|  |
| --- |
|  |
| 物流管理系统 |
| 软件体系结构描述文档 |
|  |
| **版本：v1.0** |
| **2015/10/19** |

|  |
| --- |
|  |

**小组成员： 汤大业**

**谭期友**

**杨三洋**

**王家玮**

**目录及索引**

[1.引言 （）](#引言)

[1.1编制目的](#编制目的)

[1.2参考文献](#参考资料)

[2.产品概述 （）](#产品概述)

[3.逻辑视角 （）](#逻辑视角)

[4.组合视角 （）](#组合视角)

[4.1开发包图](#开发包图)

[4.2运行时进程](#运行时进程)

[4.3物理部署](#物理部署)

[5.接口视角 （）](#接口视角)

[5.1模块的职责](#模块的职责)

[5.2 用户界面分层解](#用户界面的分解)

[5.3业务逻辑层的分解](#业务逻辑层的分解)

[5.4数据层的分解](#数据层的分解)

[6.信息视角 （）](#信息视角)

[6.1数据持久化对象](#数据持久化对象)

[6.2 Txt持久化格式](#Txt持久化格式)

1. **引言**

**1.1编制目的**

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

本报告面向开发人员、测试人员及最终的用户而编写，是了解系统的导航

**1.2参考文献**

1. IEE标准

2.项目实践V4 南京大学软件学院 刘钦 2015

3.软件工程与计算Ⅱ 南京大学软件学院 刘钦 丁二玉 2015

4.物流管理系统用例文档V1.0

5.物流管理系统软件需求规格说明

**2.产品概述**

参考物流信息管理系统用例文档和物流信息管理系统软件需求规格说明中对产品的概括描述

**3.逻辑视角**

物流信息管理系统中，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）能够很好地示意整个高层抽象。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下图1和图2所示。

