快递物流系统

软件详细设计描述文档

软件学院

B401

目录

[1.引言 2](#_Toc435388649)

[1.1编制目的 2](#_Toc435388650)

[1.2词汇表 2](#_Toc435388651)

[1.3参考资料 2](#_Toc435388652)

[2.产品概述 2](#_Toc435388653)

[2.1商品前景 2](#_Toc435388654)

[2.1.1背景与机遇 2](#_Toc435388655)

[2.1.2业务需求 3](#_Toc435388656)

[2.2商品功能 3](#_Toc435388657)

[2.3用户特征 3](#_Toc435388658)

[2.4约束 4](#_Toc435388659)

[2.5假设和依赖 4](#_Toc435388660)

[3.体系结构设计概述 4](#_Toc435388661)

[4.结构视角 5](#_Toc435388662)

[4.1用户界面层的分解 5](#_Toc435388663)

[4.1.1登陆界面 11](#_Toc435388664)

[4.1.2游客界面 11](#_Toc435388665)

[4.1.3快递员界面 11](#_Toc435388666)

[4.1.4营业厅业务员界面 12](#_Toc435388667)

[4.1.5中转中心业务员界面 14](#_Toc435388668)

[4.1.6中转中心仓库管理员界面 15](#_Toc435388669)

[4.1.7总经理界面 16](#_Toc435388670)

[4.1.8财务人员界面 20](#_Toc435388671)

[4.1.9管理员界面 23](#_Toc435388672)

[4.2业务逻辑层的分解 23](#_Toc435388673)

[4.2.1 Userbl模块 24](#_Toc435388674)

[4.2.2 Commoditybl模块 25](#_Toc435388675)

[4.2.3 Billsbl模块 27](#_Toc435388676)

[4.2.4 Staffbl 模块 34](#_Toc435388677)

[4.2.5 Organizationbl模块 37](#_Toc435388678)

[4.2.6 Managebl模块 39](#_Toc435388679)

[4.3数据层的分解 45](#_Toc435388680)

[4.3.1 BillsDataService模块 45](#_Toc435388681)

[4.3.2 StaffDataService模块 46](#_Toc435388682)

[4.3.3 OrganizationDataService模块 46](#_Toc435388683)

[4.3.4 ManageDataService模块 47](#_Toc435388684)

[4.3.5 CommodityDataService模块 48](#_Toc435388685)

[4.3.6 UserDataService模块 49](#_Toc435388686)

[5.依赖视角 50](#_Toc435388687)

# 1.引言

## 1.1编制目的

本报告详细完成对快递物流系统的详细设计，达到指导后续软件构造的目的，同时实现和测试人员及用户的沟通。

本报告面向开发人员、测试人员以及最终用户而编写，是了解系统的导航。

## 1.2词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
|  |  |  |
|  |  |  |

## 1.3参考资料

* 用例文档
* 软件需求规格说明文档
* 测试用例文档
* 体系结构设计文档

# 2.产品概述

## 2.1商品前景

### 2.1.1背景与机遇

一本地民营物流企业，希望通过对城市网点建设，扩大快运服务通达的片区。随着现公司规模扩大，企业业务量、办公场所、员工数都发生增长，无法适应新的环境，不能提高工作效率和用户满意度。

物流管理系统就是为满足该民营企业新的业务发展要求而开发的，它包括一个数据集中服务器和多个客户端。数据集中服务器将所有的数据存储起来进行维护。用户通过客户端完成日常任务，客户端与数据集中服务器才是实时通信的方式完成数据交换。

### 2.1.2业务需求

BR1：在系统使用三个月后车辆调运不及时现象要减少50%。

BR2：在系统使用三个月后人员工作效率提高50%。

BR3：在系统使用三个月后运营成本降低10%。

范围：运输费用和人员工资。

度量：检查财务人员每月的付款单金额。

## 2.2商品功能

SF1：记录货物运输过程。

SF2：统筹车辆司机管理。

SF3：明确收入支出金额。

SF4：制定价格变更策略。

SF5：处理货物出库入库。

SF6：掌握员工机构变动。

SF7：允许财务统计报表。

SF8：发展企业扩展城市。

SF9：帮助经理规划路线。

## 2.3用户特征

|  |  |
| --- | --- |
| 快递员 | 每个营业厅有50名快递员，他们每天都要接收顾客的快递回到营业厅和带着派件快递前往收件人处。在收到快递时，由于快递信息繁多，且顾客对时间要求苛刻，他们必须能够根据快递路线与货运方式直接告知客户快递送达的大致日期与需要金额。在派送快递时，他们也需要将快递运输过程中的情况详细反应给顾客。 |
| 营业厅业务员 | 每个营业厅有5名业务员，他们的任务是接收送达营业厅的快递，并安排人员将它们派送到收件人处或装车运输到中转中心。由于每个营业厅配有20名司机，业务员需要对司机信息与车辆信息严格管理，避免出现车辆故障或司机驾驶证到期等情况。 |
| 中转中心业务员 | 每个中转中心有10到20名业务员，负责对快递转运的管理，能够简单使用办公信息化系统。他们的任务是接收由其他中转中心或营业厅送来的货物，并且安排货物中转到下一中转中心或送达本城市的营业厅。 |
| 中转中心仓库管理员 | 每个中转中心有1到2名仓库管理员，他们的任务是详细记录仓库内的库存情况。他们常常为仓库库存堆积发现不及时导致新运送来的货物无法及时入库而烦恼。每天早晨八点，他们需要进行一次盘点，即检查仓库内存货与记录信息是否属实。有时他们也因为库区的调整导致的信息变化而苦恼。 |
| 财务人员 | 公司有1到两名财务人员，他们的任务是管理账户、成本管理、查看经营情况表、查看成本收益表以及期初建账。他们的办公信息化能力很强，对新系统持支持态度，不希望增加现有工作量 |
| 总经理 | 公司有2名总经理，他们能够熟练使用办公信息化系统。他们负责审批单据、人员管理、机构管理。他们对新系统持支持态度。 |
| 管理员 | 公司有一个管理员，任务是管理用户账号以及管理权限。办公信息化能力较强，对新系统持支持态度。 |

## 2.4约束

CON1：系统将运行在Win 8操作系统上。

CON2：系统将使用图形界面。

CON3：开发过程中，开发者要提交软件需求规格说明文档、设计描述文档和测试报告。

CON4：在开发中，开发者要提交软件需求规格说明文档、设计描述文档和测试报告。

## 2.5假设和依赖

AE1：顾客在快递运输过程中不可提出退货。

AE2：车辆与司机信息由营业厅管理，但是在中转中心也有配备。

AE3：司机在执行完任务回到营业厅时需要向业务员报道。

AE4：营业厅和中转中心在派车时，需人工确认某辆车是否正在执行任务。

AE5：货物送出必有人签收，且不存在退货。

# 3.体系结构设计概述

快递物流系统中，选择了分层体系结构风格，将系统分为三层，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。

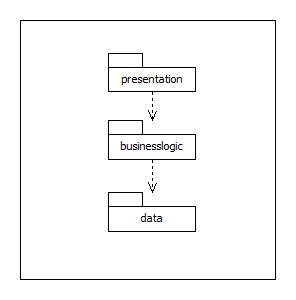


图1 参照体系结构风格的包图表达逻辑视角

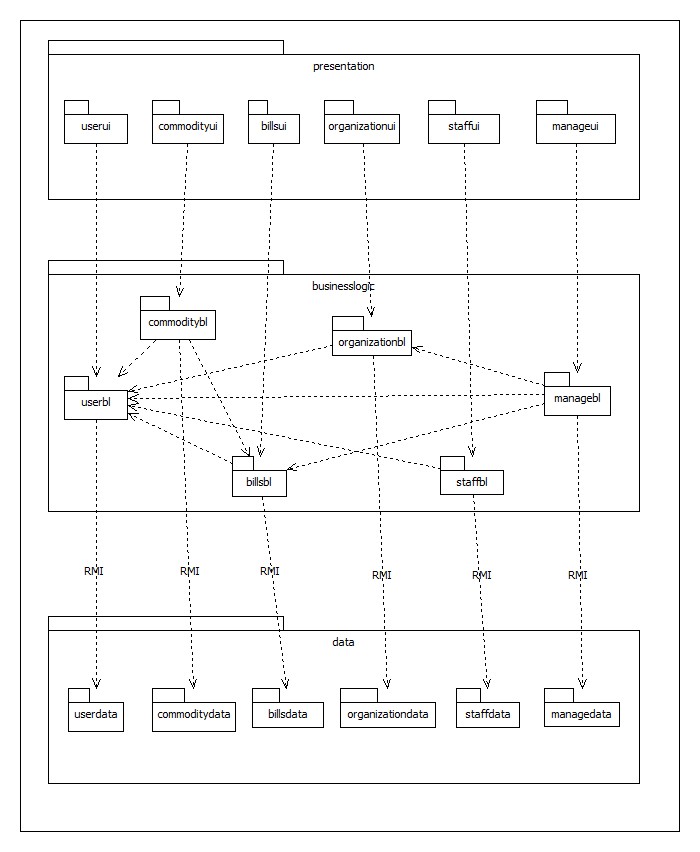
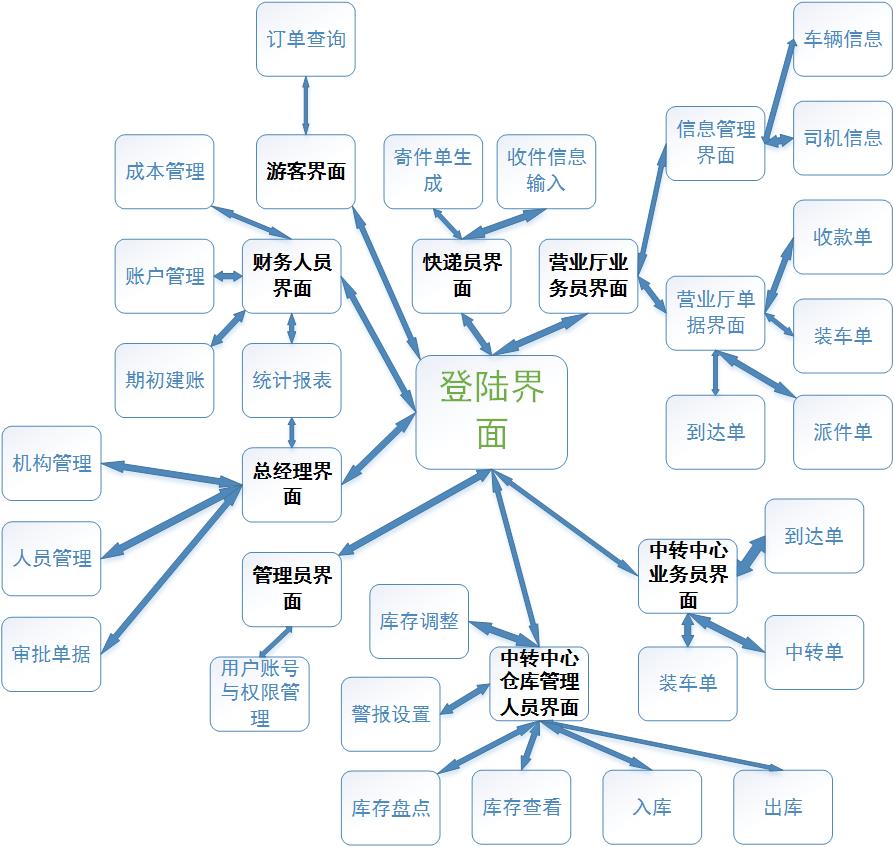


