“OK帮云进销存”软件开发文档

# 1. 引言

## 1.1选题背景

进销存软件概念起源于上世纪80年代，兴于2000年以后，很大程度上解决了企业日常经营中一直存在的商品太多无法记住价格及库存的问题。但这十几年中，进销存软件的发展可以说是非常缓慢，大部分进销存软件开发端还停留在功能堆叠的单机时代，适应不了当下销售潮流，造成了“用也不是，不用也不是”的尴尬局面。

OK帮科技在河南地区调查了多家小商品批发商，发现居然还有80%商家还使用纸笔记账，理由是“买不起，也不是很好用”；剩下的已经使用进销存软件的商家中，全部使用了单机版进销存（或通过远程桌面、离线同步联网），均存在“很多不便，什么都要跑到电脑前操作，但没办法，货品太多人脑记不了，将就着用吧”的问题。

而如今，已经进入全民移动时代，人手一部智能机，随时随地可高速上网；另一方面，企业及个体户所需要面对的形势更为复杂、各类成本日益攀升。OK帮科技认为大变革的时代到了，OK帮云进销存提供手机端，根据移动的特性，对库存管理、销售管理及采购管理等做了大量的场景针对性开发。

OK帮云进销存系统具有传统进销存系统无法比拟的优越性，它能使业务摆脱时间和场所局限，随时进行随地与公司业务平台沟通，有效提高管理效率，推动企业效益增长。

OK帮云进销存也将会采取互联网时代最流行的免费模式，所有商家可以终身免费使用。

未来，办公类应用一定会在商户中普及。OK帮科技会不断联合内外部资源，加快企业级SAAS应用在中国的普及进程，为行业信息化建设、社会效率提升贡献应有价值。

## 1.2范围

该系统主要应用于商户的信息管理，主要包括以下业务：

该系统主要有用户管理子系统、角色权限管理子系统、销售管理子系统、库存管理子系统、分析管理子系统五大子系统。

其中：

用户管理子系统包括：个人信息管理、公司信息管理、用户注册登陆、店铺注册等模块。

角色权限管理子系统包括：角色信息管理、员工信息管理、角色权限管理、员工角色管理等模块。

销售管理子系统包括：客户信息管理、商品信息管理、分类信息管理、商品分类管理、销售单信息管理、支付管理等模块。

库存管理子系统包括：库存信息管理、供应商信息管理、商品发货管理等模块。

分析管理子系统包括：销售报表分析、滞/热销商品计算、客户对账、供应商对账、支付账单流水等模块。

此系统为平台系统，以上模块，用户仅在自己店铺中可以管理自己的信息。

2.需求分析

2.1 系统功能需求

2.1.1 用户管理子系统

个人信息管理：个人具体信息的查看，修改：要求能够对用户进行修改，包含用户昵称，性别，生日，邮箱等。

2.1.2 角色权限管理子系统

2.1.2.1角色信息管理

系统注册者用户可以查看自己商店的角色信息及系统原始角色信息；在自己的店铺中：系统注册者可以添加、删除、修改角色信息（系统原始角色不可修改）。

2.1.2.2员工信息管理

拥有员工管理权限的用户可以注册员工账号；删除、修改员工信息及为员工赋予角色。员工可以使用已注册的员工账号登陆系统。

2.1.2.3角色权限管理

系统注册者用户可以在增加角色时为角色选择权限，也可以在修改角色时修改角色已有权限（权限对应页面功能，为系统固定权限，不可编辑）。

2.1.2.4员工角色管理

拥有员工管理权限的用户可以在注册员工时为员工选择角色，也可以在修改员工时修改角色。角色选择可以为多个，权限重叠部分自动合并。

2.1.3 销售管理子系统

2.1.3.1客户信息管理

