物流管理系统（LMS） 软件详细设计描述文档

编写者

崔浩 杜天蛟 高露 倪安松



# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 编写者 | 更新内容 | 版本号 |
| 2015/11/05 | 高露 | 初稿 | V1.0 |
|  |  |  |  |
|  |  |  |  |

目录

更新历史 2

1.引言 3

1.1编制目的 3

1.2词汇表 4

1.3参考资料 4

2.产品概述 4

3.体系结构设计概述 4

4.详细设计的原理 5

5.结构视角 5

5.1业务逻辑层的分解 5

5.1.1Transportbl模块 5

5.1.2Warehousebl模块 10

5.1.3Personnelbl模块 10

5.1.4Departmentbl模块 10

5.1.5Financebl模块 19

5.1.6Listbl模块 19

5.1.7Userbl模块 23

5.1.8Logbl模块 29

6.依赖视角 30

# 1.引言

## 1.1编制目的

本报告详细完成灯具照明行业进销存系统的详细设计，达到指导后续软件构造的目的，同时实现和测试人员及用户的沟通。

本报告面向开发人员、测试人员及最终用户而编写，是了解系统的导航。

## 1.2词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| LMS | 物流管理系统 |  |
| BL | 业务逻辑 |  |
| PO | 数据持久化对象 |  |
| VO | 值对象 |  |

## 1.3参考资料

1.3.1 IEEE标准

1.3.2 物流管理系统（LMS）用例文档v2

1.3.3 物流管理系统（LMS）软件需求规格说明文档v2

1.3.4 项目实践v4

1.3.5 物流管理系统（LMS）体系结构描述文档

# 2.产品概述

参考物流管理系统用例文档和物流管理系统软件规格说明中对产品的概括描述。

# 3.体系结构设计概述

参考物流管理系统体系结构描述文档对体系结构设计的概述。

# 4.详细设计的原理

# 5.结构视角

## 5.1业务逻辑层的分解

业务逻辑层的开发包图参见物流管理系统体系结构描述文档。

### 5.1.1Transportbl模块

（1）模块概述

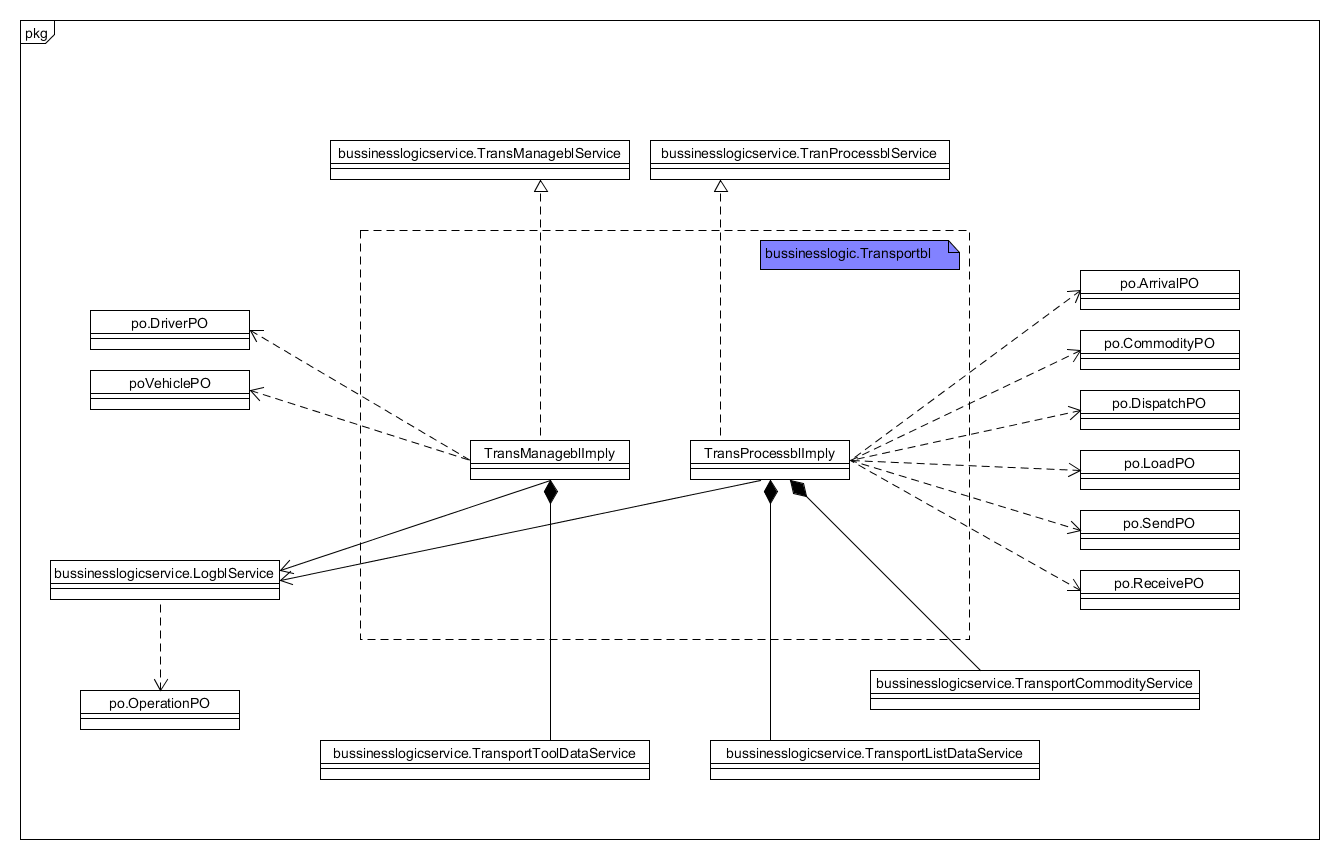
Transportbl模块承担的需求参见需求规格说明文档功能需求和相关非功能需求。

Salesbl模块的职责及接口参见软件系统结构描述文档

（2）整体结构

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

Transportbl模块的设计如图所示：



Transportbl模块各个类的职责如表所示：

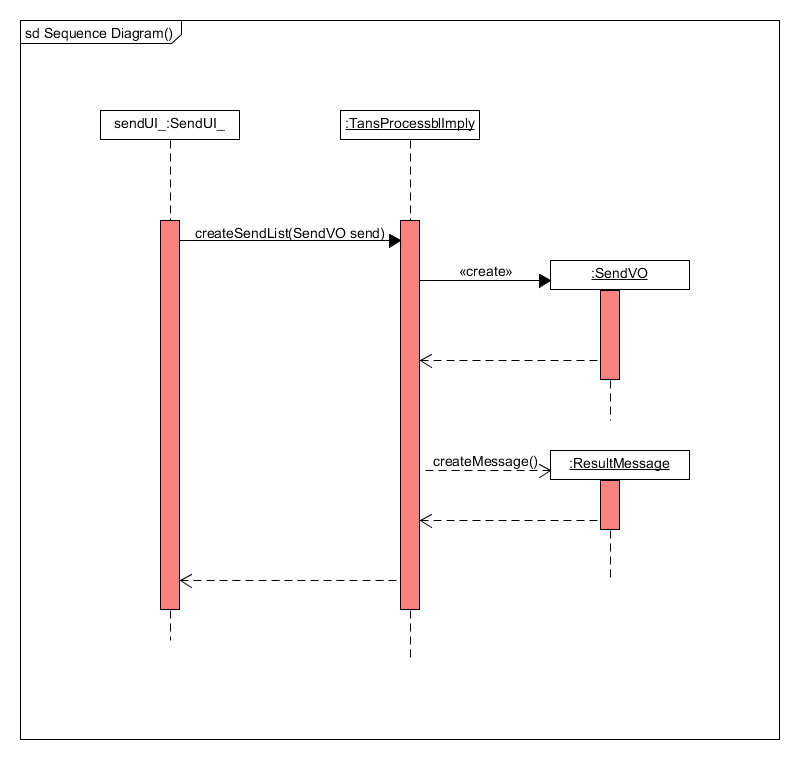
|  |  |
| --- | --- |
| 模块 | 职责 |
| TransManageblImpl | 负责处理司机和车辆管理的业务逻辑实现 |
| TransProcessblImpl | 负责生成和保存运输过程中产生的单据，包括寄件单，转运单，到达单，派件单，收件单。 |

