进销存系统

体系结构描述文档

团队：谁改需求就打死他

2014年10月18日

# 版本控制

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **修订人** | **内容** | **时间** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

# 目录

1.引言……………………………………………………………………………

1.1编制目的

1.2参考资料

2.产品概述

3.逻辑视角

4.组合视角

4.1开发包图

4.2运行时进程

4.3物理部署

5.接口视角

5.1模块职责

5.2用户界面层的分解

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

5.3.3用户界面层模块设计原理

5.3业务逻辑层的分解

5.3.1业务逻辑层的职责

5.3.2业务逻辑层模块的接口规范

5.4数据层的分解

5.4.1数据层的职责

5.4.2数据层模块的接口规范

6.信息视角

6.1数据持久化对象

# 引言

## 1.1编制目的

本报告详细完成对进销存系统的概要设计，达到指导详细设计和开发的目的，同时实现和测试人员及用户的沟通。

本报告面向开发人员，测试人员及最终用户而编写，是了解系统的导航。

## 1.2参考资料

软工二教材。

# 2.产品概述

参考进销存系统用例文档和进销存系统软件需求规格说明中对产品的概括描述。

# 3.逻辑视角

进销存系统中，选择了分层体系结构风格，将系统分为3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。展示成包含GUI页面的实现，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图1和图2所示。

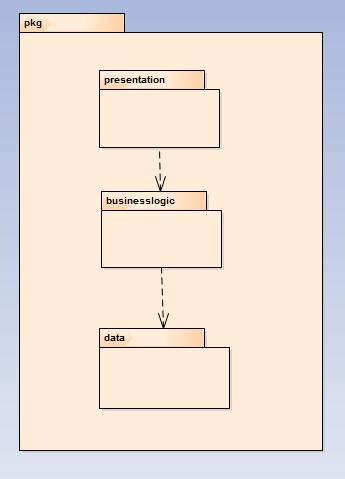


图1 参照体系结构风格的包图表达逻辑视角

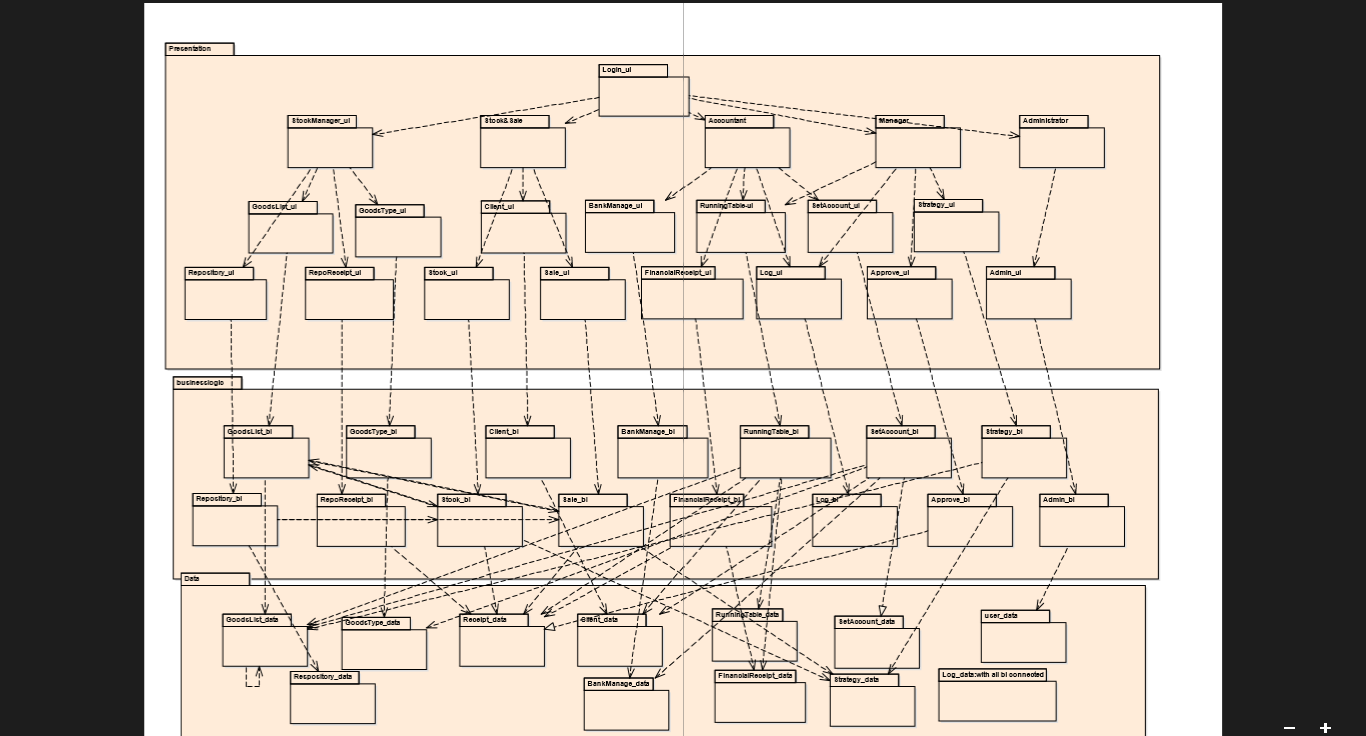


图2 软件体系结构逻辑设计方案

待修订

# 4.组合视角

## 4.1开发包图

进销存系统的最终开发包设计如表1所示。

进销存系统客户端开发包图如图3所示。

图3进销存系统客户端开发包

进销存系统服务器开发包图如图4所示。

## 4.2运行时进程

在进销存系统中，有多个客户端进程和一个服务器端进程，其进程图如图5所示，结合部署图，客户端进程是在客户端进程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。



图5进程图

## 4.3物理部署

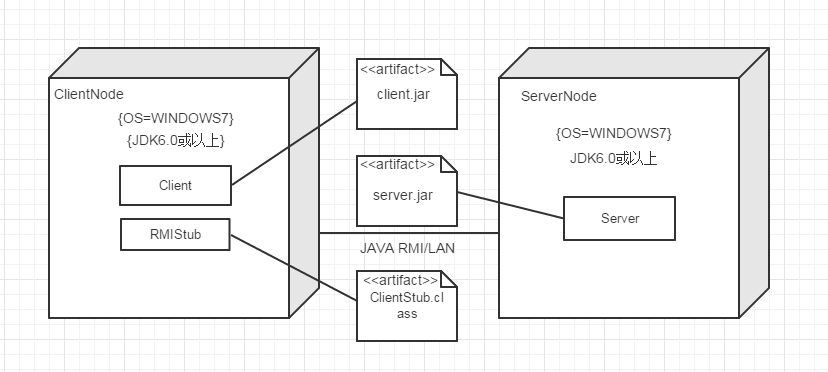
进销存系统中客户端构建放在客户端机器上，服务端构建放在服务端机器上。在客户端节点上，需部署RMIStub构件。JAVA RMI在系统JDK环境已设置好时，无需再部署。部署图如图6所示。

图6 部署图

# 5.1接口视角

## 5.1模块职责

客户端模块和服务器端模块的视图分别如图7和图8所示。客户端各层和服务器端各层的职责分别入表2和表3所示。

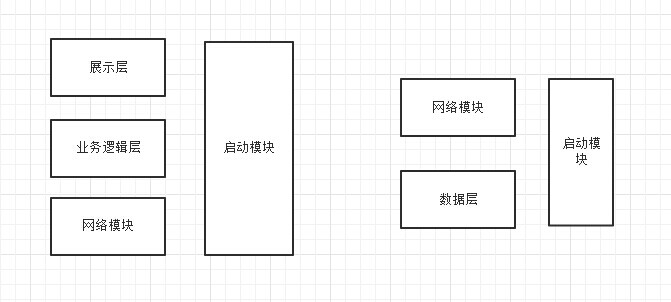


图7客户端模块视图 图8 服务器端模块视图

表2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 |
| 用户界面层 | 基于窗口的进销存系统客户端用户界面 |
| 业务逻辑层 | 对于用户界面的输入进行相应并进行业务处理逻辑 |
| 客户端网络模块 | 利用JAVA RMI机制查找RMI服务 |

表3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 |
| 数据层 | 负责数据的持久化及数据访问接口 |
| 服务器端网络模块 | 利用JAVA RMI机制开启RMI服务，注册RMI服务 |

