**物流管理系统（LMS） 软件体系结构描述文档**

**作者**

崔浩 杜天蛟 高露 倪安松

更新记录

|  |  |  |  |
| --- | --- | --- | --- |
| 作者 | 修改日期 | 变更原因 | 版本号 |
| 倪安松 | 2015/10/22 | 初稿 | V1.0 |
| 倪安松 | 2015/10/25 | 删除重复内容 | V1.1 |
|  |  |  |  |
|  |  |  |  |

目录

1.引言 4

1.1编制目的 4

1.2词汇表 4

1.3参考资料 4

2.产品概述 4

3.逻辑视角 4

4.组合视角 6

4.1开发包图 6

4.2运行时进程 10

4.3物理部署 11

5.接口视角 11

5.1模块的职责 11

5.2用户界面层的分解 13

5.3业务逻辑层的分解 19

5.4数据层的分解 42

6.信息视角 62

6.1数据持久化对象 62

6.2数据持久化格式 64

## 1.引言

### 1.1编制目的

本报告详细完成对物流管理系统的概要设计，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 |
| LMS | 物流管理系统 | 。。。 |
| 。。。 | 。。。 | 。。。 |

### 1.3参考资料

## 2.产品概述

参考物流管理系统用例文档和物流管理系统软件需求规格说明文档中对产品的概括描述。

## 3.逻辑视角

物流信息管理系统中，选择了分层体系结构的风格，将系统分为3层（展示层、业务逻辑层、数据层）能够很好的示意整个高层抽象。展示层包括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。

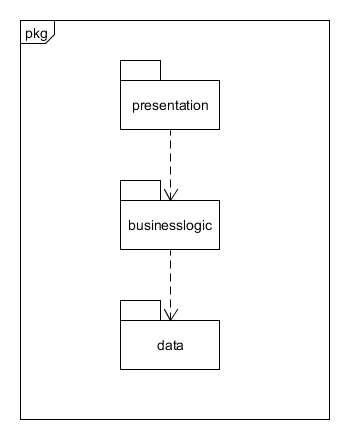


图1 参照体系结构风格的包图表达逻辑视角

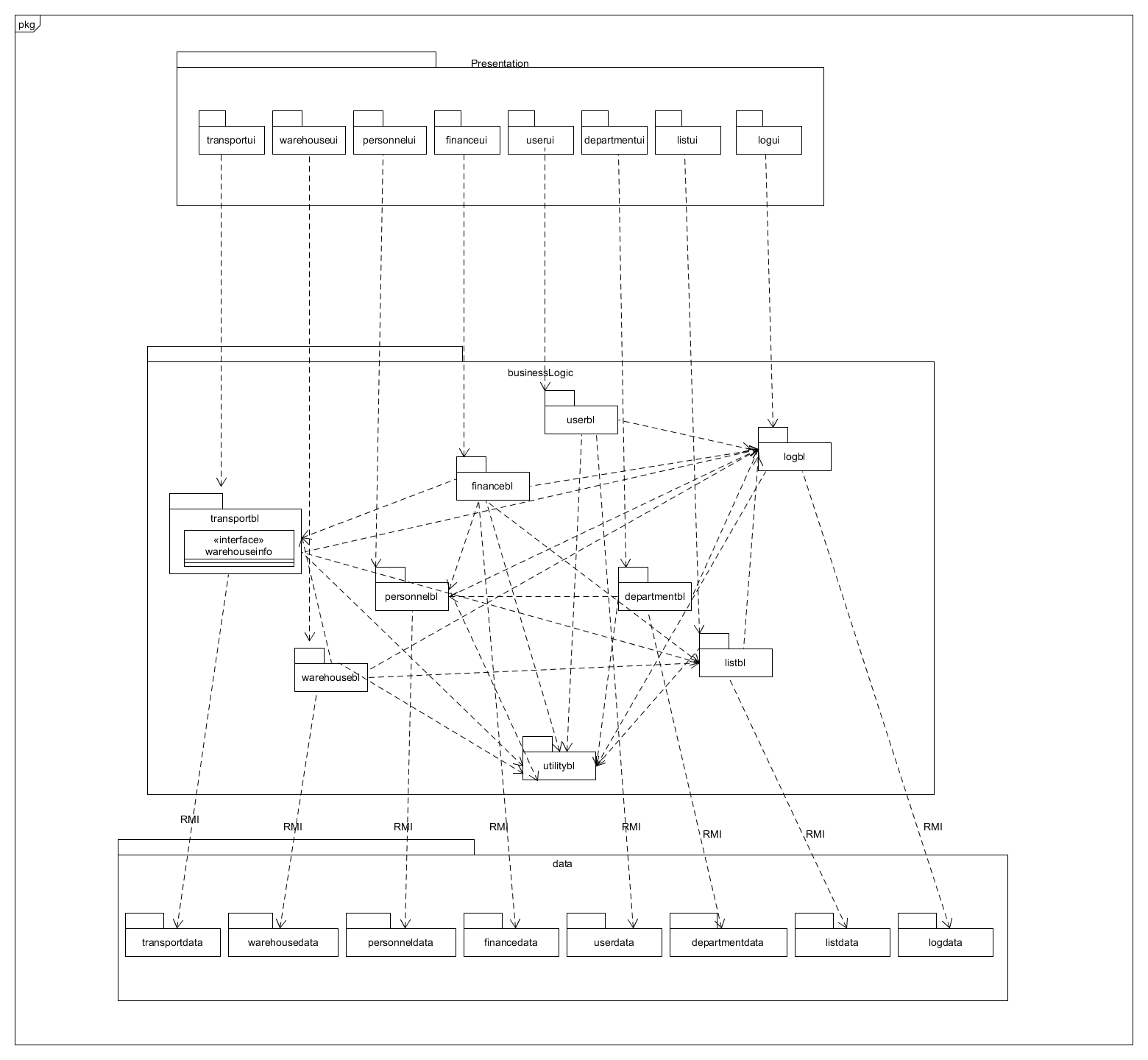


图2 软件体系结构逻辑设计方案

## 4.组合视角

### 4.1开发包图

物流信息管理系统的最终开发包图设计如表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 | Transportui,Warehoseui,Personnelui,Userui,Financeui,  Departmentui,Listui,Logui,Vo |
| Transportui | Transportblservice,界面类库包,Vo |
| Transportblservice | Vo |
| Transportbl | Transportblservice,Transportdata,Warehousebl,Logbl,  Po |
| Transportdataservice | Po,Java RMI |
| Transportdata | Po,Java RMI |
| Warehouseui | Warehouseblservice,界面类库包,Vo |
| Warehouseblservice | Vo |
| Warehousebl | Warehouseblservice,Warehousedata,Transportbl,Listbl,  Logbl,Po |
| Warehousedataservice | Po,Java RMI |
| Warehousedata | Po,Java RMI |
| Personnelui | Personnelblservice,界面类库包,Vo |
| Personnelblservice | Vo |
| Personnelbl | Personnelblservice,Personneldata,Logbl,Po |
| Personneldataservice | Po,Java RMI |
| Personneldata | Po,Java RMI |
| Userui | Userblservice,界面类库包,Vo |
| Userblservice | Vo |
| Userbl | Userblservice,Userdata,Logbl,Po |
| Userdataservice | Po,Java RMI |
| Userdata | Po,Java RMI |
| Financeui | Financeblservice,界面类库包,Vo |
| Financeblservice | Vo |
| Financebl | Financeblservice,Financedata,Transportbl,Personnelbl,  Listbl,Logbl,Po |
| Financedataservice | Po,,Java RMI |
| Financedata | Po,Java RMI |
| Departmentui | Departmentblservice,界面类库包,Vo |
| Departmentblservice | Vo |
| Departmentbl | Departmentbl,Departmentdata,Personnelbl,Logbl,Po |
| Departmentdataservice | Po,Java RMI |
| Departmentdata | Po,Java RMI |
| Listui | Listblservice,界面类库包,Vo |
| Listblservice | Vo |
| Listbl | Listblservice,Listdata,Logbl,Po |
| Listdataservice | Po,Java RMI |
| Listdata | Po,Java RMI |
| Logui | Logblservice,界面类库包,Vo |
| Logblservice | Vo |
| Logbl | Logblservice,Logdata,Po |
| Logdataservice | Po,Java RMI |
| Logdata |  |
| Vo |  |
| Po |  |
| Utilitybl |  |
| 界面类库包 |  |
| Java RMI |  |

