|  |
| --- |
| 互联网酒店管理系统  Hotel Property Management System |
| 软件详细设计描述文档 |
| V1.1 |

|  |
| --- |
| 焦立群、金灵益、黎奥邦、李恩铭  2016-10-29 |

更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| 修改人员 | 日期 | 变更原因 | 版本号 |
| 黎奥邦 | 2016-10-28 | 最初文档模板 | V1.0 |
| 黎奥邦 | 2016-10-29 | 初步完成文档，主要完成了业务逻辑层的编写 | V1.1 |

目录

[更新历史 - 1 -](#_Toc465544036)

[1.引言 - 3 -](#_Toc465544037)

[1.1编制目的 - 3 -](#_Toc465544038)

[1.2词汇表 - 3 -](#_Toc465544039)

[1.3参考资料 - 3 -](#_Toc465544040)

[2.产品概述 - 3 -](#_Toc465544041)

[3.体系结构设计描述 - 3 -](#_Toc465544042)

[4.结构视角 - 3 -](#_Toc465544043)

[4.1用户界面层的分解 - 3 -](#_Toc465544044)

[4.1.1UI层设计概要 - 3 -](#_Toc465544045)

[4.1.2UI层内部包的职责与划分 - 3 -](#_Toc465544046)

[4.2业务逻辑层的分解 - 4 -](#_Toc465544047)

[4.2.1 Orderbl模块 - 4 -](#_Toc465544048)

[4.2.2 UserInfobl模块 - 10 -](#_Toc465544049)

[4.2.3 Hotelbl模块 - 16 -](#_Toc465544050)

[4.2.4 Accountbl模块 - 21 -](#_Toc465544051)

[4.2.5 Promotionbl模块 - 27 -](#_Toc465544052)

[4.3数据层的分解 - 32 -](#_Toc465544053)

[5.依赖视角 - 32 -](#_Toc465544054)

1.引言

1.1编制目的

本报告详细完成对互联网酒店管理系统的详细设计，达到指导后续软件构造目的，同时实现和测试人员和用户的沟通。

本报告面向开发人员、测试人员及最终用户编写，是了解系统的导航。

1.2词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| HPMS | 互联网酒店管理系统 | Hotel Property Management System |
| Order | 订单 | 无 |
| UserInfo | 用户信息 | User Information |
| Hotel | 酒店 | 无 |
| Account | 账户 | 无 |
| Promotion | 促销策略 | 无 |

1.3参考资料

1. 互联网酒店管理系统HPMS用例文档\_V1.1正式版

2. 互联网酒店管理系统HPMS软件需求规格说明\_V1.0正式版

3. 互联网酒店管理系统HPMS软件体系结构描述文档

2.产品概述

参考互联网酒店管理系统用例文档和互联网酒店管理系统软件需求规格说明文档中对产品的概括描述。

3.体系结构设计描述

参考互联网酒店管理系统概要设计文档中对体系结构设计的概述。

4.结构视角

4.1用户界面层的分解

4.1.1UI层设计概要

4.1.2UI层内部包的职责与划分

4.2业务逻辑层的分解

4.2.1 Orderbl模块

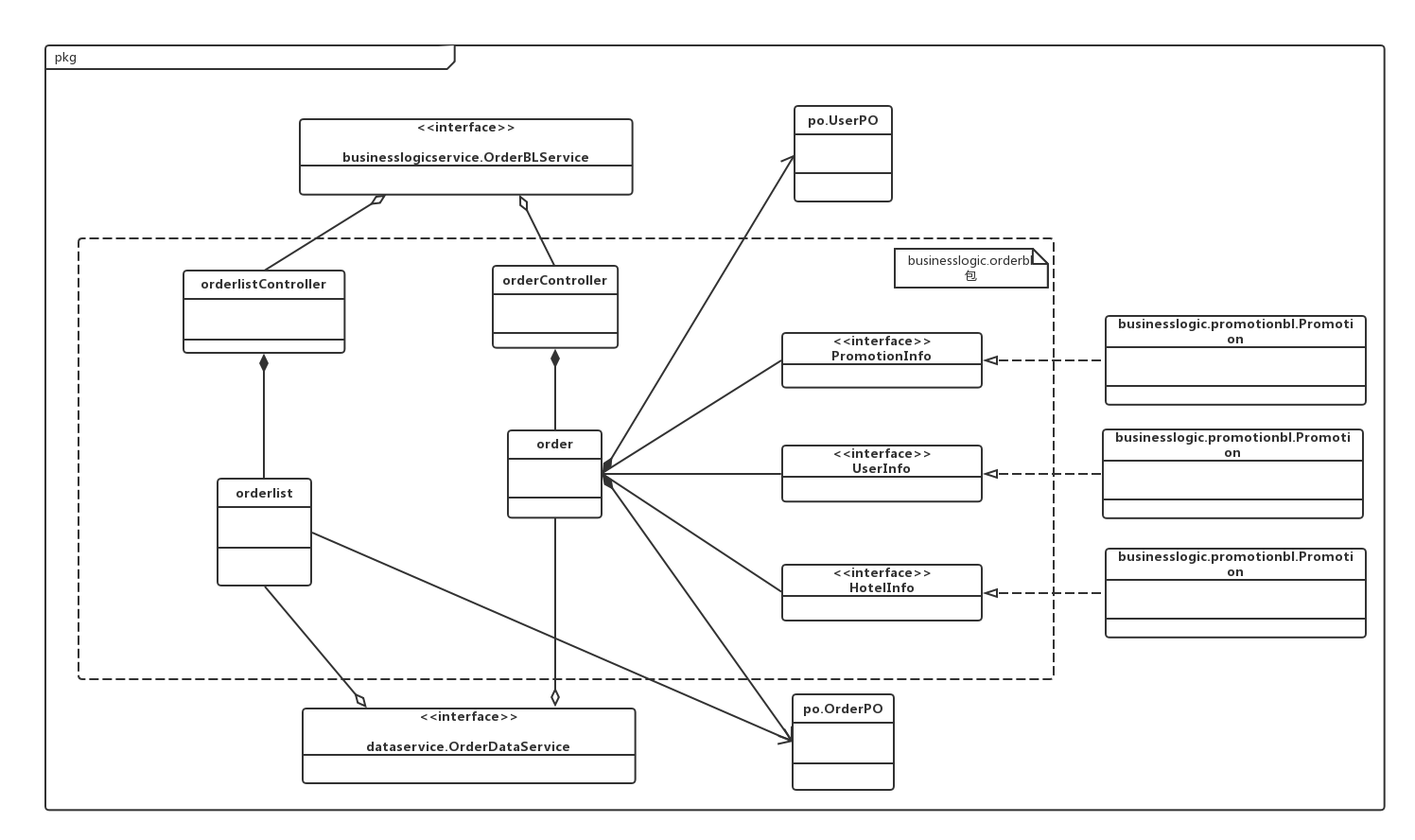
（1）模块概述

Orderbl模块承担的具体功能需求和非功能需求可参见需求规格说明文档；Orderbl模块的职责及接口参见体系结构设计文档。

（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。比如展示层和业务逻辑层之间，我们添加businesslogicservice.OrderBLService接口。业务逻辑层和数据层之间添加dataservice.OrderDataService接口。为了隔离业务逻辑职责和逻辑控制职责，我们添加了OrderController，这样OrderController会将对订单的逻辑业务处理委托给Order对象。OrderPO是作为订单的持久化对象被添加到设计模型中去的。

Order模块的设计如下图所示。



Order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 |
| OrderController | 负责实现订单管理所需要的服务 |
| Order | 订单的领域模型对象，拥有订单的信息，可以帮助完成订单管理所需要的服务 |
| OrderListController | 负责实现订单列表批量管理所需要的服务 |
| OrderList | 订单列表的数据 |

（3）模块内部类的接口规范

OrderController和Order的接口规范如下表所示。

**OrderController**的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| OrderController. getInfo | 语法 | Public ResultMessage getInfo(String orderId) | |
| 前置条件 | 创建一个Order对象 | |
| 后置条件 | 调用Order的方法 | |
| OrderController. submit | 语法 | Public ResultMessage submit(String orderId) | |
| 前置条件 | 创建一个Order对象 | |
| 后置条件 | 调用Order的方法 | |
| OrderController. cancel | 语法 | Public ResultMessage cancel(String orderId) | |
| 前置条件 | 创建一个Order对象 | |
| 后置条件 | 调用Order的方法 | |
| OrderController. execute | 语法 | Public ResultMessage execute(String orderId) | |
| 前置条件 | 创建一个Order对象 | |
| 后置条件 | 调用Order的方法 | |
| OrderController. recover | 语法 | Public ResultMessage recover(String orderId) | |
| 前置条件 | 创建一个Order对象 | |
| 后置条件 | 调用Order的方法 | |
| OrderController. revoke（网站营销人员撤销异常订单） | 语法 | Public ResultMessage revoke(String orderId) | |
| 前置条件 | 创建一个Order对象 | |
| 后置条件 | 调用Order的方法 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Order.getInfo(String orderId) | | | 显示此订单的信息 |
| Order.submit(String orderId) | | | 系统更新订单信息 |
| Order.cancel(String orderId) | | | 系统删除此订单的信息 |
| Order.execute(String orderId) | | | 系统将此订单置于正在执行状态 |
| Order.recover(String orderId) | | | 系统将此订单置于正在执行状态 |
| Order.revoke(String orderId) | | | 异常订单撤销，信用值恢复，记录撤销时间、信用值 |

