袁楚宏 袁阳阳 张鑫龙 周小帆

China Nanjing University

快递物流系统 软件详细设计描述文档

目录

[1. 引言 3](#_Toc434657697)

[1.1 编制目的 3](#_Toc434657698)

[1.2 词汇表 3](#_Toc434657699)

[1.3 参考资料 3](#_Toc434657700)

[2. 产品概述 4](#_Toc434657701)

[3. 体系结构设计概述 4](#_Toc434657702)

[4. 结构视角 4](#_Toc434657703)

[4.1 业务逻辑层分解 4](#_Toc434657704)

[4.1.1 userLogic模块 4](#_Toc434657705)

[（1） 模块概述 4](#_Toc434657706)

[（2） 整体结构 4](#_Toc434657707)

[（3） 模块内部类的接口规范 5](#_Toc434657708)

[（4） 业务逻辑层的动态模型 6](#_Toc434657709)

[（5） 业务逻辑层的设计原理 6](#_Toc434657710)

[4.1.2 transitInfoLogic模块 6](#_Toc434657711)

[（1） 模块概述 6](#_Toc434657712)

[（2） 整体结构 6](#_Toc434657713)

[（3） 模块内部类的接口规范 7](#_Toc434657714)

[（4） 业务逻辑层的动态模型 8](#_Toc434657715)

[（5） 业务逻辑层的设计原理 8](#_Toc434657716)

[4.1.3 orderNewLogic模块 8](#_Toc434657717)

[（1） 模块概述 8](#_Toc434657718)

[（2） 整体结构 8](#_Toc434657719)

[（3） 模块内部类的接口规范 10](#_Toc434657720)

[（4） 业务逻辑层的动态模型 14](#_Toc434657721)

[（5） 业务逻辑层的设计原理 14](#_Toc434657722)

[4.1.4 workOrgManLogic模块 14](#_Toc434657723)

[（1） 模块概述 14](#_Toc434657724)

[（2） 整体结构 14](#_Toc434657725)

[（3） 模块内部类的接口规范 15](#_Toc434657726)

[（4） 业务逻辑层的动态模型 19](#_Toc434657727)

[（5） 业务逻辑层的设计原理 19](#_Toc434657728)

[4.1.5 cityLogic模块 19](#_Toc434657729)

[（1）模块概述 19](#_Toc434657730)

[（2）整体结构 19](#_Toc434657731)

[（3）模块内部类的接口规范 20](#_Toc434657732)

[（4）业务逻辑层的动态模型 21](#_Toc434657733)

[（5）业务逻辑层的设计原理 21](#_Toc434657734)

[4.1.6 accountLogic模块 21](#_Toc434657735)

[（1） 模块概述 21](#_Toc434657736)

[（2） 整体结构 21](#_Toc434657737)

[（3） 模块内部类的接口规范 22](#_Toc434657738)

[（4） 业务逻辑层的动态模型 23](#_Toc434657739)

[（5） 业务逻辑层的设计原理 24](#_Toc434657740)

[4.1.7 stockLogic模块 24](#_Toc434657741)

[（1） 模块概述 24](#_Toc434657742)

[（2） 整体结构 24](#_Toc434657743)

[（3） 模块内部类的接口规范 25](#_Toc434657744)

[（4） 业务逻辑层的动态模型 27](#_Toc434657745)

[（5） 业务逻辑层的设计原理 28](#_Toc434657746)

[4.1.8 checkLogic模块 28](#_Toc434657747)

[（1）模块概述 29](#_Toc434657748)

[（2）整体结构 29](#_Toc434657749)

[（3）模块内部类的接口规范 29](#_Toc434657750)

[（4）业务逻辑层的动态模型 31](#_Toc434657751)

[（5）业务逻辑层的设计原理 31](#_Toc434657752)

[4.1.9 orderApproveLogic模块 31](#_Toc434657753)

[（1） 模块概述 31](#_Toc434657754)

[（2） 整体结构 31](#_Toc434657755)

[（3） 模块内部类的接口规范 32](#_Toc434657756)

[（4） 业务逻辑层的动态模型 33](#_Toc434657757)

[（5） 业务逻辑层的设计原理 34](#_Toc434657758)

[4.2 数据层分解 34](#_Toc434657759)

[4.2.1 userData模块 34](#_Toc434657760)

[（1） 模块概述 34](#_Toc434657761)

[（2） 整体结构 34](#_Toc434657762)

[（3） 模块内部类的接口规范 35](#_Toc434657763)

[（4） 数据层的动态模型 35](#_Toc434657764)

[（5） 数据层的设计原理 36](#_Toc434657765)

[4.2.2 transitInfoData模块 36](#_Toc434657766)

[（1） 模块概述 36](#_Toc434657767)

[（2） 整体结构 36](#_Toc434657768)

[（3） 模块内部类的接口规范 36](#_Toc434657769)

[（4） 数据层的动态模型 37](#_Toc434657770)

[（5） 数据层的设计原理 37](#_Toc434657771)

[4.2.3 orderNewData模块 37](#_Toc434657772)

[（1）模块概述 37](#_Toc434657773)

[（2）整体结构 37](#_Toc434657774)

[（3）模块内部类的接口规范 38](#_Toc434657775)

[（4）数据层的动态模型 41](#_Toc434657776)

[（5）数据层的设计原理 42](#_Toc434657777)

[4.2.4 workOrgManData模块 42](#_Toc434657778)

[（1） 模块概述 42](#_Toc434657779)

[（2） 整体结构 42](#_Toc434657780)

[（3） 模块内部类的接口规范 43](#_Toc434657781)

[（4） 数据层的动态模型 45](#_Toc434657782)

[（5） 数据层的设计原理 46](#_Toc434657783)

[4.2.5 CityData模块 46](#_Toc434657784)

[（1）模块概述 46](#_Toc434657785)

[（2）整体结构 46](#_Toc434657786)

[（3）模块内部类的接口规范 46](#_Toc434657787)

[（4）数据层的动态模型 47](#_Toc434657788)

[（5）数据层的设计原理 47](#_Toc434657789)

[4.2.6 accountData模块 47](#_Toc434657790)

[（1） 模块概述 47](#_Toc434657791)

[（2） 整体结构 47](#_Toc434657792)

[（3） 模块内部类的接口规范 48](#_Toc434657793)

[（4） 数据层的动态模型 50](#_Toc434657794)

[（5） 数据层的设计原理 50](#_Toc434657795)

[4.2.7 checkData模块 50](#_Toc434657796)

[（1）模块概述 50](#_Toc434657797)

[（2）整体结构 50](#_Toc434657798)

[（3）模块内部类的接口规范 51](#_Toc434657799)

[（4）数据层的动态模型 52](#_Toc434657800)

[（5）数据层的设计原理 52](#_Toc434657801)

[4.2.8 orderApproveData模块 52](#_Toc434657802)

[（1） 模块概述 52](#_Toc434657803)

[（2） 整体结构 53](#_Toc434657804)

[（3） 模块内部类的接口规范 53](#_Toc434657805)

[（4） 数据层的动态模型 54](#_Toc434657806)

[（5） 数据层的设计原理 54](#_Toc434657807)

[5. 依赖视角 54](#_Toc434657808)

1. 引言
   1. 编制目的

本报告详细完成对连锁商店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. 词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| ELS | 快递物流系统 |  |
| RMI | 远程接口调用 | 一般使用JavaRMI包 |

* 1. 参考资料

1. 快递物流系统软件体系结构描述文档
2. 快递物流系统用例文档
3. 快递物流系统需求规格文档
4. 产品概述

参考快递物流系统用例文档和快递物流系统需求规格说明文档中对产品的概括描述。

1. 体系结构设计概述

参考快递物流系统软件体系结构描述文档中体系结构设计的描述。

1. 结构视角
   1. 业务逻辑层分解

业务逻辑层的开发包图参见软件体系结构文档图3。

4.1.1 userLogic模块

1. 模块概述

userLogic模块承担的需求参见需求规格说明文档需求及相关非功能需求。

userLogic模块的职责及接口参见软件体系结构描述文档5.3.2中userLogic模块的接口规范。

1. 整体结构

根据体系结构的设计，分层结构分为界面层，逻辑层和数据层。

界面层和逻辑层之间添加LoginLSer作为登录界面的逻辑层接口，UserManLSer作为用户管理界面的逻辑层接口。逻辑层和数据层之间添加LoginDSer作为登录逻辑的数据层接口，UserManDSer作为用户管理逻辑的数据层接口。UserVO作为用户的界面值对象，UserPO作为用户的持久化对象。

UserLogic模块的设计如图所示



userLogic各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| LoginLogic | 负责实现登录界面所需要的数据处理 |
| UserManLogic | 负责实现用户管理界面所需要的业务逻辑 |

1. 模块内部类的接口规范

LoginLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| LoginLogic.login | 语法 | public ResultMessage login(String id,String password) |
| 前置条件 | 已输入并确认账号和密码 |
| 后置条件 | 返回相应用户信息 |
| 需要的服务（需接口） | | |
| LoginDSer.findUser(String id,String password) | 根据id和password查找UserPO，返回查找结果 | |

UserManLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserManLogic.newUser | 语法 | public boolean newUser(UserVO user) |
| 前置条件 | 已输入并确认用户信息 |
| 后置条件 | 返回是否成功新建 |
| UserManLogic.deleteUser | 语法 | public boolean deleteUser(String id) |
| 前置条件 | 已选择用户并确认删除 |
| 后置条件 | 返回是否成功删除 |
| UserManLogic.reviseUser | 语法 | public boolean reviseUser(UserVO user) |
| 前置条件 | 已输入并确认用户信息 |
| 后置条件 | 返回是否成功修改 |
| UserManLogic.findUser | 语法 | public ResultMessage findUser(String keyword) |
| 前置条件 | 已输入并确认关键字 |
| 后置条件 | 若成功，返回相应的用户列表，若失败，返回失败提示 |
| 需要的服务（需接口） | | |
| UserManDSer.addUser(UserPO user) | 新建一个userPO，返回是否新建成功 | |
| UserManDSer.deleteUser(String id) | 根据id删除UserPO，返回是否删除成功 | |
| UserManDSer.reviseUser(UserPO user) | 更新对应UserPO的信息，返回是否更新成功 | |
| UserManDSer.findUser(String keyword) | 根据关键词查找对应的UserPO，将所有对应的UserPO复制到一个ArrayList中后返回它 | |

1. 业务逻辑层的动态模型

登陆顺序图，表明了快递物流系统中，当用户输入用户名和密码进行登陆时，登陆服务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，每个界面需要访问的业务逻辑由对应对象负责。

* + 1. transitInfoLogic模块

1. 模块概述

transitInfoLogic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。

transitInfoLogic模块的职责及接口参见软件体系结构描述文档5.3.2中transitInfo模块的接口规范

1. 整体结构

根据体系结构的设计，分层结构分为界面、逻辑和数据层。层与层之间添加Service接口。界面层和数据层之间添加CourierLSer作为快递员查询订单信息的接口，CustomerLogicSer作为查询物流信息的接口。逻辑层和数据层之间添加CourierDSer作为快递员查询订单信息的接口，CustomerDSer作为查询物流信息的接口。ExpressPO存放持久化的订单数据，ResultMessage存放操作是否成功和订单的值对象ExpressVO。

transitInfoLogic模块的设计如图所示。

transitInfoLogic模块各个类的职责如表所示。

transitInfoLogic模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| CourierLogic | 负责实现订单信息查询界面所需要的数据处理 |
| CustomerLogic | 负责实现物流信息查询界面所需要的数据处理 |