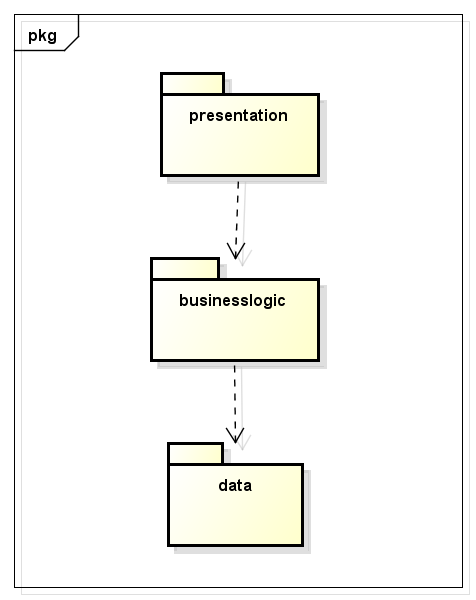


图1 参照体系结构风格的包图表达逻辑视角

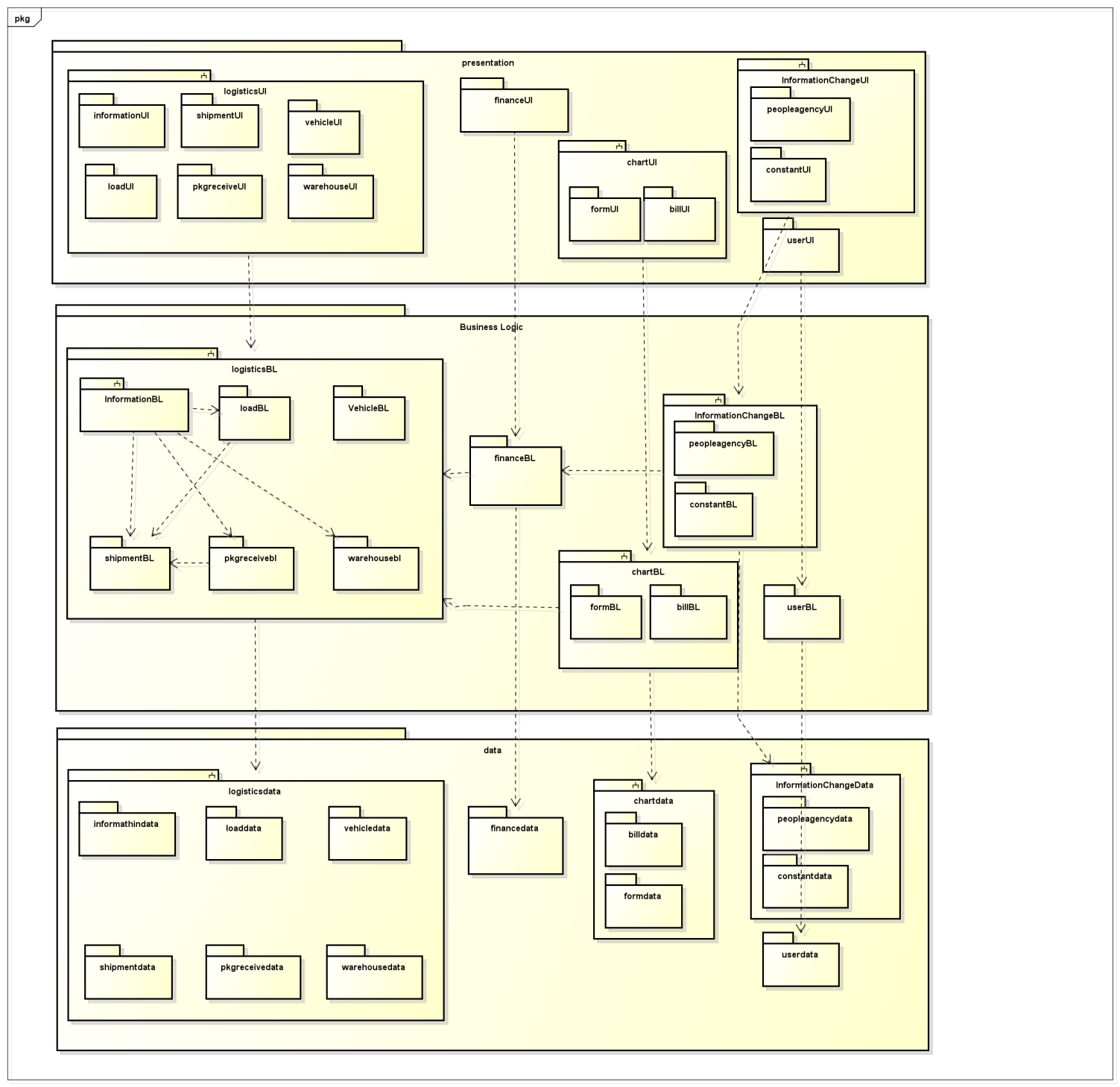


图2 软件体系结构逻辑设计方案

**4.组合视角**

**4.1开发包图**

物流管理系统的最终开发包设计如表1所示。

**表1 物流管理系统的最终开发包设计**

|  |  |
| --- | --- |
| 开发（物理）包 | 依赖的其他开发包 |
| Mainui | Logisticsui,financeui,chartui,informationchangeui,userui,vo |
| Logisticsui | Logisticsbl,界面类库包,vo |
| Logisticsservice |  |
| Logisticsbl | Logisticsservice, logisticsdataservice,po |
| Logisticsdataservice | Java RMI,po |
| Logisticsdata | Databaseutility,po, logisticsdataservice |
| Financeui | Financeblservice,界面类库包 |
| Financeservice |  |
| Financebl | Financeblservice,financedataservice,po,logisticsbl |
| Financedataservice | Java RMI,po |
| Financedata | Java RMI,po,databaseutility |
| Chartui | Chartbl,界面类库包 |
| Chartservice |  |
| Chartbl | Chartblservice,chartdataservice,po,logisticbl |
| Chartdataservice | Java RMI,po |
| Chartdata | Java RMI,po,databaseutility |
| Informationchangeui | Informationchangebl,界面类库包 |
| Informationchangeservice |  |
| Informationchangebl | Informationchangeblservice,informationchangedataservice,po,financebl |
| Informationchangedataservice | Java RMI,po |
| Informationchangedata | Java RMI,po,databaseutility |
| Userui | Userblservice，界面类库包 |
| Userservice |  |
| Userbl | UserInterface,UserDataClient,UserPo |
| Userdataservice | Java RMI,po |
| Userdata | Java RMI,po,databaseutility |
| Vo |  |
| Po |  |
| Utilitybl |  |
| 界面类库包 |  |
| Java RMI |  |
| Databaseutility | JDBC |