图2 软件体系结构逻辑设计方案

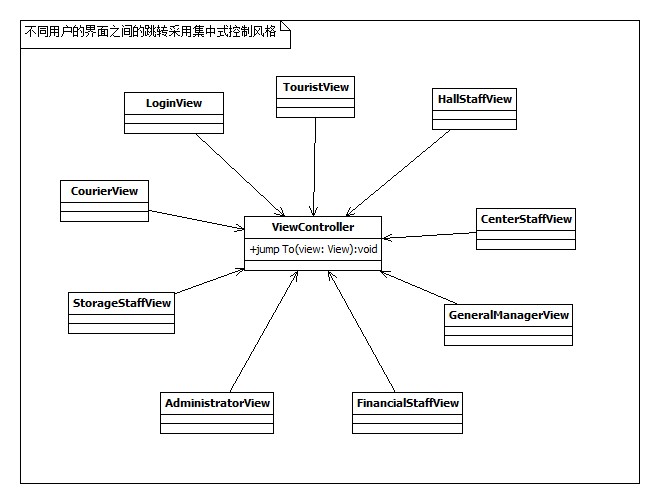
# 4.结构视角

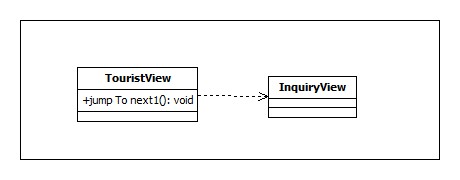
## 4.1用户界面层的分解

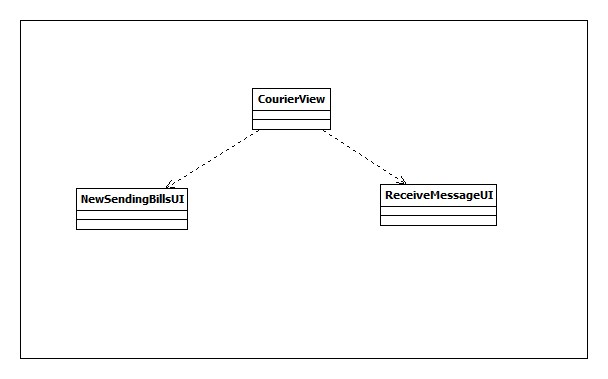
根据需求，系统界面跳转如下图所示。

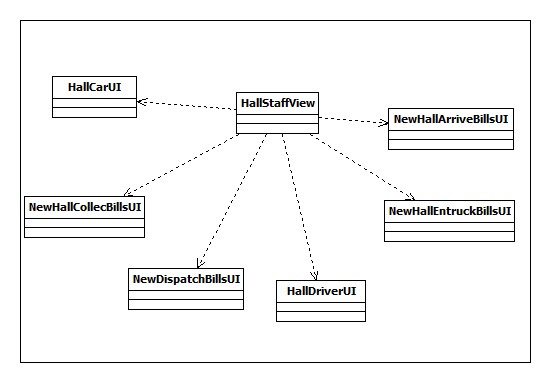


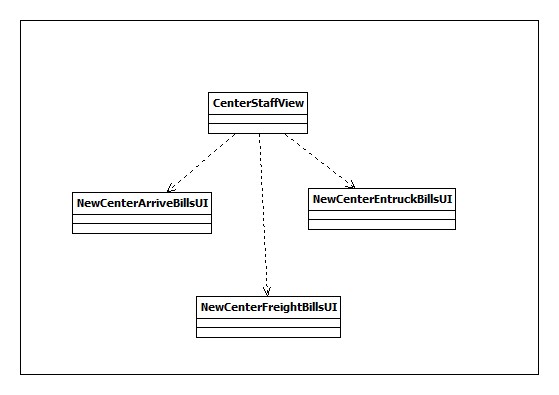
具体见下图。

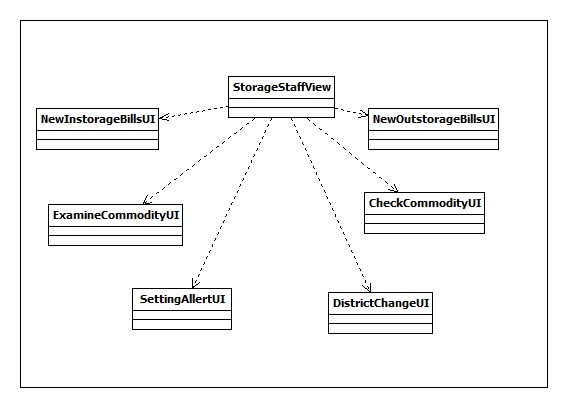


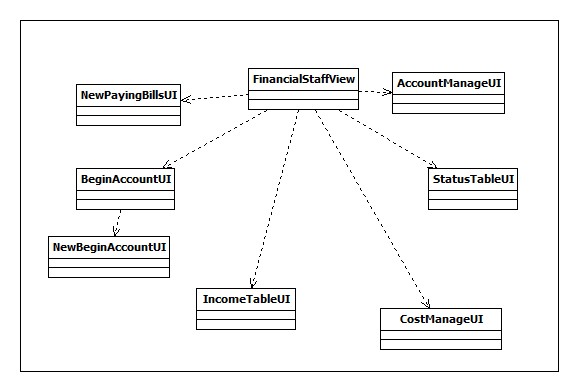


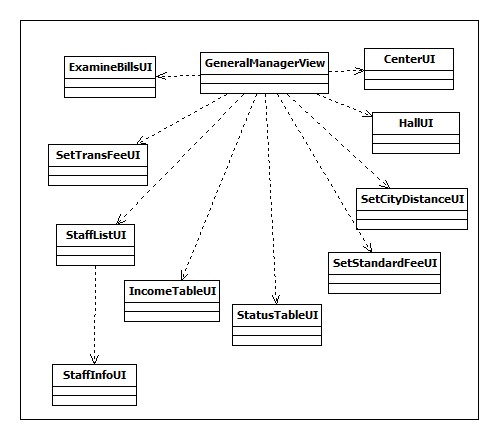


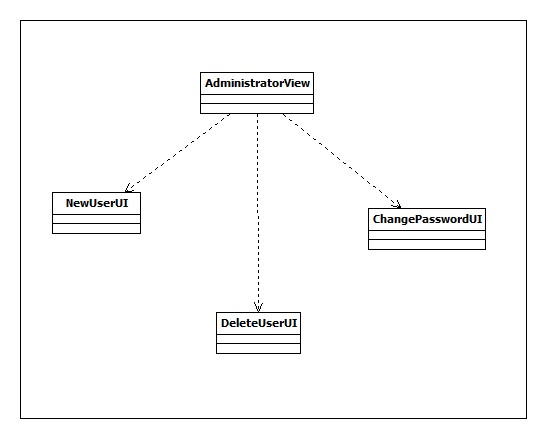












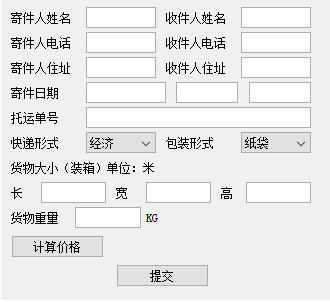
### 4.1.1登陆界面

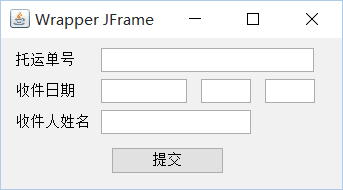


### 4.1.2游客界面

### 4.1.3快递员界面

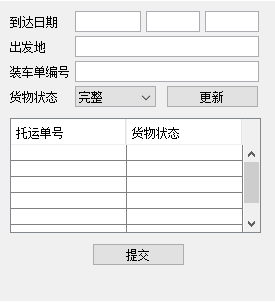
寄件单生成



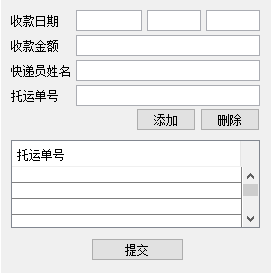
收件信息输入  


### 4.1.4营业厅业务员界面

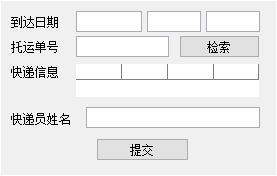
营业厅到达单生成



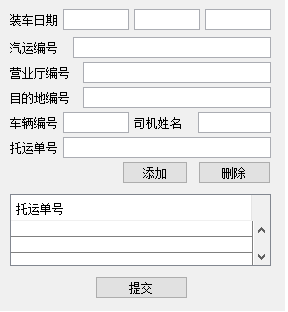
营业厅收款单生成



营业厅派件单生成



营业厅装车单生成



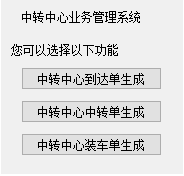
营业厅车辆信息管理界面



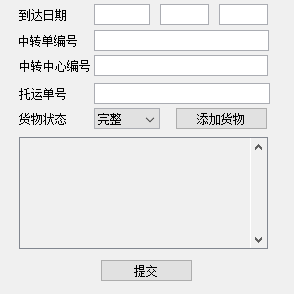
营业厅司机信息管理界面



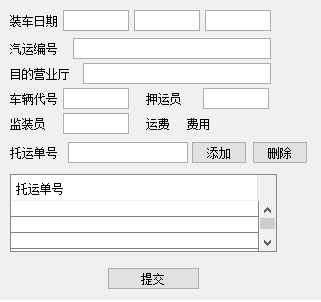
### 4.1.5中转中心业务员界面



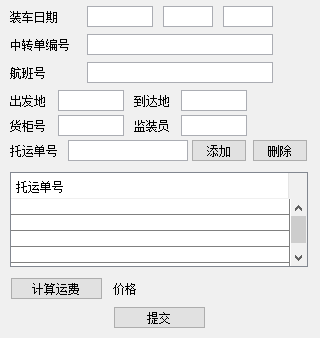
中转中心到达单生成



中转中心装车单生成

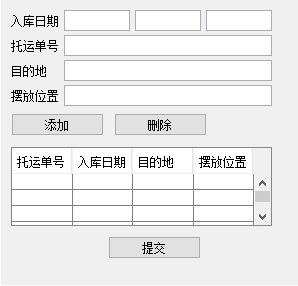


中转中心中转单生成

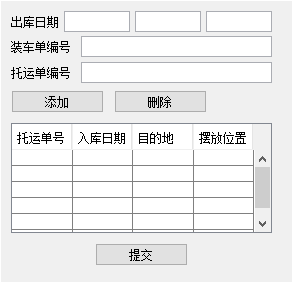


### 4.1.6中转中心仓库管理员界面

入库单生成



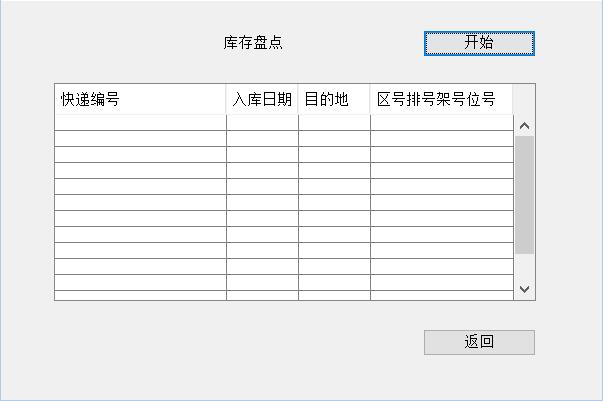
出库单生成



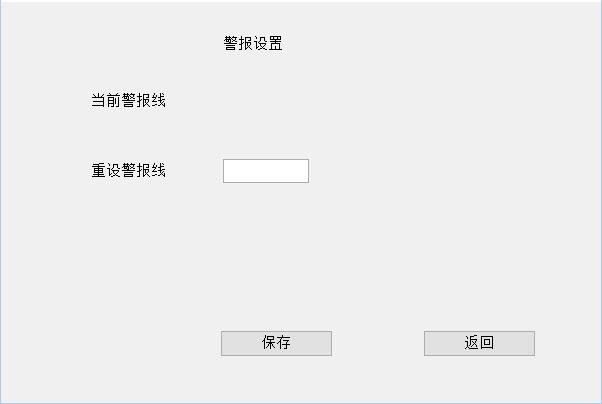
库存查看界面



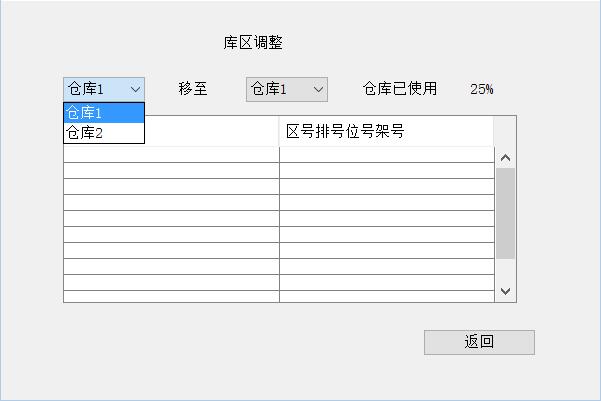
库存盘点界面



警报设置界面

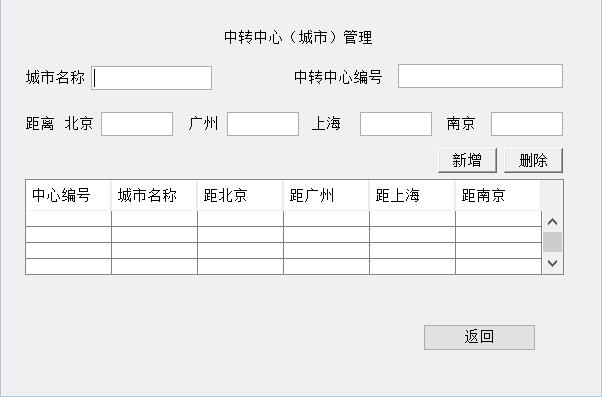


库区调整界面



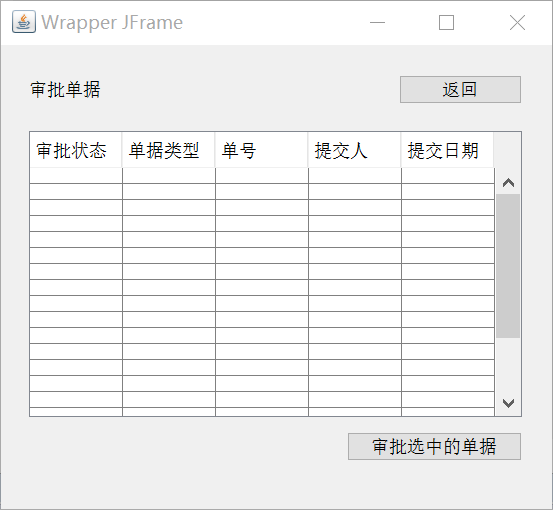
### 4.1.7总经理界面

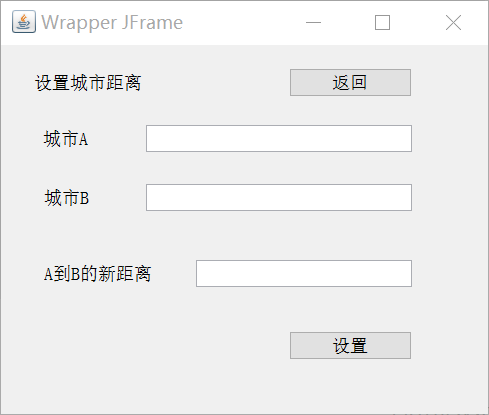
城市管理界面

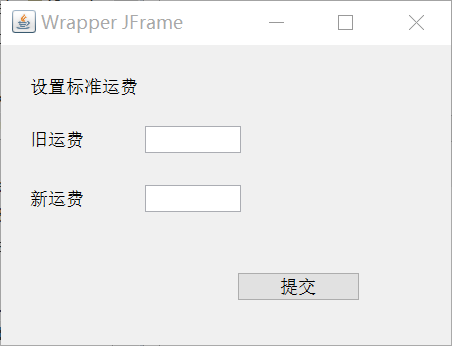


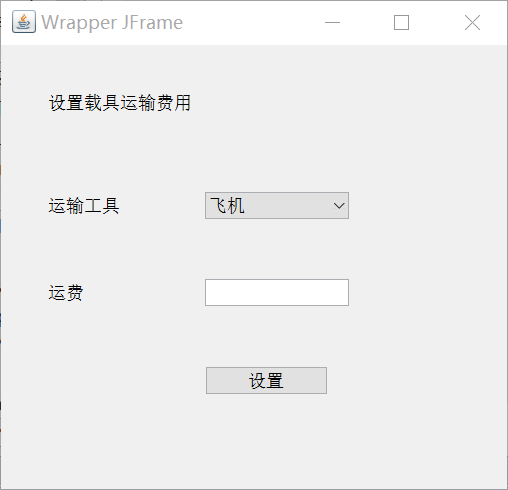