每一层只是使用下方直接接触的层。层与层之间仅仅是通过接口的调用来完成的。层之间调用的接口如表4所示。

|  |  |  |
| --- | --- | --- |
| 接口 | 服务调用方 | 服务提供方 |
| BankManageBlService  FinancialReceiptBlService  RunningTableBlService  SetAccountBlService  ApproveBlService  StrategyBlService  AdminBlService  ClientBlService  StockBlService  SaleBlService  GoodsListBlService  GoodsTypeBlService  RepositoryCheckBlService  RepositoryExaminBlService  RepoReceiptBlService | 客户端展示层 | 客户端业务逻辑层 |
| BankManageBlService  FinancialReceiptBlService  RunningTableBlService  SetAccountBlService  ApproveBlService  StrategyBlService  AdminBlService  ClientBlService  StockBlService  SaleBlService  GoodsListBlService  GoodsTypeBlService  RepositoryCheckBlService  RepositoryExaminBlService  RepoReceiptBlService | 客户端业务逻辑层 | 服务器端数据层 |

**缺少示例**

## 5.2用户界面层的分解

系统存在22个用户界面：登陆界面，财务人员界面，银行账户管理界面，财务单据管理界面，账目管理界面，查看经营表格界面，审批单据界面，总经理界面，制定促销策略界面，查看日志界面，用户账户界面，管理员界面，库存人员界面，商品列表，商品分类，库存盘点，单据管理，库存查看，进货销售人员界面，销售单界面，进货单界面，客户管理界面。界面跳转如图10所示。

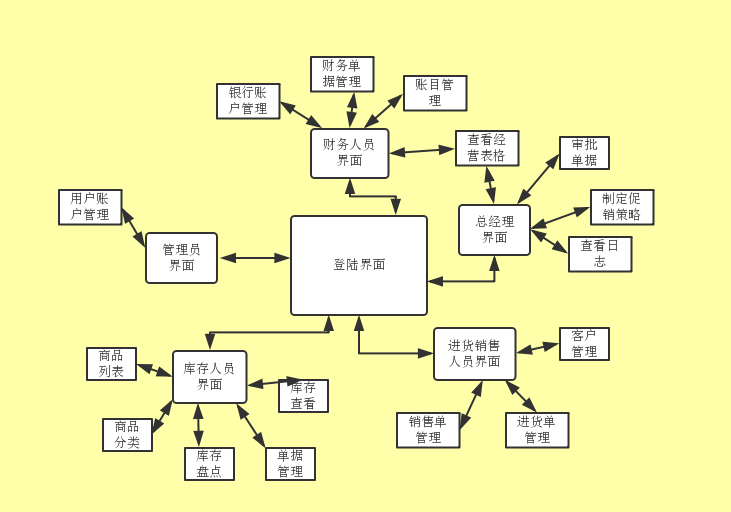


图10用户界面跳转图

用户界面类如图11所示。

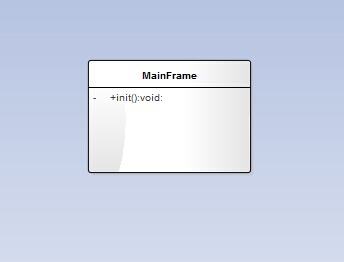


图11 用户界面类

### 5.2.1用户界面层模块的职责

如表5所示为用户界面层模块的职责。

表5用户界面层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| MainFrame | 界面Frame，负责界面的显示和界面的跳转 |

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6所示。

表6用户界面层模块的接口规范

|  |  |  |
| --- | --- | --- |
| MainFrame | 语法 | Init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示Frame以及LoginPanel |

用户界面层需要的服务接口如表7所示

表7用户界面层需要的服务接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| BusinessLogicService.LoginBlService | 登陆界面的业务逻辑接口 |
| BusinessLogicService.\*BlService | 每个界面都有相应的业务逻辑接口 |

### 5.2.3用户界面模块设计原理

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。业务逻辑层的设计如图12所示。

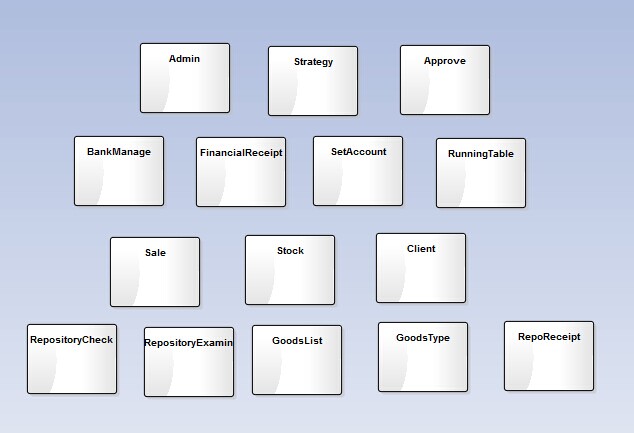


图12 业务逻辑层的设计

### 5.3.1业务逻辑层模块的职责

业务逻辑层模块的职责如表8所示。

表8，业务逻辑层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| BankManageBl | 负责实现对银行账户的操作，包括增删改查，转账入账 |
| FinancialReceiptBl | 负责实现对财务类单据的增加，查看操作，包括付款单，收款单，现金费用单 |
| RunningTableBl | 负责实现查看一段时间内某商品销售明细，查看一段时间内收支情况，查看各类单据，并提供红冲，导出表格功能 |
| SetAccountBl | 负责实现期初建账的一系列操作，包括继承，设置账目初始信息，存储初始信息 |
| StrategyBl | 负责实现处理促销策略所需要的服务 |
| ApproveBl | 负责实现单据审批所需要的服务 |
| AdminBl | 负责实现用户管理所需要的服务 |
| LogBl | 负责实现日志的处理所需要的服务 |
| ClientBL | 负责实现客户管理界面所需要的服务 |
| StockBL | 负责实现进货(退货)单管理界面所需要的服务 |
| SaleBL | 负责实现销售(退货)单管理界面所需要的服务 |
| GoodsListBl | 负责实现对于商品目录的管理的逻辑服务 |
| GoodsTypeBl | 负责实现对于商品类型的管理的服务 |
| RepositoryExaminBl | 负责实现库存盘点管理 |
| RepositoryCheckBl | 负责实现库存查看管理 |
| RepoReceiptBl | 负责实现库存单据管理 |

### 5.3.2业务逻辑层模块的接口规范

各BL模块接口规范如表9到23所示。

表9 BankManageBl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| BankManage.add | 语法 | Public Boolean add(BankVo vo) |
| 前置条件 | 无 |
| 后置条件 | 新增一个银行账户，刷新列表 |
| BankManage.delete | 语法 | Public Boolean delete(BankVo vo) |
| 前置条件 | 已选中列表某账户 |
| 后置条件 | 删除一个银行账户 |
| BankManage.modify | 语法 | Public Boolean modify(BankVo vo) |
| 前置条件 | 已选中列表某账户 |
| 后置条件 | 更新一个银行账户的信息 |
| BankManage.search | 语法 | Public BankVo[] search(String field) |
| 前置条件 | 无 |
| 后置条件 | UI列表中显示符合条件的结果 |
| BankManage.setup | 语法 | Public boolean setup(String ID) |
| 前置条件 | 无 |
| 后置条件 | 新建一个银行账户数据库 |
| BankManage.addData | 语法 | Public Boolean addData(String id,BankVo[] vo) |
| 前置条件 | 无 |
| 后置条件 | 在指定数据库添加银行账户 |
| BankManage.getData | 语法 | Public String getData(String id) |
| 前置条件 | 无 |
| 后置条件 | 返回指定帐目的银行账户数据库名称 |
| BankManage.update | 语法 | public void update(Observable o, Object arg) |
| 前置条件 | 单据被审批 |
| 后置条件 | 修改银行账户款额 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| BankManageDataService.find(String field) | 根据field查找持久化对象 |
| BankManageDataService.delete(BankPo po) | 删除单一持久化对象 |
| BankManageDataService.update(BankPo po) | 更新单一持久化对象 |
| BankManageDataService.insert(BankPo po) | 插入单一持久化对象 |
| BankManageDataService.init() | 初始化数据库 |
| BankManageDataService.finish() | 结束持久化数据库的使用 |
| BankManageDataService.finds() | 获得数据库所有数据 |
| BankManageDataService.setup(String ID) | 新建一个银行账户数据库 |
| BankManageDataService.getID(String account) | 返回指定帐目的数据库名称 |

