李凯 朴圣哲 狄逸枫 傅林华

物流快递系统

软件体系结构描述文档

物流快递系统软件体系结构描述文档

**1.引言**

**1.1** 编制目的

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

本报告面向开发人员、测试人员及最终用户而编写,是了解系统的导航。

**1.2** 参考资料

**2.产品概述**

参考物流快递系统用例文档和物流快递系统软件需求规格说明中对产品的概括描述。

**3.逻辑视角**

物流快递系统管理系统中,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、数据层能够很好的示意整个高层抽象。展示层中间包GUI 页面的实现、业务逻辑层中间包业务逻辑处理的实现、数据层负责数据的持久化和访问。

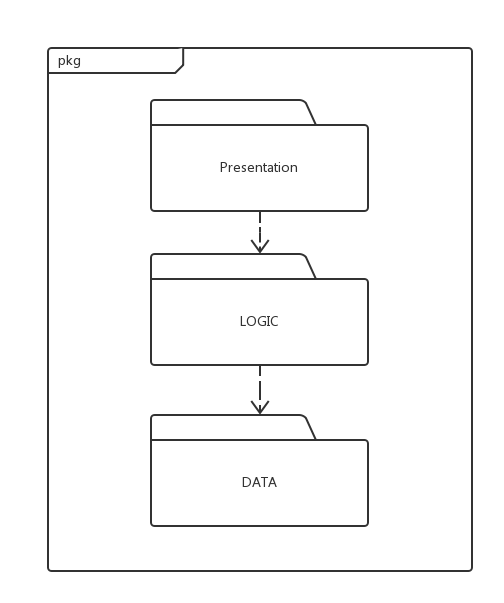


图1 参照体系结构风格的包图表达逻辑视角

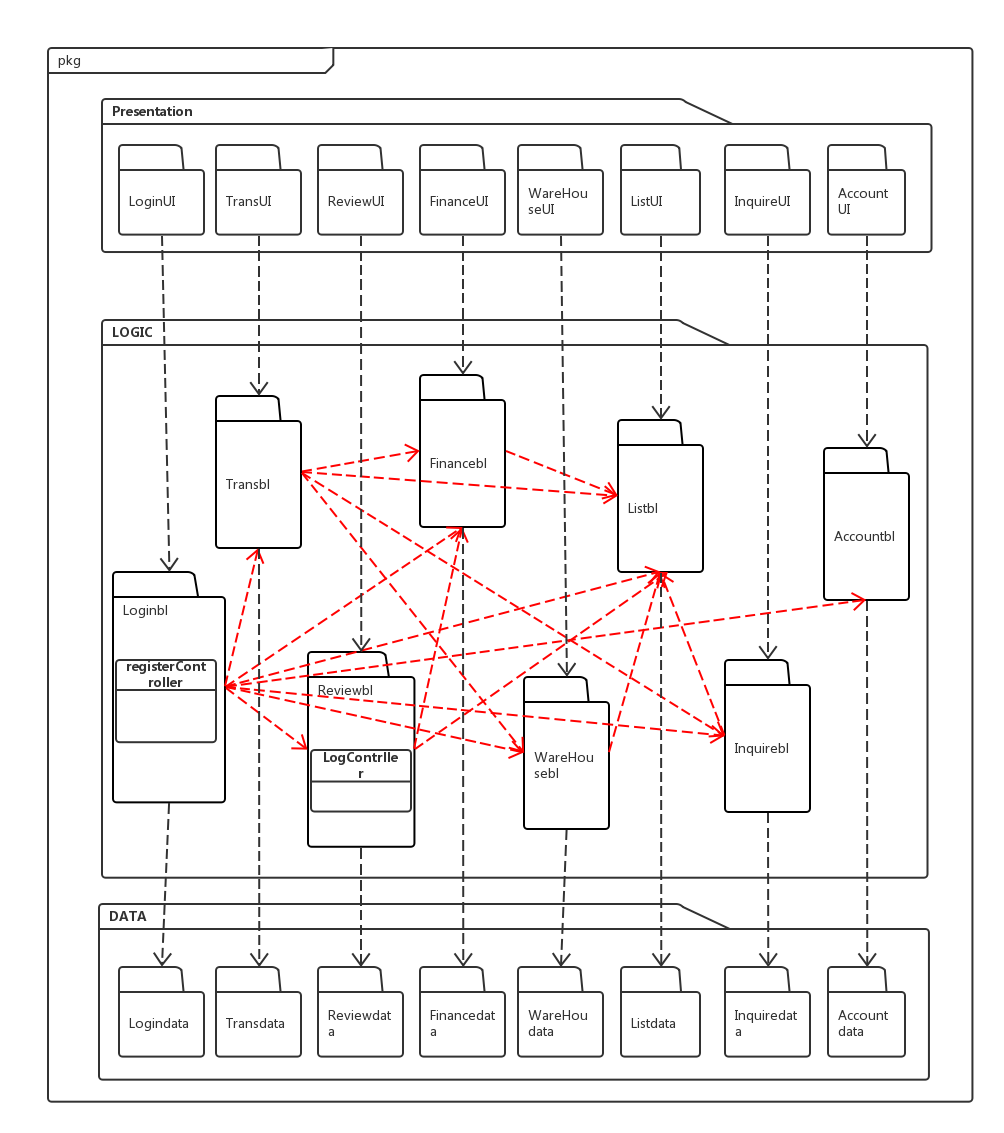


图2 软件体系结构逻辑设计方案

**4.组合视角**

4.1 开发包图

物流快递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 |
| UI | LoginBLSERVICE, TransBLSERVICE, ReviewBLSERVICE, FinanceBLSERVICE, WareHouseBLSERVICE, ListBLSERVICE, InquireBLSERVICE, AccountBLSERVICE |
| LoginUI | Loginblservice,界面类库包，VO |
| Loginblservice | PO |
| Loginbl | Loginblservice, datafactory, datafactoryservice, logindataservice,PO,VO |
| Logindataservice | Java RMI ,PO |
| Logindata | Logindataservice,PO,databaseutility, reviewdataservice, Java RMI |
| TransUI | Transblservice,界面类库包，VO |
| Transblservice | PO |
| Transbl | Listblservice,Inquireblservice,,PO,Transdataservice,datafactory,datafactoryservice,VO |
| Transdataservice | Java RMI,PO |
| Transdata | Transdataservice,PO,databaseutility, Java RMI |
| ReviewUI | Reviewblservice，界面类库包，VO |
| Reviewblservice | PO |
| Reviewbl | warehousedataservice,PO,Reviewdataservice,datafactory,datafactoryservice,VO |
| Reviewdataservice | Java RMI,PO |
| Reviewdata | Reviewdataservice,PO,databaseutility,datafactory,datafactoryservice, Java RMI |
| FinanceUI | Financeblservice，界面类库包，VO |
| Financeblservice | PO |
| Financebl | listdataservice,PO,Financedataservice,VO |
| Financedataservice | Java RMI,PO |
| Financedata | Financedataservice,PO,databaseutility, Java RMI |
| WareHouseUI | WareHouseblservice，界面类库包，VO |
| WareHouseblservice | PO |
| WareHousebl | WareHousedataservice,PO,listdataservice,datafactory,datafactoryservice,VO |
| WareHousedataservice | Java RMI,PO |
| WareHousedata | WareHousedataservice,PO,databaseutility, Java RMI |
| LisstUI | Listblservice，界面类库包，VO |
| Listblservice | PO |
| Listbl | PO,inquiredataservice,listdataservice,datafactoryservice,datafactory,VO |
| Listdataservice | Java RMI,PO |
| Listdata | Listdataservice,PO,databaseutility, Java RMI |
| InquireUI | Inquireblservice，界面类库包，VO |
| Inquireblservice | PO |
| Inquirebl | Inquiredataservice, PO, datafactoryservice, datafactory,VO, |
| Inquiredataservice | Java RMI,PO |
| Inquiredata | Inquiredataservice,PO,databaseutility, Java RMI |
| AccountUI | Accountblservice，界面类库包，VO |
| Accountblservice | PO |
| Accountbl | Logindataservice,Accountdataservice,PO,reviewdataservice,datafactoryservice, datafactory,VO, PO |
| Accoutdataservice | Java RMI,PO |
| Accountdata | Accountdataservice,PO,databaseutility, Java RMI |
| databaseutility | JDBC |