营业厅管理界面



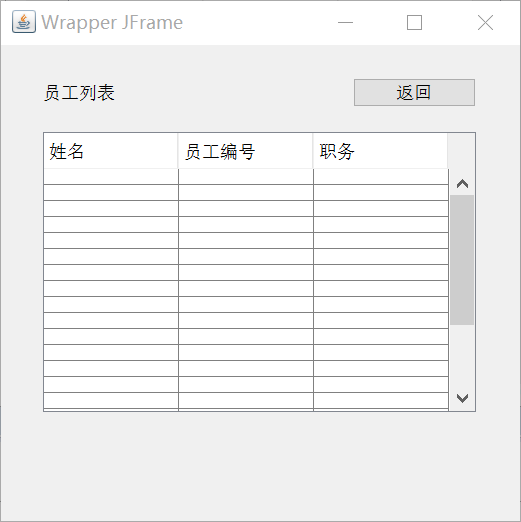






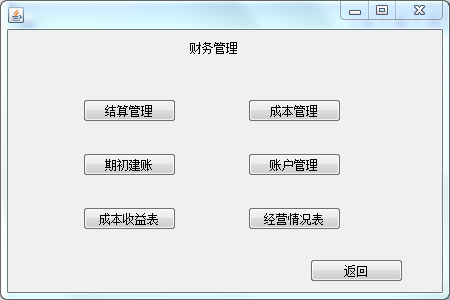




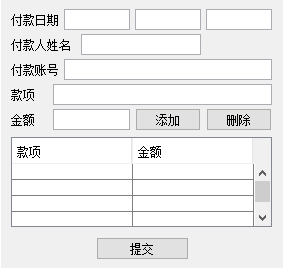


### 4.1.8财务人员界面

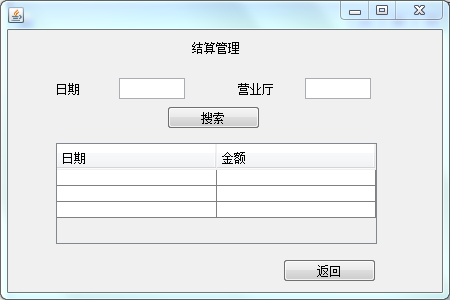
财务人员主界面



付款单生成



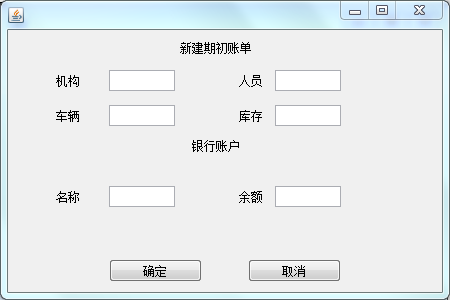
结算管理界面



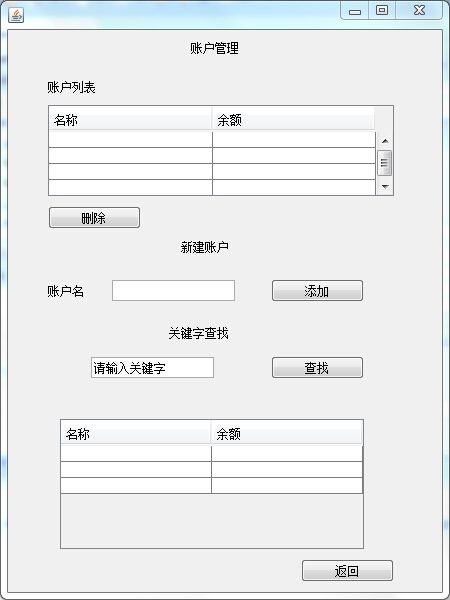
期初建账界面



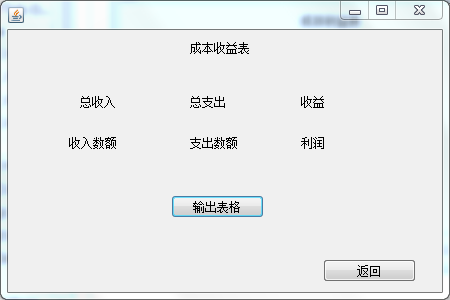
新建期初账单界面



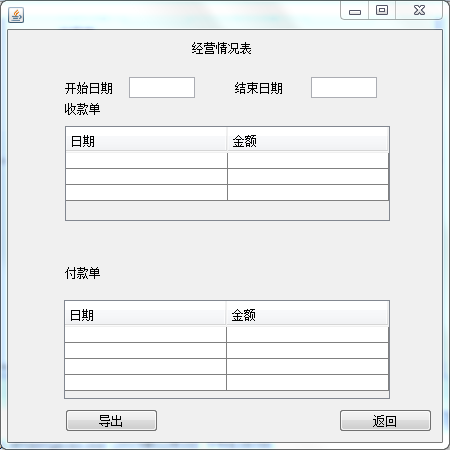
账户管理界面



成本收益表界面



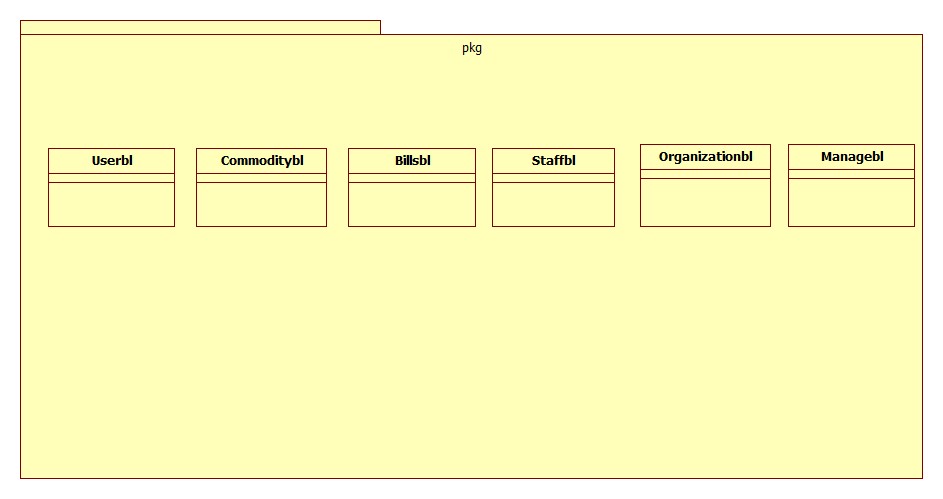
经营情况表界面



### 4.1.9管理员界面

## 4.2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下图所示。



### 4.2.1 Userbl模块

1. 模块概述

Userbl模块承担的需求参见需求规格说明文档

Userbl模块的职责及接口参见软件系统结构描述文档

1. 整体结构

根据体系结构设计，我们将系统分为展示层、业务逻辑层、数据层，每一层之间添加接口。业务逻辑层和展示层之间有若干BLService接口，业务逻辑层和数据层之间有若干DataService接口。UserPO是作为登录人员的持久化对象被添加到设计模型中去的。

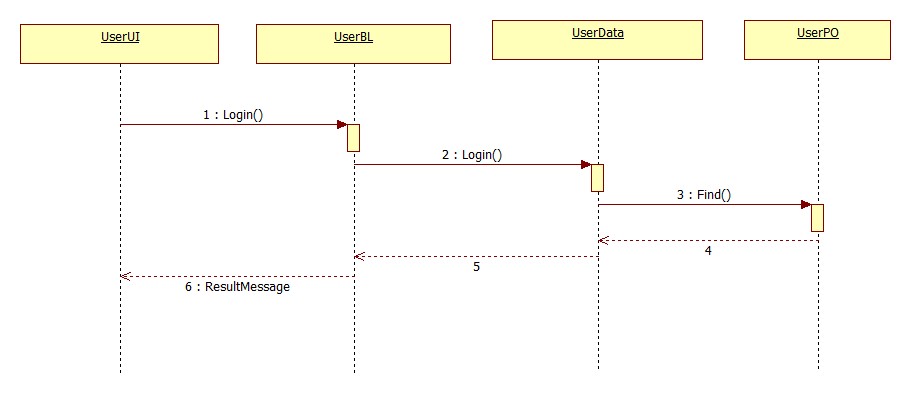
UserBL类职责：负责检查登录信息并准许（禁止）登录。

1. 模块内部的接口规范

UserBL类接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| UserBL.login | 语法 | **public** ResultMessage login(String id,String password) **throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 返回登录结果 |
| UserBL. getUser | 语法 | **public** UserPO getUser(String account) **throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 返回用户信息 |
| UserBL. delUser | 语法 | **public** ResultMessage delUser(String account) **throws** RemoteException ; |
| 前置条件 |  |
| 后置条件 | 系统从员工信息中删除员工 |
| UserBL. addUser | 语法 | **public** ResultMessage addUser(UserPO po) **throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 系统向员工信息中添加员工 |
| UserBL. getList | 语法 | **public** Vector<UserVO> getList(); |
| 前置条件 |  |
| 后置条件 | 系统返回员工信息简略表 |