拥有客户管理权限用户可以查看自己商店的客户信息、添加、删除、修改客户信息。

2.1.3.2商品信息管理

拥有商品管理权限的用户可以添加、删除、修改商品信息。所有的本店用户可以查询商品信息。

2.1.3.3分类信息管理

拥有商品管理权限的用户可以添加、删除、修改分类信息。所有的本店用户可以查询分类信息。

2.1.3.4商品分类管理

拥有商品管理权限的用户可以添加商品时选择分类或者修改商品信息时改变其所属分类。

2.1.3.5销售单信息管理

拥有销售单信息管理权限的用户可以添加、查看销售单。（销售单不可修改、删除）

2.1.3.6支付管理

拥有支付权限的用户可以在添加销售单后对销售单进行支付操作；对尚有欠款销售单进行还款支付操作。

2.1.4 库存管理子系统

2.1.4.1供应商信息管理

拥有供应商管理权限用户可以查看自己商店的客户信息、添加、删除、修改供应商信息。

2.1.4.2商品库存信息管理

拥有商品库存管理权限的用户可以查看、添加、删除商品库存信息。

2.1.4.3入库单信息管理

拥有入库单信息管理权限的用户可以添加、查看入库单。（销售单不可修改、删除）

2.1.4.4仓库管理

拥有仓库管理权限用户可以查看自己商店的客户信息、添加、删除、修改仓库信息。

2.1.5 分析管理子系统

2.1.5.1销售报表分析

拥有分析管理权限用户可以查看自己商店的销售分析信息。

2.1.5.2滞/热销商品计算

拥有商品管理权限或者分析管理权限或者销售管理权限的用户可以查询最近日期的滞/热商品信息以便快速出售商品。

2.1.5.3客户对账

拥有分析管理权限的用户可以查看客户账单信息，重点显示当前欠款客户，以便提示店铺收款。

2.1.5.4供应商对账

拥有分析管理权限的用户可以查看供应商账单信息，以便店铺对货。

2.1.5.5支付账单流水

拥有分析管理权限的用户可以查看销售支付流水汇总。

2.2 非功能性需求

2.2.1 性能需求

（1）客户端一般响应时间（除报表统计、数据导入）不超过3秒。

（2）报表统计时间不超过30秒。

（3）支持1000单销售单信息的同时并发，并发时间不超过300秒。

（4）支持5000名用户并发使用，并保证性能不受影响。

2.2.2 安全性需求

（1）权限控制

根据不同用户角色，设置相应权限，用户的重要操作都做相应的日志记录以备查看，没有权限的用户禁止使用系统。

（2）重要数据加密

本系统对一些重要的数据按一定的算法进行加密，如用户口令等。

（3）数据备份

可以配置定时进行数据的备份，以弥补数据的破坏和丢失。

（4）记录日志

本系统应该能够记录系统运行时所发生的错误信息，这些错误记录便于查找错误的原因。日至同时记录用户的关键性操作的信息。

2.2.3 可用性需求

（1）方便操作，操作流程合理

尽量从用户角度出发，以方便使用本产品。如：可以通过摄像头扫描商品条码代替手动输入。

（2）操作简单

支持没有计算机使用经验、计算机使用经验较少及有较多计算机使用经验的用户均能方便地使用本系统。

（3）数据校验

本系统能够对必填项、特殊格式（电话、邮箱等）的项目进行控制，使用户能够确保信息录入的完整。同时进行有效的统一的提示。

（4）容错能力

系统具有一定的容错和抗干扰能力，在非硬件故障或非通讯故障时，系统能够保证正常运行，并有足够的提示信息帮助用户有效正确地完成任务。

（5）操作完成时有统一规范的提示信息