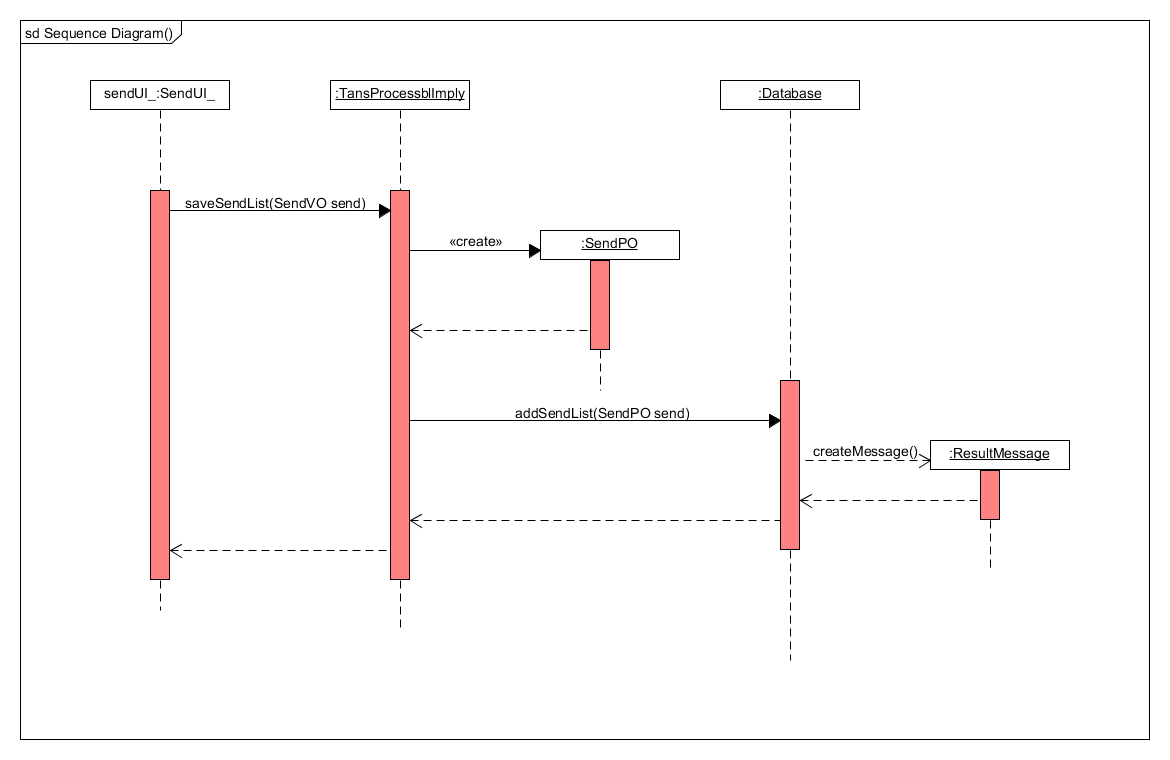
（3）模块内部的接口规范

参见体系结构设计TransManageblService和TransProcessblService的接口规范。

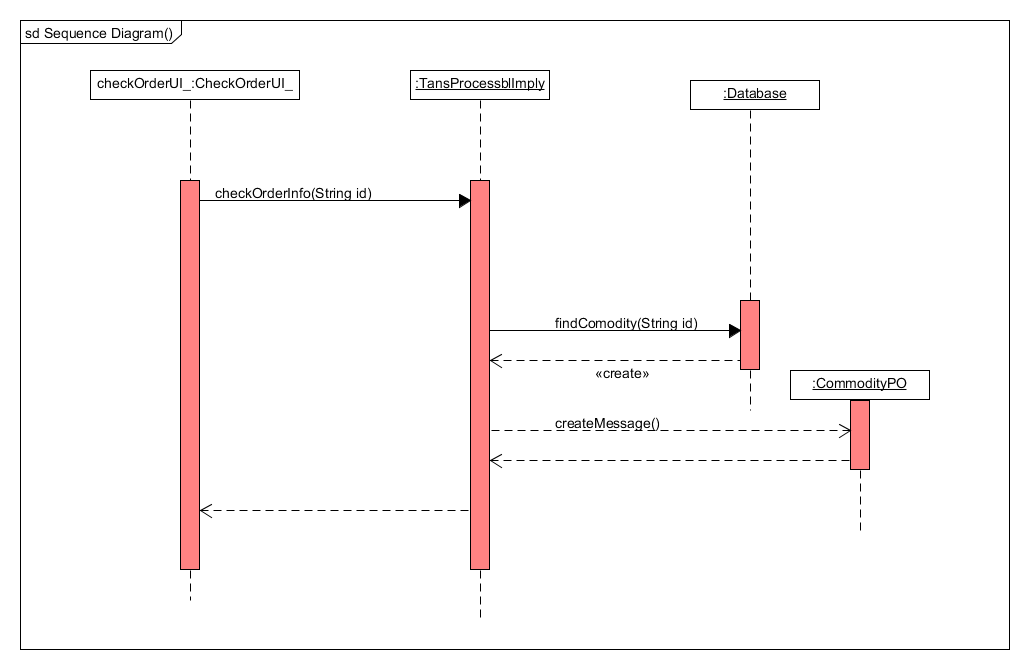
（4）业务逻辑层的动态模型

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

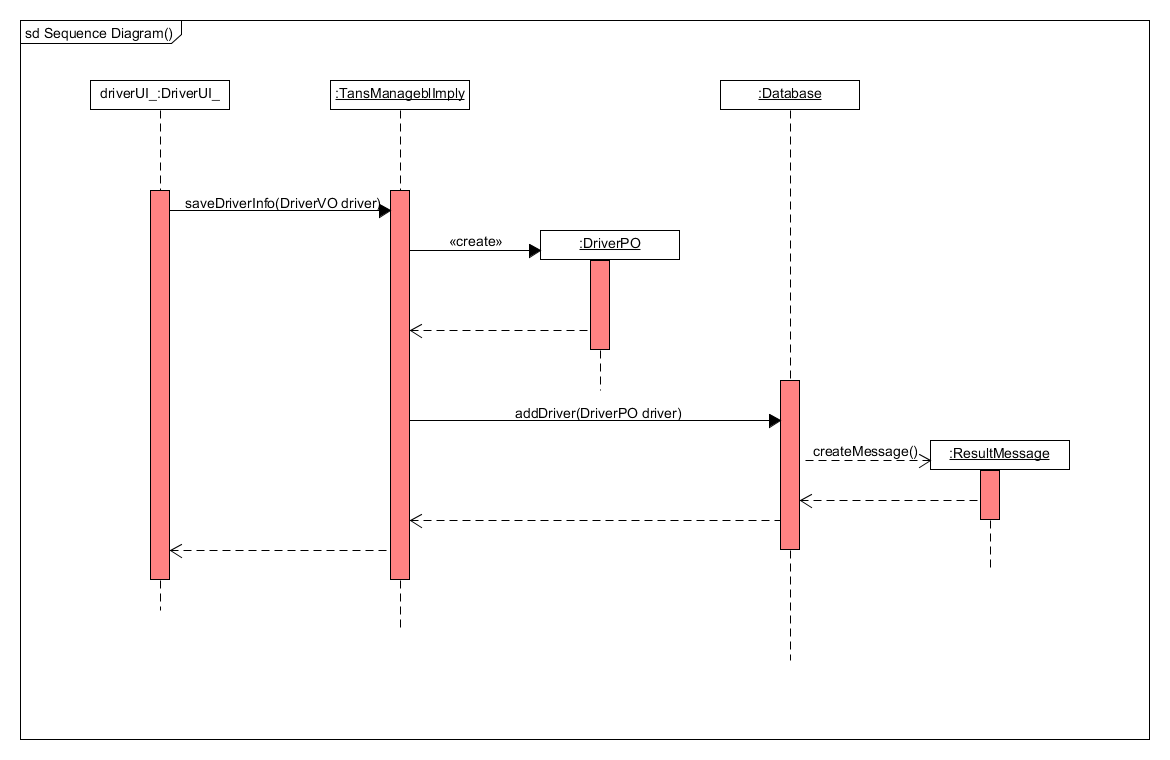
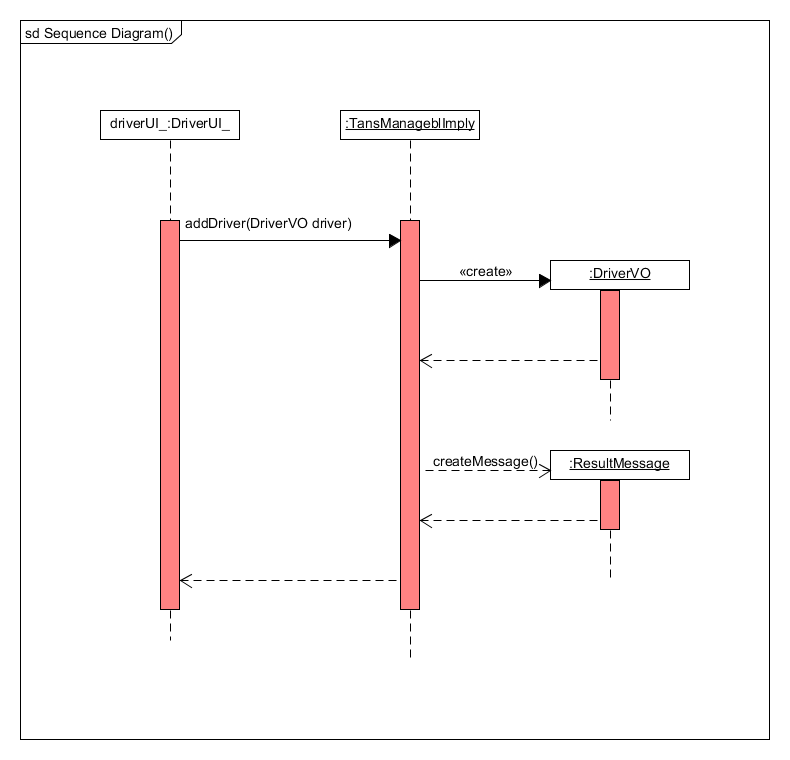