1. 动态模型



1. 设计原理

分布式控制

### 4.2.2 Commoditybl模块

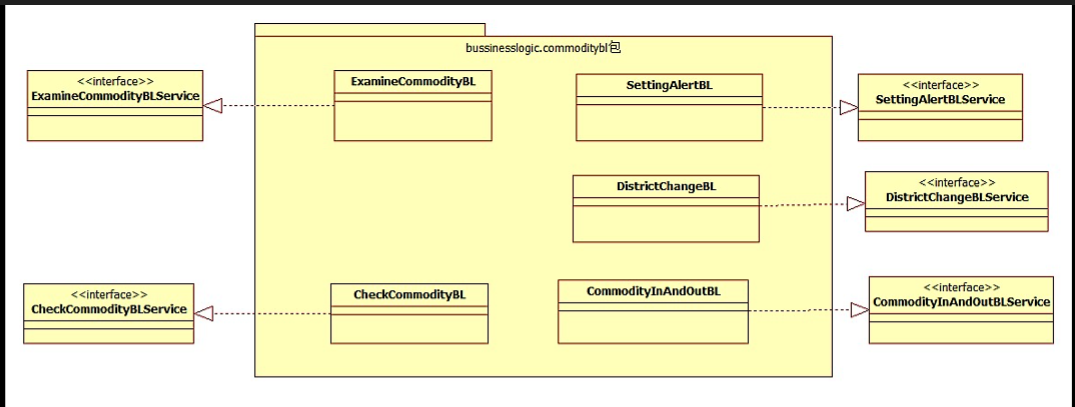
1. 模块概述

Commoditybl模块承担的需求参见需求规格说明文档

Commoditybl模块的职责及接口参见软件系统结构描述文档

1. 整体结构

根据体系结构设计，我们将系统分为展示层、业务逻辑层、数据层，每一层之间添加接口。业务逻辑层和展示层之间有若干BLService接口，业务逻辑层和数据层之间有若干DataService接口。ComPO是作为仓库库存的持久化对象被添加到设计模型中去的。



Commoditybl模块各个类的职责

|  |  |
| --- | --- |
| 类 | 职责 |
| CheckCommodityBL | 负责实现对于库存盘点界面所需要的服务 |
| ExamineCommodityBL | 负责实现对于库存查看界面所需要的服务 |
| CommodityInAndOutBL | 负责实现对于入库单生成和出库单生成界面Commodity模块所需要的服务 |
| SettingAlertBL | 负责实现对于警报设置界面所需要的服务 |
| DistrictChangeBL | 负责实现对于库区调整界面所需要的服务 |

(3)模块内部的接口规范

CheckCommodityBL类接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| CheckCommodityBL.startCheckCommodity | 语法 | **public** **boolean** startCheckCommodity(myDate time,String centerNum) **throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 调用checkCommodityInf |
| CheckCommodityBL.checkCommodityInf | 语法 | **public** Vector<CommodityVO> checkCommodityInf(String centerNum); |
| 前置条件 | startCheckCommodity返回值为真 |
| 后置条件 | 返回所有符合条件的Commodity详细信息 |

ExamineCommodityBL类接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| ExamineCommodityBL.examineCommodity | 语法 | **public** Vector<ExamineVO> examineCommodity(myDate timestart, myDate timeend,String centerNum) **throws** RemoteException; |
| 前置条件 | 起始时间、终止时间格式正确不存在冲突 |
| 后置条件 | 返回所有入库数量int[0]和出库数量int[1] |

CommodityInAndOutBL类接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| CommodityInAndOutBL.Commodityin | 语法 | **public** ResultMessage Commodityin(ComPO po) **throws** RemoteException; |
| 前置条件 | 入库单生成 |
| 后置条件 | 返回是否入库成功 |
| CommodityInAndOutBL.Commodityout | 语法 | **public** ResultMessage Commodityout(ComPO po) **throws** RemoteException; |
| 前置条件 | 出库单生成 |
| 后置条件 | 返回是否出库生成 |

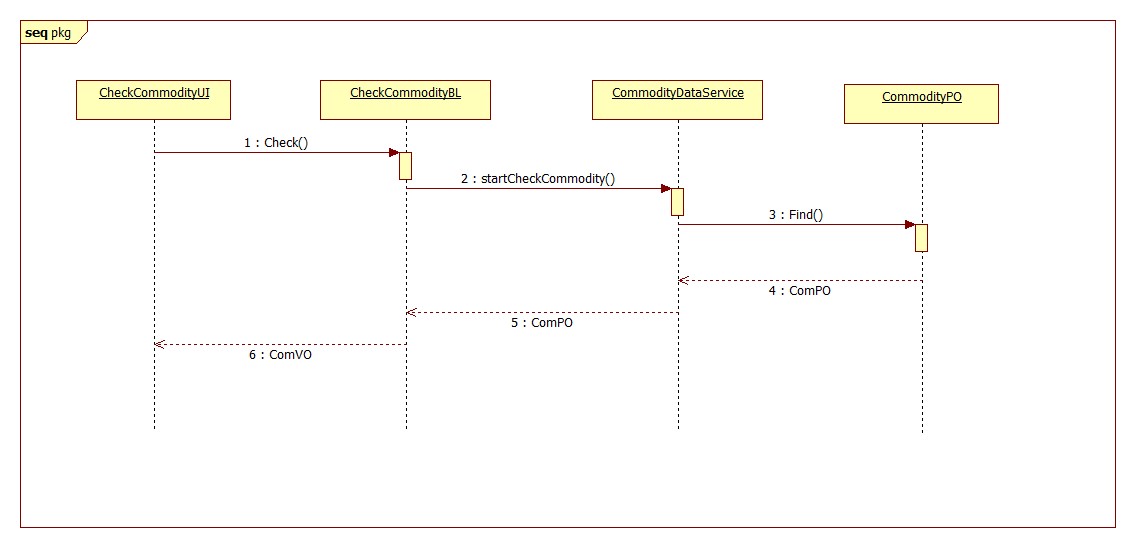
SettingAlertBL类接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| SettingAlertBL.settingAlert | 语法 | **public** **boolean** settingAlert(String centerNum, **double** alertLine) **throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 返回是否设置成功，修改警报线 |
| SettingAlertBL. getAlert | 语法 | **public** **double** getAlert(String centerNum) **throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 获得警报线 |
| SettingAlertBL. ALERT | 语法 | **public** ResultMessage ALERT(String centerNum) **throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 判断是否超过警报 |

DistrictChangeBL类接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| CommodityInAndOutBL.Commodityin | 语法 | **public** ResultMessage change(String from,String to,**int** index,String location) **throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 返回新的ComPO |
| CommodityInAndOutBL. getDistrictCommodity | 语法 | **public** Vector<CommodityVO> getDistrictCommodity(String centerNum,String from) **throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 返回当前仓库中的货物信息列表 |
| CommodityInAndOutBL. changeDistrict | 语法 | **public** ResultMessage changeDistrict(CommodityVO vo) **throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 更新货物的位置信息 |
| CommodityInAndOutBL. getArea | 语法 | **public** List<String> getArea(String centerNum) **throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 返回货物的当前位置 |

1. 动态模型



图为库存盘点的顺序图，其他任务的顺序图可以参考此图

1. 设计原理

分布式控制

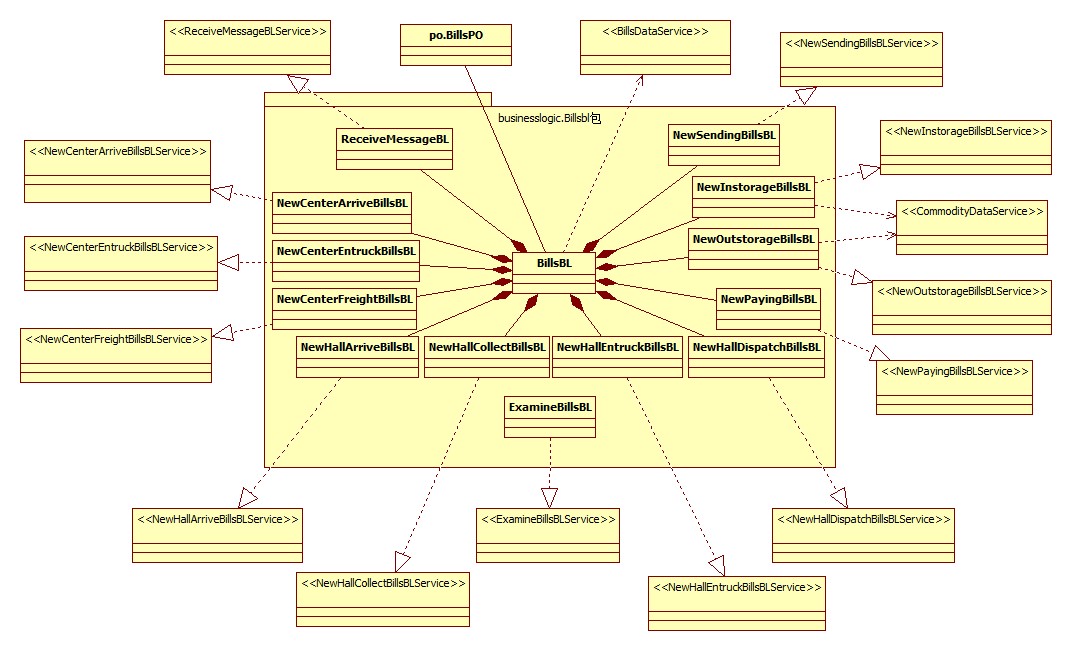
### 4.2.3 Billsbl模块

(1)模块概述

Billsbl模块承担的需求参见需求规格说明文档

Billsbl模块的职责及接口参见软件系统结构描述文档

(2)整体结构



|  |  |
| --- | --- |
| BillsBL模块各个类的职责 | |
| 模块 | 职责 |
| NewSendingBillsBL | 负责实现对应于NewSendingBillsUI界面所需要的服务 |
| ReceiveMessageBL | 负责实现对应于ReceiveMessageUI界面所需要的服务 |
| NewCenterArriveBillsBL | 负责实现对应于NewCenterArriveBillsUI界面所需要的服务 |
| NewCenterEntruckBillsBL | 负责实现对应于NewCenterEntruckBillsUI界面所需要的服务 |
| NewCenterFreightBillsBL | 负责实现对应于NewCenterFreightBillsUI的界面所需要的服务 |
| NewHallArriveBillsBL | 负责实现对应于NewHallArriveBillsUI界面所需要的服务 |
| NewHallCollectBillsBL | 负责实现对应于NewHallCollectBillsUI界面所需要的服务 |
| NewHallDispatchBillsBL | 负责实现对应于NewHallDispatchBillsUI界面所需要的服务 |
| NewHallEntruckBillsBL | 负责实现对应于NewHallEntruckBillsUI界面所需要的服务 |
| NewInstorageBillsBL | 负责实现对应于NewInstorageBillsUI界面所需要的服务 |
| NewOutstorageBillsBL | 负责实现对应于NewOutstorageBillsUI界面所需要的服务 |
| NewPayingBillsBL | 负责实现对应于NewNewPayingBillsUI界面所需要的服务 |
| ExamineBillsBL | 负责实现对应于ExamineBillsUI界面所需要的服务 |