表10 FinancialReceiptBl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| FinancialReceipt.Payment | 语法 | Public Boolean makePayment(PayVo vo) |
| 前置条件 | 无 |
| 后置条件 | 生成付款单，提交给总经理 |
| FinancialReceipt.Receive | 语法 | Public Boolean makeReceive(RecVo vo) |
| 前置条件 | 无 |
| 后置条件 | 生成收款单，提交给总经理 |
| FinancialReceipt.Cash | 语法 | Public Boolean makeCash(CashVo vo) |
| 前置条件 | 无 |
| 后置条件 | 生成现金费用单，提交给总经理 |
| FinancialReceipt.get | 语法 | Public ReceiptVo[] get(ReceiptConditionVo vo) |
| 前置条件 | 无 |
| 后置条件 | 返回符合条件的单据 |
| FinancialReceipt.creat | 语法 | Public boolean creat(ReceiptVo vo) |
| 前置条件 | 无 |
| 后置条件 | 生成财务类单据 |
| FinancialReceipt.update | 语法 | Public boolean update(ReceiptVo[] receipt) |
| 前置条件 | 无 |
| 后置条件 | 更新单据的审批信息 |
| FinancialReceipt.getReceipt | 语法 | Public ReceiptVo[] getReceipt() |
| 前置条件 | 无 |
| 后置条件 | 返回待审批的单据 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| FinancialReceiptDataService.addPay(PayPo po) | 添加一个持久化对象PayPo |
| FinancialReceiptDataService.addRec(RecPo po) | 添加一个持久化对象RecPo |
| FinancialReceiptDataService.addCash(CashPo po) | 添加一个持久化对象CashPo |
| FinancialReceiptDataService.update(Receipt  Po[] receipt) | 更新数据库持久化对象信息 |
| BankManageDataService.finds() | 获得银行账户列表 |
| Client.queryClient() | get客户列表 |
| Admin.getCurrentUsers | get当前操作员 |

表11 RunningTaBl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| RunningTable.SaleTable | 语法 | Public GoodsRecord[] getSaleTable ( SaleConditionVo vo) |
| 前置条件 | 无 |
| 后置条件 | 返回符合条件的商品销售记录 |
| RunningTable.ProgressTable | 语法 | Public Receipt[] getReceipt(ReceiptConditionVo vo) |
| 前置条件 | 无 |
| 后置条件 | 返回符合条件的单据 |
| RunningTable.FinanceTable | 语法 | Public profitVo getProfit(TimeConditionVo vo) |
| 前置条件 | 无 |
| 后置条件 | 返回符合条件的单据 |
| RunningTable.ProgressTable.CreditNote | 语法 | Public boolean CreditNote(ReceiptVo vo) |
| 前置条件 | 选中列表中某单据 |
| 后置条件 | 创建红冲单 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| FinancialReceipt.get(ReceiptConditionVo vo) | 给定条件返回单据（条件：时间区间，客户，操作员） |
| Stock.queryReceipt(ReceiptFilter[] filters) | 给定条件返回单据（条件：时间区间，客户，操作员，仓库） |
| RepoReceipt.find(ReceiptConditionVo Vo) | 给定条件返回单据（条件：时间区间，客户，操作员，仓库） |
| Sale.queryProduct(SaleConditionVo Vo) | 给定条件返回某一商品销售记录（条件：时间区间，商品名，客户，业务员，仓库） |
| FinancialReceipt.creat(ReceiptVo vo) | 生成财务类单据 |
| Sale.makeReceipt(SaleReceiptVO receipt) | 生成销售单据 |
| Stock.makeReceipt(StockReceiptVo receipt) | 生成进货单据 |

表12 SetAccountBl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| SetAccount.inherit | 语法 | Public void inherit(String name , String account Name ) |
| 前置条件 | 系统已有过往账目 |
| 后置条件 | 继承该账目所有信息 |
| SetAccount.setGoodsTy  Pe | 语法 | Public Boolean setGoodsType() |
| 前置条件 | 无 |
| 后置条件 | 建立（或更新）新的商品类型数据库 |
| SetAccount.addGoods | 语法 | Public Boolean addGoods(GoodsVo[] vo,String name) |
| 前置条件 | 已设置商品类型 |
| 后置条件 | 建立（或更新）新的商品信息数据库 |
| SetAccount.addClient | 语法 | Public Boolean addClient(ClientVo[] vo,String name) |
| 前置条件 | 已添加商品信息完毕 |
| 后置条件 | 建立(或更新)新的客户信息数据库 |
| SetAccount.addBank | 语法 | Public Boolean addBank(BankVo[] vo,String name) |
| 前置条件 | 已添加客户信息完毕 |
| 后置条件 | 建立（或更新）新的银行账户数据库 |
| SetAccount.setUp | 语法 | Public boolean setUp(String name) |
| 前置条件 | 新账目信息已填写完毕 |
| 后置条件 | 保存期初数据，更新系统默认账目 |
| SetAccount.checkGoodsType | 语法 | Public void checkGoodsType(String name) |
| 前置条件 | 无 |
| 后置条件 | 显示期初商品分类 |
| SetAccount.checkGoods | 语法 | Public GoodsVo[] checkGoods(String name) |
| 前置条件 | 无 |
| 后置条件 | 显示期初商品信息 |
| SetAccount.checkClient | 语法 | Public ClientVo[] checkClient(String name) |
| 前置条件 | 无 |
| 后置条件 | 显示期初客户信息 |
| SetAccount.checkBank | 语法 | Public BankVo[] checkBank(String name) |
| 前置条件 | 无 |
| 后置条件 | 显示期初银行账户信息 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| SetAccountDataService.SetUp(string[] name) | 保存所有期初数据库文件 |
| SetAccountDataService.Goods() | 查看期初所有商品信息 |
| SetAccountDataService.Bank() | 查看期初所有银行账户信息 |
| SetAccountDataService.Client() | 查看期初所有客户信息 |
| SetAccountDataService.GoodsType(  ) | 查看期初所有商品类别信息 |
| Client.makeTable(String account) | 传递账目名，新建客户数据库 |
| BankManage.setup(String  account ) | 传递账目名，新建银行账户数据库 |
| GoodsList.initialize(String account) | 传递账目名，新建商品分类数据库，商品信息数据库 |
| Client.queryTableName(String account) | 传递账目名，获得客户数据库名称 |
| BankManage.getData(String account) | 传递账目名，获得银行账户数据库名称 |
| GoodsList.getDatabase(String account) | 传递账目名，获得商品信息数据库名称 |

表13 ApproveBl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Approve.List.showList | 语法 | Public ArrayList<Receipt> showList() |
| 前置条件 | 用户登录，从数据层获得单据信息 |
| 后置条件 | 将单据信息传至界面层 |
| Approve.List.refresh | 语法 | Public ArrayList<Receipt> refresh() |
| 前置条件 | 用户选择刷新或单据列表发生了修改 |
| 后置条件 | 重新请求单据信息，并传至界面层 |
| Approve.List.pass | 语法 | Public Bool passList(ArrayList<Receipt>) |
| 前置条件 | 用户单据审批完毕准备提交审批结果 |
| 后置条件 | 将审批结果上传至服务器。 |
| Approve.Detail.showDetail | 语法 | Public Receipt show(int) |
| 前置条件 | 用户双击一条单据 |
| 后置条件 | 单据的详细信息被显示 |
| Approve.Detail.Pass | 语法 | Public Void confirm() |
| 前置条件 | 审批完单据内容 |
| 后置条件 | 将该单据定为审核通过 |
| Approve.Upload | 语法 | Public ArrayList<Receipt>Upload（） |
| 前置条件 | 审批完所有单据内容 |
| 后置条件 | 将审批后的单据传递 |
| Approve.Message | 语法 | Public String message(String UserName) |
| 前置条件 | 服务器反馈单据审批完成 |
| 后置条件 | 接到消息的用户去完成系统外的操作 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| ApproveDataService.List.showList() | 得到单据信息 |
| Stock.update(Observable o, Object arg) | 发送进货退货单（审批后） |
| Sale.update(Observable o, Object arg) | 发送销售退货单（审批后） |
| RepoReceipt.receiveReceipt(RepoReceiptVO) | 发送库存单据（审批后） |
| FinancialReceipt.update(ReceiptVo[] receipt) | 发送财务单据（审批后） |
| Stock.queryReceipt(ReceiptFilter[] filters) | 得到进货退货单 |
| Sale.queryReceipt(ReceiptFilter[] filters) | 得到销售退货单 |
| RepoReceipt.sendReceipt() | 得到库存单据 |
| FinancialReceipt.getReceipt() | 得到财务单据 |