例如删除操作时，系统可提示警示框“您确认删除记录吗？操作不可恢复！”，用户点击确认后，系统才执行删除操作。

（6）用户可自定义

为了满足业务的不断变化，一些重要的系统参数应该可以灵活设置。

2.2.4 其他需求

（1）支持多浏览器，可跨平台（手机、平板）。

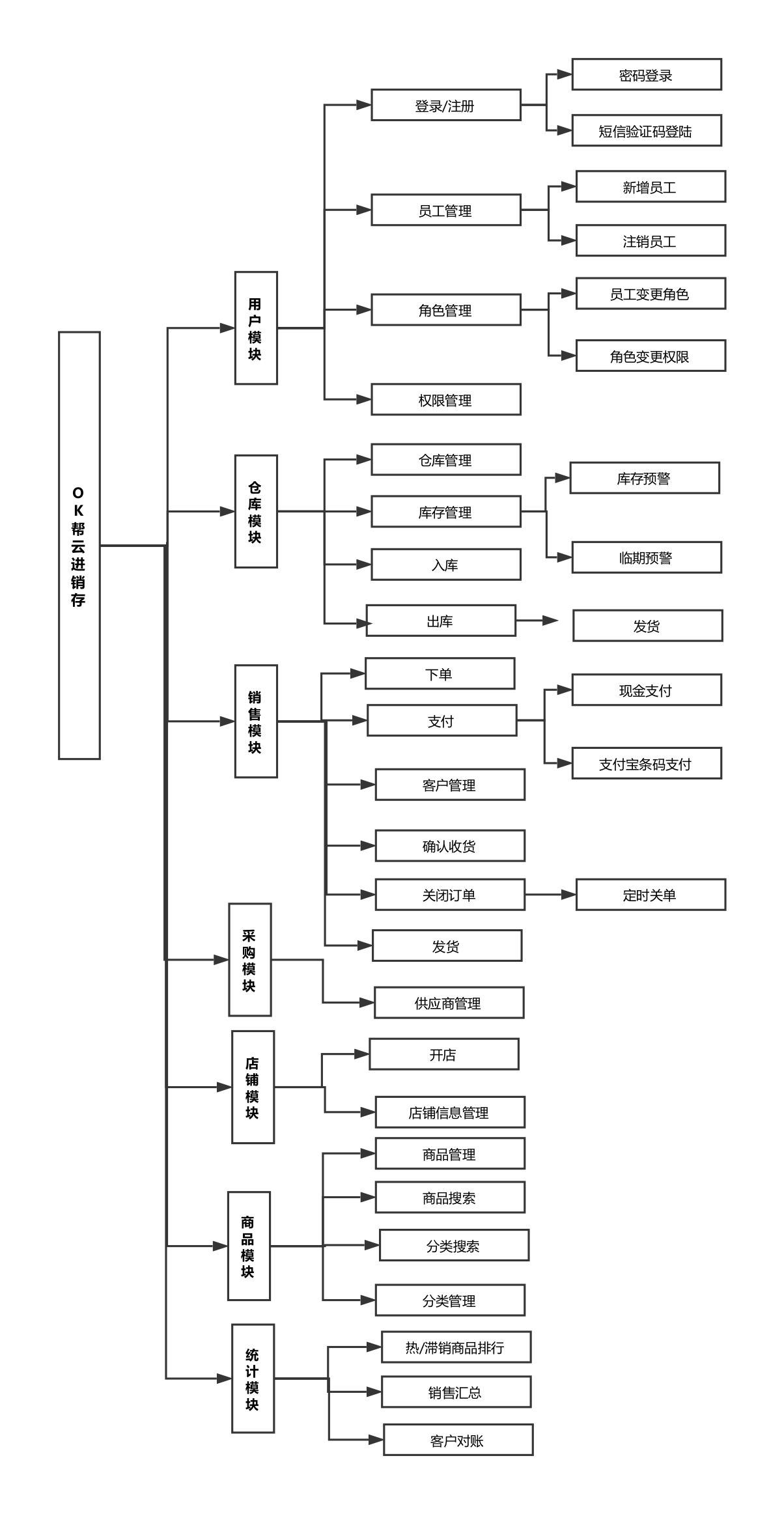
（2）有苹果系统app和安卓系统的apk，便于用户根据需要选择安装

（3）系统安装方便，易于维护。

3.概要设计

3.1 系统逻辑设计

3.1.1系统结构设计，如图1系统结构图



3.1.2体系设计，如图1系统架构图

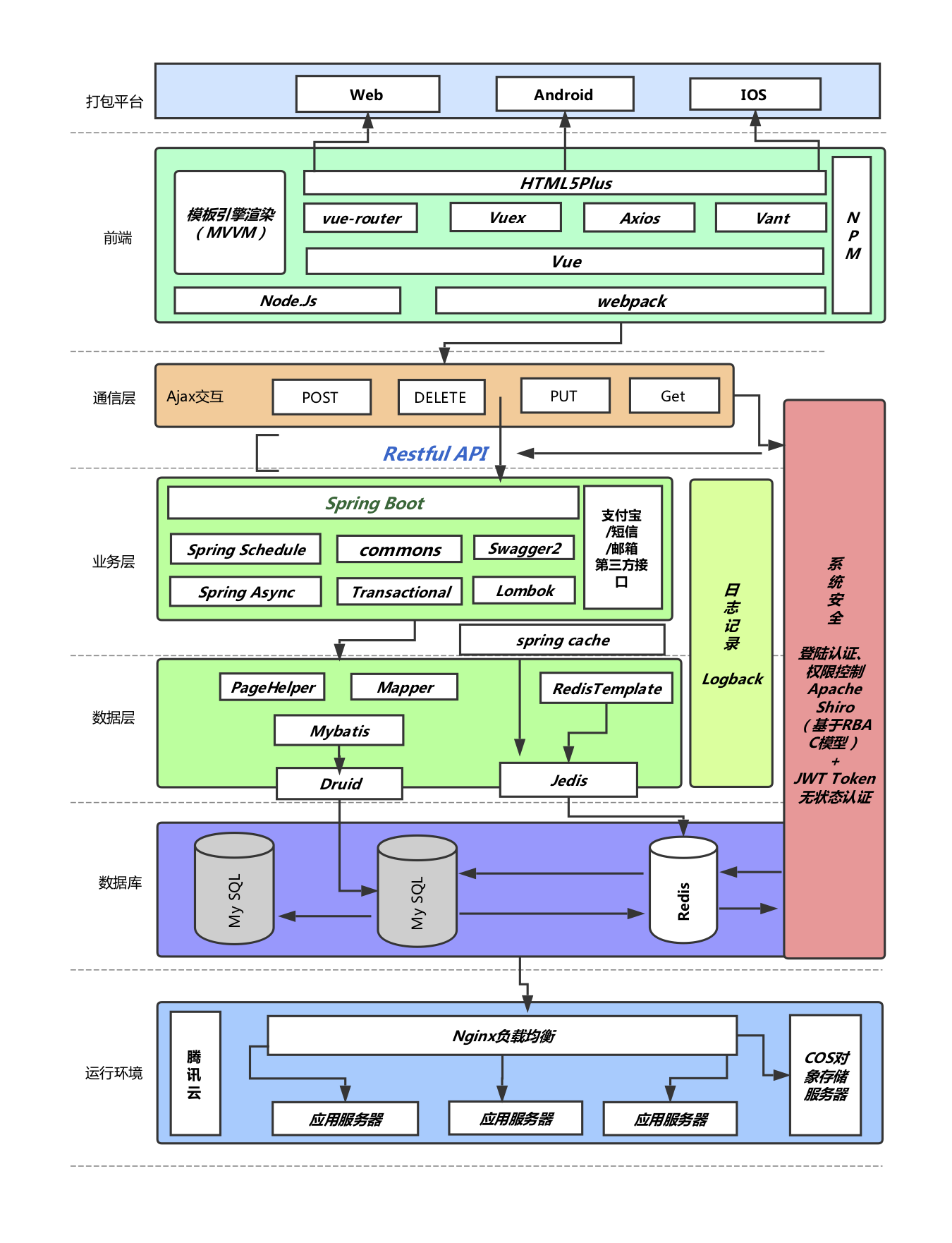


图1系统架构图

3.1.3接口设计

本人负责实现销售子系统、角色权限子系统和分析子系统的接口设计，本系统采用AJAX调用接口。

（1）接口规范

|  |  |
| --- | --- |
| 项目 | 内容 |
| 通信协议 | HTTP |
| 数据类型 | JSON |
| 调用方法 | AJAX |

（2）接口列表

|  |  |  |
| --- | --- | --- |
| 模块 | 描述 | 接口 |
| 角色 | 获取角色列表 | /api/role |
| 添加角色 | /api/role |
| 权限 | 获取我的权限列表 | /api/permission/me |
| 获取我的菜单列表 | /api/permission/menu/me |
| 客户 | 查询所有客户 | /api/customer |
| 添加客户 | /api/customer |
| 查询单个客户 | /api/customer/{id} |
| 修改客户信息 | /api/customer/{id} |
| 删除客户 | /api/customer/{id} |
| 仓库 | 查询所有仓库 | /api/warehouse |
| 添加仓库 | /api/warehouse |
| 查询单个仓库 | /api/warehouse/{id} |
| 修改仓库信息 | /api/warehouse/{id} |
| 删除仓库 | /api/warehouse/{id} |
| 供应商 | 查询所有供应商 | /api/supplier |
| 添加供应商 | /api/supplier |
| 查询单个供应商 | /api/supplier/{id} |
| 修改供应商信息 | /api/supplier/{id} |
| 删除供应商 | /api/supplier/{id} |
| 存储 | 所有入库单 | /api/storage |
| 入库操作 | /api/storage |
| 查询供应商入库单 | /api/storage/supplier/{supplierId} |
| 入库单查询 | /api/storage/{number} |
| 分类 | 所有类查询 | /api/categorys |
| 添加分类 | /api/categorys |
| 添加分类 | /api/categorys/{id} |
| 删除分类 | /api/categorys/{id} |
| 出货 | 发货/出库 | /api/delivery/deliver\_goods/{id} |
| 查询未发货订单 | /api/delivery/unsend\_orders |