(3)模块内部的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| NewSendingBillsBL | | | |
| 提供的服务（供接口） | | | |
| NewSendingBillsBL.addSendingBills | 语法 | Public ResultMessage addSendingBills(SBVO vo); | |
| 前置条件 | 需要填写的信息已填写完整 | |
| 后置条件 | 系统将生成的单据添加到数据库 | |
| NewSendingBillsBL.getPrice | 语法 | Public long getPrice(String SenderLocation, String ReceiverLocation, BOXTYPE boxtype, SENDTYPE sendtype, String length, String width, String height, String weight); | |
| 前置条件 | 需要填写的信息已填写完整 | |
| 后置条件 | 返回经过计算后的价格 | |
| NewSendingBillsBL.updateSendingBills | 语法 | public ResultMessage updateSendingBills(SBVO vo) | |
| 前置条件 | 无 | |
| 后置条件 | 系统将更新的单据添加到数据库 | |
| NewSendingBillsBL.searchState | 语法 | public String searchState(String goodNum) | |
| 前置条件 | 订单存在 | |
| 后置条件 | 返回快递单的货运状态 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| BillsDataService.insert(BillsPO po) | | | 插入单一持久化对象 |
| OrganizationDataService.getCityAndDistance() | | | 返回城市列表和城市间的距离 |

|  |  |  |  |
| --- | --- | --- | --- |
| NewCenterArriveBillsBL | | | |
| 提供的服务（供接口） | | | |
| NewCenterArriveBillsBL.addCenterArriveBills | 语法 | Public ResultMessage addCenterArriveBills(CABVO vo) | |
| 前置条件 | 需要填写的信息已填写完整 | |
| 后置条件 | 系统将生成的单据添加到数据库 | |
| NewCenterArriveBillsBL.addGoodState | 语法 | Public ResultMessage updateCenterArriveBills(CABVO vo) | |
| 前置条件 | 无 | |
| 后置条件 | 将更新的数据添加到数据库 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| BillsDataService.insert(BillsPO po) | | | 插入单一持久化对象 |

|  |  |  |  |
| --- | --- | --- | --- |
| NewCenterEntruckBillsBL | | | |
| 提供的服务（供接口） | | | |
| NewCenterEntruckBillsBL.addCenterEntruckBills | 语法 | Public ResultMessage addCenterEntruckBills(CEBVO vo) | |
| 前置条件 | 需要填写的信息已填写完整 | |
| 后置条件 | 系统将生成的单据添加到数据库 | |
| NewCenterEntruckBillsBL. updateCenterEntruckBills | 语法 | Public ResultMessage updateCenterEntruckBills(CEBVO vo) | |
| 前置条件 | 无 | |
| 后置条件 | 系统将更新的单据添加到数据库 | |
| NewCenterEntruckBillsBL.getPrice | 语法 | Public long getPrice(String HallNum); | |
| 前置条件 | 需要填写的信息已填写完整 | |
| 后置条件 | 返回经过计算后的价格 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| BillsDataService.insert(BillsPO po) | | | 插入单一持久化对象 |
| OrganizationDataService.getCityAndDistance() | | | 返回城市列表和城市间的距离 |

|  |  |  |  |
| --- | --- | --- | --- |
| NewCenterFreightBillsBL | | | |
| 提供的服务（供接口） | | | |
| NewCenterFreightBillsBL.addCenterFreightBills | 语法 | Public ResultMessage addCenterFreightBills(CFBVO vo) | |
| 前置条件 | 需要填写的信息已填写完整 | |
| 后置条件 | 系统将生成的单据添加到数据库 | |
| NewCenterFreightBillsBL. updateCenterFreightBills | 语法 | Public ResultMessage updateCenterFreightBills(CFBVO vo) | |
| 前置条件 | 无 | |
| 后置条件 | 系统将更新的单据添加到数据库 | |
| NewCenterFreightBillsBL.getPrice | 语法 | public long getPrice(SENDSTYPE send, String StartPlace, String EndPlace) | |
| 前置条件 | 需要填写的信息已填写完整 | |
| 后置条件 | 返回经过计算后的价格 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| BillsDataService.insert(BillsPO po) | | | 插入单一持久化对象 |
| OrganizationDataService.getCityAndDistance() | | | 返回城市列表和城市间的距离 |

|  |  |  |  |
| --- | --- | --- | --- |
| NewHallArriveBillsBL | | | |
| 提供的服务（供接口） | | | |
| NewHallArriveBillsBL.addHallArriveBills | 语法 | Public ResultMessage addHallArriveBills(HABVO vo); | |
| 前置条件 | 需要填写的信息已填写完整 | |
| 后置条件 | 系统将生成的单据添加到数据库 | |
| NewHallArriveBillsBL.updateHallArriveBills | 语法 | Public ResultMessage updateHallArriveBills(HABVO vo); | |
| 前置条件 | 无 | |
| 后置条件 | 系统将更新的单据添加到数据库 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| BillsDataService.insert(BillsPO po) | | | 插入单一持久化对象 |

|  |  |  |  |
| --- | --- | --- | --- |
| NewHallCollectBillsBL | | | |
| 提供的服务（供接口） | | | |
| NewHallCollectBillsBL.addHallCollectBills | 语法 | Public ResultMessage addHallCollectionBills(HCBVO vo) | |
| 前置条件 | 需要填写的信息已填写完整 | |
| 后置条件 | 系统将生成的单据添加到数据库 | |
| NewHallCollectBillsBL.updateHallCollectionBills | 语法 | Public ResultMessage updateHallCollectionBills(HCBVO vo) | |
| 前置条件 | 无 | |
| 后置条件 | 系统将更新的单据添加到数据库 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| BillsDataService.insert(BillsPO po) | | | 插入单一持久化对象 |

|  |  |  |  |
| --- | --- | --- | --- |
| NewHallDispatchBillsBL | | | |
| 提供的服务（供接口） | | | |
| NewHallDispatchBillsBL.addHallDispatchBill | 语法 | Public ResultMessage newHallDispatchBill(HDBVO vo) | |
| 前置条件 | 需要填写的信息已填写完整 | |
| 后置条件 | 系统将生成的单据添加到数据库 | |
| NewHallDispatchBillsBL. updateHallDispatchBill | 语法 | Public ResultMessage updateHallDispatchBill (HDBVO vo) | |
| 前置条件 | 无 | |
| 后置条件 | 系统将更新的单据添加到数据库 | |
| NewHallDispatchBillsBL.searchTheGoods | 语法 | Public String searchTheGoods(String GoodsNum); | |
| 前置条件 | 需要填写的信息已填写完整 | |
| 后置条件 | 将货物信息添加到列表中 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| BillsDataService.insert(BillsPO po) | | | 插入单一持久化对象 |

|  |  |  |  |
| --- | --- | --- | --- |
| NewHallEntruckBillsBL | | | |
| 提供的服务（供接口） | | | |
| NewHallEntruckBillsBL.addHallEntruckBills | 语法 | Public ResultMessage addHallEntruckBills(HEBVO vo) | |
| 前置条件 | 需要填写的信息已填写完整 | |
| 后置条件 | 系统将生成的单据添加到数据库 | |
| NewHallEntruckBillsBL.updateHallEntruckBills | 语法 | Public ResultMessage updateHallEntruckBills(HEBVO vo) | |
| 前置条件 | 无 | |
| 后置条件 | 系统将更新的单据添加到数据库 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| BillsDataService.insert(BillsPO po) | | | 插入单一持久化对象 |
| OrganizationDataService.getCityAndDistance() | | | 返回城市列表和城市间的距离 |

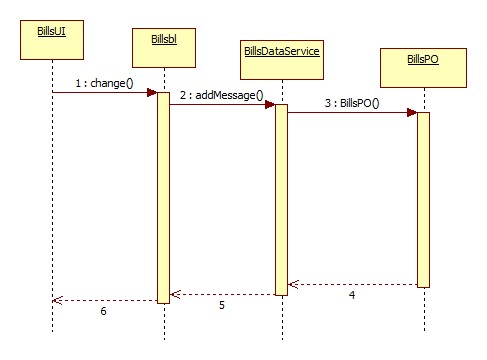
|  |  |  |  |
| --- | --- | --- | --- |
| NewInstorageBillsBL | | | |
| 提供的服务（供接口） | | | |
| NewInstorageBillsBL.addGoods | 语法 | Public ResultMessage newInstorageBill(IBVO vo) | |
| 前置条件 | 需要填写的信息已填写完整 | |
| 后置条件 | 系统将生成的单据添加到数据库 | |
| NewInstorageBillsBL.addCommodity | 语法 | public void addCommodity(ArrayList<ComPO> list) | |
| 前置条件 | 数据库中无货物单号 | |
| 后置条件 | 系统将list中的货物信息添加到仓库数据 | |
| NewInstorageBillsBL.updateInstorageBill | 语法 | public ResultMessage updateInstorageBill(IBVO vo) | |
| 前置条件 | 无 | |
| 后置条件 | 系统将更新的单据添加到数据库 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| BillsDataService.insert(BillsPO po) | | | 插入单一持久化对象 |
| CommodityDataService.insert(ComPO po) | | | 插入单一持久化对象 |

|  |  |  |  |
| --- | --- | --- | --- |
| NewOutstorageBillsBL | | | |
| 提供的服务（供接口） | | | |
| NewOutstorageBillsBL. addOutstorageBills | 语法 | Public CommodityDate addOutstorageBills(OBVO vo) | |
| 前置条件 | 需要填写的信息已填写完整 | |
| 后置条件 | 系统将生成的单据添加到数据库 | |
| NewOutstorageBillsBL. deleteCommodity | 语法 | public void deleteCommodity(ArrayList<ComPO> list) | |
| 前置条件 | 数据库中已存在货物单号 | |
| 后置条件 | 从列表中删除单号 | |
| NewOutstorageBillsBL. creatVO | 语法 | public CommodityVO creatVO(String goodNum) | |
| 前置条件 | 数据库中已存在货物单号 | |
| 后置条件 | 利用数据创建CommodityVO | |
| NewOutstorageBillsBL. creatPO | 语法 | public ComPO creatPO(String goodNum) | |
| 前置条件 | 数据库中已存在货物单号 | |
| 后置条件 | 利用数据创建ComPO | |
| NewOutstorageBillsBL. updateOutstorageBills | 语法 | public ResultMessage updateOutstorageBills(OBVO vo) | |
| 前置条件 | 无 | |
| 后置条件 | 系统将更新的单据添加到数据库 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| BillsDataService.insert(BillsPO po) | | | 插入单一持久化对象 |
| CommodityDataService.delete(ComPO po) | | | 插入单一持久化对象 |

|  |  |  |  |
| --- | --- | --- | --- |
| NewPayingBillsBL | | | |
| 提供的服务（供接口） | | | |
| NewPayingBillsBL.addPayingBills | 语法 | addPayingBills(PayingBills bill) | |
| 前置条件 | 需要填写的信息已填写完整 | |
| 后置条件 | 系统将生成的单据添加到数据库 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| BillsDataService.insert(BillsPO po) | | | 插入单一持久化对象 |

|  |  |  |
| --- | --- | --- |
| ExamineBillsBL的接口规范 | | |
| 提供的服务 | | |
| ExamineBillsBL.examineBills | 语法 | public ResultMessage examineBills(int[] rows) |
| 前置条件 | 用户批量审批单据 |
| 后置条件 | 数据层更新 |
| 需要的服务 | | |
| BillsDataService.examine(BillsPO) | 根据传入的BillsPO,修改相应的单据状态为审批后状态并保存 | |

