体系结构设计文档

1 引言

1.1 编制目的

本报告详细完成对灯具进销存管理系统的概要设计，从而达到指导详细设计和开发的目的，同时实现和测试人员以及用户的沟通。 本报告面向开发人员、测试人员和用户编写，是了解系统的导航

1.2 词汇表

词汇名称	词汇含义	备注
ERP	进销存管理系统	无

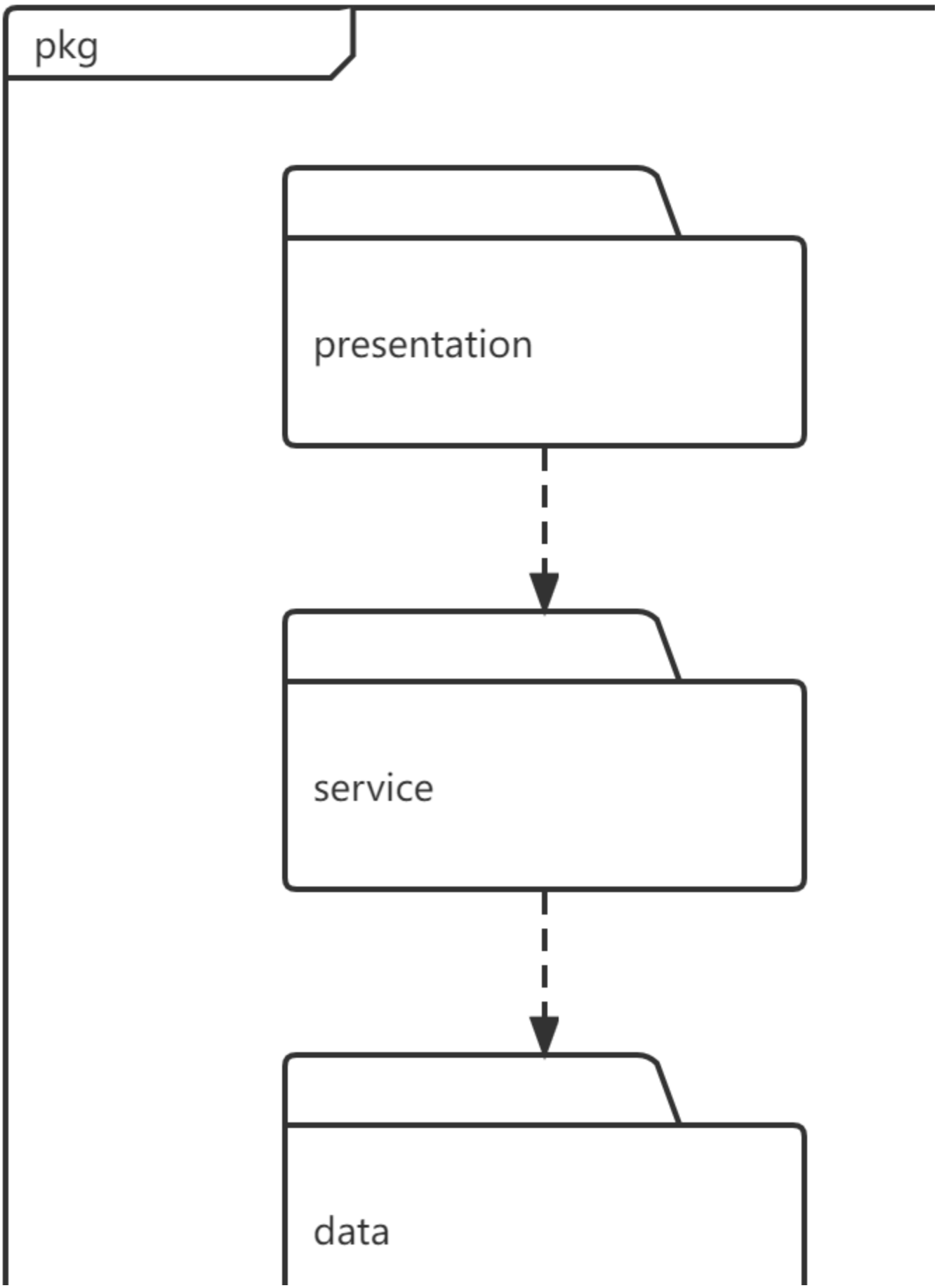
1.3 参考资料

2 产品描述

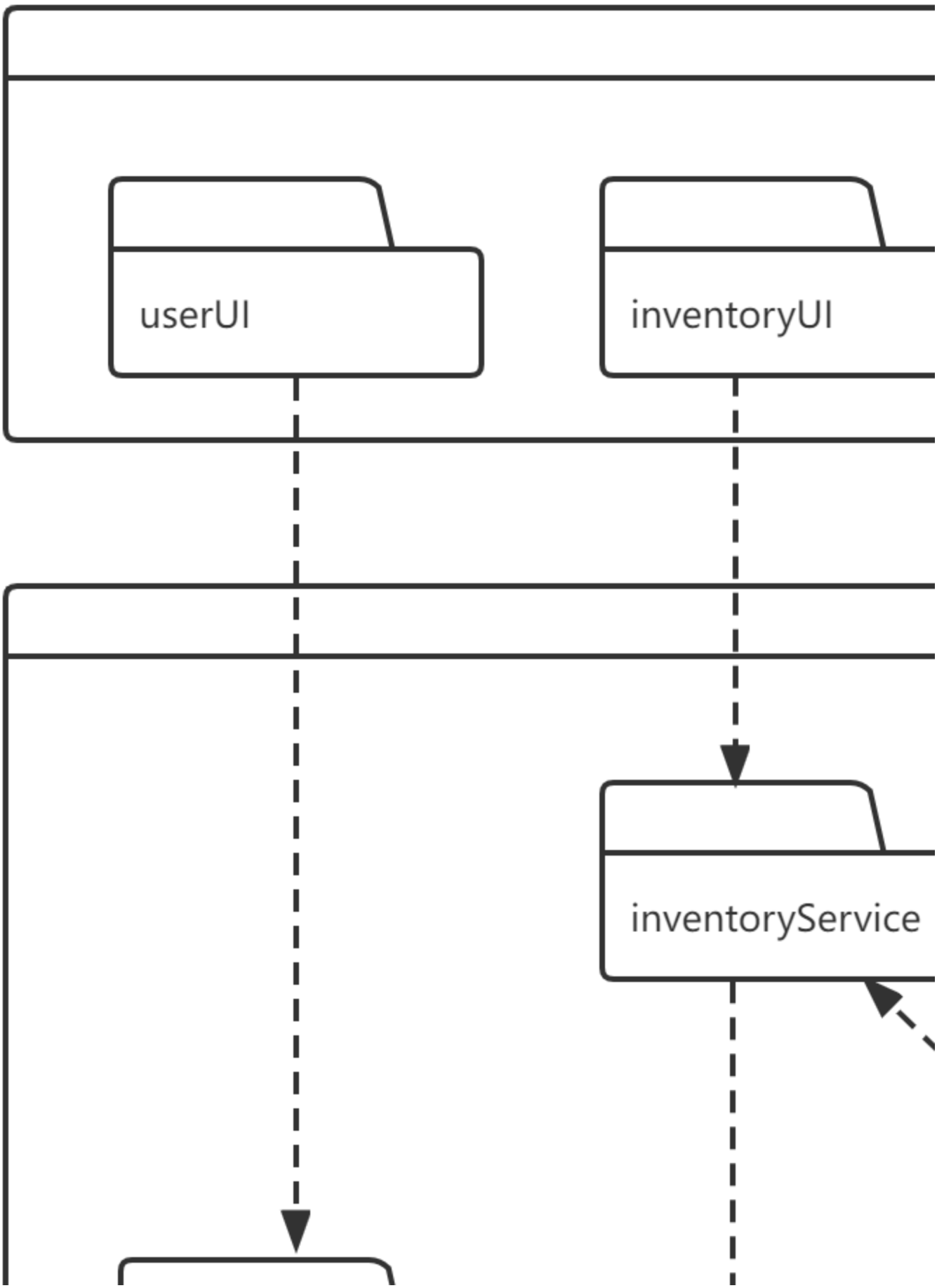
参考灯具进销存系统用例文档和灯具进销存需求规格说明文档中对产品的概括描述。

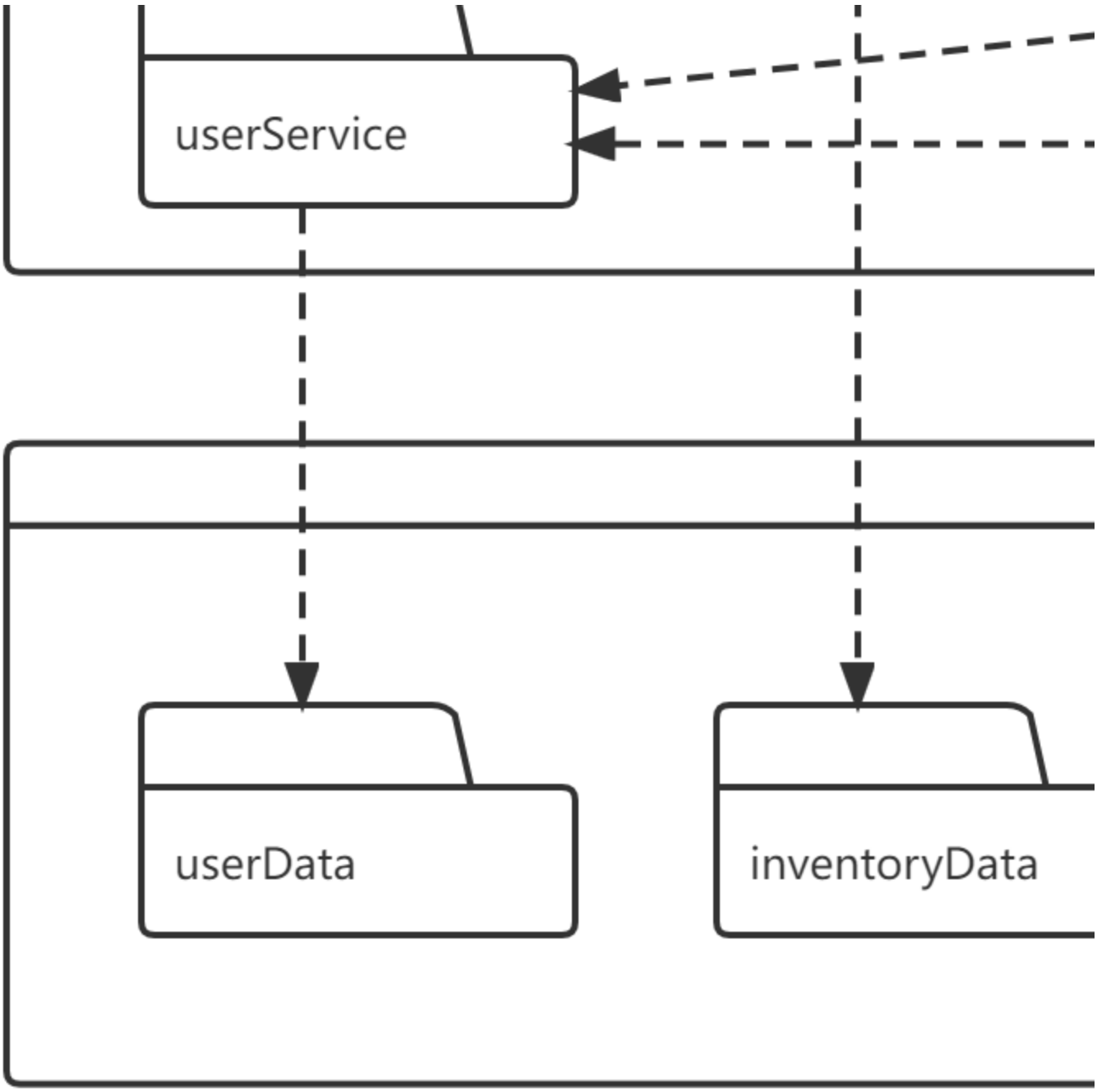
3 逻辑视角

灯具进销存管理系统中，选择了分层体系结构风格，将系统分为三层（展示层、业务逻辑层、数据层）能够很好地示意整个高层抽象。展示层包含GUI界面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。分层体系结构的逻辑视角和逻辑设计方案如下图所示。