物流管理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。

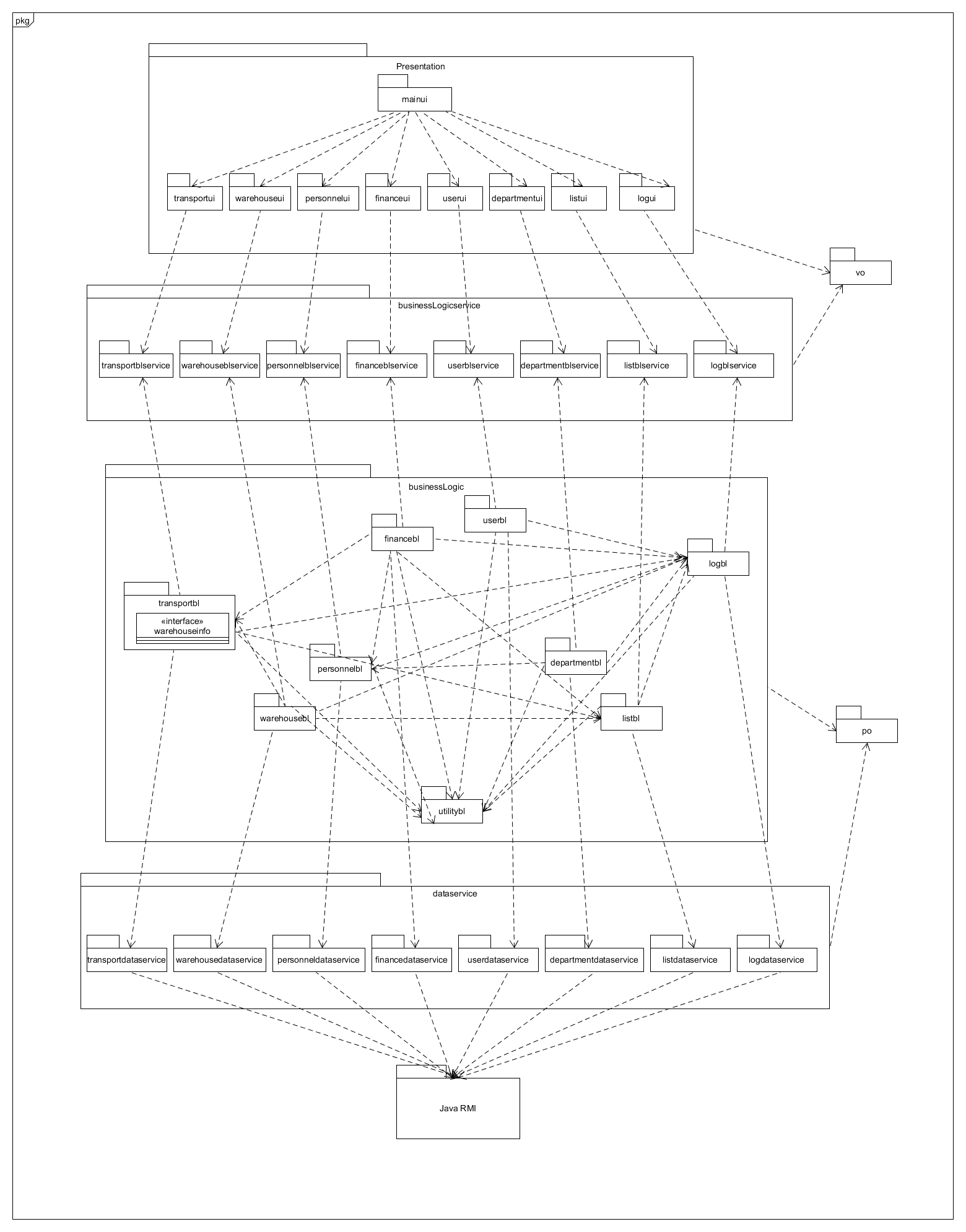


图3物流管理系统客户端开发包图

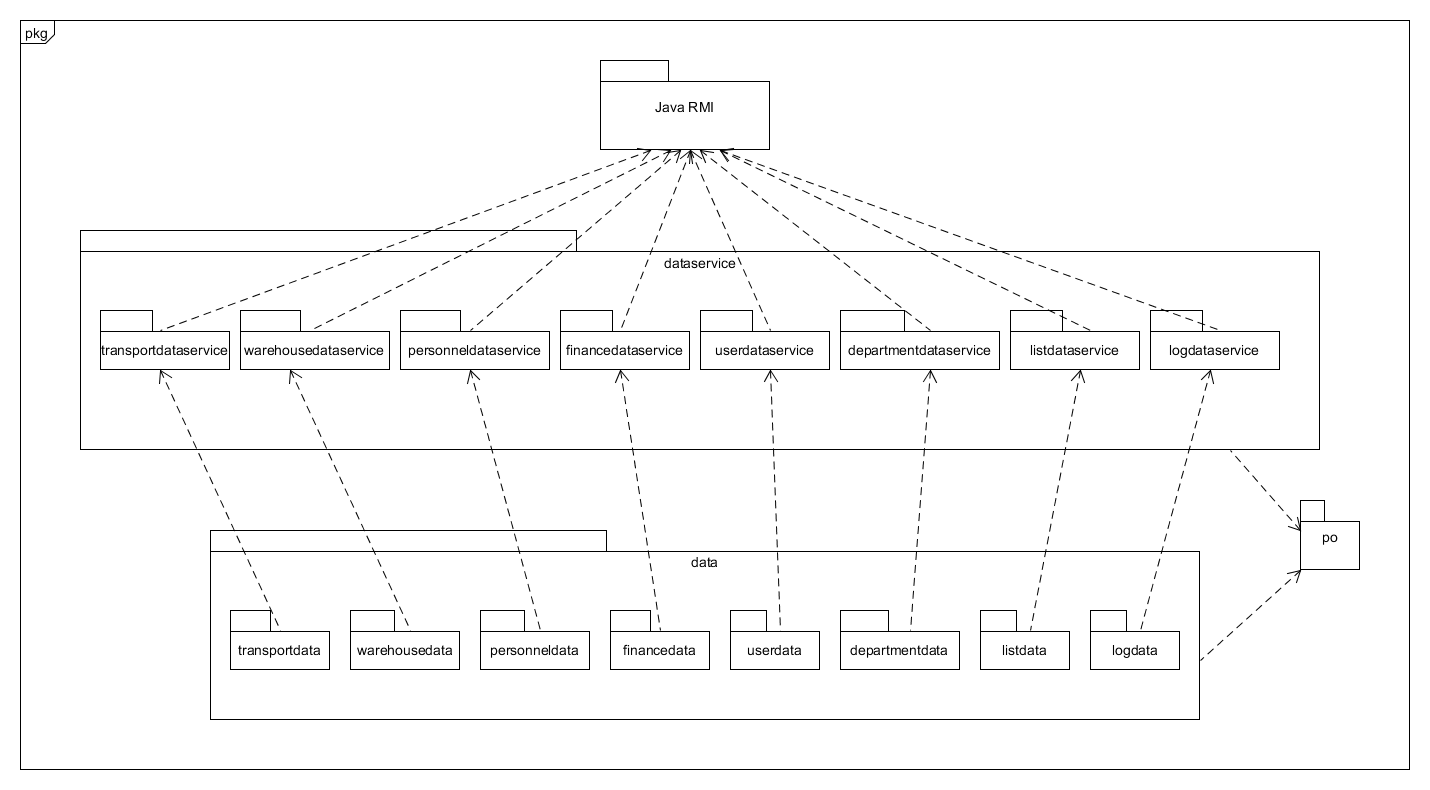


图4物流管理系统服务器端开发包图

### 4.2运行时进程

在物流信息管理系统中，会有多个服务端进程和一个客户端进程，其进程图如图5所示。结合部署图，客户端进程是在客户端上运行，服务器端进程在服务器端机器上运行。



图5 进程图

### 4.3物理部署

物流信息管理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JDK 8.0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。

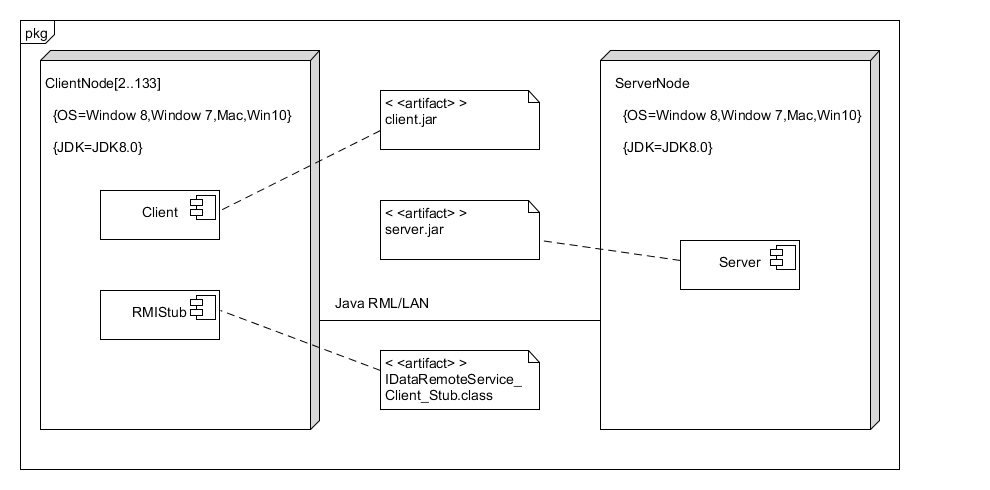


图6 部署图

## 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。

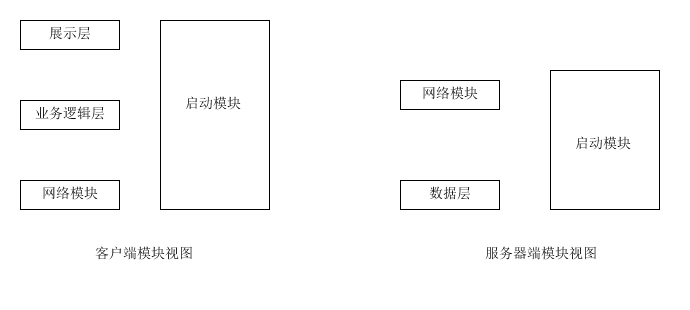


表2 客户端各层的职责

|  |  |
| --- | --- |
| 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 | 基于窗口的物流信息管理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 |