**Order**的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Order.getInfo | 语法 | Public ResultMessage getInfo(String orderId) | |
| 前置条件 | 此订单存在 | |
| 后置条件 | 显示此订单的信息 | |
| Order.submit | 语法 | Public ResultMessage submit(String orderId) | |
| 前置条件 | 客户已经填写好此订单的完整信息 | |
| 后置条件 | 系统更新订单信息 | |
| Order.cancel | 语法 | Public ResultMessage cancel(String orderId) | |
| 前置条件 | 系统已更新此订单的信息 | |
| 后置条件 | 系统删除此订单的信息 | |
| Order.execute | 语法 | Public ResultMessage execute(String orderId) | |
| 前置条件 | 系统已经更新此订单的信息 | |
| 后置条件 | 系统将此订单置于正在执行状态 | |
| Order.recover | 语法 | Public ResultMessage recover(String orderId) | |
| 前置条件 | 系统已经更新此订单的信息并且顾客晚于规定入住时间到店 | |
| 后置条件 | 系统将此订单置于正在执行状态并恢复客户信用值 | |
| Order.revoke（网站营销人员撤销异常订单） | 语法 | Public ResultMessage revoke(String orderId) | |
| 前置条件 | 异常订单未被撤销 | |
| 后置条件 | 异常订单撤销，信用值恢复，记录撤销时间、信用值 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| OrderDataService.find(String id) | | | 根据ID进行查找单一持久化对象 |
| OrderDataService.find(String field, String value) | | | 根据字段名和值进行查找多个持久化对象 |
| OrderDataService.delete(OrderPO po) | | | 删除单一持久化对象 |
| OrderDataService.update(OrderPO po) | | | 更新单一持久化对象 |
| DatabaseFactory.getOrderDatabase | | | 得到Order数据库的服务的引用 |
| DataBaseFactory.insert(OrderPO po) | | | 在数据库中插入PO对象 |
| DataBaseFactory.delete (OrderPO po) | | | 在数据库中删除PO对象 |
| DataBaseFactory.delete(OrderPO po) | | | 在数据库中更新单一持久化对象 |

**OrderListController**的接口规范

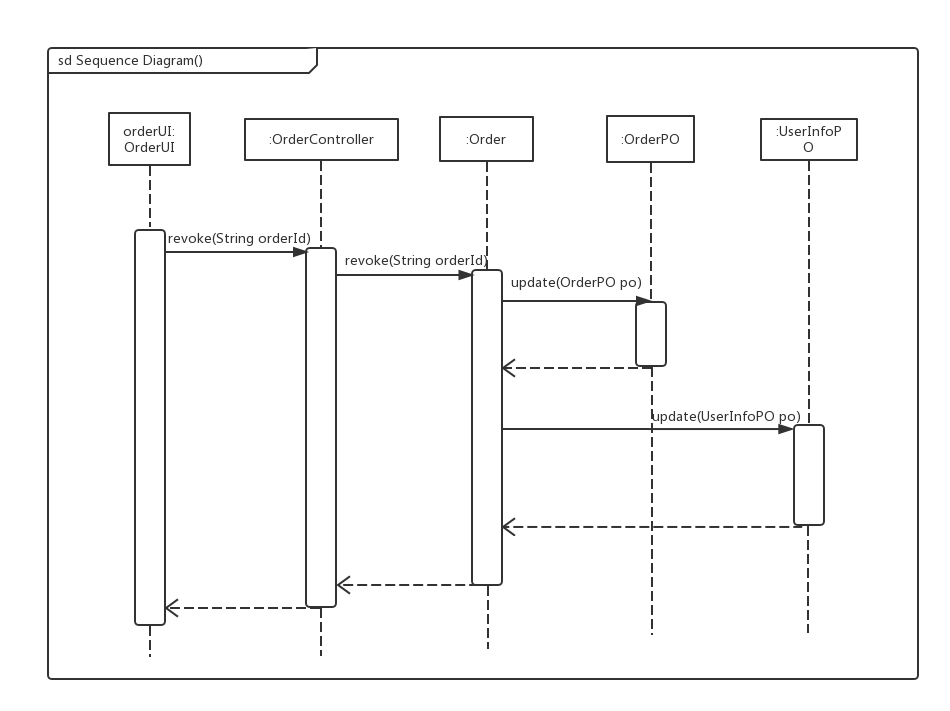
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| OrderListController.getInfo | 语法 | Public ArrayList<OrderVO> getInfo(String field, String value) | |
| 前置条件 | 创建一个OrderList对象 | |
| 后置条件 | 调用OrderList的方法 | |
| OrderListController.notExeAnalysis | 语法 | Public ResultMessage notExeAnalysis() | |
| 前置条件 | 创建一个OrderList对象 | |
| 后置条件 | 调用OrderList的方法 | |
| OrderListController.show | 语法 | Public ArrayList<OrderVO> show(String memberID) | |
| 前置条件 | 创建一个OrderList对象 | |
| 后置条件 | 调用OrderList的方法 | |
| OrderListController.showFinished | 语法 | public ArrayList<OrderVO> showFinished(String memberID) | |
| 前置条件 | 创建一个OrderList对象 | |
| 后置条件 | 调用OrderList的方法 | |
| OrderListController.showSubmitted | 语法 | public ArrayList<OrderVO> showSubmitted(String memberID) | |
| 前置条件 | 创建一个OrderList对象 | |
| 后置条件 | 调用OrderList的方法 | |
| OrderListController.showSubmitted | 语法 | public ArrayList<OrderVO> showSubmitted() | |
| 前置条件 | 创建一个OrderList对象 | |
| 后置条件 | 调用OrderList的方法 | |
| OrderListController.showFailed | 语法 | public ArrayList<OrderVO> showFailed(String memberID) | |
| 前置条件 | 创建一个OrderList对象 | |
| 后置条件 | 调用OrderList的方法 | |
| OrderListController.showFailed | 语法 | public ArrayList<OrderVO> showFailed() | |
| 前置条件 | 创建一个OrderList对象 | |
| 后置条件 | 调用OrderList的方法 | |
| OrderListController.showCanceled | 语法 | public ArrayList<OrderVO> showCanceled(String memberID) | |
| 前置条件 | 创建一个OrderList对象 | |
| 后置条件 | 调用OrderList的方法 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| OrderList.getInfo(String field, String value) | | | 根据字段名和值进行返回多个订单 |
| OrderList.notExeAnalysis() | | | 分析未执行订单并记录结果 |
| OrderListController.show(String memberID) | | | 根据客户编号返回所有订单 |
| OrderList.showFinished(String memberID) | | | 根据客户编号返回已执行订单列表 |
| OrderList.showSubmitted(String memberID) | | | 根据客户编号返回未执行订单列表 |
| OrderList. showSubmitted() | | | 返回未执行订单列表 |
| OrderList.showFailed(String memberID) | | | 根据客户编号返回异常订单列表 |
| OrderList.showFailed() | | | 返回异常订单列表 |
| OrderList.showCanceled(String memberID) | | | 根据客户编号返回已撤销订单列表 |