1. 模块内部类的接口规范

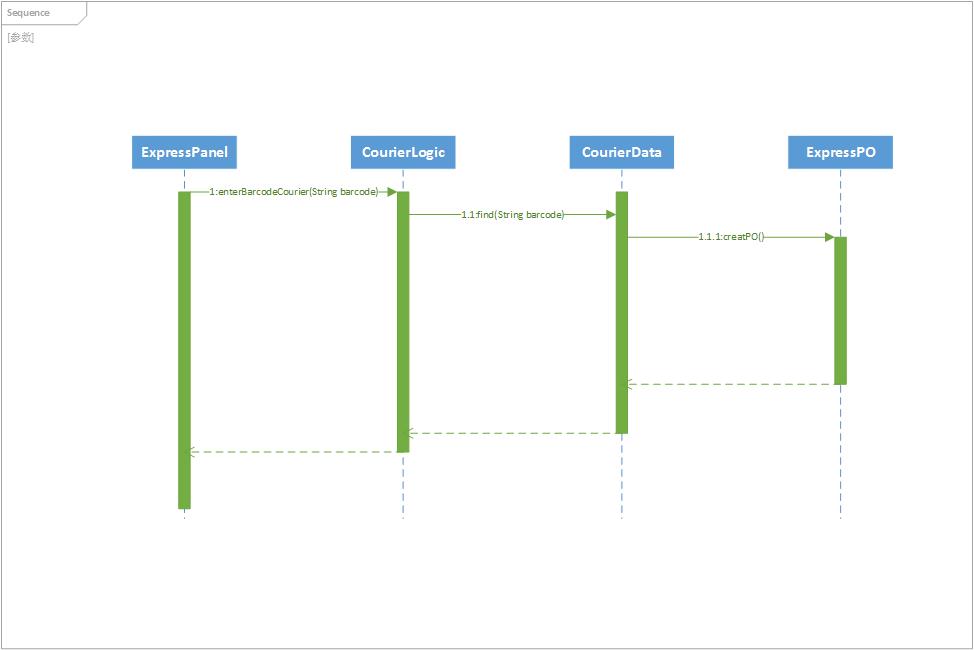
CourierLogic的接口规范如表所示。

CourierLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CourierLogic.enterBarcodeCourier | 语法 | public ResultMessage enterBarcodeCourier(String barcode) |
| 前置条件 | 已输入并确认订单条形码号 |
| 后置条件 | 根据条形码号从数据层获取订单信息 |
| 需要的服务（需接口） | | |
| CourierDSer.find(String barcode) | 根据barcode查找订单信息，返回查找结果 | |
|  |  | |

CustomerLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CustomerLogic.enterBarcodeCustomer | 语法 | public ResultMessage enterBarcodeCourier(String barcode) |
| 前置条件 | 已输入并确认订单条形码号 |
| 后置条件 | 根据条形码号从数据层获取订单的物流信息 |
| 需要的服务（需接口） | | |
| Customer.findTransit(String barcode) | 根据barcode查找订单物流信息，返回查找结果 | |

1. 业务逻辑层的动态模型

上图表明了快递物流系统中，当快递员输入快递单号之后，快递员订单信息查询逻辑的相关对象之间的协作。

1. 业务逻辑层的设计原理

见4.1.1.（5）

* + 1. orderNewLogic模块

1. 模块概述

OrderNewLogic模块承担的需求参见需求规格说明文档需求及相关非功能需求。OrderNewLogic模块的职责及接口参见软件体系结构描述文档5.3.2中OrderNewLogic模块的接口规范。

1. 整体结构

根据体系结构的设计，分层结构分为界面层，逻辑层和数据层。界面层和逻辑层之间添加ArriveOrderLSer作为中转中心到达单生成界面的逻辑层接口，CenterLoadingOrderLSer作为中转中心装车单生成界面的逻辑层接口，ExpressOrderLSer作为订单生成界面的逻辑层接口，HallLoadingOrderLSer作为营业厅装车单生成界面的逻辑层接口，IncomeOrderLSer作为入款单生成界面的逻辑层接口，PaymentOrderLSer作为付款单生成界面的逻辑层接口，ReceiveOrderLSer作为收件单生成界面的逻辑层接口，RecipientOrderLSer作为营业厅到达单生成界面的逻辑层接口，SendOrderLSer作为派件单生成界面的逻辑层接口，StockInOrderLSer作为入库单生成界面的逻辑层接口，StockOutOrderLSer作为出库单生成界面的逻辑层接口，TransitOrderLSer作为中转单生成界面的逻辑层接口。逻辑层和数据层之间添加ArriveOrderDSer作为中转中心到达单生成逻辑的数据层接口，CenterLoadingOrderDSer作为中转中心装车单生成逻辑的数据层接口，ExpressOrderDSer作为订单生成逻辑的数据层接口，HallLoadingOrderDSer作为营业厅装车单生成逻辑的数据层接口，IncomeOrderDSer作为入款单生成逻辑的数据层接口，PaymentOrderDSer作为付款单生成逻辑的数据层接口，ReceiveOrderDSer作为收件单生成逻辑的数据层接口，RecipientOrderDSer作为营业厅到达单生成逻辑的数据层接口，SendOrderDSer作为派件单生成逻辑的数据层接口，StockInOrderDSer作为入库单生成逻辑的数据层接口，StockOutOrderDSer作为出库单生成逻辑的数据层接口，TransitOrderDSer作为中转单生成逻辑的数据层接口。ArriveVO，CenterLoadingVO,ExpressVO,HallLoadingVO,IncomeVO,PaymentVO,ReceiveVO,RecipientVO,SendVO,StockIn,StockOut,TransitVO作为单据的界面值对象，相应PO作为相应单据的持久化对象。

OrderNewLogic模块的设计如图所示

OrderNewLogic各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| ArriveOrderLogic | 负责实现中转中心到达单生成界面所需要的业务逻辑 |
| CenterLoadingOrderLogic | 负责实现中转中心装车单生成界面所需要的业务逻辑 |
| ExpressOrderLogic | 负责实现订单生成界面所需要的业务逻辑 |
| HallLoadingOrderLogic | 负责实现营业厅装车单生成界面所需要的业务逻辑 |
| IncomeOrderLogic | 负责实现入款单生成界面所需要的业务逻辑 |
| PaymentOrderLogic | 负责实现付款单生成界面所需要的业务逻辑 |
| ReceiveOrderLogic | 负责实现收件单生成界面所需要的业务逻辑 |
| RecipientOrderLogic | 负责实现营业厅到达单生成界面所需要的业务逻辑 |
| SendOrderLogic | 负责实现派件单生成界面所需要的业务逻辑 |
| StockInOrderLogic | 负责实现入库单生成界面所需要的业务逻辑 |
| StockOutOrderLogic | 负责实现出库单生成界面所需要的业务逻辑 |
| TransitOrderLogic | 负责实现中转单生成界面所需要的业务逻辑 |

1. 模块内部类的接口规范

ArriveOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ArriveOrderLogic.Create | 语法 | public boolean Create(ArriveVO order) |
| 前置条件 | 到达单的各项信息已输入并确认 |
| 后置条件 | 将该新增到达单信息上传至数据库，返回上传是否成功 |
| ArriveOrderLogic.getCities | 语法 | public ResultMessage getCities() |
| 前置条件 | 用户输入城市 |
| 后置条件 | 在数据库中查找所有城市，并返回城市名称列表 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| ArriveOrderDSer.create (ArrivePO order) | 在数据库中添加ArrivePO对象 |
| ArriveOrderDSer. getCities() | 在数据库中获取并返回城市信息 |

CenterLoadingOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CenterLoadingOrderLogic.Create | 语法 | public boolean Create(CenterLoadingVO order) |
| 前置条件 | 中转中心装车单的各项信息已输入并确认 |
| 后置条件 | 将该新增中转中心装车单信息上传至数据库，返回上传是否成功 |
| CenterLoadingOrderLogic.getHalls | 语法 | public ResultMessage getHalls() |
| 前置条件 | 用户输入到达地（营业厅） |
| 后置条件 | 在数据库中查找所有营业厅，并返回营业厅名称列表 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| CenterLoadingOrderDSer.create(CenterLoadingPO order) | 在数据库中添加CenterLoadingPO对象 |
| CenterLoadingOrderDSer. getHalls() | 在数据库中获取并返回营业厅信息 |

ExpressOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ExpressOrderLogic.Create | 语法 | public boolean Create(ExpressVO order) |
| 前置条件 | 寄件单的各项信息已输入并确认 |
| 后置条件 | 将该寄件单信息上传至数据库，返回上传是否成功 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| ExpressOrderDSer.create(ExpressPO order) | 在数据库中添加ExpressPO对象 |

HallLoadingOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HallLoadingOrderLogic.Create | 语法 | public boolean Create(HallLoadingVO order) |
| 前置条件 | 营业厅装车单的各项信息已输入并确认 |
| 后置条件 | 将该新增营业厅装车单上传至数据库，返回上传是否成功 |
| HallLoadingOrderLogic.getOrgs | 语法 | public ResultMessage getOrgs() |
| 前置条件 | 用户输入到达地 |
| 后置条件 | 查找数据库中所有到达地，返回到达地名称列表 |
| HallLoadingOrderLogic.getVans | 语法 | public ResultMessage getVan() |
| 前置条件 | 用户输入车辆代号 |
| 后置条件 | 查找数据库中所有车辆，返回车辆代号列表 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| HallLoadingOrderDSer.create(HallLoadingPO order) | 在数据库中添加HallLoadingPO对象 |
| HallLoadingOrderDSer. getOrgs() | 在数据库中获取并返回机构信息 |
| HallLoadingOrderDSer. getVans() | 在数据库中获取并返回货车信息 |

IncomeOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| IncomeOrderLogic.Create | 语法 | public boolean Create(IncomeVO order) |
| 前置条件 | 收款单的各项信息已输入并确认 |
| 后置条件 | 将该新增收款单信息上传至数据库，返回上传是否成功 |
| IncomeOrderLogic.getCouriers | 语法 | public ResultMessage getCouriers(String id) |
| 前置条件 | 该id对应的营业厅在数据库中存在 |
| 后置条件 | 根据id在数据库中查找对应营业厅并返回该营业厅快递员姓名列表 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| IncomeOrderDSer.create(IncomePO order) | 在数据库中添加IncomePO对象 |
| IncomeOrderDSer.getCouriers() | 在数据库中获取并返回快递员信息 |

PaymentOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PaymentOrderLogic.Create | 语法 | public boolean Create(PaymentVO order) |
| 前置条件 | 付款单的各项信息已输入并确认 |
| 后置条件 | 将该新增付款单信息上传至数据库，返回上传是否成功 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| PaymentOrderDSer.create(PaymentPO order) | 在数据库中添加PaymentPO对象 |

ReceiveOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ReceiveOrderLogic.Create | 语法 | public boolean Create(ReceiveVO order) |
| 前置条件 | 收件单的各项信息已输入并确认 |
| 后置条件 | 将该新增收件信息上传至数据库，返回上传是否成功 |
| ReceiveOrderLogic.getExpress | 语法 | public ResultMessage getExpress(String id) |
| 前置条件 | 该id对应的订单信息在数据库中存在 |
| 后置条件 | 根据id在数据库中查找对应的订单信息并返回 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| ReceiveOrderDSer.create(ReceivePO order) | 在数据库中添加ReceivePO对象 |
| ReceiveOrderDSer.create(RecipientPO order) | 在数据库中添加RecipientPO对象 |

RecipientOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| RecipientOrderLogic.Create | 语法 | public boolean Create(RecipientVO order) |
| 前置条件 | 接收单的各项信息已输入并确认 |
| 后置条件 | 将该新增接收单信息上传至数据库，返回上传是否成功 |
| RecipientOrderLogic.getCities | 语法 | public ResultMessage getCities() |
| 前置条件 | 用户新建接收单 |
| 后置条件 | 在数据库中查找所有城市，并返回城市名称列表 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| RecipientOrderDSer.create(RecipientPO order) | 在数据库中添加RecipientPO对象 |
| RecipientOrderDSer.getCities() | 在数据库中查找并返回所有城市名称 |

SendOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| SendOrderLogic.Create | 语法 | public boolean Create(SendOrderVO order) |
| 前置条件 | 派件单的各项信息已输入并确认 |
| 后置条件 | 将该新增派件信息上传至数据库，返回上传是否成功 |
| SendOrderLogic.getCouriers | 语法 | public ResultMessage getCouriers(String id) |
| 前置条件 | 该id对应的营业厅在数据库中存在 |
| 后置条件 | 根据id在数据库中查找对应营业厅并返回该营业厅快递员姓名列表 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| SendOrderDSer.create(SendPO order) | 在数据库中添加SendPO对象 |
| SendOrderDSer.getCouriers(String id) | 在数据库中查找并返回对应快递员 |

StockInOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockInOrderLogic.Create | 语法 | public boolean Create(StockInOrderVO order) |
| 前置条件 | 入库单的各项信息已输入并确认 |
| 后置条件 | 将该新增入库信息上传至数据库，返回上传是否成功 |
| StockInOrderLogic.getCities | 语法 | public ResultMessage getCities() |
| 前置条件 | 用户输入城市 |
| 后置条件 | 在数据库中查找所有城市，并返回城市名称列表 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| StockInOrderDSer.create(StockInPO order) | 在数据库中添加StockInPO对象 |
| StockInOrderDSer.getCities() | 在数据库中查找并返回所有城市 |

StockOutOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockOutOrderLogic.Create | 语法 | public boolean Create(StockoutOrderVO order) |
| 前置条件 | 出库单的各项信息已输入并确认 |
| 后置条件 | 将该新增出库信息上传至数据库，返回上传是否成功 |
| StockOutOrderLogic.getCities | 语法 | public ResultMessage getCities() |
| 前置条件 | 用户输入城市 |
| 后置条件 | 在数据库中查找所有城市，并返回城市名称列表 |

