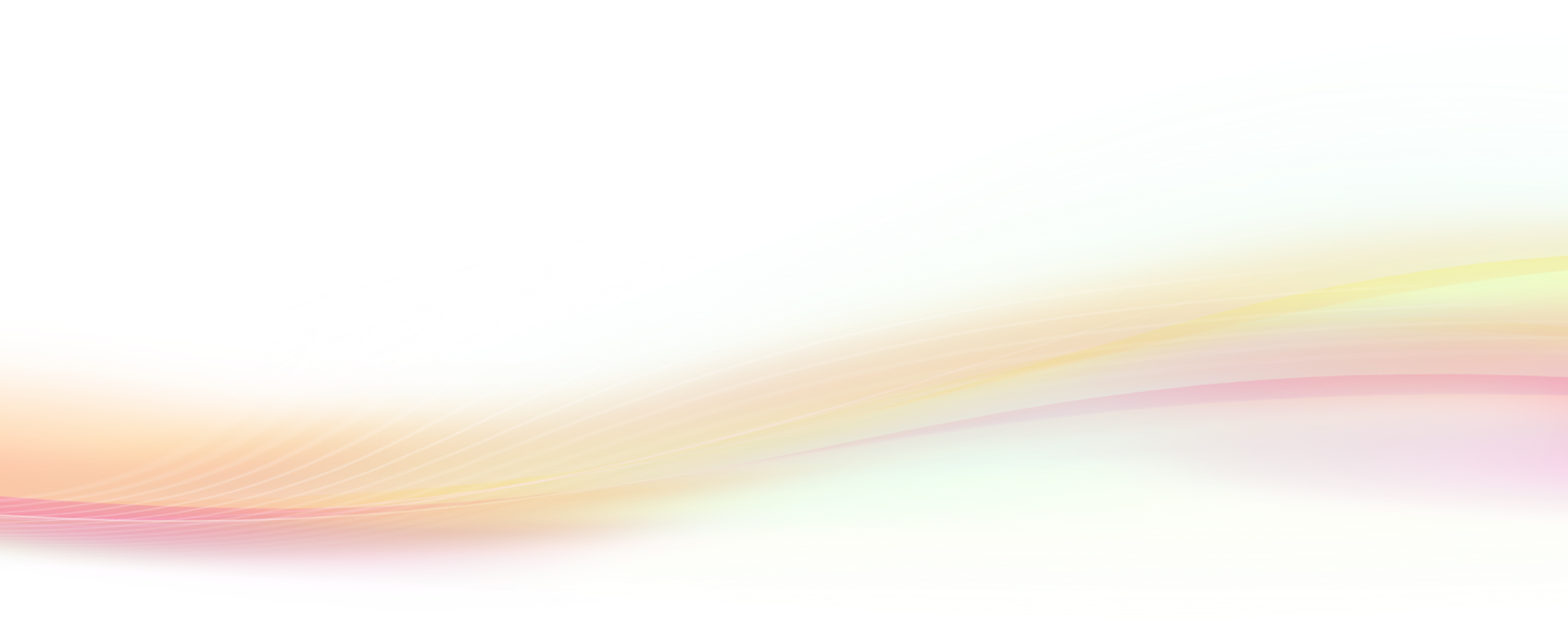
物流管理系统（LMS） 软件详细设计描述文档

编写者

崔浩 杜天蛟 高露 倪安松



# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 编写者 | 更新内容 | 版本号 |
| 2015/11/05 | 高露 | 初稿 | V1.0 |
| 2015/11/07 | 倪安松 | 添加界面顺序图 | V2.0 |
| 2015/11/15 | 倪安松 | 补充设计模型 | V3.0 |
| 2015/12/8 | 高露 | 代码完成后修正部分文档 | V4.0 |

目录

更新历史 2

1.引言 4

1.1编制目的 4

1.2词汇表 4

1.3参考资料 4

2.产品概述 5

3.体系结构设计概述 5

3.1 逻辑视角 5

3.2开发包图 7

4.人机交互设计 9

4.1 界面分解示意图 9

4.2界面跳转顺序图 11

4.2.1 财务人员界面跳转顺序图 11

4.2.2查询物流信息界面跳转顺序图 12

4.2.3管理员界面跳转顺序图 13

4.2.4 库存管理人员界面跳转顺序图 14

4.2.5快递员界面跳转顺序图 15

4.2.6 营业厅业务员界面跳转顺序图 16

4.2.7中转中心业务员界面跳转顺序图 17

4.2.8 总经理界面跳转顺序图 18

4.3界面原型设计图 18

5.结构视角 44

5.1业务逻辑层的分解 44

5.1.1Transportbl模块 45

5.1.2Warehousebl模块 57

5.1.3Personnelbl模块 64

5.1.4Departmentbl模块 69

5.1.5Financebl模块 76

5.1.6Listbl模块 99

5.1.7Userbl模块 103

5.1.8Logbl模块 109

6.依赖视角 111

# 1.引言

## 1.1编制目的

本报告详细完成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 |
| LMS | 物流管理系统 |  |
| BL | 业务逻辑 |  |
| PO | 数据持久化对象 |  |
| VO | 值对象 |  |

## 1.3参考资料

1.3.1 IEEE标准

1.3.2 物流管理系统（LMS）用例文档v2

1.3.3 物流管理系统（LMS）软件需求规格说明文档v2

1.3.4 项目实践v4

1.3.5 物流管理系统（LMS）体系结构描述文档

# 2.产品概述

物流管理系统（LMS）是为XXX物流公司开发的业务系统，开发的目标是帮助该物流公司处理日常的业务，经营和管理，包括寄收件管理，运输管理，人员管理，财务管理，仓库管理的功能。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储存起来进行维护，系统的用户通过客户端以远程操作的方式完成业务活动。客户端与数据集中服务器通过实时通信的方式完成数据交换。

# 3.体系结构设计概述

## 3.1 逻辑视角

物流信息管理系统中，选择了分层体系结构的风格，将系统分为3层（展示层、业务逻辑层、数据层）能够很好的示意整个高层抽象。展示层包括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。

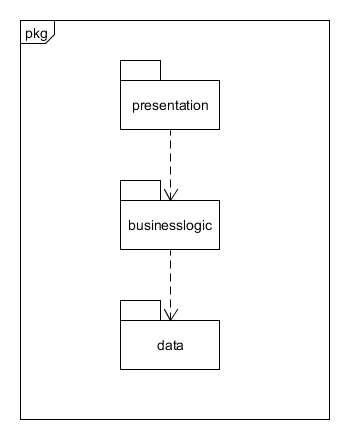


图1 参照体系结构风格的包图表达逻辑视角

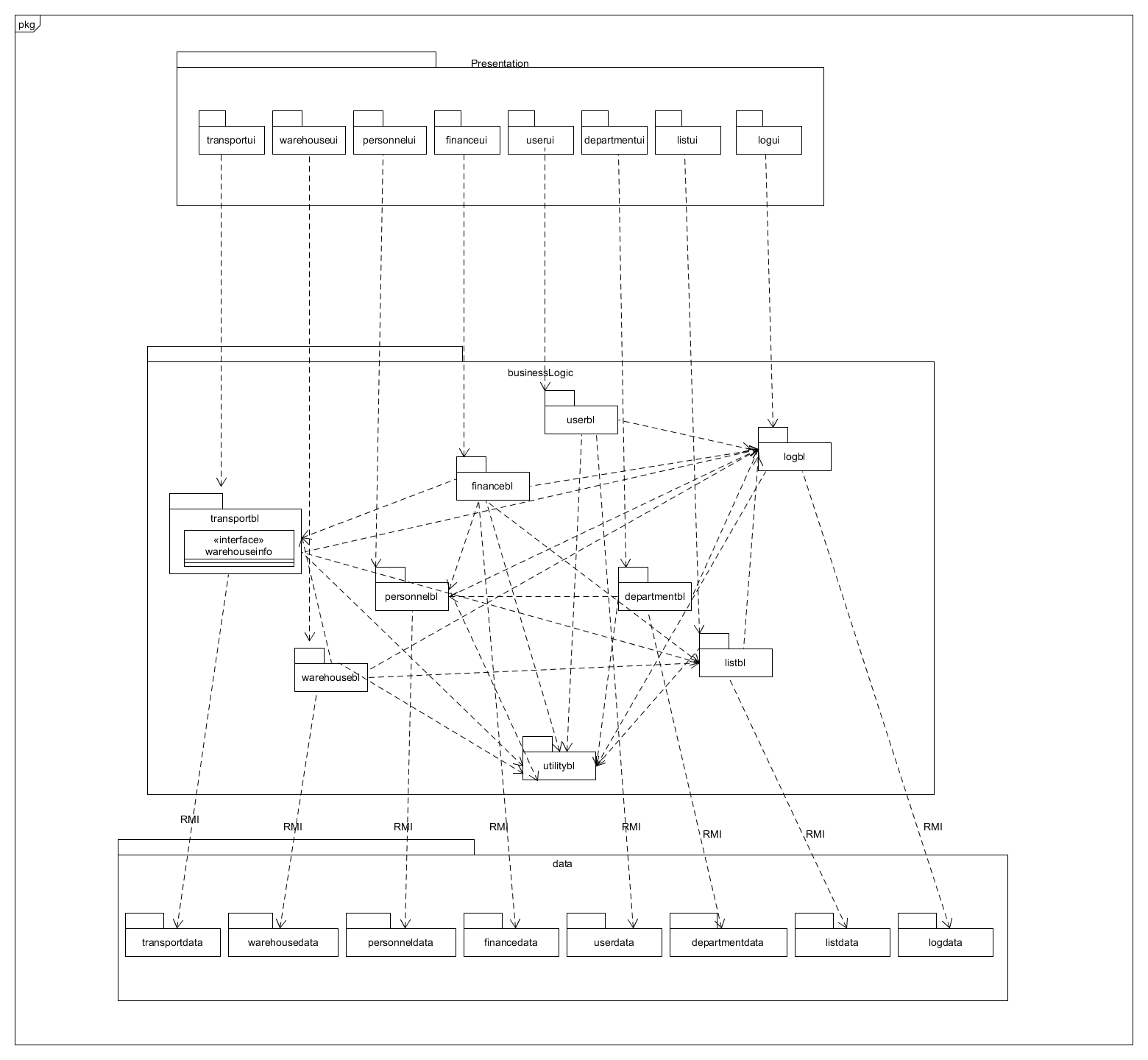


图2 软件体系结构逻辑设计方案

## 3.2开发包图

物流信息管理系统的最终开发包图设计如表1所示

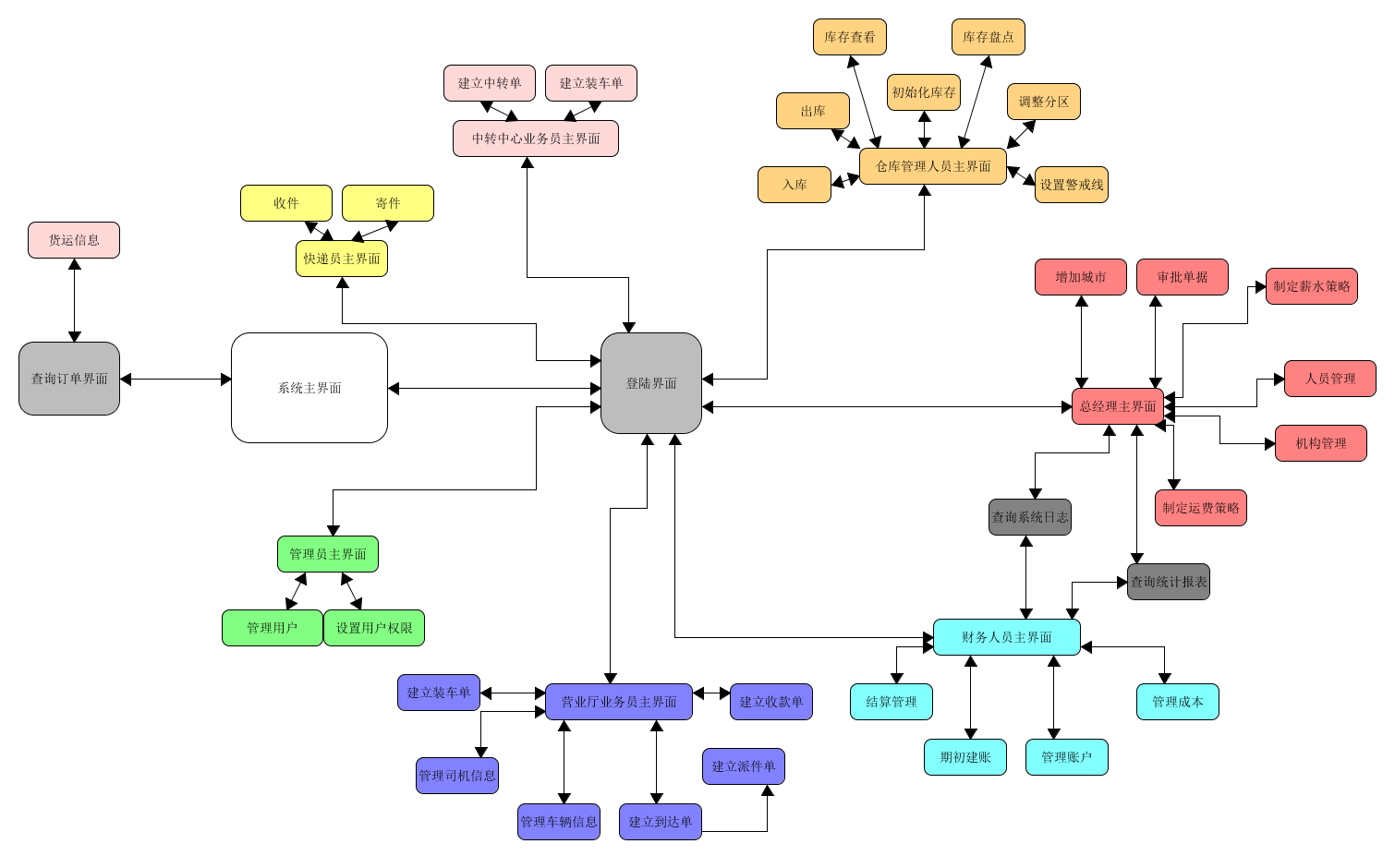
表1 快递物流系统的最终开发包设计

|  |  |
| --- | --- |
| 开发（物理）包 | 依赖的其他开发包 |
| mainui | Transportui,Warehoseui,Personnelui,Userui,Financeui,  Departmentui,Listui,Logui,Vo |
| Transportui | Transportblservice,界面类库包,Vo |
| Transportblservice | Vo |
| Transportbl | Transportblservice,Transportdata,Warehousebl,Logbl,  Po |
| Transportdataservice | Po,Java RMI |
| Transportdata | Po,Java RMI |
| Warehouseui | Warehouseblservice,界面类库包,Vo |
| Warehouseblservice | Vo |
| Warehousebl | Warehouseblservice,Warehousedata,Transportbl,Listbl,  Logbl,Po |
| Warehousedataservice | Po,Java RMI |
| Warehousedata | Po,Java RMI |
| Personnelui | Personnelblservice,界面类库包,Vo |
| Personnelblservice | Vo |
| Personnelbl | Personnelblservice,Personneldata,Logbl,Po |
| Personneldataservice | Po,Java RMI |
| Personneldata | Po,Java RMI |
| Userui | Userblservice,界面类库包,Vo |
| Userblservice | Vo |
| Userbl | Userblservice,Userdata,Logbl,Po |
| Userdataservice | Po,Java RMI |
| Userdata | Po,Java RMI |
| Financeui | Financeblservice,界面类库包,Vo |
| Financeblservice | Vo |
| Financebl | Financeblservice,Financedata,Transportbl,Personnelbl,  Listbl,Logbl,Po |
| Financedataservice | Po,,Java RMI |
| Financedata | Po,Java RMI |
| Departmentui | Departmentblservice,界面类库包,Vo |
| Departmentblservice | Vo |
| Departmentbl | Departmentbl,Departmentdata,Personnelbl,Logbl,Po |
| Departmentdataservice | Po,Java RMI |
| Departmentdata | Po,Java RMI |
| Listui | Listblservice,界面类库包,Vo |
| Listblservice | Vo |
| Listbl | Listblservice,Listdata,Logbl,Po |
| Listdataservice | Po,Java RMI |
| Listdata | Po,Java RMI |
| Logui | Logblservice,界面类库包,Vo |
| Logblservice | Vo |
| Logbl | Logblservice,Logdata,Po |
| Logdataservice | Po,Java RMI |
| Logdata |  |
| Vo |  |
| Po |  |
| Utilitybl |  |
| 界面类库包 |  |
| Java RMI |  |

# 4.人机交互设计

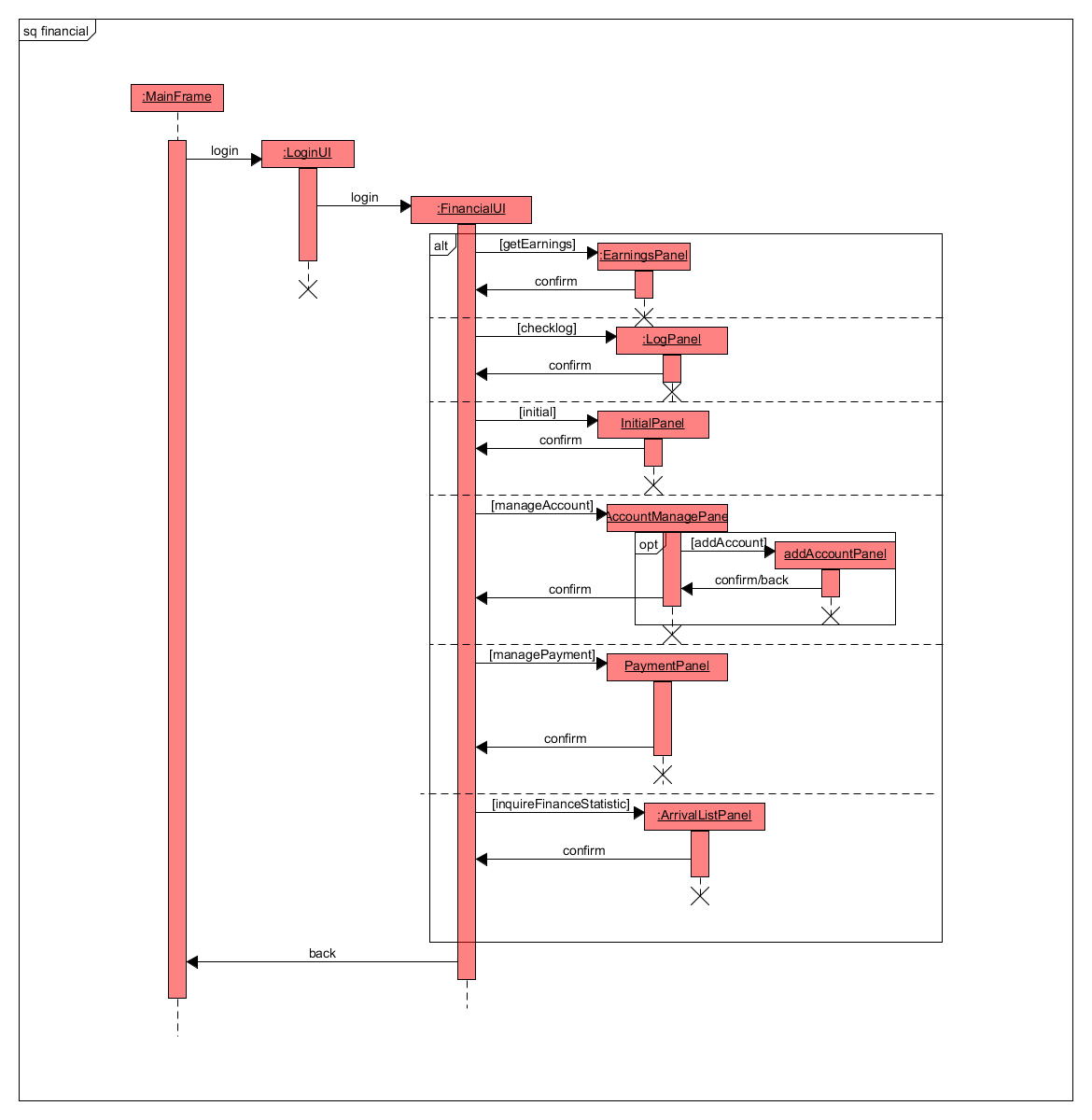
## 4.1 界面分解示意图

根据需求，系统存在42个用户界面：系统主界面、查询订单界面、货运信息界面、登陆界面、快递员主界面、收件界面、寄件界面、管理员主界面、管理用户界面、设置用户权限界面、中转中心业务员主界面、建立中转单界面、建立装运单界面、营业厅业务员主界面、建立装车单界面、管理司机信息界面、管理车辆信息界面、建立到达单界面、建立派件单界面、建立收款单界面、仓库管理人员主界面、入库界面、出库界面、库存查看界面、初始化库存界面、库存盘点界面、调整分区界面、设置警戒线界面、财务人员主界面、结算管理界面、期初建账界面、管理账户界面、管理成本界面、查看系统日志界面、查询统计报表界面、总经理主界面、增加城市界面、审批单据界面、制定运费策略界面、机构管理界面、人员管理界面和制定薪水策略。界面跳转如图所示。

