @RequestParam(value    =    "size",
defaultValue = "10") Integer size){
        if(StringUtils.isEmpty(openid)){
            log.error("【查询订单列表】 openid 为空");
            throw new SellException(ResultEnum.PARAM_ERROR);
        }

        PageRequest request = PageRequest.of(page, size);
        Page<OrderDTO> orderDTOPage = orderService.findList(openid, request);
        return ResultVOUtil.success(orderDTOPage.getContent());

    }

    //订单详情
    @GetMapping(value = "/detail")
    public ResultVO<OrderDTO> detail(@RequestParam("openid") String openid,
                                     @RequestParam("orderId")          String
orderId){
        OrderDTO orderDTO = buyerService.findOrderOne(openid, orderId);
        return ResultVOUtil.success(orderDTO);
```

```
    }

    //取消订单
    @PostMapping(value = "/cancel")
    public ResultVO<OrderDTO> cancel(@RequestParam("openid") String openid,
                                     @RequestParam("orderId")          String
orderId){
        buyerService.cancelOrder(openid, orderId);
        return ResultVOUtil.success();
    }

}
```

②用户端查看商品：

```
@RestController
@RequestMapping("/buyer/product")
public class BuyerProductController {

    @Autowired
    private ProductService productService;

    @Autowired
    private CategoryService categoryService;

    @GetMapping("/allList")
    public ResultVO list(){
        //1. 查询所有上架商品
        List<ProductInfo> productInfoList = productService.findUpAll();

        //2. 查询类目（一次性查询）
        List<Integer>     categoryTypeList     =     new     ArrayList<>();
//categoryType 实际上是类目编号，别忘了
        //传统方法
        for(ProductInfo productInfo: productInfoList){
            categoryTypeList.add(productInfo.getCategoryType());
        }
        //精简方法(java8, lambda 表达式)
//        List<Integer> categoryTypeList = productInfoList.stream().
//                map(e -> e.getCategoryType()).
//                collect(Collectors.toList());
        List<ProductCategory>              productCategoryList                =
categoryService.findByCategoryTypeIn(categoryTypeList);     //相同编号的会算
作一种？

        //3. 数据拼装（一层一层的嵌套，让其格式符合前端所需要求）
        List<ProductVO> productVOList = new ArrayList<>();
```

```
        for(ProductCategory productCategory : productCategoryList){
            ProductVO productVO = new ProductVO();
            productVO.setCategoryType(productCategory.getCategoryType());
            productVO.setCategoryName(productCategory.getCategoryName());
            List<ProductInfoVO> productInfoVOList = new ArrayList<>();
            for(ProductInfo productInfo : productInfoList){

if(productInfo.getCategoryType().equals(productCategory.getCategoryType())){
                    ProductInfoVO productInfoVO = new ProductInfoVO();
                    BeanUtils.copyProperties(productInfo,     productInfoVO);
//直接拷贝属性，简化繁琐的 set 方法
                    productInfoVOList.add(productInfoVO);
                }
            }
            productVO.setProductInfoVOList(productInfoVOList);
            productVOList.add(productVO);
        }
        return  ResultVOUtil.success(productVOList);        //success 方法用了
static，甚至不需要创建对象
    }

    @GetMapping("/list")
    public ResultVO proList(@RequestParam("shopId") String shopId){
        List<ProductInfo>                  productInfoList                  =
productService.findShopUpAll(shopId);

        List<Integer>     categoryTypeList     =     new     ArrayList<>();
//categoryType 实际上是类目编号，别忘了
        //传统方法
        for(ProductInfo productInfo: productInfoList){
            categoryTypeList.add(productInfo.getCategoryType());
        }

        List<ProductCategory>              productCategoryList              =
categoryService.findByCategoryTypeIn(categoryTypeList);      //相同编号的会算
作一种？