表14 StrategyBl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Strategy.returnStrategy | 语法 | Public Puic ArrayList<Strategy> returnStrategy（int,ArrayList<Goods>） |
| 前置条件 | 进销人员输入顾客信息和购买的商品 |
| 后置条件 | 返回可使用的优惠条件供顾客选择 |
| Strategy.List.show | 语法 | Public ArrayList<Strategy> show（） |
| 前置条件 | 用户进入制定促销策略 |
| 后置条件 | 系统显示促销策略列表 |
| Strategy.List.Refresh | 语法 | Public ArrayList<Strategy> refresh（） |
| 前置条件 | 确定并提交促销策略，或手动刷新 |
| 后置条件 | 系统显示更新的促销策略列表 |
| Strategy.New.new | 语法 | Public Strategy new（） |
| 前置条件 | 界面层要求新建促销规则 |
| 后置条件 | 生成一个缺省促销规则的vo |
| Strategy.New.Confirm | 语法 | Public Bool confirm（） |
| 前置条件 | 制定完一条促销策略的条件，方式，时间 |
| 后置条件 | 检查输入合法性，之后传送strategy po至数据层 |
| Strategy.New.ExaminePrice | 语法 | Public Bool examinePrice（） |
| 前置条件 | 确认制定促销策略 |
| 后置条件 | 根据输入是否合法决定成功还是回去修改 |
| Strategy.New.ExamineDate | 语法 | Public Bool examineDate（） |
| 前置条件 | 确认制定促销策略 |
| 后置条件 | 根据输入是否合法决定成功还是回去修改 |
| Strategy.queryValidStrategy | 语法 | Public ArrayList<Strategy> queryValidStrategy(StrategyFilter[] filter) |
| 前置条件 | 需要获取满足条件的促销策略 |
| 后置条件 | 从促销策略列表中筛选出满足条件的策略列表 |
| 需要的服务（需接口） | | |
| 未找到 | 获得商品列表 | |
| StrategyDataService.confirm | 传送Strategy po | |
| StrategyDataService.List.show | 获得促销策略列表 | |

表15 AdminBl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Admin.show | 语法 | Public ArrayList<User> show（） |
| 前置条件 | 用户进入用户管理界面 |
| 后置条件 | 系统显示已注册用户列表 |
| Admin.new | 语法 | Public User new（） |
| 前置条件 | 界面层要求新建用户 |
| 后置条件 | 生成一个缺省用户的vo |
| Admin.delete | 语法 | Public Bool delete（User） |
| 前置条件 | 界面层要求新建用户 |
| 后置条件 | 生成一个缺省用户的vo |
| Admin.confirm | 语法 | Public Bool confirm（） |
| 前置条件 | 用户信息添加或修改完成 |
| 后置条件 | 将用户信息po传至数据层 |
| Admin.getCurrentUsers | 语法 | ArrayList<User> getCurrentUsers() |
| 前置条件 | 需要获得当前登录操作员 |
| 后置条件 | 返回当前登录操作员列表 |
| 需要的服务（需接口） | | |
| AdminDataService.confirm | 传送新的促销策略 | |
| AdminDataService.update | 请求用户列表 | |
| AdminDataService.show | 要求删除用户 | |

表16 ClienBl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Client.addClient | 语法 | Public void addClient(ClientVO client); |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 添加客户 |
| Client.deleteClient | 语法 | Public void deleteClient(String id); |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 查找是否存在客户ID，存在则删除客户，否则抛出异常 |
| Client.modifyClient | 语法 | Public void modifyClient(ClientVO newClient); |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 查找是否存在客户ID，存在则修改客户，否则抛出异常 |
| Client.queryClient | 语法 | Public Vector<ClientVO> queryClient(ClientFilter[] filters); |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 查找是否存在满足过滤条件的客户，如果存在则添加到类表中，否则返回空列表 |
| Client.queryClientById | 语法 | public ClientVO queryClientById(String id); |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 查找是否存在此客户编号，如果存在则添加到类表中，否则返回空列表 |
| Client.update | 语法 | public void update(Observable o, Object arg); |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 单据生效后修改客户的应收应付 |
| Client.makeTable | 语法 | public String makeTable(String account); |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 生成新的表 |
| Client.queryTableName | 语法 | Public String queryTableName(String account); |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 返回表名 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| ClientData.addClientData(ClientPO client) | 将新客户插入数据库 |
| ClientData.deleteClientData(ClientPO client) | 将客户从数据库移除 |
| ClientData.modifyClientData(ClientPO orignClient, ClientVO newClient) | 修改数据库中的客户数据 |
| ClientData.queryClientDataById(String id) | 根据ID查找数据库中的客户资料 |
| ClientData.queryClientData(ClientFilter[] filters) | 根据过滤条件查找客户 |

表17 StockBl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Stock.makeReceipt | 语法 | Public void makeReceipt(StockReceiptVO receipt); |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 生成进货（退货）单据 |
| Stock.queryReceipt | 语法 | Public Vector<StockReceiptVO> queryReceipt(ReceiptFilter[] filters); |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 返回符合过滤条件的单据 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| Client.queryClientById(String id) | 检查客户是否存在 |
| Admin.getCurrentUsers | 获取操作员 |
| GoodsList.checkGoodsExist(GoodsVO) | 检查商品是否存在 |
| 未找到Product.queryProdcut(ProductFilter[] filters) | 获取商品列表 |
| StockData.makeReceiptData(StockReceiptPO receipt) | 向数据库中插入进货（退货）单据 |
| StockData.queryReceiptData(ReceiptFilter[] filters) | 根据过滤条件查询单据 |

表18 SaleBl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Sale.makeReceipt | 语法 | Public void makeReceipt(SaleReceiptVO receipt); |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 生成销售（退货）单据 |
| Sale.queryReceipt | 语法 | Public Vector<SaleReceiptVO > queryReceipt(ReceiptFilter[] filters); |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 返回符合过滤条件的单据 |
| Sale.queryProduct | 语法 | Public Vector<ReceiptConditionVo> queryProduct(SaleConditionVo condition); |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 返回符合过滤条件的单据 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| Client.queryClientById(String id) | 检查客户是否存在 |
| Admin.getCurrentUsers() | 获取操作员 |
| Strategy.queryValidStrategy(StrategyFilter[] filter) | 获取有效的销售策略 |
| GoodsList.checkGoodsExist(GoodsVO) | 检查商品是否存在 |
| 未找到Product.getAmount(GoodsVO) | 获取商品数量（不确定是否是Product提供） |
| SaleData.makeReceiptData(SaleReceiptPO receipt) | 数据库中插入销售（退货）单据 |
| SaleData.queryReceiptData(ReceiptFilter[] filters) | 根据过滤条件查询单据 |

