4.1.1 commoditybl模块

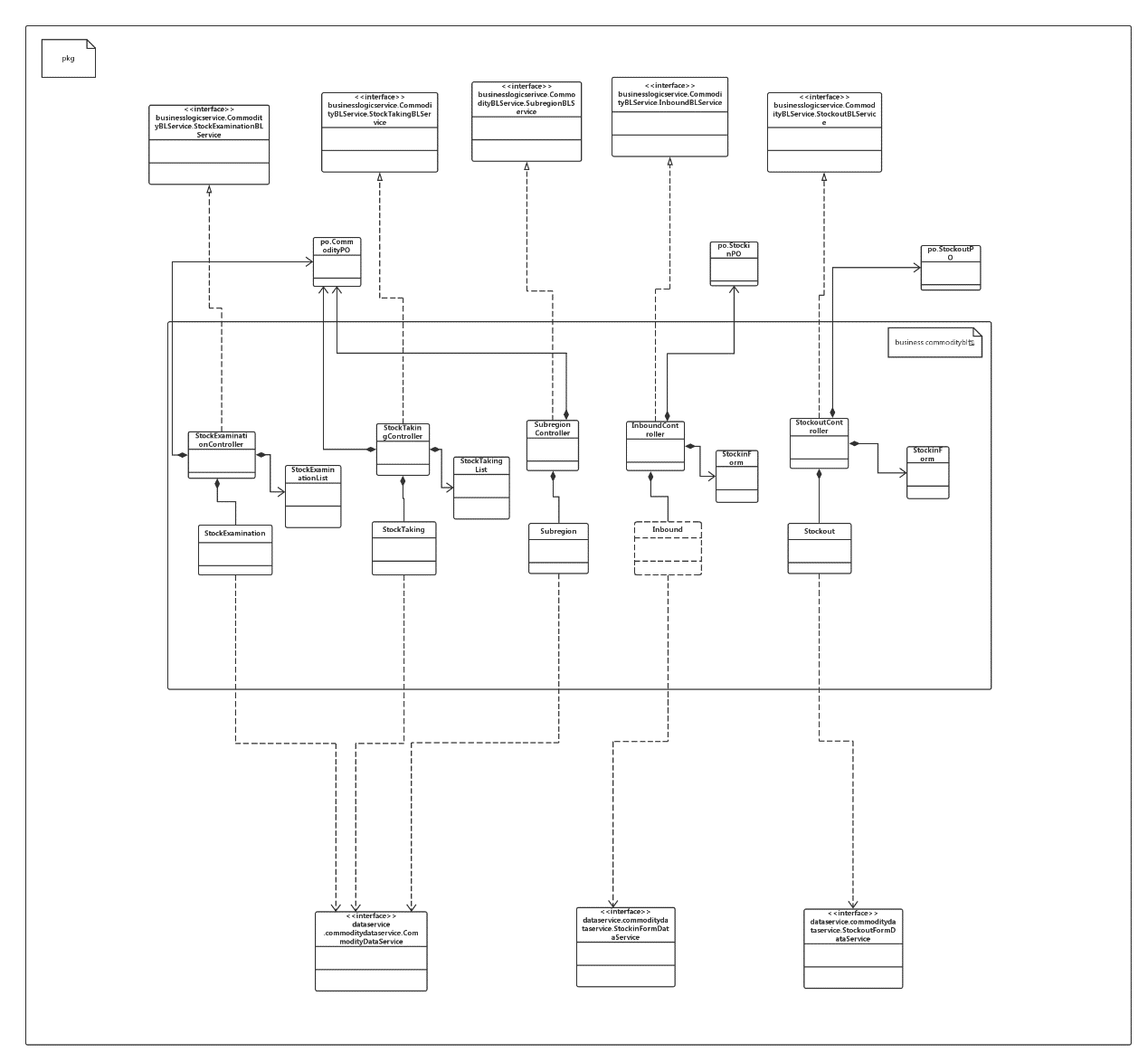
(1)模块概述

commoditybl模块承担的需求参见需求规格说明文档功能需求和相关非功能性需求。 commoditybl模块的职责及接口参见软件系统结构描述文档。

(2)整体结构

根据体系结构的设计，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。比如展示层与业务逻辑层之间，我们添加businesslogicservice.commodityblservice.StockExaminatonBlService接口。业务逻辑层和数据层之间添加dataservice.commoditydataservice.ComodityDataService接口。为了隔离业务逻辑职责和逻辑控制职责，我们增加了这样StockExaminationController,StockTakingController,SubreigonController,InboundController,StockoutController会将对StockExamination,StockTaking,Subreigon,Inbound,Stockout的业务逻辑处理委托给StockExamination,StockTaking,Subreigon,Inbound,Stockout对象。CommodityPO,StockinPO,StockoutPO是作为仓库记录的持久化对象被添加到设计模型中去的。