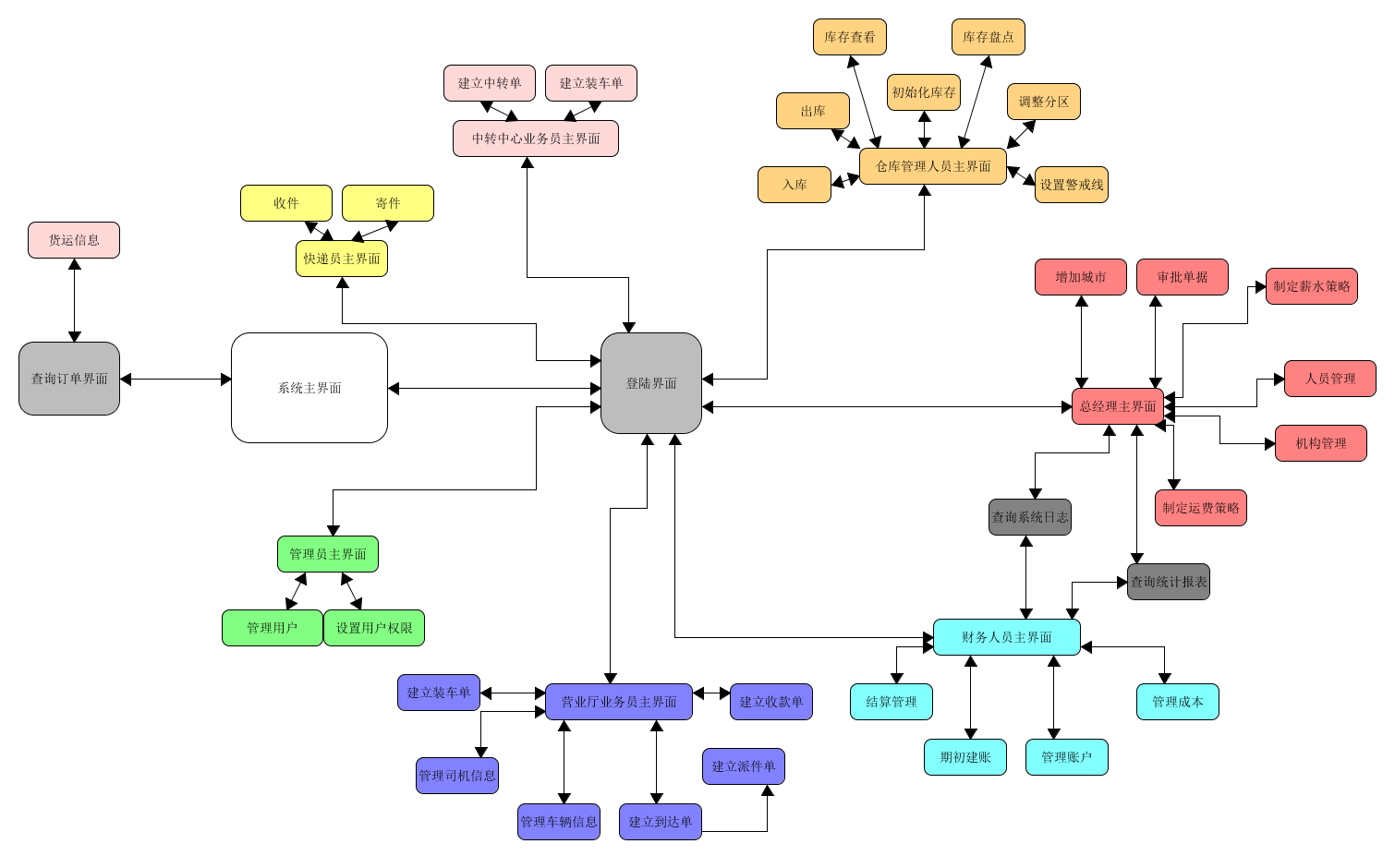
每一层只是使用下方直接接触的层。层与层之间仅仅是通过接口的调用来完成的。层之间调用的接口如表4所示。

表4 层之间调用的接口

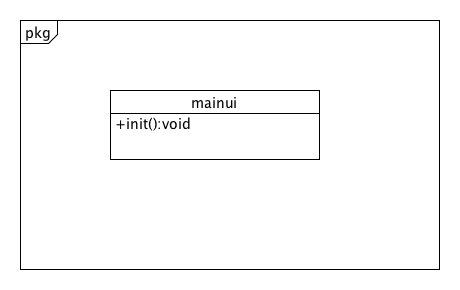
|  |  |  |
| --- | --- | --- |
| 接口 | 服务调用方 | 服务提供方 |
| TransportBLService  WareHouseBLService  UserBLService  PersonnelBLService  FinanceBLService  ListBLService  LogBLService  DepartmentBLService | 客户端展示层 | 客户端业务逻辑层 |
| TransportDataService  WareHouseDataService  UserDataService  PersonnelDataService  FinanceDataService  ListDataService  LogDataService  DepartmentDataService | 客户端业务逻辑层 | 服务器端数据层 |

### 5.2用户界面层的分解

根据需求，系统存在42个用户界面：系统主界面、查询订单界面、货运信息界面、登陆界面、快递员主界面、收件界面、寄件界面、管理员主界面、管理用户界面、设置用户权限界面、中转中心业务员主界面、建立中转单界面、建立装运单界面、营业厅业务员主界面、建立装车单界面、管理司机信息界面、管理车辆信息界面、建立到达单界面、建立派件单界面、建立收款单界面、仓库管理人员主界面、入库界面、出库界面、库存查看界面、初始化库存界面、库存盘点界面、调整分区界面、设置警戒线界面、财务人员主界面、结算管理界面、期初建账界面、管理账户界面、管理成本界面、查看系统日志界面、查询统计报表界面、总经理主界面、增加城市界面、审批单据界面、制定运费策略界面、机构管理界面、人员管理界面和制定薪水策略。界面跳转如图所示。



服务器端和客户端的用户界面设计接口是一致的，只是具体的界面不一样。用户界面类如图所示。



#### 5.2.1用户界面层模块的职责

mainui为界面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和需要的服务接口如表所示。

**Mainui模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Mainui | 语法 | init(args:String[]) |
| 前置条件 | 用户触发要显示界面对应的条件 |
| 后置条件 | 显示frame并加载panel |

**Transportui模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| TransportService. checkOrderInfor(long orderNum) | | 得到查询的订单的信息 |
| TransportService. createSendList(SendVO vo) | | 得到完整的寄件单信息 |
| TransportService. saveSendList(SendVO vo) | | 得到寄件单是否保存成功的信息 |
| TransportService. createReceipt(ReceiveVO vo) | | 得到完整的收件单信息 |
| TransportService.saveReceipt(ReceiveVO vo) | | 得到收件单是否保存成功的信息 |
| TransportService. createLoadList(LoadVO vo) | | 得到完整的装车单信息 |
| TransportService. saveLoadList(LoadVO vo) | | 得到装车单是否保存成功的信息 |
| TransportService. saveTransportList(TransportVO vo) | | 得到装运单是否保存成功的信息 |
| TransportService. createArrivalList(ArrivalState state,long orderNum) | | 得到完整的到达单信息 |
| TransportService. saveArrivalList(ArrivalVO vo) | | 得到到达单是否保存成功的信息 |
| TransportService. createDispatchList(long courierNum) | | 得到完整的派件单信息 |
| TransportService. saveDispatchList(DispatchVO vo) | | 得到派件单是否保存成功的信息 |
| TransportService.showVihicleInfor(long id) | | 得到对应车辆的完整信息 |
| TransportService. addVehicle(VehicleVO vo) | | 得到增加车辆是否成功的信息 |
| TransportService. deleteVehicle(long id) | | 得到删除车辆是否成功的信息 |
| TransportService.modifyVehicle(VehicleVO vo) | | 得到修改车辆信息是否成功的信息 |
| TransportService.showDriverInfor(long id) | | 得到对应司机的完整信息 |
| TransportService. addDriver(DriverVO vo) | | 得到增加司机是否成功的信息 |
| TransportService. deleteDriver(long id) | | 得到删除司机是否成功的信息 |
| TransportService.modifyDriver(DriverVO vo) | | 得到修改司机信息是否成功的信息 |

**Warehouseui模块的接口规范**

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| WarehouseService.createCheckinList(CheckinVO vo,long warehouseNum) | 得到完整的入库单信息 |
| WarehouseService.saveCheckinList(CheckinVO vo,long warehouseNum) | 得到入库单是否保存成功的返回信息 |
| WarehouseService.createCheckoutList(CheckoutVO vo,long warehouseNum) | 得到完整的出库单信息 |
| WarehouseService.saveCheckoutList(CheckoutVO vo,long warehouseNum) | 得到出库单是否保存成功的信息 |
| WarehouseService.checkWarehouseInfor( Calendar start, Calendar end, long warehouseNum) | 得到选择时间段内的库存信息 |
| WarehouseService.exportExcel(nventoryExcelVO vo, long warehouseNum) | 得到导出excel表格是否成功的信息 |
| WarehouseService.setCordon(double cordon, long warehouseNum) | 得到设置警戒线值是否成功的信息 |
| WarehouseService.showPartition(double cordon, long warehouseNum) | 得到对应仓库的分区信息 |
| WarehouseService.modifyPartition(long warehouseNum) | 得到修改分区是否成功的信息 |
| WarehouseService.initialize(PartitionVO vo, double cordon, long warehouseNum) | 得到初始化是否成功的信息 |