表19 GoodsListBl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| GoodsList.adds | 语法 | Public ResultMessage add（GoodsVO goods）; | |
| 前置条件 | 商品信息（见用例1扩展流程）输入符合输入规则 | |
| 后置条件 | 查找是否存在相应的商品，根据输入的信息返回是否添加成功 | |
| GoodsList.add | 语法 | Public ResultMessage add（ArrayList<GoodsVO> goods）; | |
| 前置条件 | 商品信息（见用例1扩展流程）输入符合输入规则 | |
| 后置条件 | 查找是否存在相应的商品，根据输入的信息返回一系列商品是否添加成功 | |
| GoodsList.delete | 语法 | Public ResultMessage delete（GoodsVO goods）; | |
| 前置条件 | 商品信息（见用例1扩展流程）输入符合输入规则 | |
| 后置条件 | 查找是否存在相应的商品及是否可以删除（见用例1特殊要求），根据输入的信息返回是否删除成功 | |
| GoodsList.iSearch | 语法 | Public GoodsVO iSearch（String goodsInfo）; | |
| 前置条件 | 商品信息（见用例1扩展流程）输入符合输入规则 | |
| 后置条件 | 查找是否存在相应的商品，根据输入的信息返回商品详细信息 | |
| GoodsList.eSearch | 语法 | Public GoodsVO eSearch（String id）; | |
| 前置条件 | 商品信息（见用例1扩展流程）输入符合输入规则 | |
| 后置条件 | 查找是否存在相应的商品，根据输入的信息返回商品详细信息 | |
| GoodsList.update | 语法 | Public ResultMessage update（GoodsVO goods）; | |
| 前置条件 | 商品信息（见用例1扩展流程）输入符合输入规则 | |
| 后置条件 | 查找是否存在相应的商品及是否可以修改其相应属性（见用例1特殊要求），根据输入的信息返回是否更新成功 | |
| GoodsList.setSignal | 语法 | Public ResultMessageVO setSignal（int amount， GoodsVO goods）; | |
| 前置条件 | 用户点击设置警戒值;商品信息输入符合格式 | |
| 后置条件 | 根据输入商品信息检验商品是否存在和是否可以设置警戒值，返回是否可以设置警戒值 | |
| GoodsList.getSignal | 语法 | Public SignalVO getSignal（int amount， GoodsVO goods）; | |
| 前置条件 | 用户点击设置警戒值;商品信息输入符合格式 | |
| 后置条件 | 根据输入商品信息检验商品是否存在和是否可以设置警戒值，返回是否可以设置警戒值 | |
| GoodsList.updateForSignal | 语法 | Public updateForSignal（）; | |
| 前置条件 | 销售人员在销售了商品之后通知库存人员 | |
| 后置条件 | 根据商品现有的数量及警戒值判断是否向总经理发送报警单据 | |
| GoodsList.sendSignal | 语法 | Public SignalVO sendSignal（）; | |
| 前置条件 | 发现库存少于报警值 | |
| 后置条件 | 向总经理发送报警单据 | |
| GoodsList.show | 语法 | Public ArrayList<GoodsVO> show（）; | |
| 前置条件 | 用户进入主界面点击显示列表 | |
| 后置条件 | 查找是否存在相应的商品，根据输入的信息返回是否添加成功 | |
| GoodsList.amount | 语法 | Public int amount（GoodsVO goods）; | |
| 前置条件 | 商品信息输入符合输入规则 | |
| 后置条件 | 查找是否存在相应的商品，根据输入的信息返回数量信息 | |
| GoodsList.addAmount | 语法 | Public ResultMessage addAmount（GoodsVO goods， int amount）; | |
| 前置条件 | 商品信息输入符合输入规则，数量符合库存 | |
| 后置条件 | 查找是否存在相应的商品，根据输入的信息返回是否添加成功 | |
| GoodsList.reduceAmount | 语法 | Public ResultMessage reduceAmount（GoodsVO goods， int amount）; | |
| 前置条件 | 商品信息输入符合输入规则 | |
| 后置条件 | 查找是否存在相应的商品，根据输入的信息返回是否添加成功 | |
| GoodsList.checkGoodsExist | 语法 | Public ResultMessage checkGoodsExist（GoodsVO goods）; | |
| 前置条件 | 商品信息（见用例1扩展流程）输入符合输入规则 | |
| 后置条件 | 查找是否存在相应的商品（及数量是否大于0;是否有该分类），根据输入的信息返回是否存在 | |
| GoodsList.checkSignalModifiable | 语法 | Public ResultMessage checkSignalModifiable（GoodsVO goods）; | |
| 前置条件 | 商品信息（见用例1扩展流程）输入符合输入规则 | |
| 后置条件 | 查找是否存在相应的商品，根据输入的信息返回是否可以修改其警戒值 | |
| GoodsList.checkBTeverhas | 语法 | Public ResultMessage checkBTeverhas（String goodsinfo）; | |
| 前置条件 | 商品信息输入符合输入规则 | |
| 后置条件 | 查找是否存在相应的商品，根据输入的信息返回是否可以修改其警戒值 | |
| GoodsList.RepoCheckGetGoodsList | 语法 | Public GoodsListPO getGoodsList（）; | |
| 前置条件 | 用户点击库存盘点 | |
| 后置条件 | 根据输入的信息返回商品列表 | |
| GoodsList.saleGetGoodsModeList | 语法 | Public GoodsListPO getGoodsModelList（）; | |
| 前置条件 | 用户点击库存盘点 | |
| 后置条件 | 根据输入的信息返回商品列表 | |
| GoodsList.initialize | 语法 | Public void initialize（String account）; | |
| 前置条件 | 期初建帐 | |
| 后置条件 | 按照GoodsType来初始化部分商品信息 | |
| GoodsList.getDatabase | 语法 | Public String getDatabase（String account）; | |
| 前置条件 | 无 | |
| 后置条件 | 返回商品信息数据库名 | |
| 需要的接口（需接口） | | | |
| 服务名 | | | 服务 |
| GoodsType.nodeCheck | | | 验证某类商品是否可以添加 |
| DatabaseFactory.getGoodsListDatabase | | | 得到GoodsList数据库的服务引用 |
| GoodsListDataService.initialize(GoodsListPO po) | | | 在数据库中更新GoodsListPO对象 |
| GoodsListDataService.insert(GoodsPO po) | | | 在数据库中插入某个GoodsPO对象 |
| GoodsListDataService.delete(GoodsPO po) | | | 在数据库中删除某个GoodsPO对象 |
| GoodsListDataService.update(GoodsPO po) | | | 在数据库中更新某个GoodsPO对象 |
| GoodsListDataService.eFind(GoodsPO po) | | | 在数据库中查找某个GoodsPO对象 |
| GoodsListDataService.iFind(GoodsPO po) | | | 在数据库中查找多个GoodsPO对象 |

表20 GoodsTypeBl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| GoodsType.add | 语法 | Public ResultMessage add（TreeNodeVO goods）; | |
| 前置条件 | 商品类型输入符合输入规则 | |
| 后置条件 | 查找是否可以添加相应的商品分类，根据输入的信息返回是否添加成功 | |
| GoodsType.delete | 语法 | Public ResultMessage delete（GoodsVO goods）; | |
| 前置条件 | 商品类型信息输入符合输入规则 | |
| 后置条件 | 查找是否存在相应的商品分类及是否可以删除（见用例2特殊要求），根据输入的信息返回是否删除成功 | |
| GoodsType.eSearch | 语法 | Public TreeNodeVO eSearch（String info）; | |
| 前置条件 | 商品类型信息输入符合输入规则 | |
| 后置条件 | 查找是否存在相应的商品分类，根据输入的信息聚焦某一节点 | |
| GoodsType.update | 语法 | Public ResultMessage update（TreeNodeVO goods）; | |
| 前置条件 | 商品类型信息输入符合输入规则 | |
| 后置条件 | 查找是否存在相应的商品分类及是否可以修改其相应属性（见用例2特殊要求），根据输入的信息返回是否更新成功 | |
| GoodsType.show | 语法 | Public ArrayList<TreeNodeVO> show（）; | |
| 前置条件 | 用户进入主界面时显示列表 | |
| 后置条件 | 查找是否存在相应的商品，根据输入的信息返回是否添加成功 | |
| GoodsType.find | 语法 | Public int find（TreeNodeVO goods）; | |
| 前置条件 | 商品信息输入符合输入规则 | |
| 后置条件 | 查找是否存在相应的商品，根据输入的信息返回数量信息 | |
| GoodsType.nodecheck | 语法 | Public ResultMessage nodecheck（String goodsinfo）; | |
| 前置条件 | 商品信息（见用例1扩展流程）输入符合输入规则 | |
| 后置条件 | 查找是否存在相应的商品，根据输入的信息返回是否存在 | |
| GoodsType.getDatabase | 语法 | Public String getDatabase（String account）; | |
| 前置条件 | 无 | |
| 后置条件 | 返回商品分类数据库名 | |
| GoodsType.initialize | 语法 | Public void initialize（String account）; | |
| 前置条件 | 起初建帐 | |
| 后置条件 | 建立部分分类 | |
| 需要的接口（需接口） | | | |
| 服务名 | | | 服务 |
| GoodsList.checkExist未找到，没参数 | | | 验证某中商品分类是否可以添加 |
| GoodsList.checkEverHas同上 | | | 验证某中商品分类是否可以删除 |
| GoodsList.initialize | | | 初始化商品信息 |
| DatabaseFactory.getGoodsTypeDatabase | | | 得到ArrayList<GoodsType>数据库的服务引用 |
| GoodsTypeDataService.initialize(GoodsTypePO po) | | | 在数据库中更新ArrayList<GoodsTypePO>对象 |
| GoodsTypeDataService.insert(TreeNodePO po) | | | 在数据库中插入某个TreeNodePO对象 |
| GoodsTypeDataService.delete(TreeNodePO po) | | | 在数据库中删除某个TreeNodePO对象 |
| GoodsTypeDataService.update(TreeNodePO po) | | | 在数据库中更新某个TreeNodePO对象 |
| GoodsTypeDataService.eFind(TreeNodePO po) | | | 在数据库中查找某个TreeNodePO对象 |
| GoodsTypeDataService.iFind(TreeNodePO po) | | | 在数据库中查找多个TreeNodePO对象 |