pkg
presentation
service
data

逻辑视角图


userUI
inventoryUI
inventoryService



//逻辑设计方案图

4 组合视角

4.1 开发包图

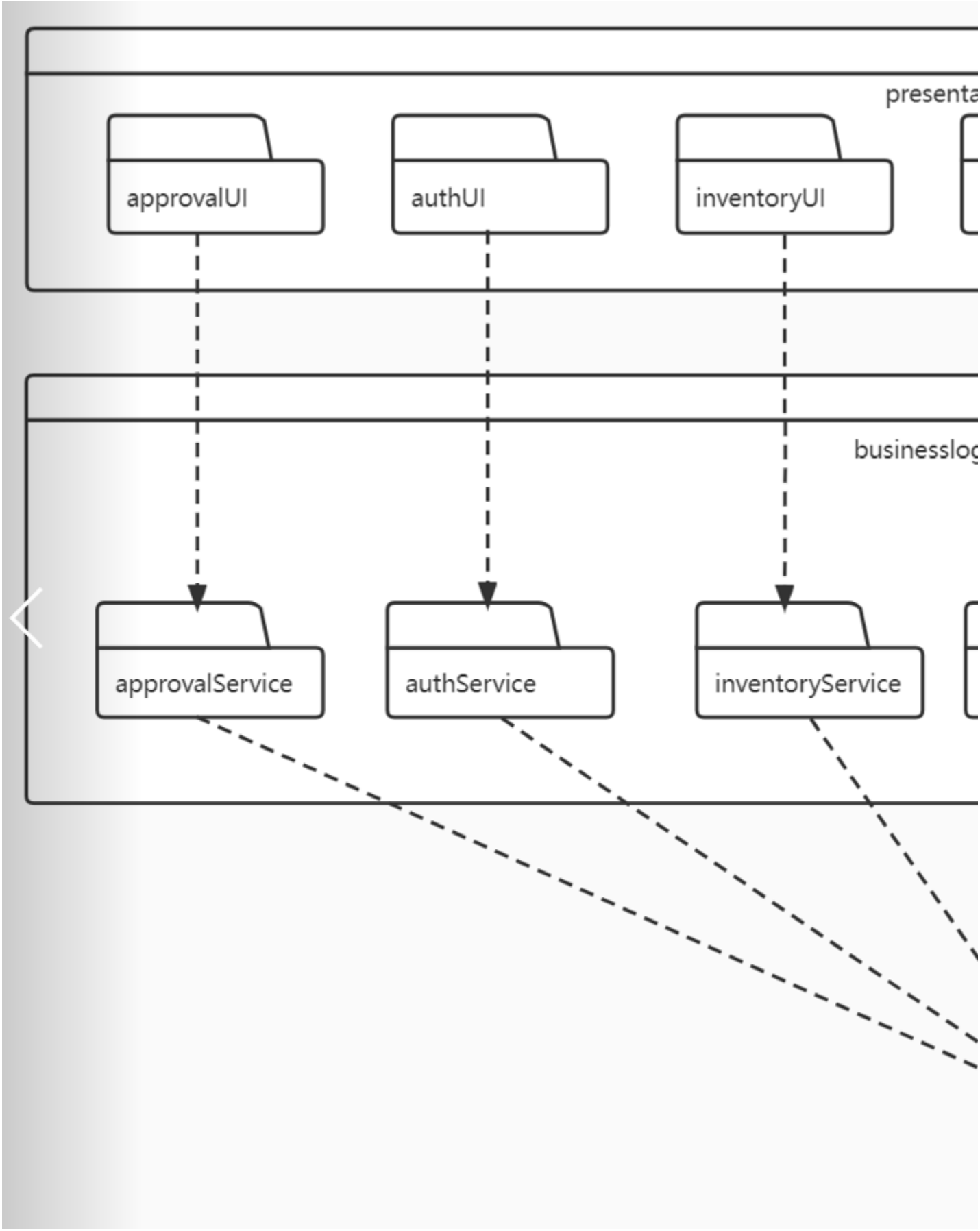
灯具进销存系统的具体开发包图如下表所示

开发（物理）包	依赖的其他开发包
CustomerController	vo

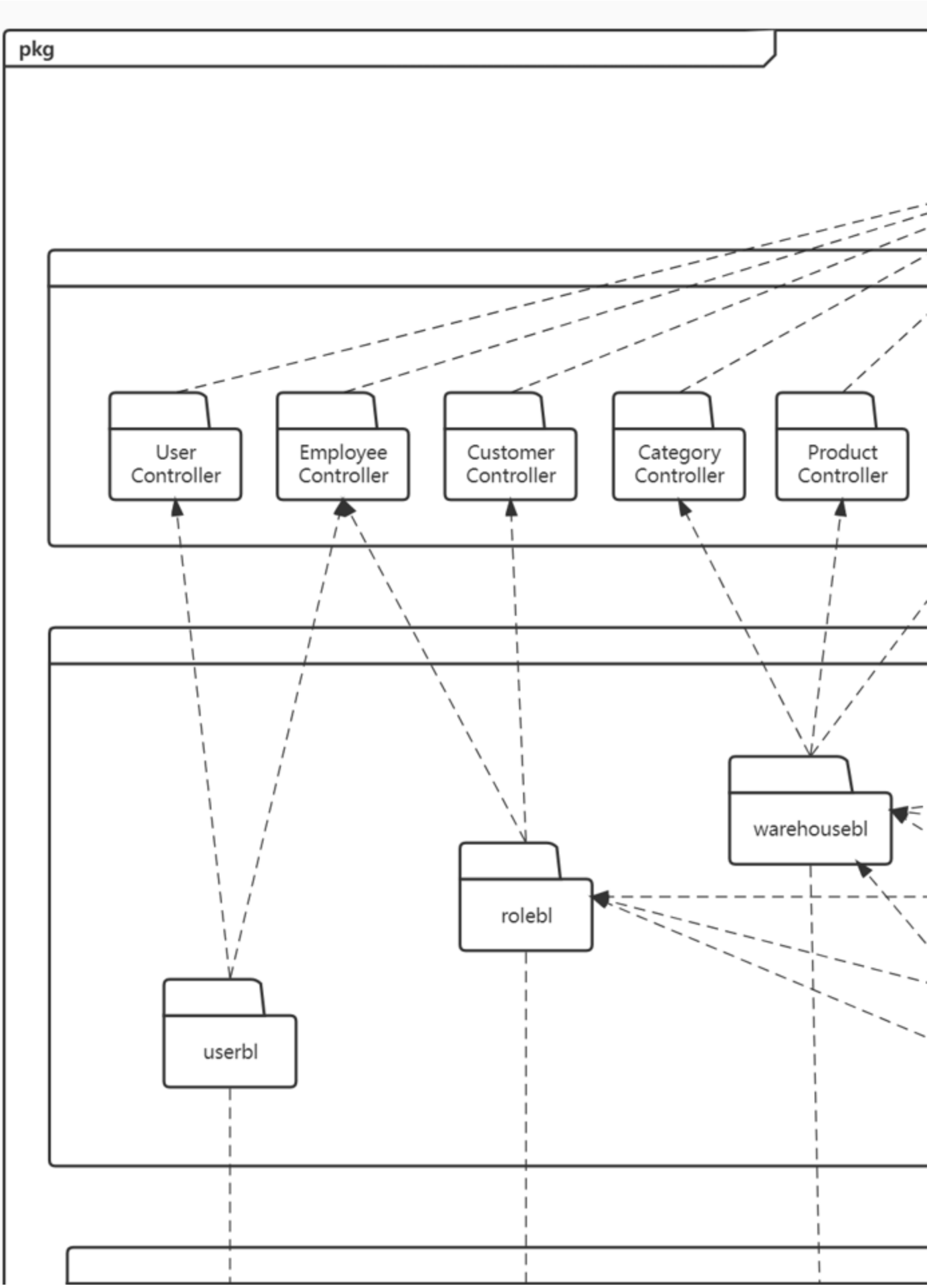
开发（物理）包	依赖的其他开发包
CategoryController	vo
EmployeeController	vo
financialbl	vo, FinancialController, utilitybl, rolebl, warehousebl, salebl, warehousebl
FinancialController	vo
financialdao	po
financialmapper	po, financialdao
ProductController	vo
purchasebl	vo, PurchaseController, PurchaseReturnController, rolebl
PurchaseController	vo
purchasedao	po
purchasemapper	po, purchasemapper
PromotionController	vo
PurchaseReturnController	vo
rolebl	vo, EmployeeController, CustomerController, purchasebl
roledao	po
rolemapper	po, roledao
salebl	vo, SaleController, SaleReturnController, PromotionController
SaleController	vo
saledao	po
salemapper	po, saledao
SaleReturnController	vo
userbl	vo, UserController, EmployeeController
UserController	vo
userdao	po

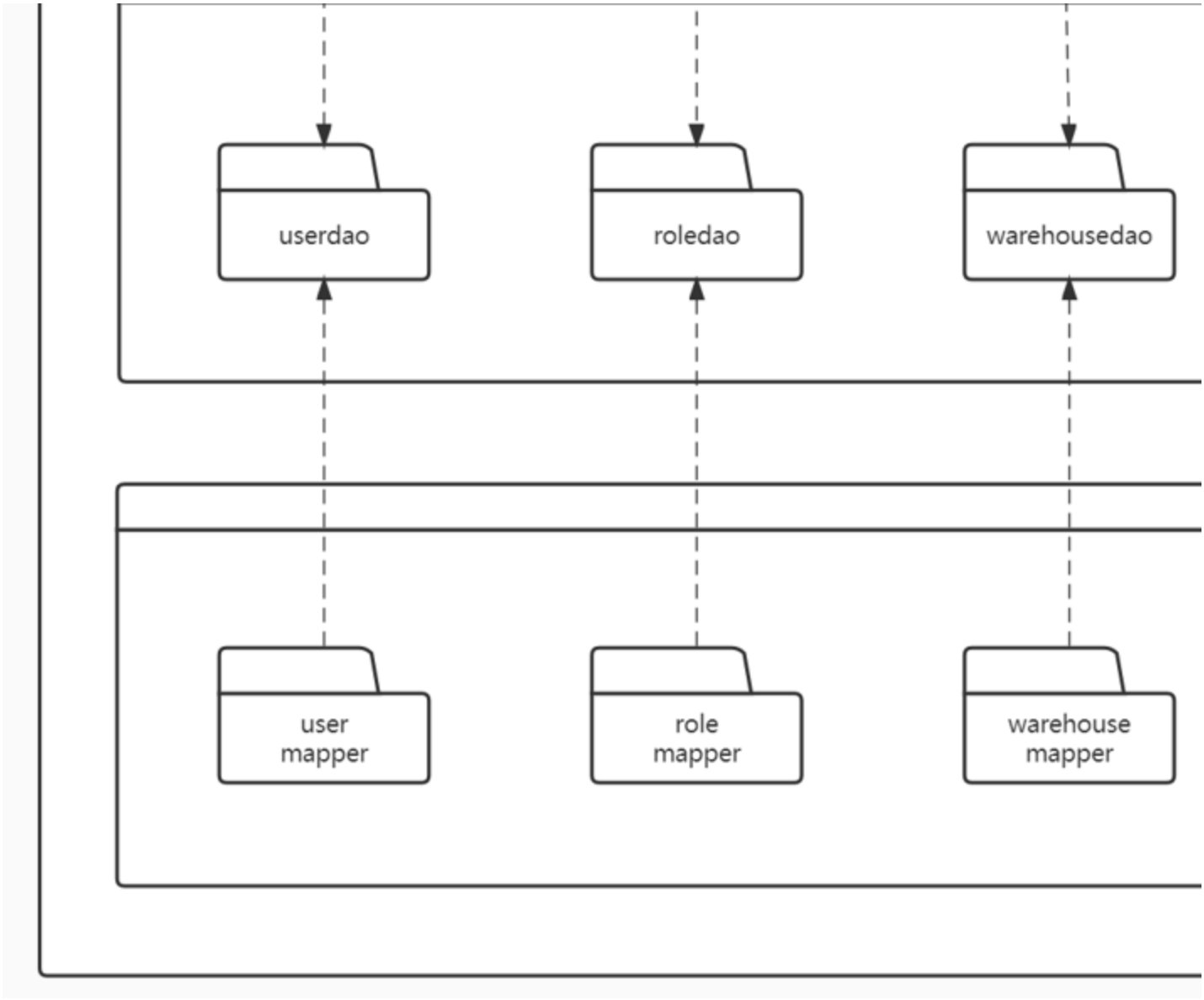
开发（物理）包	依赖的其他开发包
usermapper	po, userdao
utilitybl	vo
warehousebl	vo, CategoryController, ProductController, WareHouseController, utilitybl
WareHouseController	vo
warehousedao	po
warehousemapper	po, warehousedao
vo	
po	