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

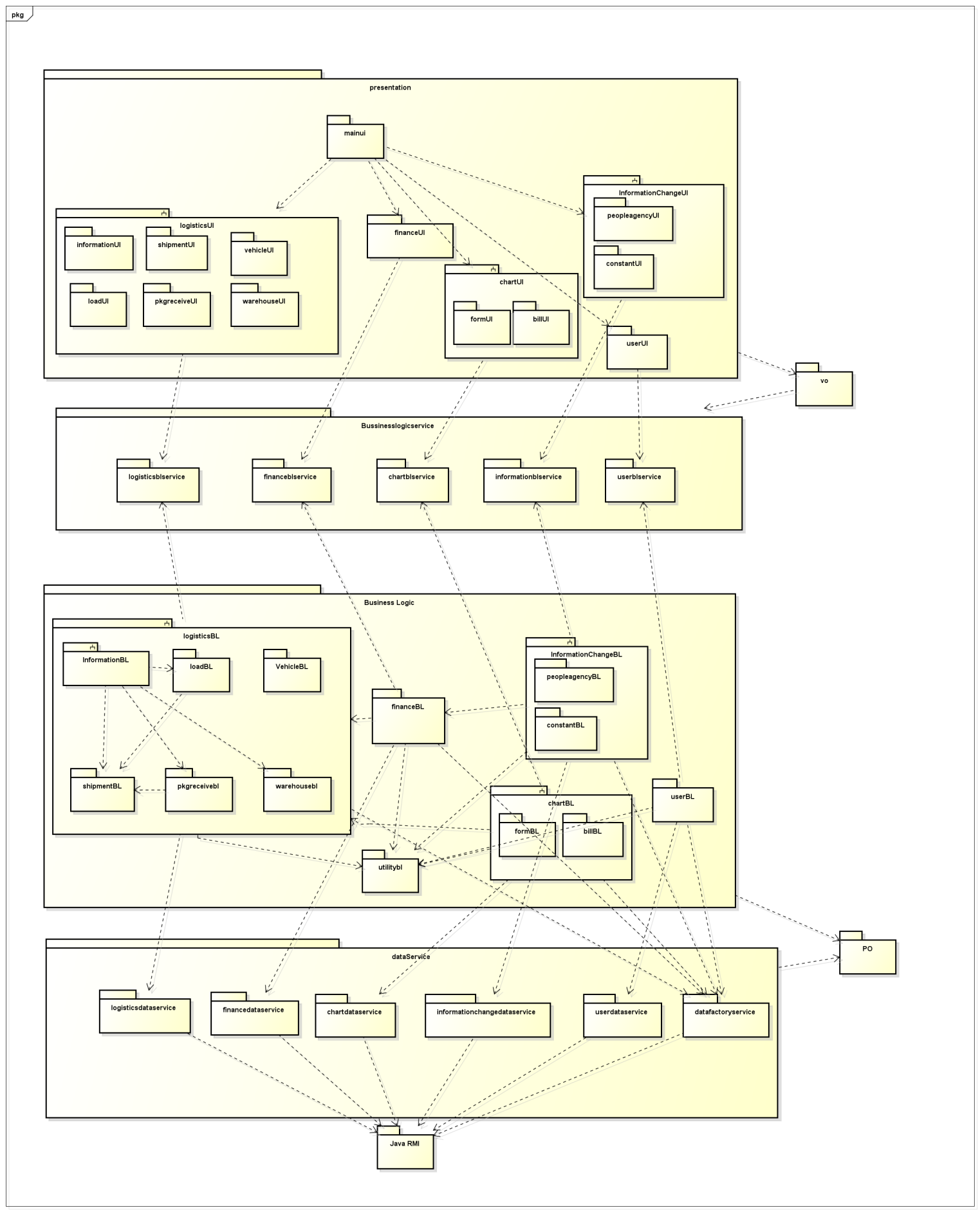


图3 物流管理系统客户端开发包图

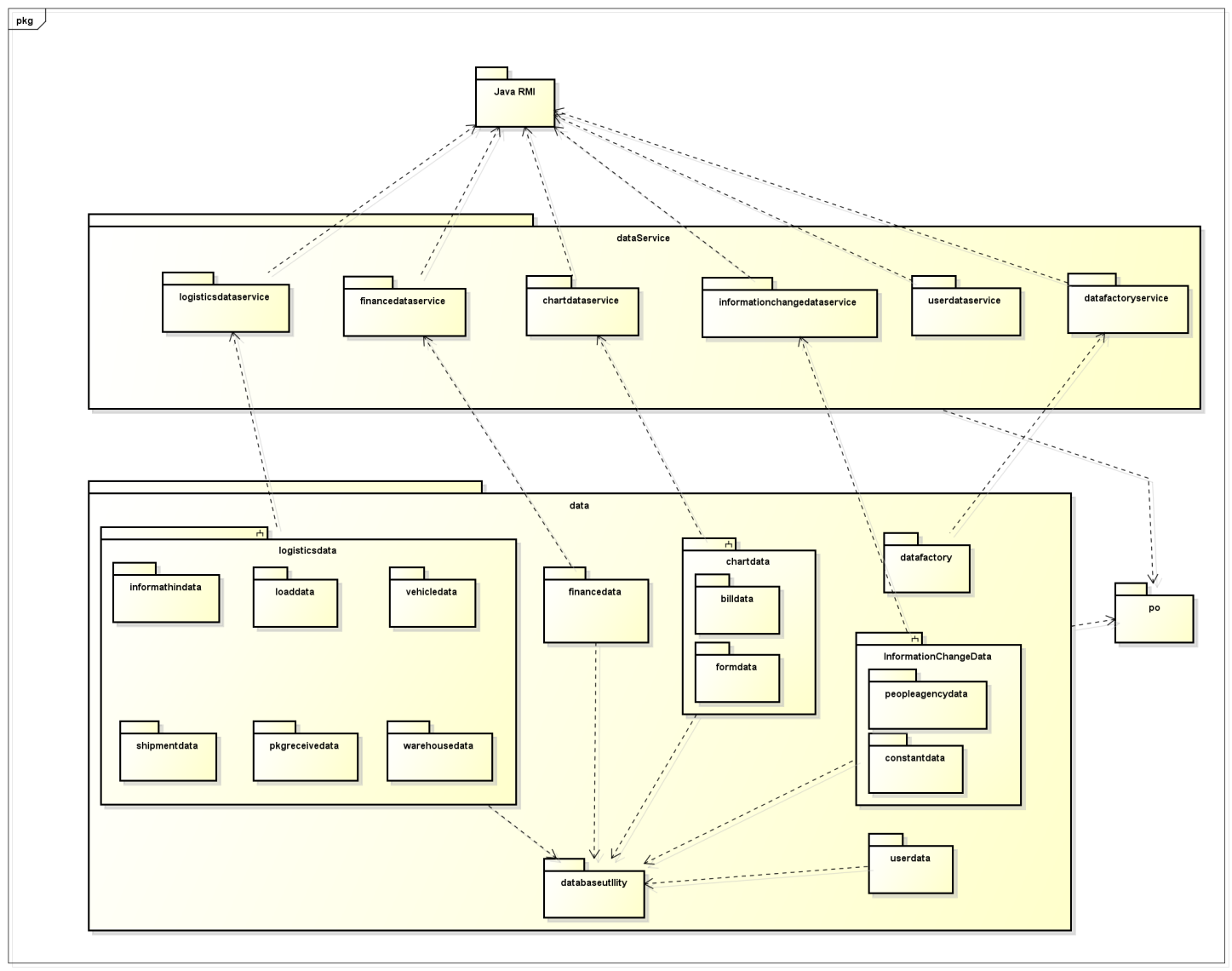


图4 物流管理系统服务器端开发包图

**4.2运行时进程**

在物流管理系统中，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，客户端是在客户端机器上运行，服务器端在服务器端机器上运行。

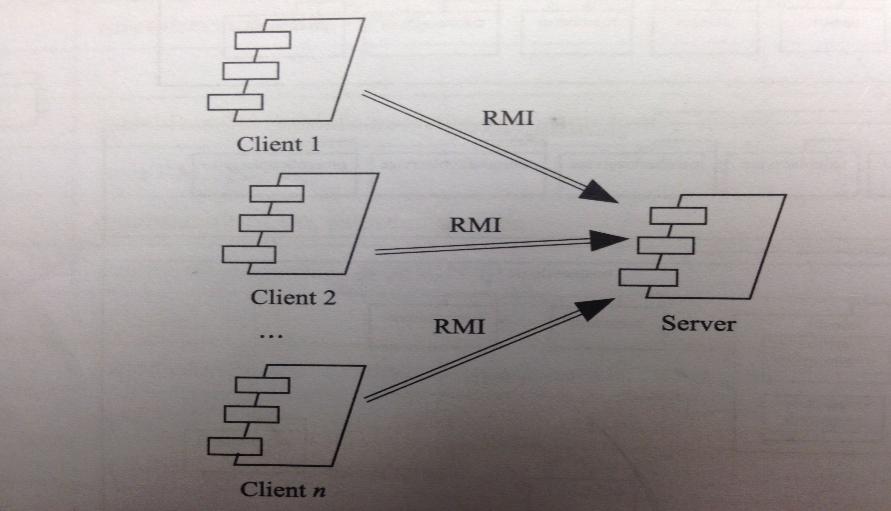


图5 进程图

**4.3物理部署**

物流管理系统中客户端构建是放在客户端机器上，服务器构建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JDK6.0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。