**Personnelui模块的接口规范**

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| PersonnelService. getPersonnelInfo(long id) | 得到查询相应人员的信息 |
| PersonnelService. deletePersonnel (long id) | 得到删除人员是否成功的信息 |
| PersonnelService. changePersonnel (PersonnelVO vo) | 得到修改人员信息是否成功的信息 |
| PersonnelService. addPersonnel (PersonnelVO vo) | 得到增加人员是否成功的信息 |

**Financeui模块的接口规范**

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| FinanceService.createDebitnote(DebitVO vo) | 得到完整的收款单信息 |
| FinanceService.saveDebitnote(DebitVO vo) | 得到收款单是否保存成功的信息 |
| FinanceService.showDebitnote(Calendar date, long department) | 得到对应时间段内对应营业厅所有收款单的信息 |
| FinanceService.getDebitsum(Calendar date) | 得到该列表中收款单的收款总和 |
| FinanceService.showDebitnoteList(Calendar date) | 得到对应时间段内所有营业厅的所有收款单信息 |
| FinanceService.showRent(int sum, int year) | 得到完整的租金付款单信息 |
| FinanceService.saveRent(RentVO vo) | 得到租金付款单是否成功保存的信息 |
| FinanceService.showFreight() | 得到当前运费付款单的信息 |
| FinanceService.saveFreight(FreightVO vo) | 得到运费付款单是否成功保存的信息 |
| FinanceService.showSalary(long person) | 得到对应员工的工资信息 |
| FinanceService.showSalarySum(int year,int month) | 得到总的工资付款单的信息 |
| FinanceService.saveSalary(SalaryVO vo) | 得到工资付款单是否保存成功的信息 |
| FinanceService.showEarning() | 得到截止到当前日期的经营情况表的信息 |
| FinanceService.exportEarning(EarningVO vo) | 得到导出excel是否成功的信息 |
| FinanceService.showAccount(long id) | 得到要查询的账户信息 |
| FinanceService.addAccount(AcountVO vo) | 得到增加账户是否成功的信息 |
| FinanceService.deleteAccount(long id) | 得到删除账户是否成功的信息 |
| FinanceService.changeAccount(AcountVO vo) | 得到修改账户是否成功的信息 |

**Userui模块的接口规范**

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| UserService.login | 获得登录验证的结果 |
| UserService.getUserInfo(long id) | 得到查询相应user的信息 |
| UserService.deleteUser(long id) | 得到删除是否成功的信息 |
| UserService.changeUser(UserVO vo) | 得到修改是否成功的信息 |
| UserService.addUser(UserVO vo) | 得到增加是否成功的信息 |

**Departmentui模块的接口规范**

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| DepartmentService. getDepartmentInfo(long id) | 得到查询相应机构的信息 |
| DepartmentService. deleteDepartment (long id) | 得到删除机构是否成功的信息 |
| DepartmentService. changeDepartment (DepartmentVO vo) | 得到修改机构信息是否成功的信息 |
| DepartmentService. addDepartment(DepartmentVO vo) | 得到增加机构是否成功的信息 |

**Listui模块的接口规范**

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| ListService.pass(ListVO List) | 得到单据审批状态修改是否成功的信息 |
| ListService. getListInfo | 得到所有未审批单据的信息 |

**Logui模块的接口规范**

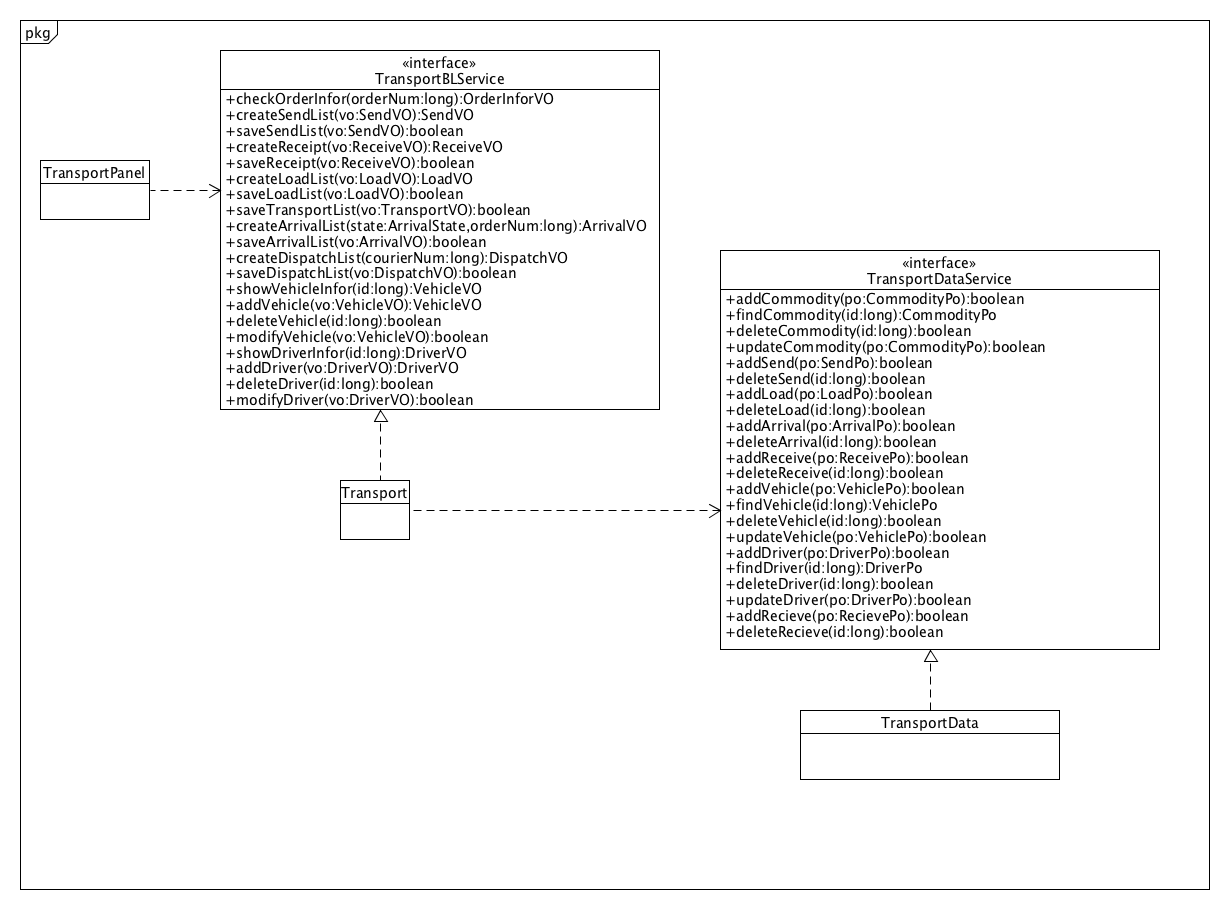
|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| LogService.getLogInfo(Calendar begin, Calendar end) | 得到查找的时间段内的操作记录的信息 |
| LogService. addLog | 得到增加系统日志是否成功的信息 |

#### 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所示（以transport模块为例）。



#### 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所示。

**业务逻辑层模块的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| transportbl | 负责实现运输相关界面所需要的服务 |
| warehousebl | 负责实现仓库管理相关界面所需要的服务 |
| departmentbl | 负责实现机构相关界面所需要的服务 |
| financebl | 负责实现财务相关界面所需要的服务 |
| listbl | 负责实现单据相关界面所需要的服务 |
| logbl | 负责实现系统日志相关界面所需要的服务 |
| personnelbl | 负责实现人员相关界面所需要的服务 |
| userbl | 负责实现用户相关界面所需要的服务 |