| 统计报表 | 客户欠款查询 | /api/report/customer\_debt |
| 热/滞销产品 | /api/report/hot\_cold\_sale |
| 获取销售汇总 | /api/sale/total |
| 用户 | 检查用户名 | /api/user/check\_username |
| 获取员工 | /api/user/employee |
| 添加员工 | /api/user/employee |
| 用户登录 | /api/user/login |
| 获取我的信息 | /api/user/me |
| 发短信 | /api/user/phoneCode |
| 短信验证 | /api/user/phoneLogin |
| 店长注册 | /api/user/register |
| 变更角色 | /api/user/role/{id} |
| 商品 | 添加商品 | /api/product |
| 修改商品 | /api/product |
| 通过条形码搜索商品 | /api/product/bar\_code/{barCode} |
| 通过分类搜索商品 | /api/product/category |
| 商品库存预警 | /api/product/lowWarning/{numbers} |
| 查询临期商品 | /api/product/nearDay/{days} |
| 通过指定条件搜索 | /api/product/search |
| 查询单个商品 | /api/product/{id} |
| 删除商品 | /api/product/{id} |
| 店铺 | 获取商店信息 | /api/store |
| 修改商店信息 | /api/store |
| 文件上传 | 上传并修改我的头像 | /api/upload/avator/me |
| 上传图片(包括商品图片、分类图片等等，用户头像请转到头像上传接口) | /api/upload/img |
| 上传并修改收款码(支付宝/微信) | /api/upload/money\_code |
| 上传并修改商铺logo | /api/upload/store\_logo |
| 销售 | 关闭订单 | /api/sale/close\_order/{id} |
| 确认收货 | /api/sale/confirm\_receipt/{id} |
| 付款 | /api/sale/payment/{id} |
| 下订单 | /api/sale/place\_order |
| 获取销售历史订单 | /api/sale/sale\_table |
| 获取指定订单 | /api/sale/{id} |

（3）接口示例

请求

|  |  |  |
| --- | --- | --- |
| 方法 | URL | 参数 |
| get | http://www.zxadwz.com/api/warehouse | id |

响应

|  |  |
| --- | --- |
| 状态 | 结果 |
| 成功 | {  "status" : 0,  "msg" : "删除成功"  } |
| 失败 | {  "status": 1,  "msg" : "需要管理权限"  } |
| {  "status" : 2,  "msg" : "需要用户登录"  } |