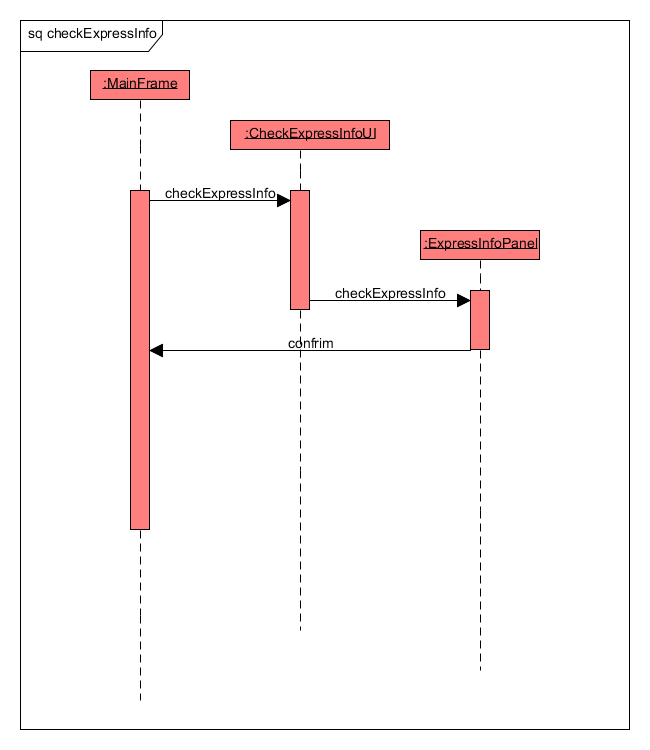


## 4.2界面跳转顺序图

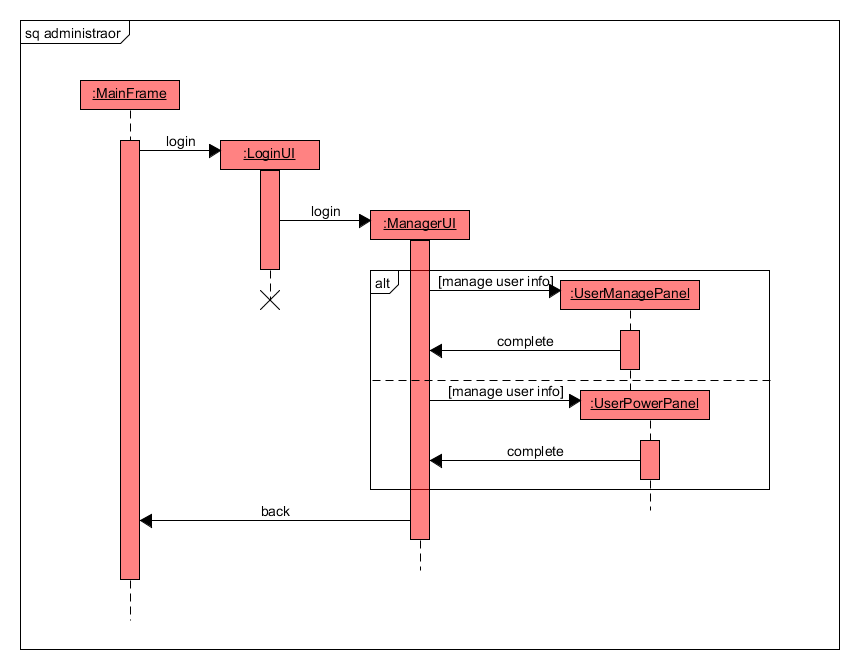
### 4.2.1 财务人员界面跳转顺序图



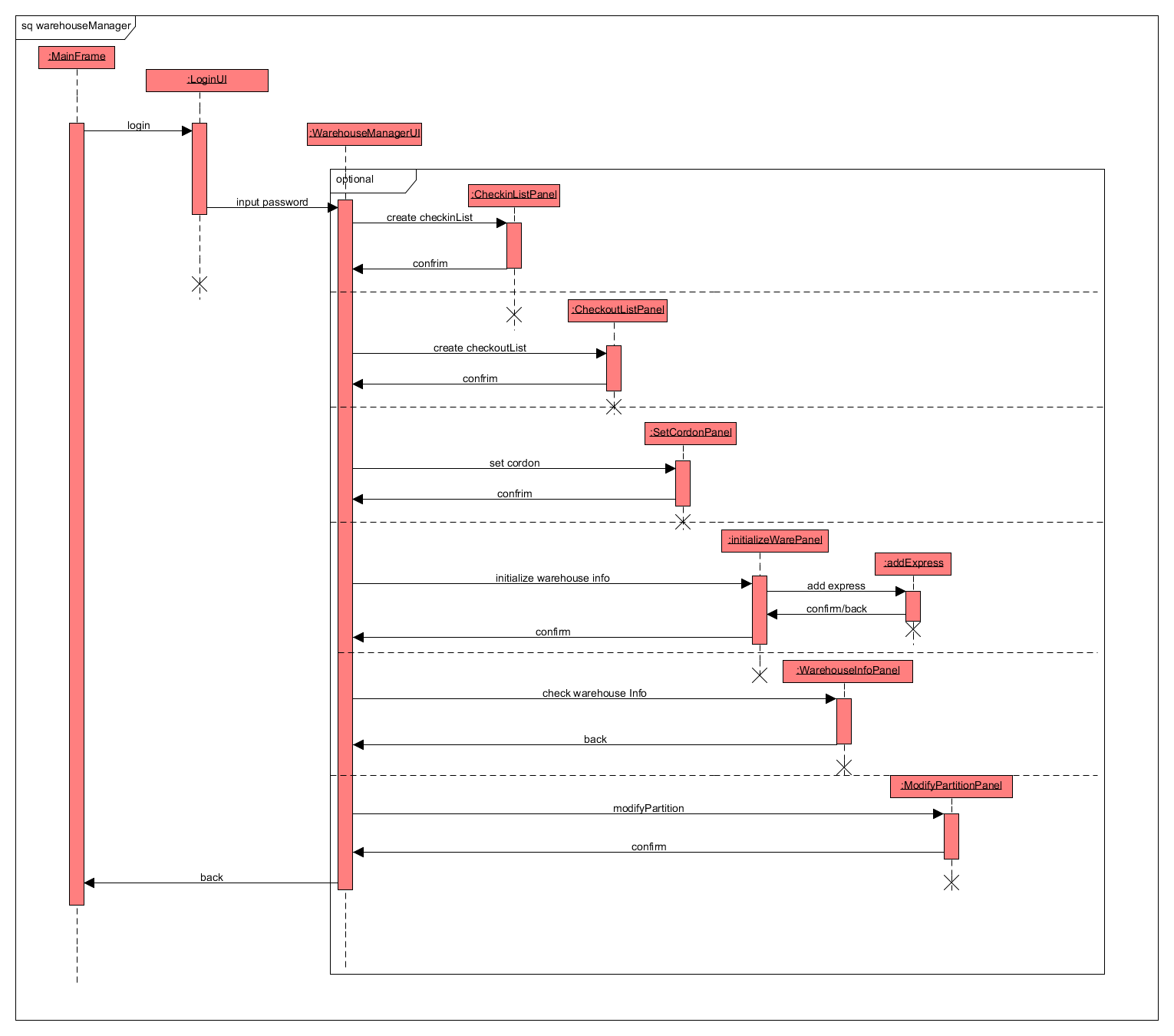
### 4.2.2查询物流信息界面跳转顺序图



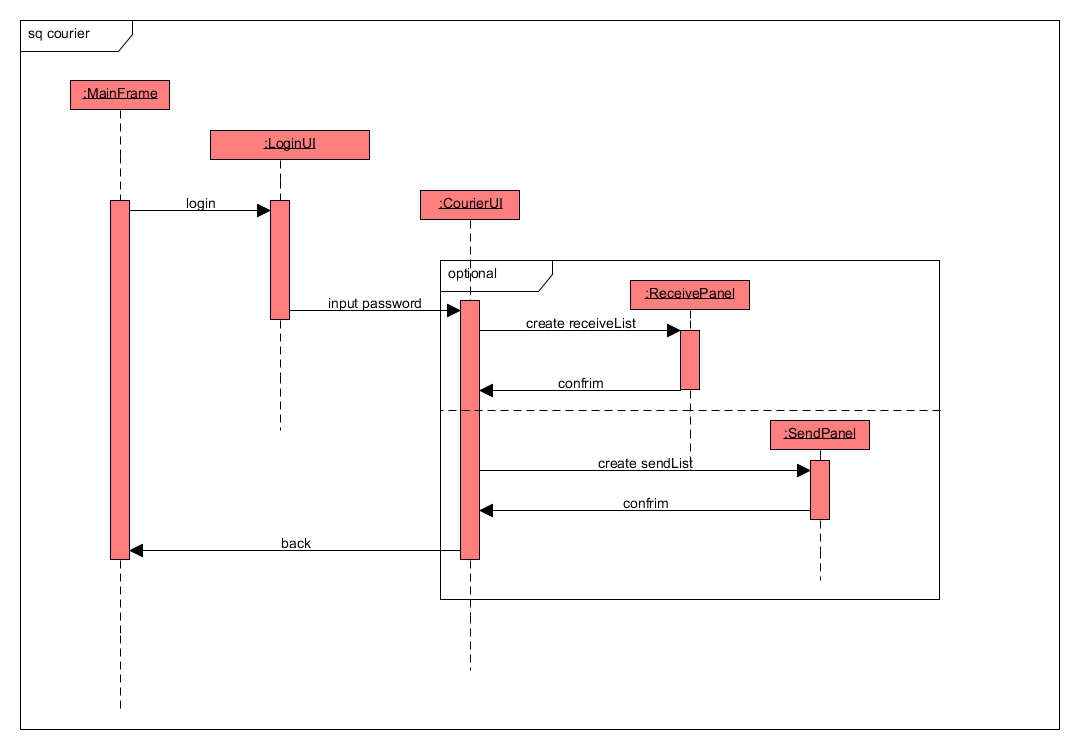
### 4.2.3管理员界面跳转顺序图



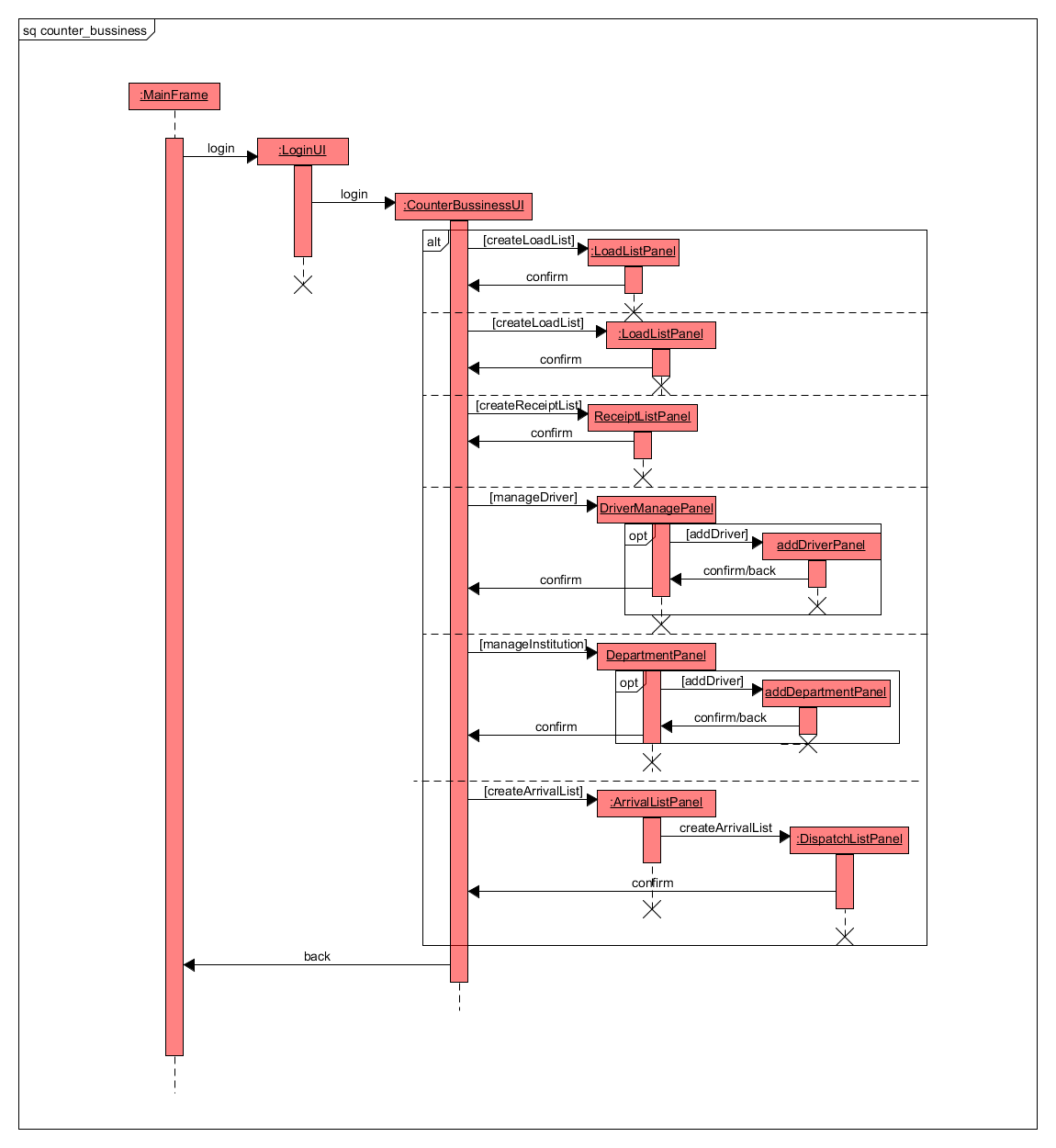
### 4.2.4 库存管理人员界面跳转顺序图



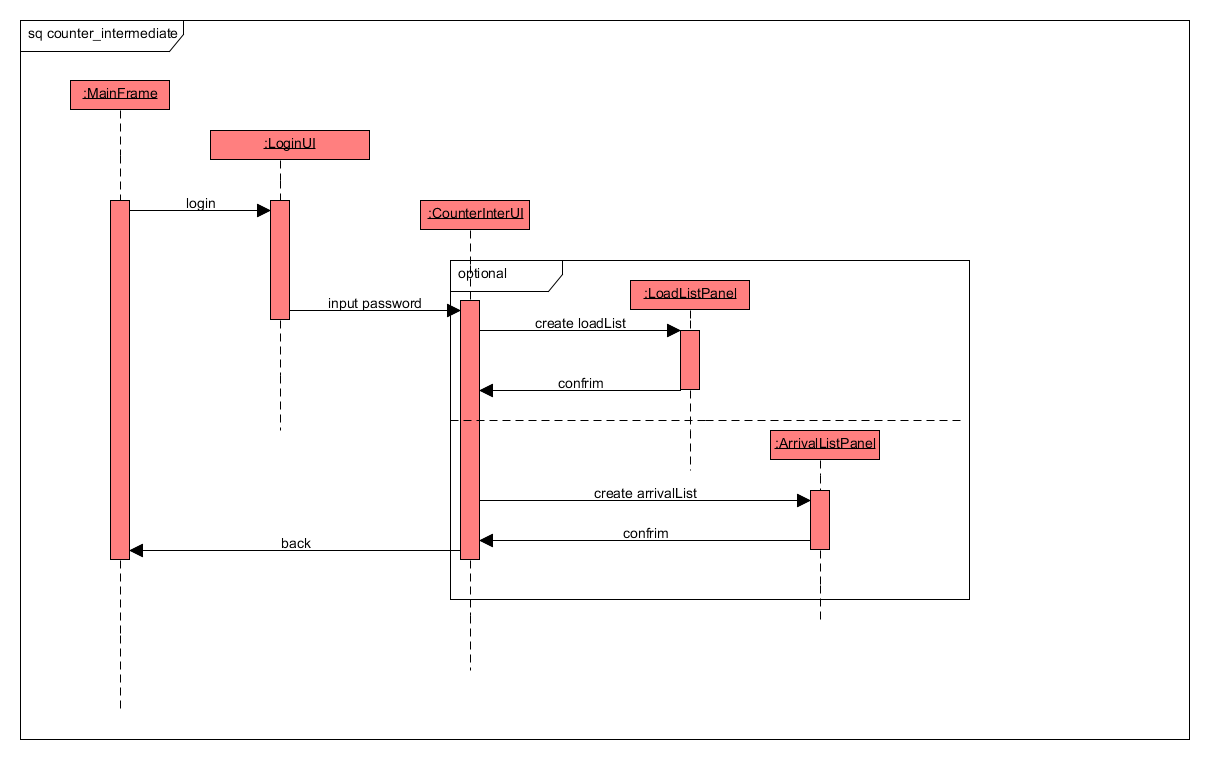
### 4.2.5快递员界面跳转顺序图



### 4.2.6 营业厅业务员界面跳转顺序图



### 4.2.7中转中心业务员界面跳转顺序图

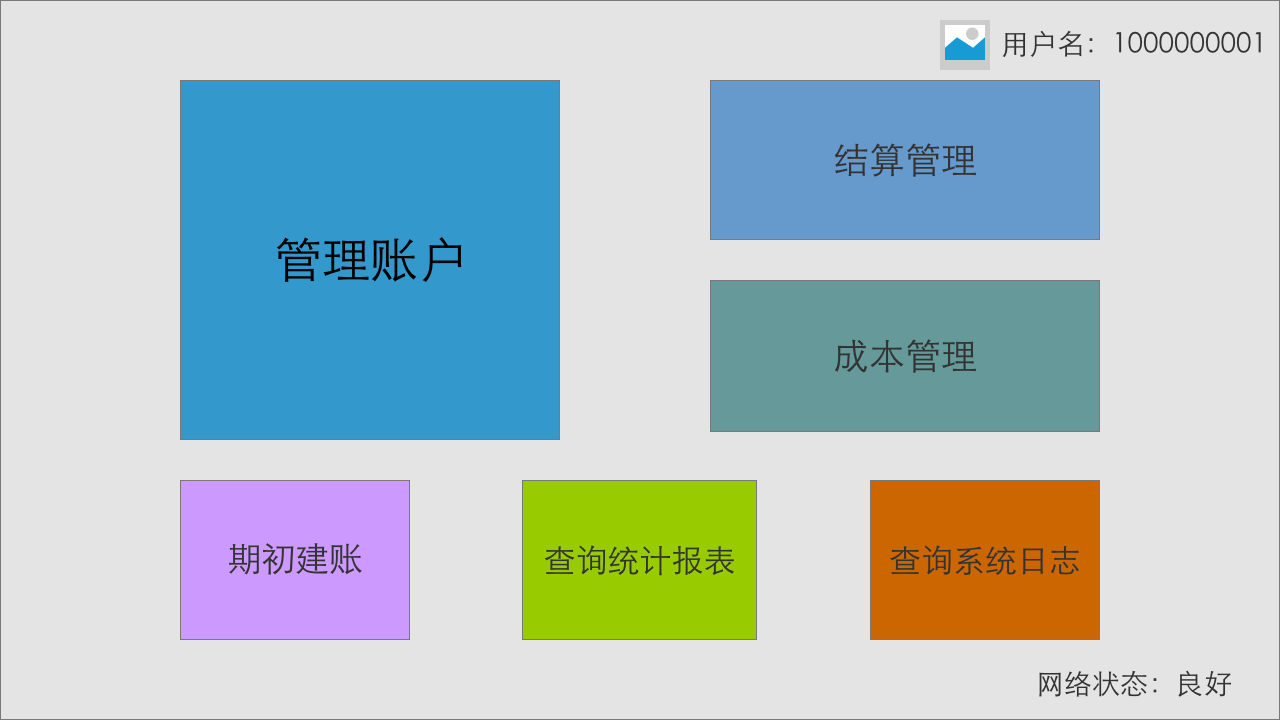


### 4.2.8 总经理界面跳转顺序图



## 4.3界面原型设计图

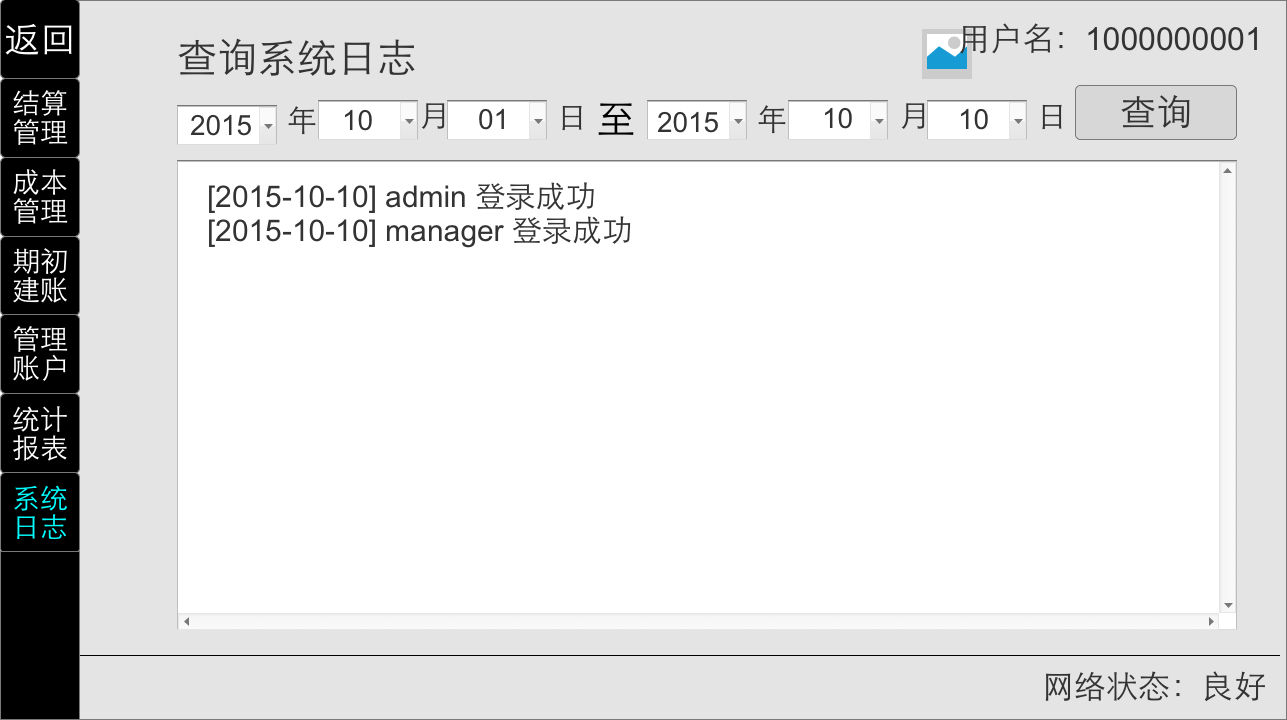
财务人员主界面



查询订单界面

查询统计报表界面



查询系统日志界面

成本管理界面

登陆界面



订单信息显示界面

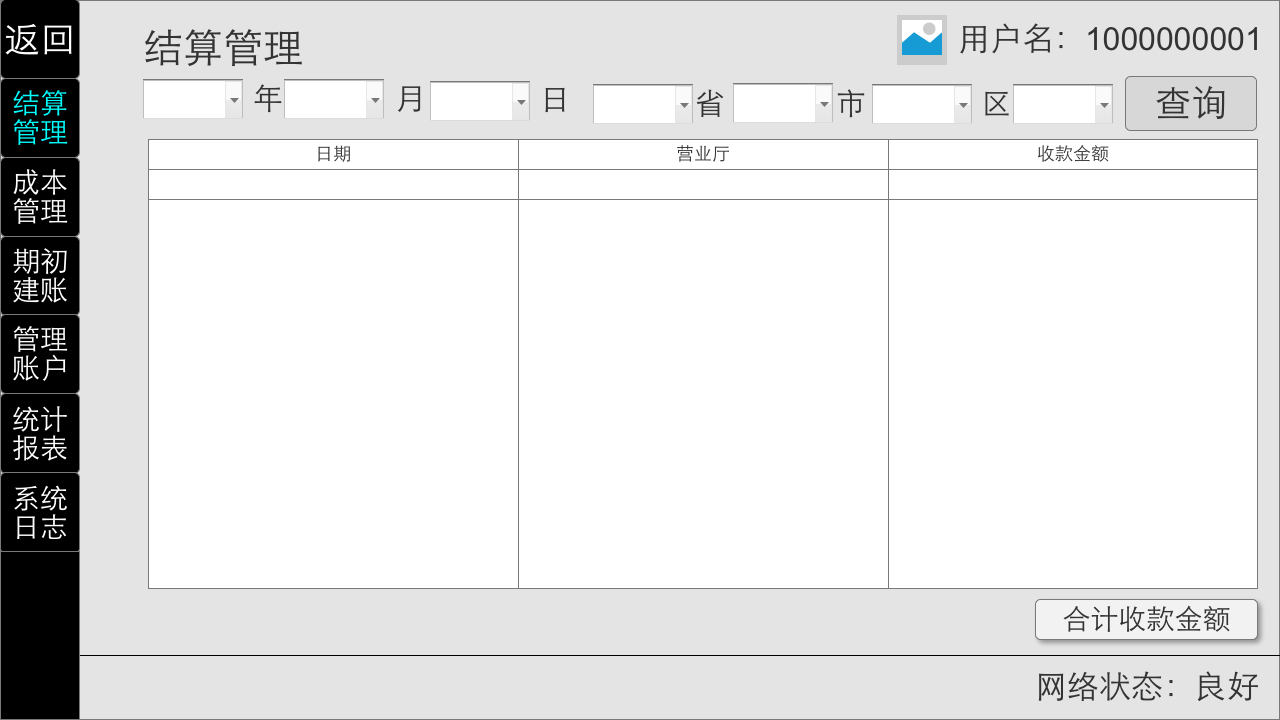
管理人员工资界面

管理运费界面

管理账户界面



管理租金界面

结算管理界面

期初建账界面



确定生成界面

生成成本收益表界面

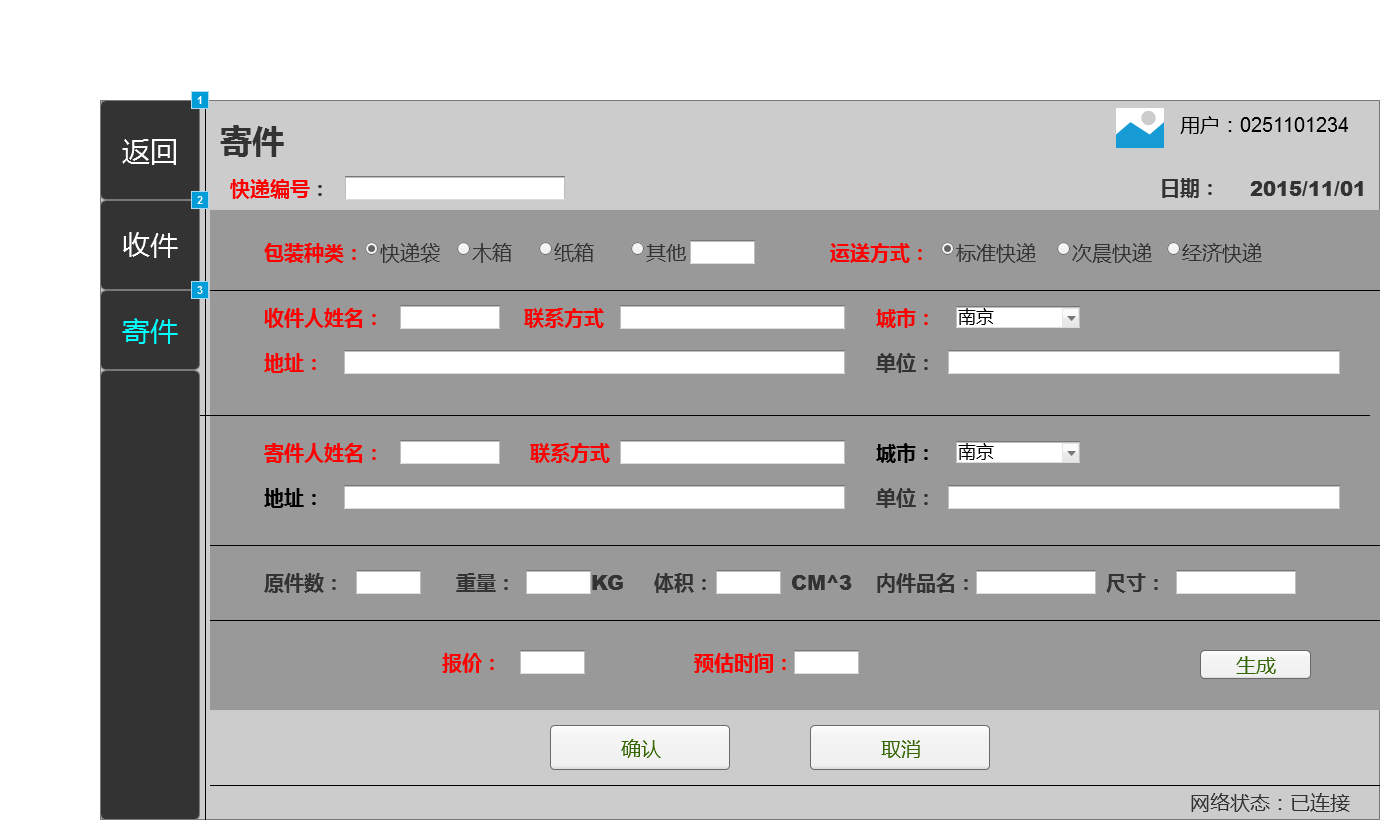
管理用户界面

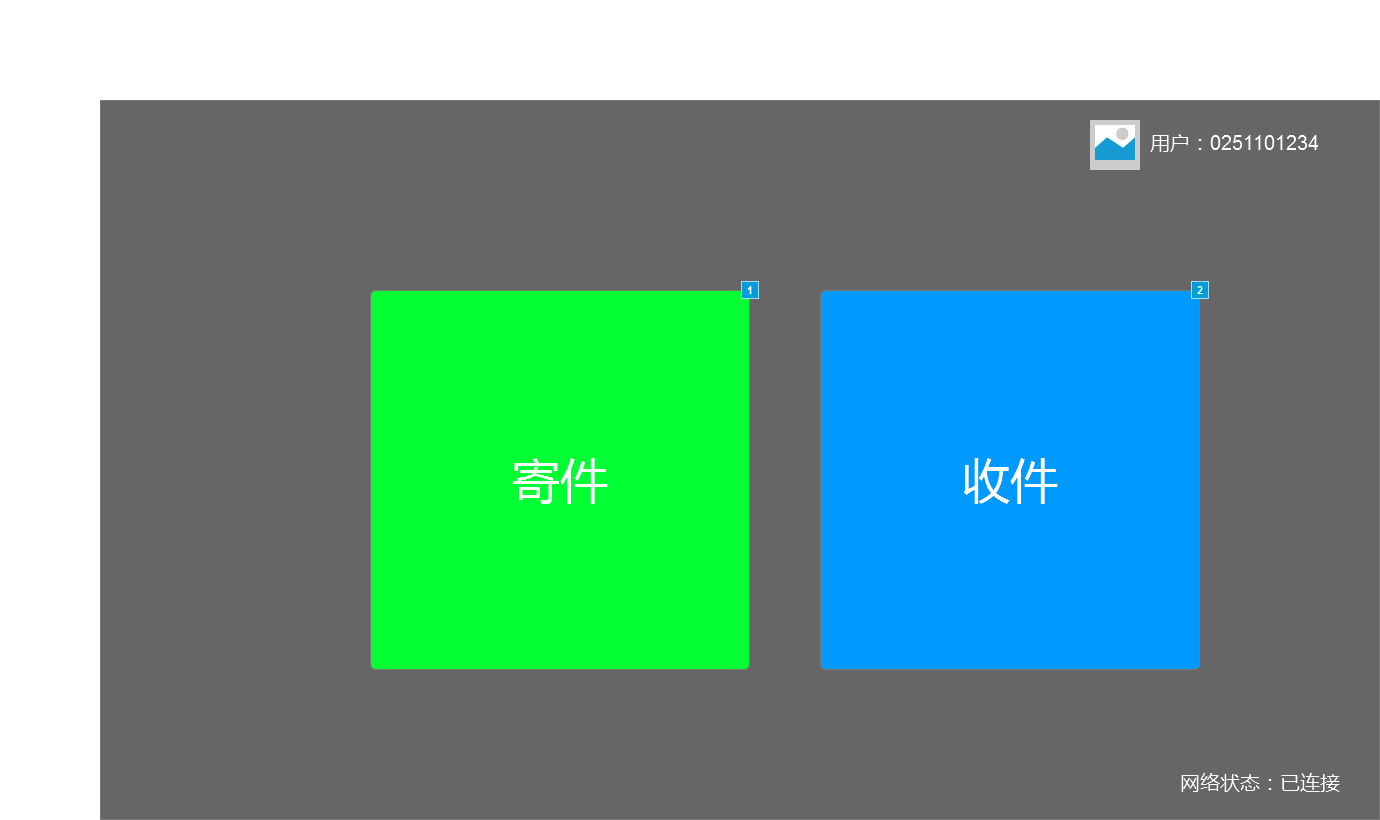


管理员主界面

设置用户权限界面

寄件界面



快递员主界面

收件界面

管理车辆信息界面



管理司机信息界面

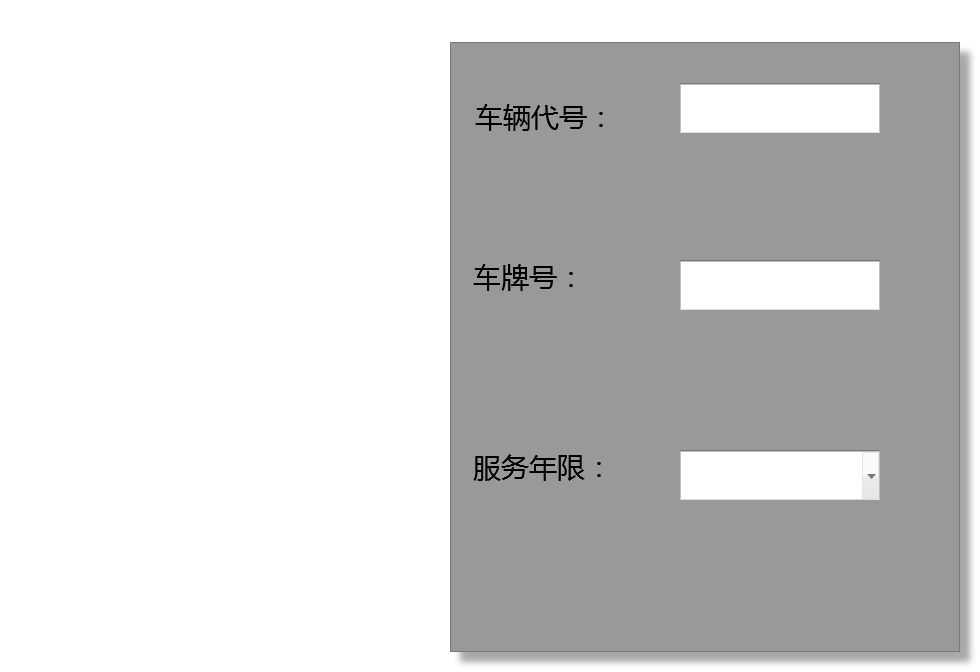