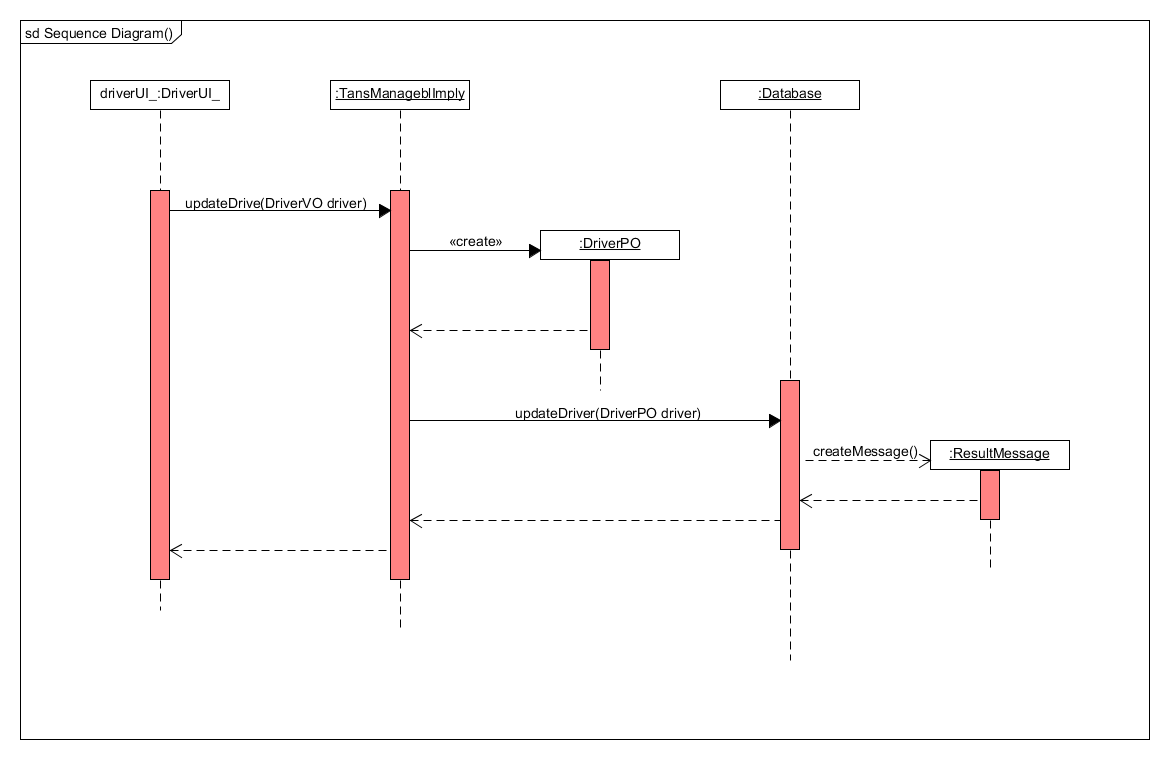


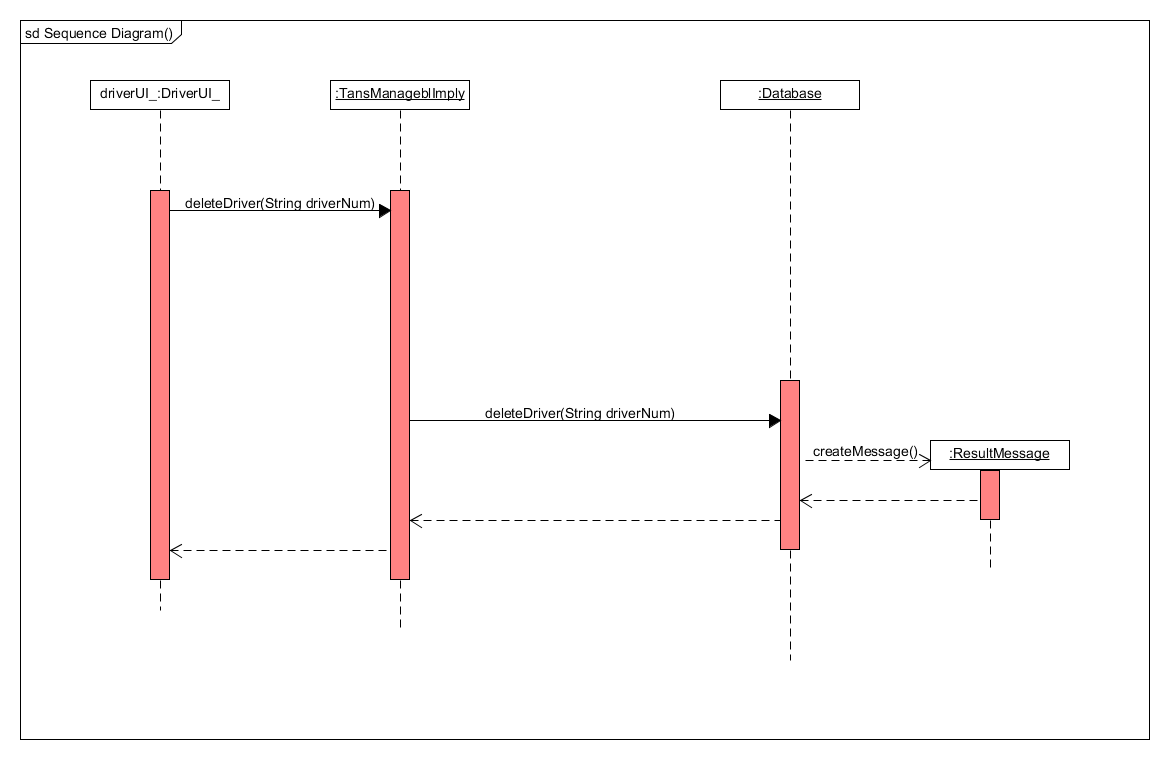
下图表示了用户查询物流状态和历史轨迹时，运输业务逻辑处理的相关对象之间的操作。



下图表示了用户管理司机信息时，运输业务逻辑处理的相关对象之间的操作。







(5)业务逻辑层的设计原理

分别创建一个类实现运输服务两个接口文件中的方法。

### 5.1.2Warehousebl模块

### 5.1.3Personnelbl模块

### 5.1.4Departmentbl模块

(1)模块概述

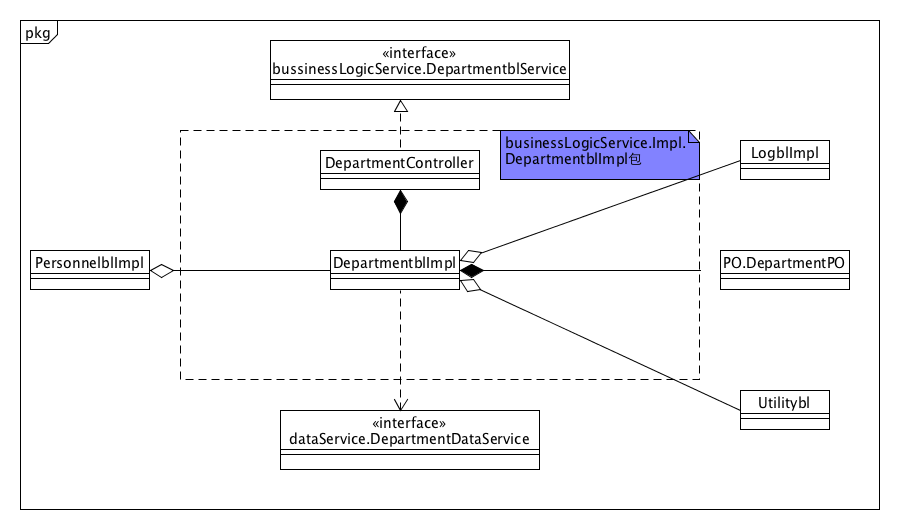
Departmentbl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

Departmentbl模块的职责和接口参见软件体系结构描述文档。

(2)整体结构

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

Departmentbl模块的设计如图所示。



Departmentbl模块的各个类的职责如下表所示。

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

(3)模块内部类的接口规范

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

DepartmentController的接口规范

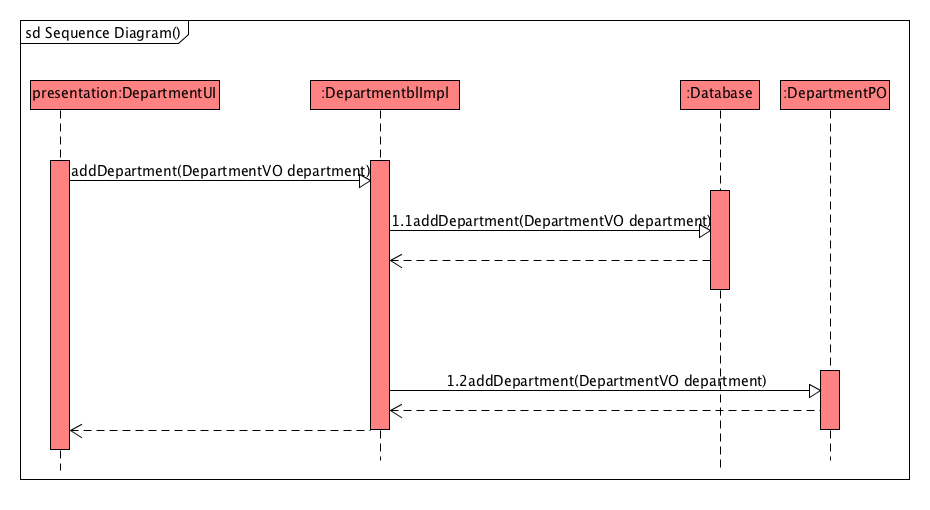
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DepartmentController.getDepartmentInfo | 语法 | Public DepartmentPO getDepartInfo(long id) |
| 前置条件 | 无 |
| 后置条件 | 查找是否存在相应的Department，如果存在返回相应的Department信息 |
| DepartmentController.deleteDepartment | 语法 | Public Boolean deleteDepartment(long id) |