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。

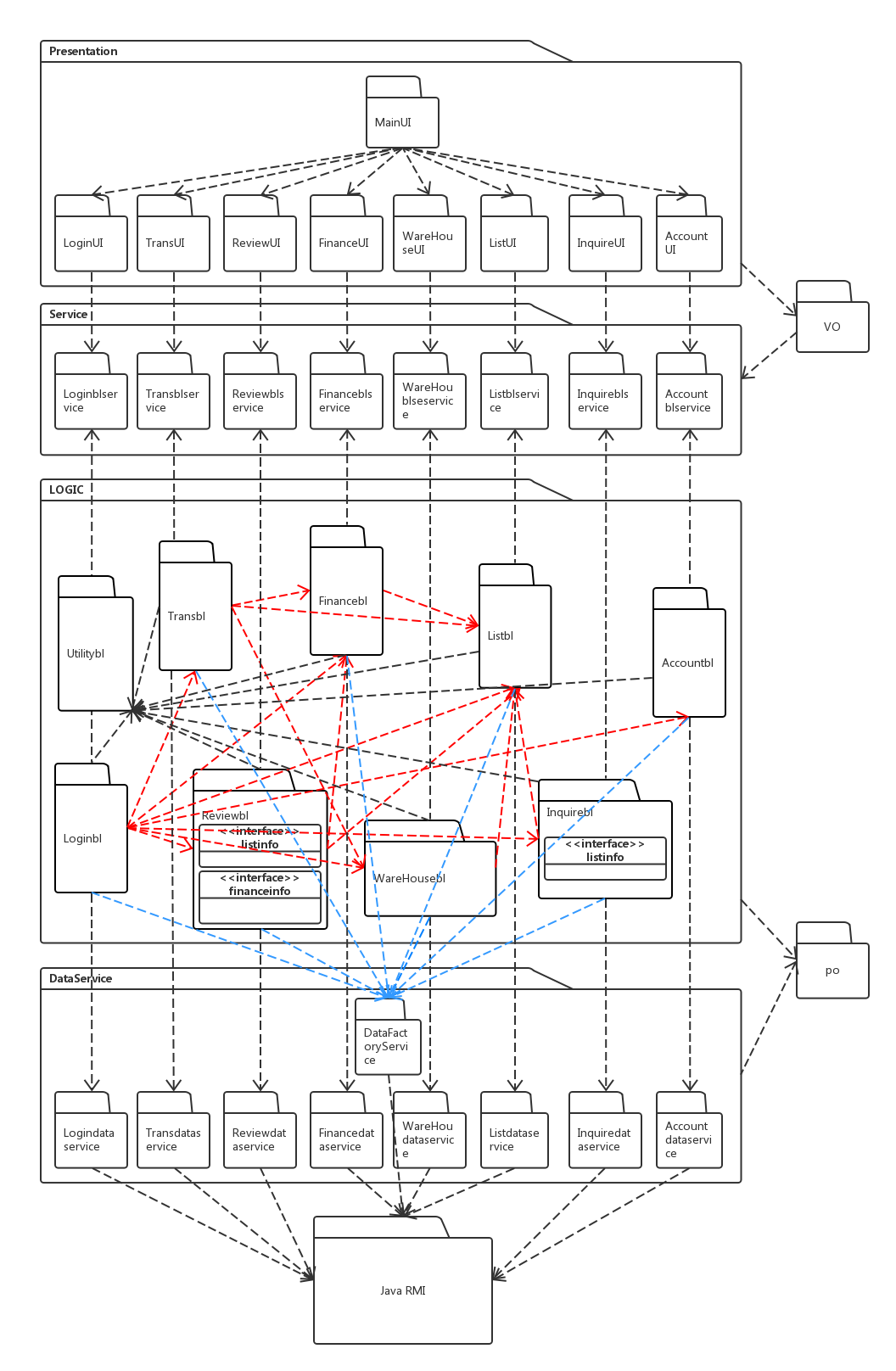


图3 物流快递系统客户端开发包图

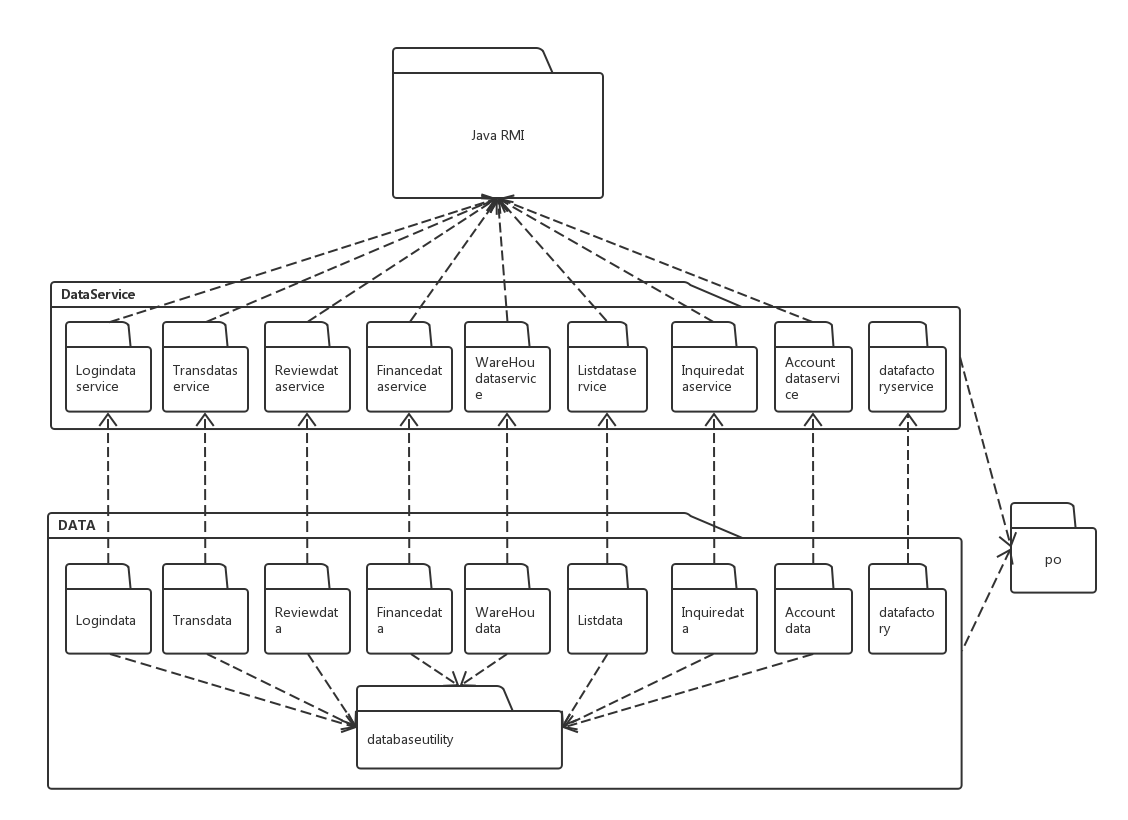


图4 物流快递系统服务器端开发包图

**4.2运行时进程**

在物流快递系统中,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。

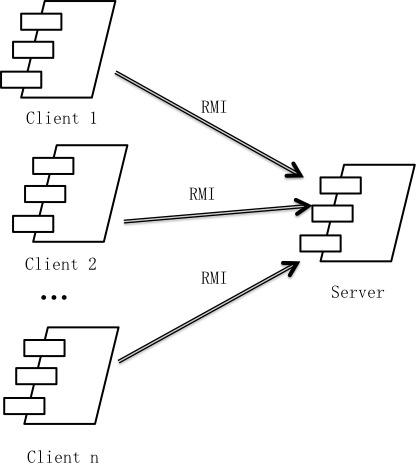


图5 进程图

**4.3物理部署**

物流快递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 6.0 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。

