

快递管理系统软件详细设计文档



2015-11-16

南京大学软件学院

林庆、张健、李芷牧、令佩棠

目录

[1. 引言 3](#_Toc30546)

[1.1编制目的 3](#_Toc299)

[1.2词汇表 3](#_Toc18051)

[1.3参考资料 3](#_Toc13954)

[2. 产品概述 3](#_Toc11355)

[3. 体系结构设计概述 4](#_Toc10111)

[4. 结构视角 4](#_Toc21879)

[4.1 业务逻辑层的分解 4](#_Toc3349)

[4.1.1 storagebl模块 4](#_Toc672)

[4.1.2 logispicsquerybl模块 13](#_Toc27132)

[4.1.3 collectionbl模块 15](#_Toc17499)

[4.1.4 transferbl模块 21](#_Toc2713)

[4.1.4 approvelbl模块 26](#_Toc25536)

[4.1.5 financebl模块 29](#_Toc4311)

[4.1.7 infobl模块 35](#_Toc27486)

[4.1.8 systembl模块 47](#_Toc21525)

[5. 依赖视角 52](#_Toc6137)

# 1. 引言

## 1.1编制目的

本报告详细完成了对快递物流管理系统的详细设计，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 |
| **YloadDoc** | 营业厅装车单 |  |
| **YArrivalDoc** | 营业厅到达单 |  |
| **ZArrivalDoc** | 中转中心到达单 |  |
| **TransferDoc** | 中转单 |  |
| **ZloadDoc** | 中转中心装车单 |  |
| **SendDoc** | 寄件单 |  |
| **InWarehouseDoc** | 入库单 |  |
| **OutWarehouseDoc** | 出库单 |  |
| **OverDoc** | 收件单 |  |
| **PayDoc** | 付款单 |  |
| **GatheringDoc** | 收款单 |  |
| **YDeliverDoc** | 派件单 |  |
| **CostInComeDocPO** | 成本收益表 |  |
| **RecordPO** | 仓库记录 |  |
| **CMASSS** | 快递管理与服务系统 |  |

## 1.3参考资料

1) 快递管理与服务系统（CMASS）用例文档v1.0

2) 快递管理与服务系统（CMASS）软件需求规格说明文档

# 2. 产品概述

CMASS是为一家本地民营物流企业开发的快递管理与服务系统，开发的目标是帮助企业wanchengrichang 的业务流程，包括：

SF1：对输送、保管、包装、流通加工等快递物流活动进行衔接活动，对装卸活动的管理，确定最恰当的装卸方式，合理配置。获得较好的经济效果。

SF2：对运输活动的管理，要求选择技术经济效果最好的运输方式及联运方式，合理确定运输路线，以实现安全、迅速、准时、价廉的要求。

SF3：处理营业厅司机，车辆的管理。

SF4：对快件的运费和到达时间有预估功能。

SF5：处理中转中心或营业厅快递的接收与派件，包括装车管理，装运管理等。

SF6：对仓库的管理，包括仓库货物的出库，入库，报警，分区，盘点等。

SF7：帮助财务人员处理公司财务状况，包括账户管理，制作成本收益，经营情况表等。

SF8：协助总经理的决策。

通过CMASS的应用，期望为民营企业提高员工工作效率，降低成本，减少企业内部交流障碍，吸引顾客，提高利润。

# 3. 体系结构设计概述

对于本快递物流信息管理系统（CMASS），系统将由两部分程序组成，一部分是安装在不同机构的客户端程序，另一部分是服务器端程序。采用分层风格完成对功能性需求的设计，通过非功能性需求约束和改进设计。

# 4. 结构视角

## 4.1 业务逻辑层的分解

业务逻辑层的开发包图见软件体系结构设计文档。

### 4.1.1 storagebl模块

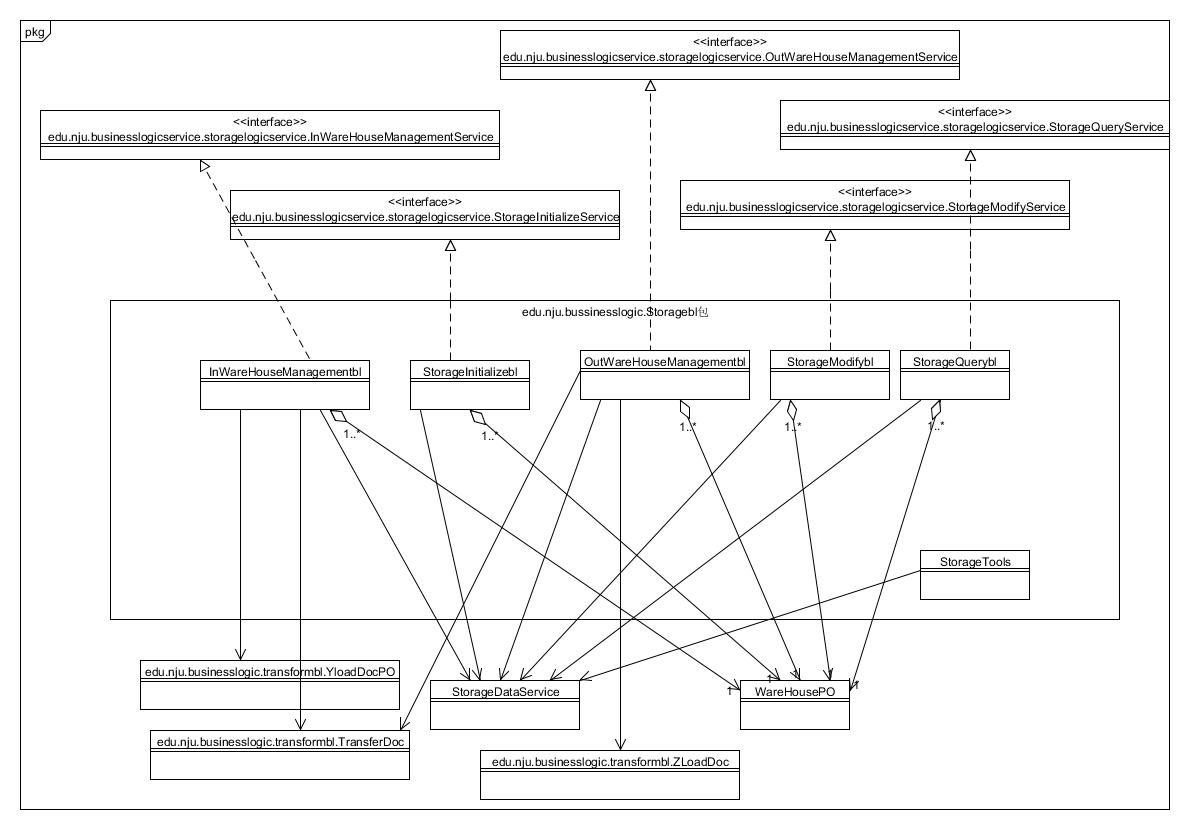
(1) 模块概述

Storagebl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

Storagebl模块的职责及接口参见软件详细设计文档。

(2) 整体结构

根据体系结构的设计，我们将系统分为展示层，业务逻辑层，数据层。每一层之间为了增加灵活性，我们加了Service层。比如逻辑层和展示层之间我们增加了businesslogicService层。Storagebl的模块设计如下所示：



Storagebl模块各个类的职责如下表所示：

|  |  |
| --- | --- |
| 模块 | 职责 |
| InWareHouseManagementbl | 负责生成入库单的VO |
| OutWareHouseManagementbl | 负责生成出库单的VO |
| StorageInitializebl | 负责库存初始化相关业务 |
| StorageModifybl | 负责库区调整的业务逻辑 |
| StorageQuerybl | 负责库存盘点的业务逻辑 |
| StorageTools | 负责为其他类提供工具服务 |

(3) 模块内部类的接口规范

InWareHouseManagementbl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| InWareHouseManagementbl.getVO\_Transfer | 语法 | public InWareHouseDocVO getInWareHouseDocVO\_Transfer(int TransferDocID) |
| 前置条件 | 已经有一个对应的中转单编号 |
| 后置条件 | 根据中转单编号生成入库单 |
| InWareHouseManagementbl.getVO\_YloadDocID | 语法 | public InWareHouseDocVO getInWareHouseDocVO\_YloadDoc(int YloadDocID) |
| 前置条件 | 已经有一个对应的营业厅装车单编号 |
| 后置条件 | 根据营业厅装车单编号生成入库单 |
| InWareHouseManagementbl.updataFile | 语法 | public void updateInWareHouseDoc(InWareHouseDocVO out) |
| 前置条件 | 已经生成了一个入库单 |
| 后置条件 | 更新入库单文件 |
| InWareHouseManagementbl.updateStorage | 语法 | public void saveInWareHouseDoc(InWareHouseDocPO in) |
| 前置条件 | 无 |
| 后置条件 | 根据审批后的入库单更新库存信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| TransferDocPO getTransferPO(int TransferID) | 根据中转单号获得中转单对象 | |
| public long getInWareHouseDocID(); | 获取当前入库单应得的编号 | |
| YloadDocPO getYloadDocPO(int YloadDocID) | 根据营业厅装车单号号获得营业厅装车单对象 | |
| updateInWareHouseDoc(InWareHouseDocPO out) | 根据一个入库单PO更新入库单文件 | |
| saveOperation | 保存操作日志 | |

OutWareHouseManagementbl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OutWareHouseManagementbl.getVO\_ ZloadDoc | 语法 | public OutWareHouseDocVO getOutWareHouseDocVO\_ZloadDoc(int ZloadDocID) |
| 前置条件 | 已有一个对应的一个中转中心装车单编号 |
| 后置条件 | 生成出库单VO给界面 |
| OutWareHouseManagementbl.getVO\_Transfer | 语法 | public OutWareHouseDocVO getOutWareHouseDocVO\_Transfer(int TransferDocID) |
| 前置条件 | 已有一个对应的中转单编号 |
| 后置条件 | 生成出库单VO给界面 |
| OutWareHouseManagementbl.updataFile | 语法 | public void updateOutWareHouseDoc(OutWareHouseDocVO out) |
| 前置条件 | 已经生成了一个出库单 |
| 后置条件 | 更新出库单文件 |
| OutWareHouseManagementbl.updateStorage | 语法 | void saveInWareHouseDoc(OutWareHouseDocPO out) |
| 前置条件 | 无 |
| 后置条件 | 根据审批后的出库单更新库存状态 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| ZloadDoc getZloadDocPO(int ZloadDocID) | 根据中转中心装车单编号获得中转中心装车单对象 | |
| TransferPO getTransferPO(int TransferID) | 根据中转单编号获得中转单对象 | |
| void updateOutWareHouseDoc(OutWareHouseDocPO out) | 根据生成的出库单PO对象，更新文件 | |
| void saveInWareHouseDoc(OutWareHouseDocPO out) | 保存操作日志 | |
| getCurrOutWare\_ID() | 得到当前出库单编号 | |

StorageInitializebl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StorageInitializebl.clear | 语法 | public void clear() |
| 前置条件 | 无 |
| 后置条件 | 清除当前库存信息 |
| StorageInitializebl. addNewStorageItem | 语法 | void addNewStorageItem(RecordPO recordPO) |
| 前置条件 | 已经输入了一条库存对象的各项数据 |
| 后置条件 | 向仓库增加了一条库存记录 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| public void clear(storageID); | 将仓库文件的数据清空 | |
| public void addNewStorageItem(RecordPO recordPO) | 根据新添加的库存对象的数据更新库存文件 | |

StorageQuerybl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StorageQuerybl.getInWareHouseDocVO\_Fly() | 语法 | public ArrayList<RecordPO> getInWareHouseDocVO\_Fly() |
| 前置条件 | 无 |
| 后置条件 | 获得当前仓库的航运区的库存对象列表 |
| StorageQuerybl.getInWareHouseDocVO\_Train() | 语法 | public ArrayList<RecordPO> getInWareHouseDocVO\_Train() |
| 前置条件 | 无 |
| 后置条件 | 获得当前仓库的货运区的库存对象列表 |
| StorageQuerybl. getInWareHouseDocVO\_Car() | 语法 | public ArrayList<RecordPO> getInWareHouseDocVO\_Car() |
| 前置条件 | 无 |
| 后置条件 | 获得当前仓库的汽运区的库存对象列表 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| public ArrayList<RecordPO> getInWareHouseDocVO\_Fly() | 数据层需要读取文件，生成航运区的库存对象list传给逻辑层 | |
| public ArrayList<RecordPO> getInWareHouseDocVO\_Train() | 数据层需要读取文件，生成货运区的库存对象list传给逻辑层 | |
| public ArrayList<RecordPO> getInWareHouseDocVO\_Car() | 数据层需要读取文件，生成汽运区的库存对象list传给逻辑层 | |

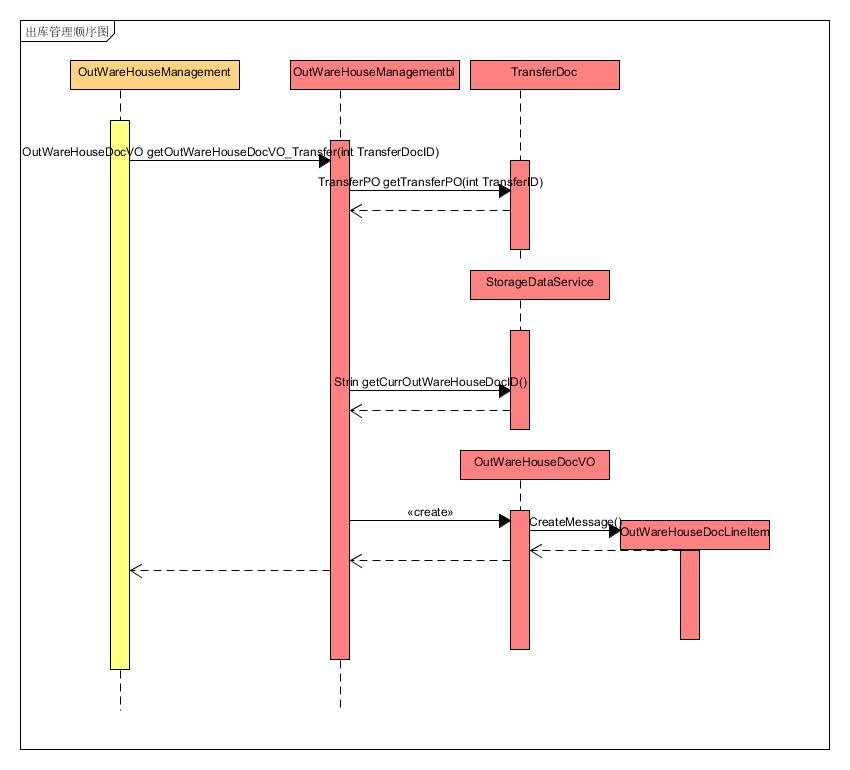
StorageModifybl的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StorageModifybl. storageModify | 语法 | void storageModify(String district,String currPersonID,String storageID,int quantity) |
| 前置条件 | 无 |
| 后置条件 | 将指定数量的机动区划归其他区 |
| StorageModifybl. storageRealease | 语法 | void storageRealease(String storageID) |
| 前置条件 | 无 |
| 后置条件 | 释放机动区空间 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| void storageModify(String to, String storageID,int quantity) | 将指定数量的机动区划归其他区 | |
| void storageRealease(String storageID) | 释放机动区空间 | |

