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 5](#_Toc437195807)

[1.1 编制目的 5](#_Toc437195808)

[1.2 词汇表 5](#_Toc437195809)

[1.3 参考资料 5](#_Toc437195810)

[2. 产品概述 5](#_Toc437195811)

[3. 体系结构设计概述 5](#_Toc437195812)

[4. 结构视角 5](#_Toc437195813)

[4.1 业务逻辑层分解 5](#_Toc437195814)

[4.1.1 userLogic模块 5](#_Toc437195815)

[（1） 模块概述 5](#_Toc437195816)

[（2） 整体结构 5](#_Toc437195817)

[（3） 模块内部类的接口规范 6](#_Toc437195818)

[（4） 业务逻辑层的动态模型 7](#_Toc437195819)

[（5） 业务逻辑层的设计原理 7](#_Toc437195820)

[4.1.2 transitInfoLogic模块 8](#_Toc437195821)

[（1） 模块概述 8](#_Toc437195822)

[（2） 整体结构 8](#_Toc437195823)

[（3） 模块内部类的接口规范 8](#_Toc437195824)

[（4） 业务逻辑层的动态模型 9](#_Toc437195825)

[（5） 业务逻辑层的设计原理 9](#_Toc437195826)

[4.1.3 orderNewLogic模块 9](#_Toc437195827)

[（1） 模块概述 9](#_Toc437195828)

[（2） 整体结构 9](#_Toc437195829)

[（3） 模块内部类的接口规范 11](#_Toc437195830)

[（4） 业务逻辑层的动态模型 15](#_Toc437195831)

[（5） 业务逻辑层的设计原理 15](#_Toc437195832)

[4.1.4 workOrgManLogic模块 15](#_Toc437195833)

[（1） 模块概述 15](#_Toc437195834)

[（2） 整体结构 15](#_Toc437195835)

[（3） 模块内部类的接口规范 16](#_Toc437195836)

[（4） 业务逻辑层的动态模型 20](#_Toc437195837)

[（5） 业务逻辑层的设计原理 20](#_Toc437195838)

[4.1.5 cityLogic模块 20](#_Toc437195839)

[（1）模块概述 20](#_Toc437195840)

[（2）整体结构 20](#_Toc437195841)

[（3）模块内部类的接口规范 21](#_Toc437195842)

[（4）业务逻辑层的动态模型 22](#_Toc437195843)

[（5）业务逻辑层的设计原理 23](#_Toc437195844)

[4.1.6 accountLogic模块 23](#_Toc437195845)

[（1） 模块概述 23](#_Toc437195846)

[（2） 整体结构 23](#_Toc437195847)

[（3） 模块内部类的接口规范 24](#_Toc437195848)

[（4） 业务逻辑层的动态模型 25](#_Toc437195849)

[（5） 业务逻辑层的设计原理 26](#_Toc437195850)

[4.1.7 stockLogic模块 26](#_Toc437195851)

[（1） 模块概述 26](#_Toc437195852)

[（2） 整体结构 26](#_Toc437195853)

[（3） 模块内部类的接口规范 27](#_Toc437195854)

[（4） 业务逻辑层的动态模型 29](#_Toc437195855)

[如图为执行库存盘点操作后，相关类的协作顺序图。 29](#_Toc437195856)

[（5） 业务逻辑层的设计原理 30](#_Toc437195857)

[4.1.8 checkLogic模块 30](#_Toc437195858)

[（1）模块概述 30](#_Toc437195859)

[（2）整体结构 31](#_Toc437195860)

[（3）模块内部类的接口规范 31](#_Toc437195861)

[（4）业务逻辑层的动态模型 32](#_Toc437195862)

[（5）业务逻辑层的设计原理 33](#_Toc437195863)

[4.1.9 orderApproveLogic模块 33](#_Toc437195864)

[（1） 模块概述 33](#_Toc437195865)

[（2） 整体结构 33](#_Toc437195866)

[（3） 模块内部类的接口规范 34](#_Toc437195867)

[（4） 业务逻辑层的动态模型 35](#_Toc437195868)

[（5） 业务逻辑层的设计原理 36](#_Toc437195869)

[4.2 数据层分解 36](#_Toc437195870)

[4.2.1 userData模块 36](#_Toc437195871)

[（1） 模块概述 36](#_Toc437195872)

[（2） 整体结构 36](#_Toc437195873)

[（3） 模块内部类的接口规范 37](#_Toc437195874)

[（4） 数据层的动态模型 37](#_Toc437195875)

[（5） 数据层的设计原理 38](#_Toc437195876)

[4.2.2 transitInfoData模块 38](#_Toc437195877)

[（1） 模块概述 38](#_Toc437195878)

[（2） 整体结构 38](#_Toc437195879)

[（3） 模块内部类的接口规范 39](#_Toc437195880)

[（4） 数据层的动态模型 39](#_Toc437195881)

[（5） 数据层的设计原理 40](#_Toc437195882)

[4.2.3 orderNewData模块 40](#_Toc437195883)

[（1）模块概述 40](#_Toc437195884)

[（2）整体结构 40](#_Toc437195885)

[（3）模块内部类的接口规范 41](#_Toc437195886)

[（4）数据层的动态模型 44](#_Toc437195887)

[（5）数据层的设计原理 44](#_Toc437195888)

[4.2.4 workOrgManData模块 44](#_Toc437195889)

[（1） 模块概述 44](#_Toc437195890)

[（2） 整体结构 45](#_Toc437195891)

[（3） 模块内部类的接口规范 45](#_Toc437195892)

[（4） 数据层的动态模型 48](#_Toc437195893)

[（5） 数据层的设计原理 48](#_Toc437195894)

[4.2.5 cityData模块 48](#_Toc437195895)

[（1）模块概述 48](#_Toc437195896)

[（2）整体结构 48](#_Toc437195897)

[（3）模块内部类的接口规范 49](#_Toc437195898)

[（4）数据层的动态模型 50](#_Toc437195899)

[（5）数据层的设计原理 50](#_Toc437195900)

[4.2.6 accountData模块 50](#_Toc437195901)

[（1） 模块概述 50](#_Toc437195902)

[（2） 整体结构 50](#_Toc437195903)

[（3） 模块内部类的接口规范 51](#_Toc437195904)

[（4） 数据层的动态模型 52](#_Toc437195905)

[（5） 数据层的设计原理 52](#_Toc437195906)

[4.2.7 stockData模块 52](#_Toc437195907)

[（1） 模块概述 52](#_Toc437195908)

[（2） 整体结构 52](#_Toc437195909)

[（3） 模块内部类的接口规范 53](#_Toc437195910)

[（4） 数据层的动态模型 55](#_Toc437195911)

[（5） 数据层的设计原理 55](#_Toc437195912)

[4.2.8 checkData模块 55](#_Toc437195913)

[（1）模块概述 55](#_Toc437195914)

[（2）整体结构 56](#_Toc437195915)

[（3）模块内部类的接口规范 56](#_Toc437195916)

[（4）数据层的动态模型 57](#_Toc437195917)

[（5）数据层的设计原理 57](#_Toc437195918)

[4.2.9 orderApproveData模块 58](#_Toc437195919)

[（1） 模块概述 58](#_Toc437195920)

[（2） 整体结构 58](#_Toc437195921)

[（3） 模块内部类的接口规范 58](#_Toc437195922)

[（4） 数据层的动态模型 59](#_Toc437195923)

[（5） 数据层的设计原理 59](#_Toc437195924)

[4.3 界面层分解 59](#_Toc437195925)

[4.3.1 orderNewUI模块 59](#_Toc437195926)

[（1） 模块概述 59](#_Toc437195927)

[（2） 整体结构 60](#_Toc437195928)

[（3） 模块内部接口规范 61](#_Toc437195929)

[（4） 界面层的动态模型 62](#_Toc437195930)

[（5） 界面层的设计原理 63](#_Toc437195931)

[4.3.2 transitInfoUI模块 63](#_Toc437195932)

[（1） 模块概述 63](#_Toc437195933)

[（2） 整体结构 63](#_Toc437195934)

[（3） 模块内部接口规范 64](#_Toc437195935)

[（4） 界面层的动态模型 64](#_Toc437195936)

[（5） 界面层的设计原理 65](#_Toc437195937)

[4.3.3 workOrgManUI模块 65](#_Toc437195938)

[（1） 模块描述 65](#_Toc437195939)

[（2） 整体结构 65](#_Toc437195940)

[（3） 模块内部类的接口规范 66](#_Toc437195941)

[（4） 界面层的动态模型 68](#_Toc437195942)

[（5） 界面层的设计原理 69](#_Toc437195943)

[4.3.4 cityUI模块 69](#_Toc437195944)

[（1） 模块描述 69](#_Toc437195945)

[（2） 整体结构 69](#_Toc437195946)

[（3） 模块内部的接口规范 70](#_Toc437195947)

[CityPanel的接口规范 70](#_Toc437195948)

[ConstantPanel的接口规范 70](#_Toc437195949)

[4.3.5 accountUI模块 71](#_Toc437195950)

[（1） 模块描述 71](#_Toc437195951)

[（2） 整体结构 71](#_Toc437195952)

[（3） 模块内部类的接口规范 72](#_Toc437195953)

[（4） 界面层的动态模型 72](#_Toc437195954)

[（5） 界面层的设计原理 73](#_Toc437195955)

[4.3.6 stockUI模块 73](#_Toc437195956)

[（1） 模块概述 73](#_Toc437195957)

[（2） 整体结构 73](#_Toc437195958)

[（3） 模块内部接口规范 74](#_Toc437195959)

[（4） 界面层的动态模型 76](#_Toc437195960)

[（5） 界面层的设计原理 77](#_Toc437195961)

[4.3.7 checkUI模块 78](#_Toc437195962)

[（1） 模块概述 78](#_Toc437195963)

[（2） 整体结构 78](#_Toc437195964)

[（3） 模块内部类的接口规范 79](#_Toc437195965)

[（4） 界面层的动态模型 80](#_Toc437195966)

[（5） 界面层的设计原理 80](#_Toc437195967)

[4.3.8 orderApproveUI模块 81](#_Toc437195968)