#### 5.3.2业务逻辑层模块的接口规范

模块的接口规范如表所示。

transportbl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Transport. checkOrderInfor | 语法 | public OrderInforVO checkOrderInfor(long orderNum); |
| 前置条件 | 订单号符合输入规则 |
| 后置条件 | 查找是否存在相应的订单，根据输入的订单号返回订单查询的结果 |
| Transport. createSendList | 语法 | public SendVO createSendList(SendVO baseMessage); |
| 前置条件 | baseMessage中的信息都符合规范 |
| 后置条件 | 返回给界面完整的寄件单信息 |
| Transport. SaveSendList | 语法 | public Boolean saveSendList(SendVO sendList); |
| 前置条件 | sendList的信息都符合规范且完整 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Transport. createLoadList | 语法 | public LoadVO createLoadList(LoadVO baseMessage); |
| 前置条件 | baseMessage中的信息都符合规范 |
| 后置条件 | 返回给界面完整的装车单信息 |
| Transport. saveLoadList | 语法 | public Boolean saveLoadList(LoadVO loadList); |
| 前置条件 | loadList的信息都符合规范且完整 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Transport. createArrivalList | 语法 | public ArrivalVO createArrivalList(ArrivalState state,long orderNum); |
| 前置条件 | 订单条号符合规范 |
| 后置条件 | 返回给界面完整的到达单信息 |
| Transport. saveArrivalList | 语法 | public Boolean saveArrivalList(ArrivalVO arrivalList); |
| 前置条件 | arrivalList的信息都完整且符合规范 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Transport. createDispatchList | 语法 | public DispatchVO createDispatchList(long courierNum); |
| 前置条件 | 快递员的工号符合规范 |
| 后置条件 | 返回给界面完整的派件单信息 |
| Transport. saveDispatchList | 语法 | public Boolean saveDispatchList(DispatchVO dipatchList); |
| 前置条件 | dispatchList的信息都符合规范 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Transport. addVehicle | 语法 | public VehicleVO addVehicle(long plateNum); |
| 前置条件 | 启动一个增加车辆的操作 |
| 后置条件 | 返回给界面完整的车辆信息 |
| Transport. saveVehicleInfor | 语法 | public Boolean saveVehicleInfor(VehicleVO vehicleInfor); |
| 前置条件 | 车辆信息都符合规范且完整 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Transport. deleteVehicle | 语法 | public Boolean deleteVehicle(long vehicleNum); |
| 前置条件 | 车辆代号符合规范 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Transport. modifyVehicle | 语法 | public Boolean modifyVehicle(VehicleVO modified ); |
| 前置条件 | 修改过后的车辆信息都符合规范且完整 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Transport. checkVehicleInfor | 语法 | public VehicleVO checkVehicleInfor(long vehicleNum); |
| 前置条件 | 车辆代号符合规范 |
| 后置条件 | 返回给界面完整的车辆信息 |
| Transport. addDriver | 语法 | public DriverVO addDriver(DriverVO baseMessage); |
| 前置条件 | 司机的基本信息完整 |
| 后置条件 | 返回给界面完整的司机信息 |
| Transport. saveDriverInfor | 语法 | public Boolean saveDriverInfor(DriverVO driverInfor); |
| 前置条件 | 司机所有信息完整 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Transport. deleteDriver | 语法 | public Boolean deleteDriver(long driverNum); |
| 前置条件 | 司机代号符合规范 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Transport. ModifyDriver | 语法 | public Boolean modifyDriver(DriverVO modified); |
| 前置条件 | 修改过后的司机信息都符合规范且完整 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Transport. checkDriverInfor | 语法 | public DriverVO checkDriverInfor(long driverNum); |
| 前置条件 | 司机代号符合规范 |
| 后置条件 | 返回给界面完整的司机信息 |
| Transport. saveTransportList | 语法 | public Boolean saveTransportList(TransportVO transportInfor); |
| 前置条件 | 装运信息完整 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Transport. | 语法 |  |
| 前置条件 |  |
| 后置条件 |  |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| TransportDataService  .addCommodity (CommodityPO commodity): boolean | 添加快递的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteCommodity (long id): boolean | 根据输入的id，删除该快递的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findCommodity (long id): CommodityPO | 根据输入的id，查找快递的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateCommodity (CommodityPO commodity): boolean | 根据传入的快递的id，找到对对应的快递持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addSend (SendPO Send): boolean | 添加寄件单的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteSend (long id): boolean | 根据输入的id，删除该寄件单的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findSend (long id): SendPO | 根据输入的id，查找寄件单的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateSend (SendPO Send): boolean | 根据传入的寄件单的id，找到对应的寄件单持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addLoad (LoadPO Load): boolean | 添加装运单的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteLoad (long id): boolean | 根据输入的id，删除该装运单的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findLoad (long id): LoadPO | 根据输入的id，查找装运单的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateLoad (LoadPO Load): boolean | 根据传入的装运单的id，找到对应的装运单持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addArrival (ArrivalPO Arrival): boolean | 添加到达单的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteArrival (long id): boolean | 根据输入的id，删除该到达单的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findArrival (long id): ArrivalPO | 根据输入的id，查找到达单的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateArrival (ArrivalPO Arrival): boolean | 根据传入的到达单的id，找到对应的到达单持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addDispatch (DispatchPO Dispatch): boolean | 添加派件单的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteDispatch (long id): boolean | 根据输入的id，删除该派件单的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findDispatch (long id): DispatchPO | 根据输入的id，查找派件单的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateDispatch (DispatchPO Dispatch): boolean | 根据传入的派件单的id，找到对应的派件单持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addVehicle (VehiclePO Vehicle): boolean | 添加司机的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteVehicle (long id): boolean | 根据输入的id，删除该司机的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findVehicle (long id): VehiclePO | 根据输入的id，查找司机的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateVehicle (VehiclePO Vehicle): boolean | 根据传入的司机的id，找到对应的司机持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addDriver (DriverPO Driver): boolean | 添加车辆的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteDriver (long id): boolean | 根据输入的id，删除该车辆的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findDriver (long id): DriverPO | 根据输入的id，查找车辆的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateDriver (DriverPO Driver): boolean | 根据传入的车辆的id，找到对应的车辆持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addWarehouse (WarehousePO Warehouse): boolean | 添加仓库的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteWarehouse (long id): boolean | 根据输入的id，删除该仓库的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findWarehouse (long id): WarehousePO | 根据输入的id，查找仓库的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateWarehouse (WarehousePO Warehouse): boolean | 根据传入的仓库的id，找到对应的仓库持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addCheckin (CheckinPO Checkin): boolean | 添加入库单的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteCheckin (long id): boolean | 根据输入的id，删除该入库单的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findCheckin (long id): CheckinPO | 根据输入的id，查找入库单的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateCheckin (CheckinPO Checkin): boolean | 根据传入的入库单的id，找到对应的入库单持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addCheckout (CheckoutPO Checkout): boolean | 添加出库单的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteCheckout (long id): boolean | 根据输入的id，删除该出库单的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findCheckout (long id): CheckoutPO | 根据输入的id，查找出库单的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateCheckout (CheckoutPO Checkout): boolean | 根据传入的出库单的id，找到对应的出库单持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addReceive (ReceivePO Receive): boolean | 添加收件单的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteReceive (long id): boolean | 根据输入的id，删除该收件单的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findReceive (long id): ReceivePO | 根据输入的id，查找收件单的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateReceive (ReceivePO Receive): boolean | 根据传入的收件单的id，找到对应的收件单持久化对象进行更新，并返回是否成功更新该持久化对象 | |

Warehousebl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Warehouse. createCheckinList | 语法 | public CheckinVO createCheckinList(CheckinVO baseMessage,long warehouseNum); |
| 前置条件 | 入库单的基本信息符合规范 |
| 后置条件 | 返回界面完整的入库单信息 |
| Warehouse.  saveCheckinList | 语法 | public Boolean saveCheckinList(CheckinVO checkinList, long warehouseNum); |
| 前置条件 | 入库单的所有信息都符合规范 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Warehouse. createCheckoutList | 语法 | public CheckoutVO createCheckoutList(CheckoutVO baseMessage, long warehouseNum); |
| 前置条件 | 出库单的基本信息符合规范 |
| 后置条件 | 返回界面完整的出库单信息 |
| Warehouse. saveCheckoutList | 语法 | Public Boolean saveCheckoutList(CheckoutVO checkoutList, long warehouseNum); |
| 前置条件 | 出库单的所有信息都符合规范 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Warehouse. checkWarehouseInfor | 语法 | public InventoryInforVO checkWarehouseInfor(Calendar start, Calendar end, long warehouseNum); |
| 前置条件 | 时间段的设定符合规范 |
| 后置条件 | 返回该时间段的库存信息 |
| Warehouse. exportExcel | 语法 | public Boolean exportExcel(InventoryExcelVO, long warehouseNum); |
| 前置条件 | 用户已进行过当天的库存查看 |
| 后置条件 | 导出excel表格 |
| Warehouse. setCordon | 语法 | public Boolean setCordon(double cordon, long warehouseNum); |
| 前置条件 | 警戒线的值符合规范 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Warehouse. showPartition | 语法 | public PartitionVO showPartition(long warehouseNum); |
| 前置条件 | 库存处于报警状态 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Warehouse. modifyPartition | 语法 | public Boolean modifyPartition(PartitionVO modified, long warehouseNum ); |
| 前置条件 | 修改过的分区信息符合规范 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Warehouse. initialize | 语法 | public Boolean initialize (PartitionVO, double cordon, long warehouseNum); |
| 前置条件 | 分区信息和警戒线都符合规范 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| WarehouseDataService  .addCheckin(CheckinPo checkin):boolean | 根据checkinPO增加入库信息持久化对象 | |
| WarehouseDataService  .addCheckout(CheckoutPo checkout):boolean | 根据checkoutPO增加出库信息持久化对象 | |
| WarehouseDataService  .findWarehouse(long id):WarehousePo | 根据id返回仓库信息持久化对象 | |
| WarehouseDataService  .updateWarehouse(WarehousePo warehouse):boolean | 查找相应的仓库持久化对象并替换成传入的仓库持久化对象 | |
| WarehouseDataService  .addWarehouse(WarehousePo warehouse):boolean | 根据warehousePO增加仓库信息持久化对象 | |

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Department.getDepartmentInfo | 语法 | | Public DepartmentPO getDepartInfo(long id) |
| 前置条件 | | 无 |
| 后置条件 | | 查找是否存在相应的Department，如果存在返回相应的Department信息 |
| Department.deleteDepartment | 语法 | | Public Boolean deleteDepartment(long id) |
| 前置条件 | | 无 |
| 后置条件 | | 查找是否存在相应的Department， 如果删除成功返回true，否则返回false |
| Department.changeDepartment | 语法 | | Public Boolean changeDepartment(DepartmentVO Department) |
| 前置条件 | | 无 |
| 后置条件 | | 根据id查找Department，并修改相应的数据，如果修改成功返回true，否则返回false |
| Department.addDepartment | 语法 | | Public Boolean addDepartment(DepartmentVO Department) |
| 前置条件 | | 无 |
| 后置条件 | | 如果增加成功返回true，否则返回false |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| DepartmentDataService  .addDepartment(DepartmentPo department)：boolean | | 增加机构的单一持久化对象 | |
| DepartmentDataService  .findDepartment(long id):DepartmentPo | | 根据id查找机构的单一持久化对象并返回 | |
| DepartmentDataService  .deleteDepartment(long id):boolean | | 根据id删除机构的单一持久化对象 | |
| DepartmentDataService  .updateDepartment(DepartmentPo department):boolean | | 查找相应的机构持久化对象并替换成传入的机构持久化对象 | |