表21 RepositoryCheckBl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| RepositoryCheck.show | 语法 | Public RepositoryVO show（）; | |
| 前置条件 | 用户点击库存盘点之后 | |
| 后置条件 | 查找当前的库存信息，根据输入的信息返回库存对象 | |
| 需要的接口（需接口） | | | |
| 服务名 | | | 服务 |
| GoodsList.RepoCheckGetGoodsList | | | 取得商品列表 |
| DatabaseFactory.getRepositoryCheckDatabase | | | 得到ArrayList<RepositoryPO>数据库的服务引用 |
| RepositoryCheckDataService.initialize(ArrayList,RepositoryPO> po) | | | 在数据库中更新RepositoryCheckPO对象 |
| RepositoryCheckDataService.insert(RepositoryPO po) | | | 在数据库中插入某个RepositoryPO对象 |
| RepositoryCheckDataService.delete(RepositoryPO po) | | | 在数据库中删除某个RepositoryPO对象 |
| RepositoryCheckDataService.update(RepositoryPO po) | | | 在数据库中更新某个RepositoryPO对象 |
| RepositoryCheckDataService.eFind(RepositoryPO po) | | | 在数据库中查找某个RepositoryPO对象 |
| RepositoryCheckDataService.iFind(RepositoryPO po) | | | 在数据库中查找多个RepositoryPO对象 |

表22 RepositoyExaminBl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| RepositoyExamin.statistic | 语法 | Public ArrayList<InOutRepoVO> statistic（Date d）; | |
| 前置条件 | 用户点击库存查看之后 | |
| 后置条件 | 查找输入时间段的出入库信息，根据输入的信息返回出入库对象 | |
| 需要的接口（需接口） | | | |
| 服务名 | | | 服务 |
| StockReceiptVO.queryReceipt | | | 取得出入库商品单据 |
| DatabaseFactory.getInOutRepoDatabase | | | 得到ArrayList<InOutRepoPO>数据库的服务引用 |
| RepositoryExaminDataService.initialize(ArrayList,RepositoryPO> po) | | | 在数据库中更新ArrayList<InOutRepoPO>对象 |
| RepositoryExaminDataService.insert(InOutRepoPO po) | | | 在数据库中插入某个InOutRepoPO对象 |
| RepositoryExaminDataService.delete(InOutRepoPO po) | | | 在数据库中删除某个InOutRepoPO对象 |
| RepositoryExaminDataService.update(InOutRepoPO po) | | | 在数据库中更新某个InOutRepoPO对象 |
| RepositoryExaminDataService.eFind(InOutRepoPO po) | | | 在数据库中查找某个InOutRepoPO对象 |
| RepositoryExaminDataService.iFind(InOutRepoPO po) | | | 在数据库中查找多个InOutRepoPO对象 |

表23 RepoReceiptBl模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| RepoReceipt.input | 语法 | Public ResultMessageVO input（int amount， GoodsVO goods）; | |
| 前置条件 | 用户输入实际数量 | |
| 后置条件 | 根据输入商品数量检验商品是否存在和是否符合统计数量 | |
| RepoReceipt.find | 语法 | Public RepoReceiptVO find（ReceiptConditionVO condition）; | |
| 前置条件 | 财务人员查询单据 | |
| 后置条件 | 根据条件进行筛选单据 | |
| RepoReceipt.sendReceipt | 语法 | Public RepoReceiptVO sendReceipt（）; | |
| 前置条件 | 用户输入数量与统计数量不同 | |
| 后置条件 | 得到总经理反馈后修改数量 | |
| RepoReceipt.receiveReceipt | 语法 | Public void receiveReceipt（RepoReceiptVO）; | |
| 前置条件 | 总经理审批单据后返回单据 | |
| 后置条件 | 得到总经理反馈后修改数量 | |
| 需要的接口（需接口） | | | |
| 服务名 | | | 服务 |
| GoodsList.amount(Goods g) | | | 在商品列表中查看某种商品的数量 |
| DatabaseFactory.getRepoReceipt() | | | 得到ArrayList<repoReceiptPO>数据库的服务引用 |
| RepoReceiptDataService.initialize(ArrayList,repoReceiptPO> po) | | | 在数据库中更新ArrayList<repoReceiptPO>对象 |
| RepoReceiptDataService.insert(repoReceiptPO po) | | | 在数据库中插入某个repoReceiptPO对象 |
| RepoReceiptDataService.delete(repoReceiptPO po) | | | 在数据库中删除某个repoReceiptPO对象 |
| RepoReceiptDataService.update(repoReceiptPO po) | | | 在数据库中更新某个repoReceiptPO对象 |
| RepoReceiptDataService.eFind(repoReceiptPO po) | | | 在数据库中查找某个repoReceiptPO对象 |
| RepoReceiptDataService.iFind(repoReceiptPO po) | | | 在数据库中查找多个repoReceiptPO对象 |

## 5.4数据层的分解

### 5.4.1数据层模块的职责

数据层模块的职责如表24所示。

表24数据层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| BankManageDataService | 持久化数据库的接口，提供集体载入，增删改查，新建数据库功能 |
| BankManageDataServiceMySqlImpl | 基于MySql数据库的持久化数据的接口，提供集体载入，增删改查，新建数据库表功能 |
| FinancialReceiptDataService | 持久化数据库的接口，提供增，查功能 |
| FinancialReceiptDataServiceMySqlImpl | 基于MySql数据库的持久化数据的接口，提供增，改，查功能 |
| SetAccountDataService | 持久化数据库的接口，提供集体载入，集体查看，新建数据库表功能 |
| SetAccountDataServiceMySqlImpl | 基于MySql数据库的持久化数据的接口，提供提供集体载入，新建数据库表功能 |
| GoodsListDataService | 持久化商品列表的接口，提供集体载入，集体保存，个体增删改查服务 |
| GoosListDataServiceMySqlImpl | 基于MySql数据库的持久化数据库接口，提供集体载入，集体保存，个体增删改查服务 |
| GoodsTypeDataService | 持久化商品类型的接口，提供集体载入，集体保存，，个体增删改查服务 |
| GoodsTypeDataServiceMySqlImpl | 基于MySql数据库的持久化的数据库接口，提供集体载入，集体保存，个体增删改查服务 |
| RepositoryExaminDataService | 持久化商品列表的接口，提供集体载入，集体保存，，个体增删改查服务 |
| RepositoryExaminDataServiceMySqlImpl | 基于MySql数据库的持久化的数据库接口，提供集体载入，集体保存，个体增删改查服务 |
| RepositoryCheckDataService | 持久化商品列表的接口，提供集体载入，集体保存，个体增删改查服务 |
| RepositoryCheckDataServiceMySqlImpl | 基于MySql数据库的持久化的数据库接口，提供集体载入，集体保存，个体增删改查服务 |
| RepoReceiptDataService | 持久化商品列表的接口，提供集体载入，集体保存，个体增删改查服务 |
| RepoReceiptDataServiceMySqlImpl | 基于MySql数据库的持久化的数据库接口，提供集体载入，集体保存，个体增删改查服务 |
| 缺YS和DZH的 |  |