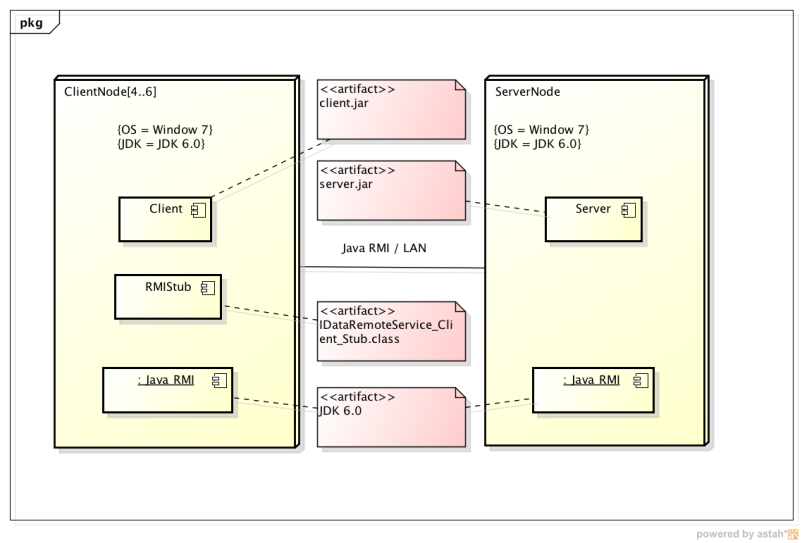


图6 部署图

**5．接口视角**

**5.1 模块的职责**

客户端模块和服务器端模块视图分别如图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。



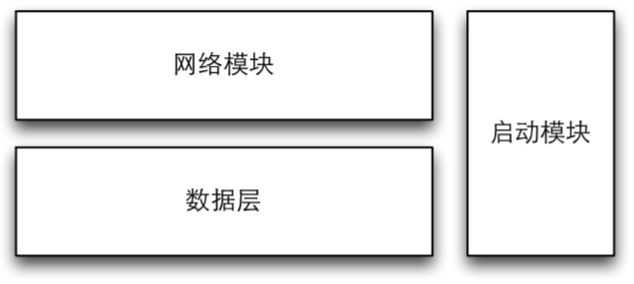


图7 客户端模块视图 图8服务器端模块视图

**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 | 基于窗口的物流快递客户端用户界面。 |
| 业务逻辑层 | 对于用户界面的输入响应和业务处理逻辑。 |
| 客户端网络模块 | 利用 Java RMI 机制查找 RMI 服务 |

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

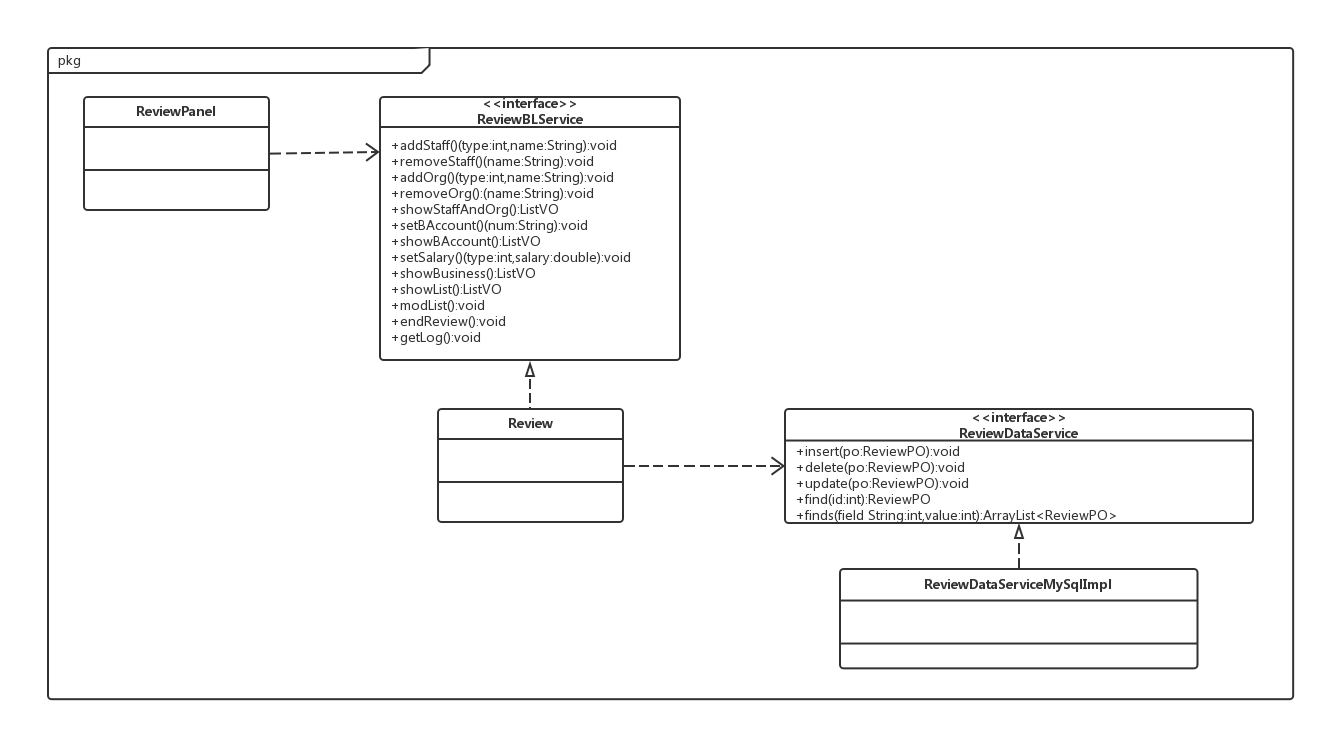
**表3 服务器端各层的职责**

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 |
| LoginBLService TransBLService ReviewBLService FinanceBLService WareHouseBLService ListBLService InquireBLService AccountBLService | 客户端展示层 | 客户端业务逻辑层 |
| LoginDataService TransDataService ReviewDataService FinanceDataService WareHouseDataService ListDataService InquireDataService AccountDataService DataFactory | 客户端业务逻辑层 | 服务器端数据层 |

借用销售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)。ReviewBLService提供了Sales界面所需要的所有业务逻辑功能。ReviewDataService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 销售用例层之间调用的接口