下图为灯具进销存系统的客户端开发包图



下图为灯具进销存系统的服务器端开发包图

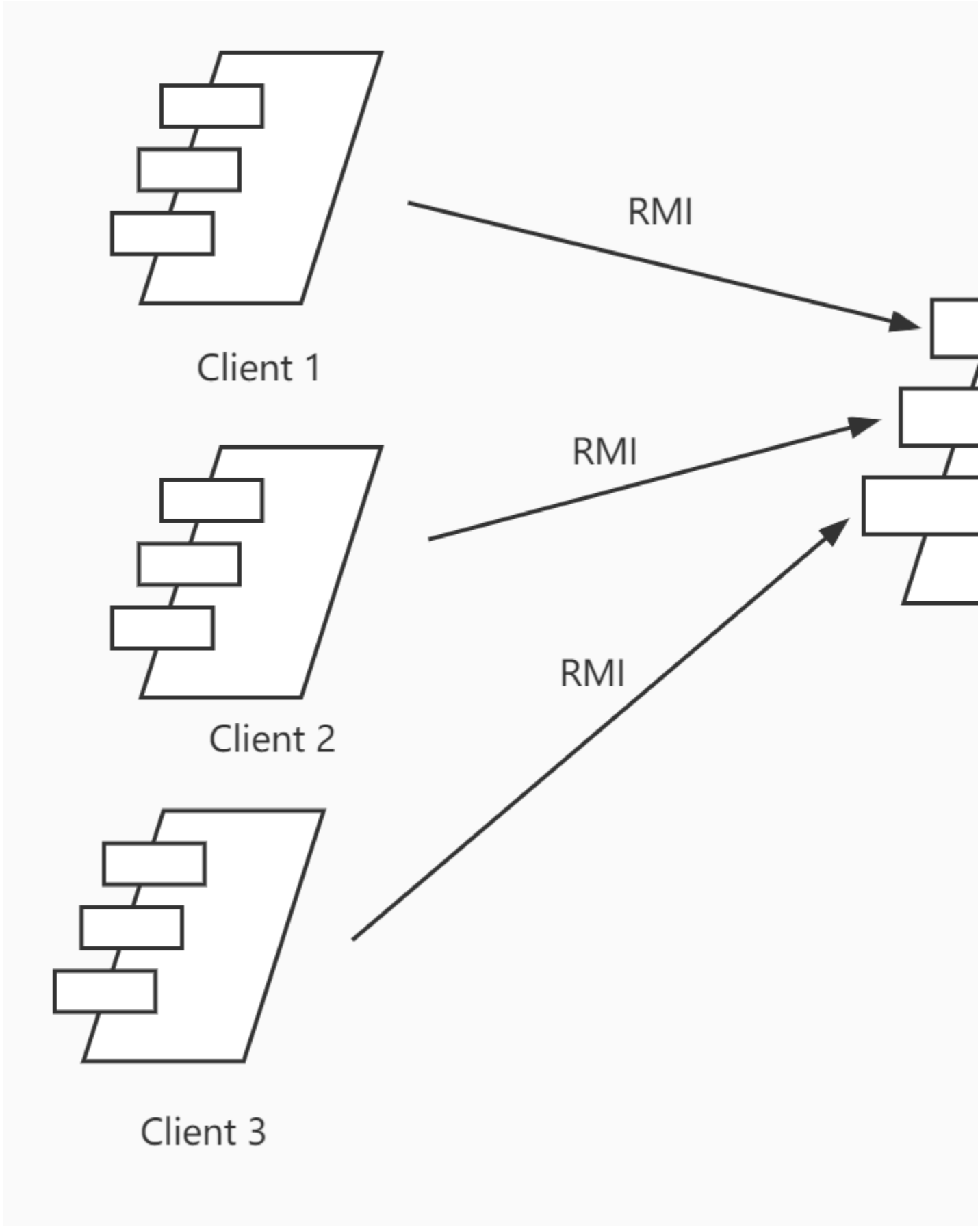

pkg
User Controller
Employee Controller
Customer Controller
Category Controller
Product Controller
warehousebl
rolebl
userbl



4.2 运行时进程

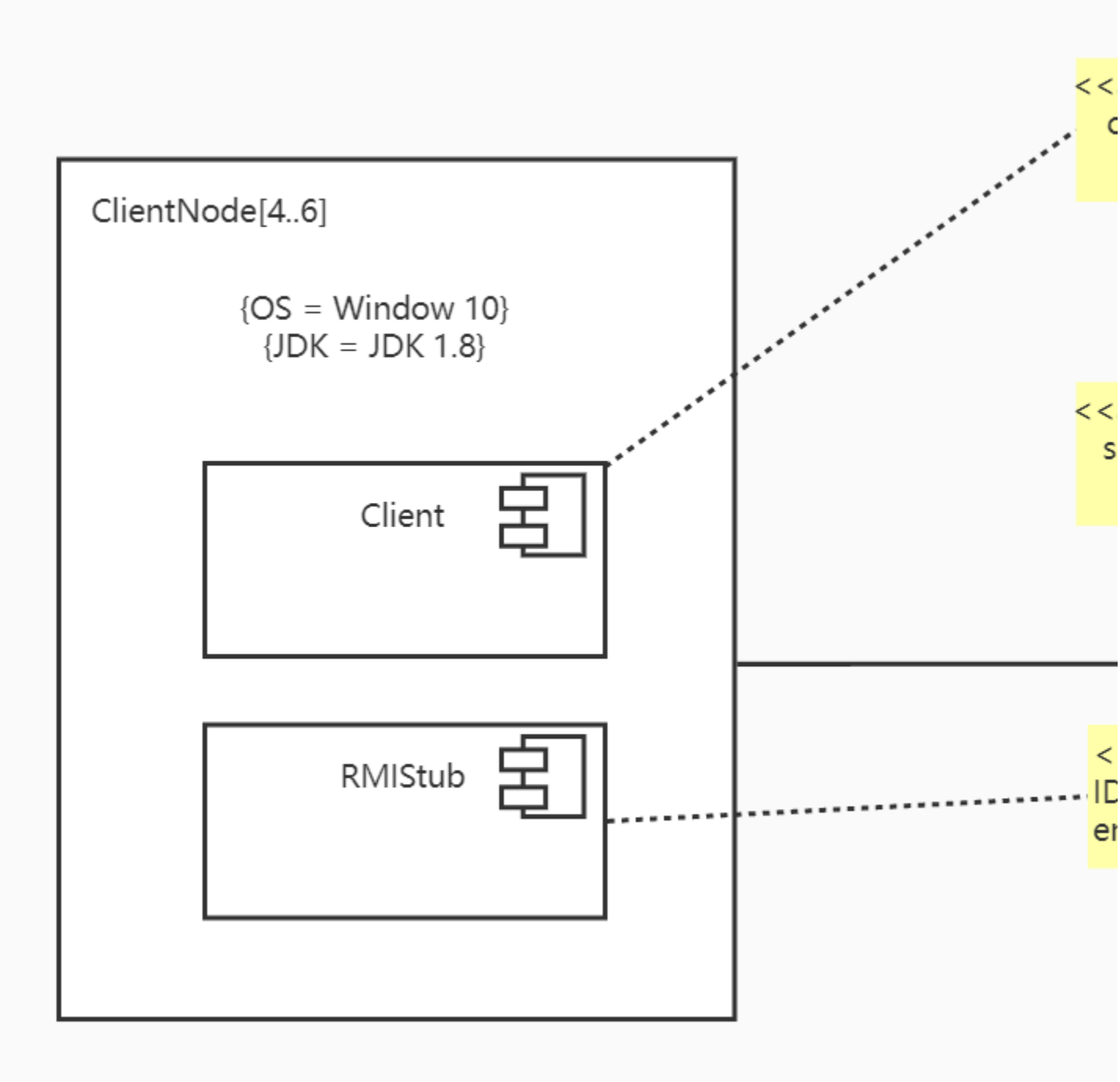
在灯具进销存管理系统中，会同时存在多个客户端进程和一个服务器端进程，其进程图如下图所示。结合部署图，客户端是在客户端机器上运行，服务器端进程是在服务器端机器上运行。

运行时进程图



4.3 物理部署

灯具进销存管理系统中的客户端构件是放在客户端机器上，服务器端构件是放在服务器端机器上的。在客户端节点上，还要部署Java RMI构件。由于Java RMI构件属于JDK1.8的一部分。所以，在系统JDK环境已经设置好的前提下，不需要再独立部署。部署图如下图所示。

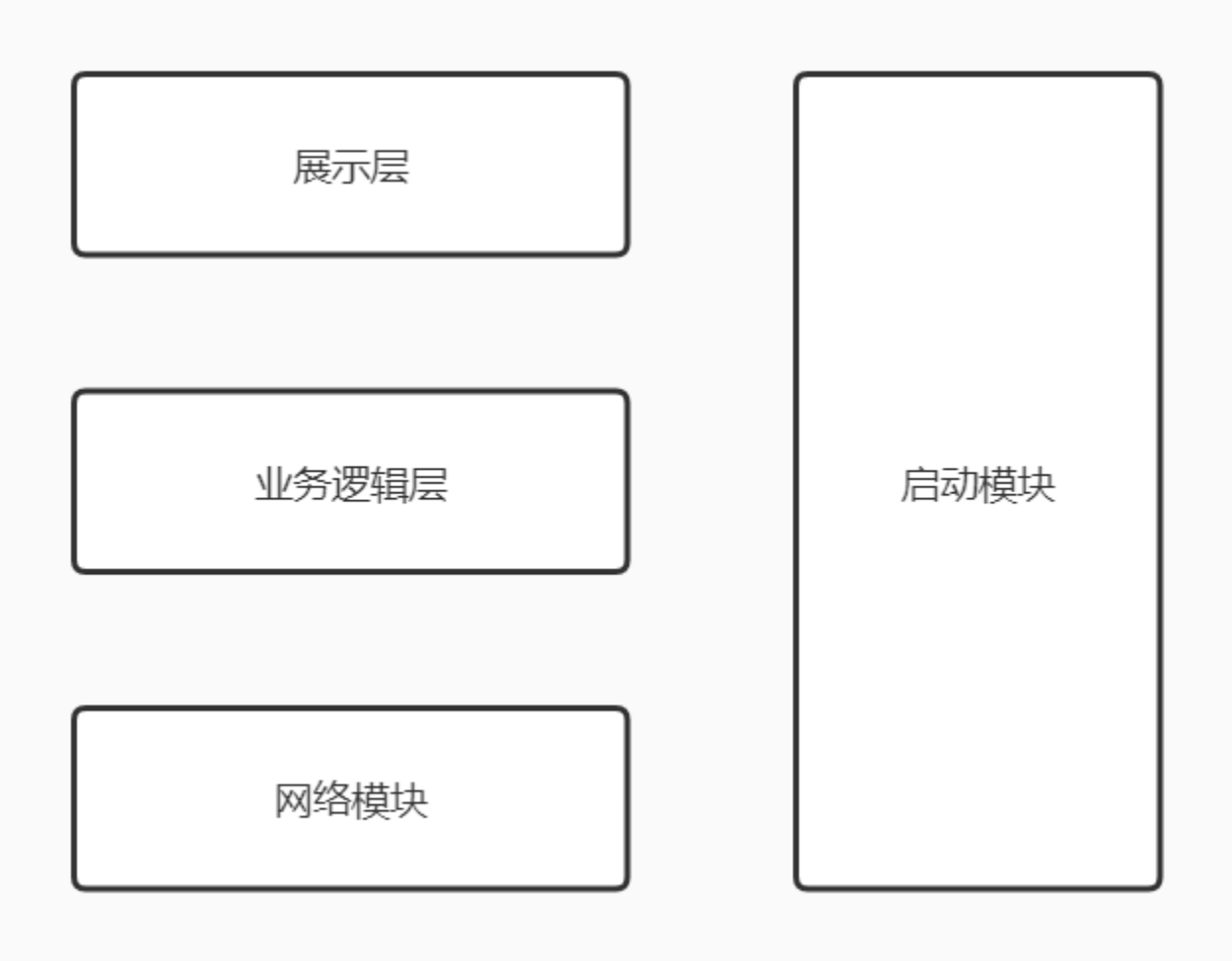


物理部署图

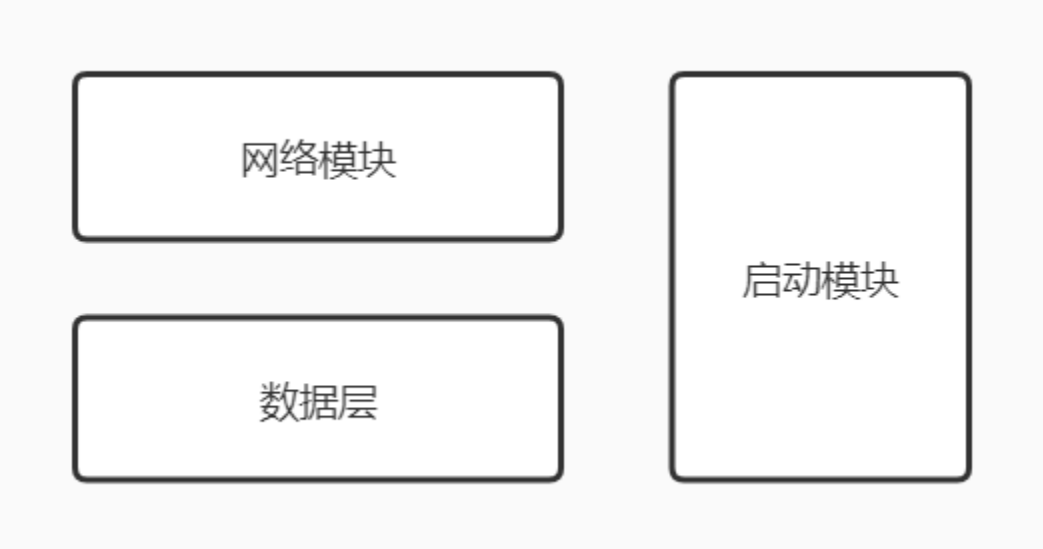
5 接口视角

5.1 模块的职责

下图为客户端的模块视图



下图为服务器端的模块视图



下表为客户端各层的职责

层	职责
启动模块	负责初始化网络通信机制，启动用户界面
展示层	基于窗口化界面的灯具进销存管理系统客户端用户界面
业务逻辑层	对用户界面的输入进行响应并进行业务逻辑处理
客户端网络模块	利用Java RMI机制查找RMI服务