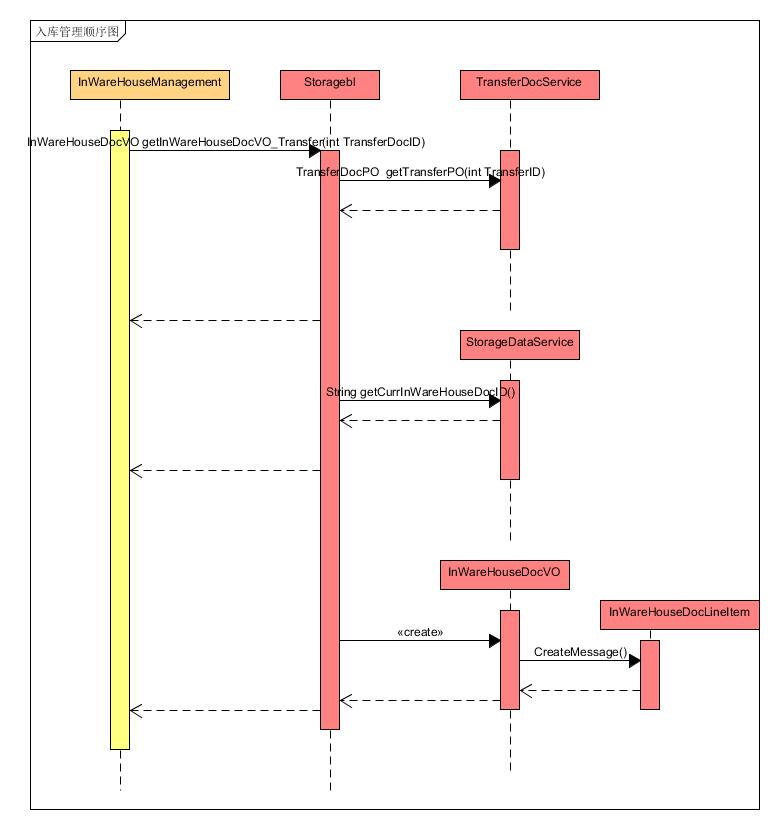
StorageTools的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StorageTools. getOutWareHouseDocList | 语法 | ArrayList<OutWareHouseDocPO> getOutWareHouseDocList() |
| 前置条件 | 无 |
| 后置条件 | 将指定数量的机动区划归其他区 |
| StorageTools. getInWareHouseDocList | 语法 | ArrayList<InWareHouseDocPO> getInWareHouseDocList() |
| 前置条件 | 无 |
| 后置条件 | 释放机动区空间 |
| StorageTools. saveInWareHouseDocPO | 语法 | void saveInWareHouseDocPO(InWareHouseDocPO in) |
| 前置条件 | 无 |
| 后置条件 | 根据审批好的入库单更新库存信息 |
| StorageTools. saveOutWareHouseDocPO | 语法 | void saveOutWareHouseDocPO(OutWareHouseDocPO out) |
| 前置条件 | 无 |
| 后置条件 | 根据审批好的出库单更新库存信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| ArrayList<OutWareHouseDocPO> getOutWareHouseDoc\_unchecked() | 获取未审批的出库单 | |
| ArrayList<InWareHouseDocPO> getInWareHouseDocList() | 获取未审批的入库单 | |
| void saveInWareHouseDoc(InWareHouseDocPO in) | 根据已审批的入库单更新仓库库存信息 | |
| void saveOutWareHouseDoc(OutWareHouseDocPO out) | 根据已审批的出库单更新仓库库存信息 | |

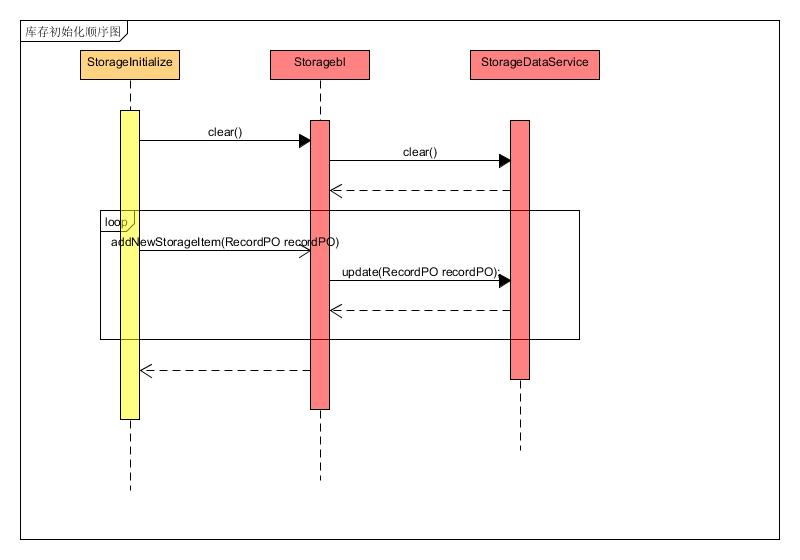
(4) 业务逻辑层的动态模型



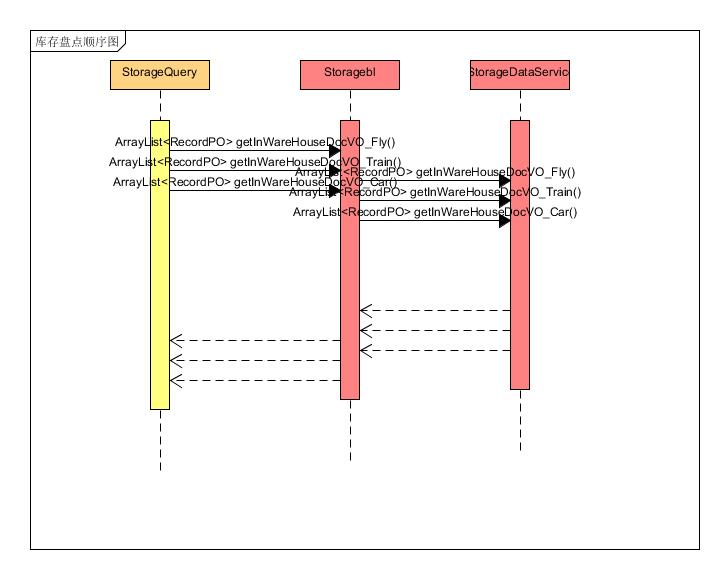
出库管理顺序图



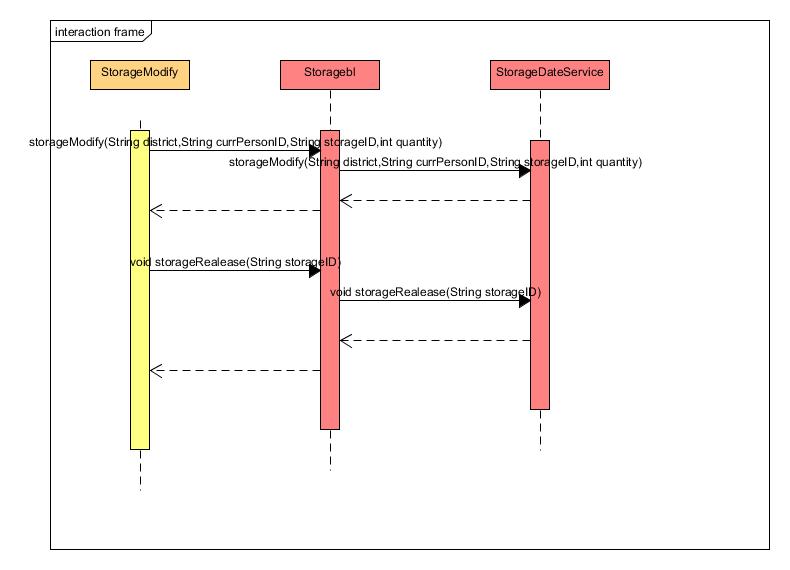
入库管理顺序图



库存初始化顺序图



库存盘点顺序图



库区调整顺序图

(5) 业务逻辑层的设计原理

采用分散式风格，每个逻辑模块内部类只负责一个功能流程。互相尽少干涉。

### 4.1.2 logispicsquerybl模块

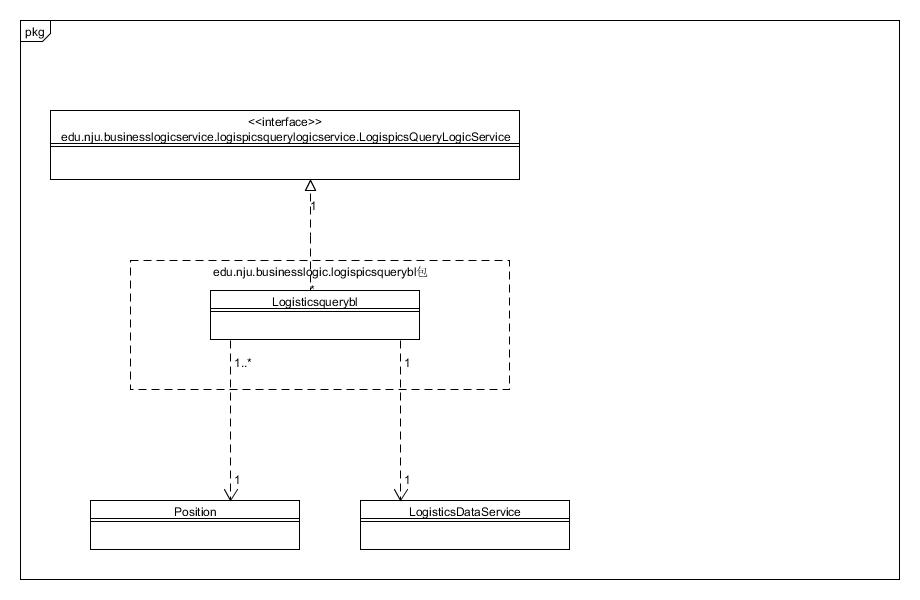
(1) 模块概述

Logispicsquerybl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

Logispicsquerybl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

物流查询这个模块分为展示层，业务逻辑层，数据层。展示层与逻辑层之间有logispicsqueryblService接口，逻辑层与数据层之间有LogisticsDataService接口。PositionPO是作为快递历史轨迹的持久化对象被添加到设计模型中去的。

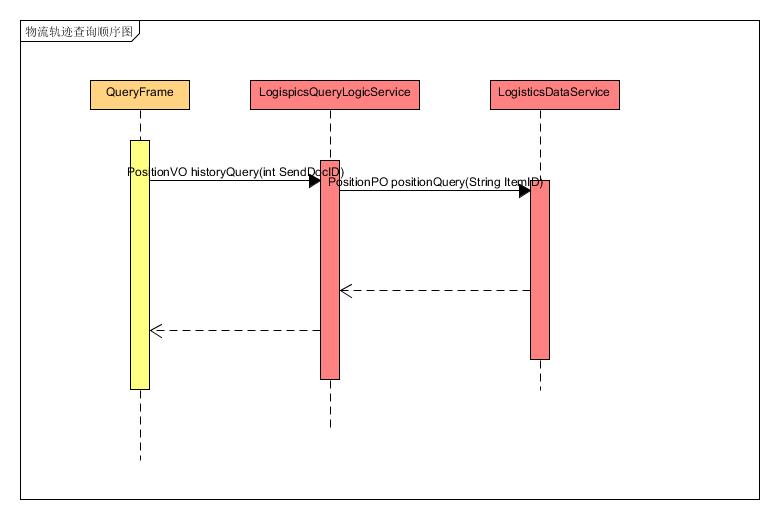


(3) 模块内部类的接口规范

Logispicsquerybl类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Logispicsquerybl. HistoryQuery | 语法 | public PositionVO historyQuery(int SendDocID) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| Logispicsquerybl.updatePosition | 语法 | public void changePosition(long ItemID,String pos) |
| 前置条件 | 无 |
| 后置条件 | 更新某快递的历史轨迹信息 |
| Logispicsquerybl.addPosition | 语法 | public void createPosition(long Item,String pos) |
| 前置条件 | 无 |
| 后置条件 | 增加某个快件的历史轨迹信息记录 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| public void changePosition(long ItemID,String pos); | 更新快递号为Item的快递的历史轨迹信息 | |
| public void createPosition(long Item,String pos); | 增加快递号为Item的快递的历史轨迹记录 | |
| PositionPO positionQuery(String ItemID); | 查询某快递号的快递的历史轨迹信息 | |

(4) 业务逻辑层的动态模型



物流轨迹查询顺序图

### collectionbl模块

1. 模块概述

Collectionbl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求

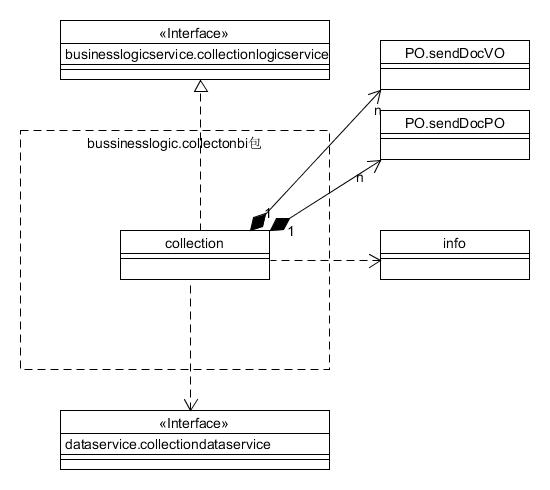
Collectionbl模块的职责及接口参见软件体系结构描述文档

(2) 整体结构

展示层、业务逻辑层、数据层间由collectionLogicService接口与collectionDataService分隔，业务逻辑完全由collection对象完成，sendDocPO是寄件单的持久化对象，sendDocVO