**5.2 用户界面层的分解**

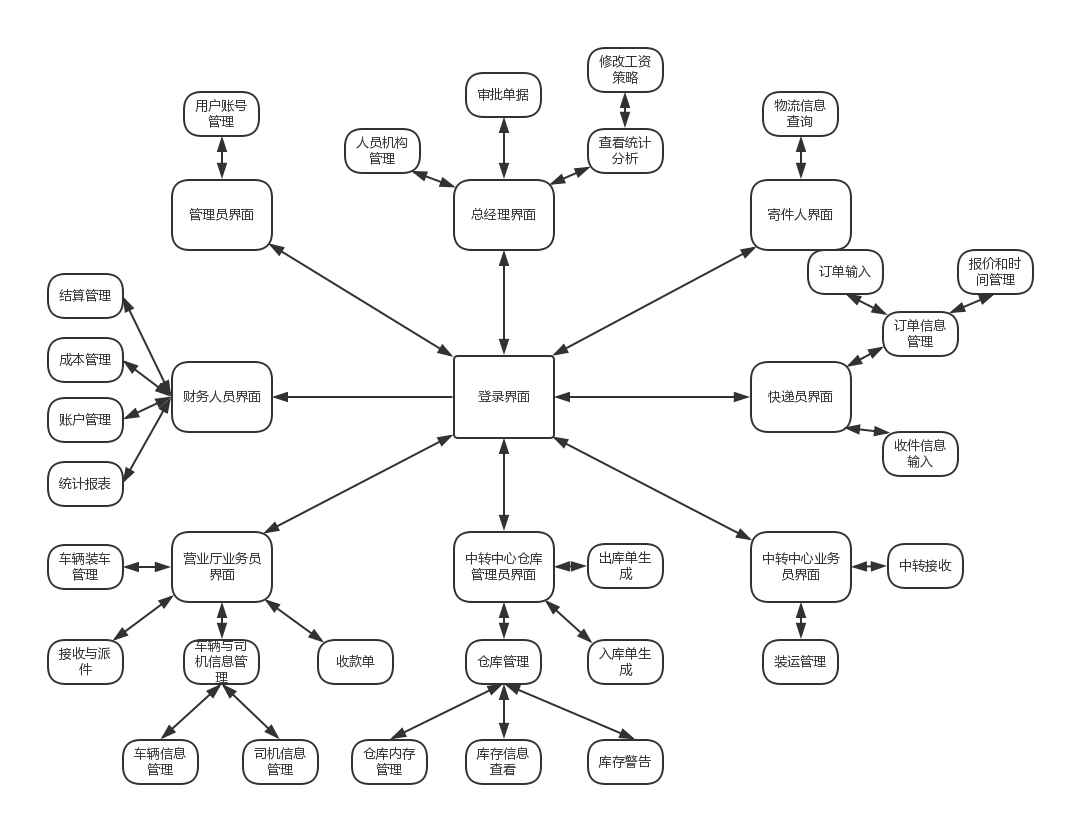
根据需求，系统存在37个用户界面：登录界面，总经理界面，人员机构管理界面，审批单据界面，修改工资界面，查看统计分析界面，管理员界面，用户账号管理界面，财务人员界面，结算管理界面，成本管理界面，账户管理界面，统计报表界面，营业厅业务员界面，车辆管理装车界面，接受与派件界面，车辆与司机信息管理界面，收款单界面，车辆信息管理界面，司机信息管理界面，中转中心仓库管理员界面，出库单生成界面，入库单生成界面，仓库管理界面，仓库内存管理界面，库存信息查看界面，库存管理界面，中转中心业务员界面，中转接收界面，装运管理界面，快递员界面，收件信息输入界面，订单信息管理界面，报价和时间管理界面，订单输入界面，寄件人界面，物流信息查询界面。界面跳转图如图10所示。

图10 用户界面跳转

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 |
| UI/Client | 界面，负责界面的显示和界面的跳转。 |

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**

|  |  |  |
| --- | --- | --- |
| LoginFrame | 语法 | init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示LoginFrame |

用户界面层需要的服务接口如表7所示。

**表7 用户界面层需要的服务接口**

|  |  |
| --- | --- |
| 服务名 | 服务 |
| LogicService.LoginBLService | 登陆界面的业务逻辑接口 |
| LogicService.\*BLService | 每个界面都有一个相应的业务逻辑接口 |

5.2.3 用户界面模块设计原理

用户界面利用Java的Swing和AWT库来实现。

**5.3 业务逻辑层的分解：**

业务逻辑层包括多个针对界面的业务逻辑处理对象。例如

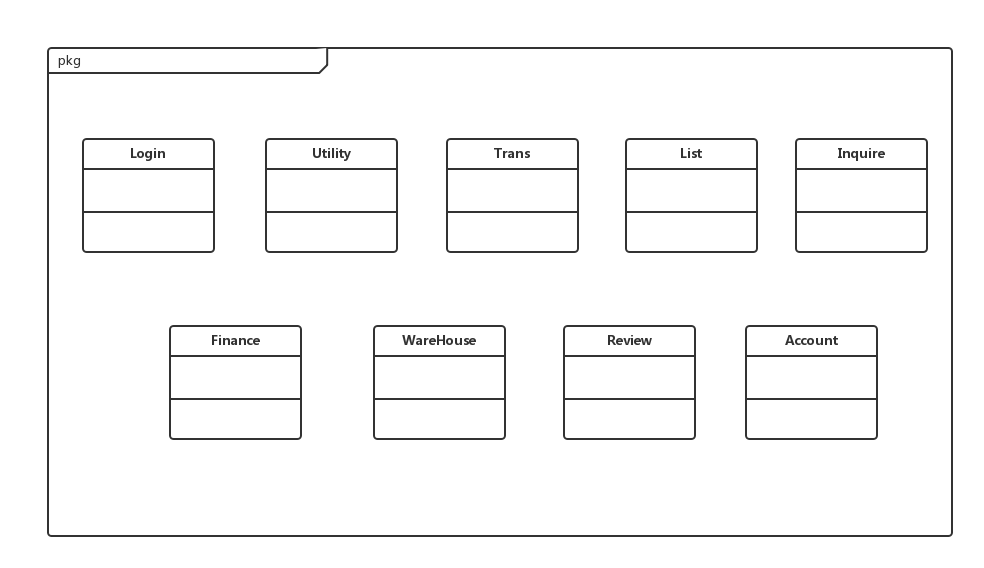


图12 业务逻辑层的设计.

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** |
| Loginbl | 负责实现登录对应界面需要的服务 |
| Transbl | 负责接收与派件界面所需的服务 |
| Listbl | 负责与单据相关任务所需的服务 |
| Inquirebl | 负责物流查询界面所需的服务 |
| Financebl | 负责财务人员界面所需的服务 |
| WareHousebl | 负责中转中心界面所需的服务 |
| Reviewbl | 负责总经理相关功能界面所需的服务 |
| Accountbl | 负责账户管理界面所需的服务 |

5.3.1 业务逻辑层模块的接口规范

**表9 Login模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Login.login | 语法 | Public AccountPO login(long account,String password) |
| 前置条件 | 账号密码符合输入规则 |
| 后置条件 | 根据对应权限，显示相对应的界面 |
| Login.addAcount | 语法 | Public ResultMessage addAcount(AccountPO po) |
| 前置条件 | Po正常 |
| 后置条件 | 创建po对应的帐号 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| public AccountPO find(String username) | 查找单一持久化对象 | |
| public ArrayList<AccountPO> findAll() | 找到所有账号 | |
| public void insert(AccountPO po) | 增加新账号 | |
| public void update(AccountPO po) | 更新账号信息 | |
| public void init() | 初始化账号信息 | |
| public void delete(long id) | 删除id对应帐号 | |
| public AccountPO findlast() | 查找最后添加的帐号 | |
| public void finish() | 结束登陆 | |
| public boolean login(String username, String password) | 进行登陆 | |

**表11 Trans模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Trans.addLoading | 语法 | public LoadingVO addLoading(TimePO loadDate, long transNum, City departPlace, City destination, long waybillNum,  String loadMonitor, String loadPerformer, double freight) |
| 前置条件 | 输入信息完整 |
| 后置条件 | 货物装车 |
| Trans.submit | 语法 | public boolean submit() |
| 前置条件 |  |
| 后置条件 | 提交信息 |
| Trans.addLoadingLists | 语法 | addLoadingLists(long id, ListType type, TimePO loadDate, long transNum, City departPlace,  City destination, long[] waybillNumList, String loadMonitor, String loadPerformer, double freight) |
| 前置条件 | 输入信息完整 |
| 后置条件 | 增加装车单 |
| Trans. getAccountPO | 语法 | public AccountPO getAccountPO() |
| 前置条件 |  |
| 后置条件 | 获取accountpo |
| Trans. addTransCenterArrivalList | 语法 | public TransCenterArrivalListVO addTransCenterArrivalList(String transcenterID, long id, TimePO arriveTime, City startCity,  GoodState state) |
| 前置条件 | 输入信息完整 |
| 后置条件 | 增加中转中心到达单 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| Transdataservice.insert(carPO po) | 在数据库中插入carPO对象 | |
| Transdataservice.insert(driverPO po) | 在数据库中插入driverPO对象 | |
| Transdataservice.insert(loadingPO po) | 在数据库中插入loadingPO对象 | |
| Transdataservice.delete(carPO po) | 在数据库中删除carPO对象 | |
| Transdataservice.delete(driverPO po) | 在数据库中删除driverPO对象 | |
| Transdataservice.find(long id) | 根据ID进行查找单一持久对象 | |
| ArrayList<CarPO> findAll() | 找出所有的车辆 | |
| ArrayList<DriverPO> findAll() | 找出所有的司机 | |