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Finance. checkUser | 语法 | public boolean isValid (UserVo user); |
| 前置条件 | User的工号不为null |
| 后置条件 | 如果该财务人员具有高级权限，返回true，否则返回false |
| Finance. createDebitnote | 语法 | Public DebitVO createDebitNote(DebitVO debit) |
| 前置条件 | Debit符合规范 |
| 后置条件 | 返回给界面完整的付款单信息 |
| Finance. saveDebitnote | 语法 | Public Boolean saveDebitnote(DebitVO debit) |
| 前置条件 | 无 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Finance. show Debitnote | 语法 | public ArrayList<debitnoteVO> showDebitnote (Calendar date, long department) |
| 前置条件 | Department编号存在 |
| 后置条件 | 返回符合该时间段内所有收款单信息 |
| Finance.getDebitSum | 语法 | public long getDebitSum(Calendar date) |
| 前置条件 | debitnote 不为空 |
| 后置条件 | 返回该列表中收款单的收款总和 |
| Finance. show DebitnoteList | 语法 | public ArrayList<debitnoteVO> showDebitnoteList (Calendar date) |
| 前置条件 | 无 |
| 后置条件 | 返回该时间段各个营业厅的收款单 |
| Finance. showRent | 语法 | public RentVO showRent(int sum, int year) |
| 前置条件 | 无 |
| 后置条件 | 返回完整的租金付款单信息 |
| Finance. saveRent | 语法 | public boolean saveRent(RentVO rent) |
| 前置条件 | 已确认该付款单 |
| 后置条件 | 结束此次管理，持久化更新设计的领域对象的数据 |
| Finance. showFreight | 语法 | Public FreightVO showFreight() |
| 前置条件 | 无 |
| 后置条件 | 返回当前的运费付款单 |
| Finance. changeFreight | 语法 | Public boolean ­­changeFreight(int freight) |
| 前置条件 | 无 |
| 后置条件 | 如果修改成功，返回true，否则返回false |
| Finance. saveFreight | 语法 | Public boolean saveFreight(FreightVO freight) |
| 前置条件 | 已确认运费付款单 |
| 后置条件 | 结束此次管理，持久化更新涉及的领域对象的数据 |
| Finance. showSalary | **语法** | public pesonVO showSalary(long person) |
| 前置条件 | 员工id符合输入规则 |
| 后置条件 | 返回该员工的工资信息 |
| Finance. showSalarySum | 语法 | Public SalaryVO showSalarySum(int year, int month) |
| 前置条件 | 无 |
| 后面条件 | 返回该月的工资付款单信息 |
| Finance. showEarning | 语法 | Public EarningVO showEarning() |
| 前置条件 | 无 |
| 后置条件 | 返回截止到当前日期的经营情况表 |
| Finance. exportEarning | 语法 | Public Boolean saveEarning(EarningVO earnings) |
| 前置条件 | 无 |
| 后置条件 | 如果导出成功，返回true，否则返回false |
| Finance. addAccount | 语法 | Public Boolean addAccount(AccountVO account) |
| 前置条件 | AccoutVO不为null |
| 后置条件 | 如果增加成功，返回true， 否则返回false |
| Finance. showAccount | 语法 | Public AccountPO showAccount(long id) |
| 前置条件 | 输入的id符合规范 |
| 后置条件 | 返回查询的账户信息 |
| Finance. deleteAccount | 语法 | Public Boolean deleteAccount(long id) |
| 前置条件 | 输入的id符合规范 |
| 后置条件 | 如果删除成功返回true，否则返回false |
| Finance. changeAccount | 语法 | Public Boolean changeAccount(AccountVO account) |
| 前置条件 | Account不为null |
| 后置条件 | 如果修改成功返回true，否则返回false |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| FinanceDataService  .addReceipt(ReceiptPo Receipt): boolean | 增加收款单的单一持久化对象 | |
| FinanceDataService  .findReceipt(Calendar date):List<ReceiptPo> | 根据输入的时间查找收款单的单一持久化对象，返回所有符合条件的收款单的列表 | |
| FinanceDataService  .deleteReceipt(long id):boolean | 根据输入的id查找收款单并删除该单一持久化对象 | |
| FinanceDataService  .updateReceipt(ReceiptPo receipt):boolean | 根据输入的收款单id查找收款单并用传入的收款单替换该单一持久化对象 | |
| FinanceDataService  .addPayment(PaymentPo payment):Boolean | 增加新的付款单单一持久化对象 | |
| FinanceDataService  .findPayment(Calendar date):List<PaymentPo> | 根据输入的date查找符合条件的付款单单一持久化对象 | |
| FinanceDataService  .deletePayment(long id):boolean | 根据输入的id查找到相应的单一持久化对象并删除 | |
| FinanceDataService  .updatePayment(PaymentPo Payment):boolean | 根据传入的PaymentPo的id查找到该单一持久化对象并替换 | |
| FinanceDataService  .addFreight(FreightPo freight):boolean | 增加新的freight单一持久化对象 | |
| FinanceDataService  .findFreight(Calendar date):List<FreightPo> | 根据输入的日期查找符合条件的单一持久化对象 | |
| FinanceDataService  .deleteFreight(long id):boolean | 按id查找相应的FreghtPo结果并删除 | |
| FinanceDataService  .updateFreight(FreightPo freight):boolean | 查找到相应的FreightPo并用传入的Freight替换 | |
| FinanceDataService  .findEarning(Calendar date):List<EarningPo> | 按date进行查找返回符合条件的成本收益表单一持久化对象 | |
| FinanceDataService  .deleteEarning(long id):boolean | 按id查找相应的持久化对象并删除 | |
| FinanceDataService  .updateEarning(EarningPo earning) | 查找到相应的EarningPo并用传入的Earning替换 | |
| FinanceDataService  .addAccount(AccountPo account):boolean | 增加新的银行账户单一持久化对象 | |
| FinanceDataService  .findAccount(long id):AccountPo | 按id进行查找返回相应的AccountPo结果 | |
| FinanceDataService  .deleteAccount(long id):boolean | 按id查找到相应的AccountPo单一持久化对象结果并删除 | |
| FinanceDataService  .updateAccount(AccountPo account) | 按id查找到AccountPo的单一持久化对象并替换 | |

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| List.getListInfo | 语法 | Public ArrayList<ListPO> getListInfo() | |
| 前置条件 | 无 | |
| 后置条件 | 查找是否存在未被审批的单据，返回所有未被审批的单据的引用 | |
| List.pass | 语法 | Public Boolean changeList(ListVO List) | |
| 前置条件 | 无 | |
| 后置条件 | 根据id查找List，并修改相应的数据，如果修改成功返回true，否则返回false | |
| 需要的服务（需接口） | | | |
| 服务名 | 服务 | | |
| ListDataService  .updateList(long id，State state)：boolean | 查找该id的list单一持久化对象并将该list的状态改为传入的state | |
| ListDataService  .findList():List<ListPo> | 查找未被审批的单据并返回单据的单一持久化对象 | |

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Log.getLogInfo | 语法 | Public Arraylist<OperationVO> getLogInfo(Calendar begin, Calendar end) |
| 前置条件 | Begin的时间在end之前 |
| 后置条件 | 查找该时间段内的操作记录并返回 |
| Log.addLog | 语法 | Public Boolean addLog(OperationVO op) |
| 前置条件 | 无 |
| 后置条件 | 如果增加成功返回true，否则返回false |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| LogDataService  .addLog(OperationPo operation):boolean | 增加传入的操作单一持久化对象 | |
| LogDataService  .findLog(Calendar begin, Calendar end):List<OperationPo> | 查找传入的时间段内的操作持久化对象并返回 | |

