

快递物流系统软件体系结构描述文档

V1.0



2015-10-15

第7组

|  |  |  |  |
| --- | --- | --- | --- |
| 变更历史 | | | |
| 版本号 | **更新时间** | **更新人** | **更新摘要** |
|  |  |  |  |
|  |  |  |  |

目录

[1引言 3](#_Toc433663595)

[1.1编制目的 3](#_Toc433663596)

[1.2词汇表 3](#_Toc433663597)

[1.3参考资料 3](#_Toc433663598)

[2产品概述 3](#_Toc433663599)

[3逻辑视角 3](#_Toc433663600)

[4组合视角 6](#_Toc433663601)

[4.1开发包图 6](#_Toc433663602)

[4.2运行时进程 9](#_Toc433663603)

[4.3物理部署 9](#_Toc433663604)

[5接口视角 10](#_Toc433663605)

[5.1模块的职责 10](#_Toc433663606)

[5.2用户界面层的分解 15](#_Toc433663607)

[5.2.1用户界面层模块的职责 16](#_Toc433663608)

[5.2.2用户界面层模块的接口规范 17](#_Toc433663609)

[5.2.3用户界面模块设计原理 17](#_Toc433663610)

[5.3业务逻辑层的分解 17](#_Toc433663611)

[5.3.1业务逻辑层模块的职责 18](#_Toc433663612)

[5.3.2业务逻辑层模块的接口规范 18](#_Toc433663613)

[5.4数据层的分解 24](#_Toc433663614)

[5.4.1数据层模块的职责 28](#_Toc433663615)

[5.4.2数据层模块的接口规范 28](#_Toc433663616)

[6信息视角 32](#_Toc433663617)

[6.1数据持久化对象 32](#_Toc433663618)

[6.2 Txt持久化格式 35](#_Toc433663619)

[6.3数据库表 35](#_Toc433663620)

# 1引言

## 1.1编制目的

本报告详细完成对快递物流系统的概要设计，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** |
| bl | 业务逻辑 |  |
|  |  |  |

## 1.3参考资料

1.快递物流系统用例文档v1.2

2.项目描述v4

3.需求规格说明文档v1.2

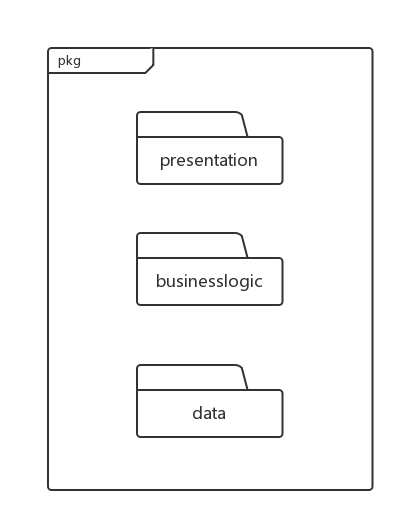
# 2产品概述

快递物流系统是为xxx物流公司开发的新的辅助办公系统，该系统开发的目标是帮助该公司高效处理日益增长的业务，提高员工的工作效率和用户的满意度。

该系统的业务目标为帮助公司保持合理库存，提高用户服务体验，增加业务额，提高公司职工的工作效率，并为总经理的决策做支持。

# 3逻辑视角

快递物流系统中，选择了分层体系结构风格，将系统分为3层（展示层、业务逻辑层、数据层）能很好的表示整个高层抽象。展示层包含GUI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，其中展示层和业务逻辑层部署在客户端，数据层部署在服务器。分层体系结构的逻辑视角如图1所示。

图1 参考体系结构风格的包图表达逻辑视角

快递物流系统的主要功能有：查询物流信息、收件、派件、车辆信息管理、司机信息管理、营业厅-营业厅转运、营业厅-中转中心转运、中转中心-中转中心转运、快件入库、快件出库、初始化仓库信息、库存查看、库存盘点、收入管理、成本管理、期初建账、银行账户管理、查询系统日志、查询报表、人员管理、机构管理、审批单据、制定薪水策略、制定价格/距离策略、用户账户管理。经过分析发现下列功能可以在设计时合并，因为它们使用的信息和行为相似：

* 查询物流信息🡪物流状态
* 收件、派件、营业厅-营业厅、营业厅-中转中心、中转中心-中转中心转运🡪运输
* 司机、车辆、人员、机构信息管理、银行账户管理、用户账户管理🡪信息管理
* 快件入库、快件出库、初始化仓库信息、库存查看、库存盘点🡪库存
* 收入管理、成本管理、期初建账、查询报表🡪财务
* 查询系统日志🡪日志
* 审批单据🡪单据
* 制定薪水策略、制定价格/距离策略🡪策略

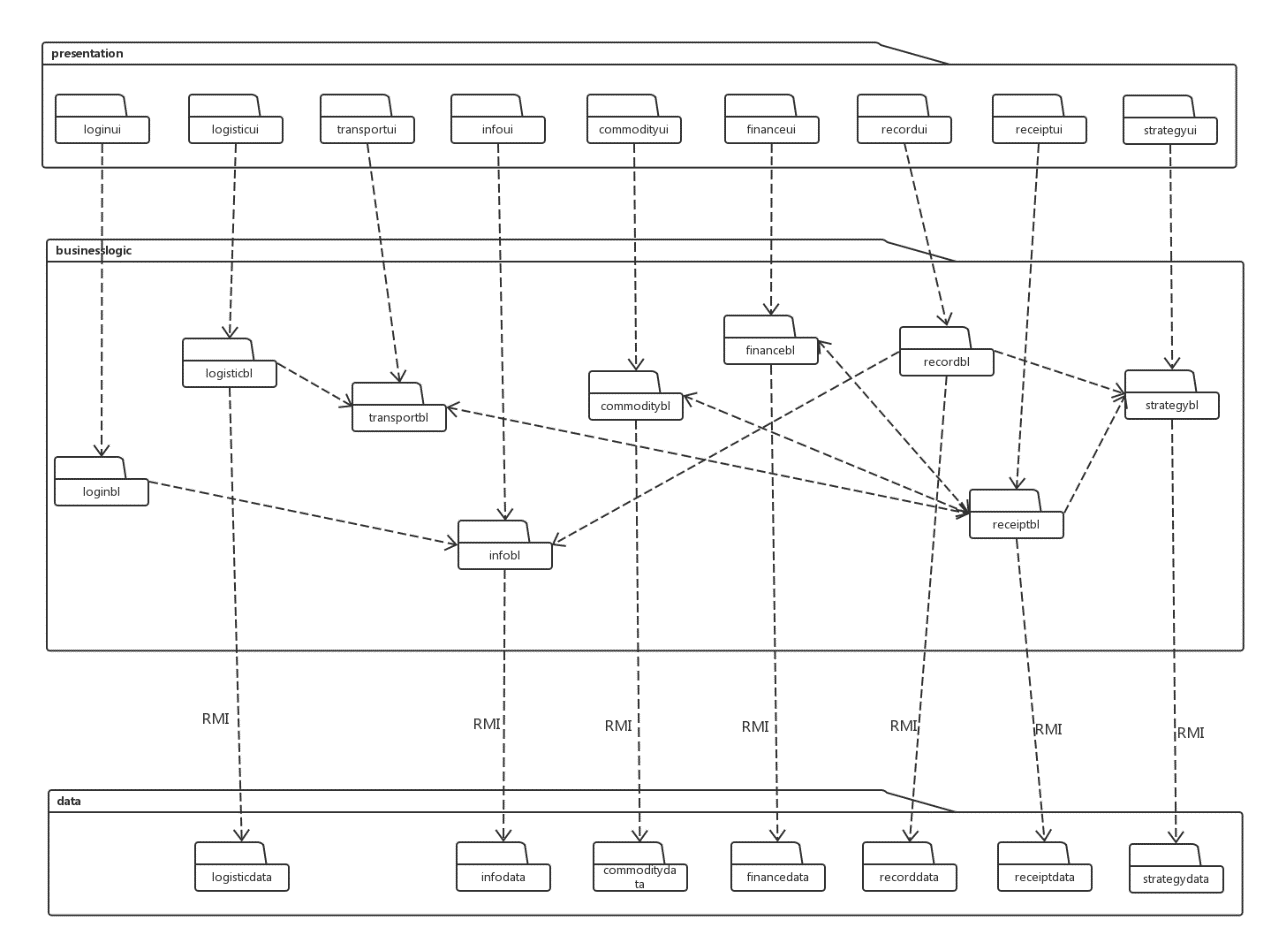
从以上合并后的功能点可以设计出对应的逻辑包：

|  |  |
| --- | --- |
| **功能** | **对应逻辑包** |
| 物流状态 | LogisticsUI, LogisticsBL, LogisticsData |
| 运输 | TransportUI, TransportBL |
| 信息管理 | InfoUI, InfoBL, InfoData |
| 库存 | CommodityUI, CommodityBL, CommodityData |
| 财务 | FinanceUI, FinanceBL, FinanceData |
| 日志 | RecordUI, RecordBL, RecordData |
| 单据 | ReceiptUI, ReceiptBL, ReceiptData |
| 策略 | StrategyUI, StrategyBL, StrategyData |

再加上不同功能之间相互信息使用，上表修改为：

|  |  |
| --- | --- |
| **功能** | **对应逻辑包** |
| 物流状态 | LogisticsUI, LogisticsBL, LogisticsData  TransportBL |
| 运输 | TransportUI, TransportBL  ReceiptBL, ReceiptData |
| 信息管理 | InfoUI, InfoBL, InfoData |
| 库存 | CommodityUI, CommodityBL, CommodityData  ReceiptBL, ReceiptData |
| 财务 | FinanceUI, FinanceBL, FinanceData  ReceiptBL, ReceiptData  StrategyBL, StrategyData |
| 日志 | RecordUI, RecordBL, RecordData  InfoBL, InfoData  StrategyBL, StrategyData |
| 单据 | ReceiptUI, ReceiptBL, ReceiptData  TransportBL  CommodityBL, CommodityData  FinanceBL, FinanceData  StrategyBL, StrategyData |
| 策略 | StrategyUI, StrategyBL, StrategyData |