是寄件单的可视化对象。

Collectionbl模块设计如图所示

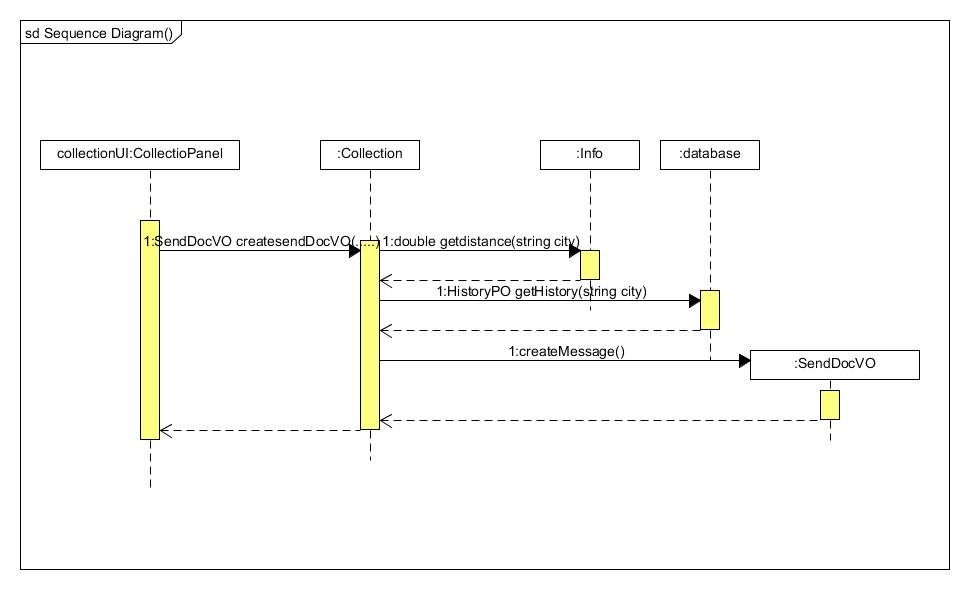


(3) 模块内部类的接口规范

Collection的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 供接口 | | | | |
| Collection.saveSendDocPO | 语法 | | public void saveSendDocPO(SendDocPO po) |
| 前置条件 | | SenddocPO存在 |
| 后置条件 | | 保存寄件单 |
| Collection.getDistance | 语法 | | public double getDistance(String city) |
| 前置条件 | | 无 |
| 后置条件 | | 得到城市距离 |
| Collection.getSequence | 语法 | | public String getSequence() |
| 前置条件 | | 无 |
| 后置条件 | | 得到寄件单序号 |
| Collection.changeSequence | 语法 | | public void changeSequence() |
| 前置条件 | | 寄件单已保存 |
| 后置条件 | | 更改序号 |
| Collection.getAllSendDoc | 语法 | | public List<SendDocPO> getAllSendDoc() |
| 前置条件 | | 无 |
| 后置条件 | | 获取所有未审批单据 |
| Collection.getCourierMoney | 语法 | | public double getCourierMoney(String courierID) |
| 前置条件 | | 快递员正确 |
| 后置条件 | | 得到快递员的业绩 |
| Collection.createSendDocVO | 语法 | | public SendDocVO createSendDocVO(String sName,String sCity,String sAddress, String sUnit,String sTelePhone,String sMobilePhone,  String rName,String rCity,String rAddress,String rUnit,String rTelePhone  ,String rMobilePhone,int itemNum,double weight,double[] volume  ,String itemKind,int packageType,int sendType) |
| 前置条件 | | 数据完整 |
| 后置条件 | | 获取寄件单vo |
| Collection.timeEstimate | 语法 | | public int timeEstimate(String sCity,String rCity) |
| 前置条件 | | 无 |
| 后置条件 | | 获取预估时间 |
| Collection.priceCalc | 语法 | | public double priceCalc(String sCity,String rCity,int packing,double[] volume,double weight,int sendType) |
| 前置条件 | | 数据完整 |
| 后置条件 | | 计算价格 |
| Collection.getSendDocPOByID | 语法 | | public SendDocPO getSendDocPOByID(String itemID) |
| 前置条件 | | Id正确 |
| 后置条件 | | 获取寄件单po |
| Collection.getSendDocsByID | 语法 | | public ArrayList<String> getSendDocsByID(String courier\_ID) |
| 前置条件 | | Id正确 |
| 后置条件 | | 获取该快递员的寄件单 |
| Collection.saveSendDocCreateGatheringDoc | 语法 | | public void saveSendDocCreateGatheringDoc(String courierID) |
| 前置条件 | | Id正确 |
| 后置条件 | | 重置快递员业绩 |
| Collection.getHistoryPO | 语法 | | public SendDocPO getSendDocPOByID(String itemID) |
| 前置条件 | | Id正确 |
| 后置条件 | | 获取寄件单po |
| Collection.getSendDocPOByID | 语法 | | ArrayList<HistoryTimePO> getHistoryPO(String sCity, String rCity) |
| 前置条件 | | 无 |
| 后置条件 | | 获取寄件历史时间记录 |
| Collection.confirmSave | 语法 | | public void confirmSave() |
| 前置条件 | | 已生成单据 |
| 后置条件 | | 确认保存 |
| 需要的服务 | | | |
| 服务名 | | 服务 | |
| CollectionDataService.saveSendDocPO(SendDocPO po) | | 保存寄件单单一持久化对象 | |
| CollectionDataService.getDistance(String city) | | 获取距离数据 | |
| CollectionDataService.getSequence() | | 获取序列号 | |
| CollectionDataService.changeSequence(string sequence) | | 改变最后序列号 | |
| CollectionDataService.getSendDocPOByID(String ID) | | 根据id获取寄件单 | |
| CollectionDataService.getAllSendDoc()t | | 获取未审批寄件单 | |
| CollectionDataService.saveCourierMoneyPO(CourierMoneyPO po | | 保存快递员业绩记录 | |
| CollectionDataService.getCourierMoneyPO(String courierID) | | 获取快递员业绩记录 | |
| 供接口 | | | | |
| Collection.saveSendDocPO | 语法 | | public void saveSendDocPO(SendDocPO po) |
| 前置条件 | | SenddocPO存在 |
| 后置条件 | | 保存寄件单 |
| Collection.getDistance | 语法 | | public double getDistance(String city) |
| 前置条件 | | 无 |
| 后置条件 | | 得到城市距离 |
| Collection.getSequence | 语法 | | public String getSequence() |
| 前置条件 | | 无 |
| 后置条件 | | 得到寄件单序号 |
| Collection.changeSequence | 语法 | | public void changeSequence() |
| 前置条件 | | 寄件单已保存 |
| 后置条件 | | 更改序号 |
| Collection.getAllSendDoc | 语法 | | public List<SendDocPO> getAllSendDoc() |
| 前置条件 | | 无 |
| 后置条件 | | 获取所有未审批单据 |
| Collection.getCourierMoney | 语法 | | public double getCourierMoney(String courierID) |
| 前置条件 | | 快递员正确 |
| 后置条件 | | 得到快递员的业绩 |
| Collection.createSendDocVO | 语法 | | public SendDocVO createSendDocVO(String sName,String sCity,String sAddress, String sUnit,String sTelePhone,String sMobilePhone,  String rName,String rCity,String rAddress,String rUnit,String rTelePhone  ,String rMobilePhone,int itemNum,double weight,double[] volume  ,String itemKind,int packageType,int sendType) |
| 前置条件 | | 数据完整 |
| 后置条件 | | 获取寄件单vo |
| Collection.timeEstimate | 语法 | | public int timeEstimate(String sCity,String rCity) |
| 前置条件 | | 无 |
| 后置条件 | | 获取预估时间 |
| Collection.priceCalc | 语法 | | public double priceCalc(String sCity,String rCity,int packing,double[] volume,double weight,int sendType) |
| 前置条件 | | 数据完整 |
| 后置条件 | | 计算价格 |
| Collection.getSendDocPOByID | 语法 | | public SendDocPO getSendDocPOByID(String itemID) |
| 前置条件 | | Id正确 |
| 后置条件 | | 获取寄件单po |
| Collection.getSendDocsByID | 语法 | | public ArrayList<String> getSendDocsByID(String courier\_ID) |
| 前置条件 | | Id正确 |
| 后置条件 | | 获取该快递员的寄件单 |
| Collection.saveSendDocCreateGatheringDoc | 语法 | | public void saveSendDocCreateGatheringDoc(String courierID) |
| 前置条件 | | Id正确 |
| 后置条件 | | 重置快递员业绩 |
| Collection.getHistoryPO | 语法 | | public SendDocPO getSendDocPOByID(String itemID) |
| 前置条件 | | Id正确 |
| 后置条件 | | 获取寄件单po |
| Collection.getSendDocPOByID | 语法 | | ArrayList<HistoryTimePO> getHistoryPO(String sCity, String rCity) |
| 前置条件 | | 无 |
| 后置条件 | | 获取寄件历史时间记录 |
| Collection.confirmSave | 语法 | | public void confirmSave() |
| 前置条件 | | 已生成单据 |
| 后置条件 | | 确认保存 |
| 需要的服务 | | | |
| 服务名 | | 服务 | |
| CollectionDataService.saveSendDocPO(SendDocPO po) | | 保存寄件单单一持久化对象 | |
| CollectionDataService.getDistance(String city) | | 获取距离数据 | |
| CollectionDataService.getSequence() | | 获取序列号 | |
| CollectionDataService.changeSequence(string sequence) | | 改变最后序列号 | |
| CollectionDataService.getSendDocPOByID(String ID) | | 根据id获取寄件单 | |
| CollectionDataService.getAllSendDoc()t | | 获取未审批寄件单 | |
| CollectionDataService.saveCourierMoneyPO(CourierMoneyPO po | | 保存快递员业绩记录 | |
| CollectionDataService.getCourierMoneyPO(String courierID) | | 获取快递员业绩记录 | |

(4) 业务逻辑层的动态模型

图示表明了系统中，快递员输入寄件单相关信息后，揽件逻辑处理的相关对象之间的协作

(5) 业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由不同的逻辑对象提供

### 4.1.4 transferbl模块

1. 模块概述

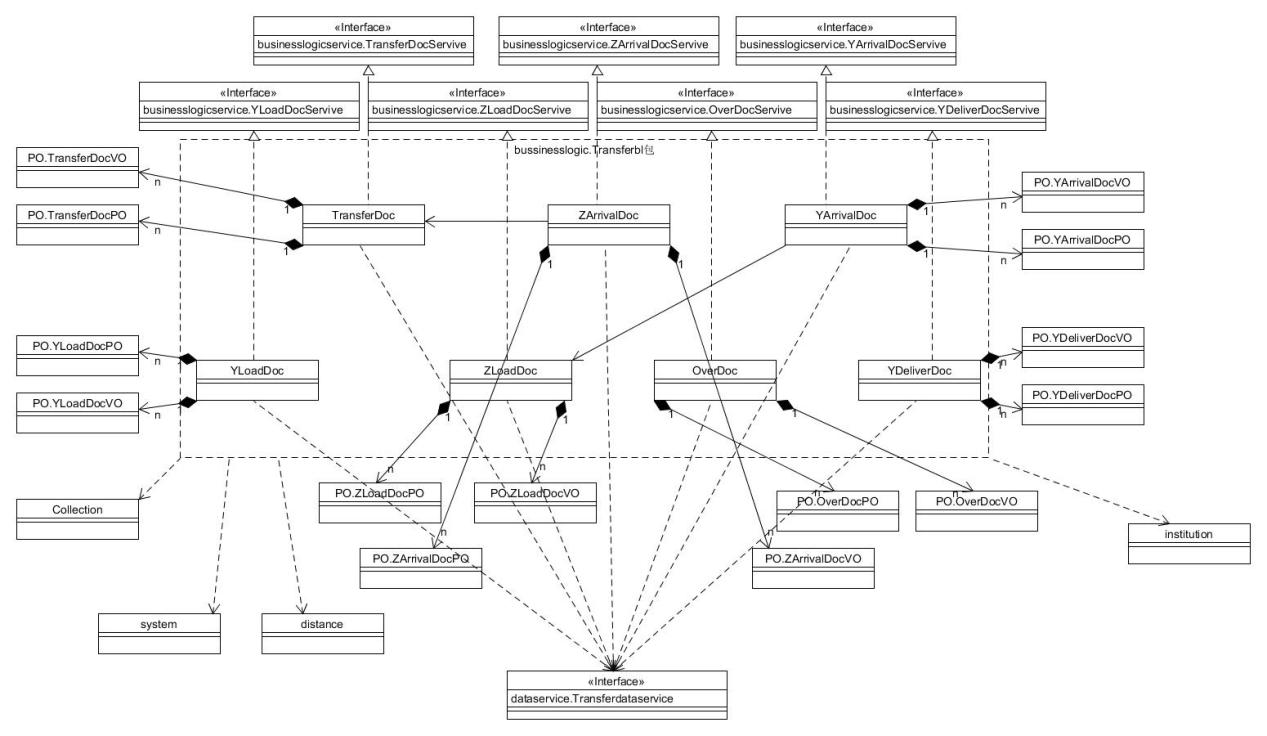
Transferbl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求

Transferbl模块的职责及接口参见软件体系结构描述文档

1. 整体结构

展示层、业务逻辑层、数据层间由TransferLogicService接口与TransferDataService分隔，业务逻辑根据不同操作对象，由TransferDoc、ZArrivalDoc等对象完成，TransferDocPO、ZArrivalDocPO等是各单据的持久化对象，TransferDocVO、ZArrivalVO等是各单据的可视化对象。

Transferbl模块设计如图所示



1. 模块内部类的接口规范

TransferDoc的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 供接口 | | | | |
| TransferDoc.getUncheckedTransferDoc | 语法 | | Public TransferDocPO getAllTransferDoc() | |
| 前置条件 | | 无 | |
| 后置条件 | | 得到待审批中转单 | |
| TransferDoc.confirmSave | 语法 | | public void confirmSave() |
| 前置条件 | | 已生成单据 |
| 后置条件 | | 确认保存 |
| Collection.getTransferDocSequence | 语法 | | public String getTransferDocSequence() |
| 前置条件 | | 无 |
| 后置条件 | | 得到TransferDoc序号 |
| Collection.changeTransferDocSequence | 语法 | | public void changeTransferDocSequence() |
| 前置条件 | | TransferDoc已保存 |
| 后置条件 | | 更改序号 |
|  | | | | |
| 需要的服务 | | | | |
| 服务名 | | 服务 | | |
| TransferDataService.saveTransferDocPO(TransferDocPO po) | | 保存中转单单一持久化对象 | | |
| TransferDataService.getTransferSequence() | | 获取中转单序列号 | | |
| TransferDataService.changeTransferSequence(int TransferSequence) | | 改变中转单最后序列号 | | |
| TransferDataService.getTransferDocPO(int TransferDocID) | | 根据ID查找中转单单一持久化对象 | | |
| TransferDataService.getUnpaidTransferDocPO() | | 获取未付款中转单 | | |
|  | |  | | |