        List<ProductVO> productVOList = new ArrayList<>();
        for(ProductCategory productCategory : productCategoryList){
            ProductVO productVO = new ProductVO();
            productVO.setCategoryType(productCategory.getCategoryType());
            productVO.setCategoryName(productCategory.getCategoryName());
            List<ProductInfoVO> productInfoVOList = new ArrayList<>();
            for(ProductInfo productInfo : productInfoList){

if(productInfo.getCategoryType().equals(productCategory.getCategoryType())){
```

```
                    ProductInfoVO productInfoVO = new ProductInfoVO();
                    BeanUtils.copyProperties(productInfo,     productInfoVO);
//直接拷贝属性，简化繁琐的 set 方法
                    productInfoVOList.add(productInfoVO);
                }
            }
            productVO.setProductInfoVOList(productInfoVOList);
            productVOList.add(productVO);
        }
        return  ResultVOUtil.success(productVOList);        //success 方法用了
static，甚至不需要创建对象
    }
}
```

③商家端管理订单控制类

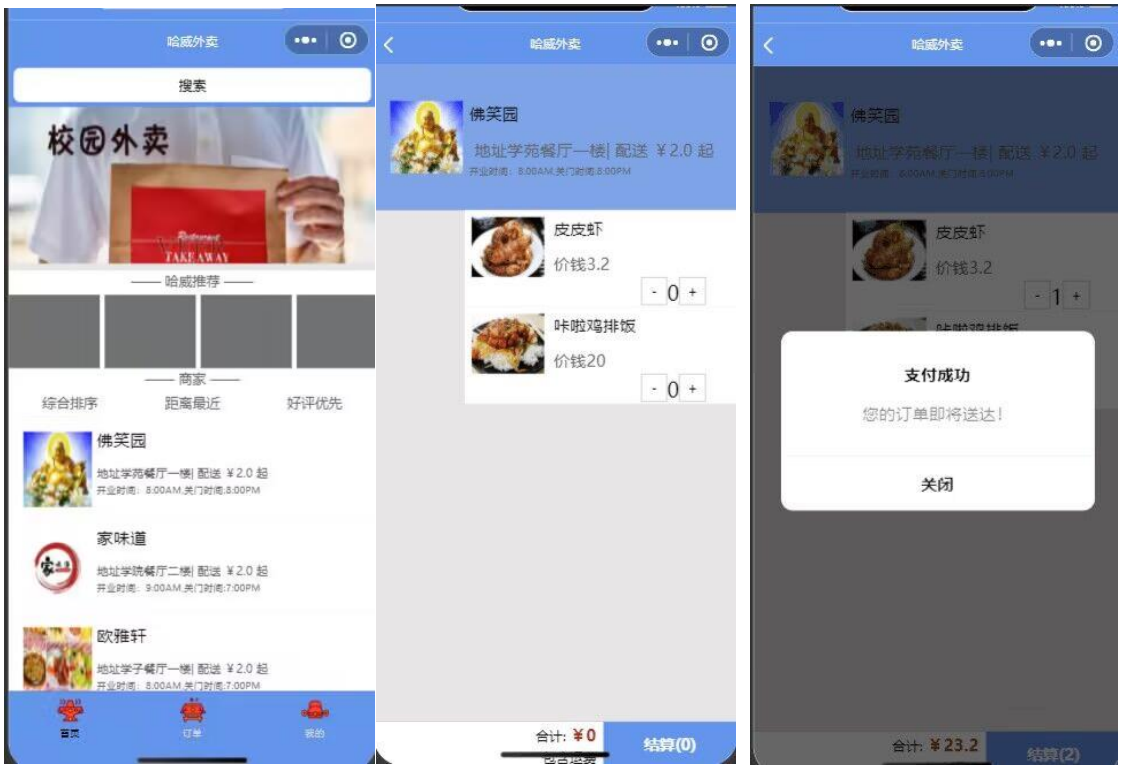
```
@Controller
@RequestMapping("/seller/order")
@Slf4j
public class SellerOrderController {

    @Autowired
    private OrderService orderService;

    @GetMapping("/list")
    public ModelAndView list(@RequestParam(value = "page", defaultValue = "1")
Integer page,
                             @RequestParam(value = "size", defaultValue = "10")
Integer size,
                             @RequestParam(value = "shopId", defaultValue =
"1") String shopId,
                             Map<String, Object> map){
        PageRequest request = PageRequest.of(page - 1, size);
        Page<OrderDTO>  orderDTOPage  =  orderService.findListInShop(shopId,
request) ;
        map.put("orderDTOPage", orderDTOPage);
        map.put("currentPage", page);
        map.put("size", size);
        map.put("shopId", shopId);
        return new ModelAndView("/order/list", map);
    }