| 前置条件 | 无 |
| 后置条件 | 查找是否存在相应的Department， 如果删除成功返回true，否则返回false |
| DepartmentController.changeDepartment | 语法 | Public Boolean changeDepartment(DepartmentVO Department) |
| 前置条件 | 无 |
| 后置条件 | 根据id查找Department，并修改相应的数据，如果修改成功返回true，否则返回false |
| DepartmentController.addDepartment | 语法 | Public Boolean addDepartment(DepartmentVO Department) |
| 前置条件 | 无 |
| 后置条件 | 如果增加成功返回true，否则返回false |
| DepartmentController. addCity | 语法 | Public Boolean addCity(CityVO city); |
| 前置条件 | 传入的城市信息符合规范 |
| 后置条件 | 如果增加成功返回true，否则返回false |
| DepartmentController. findCity | 语法 | Public CityVO findCity(long id); |
| 前置条件 | 系统存在相应的id |
| 后置条件 | 返回城市信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| Department.getDepartmentInfo(long id) | 根据id得到一个机构对象的信息 | |
| Department.deleteDepartment(long id) | 根据id删除一个机构对象 | |
| Department.changeDepartment(DepartmentVO Department) | 修改一个机构对象 | |
| Department.addDepartment(DepartmentVO Department) | 新增一个机构对象 | |
| Department. addCity(CityVO city) | 新增一个城市对象 | |
| Department. findCity(long id) | 根据id得到一个城市对象的信息 | |

