**快递物流系统**

**(Express Logistics System)**

**体系结构设计文档**



**学 院：南京大学软件学院**

**团 队：伪装者W小组**

**成 员：张词校 张家盛 王丽莉 魏彦淑**

**完成日期：2015年10月26日**

# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| 修改人员 | 日期 | 变更原因 | 版本号 |
| 全体成员 | 2015.10.25 | 完成初稿 | V0.1 |
| 魏彦淑、王丽莉 | 2015.10.26 | 对初稿进行汇总完善 | V1.0 |

# 1.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词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| ELS | 快递物流系统（Express Logistics System） | 无 |
| BL | 业务逻辑（BusinessLogic） | 无 |

## 1.3参考资料

[IEEE1471-2000]

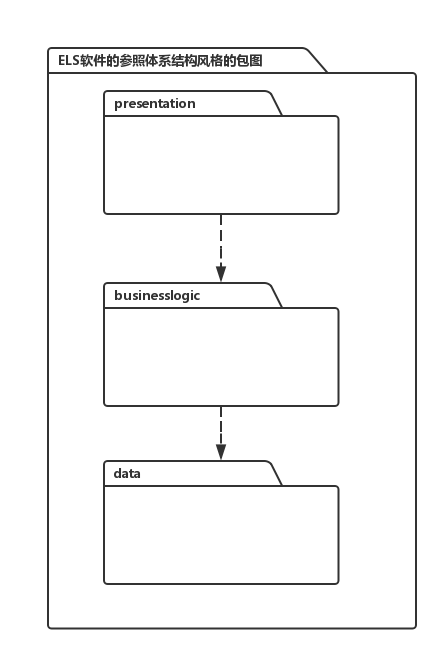
快递物流系统用例文档和快递物流系统需求规格说明。

# 2.产品概述

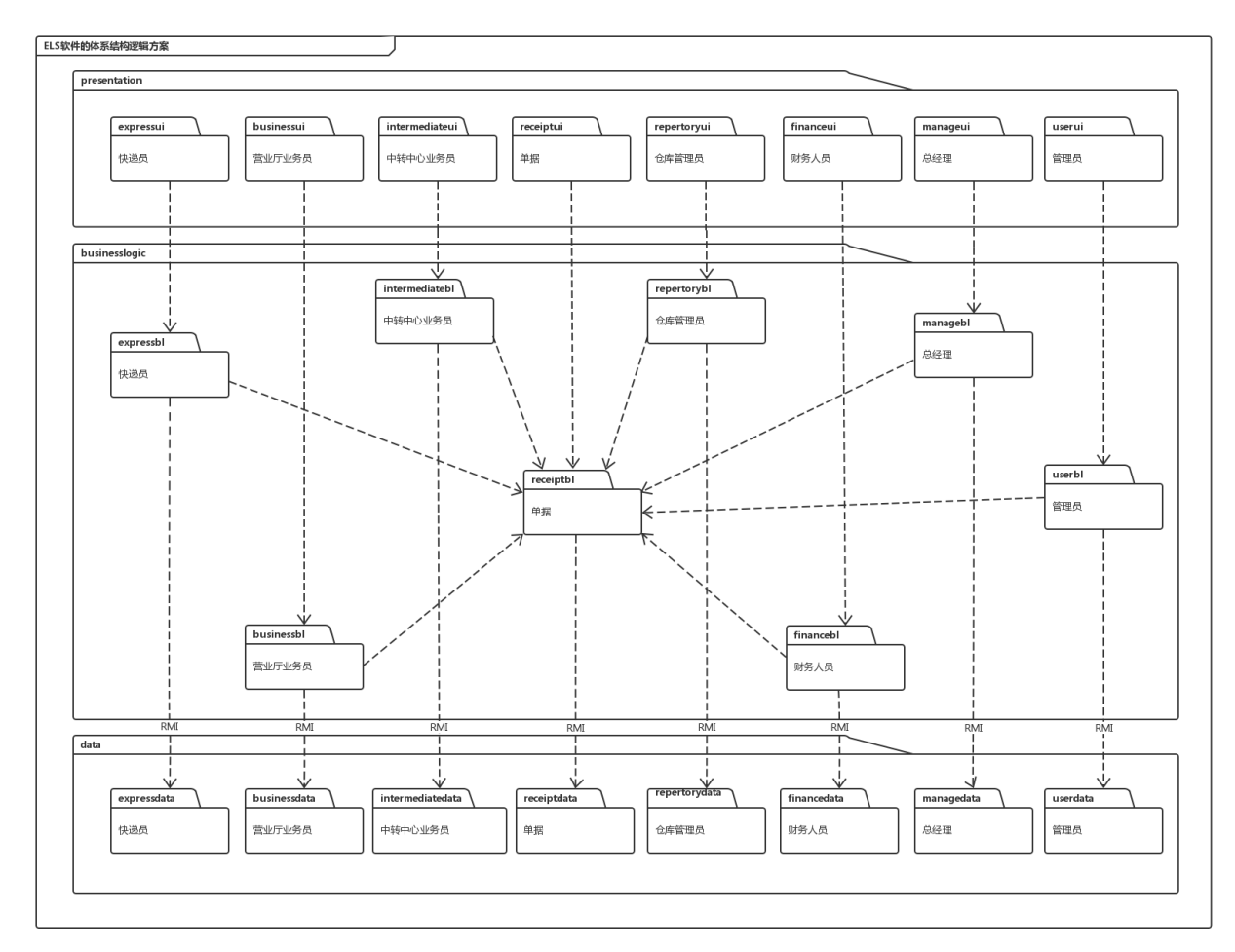
参考快递物流系统用例文档和快递物流系统需求规格说明对产品的概括描述。

# 3.逻辑视角

在快递物流系统中，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）能够很好地展示整个高层抽象。展示层包含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 ELS参照体系结构风格的包图表达逻辑视角**



**图2 ELS体系结构初始逻辑设计包图表达逻辑视角**

# 4.组合视角

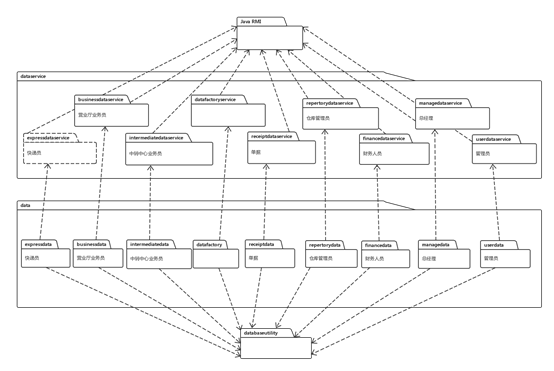
## 4.1开发包图

快递物流系统的最终开发包设计如表1所示

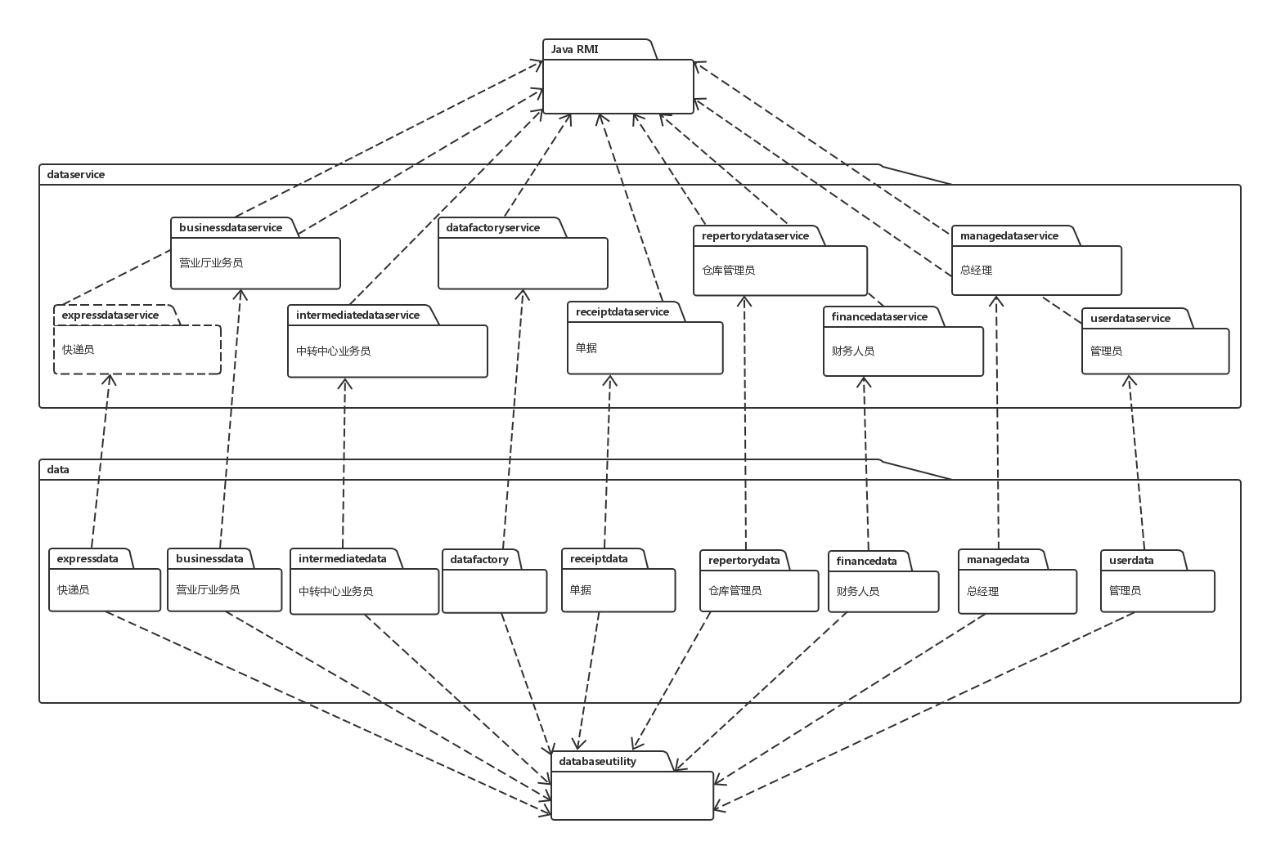
**表1 快递物流系统的最终开发包设计**

|  |  |
| --- | --- |
| 开发（物理）包 | 依赖的其他开发包 |
| mainui | expressui, businessui, intermediateui, financeui, repertoryui, manageui, userui, receiptui,vo |
| expressui | expressblservice,vo,界面类库包 |
| expressblservice |  |
| expressbl | expressblservice，expressdataservice，po,receiptbl,repertorybl |
| expressdataservice | Java RMI,po |
| expressdata | Java RMI,databaseutility,po |
| businessui | businessblservice ,vo,界面类库包 |
| businessblservice |  |
| businessbl | businessblservice, businessdataservice,po, receiptbl,repertorybl |
| businessdataservice | Java RMI,po |
| businessdata | Java RMI,po,databaseutility |
| intermediateui | intermediateblservice,vo,界面类库包 |
| intermediateblservice |  |
| intermediatebl | intermediateblservice, intermediatedataservice,po, receiptbl,repertorybl |
| intermediatedataservice | Java RMI,po |
| intermediatedata | Java RMI,po,databaseutility |
| financeui | financeblservice,vo,界面类库包 |
| financeblservice |  |
| financebl | financeblservice,financedataservice,po,receiptbl,userbl,managebl |
| financedataservice | Java RMI,po |
| financedata | Java RMI,po,databaseutility |
| repertoryui | repertoryblservice,vo,界面类库包 |
| repertoryblservice |  |
| repertorybl | repertoryblservice,repertorydataservice,po, receiptbl |
| repertorydataservice | Java RMI,po |
| repertorydata | Java RMI,po,databaseutility |
| manageui | manageblservice,vo,界面类库包 |
| manageblservice |  |
| managebl | manageblservice,managedataservice,po |
| managedataservice | Java RMI,po |
| managedata | Java RMI,po,databaseutility |
| userui | userblservice,vo,界面类库包 |
| userblservice |  |
| userbl | userblservice, userdataservice,UserPO |
| userdataservice | Java RMI,po |
| userdata | Java RMI,po,databaseutility |
| receiptui | receiptblservice, vo,界面类库包 |
| receiptblservice |  |
| receiptbl | receiptblservice,receiptdataservice,po,userbl,intermediabl |
| receiptdataservice | Java RMI,po |
| receiptdata | Java RMI, po,databaseutility |
| vo |  |
| po |  |
| utilitybl |  |
| 界面类库包 |  |
| Java RMI |  |
| databaseutility | JDBC,io类库包 |

快递物流系统客户端开发包图如图3所示，服务器端开发包图如图4所示



**图3 快递物流系统客户端开发包图**

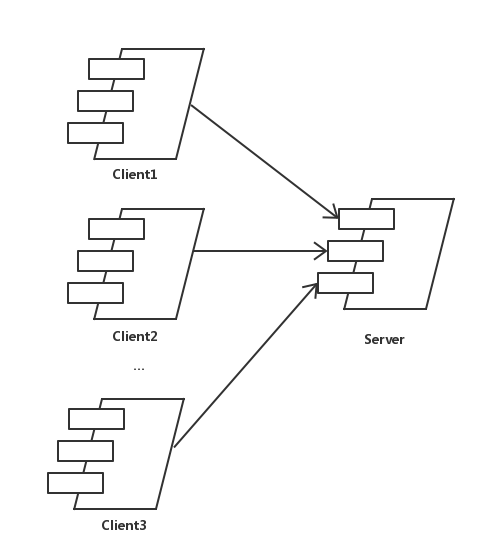


**图4 快递物流系统服务器端开发包图**

## 4.2运行时进程

在快递物流系统中，会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机器上运行，服务器端进程在服务器端机器上运行。

进程图如图5所示。

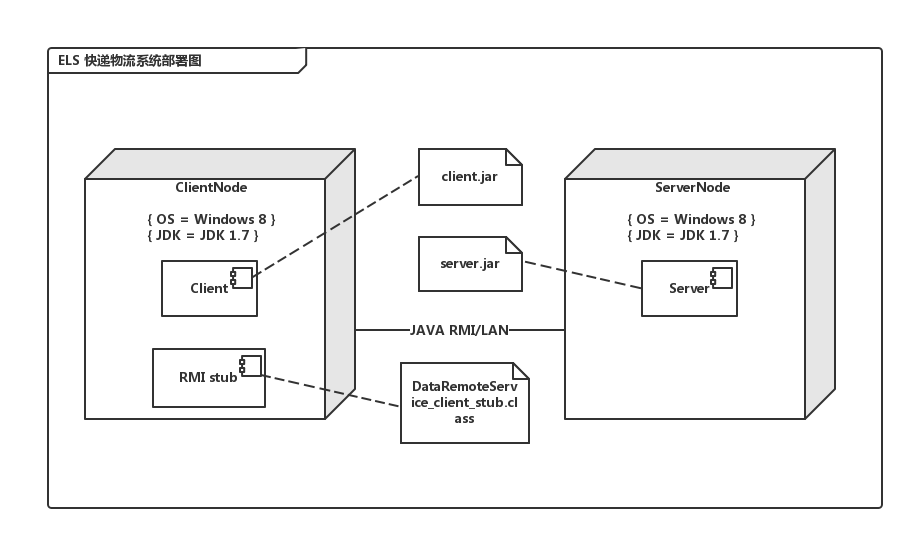


**图5 快递物流系统进程图**

## 4.3物理部署

快递物流系统中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。在客户端节点上，还要部署RMIStub构件。由于Java RMI构件属于JDK7.0的一部分。所以，在系统JDK环境已经设置好的情况下，不需要再独立部署。