**OrderList**的接口规范

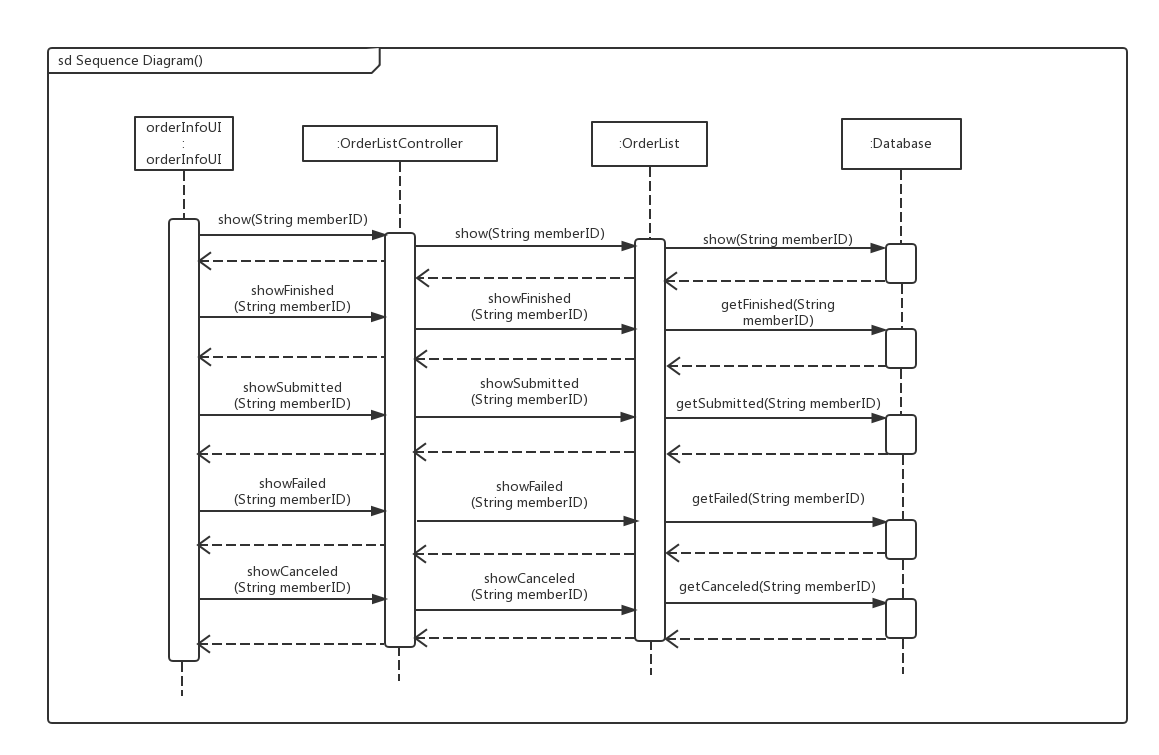
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| OrderList. getInfo | 语法 | Public ArrayList<OrderVO> getInfo(String field, String value) | |
| 前置条件 | 订单存在 | |
| 后置条件 | 根据字段名和值进行返回多个订单 | |
| OrderList. notExeAnalysis | 语法 | Public ResultMessage notExeAnalysis() | |
| 前置条件 | 无 | |
| 后置条件 | 系统显示分析结果 | |
| OrderList.show | 语法 | Public ArrayList<OrderVO> show(String memberID) | |
| 前置条件 | 客户查看订单列表 | |
| 后置条件 | 根据客户编号返回所有订单 | |
| OrderList.showFinished | 语法 | public ArrayList<OrderVO> showFinished(String memberID) | |
| 前置条件 | 客户查看已执行订单 | |
| 后置条件 | 根据客户编号返回已执行订单列表 | |
| OrderList.showSubmitted | 语法 | public ArrayList<OrderVO> showSubmitted(String memberID) | |
| 前置条件 | 客户查看未执行订单 | |
| 后置条件 | 根据客户编号返回未执行订单列表 | |
| OrderList.showSubmitted | 语法 | public ArrayList<OrderVO> showSubmitted() | |
| 前置条件 | 网站营销人员查看未执行订单 | |
| 后置条件 | 返回未执行订单列表 | |
| OrderList.showFailed | 语法 | public ArrayList<OrderVO> showFailed(String memberID) | |
| 前置条件 | 客户、网站营销人员查看异常订单 | |
| 后置条件 | 根据客户编号返回异常订单列表 | |
| OrderList.showFailed | 语法 | public ArrayList<OrderVO> showFailed() | |
| 前置条件 | 网站营销人员查看异常订单 | |
| 后置条件 | 返回异常订单列表 | |
| OrderList.showCanceled | 语法 | public ArrayList<OrderVO> showCanceled(String memberID) | |
| 前置条件 | 客户查看已撤销订单 | |
| 后置条件 | 根据客户编号返回已撤销订单列表 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| DatabaseFactory. getOrderDatabase | | | 得到Order数据库的服务的引用 |
| OrderDataService.find(String field, String value) | | | 根据字段名和值进行查找多个订单 |
| OrderDataService.show(String memberID) | | | 根据客户编号返回所有订单 |
| OrderDataService.getFinished(String memberID) | | | 根据客户编号返回已执行订单列表 |
| OrderDataService.getSubmitted(String memberID) | | | 根据客户编号返回未执行订单列表 |
| OrderDataService.getSubmitted() | | | 返回未执行订单列表 |
| OrderDataService.getFailed(String memberID) | | | 根据客户编号返回异常订单列表 |
| OrderDataService.getFailed() | | | 返回异常订单列表 |
| OrderDataService.getCanceled(String memberID) | | | 根据客户编号返回已撤销订单列表 |

（4）业务逻辑层的动态模型

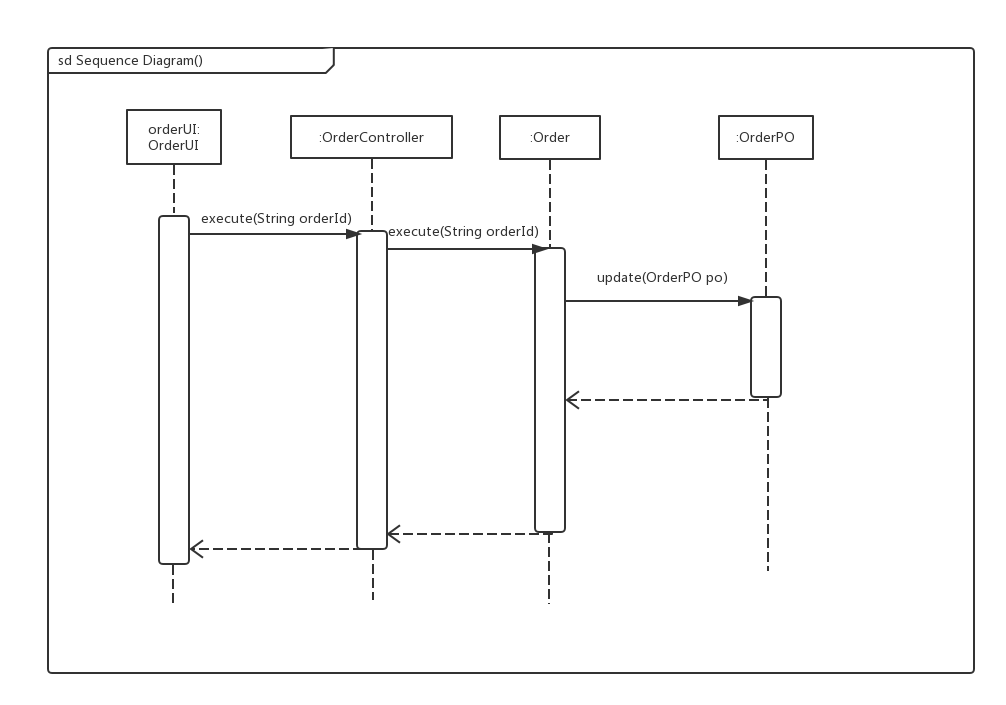
如图表明了当网站营销人员想要撤销异常订单时，订单业务逻辑处理的相关对象之间的协作。



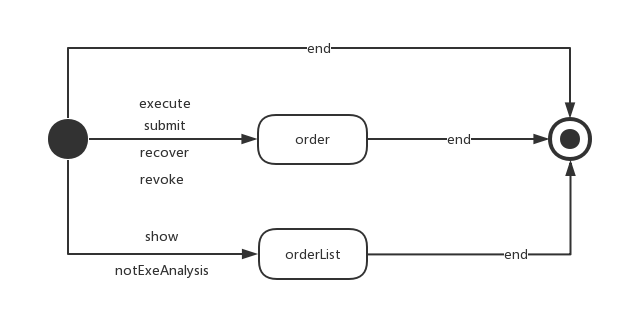
如图表明了当客户想要查看历史订单时，订单业务逻辑处理的相关对象之间的协作。



如图表明了当酒店工作人员执行订单时，订单业务逻辑处理的相关对象之间的协作。



如图所示的状态图描述了order对象的生存期间的状态序列、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领域对象。