Departmentbl的接口规范

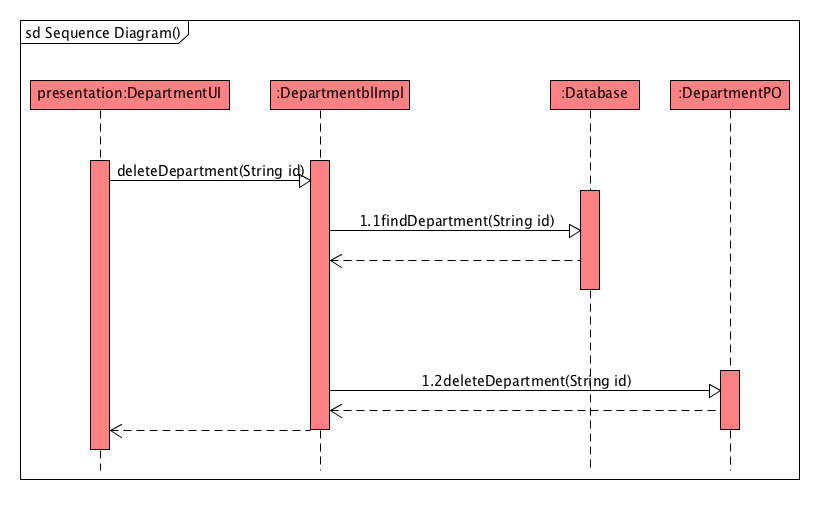
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Department.getDepartmentInfo | 语法 | Public DepartmentPO getDepartInfo(long id) |
| 前置条件 | 无 |
| 后置条件 | 查找是否存在相应的Department，如果存在返回相应的Department信息 |
| Department.deleteDepartment | 语法 | Public Boolean deleteDepartment(long id) |
| 前置条件 | 无 |
| 后置条件 | 查找是否存在相应的Department， 如果删除成功返回true，否则返回false |
| Department.changeDepartment | 语法 | Public Boolean changeDepartment(DepartmentVO Department) |
| 前置条件 | 无 |
| 后置条件 | 根据id查找Department，并修改相应的数据，如果修改成功返回true，否则返回false |
| Department.addDepartment | 语法 | Public Boolean addDepartment(DepartmentVO Department) |
| 前置条件 | 无 |
| 后置条件 | 如果增加成功返回true，否则返回false |
| Department. addCity | 语法 | Public Boolean addCity(CityVO city); |
| 前置条件 | 传入的城市信息符合规范 |
| 后置条件 | 如果增加成功返回true，否则返回false |
| Department. findCity | 语法 | Public CityVO findCity(long id); |
| 前置条件 | 系统存在相应的id |
| 后置条件 | 返回城市信息 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| DepartmentDataService  .addDepartment(DepartmentPo department)：boolean | 增加机构的单一持久化对象 | |
| DepartmentDataService  .findDepartment(long id):DepartmentPo | 根据id查找机构的单一持久化对象并返回 | |
| DepartmentDataService  .deleteDepartment(long id):boolean | 根据id删除机构的单一持久化对象 | |
| DepartmentDataService  .updateDepartment(DepartmentPo department):boolean | 查找相应的机构持久化对象并替换成传入的机构持久化对象 | |
| DepartmentDataService  .addCity(CityPo city):boolean | 增加城市持久化对象 | |
| DepartmentDataService  .findCity(long id):CityPO | 根据id返回城市持久化对象 | |

(4)业务逻辑层的动态模型

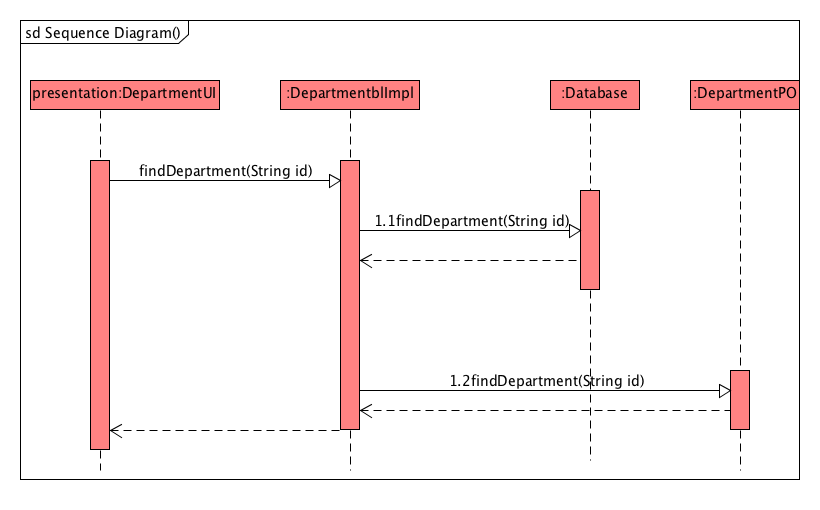
下图表示了物流管理系统中，当用户选择增加机构信息时，Departmentbl处理的相关对象的协作。



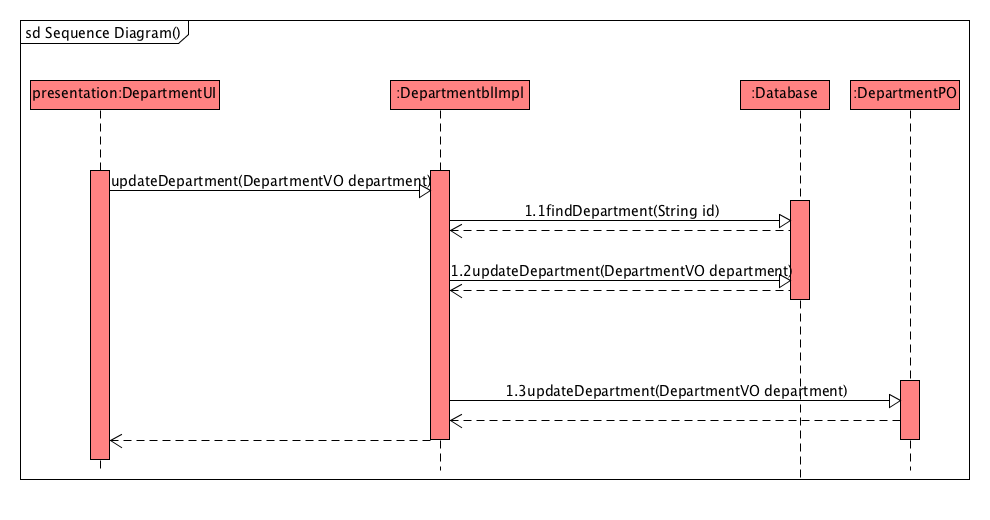
下图表示了物流管理系统中，当用户输入要删除的机构id时，Departmentbl处理的相关对象的协作。