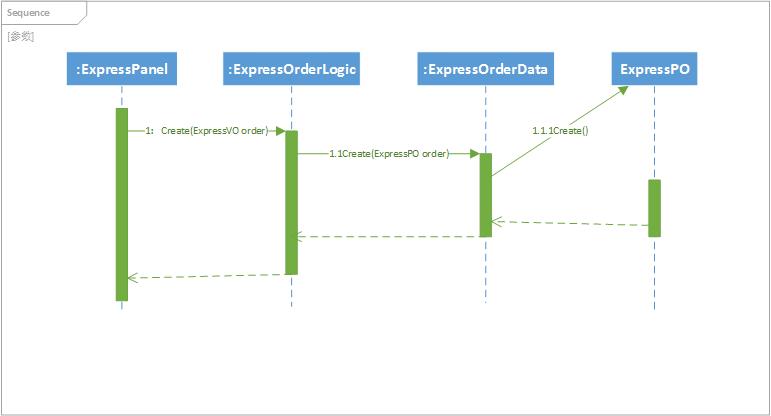
|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| StockOutOrderDSer.create(StockOutPO order) | 在数据库中添加StockOutPO对象 |
| StockOutOrderDSer.getCities() | 在数据库中查找并返回所有城市 |

TransitOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| TransitOrderLogic.Create | 语法 | public boolean Create(TransitOrderVO order) |
| 前置条件 | 中转单的各项信息已输入并确认 |
| 后置条件 | 将该新增中转单信息上传至数据库，返回上传是否成功 |
| TransitOrderLogic.getCities | 语法 | public ResultMessage getCities() |
| 前置条件 | 用户输入城市 |
| 后置条件 | 在数据库中查找所有城市，并返回城市名称列表 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| TransitOrderDSer.create(TransitPO order) | 在数据库中添加TransitPO对象 |
| TransitOrderDSer.getCities() | 在数据库中查找并返回所有城市 |

1. 业务逻辑层的动态模型

订单创建顺序图，表明了快递物流系统中，当快递员创建订单时，订单创建服务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

见4.1.1.（5）

* + 1. workOrgManLogic模块

1. 模块概述

workOrgManLogic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。

workOrgManLogic模块的职责及接口参见软件体系结构描述文档5.3.2中workOrgManLogic模块的接口规范

1. 整体结构

根据体系结构的设计，分层结构分为界面、逻辑和数据层。界面层和逻辑层之间添加DriverManLSer作为司机管理界面的逻辑层接口，OrgManLSer作为机构管理界面的逻辑层接口，VanManLSer作为车辆管理界面的逻辑层接口，WageManLSer作为薪水管理界面的逻辑层接口，WorkManLSer作为人员管理界面的逻辑层接口。逻辑层和数据层之间添加DriverManDSer作为司机管理逻辑的数据层接口，OrgManDSer作为机构管理逻辑的数据层接口，VanManDSer作为车辆管理逻辑的数据层接口，WageManDSer作为薪水管理逻辑的数据层接口，WorkManDSer作为人员管理逻辑的数据层接口。DriverVO,OrgVO,VanVO,WageVO,WorkVO分别作为司机、机构、车辆、薪水和人员的界面值对象。相应的PO分别作为相应持久化对象。UtilityLogic作为逻辑层共有的初始化操作的集合，依赖于数据层的UtilityDSer接口。CityPO含有城市信息。

workOrgManLogic模块的设计如图所示。

workOrgManLogic模块各个类的职责如表所示

workOrgManLogic模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| DriverManLogic | 负责司机管理界面所需要的业务逻辑 |
| OrgManLogic | 负责机构管理界面所需要的业务逻辑 |
| VanManLogic | 负责车辆管理界面所需要的业务逻辑 |
| WageManLogic | 负责薪水管理界面所需要的业务逻辑 |
| WorkManLogic | 负责人员管理界面所需要的业务逻辑 |

1. 模块内部类的接口规范

DriverManLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DriverManLogic.addDriver(DriverVO Driver) | 语法 | public boolean addDriver(DriverVO Driver) |
| 前置条件 | 已输入完整的司机信息，确认新增 |
| 后置条件 | 将司机信息发送到数据层并添加到数据库 |
| DriverManLogic.deleteDriver(String id) | 语法 | public boolean deleteDriver(String id) |
| 前置条件 | 选择司机，确认删除 |
| 后置条件 | 将ID发送到数据层并从库中删除相应司机的信息 |
| DriverManLogic.reviseDriver(DriverVO Driver) | 语法 | public boolean reviseDriver(DriverVO Driver) |
| 前置条件 | 确认修改司机信息 |
| 后置条件 | 将司机信息发送到数据层，并修改库中相应信息 |
| DriverManLogic.checkDriver(String id) | 语法 | public ResultMessage checkDriver(String id) |
| 前置条件 | 从司机列表中选择司机选择查看 |
| 后置条件 | 从数据层获得相应司机的信息返回 |
| DriverManLogic.searchDriver(String keyword) | 语法 | public ResultMessage searchDriver(String keyword) |
| 前置条件 | 打开司机管理界面或输入关键词检索 |
| 后置条件 | 从数据层获取司机列表或含有相应关键词的司机列表返回 |
| 需要的服务（需接口） | | |
| DriverManDSer.addDriver(DriverPO driver) | 将PO中的司机信息添加进数据库 | |
| DriverManDSer.deleteDriver(String id) | 将数据库中相应ID的司机信息删除 | |
| DriverManDSer.reviseDriver(DriverPO driver) | 在数据库中修改PO中信息相应的司机信息 | |
| DriverManDSer.checkDriver(String id) | 在数据库中获得相应ID司机的信息并返回 | |
| DriverManDSer.findDriver(String keyword) | 在数据库中获得含有相应关键词（若关键词为空则返回全部）的司机信息列表返回。 | |

OrgManLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrgManLogic.addOrg(OrgVO org) | 语法 | public boolean addOrg(OrgVO org) |
| 前置条件 | 已输入完整的机构信息，确认新增 |
| 后置条件 | 将机构信息发送到数据层并添加到数据库 |
| OrgManLogic.deleteOrg(String id) | 语法 | public boolean deleteOrg(String id) |
| 前置条件 | 选择机构，确认删除 |
| 后置条件 | 将ID发送到数据层并从库中删除相应机构的信息 |
| OrgManLogic.reviseOrg(OrgVO org) | 语法 | public boolean reviseOrg(OrgVO org) |
| 前置条件 | 确认修改机构信息 |
| 后置条件 | 将机构信息发送到数据层，并修改库中相应信息 |
| OrgManLogic.searchOrg(String keyword) | 语法 | public ResultMessage searchOrg(String keyword) |
| 前置条件 | 打开机构管理界面或输入关键词检索 |
| 后置条件 | 从数据层获取机构列表或含有相应关键词的机构列表返回 |
| OrgManLogic.getCities() | 语法 | public ResultMessage getCities() |
| 前置条件 | 打开机构管理界面 |
| 后置条件 | 获得城市信息列表返回 |
| 需要的服务（需接口） | | |
| OrgManDSer.addOrg(OrgPO Org) | 将PO中的机构信息添加进数据库 | |
| OrgManDSer.deleteOrg(String id) | 将数据库中相应ID的机构信息删除 | |
| OrgManDSer.reviseOrg(OrgPO Org) | 在数据库中修改PO中信息相应的机构信息 | |
| OrgManDSer.findOrg(String keyword) | 在数据库中获得含有相应关键词（若关键词为空则返回全部）的机构信息列表返回。 | |
| UtilityDSer.getCities() | 在数据库中获得城市信息列表并返回 | |

VanManLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| VanManLogic.addVan(VanVO Van) | 语法 | public boolean addVan(VanVO Van) |
| 前置条件 | 已输入完整的车辆信息，确认新增 |
| 后置条件 | 将车辆信息发送到数据层并添加到数据库 |
| VanManLogic.deleteVan(String id) | 语法 | public boolean deleteVan(String id) |
| 前置条件 | 选择车辆，确认删除 |
| 后置条件 | 将ID发送到数据层并从库中删除相应车辆的信息 |
| VanManLogic.reviseVan(VanVO Van) | 语法 | public boolean reviseVan(VanVO Van) |
| 前置条件 | 确认修改车辆信息 |
| 后置条件 | 将车辆信息发送到数据层，并修改库中相应信息 |
| VanManLogic.checkVan(String id) | 语法 | public ResultMessage checkVan(String id) |
| 前置条件 | 从车辆列表中选择车辆选择查看 |
| 后置条件 | 从数据层获得相应车辆的信息返回 |
| VanManLogic.searchVan(String keyword) | 语法 | public ResultMessage searchVan(String keyword) |
| 前置条件 | 打开车辆管理界面或输入关键词检索 |
| 后置条件 | 从数据层获取车辆列表或含有相应关键词的车辆列表返回 |
| 需要的服务（需接口） | | |
| VanManDSer.addVan(VanPO Van) | 将PO中的车辆信息添加进数据库 | |
| VanManDSer.deleteVan(String id) | 将数据库中相应ID的车辆信息删除 | |
| VanManDSer.reviseVan(VanPO Van) | 在数据库中修改PO中信息相应的车辆信息 | |
| VanManDSer.checkVan(String id) | 在数据库中获得相应ID车辆的信息并返回 | |
| VanManDSer.findVan(String keyword) | 在数据库中获得含有相应关键词（若关键词为空则返回全部）的车辆信息列表返回。 | |

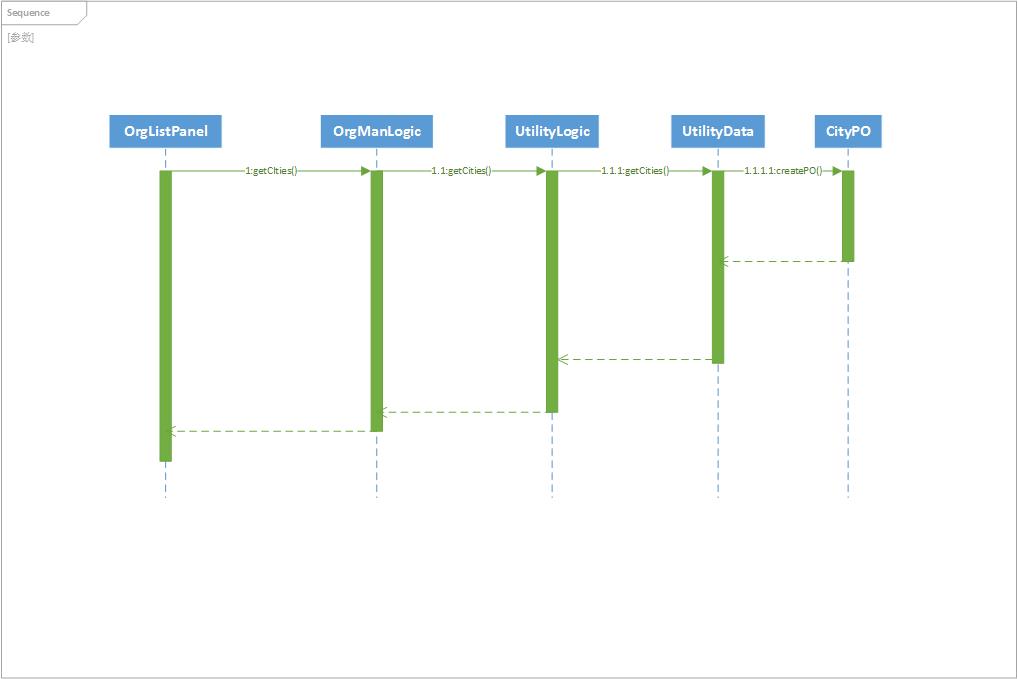
WageManLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| WageManLogic.searchWorker(String keyword) | 语法 | public ResultMessage searchWorker(String keyword) |
| 前置条件 | 打开薪水管理界面 |
| 后置条件 | 从数据库获得人员薪水列表返回 |
| WageManLogic.reviseWage(WageVO wage) | 语法 | public boolean reviseWage(WageVO wage) |
| 前置条件 | 确认薪水策略制定 |
| 后置条件 | 将薪水策略发送到数据层并修改数据库中相应人员的薪水策略 |
| 需要的服务（需接口） | | |
| WageManDSer.findWork(String keyword) | 在数据库中获得含有相应关键词（若关键词为空则返回全部）的人员薪水信息列表返回。 | |
| WageManDSer.reviseVan(WagePO Van) | 在数据库中修改PO中相应人员的薪水策略信息 | |

WorkManLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| WorkManLogic.addWork(WorkVO work) | 语法 | public boolean addWork(WorkVO Work) |
| 前置条件 | 已输入完整的人员信息，确认新增 |
| 后置条件 | 将人员信息发送到数据层并添加到数据库 |
| WorkManLogic.deleteWork(String id) | 语法 | public boolean deleteWork(String id) |
| 前置条件 | 选择人员，确认删除 |
| 后置条件 | 将ID发送到数据层并从库中删除相应人员的信息 |
| WorkManLogic.reviseWork(WorkVO work) | 语法 | public boolean reviseWork(WorkVO Work) |
| 前置条件 | 确认修改人员信息 |
| 后置条件 | 将人员信息发送到数据层，并修改库中相应信息 |
| WorkManLogic.searchWork(String keyword) | 语法 | public ResultMessage searchWork(String keyword) |
| 前置条件 | 打开机构管理界面或输入关键词检索 |
| 后置条件 | 从数据层获取机构列表或含有相应关键词的机构列表返回 |
| WorkManLogic.orgList() | 语法 | public ResultMessage orgList() |
| 前置条件 | 打开人员管理界面 |
| 后置条件 | 获得机构信息列表返回 |
| 需要的服务（需接口） | | |
| WorkManDSer.addWork(WorkPO Work) | 将PO中的人员信息添加进数据库 | |
| WorkManDSer.deleteWork(String id) | 将数据库中相应ID的人员信息删除 | |
| WorkManDSer.reviseWork(WorkPO Work) | 在数据库中修改PO中信息相应的人员信息 | |
| WorkManDSer.findWork(String keyword) | 在数据库中获得含有相应关键词（若关键词为空则返回全部）的人员信息列表返回。 | |
| UtilityDSer.getOrgs() | 在数据库中获得机构信息列表并返回 | |

1. 业务逻辑层的动态模型

下图表明了在打开机构管理界面后，获得城市信息列表时相关对象之间的协作。

1. 业务逻辑层的设计原理

见4.1.1.（5）

* + 1. cityLogic模块

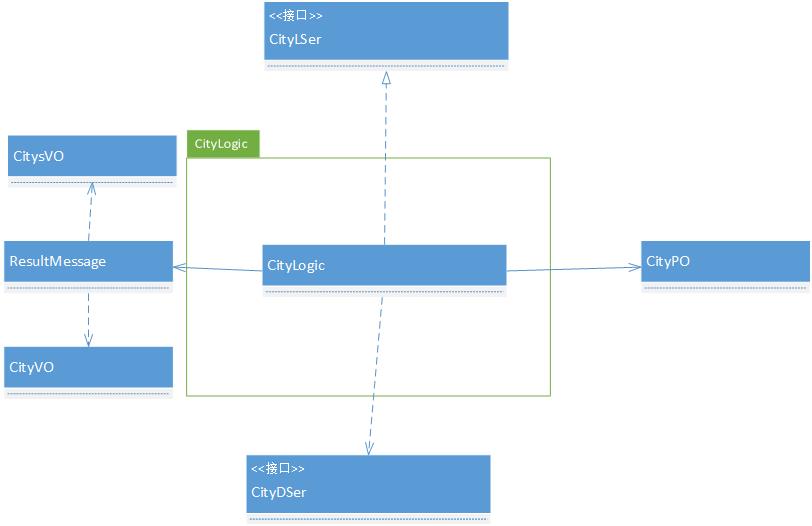
（1）模块概述

CityLogic模块承担的需求参见需求规格说明文档需求及相关非功能需求。CityLogic模块的职责及接口参见软件体系结构描述文档5.3.2中CityLogic模块的接口规范。

（2）整体结构

根据体系结构的设计，分层结构分为界面层，逻辑层和数据层。界面层和逻辑层之间添加CityLSer作为城市管理界面的逻辑层接口。逻辑层和数据层之间添加CityDSer作为城市管理逻辑的数据层接口。CityVO作为城市的界面值对象，CityPO作为城市的持久化对象。

CityLogic模块的设计如图所示



|  |  |
| --- | --- |
| 模块 | 职责 |
| CityLogic | 负责实现城市管理界面所需要的业务逻辑 |

（3）模块内部类的接口规范

cityLogic模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| cityLogic.addCity | 语法 | public boolean addCity(CityVO city); |
| 前置条件 | 增加一个城市 |
| 后置条件 | 将该城市信息加入数据库，返回添加结果 |
| cityLogic.reviseCity | 语法 | public boolean reviseCity(CityVO city); |
| 前置条件 | 修改了一个城市信息 |
| 后置条件 | 更新数据库已有信息，返回修改结果 |
| cityLogic.reviseCities | 语法 | public boolean reviseCities(CitiesVO city); |
| 前置条件 | 总经理已经选定了两个城市 |
| 后置条件 | 将两个城市间的距离和价格信息加入数据库或更新数据库中已有的距离与价格信息 |
| cityLogic.cityList | 语法 | public ResultMessage cityList(); |
| 前置条件 | 需要获取城市对象 |
| 后置条件 | 查找数据库中所有城市并返回城市对象列表 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| cityDSer.add(cityPO po) | 新增城市对象 |
| cityDSer.revise(CityPO city) | 更新城市对象信息 |
| cityDSer.reviseCities(CityPO cities) | 更新城市之间距离和价格 |
| cityDSer.getCities() | 查找并获取所有城市对象列表 |

（4）业务逻辑层的动态模型

新建城市顺序图，表明了快递物流系统中，当用户新增城市时，城市管理服务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

见4.1.1.（5）

* + 1. accountLogic模块

1. 模块概述

accountLogic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。

accountLogic模块的职责及接口参见软件体系结构描述文档5.3.2中accountLogic模块的接口规范

1. 整体结构

根据体系结构的设计，分层结构分为界面、逻辑和数据层。界面和逻辑层间添加AccountLSer作为账户管理界面的业务逻辑接口，CountLSer作为账管理界面的业务逻辑接口。逻辑层和数据层间添加AccountDSer作为账户管理逻辑的数据层接口，CountDSer作为账管理逻辑的数据层接口。AccountVO和CountVO分别为账户和账的界面值对象，相应的PO为持久化对象。ResultMessage存放操作是否成功和要返回界面的信息。UtilityLogic负责处理一些逻辑初始化操作。

accountLogic模块的设计如图所示。

accountLogic模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| AccountLogic | 负责账户管理界面所需要的业务逻辑 |
| CountLogic | 负责期初建账和账信息界面所需要的业务逻辑 |

1. 模块内部类的接口规范

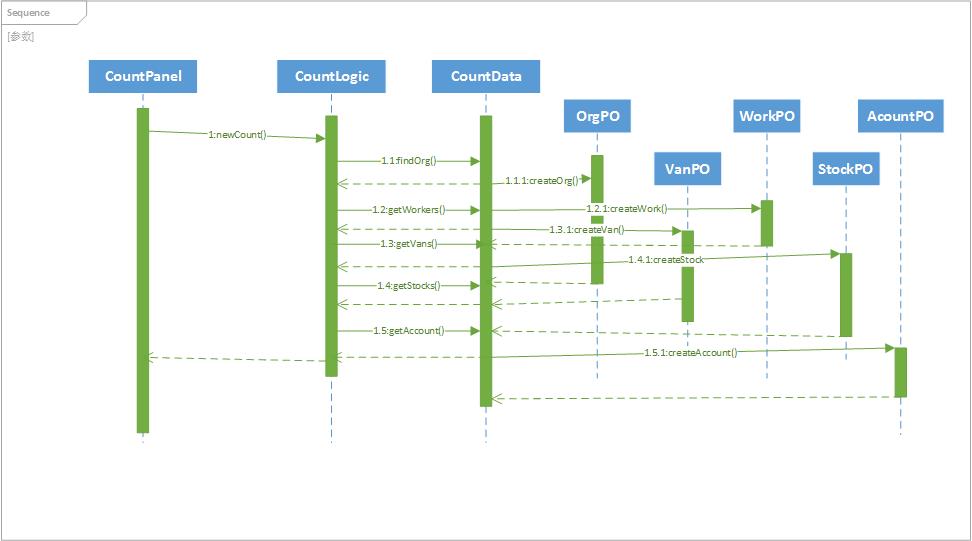
AccountLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| AccountLogic.addAccount(AccountVO account) | 语法 | public boolean addAccount(AccountVO account) |
| 前置条件 | 已输入完整的账户信息，确认新增 |
| 后置条件 | 将账户信息发送到数据层并添加到数据库 |
| AccountLogic.deleteAccount(String id) | 语法 | public boolean deleteAccount(String id) |
| 前置条件 | 选择账户，确认删除 |
| 后置条件 | 将ID发送到数据层并从库中删除相应账户的信息 |
| AccountLogic.reviseAccount(AccountVO account) | 语法 | public boolean reviseAccount(AccountVO account) |
| 前置条件 | 确认修改账户信息 |
| 后置条件 | 将账户信息发送到数据层，并修改库中相应信息 |
| AccountLogic.searchAccount(String keyword) | 语法 | public ResultMessage searchAccount(String keyword) |
| 前置条件 | 打开账户管理界面或输入关键词检索 |
| 后置条件 | 从数据层获取账户列表或含有相应关键词的账户列表返回 |
| 需要的服务（需接口） | | |
| AccountDSer.addAccount(AccountPO account) | 将PO中的账户信息添加进数据库 | |
| AccountDSer.deleteAccount(String id) | 将数据库中相应ID的账户信息删除 | |
| AccountDSer.reviseAccount(AccountPO account) | 在数据库中修改PO中信息相应的账户信息 | |
| AccountDSer.findAccount(String keyword) | 在数据库中获得含有相应关键词（若关键词为空则返回全部）的账户信息列表返回。 | |

CountLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CountLogic.newCount() | 语法 | public boolean newCount() |
| 前置条件 | 用户选择了新增期初信息 |
| 后置条件 | 获得当前所有机构、人员、车辆、库存和账户信息，将其处理存储为期初信息。 |
| CountLogic.checkInitInfo() | 语法 | public ResultMessage checkInitInfo() |
| 前置条件 | 用户选择账，选择查看期初信息 |
| 后置条件 | 从数据层获得期初信息返回给界面 |
| 需要的服务（需接口） | | |
| CountDSer.addCount(CountPO count) | 将PO中的账信息添加到数据库 | |
| CountDSer.findInitInfo() | 从数据库中获得期初信息 | |
| CountDSer.findOrg() | 从数据库中获得所有机构的信息 | |
| CountDSer.getWorkers() | 从数据库中获得所有人员的信息 | |
| CountDSer.getVans() | 从数据库中获得所有车辆的信息 | |
| CountDSer.getStocks() | 从数据库中获得所有库存的信息 | |
| CountDSer.getAccount() | 从数据库中获得所有账户的信息 | |

1. 业务逻辑层的动态模型

下图表现了在选择新建账之后各个类协作的顺序情况

1. 业务逻辑层的设计原理

见4.1.1.（5）

* + 1. stockLogic模块

1. 模块概述

stockLogic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。

stockLogic模块的职责及接口参见软件体系结构描述文档5.3.2中stockLogic模块的接口规范

1. 整体结构

根据体系结构的设计，分层结构分为界面、逻辑和数据层。界面和逻辑层间添加StockCheckLSer作为库存查看界面的业务逻辑接口，StockCheckNowLSer作为库存盘点界面的业务逻辑接口，StockCheckWarnLSer作为库存报警界面的业务逻辑接口，StockDivideLSer作为库存分区界面的业务逻辑接口，StockWarningLSer作为库存报警设置界面的业务逻辑接口。逻辑层和数据层之间添加StockCheckDSer作为库存查看逻辑的数据层接口，StockCheckNowDSer作为库存盘点业务逻辑的数据层接口，StockCheckWarnDSer作为库存报警业务逻辑的数据层接口，StockDivideDSer作为库存分区业务逻辑的数据层接口，StockWarningDSer作为库存报警设置业务逻辑的数据层接口。StockVO,ShelfVO,GoodVO,StockInVO,StockOutVO分别存储库存、架、货物、入库单和出库单的值对象，相应的PO存储相应的持久化对象。ResultMessage存储操作是否成功和返回的信息。StockInOrderLogic是入库单生成逻辑，需要通过StockCheckWarnLogic检测是否需要报警。

stockLogic模块的设计如图所示。

stockLogic模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| StockCheckLogic | 负责库存查看界面所需要的业务逻辑 |
| StockCheckNowLogic | 负责库存盘点界面所需要的业务逻辑 |
| StockCheckWarnLogic | 负责库存报警界面所需要的业务逻辑 |
| StockDivideLogic | 负责库存分区界面所需要的业务逻辑 |
| StockWarningLogic | 负责库存报警设置界面所需要的业务逻辑 |