4.2.2 UserInfobl模块

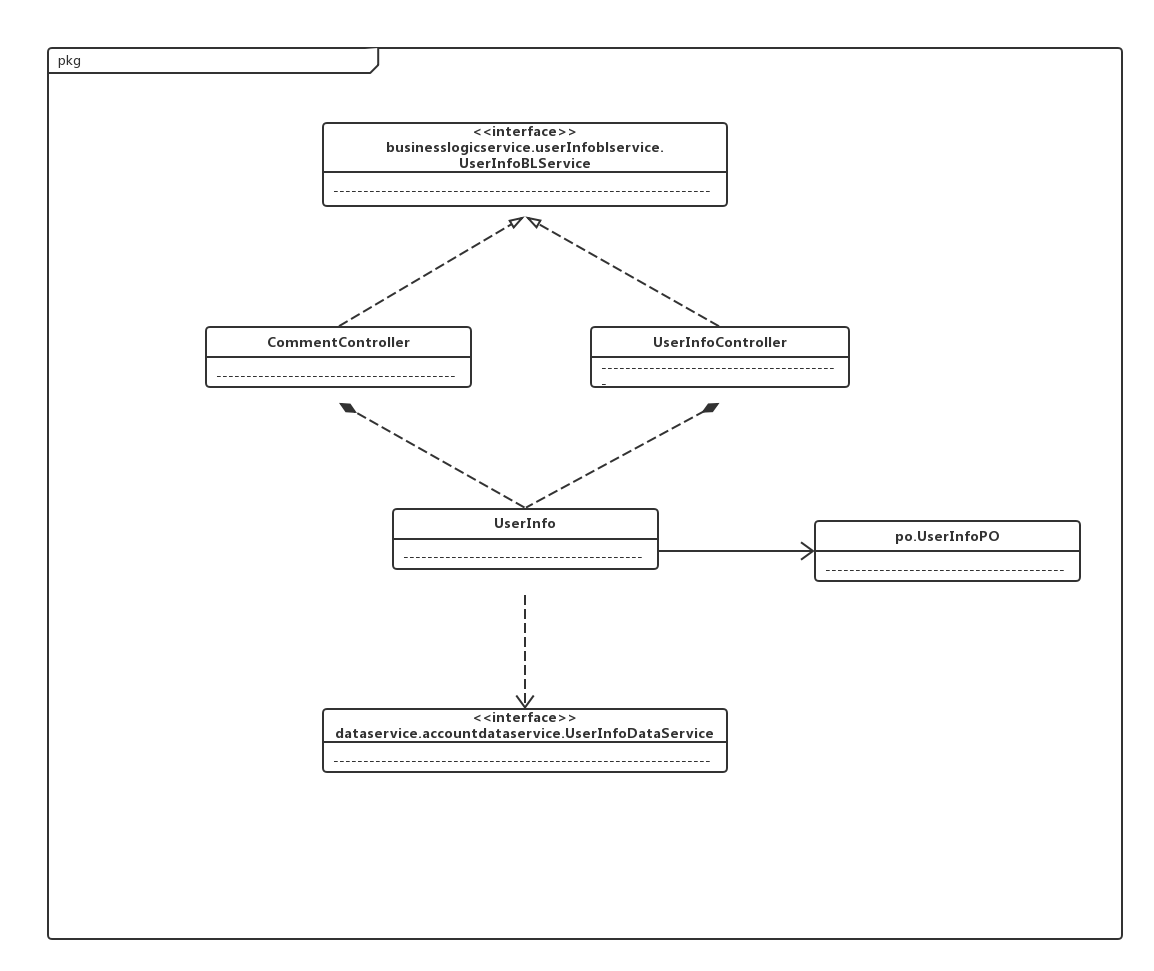
（1）模块概述

UserInfobl模块负责实现对客户的账号信息的管理，客户可以对自己的账号信息进行查询，酒店可以对每一个客户所有的信息进行查看。具体功能需求和非功能需求可参见需求规格说明文档和体系结构设计文档。

（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。比如展示层和业务逻辑层之间，我们添加bussinesslogicservice.userInfoBLService接口。业务逻辑层和数据层之间添加dataservice.userInfoDataService接口。为了隔离业务逻辑职责和逻辑控制职责，我们增加了UserInfoController，这样UserInfoController会将对账号信息的逻辑业务处理委托给UserInfo对象。UserInfoPO是作为账号信息的持久化对象被添加到设计模型中去的。

userInfobl模块的设计如下图所示



userInfobl模块各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| **CreditController** | 负责实现对应于客户账号信用充值界面所需要的服务 |
| **CommentController** | 负责实现对应于客户对酒店评分与文字评价界面所需要的服务 |
| **UserInfoController** | 负责实现对应于客户账号信息管理界面所需要的服务 |
| **UserInfo** | 客户信息的领域模型对象，拥有客户账号的信息，可以帮助完成账号查询、信用充值和酒店评价界面所需要的服务 |

（3）模块内部类的接口规范

**CreditController**的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CreditController.chargeCredit | 语法 | public ResultMessage modifyUserInfo(LIMIT limit, UserVO vo); |
| 前置条件 | vo合法 |
| 后置条件 | 调用UserInfo的对应方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| UserInfo.chargeCredit  (LIMIT limit, UserVO vo); | 对客户信用进行充值 | |

**CommentController** 的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CommentController.comment | 语法 | public ResultMessage comment(String hotelId, String comment) |
| 前置条件 | hotelId合法 |
| 后置条件 | 调用Hotel的文字评价方法 |
| CommentController.score | 语法 | public ResultMessage score(String hotelId, SCORE score) |
| 前置条件 | hotelId合法 |
| 后置条件 | 调用Hotel的等级评分方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| Hotel.comment  (String hotelId, String comment) | 返回评价成功/失败信息，持久化更新涉及的领域对象的数据 | |
| Hotel.score (String hotelId, SCORE score) | 返回评分成功/失败信息，持久化更新涉及的领域对象的数据 | |

**UserInfoController**的接口规范

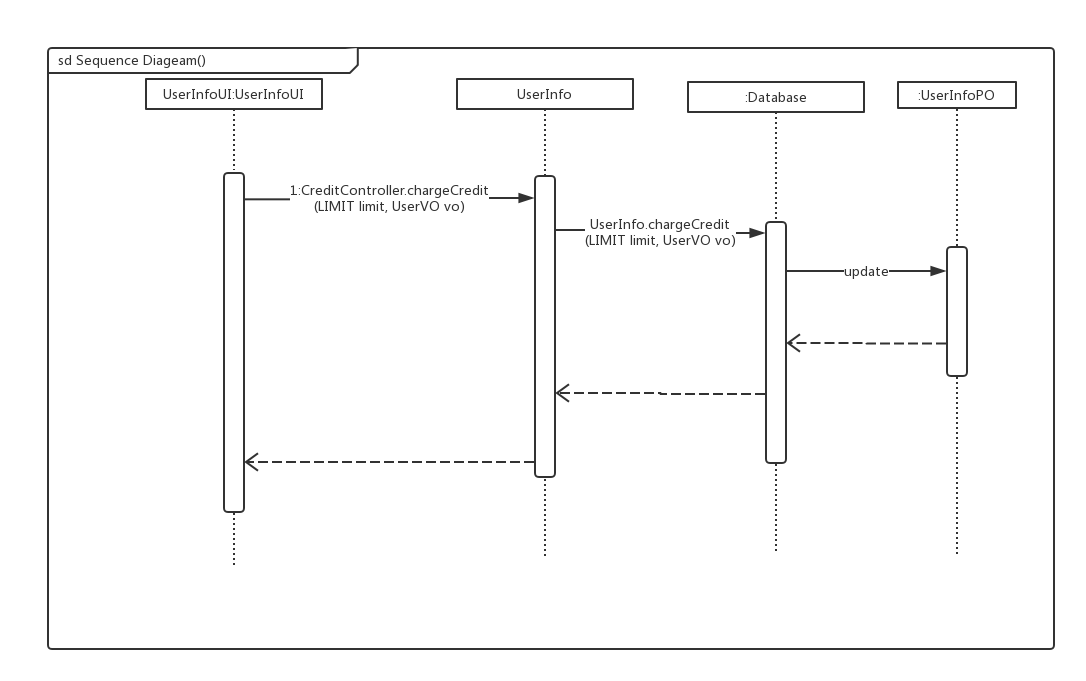
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserInfoController.getInfo | 语法 | Public UserInfoVO getInfo(String userId) |
| 前置条件 | userId合法,用于用户,酒店工作人员 |
| 后置条件 | 调用UserInfo的方法 |
| UserInfoController.findUser | 语法 | public UserInfoVO findUser(String userId) |
| 前置条件 | userId合法,用于网站管理人员 |
| 后置条件 | 调用UserInfo的方法 |
| UserInfoController.checkInfo | 语法 | Public ResultMessage checkInfo(String info) |
| 前置条件 | 无 |
| 后置条件 | 调用UserInfo的方法 |
| UserInfoController.modifyUserInfo | 语法 | public ResultMessage modifyUserInfo(LIMIT limit, UserVO vo) |
| 前置条件 | vo合法 |
| 后置条件 | 调用UserInfo的方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| UserInfo.getInfo(String userId) | 用于用户,酒店工作人员查看其个人基本信息 | |
| UserInfo.findUser(String userId) | 用于网站管理人员查看任何人的基本信息 | |
| UserInfo.checkInfo(String info) | 用于检查输入 | |
| UserInfo.modifyUserInfo(LIMIT limit, UserVO vo) | 用户用户,酒店工作人员修改个人基本信息 | |