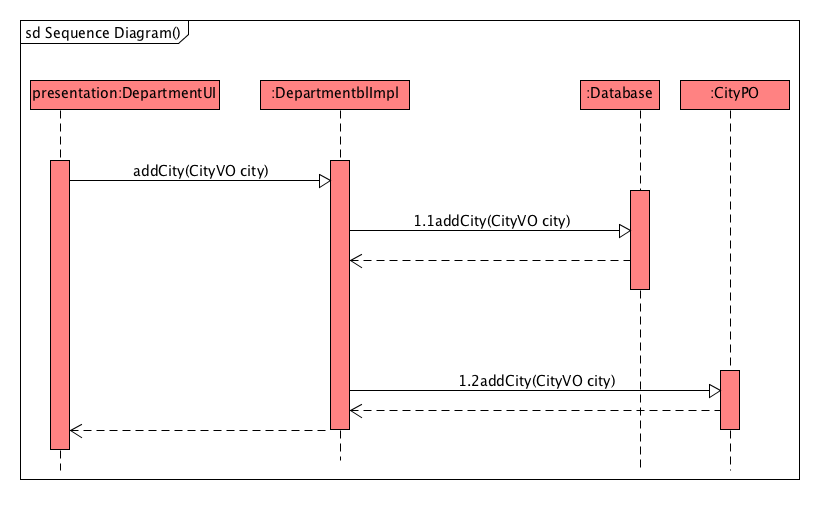
下图表示了物流管理系统中，当用户输入要查询的机构id时，Departmentbl处理的相关对象的协作。



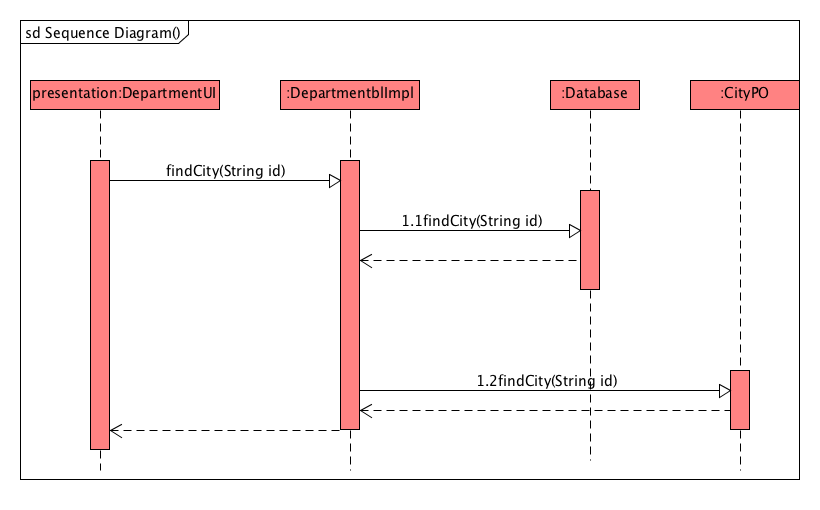
下图表示了物流管理系统中，当用户选择修改机构信息时，Departmentbl处理的相关对象的协作。



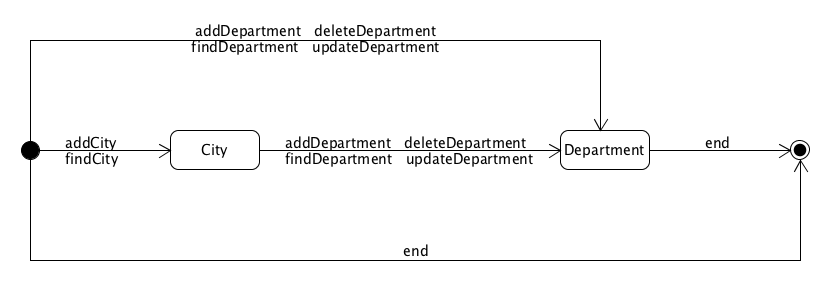
下图表示了物流管理系统中，当用户选择增加城市信息时，Departmentbl处理的相关对象的协作。



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



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



(5)业务逻辑层的设计原理

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

### 5.1.5Financebl模块

### 5.1.6Listbl模块

(1)模块概述

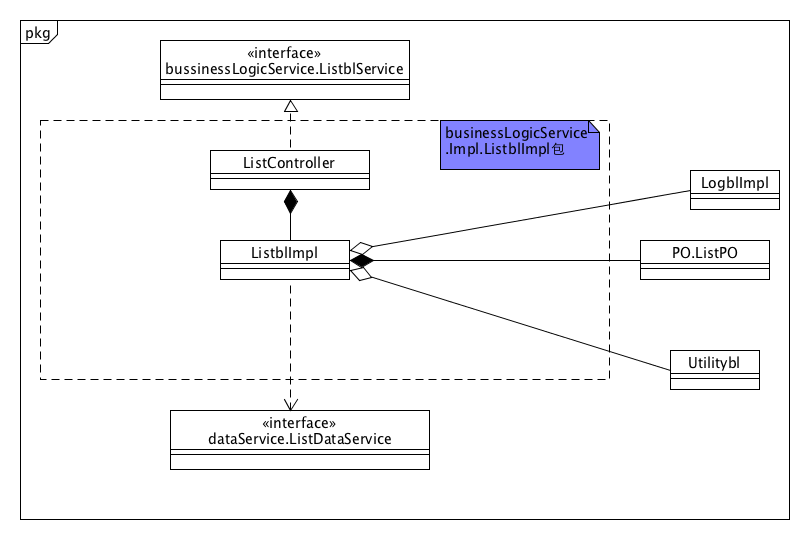
Listbl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

Listbl模块的职责和接口参见软件体系结构描述文档。

(2)整体结构

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

Listbl模块的设计如图所示。



Listbl模块的各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| LoginController | 负责实现对应于登陆界面所需要的服务 |
| ListController | 负责实现用户管理界面所需要的服务 |
| ListblImpl | 用户管理领域的模型对象，可以帮助完成用户管理界面所需要的服务 |

(3)模块内部类的接口规范

ListController和Listbl的接口规范如下表所示。

ListController的接口规范

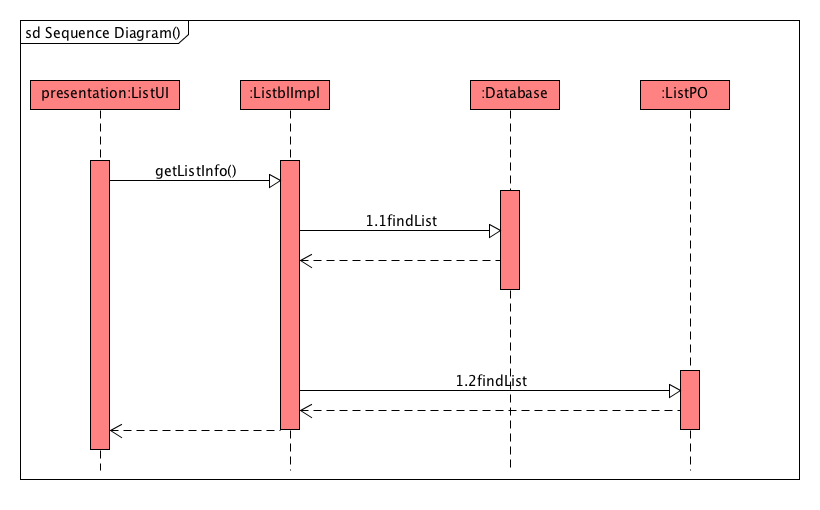
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ListController. getListInfo | 语法 | Public ArrayList<ListPO> getListInfo() |
| 前置条件 | 无 |
| 后置条件 | 查找是否存在未被审批的单据，返回所有未被审批的单据的引用 |
| ListController. pass | 语法 | Public Boolean changeList(ListVO List) |
| 前置条件 | 无 |
| 后置条件 | 根据id查找List，并修改相应的数据，如果修改成功返回true，否则返回false |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| List. getListInfo() | 得到所有单据信息 | |
| List. changeList(ListVO List) | 根据输入的单据信息修改单据内容与状态 | |