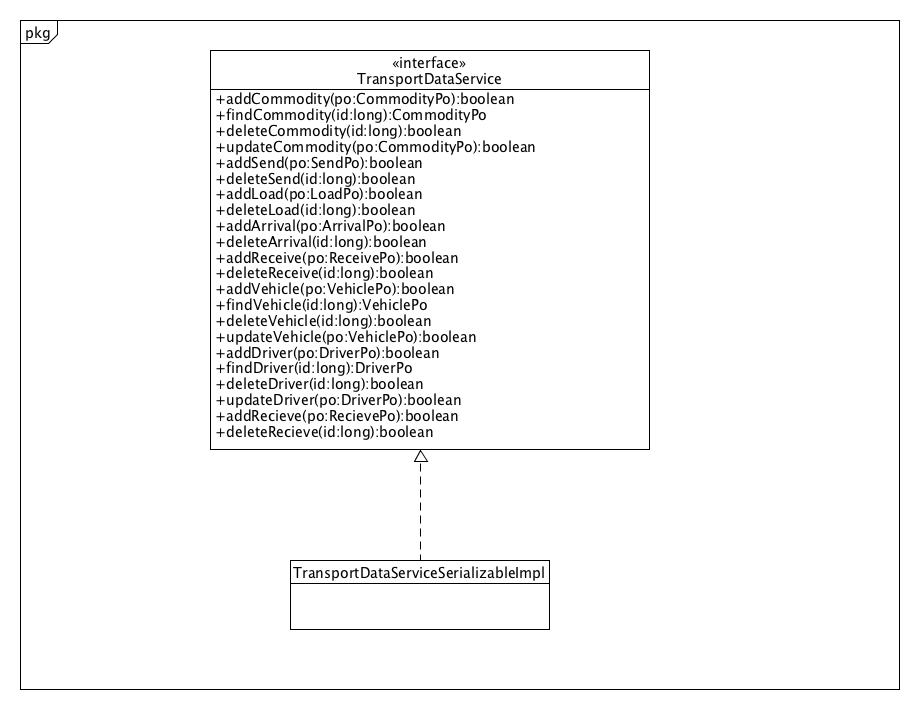
Personnelbl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Personnel. getPersonnelInfo | 语法 | public PersonnelPO getPersonInfo(long id) |
| 前置条件 | ID符合规范 |
| 后置条件 | 查找是否存在相应的Personnel，如果存在返回相应的Personnel信息 |
| Personnel. deletePersonnel | 语法 | public Boolean deletePersonnel(long id) |
| 前置条件 | ID符合规范 |
| 后置条件 | 查找是否存在相应的Personnel， 如果删除成功返回true，否则返回false |
| Personnel. changePersonnel | 语法 | Public Boolean changePersonnel(PersonnelVO Personnel) |
| 前置条件 | 更改过的人员信息符合规范 |
| 后置条件 | 根据id查找Personnel，并修改相应的数据，如果修改成功返回true，否则返回false |
| Personnel. addPersonnel | 语法 | public Boolean addPersonnel(PersonnelVO Personnel) |
| 前置条件 | 无 |
| 后置条件 | 如果增加成功返回true，否则返回false |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| PersonnelDataService  .findPersonnel(long id):PersonnelPo | 根据id返回一个人员持久化对象 | |
| PersonnelDataService  .deletePersonnel(long id):boolean | 根据id删除一个人员持久化对象 | |
| PersonnelDataService  .updatePersonnel(PersonnelPo personnel):boolean | 查找相应的人员持久化对象并替换掉 | |
| PersonnelDataService  .addPersonnel(PersonnelPo personnel):boolean | 根据传入信息增加一个人员持久化对象 | |

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| User.login | 语法 | Public String login(long id, String password) |
| 前置条件 | Password符合输入规则 |
| 后置条件 | 查找是否存在相应的User，根据输入的password返回输入的结果 |
| User.getUserInfo | 语法 | Public UserPO getUserInfo(long id) |
| 前置条件 | 无 |
| 后置条件 | 查找是否存在相应的user，如果存在返回相应的user信息 |
| User.deleteUser | 语法 | Public Boolean deleteUser(long id) |
| 前置条件 | 无 |
| 后置条件 | 查找是否存在相应的user， 如果删除成功返回true，否则返回false |
| User.changeUser | 语法 | Public Boolean changeUser(UserVO user) |
| 前置条件 | 无 |
| 后置条件 | 根据id查找user，并修改相应的数据，如果修改成功返回true，否则返回false |
| User.addUser | 语法 | Public Boolean addUser(UserVO user) |
| 前置条件 | 无 |
| 后置条件 | 如果增加成功返回true，否则返回false |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| UserDataService  .findUser(long id):UserPO | 根据输入的id返回一个用户持久化对象 | |
| UserDataService  .deleteUser(long id):boolean | 根据输入的id删除一个用户持久化对象 | |
| UserDataService  .updateUser(UserPo user):boolean | 根据输入的信息修改原有的用户持久化对象 | |
| UserDataService  .addUser(UserPo user):boolean | 根据输入的信息增加用户持久化对象 | |

### 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。持久化数据保存采用序列化文件。数据层模块的具体描述如图（以transport模块为例）。



#### 5.4.1数据层模块的职责

数据层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| \*DataService | 持久化数据的接口，提供集体载入、集体保存、增删改查服务。 |
| \*DataServiceSerializableFileImpl | 基于序列化文件的持久化数据的接口，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

数据层模块的接口规范如表所示。

**TransportData的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| TransportDataService  .addCommodity | 语法 | public boolean addCommodity(CommodityPo commodity) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| TransportDataService  .findCommodity | 语法 | Public CommodityPo findCommodity(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id进行查找返回相应的CommodityPo结果 |
| TransportDataService  .deleteCommodity | 语法 | Public boolean deleteCommodity(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的CommodityPo结果并删除 |
| TransportDataService  .updateCommodity | 语法 | Public Boolean updateCommodity(CommodityPo commodity) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 查找到相应的CommodityPo并用传入的commodity替换 |
| TransportDataService  .addSend | 语法 | public boolean addSend(SendPo send) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| TransportDataService  .deleteSend | 语法 | Public boolean deleteSend(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的LoadPo结果并删除 |
| TransportDataService  .addLoad | 语法 | public boolean addLoad(LoadPo load) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| TransportDataService  .deleteLoad | 语法 | Public boolean deleteLoad(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的LoadPo结果并删除 |
| TransportDataService  .addArrival | 语法 | public boolean addArrival(ArrivalPo arrival) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| TransportDataService  .deleteArrival | 语法 | Public boolean deleteArrival(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的ArrivalPo结果并删除 |
| TransportDataService  .addDispatch | 语法 | public boolean addDispatch(DispatchPo Dispatch) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| TransportDataService  .deleteDispatch | 语法 | Public boolean deleteDispatch(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的DispatchPo结果并删除 |
| TransportDataService  .addVehicle | 语法 | public boolean addVehicle(VehiclePo vehicle) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| TransportDataService  .findVehicle | 语法 | Public VehiclePo findVehicle(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id进行查找返回相应的VehiclePo结果 |
| TransportDataService  .deleteVehicle | 语法 | Public boolean deleteVehicle(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的vehiclePo结果并删除 |
| TransportDataService  .updateVehicle | 语法 | Public Boolean updateVehicle(VehiclePo vehicle) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 查找到相应的vehiclePo并用传入的vehicle替换 |
| TransportDataService  .addDriver | 语法 | public boolean addDriver(DriverPo driver) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| TransportDataService  .findDriver | 语法 | Public DriverPo findDriver(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id进行查找返回相应的driverPo结果 |
| TransportDataService  .deleteDriver | 语法 | Public boolean deleteDriver(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的driverPo结果并删除 |
| TransportDataService  .updateDriver | 语法 | Public Boolean updateDriver(DriverPo driver) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 查找到相应的driverPo并用传入的driver替换 |
| TransportDataService  .addRecieve | 语法 | public boolean addRecieve(RecievePo recieve) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| TransportDataService  .deleteRecieve | 语法 | Public boolean deleteRecieve(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的recievePo结果并删除 |

**WarehouseData的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| WarehouseDataService  .addWarehouse | 语法 | public boolean addWarehouse(WarehousePo warehouse) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| WarehouseDataService  .findWarehouse | 语法 | Public WarehousePo findWarehouse(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id进行查找返回相应的warehousePo结果 |
| WarehouseDataService  .deleteWarehouse | 语法 | Public boolean deleteWarehouse(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的warehousePo结果并删除 |
| WarehouseDataService  .updateWarehouse | 语法 | Public Boolean updateWarehouse(WarehousePo warehouse) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 查找到相应的WarehousePo并用传入的Warehouse替换 |
| WarehouseDataService  .addCheckin | 语法 | public boolean addCheckin(CheckinPo checkin) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| WarehouseDataService  .findCheckin | 语法 | Public CheckinPo findCheckin(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id进行查找返回相应的CheckinPo结果 |
| WarehouseDataService  .deleteCheckin | 语法 | Public boolean deleteCheckin(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的CheckinPo结果并删除 |
| WarehouseDataService  .updateCheckin | 语法 | Public Boolean updateCheckin(CheckinPo Checkin) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 查找到相应的CheckinPo并用传入的Checkin替换 |
| WarehouseDataService  .addCheckout | 语法 | public boolean addCheckout(CheckoutPo checkout) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| WarehouseDataService  .findCheckout | 语法 | Public CheckoutPo findCheckout(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id进行查找返回相应的CheckoutPo结果 |
| WarehouseDataService  .deleteCheckout | 语法 | Public boolean deleteCheckout(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的CheckoutPo结果并删除 |
| WarehouseDataService  .updateCheckout | 语法 | Public Boolean updateCheckout(CheckoutPo Checkout) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 查找到相应的CheckoutPo并用传入的Checkout替换 |

**UserData的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserDataService  .addUser | 语法 | public boolean addUser(UserPo user) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| UserDataService  .findUser | 语法 | Public UserPo findUser(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id进行查找返回相应的UserPo结果 |
| UserDataService  .deleteUser | 语法 | Public boolean deleteUser(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的UserPo结果并删除 |
| UserDataService  .updateUser | 语法 | Public Boolean updateUser(UserPo user) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 查找到相应的UserPo并用传入的user替换 |

**PersonnelData的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PersonnelDataService  .addPersonnel | 语法 | public boolean addPersonnel(PersonnelPo personnel) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| PersonnelDataService  .findPersonnel | 语法 | Public PersonnelPo findPersonnel(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id进行查找返回相应的PersonnelPo结果 |
| PersonnelDataService  .deletePersonnel | 语法 | Public boolean deletePersonnel(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的PersonnelPo结果并删除 |
| PersonnelDataService  .updatePersonnel | 语法 | Public Boolean updatePersonnel(PersonnelPo personnel) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 查找到相应的PersonnelPo并用传入的Personnel替换 |