下表为服务器端各层的职责

层	职责
启动模块	负责初始化网络通信机制，启动后端服务器
数据层	负责数据的持久化和访问接口
业务逻辑层	处理请求，对数据进行操作或给予适当相应
网络模块	实现基于HTTP协议的数据传输

每一层只使用下方直接接触的层。层与层之间仅仅通过接口的调用来完成。层之间的接口如下表所示：

接口	服务调用方	服务提供方
approvalblservice commodityblserv		
ice authblservice financeblservice inventoryblservice HRblservice commodityblservice purchaseblservice saleblservice sheetblservice	客户端展示层	客户端业务逻辑层
categorydataservice commoditydataservice accountdataservice customerdataservice billdataservice initdataservice promotiondataservice	客户端业务逻辑层	服务端数据层

接口	服务调用方	服务提供方
userdataservice logdataservice		

5.2 用户界面层的分解

根据需求，系统存在26个用户界面：进销存系统登录界面，系统管理人员主界面，库存管理人员主界面，进货销售人员主界面，财务人员主界面，总经理主界面，用户账户管理界面，商品分类管理界面，商品管理界面，库存管理界面，库存调整界面，客户管理界面，制定进货单界面，制定进货退货单界面，制定销售单界面，制定销售退货单界面，银行账户管理界面，制定收付款单界面，制定现金费用单界面，期初建账界面，查看日志界面，查看销售明细表界面，查看经营状况表界面，查看经营历程表界面，审批单据界面，制定促销策略界面。

5.2.1 用户界面层的职责

如表5-2-1所示为用户界面层模块的职责

```
<tr>
    <td><strong><div class="text" style=" text-align:center;">模块</div></strong></td>
    <td><strong><div class="text" style=" text-align:center;">职责</div></strong></td>
</tr>
<tr>
    <td>MainFrame</td>
    <td>界面Frame，负责界面的显示和界面的跳转</td>
</tr>
```

表5-2-1用户界面层模块的职责

5.2.2 用户界面层模块的接口规范

用户界面层模块的接口规范如表5-2-2-1所示

```
<tr>
    <td rowspan="3">MainFrame</td>
    <td>语法</td>
    <td>init(args:String[])</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>无</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>显示Frame以及LoginPanel</td>
</tr>
```

用户界面层需要的服务接口如表5-2-2-2所示

表5-2-2-1用户界面层模块的接口规范

```
<tr>
    <td>UserBLService.login(String id, String password)</td>
    <td>登录界面的业务逻辑接口</td>
</tr><tr>
    <td>*BLService</td>
    <td>每个界面都有一个相应的业务逻辑接口</td>
</tr>
```

表5-2-2-2用户界面层模块需要的服务接口

服务名	服务

5.2.3 用户界面模块的设计原理

用户界面利用java的Swing和AWT库来实现

5.3 业务逻辑层的分解

业务逻辑层包括多个针对界面的业务逻辑处理对象。

5.3.1 业务逻辑层模块的职责

```
<tr>
    <td><strong><div class="text" style=" text-align:center;">模块</div></strong></td>
    <td><strong><div class="text" style=" text-align:center;">职责</div></strong></td>
</tr>
<tr>
    <td>financialbl</td>
    <td>负责财务相关表单的增删等操作</td>
</tr>
<tr>
    <td>purchasebl</td>
    <td>负责进货购买时所需要的服务</td>
</tr>
<tr>
    <td>rolebl</td>
    <td>负责对员工以及客户角色管理的服务</td>
</tr>
<tr>
    <td>salebl</td>
    <td>负责销售模块所需要的服务</td>
</tr>
<tr>
    <td>userbl</td>
    <td>负责系统用户管理界面所需要的服务</td>
```

```
</tr>
<tr>
    <td>utilitybl</td>
    <td>负责其他服务执行时提供所需工具的服务</td>
</tr>
<tr>
    <td>warehousebl</td>
    <td>负责仓库管理所需要的服务</td>
</tr>
```

表8 业务逻辑层模块的职责

5.3.2 业务逻辑层模块的接口规范

```
<tr>
    <td colspan="3"><strong><div class="text" style=" text-align:center;">提供的服务（供接口
</tr>
<tr>
    <td rowspan="3">Account.addBankAccount</td>
    <td>语法</td>
    <td>void addBankAccount(BankAccountVO bankAccountVO)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>操作人为财务人员或最高权限</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>系统保存新帐户信息</td>
</tr>
<tr>
    <td rowspan="3">Account.deleteBankAccount</td>
    <td>语法</td>
    <td> void deleteBankAccount(String name)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>操作人为财务人员或最高权限</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>依据账户名称删除银行账户</td>
</tr>
<tr>
    <td rowspan="3">Account.updateBankAccount</td>
    <td>语法</td>
    <td>void updateBankAccount(BankAccountVO bankAccountVO)</td>
</tr>
<tr>
    <td>前置条件</td>
```

```
    <td>操作人为财务人员或最高权限</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>依据账户ID持久化更新银行账户</td>
</tr>
<tr>
    <td rowspan="3">Account.selectBankAccount</td>
    <td>语法</td>
    <td>List selectBankAccount(String name)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>操作人为财务人员或最高权限</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>根据名称模糊查找并返回所有符合条件的账户</td>
</tr>
<tr>
    <td rowspan="3">Account.showAllBankAccount</td>
    <td>语法</td>
    <td>List showAllBankAccount()</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>操作人为财务人员或最高权限</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回所有账户信息</td>
</tr>
<tr>
    <td rowspan="3">BusinessHistory.showBusinessHistoryBySomeConditions</td>
    <td>语法</td>
    <td>List showBusinessHistoryBySomeConditions(BusinessHistorySheetSearchCriteriaVO searc
</tr>
<tr>
    <td>前置条件</td>
    <td>已在经营过程中产生各类单据</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回所有符合条件（时间区间、单据类型、客户、业务员）的单据</td>
</tr>
<tr>
    <td rowspan="3">CashExpense.makeCashExpenseSheet</td>
    <td>语法</td>
    <td>void makeCashExpenseSheet(UserVO userVO, CashExpenseSheetVO cashExpenseSheetVO)</td
</tr>
<tr>
    <td>前置条件</td>
    <td>操作人为财务人员或最高权限</td>
```

```
</tr>
<tr>
    <td>后置条件</td>
    <td>系统生成并持久化现金费用单</td>
</tr>
<tr>
    <td rowspan="3">CashExpense.findCashExpenseSheetIdBySomeConditions</td>
    <td>语法</td>
    <td>List findCashExpenseSheetIdBySomeConditions(BusinessHistorySheetSearchCriteriaPO bu
</tr>
<tr>
    <td>前置条件</td>
    <td>操作人为财务人员或最高权限</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回所有符合条件（时间区间、单据类型、客户、业务员）的现金费用单</td>
</tr>
<tr>
    <td rowspan="3">OpeningAccount.openingAccountEstablishment</td>
    <td>语法</td>
    <td>OpeningAccountVO openingAccountEstablishment()</td>
</tr>
<tr>
    <td>前置条件</td>
    <td></td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回期初建账信息（所有商品分类、所有商品、客户信息、银行账户信息）</td>
</tr>
<tr>
    <td rowspan="3">OperationStatement.showStatementOfOperation</td>
    <td>语法</td>
    <td>OperatingConditionsSheetVO showStatementOfOperation(String beginDateStr, String end
</tr>
<tr>
    <td>前置条件</td>
    <td>传入的时间字符串符合“yyyy-MM-dd HH:mm:ss”格式，且不为空</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回指定时间区间内的经营情况表</td>
</tr>
<tr>
    <td rowspan="3">PaymentOrder.makePaymentOrder</td>
    <td>语法</td>
    <td>void makePaymentOrder(UserVO userVO, PaymentOrderVO paymentOrderVO)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>操作人为财务人员或最高权限</td>
</tr>
```

```
<tr>
    <td>后置条件</td>
    <td>系统生成并持久化付款单</td>
</tr>
<tr>
    <td rowspan="3">PaymentOrder.findPaymentOrderSheetIdBySomeConditions</td>
    <td>语法</td>
    <td>List findPaymentOrderSheetIdBySomeConditions(BusinessHistorySheetSearchCriteriaPO b
</tr>
<tr>
    <td>前置条件</td>
    <td>操作人为财务人员或最高权限</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回所有符合条件（时间区间、单据类型、客户、业务员）的付款单</td>
</tr>
<tr>
    <td rowspan="3">Payroll.makePayroll</td>
    <td>语法</td>
    <td>void makePayroll(UserVO userVO, PayrollVO payrollVO)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>操作人为财务人员或最高权限</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>系统生成并持久化工资单</td>
</tr>
<tr>
    <td rowspan="3">Payroll.findPayrollSheetIdBySomeConditions</td>
    <td>语法</td>
    <td>List findPayrollSheetIdBySomeConditions(BusinessHistorySheetSearchCriteriaPO busine
</tr>
<tr>
    <td>前置条件</td>
    <td>操作人为财务人员或最高权限</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回所有符合条件（时间区间、单据类型、客户、业务员）的工资单</td>
</tr>
<tr>
    <td rowspan="3">Receipt.makeReceipt</td>
    <td>语法</td>
    <td>void makeReceipt(UserVO userVO, ReceiptVO receiptVO)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>操作人为财务人员或最高权限</td>
</tr>
<tr>
```

```
    <td>后置条件</td>
    <td>系统生成并持久化收款单</td>
</tr>
<tr>
    <td rowspan="3">Receipt.findReceiptSheetIdBySomeConditions</td>
    <td>语法</td>
    <td>List findReceiptSheetIdBySomeConditions(BusinessHistorySheetSearchCriteriaPO busine
</tr>
<tr>
    <td>前置条件</td>
    <td>操作人为财务人员或最高权限</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回所有符合条件（时间区间、单据类型、客户、业务员）的收款单</td>
</tr>
<tr>
    <td rowspan="3">SaleDetails.showSaleDetailsBySomeConditions</td>
    <td>语法</td>
    <td>List showSaleDetailsBySomeConditions(SaleDetailsSheetSearchCriteriaVO searchCriteri
</tr>
<tr>
    <td>前置条件</td>
    <td>操作人为财务人员或最高权限</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回所有符合条件（时间区间、单据类型、客户、业务员）的销售明细表</td>
</tr>
```

表9 financialbl模块的接口规范

需要的接口（需接口）	
服务名	服务
BankAccountDao.insertBankAccount(BankAccountPO bankAccountPO)	持久化一条银行账户信息
BankAccountDao.deleteBankAccountByName(String name)	根据名称删除一条银行账户信息
BankAccountDao.updateBankAccount(BankAccountPO bankAccountPO)	更新一条银行账户信息
BankAccountDao.selectBankAccountByName(String name)	查询并返回名称模糊匹配到的所有银行账户信息
BankAccountDao.selectAll()	查询并返回所有银行账户信息

SaleService.findSheetIdBySomeConditions(SearchCriteriaPO searchCriteriaPO)	返回所有符合条件（时间区间、单据类型、客户、业务员）的表单
saleReturnsService.findSheetIdBySomeConditions (SearchCriteriaPO searchCriteriaPO)	返回所有符合条件（时间区间、单据类型、客户、业务员）的表单
purchaseService.findSheetIdBySomeConditions (SearchCriteriaPO searchCriteriaPO)	返回所有符合条件（时间区间、单据类型、客户、业务员）的表单
purchaseReturnsService.findSheetIdBySomeConditions (SearchCriteriaPO searchCriteriaPO)	返回所有符合条件（时间区间、单据类型、客户、业务员）的表单
receiptService.findReceiptSheetIdBySomeConditions (SearchCriteriaPO searchCriteriaPO)	返回所有符合条件（时间区间、单据类型、客户、业务员）的表单
paymentOrderService. findPaymentOrderSheetIdBySomeConditions(SearchCriteriaPO searchCriteriaPO)	返回所有符合条件（时间区间、单据类型、客户、业务员）的表单
cashExpenseService. findCashExpenseSheetIdBySomeConditions(SearchCriteriaPO searchCriteriaPO)	返回所有符合条件（时间区间、单据类型、客户、业务员）的表单
payrollService.findPayrollSheetIdBySomeConditions (SearchCriteriaPO searchCriteriaPO)	返回所有符合条件（时间区间、单据类型、客户、业务员）的表单
UserDataService.findByKeyWord(String keyWord)	根据字段名和值进行查找多个持久化对象
TermSheetDao.saveBatchSheetContent(List termSheetPOS)	插入多个持久化对象
CashExpenseSheetDao.saveSheet(CashExpenseSheetPO cashExpenseSheetPO)	插入单一持久化对象
CashExpenseSheetDao.findSheetIdBySomeConditions (SearchCriteriaPO searchCriteriaPO)	返回所有符合条件（时间区间、单据类型、客户、业务员）的单据
SaleService.findSheetIdByTime(String beginDate, String endDate)	返回所有符合条件的持久化对象
TransferRecordDao.saveBatchSheetContent(List transferRecordPOS)	插入多个持久化对象
PaymentOrderDao.saveSheet(PaymentOrderPO paymentOrderPO)	插入单一持久化对象
PaymentOrderDao.findSheetIdBySomeConditions (SearchCriteriaPO searchCriteriaPO)	返回所有符合条件（时间区间、单据类型、客户、业务员）的单据