3.2 数据库设计

本系统数据库数据表列如下图。如图3数据库的物理模型图

所示：

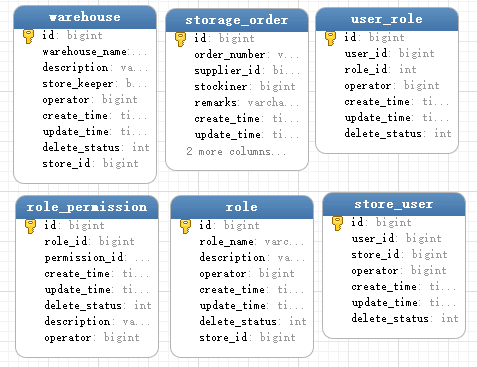
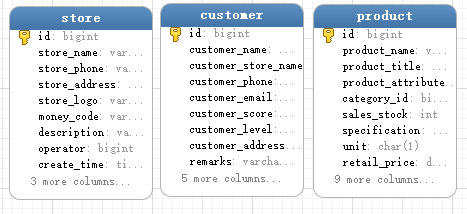
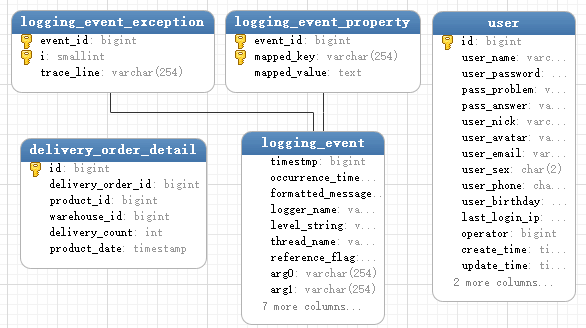
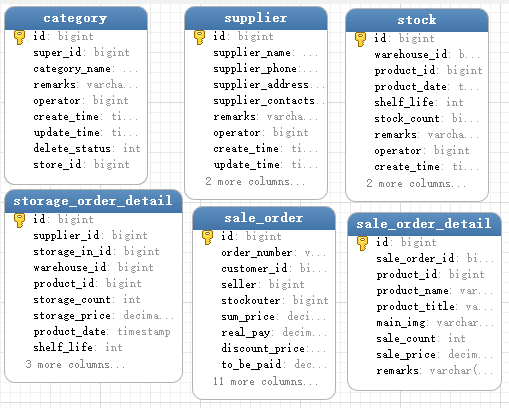
 

图3 数据库的物理模型图

4.详细设计

本人负责架构设计，销售模块、角色权限模块接口设计和后端代码编写。

4.1 角色权限子系统

角色权限子系统可以保证用户登陆之后只在自己的权限下进行操作，例如，系统注册者（店长）拥有所有权限可以进行销售管理、商品管理等，而店长注册的员工账号只有部分权限，如图4店长账号登录，如图5无权限员工登陆。

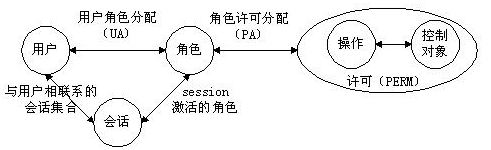


图4 RBAC模型

);

}

4.2 销售子系统

销售子系统包含客户信息管理、商品信息管理、分类信息管理、商品分类管理、销售单信息管理、支付管理等模块。

核心模块为商品信息管理、分类信息展示、销售单管理和支付管理。如图6扫描条码，图7无限层级分类树，图8销售单，图9支付

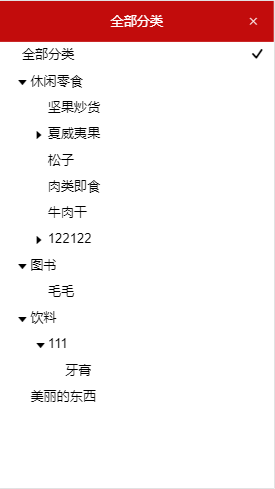
 

图6扫描条码 图7无限层级分类树

图8销售单 图9支付

4.2.1 文件上传

@PostMapping(value = **"/upload/img"**)  
@ApiOperation(value = **"上传图片(包括商品图片、分类图片等等，用户头像请转到头像上传接口)"**, notes = **"此接口只上传图片，无业务逻辑"**)  
@ApiImplicitParams(@ApiImplicitParam(name = **"file"**, value = **"文件"**, required = **true**, dataType = **"File"**))  
**public** ServerResponse<UploadVo> uploadImg(@RequestParam(**"file"**) MultipartFile file, HttpServletRequest request) {  
 **if** (!file.isEmpty() && file.getContentType().startsWith(**"image"**)) {  
 *//定义临时文件夹* String tmp\_path = request.getSession().getServletContext().getRealPath(**"tmp"**);  
 String targetFileName = **uploadService**.upload(file, tmp\_path, **cosPathImg**);  
 String url = **cosHttpPrefix** + **cosPathImg** + targetFileName;  
 UploadVo uploadVo = **new** UploadVo(targetFileName, url);  
 **return** ServerResponse.*createBySuccess*(uploadVo);  
 } **else** {  
 **throw new** IllegalException(**"请求文件文件格式不对"**);  
 }  
  