commoditybl模块的设计如下图所示。



Commodity模块各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| LoginController | 负责实现对应于登录界面所需要的服务 |
| CommodityController | 负责实现库存界面所需要的服务 |
| User | 系统用户的领域模型对象，拥有用户数据的姓名和密码，可以解决登录问题 |
| Commodity | 库存的领域模型对象，拥有一次库存事件处理所持有的库存信息，车辆信息，航班号等信息，可以帮助完成库存界面所需要的服务。 |

(3)模块内部类的接口规范

CommodityController和Commodity的接口规范如表C1和C2所示。

表C1 CommodityController的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CommodityController.StockExamination | 语法 | Public StockExaminationVO StockExamination (long startTime, long endTime) |
| 前置条件 | 已创建一个Commodity领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的StockExamination方法 |
| CommodityController.Stocktaking | 语法 | public StocktakingVO Stocktaking() |
| 前置条件 | 已创建一个Commodity领域对象 |
| 后置条件 | 调用Commodity领域对象的Stocktaking方法 |
| CommodityController.Inbound | 语法 | public ResultMessage Inbound(InboundVO vo) |
| 前置条件 | 已创建一个Commodity领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的Inbound方法 |
| CommodityController.Stockout | 语法 | public ResultMessage Stockout(StockoutVO vo) |
| 前置条件 | 已创建一个Commodity领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的Stockout方法 |
| CommodityController.Subregion | 语法 | public ResultMessage Subregion(Location beforeLocation, Location afterLocation) |
| 前置条件 | 已创建一个Commodity领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的Subregion t方法 |
| CommodityController.getRegionInfo | 语法 | public RegionInfoVO getRegionInfo (Region region) |
| 前置条件 | 已创建一个Commodity领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Commodity领域对象的getRegionInfo方法 |
| 需要的服务(需接口) | | |
| Commodity.StockExamination(long startTime, long endTime) | 进行一次库存查看 | |
| Commodity.Stocktaking() | 进行一次库存盘点 | |
| Commodity.Inbound(InboundVO vo) | 加入一个入库单 | |
| Commodity.Stockout(StockoutVO vo) | 加入一个出库单 | |
| Commodity.Subregion(Location beforeLocation, Location afterLocation) | 进行一次库存分区调整 | |
| Commodity.getRegionInfo(Region region) | 进行一次库存分区查看 | |