PayrollDao.saveSheet(PayrollPO payrollPO)	插入单一持久化对象
PayrollDao.getLatestSheet()	返回上一个工资单
IdGenerator.generateSheetId(String id, String prefix)	根据上一份单据生成
ReceiptDao.getLatestSheet()	返回上一个收款单
TransferRecordDao.saveBatchSheetContent (TransferRecordPOS transferRecordPOS)	插入多个持久化对象
ReceiptDao.saveSheet(ReceiptPO receiptPO)	插入单一持久化对象
ReceiptDao.findSheetIdBySomeConditions(SearchCriteriaPO searchCriteriaPO)	返回所有符合条件（时间区间、单据类型、客户、业务员）的单据
SaleService.findSaleSheetsBySomeConditions(SearchCriteriaPO searchCriteriaPO)	返回所有符合条件（时间区间、单据类型、客户、业务员）的单据
SaleReturnsService.findSaleSheetsBySomeConditions (SearchCriteriaPO searchCriteriaPO)	返回所有符合条件（时间区间、单据类型、客户、业务员）的单据

```
<tr>
    <td colspan="3"><strong><div class="text" style=" text-align:center;">提供的服务（供接口
</tr>
<tr>
    <td rowspan="3">PurchaseService.makePurchaseSheet</td>
    <td>语法</td>
    <td>void makePurchaseSheet(UserVO userVO, PurchaseSheetVO purchaseSheetVO)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">PurchaseService.getPurchaseSheetByState</td>
    <td>语法</td>
    <td>List getPurchaseSheetByState(PurchaseSheetState state)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
```

```
</tr>
<tr>
    <td rowspan="3">PurchaseService.approval</td>
    <td>语法</td>
    <td>void approval(String purchaseSheetId, PurchaseSheetState state)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">PurchaseService.getPurchaseSheetById</td>
    <td>语法</td>
    <td>PurchaseSheetVO getPurchaseSheetById(String purchaseSheetId)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">PurchaseService.findSheetIdByOperator</td>
    <td>语法</td>
    <td>List findSheetIdByOperator(String operator)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">PurchaseService.findSheetIdByTime</td>
    <td>语法</td>
    <td>List findSheetIdByTime(Date beginDate, Date endDate)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
```

```
<tr>
    <td rowspan="3">PurchaseService.findSheetIdBySomeConditions</td>
    <td>语法</td>
    <td>List findSheetIdBySomeConditions(BusinessHistorySheetSearchCriteriaPO businessHisto
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">PurchaseReturnsService.makePurchaseReturnsSheet</td>
    <td>语法</td>
    <td>void makePurchaseReturnsSheet(UserVO userVO, PurchaseReturnsSheetVO purchaseReturns
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">PurchaseReturnsService.getPurchaseReturnsSheetByState</td>
    <td>语法</td>
    <td> List getPurchaseReturnsSheetByState(PurchaseReturnsSheetState state)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">PurchaseReturnsService.approval</td>
    <td>语法</td>
    <td>void approval(String purchaseReturnsSheetId, PurchaseReturnsSheetState state)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
```

```
    <td rowspan="3">PurchaseReturnsService.findSheetIdByOperator</td>
    <td>语法</td>
    <td> List findSheetIdByOperator(String operator)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">PurchaseReturnsService.findSheetIdByTime</td>
    <td>语法</td>
    <td>List findSheetIdByTime(Date beginDate, Date endDate)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">PurchaseReturnsService.findSheetIdBySomeConditions</td>
    <td>语法</td>
    <td> List findSheetIdBySomeConditions(BusinessHistorySheetSearchCriteriaPO businessHist
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">PurchaseReturnsService.getPurchaseRetuenSheetById</td>
    <td>语法</td>
    <td>PurchaseReturnsSheetVO getPurchaseRetuenSheetById(String id)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
```

表10 purchasebl模块的接口规范

需要的接口（需接口）	
服务名	服务
purchaseReturnsSheetDao.saveBatch(pContentPOList)	生成一条与之前不同的客户编号
purchaseReturnsSheetDao.save(purchaseReturnsSheetPO)	生成一条与之前不同的客户编号
purchaseSheetDao.findContentByPurchaseSheetId	生成一条与之前不同的客户编号
IdGenerator.generateSheetId(String id, String prefix)	生成一条与之前不同的客户编号
purchaseReturnsSheetDao.findAll()	生成一条与之前不同的客户编号
purchaseReturnsSheetDao.findAllByState(state)	生成一条与之前不同的客户编号
purchaseReturnsSheetDao.findContentByPurchaseReturnsSheetId(po.getId())	生成一条与之前不同的客户编号
purchaseReturnsSheetDao.updateState(purchaseReturnsSheetId, state)	生成一条与之前不同的客户编号
purchaseReturnsSheetDao.findOneById(purchaseReturnsSheetId)	生成一条与之前不同的客户编号
purchaseReturnsSheetDao.findBatchId(purchaseReturnsSheetId)	生成一条与之前不同的客户编号
purchaseReturnsSheetDao.findContentByPurchaseReturnsSheetId (purchaseReturnsSheetId)	生成一条与之前不同的客户编号
warehouseDao.findOneByPidAndBatchId(pid, batchId)	生成一条与之前不同的客户编号
warehouseDao.deductQuantity(warehousePO)	生成一条与之前不同的客户编号
productService.updateProduct(productInfoVO)	生成一条与之前不同的客户编号

PurchaseReturnsSheetDao.updateState(String purchaseReturnsSheetId, PurchaseReturnsSheetState state)	生成一条与之前不同的客户编号
purchaseReturnsSheetDao.findSheetIdByOperator(operator)	生成一条与之前不同的客户编号
purchaseReturnsSheetDao.findSheetIdByTime(beginDate, endDate)	生成一条与之前不同的客户编号
purchaseReturnsSheetDao.findSheetIdBySomeConditions (BusinessHistorySheetSearchCriteriaPO searchCriteriaPO)	生成一条与之前不同的客户编号
purchaseSheetDao.getLatest()	生成一条与之前不同的客户编号
purchaseSheetDao.saveBatch(pContentPOList)	生成一条与之前不同的客户编号
purchaseSheetDao.save(purchaseSheetPO)	生成一条与之前不同的客户编号
purchaseSheetDao.findAll()	生成一条与之前不同的客户编号
purchaseSheetDao.findAllByState(state)	生成一条与之前不同的客户编号
purchaseSheetDao.findContentByPurchaseSheetId(po.getId())	生成一条与之前不同的客户编号
purchaseSheetDao.findSheetIdByOperator(operator)	生成一条与之前不同的客户编号
purchaseSheetDao.findSheetIdByTime(beginDate, endDate)	生成一条与之前不同的客户编号
purchaseSheetDao.findSheetIdBySomeConditions (BusinessHistorySheetSearchCriteriaPO searchCriteriaPO)	生成一条与之前不同的客户编号

```
<tr>
    <td colspan="3"><strong><div class="text" style=" text-align:center;">提供的服务（供接口
</tr>
```

```
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
```

```
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">UserService.register</td>
    <td>语法</td>
    <td>void register(UserVO userVO)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">UserService.auth</td>
    <td>语法</td>
    <td>UserVO auth(String token)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
```

表11 userbl模块的接口规范

UserService.login	语法	Map login(UserVO userVO)

需要的接口（需接口）	
服务名	服务
User findByUsernameAndPassword(String username, String password)	生成一条与之前不同的客户编号
int createUser(User user)	生成一条与之前不同的客户编号
User findByUsername(String username)	生成一条与之前不同的客户编号

```
<tr>
    <td colspan="3"><strong><div class="text" style=" text-align:center;">提供的服务（供接口
</tr>
```

```
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">CustomerService.getCustomersByType</td>
    <td>语法</td>
    <td>List<CustomerPO> getCustomersByType(CustomerType type)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">CustomerService.findCustomerById</td>
    <td>语法</td>
    <td>CustomerPO findCustomerById(Integer supplier)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">CustomerService.addCustomer</td>
    <td>语法</td>
    <td>void addCustomer(CustomerVO customerVO)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">CustomerService.deleteCustomerById</td>
    <td>语法</td>
    <td>void deleteCustomerById(Integer id)</td>
```

```
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">CustomerService.updateCustomer</td>
    <td>语法</td>
    <td>void updateCustomer(CustomerVO customerVO)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">CustomerService.queryAllCustomer</td>
    <td>语法</td>
    <td>List<CustomerVO> queryAllCustomer()</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">EmployeeService.addEmployee</td>
    <td>语法</td>
    <td>void addEmployee(EmployeeVO employeeVO)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">EmployeeService.deleteEmployeeById</td>
    <td>语法</td>
    <td>void deleteEmployeeById(Integer id)</td>
</tr>
```

```
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">EmployeeService.updateEmployee</td>
    <td>语法</td>
    <td>void updateEmployee(EmployeeVO employeeVO)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">EmployeeService.addPunch</td>
    <td>语法</td>
    <td>void addPunch(EmployeePunchVO employeePunchVO)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">EmployeeService.getALLEmployees</td>
    <td>语法</td>
    <td>List<EmployeeVO> getALLEmployees()</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">EmployeeService.getEmployeeSalaryGrantingModeById</td>
    <td>语法</td>
    <td>String getEmployeeSalaryGrantingModeById(Integer id)</td>
</tr>
<tr>
```

```
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">EmployeeService.getEmployeeSalaryCalculatingModeById</td>
    <td>语法</td>
    <td>String getEmployeeSalaryCalculatingModeById(Integer id)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">EmployeeService.getPunchedTimesByEmployeeName</td>
    <td>语法</td>
    <td>int getPunchedTimesByEmployeeName(String name)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">EmployeeService.showPunchByEmployeeName</td>
    <td>语法</td>
    <td>List<EmployeePunchVO> showPunchByEmployeeName(String name)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
```

表12 rolebl模块的接口规范

CustomerService.updateCustomer	语法	void updateCustomer(CustomerPO customerPO)

需要的接口（需接口）	
服务名	服务
CustomerDataService.generateId()	生成一条与之前不同的客户编号

```
<tr>
    <td colspan="3"><strong><div class="text" style=" text-align:center;">提供的服务（供接口
</tr>
```

```
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">CategoryService.queryAllCategory</td>
    <td>语法</td>
    <td>    List<CategoryVO> queryAllCategory()</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">CategoryService.updateCategory</td>
    <td>语法</td>
    <td>    CategoryVO updateCategory(Integer id, String name)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">CategoryService.deleteCategory</td>
    <td>语法</td>
    <td>    void deleteCategory(Integer id)</td>
```

```
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">ProductService.createProduct</td>
    <td>语法</td>
    <td>    ProductInfoVO createProduct(CreateProductVO inputVO)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">ProductService.updateProduct</td>
    <td>语法</td>
    <td>    ProductInfoVO updateProduct(ProductInfoVO productInfoVO)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">ProductService.queryAllProduct</td>
    <td>语法</td>
    <td>    List<ProductInfoVO> queryAllProduct()</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">ProductService.deleteById</td>
    <td>语法</td>
    <td>    void deleteById(String id)</td>
</tr>
```

```
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">ProductService.getOneProductByPid</td>
    <td>语法</td>
    <td>    ProductInfoVO getOneProductByPid(String pid)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">ProductService.productWarehousing</td>
    <td>语法</td>
    <td>    void productWarehousing(WarehouseInputFormVO warehouseInputFormVO)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">WarehouseService.productOutOfWarehouse</td>
    <td>语法</td>
    <td>    void productOutOfWarehouse(WarehouseOutputFormVO warehouseOutputFormListVO)</td
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">WarehouseService.getWareProductInfo</td>
    <td>语法</td>
    <td>    List<WarehouseOneProductInfoVO> getWareProductInfo(GetWareProductInfoParamsVO p
</tr>
<tr>
```

```
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">WarehouseService.approvalInputSheet</td>
    <td>语法</td>
    <td>    void approvalInputSheet(UserVO user, String warehouseInputSheetId, WarehouseInp
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">WarehouseService.getWareHouseInputSheetByState</td>
    <td>语法</td>
    <td>    List<WarehouseInputSheetPO> getWareHouseInputSheetByState(WarehouseInputSheetSt
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">WarehouseService.getWareHouseOutSheetByState</td>
    <td>语法</td>
    <td>    List<WarehouseOutputSheetPO> getWareHouseOutSheetByState(WarehouseOutputSheetSt
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">WarehouseService.approvalOutputSheet</td>
    <td>语法</td>
    <td>    void approvalOutputSheet(UserVO user, String sheetId, WarehouseOutputSheetState
</tr>
<tr>
    <td>前置条件</td>
```

```
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">WarehouseService.getWHISheetContentById</td>
    <td>语法</td>
    <td>    List<WarehouseInputSheetContentPO> getWHISheetContentById(String sheetId)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">WarehouseService.getWHOSheetContentById</td>
    <td>语法</td>
    <td>    List<WarehouseOutputSheetContentPO> getWHOSheetContentById(String sheetId)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">WarehouseService.getWarehouseIODetailByTime</td>
    <td>语法</td>
    <td>    List<WarehouseIODetailPO> getWarehouseIODetailByTime(String beginDateStr,String
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">WarehouseService.getWarehouseInputProductQuantityByTime</td>
    <td>语法</td>
    <td>    int getWarehouseInputProductQuantityByTime(String beginDateStr,String endDateSt
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
```

```
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">WarehouseService.getWarehouseOutProductQuantityByTime</td>
    <td>语法</td>
    <td>    int getWarehouseOutProductQuantityByTime(String beginDateStr,String endDateStr)
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>    <tr>
    <td rowspan="3">WarehouseService.warehouseCounting</td>
    <td>语法</td>
    <td>    List<WarehouseCountingVO> warehouseCounting()</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
```