建立到达单界面

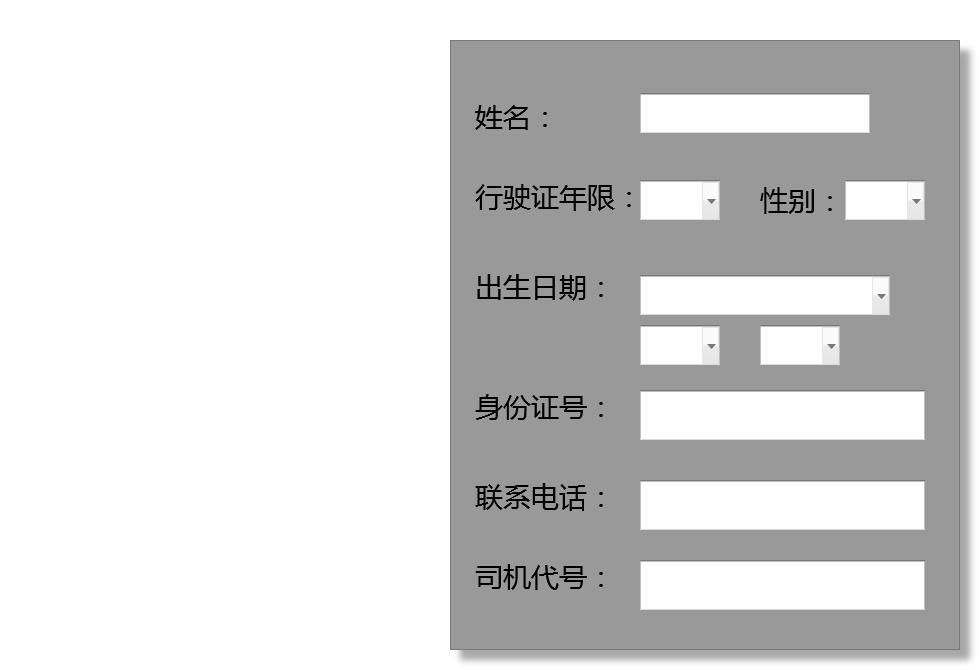
建立派件单界面

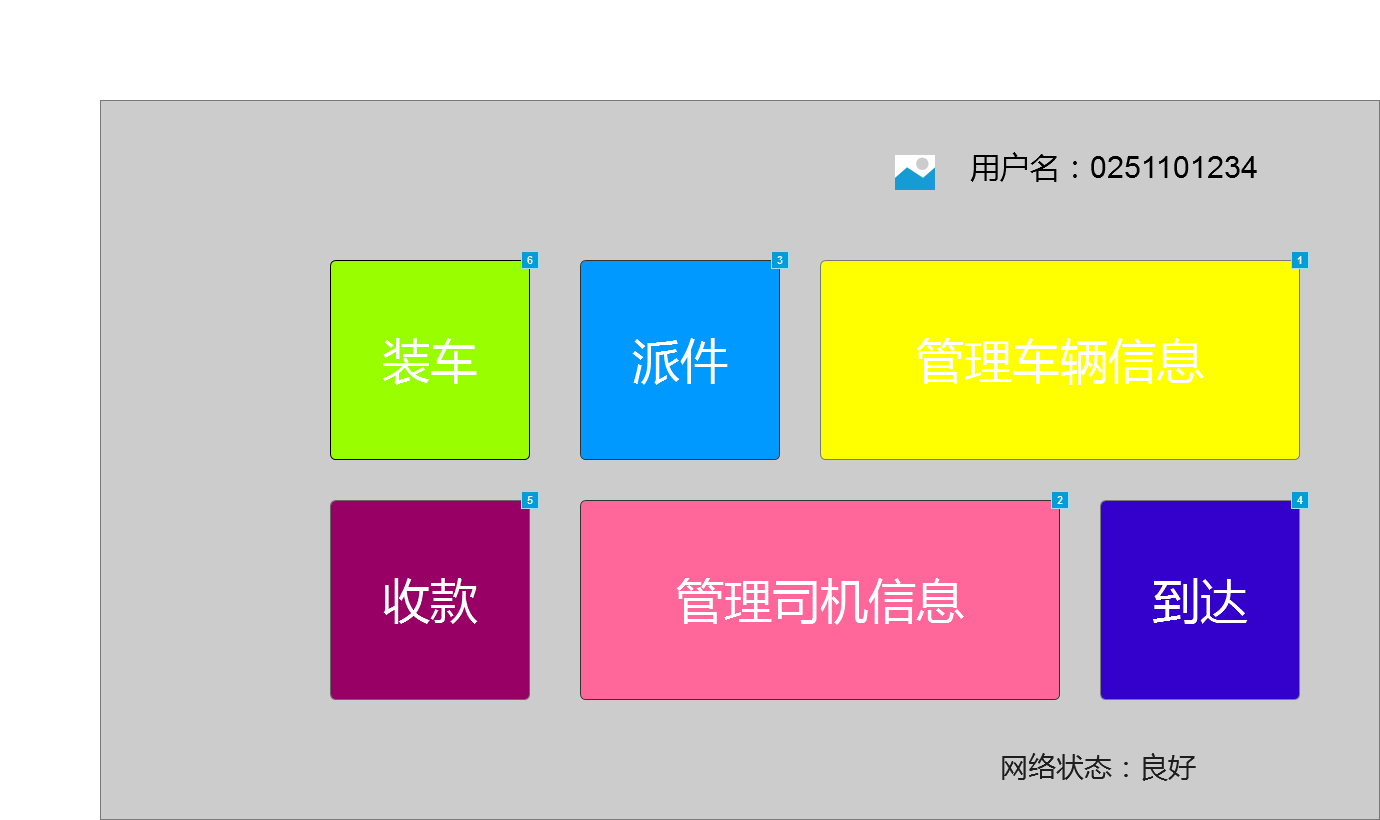
建立收款单界面



建立装车单界面

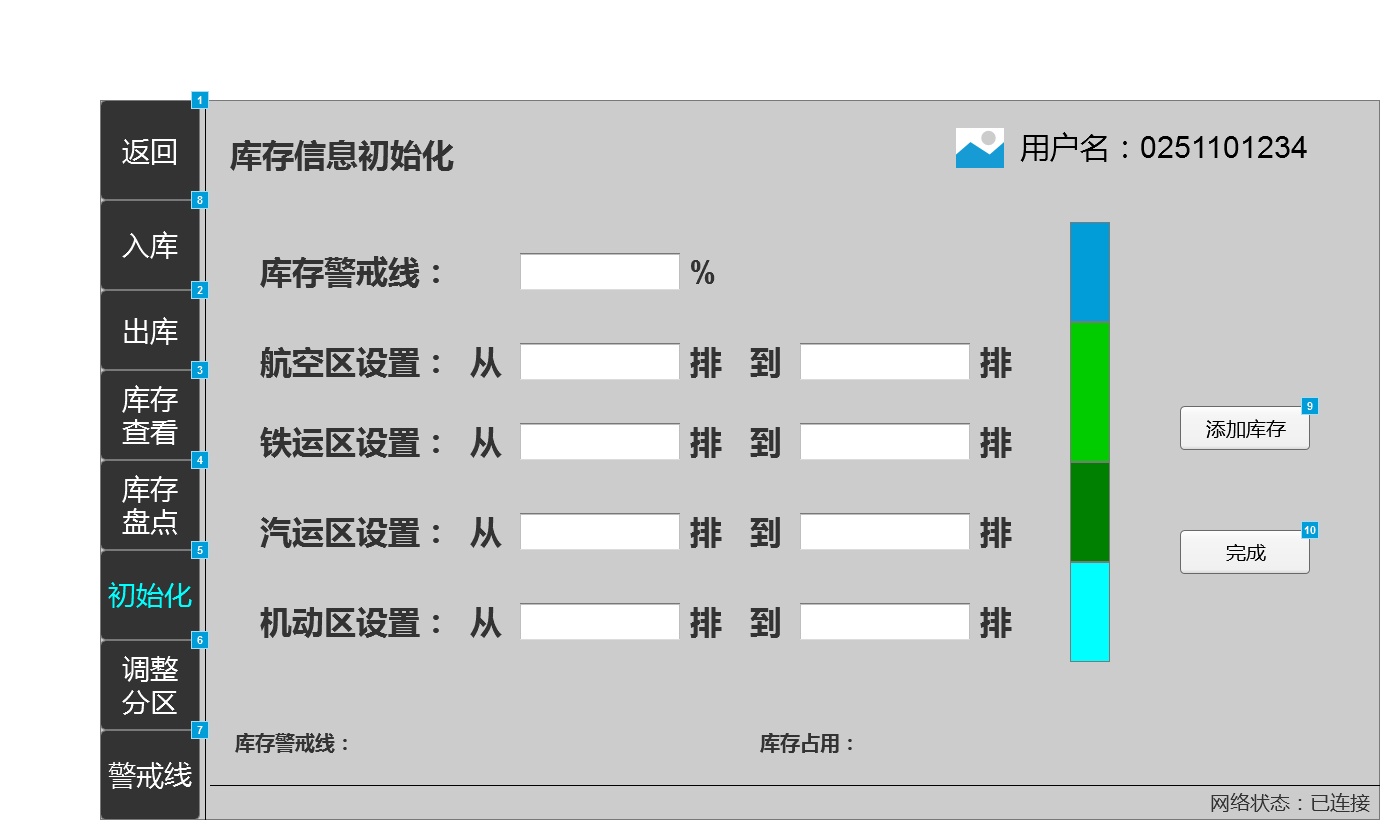
添加车辆界面

添加司机界面

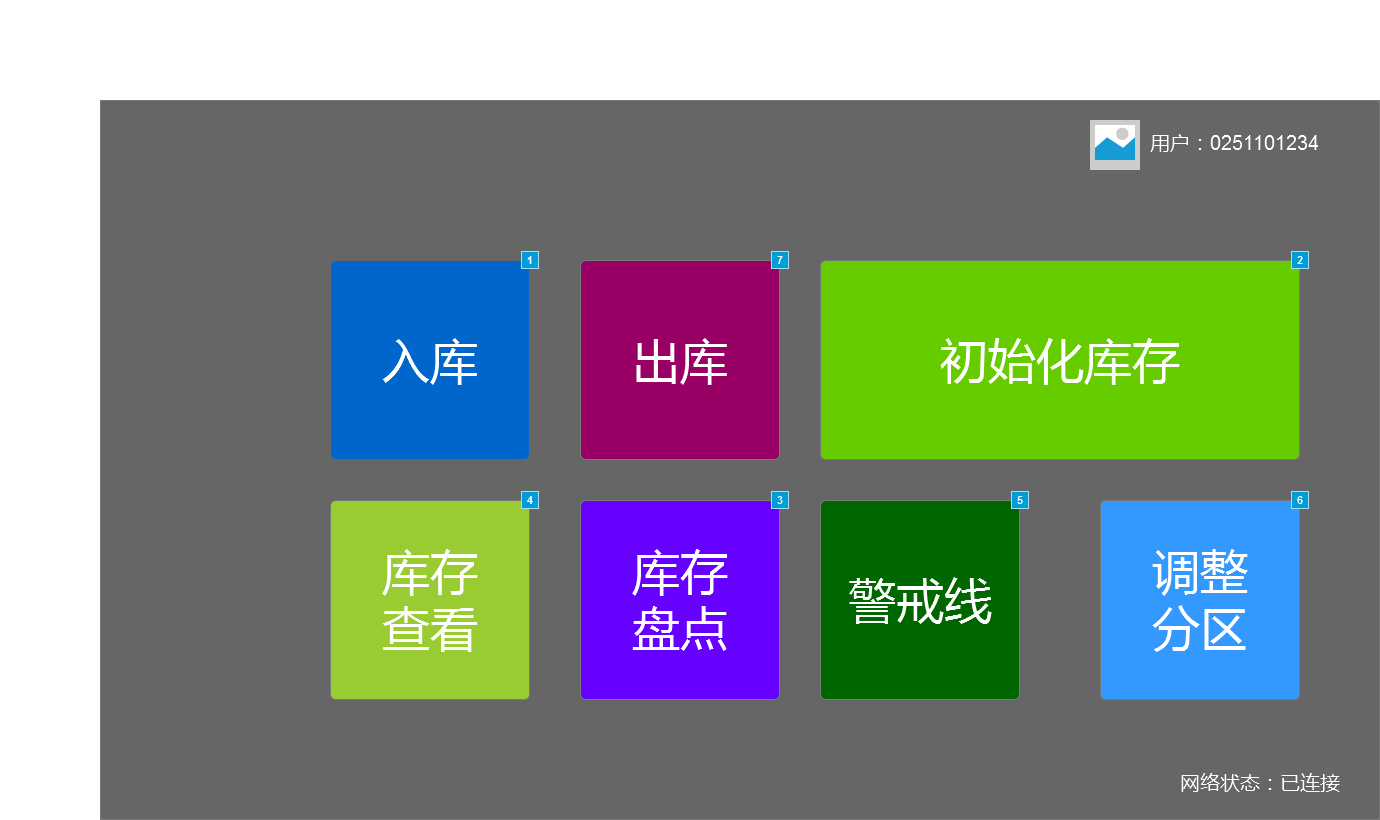
营业厅业务员主界面

出库界面

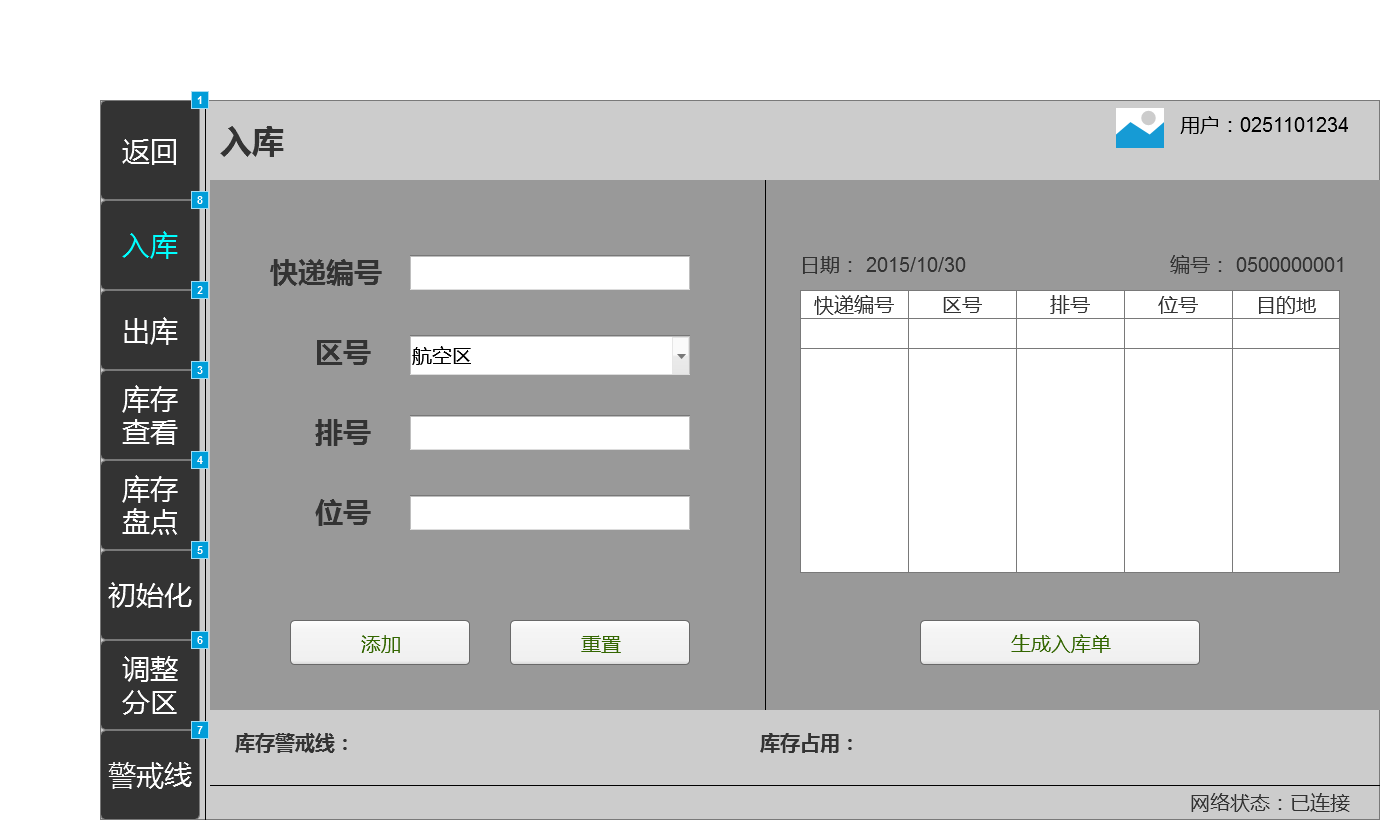


初始化库存界面

库存查看界面

库存管理人员主界面

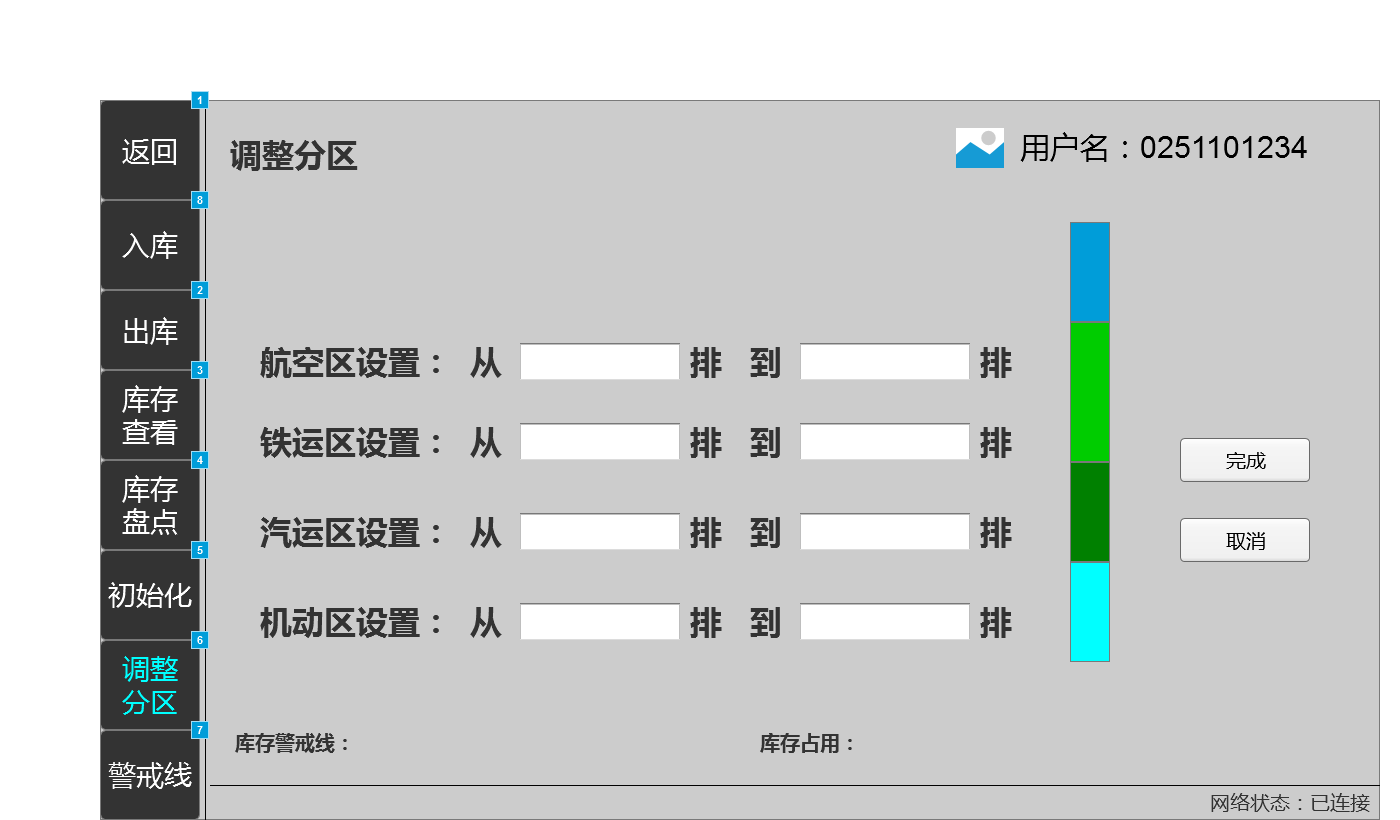
库存盘点界面

入库界面

设置警戒线界面

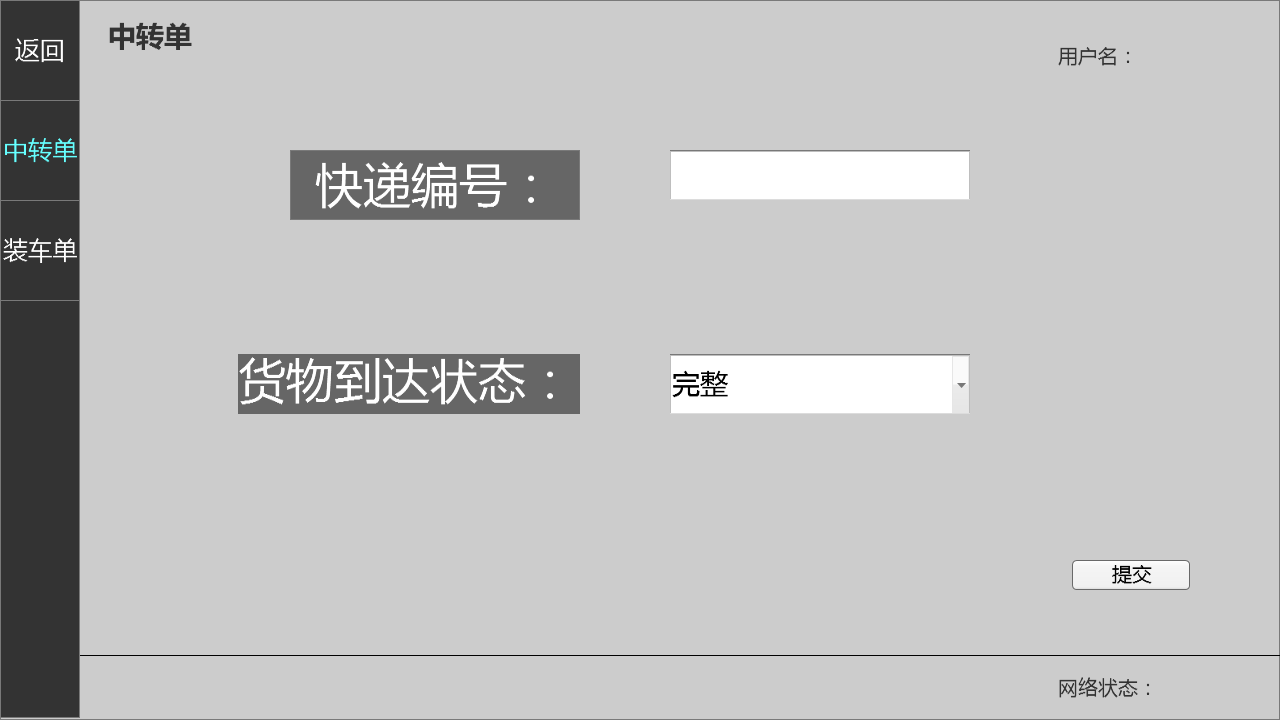
添加界面

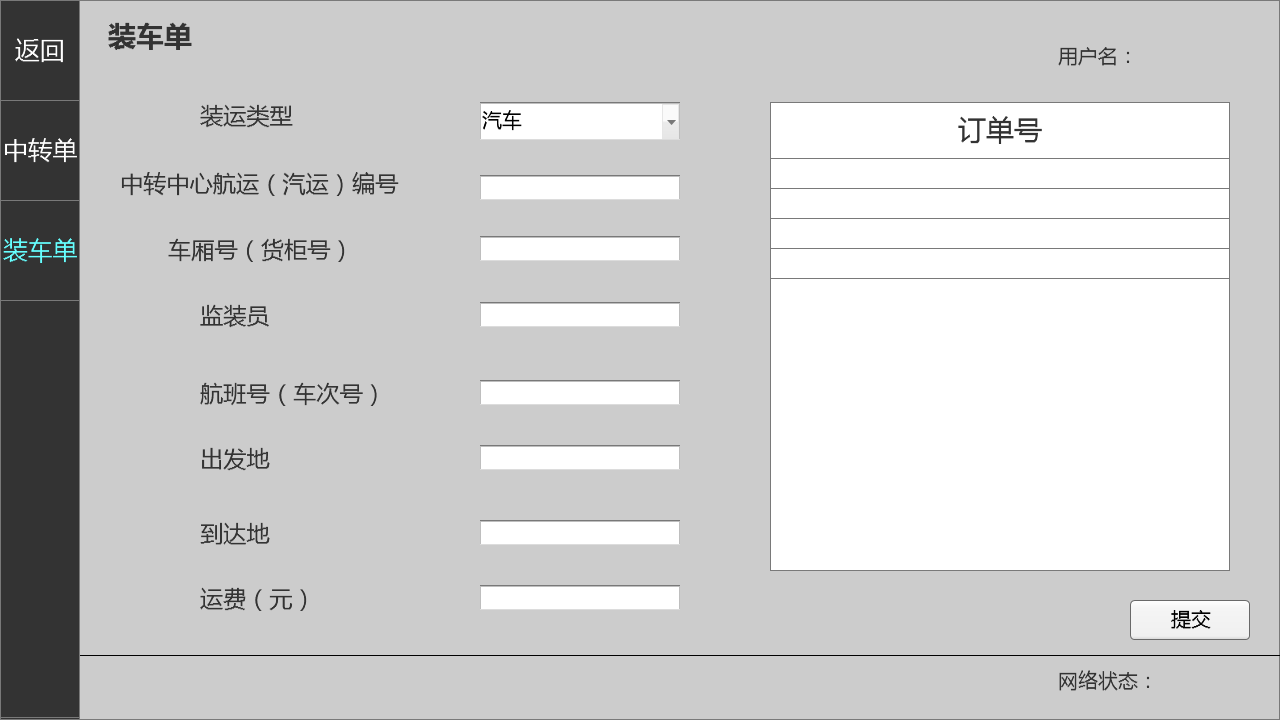


调整分区界面

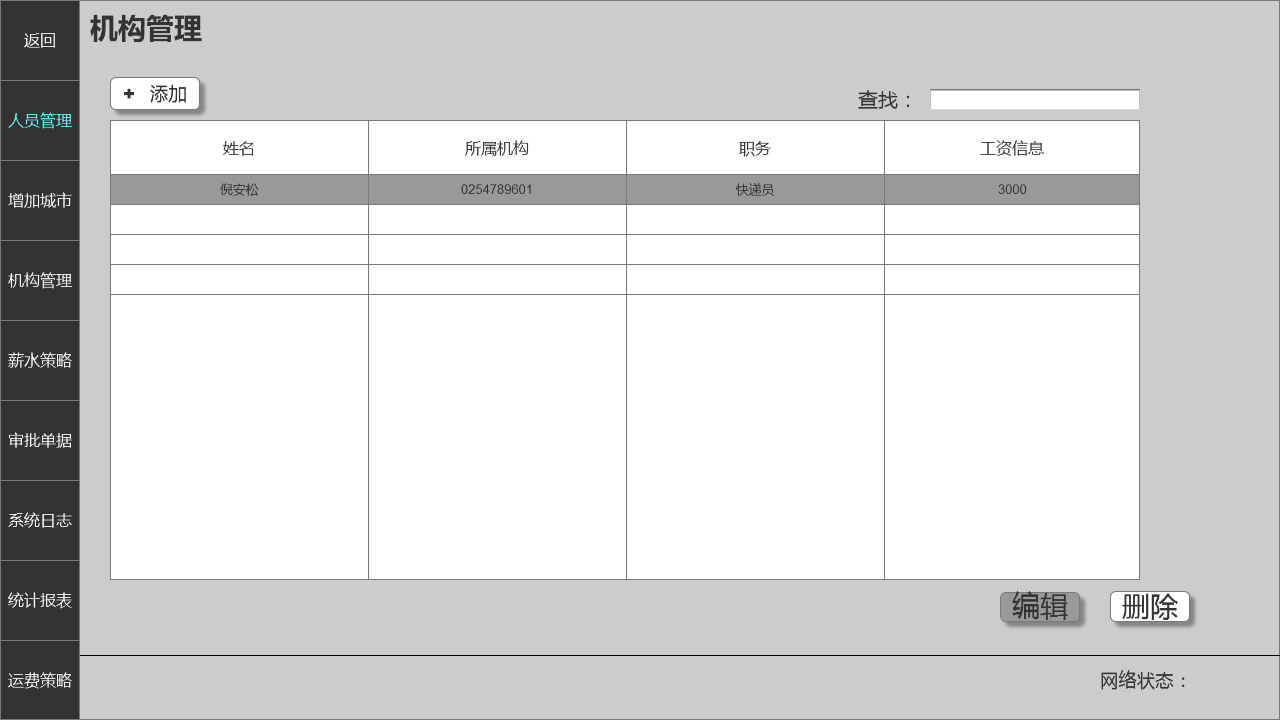
完整中转单界面



建立中转单界面

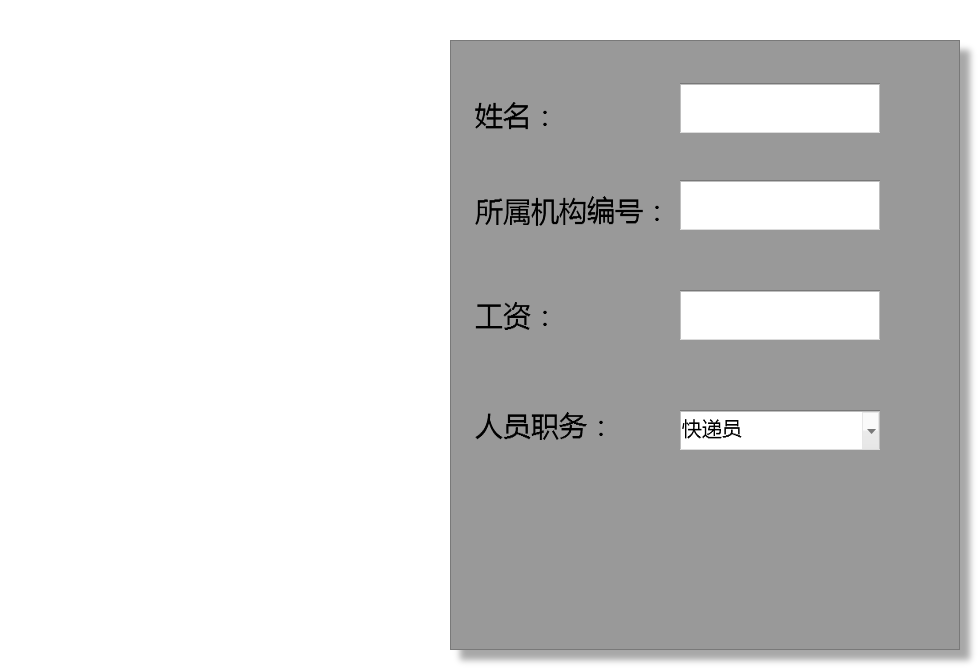
建立装车单界面机构管理界面



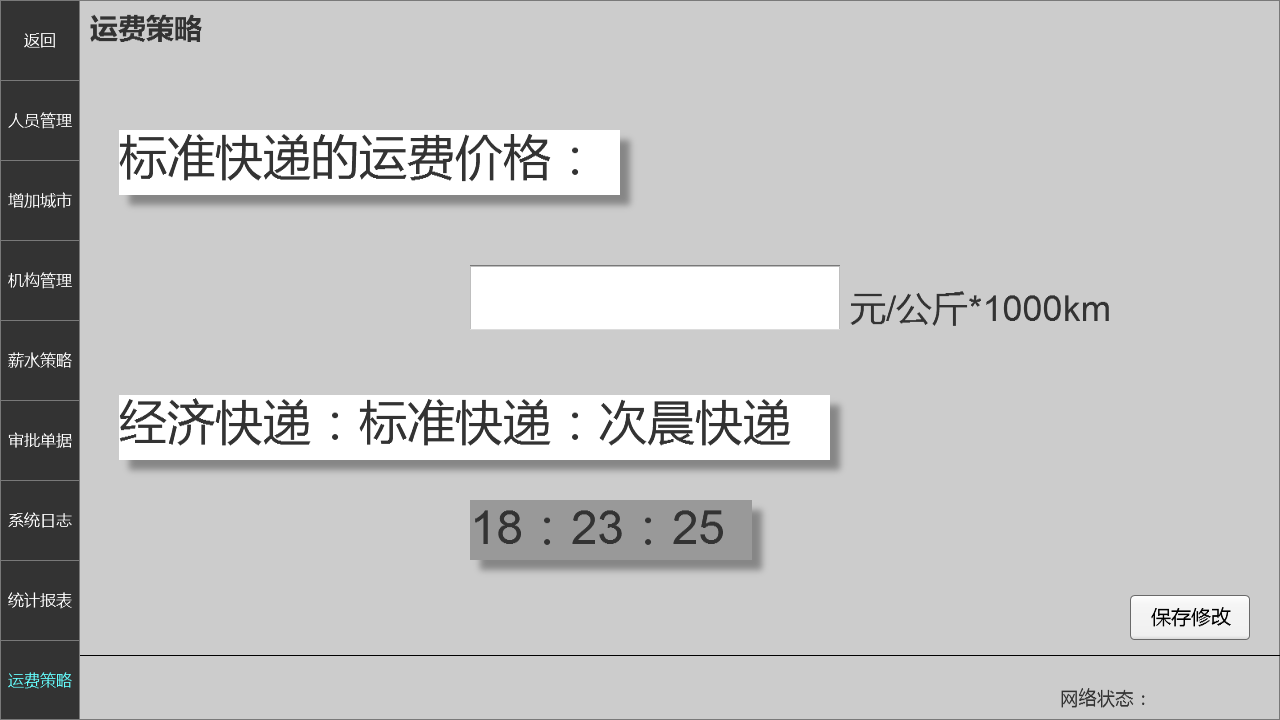
人员管理界面

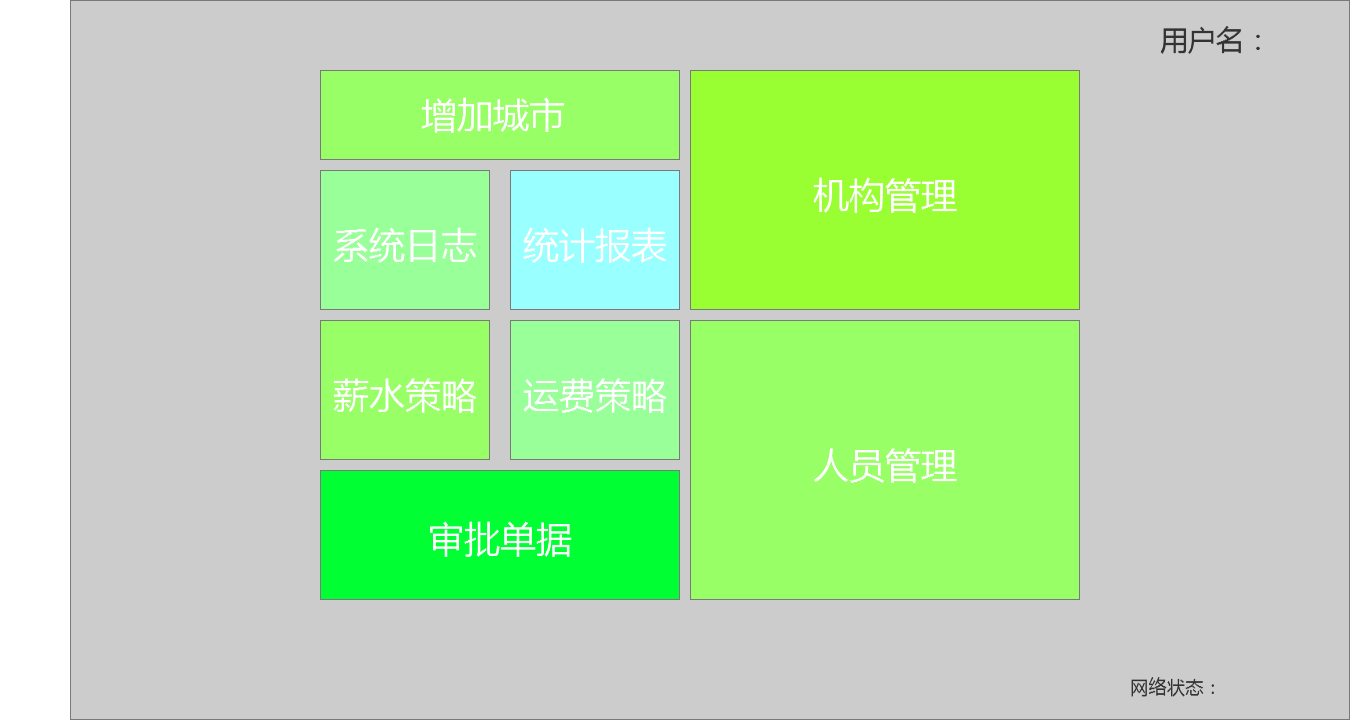
审批单据界面

添加机构界面

添加人员界面

薪水策略界面

运费策略界面

增加城市界面总经理主界面

# 5.结构视角

## 5.1业务逻辑层的分解

业务逻辑层的开发包图参见物流管理系统体系结构描述文档。

### 5.1.1Transportbl模块

（1）模块概述

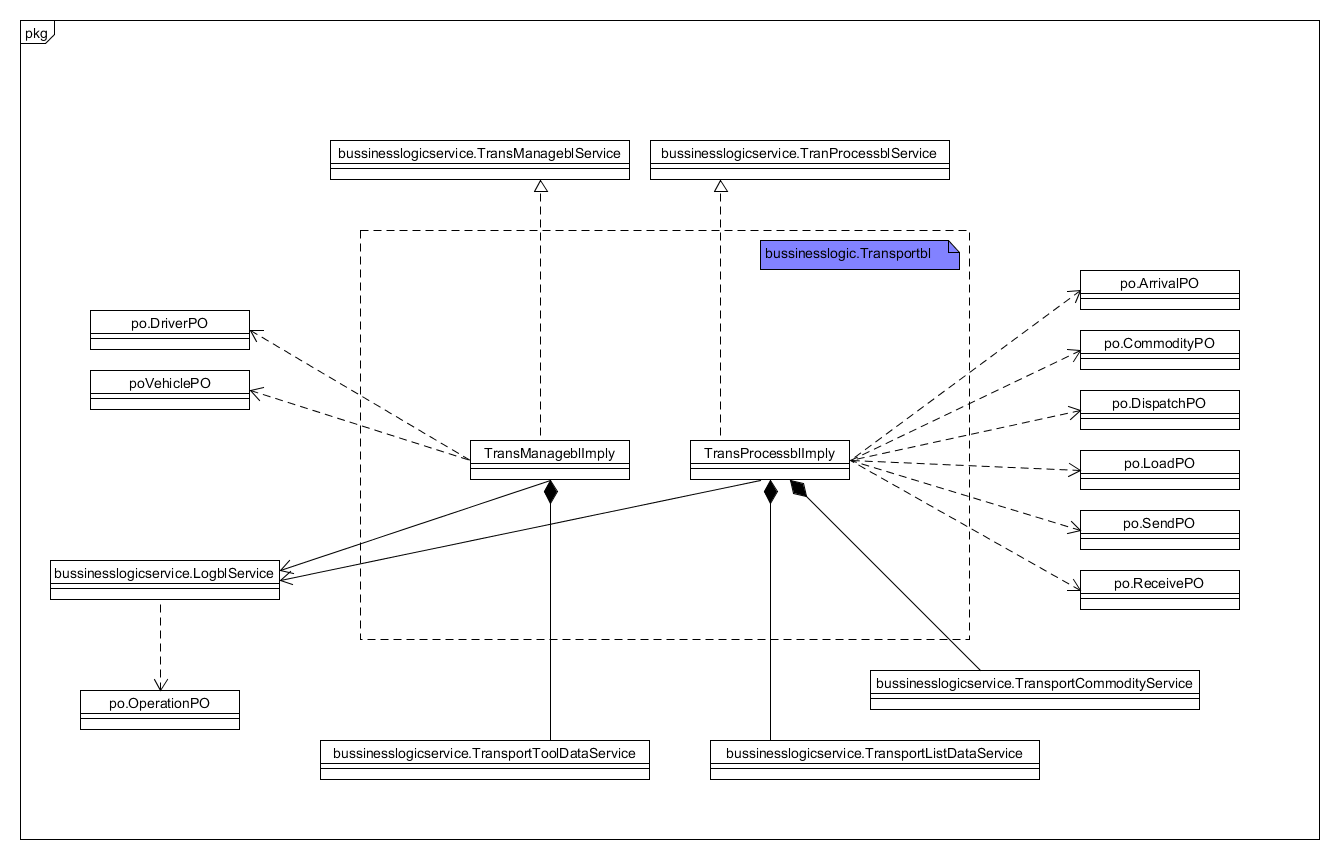
Transportbl模块承担的需求参见需求规格说明文档功能需求和相关非功能需求。