Listbl的接口规范

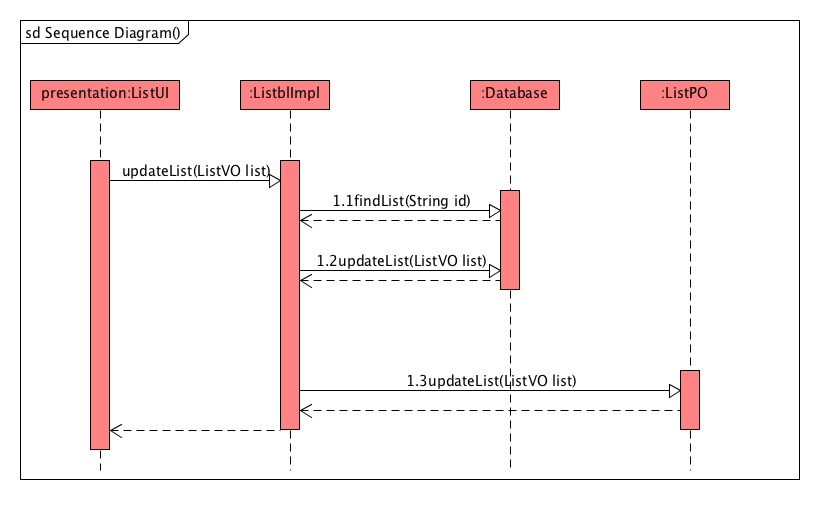
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| List.getListInfo | 语法 | Public ArrayList<ListPO> getListInfo() | |
| 前置条件 | 无 | |
| 后置条件 | 查找是否存在未被审批的单据，返回所有未被审批的单据的引用 | |
| List.pass | 语法 | Public Boolean changeList(ListVO List) | |
| 前置条件 | 无 | |
| 后置条件 | 根据id查找List，并修改相应的数据，如果修改成功返回true，否则返回false | |
| 需要的服务（需接口） | | | |
| 服务名 | 服务 | | |
| ListDataService  .updateList(long id，State state)：boolean | 查找该id的list单一持久化对象并将该list的状态改为传入的state | |
| ListDataService  .findList():List<ListPo> | 查找未被审批的单据并返回单据的单一持久化对象 | |

(4)业务逻辑层的动态模型

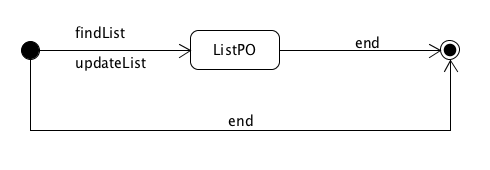
下图表示了物流管理系统中，当用户选择查看单据信息时，Listbl处理的相关对象的协作。



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



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



(5)业务逻辑层的设计原理

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

### 5.1.7Userbl模块

(1)模块概述

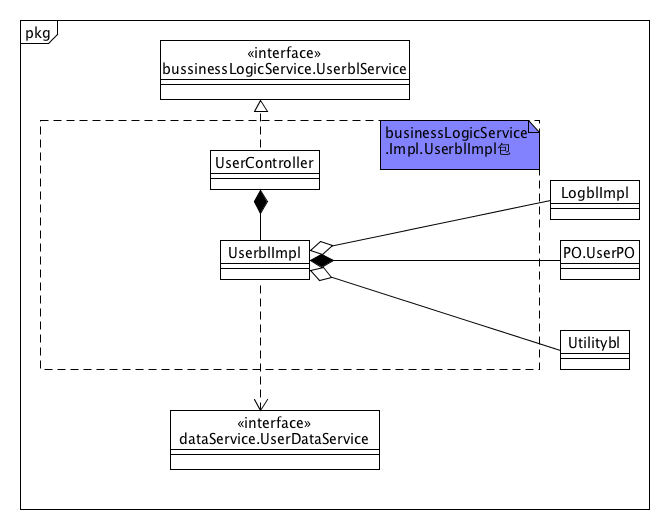
userbl模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

userbl模块的职责和接口参见软件体系结构描述文档。

(2)整体结构

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

Userbl模块的设计如图所示。



Userbl模块的各个类的职责如下表所示。

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

(3)模块内部类的接口规范

UserController和Userbl的接口规范如下表所示。

UserController的接口规范

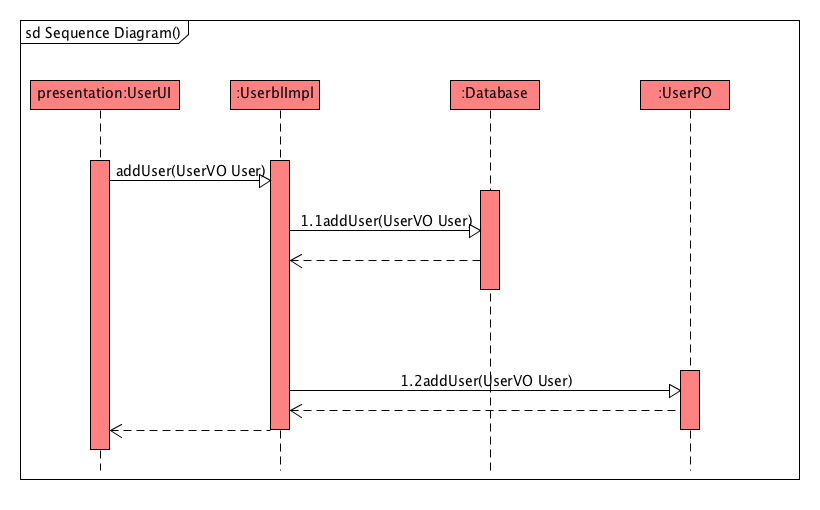
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserController.login | 语法 | Public String login(long id, String password) |
| 前置条件 | Password符合输入规则 |
| 后置条件 | 查找是否存在相应的User，根据输入的password返回输入的结果 |
| UserController.getUserInfo | 语法 | Public UserPO getUserInfo(long id) |
| 前置条件 | 无 |
| 后置条件 | 查找是否存在相应的user，如果存在返回相应的user信息 |
| UserController.deleteUser | 语法 | Public Boolean deleteUser(long id) |
| 前置条件 | 无 |
| 后置条件 | 查找是否存在相应的user， 如果删除成功返回true，否则返回false |
| UserController.changeUser | 语法 | Public Boolean changeUser(UserVO user) |
| 前置条件 | 无 |
| 后置条件 | 根据id查找user，并修改相应的数据，如果修改成功返回true，否则返回false |
| UserController.addUser | 语法 | Public Boolean addUser(UserVO user) |
| 前置条件 | 无 |
| 后置条件 | 如果增加成功返回true，否则返回false |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| User. login(long id, String password) | 登录管理用户界面 | |
| User.getUserInfo(long id) | 根据输入的id得到一个用户信息 | |
| User. deleteUser(long id) | 根据输入的id删除一个用户信息 | |
| User. changeUser(UserVO user) | 根据输入的信息修改一个用户信息 | |
| User. addUser(UserVO user) | 根据输入的信息增加一个用户信息 | |