表C2 Commodity模块接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Commodity.StockExamination | 语法 | public StockExaminationVO StockExamination (long startTime, long endTime) | |
| 前置条件 | 数据库存在库存信息 | |
| 后置条件 | 无 | |
| Commodity.Stocktaking | 语法 | public StocktakingVO Stocktaking() | |
| 前置条件 | 数据库存在库存信息 | |
| 后置条件 | 无 | |
| Commodity.Inbound | 语法 | public ResultMessage Inbound(InboundVO vo) | |
| 前置条件 | 输入信息符合规范 | |
| 后置条件 | 数据库增加该入库单 | |
| Commodity.Stockout | 语法 | public ResultMessage Stockout(StockoutVO vo) | |
| 前置条件 | 输入信息规范 | |
| 后置条件 | 数据库增加该出库单 | |
| Commodity.Subregion | 语法 | public ResultMessage Subregion(Location beforeLocation, Location afterLocation) | |
| 前置条件 | 存在该货物 | |
| 后置条件 | 更改货物位置 | |
| Commodity.getRegionInfo | 语法 | public RegionInfoVO getRegionInfo (Region region) | |
| 前置条件 | 数据库存在分区信息 | |
| 后置条件 | 无 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| DatabaseFactory.getCommodityDatabase | | | 得到Commodity数据库的服务的引用 |
| CommodityDataService.getallCommodity(  ) | | | 得到数据库中所有CommodityPO对象 |
| CommodityDataService.getCommodityBetween(  Int starttime,int endtime) | | | 得到所有入库时间在starttime和endtime之间的CommodityPO对象 |
| CommodityDataService.addnew(CommodityPO po) | | | 向数据库中加入一个CommodityPO对象 |
| CommodityDataService.setOuttime(CommodityPO  po,int endtime) | | | 给某一Commodity对象填写出库时间 |
| CommodityDataService.changeLocation(CommodityPO  Po, blocknum, linenum, shelfnum, locationnum) | | | 给数据库中某一CommodityPO对象修改位置 |
| CommodityDataService.addStockinForm（StockinPO po） | | | 向数据库中插入StockinPO对象（付款单） |
| CommodityDataService.getStockinForm(Formstate state) | | | 从数据库中得到StockinPO对象 |
| CommodityDataService.updateStockinForm(StockinPO po) | | | 向数据库中更新StockinPO对象 |
| CommodityDataService.addStockoutForm（StockoutPO po） | | | 向数据库中插入StockoutPO对象（付款单） |
| CommodityDataService.getStockoutForm(Formstate state) | | | 从数据库中得到StockoutPO对象 |
| CommodityDataService.updateStockoutForm(StockoutPO po) | | | 向数据库中更新StockoutPO对象 |