1. 模块内部类的接口规范

StockCheckLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockCheckLogic.checkStockIn(Date start, Date end, String id) | 语法 | public ResultMessage checkStockIn(Date start, Date end, String id) |
| 前置条件 | 输入了开始日期和结束日期 |
| 后置条件 | 从数据层获取开始和结束日期之间的入库单 |
| StockCheckLogic.checkStockOut(Date start, Date end, String id) | 语法 | public ResultMessage checkStockOut(Date start, Date end, String id) |
| 前置条件 | 输入了开始日期和结束日期 |
| 后置条件 | 从数据层获取开始和结束日期之间的出库单 |
| 需要的服务（需接口） | | |
| StockCheckDSer.getIn(Date start,Date end,String id) | 从数据库中获得开始和结束日期之间的入库单 | |
| StockCheckDSer.getOut(Date start,Date end,String id) | 将数据库中获得开始和结束日期之间的出库单 | |

StockCheckNowLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockCheckNowLogic.checkNow(String id) | 语法 | public ResultMessage checkNow(String id) |
| 前置条件 | 打开库存盘点界面 |
| 后置条件 | 从数据层获取当前库存快照信息 |
| StockCheckLogic.setPoint(String id, Date date) | 语法 | public boolean setPoint(String id, Date date) |
| 前置条件 | 用户确认盘点 |
| 后置条件 | 将当前时间作为截止点传往数据层，存入数据库 |
| StockCheckLogic.outputExcel(String location, StockVO stock) | 语法 | public boolean outputExcel(String location, StockVO stock) |
| 前置条件 | 用户选择路径并确认导出为Excel |
| 后置条件 | 导出Excel并打开，返回操作是否成功 |
| 需要的服务（需接口） | | |
| StockCheckNowDSer.getStock(String id) | 从数据库中获得当前库存快照 | |
| StockCheckDNowSer.setPoint(String id,Date date) | 在数据库中添加库存盘点截止点 | |

StockCheckWarnLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockCheckWarnLogic.checkWarning (String id) | 语法 | public ResultMessage checkWarning(String id) |
| 前置条件 | 新建了一个入库单 |
| 后置条件 | 比较当前库存量和库存警戒线比例，返回是否超出警戒线 |
| 需要的服务（需接口） | | |
| StockCheckWarnDSer.getWarning(String id) | 从数据库中获得库存警戒线 | |
| StockCheckWarnDSer.getStock() | 从数据库获得当前库存情况 | |

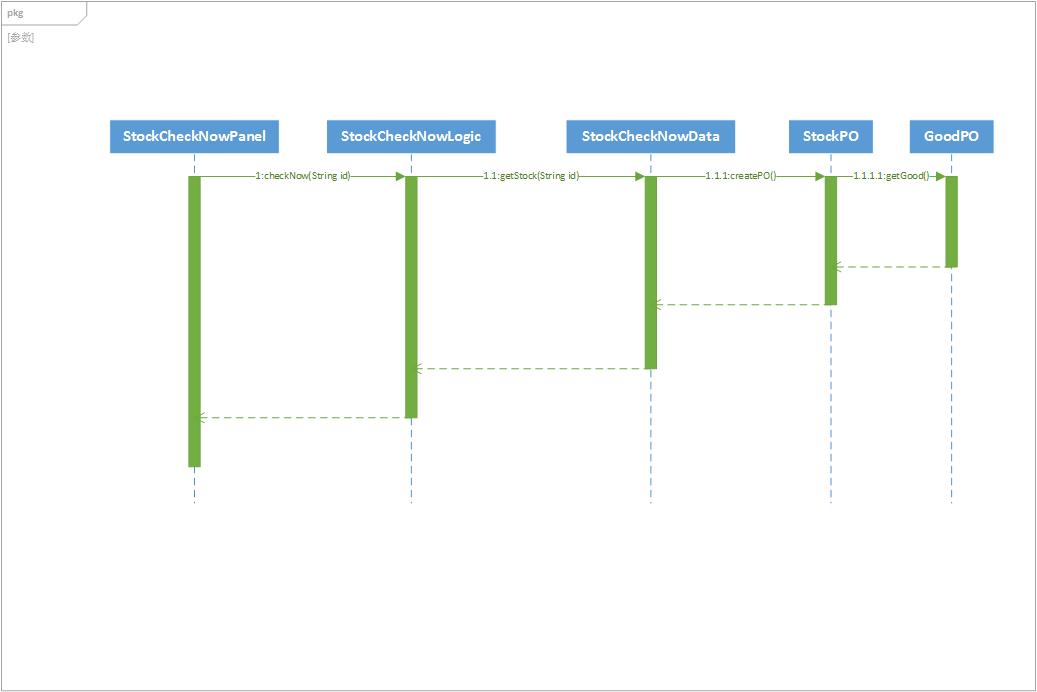
StockDivideLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockDivideLogic.searchShelf(String id,String keyword) | 语法 | public ResultMessage searchShelf(String id,String keyword) |
| 前置条件 | 打开库存分区界面或输入关键词搜索架 |
| 后置条件 | 从数据库获得架列表或含有关键词的架列表 |
| StockDivideLogic.addShelf(ShelfVO shelf) | 语法 | public boolean addShelf(ShelfVO shelf) |
| 前置条件 | 输入了完整的架信息，选择添加 |
| 后置条件 | 将架信息发送到数据层并添加入库 |
| StockDivideLogic.deleteShelf(String id) | 语法 | public boolean deleteShelf(String id) |
| 前置条件 | 选择一个架，确认删除 |
| 后置条件 | 从数据库中将相应ID的架信息清除 |
| StockDivideLogic.reviseShelf(ShelfVO shelf) | 语法 | public boolean reviseShelf(ShelfVO shelf) |
| 前置条件 | 修改一个架的信息，确认修改 |
| 后置条件 | 将架信息发送到数据层并修改库中的相应信息 |
| 需要的服务（需接口） | | |
| StockDivideDSer.getShelves(String id,String keyword) | 从数据库获得架列表或含有关键字的架列表 | |
| StockDivideDSer.addShelf(ShelfPO shelf) | 向数据库中添加架信息 | |
| StockDivideDSer.deleteShelf(String id) | 从数据库中删除ID相应的架信息 | |
| StockDivideDSer.reviseShelf(ShelfPO shelf) | 在数据库中修改相应ID的架信息 | |

StockWarningLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockWarningLogic.setWarning(int w, Part part, String id) | 语法 | public boolean setWarning(int w, Part part, String id) |
| 前置条件 | 设置了一个区的警戒线 |
| 后置条件 | 将警戒线发送到数据层并修改数据库中的相应警戒线 |
| StockWarningLogic.getWarning(String id) | 语法 | public ResultMessage getWarning(String id) |
| 前置条件 | 打开库存分区界面 |
| 后置条件 | 从数据层中获得库存各区的警戒线 |
| 需要的服务（需接口） | | |
| StockWarningDSer.getWarning(String id) | 从数据库中获得库存警戒线 | |
| StockWarningDSer.setWarning(int w,Part part,String id) | 修改数据库中相应区的警戒线 | |

1. 业务逻辑层的动态模型

如图为执行库存盘点操作后，相关类的协作顺序图。

下图表现了库存报警的状态转换情况。

1. 业务逻辑层的设计原理

见4.1.1。（5）

* + 1. checkLogic模块

（1）模块概述

CheckLogic模块承担的需求参见需求规格说明文档需求及相关非功能需求。CheckLogic模块的职责及接口参见软件体系结构描述文档5.3.2中CheckLogic模块的接口规范。

（2）整体结构

根据体系结构的设计，分层结构分为界面层，逻辑层和数据层。界面层和逻辑层之间添加CostBeneLSer作为成本收益表查看和导出界面的逻辑层接口，BusiCircumLSer作为经营情况表查看和导出界面的逻辑层接口，CollectFromLSer作为收款记录查看界面的逻辑层接口，DiaryLSer作为日志记录查看界面的逻辑层接口。逻辑层和数据层之间添加CostBeneDSer作为成本收益表查看和导出逻辑的数据层接口，BusiCircumDSer作为经营情况表查看和导出逻辑的数据层接口，CollectFromDSer作为收款记录查看逻辑的数据层接口，DiaryDSer作为日志记录查看逻辑的数据层接口。IncomeVO作为入款单的界面值对象，PaymentVO作为付款单的界面值对象，DiaryVO作为日志记录的界面值对象，IncomePO作为入款单的持久化对象，PaymentPO作为付款单的持久化对象，DiaryPO作为日志记录的持久化对象。

CheckLogic模块的设计如图所示



CheckLogic各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| CostBeneLogic | 负责实现成本收益表查看和导出界面所需要的业务逻辑 |
| BusiCircumLogic | 负责实现经营情况表查看和导出界面所需要的业务逻辑 |
| DiaryLogic | 负责实现操作日志查看界面所需要的业务逻辑 |
| CollectRecordLogic | 负责实现收款单记录查看界面所需要的业务逻辑 |

（3）模块内部类的接口规范

CostBeneLogic模块的接口规范

|  |  |  |
| --- | --- | --- |
| CostBeneLogic. getIncome | 语法 | public ResultMessage getIncome () |
| 前置条件 | 无 |
| 后置条件 | 返回入款单 |
| CostBeneLogic. getPayment | 语法 | public ResultMessage getPayment () |
| 前置条件 | 无 |
| 后置条件 | 返回付款单 |
| CostBeneLogic.outputExcel | 语法 | public boolean outputExcel(ArrayList<IncomeVO>)income，ArrayList<PaymentVO> payment，String location) |
| 前置条件 | 用户选择导出成本收益表 |
| 后置条件 | 导出经营情况表到相应地址 |
| 提供的服务（供接口） | | |

|  |  |
| --- | --- |
| CostBeneDSer. getIncome() | 从数据库中获得入款单记录，返回数据 |
| CostBeneDSer. getPayment() | 从数据库中获得付款单记录，返回数据 |

BusiCircumLogic模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| BusiCircumLogic.seeStatementSheet | 语法 | public ResultMessage seeBusinessCircum (Date start, Date end) |
| 前置条件 | 输入的日期符合输入规则 |
| 后置条件 | 根据输入的开始和结束日期，返回由入款单和收款单信息组成的经营情况表 |
| BusiCircumLogic.getHall | 语法 | public ResultMessage getHall() |
| 前置条件 | 用户选择查看经营情况表 |
| 后置条件 | 返回所有营业厅的名称 |
| BusiCircumLogic.outputExcel | 语法 | public boolean outputExcel(ArrayList<IncomeVo> income, String location) |
| 前置条件 | 用户选择导出经营情况表 |
| 后置条件 | 导出经营情况表到相应地址 |
| 提供的服务（供接口） | | |

|  |  |
| --- | --- |
| BusiCircumDSer.findCollects(Date start, Date end) | 根据开始和结束日期从数据库中获得收款单记录，返回数据 |
| BusiCircumDSer.getHall() | 返回所有营业厅的编号和名称 |

DiaryLogic模块的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| DiaryLogic.seeDiary | 语法 | public ResultMessage seeDiary(Date start, Date end) |
| 前置条件 | 输入的日期符合输入规则 |
| 后置条件 | 根据输入的开始和结束日期，返回相应的日志记录 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| DiaryDSer.seeDiary(Date start, Date end) | 根据开始和结束日期从数据库中获得日志记录，返回数据 |

CollectRecordLogic模块的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| CollectRecordLogic.seeCollectRecord | 语法 | public ResultMessage seeCollectRecord(Date date, String id) |
| 前置条件 | 输入了日期，选择了营业厅 |
| 后置条件 | 根据输入的日期，返回当日的收款记录 |
| CollectRecordLogic.total | 语法 | public ResultMessage total(ArrayList<IncomeVO> income) |
| 前置条件 | 用户选择合计收款 |
| 后置条件 | 返回合计的收款总额 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| CollectRecordDSer.getCollectRec (Date date, String id) | 根据给的日期和营业厅编号从数据库中获得当日该营业厅收款记录，返回数据 |
| CollectRecordDSer.getHall() | 返回所有营业厅的编号和名称 |

（4）业务逻辑层的动态模型

查看收款记录顺序图，表明了快递物流系统中，当用户查看收款记录时，收款记录查看服务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