Userbl的接口规范

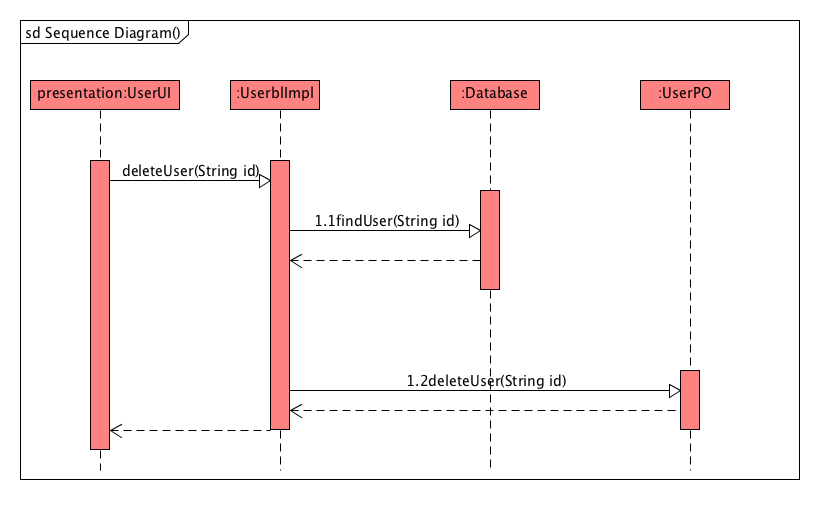
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| User.login | 语法 | Public String login(long id, String password) |
| 前置条件 | Password符合输入规则 |
| 后置条件 | 查找是否存在相应的User，根据输入的password返回输入的结果 |
| User.getUserInfo | 语法 | Public UserPO getUserInfo(long id) |
| 前置条件 | 无 |
| 后置条件 | 查找是否存在相应的user，如果存在返回相应的user信息 |
| User.deleteUser | 语法 | Public Boolean deleteUser(long id) |
| 前置条件 | 无 |
| 后置条件 | 查找是否存在相应的user， 如果删除成功返回true，否则返回false |
| User.changeUser | 语法 | Public Boolean changeUser(UserVO user) |
| 前置条件 | 无 |
| 后置条件 | 根据id查找user，并修改相应的数据，如果修改成功返回true，否则返回false |
| User.addUser | 语法 | Public Boolean addUser(UserVO user) |
| 前置条件 | 无 |
| 后置条件 | 如果增加成功返回true，否则返回false |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| UserDataService  .findUser(long id):UserPO | 根据输入的id返回一个用户持久化对象 | |
| UserDataService  .deleteUser(long id):boolean | 根据输入的id删除一个用户持久化对象 | |
| UserDataService  .updateUser(UserPo user):boolean | 根据输入的信息修改原有的用户持久化对象 | |
| UserDataService  .addUser(UserPo user):boolean | 根据输入的信息增加用户持久化对象 | |

(4)业务逻辑层的动态模型

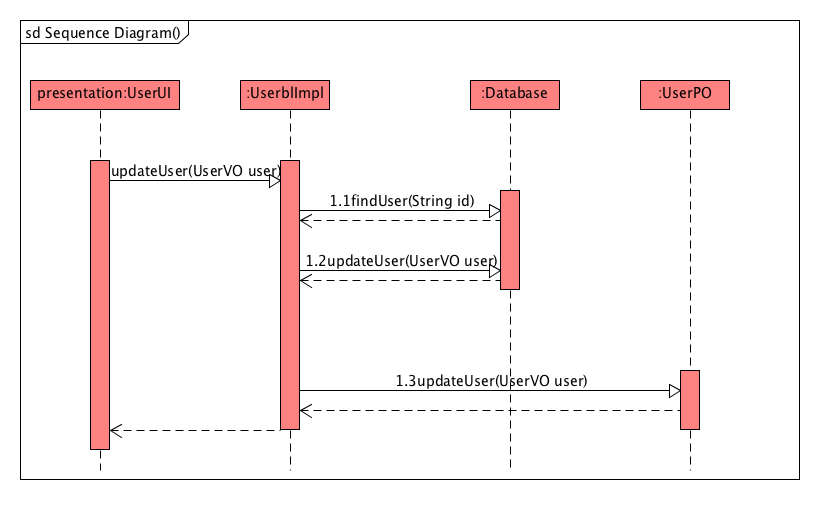
下图表示了物流管理系统中，当用户输入要增加的用户信息时，Userbl处理的相关对象的协作。



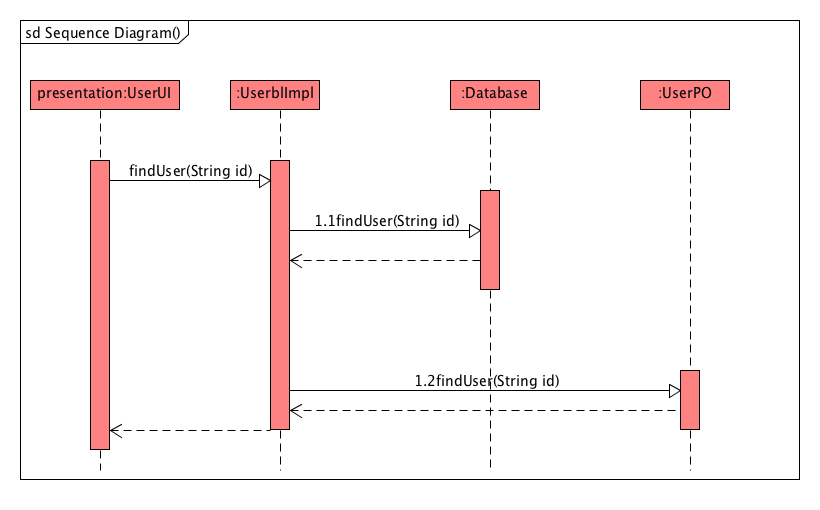
下图表示了物流管理系统中，当用户输入要删除的用户id时，Userbl处理的相关对象的协作。



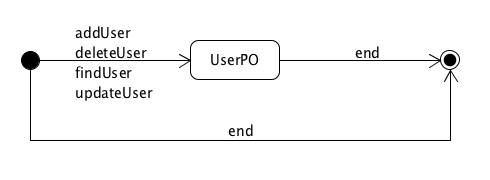
下图表示了物流管理系统中，当用户输入要修改的用户信息时，Userbl处理的相关对象的协作。



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



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



(5)业务逻辑层的设计原理

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

### 5.1.8Logbl模块

# 6.依赖视角

下图分别是客户端和服务器端各自的包之间的依赖关系。