YDeliverDoc的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 供接口 | | | | |
| YDeliverDoc.getUncheckedYDeliverDoc | 语法 | | Public YDeliverDocPO DegetYDeliverDoc() | |
| 前置条件 | | 得到派件单 | |
| 后置条件 | | 无 | |
| YDeliverDoc.confirmSave | 语法 | | public void confirmSave() |
| 前置条件 | | 已生成单据 |
| 后置条件 | | 确认保存 |
| Collection.getYDeliverDocSequence | 语法 | | public String getYDeliverDocSequence() |
| 前置条件 | | 无 |
| 后置条件 | | 得到YDeliverDoc序号 |
| Collection.changeYDeliverDocSequence | 语法 | | public void changeYDeliverDocSequence() |
| 前置条件 | | YDeliverDoc已保存 |
| 后置条件 | | 更改序号 |
|  | | | | |
| 需要的服务 | | | | |
| 服务名 | | 服务 | | |
| TransferDataService.saveYDeliverDocPO(YDeliverDocPO po) | | 保存派件单单一持久化对象 | | |
| TransferDataService.getYDeliverSequence() | | 获取派件单序列号 | | |
| TransferDataService.changeYDeliverSequence(int YDeliverSequence) | | 改变派件单最后序列号 | | |
| TransferDataService.getYDeliverDocPO(int DeliverDocID) | | 根据ID查找派件单单一持久化对象 | | |
|  | |  | | |

OverDoc的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 供接口 | | | | |
| OverDoc.getUncheckedOverDoc | 语法 | | Public OverDocPO getUncheckedOverDoc() | |
| 前置条件 | | 得到收件单 | |
| 后置条件 | | 无 | |
| OverDoc.confirmSave | 语法 | | public void confirmSave() |
| 前置条件 | | 已生成单据 |
| 后置条件 | | 确认保存 |
| Collection.getOverDocSequence | 语法 | | public String getOverDocSequence() |
| 前置条件 | | 无 |
| 后置条件 | | 得到OverDoc序号 |
| Collection.changeOverDocSequence | 语法 | | public void changeOverDocSequence() |
| 前置条件 | | OverDoc已保存 |
| 后置条件 | | 更改序号 |
|  | | | | |
| 需要的服务 | | | | |
| 服务名 | | 服务 | | |
| TransferDataService.saveOverDocPO(OverDocPO po) | | 保存收件单单一持久化对象 | | |
| TransferDataService.getOverSequence() | | 获取收件单序列号 | | |
| TransferDataService.changeOverSequence(int OverSequence) | | 改变收件单最后序列号 | | |
| TransferDataService.getOverDocPO(int OverDocID) | | 根据ID查找收件单单一持久化对象 | | |
|  | |  | | |

YLoadDoc的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 供接口 | | | | |
| YLoadDoc.getUncheckedYloadDoc | 语法 | | Public YloadDocPO getAllYloadDoc() | |
| 前置条件 | | 无 | |
| 后置条件 | | 得到待审批营业厅装车单 | |
| YLoadDoc.confirmSave | 语法 | | public void confirmSave() |
| 前置条件 | | 已生成单据 |
| 后置条件 | | 确认保存 |
| Collection.getYLoadDocSequence | 语法 | | public String getYLoadDocSequence() |
| 前置条件 | | 无 |
| 后置条件 | | 得到YLoadDoc序号 |
| Collection.changeYLoadDocSequence | 语法 | | public void changeYLoadDocSequence() |
| 前置条件 | | YLoadDoc已保存 |
| 后置条件 | | 更改序号 |
|  | | | | |
| 需要的服务 | | | | |
| 服务名 | | 服务 | | |
| TransferDataService.saveYloadDocPO(ZloadDocPO po) | | 保存装车单单一持久化对象 | | |
| TransferDataService.getYloadSequence() | | 获取装车单序列号 | | |
| TransferDataService.changeYloadSequence(int YloadSequence) | | 改变装车单最后序列号 | | |
| TransferDataService.getYloadDocPO(int YloadDocID) | | 根据ID查找装车单单一持久化对象 | | |
| TransferDataService.getUnpaidYloadDoc() | | 获取未付款装车单 | | |
|  | |  | | |

ZLoadDoc的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 供接口 | | | | |
| ZLoadDoc.getUncheckedZloadDoc | 语法 | | Public ZloadDocPO getAllZloadDoc() | |
| 前置条件 | | 无 | |
| 后置条件 | | 得到待审批营业厅装车单 | |
| ZLoadDoc.confirmSave | 语法 | | public void confirmSave() |
| 前置条件 | | 已生成单据 |
| 后置条件 | | 确认保存 |
| Collection.getZLoadDocSequence | 语法 | | public String getZLoadDocSequence() |
| 前置条件 | | 无 |
| 后置条件 | | 得到ZLoadDoc序号 |
| Collection.changeZLoadDocSequence | 语法 | | public void changeZLoadDocSequence() |
| 前置条件 | | ZLoadDoc已保存 |
| 后置条件 | | 更改序号 |
|  | | | | |
| 需要的服务 | | | | |
| 服务名 | | 服务 | | |
| TransferDataService.saveZloadDocPO(ZloadDocPO po) | | 保存装车单单一持久化对象 | | |
| TransferDataService.getZloadSequence() | | 获取装车单序列号 | | |
| TransferDataService.changeZloadSequence(int ZloadSequence) | | 改变装车单最后序列号 | | |
| TransferDataService.getZloadDocPO(int ZloadDocID) | | 根据ID查找装车单单一持久化对象 | | |
| TransferDataService.getUnpaidZloadDoc() | | 获取未付款装车单 | | |
|  | |  | | |

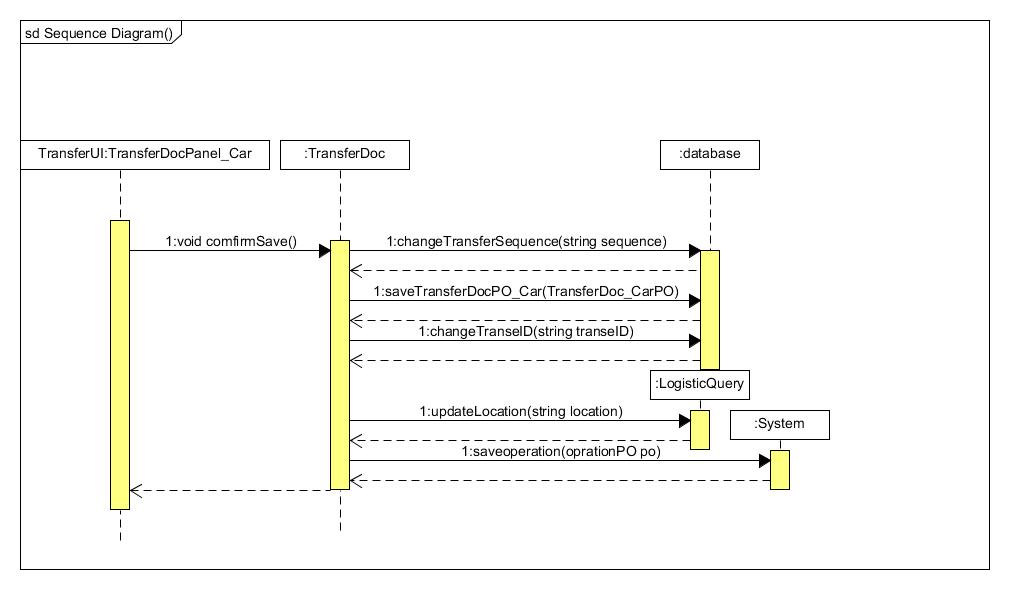
YArrivalDoc的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 供接口 | | | | |
| YArrivalDoc.getUncheckedYArrivalDoc | 语法 | | Public YArrivalDocPO getAllYArrivalDoc() | |
| 前置条件 | | 无 | |
| 后置条件 | | 得到待审批到达单 | |
| YArrivalDoc.confirmSave | 语法 | | public void confirmSave() |
| 前置条件 | | 已生成单据 |
| 后置条件 | | 确认保存 |
| Collection.getYArrivalDocSequence | 语法 | | public String getYArrivalDocSequence() |
| 前置条件 | | 无 |
| 后置条件 | | 得到YArrivalDoc序号 |
| Collection.changeYArrivalDocSequence | 语法 | | public void changeYArrivalDocSequence() |
| 前置条件 | | YarrivalDoc已保存 |
| 后置条件 | | 更改序号 |
|  | | | | |
| 需要的服务 | | | | |
| 服务名 | | 服务 | | |
| TransferDataService.saveYArrivalDocPO(YArrivalDocPO po) | | 保存营业厅到达单单一持久化对象 | | |
| TransferDataService.getYArrivalSequence() | | 获取营业厅到达单序列号 | | |
| TransferDataService.changeYArrivalSequence(int YArrivalSequence) | | 改变营业厅到达单最后序列号 | | |
| TransferDataService.getYArrivalDocPO(int YArrivalDocID) | | 根据ID查找营业厅到达单单一持久化对象 | | |
|  | |  | | |

ZArrivalDoc的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 供接口 | | | | |
| ZArrivalDoc.getUncheckedZArrivalDoc | 语法 | | Public ZArrivalDocPO getAllZArrivalDoc() | |
| 前置条件 | | 无 | |
| 后置条件 | | 得到待审批中转中心到达单 | |
| ZArrivalDoc.confirmSave | 语法 | | public void confirmSave() |
| 前置条件 | | 已生成单据 |
| 后置条件 | | 确认保存 |
| Collection.getZArrivalDocSequence | 语法 | | public String getSequence() |
| 前置条件 | | 无 |
| 后置条件 | | 得到ZArrivalDoc序号 |
| Collection.changeZArrivalDocSequence | 语法 | | public void changeSequence() |
| 前置条件 | | ZArrivalDoc已保存 |
| 后置条件 | | 更改序号 |
|  | | | | |
| 需要的服务 | | | | |
| 服务名 | | 服务 | | |
| TransferDataService.saveZArrivalDocPO(ZArrivalDocPO po) | | 保存中转中心到达单单一持久化对象 | | |
| TransferDataService.getZArrivalSequence() | | 获取中转中心到达单序列号 | | |
| TransferDataService.changeZArrivalSequence(int ZArrivalSequence) | | 改变中转中心到达单最后序列号 | | |
| TransferDataService.getZArrivalDocPO(int ZArrivalDocID) | | 根据ID查找中转中心到达单单一持久化对象 | | |
|  | |  | | |

1. 业务逻辑层的动态模型

图示表明了系统中，中转中心业务员选择汽运中转，输入到达地，车次号，监装员和所有单号后，揽件逻辑处理的相关对象之间的协作

1. 业务逻辑层的设计原理

利用分散式控制风格，每个操作不同对象的业务逻辑由不同的逻辑对象提供

### approvelbl模块

1. 模块概述

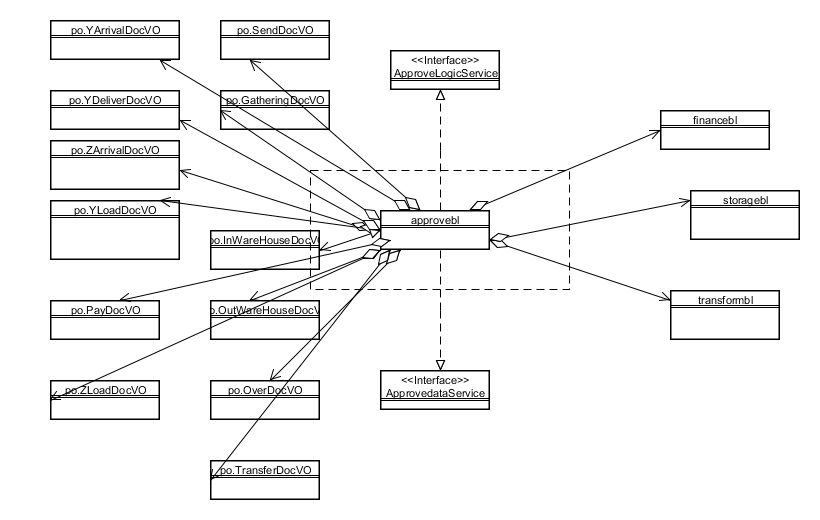
approvebl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求

approvebl模块的职责及接口参见软件体系结构描述文档

1. 整体结构

展示层、业务逻辑层、数据层间由approveLogicService接口与approvedataService分隔，业务逻辑完全由approvebl对象完成，

approvebl模块设计如图所示



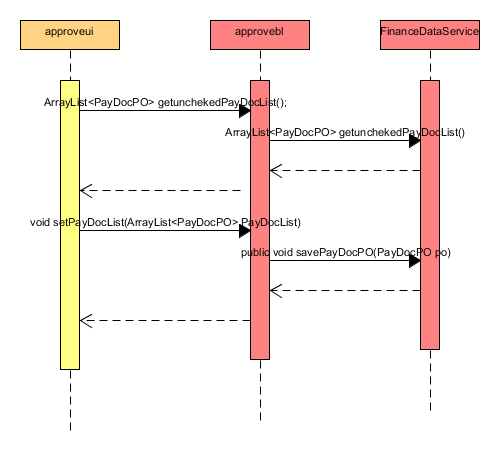
1. 模块内部类的接口规范

approvebl的接口规范

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| StorageTools.getOutWarehouseDocList | 得到出库单 |
| StorageTools.getInWarehouseDocList | 得到入库单 |
| YDeliverDoc.getuncheckedDeliverDocPOs | 得到派件单 |
| YArrivalDoc.getuncheckedYArrivalDocPOs | 得到营业厅到达单 |
| ZArrivalDoc.getuncheckedZArrivalDocPOs | 得到中转中心到达单 |
| TransferDoc.getuncheckedTransferDocPOs | 得到中转单 |
| OverDoc.getUncheckedOverDocPOs | 得到收件单 |
| YLoadDoc.getuncheckedYloadDocPOs | 得到营业厅装车单 |
| ZLoadDoc.getuncheckedZloadPOs | 得到中转中心装车单 |
| Collectionbl.getAllSendDoc() | 得到寄件单 |
| Financebl.getuncheckedPayDocList | 得到付款单 |
| Financebl.getuncheckedGatheringDocList | 得到收款单 |
| Collectionbl.saveSendDocPO(SendDocPO) | 修改寄件单的审批状态 |
| Financebl.saveGatheringDocPO(GatheringDocPO po) | 修改收款单的审批状态 |
| StorageTools.saveInWareHouseDocPO(InWareHouseDocPO) | 修改入库单的审批状态 |
| StorageTools.saveOutWareHouseDocPO(OutWareHouseDocPO) | 修改出库单的审批状态 |
| TransferDoc.saveTransferDocPO(TransferDocPO po) | 修改中转单的审批状态 |
| OverDoc.saveOverDocPO(OverDocPO) | 修改收件单的审批状态 |
| Financebl.savePayDocPO(PayDocPO) | 修改付款单的审批状态 |
| YArrivalDoc.saveYArrivalDocPO(YArrivalDocPO) | 修改营业厅到达单的审批状态 |
| YDeliverDoc.saveYDeliverDocPO(YDeliverDocPO ) | 修改派件单的审批状态 |
| ZArrivalDoc.saveZArrivalDocPO(ZArrivalDocPO) | 修改中转中心到达单的审批状态 |
| YLoadDoc.saveYLoadDocPO(YLoadDocPO) | 修改营业厅装车单的审批状态 |
| ZLoadDoc.saveZLoadDocPO(ZLoadDocPO) | 修改中转中心装车单的审批状态 |