**UserInfo** 的接口规范

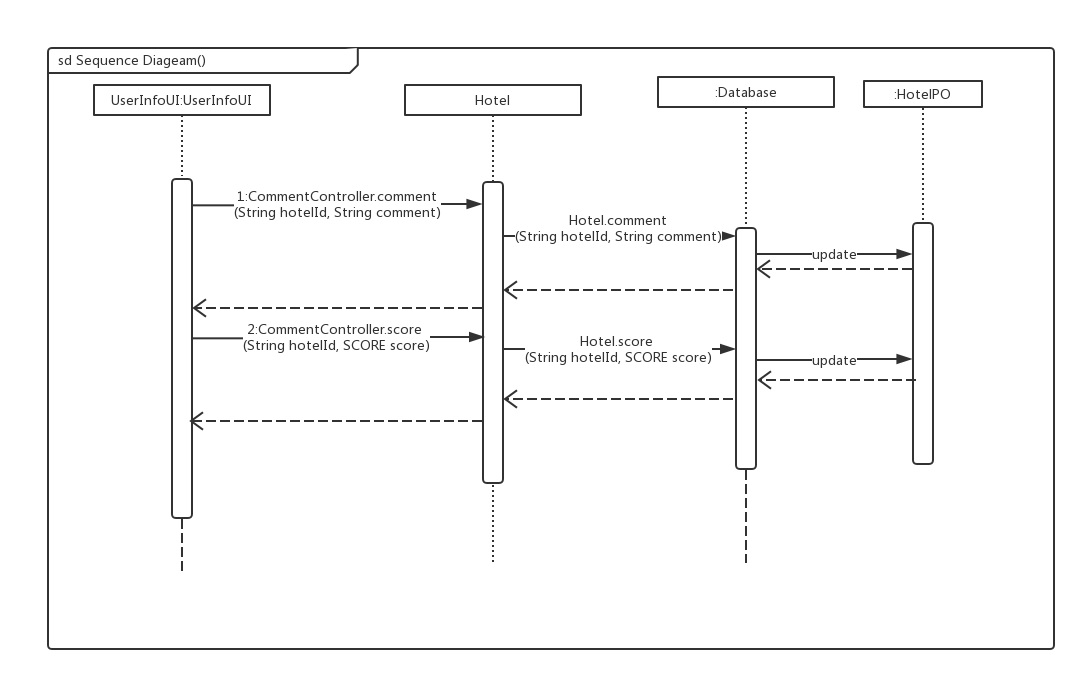
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| UserInfo.getInfo | 语法 | | Public UserInfoVO getInfo(String userId) |
| 前置条件 | | userId合法,用于用户,酒店工作人员 |
| 后置条件 | | 返回用户信息 |
| UserInfo.findUser | 语法 | | public UserInfoVO findUser(String userId) |
| 前置条件 | | userId合法,用于网站管理人员 |
| 后置条件 | | 返回用户的详细信息 |
| UserInfo.checkInfo | 语法 | | Public ResultMessage checkInfo(String info) |
| 前置条件 | | 无 |
| 后置条件 | | 返回检查结果 |
| UserInfo. modifyUserInfo | 语法 | | public ResultMessage modifyUserInfo(LIMIT limit, UserVO vo) |
| 前置条件 | | vo合法 |
| 后置条件 | | 返回修改成功/失败信息，持久化更新所涉及的领域对象的数据 |
| UserInfo.chargeCredit | 语法 | | public ResultMessage chargeCredit(LIMIT limit, UserVO vo); |
| 前置条件 | | vo合法 |
| 后置条件 | | 对客户信用进行充值 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| UserInfoDataService.update(AccountPO po) | | 在数据库中更新单一持久化对象 | |
| UserInfoDataService.find(String name) | | 根据用户名查找单一持久化对象 | |

（4）业务逻辑层的动态模型

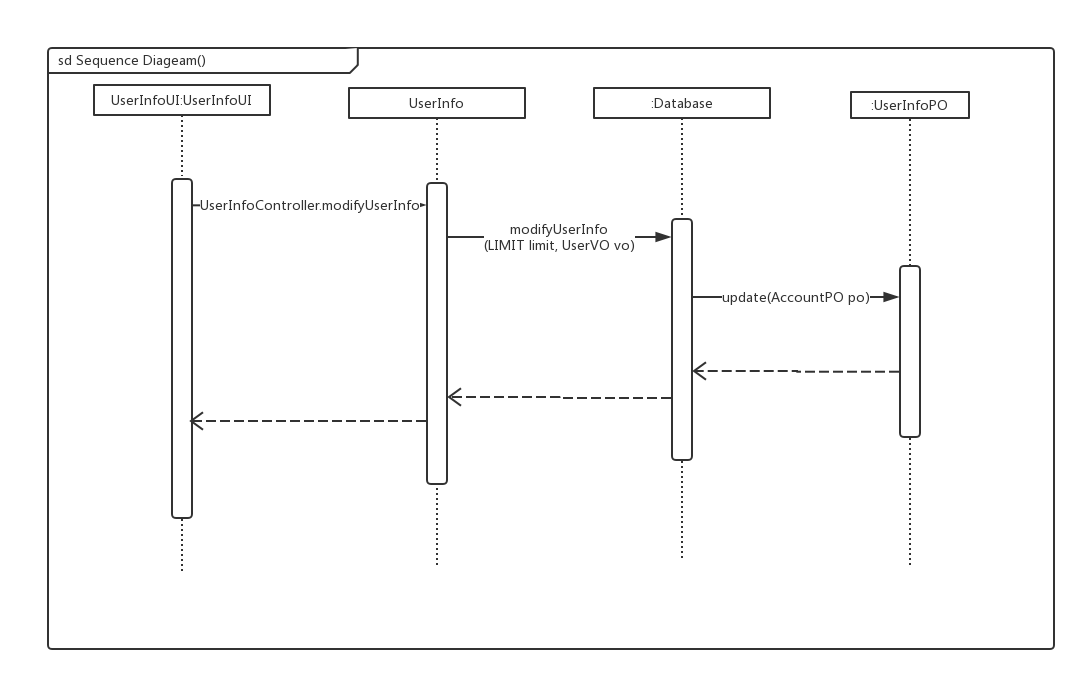
如图表明了当客户进行信用充值账号，用户信息业务逻辑处理的相关对象之间的协作。



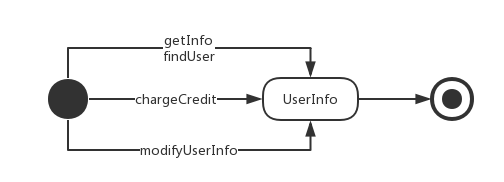
如图表明了当客户想要进行酒店评价，用户信息逻辑处理的相关对象之间的协作。



如图表明了当客户想要修改个人基本信息的时候，用户信息业务逻辑处理的相关对象之间的协作。



如图所示的状态图描述了UserInfo对象的生存期间的状态序列、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领域对象。