由上表再加上登录模块（即新增LoginUI, LoginBL）可以生成软件体系结构逻辑设计方案，如图2所示。

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

快递物流系统的最终开发包设计如表1所示。

**表1 快递物流系统的最终开发包设计**

|  |  |
| --- | --- |
| **开发（物理）包** | **依赖的其他开发包** |
| LoginUI | LoginBLService, LogisticUI, TransportUI, InfoUI, CommodityUI, FinanceUI, RecordUI, ReceiptUI, StrategyUI, vo, 界面类库包 |
| LoginBLService | vo |
| LoginBL | LoginBLService, InfoBL, UtilityBL, vo |
| LogisticUI | LogisticBLService, vo, 界面类库包 |
| LogisticBLService | vo |
| LogisticBL | LogisticBLService, LogisticDataService, TransportBL, UtilityBL |
| LogisticDataService | Java RMI, po |
| LogisticData | Java RMI, po, DatabaseUtility |
| TransportUI | TransportBLService, vo, 界面类库包 |
| TransportBLService | vo |
| TransportBL | TransportBLService, UtilityBL |
| InfoUI | InfoBLService, vo, 界面类库包 |
| InfoBLService | vo |
| InfoBL | InfoBLService, InfoDataService, UtilityBL |
| InfoDataService | Java RMI, po |
| InfoData | Java RMI, po, DatabaseUtility |
| CommodityUI | CommodityBLService, vo, 界面类库包 |
| CommodityBLService | vo |
| CommodityBL | CommodityBLService, CommodityDataService, UtilityBL |
| CommodityDataService | Java RMI, po |
| CommodityData | Java RMI, po, DatabaseUtility |
| FinanceUI | FinanceBLService, vo, 界面类库包 |
| FinanceBLService | vo |
| FinanceBL | FinanceBLService, FinanceDataService, StrategyBL, UtilityBL |
| FinanceDataService | Java RMI, po |
| FinanceData | Java RMI, po, DatabaseUtility |
| RecordUI | RecordBLService, vo, 界面类库包 |
| RecordBLService | vo |
| RecordBL | RecordBLService, RecordDataService, InfoBL, StrategyBL, UtilityBL |
| RecordDataService | Java RMI, po |
| RecordData | Java RMI, po, DatabaseUtility |
| ReceiptUI | ReceiptBLService, vo, 界面类库包 |
| ReceiptBLService | vo |
| ReceiptBL | ReceiptBLService, ReceiptDataService, TransportBL, CommodityBL, FinanceBL, StrategyBL, UtilityBL |
| ReceiptDataService | Java RMI, po |
| ReceiptData | Java RMI, po, DatabaseUtility |
| StrategyUI | StrategyBLService, vo, 界面类库包 |
| StrategyBLService | vo |
| StrategyBL | StrategyBLService, StrategyDataService, UtilityBL |
| StrategyDataService | Java RMI, po |
| StrategyData | Java RMI, po, DatabaseUtilityBL |
| vo |  |
| po |  |
| Java RMI |  |
| 界面类库包 |  |
| UtilityBL |  |
| DatabaseUtility | JDBC |