部署图如图6所示。



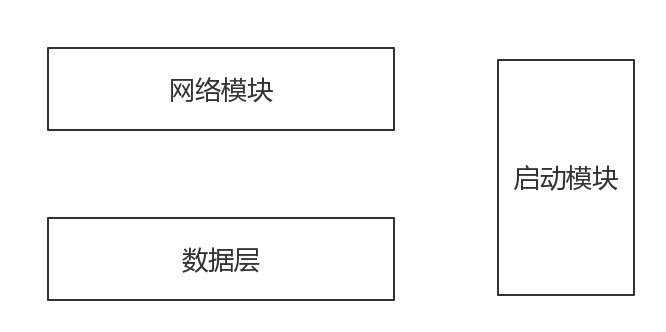
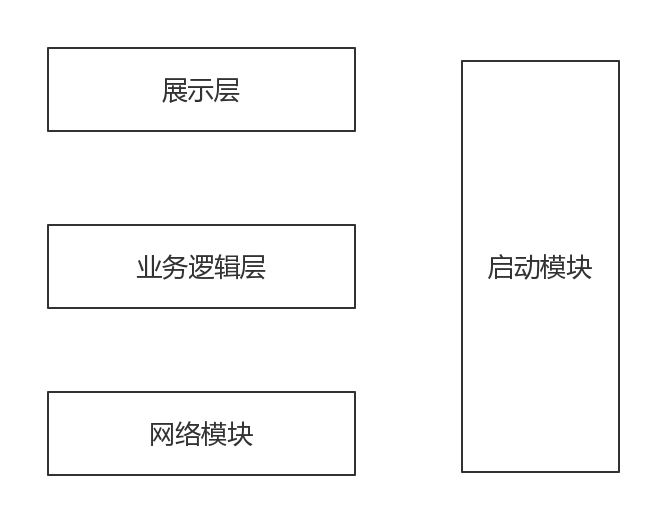
**图6 快递物流系统部署图**

# 5．接口视角

## 5.1模块的职责

客户端模块和服务器模块视图分别如图7和图8所示。客户端各层和服务器端各层的职责分别如表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 | 基于窗口的快递物流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响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**

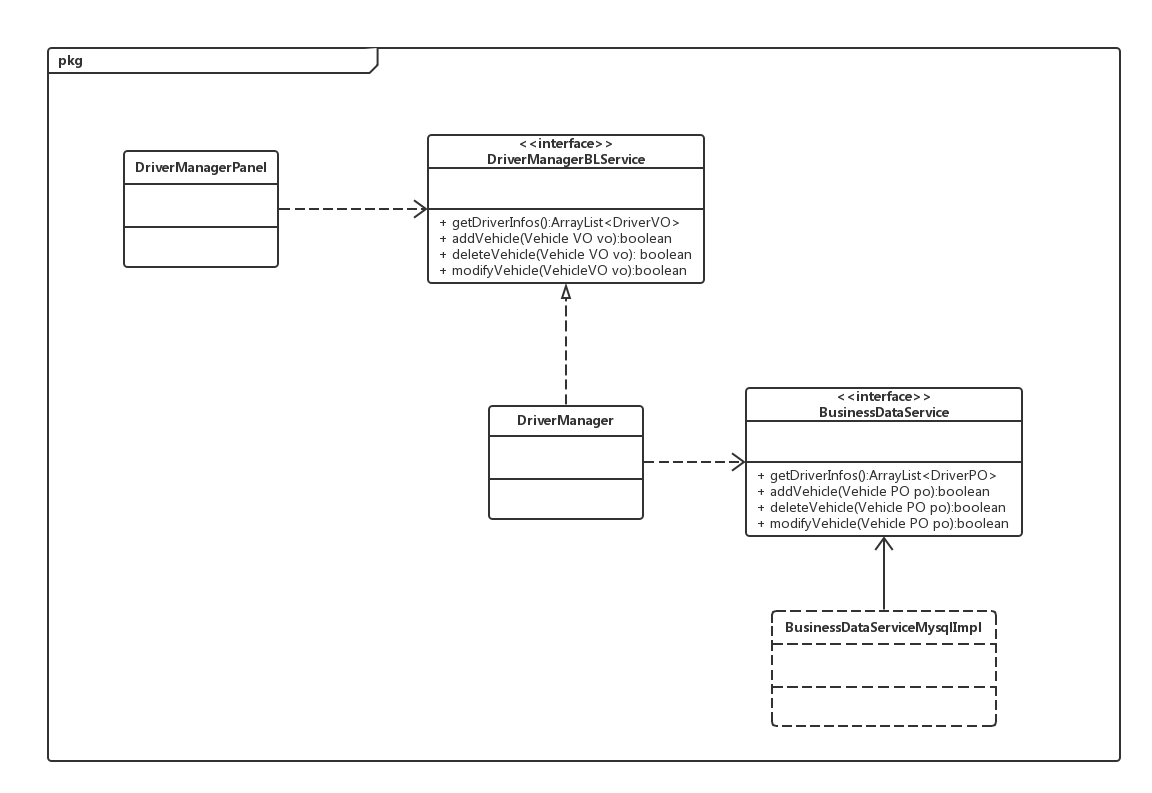
|  |  |
| --- | --- |
| 层 | 职责 |
| 启动模块 | 负责初始化网络通信机制，启动用户界面 |
| 数据层 | 负责数据的持久化及数据访问接口 |
| 服务器端网络模块 | 利用Java RMI机制开启RMI服务，注册RMI服务 |

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。

**表4 层之间调用的接口**

|  |  |  |
| --- | --- | --- |
| 接口 | 服务调用方 | 服务提供方 |
| ExpressBLService  BusinessBLService  IntermediaBLService  UserBLService  ManageBLService  FinanceBLService  ReceiptBLService  RepertoryBLService | 客户端展示层 | 客户端业务逻辑层 |
| ExpressDataService  BusinessDataService  IntermediaDataService  UserDataService  ManageDataService  FinanceDataService  ReceiptDataService  RepertoryDataService | 客户端业务逻辑层 | 服务器端数据层 |

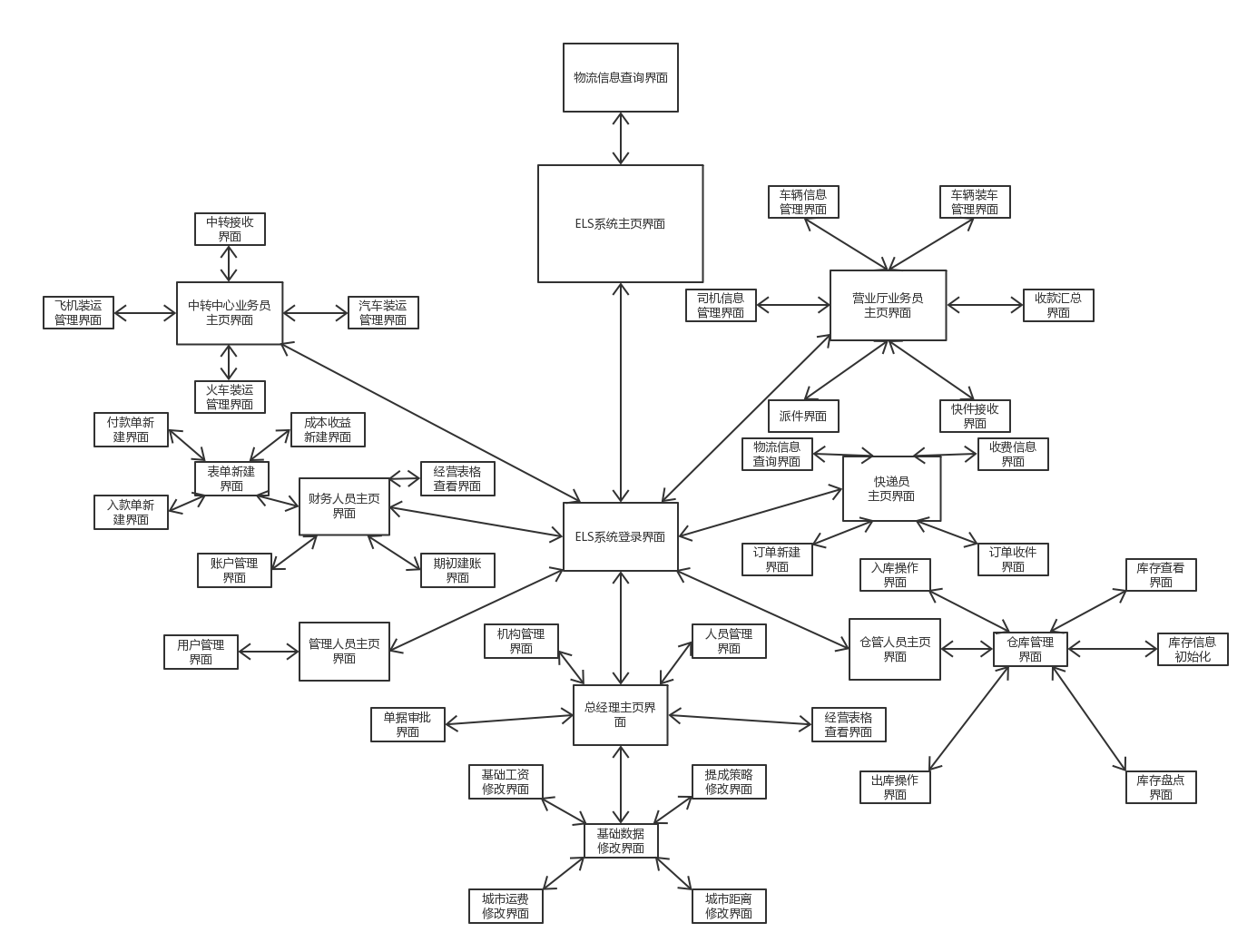
借用司机信息管理用例来说明层之间的调用，如图9所示。每一层之间都是由上层依赖了一个接口（需接口），而下层实现这个接口（供接口）。DriverManagerBLService提供了DriverManager界面所需要的所有业务逻辑功能。BusinessDataService提供了对数据库的增、删、改、查等操作。这样的实现就大大降低了层与层之间的耦合。



**图9 司机信息管理用例层之间调用的接口**

## 5.2用户界面层的分解

根据需求，系统存在共40个界面：ELS系统主页界面、物流信息查询界面、ELS系统登录界面、总经理主页界面、机构管理界面、单据审批界面、人员管理界面、经营表格查看界面、基础数据修改界面、基础工资修改界面、提成策略修改界面、城市运费修改界面、城市距离修改界面、管理人员主页界面、用户管理界面、财务人员主页界面、经营表格查看界面、期初建账界面、账户管理界面、表单新建界面、成本收益表新建界面、付款单新建界面、入款单新建界面、快递员主页界面、收费信息界面、物流信息查询界面、订单新建界面、订单收件界面、营业厅业务员主页界面、司机信息管理界面、车辆信息管理界面、派件界面、快件接收界面、车辆装车管理界面、收款汇总界面、中转中心业务员主页界面、汽车装运管理界面、火车装运管理界面、飞机装运管理界面、中转接收界面。界面跳转图如图10所示。



**图10 用户界面跳转图**

服务器端和客户端的用户界面设计接口是一致的，只是具体的页面不一样。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，负责界面的显示和跳转 |
| ExpressFrame | 快递员部分界面Frame，负责快递员部分界面的显示与跳转 |
| BusinessFrame | 营业厅部分界面Frame，负责营业厅部分界面的显示和跳转 |
| IntermediateFrame | 中转中心部分界面Frame，负责中转中心部分界面的显示和跳转 |
| ReceiptFrame | 单据部分界面Frame，负责单据部分界面的显示 |
| FinanceFrame | 财务部分界面Frame，负责财务部分界面的显示和跳转 |
| RepertoryFrame | 仓库部分界面Frame，负责仓库部分界面的显示与跳转 |
| ManageFrame | 总经理部分界面Frame，负责总经理部分界面的显示和跳转 |
| UserFrame | 用户部分界面Frame，负责用户登录界面和管理员增删改查用户界面的显示和跳转 |

### 5.2.2用户界面模块的接口规范

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 | Frameinit（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 显示Frame |
| ExpressFrame | 语法 | ExpressFrameinit（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 显示快递员部分界面的Frame以及订单新建，订单完成，订单查询和订单确认收费的panel |
| BusinessFrame | 语法 | BusinessFrameinit（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 显示营业厅部分界面的Frame以及装车管理，车辆和司机的信息管理，派件和收费汇总的panel |
| IntermediateFrame | 语法 | IntermediateFrameinit（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 显示中转中心部分界面的Frame以及飞机，火车，汽车装运管理和中转接收的panel |
| ReceiptFrame | 语法 | ReceiptFrameFrameini（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 显示单据部分界面的Frame以及各种单据查增删改的panel |
| FinanceFrame | 语法 | FinanceFrameinit(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示财务界面的Frame以及表单新建，报表查看，期初建账和账户管理的panel |
| RepertoryFrame | 语法 | RepertoryFrameinit(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示仓库界面的Frame以及商品出、入库管理,库存初始化，库存报警，库存查看和库存盘点的panel |
| ManageFrame | 语法 | ManageFrameinit(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示总经理界面的Frame以及机构管理，人员管理，基础数据设置，财务报表查看的panel |
| UserFrame | 语法 | UserLogInFrameinit（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 显示用户登录界面的Frame以及跳转 |
| UserFrame | 语法 | UserManageFrameinit(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示管理员界面的Frame以及账号管理的panel |