(4)动态模型



其他过程类似，参考上图。

(5)设计原理

分布式控制风格。

### 4.2.4 Staffbl 模块

(1)模块概述

|  |
| --- |
| StaffBL模块类的职责 |

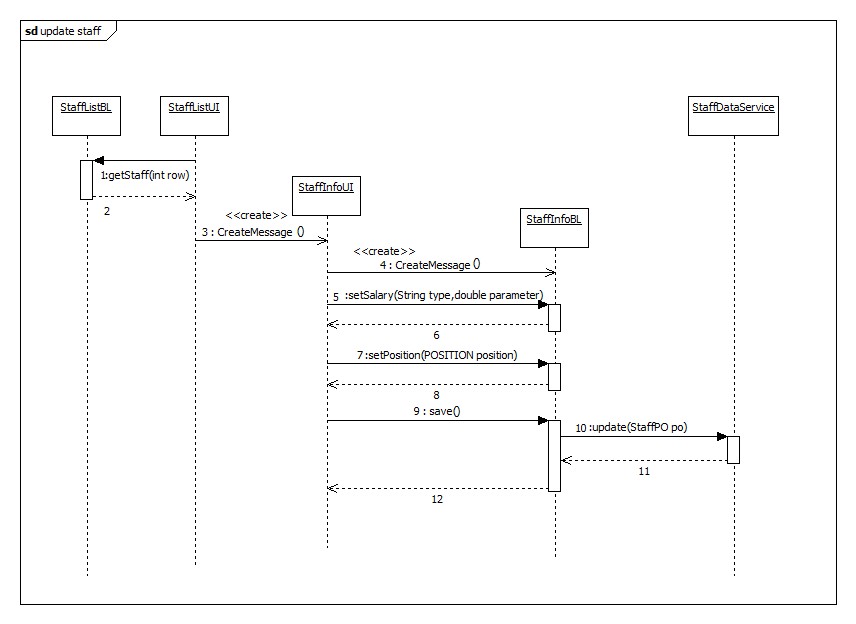
(2)整体结构

(3)模块内部的接口规范

|  |  |  |
| --- | --- | --- |
| StaffBL的接口规范 | | |
| 提供的服务 | | |
| StaffBL. newStaff | 语法 | **public** ResultMessage newStaff(StaffVO vo); |
| 前置条件 |  |
| 后置条件 | 系统将新员工信息添加到数据库 |
| StaffBL. deleteStaff | 语法 | **public** ResultMessage deleteStaff(String num); |
| 前置条件 |  |
| 后置条件 | 系统将选中的员工信息删除 |
| StaffBL. updateStaff | 语法 | **public** ResultMessage updateStaff(StaffVO vo); |
| 前置条件 |  |
| 后置条件 | 系统将员工的新信息添加到数据库 |
| StaffBL. getAllStaff | 语法 | **public** ArrayList<StaffVO> getAllStaff(); |
| 前置条件 |  |
| 后置条件 | 获取所有的员工信息 |
| StaffBL.getList | 语法 | **public** Vector<Vector<String>> getListData(); |
| 前置条件 |  |
| 后置条件 | 获取所有员工的简略信息 |
| StaffBL. findStaff | 语法 | **public** StaffVO findStaff(String num); |
| 前置条件 |  |
| 后置条件 | 返回员工号对应的员工信息 |
| 需要的服务 | | |
| StaffDataService.add(StaffPO po) | 添加一个员工信息 | |
| StaffDataService.remove(String num) | 删除一个员工信息 | |
| StaffDataService.update(StaffPO po) | 更新一个员工信息 | |
| StaffDataService.find(String num) | 搜索一个员工信息 | |
| StaffDataService.getAll() | 返回所有员工信息 | |
| StaffDataService.getAllStaffInfo() | 返回所有员工简略信息 | |

(4)动态模型

下图表明了更改员工数据的过程中相关对象的协作：



其他过程类似，参考上图。

(5)设计原理

分布式控制风格。

### 4.2.5 Organizationbl模块

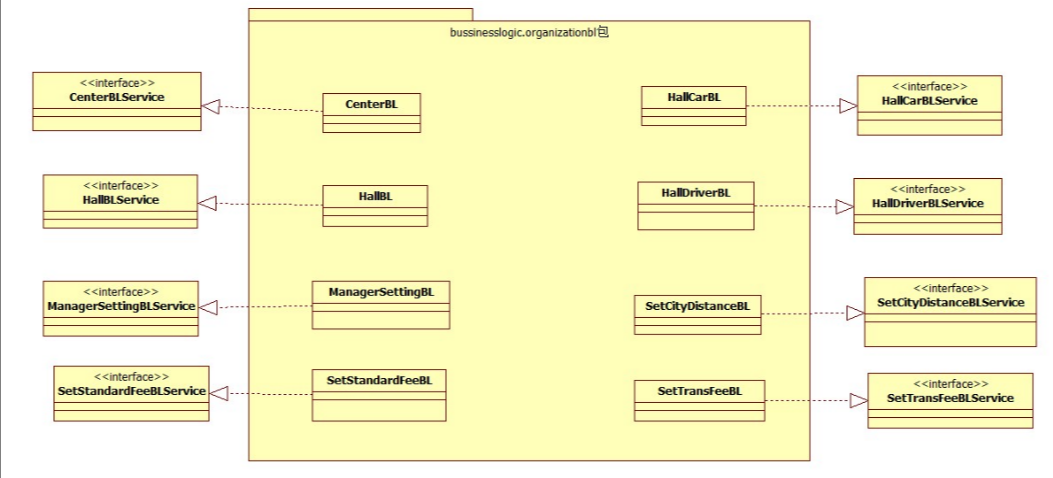
1. 模块概述

Organizationbl模块承担的需求参见需求规格说明文档

Organizationbl模块的职责及接口参见软件系统结构描述文档

1. 整体结构

根据体系结构设计，我们将系统分为展示层、业务逻辑层、数据层，每一层之间添加接口。业务逻辑层和展示层之间有若干BLService接口，业务逻辑层和数据层之间有若干DataService接口。CenterPO是作为中转中心的持久化对象、HallPO是作为营业厅的持久化对象、CarPO是作为车辆的持久化对象、DriverPO是作为司机的持久化对象被添加到设计模型中去的。



Organizationbl模块各个类的职责

|  |  |
| --- | --- |
| 类 | 职责 |
| CenterBL | 负责实现对于中转中心管理界面所需要的服务 |
| HallBL | 负责实现对于营业厅管理界面所需要的服务 |
| HallCarBL | 负责实现对于车辆管理界面所需要的服务 |
| HallDriverBL | 负责实现对于司机管理界面所需要的服务 |
| ManagerSettingBL | 负责实现对于总经理设置经营参数所需要的服务 |

(3)模块内部的接口规范

CenterBL类接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| CenterBL.addCenterbl | 语法 | **public** **boolean** addCenter(CenterVO vo); |
| 前置条件 |  |
| 后置条件 | 添加新的CenterPO |
| CenterBL.delCenter | 语法 | **public** **boolean** delCenter(String CenterNum); |
| 前置条件 |  |
| 后置条件 | 删除选中的CenterPO |
| CenterBL.updateCenter | 语法 | **public** **boolean** updateCenter(CenterVO vo); |
| 前置条件 |  |
| 后置条件 | 将更新的数据存入数据库 |
| CenterBL. getList |  | **public** Vector<CenterVO> getList(); |
|  |  |
|  | 获取Center的列表信息 |

HallBL类接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| HallBL.AddHallbl | 语法 | **public** **boolean** addHall(HallVO vo); |
| 前置条件 |  |
| 后置条件 | 返回新的HallPO |
| HallBL.DelHallbl | 语法 | **public** **boolean** delHall(String HallNum); |
| 前置条件 |  |
| 后置条件 | 返回是否删除成功 |
| HallBL. updateHall | 语法 | **public** **boolean** updateHall(HallVO vo); |
| 前置条件 |  |
| 后置条件 | 更新Hall的数据 |
| HallBL. getList | 语法 | **public** Vector<HallVO> getList(); |
| 前置条件 |  |
| 后置条件 | 获取Hall的列表信息 |

HallCarBL类接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| HallCarBL.addCar | 语法 | **public** ResultMessage addCar(CarPO po)； |
| 前置条件 |  |
| 后置条件 | 返回ResultMessage |
| HallCarBL.delCar | 语法 | **public** ResultMessage delCar(String CarNum) |
| 前置条件 |  |
| 后置条件 | 返回ResultMessage |
| HallCarBL.getList | 语法 | **public** Vector<CarVO> getList() |
| 前置条件 |  |
| 后置条件 | 返回CarVO的列表 |

HallDriverBL类接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| HallDriverBL.addDriver | 语法 | **public** ResultMessage addDriver(DriverPO po) |
| 前置条件 |  |
| 后置条件 | 返回ResultMessage |
| HallDriverBL.delDriver | 语法 | **public** ResultMessage delDriver(String DriverNum) |
| 前置条件 |  |
| 后置条件 | 返回ResultMessage |
| HallDriverBL.getList | 语法 | **public** Vector<DriverVO> getList() |
| 前置条件 |  |
| 后置条件 | 返回DriverVO的列表 |

|  |  |  |
| --- | --- | --- |
| ManagerSettingBL的接口规范 | | |
| SetCityDistanceBL.setCityDistance | 语法 | **public** **boolean** setCityDistance(String city1, String city2,**double** distance); |
| 前置条件 | 用户已在界面上选好了两个城市并输入了距离 |
| 后置条件 | 数据层更新数据 |
| SetCityDistanceBL. getCityDistance | 语法 | **public** **double** getCityDistance(String city1, String city2); |
| 前置条件 |  |
| 后置条件 | 返回两个城市间的距离 |
| SetCityDistanceBL. getCad | 语法 | **public** CityAndDistancePO getCad(); |
| 前置条件 |  |
| 后置条件 | 返回城市距离的PO |
| SetCityDistanceBL. addCity | 语法 | **public** **boolean** addCity(String cityName); |
| 前置条件 |  |
| 后置条件 | 将新城市添加到数据库 |
| SetCityDistanceBL. delCity | 语法 | **public** **boolean** delCity(String cityName); |
| 前置条件 |  |
| 后置条件 | 从数据库中删除城市 |
| SetCityDistanceBL. getExpressFee | 语法 | **public** **double** getExpressFee(SENDSTYPE type); |
| 前置条件 |  |
| 后置条件 | 返回交通方式对应的价格参数 |
| SetCityDistanceBL. setExpressFee | 语法 | **public** **boolean** setExpressFee(SENDSTYPE type,**double** fee); |
| 前置条件 |  |
| 后置条件 | 系统将新设的参数保存入数据库 |
| SetCityDistanceBL. getTransFee | 语法 | **public** **double** getTransFee (TRANSPORTATION type); |
| 前置条件 |  |
| 后置条件 | 返回交通方式对应的价格参数 |
| SetCityDistanceBL. setTransFee | 语法 | **public** **boolean** setTransFee (TRANSPORTATION type, **double** fee); |
| 前置条件 |  |
| 后置条件 | 系统将新设的参数保存入数据库 |

### 4.2.6 Managebl模块

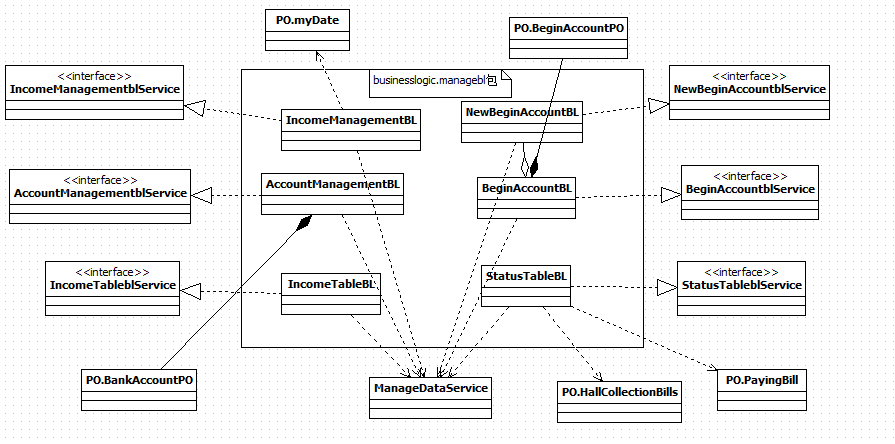
(1)模块概述

Managebl模块承担的需求参见需求规格说明文档

Managebl模块的职责及接口参见软件系统结构描述文档

(2)整体结构

根据体系结构设计，我们将系统分为展示层、业务逻辑层、数据层，每一层之间添加接口。业务逻辑层和展示层之间有若干BLService接口，业务逻辑层和数据层之间有若干DataService接口。BeginAccountPO是作为期初账单的持久化对象，BankAccountPO是作为银行账户的持久化对象被添加到设计模型中去的。