1. 业务逻辑层的动态模型

表明了在系统中当总经理需要审批单据的时候，一旦定下了审批单据的类别，就直接在approvebl中获得其未审批的列表，而approvebl中的未审批列表则是从负责该单据的模块中得到



1. 业务逻辑层的设计原理

利用集中式控制风格，每个业务逻辑都由一个类来集中完成

### financebl模块

1. 模块概述

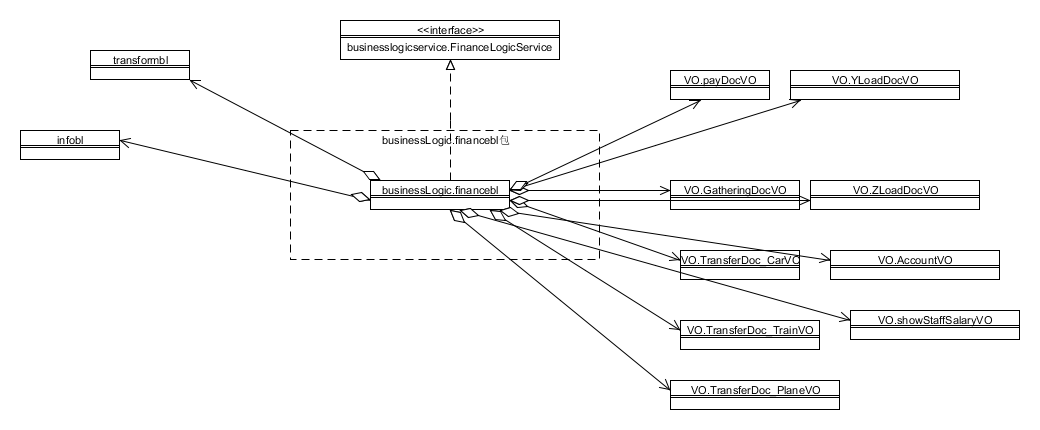
financebl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求

financebl模块的职责及接口参见软件体系结构描述文档

1. 整体结构

展示层、业务逻辑层、数据层间由finanaceLogicService接口与financeDataService分隔，业务逻辑完全由financebl对象完成

financebl模块设计如图所示

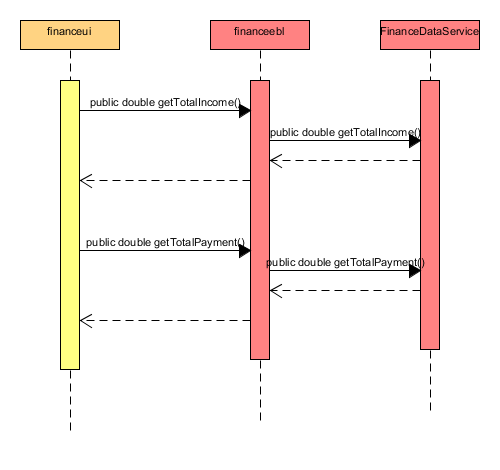


1. 模块内部类的接口规范
2. **表16 Financebl的接口规范**

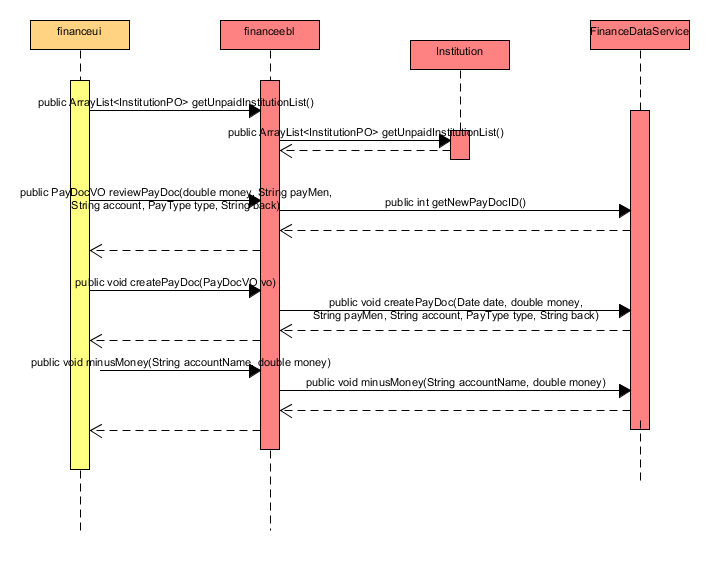
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Financebl.getuncheckedPayDocList() | 语法 | public ArrayList<PayDocPO> getuncheckedPayDocList(); |
|  | 前置条件 | 总经理已经被识别和授权 |
|  | 后置条件 | 总经理得到未审批的付款单列表 |
| Financebl.getuncheckedPayDocList() | 语法 | Public ArrayList<GatheringDocPO>getuncheckedGatheringDocList() |
|  | 前置条件 | 总经理已经被识别和授权 |
|  | 后置条件 | 总经理得到未审批的收款单列表 |
| Financebl.savePayDocPO(PayDocPO ) | 语法 | Public void savePayDocPO(PayDocPO po) |
|  | 前置条件 | 总经理已经被识别和授权 |
|  | 后置条件 | 修改单据的审批状态 |
| Financebl.saveGatheringDocPO(GatheringDocPO) | 语法 | Public void saveGatheringDocPO(GatheringDocPO po) |
|  | 前置条件 | 总经理已经被识别和授权 |
|  | 后置条件 | 修改单据的审批状态 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| DataFactory.getFinanceDataService() | 通过rmi得到数据层的服务 | |
| FinanceDataServiec.getTotalIncome() | 得到公司截至当前的总收入 | |
| Financedataservice.getTotalPayment() | 得到公司截至当前的总支出 | |
| Time.stringToDate | 将string转换为Date类型变量 | |
| FinanceDataServiece.getPayDoc(Date ,Date) | 得到期间的所有付款单对象 | |
| Financedataservice.getGatheringDoc(Date ,Date) | 得到期间所有的收款单对象 | |
| Institutin.getUnpaidInstitutionList | 得到所有尚未付款的机构 | |
| YLoadDoc.getUnpaidYLoadDocPOs | 得到所有未付款的营业厅装车单 | |
| ZLoadDoc.getUnpaidZLoadDocPOs | 得到所有未付款的中转中心装车单 | |
| Institution.getUnpaidStaffList() | 得到所有未支付工资的员工列表 | |
| Financedataservice.addAccountPO(String ) | 增加一个新的账户 | |
| Financedataservice.deleteAccountPO(String) | 删除一个账户 | |
| Financedataservice.modifyAccountPO(String,string) | 用新的账户名替换原账户名 | |
| Financedataservice.checkAccountPO(String,string) | 查询一个账户信息 | |
| Financedataservice.getunchekedPayDocList() | 得到未审批的付款单列表 | |
| Financedataservice.getunchekedGatheringDocList() | 得到未审批的收款单列表 | |
| TransferDoc.getUnpaidTransferDocPOs() | 得到所有未付款的中转单 | |
| Financedataservice.savePayDocPO(PayDocPO ) | 保存付款单的审批状态 | |
| Financedataservice.saveGatheringDocPO(GatheringDocPO) | 保存收款单的审批状态 | |
| Institution.saveRent(String) | 将特定机构设为已付款 | |
| Institution.saveStaff(StaffPO po) | 将员工设为已付款 | |
| Collectionbl.getSendDocsByID() | 根据ID得到寄件单 | |
| FinanceDataService.getNewGatheringDocID | 自动生成编号 | |
| FinanceDataService.createGatheringDoc(Date,double,string,ArrayLst<String>,String) | 生成收款单 | |
| Collectionbl.saveSendDocCreateGatheringDoc(String) | 保存寄件单对象 | |
| FinanceDataService.createPayDoc(Date,double,String,String,PayType,double) | 创建付款单 | |
| FinanceDataService.getPayDocPO(String) | 获得付款单对象 | |
| Institution.getStaffName(String) | 获取当前工作人员的名字 | |
| Institution.getPosition(String) | 获取员工的工作岗位（与计算工资的策略相关） | |
| Institution.getBase(String) | 获得基础工资 | |
| Institution.getBonus(String) | 获得奖金 | |
| InstItution.getPercentage(String) | 获得提成 | |
| FinanceDataService.getCourierMoney(String,Date) | 获取快递员的收款额 | |
| Driver,getDriverCommision(String) | 获得司机运送一次的工钱 | |
| YLoadDoc.getDriverTime(String) | 获得司机的运送次数 | |
| Institution.resetRent() | 重置租金 | |
| Institution.resetSalary() | 重置工资 | |
| FinanceDataServiece.getAccountList() | 获取所有的账户列表供财务人员选择 | |
| FinanceDataService.addMoney(String,double) | 根据收款情况修改账户的余额 | |
| FinanceDataService.minusMoney(String,double) | 根据付款情况修改账户的余额 | |
| Institution.saveSalary(SalaryPO po) | 保存员工工资 | |

1. 业务逻辑层的动态模型

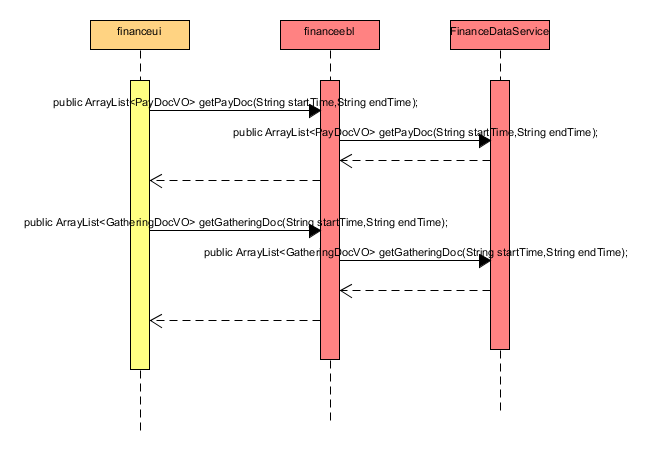
查看成本收益表



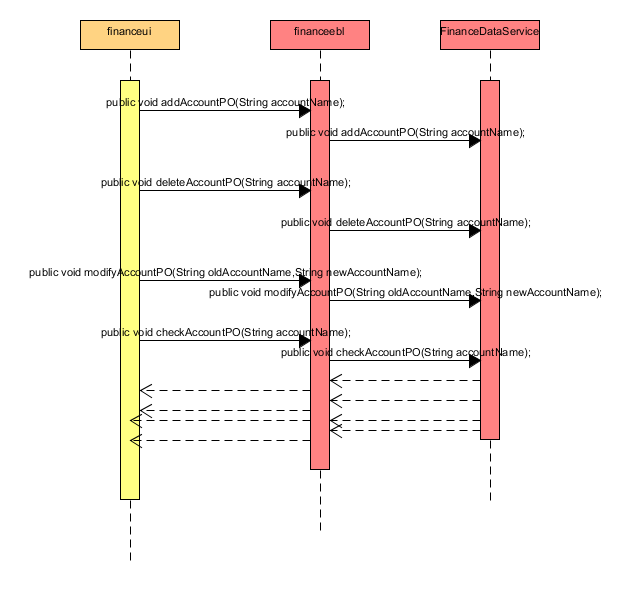
成本管理



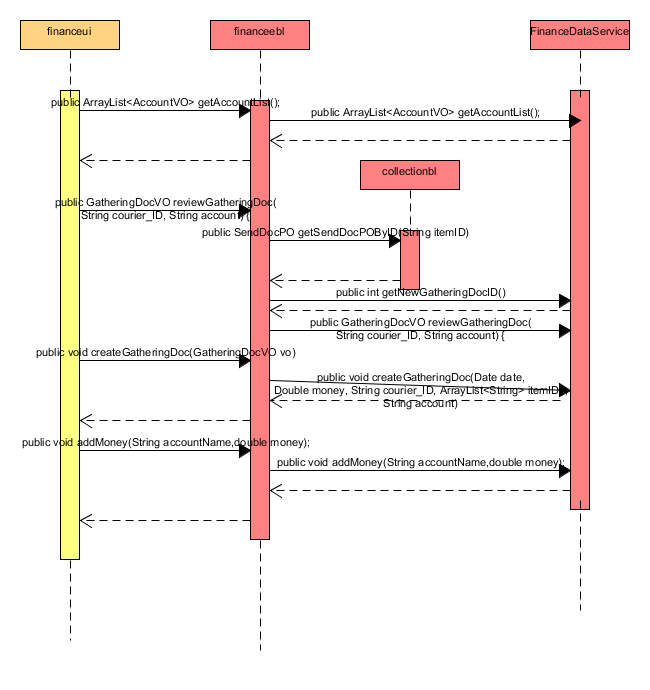
查看经营情况表



账户管理



生成收款单



1. 业务逻辑层的设计原理

利用集中式控制风格，每个业务逻辑都由一个类来集中完成

4.1.7 infobl模块

（1）模块概述

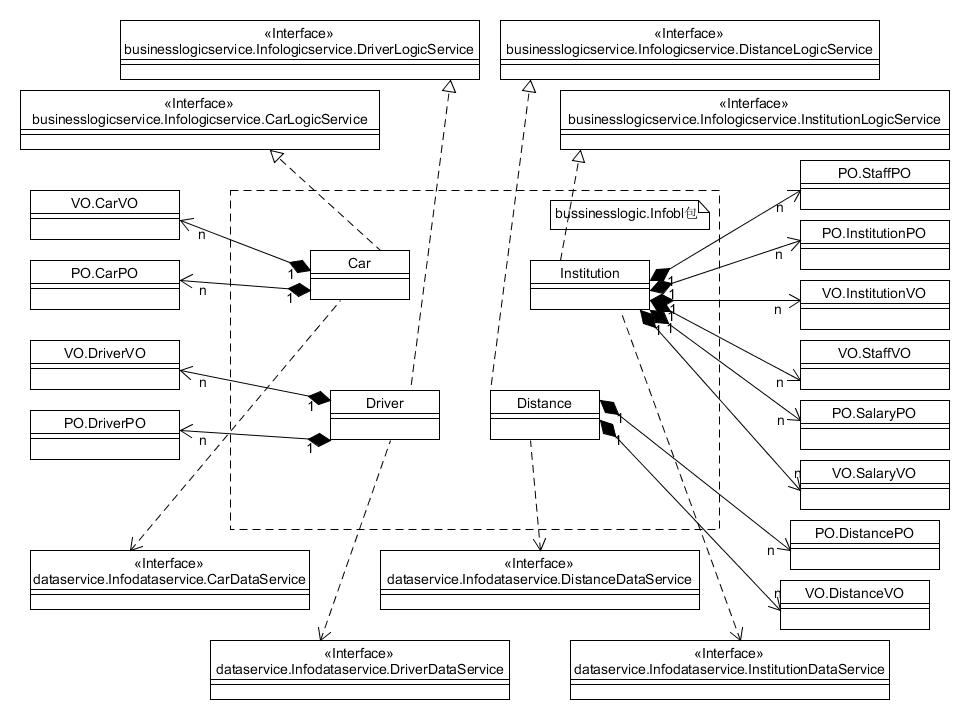
InfoBl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