**表12 List模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| List.myGetListId | 语法 | public long myGetListId(ArrivalListDataService od, TimePO time) |
| 前置条件 | 输入合法且存在 |
| 后置条件 | 获得单据id |
| List.addList | 语法 | public ArrivaListVO addList(long transid, TimePO time, Long id, City StartCity, GoodState state) |
| 前置条件 | 输入合法且存在 |
| 后置条件 | 增加单据 |
| List.addware | 语法 | public DeliveryListVO addware(TimePO time, long code, String name) |
| 前置条件 | 输入合法且存在 |
| 后置条件 | 增加货物 |
| List.findAllExist | 语法 | public ArrayList<MoneyInListVO> findAllExist(AccountPO po) |
| 前置条件 | 帐号存在 |
| 后置条件 | 找出所有存在的单据 |
| List.addOrderList | 语法 | public OrderListVO addOrderList(ListType listtype, String senderName, String senaderAddress,  String senderOrganization, String senderTphone, String senderCphone, String receiverName,  String receiverAddress, String receiverOrganization, String receiverTphone, String receiverCphone,  WarePO ware, long id, String account) |
| 前置条件 | 输入合法 |
| 后置条件 | 增加订单 |
| List. addReceiveCourierList | 语法 | public ReceiveCourierListVO addReceiveCourierList(TimePO time, long id, String name, String cellphoneNum,ListState state) |
| 前置条件 | 输入合法 |
| 后置条件 | 增加接收单 |
| List.submit | 语法 | public boolean submit() |
| 前置条件 |  |
| 后置条件 | 提交 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| Listdataservice.insert(ListPO po) | 在数据库中新增以输入id标识的list信息 | |
| Listdataservice.update(ListPO po) | 在数据库中修改以输入id标识的list信息 | |
| Listdataservice.delete(ListPO po) | 在数据库中删除以输入id标识的list信息 | |
| Listdataservice.find(long id) | 查找单一持久化对象 | |
| Listdataservice.finish() | 结束持久化 | |
| Listdataservice.init() | 初始化 | |

**表13 Inquire模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Inquire.inquire | 语法 | public ArrayList<TransVO> inquire(String id) |
| 前置条件 | 输入的快件id合法且存在 |
| 后置条件 | 显示物流信息 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| Inquiredataservice.find(long id) | 查找单一持久化对象 | |
| Inquiredataservice.insert(TransPO transPO) | 插入历史轨迹信息对象 | |
| Inquiredataservice.init() | 初始化历史轨迹信息 | |
| Inquiredataservice.finish() | 结束持久化 | |

**表14 Finance模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Finance.addStaff | 语法 | public BaccountVO addStaff(String id, String city) |
| 前置条件 | 输入的信息完整 |
| 后置条件 | 增加职员 |
| Finance.deleteStaff | 语法 | public boolean deleteStaff(String id) |
| 前置条件 | Id存在 |
| 后置条件 | 删除职员 |
| Finance.update | 语法 | public Boolean update (ArrayList<BaccountVO> voList) |
| 前置条件 | voList存在且正确 |
| 后置条件 | 更新账户信息 |
| Finance.findAll | 语法 | public ArrayList<MoneyInListVO> findAll(TimePO time, String city) |
| 前置条件 | 输入信息正确 |
| 后置条件 | 返回所有的收款单 |
| Finance.addMOList | 语法 | public MoneyOutListVO addMOList(long id, TimePO time, double money, String name, BaccountPO account, Entry entry,  String note, ListState lst) |
| 前置条件 | 输入信息正确 |
| 后置条件 | 增加出款单 |
| Finance.myGetListId | 语法 | public long myGetListId(TimePO time) |
| 前置条件 | 输入时间正确 |
| 后置条件 | 返回出款单id |
| Finance.findList | 语法 | public ArrayList<MoneyOutListPO> findList() |
| 前置条件 |  |
| 后置条件 | 返回所有出款单 |
| Finance.submit | 语法 | public void submit() |
| 前置条件 |  |
| 后置条件 | 提交财务信息 |
| Finance.setBAccount | 语法 | public ResultMessage setBAccount(String num) |
| 前置条件 | 启动审查中的查看账户 |
| 后置条件 | 在查看账户中，修改账户信息 |
| Finance.addBAccount | 语法 | public ResultMessage addBAccount(String num) |
| 前置条件 | 启动审查中的查看账户 |
| 后置条件 | 在查看账户中，增加账户信息 |
| Finance.deleteBAccount | 语法 | public ResultMessage addBAccount(String num) |
| 前置条件 | 启动审查中的查看账户 |
| 后置条件 | 在查看账户中，增加账户信息 |
| **需要的服务** | | |
| 服务名 | 服务 | |
| Financedataservice.find(long id) | 查找单一持久化对象 | |
| Financedataservice.insert() | 在数据库中插入对象 | |
| Financedataservice.updata () | 在数据库中更新对象 | |
| Financedataservice.delete() | 在数据库中删除对象 | |

**表15 WareHouse模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| WareHouse.addLoading | 语法 | | public LoadingVO addLoading(TimePO loadDate, long transNum, City departPlace, City destination, long waybillNum,  String loadMonitor, String loadPerformer, double freight) |
| 前置条件 | | 输入的信息正确完整 |
| 后置条件 | | 增加装车信息 |
| WareHouse.addLoadingLists | 语法 | | Public LoadingListVO addLoadingLists(long id, ListType type, TimePO loadDate, long transNum, City departPlace,City destination, long[] waybillNumList, String loadMonitor, String loadPerformer, double freight) |
| 前置条件 | | 输入的信息正确完整 |
| 后置条件 | | 增加装车单 |
| WareHouse.myGetListId | 语法 | | Public long myGetListId(LoadingListDataService od, TimePO time) |
| 前置条件 | | 输入的信息正确完整 |
| 后置条件 | | 返回单据id |
| WareHouse.addWareIn | 语法 | | public WareInInputVO addWareIn(long id, TimePO time, City destination, long transid) |
| 前置条件 | | 输入的信息正确完整 |
| 后置条件 | | 增加入库信息 |
| WareHouse.getPlace | 语法 | | Public ArrayList<GarageBodyPO> getPlace(long transid) |
| 前置条件 | | Id存在 |
| 后置条件 | | 返回存放的位置 |
| WareHouse.addtotxt | 语法 | | public void addtotxt(WareInListPO po) |
| 前置条件 | | 输入的信息正确完整 |
| 后置条件 | | 提交信息，增加到数据 |
| WareHouse.submit | 语法 | | public void submit() |
| 前置条件 | |  |
| 后置条件 | | 提交仓库信息 |
| WareHouse. addWareOut | 语法 | | public void addWareOut(long id, TimePO time, City destination, Vehicle vehicle, long transid, GaragePlacePO place) |
| 前置条件 | | 输入的信息正确完整 |
| 后置条件 | | 增加出库信息 |
| WareHouse.findWareIn | 语法 | | public boolean findWareIn(long id) |
| 前置条件 | |  |
| 后置条件 | | 返回是否入库 |
| WareHouse.getWareIn | 语法 | | public ArrayList<GarageBodyPO> getWareIn(TimePO start, TimePO end) |
| 前置条件 | | 输入时间正确 |
| 后置条件 | | 返回这一时间段内的库存信息 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| WareHouseDataService.insert(String address, garageitem item) | | 插入单一持久化对象 | |
| WareHouseDataService.delete(String address, long id) | | 删除单一持久化队形 | |
| WareHouseDataService.rate(String address) | | 获取比例 | |
| WareHouseDataService. destroy(String address) | | 摧毁address对应仓库 | |
| WareHouseDataService. show (String address) | | 展示单一持久化对象 | |
| WareHouseDataService. insertByPlace (String address, garageitem item, GaragePlacePO place) | | 根据位置进行插入 | |
| WareHouseDataService. getGarage (String address) | | 获取仓库 | |
| WareHouseDataService. init(String address) | | 初始化 | |