Salesbl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，我们在在展示层和业务逻辑层之间为该功能添加接口bussinesslogicservice. tranProcessblService和bussinesslogicservive. tranManageblService。业务逻辑层和数据层之间添加dataService.TransportCommodityDataService, dataService.TransportToolDataService, dataService.TransportListDataService接口。SendPO, LoadPO, ReceivePO, ArrivalPO, DispatchPO是作为运输过程中的持久化对象被添加到设计模型中的。此外，所有的操作都需要生成操作记录，因此需要logbl的协作。

Transportbl模块的设计如图所示：



Transportbl模块各个类的职责如表所示：

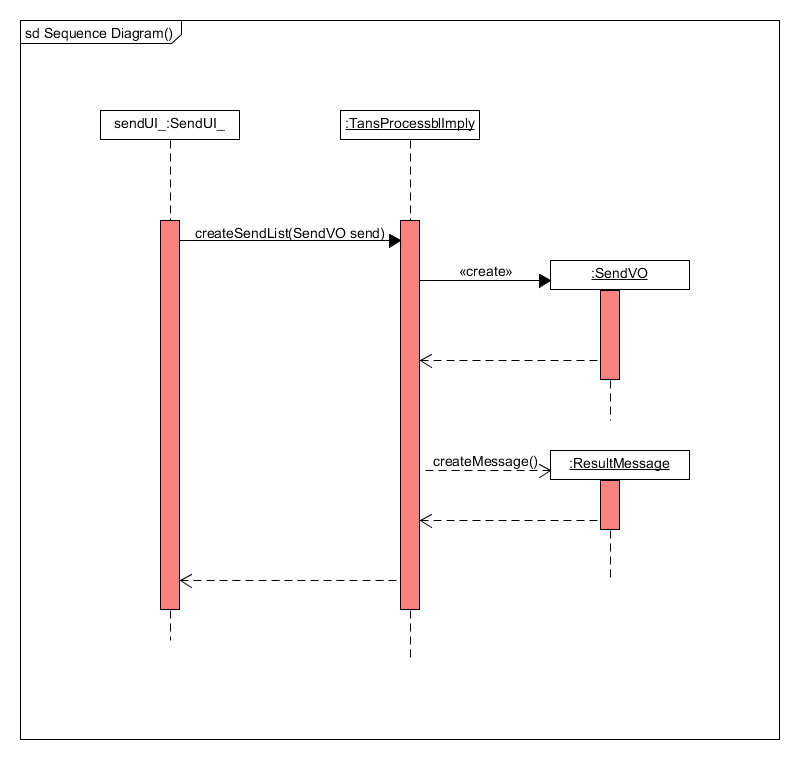
|  |  |
| --- | --- |
| 模块 | 职责 |
| TransportController | 负责提供运输界面所需要的服务 |
| TransHistoryTrqck | 负责物流信息管理部分业务逻辑的实现 |
| TransManageblImpl | 负责处理司机和车辆管理的业务逻辑实现 |
| TransProcessblImpl | 负责生成和保存运输过程中产生的单据，包括寄件单，转运单，到达单，派件单，收件单。 |

（3）模块内部的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| TransportController. checkOrderInfor | 语法 | public OrderInforVO checkOrderInfor(String orderNum); |
| 前置条件 | 订单号符合输入规则 |
| 后置条件 | 查找是否存在相应的订单，根据输入的订单号返回订单查询的结果 |
| TransportController. createSendList | 语法 | public SendVO createSendList(SendVO baseMessage); |
| 前置条件 | baseMessage中的信息都符合规范 |
| 后置条件 | 返回给界面完整的寄件单信息 |
| TransportController. SaveSendList | 语法 | public ResultMessage saveSendList(SendVO sendList); |
| 前置条件 | sendList的信息都符合规范且完整 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| TransportController. findSendList | 语法 | public SendVO findSendList(String id); |
| 前置条件 | sendList的信息都符合规范且完整 |
| 后置条件 | 返回涉及的领域对象的数据 |
| TransportController. createLoadList | 语法 | public LoadVO createLoadList(LoadVO baseMessage); |
| 前置条件 | baseMessage中的信息都符合规范 |
| 后置条件 | 返回给界面完整的装车单信息 |
| TransportController. saveLoadList | 语法 | public ResultMessage saveLoadList(LoadVO loadList); |
| 前置条件 | loadList的信息都符合规范且完整 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| TransportController. findUnpaidLoad | 语法 | public ArrayList<LoadVO> findUnpaidLoad(); |
| 前置条件 |  |
| 后置条件 | 返回涉及的领域对象的数据 |
| Transport. createArrivalList | 语法 | public ArrivalVO createArrivalList(ArrivalVO arrival); |
| 前置条件 | 订单条号符合规范 |
| 后置条件 | 返回给界面完整的到达单信息 |
| Transport. saveArrivalList | 语法 | public Boolean saveArrivalList(ArrivalVO arrivalList); |
| 前置条件 | arrivalList的信息都完整且符合规范 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| TransportController. findArrivalList | 语法 | public ArrivalVO findArrivalList(String id); |
| 前置条件 | ArrivalList的信息都符合规范且完整 |
| 后置条件 | 返回涉及的领域对象的数据 |
| TransportController. createDispatchList | 语法 | public DispatchVO createDispatchList(DispatchVO dipatchList); |
| 前置条件 |  |
| 后置条件 | 返回给界面完整的派件单信息 |
| TransportController. saveDispatchList | 语法 | public ResultMessage saveDispatchList(DispatchVO dipatchList); |
| 前置条件 | dispatchList的信息都符合规范 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| TransportController. findDispatchList | 语法 | public DispatchVO findDispatchList(String id) |
| 前置条件 | DispatchlList的信息都符合规范且完整 |
|  |  |
| 后置条件 | 返回涉及的领域对象的数据 |
| TransportController. createReceiveList | 语法 | public ReceiveVO createReceiveList(ReceiveVO ReceiveList); |
| 前置条件 |  |
| 后置条件 | 返回给界面完整的收件单信息 |
| TransportController. saveReceiveList | 语法 | public ResultMessage saveReceiveList(DispatchVO ReceiveList); |
| 前置条件 | ReceiveList的信息都符合规范 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| TransportController. findReceiveList | 语法 | public ReceiveVO findReceiveList(String id); |
| 前置条件 | ReceiveList的信息都符合规范且完整 |
| 后置条件 | 返回涉及的领域对象的数据 |
| TransportController. createLoadcarList | 语法 | public LoadcarVO createLoadcarList(LoadcarVO LoadcarList); |
| 前置条件 |  |
| 后置条件 | 返回给界面完整的装车单信息 |
| TransportController. saveLoadcarList | 语法 | public ResultMessage saveLoadcarList(LoadcarVO LoadcarList); |
| 前置条件 | LoadcarList的信息都符合规范 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| TransportController. findUnpiadLoadCar | 语法 | public ArrayList<LoadCarVO> findUnpaidLoadCar() |
| 前置条件 |  |
| 后置条件 | 返回当前所有未付款的装车单的信息 |
| TransportController. addVehicle | 语法 | public VehicleVO addVehicle(long plateNum); |
| 前置条件 | 启动一个增加车辆的操作 |
| 后置条件 | 返回给界面完整的车辆信息 |
| TransportController. saveVehicleInfor | 语法 | public ResultMessage saveVehicleInfor(VehicleVO vehicleInfor); |
| 前置条件 | 车辆信息都符合规范且完整 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| TransportController. deleteVehicle | 语法 | public ResultMessage deleteVehicle(String vehicleNum); |
| 前置条件 | 车辆代号符合规范 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| TransportController. updateVehicle | 语法 | public ResultMessage updateVehicle(VehicleVO modified ); |
| 前置条件 | 修改过后的车辆信息都符合规范且完整 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| TransportController. findVehicle | 语法 | public VehicleVO findVehicle(String vehicleNum); |
| 前置条件 | 车辆代号符合规范 |
| 后置条件 | 返回给界面完整的车辆信息 |
| TransportController.  showAllVehicle | 语法 | public ArrayList<VehicleVO> showAllVehicle() |
| 前置条件 | 无 |
| 后置条件 | 返回当前系统所有的车辆信息 |
| TransportController.addDriver | 语法 | public DriverVO addDriver(DriverVO baseMessage); |
| 前置条件 | 司机的基本信息完整 |
| 后置条件 | 返回给界面完整的司机信息 |
| TransportController. saveDriverInfor | 语法 | public ResultMessage saveDriverInfor(DriverVO driverInfor); |
| 前置条件 | 司机所有信息完整 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| TransportController. deleteDriver | 语法 | public ResultMessage deleteDriver(String driverNum); |
| 前置条件 | 司机代号符合规范 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| TransportController. updateDriver | 语法 | public ResultMessage updateDriver(DriverVO modified); |
| 前置条件 | 修改过后的司机信息都符合规范且完整 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| TransportController. findDriverInfor | 语法 | public DriverVO findDriverInfor(String driverNum); |
| 前置条件 | 司机代号符合规范 |
| 后置条件 | 返回给界面完整的司机信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| TransportDataService  .addCommodity (CommodityPO commodity): boolean | 添加快递的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteCommodity (long id): boolean | 根据输入的id，删除该快递的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findCommodity (long id): CommodityPO | 根据输入的id，查找快递的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateCommodity (CommodityPO commodity): boolean | 根据传入的快递的id，找到对对应的快递持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addSend (SendPO Send): boolean | 添加寄件单的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteSend (long id): boolean | 根据输入的id，删除该寄件单的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findSend (long id): SendPO | 根据输入的id，查找寄件单的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateSend (SendPO Send): boolean | 根据传入的寄件单的id，找到对应的寄件单持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addLoad (LoadPO Load): boolean | 添加装运单的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteLoad (long id): boolean | 根据输入的id，删除该装运单的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findLoad (long id): LoadPO | 根据输入的id，查找装运单的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateLoad (LoadPO Load): boolean | 根据传入的装运单的id，找到对应的装运单持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addArrival (ArrivalPO Arrival): boolean | 添加到达单的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteArrival (long id): boolean | 根据输入的id，删除该到达单的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findArrival (long id): ArrivalPO | 根据输入的id，查找到达单的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateArrival (ArrivalPO Arrival): boolean | 根据传入的到达单的id，找到对应的到达单持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addDispatch (DispatchPO Dispatch): boolean | 添加派件单的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteDispatch (long id): boolean | 根据输入的id，删除该派件单的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findDispatch (long id): DispatchPO | 根据输入的id，查找派件单的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateDispatch (DispatchPO Dispatch): boolean | 根据传入的派件单的id，找到对应的派件单持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addVehicle (VehiclePO Vehicle): boolean | 添加司机的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteVehicle (long id): boolean | 根据输入的id，删除该司机的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findVehicle (long id): VehiclePO | 根据输入的id，查找司机的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateVehicle (VehiclePO Vehicle): boolean | 根据传入的司机的id，找到对应的司机持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addDriver (DriverPO Driver): boolean | 添加车辆的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteDriver (long id): boolean | 根据输入的id，删除该车辆的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findDriver (long id): DriverPO | 根据输入的id，查找车辆的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateDriver (DriverPO Driver): boolean | 根据传入的车辆的id，找到对应的车辆持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addWarehouse (WarehousePO Warehouse): boolean | 添加仓库的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteWarehouse (long id): boolean | 根据输入的id，删除该仓库的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findWarehouse (long id): WarehousePO | 根据输入的id，查找仓库的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateWarehouse (WarehousePO Warehouse): boolean | 根据传入的仓库的id，找到对应的仓库持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addCheckin (CheckinPO Checkin): boolean | 添加入库单的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteCheckin (long id): boolean | 根据输入的id，删除该入库单的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findCheckin (long id): CheckinPO | 根据输入的id，查找入库单的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateCheckin (CheckinPO Checkin): boolean | 根据传入的入库单的id，找到对应的入库单持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addCheckout (CheckoutPO Checkout): boolean | 添加出库单的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteCheckout (long id): boolean | 根据输入的id，删除该出库单的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findCheckout (long id): CheckoutPO | 根据输入的id，查找出库单的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateCheckout (CheckoutPO Checkout): boolean | 根据传入的出库单的id，找到对应的出库单持久化对象进行更新，并返回是否成功更新该持久化对象 | |
| TransportDataService  .addReceive (ReceivePO Receive): boolean | 添加收件单的单一持久化对象，并返回是否成功添加该持久化对象 | |
| TransportDataService  .deleteReceive (long id): boolean | 根据输入的id，删除该收件单的单一持久化对象，并返回是否删除成功该持久化对象 | |
| TransportDataService  .findReceive (long id): ReceivePO | 根据输入的id，查找收件单的单一持久化对象，并返回该持久化对象 | |
| TransportDataService  .updateReceive (ReceivePO Receive): boolean | 根据传入的收件单的id，找到对应的收件单持久化对象进行更新，并返回是否成功更新该持久化对象 | |

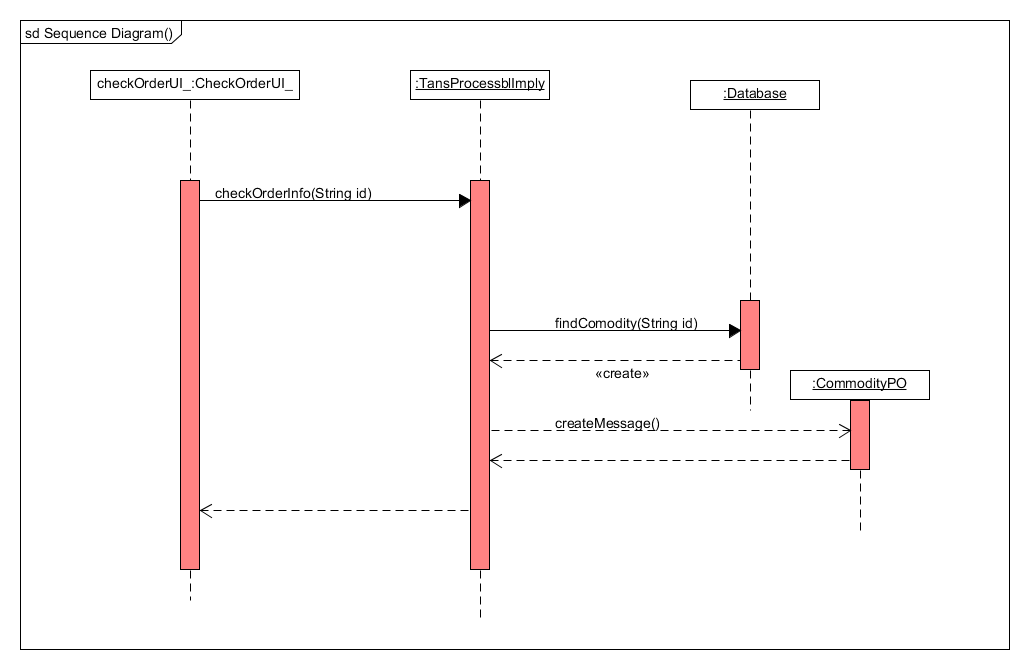
（4）业务逻辑层的动态模型

下图表明了快递物流管理系统中，当用户生成并保存寄件单时，运输业务逻辑处理的相关对象之间的协作。

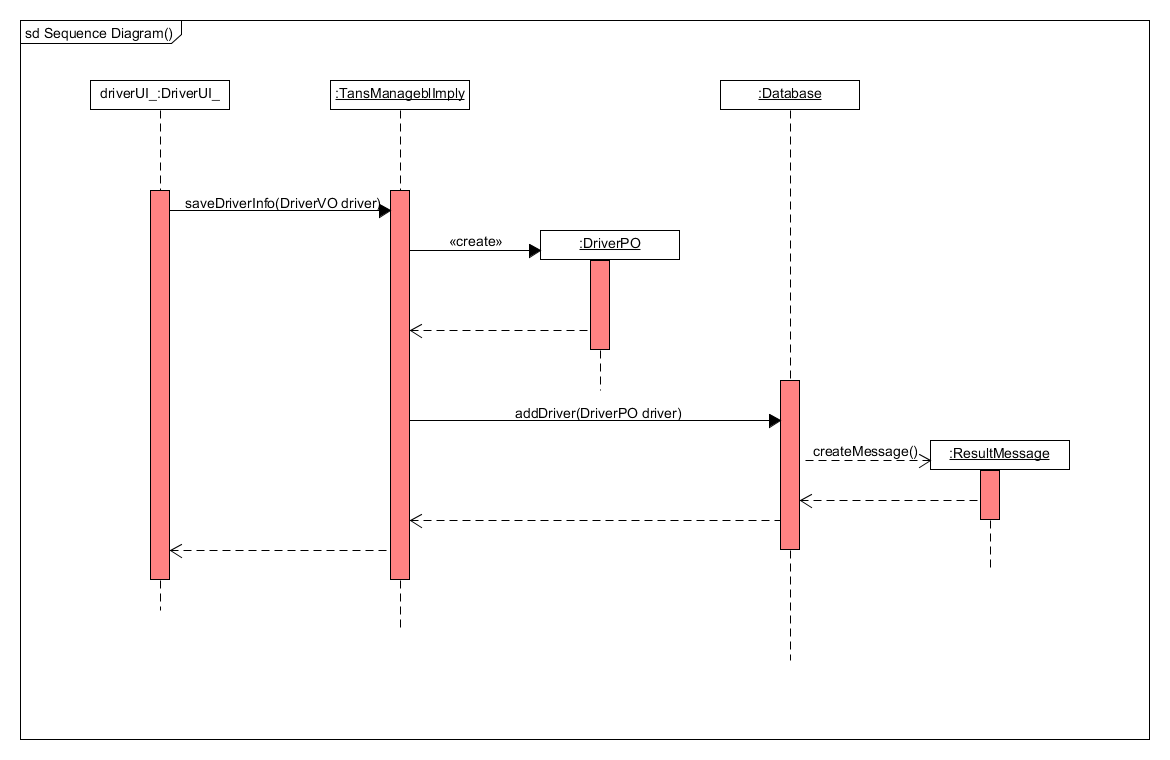
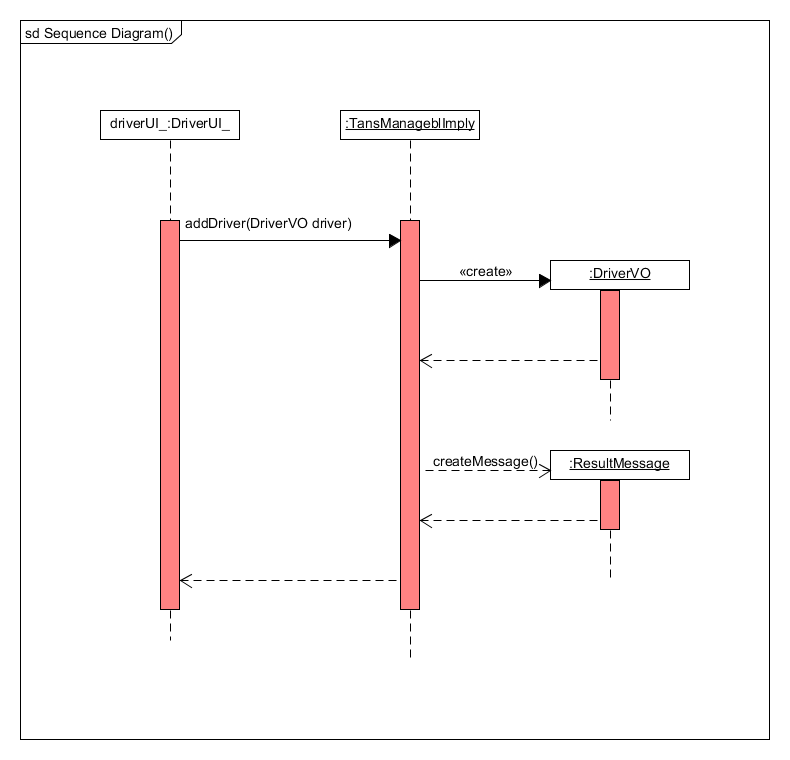


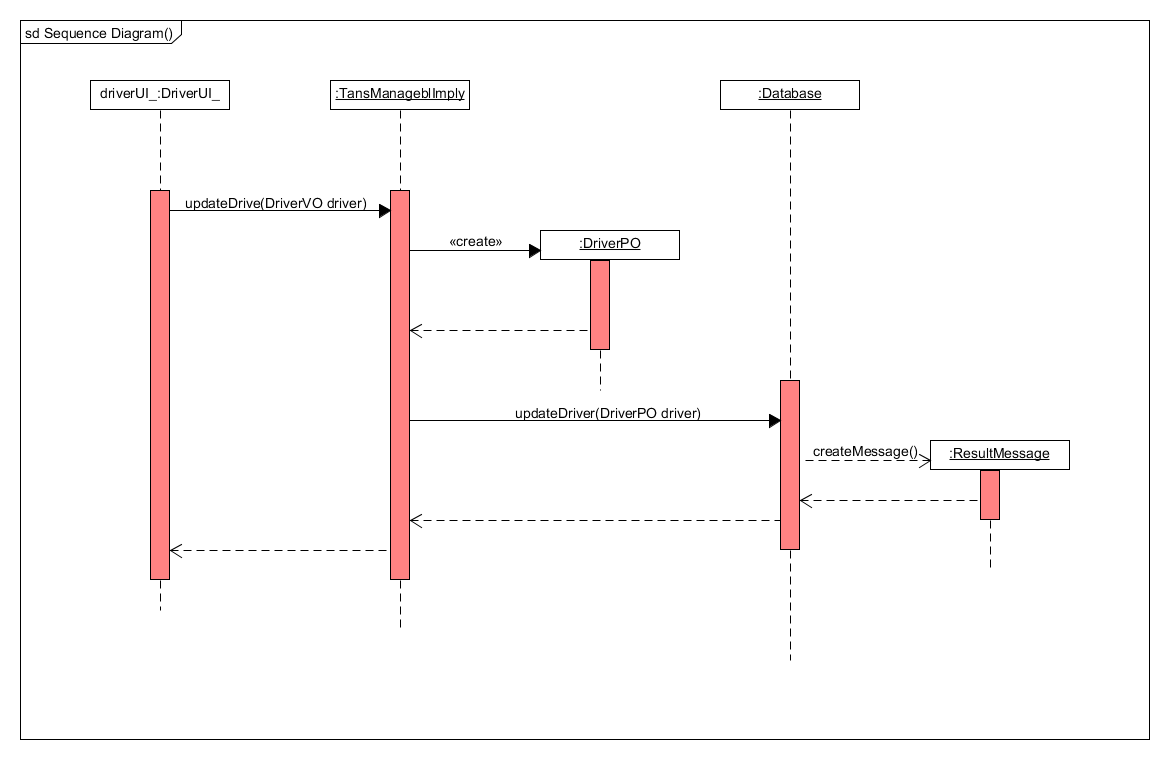


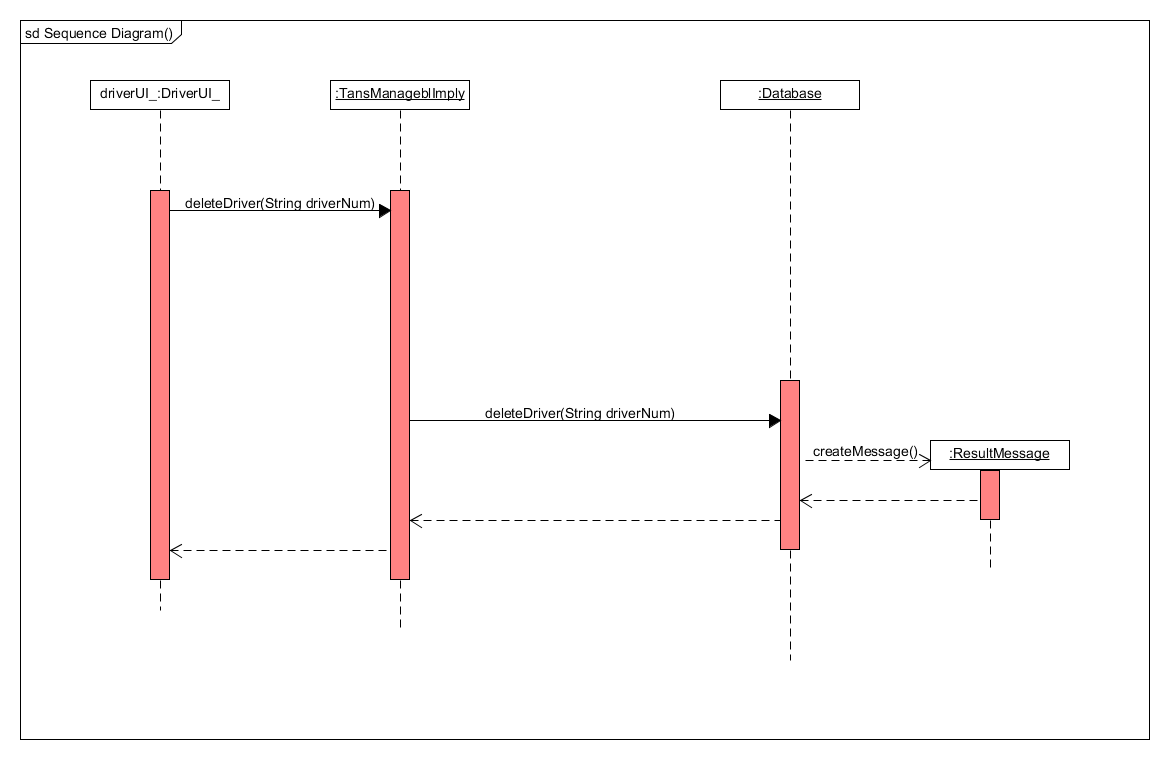
下图表示了用户查询物流状态和历史轨迹时，运输业务逻辑处理的相关对象之间的操作。



下图表示了用户管理司机信息时，运输业务逻辑处理的相关对象之间的操作。







(5)业务逻辑层的设计原理

分别创建一个类实现运输服务两个接口文件中的方法。

### 5.1.2Warehousebl模块

1. 模块概述

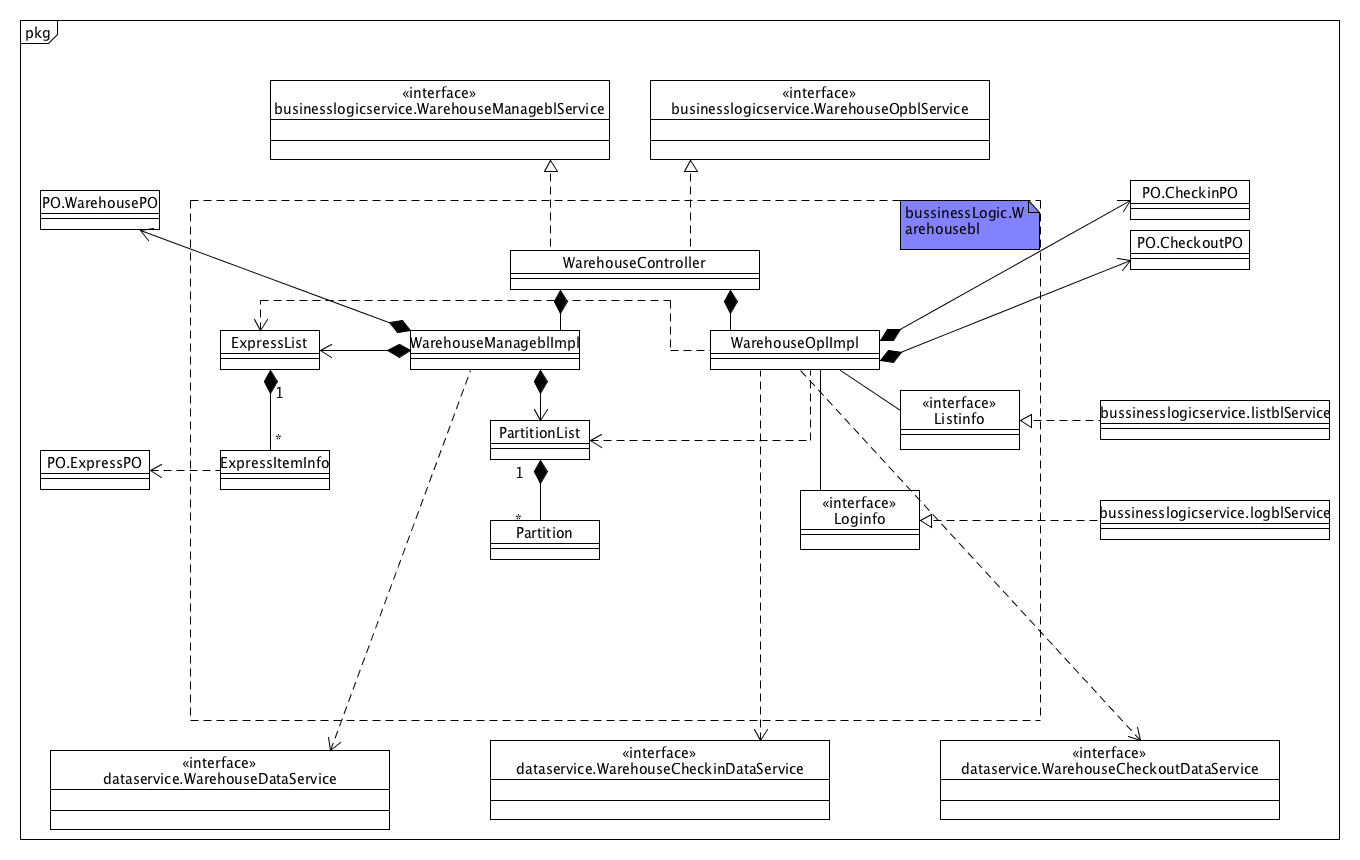
Warehousebl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

Warehousebl模块的职责及接口参见软件体系机构描述文档5.3.2 Warehousebl模块的接口规范。

（2）整体结构

在展示层和业务逻辑层之间，我们添加了businesslogicservice.WareHouseManageblService和WarehouseOpblService接口，分别处理仓库管理和出入库操作。在业务逻辑层和数据层之前，增加了dataService.WarehouseCheckinDataService和dataService.WarehouseCheckoutDataService和dataService.WarehouseDataService。为了隔离控制和实现，增加

了WarehouseController，这样WarehouseController可以把这两类操作分别委托给两个接口的实现。WarehousePO是存储仓库信息的持久化对象。ExpressList存储的是当前仓库所存储的所有快递的信息，持有ExpressItemIno的引用。ExpressPO是存储快递信息的持久化对象。类似的，PartitionList存储的是当前仓库的四个分区，持有Partition的引用。因为每次出入库的操作都会有日志记录和单据存储，而为了消除循环依赖，添加了Listinfo、Loginfo接口。



Warehousebl模块各个类的职责如下表所示。

Warehouse模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| WarehouseController | 负责提供仓库管理人员界面所需要的服务 |
| WarehouseManageblImpl | 实现仓库管理方面的服务 |
| WarehouseOpblImpl | 实现出入库的操作 |