快递物流系统的客户端开发包图如图3所示，服务器端开发包图如图4所示

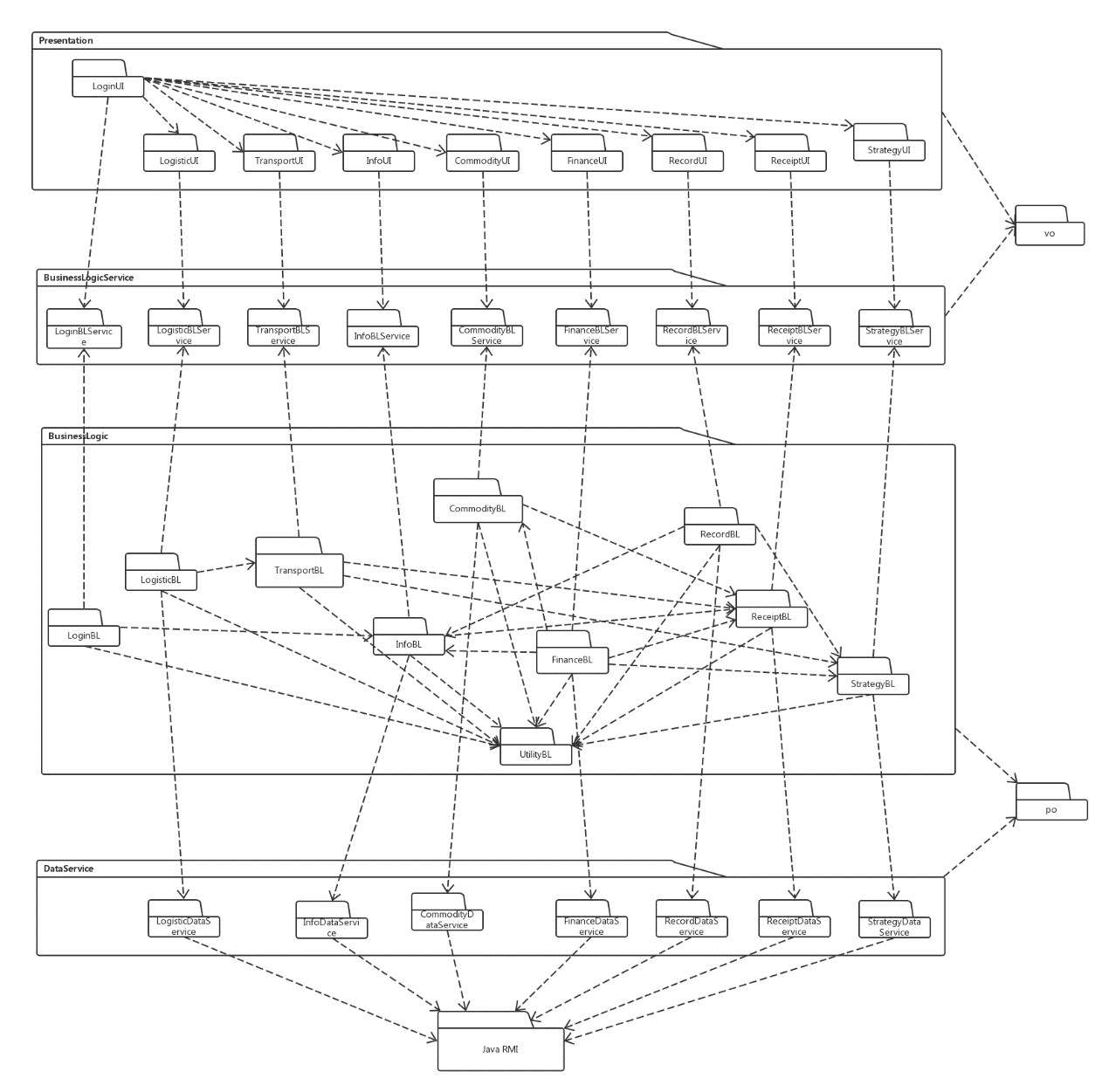


图3 快递物流系统客户端开发包图

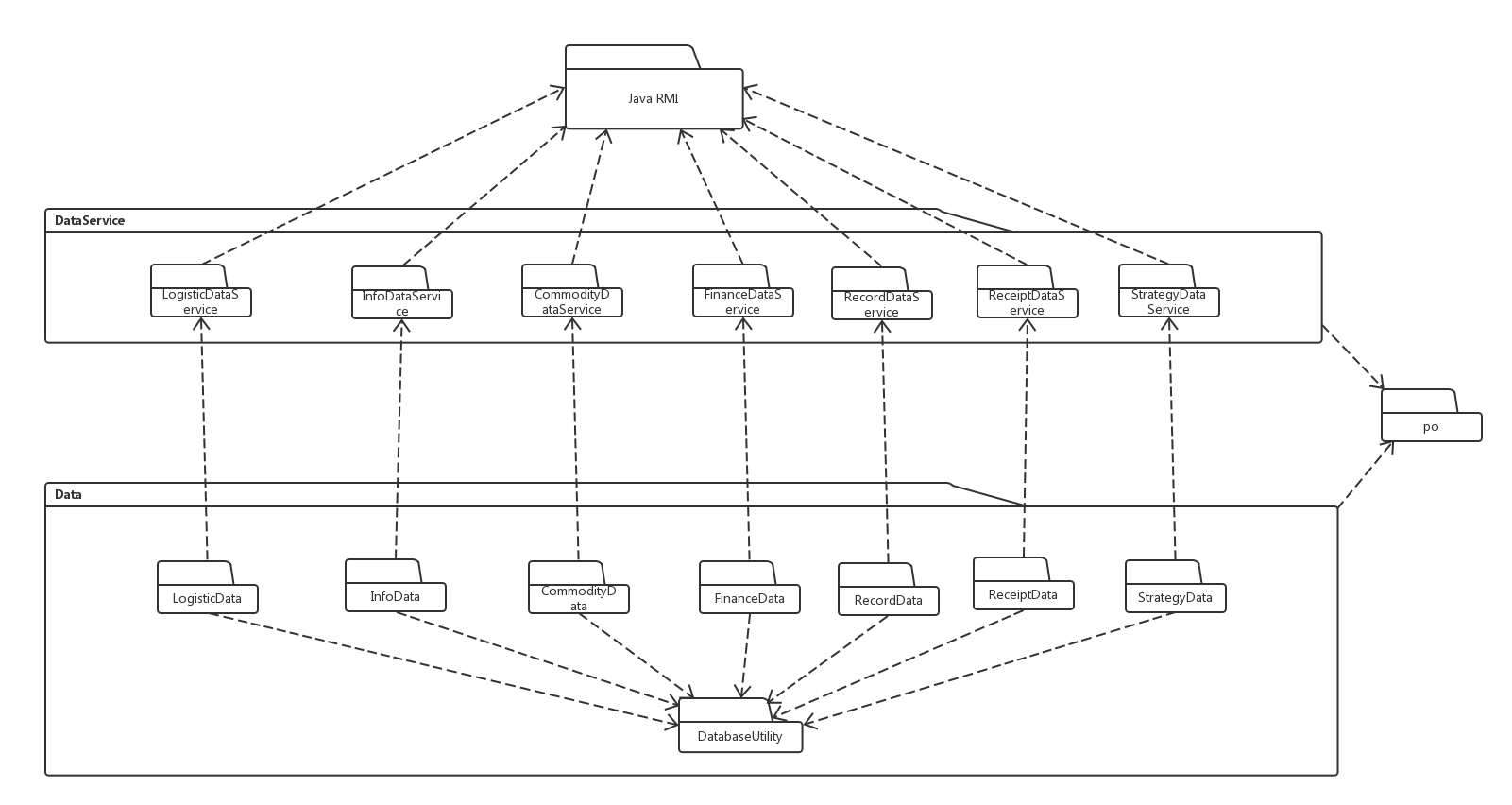
****

图4 快递物流系统服务器端开发包图

## 4.2运行时进程

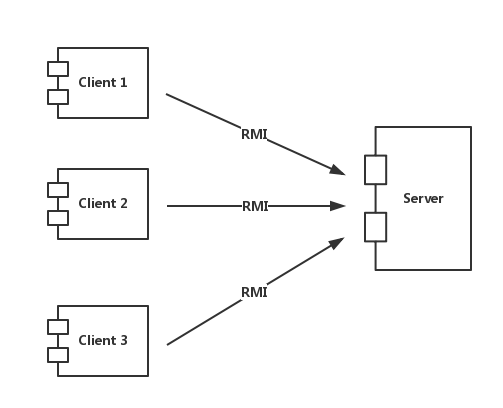
 在快递物流系统中，会有多个客户端进程和一个服务器进程，其进程图如图5所示。结合部署图，客户端进程在客户端机器上运行，服务器进程在服务器端机器上运行。

图5 进程图

## 4.3物理部署

快递物流系统中客户端构件是放在客户端机器上的，服务器端构件是放在服务器端机器上的。在客户端节点上，还要部署RMIStub构件。由于Java RMI属于JDK6.0及其以后版本的一部分，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。

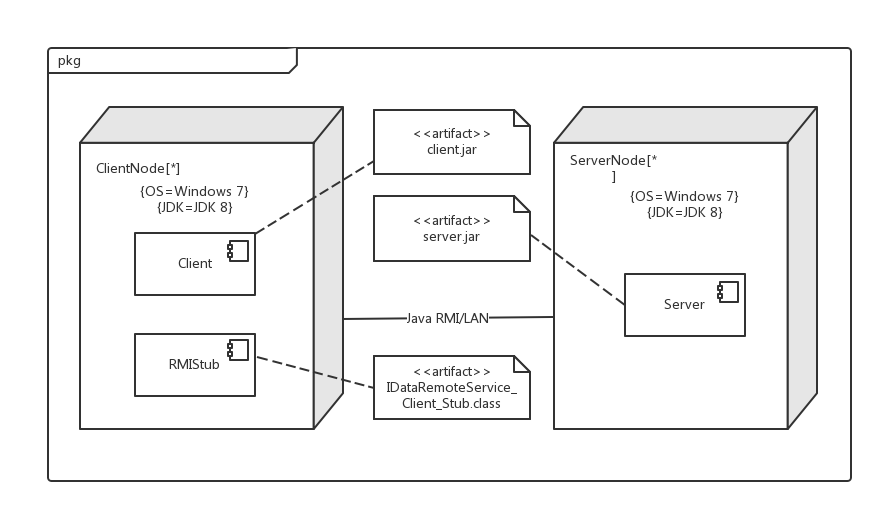
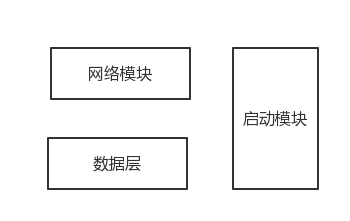
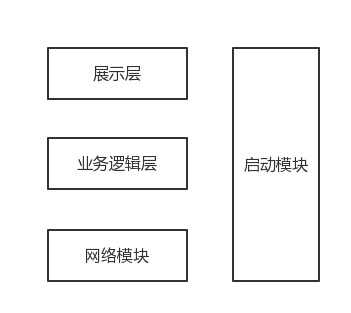


图6 部署图

# 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 |
| 展示层 | 基于窗口的快递物流系统客户端用户界面 |
| 业务逻辑层 | 对于用户界面的输入进行响应并进行业务处理逻辑 |
| 客户端网络模块 | 利用Java RMI机制查找RMI服务 |

**表3 服务器端各层的职责**

|  |  |
| --- | --- |
| **层** | **职责** |
| 启动模块 | 负责初始化网络通信机制 |
| 数据层 | 负责数据的持久化金数据访问接口 |
| 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** |
| LoginBLService  LogisticBLService  TranportBLService  InfoBLService  CommodityBLService  FinanceBLService  RecordBLService  ReceiptBLService  StrategyBLService | 客户端展示层 | 客户端业务逻辑层 |
| LogisticDataService  InfoDataService  CommodityDataService  FinanceDataService  RecordDataService  ReceiptDataService  StrategyDataService | 客户端业务逻辑层 | 服务器端数据层 |

图9-图17展示了各个功能层之间的接口调用。

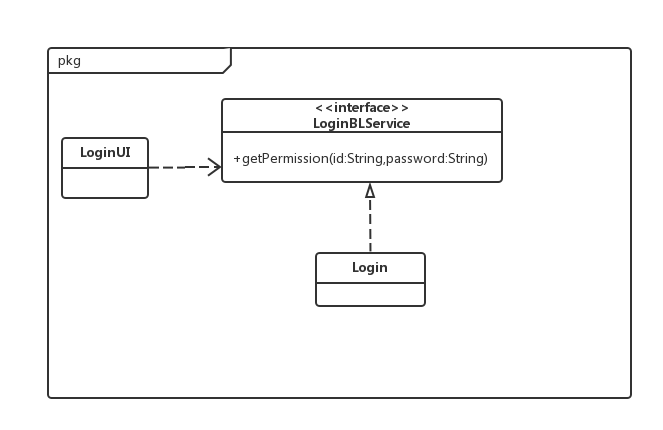


图9 登录层之间调用的接口

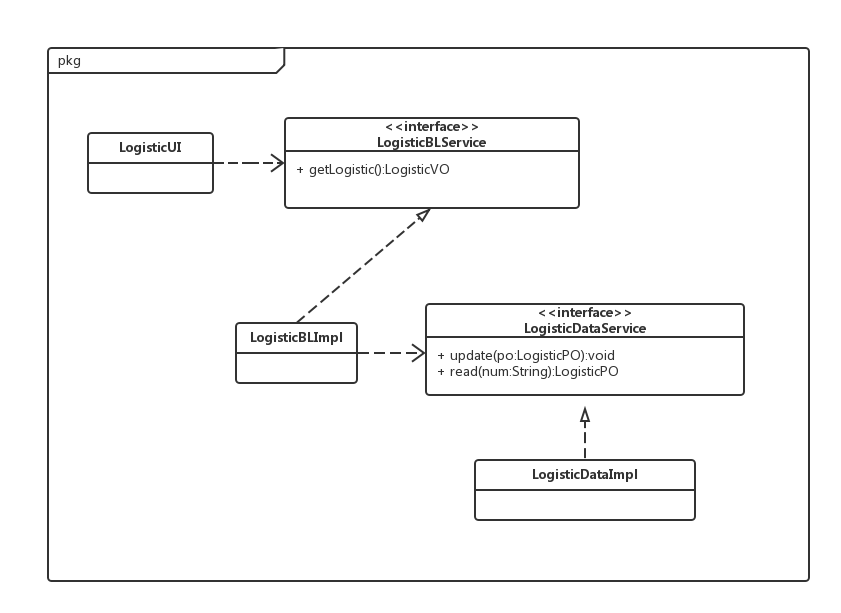


图10 物流信息用例层之间调用的接口

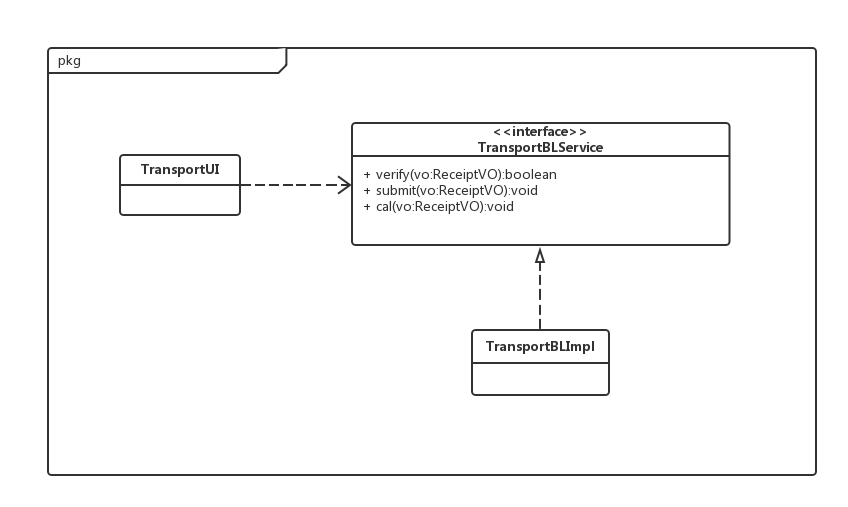
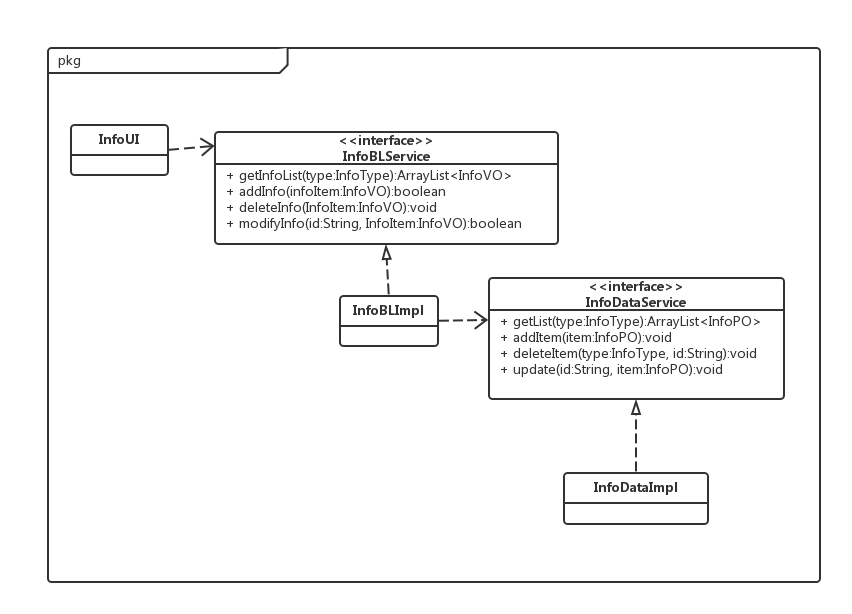