**表16 Review模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Review.addStaff | 语法 | public ResultMessage addStaff(int type,String name) |
| 前置条件 | 启动审查中的人员和机构管理 |
| 后置条件 | 在人员和机构管理中，增加人员信息 |
| Review.removeStaff | 语法 | public ResultMessage removeStaff(int type,String name) |
| 前置条件 | 启动审查中的人员和机构管理 |
| 后置条件 | 在人员和机构管理中，移除人员信息 |
| Review.addOrg | 语法 | public ResultMessage addOrg(int type,String name) |
| 前置条件 | 启动审查中的人员和机构管理 |
| 后置条件 | 在人员和机构管理中，增加机构信息 |
| Review.removeOrg | 语法 | public ResultMessage removeOrg(int type,String name) |
| 前置条件 | 启动审查中的人员和机构管理 |
| 后置条件 | 在人员和机构管理中，删除机构信息 |
| Review.setSalary | 语法 | public ResultMessage setSalary(int type,double salary) |
| 前置条件 | 启动审查中的制定薪水策略 |
| 后置条件 | 在制定薪水策略中，制定薪水策略 |
| Review.findAll | 语法 | public ArrayList<LogVO> findAll () |
| 前置条件 |  |
| 后置条件 | 返回所有日志信息 |
| Review. findAllConstant | 语法 | public ArrayList<ConstantVO> findAllConstant() |
| 前置条件 |  |
| 后置条件 | 返回所有常量信息 |
| Review. addConstant | 语法 | public boolean addConstant  (ArrayList<ConstantVO> list) |
| 前置条件 | 输入信息正确 |
| 后置条件 | 同步常量信息 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| ReviewDataService.insert() | 插入单一持久化对象 | |
| ReviewDataService.delete() | 删除单一持久化队形 | |
| ReviewDataService.update() | 更新单一持久化对象 | |
| ReviewDataService.init() | 初始化 | |
| ReviewDataService.find(long id) | 根据ID进行查找 | |

**表17 Account模块的接口规范**

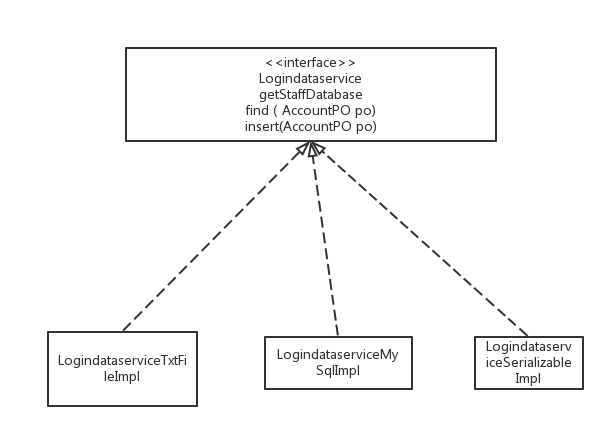
|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Account.findAll | 语法 | public ArrayList<AccountVO> findAll() |
| 前置条件 |  |
| 后置条件 | 返回所有的帐号 |
| Account.permissionUpate | 语法 | Public ResultMessage permissionUpate (ArrayList<AccountVO> voList) |
| 前置条件 | volist存在 |
| 后置条件 | 更新权限 |
| Account.addAccount | 语法 | public AccountVO addAccount(Permission permission, String username, String password, String staffid) |
| 前置条件 | 输入信息正确 |
| 后置条件 | 增加帐号 |
| Account.deleteAccount | 语法 | public boolean deleteAccount(long id) |
| 前置条件 | Id存在 |
| 后置条件 | 删除帐号 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| Accountdataservice.delete (AccountPO po) | 在数据库中删除AccountPO对象 | |
| Accountdataservice.insert(AccountPO po) | 在数据库中插入AccountPO对象 | |
| Accountdataservice.updata (AccountPO po) | 在数据库中更新AccountPO对象 | |
| Accountdataservice.find(long id) | 查找单一持久化对象 | |

**5.4 数据层的分解**

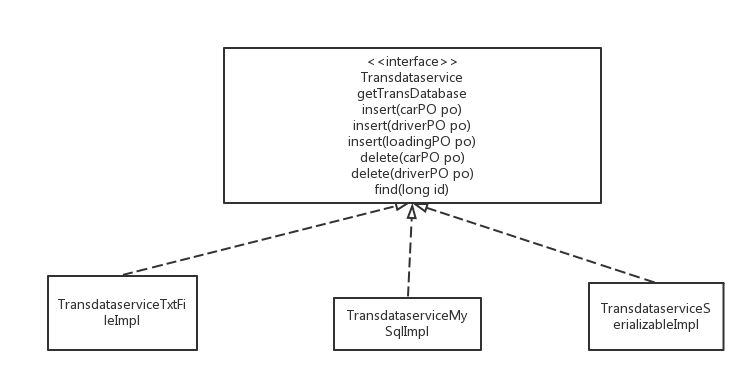
数据层主要给业务逻辑层提供数据访问服务。包括对于持久化数据的、增、删、改、查。

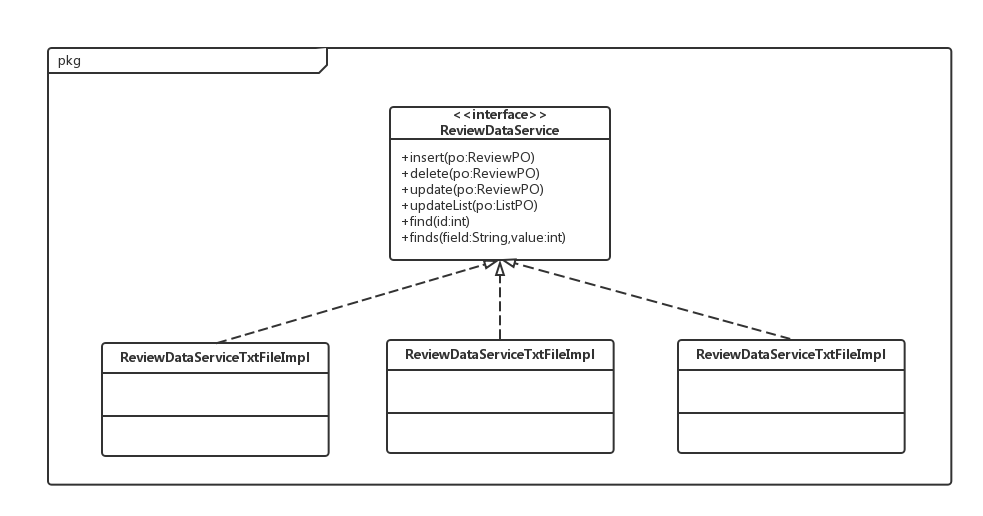
这些服务由 DataService 接口提供。由于持久化数据的保存可能存在多种形式:Txt 文件、序 列化文件、数据库等。而且还可能出现混搭的可能。具体如图13所示。