### 5.4.2数据层模块的接口规范

各数据层模块的接口如表25到

表25 BankManageData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（接口） | | |
| BankManageDataService.find | 语法 | Public BankPo[] find(String field) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按关键字进行查找，返回符合条件的BankPO |
| BankManageDataService.insert | 语法 | Public Boolean insert(BankPo po) throws RemoteException |
| 前置条件 | 同样ID的po在数据库中不存在 |
| 后置条件 | 数据库中添加该po |
| BankManageDataService.delete | 语法 | Public Boolean delete(BankPo po) throws RemoteException |
| 前置条件 | 同样ID的po在数据库中存在且未使用过 |
| 后置条件 | 数据库中删除该po |
| BankManageDataService.update | 语法 | Public Boolean update(BankPo po) throws RemoteException |
| 前置条件 | 同样ID的po在数据库中存在 |
| 后置条件 | 数据库中更新该po |
| BankManageDataService.init | 语法 | Public void init() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化数据库 |
| BankManageDataService.finish | 语法 | Public void finish() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |
| BankManageDataService.finds | 语法 | Public BankPo[] finds() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回数据库所有数据 |
| BankManageDataService.setup | 语法 | Public boolean setup(String account) |
| 前置条件 | 无 |
| 后置条件 | 新建一个数据库 |
| BankManageDataService.getID | 语法 | Public String getID(String account) |
| 前置条件 | 无 |
| 后置条件 | 返回指定帐目的的数据库名称 |

表26 FinancialReceiptData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| FinancialReceiptDataService.addPay | 语法 | Public Boolean addPay(PayPo po) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 数据库中添加该PayPo |
| FinancialReceiptDataService.addRec | 语法 | Public Boolean addRec(RecPo po) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 数据库中添加该RecPo |
| FinancialReceiptDataService.addCash | 语法 | Public Boolean addCash(CashPo po) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 数据库中添加该CashPo |
| FinancialReceiptdateservice.update | 语法 | Public boolean update(ReceiptPo[] receipt) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 数据库更新单据信息 |

表27 SetAccountData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| SetAccountDataService.Setup | 语法 | Public boolean setup(String[] name) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 创建期初数据库 |
| SetAccountDataService.Good | 语法 | Public GoodsPo[] Goods() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回所有GoodsPo |
| SetAccountDataService.Bank | 语法 | Public BankPo[] Bank() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回所有BankPo |
| SetAccountDataService.Client | 语法 | Public ClientPo[] client() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回所有ClientPo |

表28 ApproveData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ApproveDataService.List.showList | 语法 | Public ArrayList<Receipt> showList() throws RemoteException |
| 前置条件 | 逻辑层请求单据信息 |
| 后置条件 | 从数据库中读取单据信息传至逻辑层 |

表29 StrategyData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StrategyDataService.confirm | 语法 | Public void confirm（Strategy）throws RemoteException |
| 前置条件 | 接到逻辑层的strategy po |
| 后置条件 | 将po保存在数据库中 |
| StrategyDataService.List.show | 语法 | Public ArrayList<Strategy> show（）throws RemoteException |
| 前置条件 | 逻辑层请求促销列表 |
| 后置条件 | 从数据库中读取促销策略列表传至逻辑层。 |

表30 AdminData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| AdminDataService.confirm | 语法 | Public Puic strategy confirm（Strategy）throws RemoteException |
| 前置条件 | 接到逻辑层的strategy po |
| 后置条件 | 将po保存在数据库中 |
| AdminDataService.show | 语法 | Public ArrayList<User> show（）throws RemoteException |
| 前置条件 | 逻辑层请求用户列表 |
| 后置条件 | 从数据库中读取用户列表上传至逻辑层 |
| AdminDataService.delete | 语法 | Public Bool delete（User）throws RemoteException |
| 前置条件 | 逻辑层要求删除用户 |
| 后置条件 | 根据用户信息匹配相应用户并删除 |

表31 ClientData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ClientDataService.addClientData | 语法 | static void addClientData(ClientPO client) throws RemoteException |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 数据库中插入ClientPO对象 |
| ClientDataService.deleteClientData | 语法 | static void deleteClientData(ClientPO client) throws RemoteException |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 数据库中移除ClientPO对象 |
| ClientDataService.modifyClientData | 语法 | static void modifyClientData(ClientPO orignClient, ClientVO newClient) throws RemoteException |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 根据newClient修改orignClient |
| ClientDataService.queryClientDataById | 语法 | static ClientPO queryClientDataById(String id) throws RemoteException |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 若此客户编号存在则返回ClientPO对象，否则返回null |
| ClientDataService.queryClientData | 语法 | static Vector<ClientPO> queryClientData(ClientFilter[] filters) throws RemoteException |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 若有满足过滤条件的客户存在则返回ClientPO数组对象，否则返回null |

表32 StockData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockDataService.makeReceiptData | 语法 | static void makeReceiptData(StockReceiptPO receipt) throws RemoteException |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 数据库中插入StockReceiptPO 对象 |
| StockDataService.queryReceiptData | 语法 | static Vector<StockReceiptPO> queryReceiptData(ReceiptFilter[] filters) throws RemoteException |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 返回符合条件的单据 |

表33 SaleData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| SaleDataService.makeReceiptData | 语法 | static void makeReceiptData(SaleReceiptPO receipt) throws RemoteException |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 数据库中插入SaleReceiptPO 对象 |
| SaleDataService.queryReceiptData | 语法 | static Vector<SaleReceiptPO > queryReceiptData(ReceiptFilter[] filters) throws RemoteException |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 返回符合条件的单据 |

表34 GoodsListData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| 服务名 | | 服务 |
| DatabaseFactory.getGoodsList | 语法 | Public GoodsListPO getGoodsList（）throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 得到GoodsListPO数据库的服务引用 |
| GoodsListDataService.initialize | 语法 | Public ResultMesPO initialize(GoodsListPO po)  throws RemoteException |
| 前置条件 | 期初建帐 |
| 后置条件 | 在数据库中更新GoodsList对象 |
| GoodsListDataService.insert | 语法 | Public ResultMes insert(Goods g) throws RemoteException |
| 前置条件 | 添加商品 |
| 后置条件 | 在数据库中插入某个商品对象 |
| GoodsListDataService.delete | 语法 | Public ResultMes delete（Goods g） throws RemoteException |
| 前置条件 | 删除商品 |
| 后置条件 | 在数据库中删除某个商品对象 |
| GoodsListDataService.update | 语法 | Public ResultMes update（Goods g）throws RemoteException |
| 前置条件 | 修改商品信息 |
| 后置条件 | 在数据库中更新某个GoodsPO对象 |
| GoodsListDataService.eFind | 语法 | Public GoodsPO eFind（String id）throws RemoteException |
| 前置条件 | 查询商品信息 |
| 后置条件 | 返回相应商品信息 |
| GoodsListDataService.iFind | 语法 | Public ArrayList<GoodsPO> iFind(FilterPO f) throws RemoteException |
| 前置条件 | 模糊查询 |
| 后置条件 | 在数据库中查找多个符合条件的GoodsPO对象 |

表35 GoodsTypeData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| 服务名 | | 服务 |
| DatabaseFactory.getGoodsType | 语法 | Public ArrayList<TreeNodePO> getGoodsType（）throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 得到ArrayList<TreeNodePO>数据库的服务引用 |
| GoodsTypeDataService.initialize | 语法 | Public ResultMes initialize(ArrayList<TreeNodePO> po) throws RemoteException |
| 前置条件 | 期初建帐 |
| 后置条件 | 在数据库中更新ArrayList<TreeNode>对象 |
| GoodsTypeDataService.insert | 语法 | Public ResultMes insert(TreeNodePO g) throws RemoteException |
| 前置条件 | 添加商品分类 |
| 后置条件 | 在数据库中插入某个商品分类对象 |
| GoodsTypeDataService.delete | 语法 | Public ResultMes delete（TreeNodePO g）throws RemoteException |
| 前置条件 | 删除商品 |
| 后置条件 | 在数据库中删除某个商品对象 |
| GoodsTypeDataService.update | 语法 | Public ResultMes update（TreeNodePO g）throws RemoteException |
| 前置条件 | 修改商品分类信息 |
| 后置条件 | 在数据库中更新某个PO对象 |
| GoodsTypeDataService.eFind | 语法 | Public TreeNodePO eFind（String id）throws RemoteException |
| 前置条件 | 查询商品分类信息 |
| 后置条件 | 返回相应商品分类信息 |
| GoodsTypeDataService.iFind(FilterPO po) | 语法 | Public ArrayType<TreeNodePO> iFind(FilterPO f) throws RemoteException |
| 前置条件 | 模糊查询商品分类 |
| 后置条件 | 在数据库中查找多个符合条件的TreeNodePO对象 |