1. 模块内部类的接口规范

WarehouseController的接口规范如下表所示。

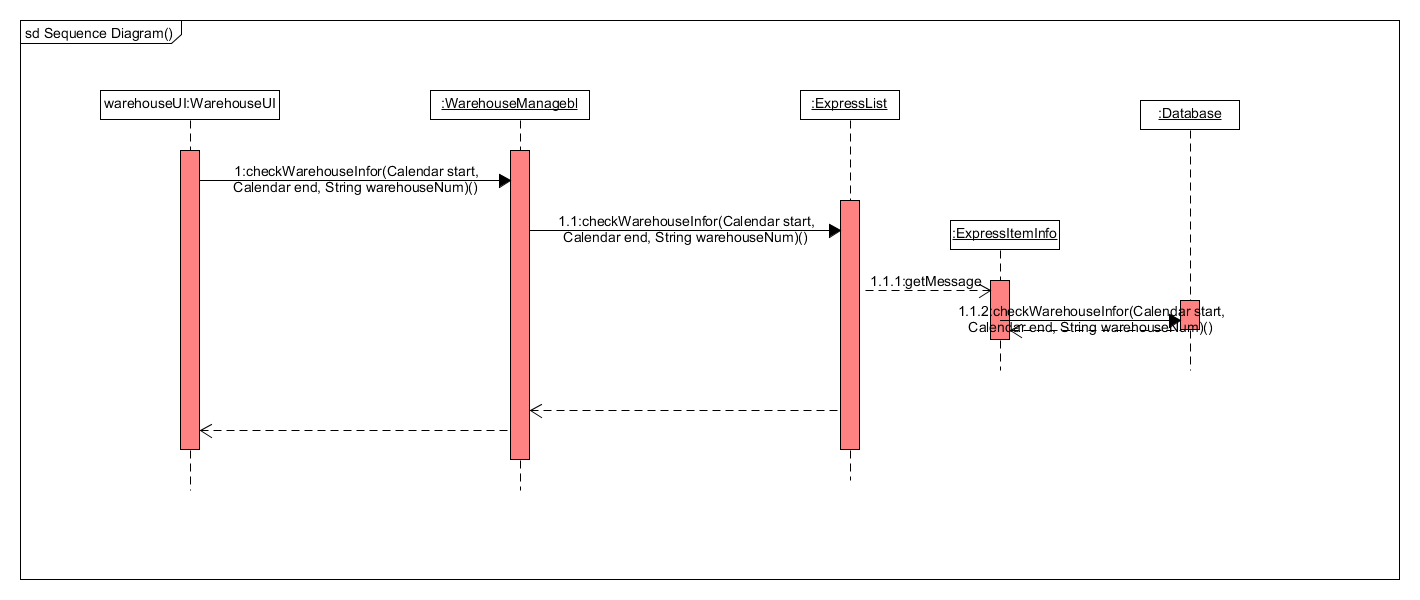
WarehouseController的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| WarehousController. createCheckinList | 语法 | public CheckinVO createCheckinList(CheckinVO baseMessage,long warehouseNum); |
| 前置条件 | 入库单的基本信息符合规范 |
| 后置条件 | 返回界面完整的入库单信息 |
| Warehouse. createCheckoutList | 语法 | public CheckoutVO createCheckoutList(CheckoutVO baseMessage, long warehouseNum); |
| 前置条件 | 出库单的基本信息符合规范 |
| 后置条件 | 返回界面完整的出库单信息 |
| Warehouse. checkWarehouseInfor | 语法 | public InventoryInforVO checkWarehouseInfor(Calendar start, Calendar end, long warehouseNum); |
| 前置条件 | 时间段的设定符合规范 |
| 后置条件 | 返回该时间段的库存信息 |
| Warehouse. exportExcel | 语法 | public Boolean exportExcel(InventoryExcelVO, long warehouseNum); |
| 前置条件 | 用户已进行过当天的库存查看 |
| 后置条件 | 导出excel表格 |
| Warehouse. setCordon | 语法 | public Boolean setCordon(double cordon, long warehouseNum); |
| 前置条件 | 警戒线的值符合规范 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Warehouse. showPartition | 语法 | public PartitionVO showPartition(long warehouseNum); |
| 前置条件 | 库存处于报警状态 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Warehouse. modifyPartition | 语法 | public Boolean modifyPartition(PartitionVO modified, long warehouseNum ); |
| 前置条件 | 修改过的分区信息符合规范 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |
| Warehouse. initialize | 语法 | public Boolean initialize (PartitionVO, double cordon, long warehouseNum); |
| 前置条件 | 分区信息和警戒线都符合规范 |
| 后置条件 | 持久化更新涉及的领域对象的数据 |

1. 业务逻辑层的动态模型

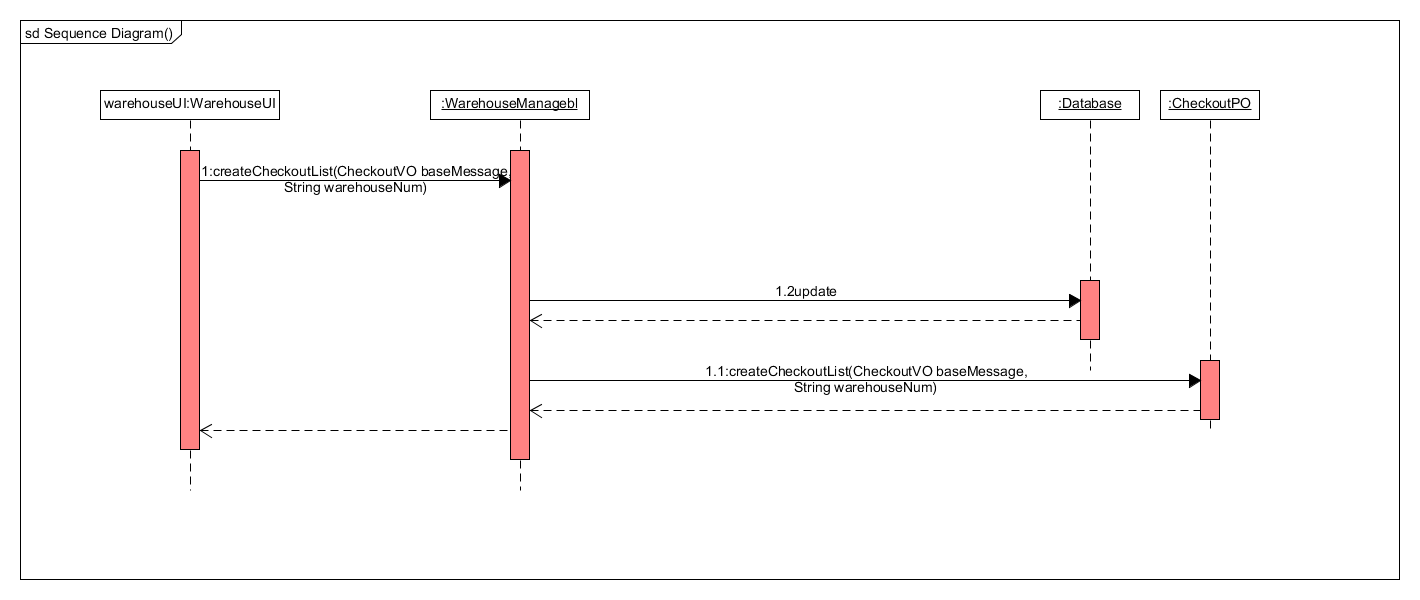
下图表明了在物流管理系统中，当用户想要查看库存信息时，输入仓库的编码和想要查看的时间段之后，仓库业务逻辑处理的相关对象的操作。



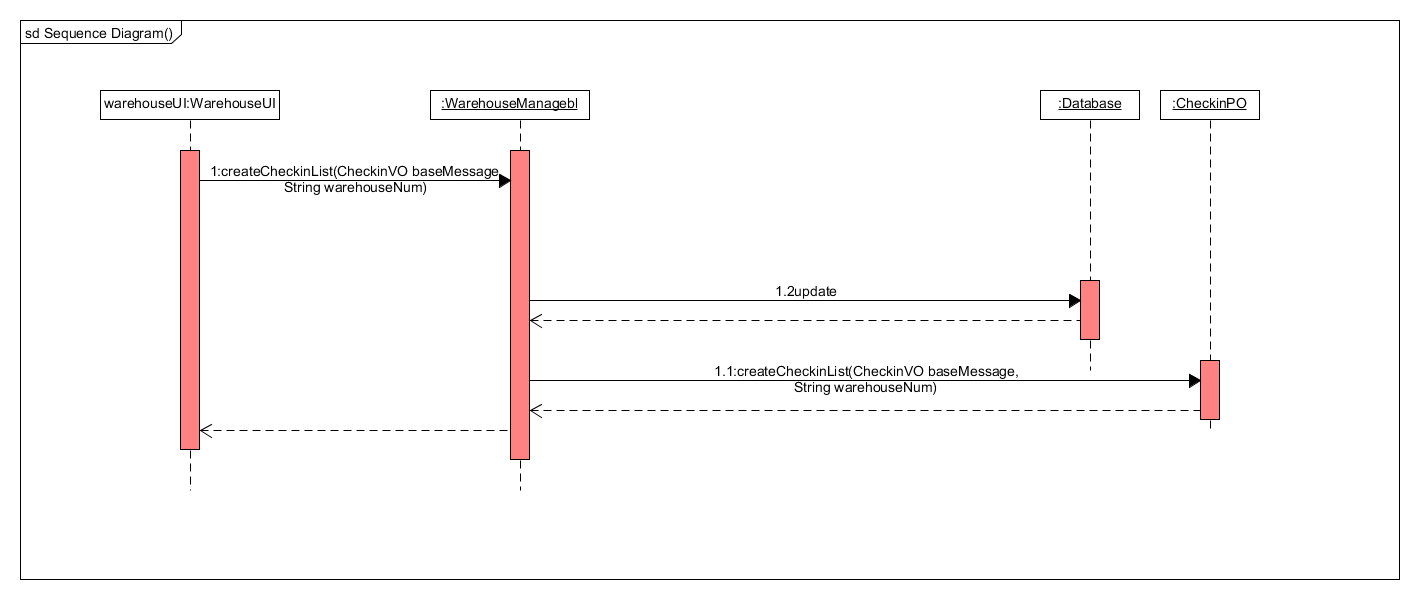
下图表明了当用户想要查看库存分区情况时，输入仓库的编码后，仓库业务逻辑处理的相关对象的操作。



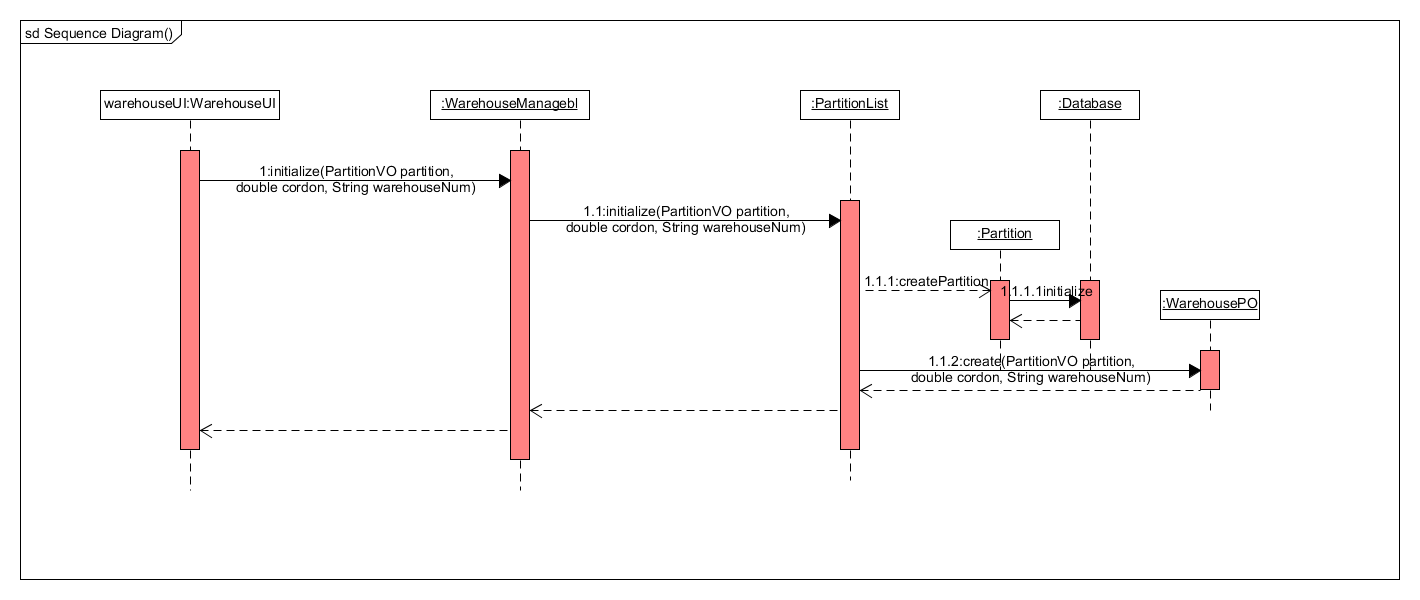
下图表明了当用户需要建立出库单时，输入仓库的编码后，仓库业务逻辑处理的相关对象的操作。



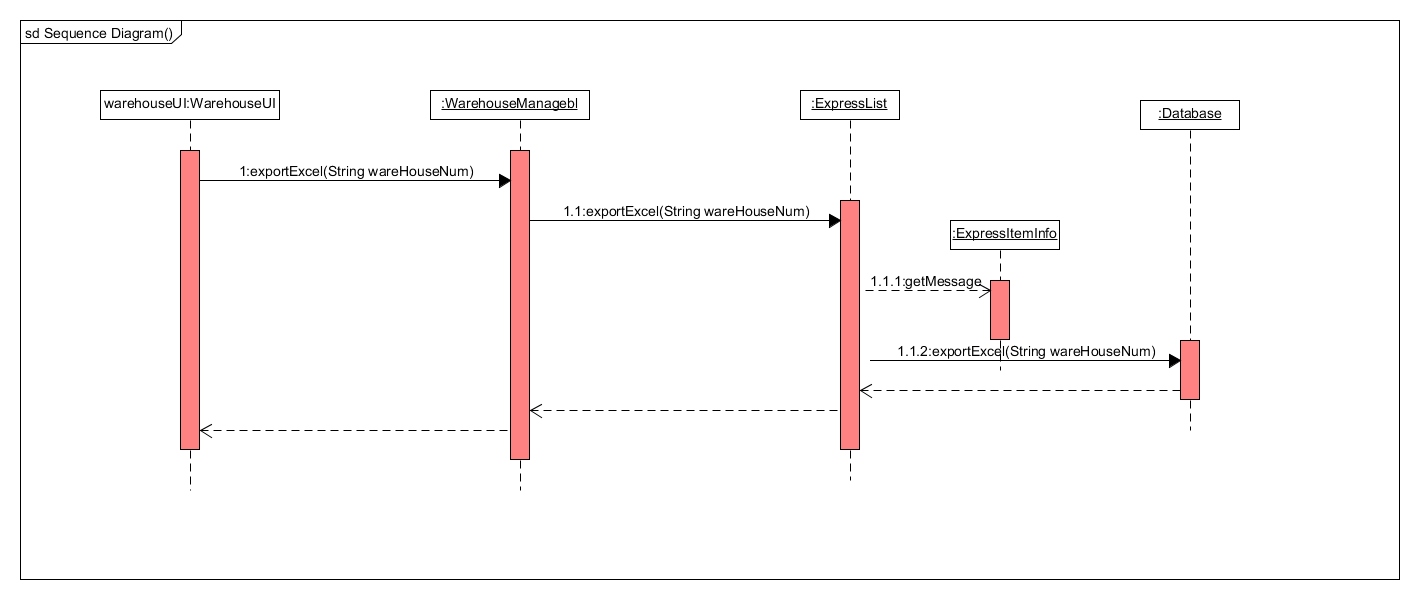
下图表明了当用户需要建立入库单时，输入仓库的编码后，仓库业务逻辑处理的相关对象的操作。



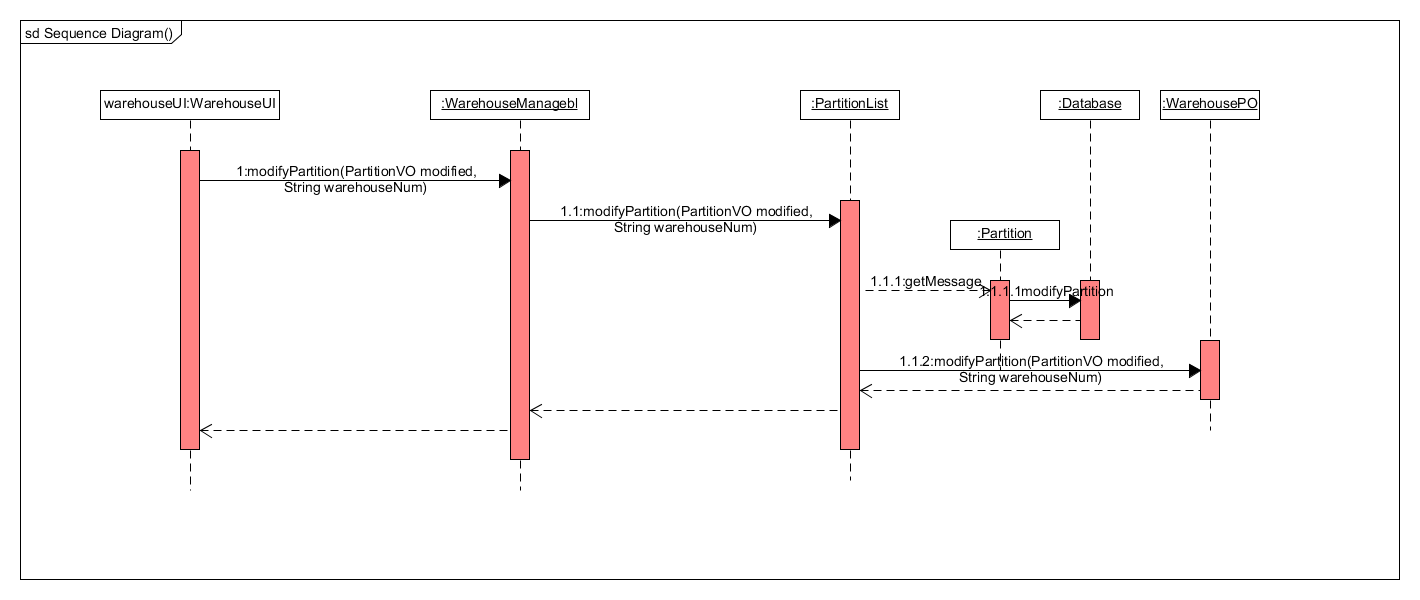
下图表明了当用户需要初始化库存信息时，输入仓库的编码后，仓库业务逻辑处理的相关对象的操作。



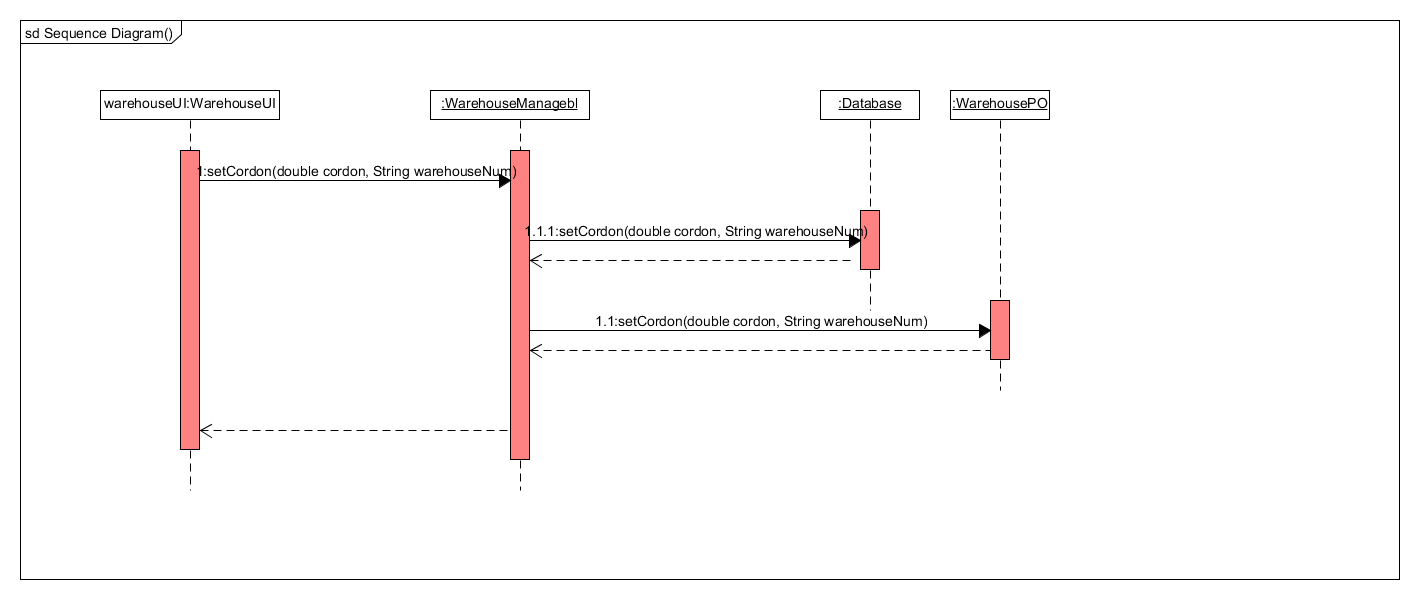
下图表明了当用户需要导出当前仓库信息为excel时，输入仓库的编码后，仓库业务逻辑处理的相关对象的操作。



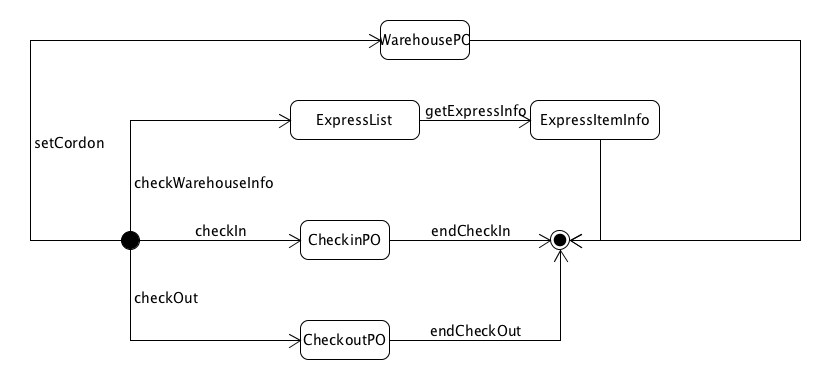
下图表明了当用户需要修改分区时，输入仓库的编码后，仓库业务逻辑处理的相关对象的操作。



下图表明了当用户需要修改警戒线时，输入仓库的编码后，仓库业务逻辑处理的相关对象的操作。



下图描述了Warehouse对象的生存期内的状态序列、引起转移的事件以及因状态转移而伴随的动作。



### 5.1.3Personnelbl模块

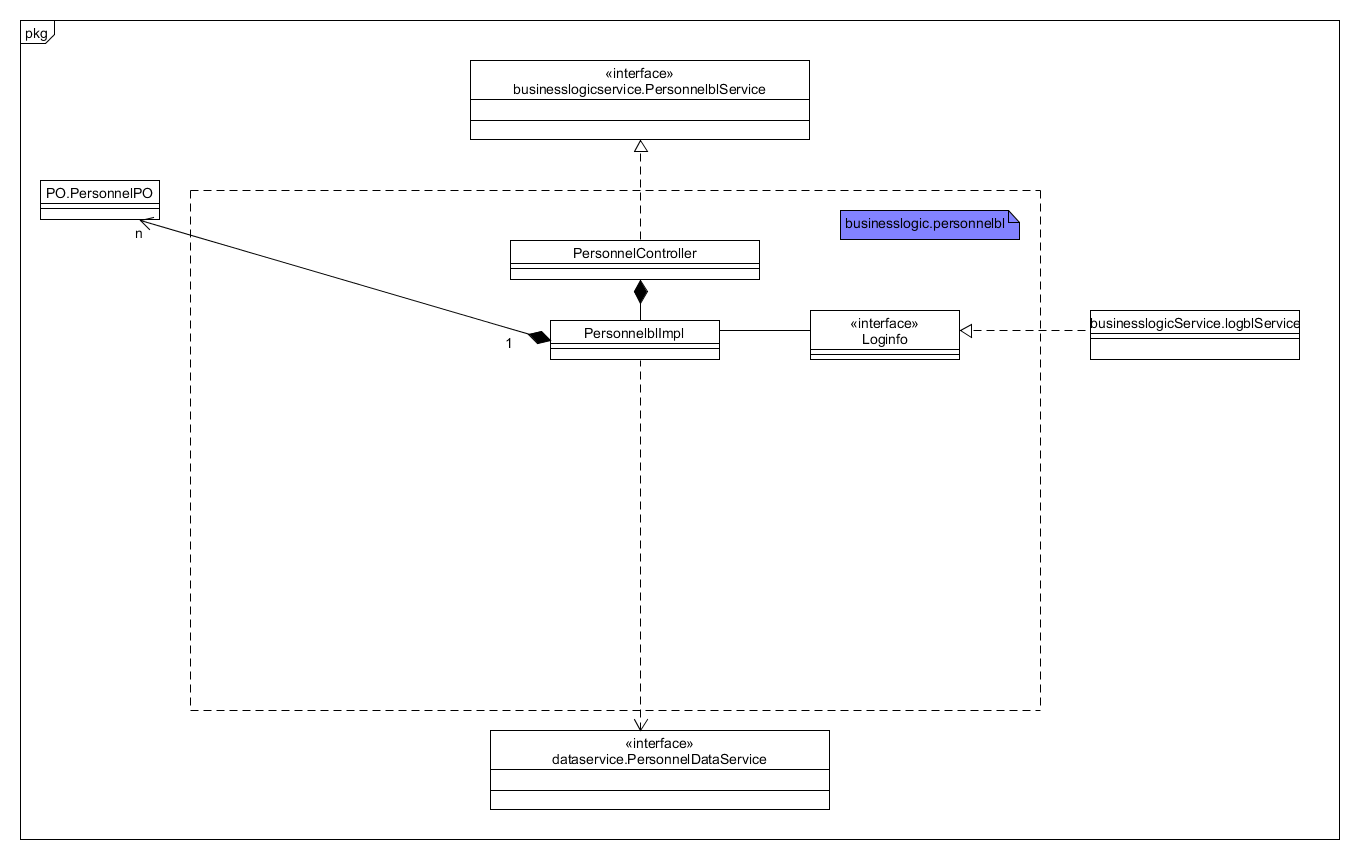
（1）模块概述

Personnelbl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。

Personnelbl模块的职责及接口参见软件体系机构描述文档5.3.2 Personnelbl模块的接口规范。

（2）整体结构

在展示层和业务逻辑层之间，我们添加了businesslogicservice.PersonnelblService接口，处理人员管理的操作。业务逻辑层和数据层之前添加了dataService.PersonnelDataService接口。为了隔离控制和实现，增加了personnelController，这样personnelController会把业务逻辑委托给PersonnelblImpl来实现。PersonnelPO是人员信息的持久化对象。每次人员管理的操作都会有日志记录，为了防止循环依赖，增加了LogInfo接口。



personnelbl模块各个类的职责如下表所示。

personnel模块各个类的职责

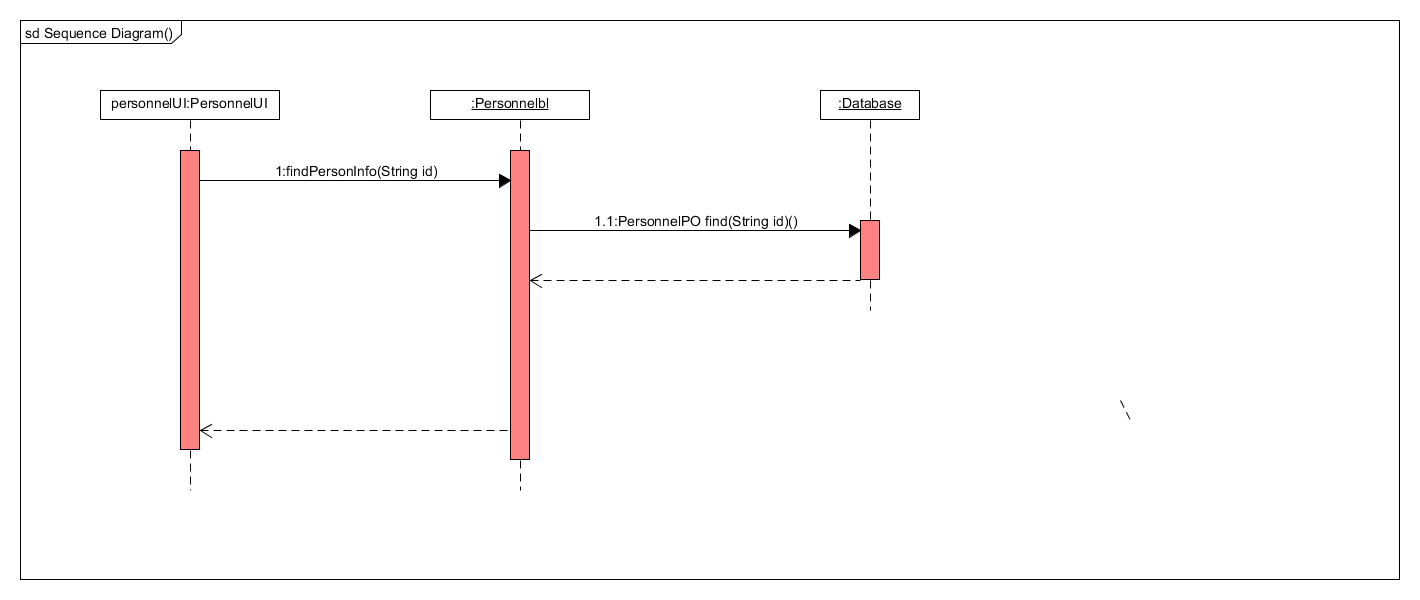
|  |  |
| --- | --- |
| 模块 | 职责 |
| PersonnelController | 负责提供人员管理界面所需要的服务 |
| PersonnelblImpl | 实现人员管理方面的服务 |

（3）模块内部的接口规范

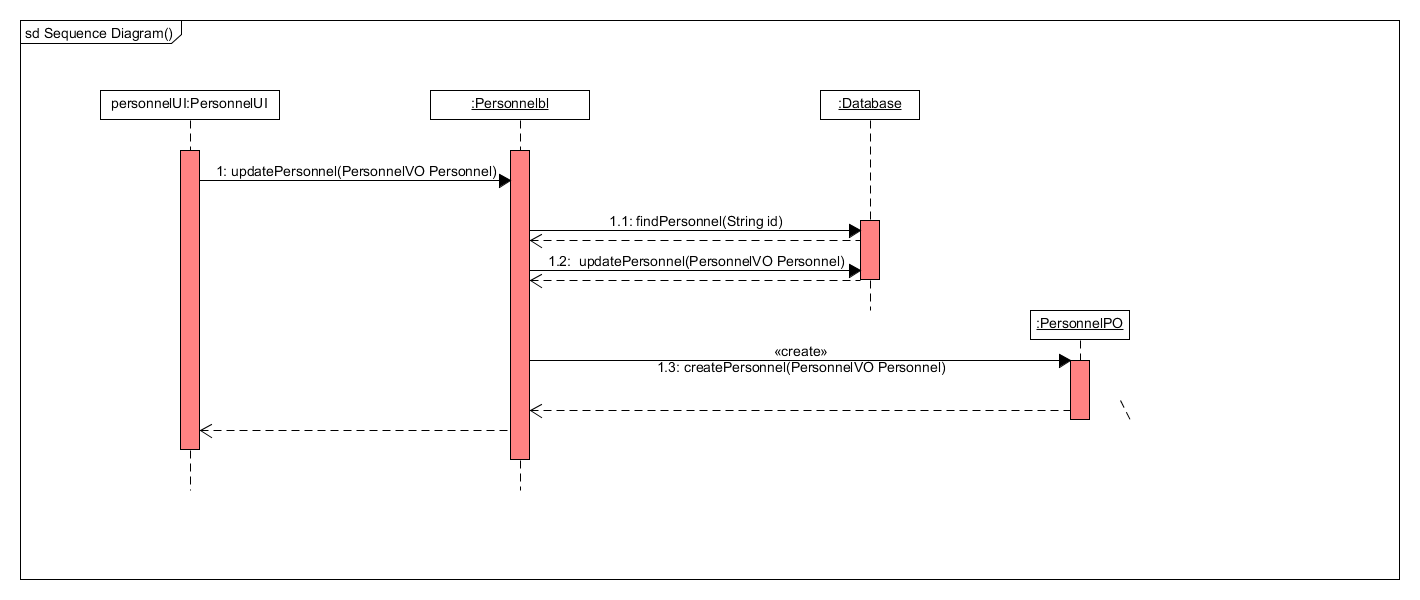
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Personnel. findPersonnelInfo | 语法 | public ArrayList<PersonnelVO> findPersonInfo(String id) |
| 前置条件 | ID符合规范 |
| 后置条件 | 查找是否存在相应的Personnel，如果存在返回相应的Personnel信息 |
| Personnel. deletePersonnel | 语法 | public ResultMessage deletePersonnel(String id) |
| 前置条件 | ID符合规范 |
| 后置条件 | 查找是否存在相应的Personnel， 如果删除成功返回true，否则返回对应的错误信息 |
| Personnel. updatePersonnel | 语法 | Public ResultMessage updatePersonnel(PersonnelVO Personnel) |
| 前置条件 | 更改过的人员信息符合规范 |
| 后置条件 | 根据id查找Personnel，并修改相应的数据，如果修改成功返回true，否则返回对应的错误信息 |
| Personnel. addPersonnel | 语法 | public ResultMessage addPersonnel(PersonnelVO Personnel) |
| 前置条件 | 无 |
| 后置条件 | 如果增加成功返回true，否则返回对应的错误信息 |
| Personnel. showAllPersonnel | 语法 | public ArrayList<PersonnelVO> showAllPersonnel () |
| 前置条件 | 无 |
| 后置条件 | 返回当前系统所有员工的信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| PersonnelDataService  .findPersonnel(String id):PersonnelPo | 根据id返回一个人员持久化对象 | |
| PersonnelDataService  .deletePersonnel(String id):boolean | 根据id删除一个人员持久化对象 | |
| PersonnelDataService  .updatePersonnel(PersonnelPo personnel):ResultMessage | 查找相应的人员持久化对象并替换掉 | |
| PersonnelDataService  .addPersonnel(PersonnelPo personnel): ResultMessage | 根据传入信息增加一个人员持久化对象 | |
| PersonnelDataService.  showAllPersonnel():ArrayList<PersonnelVO> | 返回系统中所有人员持久化对象 | |