    @GetMapping("/cancel")
    public ModelAndView cancel(@RequestParam("orderId") String orderId,
                               @RequestParam(value = "shopId", defaultValue =
"1") String shopId,
```

```
                                Map<String, Object> map){
        try {
            OrderDTO orderDTO = orderService.findById(orderId);
            orderService.cancel(orderDTO);
        }catch (SellException e){
            log.error("【卖家端取消订单】发生异常{}", e);
            map.put("msg", e.getMessage());
            map.put("url", "/sell/seller/order/list?shopId=" + shopId);
            return new ModelAndView("common/error", map);
        }
        map.put("msg", ResultEnum.ORDER_CANCEL_SUCCESS.getMessage());
        map.put("url", "/sell/seller/order/list?shopId=" + shopId);
        return new ModelAndView("common/success");
    }

    @GetMapping("/detail")
    public ModelAndView detail(@RequestParam("orderId") String orderId,
                               @RequestParam(value = "shopId", defaultValue =
"1") String shopId,
                               Map<String, Object> map){
        OrderDTO orderDTO = new OrderDTO();
        try{
            orderDTO = orderService.findById(orderId);
        }catch (SellException e){
            log.error("【卖家端查询订单详情】发生异常{}", e);
            map.put("msg", e.getMessage());
            map.put("url", "/sell/seller/order/list?shopId=" + shopId);
            map.put("shopId", shopId);
            return new ModelAndView("common/error", map);
        }
        map.put("orderDTO", orderDTO);
        map.put("shopId", shopId);
        return new ModelAndView("order/detail", map);
    }

    @GetMapping("/finish")
    public ModelAndView finish(@RequestParam("orderId") String orderId,
                               @RequestParam(value = "shopId", defaultValue =
"1") String shopId,
                               Map<String, Object> map){
        try {
            OrderDTO orderDTO = orderService.findById(orderId);
            orderService.finish(orderDTO);
        }catch (SellException e){
            log.error("【卖家端完结订单】发生异常{}", e);
            map.put("msg", e.getMessage());
```