用户界面层需要的服务接口如表7所示。

**表7 用户界面层模块需要的服务接口**

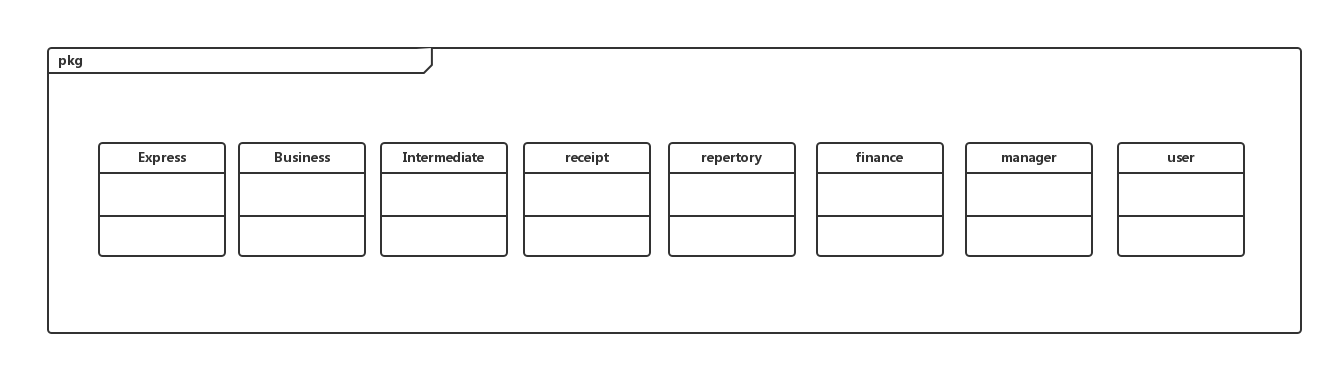
|  |  |
| --- | --- |
| **服务名** | **服务** |
| businesslogicservice.userblservice.UserBLService | 用户信息的增删改查的业务逻辑接口和登录界面的业务逻辑接口 |
| businesslogicservice.businessblservice.DriverManagerBLService  businesslogicservice.businessblservice.VehicleManagerBLService  businesslogicservice.businessblservice.EnVehicleBLService  Businesslogicservice.businessblservice.GatheringBLService  businesslogicservice.businessblservice.DistributeOrderBLService  businesslogicservice.businessblservice.AcceptCargoBLService | 营业厅司机信息的增删改查的业务逻辑接口  营业厅车辆信息的增删改查的业务逻辑接口  营业厅车辆装车安排及的业务逻辑接口  营业厅每日收款单列表提交的业务逻辑接口  营业厅派件的业务逻辑接口  营业厅接收货物的业务逻辑接口 |
| Businesslogicservice.expressblservice.LogisticQueryBLService  Businesslogicservice.expressblservice.AddOrderBLService  Businesslogicservice.expressblservice.ChargeCollectionBLService  Businesslogicservice.expressblservice.ReceiptOrderBLService | 快递员查询订单的业务逻辑接口  快递员输入订单的业务逻辑接口  快递员进行收费信息汇总的业务逻辑接口  快递员收件信息输入的业务逻辑接口 |
| businesslogicservice.financeblservice.CollectionReceiptBLService  businesslogicservice.financeblservice.PaymentReceiptBLService  businesslogicservice.financeblservice.CostIncomeReceiptBLService  businesslogicservice.financeblservice.BusinessstatementReceiptBLService  businesslogicservice.financeblservice.InitialStockBLService  businesslogicservice.financeblservice.AccountBLService | 结算管理的业务逻辑接口  新建付款单的业务逻辑接口  新建成本收益表的业务逻辑接口  查看经营情况表的业务逻辑接口  期初建账的业务逻辑接口  账户的增删改查的逻辑接口 |
| businesslogicservice.intermediateblservice.EnplaningBLService  businesslogicservice.intermediateblservice.EntrainingBLService  businesslogicservice.intermediateblservice.EntruckingBLService  businesslogicservice.intermediateblservice.TransferingBLService | 飞机装运管理的业务逻辑接口  火车装运管理的业务逻辑接口  汽车装运管理的业务逻辑接口  中转接收的业务逻辑接口 |
| businesslogicservice.repertoryblservice.RepertoryBLService | 出库、入库、库存查看、库存盘点、库存信息初始化的业务逻辑接口 |
| businesslogicservice.manageblservice.OrganizationBLService  businesslogicservice.manageblservice.PerWageBLService  businesslogicservice.manageblservice.BasicSalaryBLService  businesslogicservice.manageblservice.CityDistanceBLService  businesslogicservice.manageblservice.CostBLService | 机构信息的增删改查的业务逻辑接口  每次工资信息的增删改查的业务逻辑接口  基础月薪信息的增删改查的业务逻辑接口  城市距离信息的增删改查的业务逻辑接口  运费系数信息的增删改查的业务逻辑接口 |

### 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。例如，userbl模块负责处理登录界面的业务逻辑，financebl负责处理与财务管理相关的界面业务逻辑。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 |
| expressbl | 负责实现快递员对订单操作界面所需要的服务 |
| businessbl | 负责实现营业厅业务员界面所需要的服务 |
| intermediatebl | 负责实现中转中心业务员转运界面所需要的服务 |
| receiptbl | 负责实现各种单据增删改查以及提交界面所需要的服务 |
| repertorybl | 负责实现仓库管理界面所需要的服务 |
| financebl | 负责实现财务人员的财务管理界面所需要的服务 |
| managebl | 负责实现总经理管理人员机构，进行基础设置界面所需要的服务 |
| userbl | 负责实现管理员管理用户界面所需要的服务 |

### 5.3.2业务逻辑层模块的接口规范

expressbl模块的接口规范如表9所示。

**表9 expressbl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Express.logisticQuery.query | 语法 | public OrderVO query(long orderNum) | |
|  | 前置条件 | 快递员经过验证有查询订单的权利，输入了有效的订单号 | |
| 后置条件 | 系统查找订单是否存在，如果找到就返回订单信息 | |
| Express.AddOrder.addOrder | 语法 | public boolean addOrder(OrderVO vo) | |
| 前置条件 | 快递员经过验证有添加订单的权利，输入了有效的待添加的订单信息 | |
| 后置条件 | 系统更新对应的订单信息，并根据更新结果返回 | |
| Express.ChargeCollection.getChargeInfo | 语法 | public ExpressVO getChargeInfo() | |
| 前置条件 | 快递员经过验证有汇总收费的权利，并要求查看当前收费情况 | |
| 后置条件 | 系统更新快递员信息，并更具更新返回结果 | |
| Express.ChargeCollection.chargeCollection | 语法 | public boolean chargeCollection(ExpressVO vo) | |
| 前置条件 | 快递员经过验证有汇总收费的权利，并输入了订单号 | |
| 后置条件 | 系统根据输入的订单号返回对应的订单信息 | |
| Express.ReceiptOrder.getOrderInfo | 语法 | public OrderVO getOrderInfo(long orderNum) | |
| 前置条件 | 快递员经过验证有确认订单的权利，并要求查看订单信息 | |
| 后置条件 | 系统根据输入的 | |
| Express.ReceiptOrde.receiptOrder | 语法 | public boolean receiptOrder(OrderVO vo) | |
| 前置条件 | 快递员经过验证有确认订单的权利，并要求确认订单 | |
| 后置条件 | 系统根据输入的订单接收信息更新订单，并返回更新结果 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| DataFactory.getOrderInfo(long orderNum) | | | 得到订单信息 |
| DataFactory.addOrder(OrderPO po) | | | 新增订单 |
| ExpressDataService.getBaseCost() | | | 得到运费计算的单位价格 |
| ExpressDataService.chargeCollection(ExpressPO po) | | | 更新快递员的收费信息总额 |
| DataFactory.receiptOrder(OrderPO po) | | | 完成订单 |

businessbl模块的接口规范如表10所示。

**表10 businessbl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Business.VehicleManager.modifyVehicle | 语法 | public boolean deleteVehicle(VehicleVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并要求改正指定车辆信息 | |
| 后置条件 | 系统删除对应车辆信息，并返回操作结果 | |
| Business.VehicleManager.addVehicle | 语法 | public boolean addVehicle(VehicleVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并输入了正确的车辆信息 | |
| 后置条件 | 系统根据输入的车辆信息更新车辆列表，并返回更新结果 | |
| Business.VehicleManager.getVehicleInfo | 语法 | public ArrayList<VehicleVO>getVehicleInfo(String ID) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并输入了车辆编号 | |
| 后置条件 | 系统返回已有车辆的信息 | |
| Business.VehicleManager.deleteVehicle | 语法 | public boolean deleteVehicle(VehicleVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并要求删除指定车辆信息 | |
| 后置条件 | 系统删除对应车辆信息，并返回操作结果 | |
| Business.AcceptCargo.acceptCargo | 语法 | public boolean acceptCargo( OrderAcceptReceiptVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行接收货物的权利，并输入了所有货物的订单号 | |
| 后置条件 | 系统生成新的营业厅到达单并更新营业厅到达单列表 | |
| Business.Gathering.getChargeInfo | 语法 | public ArrayList<ExpressVO>getChargeInfo() | |
| 前置条件 | 营业厅业务员经过验证有进行收款汇总的权利，并要求查看所有营业厅业务员的收款与订单信息 | |
| 后置条件 | 系统返回该营业厅所有快递员的信息（订单与收款） | |
| Business.Gathering.gathing | 语法 | public double gathering(GatheringReceiptVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行收款汇总的权利，并要求进行收款汇总任务 | |
| 后置条件 | 系统汇总收款并返回结果 | |
| Business.TruckManager.autoTruckLoading | 语法 | public String autoTruckLoading() | |
| 前置条件 | 营业厅业务员经过验证有进行装车管理的权利，并要求进行装车管理任务 | |
| 后置条件 | 系统自动根据已有信息生成装车单，并更新到装车单列表 | |
| Business.TruckManager.getFreeVehicles | 语法 | public ArrayList<VehicleVO> getFreeVehicles() | |
| 前置条件 | 营业厅业务员经过验证有进行装车管理的权利，并要求查看当前的空闲车辆 | |
| 后置条件 | 系统返回当前的空闲车辆的信息 | |
| Business.TruckManager.getTransferOrders | 语法 | public ArrayList<OrderVO>getTransferOrders() | |
| 前置条件 | 营业厅业务员经过验证有进行装车管理的权利，并要求查看当前待转发的订单 | |
| 后置条件 | 系统返回订单待转发的订单信息 | |
| Business.DistributeOrder.distributeOrder | 语法 | public String distributeOrder() | |
| 前置条件 | 营业厅业务员经过验证有进行分配订单的权利，并要求进行分配订单任务 | |
| 后置条件 | 系统根据已有信息生成装车单，并返回分配结果 | |
| Business.DistributeOrder.getExpressInfos | 语法 | public ArrayList<ExpressVO> getExpressInfos() | |
| 前置条件 | 营业厅业务员经过验证有进行分配订单的权利，并要求查看该营业厅所有快递员的信息 | |
| 后置条件 | 系统返回该营业厅所有快递员的信息 | |
| Business.DistributeOrder.getSendOrde | 语法 | public ArrayList<OrderVO>getSendOrder() | |
| 前置条件 | 营业厅业务员经过验证有进行分配订单的权利，并要求查看该营业厅所有待派发的订单信息 | |
| 后置条件 | 系统返回该营业厅所有待派发的订单信息 | |
| Business.DriverManager.addVehicle | 语法 | public boolean addVehicle(Vehicle vo) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并输入了正确的车辆信息 | |
| 后置条件 | 系统根据输入的车辆信息更新车辆列表，并返回更新结果 | |
| Business.DriverManager.getDriverInfo | 语法 | Public ArrayList<VehicleVO>getVehicleInfo(String ID) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并输入正确的车辆编号 | |
| 后置条件 | 系统返回已有车辆的信息 | |
| Business.DriverManager.deleteDriver | 语法 | public boolean deleteVehicle(VehicleVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并要求删除指定车辆信息 | |
| 后置条件 | 系统删除对应车辆信息，并返回操作结果 | |
| Business.DriverManager.modifyDriver | 语法 | public boolean modifyVehicle(VehicleVO vo) | |
| 前置条件 | 营业厅业务员经过验证有进行车辆管理的权利，并要求修改 | |
| 后置条件 | 系统删除对应车辆信息，并返回操作结果 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| BusinessDataService.addVehicle(VehiclePO po) | | | 增加车辆信息 |
| BusinessDataService.getFreeVehicle() | | | 得到所有空闲的车辆的信息 |
| BusinessDataService.deleteVehicle(VehiclePO po) | | | 删除车辆信息 |
| BusinessDataService.modifyVehicle(VehiclePO po) | | | 修改车辆信息 |
| DataFactory.acceptCargo(OrderAcceptReceiptPO po) | | | 生成营业厅接收单 |
| ExpressDataService.getExpressInfos() | | | 获得该营业厅所有快递员信息 |
| DataFactory.gathering(GatheringReceiptPO po) | | | 将快递员的收费汇总并生成收款单 |
| BusinessDataService.getDriverInfo() | | | 得到ID对应的司机信息 |
| BusinessDataService.addDriver(DriverPO po) | | | 增加司机信息 |
| BusinessDataService.deleteDriver(DriverPO po) | | | 删除司机信息 |
| BusinessDataService.modifyDriver(DriverPO po) | | | 修改司机信息 |
| BusinessDataService.getTransferOrders() | | | 获得待转运的订单信息 |
| BusinessDataService.getSendOrder() | | | 得到所有待派送的订单信息 |