[（1） 模块描述 81](#_Toc437195969)

[（2） 整体结构 81](#_Toc437195970)

[（3） 模块内部类的接口规范 81](#_Toc437195971)

[（4） 界面层的动态模型 83](#_Toc437195972)

[（5） 界面层的设计原理 83](#_Toc437195973)

[4.3.9 UserPanel模块 83](#_Toc437195974)

[（1） 模块概述 83](#_Toc437195975)

[（2） 整体结构 84](#_Toc437195976)

[（3） 模块内部类的接口规范 84](#_Toc437195977)

[（4） 界面的动态模型 85](#_Toc437195978)

[5. 依赖视角 86](#_Toc437195979)

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 | 将账号和密码传给数据层，返回登陆是否成功 |
| 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 | 将新增用户信息传给数据层，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 | 将要删除的用户告知数据层，返回是否成功删除 |
| UserManLogic.reviseUser | 语法 | public boolean reviseUser(UserVO user，int index) |
| 前置条件 | 已输入并确认用户信息，参数已传入 |
| 后置条件 | 将修改后的用户信息传到数据层，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。

1. 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 | |

1. 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） | | |
| CustomerDSer.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

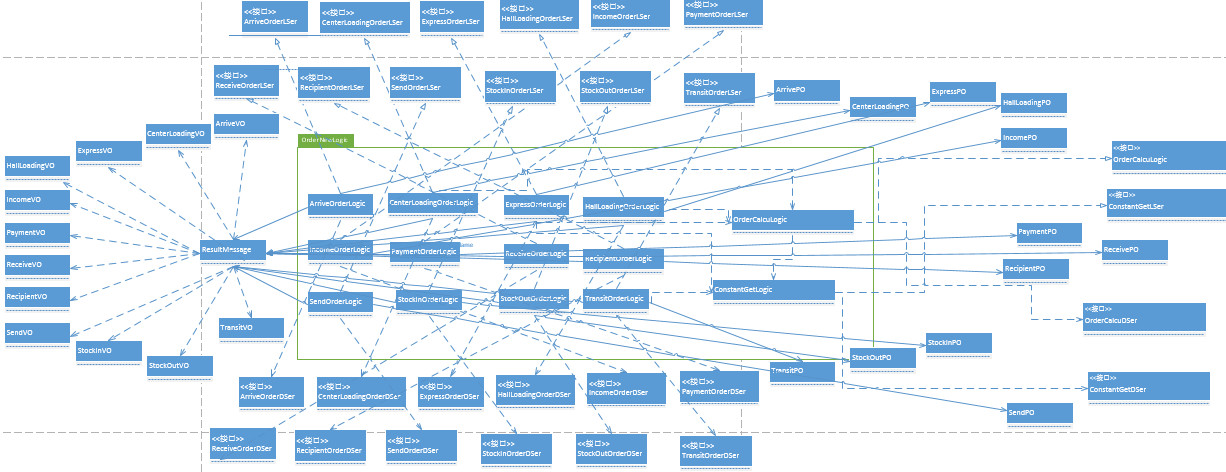
1. 模块概述

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 |
| ConstantGetLogic | 负责获取价格常量的业务逻辑 |
| OrderCalcuLogic | 负责计算费用的业务逻辑 |

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 Result 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 | 用户选择城市输入 |
| 后置条件 | 在数据库中查找所有城市，并返回城市名称列表 |
| ArriveOrderLogic.getLocalHallsAndAllCenter | 语法 | Public ResultMessage getLocalHallsAndAllCenter () |
| 前置条件 | 用户选择到达地输入 |
| 后置条件 | 在数据库中查找所有营业厅和中转中心，并返回名称列表 |

|  |  |
| --- | --- |
| 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 |
| UtilityLogic.getLocalHallAndAllCenter(String orgId) | 在数据库中获取并返回营业厅和中转中心信息 |

CenterLoadingOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CenterLoadingOrderLogic.create | 语法 | public Result 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 | 用户选择到达地（营业厅）输入 |
| 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 |
| CenterLoadingOrderLogic.getPrice | 语法 | public ResultMessage getPrice(String org1,String org2,List<String>expressList) |
| 前置条件 | 用户选择到达地（营业厅）输入 |
| 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 |
| UtilityLogic. getHall() | 通过utilitylogic提供的方法在数据库中获取并返回营业厅信息 |
| UtilityLogic. getPrice() | 通过utilitylogic提供的方法在数据库中获取并计算返回运费信息 |
| UpdateTransitInfoLogic.update | 通过UpdateTransitInfoLogic的静态方法，将信息更新至数据库 |

ExpressOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ExpressOrderLogic.create | 语法 | public Result 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 |
| ExpressOrderLogic.getPrice | 语法 | public String getPrice (String city1,String city2,int expressKind,int pakKind,double weight) |
| 前置条件 | 用户输入城市，界面要求获取运费 |
| 后置条件 | 从数据库中获取城市距离并计算返回运费 |
| ExpressOrderLogic.getTime | 语法 | public String getTime(String city1,String city2) |
| 前置条件 | 寄件单的各项信息已输入并确认 |
| 后置条件 | 从数据库中获取城市距离等并计算返回运输时间 |

|  |  |
| --- | --- |
| 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 |
| UtilityLogic.getDistance(String city1，String city2) | 在数据库中获取并返回城市距离 |
| UtilityLogic.getExpressCost() | 从数据库中获得订单运费 |

HallLoadingOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HallLoadingOrderLogic.create | 语法 | public Result 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 | 用户选择机构输入 |
| 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 | 用户选择车辆代号输入 |
| 后置条件 | 查找数据库中所有车辆，返回车辆代号列表 |
| HallLoadingOrderLogic.getPrice | 语法 | public ResultMessage getPrice(String org1,String org2) |
| 前置条件 | 用户选择两个机构 |
| 后置条件 | 查找数据库中机构信息，计算返回运费 |
| HallLoadingOrderLogic.getLocalHallsAndAllCenter | 语法 | public ResultMessage getLocalHallsAndAllCenter (String orgId) |
| 前置条件 | 用户选择到达地 |
| 后置条件 | 查找数据库中到达地信息，并返回 |

|  |  |
| --- | --- |
| 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 |
| UtilityLogic.getLocalHallsAndAllCenter (String id) | 在数据库中获取并返回营业厅和中转中心信息 |

IncomeOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| IncomeOrderLogic.create | 语法 | public Result 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 | 选择快递员输入并输入营业厅id |
| 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 Result 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 |
| PaymentOrderLogic.getAccount | 语法 | public ResultMessage getAccount() |
| 前置条件 | 付款单的各项信息已输入并确认 |
| 后置条件 | 从数据层获取账户信息并返回 |

|  |  |
| --- | --- |
| 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 |
| UtilityLogic.getAccount() | 在数据库中获取账户信息 |

ReceiveOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ReceiveOrderLogic.create | 语法 | public Result 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(ExpressPO order) | 在数据库中修改完善ExpressPO对象 |
| CourierDSer.find(String id) | 在数据库中获取相应编号订单，并返回ExpressPO |

RecipientOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| RecipientOrderLogic.Create | 语法 | public ResultCreate(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 |
| RecipientOrderLogic.isTransitValid | 语法 | public boolean isTransitValid () |
| 前置条件 | 用户输入中转单编号 |
| 后置条件 | 在数据库中查找所有中转单，并返回是否存在 |
| RecipientOrderLogic.expressList | 语法 | public List<String> expressList (String id) |
| 前置条件 | 用户需要获取订单列表 |
| 后置条件 | 在数据库中查找所有订单编号并返回订单列表 |

|  |  |
| --- | --- |
| 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 |
| UtilityLogic.getCities() | 在数据库中查找并返回所有城市名称 |
| UtilityLogic. isTransitValid(String id) | 在数据库中查找所有中转单，并返回是否存在 |

SendOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| SendOrderLogic.Create | 语法 | public Result Create(SendVO 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 |
| UtilityDSer.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 Result 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 | 用户选择城市输入 |
| 后置条件 | 在数据库中查找所有城市，并返回城市名称列表 |
| StockInOrderLogic.checkWarning | 语法 | public ResultMessage checkWarning (String id) |
| 前置条件 | 用户选择库存报警 |
| 后置条件 | 在数据库中查找警戒线并返回 |
| StockInOrderLogic.getShelves | 语法 | public ResultMessage getShelves (String orgId) |
| 前置条件 | 用户选择城市输入架信息 |
| 后置条件 | 在数据库中查找所有架，并返回名称列表 |
| StockInOrderLogic.isExpressValid | 语法 | public boolean isExpressValid (String id) |
| 前置条件 | 用户选择输入订单编号 |
| 后置条件 | 在数据库中查找所有订单编号，并返回是否存在 |

|  |  |
| --- | --- |
| 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 |
| UtilityDSer.getCities() | 在数据库中查找并返回所有城市 |
| StockCheckWarnLSer.checkWarning() | 在数据库中查找并返回警戒线信息 |
| StockInOrderDSer.getShelves(String orgId) | 在数据库中查找并返回库存中所有架 |
| UtilityLogic. isExpressValid(String id) | 在数据库中查找所有订单编号，并返回是否存在 |

StockOutOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockOutOrderLogic.create | 语法 | public Result 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 | 用户选择城市输入 |
| 后置条件 | 在数据库中查找所有城市，并返回城市名称列表 |
| StockOutOrderLogic.isExpressValid | 语法 | public boolean isExpressValid (String id) |
| 前置条件 | 用户输入订单编号 |
| 后置条件 | 从数据层查找并返回订单编号是否存在 |
| StockOutOrderLogic.isTransitValid | 语法 | public boolean isTransitValid(String id) |
| 前置条件 | 用户输入中转单 |
| 后置条件 | 在数据库中查找并返回中转单是否存在 |
| StockOutOrderLogic.isConveyValid | 语法 | public boolean isConveyValid(String id) |
| 前置条件 | 用户选择输入装车单 |
| 后置条件 | 在数据库中查找所有装车单编号，并返回是否存在 |

|  |  |
| --- | --- |
| 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 |
| UtilityLogic. getCities | 在数据库中查找并返回所有城市 |
| UtilityLogic. isTransitValid | 在数据库中查找并返回中转单编号是否存在 |
| UtilityLogic isExpressValid | 在数据库中查找并返回订单编号是否存在 |
| UtilityLogic isConveyValid | 在数据库中查找并返回装车单是否存在 |

TransitOrderLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| TransitOrderLogic.create | 语法 | public Result 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 | 用户选择城市输入 |
| 后置条件 | 在数据库中查找所有城市，并返回城市名称列表 |
| TransitOrderLogic.getCenters | 语法 | public ResultMessage getCenters() |
| 前置条件 | 用户选择中转中心输入 |
| 后置条件 | 在数据库中查找所有中转中心并返回 |
| TransitOrderLogic.isExpressValid | 语法 | public Boolean isExpressValid (List<String>expresslist) |
| 前置条件 | 用户输入订单编号 |
| 后置条件 | 在数据库中查找所有订单编号并确定是否存在 |

|  |  |
| --- | --- |
| 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 |
| UtilityLogic.getCities | 在数据库中查找并返回所有城市 |
| UtilityLogic.getCenters | 在数据库中查找并返回所有中转中心 |
| UtilityLogic. isExpressValid | 在数据库中查找并判断订单编号是否存在 |

ConstantGetLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ConstantGetLogic.getExpressCost | 语法 | Public ResutMessage getExpressCost() |
| 前置条件 | 需要获取快递类型类型对应的价格常量 |
| 后置条件 | 返回所有快递类型价格列表 |
| ConstantGetLogic.getTransitCost | 语法 | Public ResultMessage getTransitCost() |
| 前置条件 | 需要获取装运类型对应的价格常量 |
| 后置条件 | 返回所有装运类型价格列表 |
| ConstantGetLogic.getFullLoad | 语法 | Public ResultMessage getFullLoad() |
| 前置条件 | 需要获取各类装运方式的最大装载量 |
| 后置条件 | 返回所有装运类型的装载量列表 |

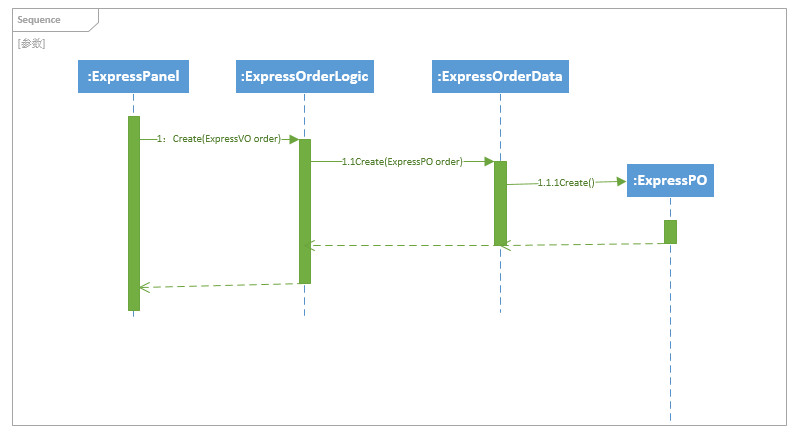
|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| ConstantGetDSer.getConstant() | 从数据层返回各种常量 |

OrderCalcuLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderCalcuLogic.getDistance() | 语法 | Public BigDecimal getDistance(String city1,String city2) |
| 前置条件 | 两城市均存在 |
| 后置条件 | 返回两城市间距离 |
| OrderCalcuLogic.getPrice() | 语法 | Public ResultMessage getPrice(String org1,String org2,Transit transit,List<String> expressList) |
| 前置条件 | 参数输入均正确 |
| 后置条件 | 返回运费或超过满载量信息 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| OrderCalcuDSer.getCitiesPO | 获取两城市间的距离信息 |
| UtilityLSer.orgs | 获取所有机构 |
| ConstantLSer.getTransitCost | 获取装运类型对应的价格常量 |
| ConstantLSer.getFullLoad | 获取装运类型对应的最大载重量 |
| OrderInfoLSer.getExpresses | 获取单据对应的信息 |

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,String name) | 语法 | public boolean addDriver(DriverVO Driver,String name) |
| 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在库中增加相应信息，返回新增是否成功 |
| DriverManLogic.deleteDriver(String id) | 语法 | public boolean deleteDriver(String id,String name) |
| 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，返回删除是否成功 |
| DriverManLogic.reviseDriver(DriverVO Driver) | 语法 | public boolean reviseDriver(DriverVO Driver,String name) |
| 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信息 |
| DriverManLogic.searchDriver(String hallId) | 语法 | public ResultMessage searchDriver (String hallId) |
| 前置条件 | 打开司机管理界面或输入机构编号检索 |
| 后置条件 | 从数据层获取司机列表或含有机构编号对应的司机列表返回 |
| 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.findDriver(String hallId) | 在数据库中获得含有对应机构（若机构编号为空则返回全部）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,String name) | 语法 | public boolean addOrg(OrgVO org,String name) |
| 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，返回新增是否成功 |
| OrgManLogic.deleteOrg(String id,String name) | 语法 | public boolean deleteOrg(String id,String name) |
| 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，返回删除是否成功 |
| OrgManLogic.reviseOrg(OrgVO org,String name) | 语法 | public boolean reviseOrg(OrgVO org,String name) |
| 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，返回修改是否成功 |
| 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,String name) |
| 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，返回新增是否成功 |
| VanManLogic.deleteVan(String id) | 语法 | public boolean deleteVan(String id,String name) |
| 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,返回删除是否成功 |
| VanManLogic.reviseVan(VanVO Van) | 语法 | public boolean reviseVan(VanVO Van, ,String name) |
| 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，返回修改是否成功 |
| VanManLogic.getNextId(String hallId) | 语法 | public ResultMessage getNextId(String hallId) |
| 前置条件 | 营业厅新增车辆 |
| 后置条件 | 返回这个新增车辆应被分配的车辆编号 |
| VanManLogic.searchVan(String keyword,String hallId) | 语法 | public ResultMessage searchVan(String keyword,String hallId) |
| 前置条件 | 打开车辆管理界面或输入关键词与营业厅检索 |
| 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.findVan(String keyword，String hallId) | 在数据库中获得含有相应关键词或营业厅（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.searchWorks(String keyword) | 语法 | public ResultMessage searchWorks(String keyword) |
| 前置条件 | 打开薪水管理界面或输入关键词检索相关人员 |
| 后置条件 | 根据关键词从数据库获得人员薪水列表返回 |
| WageManLogic.setWage(List<WageVO wageList,int[] index,String name) | 语法 | public ResultMessage setWage (List<WageVO wageList,int[] index,String name)) |
| 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，返回修改是否成功 |
| 需要的服务（需接口） | | |
| WageManDSer.searchWork(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.updateWage(WorkPO work) | 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 | 语法 | public ResultMessage n addWork(WorkVO Work,String name)) |
| 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,返回新增是否成功 |
| WorkManLogic.deleteWork | 语法 | public ResultMessage n deleteWork(String id,String name) |
| 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,返回删除是否成功 |
| WorkManLogic.reviseWork | 语法 | public ResultMessage reviseWork(WorkVO Work,int index,String name) |
| 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,返回修改是否成功 |
| WorkManLogic.searchWork | 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(int 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。 | |

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，ConstantLSer作为常量制定的逻辑层接口，LocLSer作为获取地址输入的逻辑层接口。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，ConstantDSer作为常量制定逻辑的数据层接口。CityVO、CitiesVO、ConstantVO作为城市的界面值对象，CityPO、CitiesPO、ConstantPO作为城市、常量的持久化对象。

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 |
| ConstantLogic | 负责实现常量制定界面所需要的业务逻辑 |
| LocLogic | 负责获取城市名称和所属省份的业务逻辑 |

（3）模块内部类的接口规范

cityLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| cityLogic.addCity | 语法 | public Result addCity(CityVO city); |
| 前置条件 | 确认增加一个城市 |
| 后置条件 | 将该城市信息及常量信息加入数据库，返回添加结果 |
| cityLogic.reviseCity | 语法 | public Result reviseCity(CityVO city，int index); |
| 前置条件 | 确认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 Result reviseCities(CitiesVO city); |
| 前置条件 | 总经理已经确认改变两个城市间的距离和价格信息 |
| 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.deleteCity | 语法 | public ResultedeleteCity(String cityName); |
| 前置条件 | 确认删除一个城市 |
| 后置条件 | 在数据库中删除该城市并返回删除是否成功 |
| 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 |
| cityLogic.citiesList | 语法 | public ResultMessage citiesList(); |
| 前置条件 | 需要获取城市间常量对象 |
| 后置条件 | 在数据库中查找所有常量对象并返回 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| cityDSer.add(cityPO po) | 新增城市对象 |
| cityDSer.addCities(citiesPO po) | 新增城市间常量对象 |
| 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.cityList() | 查找并获取所有城市对象列表 |
| cityDSer.deleteCity(String cityName) | 删除城市对象 |
| cityDSer.citiesList() | 查找并获取所有的城市间常量列表 |

constantLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| constantLogic.setConstant | 语法 | public Result setConstant(ConstantVO vo) |
| 前置条件 | 确认制定运输常量 |
| 后置条件 | 将该常量信息加入数据库，返回添加结果 |
| constantLogic.getConstant | 语法 | public ResultMessage getConstant() |
| 前置条件 | 获取运输常量信息 |
| 后置条件 | 在数据库中查找并返回运输常量信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| constantDSer.setConstant(ConstantPO constant) | 设置运输常量信息 | |
| constantDSer.getConstant () | 获取运输常量信息 | |

LocLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| locLogic.getCities() | 语法 | public ResultMessage getCities() |
| 前置条件 | 获取城市名称和城市所属省份 |
| 后置条件 | 在数据库中查找城市对象，返回名称和省份 |
| 需要的服务（需接口） | | |
| 无 | | |