```
            map.put("url", "/sell/seller/order/list?shopId=" + shopId);
            return new ModelAndView("common/error", map);
        }
        map.put("msg", ResultEnum.ORDER_FINISH_SUCCESS.getMessage());
        map.put("url", "/sell/seller/order/list?shopId=" + shopId);
        return new ModelAndView("common/success");
    }
}
```

功能截图：

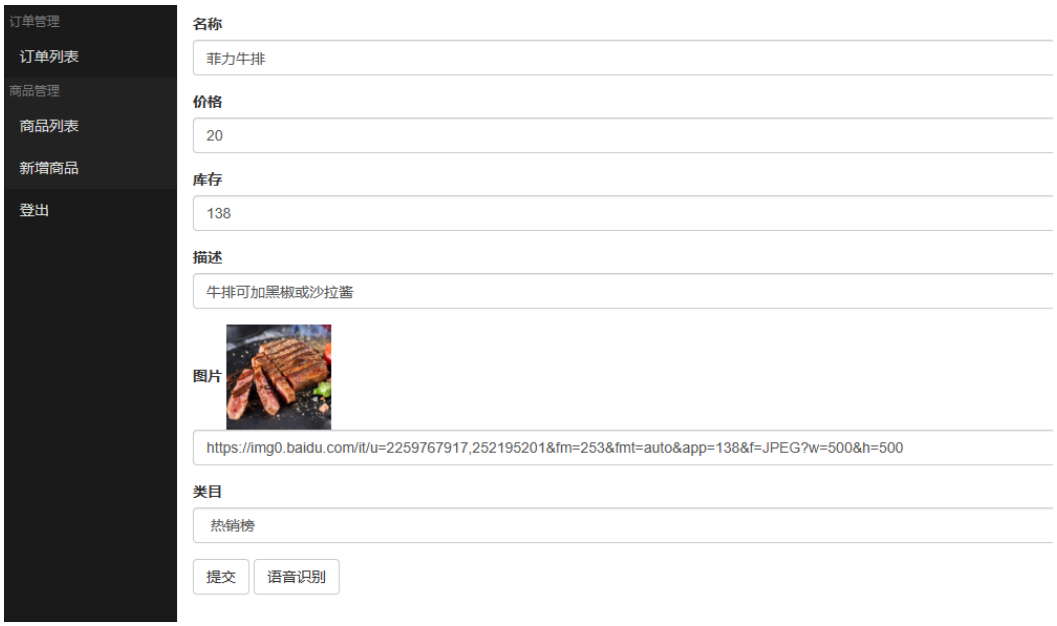
用户端 查看店家 查看商品 支付成功



商家端

查看菜品

创建/修改菜品信息：



查看订单信息：

订单id	姓名	手机号	地址	金额	订单状态	支付状态	创建时间	操作	
123456	阿廷	17650022293	愉景湾	2.3	新订单	等待支付	2022年1月26日 上午3:33:26	详情	取消
123457	阿廷	17650022293	愉景湾	2.5	新订单	等待支付	2022年1月26日 上午3:34:38	详情	取消