InfoBl模块的职责及接口参见软件系统结构描述文档相关表格。

（2）整体结构

根据体系结构的设计，我们将系统分为展示层，业务逻辑层，数据层。每一层之间为了增加灵活性，我们添加了接口。业务逻辑层和展示层之间有businesslogicservice.InfoBlService接口，业务逻辑层和数据层之间添加dataservice.InfoDataService接口。DriverPO，CarPO，InstitutionPO，StaffPO,SalaryPO,DistancePO分别是对应的持久化对象。

模块的设计模式如图

模块内部类的接口规范

InfoBl.Car的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Infobl.Car.saveCar | 语法 | | public void saveCar(CarPO po) |
| 前置条件 | | 无 |
| 后置条件 | | 保存新增或修改的车辆信息 |
| Infobl.Car.getCarList | 语法 | | public ArrayList<CarVO> getCarList(String institutionName) |
| 前置条件 | | 机构名称正确 |
| 后置条件 | | 得到车辆信息的VO列表 |
| Infobl.Car.deleteCar | 语法 | | public void deleteCar(CarPO po) |
| 前置条件 | | 无 |
| 后置条件 | | 删除指定的车辆信息 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| InfoDataService.Car.findCar(String institutionName) | | 获得车辆信息列表 | |
| InfoDataService.Car.addCar(CarPO po) | | 新增一个车辆PO | |
| InfoDataService.Car.changeCar(CarPO po) | | 修改一个车辆PO | |
| InfoDataService.Car.deleteCar(CarPO po) | | 删除一个车辆PO | |
| InfoDataService.Car.isVaild(String id,String institutionName) | | 判断该车辆是否存在 | |

nfoBl.Driver的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Infobl.Driver.saveDriver | 语法 | public void saveDriver(DriverPO po) |
| 前置条件 | 无 |
| 后置条件 | 保存新增或修改的司机信息 |
| Infobl.Driver.getDriveVOList | 语法 | Public ArrayList<DriverVO> getDriveVOList(String institutionName) |
| 前置条件 | 机构名称正确 |
| 后置条件 | 得到司机的VO列表 |
| Infobl.Driver.deleteDriver | 语法 | public void deleteDriver(DriverPO po) |
| 前置条件 | 无 |
| 后置条件 | 删除指定的司机信息 |
| Infobl.Driver.saveDriverSalary | 语法 | public void saveDriverSalary(String id) |
| 前置条件 | 正确的id信息 |
| 后置条件 | 保存司机薪水策略 |
| Infobl.Driver.getDriverCommision | 语法 | double getDriverCommision(String driverID) |
| 前置条件 | 正确的司机id |
| 后置条件 | 得到司机的薪水策略 |
| Infobl.Driver.getDirverID | 语法 | public ArrayList<String> getDirverID(String institutionName) |
| 前置条件 | 正确的机构名称 |
| 后置条件 | 得到该机构所有司机id的列表 |
| Infobl.Driver.getDirverName | 语法 | public String getDirverName(String driverID, String institutionName) |
| 前置条件 | 正确的司机id和机构名称 |
| 后置条件 | 得到该机构里该id对应的司机姓名 |
| Infobl.Driver.getDriver | 语法 | public DriverPO getDriver(String driverID) |
| 前置条件 | 正确的司机id |
| 后置条件 | 得到该id对应的司机PO |
| Infobl.Driver.saveMoney | 语法 | public void saveMoney(Double money,String id) |
| 前置条件 | 司机id正确 |
| 后置条件 | 保存司机的薪水策略 |
| 需要的服务（需接口） | | |
| 服务名 | | 服务 |
| InfoDataService.Driver.findDriver(String InstitutionName) | | 获得司机信息列表 |
| InfoDataService.Driver.addDriver(DriverPO po) | | 新增一个司机PO |
| InfoDataService.Driver.changeDriver(DriverPO po) | | 修改一个司机PO |
| InfoDataService.Driver.deleteDriver(DriverPO po) | | 删除一个司机PO |
| InfoDataService.Driver.isVaild(String id,String institutionName) | | 判断该司机是否存在这个机构中 |
| InfoDataService.Driver.getDriver(String id) | | 根据司机id获得一个司机信息 |
| InfoDataService.Driver.saveMoney(Double money,String id) | | 保存司机的薪水信息 |

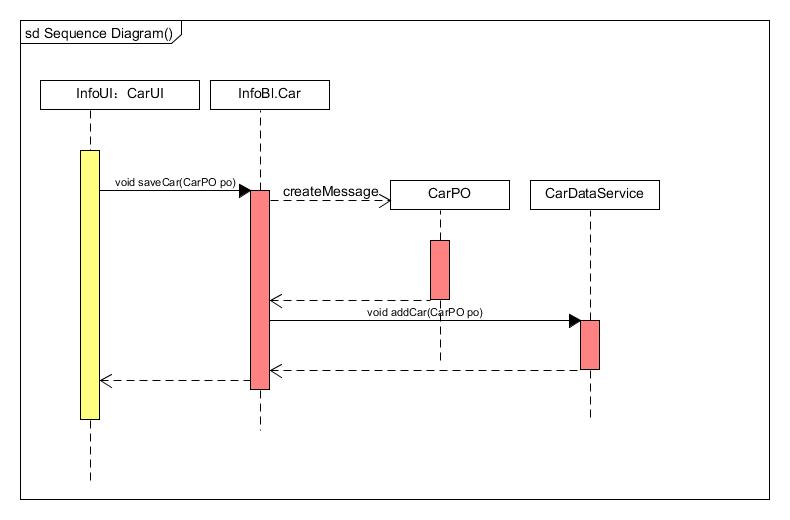
InfoBl.Distance的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Infobl.Distance.getDistance | 语法 | public double getDistance(String city1,String city2) |
| 前置条件 | 城市存在 |
| 后置条件 | 得到这两个城市间的距离 |
| Infobl.Distance.changeDistance | 语法 | public void changeDistance(DistancePO po) |
| 前置条件 | 无 |
| 后置条件 | 保存修改的城市PO |
| Infobl.Distance.addDistance | 语法 | public void addDistance(DistancePO po) |
| 前置条件 | 无 |
| 后置条件 | 新增一个城市时保存PO |
| Infobl.Distance.getCityList | 语法 | public ArrayList<String> getCityList() |
| 前置条件 | 无 |
| 后置条件 | 得到所有城市的列表 |
| 需要的服务（需接口） | | |
| 服务名 | | 服务 |
| InfoDataService.Distance.getDistance(String city1,String city2) | | 获得两个城市间的距离 |
| InfoDataService.Distance.changeDistance(DistancePO po) | | 修改一个距离PO |
| InfoDataService.Distance.addDistance(DistancePO po) | | 新增一个城市PO |
| InfoDataService.Distance.getCityList() | | 获得所有城市的列表 |

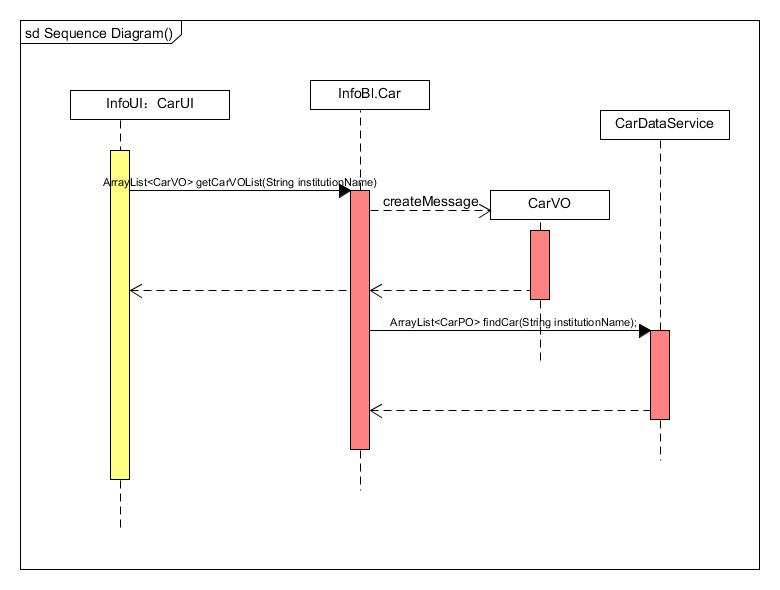
InfoBl.Institution的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
|  | 后置条件 | 得到所有城市的列表 |
| Infobl.Institution.getStaffVOList | 语法 | public ArrayList<StaffVO> getStaffVOList(String institutionName) |
| 前置条件 | 机构名称正确 |
| 后置条件 | 得到该机构的所有人员的VO列表 |
| Infobl.Institution.getInstitutionVOList | 语法 | public ArrayList<InstitutionVO> getInstitutionVOList() |
| 前置条件 | 无 |
| 后置条件 | 得到所有机构的VO列表 |
| Infobl.Institution.saveSalary | 语法 | public void saveSalary (SalaryPO po) |
| 前置条件 | 无 |
| 后置条件 | 保存员工薪水PO |
| Infobl.Institution.getSalary | 语法 | public SalaryPO getSalary(String staffID) |
| 前置条件 | 正确的员工id |
| 后置条件 | 得到该员工的薪水策略 |
| Infobl.Institution.saveStaff | 语法 | public void saveStaff(StaffPO po) |
| 前置条件 | 无 |
| 后置条件 | 保存新增或修改的员工PO |
| Infobl.Institution.saveInstitution | 语法 | public void saveInstitution (InstitutionPO po) |
| 前置条件 | 无 |
| 后置条件 | 保存新增或修改的机构信息 |
| Infobl.Institution.deleteInstitution | 语法 | public void deleteInstitution(String Institution) |
| 前置条件 | 正确的机构id |
| 后置条件 | 删除该id对应的机构信息 |
| Infobl.Institution.deleteStaff | 语法 | public void deleteStaff(StaffPO po) |
| 前置条件 | 无 |
| 后置条件 | 删除一个员工PO |
| Infobl.Institution.getCourierID | 语法 | public ArrayList<String> getCourierID(String institutionID) |
| 前置条件 | 正确的机构id |
| 后置条件 | 得到该机构快递员的id列表 |
| Infobl.Institution.getTransferCenterID | 语法 | public String getTransferCenterID(String institutionID) |
| 前置条件 | 正确的机构id |
| 后置条件 | 根据营业厅的id得到上级中转中心的id |
| Infobl.Institution.getSubInstitutionID | 语法 | public ArrayList<String> getSubInstitutionID(String institutionID) |
| 前置条件 | 正确的机构id |
| 后置条件 | 根据中转中心id得到营业厅的id列表 |
| Infobl.Institution.getID | 语法 | public String getID(String institutionName) |
| 前置条件 | 正确的机构名称 |
| 后置条件 | 得到机构id |
| Infobl.Institution.getName | 语法 | public String getName(String institutionId) |
| 前置条件 | 正确的机构id |
| 后置条件 | 根据机构id得到机构名称 |
| Infobl.Institution.getStaffName | 语法 | public String getStaffName(String staffid) |
| 前置条件 | 正确的员工id |
| 后置条件 | 得到员工的姓名 |
| Infobl.Institution.getInstitutionName | 语法 | public String getInstitutionName(String staffid) |
| 前置条件 | 正确的员工id |
| 后置条件 | 得到所在机构的名称 |
| Infobl.Institution.getInstitutionID | 语法 | public String getInstitutionID(String Staffid) |
| 前置条件 | 正确的员工id |
| 后置条件 | 得到所在机构的id |
| Infobl.Institution.getCity | 语法 | public String getCity(String institutionid) |
| 前置条件 | 正确的机构id |
| 后置条件 | 得到该机构所在城市信息 |
| Infobl.Institution.getUnpaidInstitutionList | 语法 | public ArrayList<InstitutionPO> getUnpaidInstitutionList() |
| 前置条件 | 无 |
| 后置条件 | 得到没有付租金的机构列表 |
| Infobl.Institution.getUnpaidStaffList | 语法 | public ArrayList<StaffPO> getUnpaidStaffList() |
| 前置条件 | 无 |
| 后置条件 | 得到没有付薪水的员工列表 |
| Infobl.Institution.resetSalary | 语法 | public void resetSalary() |
| 前置条件 | 无 |
| 后置条件 | 所有员工的属性改为未付款 |
| Infobl.Institution.saveSalary | 语法 | public void saveSalary(String id) |
| 前置条件 | 正确的员工id |
| 后置条件 | 该员工设为已付款 |
| Infobl.Institution.saveRent | 语法 | public void saveRent(String id) |
| 前置条件 | 正确的机构id |
| 后置条件 | 该机构设置为已付租金 |
| Infobl.Institution.resetRent | 语法 | public void resetRent() |
| 前置条件 | 无 |
| 后置条件 | 所有机构设置为未付款 |
| Infobl.Institution.getPosition | 语法 | Post getPosition(String staffID) |
| 前置条件 | 正确的员工id |
| 后置条件 | 获得该员工的职位 |
| Infobl.Institution.getBase | 语法 | double getBase(String staffID) |
| 前置条件 | 正确的员工id |
| 后置条件 | 得到该员工的基础工资 |
| Infobl.Institution.getBonus | 语法 | double getBonus(String staffID) |
| 前置条件 | 正确的员工id |
| 后置条件 | 得到该员工的奖金 |
| Infobl.Institution.getPercentage | 语法 | double getPercentage(String staffID) |
| 前置条件 | 正确的员工id |
| 后置条件 | 得到该员工的提成 |
| 需要的服务（需接口） | | |
| 服务名 | | 服务 |
| InfoDataService.Institution.getStaffList(String institutionId) | | 根据机构id获得机构中所有员工列表 |
| InfoDataService.Institution.addStaff(StaffPO po) | | 新增一个员工PO |
| InfoDataService.Institution.changeStaff(StaffPO po) | | 修改一个员工PO |
| InfoDataService.Institution.deleteStaff(StaffPO po) | | 删除一个员工PO |
| InfoDataService.Institution.findInstitution() | | 获得所有的机构信息列表 |
| InfoDataService.Institution.addInstitution (InstitutionPO po) | | 新增一个机构PO |
| InfoDataService.Institution.changeInstitution(InstitutionPO po) | | 修改一个机构PO |
| InfoDataService.Institution.deleteInstitution(InstitutionPO po) | | 删除一个机构PO |
| InfoDataService.Institution.getInstitution(String id) | | 根据机构id获得机构信息 |
| InfoDataService.Institution.getStaff(String id) | | 根据员工id获得员工信息 |
| InfoDataService.Institution.getSalaryPOs() | | 获得所有员工的薪水策略 |
| InfoDataService.Institution.getSalaryPO(String staffID) | | 根据员工id获得对应的薪水PO |
| InfoDataService.Institution.changeSalary(SalaryPO po) | | 修改一个薪水PO |
| InfoDataService.Institution.isStaffVaild(StaffPO po) | | 判断一个员工PO是否存在 |
| InfoDataService.Institution.isInstitutionVaild(InstitutionPO po) | | 判断一个机构PO是否存在 |