（4）业务逻辑层的动态模型

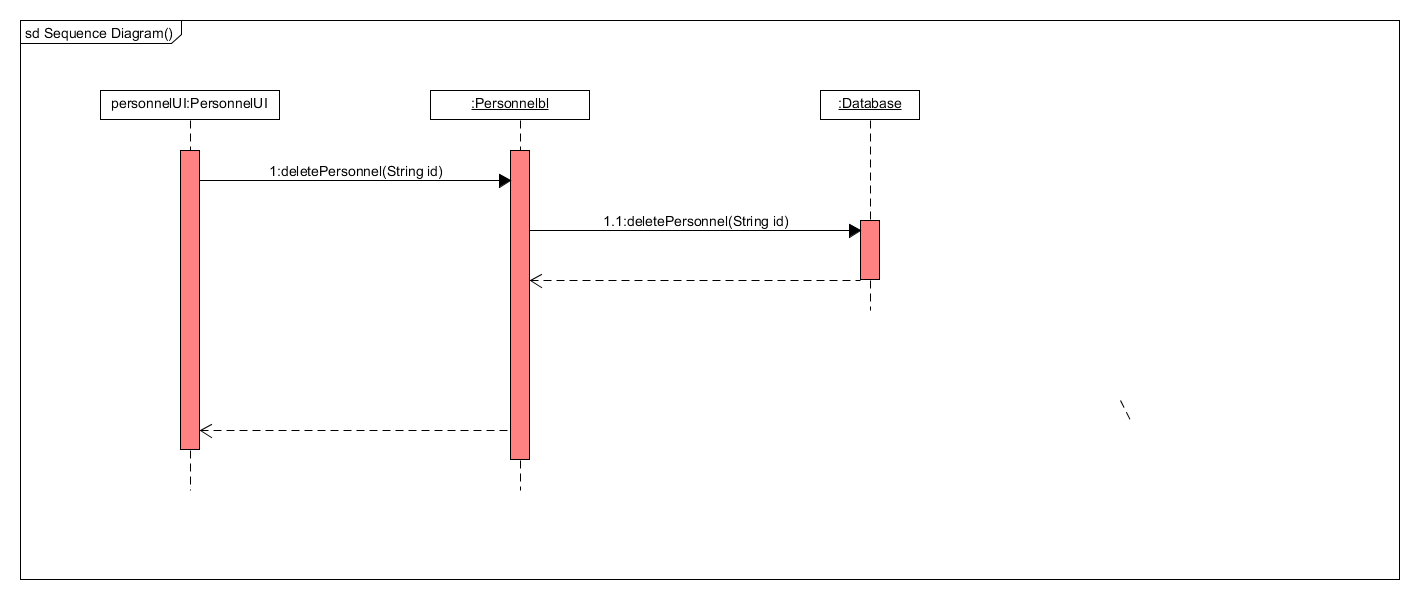
下图表示了当用户想要查看人员信息时，输入人员的id后，人员业务逻辑处理的相关对象之间的操作。



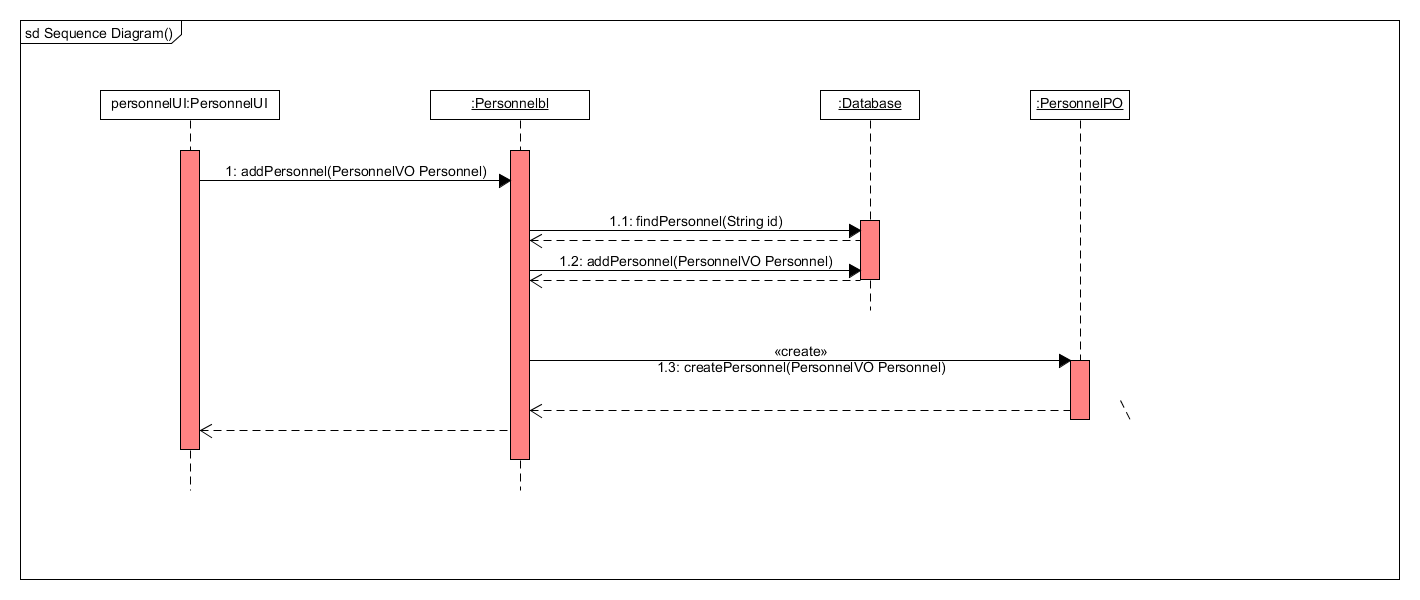
下图表示了当用户想要修改人员信息时，输入人员的id和要要修改的信息后，人员业务逻辑处理的相关对象之间的操作。



下图表示了当用户想要删除人员信息时，输入人员的id后，人员业务逻辑处理的相关对象之间的操作。



下图表示了当用户想要增加人员信息时，输入人员的id和其他信息后，人员业务逻辑处理的相关对象之间的操作。



### 5.1.4Departmentbl模块

(1)模块概述

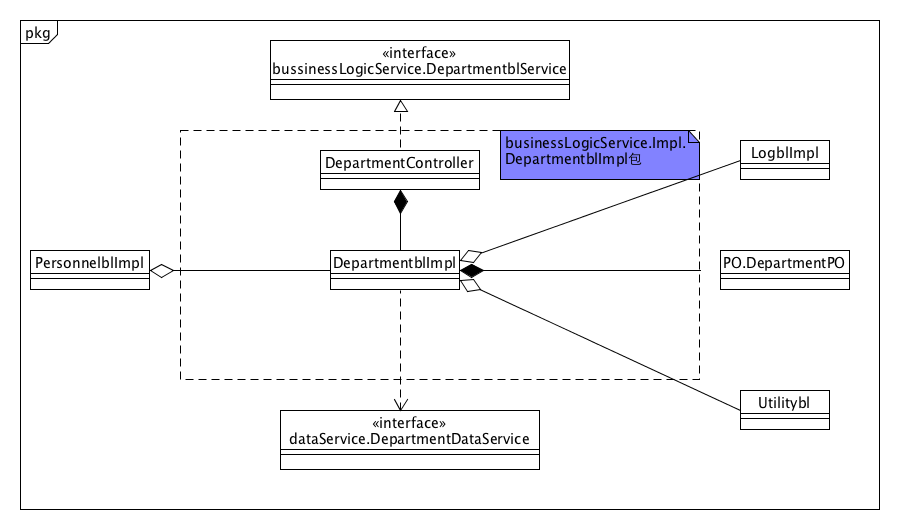
Departmentbl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

Departmentbl模块的职责和接口参见软件体系结构描述文档。

(2)整体结构

根据体系结构的设计，系统分为展示层、业务逻辑层和数据层，并在展示层和业务逻辑层之间添加bussinssLogicServi ce.DepartmentblService接口、在业务逻辑层和数据层之间添加dataService.DepartmentDataService接口，从而增加了系统的灵活性。我们增加了DepartmentController从而隔离了业务逻辑职责和逻辑控制职责，这样DepartmentController会将业务逻辑处理委托给DepartmentblImpl对象。DepartmentPO作为用户信息的持久化对象被添加到设计模型中去的。LogImpl用来进行对管理用户操作时产生的系统日志的记录。

Departmentbl模块的设计如图所示。



Departmentbl模块的各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| DepartmentController | 负责实现用户管理界面所需要的服务 |
| DepartmentblImpl | 用户管理领域的模型对象，可以帮助完成用户管理界面所需要的服务 |

(3)模块内部类的接口规范

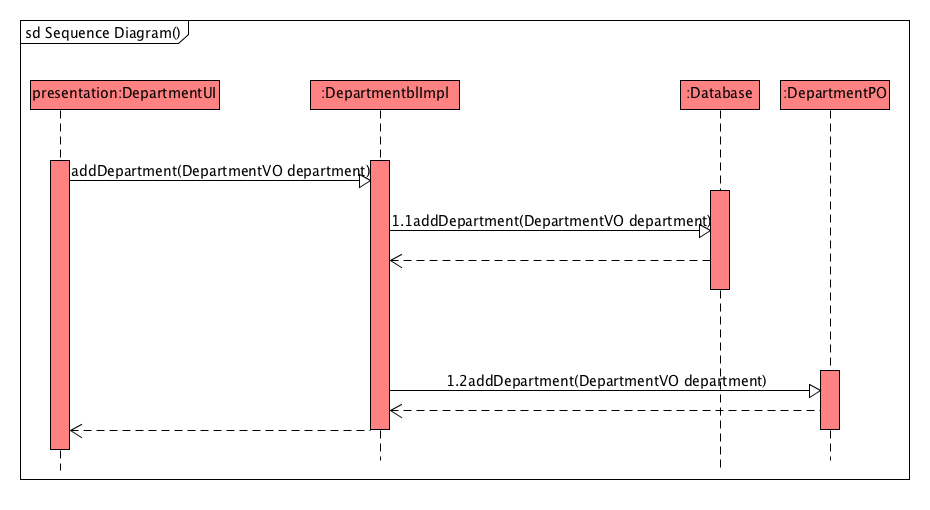
DepartmentController和Departmentbl的接口规范如下表所示。

DepartmentController的接口规范

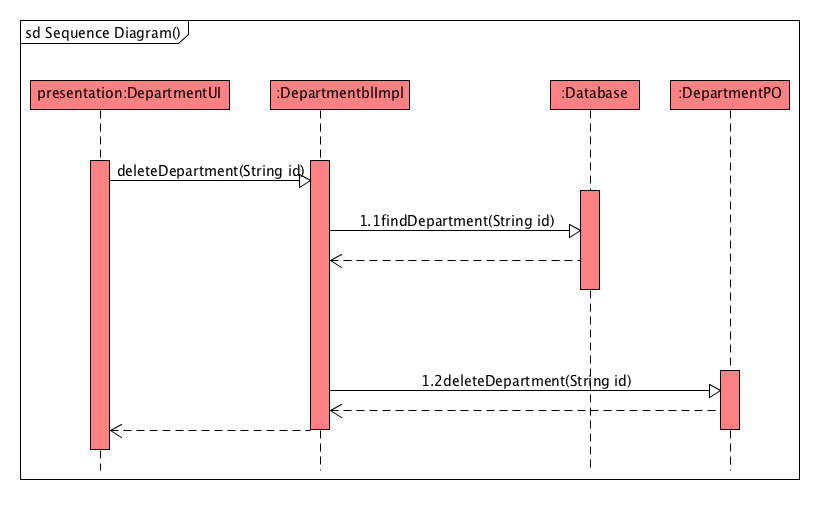
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DepartmentController. getDepartmentInfo | 语法 | Public DepartmentVO getDepartInfo(String id) |
| 前置条件 | 无 |
| 后置条件 | 查找是否存在相应的Department，如果存在返回相应的Department信息 |
| DepartmentController.  deleteDepartment | 语法 | Public ResultMessage deleteDepartment(String id) |
| 前置条件 | 无 |
| 后置条件 | 查找是否存在相应的Department， 如果删除成功返回true，否则返回对应的错误信息 |
| DepartmentController. updateDepartment | 语法 | Public ResultMessage updateDepartment(DepartmentVO Department) |
| 前置条件 | 无 |
| 后置条件 | 根据id查找Department，并修改相应的数据，如果修改成功返回true，否则返回对应的错误信息 |
| DepartmentController. addDepartment | 语法 | Public ResultMessage addDepartment(DepartmentVO Department) |
| 前置条件 | 无 |
| 后置条件 | 如果增加成功返回true，否则返回对应的错误信息 |
| DepartmentController. addCity | 语法 | Public ResultMessage addCity(CityVO city); |
| 前置条件 | 传入的城市信息符合规范 |
| 后置条件 | 如果增加成功返回true，否则返回对应的错误信息 |
| DepartmentController. findCity | 语法 | Public CityVO findCity(String id); |
| 前置条件 | 系统存在相应的id |
| 后置条件 | 返回城市信息 |
| DepartmentController. deleteCity | 语法 | Public ResultMessage deleteCity(String id); |
| 前置条件 | 无 |
| 后置条件 | 如果删除成功返回true，否则返回对应的错误信息 |
| DepartmentController. updateCity | 语法 | Public ResultMessage updateCity(CityVO city); |
| 前置条件 | 无 |
| 后置条件 | 根据id查找City，并修改相应的数据，如果修改成功返回true，否则返回对应的错误信息 |
| DepartmentController. showAllCities | 语法 | Public ArrayList<CityVO> showAllCities(); |
| 前置条件 | 无 |
| 后置条件 | 返回当前系统中所有城市的信息 |
| DepartmentController. showAllDepartments | 语法 | Public ArrayList<DepartmentVO> showAllDepartments(); |
|  | 前置条件 | 无 |
|  | 后置条件 | 返回当前系统中所有机构的信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| DepartmentDataService  .addDepartment(DepartmentPo department)：ResultMessage | 增加机构的单一持久化对象 | |
| DepartmentDataService  .findDepartment(String id):DepartmentPo | 根据id查找机构的单一持久化对象并返回 | |
| DepartmentDataService  .deleteDepartment(String id):ResultMessage | 根据id删除机构的单一持久化对象 | |
| DepartmentDataService  .updateDepartment(DepartmentPo department):ResultMessage | 查找相应的机构持久化对象并替换成传入的机构持久化对象 | |
| DepartmentDataService.  showAllDepartments() | 返回所有机构的单一持久化对象 | |
| DepartmentDataService  .addCity(CityPo city):ResutMessage | 增加城市持久化对象 | |
| DepartmentDataService  .findCity(String id):CityPO | 根据id返回城市持久化对象 | |
| DepartmentDataService  .updateCity(CityPo city):ResutMessage | 更新城市持久化对象 | |
| DepartmentDataService  .deleteCity(String id):ResultMessage | 删除城市持久化对象 | |
| DepartmentDataService.  showAllCities() | 返回系统中所有城市的单一持久化对象 | |
| LogController.addLog(String explain):ResultMessage | 根据操作增加系统日志持久化对象 | |

(4)业务逻辑层的动态模型

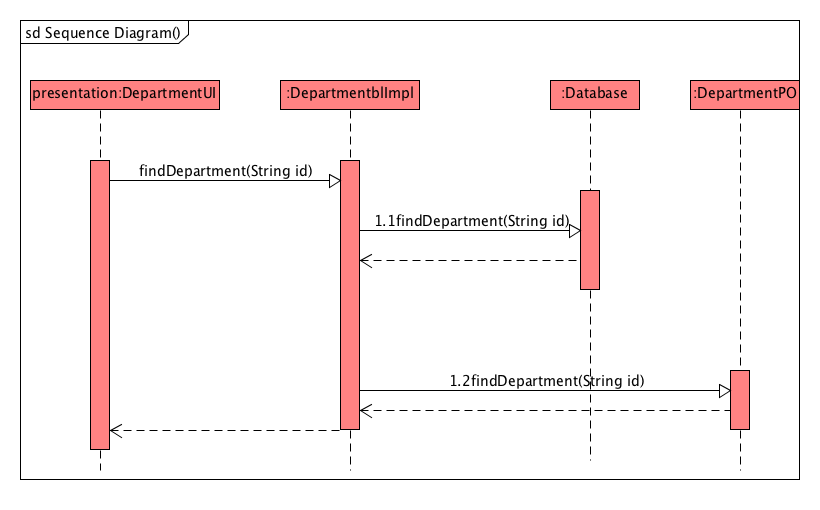
下图表示了物流管理系统中，当用户选择增加机构信息时，Departmentbl处理的相关对象的协作。



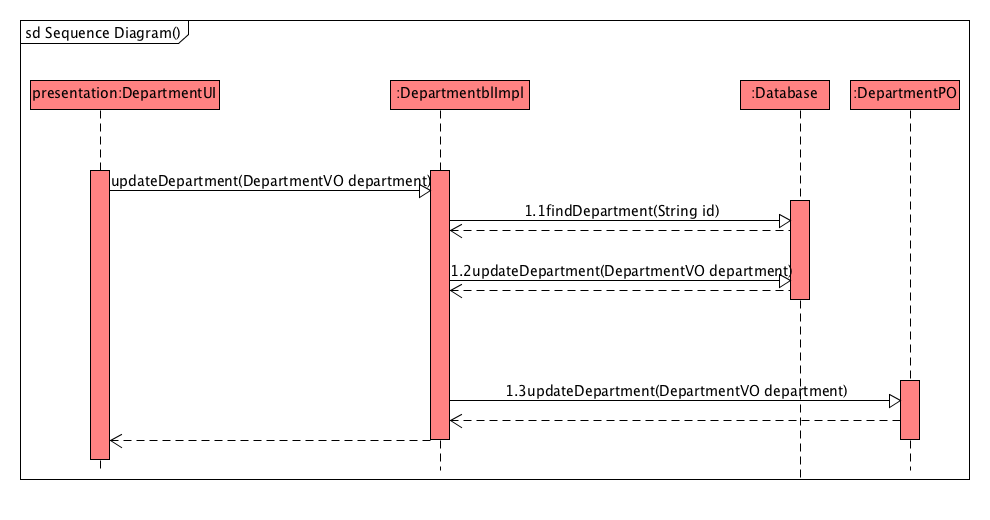
下图表示了物流管理系统中，当用户输入要删除的机构id时，Departmentbl处理的相关对象的协作。



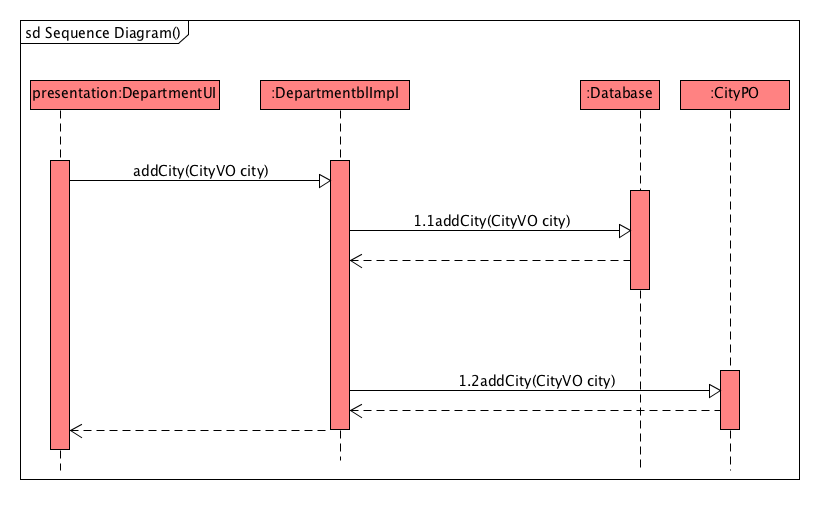
下图表示了物流管理系统中，当用户输入要查询的机构id时，Departmentbl处理的相关对象的协作。



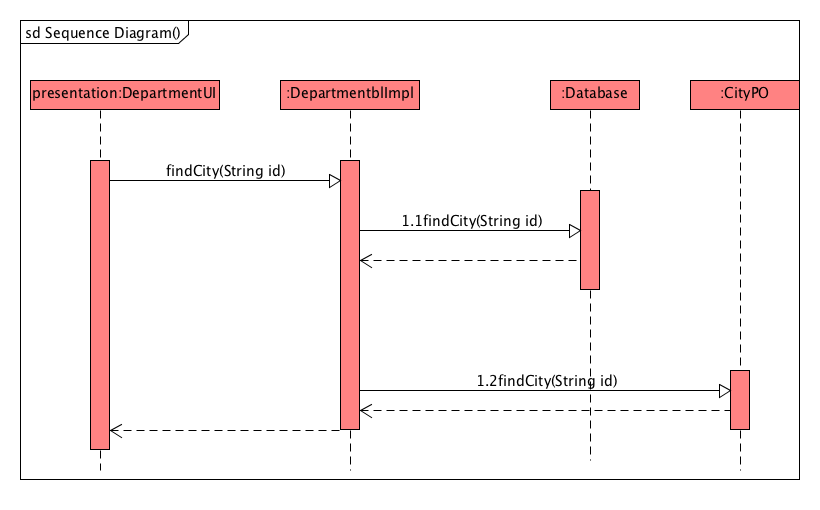
下图表示了物流管理系统中，当用户选择修改机构信息时，Departmentbl处理的相关对象的协作。



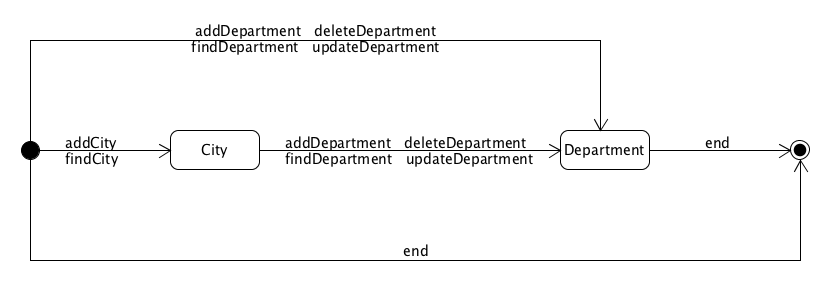
下图表示了物流管理系统中，当用户选择增加城市信息时，Departmentbl处理的相关对象的协作。



下图表示了物流管理系统中，当用户输入要查询的城市id时，Departmentbl处理的相关对象的协作。



如下图所示的状态图描述了Departmentbl对象的生存期内的状态序列、引起转移的事件以及因状态转移而伴随的动作。



(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各自的控制器委托给不同的领域对象。

### 5.1.5Financebl模块

(1)模块概述

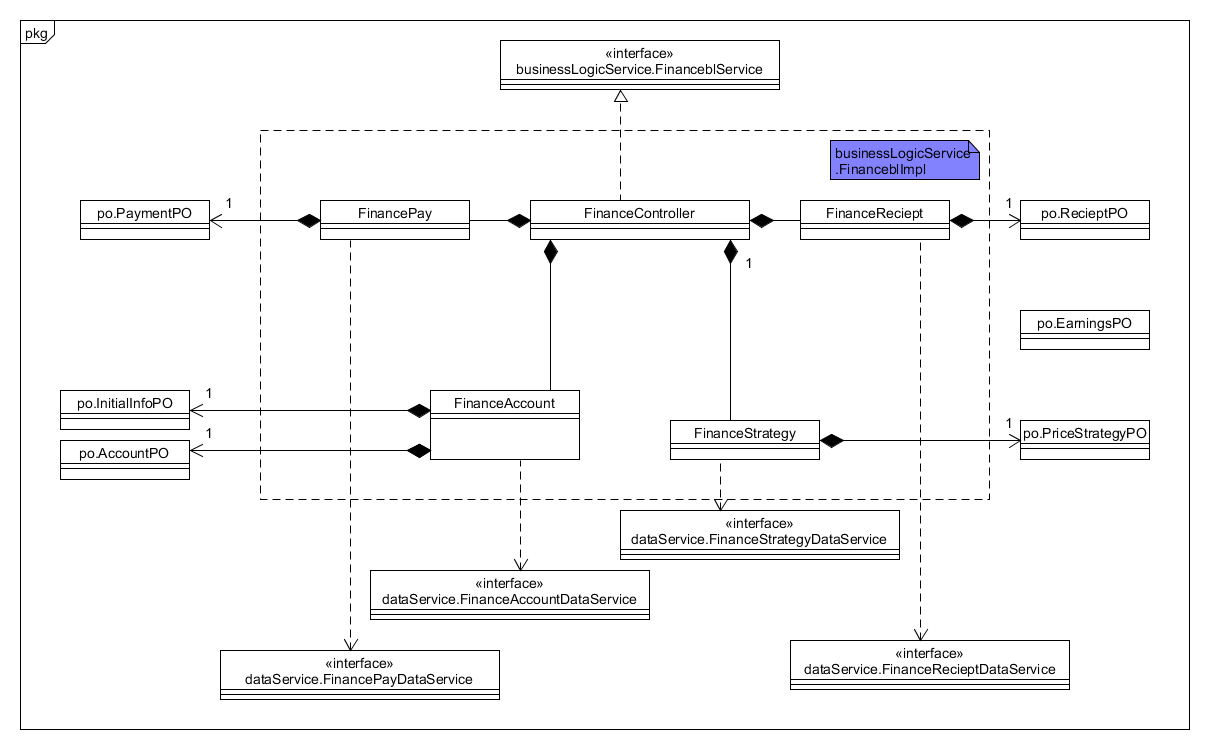
Financebl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

Financebl模块的职责和接口参见软件体系结构描述文档。

(2)整体结构

根据体系结构的设计，系统分为展示层、业务逻辑层和数据层，并在展示层和业务逻辑层之间添加bussinssFinanceicServi ce.FinanceAccountblService, FinancePayblService, FinanceReceiptblService, FinanceStrategyblService接口、在业务逻辑层和数据层之间添加dataService.FinanceAccountDataService，FinancePaymentDataService，FinanceReceiptDataService，FinanceStrategyDataService接口，从而增加了系统的灵活性。我们增加了FinanceController从而隔离了业务逻辑职责和逻辑控制职责，这样FinanceController会将业务逻辑处理委托给Financebl对象。PaymentPO,ReceiptPO,InitialInfoPO,EarningPO,PriceStrategyPO,SalaryStrategyPO作为用户信息的持久化对象被添加到设计模型中去的。LogImpl用来进行对管理用户操作时产生的系统日志的记录。

Financebl模块的设计如图所示。



Financebl模块的各个类的职责如下表所示。

Financebl模块的各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| FinanceController | 负责实现付款、收款、账户、策略四大模块的协调与调度 |
| FinancePayblImpl | 负责实现付款单建立及付款所需要的服务 |
| FinanceAccountblImpl | 负责实现账户的添加，删除，修改，查找和添加期初建账信息所需要的服务 |
| FinanceStrategyblImpl | 负责实现价格策略、薪水策略的制定和修改所需要的服务 |
| FinanceReceiptblImpl | 负责实现收款单建立及收款所需要的服务 |