123458	阿廷	17650022293	愉景湾	2.5	新订单	等待支付	2022年5月13日 上午11:59:47	详情	取消
1643274237302532732	李彦廷	17650022293	愉景湾	20	已取消	等待支付	2022年1月27日 上午4:25:11	详情	
1643274589837111258	李彦廷	17650022293	愉景湾	26.4	完结	等待支付	2022年1月27日 上午4:31:33	详情	
1643548108508915694	李彦廷	17650022293	愉景湾	26.4	已取消	等待支付	2022年1月30日 下午12:02:55	详情	
1643548449740332261	张三	18868822111	哈工大总部	40	已取消	等待支付	2022年1月30日 下午12:09:05	详情	
1644027499304700632	张三	99999999	哈工大总部	40	新订单	等待支付	2022年2月5日 上午2:18:19	详情	取消
1644722362302142838	张三	12122122112	哈工大总部	29.6	新订单	等待支付	2022年2月13日 上午3:19:22	详情	取消
1644723756208532204	张三	12122122112	哈工大总部	23.2	新订单	等待支付	2022年2月13日 上午3:42:36	详情	取消

6. 团队协作及完成进度安排

（在团队中的任务及任务完成进度表）

时间安排：

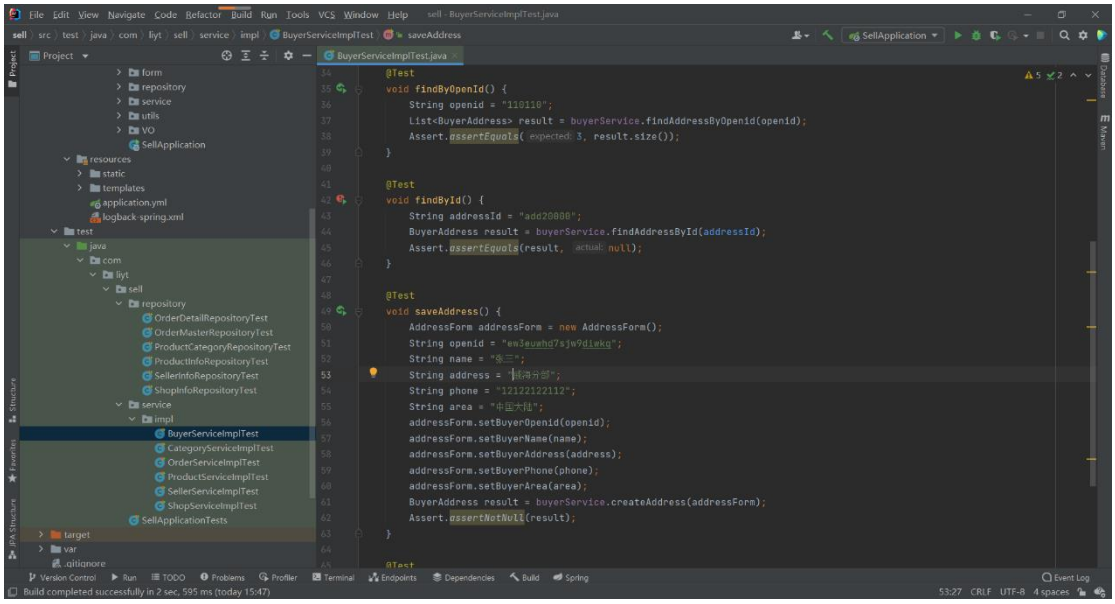
5 月 29 日至 5 月 30 日：
确定任务主题与实现方向，选用的技术栈，撰写任务计划书。
5 月 31 日至 6 月 1 日：
确定所有模块，设计总体类图，共同协作完成 API 文档与数据库的设计。
6 月 2 日至中检：
完成用户端的所有后端的完整设计。
中检至 6 月 7 日：
完成卖家端的部分前端和所有后端的完整设计。
6 月 8 日至验收：
测试客户端和卖家端所有功能，以及二者交互的功能正常

主要任务：
①商家端后端编码
②用户端后端编码
③设计并创建数据库

7. 运行测试与分析

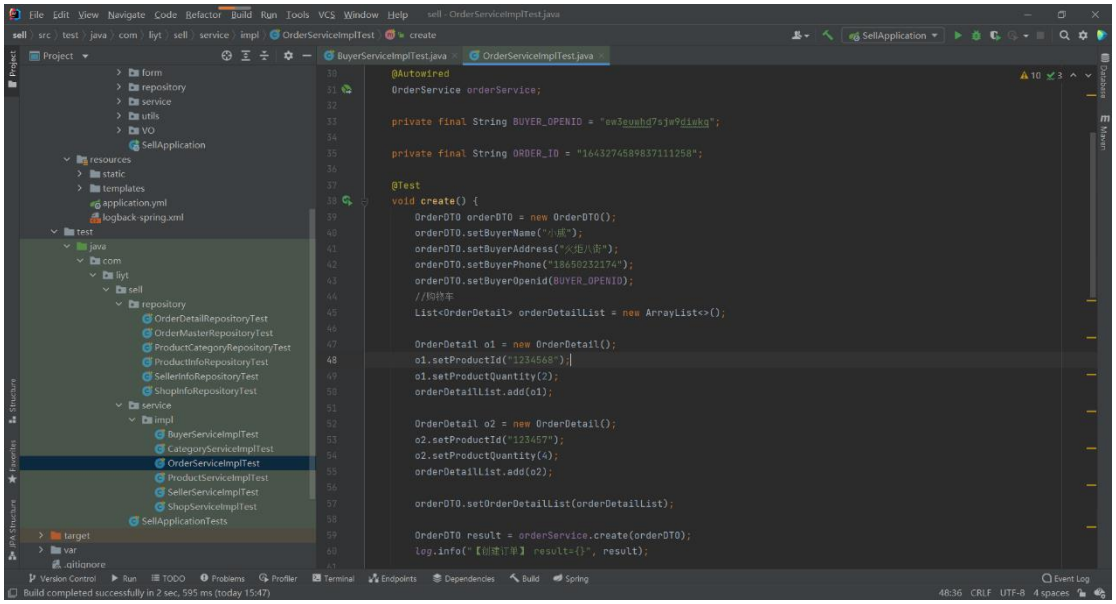
（要有多组实际测试数据，和测试用例并给出相应运行结果截图。具体：（1)完整性约束测试过程设计，结果验证；(2)能给出软件操作的界面截图来。(3）软件开发调试过程中遇到的问题及解决过程；符合实际情况的数据测试及功能的改进设想等）

买家信息测试类:



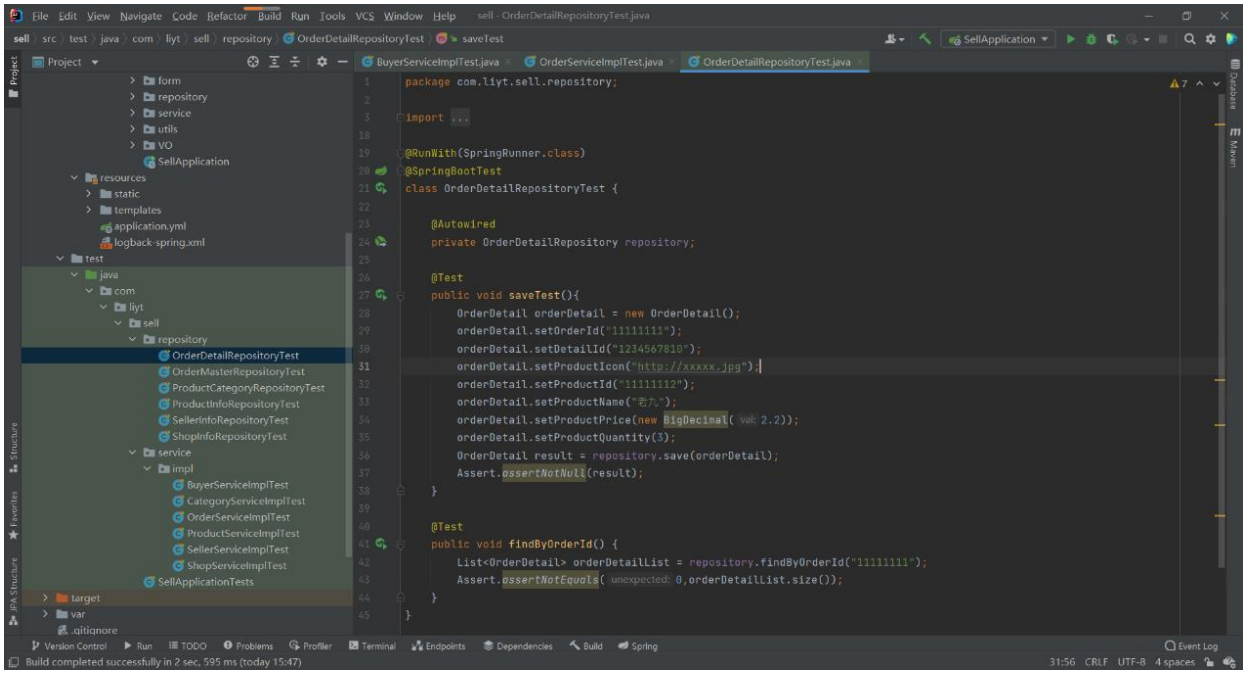
测试结果：买家填写地址时，成功会录入数据库错误会提示错误信息

点单测试类：



点单测试结果：点单过程无错误则计入订单，有错误则可以弹出错误信息

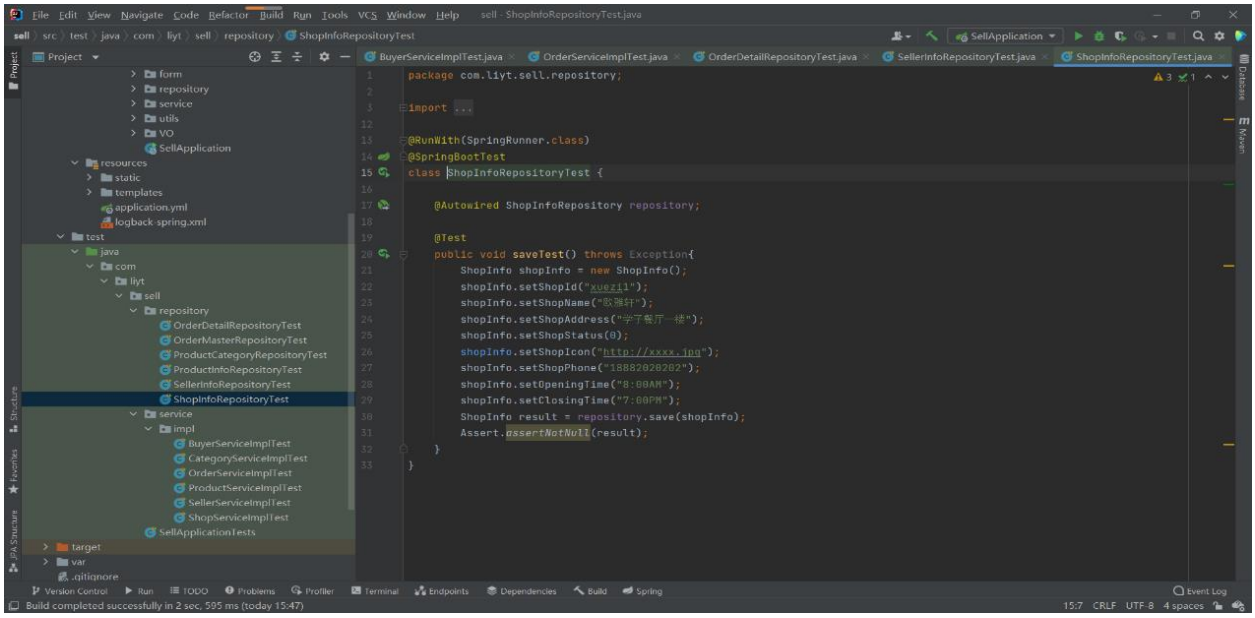
订单详情测试类：