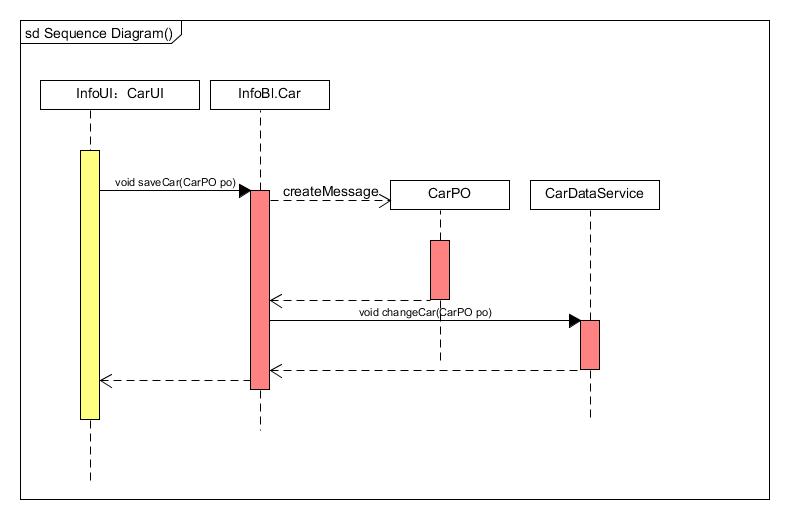
（4）业务逻辑层的动态模型



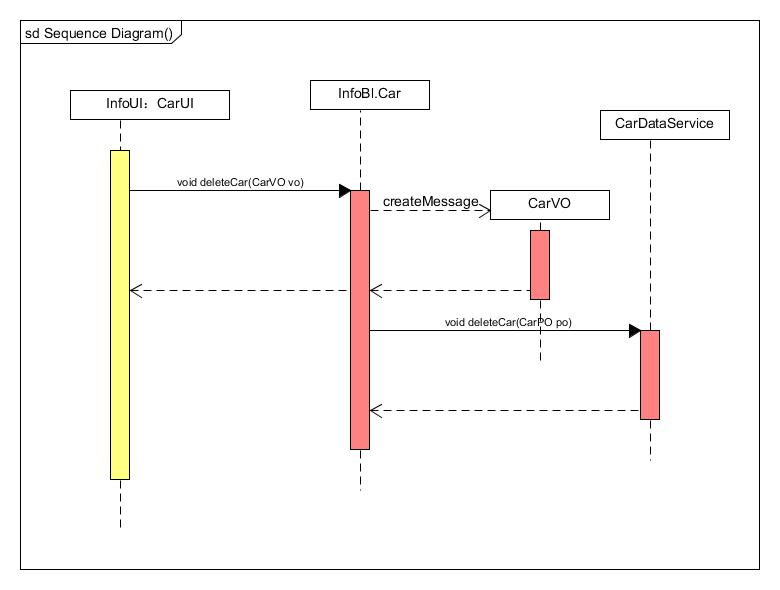
增加车辆信息的顺序图



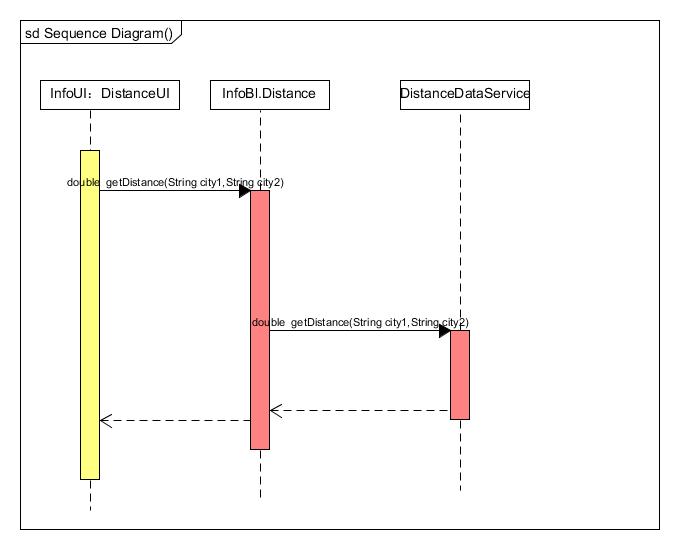
查询车辆信息的顺序图



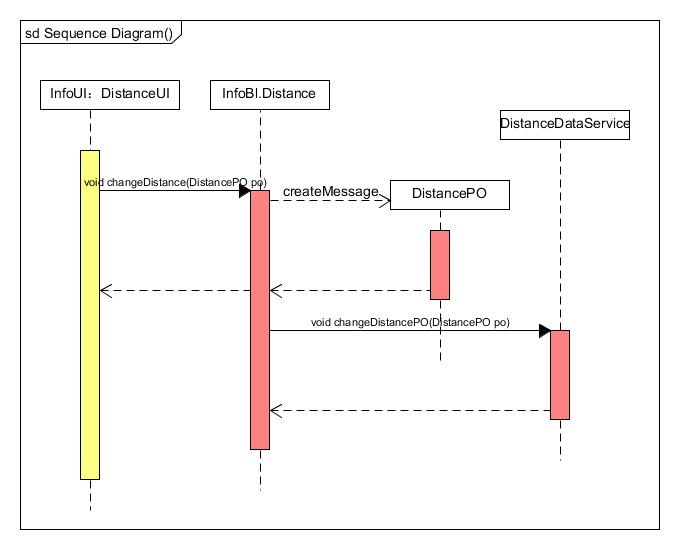
修改车辆信息的顺序图



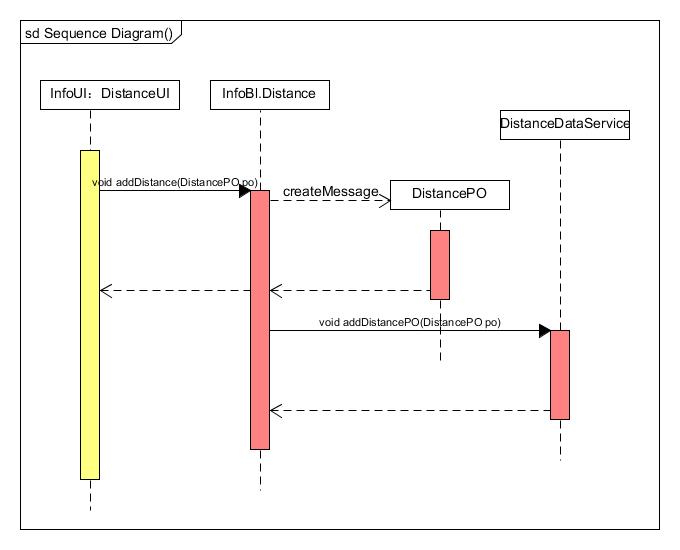
删除车辆信息的顺序图



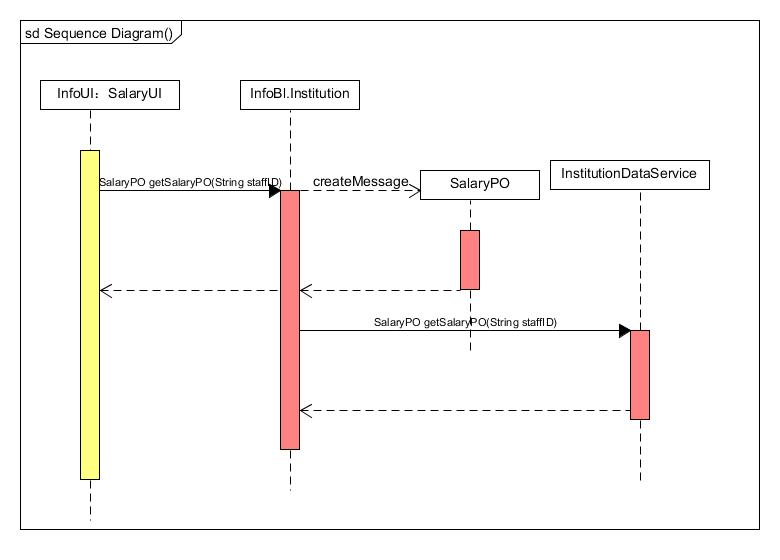
查看城市距离的顺序图



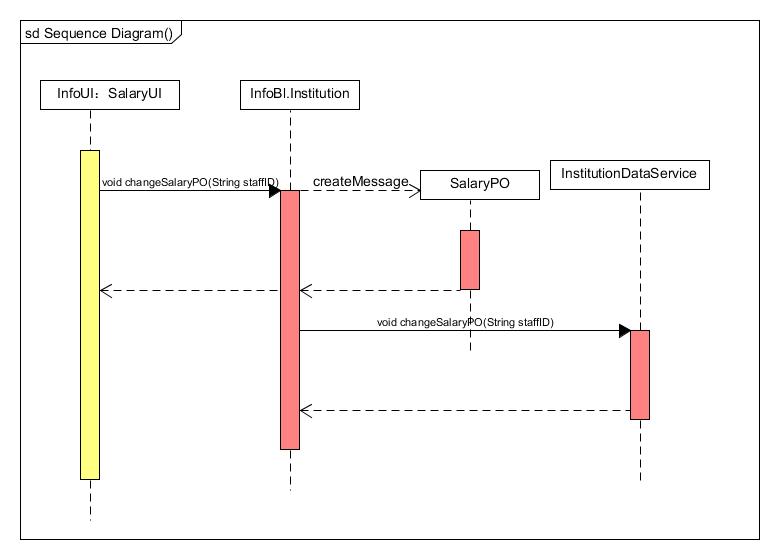
距离修改的顺序图



新增城市的顺序图



查询薪水策略的顺序图



薪水策略修改的顺序图

司机的增删改查，机构的增删改查，人员的增删改查与车辆的增删改查类似，参见车辆增删改查的顺序图。

（5）业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象。

4.1.8 systembl模块

（1）模块概述

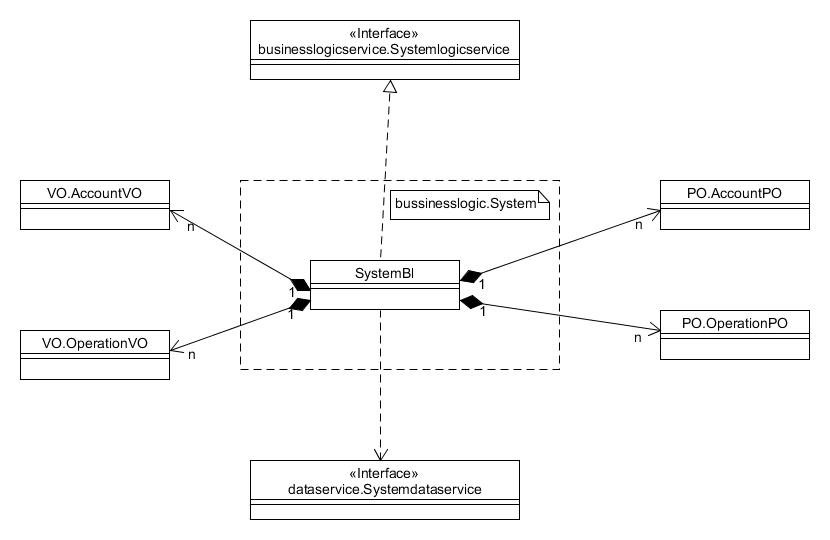
SystemBl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

SystemBl模块的职责及接口参见软件系统结构描述文档相关表格。

（2）整体结构

根据体系结构的设计，我们将系统分为展示层，业务逻辑层，数据层。每一层之间为了增加灵活性，我们添加了接口。业务逻辑层和展示层之间有businesslogicservice. SystemBlService接口，业务逻辑层和数据层之间添加dataserviceSystemDataService接口。OperationPO,AccountPO分别是对应的持久化对象。