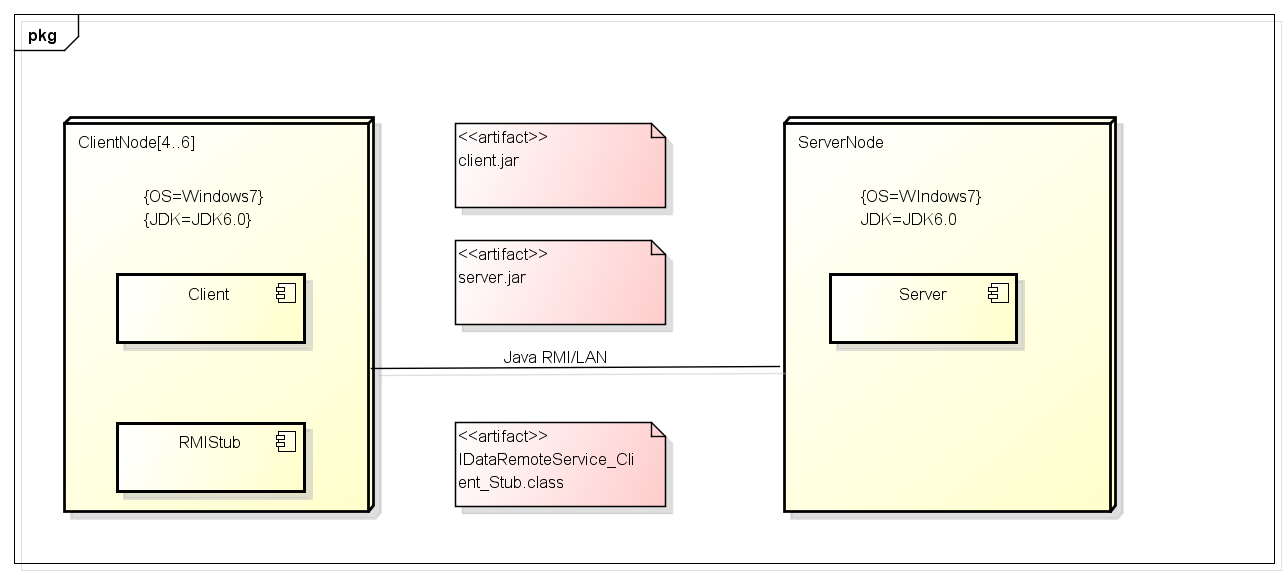


图6 部署图

**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。

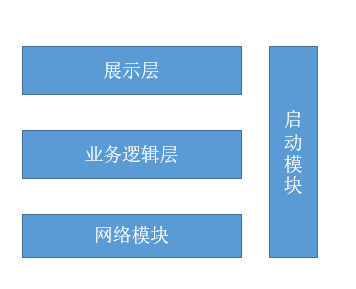


图7 客户端模块视图

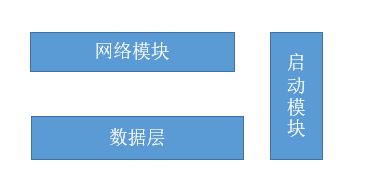


图8 服务器端模块视图

**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相应并进行业务处理逻辑 |
| 客户端网络模块 | 利用Java RMI机制查找RMI服务 |

**表3 服务器端各层的职责**

|  |  |
| --- | --- |
| **层** | **职责** |
| 启动模块 | 负责初始化网络通信机制，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 |
| Logisticsblservice  Financeblservice  Chartblservice  Informationblservice  Userblservice | 用户端界面层 | 用户端业务逻辑层 |
| Logisticsddateservice  Financedateservice  Chartdateservice  Informationdateservice  Userdateservice | 用户端业务逻辑层 | 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31个界面。界面跳转如图9所示。