见4.1.1.（5）

* + 1. orderApproveLogic模块

1. 模块概述

orderApproveLogic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。

orderApproveLogic模块的职责及接口参见软件体系结构描述文档5.3.2中orderApproveLogic模块的接口规范

1. 整体结构

根据体系结构的设计，分层结构分为界面、逻辑和数据层。界面和逻辑层间添加OrderApproveLSer作为单据审批的逻辑层接口，OrderResubmitLSer作为未通过单据处理的逻辑层接口。逻辑和数据层间添加OrderApproveDSer作为单据审批的数据层接口，OrderResubmitDSer作为未通过单据处理的数据层接口。OrderVO存放单据的类型、操作者、编号和生成日期，是值对象。OrderPO是相应的持久化对象。ResultMessage包含返回是否成功和要返回的信息。ArriveVO,CenterLoadingVO,ExpressVO,HallLoadingVO,IncomeVO,PaymentVO,ReceiveVO,RecipientVO,SendVO,StockInVO,StockOutVO,TransitVO分别为到达单、中转中心装车单、快递单、营业厅装车单、收款单、付款单、收件单、接收单、派件单、入库单、出库单和中转单的值对象，相应PO为其持久化对象。

如图为orderApproveLogic模块的设计。

orderApproveLogic模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| OrderApproveLogic | 负责单据审批界面所需要的业务逻辑 |
| OrderResubmitLogic | 负责获得未通过单据所需要的业务逻辑 |

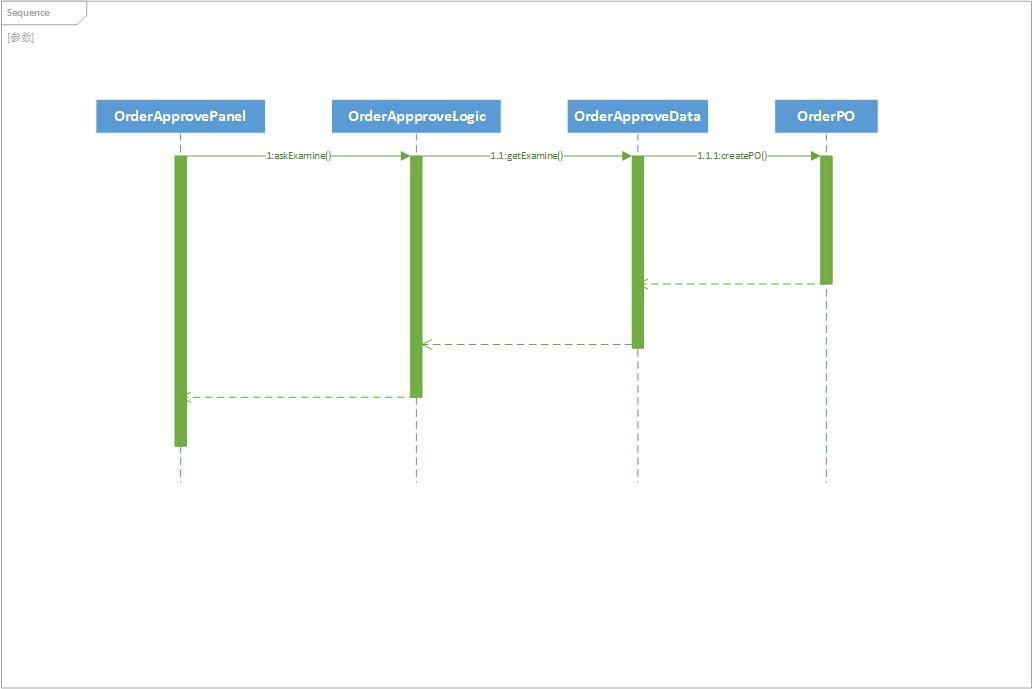
1. 模块内部类的接口规范

OrderApproveLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderApproveLogic.askExamine() | 语法 | public ResultMessage askExamine() |
| 前置条件 | 打开审批单据界面 |
| 后置条件 | 从数据层获取所有未审批单据返回 |
| OrderApproveLogic.examine(boolean approve, ArrayList<String> id) | 语法 | public boolean examine(boolean approve, ArrayList<String> id) |
| 前置条件 | 选择一批单据，选择通过或不通过 |
| 后置条件 | 将该批单据的通过情况传入数据层并改变单据在数据库中的存放位置 |
| OrderApproveLogic.chooseOrder(String id) | 语法 | public ResultMessage chooseOrder(String id) |
| 前置条件 | 选择一个单据，选择查看 |
| 后置条件 | 获取单据信息并返回 |
| 需要的服务（需接口） | | |
| OrderApproveDSer.getExamine() | 从数据库中获得未审批单据 | |
| OrderApproveDSer.update(boolean isPassed,ArrayList<String> id) | 更新数据库中ID相应单据的审批情况并改变其存放状态 | |
| OrderApproveDSer.getOrder(String id) | 从数据库中获得ID相应单据的信息 | |

OrderResubmitLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderResubmitLogic.getOrders(String id) | 语法 | public ResultMessage getOrders(String id) |
| 前置条件 | 有生成单据功能的用户已经登录 |
| 后置条件 | 轮询从数据库获取自己机构的未通过单据 |
| 需要的服务（需接口） | | |
| OrderResubmitDSer.getOrder(String id) | 从数据库中相应机构所有的未通过单据 | |

1. 业务逻辑层的动态模型

上图表示了单据审批界面打开后获得未审批单据列表时各个类的协作。

上图表示了轮询获得未通过单据的状态转换情况。

1. 业务逻辑层的设计原理

见4.1.1.（5）

* 1. 数据层分解

数据层的开发包图参见软件体系结构文档。

* + 1. userData模块

1. 模块概述

userData模块承担的需求参见需求规格说明文档需求及相关非功能需求。userData模块的职责及接口参见软件体系结构描述文档5.3.2中userData模块的接口规范。

1. 整体结构

根据体系结构的设计，分层结构分为界面层，逻辑层和数据层。逻辑层和数据层之间添加LoginDSer作为登录逻辑的数据层接口，UserManDSer作为用户管理逻辑的数据层接口。UserPO作为用户的持久化对象。

UserData模块的设计如图所示



UserData各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| LoginData | 负责实现登录逻辑所需要的数据处理 |
| UserManData | 负责实现用户管理逻辑所需要的数据处理 |

1. 模块内部类的接口规范

LoginData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| LoginData.findUser | 语法 | public UserPO login(String id,String password) |
| 前置条件 | 逻辑层请求登录 |
| 后置条件 | 从数据库中找到并返回相应用户信息 |

UserManData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserManData.newUser | 语法 | public boolean newUser(UserPO user) |
| 前置条件 | 逻辑层请求新建 |
| 后置条件 | 在数据库中增加UserPO |
| UserManData.deleteUser | 语法 | public boolean deleteUser(String id) |
| 前置条件 | 逻辑层请求删除 |
| 后置条件 | 在数据库中删除对应UserPO |
| UserManData.reviseUser | 语法 | public boolean reviseUser(UserPO user) |
| 前置条件 | 逻辑层请求修改 |
| 后置条件 | 在数据库中修改对应UserPO |
| UserManData.findUser | 语法 | public ArrayList<UserPO> findUser(String keyword) |
| 前置条件 | 逻辑层请求查询 |
| 后置条件 | 在数据库中查找并返回对应UserPO |

1. 数据层的动态模型

登陆顺序图，表明了快递物流系统中，当用户输入用户名和密码进行登陆时，登陆数据服务处理的相关对象之间的协作。



1. 数据层的设计原理

利用分散式控制风格，每个界面需要访问的业务逻辑由对应对象负责。

* + 1. transitInfoData模块

1. 模块概述

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

transitInfoData模块的职责及接口参见软件体系结构描述文档5.3.2中transitInfoData模块的接口规范。

1. 整体结构

根据体系结构的设计，分层结构分为界面层，逻辑层和数据层。界面层和逻辑层之间添加transitInfoLogicSer作为账户管理界面的逻辑层接口。数据层和逻辑层之间添加transitInfoDSer作为账户管理逻辑的数据层接口。transitInfoDataSerImpl是基于Mysql数据库的持久化数据接口，实现TransitInfoDataSer的服务。

模块各个类的职责如表所示。

transitInfoData各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| CourierData | 负责实现快递员的物流信息查询数据处理 |
| CustomerData | 负责实现用户的物流信息查询数据处理 |

transitInfoData模块设计如图所示。



1. 模块内部类的接口规范

CourierData的接口规范

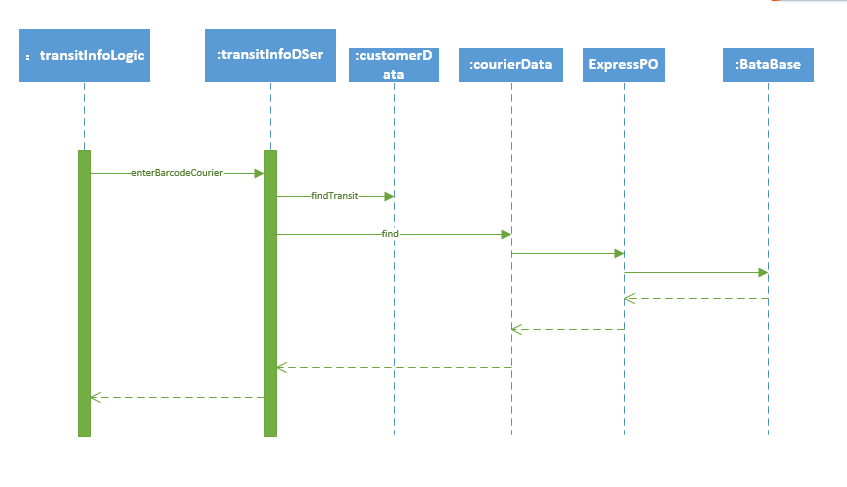
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Courier.find | 语法 | public ExpressPO find(String barcode) throws RemoteException |
| 前置条件 | 有相应条形码存在 |
| 后置条件 | 按barcode进行查找返回相应的ExpressPO结果 |

CustomerData模块的接口规范

|  |  |  |
| --- | --- | --- |
| CustomerData.findTransit | 语法 | public ArrayList<String> findTransit(String barcode) throws RemoteException |
| 前置条件 | 有相应条形码存在 |
| 后置条件 | 按barcode进行查找返回相应订单的物流轨迹和货运信息 |

1. 数据层的动态模型

物流信息查询顺序图，表明了物流信息查询过程相关数据对象之间的协作。



1. 数据层的设计原理

见4.2.1.(5)

* + 1. orderNewData模块

（1）模块概述

OrderNewData模块承担的需求参见需求规格说明文档需求及相关非功能需求。OrderNewData模块的职责及接口参见软件体系结构描述文档5.3.2中OrderNewData模块的接口规范。

（2）整体结构

根据体系结构的设计，分层结构分为界面层，逻辑层和数据层。逻辑层和数据层之间添加ArriveOrderDSer作为中转中心到达单生成逻辑的数据层接口，CenterLoadingOrderDSer作为中转中心装车单生成逻辑的数据层接口，ExpressOrderDSer作为订单生成逻辑的数据层接口，HallLoadingOrderDSer作为营业厅装车单生成逻辑的数据层接口，IncomeOrderDSer作为入款单生成逻辑的数据层接口，PaymentOrderDSer作为付款单生成逻辑的数据层接口，ReceiveOrderDSer作为收件单生成逻辑的数据层接口，RecipientOrderDSer作为营业厅到达单生成逻辑的数据层接口，SendOrderDSer作为派件单生成逻辑的数据层接口，StockInOrderDSer作为入库单生成逻辑的数据层接口，StockOutOrderDSer作为出库单生成逻辑的数据层接口，TransitOrderDSer作为中转单生成逻辑的数据层接口。

ArrivePO,CenterLoadingPO,ExpressPO,HallLoadingPO,IncomePO,PaymentPO,ReceivePO,RecipientPO,SendPO,StockInPO,StockOutPO,TransitPO作为单据的持久化对象。

OrderNewData模块的设计如图所示

OrderNewData各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| ArriveOrderData | 负责实现中转中心到达单生成逻辑所需要的数据服务 |
| CenterLoadingOrderData | 负责实现中转中心装车单生成逻辑所需要的数据服务 |
| ExpressOrderData | 负责实现订单生成逻辑所需要的数据服务 |
| HallLoadingOrderData | 负责实现营业厅装车单生成逻辑所需要的数据服务 |
| IncomeOrderData | 负责实现入款单生成逻辑所需要的数据服务 |
| PaymentOrderData | 负责实现付款单生成逻辑所需要的数据服务 |
| ReceiveOrderData | 负责实现收件单生成逻辑所需要的数据服务 |
| RecipientOrderData | 负责实现营业厅到达单生成逻辑所需要的数据服务 |
| SendOrderData | 负责实现派件单生成逻辑所需要的数据服务 |
| StockInOrderData | 负责实现入库单生成逻辑所需要的数据服务 |
| StockOutOrderData | 负责实现出库单生成逻辑所需要的数据服务 |
| TransitOrderData | 负责实现中转单生成逻辑所需要的数据服务 |