（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 Result addAccount(AccountVO account,String opname) |
| 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,返回新增是否成功 |
| AccountLogic.deleteAccount(int index) | 语法 | public Result deleteAccount(int index,String opName) |
| 前置条件 | 选择账户，确认删除 |
| 后置条件 | 由index在数据库中删除相应账户，返回删除是否成功 |
| AccountLogic.reviseAccount(AccountVO account，int index) | 语法 | public Result reviseAccount(AccountVO account,int index,String opName) |
| 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,返回修改是否成功 |
| 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(int 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 Result newCount(String name) |
| 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.deleteCount(int index) | 语法 | public Result deleteCount(int index,String name) |
| 前置条件 | 用户确认删除某账 |
| 后置条件 | 从数据库删除该账信息并返回删除是否成功 |
| CountLogic.getCount() | 语法 | public ResultMessage getCount() |
| 前置条件 | 用户打开账管理 |
| 后置条件 | 从数据层获得账列表返回给界面 |

|  |  |
| --- | --- |
| 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.getCount() | 获得数据库中的账列表 |
| CountDSer.deleteCount(int id) | 在数据库中删除对应的账信息 |

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(Calendar start, Calendar end, String id) | 语法 | public ResultMessage checkStockIn(Calendar start, Calendar 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(Calendar start, Calendar end, String id) | 语法 | public ResultMessage checkStockOut(Calendar start, Calendar 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(Calendar start, Calendar 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(Calendar start, Calendar 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.outputExcel(String location, String name) | 语法 | public Result 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 | |
| StockCheckNowDSer.getGoods() | 在数据库中获取商品信息列表 | |
| UtilityLSer.outputExcel(String[][] data,String name,String location) | 将信息列表输出为Excel | |

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 | |
| StockPO.getGoods() | 从数据库获得当前库存商品情况 | |

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 Result 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 Result 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 Result 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(ArrayList<Integer> warnings, String id) | 语法 | public Result 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的接口规范

|  |  |  |
| --- | --- | --- |
| CostBeneLogic. getCostBene | 语法 | public ResultMessage getCostBene () |
| 前置条件 | 用户需要获得成本收益表 |
| 后置条件 | 返回成本收益 |
| 提供的服务（供接口） | | |

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

BusiCircumLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| BusiCircumLogic.getBusiCircum | 语法 | public ResultMessage getBusiCircum (Calendar start, Calendar 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.outputExcel | 语法 | public boolean outputExcel(String name, String location) |
| 前置条件 | 用户选择导出经营情况表 |
| 后置条件 | 导出相应名称的经营情况表到相应地址 |
| 提供的服务（供接口） | | |

|  |  |
| --- | --- |
| BusiCircumDSer.findIncome(Calendar start, Calendar end) | 根据开始和结束日期从数据库中获得收款单列表，返回数据 |
| BusiCircumDSer.findPaymentl(Calendar start, Calendar end) | 根据开始和结束日期从数据库中获得付款单列表，返回数据 |

DiaryLogic的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| DiaryLogic.seeDiary | 语法 | public ResultMessage seeDiary(Calendar start, Calendar 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(Calendar start, Calendar 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的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| CollectRecordLogic.seeIncomeList | 语法 | public ResultMessage seeIncomeList(Calendar 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 (Calendar 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 |

（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 ResultMessage examine(boolean approve, ArrayList<Integer > indexs) |
| 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,Order kind) | 语法 | public ResultMessage chooseOrder(String id,Order kind) |
| 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,Order kind) | 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,Order kind) | 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.getUnpassed (String id) | 语法 | public ResultMessage get Unpassed (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未通过单据的数量 |
| OrderResubmitLogic.getOrdersDisplay() | 语法 | public ResultMessage getOrdersDisplay() |
| 前置条件 | 用户查看未通过审批的单据 |
| 后置条件 | 显示未通过的单据列表 |
| OrderResubmitLogic.getOrder(int index) | 语法 | public ResultMessage getOrder(int index) |
| 前置条件 | 用户查看未通过单据具体信息 |
| 后置条件 | 显示相应单据具体信息，可编辑 |

|  |  |
| --- | --- |
| 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.addUser | 语法 | public boolean addUser(UserPO user) |
| 前置条件 | 逻辑层请求新建用户 |
| 后置条件 | 在数据库中增加UserPO,返回增加是否成功 |
| UserManData.deleteUser | 语法 | public boolean deleteUser(String id) |
| 前置条件 | 逻辑层请求删除用户 |
| 后置条件 | 在数据库中删除对应UserPO，返回删除是否成功 |
| UserManData.reviseUser | 语法 | public boolean reviseUser(UserPO user) |
| 前置条件 | 逻辑层请求修改用户 |
| 后置条件 | 在数据库中修改对应UserPO，返回修改是否成功 |
| UserManData.findUser | 语法 | public ArrayList<UserPO> findUser(String keyword) |
| 前置条件 | 逻辑层请求查询用户 |
| 后置条件 | 在数据库中查找并返回对应UserPO |
| UserManData.checkIsUsed | 语法 | public Result checkIsUsed(String id) |
| 前置条件 | 逻辑层新增用户 |
| 后置条件 | 在数据库中查找返回本用户名是否已被使用 |

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

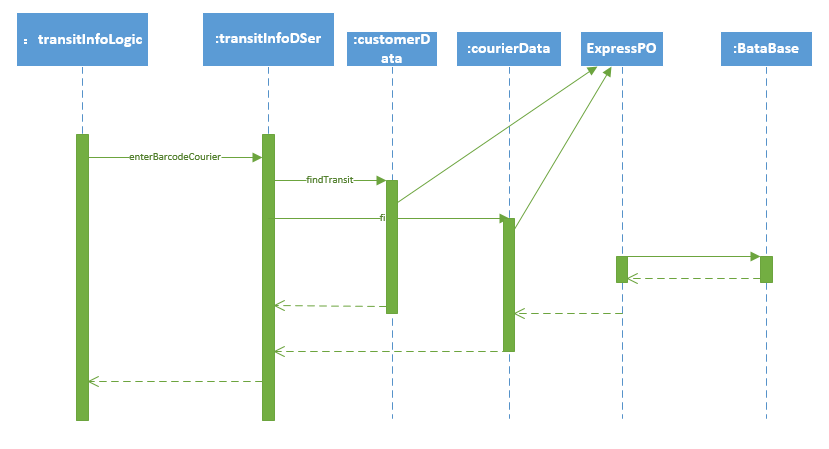
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CourierData.find | 语法 | public ExpressPO find(String barcode) throws RemoteException |
| 前置条件 | 有相应条形码存在 |
| 后置条件 | 数据库中查找返回相应的ExpressPO结果 |

CustomerData的接口规范

|  |  |  |
| --- | --- | --- |
| CustomerData.findTransit | 语法 | public ArrayList<String> findTransit(String barcode) throws RemoteException |
| 前置条件 | 输入相应的条形码 |
| 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 Result 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 |

CenterLoadingOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| CenterLoadingOrderData.Create | 语法 | public Result 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 |

ExpressOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| ExpressOrderData.Create | 语法 | public Result 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 Result 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 |

IncomeOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| IncomeOrderData.Create | 语法 | public Result 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 |

PaymentOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| PaymentOrderData.Create | 语法 | public Result 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 Result 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 |

RecipientOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| RecipientOrderData.Create | 语法 | public Result 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 |

SendOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| SendOrderData.create | 语法 | public Result 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 |

StockInOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| StockInOrderData.create | 语法 | public Result 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 |
| StockOutOrderData.getShelves | 语法 | public ArrayList<String> getShelves () throws RemoteException |
| 前置条件 | 单据新建逻辑请求架信息 |
| 后置条件 | 在数据库中获取并返回架信息 |

StockOutOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| StockOutOrderData.create | 语法 | public Result 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 |

TransitOrderData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| TransitOrderData.create | 语法 | public Result 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 |

UtilityData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| UtilityData.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 |
| UtilityData.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 |
| UtilityData.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 |
| UtilityData.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 |
| UtilityData.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 |
| UtilityData.getWorkers | 语法 | public ArrayList<String> getWorkers () throws RemoteException |
| 前置条件 | 新建单据逻辑请求工作人员信息 |
| 后置条件 | 在数据库中获取并返回工作人员信息 |
| UtilityData.getStocks | 语法 | public ArrayList<String> getStocks () throws RemoteException |
| 前置条件 | 新建单据逻辑请求库存信息 |
| 后置条件 | 在数据库中获取并返回库存信息 |

（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 Result 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 Result 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 Result 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.findDriver(String hallId) | 语法 | public ArrayList<DriverPO> searchDriver(String hallId) |
| 前置条件 | 打开司机管理界面或输入机构编号检索 |
| 后置条件 | 从数据库获取司机列表或含有相应机构的司机列表返回 |

OrgManData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrgManData.addOrg(OrgPO org) | 语法 | public Result 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,返回新增是否成功 |
| OrgManData.deleteOrg(String id) | 语法 | public Result 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,返回删除是否成功 |
| OrgManData.reviseOrg(OrgPO org) | 语法 | public Result 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,返回修改是否成功 |
| 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。 |

VanManData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| VanManData.addVan(VanPO Van) | 语法 | public Result 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,返回新增是否成功 |
| VanManData.deleteVan(String id) | 语法 | public Result 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,返回删除是否成功 |
| VanManData.reviseVan(VanPO Van) | 语法 | public Result 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，返回修改是否成功 |
| VanManData.findVan(String keyword，String hallId) | 语法 | public ArrayList<VanPO> searchVan(String keyword,String hallId) |
| 前置条件 | 打开车辆管理界面或输入关键词或按营业厅检索 |
| 后置条件 | 在数据库中获得含有相应关键词或对应营业厅（若关键词、营业厅为空则返回全部）的车辆信息列表返回。 |

WageManData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| WageManData.searchWork(String keyword) | 语法 | public Result searchWork(String keyword) |
| 前置条件 | 打开薪水管理界面或输入关键词检索 |
| 后置条件 | 从数据库获得对应人员薪水列表返回 |
| WageManData.updateWork(WorkPO work) | 语法 | public Result updateWork(WorkPO work) |
| 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 |

WorkManData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| WorkManData.addWork(WorkPO work) | 语法 | public Result 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,返回新增是否成功 |
| WorkManData.deleteWork(int id) | 语法 | public Result deleteWork(int 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，返回删除是否成功 |
| WorkManData.reviseWork(WorkPO work) | 语法 | public Result 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，返回修改是否成功 |
| 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。 |