表13 warehousebl模块的接口规范

CategoryService.createCategory	语法	CategoryVO createCategory(Integer parentId, String name)

需要的接口（需接口）	
服务名	服务
CustomerDataService.generateId()	生成一条与之前不同的客户编号

```
<tr>
    <td colspan="3"><strong><div class="text" style=" text-align:center;">提供的服务（供接口
</tr>
```

```
<tr>
    <td>前置条件</td>
```

```
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">PromotionService.addPromotion</td>
    <td>语法</td>
    <td>void addPromotion(Object promotionVO)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">PromotionService.getPromotions</td>
    <td>语法</td>
    <td>List<Object> getPromotions()</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">SaleReturnsService.makeSaleReturnsSheet</td>
    <td>语法</td>
    <td>void makeSaleReturnsSheet(UserVO userVO, SaleReturnsSheetVO saleReturnsSheetVO)</td
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">SaleReturnsService.getSaleReturnsSheetByState</td>
    <td>语法</td>
    <td>List<SaleReturnsSheetVO> getSaleReturnsSheetByState(SaleReturnsSheetState state)</t
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
```

```
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">SaleReturnsService.approval</td>
    <td>语法</td>
    <td>void approval(String saleReturnsSheetId, SaleReturnsSheetState state)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">SaleReturnsService.findSheetIdByOperator</td>
    <td>语法</td>
    <td>List<String> findSheetIdByOperator(String operator)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">SaleReturnsService.findSheetIdByTime</td>
    <td>语法</td>
    <td>List<String> findSheetIdByTime(Date beginDate, Date endDate)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">SaleReturnsService.findSaleSheetsBySomeConditions</td>
    <td>语法</td>
    <td>List<SalesDetailsSheetVO> findSaleSheetsBySomeConditions(SaleDetailsSheetSearchCrit
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
```

```
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">SaleReturnsService.findSheetIdBySomeConditions</td>
    <td>语法</td>
    <td>List<String> findSheetIdBySomeConditions(BusinessHistorySheetSearchCriteriaPO busir
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">SaleReturnsService.</td>
    <td>语法</td>
    <td>SaleReturnsSheetVO findSheetIdById(String id)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">SaleService.makeSaleSheet</td>
    <td>语法</td>
    <td>void makeSaleSheet(UserVO userVO, SaleSheetVO saleSheetVO)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">SaleService.getSaleSheetByState</td>
    <td>语法</td>
    <td>List<SaleSheetVO> getSaleSheetByState(SaleSheetState state)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
```

```
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">SaleService.approval</td>
    <td>语法</td>
    <td>void approval(String saleSheetId, SaleSheetState state)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">SaleService.getMaxAmountCustomerOfSalesmanByTime</td>
    <td>语法</td>
    <td>CustomerPurchaseAmountPO getMaxAmountCustomerOfSalesmanByTime(String salesman,Strir
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">SaleService.getSaleSheetById</td>
    <td>语法</td>
    <td>SaleSheetVO getSaleSheetById(String saleSheetId)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">SaleService.findSheetIdByOperator</td>
    <td>语法</td>
    <td>List<String> findSheetIdByOperator(String operator)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
```

```
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">SaleService.findSheetIdByTime</td>
    <td>语法</td>
    <td>List<String> findSheetIdByTime(Date beginDate, Date endDate)</td>
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">SaleService.findSaleSheetsBySomeConditions</td>
    <td>语法</td>
    <td>List<SalesDetailsSheetVO> findSaleSheetsBySomeConditions(SaleDetailsSheetSearchCrit
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
<tr>
    <td rowspan="3">SaleService.findSheetIdBySomeConditions</td>
    <td>语法</td>
    <td>List<String> findSheetIdBySomeConditions(BusinessHistorySheetSearchCriteriaPO searc
</tr>
<tr>
    <td>前置条件</td>
    <td>启动客户管理任务</td>
</tr>
<tr>
    <td>后置条件</td>
    <td>返回查询的客户信息的结果</td>
</tr>
```

表14 salebl模块的接口规范

PromotionService.deletePromotionById	语法	void deletePromotionById(int id)

需要的接口（需接口）	
服务名	服务

CustomerDataService.generateId()	生成一条与之前不同的客户编号

5.4 数据层的分解

5.4.1 数据层模块的职责

数据层模块的职责

模块	职责
BankAccountDao	持久化数据接口,提供BankAccountPO的集体载入、集体保存、增删改查服务
CashExpenseSheetDao	持久化数据接口,提供CashExpenseSheetPO的集体载入、集体保存、增删改查服务
CategoryDao	持久化数据接口,提供CategoryPO的集体载入、集体保存、增删改查服务
CombinePromotionDao	持久化数据接口,提供CombinePromotionPO的集体载入、集体保存、增删改查服务
CustomerDao	持久化数据接口,提供CustomerPO的集体载入、集体保存、增删改查服务
CustomerPromotionDao	持久化数据接口,提供CustomerPromotionPO的集体载入、集体保存、增删改查服务
EmployeeDao	持久化数据接口,提供EmployeePO的集体载入、集体保存、增删改查服务
PaymentOrderDao	持久化数据接口,提供PaymentOrderPO的集体载入、集体保存、增删改查服务
PayrollDao	持久化数据接口,提供PayrollPO的集体载入、集体保存、增删改查服务
ProductDao	持久化数据接口,提供ProductPO的集体载入、集体保存、增删改查服务
PurchaseReturnsSheetDao	持久化数据接口,提供PurchaseReturnsSheetPO的集体载入、集体保存、增删改查服务
PurchaseSheetDao	持久化数据接口,提供PurchaseSheetPO的集体载入、集体保存、增删改查服务
ReceiptDao	持久化数据接口,提供ReceiptPO的集体载入、集体保存、增删改查服务
SaleReturnsSheetDao	持久化数据接口,提供SaleReturnsSheetPO的集体载入、集体保存、增删改查服务

SaleSheetDao	持久化数据接口,提供SaleSheetPO的集体载入、集体保存、增删改查服务
TermSheetDao	持久化数据接口,提供TermSheetPO的集体载入、集体保存、增删改查服务
TotalPromotionDao	持久化数据接口,提供TotalPromotionPO的集体载入、集体保存、增删改查服务
TransferRecordDao	持久化数据接口,提供TransferRecordPO的集体载入、集体保存、增删改查服务
UserDao	件的持久化数据接口,提供UserPO的集体载入、集体保存、增删改查服务
WarehouseDao	持久化数据接口,提供WarehousePO的集体载入、集体保存、增删改查服务
WarehouseInputSheetDao	持久化数据接口,提供WarehouseInputSheetPO的集体载入、集体保存、增删改查服务
WarehouseOutputSheetDao	久化数据接口,提供WarehouseOutputSheetPO的集体载入、集体保存、增删改查服务
BankAccountMapper	基于Mysql数据库的持久化数据接口,提供BankAccountPO的集体载入、集体保存、增删改查服务
CashExpenseSheetMapper	基于Mysql数据库的持久化数据接口,提供CashExpenseSheetPO的集体载入、集体保存、增删改查服务
CategoryMapper	基于Mysql数据库的持久化数据接口,提供CategoryPO的集体载入、集体保存、增删改查服务
CombinePromotionMapper	基于Mysql数据库的持久化数据接口,提供CombinePromotionPO的集体载入、集体保存、增删改查服务
CustomerMapper	基于Mysql数据库的持久化数据接口,提供CustomerPO的集体载入、集体保存、增删改查服务
CustomerPromotionMapper	基于Mysql数据库的持久化数据接口,提供CustomerPromotionPO的集体载入、集体保存、增删改查服务
EmployeeMapper	基于Mysql数据库的持久化数据接口,提供EmployeePO的集体载入、集体保存、增删改查服务
PaymentOrderMapper	基于Mysql数据库的持久化数据接口,提供PaymentOrderPO的集体载入、集体保存、增删改查服务
PayrollMapper	基于Mysql数据库的持久化数据接口,提供PayrollPO的集体载入、集体保

	存、增删改查服务
ProductMapper	基于Mysql数据库的持久化数据接口,提供ProductPO的集体载入、集体保存、增删改查服务
PurchaseReturnsSheetMapper	基于Mysql数据库的持久化数据接口,提供PurchaseReturnsSheetPO的集体载入、集体保存、增删改查服务
PurchaseSheetMapper	基于Mysql数据库的持久化数据接口,提供PurchaseSheetPO的集体载入、集体保存、增删改查服务
ReceiptMapper	基于Mysql数据库的持久化数据接口,提供ReceiptPO的集体载入、集体保存、增删改查服务
SaleReturnsSheetMapper	基于Mysql数据库的持久化数据接口,提供SaleReturnsSheetPO的集体载入、集体保存、增删改查服务
SaleSheetMapper	基于Mysql数据库的持久化数据接口,提供SaleSheetPO的集体载入、集体保存、增删改查服务
TermSheetMapper	基于Mysql数据库的持久化数据接口,提供TermSheetPO的集体载入、集体保存、增删改查服务
TotalPromotionMapper	基于Mysql数据库的持久化数据接口,提供TotalPromotionPO的集体载入、集体保存、增删改查服务
TransferRecordMapper	基于Mysql数据库的持久化数据接口,提供TransferRecordPO的集体载入、集体保存、增删改查服务
UserMapper	基于Mysql数据库的持久化数据接口,提供UserPO的集体载入、集体保存、增删改查服务
WarehouseMapper	基于Mysql数据库的持久化数据接口,提供WarehousePO的集体载入、集体保存、增删改查服务
WarehouseInputSheetMapper	基于Mysql数据库的持久化数据接口,提供WarehouseInputSheetPO的集体载入、集体保存、增删改查服务
WarehouseOutputSheetMapper	基于Mysql数据库的持久化数据接口,提供WarehouseOutputSheetPO的集体载入、集体保存、增删改查服务

5.4.2 数据层模块的接口规范

数据层模块(userdao)的接口规范

提供的服务（供接口）		
UserDao.	语法	public User findByUsernameAndPassword(String username,

findByUsernameAndPassword		String password)
	前置条件	无
	后置条件	返回用户名和密码所对应的记录
UserDao.findByUsername	语法	public User findByUsername(String username)
	前置条件	无
	后置条件	返回用户名所对应的记录
UserDao.createUser	语法	public int createUser(User user)
	前置条件	user中个字段值均不为null
	后置条件	返回创建的user的id

```
<td>语法</td>
    <td>int updateOne(CustomerPO customerPO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CustomerDao.findOneById</td>
    <td>语法</td>
    <td>CustomerPO findOneById(Integer supplier)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CustomerDao.findAllByType</td>
    <td>语法</td>
    <td>List findAllByType(CustomerType customerType)</td>
<tr>
```

```
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CustomerDao.createCustomer</td>
    <td>语法</td>
    <td>int createCustomer(CustomerPO customerPO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CustomerDao.deleteCustomerById</td>
    <td>语法</td>
    <td>int deleteCustomerById(Integer id)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CustomerDao.findAll</td>
    <td>语法</td>
    <td>List findAll()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>EmployeeDao.createEmployee</td>
    <td>语法</td>
    <td>int createEmployee(EmployeePO employeePO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>EmployeeDao.deleteEmployeeById</td>
    <td>语法</td>
```

```
    <td>int deleteEmployeeById(Integer id)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>EmployeeDao.updateEmployee</td>
    <td>语法</td>
    <td>int updateEmployee(EmployeePO employeePO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>EmployeeDao.findAllEmployees</td>
    <td>语法</td>
    <td>List findAllEmployees()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>EmployeeDao.findEmployeeById</td>
    <td>语法</td>
    <td>EmployeePO findEmployeeById(Integer id)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>EmployeeDao.getPunchByEmployeeName</td>
    <td>语法</td>
    <td>List getPunchByEmployeeName(String name)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
```

```
<td rowspan=3>EmployeeDao.punch</td>
    <td>语法</td>
    <td>int punch(EmployeePunchPO employeePunchPO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>EmployeeDao.getPunchById</td>
    <td>语法</td>
    <td>EmployeePunchPO getPunchById(Integer id)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
```