}

4.2.2 下单

*/\*\*  
 \** ***@Author*** *zc  
 \** ***@Date*** *2018/4/29 上午11:00  
 \** ***@Param*** *[storeId, seller, placeOrderDto]  
 \** ***@Return*** *java.lang.String 返回订单vo  
 \** ***@Description:下单*** *\*/* @Override  
 @Transactional  
 **public** PlaceOrderVo placeOrder(Long storeId, Long seller, PlaceOrderDto placeOrderDto) {  
 List<PlaceOrderItemDto> placeOrderItemDtos = placeOrderDto.getPlaceOrderItemDtos();  
  
 *//加入销售订单* SaleOrder saleOrder = **new** SaleOrder();  
*// saleOrder.setToBePaid(toBePaid);* BeanUtils.*copyProperties*(placeOrderDto, saleOrder);  
 saleOrder.setOrderNumber(NumberGenerator.*generatorOrderNumber*(ConstStr.***ODERTR\_NUM\_PREFIX\_SALE***, seller));  
 saleOrder.setOrderStatus(ConstEnum.***SALESTATUS\_NOPAYMENT***.getCode());  
 saleOrder.setStoreId(storeId);  
 saleOrder.setSeller(seller);  
 saleOrder.setLogisticsStatus(ConstEnum.***LOGISTICSSTATUS\_NOSEND***.getCode());  
  
 **saleOrderMapper**.insertSelective(saleOrder);  
  
 **if** (CollectionUtils.*isNotEmpty*(placeOrderItemDtos)) {  
 *//插入订单子项* assembleSaleOrderDetail(placeOrderItemDtos, saleOrder.getId());  
 *//减商品库存* **otherService**.checkAndCutStock(placeOrderItemDtos);  
 }**else** {  
 **throw new** IllegalException(**"请添加商品"**);  
 }  
 *//返回vo* PlaceOrderVo placeOrderVo = **new** PlaceOrderVo();  
 BeanUtils.*copyProperties*(saleOrder, placeOrderVo);  
 **return** placeOrderVo;  
 }

4.2.4 支付实现

支付模块使用了支付宝的沙箱环境，后端通过调用后端接口实现，后端可以通过扫描别人的二维码进行扣款。

*/\*\*  
 \** ***@Author*** *zc  
 \** ***@Date*** *2018/5/5 下午5:24  
 \** ***@Param*** *[saleOrderId, paymentDto]  
 \** ***@Return*** *void  
 \** ***@Description:支付*** *\*/***public void** payment(Long saleOrderId, PaymentDto paymentDto) {  
 SaleOrder saleOrder = **saleOrderMapper**.selectByPrimaryKey(saleOrderId);  
 **if** (saleOrder == **null**) {  
 **throw new** NotFoundException(**"订单不存在"**);  
 }  
 **if** (ObjectUtils.*notEqual*(saleOrder.getStoreId(), JWTUtil.*getStoreId*())) {  
 **throw new** AuthorizationException(**"资源不在你当前商铺查看范围"**);  
 }  
 **if** (ConstEnum.***SALESTATUS\_CLOSE***.getCode() == saleOrder.getOrderStatus()) {  
 **throw new** IllegalException(**"订单已关闭"**);  
 }  
 **if** (ConstEnum.***SALESTATUS\_SUCCESS***.getCode() == saleOrder.getOrderStatus()) {  
 **throw new** IllegalException(**"交易已完成"**);  
 }  
 **if** (ConstEnum.***SALESTATUS\_PAID***.getCode() == saleOrder.getOrderStatus()) {  
 **throw new** IllegalException(**"订单已支付全款，不要再支付了"**);  
 }  
 **if** (saleOrder.getToBePaid().subtract(paymentDto.getRealPay()).doubleValue()<0.0) {  
 **throw new** IllegalException(**"实付金额不能大于应付金额"**);  
 }  
  
 *//生成支付流水号(我们自己的)* String payNumber  
 = NumberGenerator.*generatorPayMentOrderNumber*(saleOrderId, paymentDto.getTradeType(), paymentDto.getPayType());  
  
 String aliPayTradeNumber=**null**;  
  
 *//支付宝扣款* **if**(String.*valueOf*(ConstEnum.***PAYTYPE\_ALIPAY***.getCode()).equals(paymentDto.getPayType())){  
 **if**(StringUtils.*isBlank*(paymentDto.getAliPayAuthCode())){  
 **throw new** IllegalException(**"支付宝付款码不能为空"**);  
 }  
 **if**(ConstEnum.***TRADETYPE\_FIRST***.getCode()==paymentDto.getTradeType()){  
 aliPayTradeNumber=**aliPayUtil**.alipay(payNumber,paymentDto.getAliPayAuthCode(),paymentDto.getRealPay().toString(),**"0.00"**,**"OK帮下单支付-"**+saleOrder.getOrderNumber());  
 }**else if**(ConstEnum.***TRADETYPE\_REPAYMENT***.getCode()==paymentDto.getTradeType()){  
 aliPayTradeNumber=**aliPayUtil**.alipay(payNumber,paymentDto.getAliPayAuthCode(),paymentDto.getRealPay().toString(),**"0.00"**,**"OK帮订单还款-"**+saleOrder.getOrderNumber());  
 }  
 }  
  