4.1.2 transportbl模块

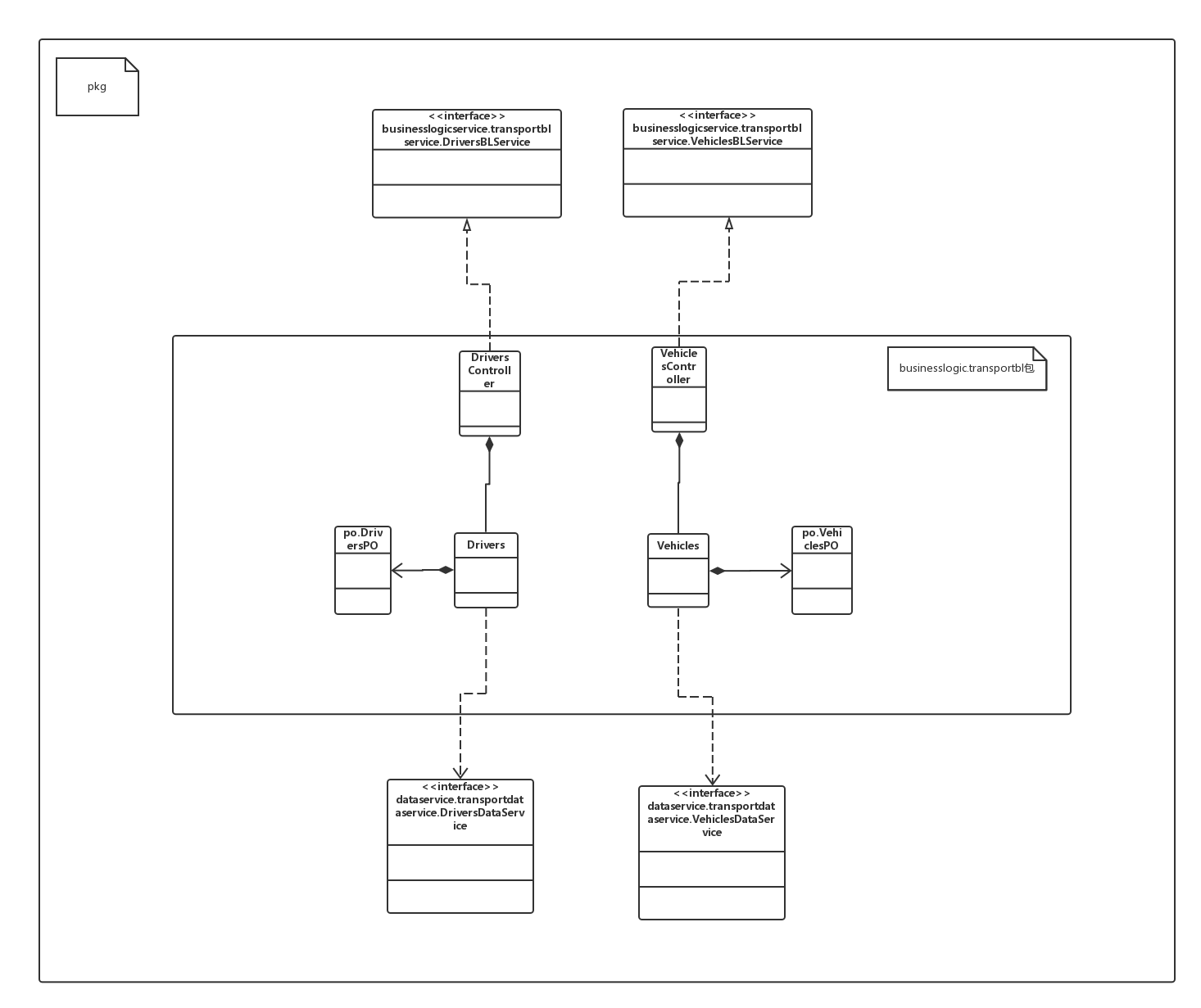
(1)模块概述

transportbl模块承担的需求参见需求规格说明文档功能需求和相关非功能性需求。 transportbl模块的职责及接口参见软件系统结构描述文档。

(2)整体结构

根据体系结构的设计，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。比如展示层与业务逻辑层之间，我们添加businesslogicservice.transportblservice.DriversBLService接口。业务逻辑层和数据层之间添加dataservice.transportdataservice.Drivers DataService接口。为了隔离业务逻辑职责和逻辑控制职责，我们增加了DriversController,VehiclesController，这样DriversController会将对Drivers的业务逻辑处理委托给Drivers对象，VehiclesController会将对Vehicles的业务逻辑处理委托给Vehicles对象。DriversPO和VehiclesPO是作为交通信息管理的持久化对象被添加进设计模型中去的。

transportbl模块的设计如下图所示。



Transport模块各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| LoginController | 负责实现对应于登录界面所需要的服务 |
| TransportController | 负责实现司机信息界面和车辆信息界面所需要的服务 |
| User | 系统用户的领域模型对象，拥有用户数据的姓名和密码，可以解决登录问题 |
| Transport | 库存的领域模型对象，拥有一次司机信息界面或者交通信息界面处理所持有的司机信息，车辆信息等信息，可以帮助完成界面所需要的服务。 |

transport模块各个类的职责如表T1，T2所示。

表T1 TransportController模块接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| TransportController.AddDriver | 语法 | public ResultMessage addDriver (DriverVO vo) |
| 前置条件 | 已创建一个Transport领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Transport领域对象的AddDriver方法 |
| TransportController.DelDriver | 语法 | public ResultMessage delDriver() |
| 前置条件 | 已创建一个Transport领域对象 |
| 后置条件 | 调用Transport领域对象的DelDriver 方法 |
| TransportController.RevDriver | 语法 | public ResultMessage revDriver (DriverVO vo) |
| 前置条件 | 已创建一个Transport领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Transport领域对象的RevDriver方法 |
| TransportController.getDriverbyDN | 语法 | public DriverVO getDriverby DN(String driverNumber) |
| 前置条件 | 已创建一个Transport领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Transport领域对象的getDriverbyDN方法 |