图11 转运用例层之间调用的接口

图12 信息管理用例层之间调用的接口

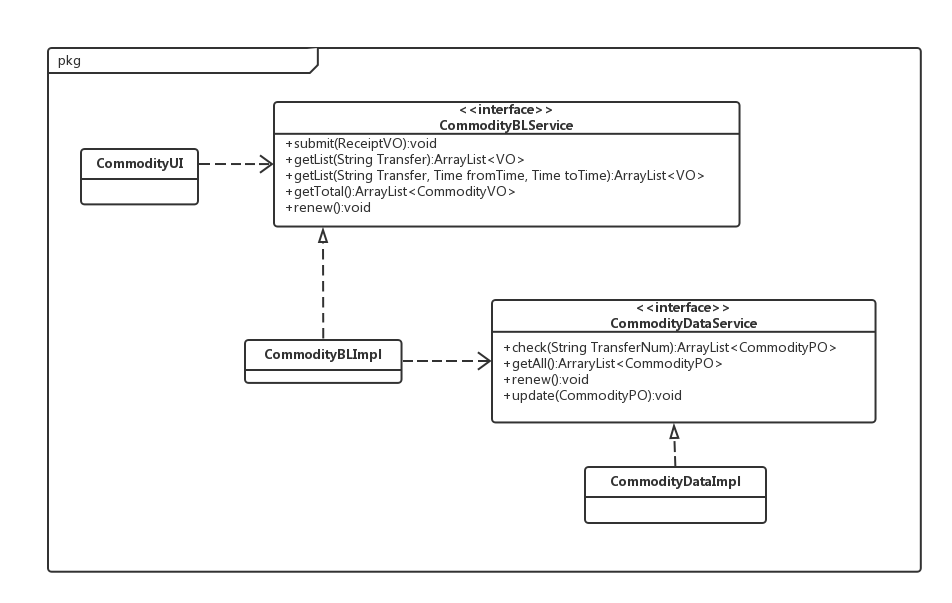


图13 库存用例层之间调用的接口

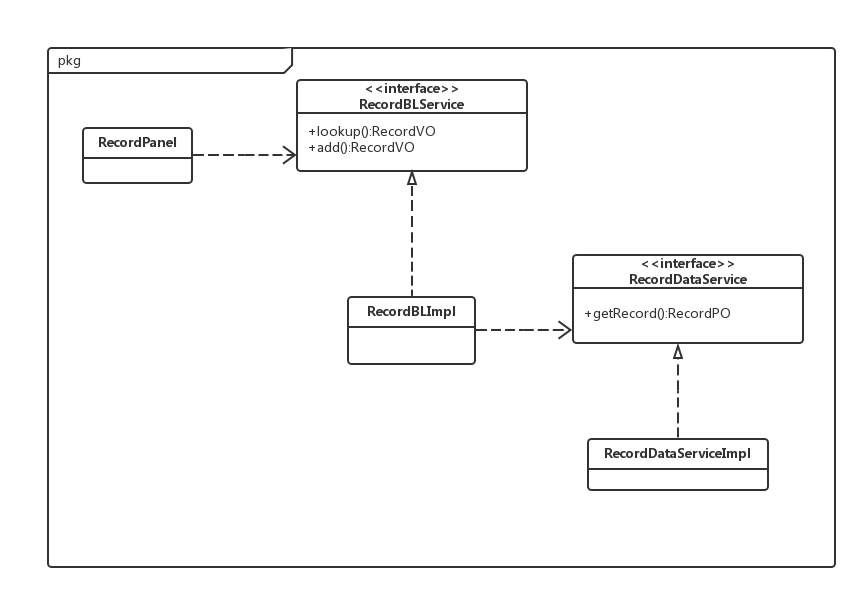


图14 系统日志用例层之间调用的接口

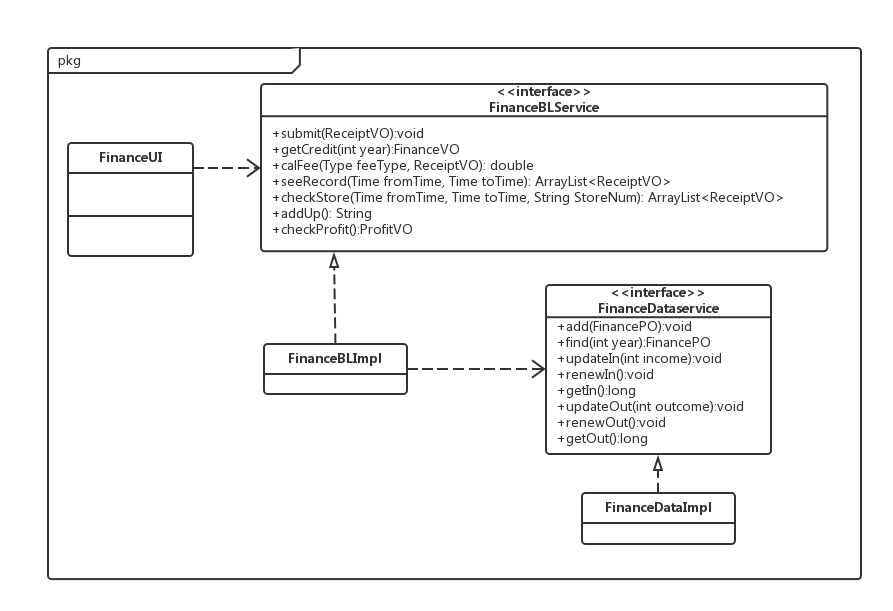


图15 财务用例层之间调用的接口

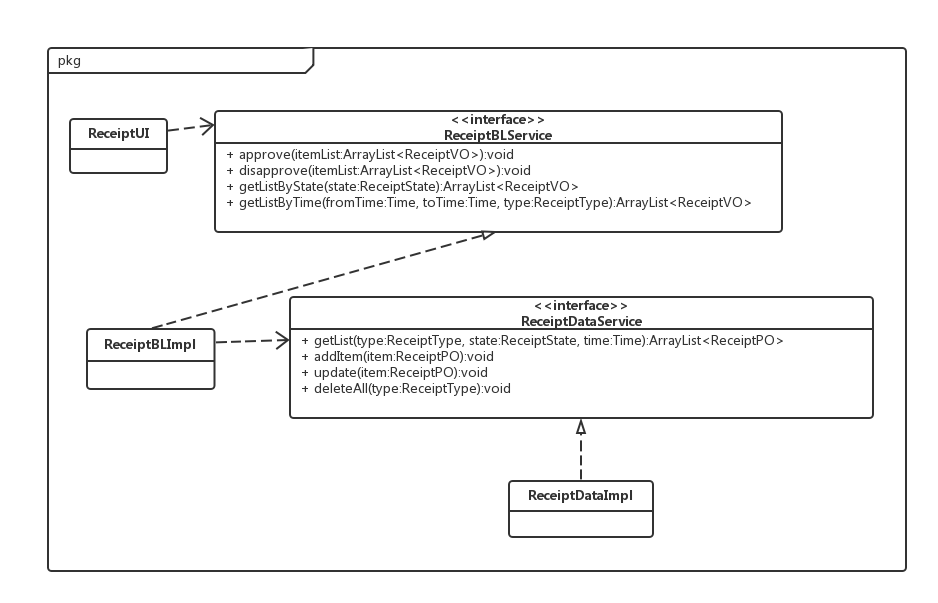


图16 单据用例层间调用的接口

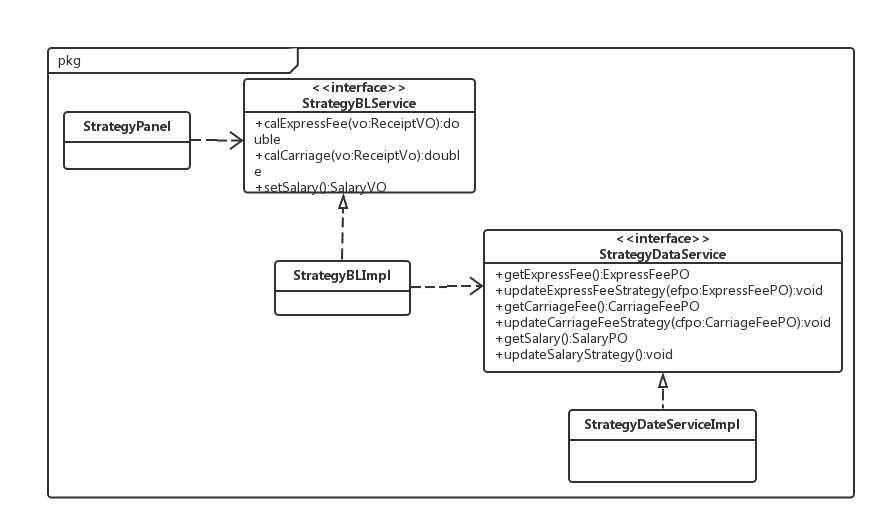


图17 策略用例层之间调用的接口

## 5.2用户界面层的分解

根据需求，系统存在26个用户界面：登录界面、快递员主界面、营业厅业务员主界面、中转中心业务员主界面、仓库管理人员主界面、总经理主界面、财务人员主界面、管理员主界面、物流信息查看界面、收件界面、派件界面、订单信息查看界面、货物运出/到达界面、司机/车辆管理界面、货物中转/到达界面、出库/入库界面、库存盘点界面、库存查看/初始化界面、用户管理界面、制定薪水策略界面、制定价格/距离策略界面、单据审批界面、机构/人员管理界面、收入/支出管理界面、银行账户管理界面、期初建账界面。界面跳转如图17所示。