```
package com.liyt.sell.repository;

import ...

@RunWith(SpringRunner.class)
@SpringBootTest
class OrderDetailRepositoryTest {

    @Autowired
    private OrderDetailRepository repository;

    @Test
    public void saveTest(){
        OrderDetail orderDetail = new OrderDetail();
        orderDetail.setOrderId("11111111");
        orderDetail.setDetailId("1234567810");
        orderDetail.setProductIcon("http://xxxxx.jpg");
        orderDetail.setProductId("11111112");
        orderDetail.setProductName("老九");
        orderDetail.setProductPrice(new BigDecimal( val: 2.2));
        orderDetail.setProductQuantity(3);
        OrderDetail result = repository.save(orderDetail);
        Assert.assertNotNull(result);
    }

    @Test
    public void findByOrderId() {
        List<OrderDetail> orderDetailList = repository.findByOrderId("11111111");
        Assert.assertNotEquals( unexpected: 0,orderDetailList.size());
    }
}
```

测试结果：可以正确查看订单详情

商家端测试类：



```
package com.liyt.sell.repository;

import ...

@RunWith(SpringRunner.class)
@SpringBootTest
class ShopInfoRepositoryTest {

    @Autowired ShopInfoRepository repository;

    @Test
    public void saveTest() throws Exception{
        ShopInfo shopInfo = new ShopInfo();
        shopInfo.setShopId("xuezi1");
        shopInfo.setShopName("欧雅轩");
        shopInfo.setShopAddress("学子餐厅一楼");
        shopInfo.setShopStatus(0);
        shopInfo.setShopIcon("http://xxxx.jpg");
        shopInfo.setShopPhone("18882020202");
        shopInfo.setOpeningTime("8:00AM");
        shopInfo.setClosingTime("7:00PM");
        ShopInfo result = repository.save(shopInfo);
        Assert.assertNotNull(result);
    }
}
```

测试结果：登录时可以弹出提示，登录失败弹出失败提示，登录成功跳转首页

8. 问题及难点

遇到的问题及难点：

①前端界面无法获取数据：与队友数据库 sql 文件存在出入，导致表中字段缺失，dao 无法获取对应数据，从而返回的 json 文件中存在空值，浪费了大量测试时间。在统一 sql 文件，重新建表并导入测试数据后问题解决。

②语音输入功能：

刚开始对于添加该功能无从下手，在 csdn 与 chatgpt 搜索后，学习了相关知识，引入了科大讯飞的语音识别接口，完成了基础项目配置，初步实现了语音识别功能。尽管取得一定成果，但在和原项目整合时存在问题。此处开头，留待以后解决。

9. 总结（收获与体会）

（如实撰写课程任务完成过程的收获和体会以及遇到问题的思考、程序调试能力的培养提升等相关内容；要求不少于 500 字，严禁雷同）

本次课程设计中，我们小组通过同步密切探讨和合理划分任务量等团队合作完成了对校园外卖小程序的整体分析和设计，完成了需求分析建模、数据库设计、前后端联合开发等内容，并对每个阶段的任务量进行合理划分，积极协作，共同解决遇到的问题，取得了有效的成果。

在这次课程设计后，我们学习并巩固了项目设计的过程，深切体会到了系统分析对一个大项目完成的必要性。只有在项目开始前，先对系统的各个部分提出预期，与小组成员有目的性地一步步分析，才能保障项目的安全性和健全性，为后续的开发工作提供清晰的指导。

在团队合作中，组内成员也遇到了一些分歧和挑战，最终通过查阅 csdn、与队友积极探讨和共同协商，问题都获得了解决。同时也在绘制各种模型图时对建模软件的使用更加熟悉，收获颇丰。同时通过这次试验，我对 Springboot 框架有了实质性的掌握，由之前的理论学习上升到了动手实践，深刻领会了该技术的使用方法。

当然我们也意识到自己的不足之处，由于时间有限，没有完整地加入语音识别功能，也没有利用缓存技术如 Redis 和安全框架。在日后改进和完善该项目的过程中，我们将进一步跟进和学习这些技术，提升自己的能力。通过这次宝贵的经验，我相信我们小组将在未来的项目中取得更加优秀的成绩。