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，ConstantDSer作为常量制定的数据层接口。CityPO CitiesPO ConstantPO作为持久化对象，。

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的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CityData.addCity | 语法 | public Result addCity(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对象，返回新增是否成功 |
| CityData.reviseCity | 语法 | Public Result reviseCity(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，返回更新是否成功 |
| CityData.reviseCities | 语法 | Public Result reviseCities(CitiesPO 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.addCities | 语法 | Public Result addCities(CitiesPO po) throws RemoteException |
| 前置条件 | 界面层制定常量 |
| 后置条件 | 在数据库中添加城市常量信息，返回添加是否成功 |
| CityData.deleteCity | 语法 | Public Result deleteCity(String cityName) throws RemoteException |
| 前置条件 | 城市对象存在 |
| 后置条件 | 在数据库中删除对应城市，返回删除是否成功 |
| CityData.deleteCities | 语法 | Public Result delateCities(CitiesPO cities) throws RemoteException |
| 前置条件 | 城市常量对象存在 |
| 后置条件 | 在数据库中删除对应城市常量，返回删除是否成功 |
| CityData.getCitiesInfo | 语法 | Public ArrayList<CitiesPO> getCitiesInfo() throws RemoteException |
| 前置条件 | 城市常量对象列表存在 |
| 后置条件 | 返回城市常量对象列表 |

（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 | 逻辑层需要查询账户 |
| 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 Result 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,返回增加是否成功 |
| AccountData.deleteAccount | 语法 | public Result deleteAccount(int 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，返回删除是否成功 |
| AccountData.reviseAccount | 语法 | public Result 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,返回更新是否成功 |

CountData的接口规范

|  |  |  |
| --- | --- | --- |
| CountData.addCount | 语法 | public Result addCount(CountPO count) throws RemoteException |
| 前置条件 | 用户新建账单 |
| 后置条件 | 在数据库中增加一个po记录,返回增加是否成功 |
| CountData.getCount | 语法 | public Result getCount() throws RemoteException |
| 前置条件 | 逻辑层需要查看账列表 |
| 后置条件 | 在数据库中获取并返回账列表信息 |
| CountData.deleteCount | 语法 | Public Result deleteCount(int id)throws RemoteException |
| 前置条件 | 逻辑层删除某套账 |
| 后置条件 | 在数据库中删除对应的账信息，返回删除是否成功 |

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. stockData模块

1. 模块概述

StockData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。StockData模块的职责及接口参见软件体系结构描述文档5.3.2中StockData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。StockPO,ShelfPO,GoodPO,StockInPO,StockOutPO分别存储库存、架、货物、入库单和出库单的持久化对象。ResultMessage存储操作是否成功和返回的信息。StockInOrderData是入库单生成数据服务，需要通过StockCheckWarnData检测是否需要报警。

StockData模块的设计如图所示。

stockData模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| StockCheckData | 负责库存查看逻辑所需要的数据服务 |
| StockCheckNowData | 负责库存盘点逻辑所需要的数据服务 |
| StockCheckWarnData | 负责库存报警逻辑所需要的数据服务 |
| StockDivideData | 负责库存分区逻辑所需要的数据服务 |
| StockWarningData | 负责库存报警设置逻辑所需要的数据服务 |

1. 模块内部类的接口规范

StockCheckData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockCheckData.getIn(Calendar start,Calendar end,String id) | 语法 | public ArrayList<StockInPO> getIn(Calendar start, Calendar end, String id) |
| 前置条件 | 输入了开始日期和结束日期 |
| 后置条件 | 从数据库获取开始和结束日期之间的入库单并返回 |
| StockCheckData.getOut(Calendar start, Calendar end,String id) | 语法 | public ArrayList<StockOutPO> getOut(Calendar start, Calendar end, String id) |
| 前置条件 | 输入了开始日期和结束日期 |
| 后置条件 | 从数据库获取开始和结束日期之间的出库单并返回 |

StockCheckNowData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockCheckNowData.getStock(String id) | 语法 | public StockPO getStock (String id) |
| 前置条件 | 打开库存盘点界面 |
| 后置条件 | 从数据库获取当前库存快照信息并返回 |

StockCheckWarnData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockCheckWarnData.getWarning(String id) | 语法 | public ArrayList<Integer> getWarning (String id) |
| 前置条件 | 逻辑层需要查看库存警戒线 |
| 后置条件 | 返回库存警戒线 |
| StockCheckWarnData.getStock() | 语法 | public StockPO getStock (String id) |
| 前置条件 | 逻辑层需要查看库存情况 |
| 后置条件 | 返回库存情况 |
| StockDivideData.getShelves(String id) | 语法 | public ArrayList<ShelfPO> getShelves (String id) |
| 前置条件 | 逻辑层需要获取架信息 |
| 后置条件 | 从数据库获得架列表或含有关键词的架列表并返回 |

StockDivideData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockDivideData.getShelves(String id,String keyword) | 语法 | public ArrayList<ShelfPO> getShelves (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并返回 |
| StockDivideData.addShelf(ShelfPO shelf) | 语法 | public Result addShelf(ShelfPO shelf) |
| 前置条件 | 输入了完整的架信息，选择添加 |
| 后置条件 | 将架信息添加到数据库 |
| StockDivideData.deleteShelf(String id) | 语法 | public Result 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 |
| StockDivideData.reviseShelf(ShelfPO shelf) | 语法 | public Result reviseShelf(ShelfPO shelf) |
| 前置条件 | 修改一个架的信息，确认修改 |
| 后置条件 | 修改数据库中的相应信息 |

StockWarningData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockWarningData.getWarning(String id) | 语法 | public ArrayList<Integer> getWarning(String id) |
| 前置条件 | 逻辑层需要获得库存警戒线 |
| 后置条件 | 从数据库中获得库存警戒线 |
| StockWarningData.setWarning(int w,Part part,String id) | 语法 | public Result getWarning(int w,Part part,String id) |
| 前置条件 | 确认修改警戒线 |
| 后置条件 | 修改数据库中相应区的警戒线 |

1. 数据层的动态模型

如图为执行库存盘点操作后，相关类的协作顺序图。

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的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CostBeneData. getIncomes | 语法 | public ArrayList<IncomePO> getIncome () |
| 前置条件 | 需要获得入款单信息 |
| 后置条件 | 返回入款单列表 |
| CostBeneData. getPayments | 语法 | public ArrayList<PaymentPO> getPayment () |
| 前置条件 | 需要获得付款单信息 |
| 后置条件 | 返回付款单列表 |

BusiCircumData的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| BusiCircumData.findIncome | 语法 | public ArrayList<IncomePO> findCollect(Calendar start,Calendar end) throw RemoteException |
| 前置条件 | 逻辑层需要获得收款单信息 |
| 后置条件 | 根据输入的开始和结束日期从数据库获得收款单列表，返回相应的数据 |
| BusiCircumData.findPayment | 语法 | public ArrayList<PaymentPO> findPayment(Calendar start,Calendar end) throws RemoteException |
| 前置条件 | 逻辑层需要获得付款单信息 |
| 后置条件 | 根据输入的开始和结束日期从数据库获得付款单列表，返回相应数据 |

DiaryData的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| DiaryData.seeDiary | 语法 | public ArrayList<DiaryPO> seeDiary(Calendar start,Calendar end) throws RemoteException |
| 前置条件 | 用户选择日志审批 |
| 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的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| CollectRecordData.getCollectRec | 语法 | public ArrayList<IncomePO> getCollectRec(Calendar date,String id) throws RemoteException |
| 前置条件 | 逻辑层需要获得收款单记录 |
| 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 |

（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接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderApproveData.update | 语法 | public Result update(boolean isPassed,ArrayList<String> id,Order kind) 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 PO getOrder(String id,Order kind) 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 | 相应id的单据存在于数据库中 |
| 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<KindGetter> getOrder(String userID) throws RemoteException |
| 前置条件 | 逻辑层需要获得未通过审核的单据 |
| 后置条件 | 从数据库获得ID相应人员的未通过单据列表 |

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. 界面层分解
     1. orderNewUI模块

1. 模块概述

orderNewUI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。

orderNewUI模块的职责及接口参见软件系统结构描述文档5.2.2表orderNewUI模块的接口规范。

1. 整体结构

根据体系结构的设计，我们将系统分为展示层，业务逻辑层，数据层。每一层之间为了增加灵活性，我们会添加接口。比如展示层和业务逻辑层之间，我们添加LogicService. OrderNewLogic接口。OrderVO是各种单据作为单据记录的持久化对象被添加到设计模型中去的。 ExpressPanel，ReceivePanel，PaymentPanel，HallLoadingPanel，IncomePanel，RecipientPanel，SendPanel，ArrivePanel，CenterLoadingPanel，TransitPanel，StockInPanel，StockOutPanel用于显示对应的单据信息。FunctionFrame负责由功能主界面到具体界面的跳转。

OrderNewUI模块的设计如图所示



OrderNewUI模块各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| FunctionFrame | 负责用户功能主界面到具体功能界面的跳转 |
| ExpressPanel | 负责显示快递单填写界面 |
| ReceivePanel | 负责显示收件单填写界面 |
| PaymentPanel | 负责显示付款单填写界面 |
| HallLoadingPanel | 负责显示营业厅装车单填写界面 |
| IncomePanel | 负责显示收款单填写界面 |
| RecipientPanel | 负责显示接收单填写界面 |
| SendPanel | 负责显示派件单填写界面 |
| ArrivePanel | 负责显示到达单填写界面 |
| CenterLoadingPanel | 负责显示中转中心装车填写界面 |
| TransitPanel | 负责显示中转单填写界面 |
| StockInPanel | 负责显示入库单填写界面 |
| StockOutPanel | 负责显示出库单填写界面 |