intermediabl模块的接口规范如表11所示。

**表11 intermediabl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Intermediate. showTransferingReceipt | 语法 | Public TransferingReceiptVO showTransferingReceipt(); |
| 前置条件 | 经过验证的中转中心业务员有查看中转中心到达单的权利，且业务员选择了中转中心到达单查看功能 |
| 后置条件 | 系统根据日期和中转中心编号显示正确的中转中心到达单 |
| Intermediate.addOrder | 语法 | public boolean addOrder(String ID); |
| 前置条件 | 经过验证的中转中心业务员有添加新寄件单到中转中心到达单的权利，且业务员输入了完整有效的订单号 |
| 后置条件 | 系统保存新增的寄件单信息并更新至中转中心到达单 |
| Intermediate.deleteOrder | 语法 | public boolean deleteOrder(String ID); |
| 前置条件 | 经过验证的中转中心业务员有删除新寄件单从中转中心到达单的权利，且业务员输入了完整有效的订单号或选择了正确的删除对象 |
| 后置条件 | 系统查找对应的订单信息，并删除该用户对象 |
| Intermediate.modifyOrder | 语法 | public boolean modifyOrder(String ID); |
| 前置条件 | 经过验证的中转中心业务员有修改新寄件单信息从中转中心到达单的权利，且业务员选择了正确的修改对象并输入了正确的订单号 |
| 后置条件 | 系统根据输入的新旧订单号更新中转中心到达单 |
| Intermediate.updateTransferingReceipt | 语法 | public void updateTransferingReceipt(TransferingReceiptVO); |
| 前置条件 | 经过验证的中转中心业务员选择更新中转中心到达单 |
| 后置条件 | 系统根据中转中心到达单将中转中心到达单数据存储至数据库中 |
| Intermediate.showPlaneList | 语法 | public ArrayList<PlaneVO> showPlaneList(); |
| 前置条件 | 经过验证的中转中心业务员有查看本中转中心飞机列表的权利，并选择查看所有飞机信息列表 |
| 后置条件 | 系统根据中转中心编号正确显示该中转中心飞机信息列表 |
| Intermediate. showPlane | 语法 | public PlaneVO showPlane(String ID); |
| 前置条件 | 经过验证的中转中心业务员有查看本中转中心某飞机当前包括已装车快件信息列表在内的所有飞机属性信息的权利，并正确输入了飞机编号或正确选择了飞机 |
| 后置条件 | 系统根据输入的飞机编号显示该飞机信息 |
| Intermediate.updateWaitingList | 语法 | public ArrayList<OrderVO> updateWaitingList(TranferingReceiptVO); |
| 前置条件 | 快件被装车或中转中心到达单更新 |
| 后置条件 | 等候装车的快件列表更新 |
| Intermediate. enplane | 语法 | public EnplaningReceiptVO enplane(ArrayList<OrderVO>); |
| 前置条件 | 经过验证的中转中心业务员有进行装车分配工作的权利，中转中心业务员业务员选择了装车分配操作 |
| 后置条件 | 等候装车的快件列表清空 |
| Intermediate. updateEnplaningReceiptList | 语法 | public ArrayList<EnplaningReceiptVO> updateEnplaningReceiptList(EnplaningReceiptVO); |
| 前置条件 | 等候装车的快件列表清空 |
| 后置条件 | 系统根据装车单将装车单数据存储至数据库中 |
| Intermediate. computeFare | 语法 | public fareVO computeFare(ArrayList<EnplaningReceiptVO>); |
| 前置条件 | 经过验证的中转中心业务员有汇总运费的权利，查看装车单后选择了运费汇总的操作 |
| 后置条件 | 系统运费合计值 |
| Intermediate. updateFare | 语法 | public boolean updateFare(fareVO); |
| 前置条件 | 中转中心业务员确认显示的运费合计值正确 |
| 后置条件 | 系统根据运费合计值将运费合计值数据更新至数据库中 |
| Intermediate.showEnplaningReceipt | 语法 | public boolean showEnplaningReceipt(ArrayList<EnplaningReceiptVO>); |
| 前置条件 | 经过验证的中转中心业务员有查看装车单的权利，且业务员选择了装车单查看功能 |
| 后置条件 | 系统根据日期和中转中心编号或飞机编号显示正确的装车单 |
| Intermediate. updateEnplaningReceipt | 语法 | public boolean updateEnplaningReceipt(ArrayList<EnplaningReceiptVO>); |
| 前置条件 | 中转中心业务员确认装车单信息正确 |
| 后置条件 | 系统根据装车单链表将装车单链表数据更新至数据库中 |
| Intermediate.showTrainList | 语法 | public ArrayList<TrainVO> showTrainList(); |
| 前置条件 | 经过验证的中转中心业务员有查看本中转中心火车列表的权利，并选择查看所有火车信息列表 |
| 后置条件 | 系统根据中转中心编号正确显示该中转中心火车信息列表 |
| Intermediate. showTrain | 语法 | public TrainVO showTrain(String ID); |
| 前置条件 | 经过验证的中转中心业务员有查看本中转中心某火车当前包括已装车快件信息列表在内的所有火车属性信息的权利，并正确输入了火车编号或正确选择了火车 |
| 后置条件 | 系统根据输入的火车编号显示该火车信息 |
| Intermediate.updateWaitingList | 语法 | public ArrayList<OrderVO> updateWaitingList(TranferingReceiptVO); |
| 前置条件 | 快件被装车或中转中心到达单更新 |
| 后置条件 | 等候装车的快件列表更新 |
| Intermediate. entrain | 语法 | public EntrainingReceiptVO entrain(ArrayList<OrderVO>); |
| 前置条件 | 经过验证的中转中心业务员有进行装车分配工作的权利，中转中心业务员业务员选择了装车分配操作 |
| 后置条件 | 等候装车的快件列表清空 |
| Intermediate. updateEntrainingReceiptList | 语法 | public ArrayList<EntrainingReceiptVO> updateEntrainingReceiptList(EntrainingReceiptVO); |
| 前置条件 | 等候装车的快件列表清空 |
| 后置条件 | 系统根据装车单将装车单数据存储至数据库中 |
| Intermediate. computeFare | 语法 | public fareVO computeFare(ArrayList<EntrainingReceiptVO>); |
| 前置条件 | 经过验证的中转中心业务员有汇总运费的权利，查看装车单后选择了运费汇总的操作 |
| 后置条件 | 系统运费合计值 |
| Intermediate. updateFare | 语法 | public boolean updateFare(fareVO); |
| 前置条件 | 中转中心业务员确认显示的运费合计值正确 |
| 后置条件 | 系统根据运费合计值将运费合计值数据更新至数据库中 |
| Intermediate.showEntrainingReceipt | 语法 | public boolean showEntrainingReceipt(ArrayList<EntrainingReceiptVO>); |
| 前置条件 | 经过验证的中转中心业务员有查看装车单的权利，且业务员选择了装车单查看功能 |
| 后置条件 | 系统根据日期和中转中心编号或火车编号显示正确的装车单 |
| Intermediate. updatEntrainingReceipt | 语法 | public boolean updateEntrainingReceipt(ArrayList<EntrainingReceiptVO>); |
| 前置条件 | 中转中心业务员确认装车单信息正确 |
| 后置条件 | 系统根据装车单链表将装车单链表数据更新至数据库中 |
| Intermediate.showTruckList | 语法 | public ArrayList<TruckVO> showTruckList(); |
| 前置条件 | 经过验证的中转中心业务员有查看本中转中心火车列表的权利，并选择查看所有火车信息列表 |
| 后置条件 | 系统根据中转中心编号正确显示该中转中心火车信息列表 |
| Intermediate. showTruck | 语法 | public TruckVO showTruck(String ID); |
| 前置条件 | 经过验证的中转中心业务员有查看本中转中心某火车当前包括已装车快件信息列表在内的所有火车属性信息的权利，并正确输入了火车编号或正确选择了火车 |
| 后置条件 | 系统根据输入的火车编号显示该火车信息 |
| Intermediate.updateWaitingList | 语法 | public ArrayList<OrderVO> updateWaitingList(TranferingReceiptVO); |
| 前置条件 | 快件被装车或中转中心到达单更新 |
| 后置条件 | 等候装车的快件列表更新 |
| Intermediate. entruck | 语法 | public EntruckingReceiptVO enTruck(ArrayList<OrderVO>); |
| 前置条件 | 经过验证的中转中心业务员有进行装车分配工作的权利，中转中心业务员业务员选择了装车分配操作 |
| 后置条件 | 等候装车的快件列表清空 |
| Intermediate. updateEntruckingReceiptList | 语法 | public ArrayList<EntruckingReceiptVO> updateEntruckingReceiptList(EntruckingReceiptVO); |
| 前置条件 | 等候装车的快件列表清空 |
| 后置条件 | 系统根据装车单将装车单数据存储至数据库中 |
| Intermediate. computeFare | 语法 | public fareVO computeFare(ArrayList<EntruckingReceiptVO>); |
| 前置条件 | 经过验证的中转中心业务员有汇总运费的权利，查看装车单后选择了运费汇总的操作 |
| 后置条件 | 系统运费合计值 |
| Intermediate. updateFare | 语法 | public boolean updateFare(fareVO); |
| 前置条件 | 中转中心业务员确认显示的运费合计值正确 |
| 后置条件 | 系统根据运费合计值将运费合计值数据更新至数据库中 |
| Intermediate.showEntruckingReceipt | 语法 | public boolean showEntruckingReceipt(ArrayList<EntruckingReceiptVO>); |
| 前置条件 | 经过验证的中转中心业务员有查看装车单的权利，且业务员选择了装车单查看功能 |
| 后置条件 | 系统根据日期和中转中心编号或火车编号显示正确的装车单 |
| Intermediate. updatEntruckingReceipt | 语法 | public boolean updateEntruckingReceipt(ArrayList<EntruckingReceiptVO>); |
| 前置条件 | 中转中心业务员确认装车单信息正确 |
| 后置条件 | 系统根据装车单链表将装车单链表数据更新至数据库中 |
| 需要的服务（需接口） | | |
| 服务名 | | 服务 |
| FileFactory.getIntermediateData | | 得到Intermediate文件的引用 |
| IntermediateDataService. showTransferingReceipt（） | | 返回一个TransferingReceiptPO对象 |
| IntermediateDataService.addOrder(String ID) | | 在用户数据文件中插入一个OrderPO对象 |
| IntermediateDataService.deleteOrder(String ID) | | 在用户数据文件中删除一个OrderPO对象 |
| IntermediateDataService.modifyOrder(String ID) | | 在用户数据文件中更新一个OrderPO对象 |
| IntermediateDataService.updateTransferingReceipt(TransferingReceiptPO) | | 在用户数据文件中插入一个TransferingReceiptPO对象 |
| IntermediateDataService.getPlaneList() | | 返回一个存储PlanePO的ArrayList对象 |
| IntermediateDataService. getPlane(String ID) | | 返回一个PlanePO对象 |
| IntermediateDataService.updateWaitingList(TranferingReceiptPO) | | 更新一个存储OrderPO对象的ArrayList对象 |
| IntermediateDataService. enplane(ArrayList<OrderPO>) | | 清空一个存储OrderPO对象的ArrayList对象 |
| IntermediateDataService. updateEnplaningReceiptList(EnplaningReceiptPO) | | 在用户数据文件中插入一个EnplaningReceiptPO对象 |
| IntermediateDataService. computeFare(ArrayList<EnplaningReceiptPO>) | | 返回一个farePO对象 |
| IntermediateDataService. updateFare(farePO) | | 在用户数据文件中插入一个farePO对象 |
| IntermediateDataService.getEnplaningReceipt(ArrayList<EnplaningReceiptPO>) | | 更新一个存储EnplaningReceiptPO对象的ArrayList对象 |
| IntermediateDataService. updateEnplaningReceipt(ArrayList<EnplaningReceiptPO>) | | 在用户数据文件中插入所有ArrayList中所有的EnplaningRecePO对象 |
| IntermediateDataService.getTrainList() | | 返回一个存储TrainPO的ArrayList对象 |
| IntermediateDataService. getTrain(String ID) | | 返回一个TrainPO对象 |
| IntermediateDataService.updateWaitingList(TranferingReceiptPO) | | 更新一个存储OrderPO对象的ArrayList对象 |
| IntermediateDataService. entrain(ArrayList<OrderPO>) | | 清空一个存储OrderPO对象的ArrayList对象 |
| IntermediateDataService. updateEntrainingReceiptList(EntrainingReceiptPO) | | 在用户数据文件中插入一个EntrainingReceiptPO对象 |
| IntermediateDataService. computeFare(ArrayList<EntrainingReceiptPO>) | | 返回一个farePO对象 |
| IntermediateDataService. updateFare(farePO) | | 在用户数据文件中插入一个farePO对象 |
| IntermediateDataService.getEntrainingReceipt(ArrayList<EntrainingReceiptPO>) | | 更新一个存储EntrainingReceiptPO对象的ArrayList对象 |
| IntermediateDataService. updatEntrainingReceipt(ArrayList<EntrainingReceiptPO>) | | 在用户数据文件中插入所有ArrayList中所有的EntrainingRecePO对象 |
| IntermediateDataService.getTruckList() | | 返回一个存储TruckPO的ArrayList对象 |
| IntermediateDataService. getTruck(String ID) | | 返回一个TruckPO对象 |
| IntermediateDataService.updateWaitingList(TranferingReceiptPO) | | 更新一个存储OrderPO对象的ArrayList对象 |
| IntermediateDataService. entruck(ArrayList<OrderPO>) | | 清空一个存储OrderPO对象的ArrayList对象 |
| IntermediateDataService. updateEntruckingReceiptList(EntruckingReceiptPO) | | 在用户数据文件中插入一个EntruckingReceiptPO对象 |
| IntermediateDataService. computeFare(ArrayList<EntruckingReceiptPO>) | | 返回一个farePO对象 |
| IntermediateDataService. updateFare(farePO) | | 在用户数据文件中插入一个farePO对象 |
| IntermediateDataService.getEntruckingReceipt(ArrayList<EntruckingReceiptPO>) | | 更新一个存储EntruckingReceiptPO对象的ArrayList对象 |
| IntermediateDataService. updatEntruckingReceipt(ArrayList<EntruckingReceiptPO>) | | 在用户数据文件中插入所有ArrayList中所有的EntruckingRecePO对象 |