数据层模块(roledao)的接口规范

提供的服务（供接口）
CustomerDao.updateOne

```
<td>语法</td>
    <td>int createCategory(CategoryPO categoryPO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CategoryDao.findByCategoryId</td>
    <td>语法</td>
    <td>CategoryPO findByCategoryId(Integer categoryId)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CategoryDao.findAll</td>
    <td>语法</td>
    <td>List findAll()</td>
```

```
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CategoryDao.updateById</td>
    <td>语法</td>
    <td>int updateById(CategoryPO categoryPO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CategoryDao.deleteById</td>
    <td>语法</td>
    <td>int deleteById(Integer id)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>ProductDao.createProduct</td>
    <td>语法</td>
    <td>int createProduct(ProductPO productPO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>ProductDao.updateById</td>
    <td>语法</td>
    <td>int updateById(ProductPO productPO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>ProductDao.findById</td>
```

```
    <td>语法</td>
    <td>ProductPO findById(String id)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>ProductDao.findAll</td>
    <td>语法</td>
    <td>List findAll()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>ProductDao.deleteById</td>
    <td>语法</td>
    <td>int deleteById(String id)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>WarehouseDao.saveBatch</td>
    <td>语法</td>
    <td>void saveBatch(List warehousePOList)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>WarehouseDao.deductQuantity</td>
    <td>语法</td>
    <td>void deductQuantity(WarehousePO warehousePO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
```

```
<tr>
<td rowspan=3>WarehouseDao.addQuantity</td>
    <td>语法</td>
    <td>void addQuantity(WarehousePO warehousePO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>WarehouseDao.findAllNotZeroByPidSortedByBatchId</td>
    <td>语法</td>
    <td>List findAllNotZeroByPidSortedByBatchId(String pid)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>WarehouseDao.findByPidOrderByPurchasePricePos</td>
    <td>语法</td>
    <td>List findByPidOrderByPurchasePricePos(String pid)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>WarehouseDao.findOneByPidAndBatchId</td>
    <td>语法</td>
    <td>WarehousePO findOneByPidAndBatchId(String pid, Integer batchId)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>WarehouseDao.findByPidOrderByQuantity</td>
    <td>语法</td>
    <td>List findByPidOrderByQuantity(String pid)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
```

```
    <td></td>
<tr>
<tr>
<td rowspan=3>WarehouseDao.findAll</td>
    <td>语法</td>
    <td>List findAll()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>WarehouseInputSheetDao.getLatest</td>
    <td>语法</td>
    <td>WarehouseInputSheetPO getLatest()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>WarehouseInputSheetDao.save</td>
    <td>语法</td>
    <td>int save(WarehouseInputSheetPO toSave)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>WarehouseInputSheetDao.saveBatch</td>
    <td>语法</td>
    <td>void saveBatch(List warehouseInputListPOSheetContent)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>WarehouseInputSheetDao.getDraftSheets</td>
    <td>语法</td>
    <td>List getDraftSheets(WarehouseInputSheetState state)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
```

```
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>WarehouseInputSheetDao.getAllSheets</td>
    <td>语法</td>
    <td>List getAllSheets()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>WarehouseInputSheetDao.getSheet</td>
    <td>语法</td>
    <td>WarehouseInputSheetPO getSheet(String id)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>WarehouseInputSheetDao.updateById</td>
    <td>语法</td>
    <td>int updateById(WarehouseInputSheetPO warehouseInputSheetPO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>WarehouseInputSheetDao.getAllContentById</td>
    <td>语法</td>
    <td>List getAllContentById(String warehouseInputSheetId)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>WarehouseInputSheetDao.getWarehouseIODetailByTime</td>
    <td>语法</td>
    <td>List getWarehouseIODetailByTime(Date beginTime,Date endTime)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
```

```
<td rowspan=3>WarehouseInputSheetDao.getWarehouseInputProductQuantityByTime</td>
    <td>语法</td>
    <td>Integer getWarehouseInputProductQuantityByTime(Date beginTime,Date endTime)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>WarehouseOutputSheetDao.getLatest</td>
    <td>语法</td>
    <td>WarehouseOutputSheetPO getLatest()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>WarehouseOutputSheetDao.save</td>
    <td>语法</td>
    <td>void save(WarehouseOutputSheetPO toSave)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>WarehouseOutputSheetDao.saveBatch</td>
    <td>语法</td>
    <td>void saveBatch(List warehouseOutputListPOSheetContent)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>WarehouseOutputSheetDao.getAllSheets</td>
    <td>语法</td>
    <td>List getAllSheets()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>WarehouseOutputSheetDao.getDraftSheets</td>
    <td>语法</td>
    <td>List getDraftSheets(WarehouseOutputSheetState state)</td>
```

```
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>WarehouseOutputSheetDao.getSheet</td>
    <td>语法</td>
    <td>WarehouseOutputSheetPO getSheet(String sheetId)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>WarehouseOutputSheetDao.updateById</td>
    <td>语法</td>
    <td>void updateById(WarehouseOutputSheetPO warehouseOutputSheetPO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>WarehouseOutputSheetDao.getAllContentById</td>
    <td>语法</td>
    <td>List getAllContentById(String sheetId)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>WarehouseOutputSheetDao.deleteContent</td>
    <td>语法</td>
    <td>void deleteContent(String sheetId)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>WarehouseOutputSheetDao.getWarehouseOutputProductQuantityByTime</td>
    <td>语法</td>
    <td>Integer getWarehouseOutputProductQuantityByTime(Date beginTime,Date endTime)</td>
<tr>
    <td>前置条件</td>
    <td></td>
```

```
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
```

数据层模块(warehousedao)的接口规范

提供的服务（供接口）

CategoryDao.createCategory

```
<td>语法</td>
    <td>void insertBankAccount(BankAccountPO bankAccountPO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>BankAccountDao.deleteBankAccountByName</td>
    <td>语法</td>
    <td>void deleteBankAccountByName(String name)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>BankAccountDao.selectBankAccountByName</td>
    <td>语法</td>
    <td>List<BankAccountPO> selectBankAccountByName(String name)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>BankAccountDao.updateBankAccount</td>
    <td>语法</td>
    <td>void updateBankAccount(BankAccountPO bankAccountPO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
```

```
<tr>
<tr>
<td rowspan=3>BankAccountDao.selectAll</td>
    <td>语法</td>
    <td>List<BankAccountVO> selectAll()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CashExpenseSheetDao.getLatestSheet</td>
    <td>语法</td>
    <td>CashExpenseSheetPO getLatestSheet()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CashExpenseSheetDao.saveSheet</td>
    <td>语法</td>
    <td>void saveSheet(CashExpenseSheetPO cashExpenseSheetPO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CashExpenseSheetDao.findSheetIdByOperator</td>
    <td>语法</td>
    <td>List<String> findSheetIdByOperator(String operator)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CashExpenseSheetDao.findSheetIdByTime</td>
    <td>语法</td>
    <td>List<String> findSheetIdByTime(Date beginDate, Date endDate)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
```

```
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CashExpenseSheetDao.findSheetIdBySomeConditions</td>
    <td>语法</td>
    <td>List<String> findSheetIdBySomeConditions(BusinessHistorySheetSearchCriteriaPO searc
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>PaymentOrderDao.getLatestSheet</td>
    <td>语法</td>
    <td>PaymentOrderPO getLatestSheet()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>PaymentOrderDao.saveSheet</td>
    <td>语法</td>
    <td>void saveSheet(PaymentOrderPO paymentOrderPO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>PaymentOrderDao.findSheetIdByOperator</td>
    <td>语法</td>
    <td>List<String> findSheetIdByOperator(String operator)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>PaymentOrderDao.findSheetIdByTime</td>
    <td>语法</td>
    <td>List<String> findSheetIdByTime(Date beginDate, Date endDate)</td>
<tr>
    <td>前置条件</td>
```

```
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>PaymentOrderDao.findSheetIdBySomeConditions</td>
    <td>语法</td>
    <td>List<String> findSheetIdBySomeConditions(BusinessHistorySheetSearchCriteriaPO searc
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>PayrollDao.saveSheet</td>
    <td>语法</td>
    <td>void saveSheet(PayrollPO payrollPO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>PayrollDao.getLatestSheet</td>
    <td>语法</td>
    <td>PayrollPO getLatestSheet()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>PayrollDao.findSheetIdByOperator</td>
    <td>语法</td>
    <td>List<String> findSheetIdByOperator(String operator)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>PayrollDao.findSheetIdByTime</td>
    <td>语法</td>
    <td>List<String> findSheetIdByTime(Date beginDate, Date endDate)</td>
```

```
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>PayrollDao.findSheetIdBySomeConditions</td>
    <td>语法</td>
    <td>List<String> findSheetIdBySomeConditions(BusinessHistorySheetSearchCriteriaPO searc
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>ReceiptDao.saveSheet</td>
    <td>语法</td>
    <td>int saveSheet(ReceiptPO toSave)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>ReceiptDao.getLatestSheet</td>
    <td>语法</td>
    <td>ReceiptPO getLatestSheet()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>ReceiptDao.findSheetIdByOperator</td>
    <td>语法</td>
    <td>List<String> findSheetIdByOperator(String operator)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>ReceiptDao.findSheetIdByTime</td>
    <td>语法</td>
    <td>List<String> findSheetIdByTime(Date beginDate, Date endDate)</td>
<tr>
```

```
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>ReceiptDao.findSheetIdBySomeConditions</td>
    <td>语法</td>
    <td>List<String> findSheetIdBySomeConditions(BusinessHistorySheetSearchCriteriaPO searc
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>TermSheetDao.saveBatchSheetContent</td>
    <td>语法</td>
    <td>void saveBatchSheetContent(List<TermSheetPO> termSheetPOS)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>TransferRecordDao.saveBatchSheetContent</td>
    <td>语法</td>
    <td>void saveBatchSheetContent(List<TransferRecordPO> transferRecordPOS)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>TransferRecordDao.findTransferBySheetId</td>
    <td>语法</td>
    <td>List<TransferRecordPO> findTransferBySheetId(String sheetId)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
```

数据层模块(financialdao)的接口规范

提供的服务（供接口）

BankAccountDao.insertBankAccount
    <td>PurchaseReturnsSheetPO getLatest()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>PurchaseReturnsSheetDao.save</td>
    <td>语法</td>
    <td>int save(PurchaseReturnsSheetPO toSave)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>PurchaseReturnsSheetDao.saveBatch</td>
    <td>语法</td>
    <td>void saveBatch(List<PurchaseReturnsSheetContentPO> PurchaseReturnsSheetContent)</td
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>PurchaseReturnsSheetDao.findAll</td>
    <td>语法</td>
    <td>List<PurchaseReturnsSheetPO> findAll()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>PurchaseReturnsSheetDao.findAllByState</td>
    <td>语法</td>
    <td>List<PurchaseReturnsSheetPO> findAllByState(PurchaseReturnsSheetState state)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>PurchaseReturnsSheetDao.findSheetIdByOperator</td>
    <td>语法</td>
    <td>List<String> findSheetIdByOperator(String operator)</td>

```
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>PurchaseReturnsSheetDao.updateState</td>
    <td>语法</td>
    <td>int updateState(String purchaseReturnsSheetId, PurchaseReturnsSheetState state)</td
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>PurchaseReturnsSheetDao.updateStateV2</td>
    <td>语法</td>
    <td>int updateStateV2(String purchaseReturnsSheetId, PurchaseReturnsSheetState prevStat
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>PurchaseReturnsSheetDao.findOneById</td>
    <td>语法</td>
    <td>PurchaseReturnsSheetPO findOneById(String purchaseReturnsSheetId)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
```

```
<td>语法</td>
    <td> List<PurchaseReturnsSheetContentPO> findContentByPurchaseReturnsSheetId(String pur
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>PurchaseReturnsSheetDao.findBatchId</td>
    <td>语法</td>
    <td>Integer findBatchId(String purchaseReturnsSheetId)</td>
<tr>
```

```
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>PurchaseReturnsSheetDao.findSheetIdByTime</td>
    <td>语法</td>
    <td>List<String> findSheetIdByTime(Date beginDate, Date endDate)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>PurchaseReturnsSheetDao.findSheetIdBySomeConditions</td>
    <td>语法</td>
    <td>List<String> findSheetIdBySomeConditions(BusinessHistorySheetSearchCriteriaPO searc
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>PurchaseReturnsSheetDao.findContentByPurchaseReturnSheetId</td>
    <td>语法</td>
    <td>List<PurchaseReturnsSheetContentPO> findContentByPurchaseReturnSheetId(String id)</
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
```

```
<td>语法</td>
    <td>PurchaseSheetPO getLatest()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>.</td>
    <td>语法</td>
    <td>int save(PurchaseSheetPO toSave)</td>
<tr>
    <td>前置条件</td>
```

```
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>.</td>
    <td>语法</td>
    <td>void saveBatch(List<PurchaseSheetContentPO> purchaseSheetContent)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<td rowspan=3>.</td>
    <td>语法</td>
    <td>List<PurchaseSheetPO> findAll()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
```

```
<td>语法</td>
    <td>List<PurchaseSheetPO> findAllByState(PurchaseSheetState state)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
```

```
<td>语法</td>
    <td>List<String> findSheetIdByOperator(String operator)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
```

```
<td>语法</td>
    <td>int updateState(String purchaseSheetId, PurchaseSheetState state)</td>
```

```
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
```

```
<td>语法</td>
    <td>int updateStateV2(String purchaseSheetId, PurchaseSheetState prevState, PurchaseShe
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
```

```
<td>语法</td>
    <td>PurchaseSheetPO findOneById(String purchaseSheetId)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
```

```
<td>语法</td>
    <td>List<PurchaseSheetContentPO> findContentByPurchaseSheetId(String purchaseSheetId)</
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
```

```
<td>语法</td>
    <td>List<String> findSheetIdByTime(Date beginDate, Date endDate)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
```

```
    <td>后置条件</td>
    <td></td>
<tr>
```

```
<td>语法</td>
    <td>List<String> findSheetIdBySomeConditions(BusinessHistorySheetSearchCriteriaPO searc
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
```