1. 模块内部接口规范

ArrivePanel的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ArrivePanel() | 语法 | public ArrivePanel() |
| 前置条件 | 用户选择生成到达单 |
| 后置条件 | 显示生成到达单界面 |
| 需要的接口（需接口） | | |
| 服务名 | 服务 | |
| ArriveOrderLSer.create | 负责提供新建单据的逻辑接口 | |
| ArriveOrderLSer.getCities | 负责提供获取城市的逻辑接口 | |
| ArriveOrderLSer.getLocalHallsAndAllCenter | 负责提供获取营业厅和中转中心的逻辑接口 | |

CenterLoadingPanel的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CenterLoadingPanel() | 语法 | public CenterLoadingPanel() |
| 前置条件 | 用户选择生成中转中心接收单 |
| 后置条件 | 显示生成中转中心接收单界面 |
| 需要的接口（需接口） | | |
| 服务名 | 服务 | |
| CenterLoadingOrderLSer.create | 负责提供新建单据的逻辑接口 | |
| CenterLoadingOrderLSer.getHalls | 负责获取营业厅选项的逻辑接口 | |

ExpressPanel的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ExpressPanel() | 语法 | public ExpressPanel() |
| 前置条件 | 用户选择生成快递单 |
| 后置条件 | 显示生成快递单界面 |
| 需要的接口（需接口） | | |
| 服务名 | 服务 | |
| ExpressOrderLSer.Create | 负责提供新建单据的逻辑接口 | |

HallLoadingPanel的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HallLoadingPanel() | 语法 | public HallLoadingPanel() |
| 前置条件 | 用户选择生成营业厅接收单 |
| 后置条件 | 显示生成营业厅接收单界面 |
| 需要的接口（需接口） | | |
| 服务名 | 服务 | |
| HallLoadingOrderLSer.create | 负责提供新建单据的逻辑接口 | |
| HallLoadingOrderLSer.getHalls | 负责获取营业厅选项的逻辑接口 | |
| HallLoadingOrderLSer.getOrgs | 负责获取到达机构的逻辑接口 | |
| HallLoadingOrderLSer. getVans | 负责获取车辆选项的逻辑接口 | |
| HallLoadingOrderLSer. getPrice | 负责获取运费的逻辑接口 | |
| HallLoadingOrderLSer. getLocalHallsAndAllCenter | 负责获取本市营业厅或中转中心的逻辑接口 | |

IncomePanel的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| IncomePanel() | 语法 | public IncomePanel () |
| 前置条件 | 用户选择生成收款单 |
| 后置条件 | 将用户填写的VO数据传入逻辑层 |
| 需要的接口（需接口） | | |
| 服务名 | 服务 | |
| IncomeOrderLSer.Create | 负责提供新建单据的逻辑接口 | |
| IncomeOrderLSer.getCourier | 负责获取快递员选项的逻辑接口 | |

PaymentPanel的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PaymentPanel() | 语法 | public PaymentPanel () |
| 前置条件 | 用户选择生成付款单 |
| 后置条件 | 将用户填写的VO数据传入逻辑层 |
| 需要的接口（需接口） | | |
| 服务名 | 服务 | |
| PaymentOrderLSer.Create | 负责提供新建单据的逻辑接口 | |
| PaymentOrderLSer. getAccount | 负责获取账户的逻辑接口 | |

ReceivePanel的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ReceivePanel() | 语法 | public ReceivePanel () |
| 前置条件 | 用户选择生成收件单 |
| 后置条件 | 显示收件单界面 |
| 需要的接口（需接口） | | |
| 服务名 | 服务 | |
| ReceiveOrderLSer.create | 负责提供新建单据的逻辑接口 | |
| ReceiveOrderLSer.getExpress | 负责获取订单的逻辑接口 | |

RecipientPanel的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| RecipientPanel() | 语法 | public RecipientPanel () |
| 前置条件 | 用户选择生成接收单 |
| 后置条件 | 显示接收单界面 |
| 需要的接口（需接口） | | |
| 服务名 | 服务 | |
| RecipientOrderLSer.Create | 负责提供新建单据的逻辑接口 | |
| RecipientOrderLSer.getCities | 负责获取城市的逻辑接口 | |

SendPanel的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| SendPanel() | 语法 | public SendPanel () |
| 前置条件 | 用户选择生成派件单 |
| 后置条件 | 显示派件单界面 |
| 需要的接口（需接口） | | |
| 服务名 | 服务 | |
| SendOrderLSer.create | 负责提供新建单据的逻辑接口 | |
| SendOrderLSer. getCouriers | 负责获取快递员的逻辑接口 | |

StockInPanel的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockInPanel() | 语法 | public StockInPanel () |
| 前置条件 | 用户选择生成入库单 |
| 后置条件 | 显示入库单界面 |
| 需要的接口（需接口） | | |
| 服务名 | 服务 | |
| StockInOrderLSer.Create | 负责提供新建单据的逻辑接口 | |
| StockInOrderLSer.getCities | 负责获取城市的逻辑接口 | |
| StockInOrderLSer.checkWarning | 负责获取警戒线的逻辑接口 | |
| StockInOrderLSer. getShelves | 负责获取架的逻辑接口 | |

StockOutPanel的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockOutPanel() | 语法 | public StockOutPanel () |
| 前置条件 | 用户选择生成出库单 |
| 后置条件 | 显示出库单界面 |
| 需要的接口（需接口） | | |
| 服务名 | 服务 | |
| StockOutOrderLSer.Create | 负责提供新建单据的逻辑接口 | |
| StockOutOrderLSer.getCities | 负责获取城市的逻辑接口 | |

TransitPanel的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| TransitPanel() | 语法 | public TransitPanel () |
| 前置条件 | 用户选择生成中转单 |
| 后置条件 | 显示出库单界面 |
| 需要的接口（需接口） | | |
| 服务名 | 服务 | |
| StockOutOrderLSer.Create | 负责提供新建单据的逻辑接口 | |
| StockOutOrderLSer.getCities | 负责获取城市的逻辑接口 | |
| StockOutOrderLSer.getCenters | 负责获取中转中心的逻辑接口 | |

1. 界面层的动态模型

下图表明了系统中，当用户填写了接收单后，订单界面间的跳转。其余单据动态模型类似如图。



1. 界面层的设计原理

见4.3.1.（5）

* + 1. transitInfoUI模块

1. 模块概述

transitInfoUI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。

transitInfoUI模块的职责及接口参见软件系统结构描述文档5.2.2表transitInfoUI模块的接口规范。

1. 整体结构

根据体系结构的设计，我们将系统分为展示层，业务逻辑层，数据层。每一层之间为了增加灵活性，我们会添加接口。比如展示层和业务逻辑层之间，我们添加LogicService. transitInfoLogic接口。StockCheckNowPanel，StockCheckPanel，StockWarningPanel，StockDividePanel用于显示对应的单据信息。FunctionFrame负责由功能主界面到具体界面的跳转。

TransitInfoUI模块的设计如图所示



TransitInfoUI模块各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| FunctionFrame | 负责用户功能主界面到具体功能界面的跳转 |

|  |  |
| --- | --- |
| TransitInfoInqPanel | 显示物流信息查询界面 |

1. 模块内部接口规范

TransitInfoUI的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| TransitInfoInqPanel() | 语法 | public TransitInfoInqPanel() |
| 前置条件 | 用户选择查看物流信息 |
| 后置条件 | 显示物流信息 |
| 需要的接口（需接口） | | |

|  |  |
| --- | --- |
| TransitInfoLSer.enterBarcodeCounrier(String  Barcode) | 查询订单信息界面的逻辑接口 |
| TransitInfoLSer.enterBarcodeCustomer(String  Barcode) | 查询物流轨迹和货运状态界面的逻辑接口 |

1. 界面层的动态模型

下图表明了系统中，当用户进行物流信息查询时界面间的跳转。



1. 界面层的设计原理

见4.3.1.（5）

* + 1. workOrgManUI模块

1. 模块描述

workOrgManUI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。

workOrgManUI模块的职责及接口参加软件系统体系结构描述文档5.2.2表workOrgManUI模块的接口规范

1. 整体结构

为了增加灵活性，我们在界面层和逻辑层之间添加logicService.WorkOrgManLogicService接口。VanListPanel DriverListPanel WorkListPanel WorkWageListPanel OrgListPanel反别作为车辆信息列表、司机信息列表、工作人员信息列表、人员薪水信息列表、机构信息列表的容器类，VanInfoPanel DriverInfoPanel WageDraftPanel用于车辆、司机、薪水详细信息查看及编辑，VanVO WorkVO DriverVO WageVO作为界面值对象。VanPO WorkPO DriverPO WagePO作为持久化对象保存。

WorkOrgManUI模块的设计如图所示



WorkOrgManUI各个类的职责如下表所示

WorkOrgManUI模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| FunctionFrame | 负责由功能主界面向具体功能界面的跳转 |
| VanInfoPanel | 负责显示车辆详细信息的面板类 |
| VanListPanel | 负责车辆信息的列表显示以及车辆信息管理操作的面板类 |
| DriverInfoPanel | 负责显示司机详细信息的面板类 |
| DriverListPanel | 负责司机信息的列表显示以及司机信息管理操作的面板类 |
| OrgListPanel | 负责机构信息的列表显示以及机构信息管理操作的面板类 |
| WageDraftPanel | 负责薪水制定操作的面板类 |
| WorkListPanel | 负责人员信息的列表显示以及人员信息管理操作的面板类 |
| WorkWageListPanel | 负责薪水信息的列表显示以及薪水信息管理操作的面板类 |

1. 模块内部类的接口规范

VanInfoPanel的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| VanInfoPanel | 语法 | Public vanInfoPanel() |
| 前置条件 | 用户打开车辆详细信息界面 |
| 后置条件 | 显示该车辆具体信息 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| public Result addVan(VanVO van) | 负责提供增加车辆的逻辑接口 |
| public Result deleteVan(String id) | 负责提供删除车辆的逻辑接口 |
| public Result reviseVan(VanVO van) | 负责提供修改车辆信息的逻辑接口 |
| public ResultMessage searchVan(String keyword,String hallId) | 负责提供车辆信息列表的逻辑接口 |
| public ResultMessage getNextId(String hallId) | 负责获取下一个车辆编号的逻辑接口 |