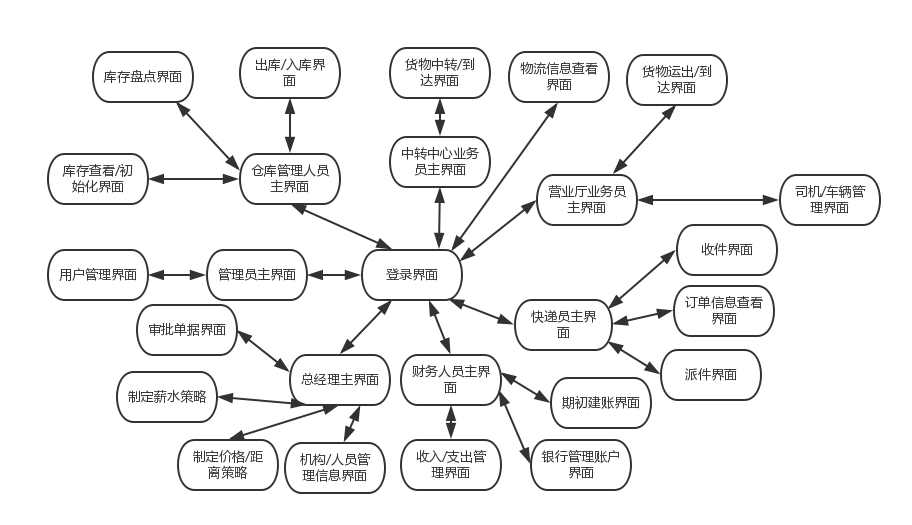


图17 用户界面跳转

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。用户界面类如图18所示。

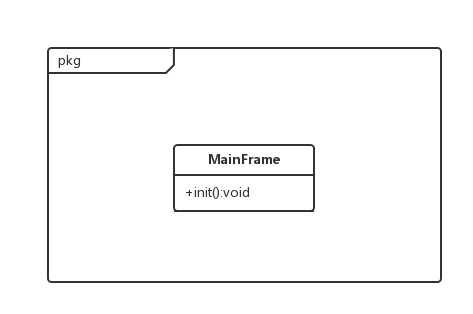


图18 用户界面类

### 5.2.1用户界面层模块的职责

如表5所示为用户界面层模块的职责。

**表5 用户界面层模块的职责**

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

### 5.2.2用户界面层模块的接口规范

用户界面层模块的接口规范如表6所示。

**表6 用户界面层模块的接口规范**

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

用户界面层需要的服务接口如表7所示。

**表7 用户界面层模块需要的服务接口**

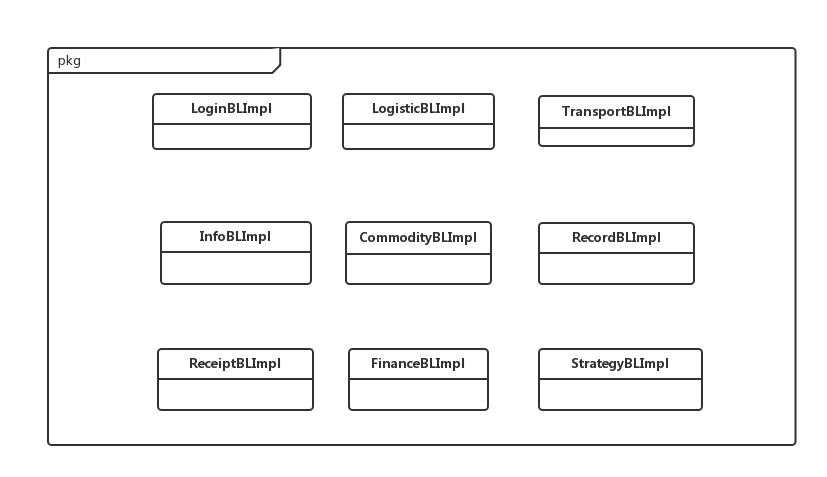
|  |  |
| --- | --- |
| **服务名** | **服务** |
| businesslogicservice.LoginBLService | 登录界面的业务逻辑接口 |
| businesslogicservice.\*BLService | 每个界面都有一个相应的业务逻辑接口 |

### 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，LoginBLImpl对象负责处理登录界面的业务逻辑；InfoBLImpl对象负责处理用户账户管理界面的业务逻辑以及LoginBLImpl对象验证密码时请求对应账户密码信息的业务逻辑。业务逻辑层的设计如图20所示。

图20 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 | 负责实现登录界面所需要的服务 |
| LogisticBL | 负责实现查询物流信息界面所需要的服务 |
| TransportBL | 负责实现快递员、营业厅业务员、中转中心业务员在快件收、派、转运过程中对应于系统操作界面的服务，并对LogisticBL模块提供物流信息更新的服务，对ReceiptBL模块提供单据状态改变时向参与者发出通知服务 |
| InfoBL | 负责实现车辆、司机、人员、机构、用户账户、银行账户管理界面所需要的服务，并对LoginBL模块提供验证登录密码的服务，对RecordBL提供信息变更时发出通知的服务，对FinanceBL模块提供获取人员信息用于计算工资的服务 |
| CommodityBL | 负责实现快件入库、出库、初始化仓库信息、库存查看、库存盘点界面所需要的服务，并对ReceiptBL模块提供单据状态改变时向参与者发出通知服务 |
| RecordBL | 负责实现查询系统日志界面所需要的服务 |
| ReceiptBL | 负责实现审批单据界面所需要的服务，并对TransportBL模块、CommodityBL模块、FinanceBL模块提供新建相应单据的服务，尤其对FinanceBL模块中查询报表功能提供查询收款单和入款单的服务 |
| FinanceBL | 负责实现收入管理、成本管理、期初建账、查询报表界面所需要的服务，并对ReceiptBL模块提供单据状态改变时向参与者发出通知服务 |
| StrategyBL | 负责实现制定策略界面所需要的服务，并对FinanceBL模块提供人员工资计算相关服务，对RecordBL模块提供策略变更时发出通知的服务，对TransportBL模块提供计算快递费用和运输费用的服务 |
| UtilityBL | 负责业务逻辑层所有模块的初始化和公用操作 |

### 5.3.2业务逻辑层模块的接口规范

**表9 LoginBL模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| LoginBLImpl.getPermission | 语法 | Public Job getPermission(String id,String password); |
| 前置条件 | Id经验证存在 |
| 后置条件 | 查找是否存在相应的用户，根据返回权限跳转到相应操作界面 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| InfoBLImpl.verifyPassword | 验证用户id和密码获得权限 | |

**表10 LogisticBL模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| LogisticBLImpl.getLogistic | 语法 | Public LogisticVO getLogistic(String orderID); |
| 前置条件 | 已输入订单号 |
| 后置条件 | 获得物流信息用于显示，如订单号不存在则显示订单不存在的信息 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| LogisticDataService.update | 更新单一持久化对象 | |
| LogisticDataService.read | 读取单一持久化对象 | |
| TransportBLImpl.register | 向TransportBLImpl注册以获得转运更新 | |

**表11 TransportBL模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| TransportBLImpl.verify | 语法 | Public boolean verify(ReceiptVO vo); |
| 前置条件 | 已输入单据 |
| 后置条件 | 检查输入内容的正确性以决定页面是否跳转 |
| TransportBLImpl.submit | 语法 | Public void submit(ReceiptVO vo); |
| 前置条件 | 输入内容经验证无误 |
| 后置条件 | 单据提交等待审批 |
| TransportBLImpl.register | 语法 | Public void register(BLImpl listener); |
| 前置条件 | 无 |
| 后置条件 | 转运更新时通知改变物流信息 |
| TransportBLImpl.cal | 语法 | Public void cal(ReceiptVO vo); |
| 前置条件 | 输入单据需要获得运费 |
| 后置条件 | 无 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| ReceiptBLImpl.register | 向ReceiptBLImpl注册以获得审批结果 | |
| ReceiptBLImpl.createReceipt | 将生成运费的单据返回给Receipt包 | |
| StrategyBLImpl.calExpressFee | 计算快递费 | |
| StrategyBLImpl.calCarriage | 计算运费 | |