| TransportController.getDriverbyName | 语法 | public ResultMessage getDriverbyName (String name) |
| 前置条件 | 已创建一个Transport领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Transport领域对象的getDriverby Name方法 |
| TransportController.AddVehicle | 语法 | public ResultMessage addVehicle (VehicleVo vo) |
| 前置条件 | 已创建一个Vehicle领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Transport领域对象的AddVehicle方法 |
| TransportController.DelVehicle | 语法 | public ResultMessage delVehicle () |
| 前置条件 | 已创建一个Vehicle领域对象 |
| 后置条件 | 调用Transport领域对象的DelVehicle方法 |
| TransportController.RevVehicle | 语法 | public ResultMessage revVehicle (VehicleVo vo) |
| 前置条件 | 已创建一个Vehicle领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Transport领域对象的RevVehicle方法 |
| TransportController.getVehiclebyVN | 语法 | public ResultMessage getVehiclebyVN (String vehicleNumber) |
| 前置条件 | 已创建一个Vehicle领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Transport领域对象的getVehiclebyVN方法 |
| TransportController.getVehiclebyPN | 语法 | public ResultMessage getVehiclebyPN (String plateNumber) |
| 前置条件 | 已创建一个Vehicle领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Transport领域对象的getVehiclebyPN方法 |
| 需要的服务（需接口） | | |
| Transport.AddDriver(DriverVO vo) | | 加入一个司机对象 |
| Transport.DelDriver(DriverVO vo) | | 删除一个司机对象 |
| Transport.RevDriver(DriverVO vo) | | 修改一个司机对象 |
| Transport.getDriverbyDN(long driverNumber) | | 根据司机编号查询一个司机对象 |
| Transport.getDriverbyName(String name) | | 根据司机姓名查询一个司机对象 |
| Transport.AddVehicle(VehicleVo vo) | | 加入一个车辆对象 |
| Transport.DelVehicle(VehicleVO vo) | | 删除一个车辆对象 |
| Transport.RevVehicle(VehicleVO vo) | | 修改一个车辆对象 |
| Transport.getVehiclebyVN(String vehicleNumber) | | 根据车辆编号查询一个车辆信息 |
| Transport.getVehiclebyPN(String plateNumber) | | 根据车牌号查询一个车辆信息 |