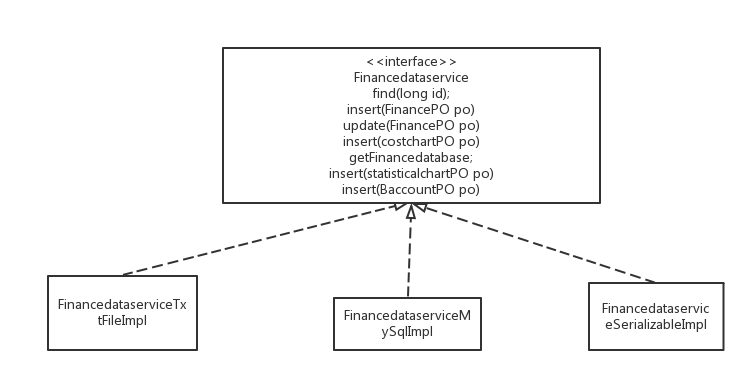
Login的数据层

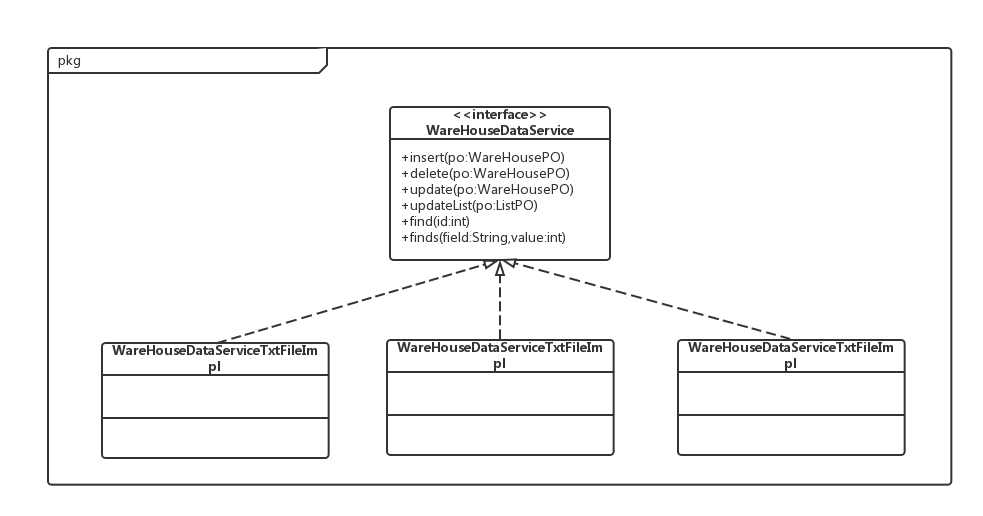


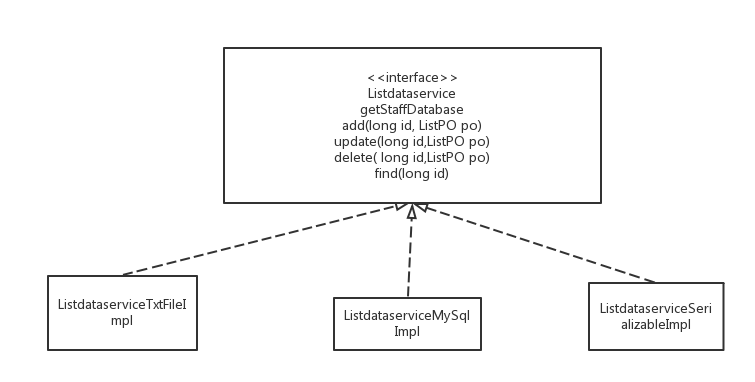
Trans的数据层

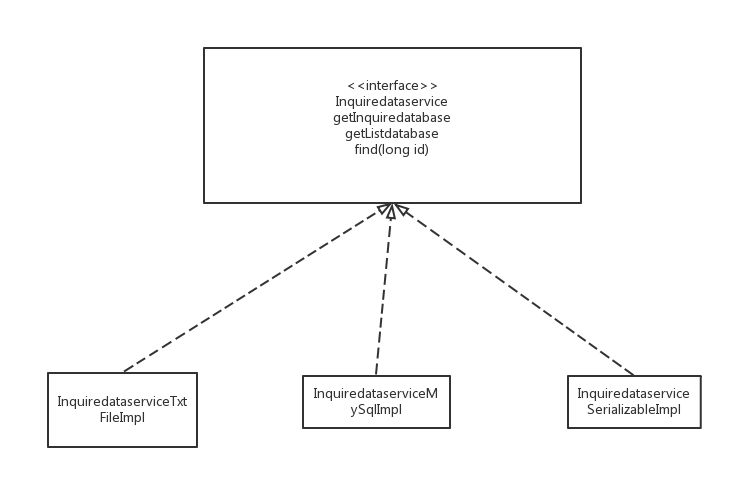
 Review的数据层

 Finance的数据层

 WareHouse的数据层

List的数据层

Inquire的数据层

Account的数据层

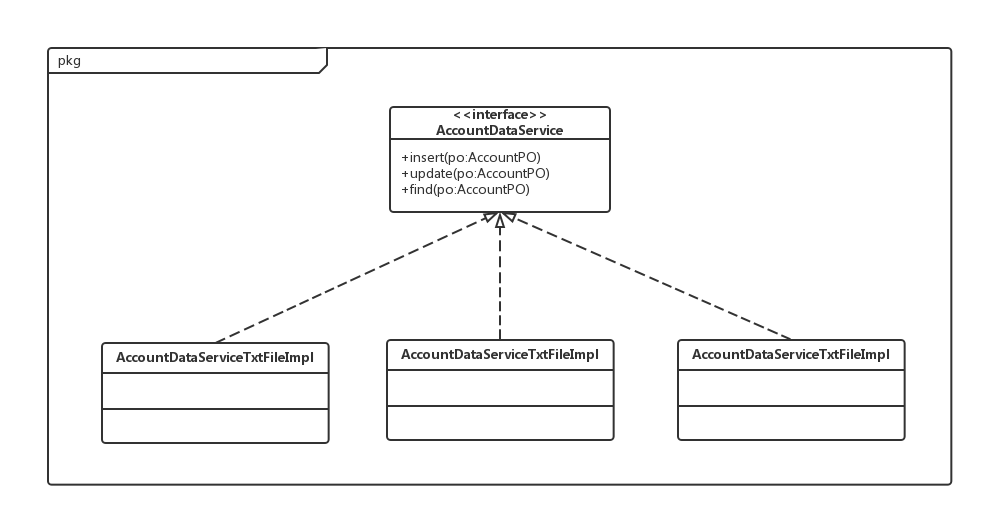


图13 数据层模块的描述

5.4.1 数据层模块的职责

数据层模块的职责如表18所示。

**表18 数据层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| Logindataservice | 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。 |
| LogindataserviceTxtFileImpl | 基于Txt文件的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。 |
| Transdataservice | 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。 |
| TransdataserviceTxtFileImpl | 基于Txt文件的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。 |
| Reviewdataservice | 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。 |
| ReviewdataserviceTxtFileImpl | 基于Txt文件的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。 |
| FinancedataserviceTxtFileImpl | 基于Txt文件的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。 |
| WareHousedataservice | 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。 |
| WareHousedataserviceTxtFileImpl | 基于Txt文件的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。 |
| WareHousedataserviceSerializableFileImpl | 基于序列化文件的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。 |
| Listdataservice | 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。 |
| ListdataserviceTxtFileImpl | 基于Txt文件的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。 |
| Inquiredataservice | 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。 |
| InquiredataserviceTxtFileImpl | 基于Txt文件的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。 |
| Accountdataservice | 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。 |
| AccountdataserviceTxtFileImpl | 基于Txt文件的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 数据层模块的规范

数据层模块的规范如表19所示。

**表19 数据层模块的规范**

**LoginDataService的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| public AccountPO find(String username) | 查找单一持久化对象 |
| public ArrayList<AccountPO> findAll() | 找到所有账号 |
| public void insert(AccountPO po) | 增加新账号 |
| public void update(AccountPO po) | 更新账号信息 |
| public void init() | 初始化账号信息 |
| public void delete(long id) | 删除id对应帐号 |
| public AccountPO findlast() | 查找最后添加的帐号 |
| public void finish() | 结束登陆 |
| public boolean login(String username, String password) | 进行登陆 |

**TransDataService的接口规范**

|  |  |
| --- | --- |
| 提供的服务（供接口） | |
| Transdataservice.insert(carPO po) | 在数据库中插入carPO对象 |
| Transdataservice.insert(driverPO po) | 在数据库中插入driverPO对象 |
| Transdataservice.insert(loadingPO po) | 在数据库中插入loadingPO对象 |
| Transdataservice.delete(carPO po) | 在数据库中删除carPO对象 |
| Transdataservice.delete(driverPO po) | 在数据库中删除driverPO对象 |
| Transdataservice.find(long id) | 根据ID进行查找单一持久对象 |
| ArrayList<CarPO> findAll() | 找出所有的车辆 |
| ArrayList<DriverPO> findAll() | 找出所有的司机 |

**FinanceDataService的接口规范**

|  |  |
| --- | --- |
| 提供的服务（供接口） | |
| Financedataservice.find(long id) | 查找单一持久化对象 |
| Financedataservice.insert() | 在数据库中插入对象 |
| Financedataservice.updata () | 在数据库中更新对象 |
| Financedataservice.delete() | 在数据库中删除对象 |

**ReviewDataService的接口规范**

|  |  |
| --- | --- |
| 提供的服务（供接口） | |
| ReviewDataService.insert() | 插入单一持久化对象 |
| ReviewDataService.delete() | 删除单一持久化队形 |
| ReviewDataService.update() | 更新单一持久化对象 |
| ReviewDataService.init() | 初始化 |
| ReviewDataService.find(long id) | 根据ID进行查找 |