  
 *//更新订单信息  
  
 //(历史已经付款金额+这次实付金额)* BigDecimal realPay = saleOrder.getRealPay().add(paymentDto.getRealPay());  
 *//欠款金额=应付金额-(历史已经付款金额+这次实付金额)* BigDecimal toBePaid  
 = saleOrder.getSumPrice().subtract(realPay);  
  
 saleOrder.setRealPay(realPay);  
 saleOrder.setToBePaid(toBePaid);  
 saleOrder.setPayTime(**new** Date());  
  
 *//判断物流状态决定是否完成订单* **if** (toBePaid.doubleValue() > 0.0) {  
 saleOrder.setOrderStatus(ConstEnum.***SALESTATUS\_DEBT***.getCode());  
 } **else** {  
 saleOrder.setOrderStatus(ConstEnum.***SALESTATUS\_PAID***.getCode());  
 **if** (ConstEnum.***LOGISTICSSTATUS\_RECEIVED***.getCode() == saleOrder.getLogisticsStatus()) {  
 saleOrder.setOrderStatus(ConstEnum.***SALESTATUS\_SUCCESS***.getCode());  
 }  
 }  
  
 *//统计支付方式,金额* **try** {  
 ObjectMapper objectMapper=**new** ObjectMapper();  
 Map<String,String> map = objectMapper.readValue(saleOrder.getPayType(), Map.**class**);  
 String dbpayMentTypePrice = map.get(paymentDto.getPayType());  
 **if**(dbpayMentTypePrice==**null**){  
 **throw new** IllegalException(**"payType参数错误"**);  
 }  
 *//数据库中该支付方式金额* BigDecimal dbpayMentTypePriceDecimal=**new** BigDecimal(dbpayMentTypePrice);  
 BigDecimal newpayMentTypePriceDecimal = dbpayMentTypePriceDecimal.add(paymentDto.getRealPay());  
 map.put(paymentDto.getPayType(),newpayMentTypePriceDecimal.toString());  
 String newPayType = objectMapper.writeValueAsString(map);  
 saleOrder.setPayType(newPayType);  
 } **catch** (IOException e) {  
 ***log***.error(**"异常：{}"**,e.getMessage());  
 *//支付宝退款* **aliPayUtil**.refund(saleOrderId.toString(),aliPayTradeNumber,realPay.toString());  
 **throw new** IllegalException(**"系统异常，支付失败"**);  
 }  
  
  
 *//更新数据库* **int** i = **saleOrderMapper**.updateByPrimaryKeySelective(saleOrder);  
 **if** (i <= 0) {  
 *//支付宝退款* **aliPayUtil**.refund(saleOrderId.toString(),aliPayTradeNumber,realPay.toString());  
 **throw new** IllegalException(**"支付失败"**);  
 }

4.3 统计分析子系统

统计分析子系统包括滞/热销排行，客户对账，供应商对账等，主要为查询信息，以便用户可以及时分析自己的经营状况，调整销售策略。如图10滞/热销排行，图11客户对账。

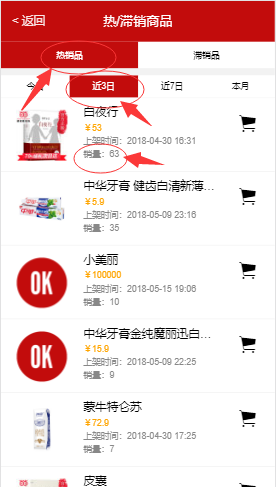
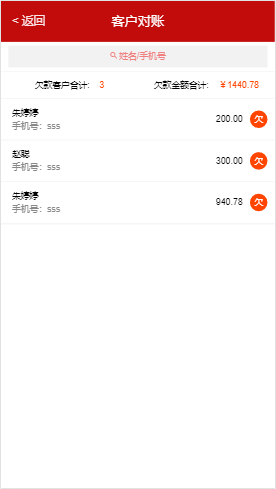
 

图10滞/热销排行 图11客户对账

4.3.1 滞/热销查询

*/\*\*  
 \** ***@Author*** *zc  
 \** ***@Date*** *2018/5/6 下午1:50  
 \** ***@Param*** *[range]  
 \** ***@Return*** *java.util.List*<*com.ok.okhelper.pojo.vo.SalesVolumeVo*>  
 *\** ***@Description:获取热销商品排行*** *\*/*@Override  
**public** List<SalesVolumeVo> getHotSaleByRedis(String range) {  
 Set<ZSetOperations.TypedTuple<String>> projectIdTop10 = **null**;  
 String zkey\_today = ConstStr.***HOT\_SALE*** + **":"** + JWTUtil.*getStoreId*() + **":"** + DateFormatUtils.*format*(**new** Date(), **"yyyyMMdd"**);  
  