模块的设计如图所示。

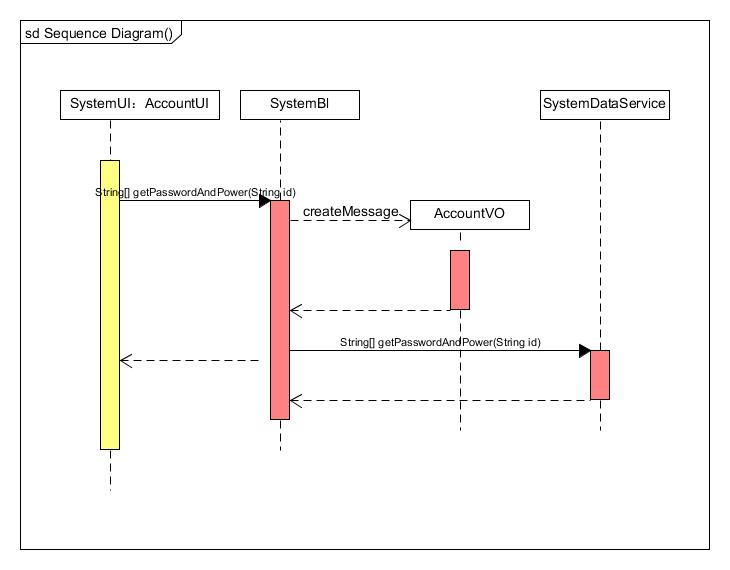


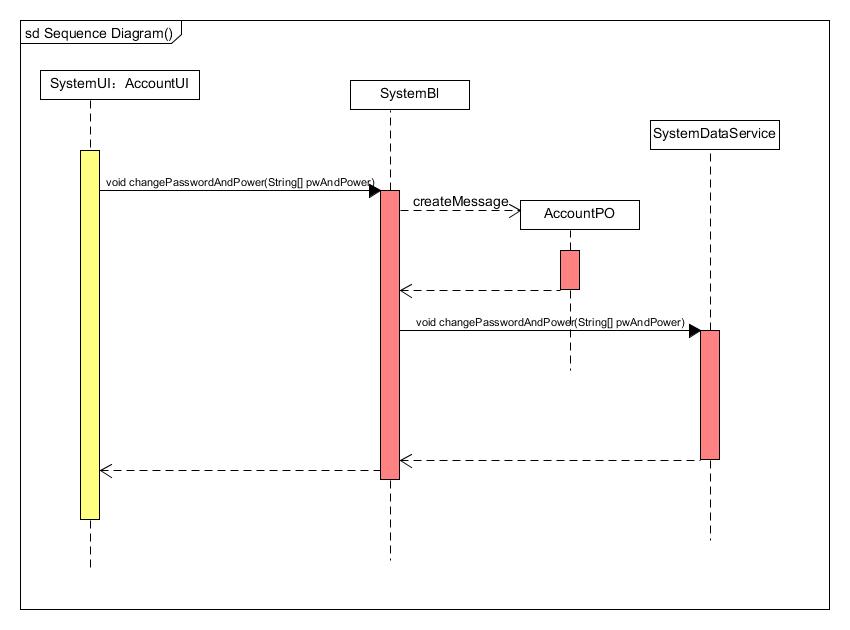
（3）模块内部类的接口规范

SystemBl接口规范

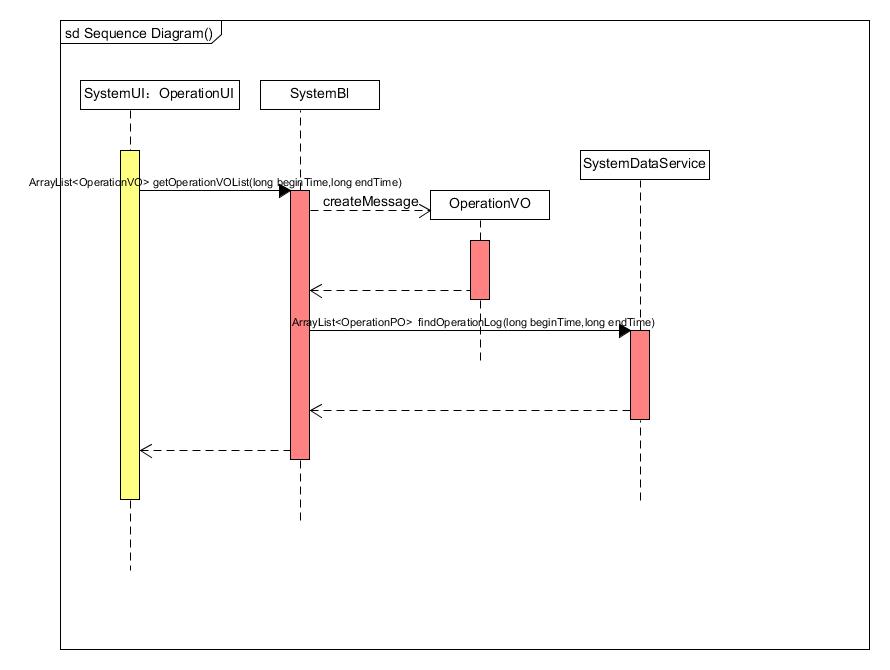
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Systembl.getOperationVOList | 语法 | public ArrayList<OperationVO> getOperationVOList(Date beginTime,Date endTime) |
| 前置条件 | 无 |
| 后置条件 | 得到该时间段内的操作日志VO列表 |
| Systembl.saveOperation | 语法 | public void saveOperation(OperationPO po) |
| 前置条件 | 无 |
| 后置条件 | 保存一个操作日志PO |
| Systembl.getPasswordAndPower | 语法 | public String[] getPasswordAndPower(String id) |
| 前置条件 | 正确的员工id |
| 后置条件 | 得到该员工的密码，权限 |
| Systembl.changePasswordAndPower | 语法 | public void changePasswordAndPower(String[] str) |
| 前置条件 | 无 |
| 后置条件 | 修改一个员工的密码权限 |
| 需要的服务（需接口） | | |
| 服务名 | | 服务 |
| SystemDataService.findOperationLog(Date beginTime,Date endTime) | | 获得一段时间内的操作日志 |
| SystemDataService.saveOperationLog(OperationPO po) | | 保存一个操作日志PO |
| SystemDataService.getPasswordAndPower(String id) | | 得到一个员工的密码权限信息 |
| SystemDataService.changePasswordAndPower(String[] str) | | 修改一个员工的密码权限 |

（4）业务逻辑层的动态模型

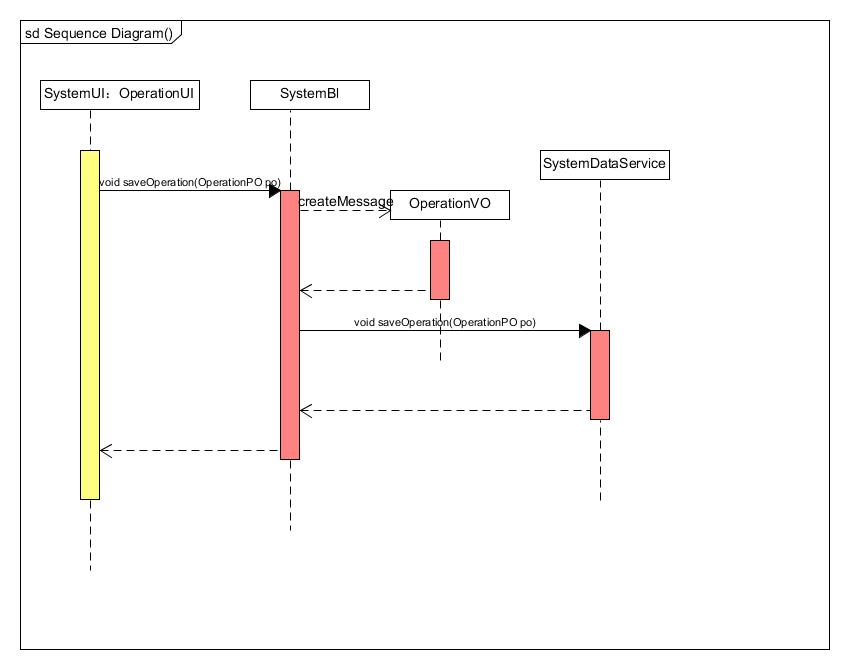
 查看账户权限的顺序图



修改账户权限的顺序图



查看系统日志的顺序图



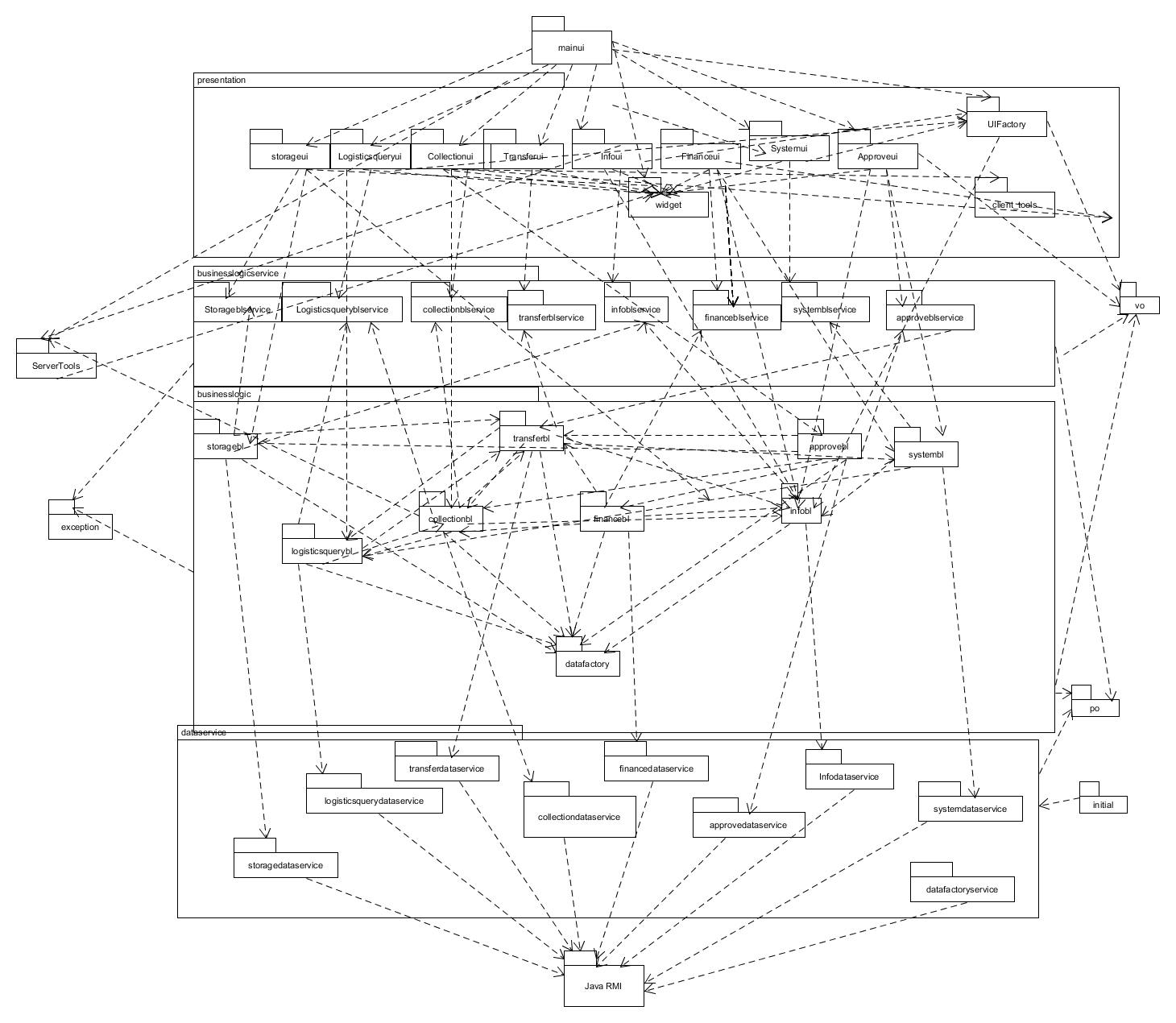
保存系统日志的顺序图

（5）业务逻辑层的设计原理

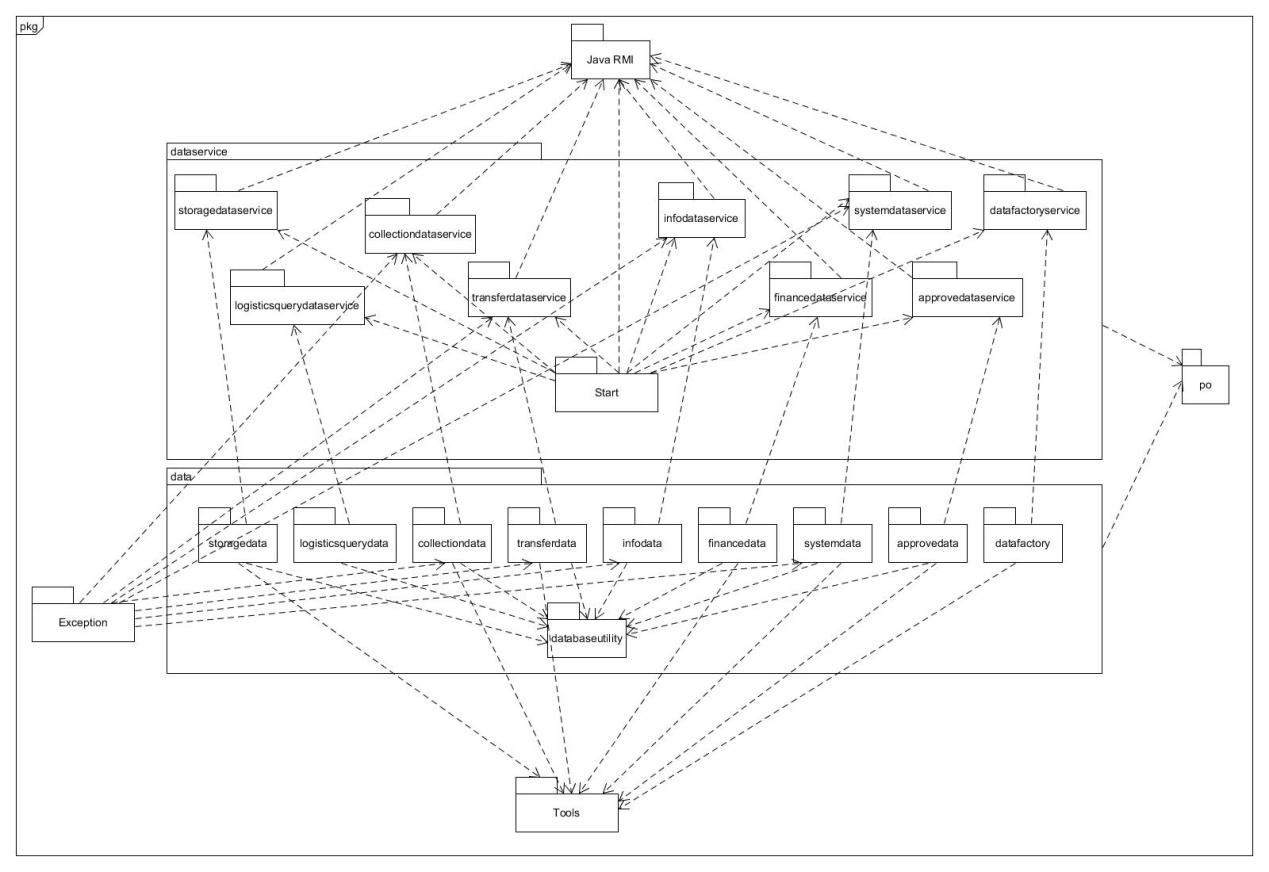
利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同的领域对象。

# 5. 依赖视角

客户端和服务器端各自包之间的依赖关系如下图所示。



客户端开发包图

 服务器开发包图