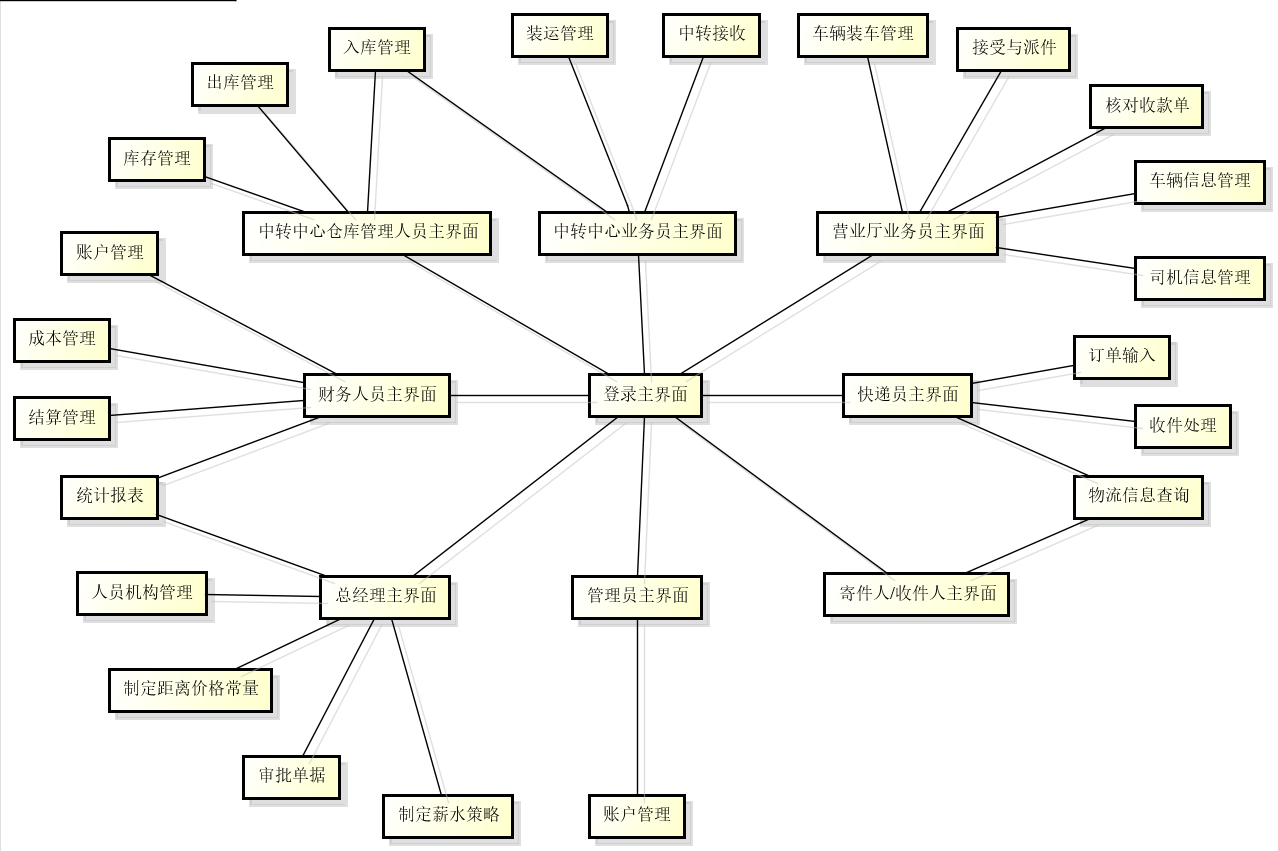


图9 用户界面跳转

服务器端和客户端的用户界面的设计接口是一致的，只是具体的页面不一样。用户界面类如图10所示。

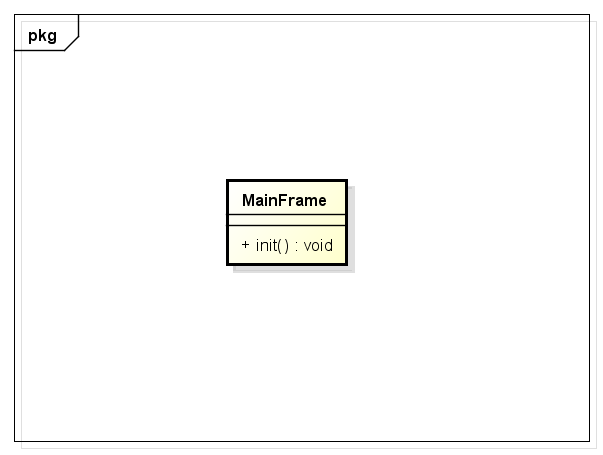


图10 用户界面类

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，负责界面的展示和界面跳转 |

5.2.2用户界面层模块的接口规范

用户界面层的接口规范如表6所示。

**表6 用户界面层模块的接口规范**

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

用户界面层需要的服务接口如表7所示

**表7 用户界面层模块需要的服务接口**

|  |  |
| --- | --- |
| 服务名 | 服务 |
| Busineesslogicservice.\*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。业务逻辑层的设计如图11所示。

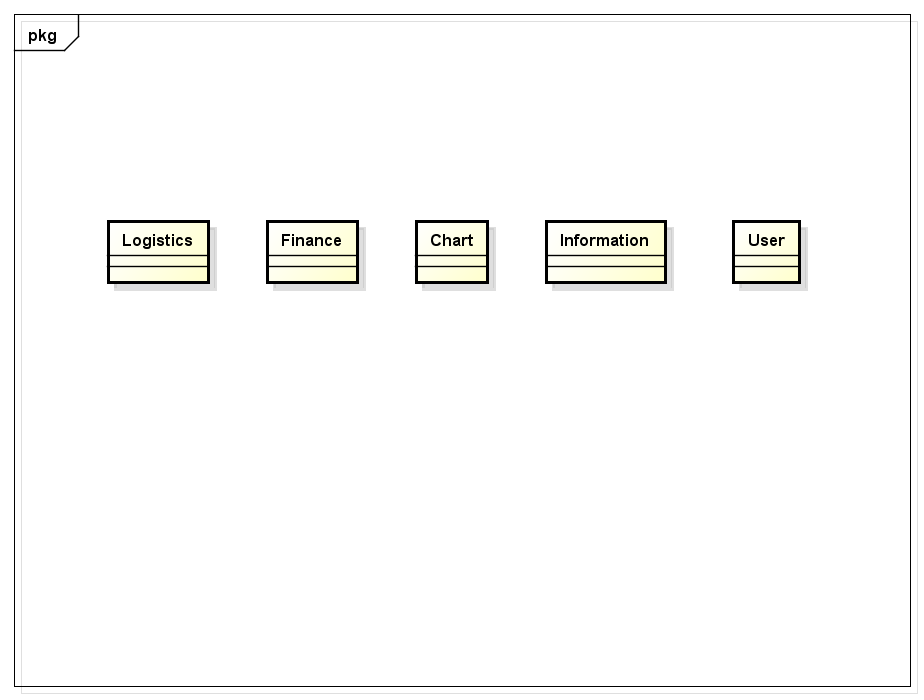


图11 业务逻辑层的设计

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** |
| Logisticsbl | 负责实现与物流系统相关所需要的服务 |
| Userbl | 负责实现对应与管理员登录管理所需要的服务 |
| Financebl | 负责实现与金融任务相关所需要的服务 |
| Informationchangebl | 负责实现与信息修改任务相关所需要的服务 |
| Chartbl | 负责实现与各种表格任务相关所需要的服务 |

5.3.2业务逻辑层模块的接口规范

**表9 Chartbl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| **Form.isFormatRight** | **语法** | **public boolean isFormatRight(String start\_day, String end\_day,ChartType chartType);** | |
| **前置条件** | **用户输入起始日期,终止日期,报表类型 申请查看报表** | |
| **后置条件** | **返回用户输入信息是否合法** | |
| **Form.getCostandBenefitChart** | **语法** | **public ArrayList<MoneyInfomationListVO> getCostandBenefitChart(String start\_day, String end\_day);** | |
| **前置条件** | **财务人员或总经理已经登陆，申请查看成本收益表** | |
| **后置条件** | **返回成本收益信息** | |
| **Form.getBusinessCircumstanceChart** | **语法** | **public ArrayList<ListVO> getBusinessCircumstanceChart(String start\_day, String end\_day);** | |
| **前置条件** | **财务人员或总经理已经登陆，申请查看经营情况表** | |
| **后置条件** | **返回经营情况表** | |
| **Form.exportChart** | **语法** | **public void exportChart(String strat\_day,String end\_day,ChartType chartType,String path);** | |
| **前置条件** | **用户已登录，申请打印导出报表** | |
| **后置条件** | **导出报表至指定路径** | |
| **Check.getAll** | **语法** | **public ArrayList<ListVO> getAll(DocType docType);** | |
| **前置条件** | **总经理选择需要审批的单据类别** | |
| **后置条件** | **返回相应的所有单据** | |
| **Check.passAllDoc** | **语法** | **public boolean passAllDoc(DocType docType);** | |
| **前置条件** | **总经理选择一键审核通过所有** | |
| **后置条件** | **审核通过所有单据，返回是否审核成功** | |
| **Check.passSelectedDoc** | **语法** | **public boolean passSelectedDoc(DocType docType,ArrayList<ListVO> listVO);** | |
| **前置条件** | **选择部分单据进行审核通过** | |
| **后置条件** | **审核通过选中单据，返回是否成功** | |
| **Check.failSelectedDoc** | **语法** | **public boolean failSelectedDoc(DocType docType,ArrayList<ListVO> listVO);** | |
| **前置条件** | **选择部分单据进行审核失败判定** | |
| **后置条件** | **审核失败选中单据，但会是否成功** | |
| **Check.ChangeDoc** | **语法** | **public boolean ChangeDoc(DocType docType,ListVO listVO);** | |
| **前置条件** | **总经理选择单据进行修改属性** | |
| **后置条件** | **修改单据内容，返回是否修改成功** | |
| **LogInquiry.getLogInfo** | **语法** | **public ArrayList<SystemLogVO> getLogInfo(String time);** | |
| **前置条件** | **用户输入正确格式的日期** | |
| **后置条件** | **系统显示匹配日的系统日志查询结果** | |
| **需要的服务（需接口）** | | | |
| **服务名** | | | **服务** |
| **CheckDataService.getDoc(DocType docType,String time)** | | | **得到特定时间内需要审批的某种单据PO对象集合** |
| **CheckDataService.changeDoc(ArrayList<ListPO> listPO)** | | | **更新PO对象** |
| **FormDataService.getBusinessCircumstanceChart(String start\_day, String end\_day);** | | | **得到经营情况PO** |
| **FormDataService.getCostandBenefitChart(String start\_day, String end\_day);** | | | **得到成本收益PO** |
| **LogInquiryDataService.getLogInfo(String time);** | | | **得到系统日志PO** |