表36 RepositoryCheckData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| 服务名 | | 服务 |
| DatabaseFactory.getRepository | 语法 | Public ArrayList<RepositoryPO> getRepository（）throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 得到ArrayList<RepositoryPO>数据库的服务引用 |
| RepositoryCheckDataService.initialize | 语法 | Public ResultMes initialize(ArrayList<RepositoryPO>) throws RemoteException |
| 前置条件 | 期初建帐 |
| 后置条件 | 在数据库中更新ArrayList<RepositoryPO>对象 |
| RepositoryCheckDataService.insert | 语法 | Public ResultMes insert(RepositoryPO g) throws RemoteException |
| 前置条件 | 添加库存信息 |
| 后置条件 | 在数据库中插入某个库存信息对象 |
| RepositoryCheckDataService.delete | 语法 | Public ResultMes delete（RepositoryPO g）throws RemoteException |
| 前置条件 | 删除库存信息 |
| 后置条件 | 在数据库中删除某个库存对象 |
| RepositoryCheckDataService.update | 语法 | Public ResultMes update（RepositoryPO g） throws RemoteException |
| 前置条件 | 修改库存信息 |
| 后置条件 | 在数据库中更新某个库存对象 |
| RepositoryCheckDataService.eFind | 语法 | Public RepositoryPO eFind（String id） throws RemoteException |
| 前置条件 | 查询库存信息 |
| 后置条件 | 返回相应库存信息 |
| RepositoryCheckDataService.iFind(FilterPO po) | 语法 | Public ArrayType<RepositoryPO> iFind(FilterPO f) throws RemoteException |
| 前置条件 | 模糊查询库存 |
| 后置条件 | 在数据库中查找多个符合条件的RepositoryPO对象 |

表37 RepositoryExaminData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| 服务名 | | 服务 |
| DatabaseFactory.getInOutRepo | 语法 | Public ArrayList<InOutRepoPO> getInOutRepo（） throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 得到ArrayList<InOutRepoPO>数据库的服务引用 |
| InOutRepoExaminDataService.initialize | 语法 | Public ResultMes initialize() throws RemoteException |
| 前置条件 | 期初建帐 |
| 后置条件 | 在数据库中更新ArrayList<InOutRepoPO>对象 |
| RepositoryExaminDataService.insert | 语法 | Public ResultMes insert(InOutRepoPO g) throws RemoteException |
| 前置条件 | 添加库存信息 |
| 后置条件 | 在数据库中插入某个出入库信息对象 |
| RepositoryExaminDataService.delete | 语法 | Public ResultMes delete（InOutRepoPO g）throws RemoteException |
| 前置条件 | 删除库存信息 |
| 后置条件 | 在数据库中删除某个出入库对象 |
| RepositoryExaminDataService.update | 语法 | Public ResultMes update（InOutRepoPO g）throws RemoteException |
| 前置条件 | 修改库存信息 |
| 后置条件 | 在数据库中更新某个出入库对象 |
| RepositoryExaminDataService.eFind | 语法 | Public InOutRepoPO eFind（String id）throws RemoteException |
| 前置条件 | 查询库存信息 |
| 后置条件 | 返回相应出入库信息 |
| RepositoryExaminDataService.iFind(FilterPO po) | 语法 | Public ArrayType<InOutRepoPO> iFind(FilterPO f) throws RemoteException |
| 前置条件 | 模糊查询出入库信息 |
| 后置条件 | 在数据库中查找多个符合条件的InOutRepoPO对象 |

表38 RepoReceiptData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| 服务名 | | 服务 |
| DatabaseFactory.getRepoReceipt | 语法 | Public ArrayList<RepoReceipt> getRepoReceipt() throws RemoteException |
| 前置条件 | 单据审批 |
| 后置条件 | 得到ArrayList<repoReceiptPO>数据库的服务引用 |
| RepoReceiptDataService.initialize | 语法 | Public ResultMes initialize(ArrayList<repoReceiptPO> po) throws RemoteException |
| 前置条件 | 期初建帐 |
| 后置条件 | 在数据库中更新ArrayList<repoReceiptPO>对象 |
| RepoReceiptDataService.insert | 语法 | Public ResultMes insert(repoReceiptPO po) throws RemoteException |
| 前置条件 | 新生单据 |
| 后置条件 | 在数据库中插入某个repoReceiptPO对象 |
| RepoReceiptDataService.delete | 语法 | Public ResultMes delete(repoReceiptPO po) throws RemoteException |
| 前置条件 | 删除单据 |
| 后置条件 | 在数据库中删除某个repoReceiptPO对象 |
| RepoReceiptDataService.update | 语法 | Public ResultMes update(repoReceiptPO po) throws RemoteException |
| 前置条件 | 更新单据 |
| 后置条件 | 在数据库中更新某个repoReceiptPO对象 |
| RepoReceiptDataService.eFind | 语法 | Public ResultMes eFind(repoReceiptPO po) throws RemoteException |
| 前置条件 | 明确搜寻单据 |
| 后置条件 | 在数据库中查找某个repoReceiptPO对象 |
| RepoReceiptDataService.iFind | 语法 | Public ResultMes iFind(repoReceiptPO po) throws RemoteException |
| 前置条件 | 模糊搜寻单据 |
| 后置条件 | 在数据库中查找多个repoReceiptPO对象 |

# 6.信息视角

## 6.1持久化数据对象

库存管理人员：

GoodsPO：保存一批商品的属性：**编号、名称、型号、进价、零售价（见用例1要求8）、**

GoodsModelPO: 保存商品统计信息： **库存数量、最近进价、最近零售价、历史、是否可以赠送**

GoodsListPO：保存GoodsPO的list和HashMap<String(id) GoodsModelPO>

GoodsTypePO：保存TreeNode 的list

TreeNode:保存每个商品分类的信息：编号、父节点、子节点list

RepositoryPO:保存库存统计信息： 盘点的是库存快照，包括当天的各种商品的**名称，型号，库存数量，库存均价，批次，批号，出厂日期**

InOutRepoPO:保存一段时间的出入库商品金额及数量

RepoReceiptPO:保存单据信息（报损，报溢）：实际数量，统计数量，单据类型

GoodsRecePO：保存赠送单信息：商品list（商品名，价格。。），总价

ResultMes:保存操作是否成功

RepoFilterPO：保存检索过滤条件

进货销售人员：

ClientPO类包含客户的编号、分类（进货商、销售商）、级别（五级，一级普通用户，五级VIP客户）、姓名、电话、地址、邮编、电子邮箱、应收额度、应收、应付、默认业务员属性。

StockPO类包含进货(退货)单的单据编号，供应商，仓库，操作员，入库商品列表，备注，总额合计。其中入库商品列表包含的信息有：商品编号，数量，单价，金额，备注。

SalePO类包含销售(退货)单的单据编号，客户，业务员，操作员，仓库，出货商品清单，折让前总额，折让，使用代金卷金额，折让后总额，备注。出货商品清单包含的信息有：商品的编号，数量，单价，金额，商品备注。

财务人员：

BankPo类包含银行账户的名称，金额，备注

PayPo类包含单据编号，客户，操作员，总额，银行转账列表（银行账户，转账金额，备注），审批情况，回执情况

RecPo类包含信息同上

CashPo类包含单据编号，操作员，总额，条目清单（条目名，金额，备注），审批情况，回执情况

总经理：

UserPO类包好用户的用户名，密码，属性

StrategyPO类包含一个促销策略的满足条件，优惠方式和起止时间

持久化用户对象UserPO的定义如下：

**public** **class** UserPO {

**int** id;

String name;

String password;

UserRole role;

**public** UserPO(**int** i ,String n,String p,UserRole r){

id = i;

name = n;

password = p;

role = r;

}

**public** **int** getId() {

**return** id;

}

**public** String getName() {

**return** name;

}

**public** String getPassword() {

**return** password;

}

**public** UserRole getRole() {

**return** role;

}

}

持久化促销策略对象StrategyPO的定义如下：

**public** **class** StrategyPO {

Condition condition;

Treatment treatment;

TimePeriod timePeriod;

**public** Condition getCondition() {

**return** condition;

}

**public** **void** setCondition(Condition condition) {

**this**.condition = condition;

}

**public** Treatment getTreatment() {

**return** treatment;

}

**public** **void** setTreatment(Treatment treatment) {

**this**.treatment = treatment;

}

**public** TimePeriod getTimePeriod() {

**return** timePeriod;

}

**public** **void** setTimePeriod(TimePeriod timePeriod) {

**this**.timePeriod = timePeriod;

}

}

## 数据库表

**待定**