**表12 InfoBL模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| InfoBLImpl.getInfoList | 语法 | public ArrayList<InfoVO> getInfoList(InfoType type) |
| 前置条件 | 无 |
| 后置条件 | 在系统中展示司机、车辆、人员、机构、用户账户或银行账户的信息列表 |
| InfoBLImpl.addInfo | 语法 | public boolean addInfo(InfoVO infoItem) |
| 前置条件 | 已经打开相关信息列表 |
| 后置条件 | 添加成功则返回true并更新相关持久化数据并通知更新系统日志，添加失败则返回false |
| InfoBLImpl.deleteInfo | 语法 | public void deleteInfo(InfoVO infoItem) |
| 前置条件 | 已经打开相关信息列表 |
| 后置条件 | 更新相关持久化数据并通知更新系统日志 |
| InfoBLImpl.modifyInfo | 语法 | public boolean modifyInfo(InfoType type, String id, InfoVO infoItem) |
| 前置条件 | 已经打开相关信息列表 |
| 后置条件 | 修改成功则返回true并更新持久化数据并通知更新系统日志，失败则返回false |
| InfoBLImpl.verifyPassword | 语法 | public Job verifyPassword(String id, String password) |
| 前置条件 | 已输入用户名和密码 |
| 后置条件 | 根据验证结果返回对应职位，密码不正确则返回“notfound”这个职位 |
| InfoBLImpl.register | 语法 | public void register(BLImpl listener); |
| 前置条件 | 无 |
| 后置条件 | 当信息发生改变时通知系统日志模块记录 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| InfoDataService.getList(InfoType type) | 得到type类型信息（如司机信息、机构信息等）的持久化对象列表 | |
| InfoDataService.addItem(InfoPO item) | 向数据库中添加一个持久化对象 | |
| InfoDataService.deleteItem(InfoType type, String id) | 从数据库中删除一个持久化对象 | |
| InfoDataService.update(String id, InfoPO item) | 更新单一持久化对象 | |
| StrategyBLImpl.setSalary() | 得到员工基本工资 | |

**表13 CommodityBL模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| CommodityBLImpl.submit | 语法 | public void submit(ReceiptVO vo) |
| 前置条件 | 启动一个库存入库（库存出库）回合 |
| 后置条件 | 在一个库存入库（库存出库）回合中，提交库存入库单（库存出库单） |
| CommodityBLImpl.getList | 语法 | public ArrayList<CommodityVO> getList(String TransferNum) |
| 前置条件 | 无 |
| 后置条件 | 返回指定中转中心当天的库存快照 |
| CommodityBLImpl.getListByTime | 语法 | public ArrayList<CommodityVO> getList(String TransferNum, Time fromTime, Time toTime) |
| 前置条件 | 无 |
| 后置条件 | 返回指定中转中心指定时间段内的出/入库数量，存储位置，库存数量合计数据 |
| CommodityBLImpl.getTotal | 语法 | public ArrayList<CommodityVO> getTotal() |
| 前置条件 | 无 |
| 后置条件 | 返回当前所有中转中心的库存信息 |
| CommodityBLImpl.renew | 语法 | Public void renew() |
| 前置条件 | 无 |
| 后置条件 | 初始化库存信息 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| ReceiptBLImpl.register(BLImpl) | 向receipt模块注册 | |
| ReceiptBLImpl.createReceipt(ReceiptVO item) | 在receipt模块插入单一持久化对象 | |
| ReceiptBLImpl.getListByTime(Time fromTime, Time toTime, ReceiptType type) | 得到对应时间段内对应type的单据列表 | |
| CommodityDataService.check(String TransferNum) | 获取指定中转中心仓库的库存信息 | |
| CommodityDataService.getAll() | 获取当前所有中转中心仓库的库存信息 | |
| CommodityDataService.renew() | 初始化库存数据块 | |
| CommodityDataService.update(CommodityPO) | 在库存数据库中更新库存信息 | |

**表14 RecordBL模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| RecordBLImpl.lookup | 语法 | public RecordVO lookup() |
| 前置条件 | 经理或财务人员请求查看系统日志 |
| 后置条件 | 显示系统主要操作日志 |
| RecordBLImpl.add | 语法 | Public void add(RecordVO vo) |
| 前置条件 | 系统有主要操作发生 |
| 后置条件 | 更新系统日志 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| RecordDataService.getRecord() | 提供系统日志 | |
| RecordDataService.record(RecordPO po) | 写入系统日志 | |
| InfoBLImpl.register() | 向info模块注册以获得其变动 | |
| StrategyBLImpl.register() | 向strategy模块注册以获得其变动 | |

**表15 ReceiptBL模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| ReceiptBLImpl.getListByState | 语法 | public ArrayList<ReceiptVO> getListByState(ReceiptState state) |
| 前置条件 | 无 |
| 后置条件 | 返回对应state的单据列表 |
| ReceiptBLImpl.getListByTime | 语法 | public ArrayList<ReceiptVO> getListByTime(myTime fromTime, myTime toTime, ReceiptType type) |
| 前置条件 | 无 |
| 后置条件 | 返回对应时间段内对应type的单据列表 |
| ReceiptBLImpl.approve | 语法 | public void approve(ArrayList<ReceiptVO> itemList) |
| 前置条件 | 已经打开待审批或未通过单据列表 |
| 后置条件 | 更新对应单据持久化数据，并且通知监听者 |
| ReceiptBLImpl.disapprove | 语法 | public void disapprove(ArrayList<ReceiptVO> itemList) |
| 前置条件 | 已经打开待审批单据列表 |
| 后置条件 | 更新对应单据持久化数据，并且通知监听者 |
| ReceiptBLImpl.createReceipt | 语法 | public void createReceipt(ReceiptVO item) |
| 前置条件 | 单据中费用信息已经计算完成 |
| 后置条件 | 新增一个单据，并更新单据持久化数据 |
| ReceiptBLImpl.register | 语法 | public void register(BLImpl listener); |
| 前置条件 | 无 |
| 后置条件 | 当单据状态发生改变时向监听者发出通知 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| ReceiptDataService.getList(ReceiptType type, ReceiptState state, Time time) | 得到数据库中对应type, state, time的单据列表 | |
| ReceiptDataService.addItem(ReceiptPO item) | 向单据数据库中添加一个持久化对象 | |
| ReceiptDataService.update(ReceiptPO item) | 更新单一持久化对象 | |
| ReceiptDataService.deleteAll(ReceiptType type) | 清空数据库中type类型的所有单据 | |

**表16 FinanceBL模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| FinanceBLImpl.submit | 语法 | Public void submit(ReceiptVO vo) |
| 前置条件 | 启动一个付款（收款）回合 |
| 后置条件 | 在一个付款（收款）回合中，新建付款（收款）单 |
| FinanceBLImpl.getCredit | 语法 | Public FinanceVO getCredit(int year) |
| 前置条件 | 无 |
| 后置条件 | 返回指定年份存储期初建账时保存的信息 |
| FinanceBLImpl.calFee | 语法 | Public double calFee(Type feeType, ReceiptVO vo) |
| 前置条件 | 无 |
| 后置条件 | 返回计算的收款（付款）值 |
| FinanceBLImpl.seeRecord | 语法 | Public ArrayList<ReceiptVO> seeRecord(Time fromTime, Time toTime) |
| 前置条件 | 无 |
| 后置条件 | 返回该时间段内的所有入库单和出库单数据列表 |
| FinanceBLImpl.checkStore | 语法 | Public ArrayList<ReceiptVO> checkStore(Time fromTIme , Time toTime, String StoreNum) |
| 前置条件 | 无 |
| 后置条件 | 返回指定营业厅指定日期的收款单数据列表 |
| FinanceBLImpl.addUp | 语法 | Public String addUp() |
| 前置条件 | 已经获取收款单数据列表 |
| 后置条件 | 返回所有收款单金额合计 |
| FinanceBLImpl.checkProfit | 语法 | Public ProfitVO checkProfit() |
| 前置条件 | 无 |
| 后置条件 | 返回截止当前日期的总收入、总支出、总利润 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| CommodityBLImpl.getTotal() | 提供当前所有中转中心仓库的库存信息 | |
| ReceiptBLImpl.register(BLImpl) | 向receipt模块注册 | |
| ReceiptBLImpl.createReceipt(ReceiptVO item) | 在receipt模块插入单一持久化对象 | |
| ReceiptBLImpl.getListByTime(Time fromTime, Time toTime, ReceiptType type) | 得到对应时间段内对应type的单据列表 | |
| StrategyBLImpl.setSalary() | 得到薪水策略 | |
| StrategyBLImpl.register(BLImpl Listener) | 向strategy模块注册 | |
| FinanceDataService.add(FinancePO) | 向账目数据库中添加一个持久化对象 | |
| FinanceDataService.find(int year) | 返回账目数据库中的一个持久化对象 | |
| FinanceDataService.updateIn(int income) | 更新账目数据库中的总收入数据 | |
| FincanceDataService.renewIn() | 初始化账目数据库中的总收入数据 | |
| FinanceDataService.getIn() | 返回账目数据库中的总收入数据 | |
| FinanceDataService.updateOut(int outcome) | 更新账目数据库中的总支出数据 | |
| FincanceDataService.renewOut() | 初始化账目数据库中的总支出数据 | |
| FinanceDataService.getOut() | 返回账目数据库中的总支出数据 | |
| InfoBLImpl.getInfoList() | 返回司机、车辆、人员、机构、用户账户或银行账户的信息列表 | |

**表17 StrategyBL模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| StrategyBLImpl.calExpressFee | 语法 | public double calExpressFee(ReceiptVO vo) |
| 前置条件 | 获得寄件单参数 |
| 后置条件 | 返回快递费 |
| StrategyBLImpl.calCarriage | 语法 | public double calCarriage(ReceiptVO vo) |
| 前置条件 | 获得中转单或装车单参数 |
| 后置条件 | 返回运费 |
| StrategyBLImpl.setSalary | 语法 | public SalaryVO setSalary() |
| 前置条件 | 财务人员请求获得薪水策略 |
| 后置条件 | 返回薪水策略 |
| StrategyBLImpl.register | 语法 | public void register(BLImpl Listener) |
| 前置条件 | 无 |
| 后置条件 | 当策略发生变动向监听者发出通知 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| StrategyDataService.getExpressFee() | 提供快递费用策略 | |
| StrategyDataService.updateExpressFeeStrategy(ExpressFeePO efpo) | 更新快递费用策略 | |
| StrategyDataService.getCarriageFee() | 提供运费策略 | |
| StrategyDataService.updateCarriageFeeStrategy(CarriageFeePO cfpo) | 更新运费策略 | |
| StrategyDataService.getSalary() | 提供薪水的策略 | |
| StrategyDataService.updateSalaryStrategy(SalaryPO po) | 更新薪水策略 | |
| TransportBLImpl.cal() | 获得单据信息用于计算 | |