(3)模块内部类的接口规范

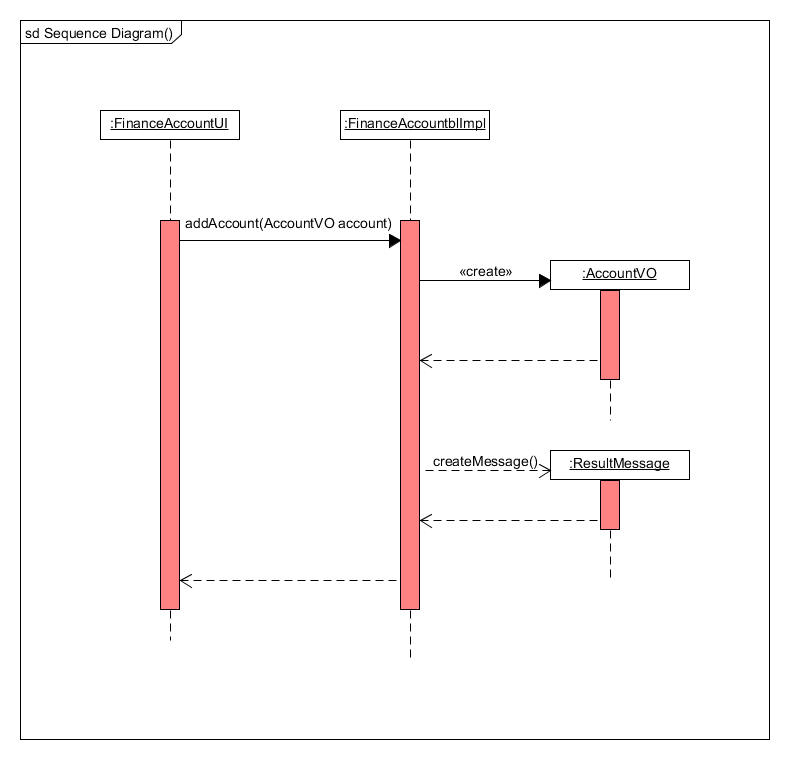
FinanceController和Financebl的接口规范如下表所示。

Finance的接口规范

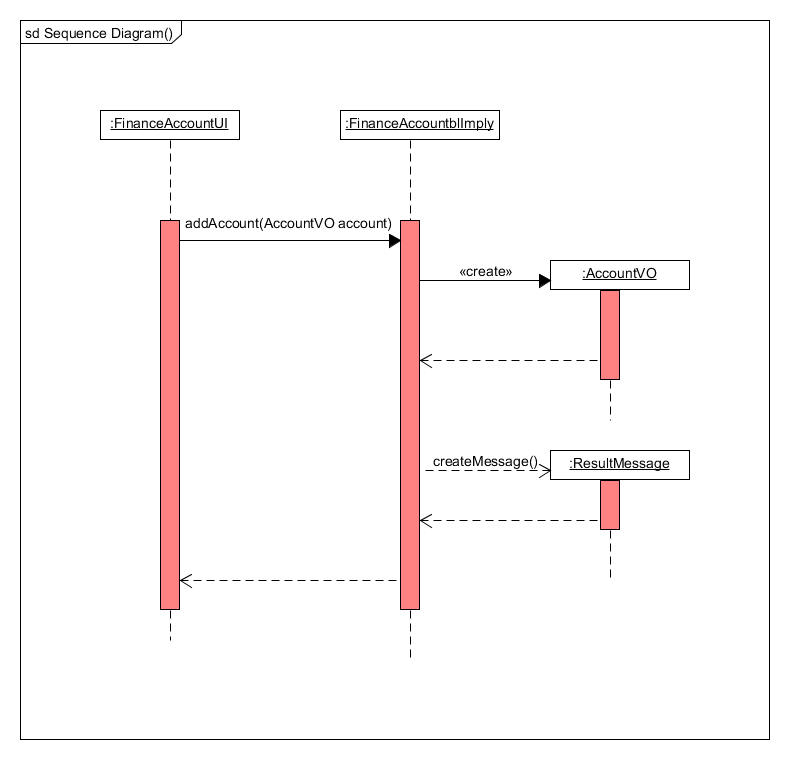
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| FinanceController. createReceipt | 语法 | public ReceiptVO createReceipt (ReceiptVO Receipt) |
| 前置条件 | Debit符合规范 |
| 后置条件 | 返回给界面完整的付款单信息 |
| FinanceController. addReceipt | 语法 | public ResultMessage addReceipt (ReceiptVO Receipt) |
| 前置条件 | 无 |
| 后置条件 | 如果增加成功返回true，否则返回对应的错误信息 |
| FinanceController. deleteReceipt | 语法 | public ResultMessage deleteReceipt (String id) |
| 前置条件 | 无 |
| 后置条件 | 如果删除成功返回true，否则返回对应的错误信息 |
| FinanceController. updateReceipt | 语法 | public ResultMessage deleteReceipt (ReceiptVO Receipt) |
| 前置条件 | 无 |
| 后置条件 | 如果更新成功返回true，否则返回对应的错误信息 |
| FinanceController. showReceiptList | 语法 | public ArrayList<debitnoteVO> showDebitnote (Calendar date, Calendar end) |
| 前置条件 | end在date之后 |
| 后置条件 | 返回符合该时间段内所有收款单信息 |
| FinanceController.getReceiptSum | 语法 | public double getReceiptSum(Calendar date) |
| 前置条件 | Receiptnote 不为空 |
| 后置条件 | 返回该列表中收款单的收款总和 |
| FinanceController. createRent | 语法 | public RentVO createRent(RentVO rent) |
| 前置条件 | 无 |
| 后置条件 | 返回完整的租金付款单信息 |
| FinanceController.saveRent | 语法 | public ResultMessage saveRent(RentVO rent) |
| 前置条件 | 已确认该付款单 |
| 后置条件 | 结束此次管理，持久化更新设计的领域对象的数据 |
| FinanceController . createFreight | 语法 | public FreightVO createFreight(String accountName) |
| 前置条件 | 无 |
| 后置条件 | 返回当前的运费付款单 |
| FinanceController. saveFreight | 语法 | public ResultMessage saveFreight(FreightVO freight) |
| 前置条件 | 已确认运费付款单 |
| 后置条件 | 结束此次管理，持久化更新涉及的领域对象的数据 |
| FinanceController. createSalary | **语法** | public SalaryVO createSalary(SalaryVO Salary) |
| 前置条件 | 员工id符合输入规则 |
| 后置条件 | 返回完整的工资付款单信息 |
| FinanceController.saveSalary | 语法 | public ResultMessage saveSalary (SalaryVO Salary) |
| 前置条件 | 已确认该付款单 |
| 后置条件 | 结束此次管理，持久化更新设计的领域对象的数据 |
| FinanceController.showAllPayment | 语法 | public ArrayList<PaymentVO> showAllPayment(Calendar start, Calendar end) |
| 前置条件 |  |
| 后置条件 | 返回选定时间内所有的付款单信息 |
| FinanceController. showEarning | 语法 | public EarningVO showEarning(Calendar start,Calendar end) |
| 前置条件 | 无 |
| 后置条件 | 返回选定日期内的经营情况表 |
| FinanceController. exportEarning | 语法 | public ResultMessage exportEarning(EarningVO earnings) |
| 前置条件 | 无 |
| 后置条件 | 如果导出成功，返回true，否则返回对应的错误信息 |
| FinanceController. addAccount | 语法 | public ResultMessage addAccount(AccountVO account) |
| 前置条件 | AccoutVO不为null |
| 后置条件 | 如果增加成功，返回true， 否则返回对应的错误信息 |
| FinanceController. showAccount | 语法 | public AccountVO showAccount(String id) |
| 前置条件 | 输入的id符合规范 |
| 后置条件 | 返回查询的账户信息 |
| FinanceController. deleteAccount | 语法 | public ResultMessage deleteAccount(String id) |
| 前置条件 | 输入的id符合规范 |
| 后置条件 | 如果删除成功返回true，否则返回对应的错误信息 |
| FinanceController. updateAccount | 语法 | public ResultMessage updateAccount(AccountVO account) |
| 前置条件 | Account不为null |
| 后置条件 | 如果修改成功返回true，否则返回对应的错误信息 |
| FinanceController.showAllAcount | 语法 | public ArrayList<AccountVO> showAllAccount() |
| 前置条件 | 无 |
| 后置条件 | 返回系统所有账户信息 |
| FinanceController.addMoney | 语法 | public ResultMessage addMoney (String name,double amount) |
| 前置条件 | 无 |
| 后置条件 | 系统更新账户信息 |
| FinanceController.addInitialInfo | 语法 | public ResultMessage addInitialInfo(InitialInfoVO initial) |
| 前置条件 | 无 |
| 后置条件 | 初始化系统 |
| FinanceController.findInitialInfo | 语法 | public ArrayList<InitialInfoVO> findInitialInfo() |
| 前置条件 | 无 |
| 后置条件 | 返回系统所有期初建账信息 |
| FinanceController. addPriceStrategy | 语法 | public ResultMessage addPriceStrategy (double std) |
| 前置条件 | 无 |
| 后置条件 | 如果增加成功，返回true， 否则返回对应的错误信息 |
| FinanceController. updatePriceStrategy | 语法 | public ResultMessage updatePriceStrategy(double std) |
| 前置条件 | 无 |
| 后置条件 | 如果修改成功返回true，否则返回对应的错误信息 |
| FinanceController. findPriceStrategy | 语法 | public PriceStrategyVO findPriceStrategy () |
| 前置条件 | 无 |
| 后置条件 | 返回查询的信息 |
| FinanceController. addSalaryStrategy | 语法 | public ResultMessage addSalaryStrategy (SalaryStrategyVO SalaryStrategy) |
| 前置条件 | 无 |
| 后置条件 | 如果增加成功，返回true， 否则返回对应的错误信息 |
| FinanceController. updateSalaryStrategy | 语法 | public ResultMessage updateSalaryStrategy (SalaryStrategyVO SalaryStrategy) |
| 前置条件 | 无 |
| 后置条件 | 如果修改成功返回true，否则返回对应的错误信息 |
| FinanceController. findSalaryStrategy | 语法 | public SalaryStrategyVO findSalaryStrategy (String id) |
| 前置条件 | 无 |
| 后置条件 | 返回查询的信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| FinanceDataService  .addReceipt(ReceiptPo Receipt): ResultMessage | 增加收款单的单一持久化对象 | |
| FinanceDataService  .findReceipt(Calendar date,Calendar end):List<ReceiptPo> | 根据输入的时间查找收款单的单一持久化对象，返回所有符合条件的收款单的列表 | |
| FinanceDataService  .deleteReceipt(String id): ResultMessage | 根据输入的id查找收款单并删除该单一持久化对象 | |
| FinanceDataService  .updateReceipt(ReceiptPo receipt): ResultMessage | 根据输入的收款单id查找收款单并用传入的收款单替换该单一持久化对象 | |
| FinanceDataService  .addPayment(PaymentPo payment): ResultMessage | 增加新的付款单单一持久化对象 | |
| FinanceDataService  .findPayment(Calendar date,Calendar end):List<PaymentPo> | 根据输入的date查找符合条件的付款单单一持久化对象 | |
| FinanceDataService  .addAccount(AccountPo account): ResultMessage | 增加新的银行账户单一持久化对象 | |
| FinanceDataService  .findAccount(String id):AccountPo | 按id进行查找返回相应的AccountPo结果 | |
| FinanceDataService  .deleteAccount(String id):boolean | 按id查找到相应的AccountPo单一持久化对象结果并删除 | |
| FinanceDataService  .updateAccount(AccountPo account) | 按id查找到AccountPo的单一持久化对象并替换 | |
| FinanceDataService.showAllAccount():  ArrayList<AccountPO> | 返回所有的account单一持久化对象 | |
| FinanceDataService.addPriceStrategy(double std):ResultMessage | 增加运费策略的单一持久化对象 | |
| FinanceDataService.updatePriceStrategy(double std):ResultMessage | 更新运费策略的单一持久化 | |
| FinanceDataService.findPriceStrategy(): PriceStrategyVO | 返回当前的运费策略单一持久化对象 | |
| FinanceDataService.addSalaryStrategy(SalaryVO Salary):ResultMessage | 增加薪水策略的单一持久化对象 | |
| FinanceDataService.updateSalaryStrategy(SalaryVO Salary):ResultMessage | 更新薪水策略的单一持久化 | |
| FinanceDataService.findSalaryStrategy():SalaryStrategyVO | 返回当前的薪水策略单一持久化对象 | |
| LogController.addLog(String explain):ResultMessage | 根据操作生成系统日志 | |
| ListController.applyNumOccupancy | 生成单据的编号 | |

(4)业务逻辑层的动态模型

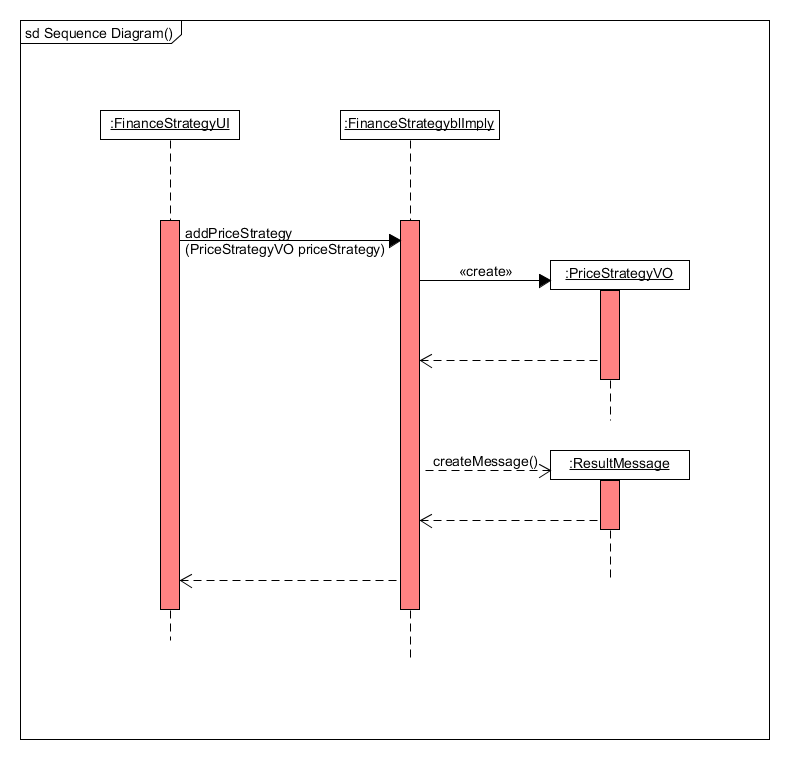
下图表示了物流管理系统中，当用户选择添加账户时，Financebl处理的相关对象的协作。



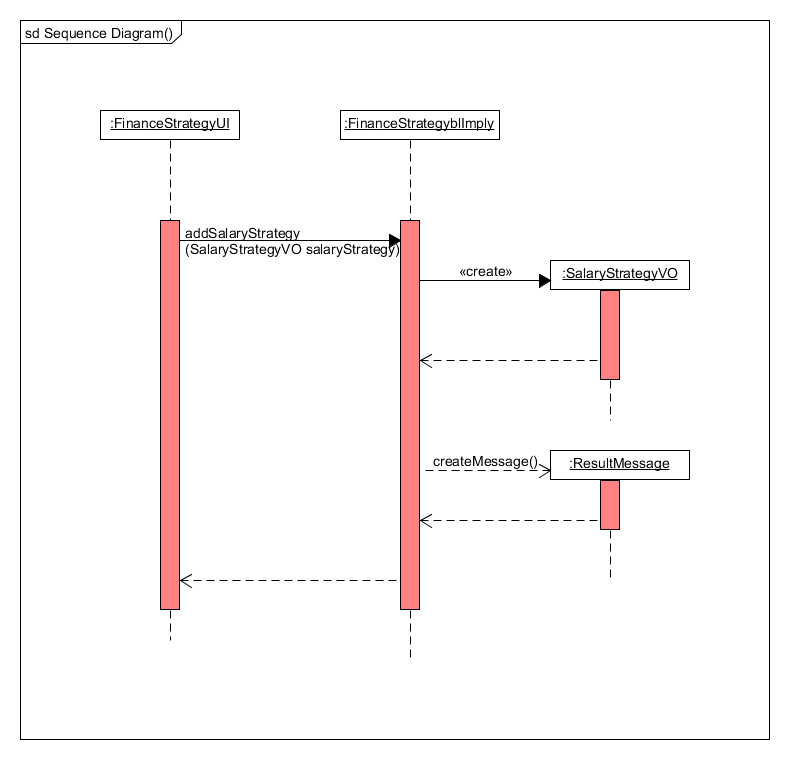
下图表示了物流管理系统中，当用户添加期初建账信息时，Financebl处理的相关对象的协作。



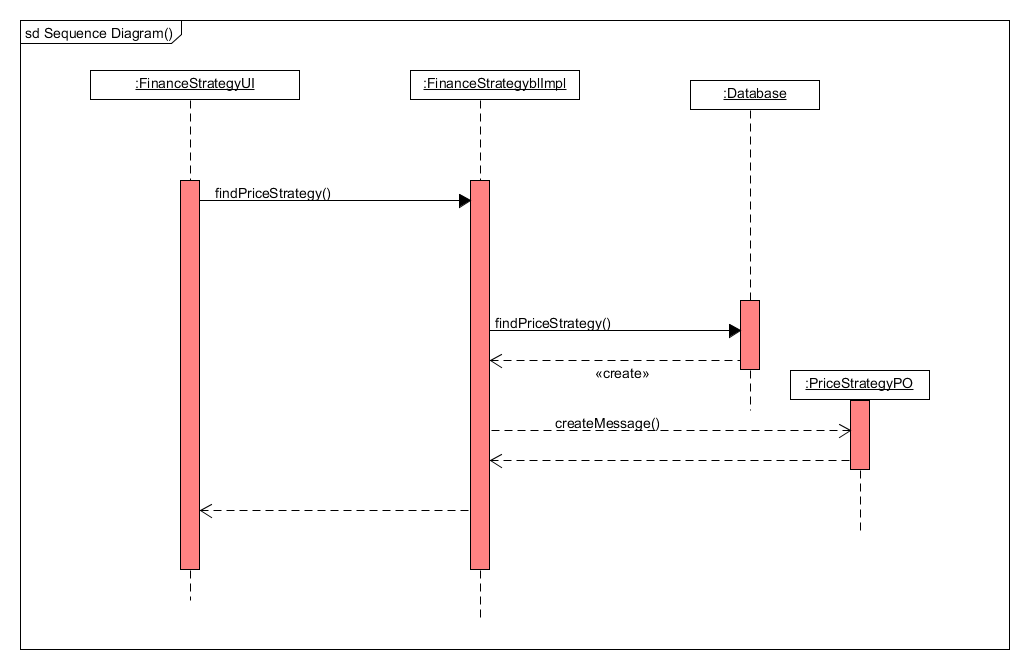
下图表示了物流管理系统中，当用户选择添加价格策略时，Financebl处理的相关对象的协作。



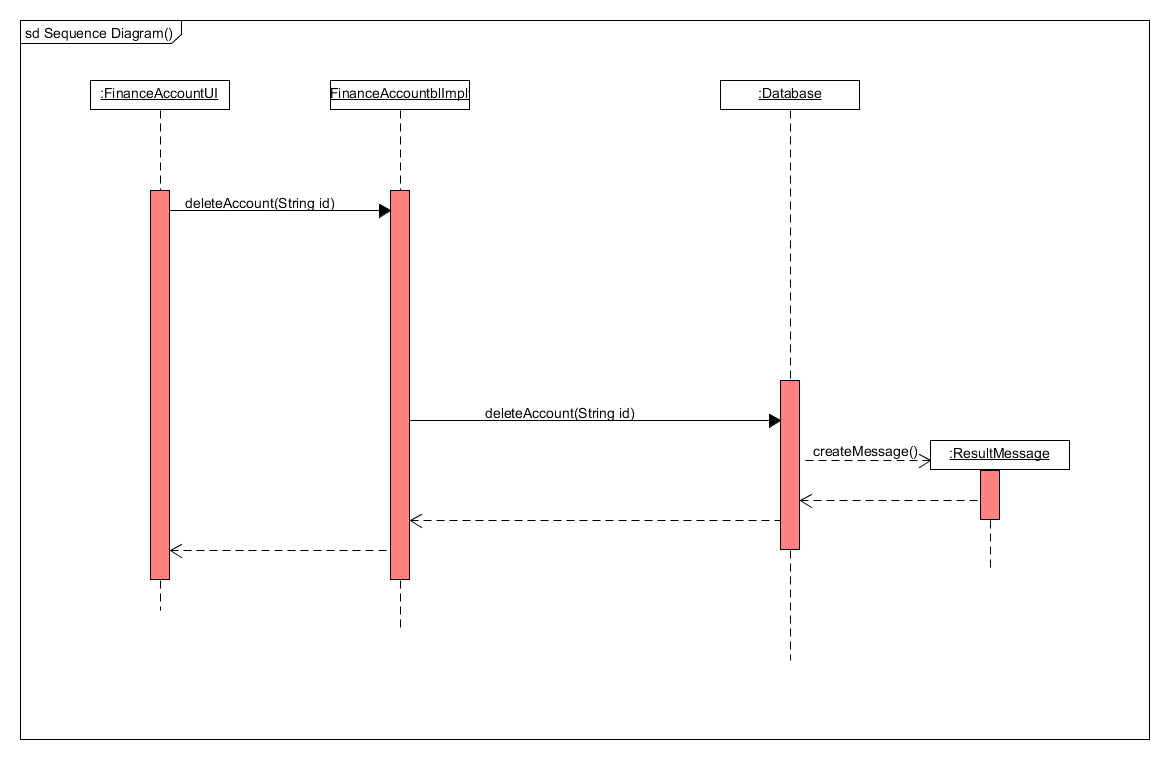
下图表示了物流管理系统中，当用户选择添加薪水策略时，Financebl处理的相关对象的协作。



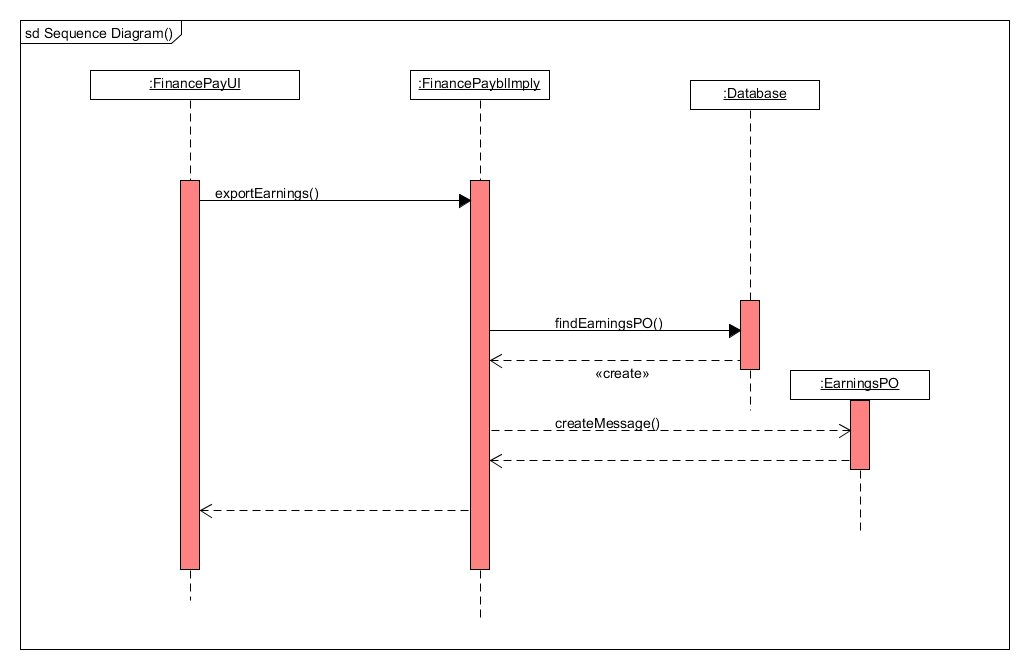
下图表示了物流管理系统中，当用户选择创建收款单时，Financebl处理的相关对象的协作。



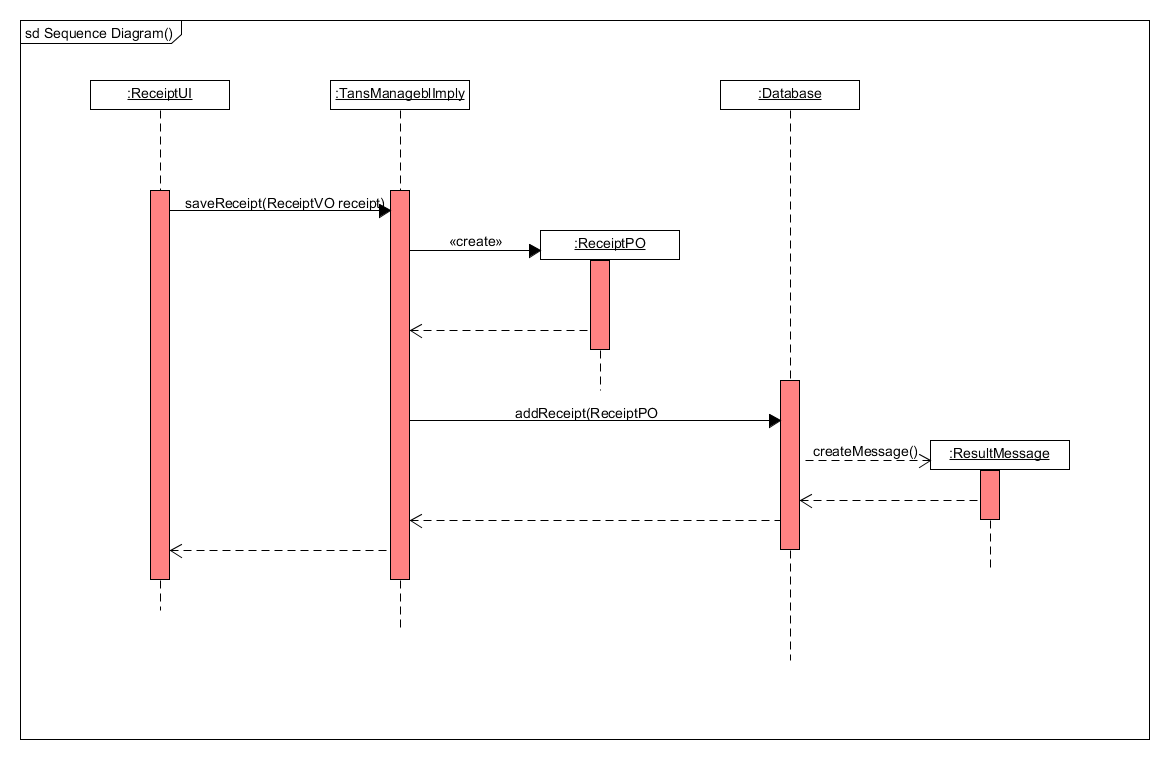
下图表示了物流管理系统中，当用户选择删除账户时，Financebl处理的相关对象的协作。



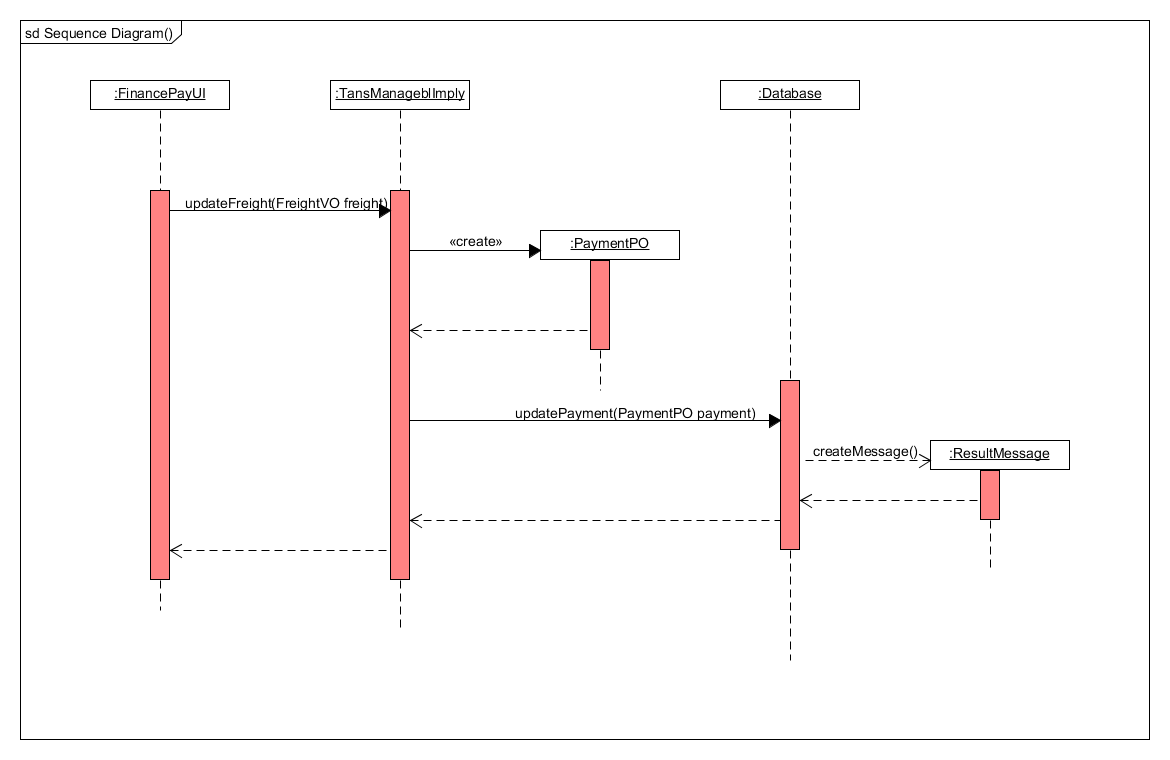
下图表示了物流管理系统中，当用户选择导出成本收益表时，financebl处理的相关对象的协作。



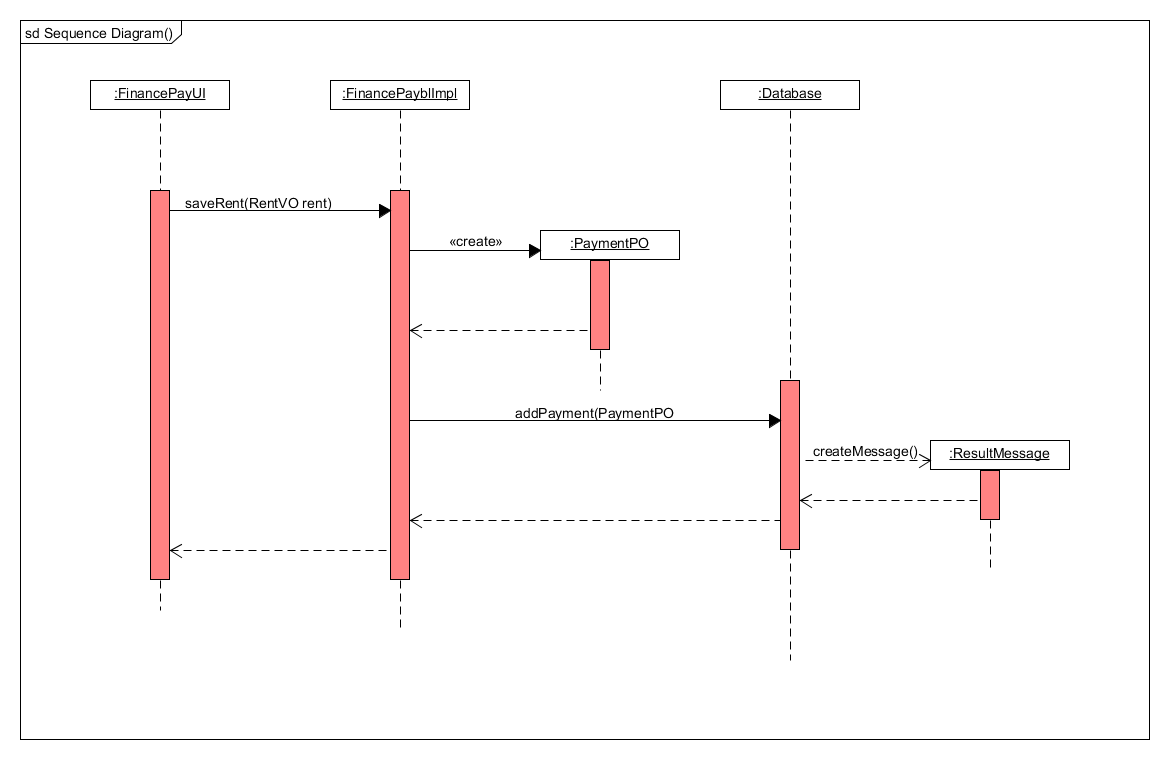
下图表示了物流管理系统中，当用户选择合计收款单时，Financebl处理的相关对象的协作。



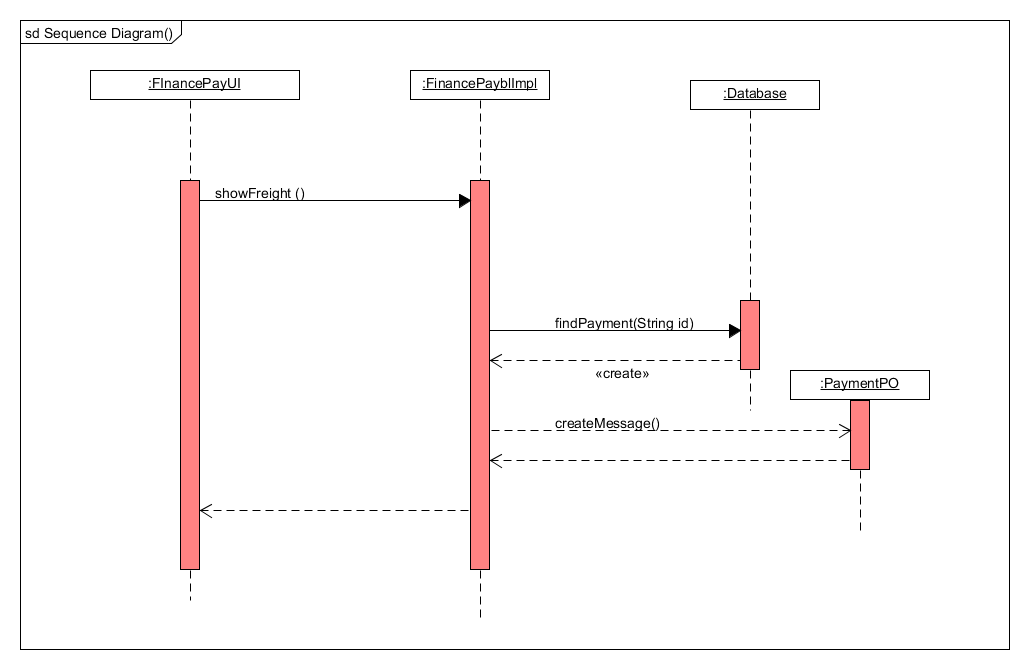
下图表示了物流管理系统中，当用户选择修改账户时，Financebl处理的相关对象的协作。