数据层模块(purchasedao)的接口规范

提供的服务（供接口）	
PurchaseReturnsSheetDao.getLatest	PurchaseReturnsSheetDao.findContentByPurchaseReturnsSheetId

```
<td>语法</td>
    <td>SaleReturnsSheetPO getLatest()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleReturnsSheetDao.save</td>
    <td>语法</td>
    <td>int save(SaleReturnsSheetPO toSave)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleReturnsSheetDao.saveBatch</td>
    <td>语法</td>
    <td>void saveBatch(List<SaleReturnsSheetContentPO> saleReturnsSheetContent)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
```

```
<tr>
<tr>
<td rowspan=3>SaleReturnsSheetDao.findAll</td>
    <td>语法</td>
    <td>List<SaleReturnsSheetPO> findAll()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleReturnsSheetDao.findAllByState</td>
    <td>语法</td>
    <td>List<SaleReturnsSheetPO> findAllByState(SaleReturnsSheetState state)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleReturnsSheetDao.findSheetIdByOperator</td>
    <td>语法</td>
    <td>List<String> findSheetIdByOperator(String operator)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleReturnsSheetDao.updateState</td>
    <td>语法</td>
    <td>int updateState(String saleReturnsSheetId, SaleReturnsSheetState state)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleReturnsSheetDao.updateStateV2</td>
    <td>语法</td>
    <td>int updateStateV2(String saleReturnsSheetId, SaleReturnsSheetState prevState, SaleR
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
```

```
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleReturnsSheetDao.findOneById</td>
    <td>语法</td>
    <td>SaleReturnsSheetPO findOneById(String saleReturnsSheetId)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleReturnsSheetDao.findContentBySaleReturnsSheetId</td>
    <td>语法</td>
    <td>List<SaleReturnsSheetContentPO> findContentBySaleReturnsSheetId(String saleReturnsS
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleReturnsSheetDao.findSheetIdByTime</td>
    <td>语法</td>
    <td>List<String> findSheetIdByTime(Date beginDate, Date endDate)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleReturnsSheetDao.findSaleSheetsBySomeConditions</td>
    <td>语法</td>
    <td>List<SalesDetailsSheetVO> findSaleSheetsBySomeConditions(SaleDetailsSheetSearchCrit
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleReturnsSheetDao.findSheetIdBySomeConditions</td>
    <td>语法</td>
    <td>List<String> findSheetIdBySomeConditions(BusinessHistorySheetSearchCriteriaPO searc
<tr>
    <td>前置条件</td>
```

```
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleReturnsSheetDao.findSheetById</td>
    <td>语法</td>
    <td>SaleReturnsSheetPO findSheetById(String id)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleSheetDao.getLatestSheet</td>
    <td>语法</td>
    <td>SaleSheetPO getLatestSheet()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleSheetDao.saveSheet</td>
    <td>语法</td>
    <td>int saveSheet(SaleSheetPO toSave)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleSheetDao.saveBatchSheetContent</td>
    <td>语法</td>
    <td>int saveBatchSheetContent(List<SaleSheetContentPO> saleSheetContent)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleSheetDao.findAllSheet</td>
    <td>语法</td>
    <td> List<SaleSheetPO> findAllSheet()</td>
```

```
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleSheetDao.findSheetById</td>
    <td>语法</td>
    <td>SaleSheetPO findSheetById(String id)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleSheetDao.findSheetByState</td>
    <td>语法</td>
    <td>List<SaleSheetPO> findSheetByState(SaleSheetState state)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleSheetDao.findSheetIdByOperator</td>
    <td>语法</td>
    <td>List<String> findSheetIdByOperator(String operator)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
```

```
<td>语法</td>
    <td>List<SaleSheetContentPO> findContentBySheetId(String sheetId)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
```

```
<td rowspan=3>SaleSheetDao.updateSheetState</td>
    <td>语法</td>
    <td>int updateSheetState(String sheetId, SaleSheetState state)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleSheetDao.updateSheetStateOnPrev</td>
    <td>语法</td>
    <td>int updateSheetStateOnPrev(String sheetId, SaleSheetState prev, SaleSheetState stat
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleSheetDao.getMaxAmountCustomerOfSalesmanByTime</td>
    <td>语法</td>
    <td>CustomerPurchaseAmountPO getMaxAmountCustomerOfSalesmanByTime(String salesman, Date
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleSheetDao.findSheetIdByTime</td>
    <td>语法</td>
    <td> List<String> findSheetIdByTime(Date beginDate, Date endDate)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>SaleSheetDao.findSaleSheetBySomeConditions</td>
    <td>语法</td>
    <td>List<SalesDetailsSheetVO> findSaleSheetBySomeConditions(SaleDetailsSheetSearchCrite
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
```

```
<tr>
<tr>
<td rowspan=3>SaleSheetDao.findSheetIdBySomeConditions</td>
    <td>语法</td>
    <td>List<String> findSheetIdBySomeConditions(BusinessHistorySheetSearchCriteriaPO searc
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CombinePromotionDao.createPromotion</td>
    <td>语法</td>
    <td>int createPromotion(CombinePromotionPO promotionPO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CombinePromotionDao.deletePromotionById</td>
    <td>语法</td>
    <td> int deletePromotionById(Integer id)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CombinePromotionDao.showPromotions</td>
    <td>语法</td>
    <td> List<CombinePromotionPO> showPromotions()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CombinePromotionDao.getPromotionById</td>
    <td>语法</td>
    <td>CombinePromotionPO getPromotionById(Integer id)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
```

```
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CustomerPromotionDao.createPromotion</td>
    <td>语法</td>
    <td>int createPromotion(CustomerPromotionPO promotionPO)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CustomerPromotionDao.deletePromotionById</td>
    <td>语法</td>
    <td>int deletePromotionById(Integer id)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>CustomerPromotionDao.showPromotions</td>
    <td>语法</td>
    <td>List<CustomerPromotionPO> showPromotions()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
```

```
<td>语法</td>
    <td>CustomerPromotionPO getPromotionById(Integer id)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>TotalPromotionDao.createPromotion</td>
    <td>语法</td>
    <td>int createPromotion(TotalPromotionPO promotionPO)</td>
<tr>
```

```
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>TotalPromotionDao.deletePromotionById</td>
    <td>语法</td>
    <td>int deletePromotionById(Integer id)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>TotalPromotionDao.showPromotions</td>
    <td>语法</td>
    <td>List<TotalPromotionPO> showPromotions()</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
<tr>
<td rowspan=3>TotalPromotionDao.getPromotionById</td>
    <td>语法</td>
    <td>TotalPromotionPO getPromotionById(int id)</td>
<tr>
    <td>前置条件</td>
    <td></td>
<tr>
    <td>后置条件</td>
    <td></td>
<tr>
```

数据层模块(saledao)的接口规范

提供的服务（供接口）		
SaleReturnsSheetDao. getMaxSheetId	SaleSheetDao. findContentBySheetId	CustomerPromotionDao. getPromotionById

6.信息视角

6.1 数据持久化对象

系统的PO类就是对应的相关实体类，系统主要的PO类如下图所示。

- BankAccountPO类：银行账户名称、银行账户余额。

- BusinessHistorySheetSearchCriteriaPO：起始时间、结束时间、客户、业务员。
- CashExpenseSheetPO：现金费用单单据编号、操作员、银行账户、总额、单据状态、创建时间。
- CategoryPO：分类id、分类名、父分类ID、是否为叶节点、商品数量、下一个商品index。
- CombinePromotionPO：id、开始时间、结束时间、条件、数量。
- CustomerPO：编号、分类(供应商)、级别、姓名、电话号码、地址、邮编、电子邮箱、应收额度、应收、应付、默认业务员。
- CustomerPromotionPO：id、开始时间、结束时间、条件、数量。
- CustomerPurchaseAmountPO：客户、用户在某时间段内通过某销售人员购买的商品的总金额。
- EmployeePO：员工id、员工姓名、员工性别、员工生日、员工电话号码、员工岗位、员工基本工资、员工岗位工资、员工薪资发放方式、员工薪资计算方式、员工账户。
- EmployeePunchPO：打卡记录id、打卡员工用户名、打卡时间。
- PaymentOrderPO：付款单单据编号、供应商id、销售商id、操作员、总额汇总、单据状态、创建时间。
- PayrollPO：单据编号、员工编号、姓名、银行账户信息、应发工资、个人所得税、失业保险、住房公积金、扣除税款实发工资、单据状态、创建时间。
- ProductPO：商品id、商品名、分类ID、商品型号、商品数量、进价、零售价、最近进价、最近零售价。
- PurchaseReturnsSheetContentPO：自增id、进货退货单id、商品id、数量、单价、总金额、备注。
- PurchaseReturnsSheetPO：进货退货单单据编号、关联的进货单id、操作员、备注、退货的总额合计、单据状态、创建时间。
- PurchaseSheetContentPO：自增id、进货单id、商品id、数量、单价、总金额、备注。
- PurchaseSheetPO：进货单单据编号、供应商id、操作员、备注、总额合计、单据状态、创建时间。
- ReceiptPO：收款单单据编号、供应商id、销售商id、操作员、总额汇总、单据状态、创建时间。
- SaleDetailsSheetSearchCriteriaPO：起始时间、结束时间、操作员、商品名、客户。
- SaleReturnsSheetContentPO：自增id、销售退货单id、商品id、数量、单价、总金额、备注。
- SaleReturnsSheetPO：销售退货单单据编号、关联的销售单id、操作员、备注、退款总额、单据状态、创建时间。
- SaleSheetContentPO：自增id、销售单id、商品id、数量、单价、总金额、备注。
- SaleSheetPO：销售单单据编号、客户/销售商id、业务员、操作员、备注、折让前总额、折让、使用代金券总额、折让后总额、单据状态、创建时间。
- TermSheetPO：自增id、对应现金费用单id、条目名、金额、备注。
- TotalPromotionPO：id、开始时间、结束时间、条件、数量。

- TransferRecordPO：自增id、收款单id、银行账户、转账金额、备注。
- User：用户ID、用户姓名、用户密码、用户身份。
- WarehouseInputSheetContentPO：入库商品列表id、入库单编号、商品id、商品数量、单价、出厂日期、备注。
- WarehouseInputSheetPO：入库单编号、批次、操作员、操作时间、关联的进货单据、单据状态。
- WarehouseIODetailPO：库存操作类型、出库单/入库单id、商品名、分类名、商品数量、商品单价、商品总价、出库单/入库单创建时间。
- WarehouseOutputSheetContentPO：出库商品列表id、商品id、出库单编号、批次、商品数量、单价、备注。
- WarehouseOutputSheetPO：出库单编号、操作员、操作时间、关联的销售单据、单据状态。
- WarehousePO：库存id、商品编号、数量、进价、批次、出厂日期。

6.2 数据库表

数据库中包含 bank_account表、 cash_expense_sheet表、 category表、 customer表、 customer_promotion_strategy表、 employee表、 employee_punch表、 payment_order_sheet表、 payroll表、 product表、 purchase_returns_sheet表、 purchase_returns_sheet_content表、 purchase_sheet表、 purchase_sheet_content表、 receipt_sheet表、 sale_returns_sheet表、 sale_returns_sheet_content表、 sale_sheet表、 sale_sheet_content表、 term_sheet_record表、 total_promotion_strategy表、 transfer_record表、 user表、 warehouse表、 warehouse_input_sheet表、 warehouse_input_sheet_content表、 warehouse_output_sheet表、 warehouse_output_sheet_content表。