**DepartmentData的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DepartmentDataService  .addDepartment | 语法 | public boolean addDepartment(DepartmentPo Department) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| DepartmentDataService  .findDepartment | 语法 | Public DepartmentPo findDepartment(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id进行查找返回相应的DepartmentPo结果 |
| DepartmentDataService  .deleteDepartment | 语法 | Public boolean deleteDepartment(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的DepartmentPo结果并删除 |
| DepartmentDataService  .updateDepartment | 语法 | Public Boolean updateDepartment(DepartmentPo Department) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 查找到相应的DepartmentPo并用传入的Department替换 |
| DepartmentDataService  .addCity | 语法 | public boolean addCity(CityPo City) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| DepartmentDataService  .findCity | 语法 | Public CityPo findCity(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id进行查找返回相应的CityPo结果 |
| DepartmentDataService  .deleteCity | 语法 | Public boolean deleteCity(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的CityPo结果并删除 |
| DepartmentDataService  .updateCity | 语法 | Public Boolean updateCity(CityPo City) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 查找到相应的CityPo并用传入的City替换 |

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| LogDataService  .addLog | 语法 | public boolean addOperation(OperationPo Operation) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| LogDataService  .findLog | 语法 | Public List<OperationPo> findOperation(Calendar begin, Calendar end) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id进行查找返回相应的OperationPo结果 |

**LogData的接口规范**

**FinanceData的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| FinanceDataService  .addReceipt | 语法 | public boolean addReceipt(ReceiptPo Receipt) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| FinanceDataService  .findReceipt | 语法 | Public List<ReceiptPo> findReceipt(Calendar date) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id进行查找返回相应的ReceiptPo结果 |
| FinanceDataService  .deleteReceipt | 语法 | Public boolean deleteReceipt(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的ReceiptPo结果并删除 |
| FinanceDataService  .updateReceipt | 语法 | Public Boolean updateReceipt(ReceiptPo Receipt) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 查找到相应的ReceiptPo并用传入的Receipt替换 |
| FinanceDataService  .addPayment | 语法 | public boolean addPayment(PaymentPo payment) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| FinanceDataService  .findPayment | 语法 | Public List<PaymentPo> findPayment(Calendar date) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id进行查找返回相应的PaymentPo结果 |
| FinanceDataService  .deletePayment | 语法 | Public boolean deletePayment(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的PaymentPo结果并删除 |
| FinanceDataService  .updatePayment | 语法 | Public Boolean updatePayment(PaymentPo Payment) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 查找到相应的PaymentPo并用传入的Payment替换 |
| FinanceDataService  .addFreight | 语法 | public boolean addFreight(FreightPo Freight) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| FinanceDataService  .findFreight | 语法 | Public List<FreightPo> findFreight(Calendar date throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id进行查找返回相应的FreightPo结果 |
| FinanceDataService  .deleteFreight | 语法 | Public boolean deleteFreight(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的FreightPo结果并删除 |
| FinanceDataService  .updateFreight | 语法 | Public Boolean updateFreight(FreightPo Freight) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 查找到相应的FreightPo并用传入的Freight替换 |
| FinanceDataService  .addEarning | 语法 | public boolean addEarning(EarningPo Earning) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| FinanceDataService  .findEarning | 语法 | Public List<EarningPo> findEarning(Calendar date) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按date进行查找返回相应的EarningPo结果 |
| FinanceDataService  .deleteEarning | 语法 | Public boolean deleteEarning(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的EarningPo结果并删除 |
| FinanceDataService  .updateEarning | 语法 | Public Boolean updateEarning(EarningPo Earning) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 查找到相应的EarningPo并用传入的Earning替换 |
| FinanceDataService  .addAccount | 语法 | public boolean addAccount(AccountPo Account) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| FinanceDataService  .findAccount | 语法 | Public AccountPo findAccount(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id进行查找返回相应的AccountPo结果 |
| FinanceDataService  .deleteAccount | 语法 | Public boolean deleteAccount(long id) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id查找到相应的AccountPo结果并删除 |
| FinanceDataService  .updateAccount | 语法 | Public Boolean updateAccount(AccountPo Account) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 查找到相应的AccountPo并用传入的Account替换 |
| FinanceDataService  .addInitialInfo | 语法 | public boolean addInitialInfo(InitialInfoPo InitialInfo) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| FinanceDataService  .findInitialInfo | 语法 | Public InitialInfoPo findInitialInfo() throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 返回InitialInfoPo结果（唯一） |
| FinanceDataService  .deleteInitialInfo | 语法 | Public boolean deleteInitialInfo() throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 删除InitialInfoPo结果（唯一） |
| FinanceDataService  .addPriceStrategy | 语法 | public boolean addPriceStrategy(PriceStrategyPo PriceStrategy) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| FinanceDataService  .findPriceStrategy | 语法 | Public PriceStrategyPo findPriceStrategy() throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 返回相应的PriceStrategyPo结果（唯一） |
| FinanceDataService  .deletePriceStrategy | 语法 | Public boolean deletePriceStrategy() throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 删除相应的PriceStrategyPO（唯一） |
| FinanceDataService  .updatePriceStrategy | 语法 | Public Boolean updatePriceStrategy(PriceStrategyPo PriceStrategy) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 用传入的PriceStrategy替换原PriceStrategyPO |
| FinanceDataService  .addSalaryStrategy | 语法 | public boolean addSalaryStrategy(SalaryStrategyPo SalaryStrategy) throws RemoteException |
| 前置条件 | 同样的Po在记录中不存在 |
| 后置条件 | 在记录中增加一条Po |
| FinanceDataService  .findSalaryStrategy | 语法 | Public SalaryStrategyPo findSalaryStrategy() throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 返回相应的SalaryStrategyPo结果（唯一） |
| FinanceDataService  .deleteSalaryStrategy | 语法 | Public boolean deleteSalaryStrategy() throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 删除相应的SalaryStrategyPO（唯一） |
| FinanceDataService  .updateSalaryStrategy | 语法 | Public Boolean updateSalaryStrategy(SalaryStrategyPo SalaryStrategy) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 用传入的SalaryStrategy替换原StrategyPO |

**ListData的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ListDataService  .findList | 语法 | Public List<ListPo> findList(Calendar begin，Calendar end) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 按id进行查找返回相应的ListPo结果 |
| ListDataService  .updateList | 语法 | Public Boolean updateList(long id，State state) throws RemoteException |
| 前置条件 | 该Po在记录中要存在 |
| 后置条件 | 查找到相应的ListPo并用传入的List替换 |

## 6.信息视角

### 6.1数据持久化对象

系统的PO类就是对应的相关实体类，在此作简单的介绍。

* commodityPO类包含快递单号，以及在运输过程中产生的所有单据PO类的引用。
* sendPO类包含寄件人姓名、住址、单位、电话、手机，收件人姓名、住址、单位、电话、手机属性，货物信息（原件数、实际重量、体积、内件品名、尺寸）属性，包装的种类信息（纸箱、木箱、快递袋、其它）属性，订单条型号码（10位），预估时间，价格，运送方式属性。
* loadPO类包含装运类型、装运日期、本中转中心货（航）运编号、车次号，出发地、到达地、监装员、押运员、本次装箱所有托运单号属性，运费。
* arrivalPO类包含货物到达状态（损坏、完整、丢失）属性，托运订单条形码号，目的地、到达日期、中转单编号属性。
* dipatchPO类包含派送员工工号、到达日期、托运订单条形码号属性。
* vehiclePO类包含车辆代号、车牌号、服役时间属性
* driverPO类包含司机编号、姓名、出生日期、身份证号、手机 、性别、行驶证期限属性。
* warehousePO类包含库存分区信息、警戒线的值、在仓库中的所有快递的编号和位置形成的键值对集合、所有有关该仓库入库单和出库单的引用。
* checkinPO类包含需要入库的所有快递编号、入库日期、目的地、区号、排号、架号、位号属性。
* checkoutPO类包含快递编号、出库日期、目的地、装运形式、中转单编号或者汽运编号。
* receivePO类包含收件人姓名和收件时间属性。
* userPO类包含用户的用户名、密码和权限属性。
* personnelPO类包含员工的姓名、所属机构编号、人员职务、工资属性。
* departmentPO类包含机构类型、机构编号、地点属性。
* operationPO类包含操作类型、时间、用户名属性。
* receiptPO类包含收款日期、收款金额、收款快递员、对应的所有快递订单条形码号属性。
* paymentPO类包含付款单类型、付款时间、付款账户属性。
* earningPO类包含截至当前日期的总收入、总支出、总利润，以及当天日期属性。
* accountPO类包含银行账户名称和余额属性。
* InitialInforPO类包含最初的机构、人员、车辆、库存、银行账户信息属性。
* listPO类包含单据编号、单据类型、单据状态。
* cityPO类包含城市名字、对应中转中心编号、所有营业厅编号。
* priceStrategyPO类包含经济快递、标准快递、次晨特快价格比，标准快递的运费价格，单位为元/公斤\*1000km。
* salaryStrategyPO类包含人员类型和对应的工资策略形成的键值对，工资策略包括按月、计次、提成。

### 6.2数据持久化格式

用序列化持久化数据，统一使用.ser格式。