Managebl模块各个类的职责

|  |  |
| --- | --- |
| 类 | 职责 |
| NewBeginAccountBL | 负责实现对应于新建期初账单界面所需要的服务 |
| BeginAccountBL | 负责实现对应于期初建账界面所需要的服务 |
| IncomeManagementBL | 负责实现对应于结算管理界面所需要的服务 |
| AccountManagementBL | 负责实现对应于账户管理界面所需要的服务 |
| IncomeTableBL | 负责实现对应于成本收益表界面所需要的服务 |
| StatusTableBL | 负责实现对应于经营情况表界面所需要的服务 |

(3)模块内部的接口规范

|  |  |  |
| --- | --- | --- |
| NewBeginAccountbl的接口规范 | | |
| 提供的服务（供接口） | | |
| NewBeginAccountblService.addBeginAccount | 语法 | public ResultMessage addBeginAccount(String organization,int people,int car,int storage,String accountname,long balance); |
| 前置条件 | 界面获取输入的期初账单信息 |
| 后置条件 | 生成期初账单并存储信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| BeginAccountDataService.addBeginAccount(BeginaccountPO account) | 存储新建期初账单数据 | |
| BeginAccountDataService.Initialize(BeginaccountPO account) | 初始化系统初始状态 | |

|  |  |  |
| --- | --- | --- |
| BeginAccountbl的接口规范 | | |
| 提供的服务（供接口） | | |
| BeginAccountblService.getBeginAccount | 语法 | public BeginAccountPO getBeginAccount(); |
| 前置条件 | 打开期初建账界面 |
| 后置条件 | 返回当前期初账单 |
| BeginAccountblService. deleteAccount | 语法 | Public ResultMessage deleteAccount(BeginAccountPO account); |
| 前置条件 | 已选择删除账单 |
| 后置条件 | 删除此账单，系统初始状态清零 |
| BeginAccountblService.getBeginAccount | 语法 | public Vector<BeginAccountVO> getBeginAccountList(); |
| 前置条件 | 打开期初建账界面 |
| 后置条件 | 返回所有期初账单 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| BeginAccountDataService.getBeginAccount | 获取期初账单数据 | |
| BeginAccountDataService. deleteAccount (BeginAccountPO account) | 从数据中删除账单数据，修改系统初始数据 | |

|  |  |  |
| --- | --- | --- |
| IncomeManagementbl的接口规范 | | |
| 提供的服务（供接口） | | |
| IncomeManagementblService. incomeSearch | 语法 | public Vector<IncomeBillVO> incomeSearch (myDate date,String hallNumber) |
| 前置条件 | 已获得日期,营业厅编号 |
| 后置条件 | 查找指定日期该营业厅的所有收款单并返回 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| IncomeManagementDataService. search(myDate date,int hallNumber) | 查找指定日期该营业厅的所有收款单并返回 | |

|  |  |  |
| --- | --- | --- |
| AccountManagementbl的接口规范 | | |
| 提供的服务（供接口） | | |
| AccountManagementblService.getAccountList | 语法 | public Vector<BankAccountVO> getAccountList (); |
| 前置条件 | 打开账户管理界面 |
| 后置条件 | 返回账户列表 |
| AccountManagementblService.addAccount | 语法 | Public ResultMessage addAccount(String name); |
| 前置条件 | 已获得新建的账户名称 |
| 后置条件 | 存储账户信息，根据系统收款单和付款单计算该账户余额 |
| AccountManagementblService. calculate | 语法 | public int getbalance (String name); |
| 前置条件 | 打开账户管理界面 |
| 后置条件 | 返回账户余额 |
| AccountManagementblService.delAccount | 语法 | public ResultMessage delAccount(String name); |
| 前置条件 | 已选择删除账号 |
| 后置条件 | 删除账号数据 |
| AccountManagementblService. changeName | 语法 | Public ResultMessage changeName(String name,String newname); |
| 前置条件 | 已获得要修改的账户名和新的账号名 |
| 后置条件 | 修改账号名数据 |
| AccountManagementblService. accountSearch | 语法 | Public Vector<BankAccountVO> accountSearch(String namepart); |
| 前置条件 | 已获得关键字 |
| 后置条件 | 查找账户，若有，则返回列表，若没有，则返回没找到 |
| AccountManagementblService. check | 语法 | Public boolean check (UserPO user) |
| 前置条件 | 将要打开账户管理界面 |
| 后置条件 | 若财务人员有最高权限，进入账户管理；若没有，则拒绝进入账户管理 |
| AccountManagementblService. changeBalance | 语法 | public ResultMessage changeBalance (String name,long change); |
| 前置条件 | 生成新的付款单或收款单 |
| 后置条件 | 修改账户余额 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| AccountManagementDataService. getAccountList | 获取账户列表 | |
| AccountManagementDataService.addAccount(String name) | 存储新建账户数据 | |
| AccountManagementDataService. calculate | 计算账户余额 | |
| AccountManagementDataService.delAccount(String name) | 删除该账户数据 | |
| AccountManagementDataService. changeName(String name,String newname) | 修改账户名数据 | |
| AccountManagementDataService. search(String namepart) | 在数据中查找含有关键字的账户名 | |
| AccountManagementDataService. changeBalance (String name,int change); | 修改账户余额数据 | |

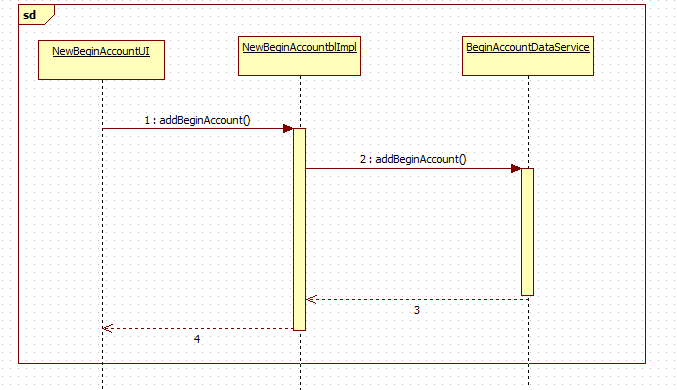
|  |  |  |
| --- | --- | --- |
| CostManagementbl的接口规范 | | |
| 提供的服务（供接口） | | |
| CostManagementBLService.getAllBill | 语法 | public Vector<PayingBillVO> getAllBill(); |
| 前置条件 | 打开成本管理界面 |
| 后置条件 | 无 |
| CostManagementBLService. changeBill | 语法 | public ResultMessage changeBill(String index,PayingBills newBill); |
| 前置条件 | 选择付款单进行修改 |
| 后置条件 | 成功修改 |
| CostManagementBLService. delBill | 语法 | public ResultMessage delBill(String index); |
| 前置条件 | 选择付款单删除 |
| 后置条件 | 删除成功 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| CostManagementDataService. getAllBill() | 获取付款单列表 | |
| CostManagementDataService. changeBill(int index,PayingBills newBill) | 修改付款单 | |
| CostManagementDataService. delBill(int index) | 删除付款单 | |

|  |  |  |
| --- | --- | --- |
| IncomeTablebl的接口规范 | | |
| 提供的服务（供接口） | | |
| IncomeTableblService.getIncome | 语法 | public long getIncome(); |
| 前置条件 | 打开成本收益表界面 |
| 后置条件 | 返回总收入 |
| IncomeTableblService.getPayment | 语法 | public long getPayment (); |
| 前置条件 | 打开成本收益表界面 |
| 后置条件 | 返回总支出 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| IncomeTableDataService.getEarning | 根据数据计算总收入 | |
| IncomeTableDataService.getPayment | 根据数据计算总支出 | |

|  |  |  |
| --- | --- | --- |
| StatusTablebl的接口规范 | | |
| 提供的服务（供接口） | | |
| StatusTableblService.searchIncomeBill | 语法 | public Vector<IncomeBillVO> searchIncomeBill (myDate date1,myDate date2); |
| 前置条件 | 已获取开始和结束日期 |
| 后置条件 | 返回收款单列表 |
| StatusTableblService.searchPayingBill | 语法 | public Vector<PayingBillVO> searchPayingBill (myDate date1,myDate date2); |
| 前置条件 | 已获取开始和结束日期 |
| 后置条件 | 返回付款单列表 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| StatusTableDataService.searchIncomeBill(myDate date1,myDate date2) | 在数据中搜索在此时间段的收款单 | |
| StatusTableblService.searchPayingBill(myDate date1,myDate date2) | 在数据中搜索在此时间段的付款单 | |

(4)动态模型

下图为新建期初账单的顺序图



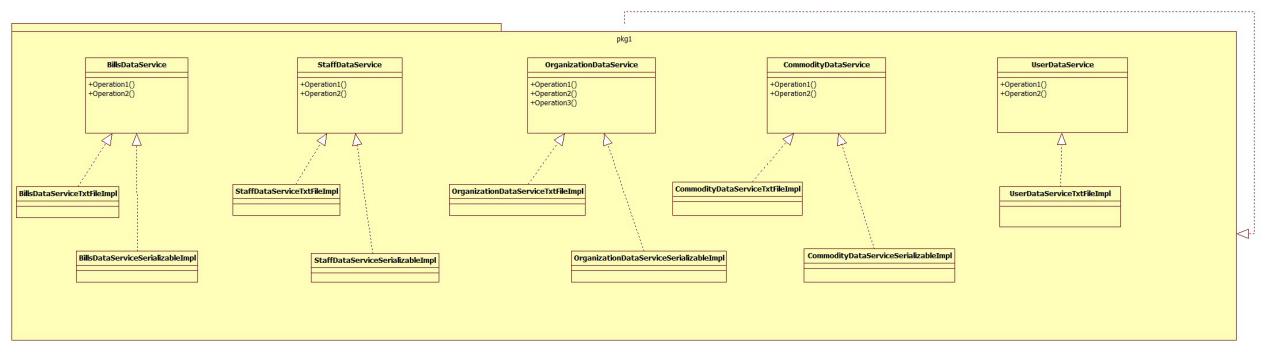
其他类似，参考上图

(5)设计原理

分布式控制风格

## 4.3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，如txt文件、序列化文件、数据库，所示抽象了数据服务。数据层模块的描述具体如下图所示。



### 4.3.1 BillsDataService模块

(1)模块概述

(2)整体结构

(3)模块内部的接口规范

|  |  |  |
| --- | --- | --- |
| 数据层模块的接口规范 | | |
| 提供的服务（供接口） | | |
| BillsDataService.addBills | 语法 | **public** ResultMessage addBills(BillsPO po)**throws** RemoteException; |
| 前置条件 | 同样单据编号的单据在BillsData中不存在 |
| 后置条件 | 在BillsData中添加一个PO |
| BillsDataService.findBills | 语法 | **public** BillsPO findBills(String BillNum)**throws** RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 返回找到的PO |
| BillsDataService.deleteBills | 语法 | **public** ResultMessage deleteBills(String BillNum)**throws** RemoteException; |
| 前置条件 | 在BillsData中存在该编号 |
| 后置条件 | 从BillsData中删除PO |
| BillsDataService.updateBills | 语法 | **public** ResultMessage updateBills(String BillNum,BillsPO bill)**throws** RemoteException; |
| 前置条件 | 在BillsData中存在该编号 |
| 后置条件 | 删除掉原先的PO，添加修改后的PO |
| BillsDataService.examine | 语法 | **public** ResultMessage examine(String BillNum)**throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 将对应的单据的状态改为审批后状态 |
| BillsDataService. getUnexaminedBillsNum | 语法 | **public** **long** getUnexaminedBillsNum()**throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 返回未审批的单据数量 |
| BillsDataService. getUnexaminedBills | 语法 | **public** Vector<Vector<String>> getUnexaminedBills()**throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 返回未审批的单据的简略信息 |
| BillsDataService. getByDate | 语法 | **public** Vector<Vector<String>> getByDate(myDate date1,myDate date2)**throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 返回限定日期内的所有单据的简略信息 |
| BillsDataService. getAllInfo | 语法 | **public** Vector<Vector<String>> getAllInfo()**throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 返回所有单据的简略信息 |
| BillsDataService. getAll | 语法 | **public** ArrayList<BillsPO> getAll()**throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 返回所有单据的PO |
| BillsDataService. save | 语法 | **public** ResultMessage save()**throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 保存数据 |