financebl模块的接口规范如表12所示。

**表12 financebl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Finance.initInfo | 语法 | | public int initInfo(InitInfoVO vo,String time); |
| 前置条件 | | 用户要求期初建账 |
| 后置条件 | | 系统初始化一套账 |
| Finance.getInitInfo | 语法 | | public InitInfoVO getInitInfo(String time); |
| 前置条件 | | 用户要求查看期初信息 |
| 后置条件 | | 返回期初信息并显示 |
| Finance.getAllInitInfo | 语法 | | public ArrayList<InitInfoVO> getAllInitInfo(); |
| 前置条件 | | 用户具有查看期初信息的权限 |
| 后置条件 | | 系统返回所有期初信息 |
| Finance.addAccount | 语法 | | public int addAccount(AccountVO vo); |
| 前置条件 | | 用户要求新建账户 |
| 后置条件 | | 在账户列表中添加该账户信息 |
| Finance.deleteAccount | 语法 | | public int deleteAccount(AccountVO vo); |
| 前置条件 | | 用户要求删除账户 |
| 后置条件 | | 从账户列表中删除该账户信息 |
| Finance.modifyAccount | 语法 | | public int modifyAccount(AccountVO vo,String name) |
| 前置条件 | | 用户要求修改账户并选中需修改的账户 |
| 后置条件 | | 在账户列表中修改该账户信息 |
| Finance.findByName | 语法 | | public AccountVO findByName(String name); |
| 前置条件 | | 用户输入账户名称并查找 |
| 后置条件 | | 查找是否存在该名称的账户并返回该账户的信息 |
| Finance.findByKeyword | 语法 | | public ArrayList<AccountVO> findByKeyword(String s); |
| 前置条件 | | 用户输入关键词并查询 |
| 后置条件 | | 查找是否存在含有该关键字的账户并返回含有该关键字的账户的信息 |
| Finance.showAll | 语法 | | public ArrayList<AccountVO> showAll(); |
| 前置条件 | | 用户要求查看所有的账户列表 |
| 后置条件 | | 返回所有账户列表 |
| Finance.showBSL | 语法 | | public BusinessstatementReceiptVO show(String beginTime,String endTime); |
| 前置条件 | | 用户输入起始日期并要求查看经营情况表 |
| 后置条件 | | 系统显示该段时间内所有入款单和付款单信息 |
| Finance.showAllBSL | 语法 | | public ArrayList<BusinessstatementReceiptVO> showAllBSList(); |
| 前置条件 | | 用户经验证有查看经营情况表的权限 |
| 后置条件 | | 系统显示所有经营情况的列表 |
| Finance.export | 语法 | | public int export(BusinessstatementReceiptVO vo); |
| 前置条件 | | 用户要求导出经营情况表的excel |
| 后置条件 | | 系统生成经营情况表的excel |
| Finance.creatCollection | 语法 | | public int creatCollection (CollectionReceiptVO vo); |
| 前置条件 | | 用户要求新建入款单并输入有效的信息 |
| 后置条件 | | 向系统输入新的入款单信息 |
| Finance.getCollection | 语法 | | Public CollectionReceiptVO getCollection (String s); |
| 前置条件 | | 用户具有查看入款单的权限 |
| 后置条件 | | 系统显示符合条件的入款单 |
| Finance.getAllCollection | 语法 | | public ArrayList<CollectionReceiptVO> getAllCollection(); |
| 前置条件 | | 用户有查看入款单的权限 |
| 后置条件 | | 系统显示所有入款单列表 |
| Finance.getGathering | 语法 | | public ArrayList<GatheringVO> getGathering(String HallID,String Time,GatheringVO vo); |
| 前置条件 | | 用户输入日期和营业厅编号并要求查看收款单 |
| 后置条件 | | 系统显示符合条件的收款单列表 |
| Finance.getMoney | 语法 | | public double[] getMoney(GatheringVO vo); |
| 前置条件 | | 所有符合条件的收款单已经获取 |
| 后置条件 | | 系统返回收款单的金额 |
| Finance.getTotalMoney | 语法 | | public double getTotalMoney(int[] money); |
| 前置条件 | | 用户要求合计金额 |
| 后置条件 | | 系统返回合计后的金额 |
| Finance.getCollectionListID | 语法 | | public String getCollectionListID(); |
| 前置条件 | | 用户制定入款单时需要根据今天已经制定的入款单来自动生成入款单编号 |
| 后置条件 | | 系统显示自动生成的入款单ID |
| Finance.createPayment | 语法 | | public int creatPayment(PaymentReceiptVO vo); |
| 前置条件 | | 用户要求新建付款单并输入有效信息 |
| 后置条件 | | 向系统输入新的付款单信息 |
| Finance.getPayment | 语法 | | Public PaymentReceiptVO getPayment(String s); |
| 前置条件 | | 用户有查看付款单的权限 |
| 后置条件 | | 系统显示符合条件的付款单列表 |
| Finance.getAllPayment | 语法 | | public ArrayList<PaymentReceiptVO> getAllPayment(); |
| 前置条件 | | 用户有查看付款单的权限 |
| 后置条件 | | 系统显示所有付款单列表 |
| Finance.getPaymentID | 语法 | | public String getPaymentListID(); |
| 前置条件 | | 用户制定付款单时需要根据今天已经制定的付款单来自动生成付款单编号 |
| 后置条件 | | 系统显示自动生成的付款单ID |
| Finance.createCostIncomeList | 语法 | | public int creatCostIncomeList(CostIncomeReceiptVO vo); |
| 前置条件 | | 用户要求新建成本收益表并输入有效信息 |
| 后置条件 | | 向系统中输入新的成本收益表信息 |
| Finance.getCostIncmeList | 语法 | | public CostIncomeReceiptVO getCostIncomeList(String s); |
| 前置条件 | | 用户有查看成本收益表的权限 |
| 后置条件 | | 系统返回符合条件的成本收益表 |
| Finance.getAllCostIncomeList | 语法 | | public ArrayList<CostIncomeReceiptVO> getAllCostIncomeList(); |
| 前置条件 | | 用户有查看成本收益表的权限 |
| 后置条件 | | 系统返回所有成本收益表列表 |
| Finance.getCollection | 语法 | | public ArrayList<CollectionReceiptVO> getCollection(); |
| 前置条件 | | 用户有查看入款单的权限 |
| 后置条件 | | 系统返回所有入款单列表 |
| Finance.getPayment | 语法 | | public ArrayList<PaymentReceiptVO>getPayment(); |
| 前置条件 | | 用户有查看付款单的权限 |
| 后置条件 | | 系统返回所有付款单列表 |
| Finance.getCost | 语法 | | public double getCost(PaymentReceiptVO vo); |
| 前置条件 | | 已获取付款单 |
| 后置条件 | | 系统返回付款单金额总和 |
| Finance.getIncome | 语法 | | public double getIncome(CollectionReceiptVO vo); |
| 前置条件 | | 已获取入款单 |
| 后置条件 | | 系统返回入款单金额总和 |
| Finance.getProfit | 语法 | | public double getProfit(double income,double cost); |
| 前置条件 | | 已获取付款单金额之和,入款单金额之和 |
| 后置条件 | | 系统返回利润（即入款单付款单金额之差） |
| Finance.getCostIncomeListID | 语法 | | public String getCostIncomeListID(); |
| 前置条件 | | 用户制定成本收益表时需要根据今天已经制定的成本收益表来自动生成编号 |
| 后置条件 | | 系统显示自动生成的成本收益表ID |
| 需要的服务（供接口） | | | |
| 服务名 | | 服务 | |
| FinanceDataService.initInfo(InitInfoPO po) | | 在数据库中添加一套账的初始化信息（InitInfoPO） | |
| FinanceDataService.getInitInfo(String time) | | 查找符合时间条件的期初信息并返回 | |
| FinanceDataService.getAllInitInfo() | | 返回并输出所有的持久化对象InitInfoPO | |
| DataFactory.getAccountData | | 得到Account数据库的引用 | |
| FinanceDataService.addAccount(AccountPO po) | | 插入单一持久化对象AccountPO | |
| FinanceDataService.deleteAccount(AccountPO po) | | 删除单一持久化对象AccountPO | |
| FinanceDataService.modifyAccount(AccountPO po,String name) | | 修改单一持久化对象AccountPO | |
| FinanceDataService.findByName(String name) | | 根据输入的名称查找单一持久化对象AccountPO | |
| public ArrayList<AccountPO> findByKeyword(String s) | | 根据输入的关键字查找持久化对象AccountPO | |
| public ArrayList<AccountPO> showAll() | | 输出持久化对象AccountPO | |
| FinanceDataService.showBSL(String beginTime,String endTime) | | 返回并输出符合时间条件的单一持久化对象BusinessstatementReceiptPO | |
| public ArrayList<BusinessstatementReceiptPO> showAllBSList() | | 返回并输出所有的单一持久化对象BusinessstatementReceiptPO | |
| public int export(BusinessstatementReceiptPO po) | | 导出单一持久化对象BusinessstatementReceiptPO到excel | |
| FinanceDataService.creatCollection(CollectionReceiptPO po) | | 插入单一持久化对象CollectionReceiptPO | |
| FinanceDataService.getAllCollection() | | 得到持久化对象CollectionReceiptPO的ArrayList | |
| FinanceDataService.getGathering(String HallID,String Time) | | 得到持久化对象GatheringReceiptPO的ArrayList | |
| FinanceDataService.getMoney(GatheringReceiptPO po) | | 得到持久化对象GatheringReceiptPO中的金额数 | |
| FinanceDataService.getTotalMoney(int[] money) | | 得到持久化对象GatheringReceiptPO中的金额总和 | |
| FinanceDataService.getNum() | | 得到当天存储的持久化对象CollectionReceiptPO的个数 | |
| FinanceDataService.findByID(String ID) | | 根据ID返回对应的一个持久化对象CollectionReceiptPO | |
| FinanceDataService.modify(CollectionReceiptPO po) | | 修改单一持久化对象CollectionReceiptPO的信息 | |
| FinanceDataService. creatPaymentReceipt(PaymentReceiptPO po) | | 插入单一持久化对象 PaymentReceiptPO | |
| FinanceDataService. getAllPaymentReceipt() | | 得到持久化对象 PaymentReceiptPO的ArrayList | |
| FinanceDataService. getNum() | | 得到当天存储的持久化对象PaymentReceiptPO的个数 | |
| FinanceDataService. findByID(String ID) | | 根据ID返回对应的一个持久化对象PaymentReceiptPO | |
| FinanceDataService. modify(PaymentReceiptPO po) | | 修改单一持久化对象CollectionReceiptPO的信息 | |
| FinanceDataService. creatCostIncomeList(CostIncomeReceiptPO po) | | 插入单一持久化对象 CostIncomeReceiptPO | |
| FinanceDataService. getAllCostIncomeList() | | 得到持久化对象CostIncomeReceiptPO的ArrayList | |
| FinanceDataService. getCollection() | | 得到持久化对象CollectionReceiptPO的ArrayList | |
| FinanceDataService. getPayment() | | 得到持久化对象PaymentReceiptPO的ArrayList | |
| FinanceDataService. getNum() | | 得到当天存储的持久化对象CostIncomeReceiptPO的个数 | |
| FinanceDataService. findByID(String ID) | | 根据ID返回对应的一个持久化对象CostIncomeReceiptPO | |
| FinanceDataService. modify(CostIncomeReceiptPO po) | | 修改单一持久化对象CostIncomeReceiptPO的信息 | |

repertorybl模块的接口规范如表13所示。

**表13 repertorybl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Repertory.inventoryInitialization | 语法 | public int inventoryInitialization(String repertoryID, int maxRow, int maxShelf, int Maxdigit, int warningRatio) | |
| 前置条件 | 经过验证的仓库管理员有库存信息初始化的权利，且仓库管理员输入了有效的库存信息值 | |
| 后置条件 | 系统保存仓库初始化信息 | |
| Repertory.enterRepertory | 语法 | public int enterRepertory(String repertoryID, String JJD-ID, int blockNum, int rowNum, int shelfNum, int digitNum, String date) | |
| 前置条件 | 经过验证的仓库管理员有将商品入库的权利，且仓库管理员输入了有效的商品信息和对应仓库位置、时间信息 | |
| 后置条件 | 系统保存商品入库的位置和时间信息，生成入库单 | |
| Repertory.leaveRepertory | 语法 | public int leaveRepertory(String repertoryID, String JJD-ID, int transType, String date) | |
| 前置条件 | 经过验证的仓库管理员有将商品出库的权利，且仓库管理员输入了有效的商品信息和转运方式、时间信息 | |
| 后置条件 | 系统保存商品出库的转运方式和时间，生成出库单 | |
| Repertory.inventoryWarning | 语法 | public boolean inventoryWarning() | |
| 前置条件 | 当某区的库存总量与某区最大容量的比值达到或者超过警戒比例时，系统提示仓库管理员该区库存已达警戒比例 | |
| 后置条件 | 无 | |
| Repertory.inventoryCheck | 语法 | public ArrayList<GoodsVO> inventoryCheck(String repertoryID, String beginDate, String endDate) | |
| 前置条件 | 经过验证的仓库管理员有库存查看的权利，且仓库管理员输入了有效的时间段信息 | |
| 后置条件 | 系统根据输入的时间段查找对应的出入库商品，并返回这段时间出/入库商品数量和金额总和，库存数量合计 | |
| Repertory.inventoryStockTaking | 语法 | public ArrayList<GoodsVO> inventoryStockTaking(String repertoryID) | |
| 前置条件 | 经过验证的仓库管理员有库存盘点的权利，且仓库管理员输入了库存盘点命令 | |
| 后置条件 | 系统自动根据提交库存盘点命令的时间生成一个截止点，并根据该截止点查找当天到截止点之前的仍在仓库中的商品，返回商品信息 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| FileFactory.getRepertoryData | | | 得到Repertory文件的引用 |
| RepertoryDataService.modifyRepertory(RepertoryPO repertorypo) | | | 在仓库数据文件中插入一个RepertoryPO对象 |
| RepertoryDataService.findRepertory(String repertoryID) | | | 在仓库数据文件中删除一个RepertoryPO对象 |
| RepertoryDataService.addGoods(String repertoryID, GoodsPO goodspo) | | | 在编号为repertoryID的仓库对应库存列表中插入一个GoodsPO对象 |
| RepertoryDataService.deleteGoods(String repertoryID, GoodsPO goodspo) | | | 在编号为repertoryID的仓库对应库存列表中删除一个GoodsPO对象 |
| RepertoryDataService.modifyGoods(String repertoryID, GoodsPO goodspo) | | | 在编号为repertoryID的仓库对应库存列表中更新一个GoodsPO对象 |
| RepertoryDataService.findGoodsbyID(String repertoryID, String JJD-ID) | | | 根据输入的仓库编号和订单号，返回对应的GoodsPO对象 |
| RepertoryDataService.findGoodsbyDate(String repertoryID, String beginDate, String endDate) | | | 根据输入的仓库编号和起始日期，返回对应的GoodsPO对象 |
| RepertoryDataService.findGoodsbyTime(String repertoryID, String time) | | | 根据输入的仓库编号和截止点时间，返回对应的GoodsPO对象 |