**WareHouseDataService的接口规范**

|  |  |
| --- | --- |
| 提供的服务（供接口） | |
| WareHouseDataService.insert(String address, garageitem item) | 插入单一持久化对象 |
| WareHouseDataService.delete(String address, long id) | 删除单一持久化队形 |
| WareHouseDataService.rate(String address) | 获取比例 |
| WareHouseDataService. destroy(String address) | 摧毁address对应仓库 |
| WareHouseDataService. show (String address) | 展示单一持久化对象 |
| WareHouseDataService. insertByPlace (String address, garageitem item, GaragePlacePO place) | 根据位置进行插入 |
| WareHouseDataService. getGarage (String address) | 获取仓库 |
| WareHouseDataService. init(String address) | 初始化 |

**ListDataService的接口规范**

|  |  |
| --- | --- |
| 提供的服务（供接口） | |
| Listdataservice.insert(ListPO po) | 在数据库中新增以输入id标识的list信息 |
| Listdataservice.update(ListPO po) | 在数据库中修改以输入id标识的list信息 |
| Listdataservice.delete(ListPO po) | 在数据库中删除以输入id标识的list信息 |
| Listdataservice.find(long id) | 查找单一持久化对象 |
| Listdataservice.finish() | 结束持久化 |
| Listdataservice.init() | 初始化 |

**InquireDataService的接口规范**

|  |  |
| --- | --- |
| 提供的服务（供接口） | |
| Inquiredataservice.find(long id) | 查找单一持久化对象 |
| Inquiredataservice.insert(TransPO transPO) | 插入历史轨迹信息对象 |
| Inquiredataservice.init() | 初始化历史轨迹信息 |
| Inquiredataservice.finish() | 结束持久化 |

**AccountDataService的接口规范**

|  |  |
| --- | --- |
| 提供的服务（供接口） | |
| Accountdataservice.delete (AccountPO po) | 在数据库中删除AccountPO对象 |
| Accountdataservice.insert(AccountPO po) | 在数据库中插入AccountPO对象 |
| Accountdataservice.updata (AccountPO po) | 在数据库中更新AccountPO对象 |
| Accountdataservice.find(long id) | 查找单一持久化对象 |

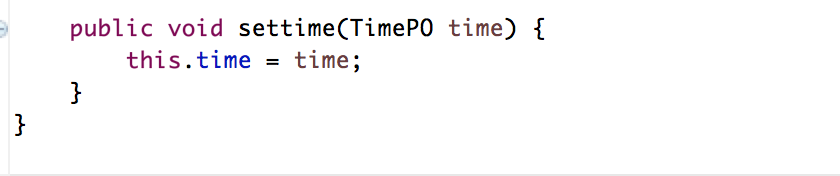
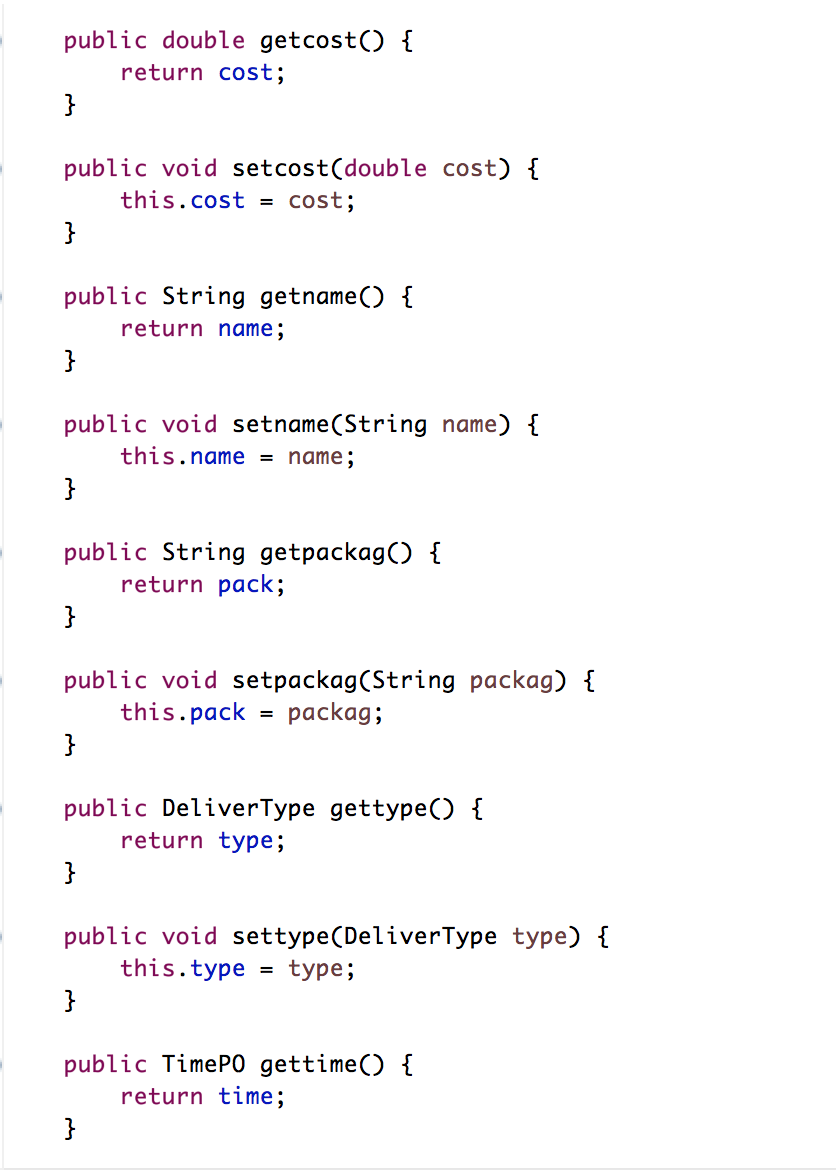
**6.信息视角**

**6.1 数据持久化对象**

系统的PO类就是对应的相关实体类，在此做简单介绍。

* AccountPO类包含用户的账号。密码和权限
* BaccountPO类包含账户信息
* warePO类包含原件数、实际重量、体积、内件品名、尺寸 包装费 订单条形码号 快递种类
* carPO类包含车辆代号、发动机号、车辆号、底盘号、购买时间、服役时间、车辆图片
* ConstantPO类包含常量名和常量值
* driverPO类包含司机编号、姓名、出生日期、身份证号、手机、车辆单位、性别、行驶证期限
* GarageBodyPO包含仓库信息
* Garageitem包涵仓库内货物信息
* GaragePlacePO包含仓库区排架位信息
* LogPO包含日志信息
* ReceiptPO包含收款单信息
* SetupPO包含建账信息
* loadingPO类包含记录装车日期、本营业厅汽运编号、出发地、到达地、监装员、押运员、本次装箱所有托运单号。运费
* ListPO包含单据种类和相应种类单据的全部信息
* FinancePO类收款日期、收款单位、收款人、收款方、收款金额、收款地点
* costchartPO类包含成本收益表的信息
* statisticalchartPO类包含经营情况表的信息
* BaccountPO po类包含账户的信息
* WareHousePO类包含库存信息
* reviewPO包含staffPO,institlePO为人员机构的 具体信息
* TimePO类包含某一个点的时间信息
* TransPO类包含物流历史信息

持久化对象WarePO定义如下。



**6.2 Txt持久化格式**

Txt数据保持格式以ware.txt为例。每行分别对应实际重量、体积、包装类型、内件品名、快递类型、运费、预计时间。中间用”:”隔开。如下所示：

23.0:23.0:纸箱:耳机:经济快递:10.0:1y1m1d1h1m1s

2.0:5.0:纸箱:手机:经济快递:8.0:1y1m1d1h1m1s

* 1. **数据库表**

数据库中包含Account表、Baccount表、car表、driver表、loading表、List表、Finance表、costchart表、statisticalchart表、Baccount表、WareHouse表、review表、staff表、institle表、Time表、Trans表。