**InformationChangebl接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| **CostantManagement.addCity** | **语法** | **public ResultMessage addCity(String name);** | |
| **前置条件** | **增加城市信息** | |
| **后置条件** | **添加城市信息并返回合法化** | |
| **CostantManagement.showDistanceChart** | **语法** | **public DistanceChartVO showDistanceChart();** | |
| **前置条件** | **显示城市距离分布表** | |
| **后置条件** | **返回距离常量表** | |
| **CostantManagement.showPriceChart** | **语法** | **public PriceChartVO showPriceChart();** | |
| **前置条件** | **显示价格常量表** | |
| **后置条件** | **返回价格常量表** | |
| **CostantManagement.setDistance** | **语法** | **public ResultMessage setDistance(String newDistance,String cityA,String cityB);** | |
| **前置条件** | **Input合法化通过** | |
| **后置条件** | **修改距离常量** | |
| **CostantManagement.setPrice** | **语法** | **public ResultMessage sePrice(String newPrice,String otherInformation);** | |
| **前置条件** | **Input合法化通过** | |
| **后置条件** | **修改价格常量** | |
| **PeopleAgency.salaryManage** | **语法** | **public ResultMessage salaryManage(String userID,String salary);** | |
| **前置条件** | **选择人员工资策略进行调整** | |
| **后置条件** | **人员工资修改** | |
| **PeopleAgency.addAgency** | **语法** | **public ResultMessage addAgency(AgencyVO agency);** | |
| **前置条件** | **总经理选择添加机构且该机构不重复** | |
| **后置条件** | **返回机构添加信息** | |
| **PeopleAgency.deleteAgency** | **语法** | **public ResultMessage deleteAgency(String agencyId);** | |
| **前置条件** | **输入id正确** | |
| **后置条件** | **机构信息删除** | |
| **PeopleAgency.updataAgency** | **语法** | **public ResultMessage updataAgency(AgencyVO agency);** | |
| **前置条件** | **机构人员信息修改格式正确** | |
| **后置条件** | **机构信息得到修改** | |
| **PeopleAgency.findAgency** | **语法** | **public AgencyVO findAgency(String id);** | |
| **前置条件** | **该id信息存在** | |
| **后置条件** | **返回一个被查找的VO数据** | |
| **需要的服务（需接口）** | | | |
| **服务名** | | | **服务** |
| **ConstantManagementDataService.submitDistance(String newDistance,String cityA,String cityB);** | | | **更新距离常量** |
| **ConstantManagementDataService.submitPrice(String newPrice,String otherInformation);** | | | **更新价格常量** |
| **ConstantManagementDataService.addCity(String name)** | | | **添加城市信息** |
| **ConstantManagementDataService.showDistanceChart** | | | **显示城市距离表信息** |
| **ConstantManagementDataService.showPriceChart** | | | **显示价格信息表** |
| **PeopleAgencyDataService.salaryManage(String userID,String salary);** | | | **更新员工收益情况** |
| **PeopleAgencyDataService.showAll()** | | | **显示人员机构信息列表** |
| **PeopleAgencyDataService.add(AgencyPO agency)** | | | **增加人员机构** |
| **PeopleAgencyDataService.delete(AgencyPO agency)** | | | **删除人员机构** |
| **PeopleAgencyDataService.update(AgencyPO agency)** | | | **更新修改人员机构** |
| **PeopleAgencyDataService.find(String id)** | | | **返回寻找的人员机构PO对象** |

**userbl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| **UserManagement.login** | **语法** | **public ResultMessage login(String admin,String password);** | |
| **前置条件** | **Password符合输入规范** | |
| **后置条件** | **查找是否存在相应User 根据password返回登录验证结果** | |
| **UserManagement.addUser** | **语法** | **public ResultMessage addUser(UserVO user);** | |
| **前置条件** | **Vo输入符合条件** | |
| **后置条件** | **增添一个用户VO** | |
| **UserManagement.removeUser** | **语法** | **public ResultMessage removeUser(String admin);** | |
| **前置条件** | **Admin输入符合规范** | |
| **后置条件** | **删除一个用户VO** | |
| **UserManagement.updataUser** | **语法** | **public ResultMessage updataUser(UserVO user);** | |
| **前置条件** | **VO修改符合规范** | |
| **后置条件** | **修改用户VO** | |
| **UserManagement.showAll** | **语法** | **public ArrayList<UserVO> showAll();** | |
| **前置条件** | **用户资料存在且合法** | |
| **后置条件** | **返回用户列表VO** | |
| **UserManagement.findonAdmin** | **语法** | **public UserVO findonAdmin(String admin);** | |
| **前置条件** | **用户admin输入合法** | |
| **后置条件** | **返回查找到的用户VO** | |
| **UserManagement.findonId** | **语法** | **public UserVO findonId(String id);** | |
| **前置条件** | **用户输入ID合法** | |
| **后置条件** | **返回查找到的用户VO** | |
| **需要的服务（需接口）** | | | |
| **服务名** | | | **服务** |
| **UserDataService.login(String admin,String password)** | | | **返回一个登录账号** |
| **UserDataService.findonId(String id)** | | | **返回一个PO** |
| **UserDataService.findonAdmin(String admin)** | | | **返回一个PO** |
| **UserDataService.insert(UserPO user)** | | | **新建一个PO** |
| **UserDataService.delete(UserPO user)** | | | **删除一个PO** |
| **UserDataService.updata(UserPO user)** | | | **更新一个PO** |
| **UserDataService.show** | | | **返回所有管理员用户PO** |