（3）模块内部类的接口规范

ArriveOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| ArriveOrderData.Create | 语法 | public boolean Create(ArrivePO order) |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个ArriveOrder记录 |
| ArriveOrderData.getCities | 语法 | public ArrayList<String> getCities () throws RemoteException |
| 前置条件 | 新建单据逻辑请求城市信息 |
| 后置条件 | 在数据库中获取并返回城市信息 |

CenterLoadingOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| CenterLoadingOrderData.Create | 语法 | public boolean create(CenterLoadingPO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个CenterLoadingOrder记录 |
| CenterLoadingOrderData.getHalls | 语法 | public ArrayList<String> getHalls () throws RemoteException |
| 前置条件 | 新建单据逻辑请求营业厅信息 |
| 后置条件 | 在数据库中获取并返回营业厅信息 |

ExpressOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| ExpressOrderData.Create | 语法 | public boolean create (ExpressPO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个Order记录 |

HallLoadingOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| HallLoadingOrderData.Create | 语法 | public boolean create (HallLoadingPO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个Order记录 |
| HallLoadingOrderData.getOrgs | 语法 | public ArrayList<String> getOrgs () throws RemoteException |
| 前置条件 | 新建单据逻辑请求机构信息 |
| 后置条件 | 在数据库中获取并返回机构信息 |
| HallLoadingOrderData.getVans | 语法 | public ArrayList<String> getVans() throws RemoteException |
| 前置条件 | 新建单据逻辑请求货车信息 |
| 后置条件 | 在数据库中获取并返回货车信息 |

IncomeOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| IncomeOrderData.Create | 语法 | public boolean create (IncomePO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个Order记录 |
| IncomeOrderData.getCouriers | 语法 | public ArrayList<String> getCouriers () throws RemoteException |
| 前置条件 | 新建单据逻辑请求快递员信息 |
| 后置条件 | 在数据库中获取并返回快递员信息 |

PaymentOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| PaymentOrderData.Create | 语法 | public boolean create (PaymentPO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个Order记录 |

ReceiveOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| ReceiveOrderData.Create | 语法 | public boolean create (ReceivePO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个Order记录 |
| ReceiveOrderData.getExpress | 语法 | public ExpressPO getExpress (String id) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中获取并返回物流信息 |

RecipientOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| RecipientOrderData.Create | 语法 | public boolean Create(RecipientVO order) |
| 前置条件 | 接收单的各项信息已输入并确认 |
| 后置条件 | 将该新增接收单信息上传至数据库，返回上传是否成功 |
| RecipientOrderData.Create | 语法 | public ArrayList<String> getCities () throws RemoteException |
| 前置条件 | 新建单据逻辑请求城市信息 |
| 后置条件 | 在数据库中获取并返回城市信息 |

SendOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| SendOrderData.create | 语法 | public boolean create (SendPO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个Order记录 |
| SendOrderData.getCouriers | 语法 | public ArrayList<String> getCouriers () throws RemoteException |
| 前置条件 | 单据新建逻辑请求快递员信息 |
| 后置条件 | 在数据库中获取并返回快递员信息 |

StockInOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| StockInOrderDSer.create | 语法 | public boolean create (StockInPO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个Order记录 |
| StockInOrderDSer.getCities | 语法 | public ArrayList<String> getCities () throws RemoteException |
| 前置条件 | 单据新建逻辑请求城市信息 |
| 后置条件 | 在数据库中获取并返回城市信息 |

StockOutOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| StockOutOrderDSer.create | 语法 | public boolean create (StockOutPO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个Order记录 |
| StockOutOrderDSer.getCities | 语法 | public ArrayList<String> getCities () throws RemoteException |
| 前置条件 | 单据新建逻辑请求城市信息 |
| 后置条件 | 在数据库中获取并返回城市信息 |

TransitOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| TransitOrderDSer.create | 语法 | public boolean create (TransitPO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个Order记录 |
| TransitOrderDSer.getCities | 语法 | public ArrayList<String> getCities () throws RemoteException |
| 前置条件 | 单据新建逻辑请求城市信息 |
| 后置条件 | 在数据库中获取并返回城市信息 |

（4）数据层的动态模型

订单创建顺序图，表明了快递物流系统中，当快递员创建订单时，订单创建数据服务处理的相关对象之间的协作。

（5）数据层的设计原理

见4.2.1.（5）

* + 1. workOrgManData模块

1. 模块概述

WorkOrgManData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。WorkOrgManData模块的职责及接口参见软件体系结构描述文档5.3.2中WorkOrgManData模块的接口规范

1. 整体结构

根据体系结构的设计，分层结构分为界面、逻辑和数据层。逻辑层和数据层之间添加DriverManDSer作为司机管理逻辑的数据层接口，OrgManDSer作为机构管理逻辑的数据层接口，VanManDSer作为车辆管理逻辑的数据层接口，WageManDSer作为薪水管理逻辑的数据层接口，WorkManDSer作为人员管理逻辑的数据层接口。DriverPO,OrgPO,VanPO,WagePO,WorkPO分别作为司机、机构、车辆、薪水和人员的持久化对象。CityPO含有城市信息。

WorkOrgManData模块的设计如图所示。



workOrgManData模块各个类的职责如表所示

workOrgManData模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| DriverManData | 负责司机管理逻辑所需要的数据服务 |
| OrgManData | 负责机构管理逻辑所需要的数据服务 |
| VanManData | 负责车辆管理逻辑所需要的数据服务 |
| WageManData | 负责薪水管理逻辑所需要的数据服务 |
| WorkManData | 负责人员管理逻辑所需要的数据服务 |

1. 模块内部类的接口规范

DriverManData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DriverManData.addDriver | 语法 | public boolean addDriver(DriverPO Driver) |
| 前置条件 | 已输入完整的司机信息，确认新增 |
| 后置条件 | 将司机信息添加到数据库，返回是否成功 |
| DriverManData.deleteDriver | 语法 | public boolean deleteDriver(String id) |
| 前置条件 | 选择司机，确认删除 |
| 后置条件 | 从数据库中删除相应司机的信息，返回是否成功 |
| DriverManData.reviseDriver | 语法 | public boolean reviseDriver(DriverPO Driver) |
| 前置条件 | 确认修改司机信息 |
| 后置条件 | 修改数据库中相应信息，，返回是否成功 |
| DriverManData.checkDriver(String id) | 语法 | public DriverPO checkDriver(String id) |
| 前置条件 | 从司机列表中选择司机选择查看 |
| 后置条件 | 从数据库获得相应司机的信息返回 |
| DriverManData.findDriver(String keyword) | 语法 | public ArrayList<DriverPO> searchDriver(String keyword) |
| 前置条件 | 打开司机管理界面或输入关键词检索 |
| 后置条件 | 从数据库获取司机列表或含有相应关键词的司机列表返回 |

OrgManData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrgManData.addOrg(OrgPO org) | 语法 | public boolean addOrg(OrgPO org) |
| 前置条件 | 已输入完整的机构信息，确认新增 |
| 后置条件 | 将PO中的机构信息添加进数据库 |
| OrgManData.deleteOrg(String id) | 语法 | public boolean deleteOrg(String id) |
| 前置条件 | 选择机构，确认删除 |
| 后置条件 | 将数据库中相应ID的机构信息删除 |
| OrgManData.reviseOrg(OrgPO org) | 语法 | public boolean reviseOrg(OrgPO org) |
| 前置条件 | 确认修改机构信息 |
| 后置条件 | 在数据库中修改PO中信息相应的机构信息 |
| OrgManData.findOrg(String keyword) | 语法 | public ArrayList<OrgPO> findOrg(String keyword) |
| 前置条件 | 打开机构管理界面或输入关键词检索 |
| 后置条件 | 在数据库中获得含有相应关键词（若关键词为空则返回全部）的机构信息列表返回。 |
| OrgManData.getCities() | 语法 | public ArrayList<CityPO> getCities() |
| 前置条件 | 打开机构管理界面 |
| 后置条件 | 获得城市信息列表返回 |

VanManData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| VanManData.addVan(VanPO Van) | 语法 | public boolean addVan(VanPO Van) |
| 前置条件 | 已输入完整的车辆信息，确认新增 |
| 后置条件 | 将PO中的车辆信息添加进数据库 |
| VanManData.deleteVan(String id) | 语法 | public boolean deleteVan(String id) |
| 前置条件 | 选择车辆，确认删除 |
| 后置条件 | 将数据库中相应ID的车辆信息删除 |
| VanManData.reviseVan(VanPO Van) | 语法 | public boolean reviseVan(VanPO Van) |
| 前置条件 | 确认修改车辆信息 |
| 后置条件 | 在数据库中修改PO中信息相应的车辆信息 |
| VanManData.checkVan(String id) | 语法 | public VanPO checkVan(String id) |
| 前置条件 | 从车辆列表中选择车辆选择查看 |
| 后置条件 | 在数据库中获得相应ID车辆的信息并返回 |
| VanManData.findVan(String keyword) | 语法 | public ArrayList<VanPO> searchVan(String keyword) |
| 前置条件 | 打开车辆管理界面或输入关键词检索 |
| 后置条件 | 在数据库中获得含有相应关键词（若关键词为空则返回全部）的车辆信息列表返回。 |

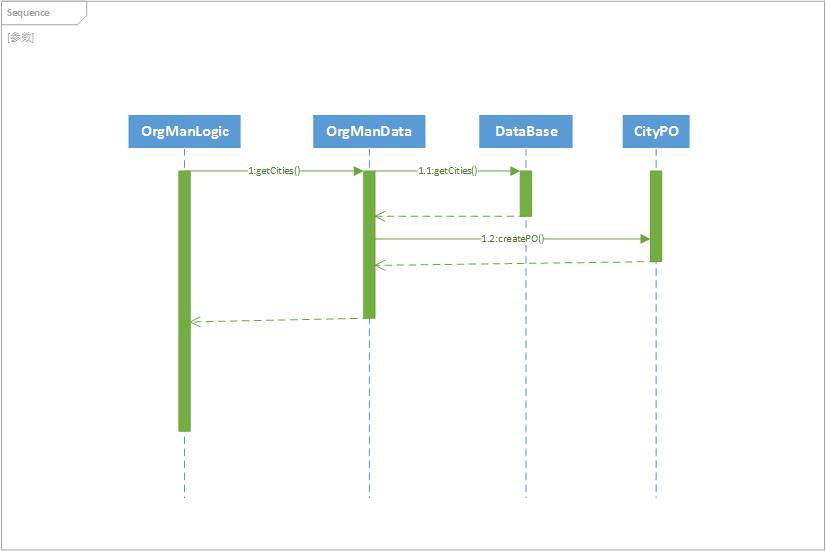
WageManLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| WageManLogic.searchWorker(String keyword) | 语法 | public ResultMessage searchWorker(String keyword) |
| 前置条件 | 打开薪水管理界面 |
| 后置条件 | 从数据库获得人员薪水列表返回 |
| WageManLogic.reviseWage(WageVO wage) | 语法 | public boolean reviseWage(WageVO wage) |
| 前置条件 | 确认薪水策略制定 |
| 后置条件 | 将薪水策略发送到数据层并修改数据库中相应人员的薪水策略 |
| 需要的服务（需接口） | | |
| WageManDSer.findWork(String keyword) | 在数据库中获得含有相应关键词（若关键词为空则返回全部）的人员薪水信息列表返回。 | |
| WageManDSer.reviseVan(WagePO Van) | 在数据库中修改PO中相应人员的薪水策略信息 | |

WorkManData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| WorkManData.addWork(WorkPO work) | 语法 | public boolean addWork(WorkPO Work) |
| 前置条件 | 已输入完整的人员信息，确认新增 |
| 后置条件 | 将PO中的人员信息添加进数据库 |
| WorkManData.deleteWork(String id) | 语法 | public boolean deleteWork(String id) |
| 前置条件 | 选择人员，确认删除 |
| 后置条件 | 将数据库中相应ID的人员信息删除 |
| WorkManData.reviseWork(WorkPO work) | 语法 | public boolean reviseWork(WorkPO Work) |
| 前置条件 | 确认修改人员信息 |
| 后置条件 | 在数据库中修改PO中信息相应的人员信息 |
| WorkManData.findWork(String keyword) | 语法 | public ArrayList<WorkPO> findWork(String keyword) |
| 前置条件 | 打开机构管理界面或输入关键词检索 |
| 后置条件 | 在数据库中获得含有相应关键词（若关键词为空则返回全部）的人员信息列表返回。 |
| WorkManData.getOrgs() | 语法 | public ArrayList<OrgPO> getOrgs() |
| 前置条件 | 打开人员管理界面 |
| 后置条件 | 在数据库中获得机构信息列表并返回 |