 Set<String> keySet = **new** HashSet<>();  
 **switch** (range) {  
 **case** ConstStr.***QUERY\_RANGE\_TODAY***:  
 projectIdTop10 = **redisTemplate**.opsForZSet().reverseRangeWithScores(zkey\_today, 0, 9);  
 **break**;  
 **case** ConstStr.***QUERY\_RANGE\_THREEDAYS***:  
 String zkey\_three\_days = ConstStr.***HOT\_SALE*** + **":"** + JWTUtil.*getStoreId*() + **":"** + **"last\_threeDays"**;  
 keySet.clear();  
 *// 生成前两天的keys* **for** (**int** i = 1; i <= 2; i++) {  
 String oneKey =  
 ConstStr.***HOT\_SALE*** + **":"** + JWTUtil.*getStoreId*() + **":"** + DateFormatUtils.*format*(DateUtils.*addDays*(**new** Date(), -i), **"yyyyMMdd"**);  
 keySet.add(oneKey);  
 }  
 **redisTemplate**.opsForZSet().unionAndStore(zkey\_today, keySet, zkey\_three\_days);  
 projectIdTop10 = **redisTemplate**.opsForZSet().reverseRangeWithScores(zkey\_three\_days, 0, 9);  
 **break**;  
 **case** ConstStr.***QUERY\_RANGE\_WEEK***:  
 String zkey\_week = ConstStr.***HOT\_SALE*** + **":"** + JWTUtil.*getStoreId*() + **":"** + **"last\_week"**;  
 keySet.clear();  
 *// 生成前6天* **for** (**int** i = 1; i <= 6; i++) {  
 String oneKey =  
 ConstStr.***HOT\_SALE*** + **":"** + JWTUtil.*getStoreId*() + **":"** + DateFormatUtils.*format*(DateUtils.*addDays*(**new** Date(), -i), **"yyyyMMdd"**);  
 keySet.add(oneKey);  
 }  
 **redisTemplate**.opsForZSet().unionAndStore(zkey\_today, keySet, zkey\_week);  
 projectIdTop10 = **redisTemplate**.opsForZSet().reverseRangeWithScores(zkey\_week, 0, 9);  
 **break**;  
 **case** ConstStr.***QUERY\_RANGE\_MONTH***:  
 String zkey\_month = ConstStr.***HOT\_SALE*** + **":"** + JWTUtil.*getStoreId*() + **":"** + **"last\_month"**;  
 keySet.clear();  
  
 **for** (**int** i = 1; i <= 29; i++) {  
 String oneKey =  
 ConstStr.***HOT\_SALE*** + **":"** + JWTUtil.*getStoreId*() + **":"** + DateFormatUtils.*format*(DateUtils.*addDays*(**new** Date(), -i), **"yyyyMMdd"**);  
 keySet.add(oneKey);  
 }  
 **redisTemplate**.opsForZSet().unionAndStore(zkey\_today, keySet, zkey\_month);  
 projectIdTop10 = **redisTemplate**.opsForZSet().reverseRangeWithScores(zkey\_month, 0, 9);  
 **break**;  
 **default**:  
 **throw new** IllegalException(**"range参数错误"**);  
 }  
  
 List<SalesVolumeVo> salesVolumeVos = projectIdTop10.stream().map(p -> {  
 Product product;  
 **try** {  
 product = **productService**.getProduct(Long.*valueOf*(p.getValue()));  
 SalesVolumeVo salesVolumeVo = **new** SalesVolumeVo();  
 BeanUtils.*copyProperties*(product, salesVolumeVo);  
 salesVolumeVo.setSalesVolume(p.getScore().intValue());  
 **return** salesVolumeVo;  
 } **catch** (IllegalException e) {  
 ***log***.debug(e.getMessage());  
 **return null**;  
 }  
 }).collect(Collectors.*toList*());  
  
 salesVolumeVos.removeIf(Objects::*isNull*);  
  
 **return** salesVolumeVos;  
}

**5.软件测试**

5.1 iOSapp与安卓apk测试详情

5.1.1欢迎页面 图12安卓欢迎页，图13ios欢迎页

图12安卓欢迎页 图13 ios欢迎页

5.1.2销售页面 图14安卓销售页，图15 ios销售页

图14安卓销售页 图15 ios销售页

5.1.3统计页面 图16安卓统计页，图17 ios统计页

图16安卓统计页 图17 ios统计页

5.2手机浏览器

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 最新 | 最新 | 9.1+ | 最新 |
| 正常 | 正常 | 正常 | 正常 |

**6.软件完成后的总结汇报**

此次项目采用前后端分离进行开发，后端使用基于SpringBoot的移动端云存储的设计与实现。在拟定课题时，我们小组考虑到现实的需求进行设计，因为后端所用到的SpringBoot技术是在以前所有项目中没有使用过也没有接触过的开发技术，所以一开始学习SpringBoot时感到无从下手，在一步步深入学习中对其有了基本的了解。可以设计出简单的页面。这些页面中的样式的使用与以前使用过的后端技术没有什么区别，主要是标签的使用有很多不同，因此需要了解SpringBoot中应该使用什么样的标签。在项目进行过程中最困难的地方在于实现逻辑代码与后台进行对接过程，这期间老师以及我的小组成员对我有很大的帮助。通过这个项目我对SpringBoot有了更深的了解，对今后的项目设计有很大的帮助。