**Financebl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| **CostManagement.createMoneyOutlist** | **语法** | **public ResultMessage createMoneyOutlist(MoneyOutListVO moneyOut);** | |
| **前置条件** | **Password符合输入规范** | |
| **后置条件** | **查找是否存在相应User 根据password返回登录验证结果** | |
| **AccountManagement.login** | **语法** | **public boolean login(String admin,String password);** | |
| **前置条件** | **Admin和密码输入格式正确** | |
| **后置条件** | **返回是否有权限插叙账户** | |
| **AccountManagement.updadaCount** | **语法** | **public ResultMessage updadaCount(BankAccountVO countVo);** | |
| **前置条件** | **修改VO格式正确** | |
| **后置条件** | **修改一个用户VO** | |
| **AccountManagement.searchCount** | **语法** | **public BankAccountVO searchCount(String keyword);** | |
| **前置条件** | **Password格式正确** | |
| **后置条件** | **返回一个查找到的VO** | |
| **AccountManagement.createCount** | **语法** | **public ResultMessage createCount(BankAccountVO countVo);** | |
| **前置条件** | **不存在相同账户** | |
| **后置条件** | **新建一个账户** | |
| **AccountManagement.removeCount** | **语法** | **public ResultMessage removeCount(String name);** | |
| **前置条件** | **Name输入合法存在** | |
| **后置条件** | **删除该账户** | |
| **AccountManagement.show** | **语法** | **public ArrayList<BankAccountVO> show();** | |
| **前置条件** | **用户进入账户管理** | |
| **后置条件** | **返回查找到的所有账户VO** | |
| **BaseDataSetting.restart** | **语法** | **public ResultMessage restart();** | |
| **前置条件** | **用户选择期初建账** | |
| **后置条件** | **新建账目** | |
| **BaseDataSetting.addInstitutionInfo** | **语法** | **public ResultMessage addInstitutionInfo(AgencyVO agencyVO);** | |
| **前置条件** | **VO格式输入正确** | |
| **后置条件** | **新建一个VO** | |
| **BaseDataSetting.addCommodityInfo** | **语法** | **public ResultMessage addCommodityInfo(RepertoryVO repertoryVO);** | |
| **前置条件** | **VO格式输入正确** | |
| **后置条件** | **新建一个VO** | |
| **BaseDataSetting.addTruckInfo** | **语法** | **public ResultMessage addTruckInfo(TruckVO truck);** | |
| **前置条件** | **VO格式输入正确** | |
| **后置条件** | **新建一个VO** | |
| **BaseDataSetting.addFinanceInfo** | **语法** | **public ResultMessage addFinanceInfo(BankAccountVO bankAccountVO);** | |
| **前置条件** | **VO格式输入正确** | |
| **后置条件** | **新建一个VO** | |
| **需要的服务（需接口）** | | | |
| **服务名** | | | **服务** |
| **CostManagementDataService.createMoneyOutlist(MoneyOutListPO moneyOut)** | | | **新建一个PO** |
| **SettlementManageDataService.search(String start\_day, String end\_day,String Hallid)** | | | **返回一个POlist包换所需要的信息** |
| **AccountDataService.createCount(BankAccountPO countpo)** | | | **新建一个PO** |
| **AccountDataService.removeCount(String name)** | | | **删除一个PO** |
| **AccountDataService.updataCount(BankAccountPO countpo)** | | | **更新一个PO** |
| **AccountDataService.searchCount(String keyword)** | | | **查找一个PO** |
| **AccountDataService.ArrayList<BankAccountPO> show()** | | | **返回所有账户PO** |
| **BaseDataSettingDataService.addInstitutionInfo(AgencyPO agencyPO)** | | | **新建一个PO** |
| **BaseDataSettingDataService.addCommodityInfo(RepertoryPO repertoryPO);** | | | **新建一个PO** |
| **BaseDataSettingDataService.addTruckInfo(TruckPO truck);** | | | **新建一个PO** |
| **BaseDataSettingDataService.addFinanceInfo(BankAccountPO bankAccountPO);** | | | **新建一个PO** |

**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，包括对持久化数据的增、删、改、查。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，所以抽象了数据服务。数据层模块的描述具体如图13所示。

图13 数据层模块的职责

5.4.1数据层模块的职责

数据层模块的职责如表11所示。

表11 数据层模块的职责

|  |  |
| --- | --- |
| **模块** | **职责** |
|  |  |
|  |  |

5.4.2数据层模块的接口规范

表12 数据层模块的接口规范

ChartData模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **CheckDataService.getDoc** | **语法** | **public ArrayList<ListPO> getDoc(DocType docType,String time);** |
| **前置条件** | **经理选择审批单据类型与时间** |
| **后置条件** | **返回用户输入信息是否合法** |
| **CheckDataService.changeDoc** | **语法** | **public boolean changeDoc(ArrayList<ListPO> listPO);** |
| **前置条件** | **经理对单据信息进行修改** |
| **后置条件** | **更新一个PO** |
| **FormDataService.getBusinessCircumstanceChart** | **语法** | **public ArrayList<ListPO> getBusinessCircumstanceChart(String start\_day, String end\_day);** |
| **前置条件** | **财务人员或总经理已经登陆并申请查看经营情况表** |
| **后置条件** | **返回收款付款PO** |
| **FormDataService.getCostandBenefitChart** | **语法** | **public ArrayList<MoneyInformationListPO> getCostandBenefitChart(String start\_day, String end\_day);** |
| **前置条件** | **财务人员或总经理选择查看成本收益表** |
| **后置条件** | **返回成本收益PO** |

InformationChangeData模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **ConstantManageDataService.setDistance** | **语法** | **public boolean setDistance(String newDistance,String cityA,String cityB);** |
| **前置条件** | **输入的城市a、b与新距离都合法** |
| **后置条件** | **设置新的距离信息** |
| **ConstantManageDataService.addCity** | **语法** | **public boolean addCity(String name);** |
| **前置条件** | **新建城市名不重复** |
| **后置条件** | **更新城市距离表** |
| **ConstantManageDataService.showDistanceChart** | **语法** | **public DistanceChartPO showDistanceChart();** |
| **前置条件** | **表单信息存在** |
| **后置条件** | **返回一个PO** |
| **ConstantManageDataService.showPriceChart** | **语法** | **public PriceChartPO showPriceChart();** |
| **前置条件** | **表单信息存在** |
| **后置条件** | **返回一个PO** |
| **ConstantManageDataService.setPrice** | **语法** | **public boolean setPrice(String newPrice,String otherInformation);** |
| **前置条件** | **输入的价格合法** |
| **后置条件** | **设置新的价格常量信息** |
| **PeopleAgencyDataService.salaryManage** | **语法** | **public void salaryManage(String userID,String salary);** |
| **前置条件** | **输入的id存在 策略合法** |
| **后置条件** | **更新员工工资策略信息** |
| **PeopleAgencyDataService.showAll** | **语法** | **public ArrayList<AgencyPO> showAll() throws RemoteException;** |
| **前置条件** | **所有机构存在** |
| **后置条件** | **返回所有机构信息PO集合** |
| **PeopleAgencyDataService.add** | **语法** | **public void add(AgencyPO agency) throws RemoteException;** |
| **前置条件** | **机构信息不重复** |
| **后置条件** | **新建一个PO** |
| **PeopleAgencyDataService.delete** | **语法** | **public void delete(AgencyPO agency) throws RemoteException;** |
| **前置条件** | **机构信息存在** |
| **后置条件** | **删除一个PO** |
| **PeopleAgencyDataService.update** | **语法** | **public void update(AgencyPO agency) throws RemoteException;** |
| **前置条件** | **机构信息修改合法** |
| **后置条件** | **修改机构信息** |
| **PeopleAgencyDataService.find** | **语法** | **public AgencyPO find(String id) throws RemoteException;** |
| **前置条件** | **机构信息存在 id输入合法** |
| **后置条件** | **返回一个PO** |