VanListPanel的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| VanListPanel | 语法 | Public VanListPanel(JFrame fr,UserVO uservo) |
| 前置条件 | 用户打开车辆列表界面 |
| 后置条件 | 显示车辆信息列表界面 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| public Result addVan(VanVO van) | 负责提供增加车辆的逻辑接口 |
| public Result deleteVan(String id) | 负责提供删除车辆的逻辑接口 |
| public Result reviseVan(VanVO van) | 负责提供修改车辆信息的逻辑接口 |
| public ResultMessage searchVan(String keyword,String hallId) | 负责提供车辆信息列表的逻辑接口 |
| public ResultMessage getNextId(String hallId) | 负责获取下一个车辆编号的逻辑接口 |

WageDraftPanel的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| WageDraftPanel | 语法 | Public WageDraftPanel(JFrame fr,WorkWageListPanel pa) |
| 前置条件 | 用户打开薪水制定界面 |
| 后置条件 | 显示薪水制定界面 |
| WageDraftPanel.isLegal | 语法 | private boolean isLegal() |
| 前置条件 | 用户输入薪水变量 |
| 后置条件 | 检查薪水数额是否在正常范围内 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| WorkOrgManSer. searchWorks(String keyword) | 负责获取人员薪水列表信息的逻辑接口 |
| WorkOrgManSer.setWage(List<WageVO> wageList,int[] index) | 设置薪水的逻辑接口 |

WorkWageListPanel的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| WorkWageListPanel | 语法 | Public WorkWageListPanel(JFrame fr) |
| 前置条件 | 用户打开人员薪水列表界面 |
| 后置条件 | 显示人员薪水列表界面 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| WorkOrgManSer. searchWorks(String keyword) | 负责获取人员薪水列表信息的逻辑接口 |
| WorkOrgManSer.setWage(List<WageVO> wageList,int[] index) | 设置薪水的逻辑接口 |

WorkListPanel的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| WorkListPanel | 语法 | Public WorkListPanel(JFrame fr) |
| 前置条件 | 用户打开人员列表界面 |
| 后置条件 | 显示人员列表界面 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| WorkManLSer.addWork（WorkVO work） | 负责提供新建人员的逻辑接口 |
| WorkManLSer.deleteWork(int index) | 负责提供删除人员的逻辑接口 |
| WorkManLSer.reviseWork(WorkVO work,int index) | 负责提供修改人员的逻辑接口 |
| WorkManLSer. orgList() | 负责获取机构列表的逻辑接口 |
| WorkManLSer.searchWork(String keywords) | 负责提供搜索特定人员的逻辑接口 |

DriverInfoPanel的接口规范参见VanInfoPanel接口规范；

DriverListPanel的接口规范参见VanListPanel的接口规范；

OrgListPanel的接口规范参见WorkListPanel的接口规范。

1. 界面层的动态模型

下图表明了在快递物流系统中，当用户使用车辆信息管理工能时，车辆管理界面处理的相关对象之间的协作。



如下图所示的状态图描述了用户在使用车辆管理功能时系统的状态序列、引起转移的事件，以及因为状态转移而伴随的动作。用户进入由功能主界面进入VanListPanel;之后通过新增或者查看已有车辆信息进入VanInfoPanel。



1. 界面层的设计原理

见4.3.1.（5）

* + 1. cityUI模块

1. 模块描述

cityUI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。

cityUI模块的职责及接口参加软件系统体系结构描述文档5.2.2表cityUI模块的接口规范

1. 整体结构

在界面层和逻辑层之间添加logicService.cityLogic logicService.constantLogic接口以增加灵活性。CityListPanel用于显示城市信息列表以及管理城市信息，ConstantPanel用于常量制定，CityVO ConstantVO作为界面值对象被添加到设计模型中去。

CityUI模块的设计如下图所示



CityUI模块类的职责

|  |  |
| --- | --- |
| CityListPanel | 负责城市管理职责的界面 |
| ConstantPanel | 负责常量信息制定职责的界面 |

1. 模块内部的接口规范

CityPanel的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| CityPanel | 语法 | Public CityListPanel(JFrame fr) |
| 前置条件 | 用户打开城市管理界面 |
| 后置条件 | 显示城市管理界面 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| CityLogic. addCity(CityVO city) | 添加城市的逻辑接口 |
| CityLogic. reviseCity(CityVO city,int index) | 修改城市信息的逻辑接口 |
| CityLogic. deleteCity(String cityName) | 删除城市的逻辑接口 |
| CityLogic. reviseCities(CitiesVO cities) | 修改城市间距离和价格的逻辑接口 |
| CityLogic. citiesList() | 获取城市距离与价格列表的逻辑接口 |
| CityLogic.cityList() | 获取城市列表的逻辑接口 |

ConstantPanel的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| ConstantPanel | 语法 | Public ConstantPanel(JFrame fr) |
| 前置条件 | 用户打开常量制定界面 |
| 后置条件 | 显示常量制定界面 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| ConstantLogic. setConstant(ConstantVO constant) | 制定常量的逻辑接口 |
| ConstantLogic. getConstant() | 获取常量信息的逻辑接口 |

（4）（5）参见4.2.5 AccountUI模块

* + 1. accountUI模块

1. 模块描述

accountUI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。

accountUI模块的职责及接口参加软件系统体系结构描述文档5.2.2表accountUI模块的接口规范。

1. 整体结构

根据体系结构的设计，我们将系统分为界面层、逻辑层、数据层。每一层之间为了增加灵活性，我们会添加接口。比如展示层和逻辑层之间，我们添加LogicService.accountLogic接口。AccountManPanel用于封装账户管理的职责，CountPanel和CountInfoPanel分别用于新建账和查看期初信息。CountVO和AccountVO作为分别作为账户和账的界面对象被添加到设计模型中。

AccountUI模块的设计如图所示



AccountUI模块各个类的设计

AccountUI模块各个类的职责如下表所示

AccountUI各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| FunctionFrame | 负责由用户功能主界面到具体功能界面的跳转 |
| AccountManPanel | 负责账户管理的面板 |
| CountPanel | 负责新建账的面板 |
| CountInfoPanel | 负责期初信息查看的面板 |

1. 模块内部类的接口规范

AccountManPanel的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| AccountManPanel | 语法 | Public AccountManPanel(JFrame fr) |
| 前置条件 | 用户打开账户管理界面 |
| 后置条件 | 显示账户管理界面 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| AccountLSer. addAccount(accountVO vo) | 根据vo新增账户对象 |
| AccountLcSer.deleteAccount(int index) | 根据index删除账户对象 |
| AccountLSer.reviseAccount(accountVO vo,int index) | 根据vo修改对应账户信息 |
| AccountLSer.searchAccount(String keywords) | 检索账户界面的逻辑接口 |

CountPanel的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| CountPanel | 语法 | Public CountPanel(JFrame fr) |
| 前置条件 | 用户打开账管理界面 |
| 后置条件 | 显示账管理界面 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| CountLSer.newCount | 新增账对象 |
| CountLSer.deleteCcount(int index) | 根据index删除账对象 |
| CountLSer.getCount() | 获得账信息 |

CountInfoPanel的模块接口

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| CountInfoPanel | 语法 | Public CountInfoPanel(CountVO count,JFrame fr) |
| 前置条件 | 用户打开账详细信息界面 |
| 后置条件 | 显示账详细信息界面 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| CountLSer.newCount | 新增账对象 |
| CountLSer.deleteCcount(int index) | 根据index删除账对象 |
| CountLSer.getCount() | 获得账信息 |

1. 界面层的动态模型

下图为使用账户管理功能时，账户管理界面处理相关对象之间的协作。



如下图所示的状态图描述了当用户使用账户管理功能时，系统的状态序列、引起转移的事件，以及因状态转移而伴随的动作呢。伴随着login方法被调用，界面跳转至FunctionFrame界面，用户选择账户管理功能后转至AccountManPanel。



AccountManPanel的状态图

1. 界面层的设计原理

见4.3.1.（5）

* + 1. stockUI模块

1. 模块概述

stockUI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。

stockUI模块的职责及接口参见软件系统结构描述文档5.2.2表stockUI模块的接口规范。

1. 整体结构

根据体系结构的设计，我们将系统分为展示层，业务逻辑层，数据层。每一层之间为了增加灵活性，我们会添加接口。比如展示层和业务逻辑层之间，我们添加LogicService. stockLogic接口。StockVO是作为库存记录的持久化对象，ShelfVO是作为货架记录的持久化对象被添加到设计模型中去的。StockCheckNowPanel，StockCheckPanel，StockWarningPanel，StockDividePanel用于显示对应的单据信息。FunctionFrame负责由功能主界面到具体界面的跳转。

stockUI模块的设计如图所示



stockUI模块各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| FunctionFrame | 负责用户功能主界面到具体功能界面的跳转 |

|  |  |
| --- | --- |
| StockCheckNowPanel | 库存查看界面，负责显示库存当前信息 |
| StockCheckPanel | 库存查看界面，负责显示库存信息 |
| StockWarningPanel | 库存警告界面，负责显示库存警告信息界面 |
| StockDividePanel | 库存分区界面，负责处理库存中架信息的管理操作 |

1. 模块内部接口规范
2. StockCheckPanel模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockCheckPanel() | 语法 | public StockCheckPanel () |
| 前置条件 | 用户选择库存查看 |
| 后置条件 | 显示时间段内库存信息 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| StockLogicSer. checkStockIn | 负责提供查看入库服务 |
| StockLogicSer.checkStockOut | 负责提供查看出库服务 |

1. StockCheckNowPanel模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockCheckNowPanel() | 语法 | public StockCheckNowPanel () |
| 前置条件 | 用户选择盘点，设置截止点 |
| 后置条件 | 显示库存盘点界面 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| CheckSer.CheckNow | 负责提供库存盘点服务 |
| CheckNowSer.outputExcel | 负责提供导出库存情况为Excel的接口 |

1. StockWarningPanel的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockWarningPanel() | 语法 | public StockWarningPanel () |
| 前置条件 | 用户打开警戒线修改界面 |
| 后置条件 | 显示原始警戒线值 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| StockWarningLSer.setWarning | 负责设置警戒线 |
| StockCheckWarnLSer.checkWarning | 负责检测库存是否超出警戒线的接口 |
| StockWarningLSer.getWarning | 负责获得警戒线 |