1. 数据层的动态模型

下图表明了在打开机构管理界面后，获得城市信息列表时相关对象之间的协作。

1. 数据层的设计原理

见4.2.1.（5）

* + 1. CityData模块

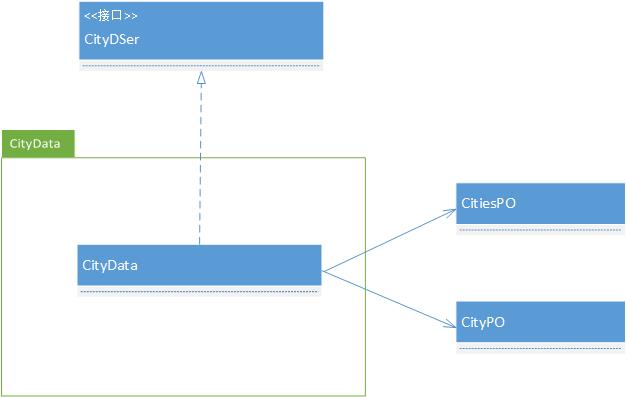
（1）模块概述

CityData模块承担的需求参见需求规格说明文档需求及相关非功能需求。CityData模块的职责及接口参见软件体系结构描述文档5.3.2中CityData模块的接口规范。

（2）整体结构

根据体系结构的设计，分层结构分为界面层，逻辑层和数据层。逻辑层和数据层之间添加CityDSer作为城市管理逻辑的数据层接口。CityPO作为城市的持久化对象。

CityData模块的设计如图所示



|  |  |
| --- | --- |
| 模块 | 职责 |
| CityData | 负责实现城市管理逻辑所需要的数据服务 |

（3）模块内部类的接口规范

cityData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CityData.add | 语法 | public boolean add(CityPO city) throws RemoteException |
| 前置条件 | 该city在数据库中不存在 |
| 后置条件 | 在数据库中增加一个city记录 |
| CityData.revise | 语法 | Public Boolean revise(CityPO city) throws RemoteException |
| 前置条件 | 该city在数据库中已存在 |
| 后置条件 | 在数据库中更新该city |
| CityData.reviseCities | 语法 | Public Boolean reviseCities(CityPO cities) throws RemoteException |
| 前置条件 | 该city在数据库中已存在 |
| 后置条件 | 改变城市之间距离和价格 |
| CityData.getCities | 语法 | Public ArrayList<CityPO> getCities() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回城市对象列表 |

（4）数据层的动态模型

新建城市顺序图，表明了快递物流系统中，当用户新增城市时，城市管理数据服务处理的相关对象之间的协作。



（5）数据层的设计原理

见4.2.1.（5）

* + 1. accountData模块

1. 模块概述

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

accountData模块的职责及接口参见软件体系结构描述文档5.3.2中accountData模块的接口规范。

1. 整体结构

根据体系结构的设计，分层结构分为界面层，逻辑层和数据层。界面层和逻辑层之间添加accountLogicSer作为账户管理界面的逻辑层接口。数据层和逻辑层之间添加accountDSer作为账户逻辑的数据层接口。AccountDataSerImpl是基于Mysql数据库的持久化数据接口，实现AccountDataSer的服务。accountPO作为用户的持久化对象。

accountData模块设计如图所示。



accountData模块各个类的职责如表所示。

accountData各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| accountData | 负责账户管理数据处理 |
| countData | 负责实现账管理的数据处理 |

1. 模块内部类的接口规范

AccountData的接口规范

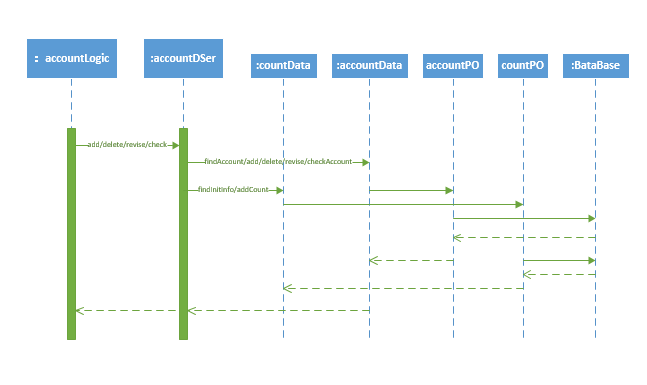
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| AccountData.findAccount | 语法 | public ArrayList<AccountPO> findAccount(String keyword) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按关键字查找返回符合条件的账户列表 |
| AccountData.addAccount | 语法 | public boolean addAccount(AccountPO account) throws RemoteException |
| 前置条件 | 同样账户名的账户在数据库中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| AccountData.deleteAccount | 语法 | public boolean deleteAccount(String id) throws RemoteException |
| 前置条件 | 该id对应的账户在数据库中存在 |
| 后置条件 | 在数据库中删除一个po记录 |
| AccountData.reviseAccount | 语法 | public boolean reviseAccount(AccountPO po) throws RemoteException |
| 前置条件 | 该账户在数据库中存在 |
| 后置条件 | 在数据库中更新该账户信息 |

CountData的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CountData.findInitInfo | 语法 | public AcountPO findInitInfo() throws RemoteException | | |
| 前置条件 | ID对应的期初信息存在 | | |
| 后置条件 | 返回该id对应的期初信息 | | |
| CountData.addCount | 语法 | public boolean addCount(CountPO count) throws RemoteException | | |
| 前置条件 | 无 | | |
| 后置条件 | 在数据库中增加一个po记录 | | |
| CountData.findOrg | | | 语法 | public ArrayList<OrgPO> findOrg() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 在数据库中获取并返回机构信息 |
| CountData.getWorkers | | | 语法 | Public ArrayList<WorkPO> getWorkers()throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 在数据库中获取并返回人员信息 |
| CountData.getVans | | | 语法 | public ArrayList<VanPO> getVans() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 在数据库中获取并返回车辆信息 |
| CountData.getStock | | | 语法 | Public ArrayList<StockPO> getStock()throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 在数据库中获取并返回库存信息 |
| CountData.getAccount | | 语法 | public ArrayList<AccountPO> getAccount() throws RemoteException | |
| 前置条件 | 无 | |
| 后置条件 | 在数据库中获取并返回账户信息 | |

1. 数据层的动态模型

账户管理顺序图，表明了账户管理过程相关数据对象之间的协作。



1. 数据层的设计原理

见4.2.1.（5）

* + 1. checkData模块

（1）模块概述

CheckData模块承担的需求参见需求规格说明文档需求及相关非功能需求。CheckData模块的职责及接口参见软件体系结构描述文档5.3.2中CheckData模块的接口规范。

（2）整体结构

根据体系结构的设计，分层结构分为界面层，逻辑层和数据层。逻辑层和数据层之间添加CostBeneDSer作为成本收益表查看和导出界面的数据层接口，BusiCircumDSer作为经营情况表查看和导出界面的数据层接口，CollectFromDSer作为收款记录查看界面的数据层接口，DiaryDSer作为日志记录查看界面的数据层接口。IncomePO作为入款单的持久化对象，PaymentPO作为付款单的持久化对象，DiaryPO作为日志记录的持久化对象。

CheckData模块的设计如图所示



CheckData各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| CostBeneData | 负责实现成本收益表查看和导出逻辑所需要的数据服务 |
| BusiCircumData | 负责实现经营情况表查看和导出逻辑所需要的数据服务 |
| DiaryData | 负责实现操作日志查看逻辑所需要的数据服务 |
| CollectRecordData | 负责实现收款单记录查看逻辑所需要的数据服务 |

（3）模块内部类的接口规范

CostBeneData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CostBeneData. getIncome | 语法 | public ArrayList<IncomePO> getIncome () |
| 前置条件 | 无 |
| 后置条件 | 返回入款单 |
| CostBeneData. getPayment | 语法 | public ArrayList<PaymentPO> getPayment () |
| 前置条件 | 无 |
| 后置条件 | 返回付款单 |

BusiCircumData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| BusiCircumData.findCollect | 语法 | public ArrayList<IncomePO> findCollect(Date start,Date end) throw RemoteException |
| 前置条件 | 无 |
| 后置条件 | 根据输入的开始和结束日期从数据库收款单记录，返回相应的数据 |
| BusiCircumData.getHall | 语法 | public ArrayList<HallPO> getHall() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 从数据库中获取并返回所有的营业厅编号和名称 |

DiaryData模块的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| DiaryData.seeDiary | 语法 | public ArrayList<DiaryPO> seeDiary(Date start,Date end) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 根据输入的开始和结束日期从数据库中获得日志记录，返回相应的数据 |

CollectRecordData模块的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| CollectRecordData.getCollectRecord | 语法 | public ArrayList<IncomePO> getCollectRec(Date date,String id) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 根据时间和营业厅名，从数据库中获取并返回该营业厅的所有收款单记录 |
| CollectRecordData.getHall() | 语法 | public ArrayList<OrgPO> getHall() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 从数据库中获取并返回所有的营业厅编号和名称 |

（4）数据层的动态模型

查看收款记录顺序图，表明了快递物流系统中，当用户查看收款记录时，收款记录查看服务逻辑处理的相关对象之间的协作。

（5）数据层的设计原理

见4.2.1.（5）

* + 1. orderApproveData模块

1. 模块概述

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

orderApproveData模块的职责及接口参见软件体系结构描述文档5.3.2中orderApproveData模块的接口规范。

1. 整体结构

根据体系结构的设计，分层结构分为界面层，逻辑层和数据层。界面层和逻辑层之间添加orderApproveLogicSer作为账户管理界面的逻辑层接口。数据层和逻辑层之间添加orderApproveDSer作为账户管理逻辑的数据层接口。orderApproveDataSerImpl是基于Mysql数据库的持久化数据接口，实现OrderApproveDataSer的服务。orderPO作为用户的持久化对象。

orderApproveData模块各个类的职责如表所示。

orderApproveData各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| OrderApproveData | 负责单据审批数据处理 |
| OrderResubmitData | 负责实现单据重新提交的数据处理 |

orderApproveData模块设计如图所示。



1. 模块内部类的接口规范

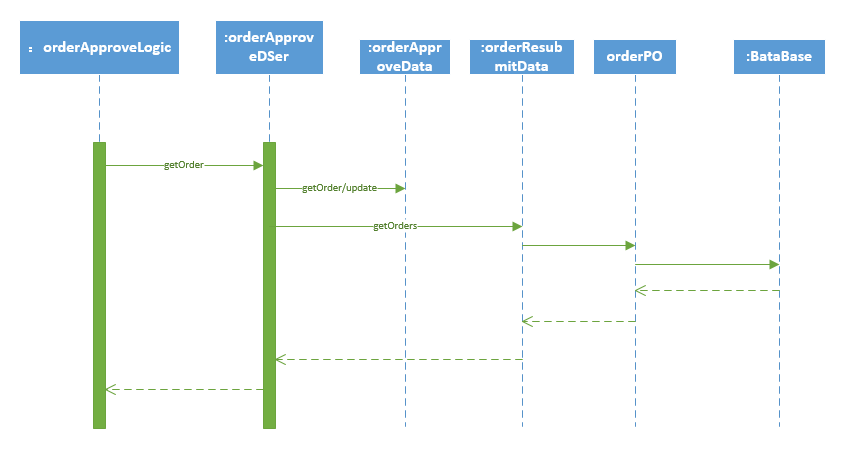
OrderApproveData模块接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderApproveData.update | 语法 | public boolean update(boolean isPassed,ArrayList<String> id) throws RemoteException |
| 前置条件 | Order存在于数据库中 |
| 后置条件 | 在数据库中更新订单审批状态 |
| OrderApproveData.getOrder | 语法 | public OrderPO getOrder(String id) throws RemoteException |
| 前置条件 | 相应ID的单据存在于数据库 |
| 后置条件 | 从数据库获得并返回相应的单据信息 |
| OrderApproveData.getExamine | 语法 | public ArrayList<OrderPO> getExamine (String id) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 从数据库获得并返回相应的单据审批信息 |

OrderResubmitData的接口规范

|  |  |  |
| --- | --- | --- |
| OrderResubmitData.getOrders | 语法 | public ArrayList<Object> getOrder(String id) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 从数据库获得ID相应机构的未通过单据列表 |

1. 数据层的动态模型

单据审批顺序图，表明了单据审批过程相关数据对象之间的协作。

1. 数据层的设计原理

见4.2.1.（5）

1. 依赖视角