4.2.3 Hotelbl模块

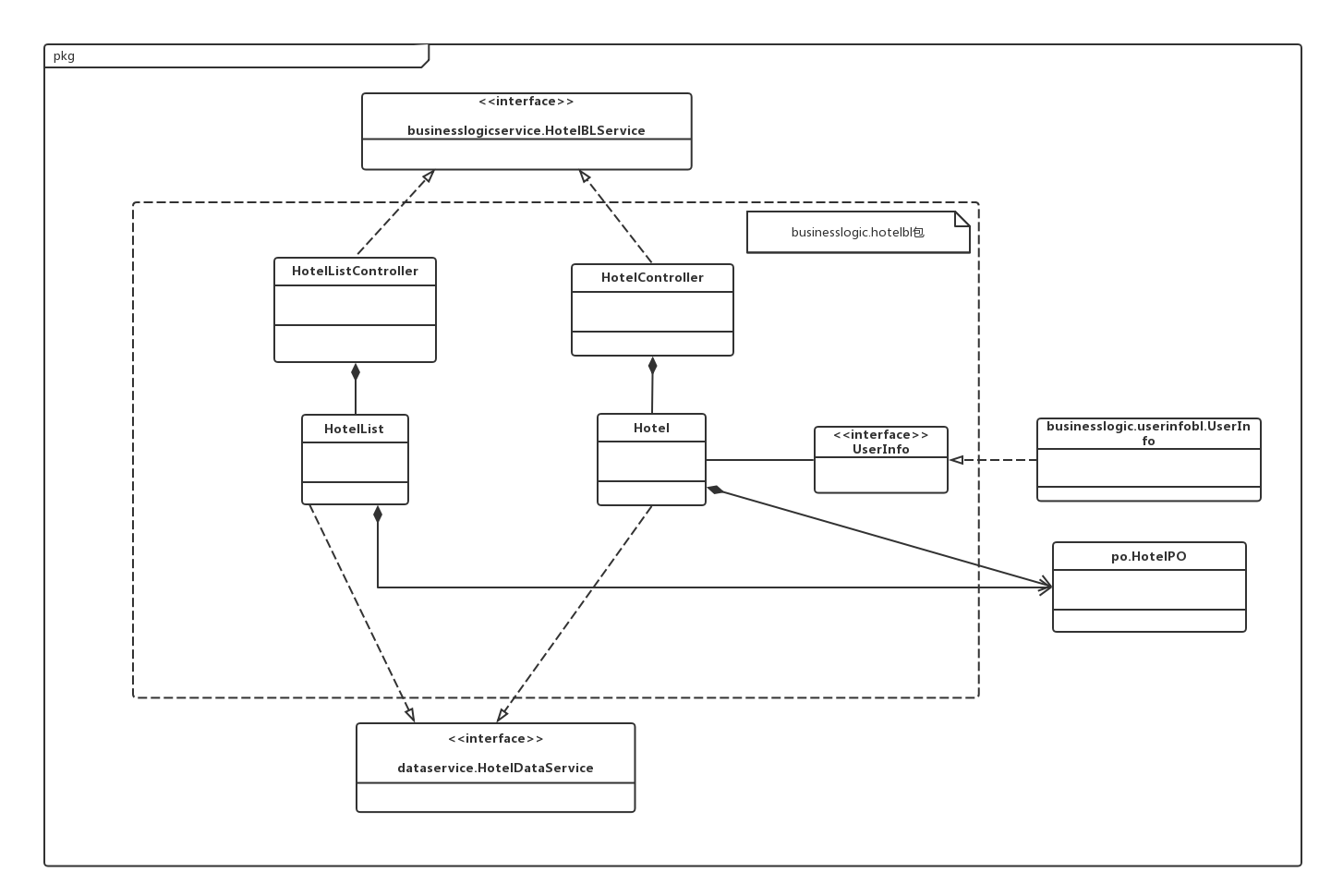
（1）模块概述

Hotelbl模块负责酒店的管理，查找酒店，显示酒店信息，酒店分类、排序，修改酒店。具体功能需求和非功能需求可参见需求规格说明文档和体系结构设计文档。

（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。比如展示层和业务逻辑层之间，我们添加businesslogicservice.HotelBLService接口。业务逻辑层和数据层之间添加dataservice.HotelDataService接口。为了隔离业务逻辑职责和逻辑控制职责，我们添加了HotelController，这样HotelController会将对酒店的逻辑业务处理委托给Hotel对象。HotelPO是作为酒店的持久化对象被添加到设计模型中去的。

Hotelbl模块的设计如图



Hotelbl模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| LoginController | 负责实现对应于登录界面所需要的服务 |
| HotelController | 负责实现酒店管理所需要的服务 |
| Hotel | 酒店的领域模型对象，拥有酒店的信息，可以帮助完成酒店管理所需要的服务 |
| HotelListController | 负责实现酒店列表管理所需要的服务 |
| HotelList | 酒店列表的数据 |

（3）模块内部类的接口规范

**HotelController**的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| HotelController.showHotelInfo | 语法 | public ResultMessage showHotelInfo(String id) | |
| 前置条件 | 创建一个Hotel对象 | |
| 后置条件 | 调用Hotel的方法 | |
| HotelController.getReadyRoom | 语法 | public int getReadyRoom() | |
| 前置条件 | 创建一个Hotel对象 | |
| 后置条件 | 调用Hotel的方法 | |
| HotelController.comment | 语法 | public void comment(String hotelId,String comment) | |
| 前置条件 | 创建一个Hotel对象 | |
| 后置条件 | 调用Hotel的方法 | |
| HotelController.getComment | 语法 | public ArrayList<String> getComment() | |
| 前置条件 | 创建一个Hotel对象 | |
| 后置条件 | 调用Hotel的方法 | |
| HotelController.givePoStrings | 语法 | public void givePoStrings(double poString) | |
| 前置条件 | 创建一个Hotel对象 | |
| 后置条件 | 调用Hotel的方法 | |
| HotelController.getPoStrings | 语法 | public double getPoStrings() | |
| 前置条件 | 创建一个Hotel对象 | |
| 后置条件 | 调用Hotel的方法 | |
| HotelController.getBusiness | 语法 | public String getBusiness() | |
| 前置条件 | 创建一个Hotel对象 | |
| 后置条件 | 调用Hotel的方法 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Hotel.showHotelInfo(String id) | | | 返回酒店的具体信息 |
| Hotel.getReadyRoom() | | | 返回空余房间数量 |
| Hotel.comment(String comment) | | | 记录客户评论 |
| Hotel.getComment() | | | 返回酒店所有评论 |
| Hotel.givePoStrings(double poString) | | | 计算并更新酒店评分 |
| Hotel.getPoStrings() | | | 返回酒店评分 |
| Hotel.getBusiness() | | | 返回酒店所属商圈 |

**Hotel**的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Hotel.showHotelInfo | 语法 | public ResultMessage showHotelInfo(String id) |
| 前置条件 | 客户选择查看酒店信息 |
| 后置条件 | 返回酒店的具体信息 |
| Hotel.getReadyRoom | 语法 | public int getReadyRoom() |
| 前置条件 | 已计算空余的酒店房间数量 |
| 后置条件 | 返回空余房间数量 |
| Hotel.comment | 语法 | public void comment(String hotelId,String comment) |
| 前置条件 | 客户进行酒店文字评价 |
| 后置条件 | 记录该评论 |
| Hotel.getComment | 语法 | public ArrayList<String> getComment() |
| 前置条件 | 客户查看酒店文字评论 |
| 后置条件 | 返回酒店所有评论 |
| Hotel.givePoStrings | 语法 | public void givePoStrings(double poString) |
| 前置条件 | 客户已进行酒店评价 |
| 后置条件 | 计算酒店的评分 |
| Hotel.getPoStrings | 语法 | public double getPoStrings() |
| 前置条件 | 客户查看酒店评分但未评价 |
| 后置条件 | 返回酒店评分 |
| Hotel.getBusiness | 语法 | public String getBusiness() |
| 前置条件 | 查看酒店的商圈 |
| 后置条件 | 返回酒店商圈 |
| 需要的服务（需接口） | | |
| 服务名 | | 服务 |
| DatabaseFactory.getHotelDatabase | | 得到Hotel数据库的服务的引用 |
| HotelDataServiceMySqlImpl.update(HotelPO po) | | 在数据库中更新一个持久化对象 |
| HotelDataService.update(HotelPO po) | | 更新一个持久化对象 |
| HotelDataService.find(String id) | | 按ID进行查找返回相应的HotelPO结果 |
| HotelDataService.getReadyRoom() | | 返回酒店空余房间数 |
| HotelDataService.getBusiness() | | 返回酒店所属商圈 |
| HotelDataService.getPoStrings() | | 返回酒店评分 |
| HotelDataService.getComment() | | 返回酒店文字评论 |