表T2 transport模块接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Transport.AddDriver | 语法 | public ResultMessage addDriver (DriverVO vo) | |
| 前置条件 | 启动一个司机信息管理任务 | |
| 后置条件 | 在数据库中新增driver信息 | |
| Transport.DelDriver | 语法 | public ResultMessage delDriver(DriverVO vo) | |
| 前置条件 | 数据库中存在对应driver信息 | |
| 后置条件 | 在数据库中删除该driver信息 | |
| Transport.RevDriver | 语法 | public ResultMessage revDriver (DriverVO vo) | |
| 前置条件 | 数据库中存在对应driver信息 | |
| 后置条件 | 在数据库中修改该driver信息 | |
| Transport.getDriverbyDN | 语法 | public DriverVO getDriverby DN(String driverNumber) | |
| 前置条件 | 数据库中存在对应driver信息 | |
| 后置条件 | 无 | |
| Transport.getDriverbyName | 语法 | public ResultMessage getDriverbyName (String name) | |
| 前置条件 | 数据库中存在对应driver信息 | |
| 后置条件 | 无 | |
| Transport.AddVehicle | 语法 | public ResultMessage addVehicle (VehicleVO vo) | |
| 前置条件 | 启动一个车辆信息管理任务 | |
| 后置条件 | 在数据库中新增Vehicle信息 | |
| Transport.DelVehicle | 语法 | public ResultMessage delVehicle (VehicleVO vo) | |
| 前置条件 | 数据库中存在对应Vehicle信息 | |
| 后置条件 | 在数据库中删除该Vehicle信息 | |
| Transport.RevVehicle | 语法 | public ResultMessage revVehicle (VehicleVO vo) | |
| 前置条件 | 数据库中存在对应Vehicle信息 | |
| 后置条件 | 在数据库中修改该Vehicle信息 | |
| Transport.getVehiclebyVN | 语法 | public ResultMessage getVehiclebyVN (String vehicleNumber) | |
| 前置条件 | 数据库中存在对应Vehicle信息 | |
| 后置条件 | 无 | |
| Transport.getVehiclebyPN | 语法 | public ResultMessage getVehiclebyPN (String plateNumber) | |
| 前置条件 | 数据库中存在对应Vehicle信息 | |
| 后置条件 | 无 | |
| 需要的服务（需接口） | | | |
| DatabaseFactory. getTransportDaTabase | | | DatabaseFactory.getTransportDa  Tabase |
| TransportDataService.insertDriver(DriversPO po) | | | 在数据库中插入DriversPO对象 |
| TransportDataService.deleteDriver(DriversPO po) | | | 在数据库中删除DriversPO对象 |
| TransportDataService.updateDriver(DriversPO po) | | | 在数据库中修改更新某一DriversPO对象 |
| TransportDataService.findDriver(String identity) | | | 在数据库中查找某一DriversPO对象 |
| TransportDataService.insertVehicle(VehiclesPO po) | | | 在数据库中插入VehiclesPO对象 |
| TransportDataService.deleteVehicle(VehiclesPO po) | | | 在数据库中删除VehiclesPO对象 |
| TransportDataService.updateVehicle(VehiclesPO po) | | | 在数据库中修改更新某一VehiclesPO对象 |
| TransportDataService.findVehicle(String identity) | | | 在数据库中查找某一VehiclesPO对象 |