## 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。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。

图21到图27展示了各个数据层模块的描述。

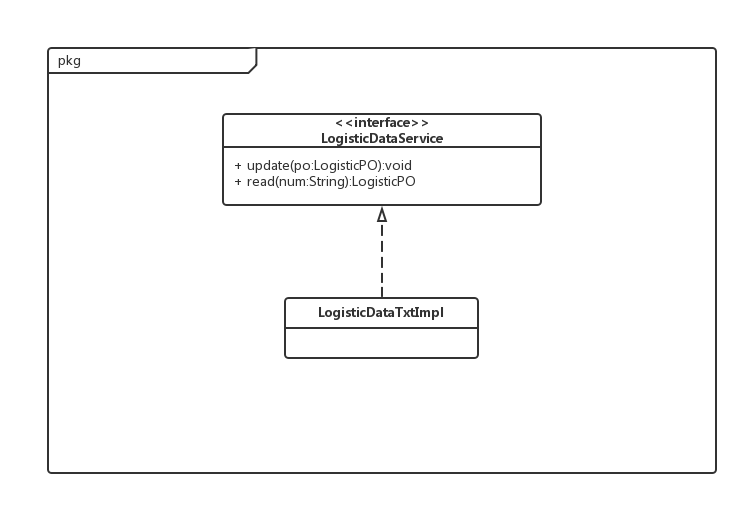


图21 物流数据层模块的描述

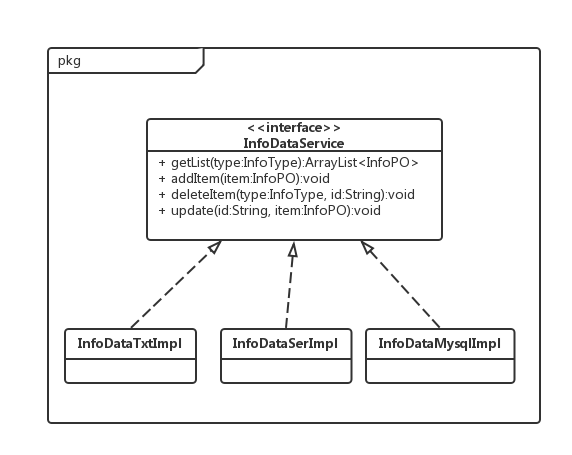


图22 信息管理数据层模块的描述

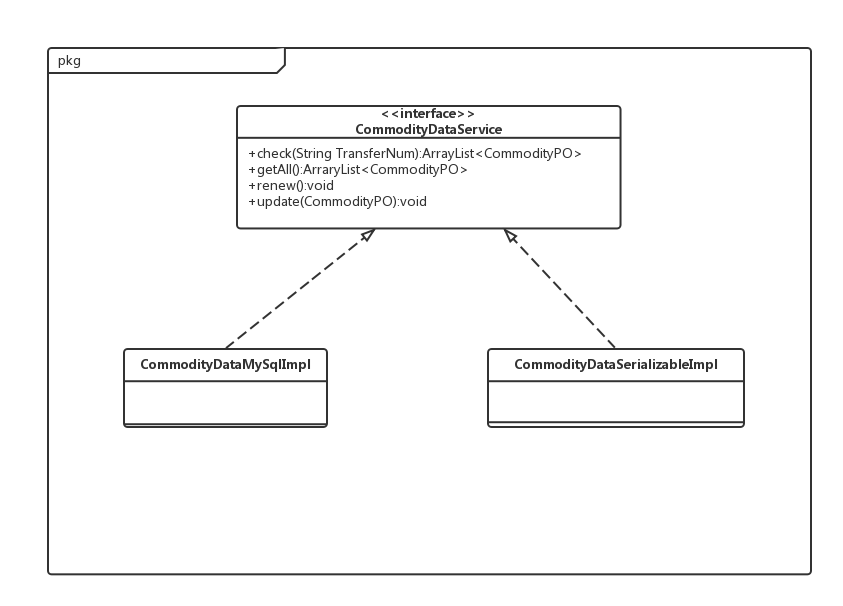


图23 库存数据层模块的描述

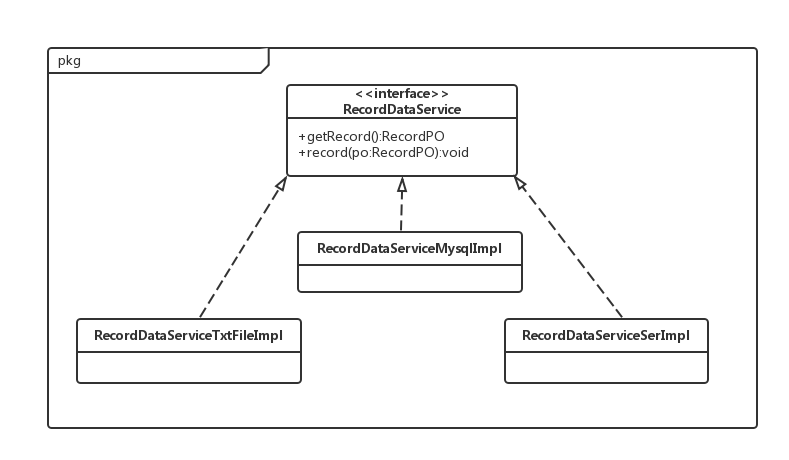


图24 系统日志数据层模块的描述

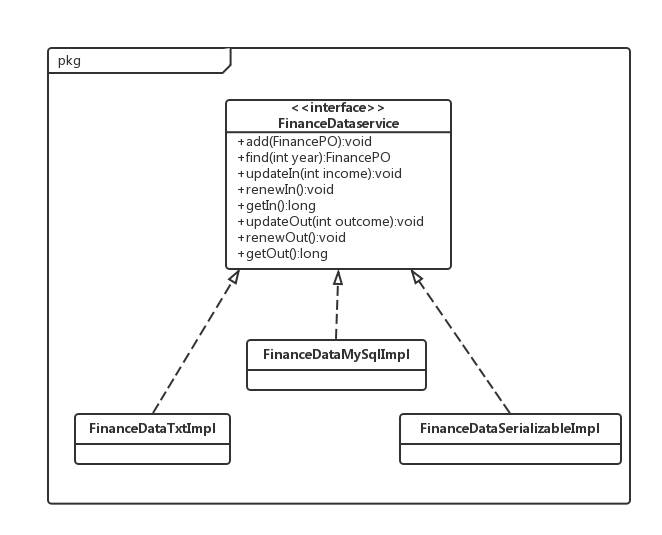


图26 财务数据层模块的描述

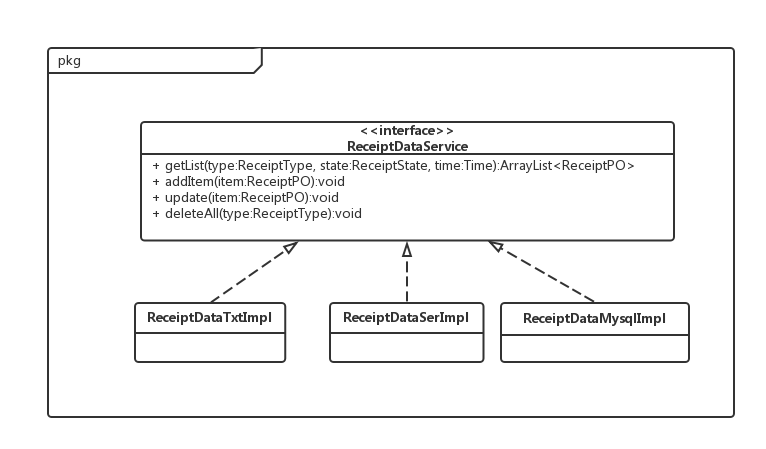


图26 单据数据层模块的描述

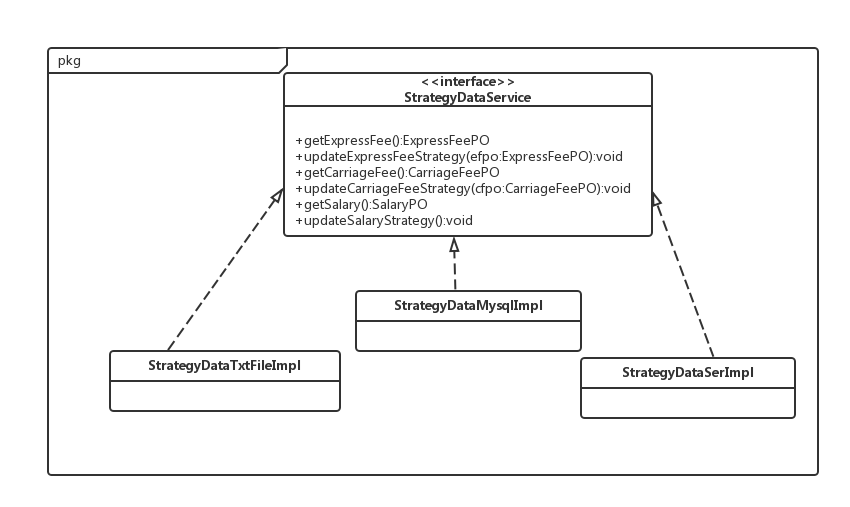


图27 策略数据层模块的描述

### 5.4.1数据层模块的职责

由于各个功能模块对应数据层模块的职责基本相同，故这里只用一张表总结描述，如表18所示。

**表18 数据层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| \*DataService | 持久化数据操作的接口，提供持久化数据的增、删、改、查服务 |
| \*DataTxtImpl | 基于txt文件的持久化数据操作的接口，提供持久化数据的增、删、改、查服务 |
| \*DataSerImpl | 基于序列化文件的持久化数据操作的接口，提供持久化数据的增、删、改、查服务 |
| \*DataMysqlImpl | 基于Mysql数据库持久化数据操作的接口，提供持久化数据的增、删、改、查服务 |