(4)动态模型

(5)设计原理

### 4.3.2 StaffDataService模块

(1)模块概述

(2)整体结构

(3)模块内部的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StaffDataService.add | 语法 | Public ResultMessage add(StaffPO po) |
| 前置条件 | BL层调用 |
| 后置条件 | 数据库新增 |
| StaffDataService.remove | 语法 | Public ResultMessage remove(String num) |
| 前置条件 | BL层调用 |
| 后置条件 | 数据库删除此PO |
| StaffDataService.update | 语法 | Public ResultMessage update(StaffPO po) |
| 前置条件 | BL层调用 |
| 后置条件 | 数据库更新此PO |
| StaffDataService.find | 语法 | Public StaffPO find(String id) |
| 前置条件 | BL层调用 |
| 后置条件 | 返回此PO给BL层 |
| StaffDataService.getAll | 语法 | Public ArrayList<StaffPO> getAll() |
| 前置条件 | BL层调用 |
| 后置条件 | 返回此ArrayList给BL层 |
| StaffDataService.getAllStaffInfo | 语法 | public Vector<Vector<String>> getAllStaffInfo() |
| 前置条件 | BL层调用 |
| 后置条件 | 返回所有员工简略信息 |

(4)动态模型

(5)设计原理

### 4.3.3 OrganizationDataService模块

(1)模块概述

(2)整体结构

(3)模块内部的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrganizationDataService.OrganizationDataService.addCar | 语法 | **public** ResultMessage addCar(CarPO po)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 新增车辆 |
| OrganizationDataService.OrganizationDataService.addDriver | 语法 | **public** ResultMessage addDriver(DriverPO po)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 新增司机 |
| OrganizationDataService.OrganizationDataService.delCar | 语法 | **public** ResultMessage delCar(CarPO po)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 删除车辆 |
| OrganizationDataService.OrganizationDataService.delDriver | 语法 | **public** ResultMessage delDriver(DriverPO po)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 删除司机 |
| OrganizationDataService.OrganizationDataService.findCar | 语法 | **public** CarPO findCar(**int** CarNum)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 查找车辆 |
| OrganizationDataService.OrganizationDataService.findDriver | 语法 | **public** DriverPO findDriver(**int** driverNum)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 查找司机 |
| OrganizationDataService.OrganizationDataService.SetCitiesDistance | 语法 | **public** CityAndDistancePO SetCitiesDistance(String city1,String city2,**double** distance)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 设置城市距离 |
| OrganizationDataService.OrganizationDataService.newCityDistance | 语法 | **public** CityAndDistancePO newCityDistance(String city,**double**[] distance)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 新增城市距离 |
| OrganizationDataService.CenterDataService. addCenter | 语法 | **public** ResultMessage addCenter(CenterPO po)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 新增中转中心 |
| OrganizationDataService.CenterDataService. delCenter | 语法 | **public** ResultMessage delCenter(String CenterNum)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 删除中转中心 |
| OrganizationDataService.CenterDataService. findCenter | 语法 | **public** CenterPO findCenter(String CenterNum)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 查找中转中心 |
| OrganizationDataService.CenterDataService.updateCenter | 语法 | **public** ResultMessage updateCenter(CenterPO po)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 更新中转中心 |
| OrganizationDataService.CenterDataService. getAll | 语法 | **public** ArrayList<CenterPO> getAll()**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 返回所有中转中心的ArrayList |
| OrganizationDataService.HallDataService. addHall | 语法 | **public** ResultMessage addHall(HallPO po)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 新增营业厅 |
| OrganizationDataService.HallDataService. delHall | 语法 | **public** ResultMessage delHall(String HallNum)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 删除营业厅 |
| OrganizationDataService.HallDataService. delByCity | 语法 | **public** ResultMessage delByCity(String cityNum)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 删除城市营业厅 |
| OrganizationDataService.HallDataService. findHall | 语法 | **public** HallPO findHall(String HallNum)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 查找营业厅 |
| OrganizationDataService.HallDataService. updateHall | 语法 | **public** ResultMessage updateHall(HallPO po)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 更新营业厅 |
| OrganizationDataService.HallDataService. getAll | 语法 | **public** ArrayList<HallPO> getAll()**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 返回所有营业厅的ArrayList |
| OrganizationDataService.ManagerSettingDataService | 语法 | **public** CityAndDistancePO getCityAndDistance()**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 返回城市间距离 |
| OrganizationDataService.ManagerSettingDataService | 语法 | **public** ResultMessage setCityAndDistance(CityAndDistancePO po)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 设置城市间距离 |
| OrganizationDataService.ManagerSettingDataService | 语法 | **public** **double** getExpressFee(SENDSTYPE type)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 返回运输方式对应的费用参数 |
| OrganizationDataService.ManagerSettingDataService | 语法 | **public** ResultMessage setExpressFee(SENDSTYPE type,**double** fee)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 设置运输方式对应的费用参数 |
| OrganizationDataService.ManagerSettingDataService | 语法 | **public** **double** getTransFee(TRANSPORTATION type)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 返回运费比例 |
| OrganizationDataService.ManagerSettingDataService | 语法 | **public** ResultMessage setTransFee(TRANSPORTATION type,**double** fee)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 设置运费比例 |

(4)动态模型

(5)设计原理

### 4.3.4 ManageDataService模块

(1)模块概述

(2)整体结构

(3)模块内部的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ManageDataServive. BeginAccountDataService.addBeginAccount | 语法 | Public ResultMessage addBeginAccount (BeginaccountPO account); |
| 前置条件 | 已经生成期初账单 |
| 后置条件 | 存储期初账单数据 |
| ManageDataServive. BeginAccountDataService.Initialize | 语法 | public void Initialize(BeginaccountPO account); |
| 前置条件 | 期初账单新建 |
| 后置条件 | 初始化期初数据 |
| ManageDataServive. BeginAccountDataService.getBeginAccount | 语法 | public BeginAccountPO getBeginAccount(); |
| 前置条件 | 打开期初建账界面 |
| 后置条件 | 返回期初账单数据 |
| ManageDataServive. BeginAccountDataService. deleteAccount | 语法 | public ResultMessage deleteAccount(BeginAccountPO account); |
| 前置条件 | 删除期初账单 |
| 后置条件 | 删除期初账单数据，初始化系统初始状态 |
| ManageDataServive. BeginAccountDataService.getBeginAccountList | 语法 | public ArrayList<BeginAccountPO> getBeginAccountList() |
| 前置条件 | 打开期初建账界面 |
| 后置条件 | 返回期初账单列表 |
| ManageDataServive. IncomeManagementDataService. search | 语法 | public ArrayList<HallCollectionBills> incomeSearch (myDate date,int hallNumber); |
| 前置条件 | 已获得日期,营业厅编号 |
| 后置条件 | 查找指定日期该营业厅的所有收款单并返回 |
| ManageDataServive. AccountManagementDataService. getAccountList | 语法 | public ArrayList<BankAccountPO> getAccountList (); |
| 前置条件 | 打开账户管理界面 |
| 后置条件 | 返回账户列表 |
| ManageDataServive. AccountManagementDataService.addAccount | 语法 | Public ResultMessage addAccount(String name); |
| 前置条件 | 已获得新建的账户名称 |
| 后置条件 | 存储账户信息，根据系统收款单和付款单计算该账户余额 |
| ManageDataServive. AccountManagementDataService. getBalance | 语法 | public long getBalance (String name); |
| 前置条件 | 已新建该账户 |
| 后置条件 | 返回账户余额 |
| ManageDataServive. AccountManagementDataService.delAccount | 语法 | public ResultMessage delAccount(String name); |
| 前置条件 | 已选择删除账号 |
|  | 删除账号数据 |
| ManageDataServive. AccountManagementDataService. changeName | 语法 | Public ResultMessage changeName(String name,String newname); |
| 前置条件 | 已获得要修改的账户名和新的账号名 |
| 后置条件 | 修改账号名数据 |
| ManageDataServive. AccountManagementDataService. search | 语法 | Public ArrayList<BankAccountPO> search(String namepart); |
| 前置条件 | 已获得关键字 |
| 后置条件 | 查找账户，若有，则返回列表，若没有，则返回没找到 |
| ManageDataServive. AccountManagementDataService. changeBalance | 语法 | public void changeBalance (String name,int change); |
| 前置条件 | 生成新的付款单和收款单 |
| 后置条件 | 修改账户余额并存储 |
| ManageDataServive. CostManagementDataService. getAllBill | 语法 | public ArrayList<PayingBills> getAllBill(); |
| 前置条件 | 无 |
| 后置条件 | 返回付款单列表 |
| ManageDataServive. CostManagementDataService. changeBill | 语法 | public ResultMessage changeBill(String index,PayingBills newBill); |
| 前置条件 | 获得修改后的付款单和序号 |
| 后置条件 | 找到改付款单并修改 |
| ManageDataServive. CostManagementDataService. delBill | 语法 | public ResultMessage delBill(String index); |
| 前置条件 | 获得删除的付款单序号 |
| 后置条件 | 找到改付款单并删除 |
| ManageDataServive. CostManagementDataService. getBillByTime | 语法 | public ArrayList<PayingBills> getBillByTime(myDate date1, myDate date2) |
| 前置条件 | 获得合理的前后日期 |
| 后置条件 | 找到在此日期内的付款单并返回 |
| ManageDataServive. IncomeTableDataService.getEarning | 语法 | public long getIncome(); |
| 前置条件 | 打开成本收益表界面 |
| 后置条件 | 返回总收入 |
| ManageDataServive. IncomeTableDataService.getPayment | 语法 | public long getPayment (); |
| 前置条件 | 打开成本收益表界面 |
| 后置条件 | 返回总支出 |
| ManageDataServive. StatusTableDataService.searchIncomeBill | 语法 | public ArrayList<HallCollectionBills> searchIncomeBill (myDate date1,myDate date2); |
| 前置条件 | 已获取开始和结束日期 |
| 后置条件 | 返回收款单列表 |
| ManageDataServive. StatusTableblService.searchPayingBill | 语法 | public ArrayList<PayingBills> searchPayingBill (myDate date1,myDate date2); |
| 前置条件 | 已获取开始和结束日期 |
| 后置条件 | 返回付款单列表 |

(4)动态模型

(5)设计原理

### 4.3.5 CommodityDataService模块

(1)模块概述

(2)整体结构

(3)模块内部的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CommodityDataService.addCom | 语法 | **public** ResultMessage addCom(ComPO po) **throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 数据库新增 |
| CommodityDataService.delCom | 语法 | **public** ResultMessage delCom(ComPO po) **throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 数据库中删除此PO |
| CommodityDataService.findCom | 语法 | **public** ComPO findCom(String GoodsNum) **throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 返回此PO |
| CommodityDataService. getAllCom | 语法 | **public** ArrayList<ComPO> getAllCom(String centerNum) **throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 返回所有PO的ArrayList |
| CommodityDataService. setAlert | 语法 | **public** ResultMessage setAlert(String centerNum, **double** line)**throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 设置报警参数 |
| CommodityDataService. getAlert | 语法 | **public** AlertPO getAlert(String centerNum) **throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 返回报警参数 |
| CommodityDataService. getComSize | 语法 | **public** **int** getComSize(String centerNum) **throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 返回仓库规模 |
| CommodityDataService. addNewCity | 语法 | **public** ResultMessage addNewCity(String centerNum) **throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 新增城市仓库 |
| CommodityDataService. delNewCity | 语法 | **public** ResultMessage delNewCity(String centerNum) **throws** RemoteException; |
| 前置条件 | BL层调用 |
| 后置条件 | 删除城市仓库 |

(4)动态模型

(5)设计原理

### 4.3.6 UserDataService模块

(1)模块概述

(2)整体结构

(3)模块内部的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserDataService. insert | 语法 | **public** ResultMessage insert(UserPO po) **throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 新增用户信息 |
| UserDataService. confirm | 语法 | **public** ResultMessage confirm(String account,String password) **throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 确认用户身份 |
| UserDataService. find | 语法 | **public** UserPO find(String account) **throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 返回用户信息 |
| UserDataService. delete | 语法 | **public** ResultMessage delete(String account) **throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 删除用户信息 |
| UserDataService. getList | 语法 | **public** ArrayList<UserPO> getList()**throws** RemoteException; |
| 前置条件 |  |
| 后置条件 | 返回用户列表 |

(4)动态模型

(5)设计原理

# 5.依赖视角