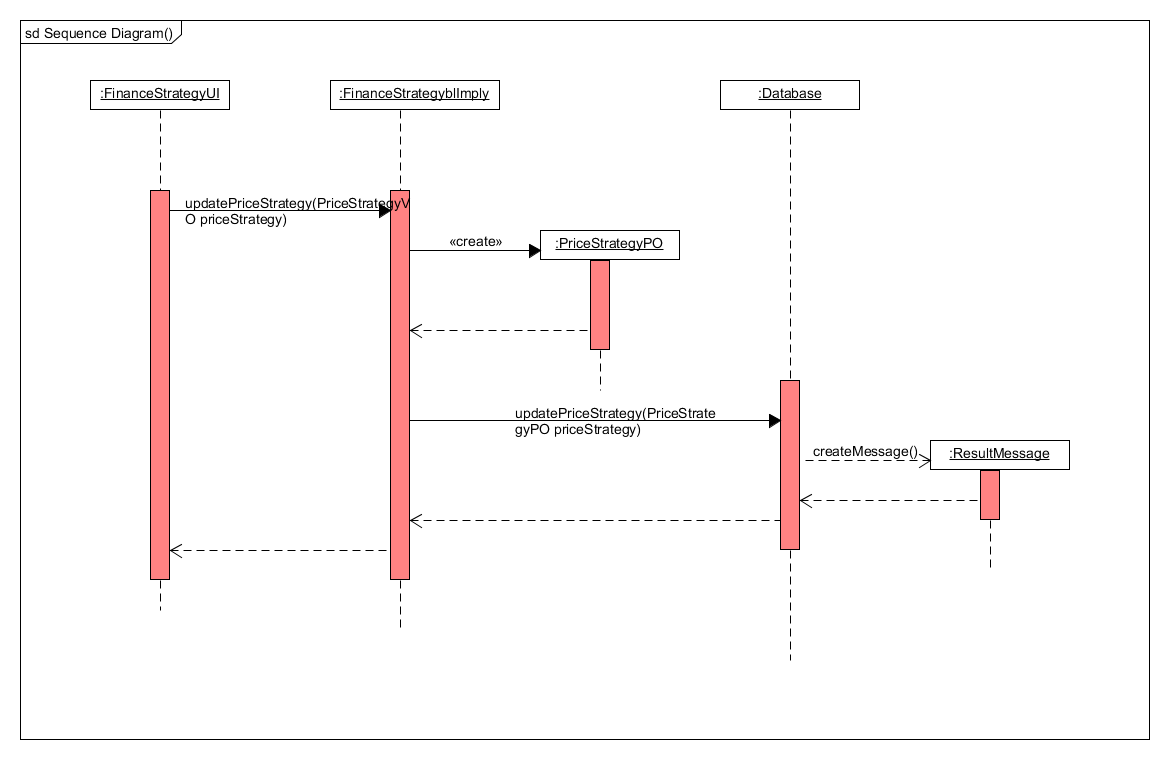
下图表示了物流管理系统中，当用户选择修改租金时，Financebl处理的相关对象的协作。



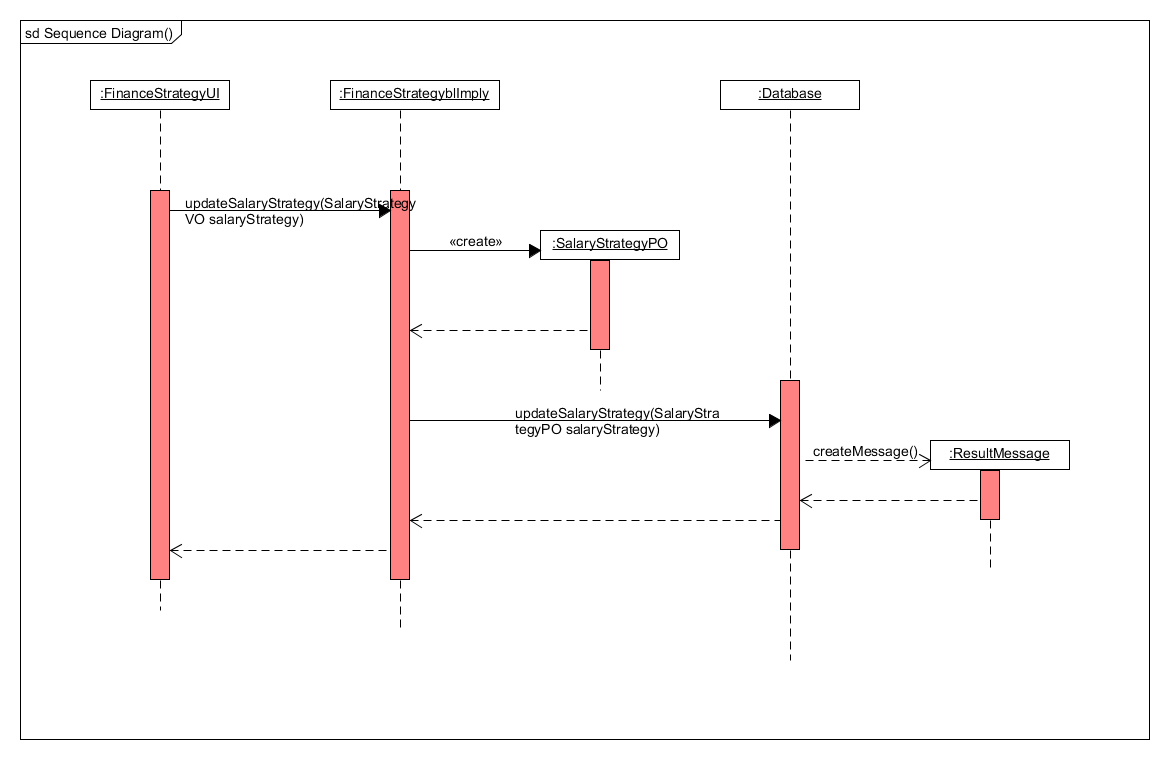
下图表示了物流管理系统中，当用户选择查看成本收益表时，Financebl处理的相关对象的协作。



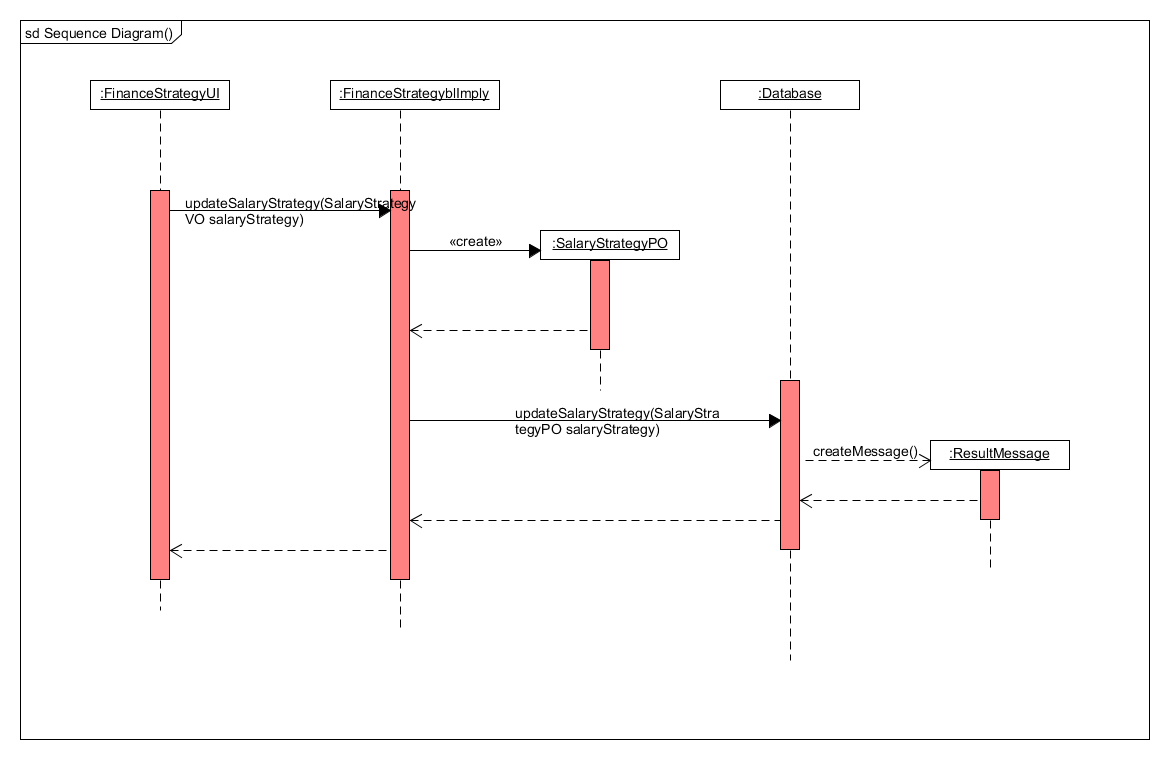
下图表示了物流管理系统中，当用户选择修改运费时，Financebl处理的相关对象的协作。



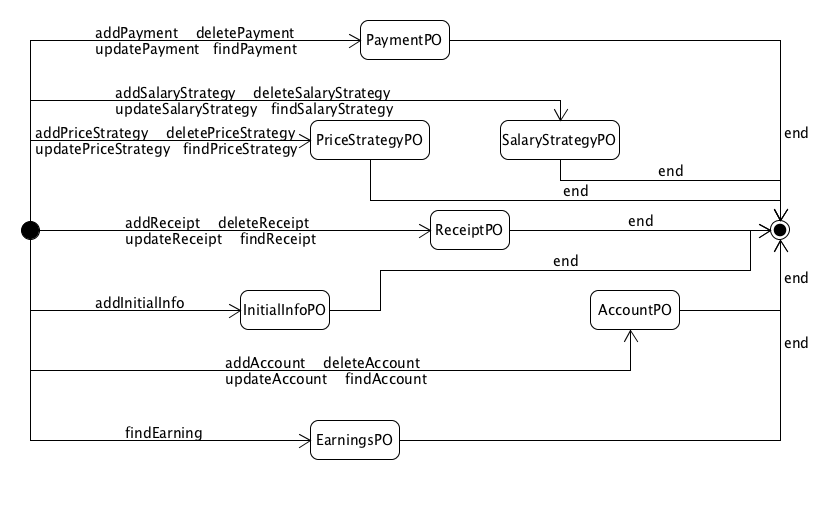
下图表示了物流管理系统中，当用户选择修改价格策略时，Financebl处理的相关对象的协作。



下图表示了物流管理系统中，当用户选择修改薪水策略时，Financebl处理的相关对象的协作。



如下图所示的状态图描述了Financebl对象的生存期内的状态序列、引起转移的事件以及因状态转移而伴随的动作。



(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各自的控制器委托给不同的领域对象。

### 5.1.6Listbl模块

(1)模块概述

Listbl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。

Listbl模块的职责和接口参见软件体系结构描述文档。

(2)整体结构

根据体系结构的设计，系统分为展示层、业务逻辑层和数据层，并在展示层和业务逻辑层之间添加bussinssLogicServi ce.ListblService接口、在业务逻辑层和数据层之间添加dataService.ListDataService接口，从而增加了系统的灵活性。我们增加了ListController从而隔离了业务逻辑职责和逻辑控制职责，这样ListController会将业务逻辑处理委托给ListblImpl对象。ListPO作为用户信息的持久化对象被添加到设计模型中去的。LogImpl用来进行对管理用户操作时产生的系统日志的记录。

Listbl模块的设计如图所示。



Listbl模块的各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| ListController | 负责实现用户管理界面所需要的服务 |
| ListblImpl | 用户管理领域的模型对象，可以帮助完成用户管理界面所需要的服务 |
| ListNumOccupancy | 负责实现各类单据编号的自动生成 |

(3)模块内部类的接口规范

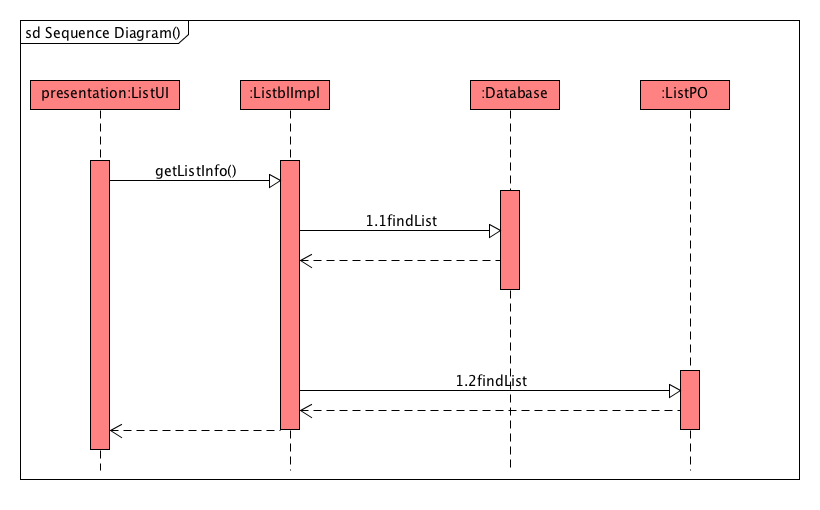
ListController和Listbl的接口规范如下表所示。

ListController的接口规范

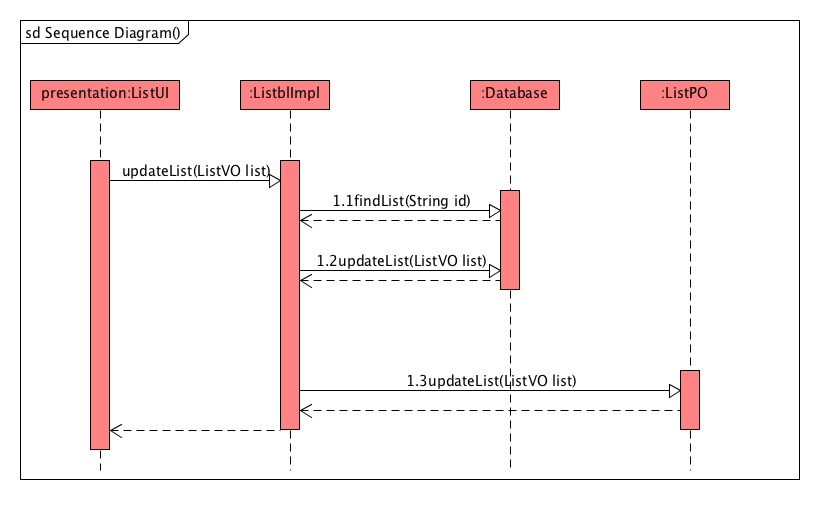
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ListController. getListInfo | 语法 | Public ArrayList<ListPO> getListInfo(ListType type) |
| 前置条件 | 无 |
| 后置条件 | 查找所需类型是否存在未被审批的单据，返回所有所需类型未被审批的单据的引用 |
| ListController. changeList | 语法 | Public ResultMessage changeList(ListVO List, ListType type) |
| 前置条件 | 无 |
| 后置条件 | 根据id查找List，并修改相应的数据，如果修改成功返回true，否则返回对应的错误信息 |
| ListController. getListInfo | 语法 | Public ListPO getListInfo(String id) |
| 前置条件 | 无 |
| 后置条件 | 根据编号查询一个单据的信息 |
| ListController. applyListNum | 语法 | Public String applyListNum(ListType type) |
| 前置条件 | 无 |
| 后置条件 | 为所选类型的单据生成一个不重复的编号 |
| ListController. saveListNum | 语法 | Public void saveListNum () |
| 前置条件 | 无 |
| 后置条件 | 更新现存的单据编号的占用情况 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| ListDataImpl. findList(ListType type):ArrayList<ListPO> | 得到一类单据持久化对象的数据 | |
| ListDataImpl. updateList(String id,ListState state,ListType type):ResultMessage | 根据输入的单据信息修改单据内容与状态 | |
| ListDataImpl.findList(ListType type,String id):ListVO | 根据输入的id得到一个单据持久化对象 | |
| ListDataImpl.getNumOccupancy(): NumOccupancyPO | 得到当前编号占用情况的持久化对象 | |
| ListDataImpl.updateNumOccupancy() | 更新当前编号占用情况的持久化对象 | |

(4)业务逻辑层的动态模型

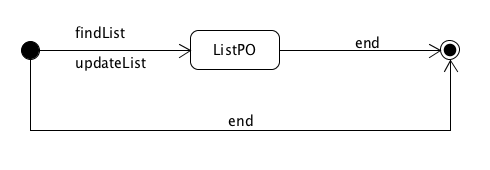
下图表示了物流管理系统中，当用户选择查看单据信息时，Listbl处理的相关对象的协作。



下图表示了物流管理系统中，当用户输入要修改的单据信息时，Listbl处理的相关对象的协作。



如下图所示的状态图描述了Listbl对象的生存期内的状态序列、引起转移的事件以及因状态转移而伴随的动作。



(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各自的控制器委托给不同的领域对象。

### 5.1.7Userbl模块

(1)模块概述

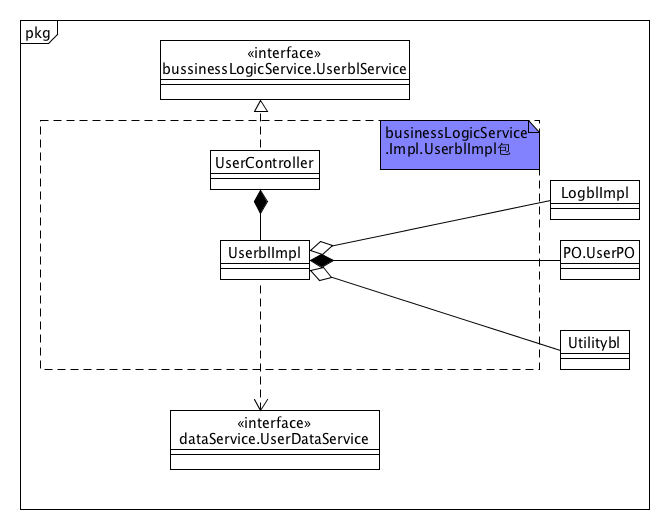
userbl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

userbl模块的职责和接口参见软件体系结构描述文档。

(2)整体结构

根据体系结构的设计，系统分为展示层、业务逻辑层和数据层，并在展示层和业务逻辑层之间添加bussinssLogicSerivce.UserblService接口、在业务逻辑层和数据层之间添加dataService.UserDataService接口，从而增加了系统的灵活性。我们增加了UserController从而隔离了业务逻辑职责和逻辑控制职责，这样UserController会将业务逻辑处理委托给UserblImpl对象。UserPO作为用户信息的持久化对象被添加到设计模型中去的。LogImpl用来进行对管理用户操作时产生的系统日志的记录。

Userbl模块的设计如图所示。



Userbl模块的各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| UserController | 负责实现用户管理界面所需要的服务 |
| UserblImpl | 用户管理领域的模型对象，可以帮助完成用户管理界面所需要的服务 |

(3)模块内部类的接口规范

UserController和Userbl的接口规范如下表所示。

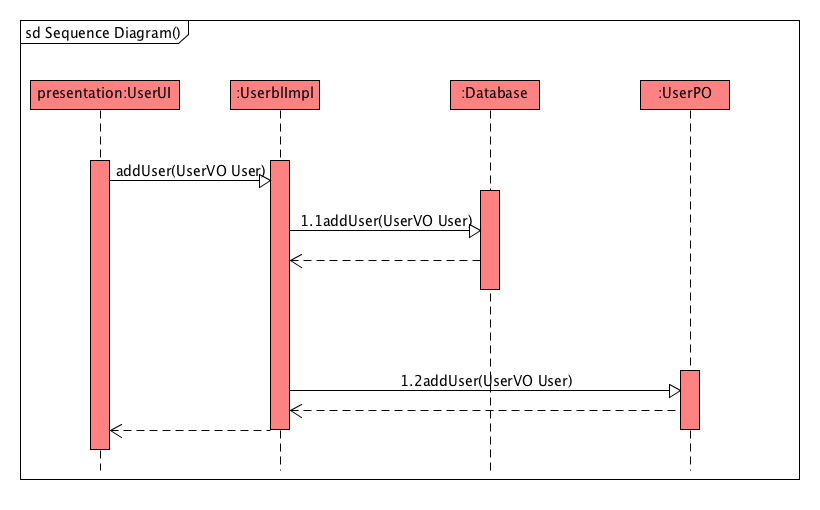
UserController的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserController.login | 语法 | Public String login(long id, String password) |
| 前置条件 | Password符合输入规则 |
| 后置条件 | 查找是否存在相应的User，根据输入的password返回输入的结果 |
| UserController.findUserInfo | 语法 | Public UserPO findUserInfo(String id) |
| 前置条件 | 无 |
| 后置条件 | 查找是否存在相应的user，如果存在返回相应的user信息 |
| UserController.deleteUser | 语法 | Public ResultMessage deleteUser(String id) |
| 前置条件 | 无 |
| 后置条件 | 查找是否存在相应的user， 如果删除成功返回true，否则返回对应的错误信息 |
| UserController.updateUser | 语法 | Public ResultMessage updateUser(UserVO user) |
| 前置条件 | 无 |
| 后置条件 | 根据id查找user，并修改相应的数据，如果修改成功返回true，否则返回对应的错误信息 |
| UserController.addUser | 语法 | Public ResultMessage addUser(UserVO user) |
| 前置条件 | 无 |
| 后置条件 | 如果增加成功返回true，否则返回对应的错误信息 |
| UserController.findAllUser | 语法 | Public ArrayList<UserVO> findAllUser() |
| 前置条件 | 无 |
| 后置条件 | 返回当前系统所有用户的信息 |

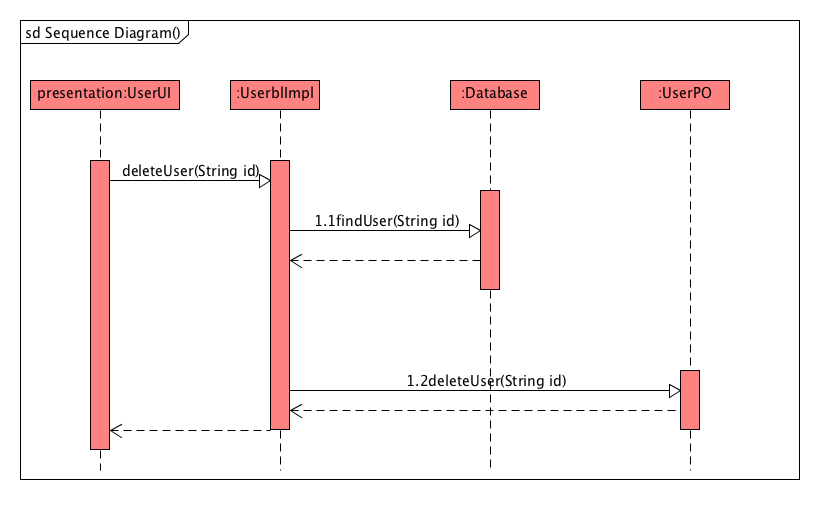
|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| UserDataService  .findUser(String id):UserPO | 根据输入的id返回一个用户持久化对象 |
| UserDataService  .deleteUser(String id):ResultMessage | 根据输入的id删除一个用户持久化对象 |
| UserDataService  .updateUser(UserPO user):ResultMessage | 根据输入的信息修改原有的用户持久化对象 |
| UserDataService  .addUser(UserPO user): ResultMessage | 根据输入的信息增加用户持久化对象 |
| UserDataService.findAllUser()  :ArraylList<UserVO> | 返回当前系统所有用户持久化对象 |
| LogController.addLog(String explain):ResultMessage | 根据操作生成系统日志 |

(4)业务逻辑层的动态模型

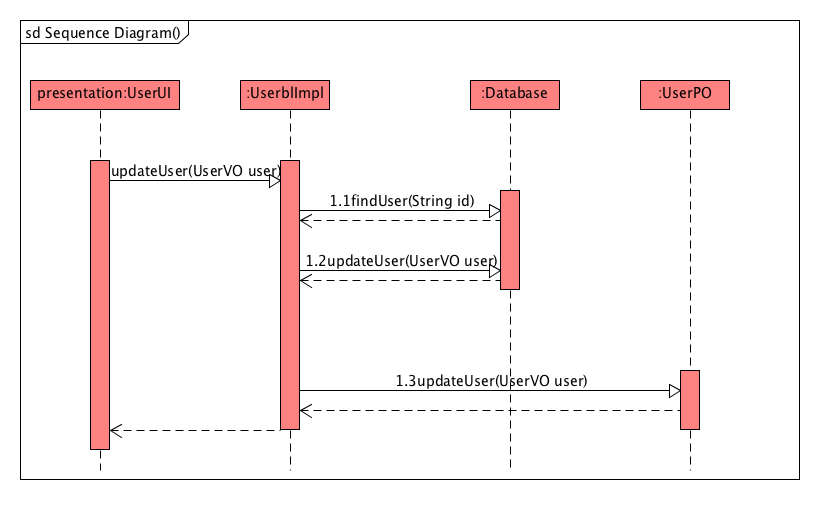
下图表示了物流管理系统中，当用户输入要增加的用户信息时，Userbl处理的相关对象的协作。



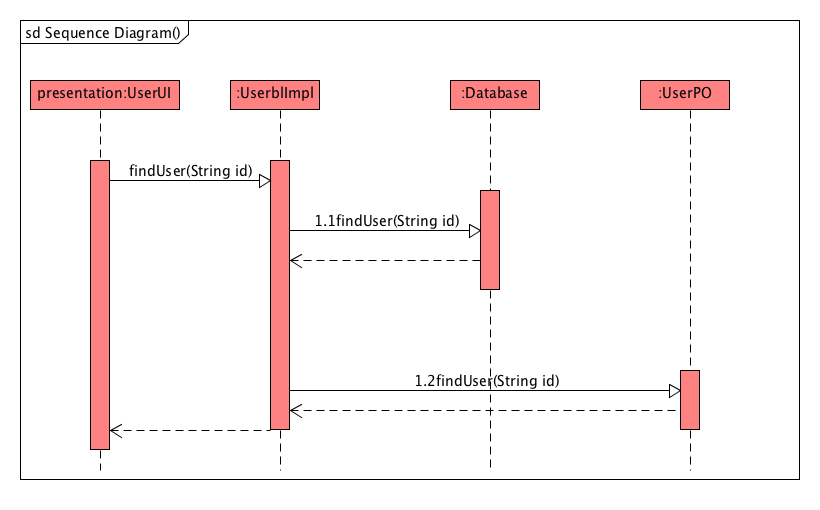
下图表示了物流管理系统中，当用户输入要删除的用户id时，Userbl处理的相关对象的协作。



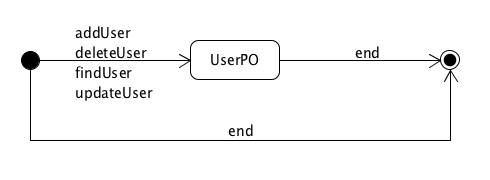
下图表示了物流管理系统中，当用户输入要修改的用户信息时，Userbl处理的相关对象的协作。



下图表示了物流管理系统中，当用户输入要查询的用户id时，Userbl处理的相关对象的协作。



如下图所示的状态图描述了Userbl对象的生存期内的状态序列、引起转移的事件以及因状态转移而伴随的动作。



(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各自的控制器委托给不同的领域对象。

### 5.1.8Logbl模块

(1)模块概述

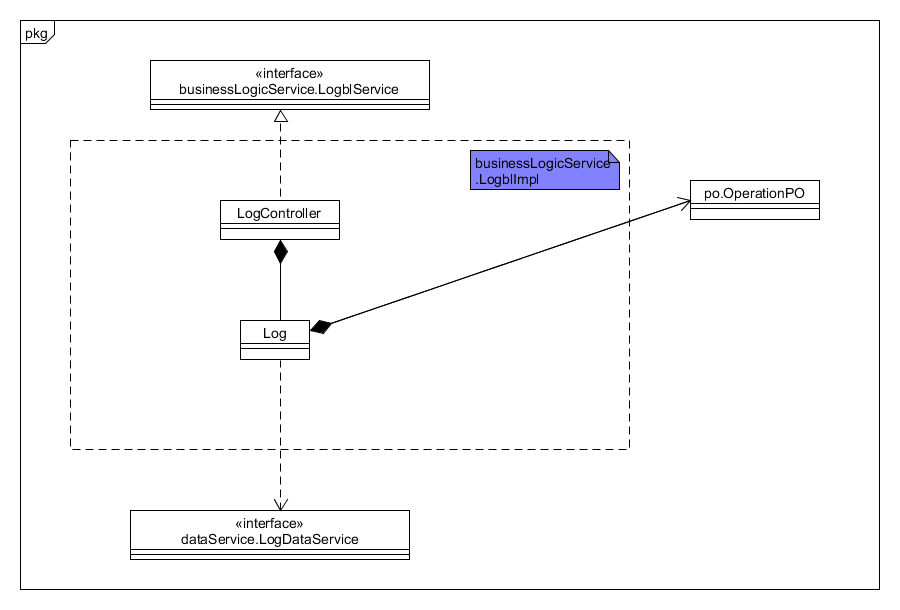
Logbl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

Logbl模块的职责和接口参见软件体系结构描述文档。

(2)整体结构

根据体系结构的设计，系统分为展示层、业务逻辑层和数据层，并在展示层和业务逻辑层之间添加bussinssLogicServi ce.LogblService接口、在业务逻辑层和数据层之间添加dataService.LogDataService接口，从而增加了系统的灵活性。我们增加了LogController从而隔离了业务逻辑职责和逻辑控制职责，这样LogController会将业务逻辑处理委托给LogblImpl对象。OperationPO作为用户信息的持久化对象被添加到设计模型中去的。

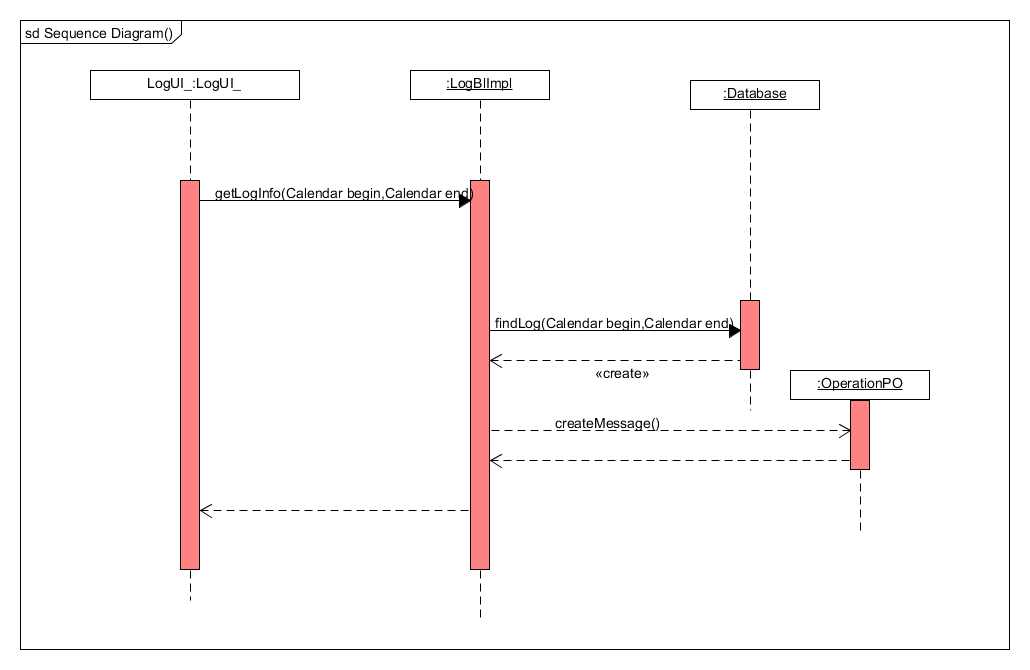
Logbl模块的设计如图所示。



Logbl模块的各个类的职责如下表所示。

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Log.getLogInfo | 语法 | public Arraylist<OperationVO> getLogInfo(Calendar begin, Calendar end) |
| 前置条件 | Begin的时间在end之前 |
| 后置条件 | 查找该时间段内的操作记录并返回 |
| Log.addLog | 语法 | public ResultMessage addLog(OperationVO op) |
| 前置条件 | 无 |
| 后置条件 | 如果增加成功返回true，否则返回对应的错误信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| LogDataService  .addLog(OperationPo operation):ResultMessage | 增加传入的操作单一持久化对象 | |
| LogDataService  .findOperation(Calendar begin, Calendar end):List<OperationPo> | 查找传入的时间段内的操作持久化对象并返回 | |

(4)业务逻辑层的动态模型

下图表示了物流管理系统中，当用户选择查询系统日志时，Logbl处理的相关对象的协作。

(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各自的控制器委托给不同的领域对象。

# 6.依赖视角

下图分别是客户端和服务器端各自的包之间的依赖关系。