**HotelListController**的接口规范

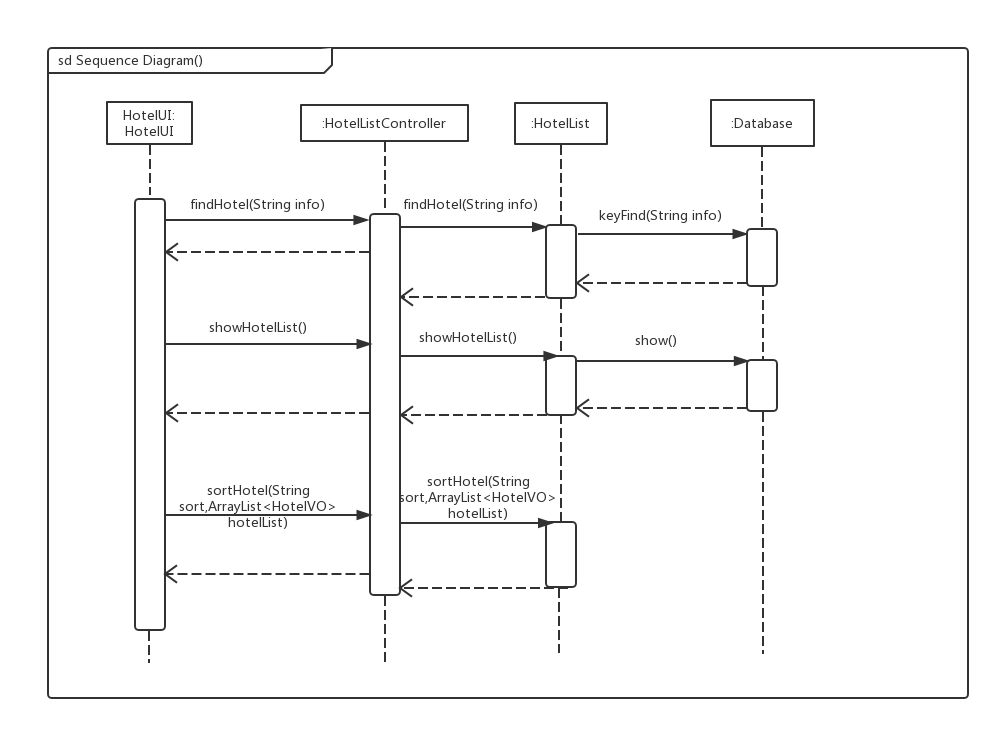
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| HotelListController.showHotelList | 语法 | public Arraylist<HotelVO>showHotelList() | |
| 前置条件 | 创建一个HotelList对象 | |
| 后置条件 | 调用HotelList的方法 | |
| HotelListController.sortHotel | 语法 | Public Arraylist<HotelVO> sortHotel(String sort,ArrayList<HotelVO> hotelList) | |
| 前置条件 | 创建一个HotelList对象 | |
| 后置条件 | 调用HotelList的方法 | |
| HotelListController.findHotel | 语法 | public ResultMessage findHotel(String info) | |
| 前置条件 | 创建一个HotelList对象 | |
| 后置条件 | 调用HotelList的方法 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| Hotel.showHotelList() | | | 返回所有的酒店列表 |
| Hotel.sortHotel(String sort,ArrayList<HotelVO> hotelList) | | | 系统按照某个分类对当前列表进行排序 |
| Hotel.findHotel(String info) | | | 查找到酒店并返回具体信息 |

**HotelList**的接口规范

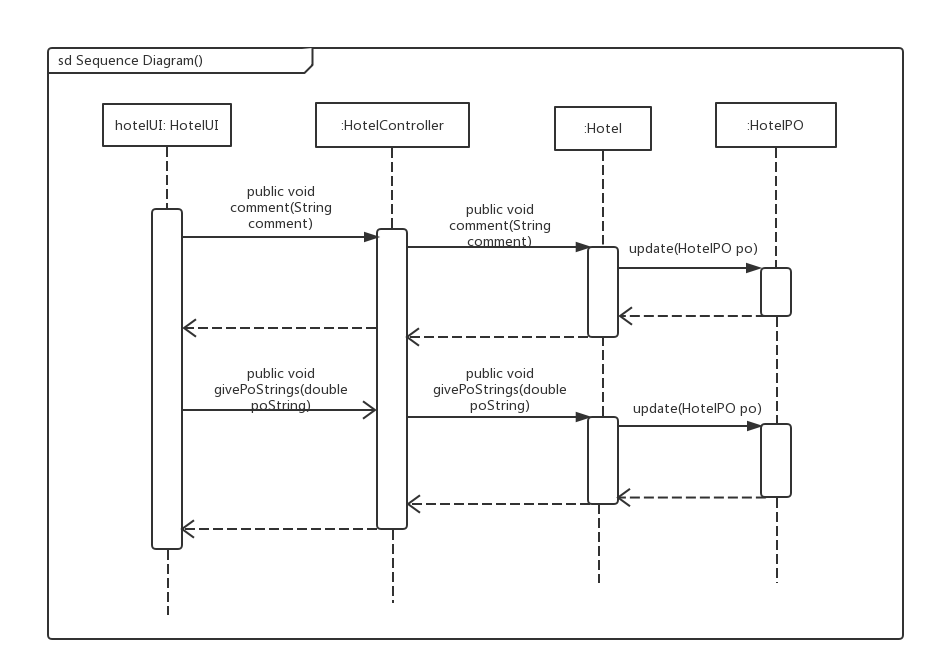
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HotelList.showHotelList | 语法 | public Arraylist<HotelVO>showHotelList() |
| 前置条件 | 创建一个HotelList对象 |
| 后置条件 | 调用HotelList的方法 |
| HotelList.sortHotel | 语法 | Public Arraylist<HotelVO> sortHotel(String sort,ArrayList<HotelVO> hotelList) |
| 前置条件 | 创建一个HotelList对象 |
| 后置条件 | 调用HotelList的方法 |
| HotelList.findHotel | 语法 | public ResultMessage findHotel(String info) |
| 前置条件 | 创建一个HotelList对象 |
| 后置条件 | 调用HotelList的方法 |
| 需要的服务（需接口） | | |
| 服务名 | | 服务 |
| DatabaseFactory.getHotelDatabase | | 得到Hotel数据库的服务的引用 |
| HotelDataService.show() | | 返回所有酒店数据 |
| HotelDataService.keyFind(String info) | | 按关键字进行查找返回相应的所有结果 |

（4）业务逻辑层的动态模型

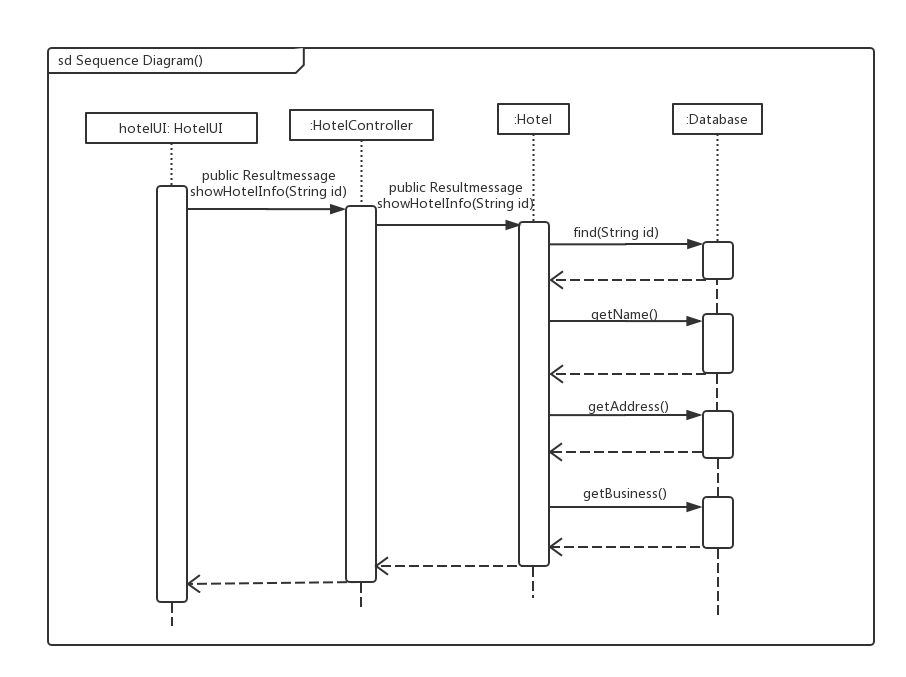
如图表明了客户想要进行酒店搜索时，酒店业务逻辑处理的相关对象之间的协作。



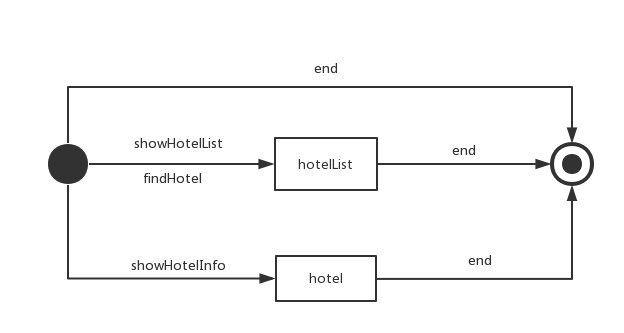
如图表明了客户在进行酒店评价时，酒店业务逻辑处理的相关对象之间的协作。



如图表明了客户在进行查看酒店信息时，酒店业务逻辑处理的相关对象之间的协作。



如图所示的状态图描述了Hotel对象的生存期间的状态序列、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领域对象。