4.1.3 Userbl模块

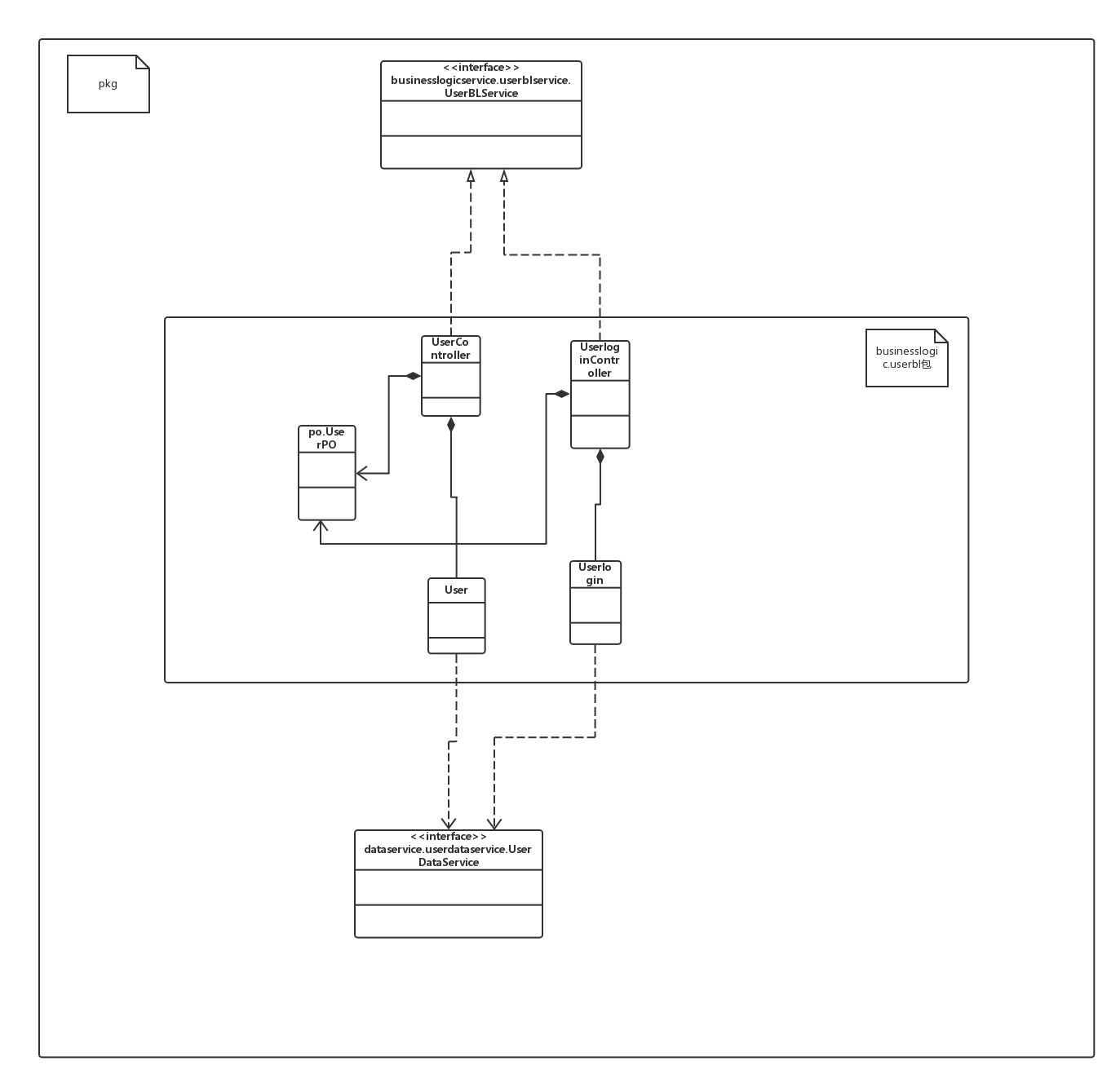
(1)模块概述

Userbl模块承担的需求参见需求规格说明文档功能需求和相关非功能性需求。 Userbl模块的职责及接口参见软件系统结构描述文档。

(2)整体结构

根据体系结构的设计，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。比如展示层与业务逻辑层之间，我们添加businesslogicservice.Userblservice.UserBLService接口。业务逻辑层和数据层之间添加dataservice.Userdataservice.UserDataService接口。为了隔离业务逻辑职责和逻辑控制职责，我们增加了UserController,UserloginController这样UserController会将对User的业务逻辑处理委托给User对象，UserloginController会将对Userlogin的业务逻辑处理委托给Userlogin对象。UserPO作为用户管理的持久化对象被添加进设计模型中去的。

userbl模块的设计如下图所示。



user模块各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| LoginController | 负责实现对应于登录界面所需要的服务 |
| UserController | 负责实现司机信息界面和车辆信息界面所需要的服务 |
| User | 系统用户的领域模型对象，拥有用户数据的姓名和密码，可以解决登录问题。拥有一次用户信息界面处理所持有的用户信息等信息，可以帮助完成界面所需要的服务。 |

User模块各个类的职责如表U1，U2所示。

表U1 UserController模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| User.Login | 语法 | public ResultMessage login(long id,String password) | |
| 前置条件 | 已创建一个User领域对象，并且输入符合输入规则 | |
| 后置条件 | 调用User领域对象的Login方法 | |
| User.AddUser | 语法 | public ResultMessage addUser(UserVO vo) | |
| 前置条件 | 已创建一个User领域对象，并且输入符合输入规则 | |
| 后置条件 | 调用User领域对象的AddUser方法 | |
| User.DelUser | 语法 | public ResultMessage delUser(UserVO vo) | |
| 前置条件 | 已创建一个User领域对象 | |
| 后置条件 | 调用User领域对象的DelUser方法 | |
| User.RevUser | 语法 | public ResultMessage revUser(UserVO vo) | |
| 前置条件 | 已创建一个User领域对象，并且输入符合输入规则 | |
| 后置条件 | 调用User领域对象的RevUser方法 | |
| User.getUser | 语法 | public UserVO getUserbyID(long id) | |
| 前置条件 | 已创建一个User领域对象，并且输入符合输入规则 | |
| 后置条件 | 调用User领域对象的getUser方法 | |
| 需要的服务(需接口) | | | |
| 服务名 | | | 服务 |
| User.Login(long id,String password) | | | 登录一个用户对象 |
| User.AddUser(UserVO vo) | | | 加入一个用户对象 |
| User.DelUser(UserVO vo) | | | 删除一个用户对象 |
| User.RevUser(UserVO vo) | | | 修改一个用户对象 |
| User.getUser(long id) | | | 根据id得到一个用户对象 |

表U2 user模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| User.Login | 语法 | public ResultMessage login(long id,String password) | |
| 前置条件 | password符合输入规则 | |
| 后置条件 | 查找是否存在相应的User，根据输入的password返回登录验证结果 | |
| User.AddUser | 语法 | public ResultMessage addUser(UserVO vo) | |
| 前置条件 | id,password合法 | |
| 后置条件 | 在数据库中新增User | |
| User.DelUser | 语法 | public ResultMessage delUser(UserVO vo) | |
| 前置条件 | 数据库中存在对应User | |
| 后置条件 | 在数据库中删除该User | |
| User.RevUser | 语法 | public ResultMessage revUser(UserVO vo) | |
| 前置条件 | 数据库中存在对应User | |
| 后置条件 | 在数据库中修改该User | |
| User.getUser | 语法 | public UserVO getUserbyID(long id) | |
| 前置条件 | 数据库中存在对应User | |
| 后置条件 | 无 | |
| 需要的服务（需接口） | | | |
| DatabaseFactory.getUserDatabase | | | 得到User数据库的服务的引用 |
| UserDataService.insert(UserPO po) | | | 在数据库中插入UserPO对象 |
| UserDataService.delete(UserPO po) | | | 在数据库中插入UserPO对象 |
| UserDataService.update(UserPO po) | | | 在数据库中修改更新某一UserPO对象 |
| UserDataService.find(long username) | | | 在数据库中根据用户名查找某一UserPO对象 |