managebl模块的接口规范如表14所示。

**表14 managebl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Manage.addOrganization | 语法 | | public int addOrganization(OrganizationVO organizationvo) |
| 前置条件 | | 经过验证的总经理有添加新机构的权利，且总经理输入了完整有效的新机构信息 |
| 后置条件 | | 系统保存新增的机构信息并更新机构列表 |
| Manage.deleteOrganization | 语法 | | public int deleteOrganization(OrganizationVO organizationvo) |
| 前置条件 | | 经过验证的总经理有删除机构的权利，且总经理输入了有效的待删除机构信息 |
| 后置条件 | | 系统查找对应的机构信息，并删除该机构对象 |
| Manage.modifyOrganization | 语法 | | public int modifyOrganization(OrganizationVO organizationvo) |
| 前置条件 | | 经过验证的总经理有修改机构信息的权利，且总经理输入了有效的待修改机构信息 |
| 后置条件 | | 系统更新机构信息 |
| Manage.findOrganization | 语法 | | public OrganizationVO findOrganization(String organizationID) |
| 前置条件 | | 经过验证的总经理有查询机构的权利，且总经理输入了有效的待查询机构编号 |
| 后置条件 | | 系统根据输入的机构编号查找机构，并返回机构信息 |
| Manage.chooseDepartment | 语法 | | public int chooseDepartment(String userID, String organizationID) |
| 前置条件 | | 经过验证的总经理有为新员工选择所属机构的权利，且总经理输入了有效的员工编号和机构编号 |
| 后置条件 | | 系统根据输入的员工编号查找对应的员工，并将该员工所属机构修改为机构编号对应的机构 |
| Manage.addPerWage | 语法 | | public int addPerWage(PerWageVO perwagevo) |
| 前置条件 | | 经过验证的总经理有添加新的每次工资的权利，且总经理输入了完整有效的新每次工资信息 |
| 后置条件 | | 系统保存新增的每次工资信息并更新每次工资列表 |
| Manage.deletePerWage | 语法 | | public int deletePerWage(PerWageVO perwagevo) |
| 前置条件 | | 经过验证的总经理有删除每次工资的权利，且总经理输入了有效的待删除每次工资信息 |
| 后置条件 | | 系统查找对应的每次工资信息，并删除该每次工资对象 |
| Manage.modifyPerWage | 语法 | | public int modifyPerWage(PerWageVO perwagevo) |
| 前置条件 | | 经过验证的总经理有修改每次工资的权利，且总经理输入了有效的待修改每次工资信息 |
| 后置条件 | | 系统更新每次工资信息 |
| Manage.findPerWage | 语法 | | public double findPerWage(int profession) |
| 前置条件 | | 经过验证的总经理有查询每次工资的权利，且总经理输入了有效的对应职业编号 |
| 后置条件 | | 系统根据输入的职业编号查找每次工资，并返回每次工资信息 |
| Manage.addBasicSalary | 语法 | | public int addBasicSalary(BasicSalaryVO basicsalaryvo) |
| 前置条件 | | 经过验证的总经理有添加新的基础月薪的权利，且总经理输入了完整有效的新基础月薪信息 |
| 后置条件 | | 系统保存新增的基础月薪信息并更新基础月薪列表 |
| Manage.deleteBasicSalary | 语法 | | public int deleteBasicSalary(BasicSalaryVO basicsalaryvo) |
| 前置条件 | | 经过验证的总经理有删除基础月薪的权利，且总经理输入了有效的待删除基础月薪信息 |
| 后置条件 | | 系统查找对应的基础月薪信息，并删除该基础月薪对象 |
| Manage.modifyBasicSalary | 语法 | | public int modifyBasicSalary(BasicSalaryVO basicsalaryvo) |
| 前置条件 | | 经过验证的总经理有修改基础月薪的权利，且总经理输入了有效的待修改基础月薪信息 |
| 后置条件 | | 系统更新基础月薪信息 |
| Manage.findBasicSalary | 语法 | | public double findBasicSalary(int profession) |
| 前置条件 | | 经过验证的总经理有查询基础月薪的权利，且总经理输入了有效的对应职业编号 |
| 后置条件 | | 系统根据输入的职业编号查找基础月薪，并返回基础月薪信息 |
| Manage.addCityDistance | 语法 | | public int addCityDistance(CityDistanceVO citydistancevo) |
| 前置条件 | | 经过验证的总经理有添加新的城市距离的权利，且总经理输入了完整有效的新城市距离信息 |
| 后置条件 | | 系统保存新增的城市距离信息并更新城市距离列表 |
| Manage.deleteCityDistance | 语法 | | public int deleteCityDistance(CityDistanceVO citydistancevo) |
| 前置条件 | | 经过验证的总经理有删除城市距离的权利，且总经理输入了有效的待删除城市距离信息 |
| 后置条件 | | 系统查找对应的城市距离信息，并删除该城市距离对象 |
| Manage.modifyCityDistance | 语法 | | public int modifyCityDistance(CityDistanceVO citydistancevo) |
| 前置条件 | | 经过验证的总经理有修改城市距离的权利，且总经理输入了有效的待修改城市距离信息 |
| 后置条件 | | 系统更新城市距离信息 |
| Manage.findCityDistance | 语法 | | public double findCityDistance(String cityA, String cityB) |
| 前置条件 | | 经过验证的总经理有查询城市距离的权利，且总经理输入了有效且存在的两个城市 |
| 后置条件 | | 系统根据输入的两个城市名查找城市距离，并返回城市距离信息 |
| Manage.addCost | 语法 | | public int addCost(CostVO costvo) |
| 前置条件 | | 经过验证的总经理有添加新运费系数的权利，且总经理输入了完整有效的新运费系数信息 |
| 后置条件 | | 系统保存新增的运费系数信息并更新运费系数列表 |
| Manage.deleteCost | 语法 | | public int deleteCost(CostVO costvo) |
| 前置条件 | | 经过验证的总经理有删除运费系数的权利，且总经理输入了有效的待删除运费系数信息 |
| 后置条件 | | 系统查找对应的运费系数信息，并删除该运费系数对象 |
| Manage.modifyCost | 语法 | | public int modifyCost(CostVO costvo) |
| 前置条件 | | 经过验证的总经理有修改运费系数的权利，且总经理输入了有效的待修改运费系数信息 |
| 后置条件 | | 系统更新运费系数信息 |
| Manage.findCost | 语法 | | public double findCost(int category) |
| 前置条件 | | 经过验证的总经理有查询运费系数的权利，且总经理输入了有效且存在的类别号 |
| 后置条件 | | 系统根据输入的类别号查找运费系数，并返回运费系数信息 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| FileFactory.getManageData | | 得到Manage文件的引用 | |
| ManageDataService.addOrganization(OrganizationPO organizationpo) | | 在管理数据包的机构数据文件中插入一个OrganizationPO对象 | |
| ManageDataService.deleteOrganization(OrganizationPO organizationpo) | | 在管理数据包的机构数据文件中删除一个OrganizationPO对象 | |
| ManageDataService.modifyOrganization(OrganizationPO organizationpo) | | 在管理数据包的机构数据文件中更新一个OrganizationPO对象 | |
| ManageDataService.findOrganization(String OrganizationID) | | 根据输入的机构编号，返回对应的OrganizationPO对象 | |
| ManageDataService.modifyUser(UserPO userpo) | | 在用户数据文件中更新一个UserPO对象 | |
| ManageDataService.addPerWage(PerWagePO perwagepo) | | 在管理数据包的每次工资数据文件中插入一个PerWagePO对象 | |
| ManageDataService.deletePerWage(PerWagePO perwagepo) | | 在管理数据包的每次工资数据文件中删除一个PerWagePO对象 | |
| ManageDataService.modifyPerWage(PerWagePO perwagepo) | | 在管理数据包的每次工资数据文件中更新一个PerWagePO对象 | |
| ManageDataService.findPerWage(int profession) | | 根据输入的职业编号，返回对应的PerWagePO对象 | |
| ManageDataService. addBasicSalary(BasicSalaryPO citydistancepo) | | 在管理数据包的基础月薪数据文件中插入一个BasicSalaryPO对象 | |
| ManageDataService.deleteBasicSalary(BasicSalaryPO citydistancepo) | | 在管理数据包的基础月薪数据文件中删除一个BasicSalaryPO对象 | |
| ManageDataService.modifyBasicSalary(BasicSalaryPO citydistancepo) | | 在管理数据包的基础月薪数据文件中更新一个BasicSalaryPO对象 | |
| ManageDataService.findBasicSalary(int profession) | | 根据输入的职业编号，返回对应的BasicSalaryPO对象 | |
| ManageDataService.addCityDistance(CityDistancePO citydistancepo) | | 在管理数据包的城市距离数据文件中插入一个CityDistancePO对象 | |
| ManageDataService.deleteCityDistance(CityDistancePO citydistancepo) | | 在管理数据包的城市距离数据文件中删除一个CityDistancePO对象 | |
| ManageDataService.modifyCityDistance(CityDistancePO citydistancepo) | | 在管理数据包的城市距离数据文件中更新一个CityDistancePO对象 | |
| ManageDataService.findCityDistance(String cityA, String cityB) | | 根据输入的两个城市名，返回对应的CityDistancePO对象 | |
| ManageDataService.addCost(CostPO costpo) | | 在管理数据包的运费系数数据文件中插入一个CostPO对象 | |
| ManageDataService.deleteCost(CostPO costpo) | | 在管理数据包的运费系数数据文件中删除一个CostPO对象 | |
| ManageDataService.modifyCost(CostPO costpo) | | 在管理数据包的运费系数数据文件中更新一个CostPO对象 | |
| ManageDataService. findCost(int category) | | 根据输入的类别，返回对应的CostPO对象 | |

userbl模块的接口规范如表15所示。

**表15 userbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| User.login | 语法 | public int login(String userID, String password) |
| 前置条件 | 快递物流系统的用户在登录界面输入了自己的用户编号和密码 |
| 后置条件 | 查找是否存在编号为userID的用户，根据输入的password返回登录验证的结果 |
| User.addUser | 语法 | public int addUser(UserVO uservo) |
| 前置条件 | 经过验证的管理员有添加新用户的权利，且管理员输入了完整有效的新用户信息 |
| 后置条件 | 系统保存新增的用户信息并更新用户列表 |
| User.deleteUser | 语法 | public int deleteUser(UserVO uservo) |
| 前置条件 | 经过验证的管理员有删除用户的权利，且管理员输入了有效的待删除用户信息 |
| 后置条件 | 系统查找对应的用户信息，并删除该用户对象 |
| User.modifyUserPassword | 语法 | public int modifyUserPassword (UserVO uservo) |
| 前置条件 | 系统中每个用户有修改自己密码的权利，且该用户输入了正确的原有密码和有效的新密码 |
| 后置条件 | 系统根据用户编号和输入的新密码更新用户信息 |
| User.modifyUserAuthority | 语法 | public int modifyUserAuthority(UserVO uservo) |
| 前置条件 | 经过验证的管理员有更改用户权限的权利，且管理员输入了有效的待更改用户信息 |
| 后置条件 | 系统根据用户编号和新的用户权限更新用户信息 |
| User.findUser | 语法 | public UserVO findUser(String userID); |
| 前置条件 | 经过验证的管理员或总经理有查询用户的权利，且管理员或总经理输入了有效的待查询用户编号 |
| 后置条件 | 系统根据输入的用户编号查找用户，并返回用户信息 |
| User.showAllUsers | 语法 | public ArrayList<UserVO> showAllUsers() |
| 前置条件 | 经过验证的管理员或总经理有查询用户的权利，且管理员或总经理没有输入 |
| 后置条件 | 系统返回所有的用户信息 |
| 需要的服务（需接口） | | |
| 服务名 | | 服务 |
| FileFactory.getUserData | | 得到User文件的引用 |
| UserDataService.check(String userID ,String password) | | 得到根据输入的ID和password验证的结果 |
| UserDataService.add(UserPO userpo) | | 在用户数据文件中插入一个UserPO对象 |
| UserDataService.delete(UserPO userpo) | | 在用户数据文件中删除一个UserPO对象 |
| userDataService.modify(UserPO userpo) | | 在用户数据文件中更新一个UserPO对象 |
| userDataService.find (String userID) | | 根据输入的用户编号，返回对应的UserPO对象 |
| userDataService.showAll () | | 返回所有的UserPO对象 |

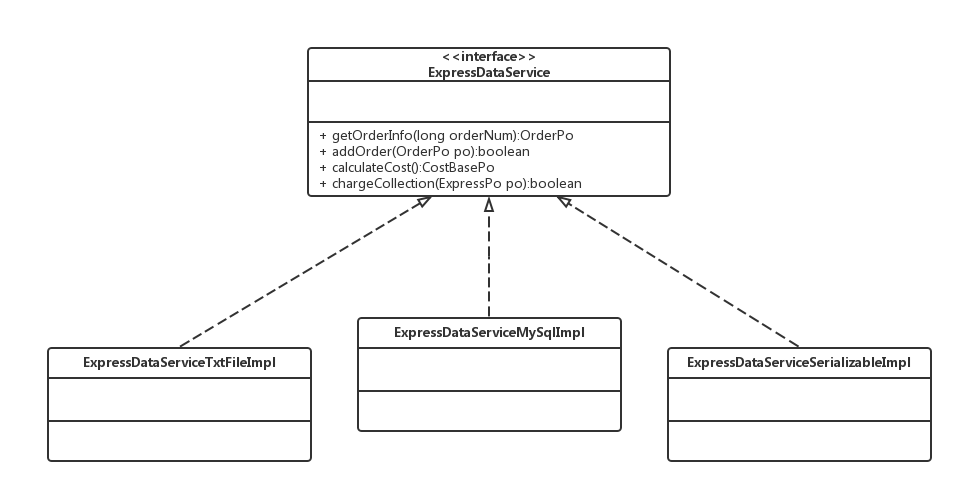
receiptbl模块的接口规范如表16所示。

**表16 receiptbl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务(供接口) | | | |
| Receipt.add | 语法 | Public int add(ReceiptVo vo) | |
| 前置条件 | User经验证，可以创建单据 | |
| 后置条件 | 向待审批单据列表库中添加该单据 | |
| Receipt.modify | 语法 | Public int modify(String ID) | |
| 前置条件 | User选中了有效单据 | |
| 后置条件 | 更新该单据信息 | |
| Receipt.batch | 语法 | Public int batch(String[] ID) | |
| 前置条件 | 当前有待审批单据，且user已选中单据 | |
| 后置条件 | 更新该单据组信息 | |
| Receipt.update | 语法 | Public void update(ReceiptVo vo) | |
| 前置条件 | 该业务单据创建成功 | |
| 后置条件 | 持久化更新涉及的领域对象的数据 | |
| Receipt.reply | 语法 | Public void reply(String userID) | |
| 前置条件 | 该单据审批完成 | |
| 后置条件 | 该单据制单人收到审批结果信息 | |
| Receipt.view | 语法 | Public ArrayList<ReceiptVO> view() | |
| 前置条件 | 已有单据记录 | |
| 后置条件 | 返回所有单据信息 | |
| Receipt.refresh | 语法 | Public ArrayList<ReceiptVO> refresh() | |
| 前置条件 | 已有单据记录 | |
| 后置条件 | 返回所有待审批单据信息 | |
| Receipt.approve | 语法 | Public int approve(String ID) | |
| 前置条件 | 有单据提交待审批 | |
| 后置条件 | 更新该单据信息 | |
| 需要的服务(需接口) | | | |
| 服务名 | | | 服务 |
| FileFactory.getReceiptData | | | 得到Receipt文件的引用 |
| ReceiptDataService.add(ReceiptPo po) | | | 插入单一持久化对象 |
| ReceiptDataService.modify(ReceiptPo po) | | | 更新单一持久化对象 |
| ReceiptDataService.modify(ReceiptPo[] po) | | | 更新单一持久化对象组 |
| ReceiptDataService.find(Type type) | | | 根据类型type查找ReceiptPo对象 |
| ReceiptDataService.find(Date date) | | | 根据日期date查找ReceiptPo对象 |
| ReceiptDataService.show() | | | 得到所得的ReceiptPo对象 |

## 5.4数据层的分解

数据层主要给业务逻辑层提供数据访问服务，包括对于持久化数据的增、删、改、查。以Express业务为例，Express业务逻辑需要的服务由ExpressDataService接口提供。由于持久化数据的保存可能存在多种形式：Txt文件、序列化文件、数据库等，所示抽象了数据服务。数据层模块的描述具体如图13所示。



**图13 数据层模块的描述**

### 5.4.1 数据层模块的职责

数据层模块的职责如表17所示。

**表17 数据层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| ExpressDataService | 持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |
| BusinessDataService | 持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |
| IntermediateDataService | 持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |
| FinanceDataService | 持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |
| RepertoryDataService | 持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |
| ManageDataService | 持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |
| UserDataService | 持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |
| ReceiptDataService | 持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |

### 5.4.2 数据层模块的接口规范

express数据层模块的接口规范如表18所示。

**表18 express数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ExpressDataService.getBaseCost | 语法 | public CostBasePO getBaseCost() |
| 前置条件 | 系统中存在订单收费的价格信息信息 |
| 后置条件 | 系统返回订单收费的价格信息 |
| ExpressDataService.chargeCollection | 语法 | public boolean chargeCollection(ExpressPO po) |
| 前置条件 | ExpressPO中存在成员charge需要更改 |
| 后置条件 | 系统更新ExpressPO的charge属性 |
| ExpressDataService.getExpressInfos | 语法 | public ExpressPO getExpressInfos() |
| 前置条件 | 系统中存在ExpressPO的信息 |
| 后置条件 | 系统返回ExpressPO |

business数据层模块的接口规范如表19所示。

**表19 business数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| BusinessDataService.addVehicle | 语法 | public boolean addVehicle(VehiclePO po) |
| 前置条件 | VehiclePOList中需要添加一个VehiclePO |
| 后置条件 | 系统在VehiclePOList中添加一个VehiclePO并返回操作结果 |
| BusinessDataService.getFreeVehicle | 语法 | public ArrayList<VehiclePO> getVehicleInfos() |
| 前置条件 | 系统存在空闲的车辆信息VehiclePO |
| 后置条件 | 系统返回空闲的车辆信息列表VehiclePOList |
| BusinessDataService.deleteVehicle | 语法 | public boolean BusinessDataService.deleteVehicle(VehiclePO po) |
| 前置条件 | 系统中存在改车辆信息VehiclePO |
| 后置条件 | 系统删除该车辆信息，并返回操作结果 |
| BusinessDataService.modifyVehicle | 语法 | public boolean BusinessDataService.modifyVehicle(VehiclePO po) |
| 前置条件 | 系统中存在该车辆信息VehiclePO |
| 后置条件 | 系统修改该车辆信息，并返回操作结果 |
| BusinessDataService.getDriverInfo | 语法 | public DriverPO getDriverInfos(String ID) |
| 前置条件 | 系统中存在该司机信息DriverPO |
| 后置条件 | 系统返回该司机信息DriverPO |
| BusinessDataService.addDriver | 语法 | public boolean addDriver(DriverPO po) |
| 前置条件 | DriverPOList中需要添加一个DriverPO |
| 后置条件 | 系统在DriverPOList中添加一个DriverPO并返回操作结果 |
| BusinessDataService.deleteDriver | 语法 | public boolean deleteDriver(DriverPO po) |
| 前置条件 | 系统中存在该司机信息DriverPO |
| 后置条件 | 系统删除该司机信息DriverPO，并返回操作结果 |
| BusinessDataService.modifyDriver | 语法 | public boolean modifyDriver(DriverPO po) |
| 前置条件 | 系统中存在该车辆信息VehiclePO |
| 后置条件 | 系统修改该车辆信息，并返回操作结果 |
| BusinessDataService.getTransferOrders | 语法 | public ArrayList<OrderVO>getTransferOrders() |
| 前置条件 | 系统中存在待转运的的OrderVO |
| 后置条件 | 系统返回待转运的的OrderVO的列表OrderVOList |
| BusinessDataService.getSendOrder | 语法 | public ArrayList<VehiclePO> getFreeVehicles() |
| 前置条件 | 系统中存在待派送的的OrderVO |
| 后置条件 | 系统返回待派送的的OrderVO的列表OrderVOList |

intermediate数据层模块的接口规范如表20所示。

**表20 intermediate数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| IntermediateDataService. showTransferingReceipt | 语法 | Public TransferingReceiptPO getTransferingReceipt(); |
| 前置条件 | 无 |
| 后置条件 | 返回一个TransferingReceiptPO对象 |
| IntermediateDataService.addOrder | 语法 | public boolean addOrder(String ID); |
| 前置条件 | 同样的ID在TransferingReceiptPO的对象中不存在 |
| 后置条件 | 在外部文件中增加一个OrderPO对象 |
| IntermediateDataService.deleteOrder | 语法 | public boolean deleteOrder(String ID); |
| 前置条件 | 在外部文件中存在一个ID符合的OrderPO对象需要删除 |
| 后置条件 | 删除一个OrderPO对象 |
| IntermediateDataService.modifyOrder | 语法 | public boolean modifyOrder(String ID); |
| 前置条件 | 在外部文件中存在一个ID符合的OrderPO对象需要修改 |
| 后置条件 | 更新一个OrderPO对象 |
| IntermediateDataService.updateTransferingReceipt | 语法 | public void updateTransferingReceipt(TransferingReceiptPO); |
| 前置条件 | 无 |
| 后置条件 | 在外部文件中增加一个TransferingReceiptPO对象 |
| IntermediateDataService.getPlaneList | 语法 | public ArrayList<PlanePO> getPlaneList(); |
| 前置条件 | 无 |
| 后置条件 | 返回一个存储PlanePO的ArrayList对象 |
| IntermediateDataService. getPlane | 语法 | public PlanePO getPlane(String ID); |
| 前置条件 | 无 |
| 后置条件 | 返回一个PlanePO对象 |
| IntermediateDataService.updateWaitingList | 语法 | public ArrayList<OrderPO> updateWaitingList(TranferingReceiptPO); |
| 前置条件 | 无 |
| 后置条件 | 存储OrderPO对象的ArrayList对象更新 |
| IntermediateDataService. enplane | 语法 | public EnplaningReceiptPO enplane(ArrayList<OrderPO>); |
| 前置条件 | 存储OrderPO对象的ArrayList对象更新，且不为空链表 |
| 后置条件 | 存储OrderPO对象的ArrayList对象为空链表 |
| IntermediateDataService. updateEnplaningReceiptList | 语法 | public ArrayList<EnplaningReceiptPO> updateEnplaningReceiptList(EnplaningReceiptPO); |
| 前置条件 | 存储OrderPO对象的ArrayList对象为空链表 |
| 后置条件 | 外部文件中新增一个EnplaningReceiptPO对象 |
| IntermediateDataService. computeFare | 语法 | Public farePO computeFare(ArrayList<EnplaningReceiptPO>); |
| 前置条件 | 同样的EnplaningReceiptPO对象存在 |
| 后置条件 | 返回一个farePO对象 |
| IntermediateDataService. updateFare | 语法 | public boolean updateFare(farePO); |
| 前置条件 | 同样的farePO对象在外部文件中不存在 |
| 后置条件 | 外部文件中新增一个farePO对象 |
| IntermediateDataService.getEnplaningReceipt | 语法 | Public boolean getEnplaningReceipt(ArrayList<EnplaningReceiptPO>); |
| 前置条件 | 存储EnplaningReceiptPO对象的ArrayList对象不为空链表，且同样的EnplaningReceiptPO对象不存在 |
| 后置条件 | ArrayList对象更新 |
| IntermediateDataService. updateEnplaningReceipt | 语法 | Public boolean updateEnplaningReceipt(ArrayList<EnplaningReceiptPO>); |
| 前置条件 | ArrayList对象更新 |
| 后置条件 | 在外部文件中新增ArrayList中所有的EnplaningRecePO对象 |
| IntermediateDataService.getTrainList1 | 语法 | public ArrayList<TrainPO> getTrainList(); |
| 前置条件 | 无 |
| 后置条件 | 返回一个存储TrainPO的ArrayList对象 |
| IntermediateDataService. getTrain | 语法 | public TrainPO getTrain(String ID); |
| 前置条件 | 无 |
| 后置条件 | 返回一个TrainPO对象 |
| IntermediateDataService.updateWaitingList | 语法 | public ArrayList<OrderPO> updateWaitingList(TranferingReceiptPO); |
| 前置条件 | 无 |
| 后置条件 | 存储OrderPO对象的ArrayList对象更新 |
| IntermediateDataService. entrain | 语法 | public EntrainingReceiptPO entrain(ArrayList<OrderPO>); |
| 前置条件 | 存储OrderPO对象的ArrayList对象更新，且不为空链表 |
| 后置条件 | 存储OrderPO对象的ArrayList对象为空链表 |
| IntermediateDataService. updateEntrainingReceiptList | 语法 | public ArrayList<EntrainingReceiptPO> updateEntrainingReceiptList(EntrainingReceiptPO); |
| 前置条件 | 存储OrderPO对象的ArrayList对象为空链表 |
| 后置条件 | 外部文件中新增一个EntrainingReceiptPO对象 |
| IntermediateDataService. computeFare | 语法 | public farePO computeFare(ArrayList<EntrainingReceiptPO>); |
| 前置条件 | 同样的EntrainingReceiptPO对象存在 |
| 后置条件 | 返回一个farePO对象 |
| IntermediateDataService. updateFare | 语法 | public boolean updateFare(farePO); |
| 前置条件 | 同样的farePO对象在外部文件中不存在 |
| 后置条件 | 外部文件中新增一个farePO对象 |
| IntermediateDataService.getEntrainingReceipt | 语法 | public boolean getEntrainingReceipt(ArrayList<EntrainingReceiptPO>); |
| 前置条件 | 存储EntrainingReceiptPO对象的ArrayList对象不为空链表，且同样的EntrainingReceiptPO对象不存在 |
| 后置条件 | ArrayList对象更新 |
| IntermediateDataService. updatEntrainingReceipt | 语法 | public boolean updateEntrainingReceipt(ArrayList<EntrainingReceiptPO>); |
| 前置条件 | ArrayList对象更新 |
| 后置条件 | 在外部文件中新增ArrayList中所有的EntrainingRecePO对象 |
| IntermediateDataService.getTruckList | 语法 | public ArrayList<TruckPO> getTruckList(); |
| 前置条件 | 无 |
| 后置条件 | 返回一个存储TruckPO的ArrayList对象 |
| IntermediateDataService. getTruck | 语法 | public TruckPO getTruck(String ID); |
| 前置条件 | 无 |
| 后置条件 | 返回一个TruckPO对象 |
| IntermediateDataService.updateWaitingList | 语法 | public ArrayList<OrderPO> updateWaitingList(TranferingReceiptPO); |
| 前置条件 | 无 |
| 后置条件 | 存储OrderPO对象的ArrayList对象更新 |
| IntermediateDataService. entruck | 语法 | public EntruckingReceiptPO enTruck(ArrayList<OrderPO>); |
| 前置条件 | 存储OrderPO对象的ArrayList对象更新，且不为空链表 |
| 后置条件 | 存储OrderPO对象的ArrayList对象为空链表 |
| IntermediateDataService. updateEntruckingReceiptList | 语法 | public ArrayList<EntruckingReceiptPO> updateEntruckingReceiptList(EntruckingReceiptPO); |
| 前置条件 | 存储OrderPO对象的ArrayList对象为空链表 |
| 后置条件 | 外部文件中新增一个EntruckingReceiptPO对象 |
| IntermediateDataService. computeFare | 语法 | public farePO computeFare(ArrayList<EntruckingReceiptPO>); |
| 前置条件 | 同样的EntruckingReceiptPO对象存在 |
| 后置条件 | 返回一个farePO对象 |
| IntermediateDataService. updateFare | 语法 | public boolean updateFare(farePO); |
| 前置条件 | 同样的farePO对象在外部文件中不存在 |
| 后置条件 | 外部文件中新增一个farePO对象 |
| IntermediateDataService.getEntruckingReceipt | 语法 | public boolean getEntruckingReceipt(ArrayList<EntruckingReceiptPO>); |
| 前置条件 | 存储EntruckingReceiptPO对象的ArrayList对象不为空链表，且同样的EntruckingReceiptPO对象不存在 |
| 后置条件 | ArrayList对象更新 |
| IntermediateDataService. updatEntruckingReceipt | 语法 | public boolean updateEntruckingReceipt(ArrayList<EntruckingReceiptPO>); |
| 前置条件 | ArrayList对象更新 |
| 后置条件 | 在外部文件中新增ArrayList中所有的EntruckingRecePO对象 |