### 5.4.2数据层模块的接口规范

表19到表25为各个功能数据层模块的接口规范。

**表19 Logistic数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| LogisticDataService.update | 语法 | Public void update(LogisticPO po) throws RemoteException; |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 更新一个po |
| LogisticDataService.read | 语法 | Public LogisticPO read(String num) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 读取物流信息 |
| LogisticDataService.init | 语法 | Public void init() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |

**表20 Info数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| InfoDataService.getList | 语法 | public ArrayList<InfoPO> getList(InfoType type) throws RemoteExceptoin; |
| 前置条件 | 无 |
| 后置条件 | 按type返回相应的InfoPO列表 |
| InfoDataService.addItem | 语法 | public void addItem(InfoPO item) throws RemoteException; |
| 前置条件 | 相同id的po在列表中不存在 |
| 后置条件 | 在持久化数据中增加一个po条目 |
| InfoDataService.deleteItem | 语法 | public void deleteItem(InfoType type, String id) throws RemoteException |
| 前置条件 | 持久化数据列表中存在一个相同type和id的po |
| 后置条件 | 删除一个po |
| InfoDataService.update | 语法 | public void update(String id, InfoPO item) throws RemoteException |
| 前置条件 | 在持久化数据列表中存在相同id的po |
| 后置条件 | 更新一个po |

**表21 Commodity数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| CommodityDataService.check | 语法 | Public ArrayList<CommodityPO> check(String TransferNum) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回指定中转中心仓库的库存信息 |
| CommodityDataService.getAll | 语法 | Public ArrayList<CommodityPO> getAll() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回所有中转中心的库存信息 |
| CommodityDataServicec.renew | 语法 | Public void renew()throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化库存数据块 |
| CommodityDataService.update | 语法 | Public void update(CommodityPO) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 库存数据块更新库存信息 |

**表22 Record数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| RecordDataService.getRecord | 语法 | public RecordPO getRecord() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回系统日志 |
| RecordDataService.insertRecord | 语法 | public void insertRecord(RecordPO po) throws RemoteException |
| 前置条件 | 系统有主要操作发生 |
| 后置条件 | 在系统日志中记录操作 |

**表23 Receipt数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| ReceiptDataService.getList | 语法 | public ArrayList<ReceiptPO> getList(ReceiptType type, ReceiptState state, Time time) throws RemoteExceptoin; |
| 前置条件 | 无 |
| 后置条件 | 按对应type,state,time返回相应ReceiptPO列表 |
| ReceiptDataService.addItem | 语法 | public void addItem(ReceiptPO item) throws RemoteException; |
| 前置条件 | 相同id的po在列表中不存在 |
| 后置条件 | 在持久化数据中增加一个po条目 |
| ReceiptDataService.update | 语法 | public void update(ReceiptPO item) throws RemoteException |
| 前置条件 | 持久化数据列表中存在一个相同id的po |
| 后置条件 | 更新一个po |
| ReceiptDataService.deleteAll | 语法 | public void deleteAll(ReceiptType type) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 清空数据库中type类型的所有单据 |

**表24** **Finance数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| FinanceDataService.add | 语法 | Public void add (FinancePO) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 在账目数据库中增加一个持久化对象 |
| FinanceDataService.find | 语法 | Public FinancePO find(int year) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 根据年份信息返回账目数据库中的一个持久化对象 |
| FinanceDataService.updateIn(int income) | 语法 | Public void updateIn(int income) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 更新账目数据库中的总收入数据 |
| FinanceDataService.renewIn() | 语法 | Public void renewIn()throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化账目数据库中的总收入数据 |
| FinanceDataService.getIn() | 语法 | Public long getIn() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回账目数据库中的总收入数据 |
| FinanceDataService.updateOut(int outcome) | 语法 | Public void updateOut(int outcome) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 更新账目数据库中的总支出数据 |
| FinanceDataService.renewOut() | 语法 | Public void renewOut() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化账目数据库中的总支出数据 |
| FinanceDataService.getOut() | 语法 | Public long getOut() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回账目数据库中的总支出数据 |

**表25 Strategy数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| StrategyDataService.getExpres  sFee | 语法 | public ExpressFeePO getExpressFee() throws RemoteException |
| 前置条件 | 请求获得计算快递费用的策略 |
| 后置条件 | 返回计算快递费用的策略 |
| StrategyDataService.update  ExpressFeeStrategy | 语法 | public void updateExpressFeeStrategy  (ExpressFeePO efpo) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 更新计算快递费用的策略 |
| StrategyDataService.getCarriage  Fee | 语法 | Public CarriagePO getCarriageFee() throws RemoteException |
| 前置条件 | 请求获得计算运费的策略 |
| 后置条件 | 返回计算运费的策略 |
| StrategyDataService.updateCarr  iageFee | 语法 | public void updateCarriageFeeStrategy  (CarriageFeePO cfpo) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 更新计算运费的策略 |
| StrategyDataService.getSalary | 语法 | public SalaryPO getSalary() throws RemoteException |
| 前置条件 | 请求获得薪水策略 |
| 后置条件 | 返回薪水策略 |
| StrategyDataService.update  SalaryStrategy | 语法 | public void updateSalaryStrategy(SalaryPO po) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 更新薪水策略 |

# 6信息视角

## 6.1数据持久化对象

系统的PO就是对应的需要持久化的类，所有的po类的介绍如下所示。

* LogisticPO类包含托运订单号、到达时间、货运状态（收件、到达寄件人营业厅、到达寄件人中转中心、到达收件人中转中心、到达收件人营业厅、派件中)。
* InfoPO类是车辆、司机、人员、机构、用户账户、银行账户的持久化信息父类，其对应的子类包括TruckPO、DriverPO、StaffPO、AgencyPO、UserAccountPO、BankAccountPO。

TruckPO类包含车辆代号、发动机号、车牌号、底盘号、购买时间、服役时间；DriverPO类包含司机编号、姓名、出生日期、身份证号、手机号、性别、行驶证期限；StaffPO类包含工号、姓名、性别、出生年月、所属部门、基本工资、快递员派件数、司机运货次数；AgencyPO类包含机构名称、机构类别（中转中心、营业厅）、机构编号、位置、占地面积、土地租金（元／平方米／月）；UserAccountPO类包含工号、姓名、所属部门、用户名、密码；BankAccountPO类包含账户名。

* CommodityPO类包含入库数量、金额、存储位置、库存变化数量。
* RecordPO类包含操作名称、发生时间。
* ReceiptPO类是寄件单、派件单、装车单、中转单、营业厅到达单、中转中心到达单、入库单、出库单、收款单、付款单的持久化信息父类，其对应的子类包括SendReceiptPO、DespatchReceiptPO、EntruckReceiptPO、TransferReceiptPO、StoreArrivalReceiptPO、HubArrivalReceiptPO、DepotInReceiptPO、DepotOutReceiptPO、ChargeReceiptPO、PayReceiptPO。

SendReceiptPO类包含寄件人姓名、住址、单位、手机，收件人姓名、住址、单位、手机；DespatchReceiptPO类包含到达日期、托运订单条形码号、派送员；EntruckReceiptPO类包含装车日期、营业厅／中转中心汽运编号、到达地、车辆代号、本次装箱所有订单条形码号、运费；

TransferReceiptPO类包含中转方式、中转日期、中转单编号、（飞机／铁路）班次或者汽车的车牌号、出发地、到达地、货柜号、本次装箱所有货物的单号；StoreArrivalReceiptPO类包含到达日期、（中转单编号）、出发地、货物到达状态（损坏、完整、丢失）；

HubArrivalReceiptPO类包含到达日期、中转单编号、出发地、货物到达状态（损坏、完整、丢失）；

DepotInReceiptPO类包含快递编号、目的地、入库日期、区号（汽运区区号、铁运区区号、航运区区号）、排号、架号、位号；

DepotOutReceiptPO类包含快递编号、出库日期、目的地、装运形式（火车、飞机、汽车）、中转单编号或者汽运编号；

ChargeReceiptPO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;

PayReceiptPO类包含付款日期、付款金额、付款人、付款账号、条目（租金（按年收）、运费（按次计算）、人员工资（按月统计）、奖励（一次性））、备注。

* FinancePO类包含机构、人员、车辆、库存、银行账户信息（名称、余额）、总收入、总支出。
* StategyPO类是价格策略、薪水策略的持久化信息的父类，其对应的子类包括和ExpressFeeVO、CarriageVO和SalaryVO。

持久化对象ReceiveReceiptPO的定义如下所示，其他po对象详见代码。

public class ReceiveReceiptPO extends ReceiptPO{

private static final long serialVersionUID = 1L;

private String receiveNum;

private String receiver;

private myTime receiveTime;

public ReceiveReceiptPO(String receiveNum,String receiver,myTime receiveTime) {

super(ReceiptType.RECEIVE);

this.setReceiveNum(receiveNum);

this.setReceiver(receiver);

this.setReceiveTime(receiveTime);

}

public String getReceiveNum() {

return receiveNum;

}

public void setReceiveNum(String receiveNum) {

this.receiveNum = receiveNum;

}

public String getReceiver() {

return receiver;

}

public void setReceiver(String receiver) {

this.receiver = receiver;

}

public myTime getReceiveTime() {

return receiveTime;

}

public void setReceiveTime(myTime receiveTime) {

this.receiveTime = receiveTime;

}

}

## 6.2 Txt持久化格式

Txt数据保持格式以Losgistic.txt为例。每行分别对应，编号后用“：”与详细内容隔开，不同类信息中间用“；”隔开，同类信息用“ ”隔开。如下所示：

0123456789:20151022 8:00；收件20151024 8:00；到达寄件人营业厅

1013456789:20151026 13:00；收件 20151026 19:00；到达寄件人营业厅 20151026 23:00；到达寄件人中转中心

## 6.3数据库表

数据库表中包含Info表，Commodity表，Receipt表。