UserData模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **UserDataService.login** | **语法** | **public UserPO login(String admin,String password);** |
| **前置条件** | **用户信息合法化** |
| **后置条件** | **返回一个PO** |
| **UserDataService.findonId** | **语法** | **public UserPO findonId(String id) throws RemoteException;** |
| **前置条件** | **用户id输入合法化** |
| **后置条件** | **返回一个PO** |
| **UserDataService.findonAdmin** | **语法** | **public UserPO findonAdmin(String admin) throws RemoteException;** |
| **前置条件** | **用户名admin输入合法化** |
| **后置条件** | **返回一个PO** |
| **UserDataService.insert** | **语法** | **public void insert(UserPO user) throws RemoteException;** |
| **前置条件** | **PO不重复** |
| **后置条件** | **新建一个PO** |
| **UserDataService.delete** | **语法** | **public void delete(UserPO user) throws RemoteException;** |
| **前置条件** | **删除的PO存在** |
| **后置条件** | **删除一个PO** |
| **UserDataService.update** | **语法** | **public void update(UserPO user) throws RemoteException;** |
| **前置条件** | **更新的数据PO合法化** |
| **后置条件** | **更新一个PO** |
| **UserDataService.show** | **语法** | **public ArrayList<UserPO> show() throws RemoteException;** |
| **前置条件** | **系统要求展现所有用户信息** |
| **后置条件** | **返回一个list<PO>** |

FinanceData模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| **CostManagementDataService.createMoneyOutlist** | **语法** | **public boolean createMoneyOutlist(MoneyOutListPO moneyOut);** |
| **前置条件** | **用户信息合法化** |
| **后置条件** | **返回一个PO** |
| **SettlementManagementDataService.search** | **语法** | **public ArrayList<MoneyOutListPO> search(String start\_day, String end\_day,String Hallid);** |
| **前置条件** | **日期与营业厅格式输入正确** |
| **后置条件** | **返回一个POList** |
| **BaseDataSettingDataService.addInstitutionInfo** | **语法** | **public boolean addInstitutionInfo(AgencyPO agencyPO);** |
| **前置条件** | **PO输入合法** |
| **后置条件** | **新建一个PO** |
| **BaseDataSettingDataService.addCommodityInfo** | **语法** | **public boolean addCommodityInfo(RepertoryPO repertoryPO);** |
| **前置条件** | **PO输入合法** |
| **后置条件** | **新建一个PO** |
| **BaseDataSettingDataService.addTruckInfo** | **语法** | **public boolean addTruckInfo(TruckPO truck);** |
| **前置条件** | **PO输入合法** |
| **后置条件** | **新建一个PO** |
| **BaseDataSettingDataService.addFinanceInfo** | **语法** | **public boolean addFinanceInfo(BankAccountPO bankAccountPO);** |
| **前置条件** | **PO输入合法** |
| **后置条件** | **新建一个PO** |
| **AccountDataService.createCount** | **语法** | **public boolean createCount(BankAccountPO countpo);** |
| **前置条件** | **PO输入合法** |
| **后置条件** | **新建一个PO** |
| **AccountDataService.removeCount** | **语法** | **public boolean removeCount(String name);** |
| **前置条件** | **Name输入合法** |
| **后置条件** | **删除一个PO** |
| **AccountDataService.updataCount** | **语法** | **public boolean updataCount(BankAccountPO countpo);** |
| **前置条件** | **PO输入合法** |
| **后置条件** | **更新一个PO** |
| **AccountDataService.searchCount** | **语法** | **public BankAccountPO searchCount(String keyword);** |
| **前置条件** | **Keyword输入合法** |
| **后置条件** | **返回一个PO** |
| **AccountDataService.show** | **语法** | **public ArrayList<BankAccountPO> show();** |
| **前置条件** | **系统要求展现所有用户信息** |
| **后置条件** | **返回一个list<PO>** |

**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。

·UserPO类包含用户的用户名、密码属性。

·constantPO类包含薪水计算、价格、距离常量等静态常量

·informationPO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；托运货物信息（原件数、实际重量、体积、内件品名、尺寸）；包装费（纸箱、木箱、快递袋、其它）；费用合计；订单条形码号（10位数）；经济快递、标准快递、特快；种类。

·shipmentPO类包含装车日期、本营业厅编号（025城市编码+000鼓楼营业厅）、汽运编号 （营业厅编号+20150921日期+00000编码 、五位数字）、到达地（本地中转中心或者其它营业厅）、车辆代号、监装员、押运员、本次装箱所有订单条形码号）、运费（运费根据出发地和目的地自动生成）

·vehiclePO类包含车辆代号（城市编号（电话号码区号南京025）+营业厅编号（000三位数字）+000三位数字）、车牌号（苏A 00000）、服役时间；司机信息（司机编号、营业厅编号、000、姓名、出生日期、身份证号、手机、性别、行驶证期限）

·LoadPO类包含装车日期，本中转中心中转单编号（区别航运货运汽运），航班号（或者火车车次，汽车车次），出发地，到达地，货柜号（车厢号），监装员，本次装箱所有托运单号，运费

·pkgreceivePO类包含

·warehousePO类包含位置（区、排、架、位机）及位置上放置的商品。

·formPO类包含表格的信息。

·billPO类包含表单的信息。

·financePO类包含账户信息。

**6.2 Txt持久化格式**

给出一个txt格式