finance数据层模块的接口规范如表21所示。

**表21 finance数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| FinanceDataService.InitInfo | 语法 | public int InitInfo(InitInfoPO po)throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 在数据库中增加一个 InitInfoPO |
| FinanceDataService.getInitInfo | 语法 | public InitInfoPO getInitInfo（String time）throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回一个 InitInfoPO |
| FinanceDataService.addAccount | 语法 | public int addAccount(AccountPO po) throws RemoteException; |
| 前置条件 | 同样ID的AccountPO在数据库中不存在 |
| 后置条件 | 在数据库中增加一个AccountPO |
| FinanceDataService.deleteAccount | 语法 | public int deleteAccount(AccountPO po) throws RemoteException; |
| 前置条件 | 在数据库中存在同样ID的AccountPO |
| 后置条件 | 删除一个AccountPO |
| FinanceDataService.modifyAccount | 语法 | public int modifyAccount(AccountPO po,String name) throws RemoteException; |
| 前置条件 | 在数据库中存在某个需要被修改的AccountPO |
| 后置条件 | 修改一个AccountPO |
| FinanceDataService.findByName | 语法 | public AccountPO findbyName(String name) throws RemoteException； |
| 前置条件 | 含有该名称的AccountPO在数据库中存在 |
| 后置条件 | 按名称进行查找返回相应的AccountPO |
| FinanceDataService.findByKeyword | 语法 | public ArrayList<AccountPO> findByKeyword(String s) throws RemoteException； |
| 前置条件 | 含有关键字的AccountPO在数据库中存在 |
| 后置条件 | 按关键字进行查找返回相应的AcountPO的ArrayList |
| FinanceDataService.showAll | 语法 | Public ArrayList<AccountPO> showAll() throws RemoteException; |
| 前置条件 | 在数据库中存在AccountPO的记录 |
| 后置条件 | 返回数据库中所有的AccountPO记录 |
| FinanceDataService.showBSL | 语法 | public BusinessstatementReceiptPO showBSL(String beginTime,String endTime) throws RemoteException; |
| 前置条件 | 数据库中存在符合条件的 BusinessstatementReceiptPO |
| 后置条件 | 返回一个BusinessstatementReceiptPO |
| FinanceDataService.showAllBSList | 语法 | public ArrayList<BusinessstatementReceiptPO> showAllBSList()throws RemoteException; |
| 前置条件 | 数据库中存在 BusinessstatementReceiptPO |
| 后置条件 | 返回 BusinessstatementReceiptPO的ArrayList |
| FinanceDataService.export | 语法 | public int export(BusinessstatementReceiptPO po)throws RemoteException; |
| 前置条件 | 数据库中存在 BusinessstatementReceiptPO |
| 后置条件 | 导出 BusinessstatementReceiptPO到excel |
| FinanceDataService.creatCollection | 语法 | public int createCollection(CollectionReceiptPO po) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 存储一张入款单 CollectionReceiptPO |
| FinanceDataService.getAllCollection | 语法 | public ArrayList<CollectionReceiptPO> getAllCollection() throws RemoteException; |
| 前置条件 | 数据库中存在 CollectionReceiptPO记录 |
| 后置条件 | 返回数据库中所有的 CollectionReceiptPO记录 |
| FinanceDataService.getGathering | 语法 | public ArrayList<GatheringReceiptPO> getGathering(String HallID,String Time) throws RemoteException; |
| 前置条件 | 数据库中存在 GatheringReceiptPO记录 |
| 后置条件 | 返回符合条件的 GatheringReceiptPO记录 |
| FinanceDataService.getMoney | 语法 | public double[] getMoney(GatheringReceiptPO po) throws RemoteException; |
| 前置条件 | 已经获得GatheringReceiptPO记录 |
| 后置条件 | 返回GatheringReceiptPO中所有金额 |
| FinanceDataService.getTotalMoney | 语法 | public double getTotalMoney(int[] money) throws RemoteException; |
| 前置条件 | 已经获得GatheringReceiptPO中的所有金额 |
| 后置条件 | 返回所有金额之和 |
| FinanceDataService.getNum | 语法 | public int getNum() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回当天已创建的入款单数 |
| FinanceDataService.findByID | 语法 | public CollectionReceiptPO findByID(String ID) throws RemoteException; |
| 前置条件 | 数据库中存在符合条件的CollectionReceiptPO |
| 后置条件 | 返回相应ID的 CollectionReceiptPO记录 |
| FinanceDataService.modify | 语法 | publicCollectionReceiptPOmodify(CollectionReceiptPO po) throws RemoteException; |
| 前置条件 | 数据库中存在符合条件的 CollectionReceiptPO |
| 后置条件 | 修改相应的 CollectionReceiptPO记录 |
| FinanceDataService.creatPaymentReceipt(PaymentReceiptPO po) | 语法 | public int creatPaymentReceipt(PaymentReceiptPO vo) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 存储一张付款单PaymentReceiptPO |
| FinanceDataService.getAllPaymentReceipt() | 语法 | public ArrayList<PaymentReceiptPO> getAllPaymentReceipt() throws RemoteException; |
| 前置条件 | 数据库中存在 PaymentReceiptPO记录 |
| 后置条件 | 返回数据库中的所有 PaymentReceiptPO记录 |
| FinanceDataService. getNum() | 语法 | public int getNum() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回当天已创建的付款单数 |
| FinanceDataService.findByID(String ID) | 语法 | public PaymentReceiptPO findByID(String ID) throws RemoteException; |
| 前置条件 | 数据库中存在符合ID的 PaymentReceiptPO |
| 后置条件 | 返回相应的 PaymentReceiptPO记录 |
| FinanceDataService.modify(PaymentReceiptPO po) | 语法 | public PaymentReceiptPO modify(PaymentReceiptPO po) throws RemoteException; |
| 前置条件 | 数据库中存在符合条件的PaymentReceiptPO |
| 后置条件 | 修改相应的PaymentReceiptPO记录 |
| FinanceDataService.creatCostIncomeList(CostIncomeReceiptPO po) | 语法 | public int creatCostIncomeList(CostIncomeReceiptPO po) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 存储一张成本收益表CostIncomeReceiptPO |
| FinanceDataService.getAllCostIncomeList() | 语法 | public ArrayList<CostIncomeReceiptPO> getAllCostIncomeList() throws RemoteException; |
| 前置条件 | 数据库中存在 CostIncomeReceiptPO记录 |
| 后置条件 | 返回数据库中的所有 CostIncomeReceiptPO记录 |
| FinanceDataService.getCollection() | 语法 | public ArrayList<CollectionReceiptPO> getCollection() throws RemoteException; |
| 前置条件 | 数据库中存在 CollectionReceiptPO记录 |
| 后置条件 | 返回数据库中的所有 CollectionReceiptPO记录 |
| FinanceDataService.getPayment() | 语法 | public ArrayList<PaymentReceiptPO>getPayment() throws RemoteException; |
| 前置条件 | 数据库中存在 PaymentReceiptPO记录 |
| 后置条件 | 返回数据库中的所有 PaymentReceiptPO记录 |
| FinanceDataService. getNum() | 语法 | public int getNum() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回当天已经创建的成本收益表数 |
| FinanceDataService.findByID(String ID) | 语法 | public CostIncomeReceiptPO findByID(String ID) throws RemoteException; |
| 前置条件 | 数据库中存在符合ID的CostIncomeReceiptPO记录 |
| 后置条件 | 返回相应的 CostIncomeReceiptPO记录 |
| FinanceDataService.modify(CostIncomeReceiptPO po) | 语法 | public CostIncomeReceiptPO modify(CostIncomeReceiptPO po) throws RemoteException; |
| 前置条件 | 数据库中存在符合条件的 CostIncomeReceiptPO |
| 后置条件 | 修改相应的CostIncomeReceiptPO记录 |

repertory数据层模块的接口规范如表22所示。

**表22 repertory数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| RepertoryDataService. modifyRepertory | 语法 | public int modifyRepertory(RepertoryPO repertorypo) |
| 前置条件 | 需要修改仓库信息 |
| 后置条件 | 更新一个po对象 |
| RepertoryDataService. findRepertory | 语法 | public RepertoryPO findRepertory(String repertoryID) |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的RepertoryPO的结果 |
| RepertoryDataService. addGoods | 语法 | public int addGoods(String repertoryID, GoodsPO goodspo) |
| 前置条件 | 无 |
| 后置条件 | 在仓库的库存信息列表中增加一个po记录 |
| RepertoryDataService. deleteGoods | 语法 | public int deleteGoods(String repertoryID, GoodsPO goodspo) |
| 前置条件 | 在仓库的库存信息列表中存在某一个po需要删除 |
| 后置条件 | 删除一个po对象 |
| RepertoryDataService. modifyGoods | 语法 | public int modifyGoods(String repertoryID, GoodsPO goodspo) |
| 前置条件 | 在仓库的库存信息列表中存在某一个po需要更新 |
| 后置条件 | 更新一个po对象 |
| RepertoryDataService. findGoodsbyID | 语法 | public GoodsPO findGoodsbyID(String repertoryID, String JJD-ID) |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的GoodsPO的结果 |
| RepertoryDataService.findGoodsbyDate | 语法 | public GoodsPO findGoodsbyDate(String repertoryID, String beginDate, String endDate) |
| 前置条件 | 无 |
| 后置条件 | 按beginDate和endDate进行查找返回相应的GoodsPO的结果 |
| RepertoryDataService.findGoodsbyTime | 语法 | public GoodsPO findGoodsbyTime(String repertoryID, String time) |
| 前置条件 | 无 |
| 后置条件 | 按time进行查找返回相应的GoodsPO的结果 |

manage数据层模块的接口规范如表23所示。

**表23 manage数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ManageDataService.addOrganization | 语法 | public int addOrganization(OrganizationPO organizationpo) |
| 前置条件 | 同样ID的organizationpo在文件中不存在 |
| 后置条件 | 在外部文件中增加一个organizationpo记录 |
| ManageDataService.deleteOrganization | 语法 | public int deleteOrganization(OrganizationPO organizationpo) |
| 前置条件 | 在外部文件中存在某一个organizationpo需要删除 |
| 后置条件 | 删除一个organizationpo对象 |
| ManageDataService.modifyOrganization | 语法 | public int modifyOrganization(OrganizationPO organizationpo) |
| 前置条件 | 在外部文件中存在某一个organizationpo需要更新 |
| 后置条件 | 更新一个organizationpo对象 |
| ManageDataService.findOrganization | 语法 | public OrganizationPO findOrganization(String organizationID) |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的OrganizationPO的结果 |
| ManageDataService.modifyUser | 语法 | public int modifyUser(UserPO userpo) |
| 前置条件 | 在外部文件中存在某一个userpo需要更新 |
| 后置条件 | 更新一个userpo对象 |
| ManageDataService.addPerWage | 语法 | public int addPerWage(PerWagePO perwagepo) |
| 前置条件 | 同样profession的perwagepo在文件中不存在 |
| 后置条件 | 在外部文件中增加一个perwagepo记录 |
| ManageDataService.deletePerWage | 语法 | public int deletePerWage(PerWagePO perwagepo) |
| 前置条件 | 在外部文件中存在某一个perwagepo需要删除 |
| 后置条件 | 删除一个perwagepo对象 |
| ManageDataService.modifyPerWage | 语法 | public int modifyPerWage(PerWagePO perwagepo) |
| 前置条件 | 在外部文件中存在某一个perwagepo需要更新 |
| 后置条件 | 更新一个perwagepo对象 |
| ManageDataService.findPerWage | 语法 | public double findPerWage(int profession) |
| 前置条件 | 无 |
| 后置条件 | 按profession进行查找返回相应的PerWage的值 |
| ManageDataService.addBasicSalary | 语法 | public int addBasicSalary(BasicSalaryPO basicsalarypo) |
| 前置条件 | 同样profession的basicsalarypo在文件中不存在 |
| 后置条件 | 在外部文件中增加一个basicsalarypo记录 |
| ManageDataService.deleteBasicSalary | 语法 | public int deleteBasicSalary(BasicSalaryPO basicsalarypo) |
| 前置条件 | 在外部文件中存在某一个basicsalarypo需要删除 |
| 后置条件 | 删除一个basicsalarypo对象 |
| ManageDataService.modifyBasicSalary | 语法 | public int modifyBasicSalary(BasicSalaryPO basicsalarypo) |
| 前置条件 | 在外部文件中存在某一个basicsalarypo需要更新 |
| 后置条件 | 更新一个basicsalarypo对象 |
| ManageDataService.findBasicSalary | 语法 | public double findBasicSalary((int profession)) |
| 前置条件 | 无 |
| 后置条件 | 按profession进行查找返回相应的BasicSalary的值 |
| ManageDataService.addCityDistance | 语法 | public int addCityDistance(CityDistancePO citydistancepo) |
| 前置条件 | 同样City A, City B的citydistancepo在文件中不存在 |
| 后置条件 | 在外部文件中增加一个citydistancepo记录 |
| ManageDataService.deleteCityDistance | 语法 | public int deleteCityDistance(CityDistancePO citydistancepo) |
| 前置条件 | 在外部文件中存在某一个citydistancepo需要删除 |
| 后置条件 | 删除一个citydistancepo对象 |
| ManageDataService.modifyCityDistance | 语法 | public int modifyCityDistance(CityDistancePO citydistancepo) |
| 前置条件 | 在外部文件中存在某一个citydistancepo需要更新 |
| 后置条件 | 更新一个citydistancepo对象 |
| ManageDataService.findCityDistance | 语法 | public double findCityDistance(String CityA, String CityB) |
| 前置条件 | 无 |
| 后置条件 | 按CityA, CityB进行查找返回相应的CityDistance的值 |
| ManageDataService.addCost | 语法 | public int addCost(CostPO costpo) |
| 前置条件 | 同样category的costpo在文件中不存在 |
| 后置条件 | 在外部文件中增加一个costpo记录 |
| ManageDataService.deleteCost | 语法 | public int deleteCost(CostPO costpo) |
| 前置条件 | 在外部文件中存在某一个costpo需要删除 |
| 后置条件 | 删除一个costpo对象 |
| ManageDataService.modifyCost | 语法 | public int modifyCost(CostPO costpo) |
| 前置条件 | 在外部文件中存在某一个costpo需要更新 |
| 后置条件 | 更新一个costpo对象 |
| ManageDataService.findCost | 语法 | public double findCost(category) |
| 前置条件 | 无 |
| 后置条件 | 按category进行查找返回相应的Cost的值 |

user数据层模块的接口规范如表24所示。

**表24 user数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserDataService.check | 语法 | public boolean check(String userID, String password) |
| 前置条件 | 无 |
| 后置条件 | 返回用户名和密码是否匹配 |
| UserDataService.add | 语法 | public int add(UserPO userpo) |
| 前置条件 | 同样ID的userpo在文件中不存在 |
| 后置条件 | 在外部文件中增加一个po记录 |
| UserDataService.delete | 语法 | public int delete(UserPO userpo) |
| 前置条件 | 在外部文件中存在某一个po需要删除 |
| 后置条件 | 删除一个userpo对象 |
| UserDataService.modify | 语法 | public int modify(UserPO userpo) |
| 前置条件 | 在外部文件中存在某一个po需要更新 |
| 后置条件 | 更新一个userpo对象 |
| UserDataService.find | 语法 | public UserPO find (String ID) |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的UserPO的结果 |
| UserDataService.showAll | 语法 | public ArrayList<UserPO> showAll() |
| 前置条件 | 无 |
| 后置条件 | 返回所有的UserPO的结果 |

receipt数据层模块的接口规范如表25所示。

**表25 receipt数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| ReceiptDataService.add | 语法 | Public int Add(ReceiptPo po) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 在数据库中增加一个po记录 |
| ReceiptDataService.find | 语法 | Public ReceiptPO find(String id)throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的ReceiptPo()结果 |
| ReceiptDataService.modify | 语法 | Public int Modify(ReceiptPo po)throws RemoteException |
| 前置条件 | 该单据id存在 |
| 后置条件 | 修改一个po |
| ReceiptDataService.show | 语法 | Public ArrayList<ReceiptPO> show()throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回所有的po |
| ReceiptDataService.init | 语法 | Public void init()throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始持久化数据库的使用 |

# 6.信息视角

## 6.1数据持久化对象

系统的PO类就是对应的相关的实体类，在此只做简单的介绍。

* ExpressPO类包括寄件人姓名、寄件人联系方式、收件人姓名、收件人联系方式、起始地、目的地、寄件日期、到达日期，运费等属性。
* BusinessPO包括货物出发地、货物目的地、车辆编号、司机编号、出车日期等属性。
* IntermediatePO包括转运方式、转运时间、经手营业厅业务员等属性。
* FinancePO类包括账户名、账户金额属性。
* ManagePO类包括机构的属性（包括机构的类别、机构编号、机构名称、机构附属仓库）以及一些基础数据（如PerWage、BasicSalary等）的属性（包括职业以及对应金额）。
* RepertoryPO类包括仓库的仓库编号、所属中转中心，对应管理员、最大排号、架号、位号、警戒比例等属性。
* UserPO类包括用户的用户名、用户编号、用户密码、职业，所属机构、对应薪水策略、对应权限、绩点属性。
* ReceiptPO类包括单据的编号、单据类别、单据状态、制定日期、提交日期、审批通过日期等属性。

以UserPO为例，持久化用户对象UserPO的定义如图14所示。



**图14 持久化用户对象UserPO的定义**

## 6.2数据库表

数据库表中包含Express表、Business表、Intermediate表、Finance表、Repertory表、Manage表、User表、Receipt表。