1. StockDividePanel模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockDividePanel() | 语法 | public StockDividePanel() |
| 前置条件 | 用户输入架的编号，选择搜索 |
| 后置条件 | 显示相应架的分区信息 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |

|  |  |
| --- | --- |
| StockDivideLSer. searchShelf | 负责提供查找架的接口 |
| StockDivideLSer.addShelf | 负责提供添加架的接口 |
| StockDivideLSer.deleteShelf | 负责提供删除架的接口 |
| StockDivideLSer.reviseShelf | 负责提供修改架的接口 |

1. 界面层的动态模型

下图表明了系统中，当用户进行库存报警时界面间的跳转。



库存报警的状态图



库存查看的状态图



1. 界面层的设计原理

见4.3.1.（5）

* + 1. checkUI模块

1. 模块概述

checkUI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。

checkUI模块的职责及接口参加软件系统体系结构描述文档5.2.2表checkUI模块的接口规范

1. 整体结构

根据体系结构的设计，我们将系统分为界面层、逻辑层、数据层。每一层之间为了增加灵活性，我们会添加接口。比如界面层和逻辑层之间，我们添加LogicService.checkLogic接口。IncomeVO PaymentVO DiaryVO BusiCircumVO作为界面对象，被添加到设计模型中去，DiaryListPanel保有显示日志记录信息的职责，CostBenePanel用于显示成本收益，IncomeListPanel用于显示收款信息，BusiCircumPabel保有查看收款单和入款单信息的职责，FunctionFrame负责由功能主界面到具体界面的跳转。

CheckUI模块的设计如图所示



CheckUI模块各个类的职责如下表所示。

CheckUI模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| FunctionFrame | 负责由用户功能主界面到具体功能界面的跳转 |
| CollectFormPanel | 负责显示成本收益表的面板 |
| DiaryListPanel | 负责显示日志记录的面板 |
| IncomeListPanel | 负责显示收款记录的面板 |
| StatementSheetPanel | 负责显示收款单和付款单信息列表的面板 |

1. 模块内部类的接口规范

CostBenePanel的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| CostBenePanel | 语法 | public CostBenePanel(JFrame fr) |
| 前置条件 | 打开成本收益界面 |
| 后置条件 | 显示成本收益界面 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| CostBeneLSer. getCostBene() | 获取成本收益 |

DiaryListPanel的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| DiaryListPanel | 语法 | Public DiaryListPanel(JFrame fr) |
| 前置条件 | 用户打开查看日志记录界面 |
| 后置条件 | 显示日志记录 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| DiaryLSer.seeDiary(Calendar start,Calendar end) | 根据开始和结束日期获得相应的日志记录 |

IncomeListPanel的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| IncomeListPanel | 语法 | Public IncomeListPanel(JFrame fr) |
| 前置条件 | 用户打开收款记录界面 |
| 后置条件 | 显示收款记录界面 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| IncomeListLSer.getHall() | 获得营业厅列表 |
| IncomeListLSer. total(ArrayList<IncomeVO> income) | 合计收款 |
| IncomeListLSer. seeIncomeList(Calendar date,String id) | 根据日期或营业厅获得所有的收款信息组成的列表 |

BusiCircumPanel的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| BusiCircumPanel | 语法 | Public BusiCircumPanel(JFrame fr) |
| 前置条件 | 用户打开经营情况表界面 |
| 后置条件 | 显示经营情况表界面 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| BusiCircumLSer. getBusiCircum(Calendar start,Calendar end) | 根据开始和结束日期获得相应的收款单和付款单信息列表。 |
| BusiCircumLSer. outputExcel(String name,String location) | 根据地址和名称导出表格 |

1. 界面层的动态模型

下图为快递物流系统中，当用户选择查看经营情况表功能时，界面层相关对象之间的协作。



下图所示的状态图描述了用户在使用查看经营情况表功能时系统界面的状态序列、引起转移的事件，以及因状态转移而伴随的动作。随着login方法被调用界面跳转至FunctionFrame,选择查看经营情况表转至StatementSheetPane，由经营情况表界面可选择跳转至IncomePanel或者PaymentPanel。



1. 界面层的设计原理

见4.3.1.（5）

* + 1. orderApproveUI模块

1. 模块描述

OrderApproveUI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。

OrderApproveUI模块的职责及接口参加软件系统体系结构描述文档5.2.2表OrderApproveUII模块的接口规范

1. 整体结构

在界面层和逻辑层之间添加OrderApproveLSer OrderResubmitLSer接口来增加灵活性。OrderApprovePanel保有查看单据列表以及审批单据的职责,ApproveTable作为OrderApprovePanel的辅助类，OrderRevisePanel用来显示审批未通过的单据，OrderVO以及各种单据的VO作为界面数据对象被添加到设计模型中。

OrderApproveUI模块的设计如下图所示



OrderApproveUI模块的类的职责如下表所示。

OrderApproveUI模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| FunctionFrame | 负责由功能主界面到具体功能界面的跳转 |
| OrderApprovePanel | 负责单据审批的面板 |
| ApproveTable | 单据审批的辅助类 |
| OrderRevisePanel | 负责显示未通过审批的单据的面板 |

1. 模块内部类的接口规范

OrderApprovePanel的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| OrderApprovePanel | 语法 | Public OrderApprovePanel(JFrame fr) |
| 前置条件 | 用户打开单据审批界面 |
| 后置条件 | 显示单据审批界面 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| orderApproveLSer.askExamine() | 提供待审批的单据信息列表 |
| orderApproveLSer. examine (boolean approve,ArrayList<Integer> indexs) | 审批单据并更新单据审批状态 |
| orderApproveLSer.chooseOrder(String id,Order kind) | 根据kinde和id查找对应的单据并提供单据的详细信息 |

ApproveTable的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| ApproveTable | 语法 | public ApproveTable(DefaultTableModel tableModel) |
| 前置条件 | 用户打开单据审批界面 |
| 后置条件 | 显示待审批单据的表格 |

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| ApproveTable. getTableHeader | 语法 | public JTableHeader getTableHeader() |
| 前置条件 | 无 |
| 后置条件 | 获取表头 |

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| ApproveTable. isCellEditable | 语法 | public boolean isCellEditable(int row, int column) |
| 前置条件 | 无 |
| 后置条件 | 返回是否可编辑 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| orderApproveLSer.askExamine() | 提供待审批的单据信息列表 |
| orderApproveLSer. examine (boolean approve,ArrayList<Integer> indexs) | 审批单据并更新单据审批状态 |
| orderApproveLSer.chooseOrder(String id,Order kind) | 根据kinde和id查找对应的单据并提供单据的详细信息 |

OrderRevisePanel的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| OrderRevisePanel | 语法 | Public OrderRevisePanel(JFrame fr) |
| 前置条件 | 用户打开单据重提交界面 |
| 后置条件 | 显示未通过审批待修改的单据 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |
| orderResubmitLSer. getUnpassed(String id) | 获取未通过审批的单据数量 |
| orderResubmitLSer. getOrdersDisplay() | 获得未通过审批的单据列表 |
| orderResubmitLSer. getOrder(int index) | 显示某单据详细信息 |

1. 界面层的动态模型

下图表明了在快递物流系统中，当用户使用单据审批功能时，单据审批界面处理的相关对象之间的协作。



下图所示的状态图描述了用户在使用单据审批功能时系统的状态序列、引起转移的事件，以及因转移而伴随的动作。随着用户登陆并选择单据审批功能，系统跳转至OrderApprovePanel, 之后可以在本界面进行全选和审批单据操作，也可以通过查看详细操作跳转至指定的单据信息界面，即\*orderPanel。



1. 界面层的设计原理

见4.3.1.（5）

* + 1. UserPanel模块

1. 模块概述

UserPanel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。

UserPanel模块的职责及接口参见软件系统体系结构描述文档5.2.2表*User*Panel*模块的接口规范*

1. 整体结构

在界面层和逻辑层之间，我们添加logicService.UserService接口以增强灵活性，各类用户的UserPanel保有让用户选择功能并跳转至相应界面的职责。

UserPanel模块的设计如图所示



UserPanel模块各个类的职责如下表所示

UserPanel模块各个类的职责

|  |  |
| --- | --- |
| AdministerPanel | 负责由管理员功能界面向具体功能界面的跳转 |
| BussinessHallPanel | 负责由营业厅业务员功能主界面向具体功能界面的跳转 |
| CourierPanel | 负责由快递员功能主界面向具体功能界面的跳转 |
| FinancePanel | 负责由普通财务人员功能主界面向具体功能界面的跳转 |
| FinanceSuperPanel | 负责由高级财务人员功能主界面向具体功能界面的跳转 |
| StockManagePanel | 负责由仓库管理员功能主界面向具体功能界面的跳转 |
| TransitCenterPanel | 负责由中转中心业务员功能主界面向具体功能界面的跳转 |

1. 模块内部类的接口规范

BusinessHallPanel模块的接口规范

|  |
| --- |
| 提供的服务（供接口） |

|  |  |  |
| --- | --- | --- |
| BusinessHallPanel | 语法 | public BusinessHallPanel(UserVO user,JFrame fr) |
| 前置条件 | 营业厅业务员登陆 |
| 后置条件 | 显示营业厅业务员功能界面 |

|  |  |
| --- | --- |
| 需要的接口（需接口） | |
| 服务名 | 服务 |

|  |  |
| --- | --- |
| UserManLSer. login(String id,String password) | 负责提供登录的逻辑接口 |

其他用户类的接口规范参照AdministerPanel的接口规范，依据用户功能的不同（参照界面图）变动方法名。

1. 界面的动态模型

下图为营业厅业务员进入系统后，界面层相关类之间的协作。



如下图所示的状态图描述了营业厅业务员由登陆至进入具体功能界面的状态序列，引起转移的事件以及由转移伴随的动作。



1. 界面层的设计原理

利用分散式分格，每个界面自主实现各自的界面职责。

1. 依赖视角

下图为客户端和服务器各包之间的依赖关系。

客户端各包之间的依赖关系

服务器端各包之间的依赖关系