4.2.4 Accountbl模块

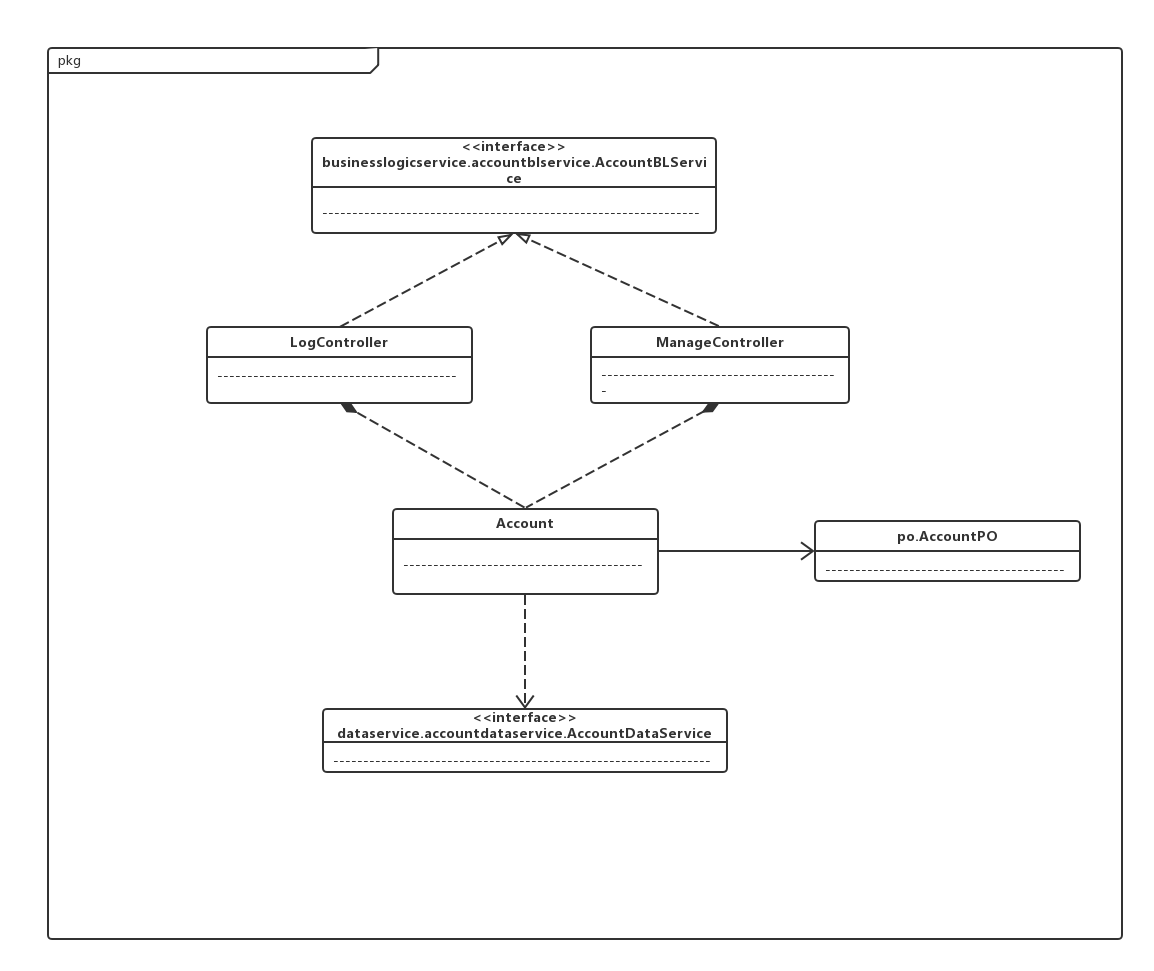
（1）模块概述

accountbl模块负责实现对四种类型用户（客户、酒店工作人员、酒店营销人员、酒店管理人员）账号的管理，对账号进行注册、查找、增加、删除和登入、登出验证，对账号密码进行修改。具体功能需求和非功能需求可参见需求规格说明文档和体系结构设计文档。

（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。比如展示层和业务逻辑层之间，我们添加bussinesslogicservice.accountBLService接口。业务逻辑层和数据层之间添加dataservice.accountDataService接口。为了隔离业务逻辑职责和逻辑控制职责，我们增加了AccountController，这样AccountController会将对账号的逻辑业务处理委托给Account对象。AccountPO是作为账号的持久化对象被添加到设计模型中去的。

accountbl模块的设计如图



accountbl模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| **LogController** | 负责实现对应于账号登录、登出和注册界面所需要的服务 |
| **ManageController** | 负责实现账号管理界面所需要的服务 |
| **Account** | 账号的领域模型对象，拥有账号的账号名和密码，可以帮助完成账号管理和登入登出界面所需要的服务 |

（3）模块内部类的接口规范

**LogController**的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| LogController.Login | 语法 | Public ResultMessage login(String memberName,String password); |
| 前置条件 | 已经创建一个Account领域对象，并且客户端尚未登录账号 |
| 后置条件 | 调用Account的对应方法 |
| logController.Logout | 语法 | Public ResultMessage logout(String memberName); |
| 前置条件 | 客户端处于已登录账号的状态 |
| 后置条件 | 调用Account的对应方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| Account.Login(String memberName,String password); | 用户登入 | |
| Account.Logout(String memberName); | 用户登出 | |

**ManageController** 的接口规范

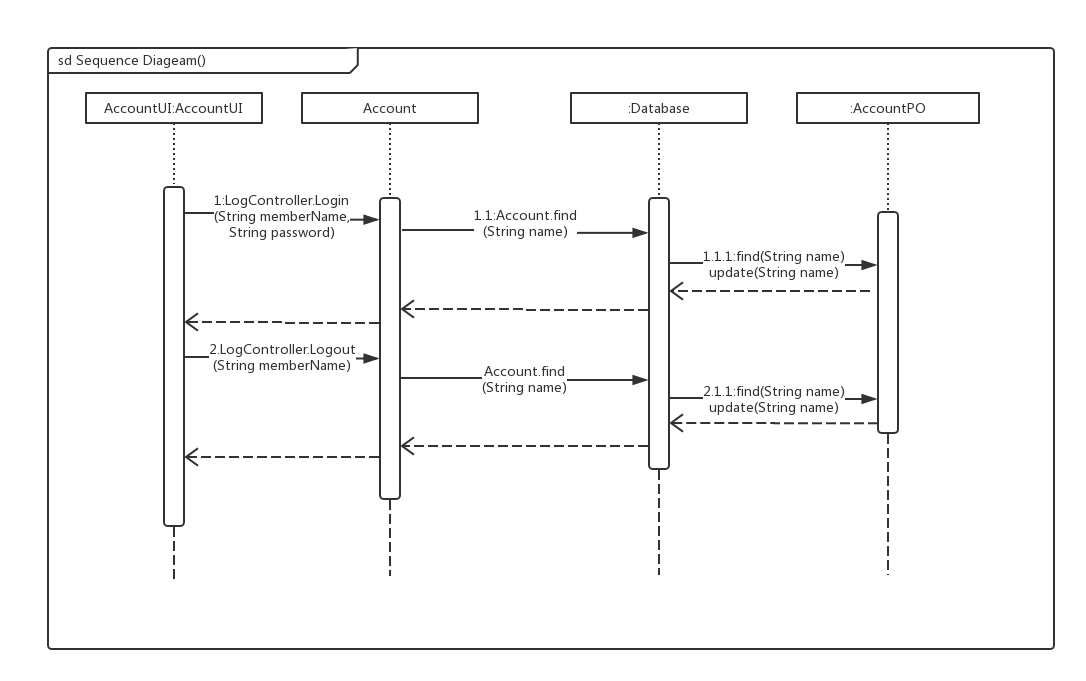
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ManageController.insert | 语法 | Public ResultMessage insert(String id) |
| 前置条件 | 无 |
| 后置条件 | 调用Account的对应方法 |
| ManageController.delete | 语法 | public ResultMessage delete(String id) |
| 前置条件 | 该账号存在 |
| 后置条件 | 调用Account的对应方法 |
| ManageController.modify | 语法 | Public ResultMessage modify(String memberName,String oldPassword,String newPassword); |
| 前置条件 | 无 |
| 后置条件 | 调用Account的对应方法 |
| ManageController.find | 语法 | Public ResultMessage find(String id) |
| 前置条件 | id符合输入规则 |
| 后置条件 | 调用Account的对应方法 |
| ManageController.Register | 语法 | public ResultMessage register(String memberName,String password); |
| 前置条件 | name和password符合输入规则 |
| 后置条件 | 调用Account的对应方法 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| Account.insert(AccountPO po) | 在数据库中插入一个单一持久化对象 | |
| Account.update(AccountPO po) | 在数据库中更新单一持久化对象 | |
| Account.update(AccountPO po) | 在数据库中删除单一持久化对象 | |
| Account.Register(String memberName,String password) | 注册账号 | |
| Account.find(String id) | 查找用户，返回信息 | |

**Account** 的接口规范

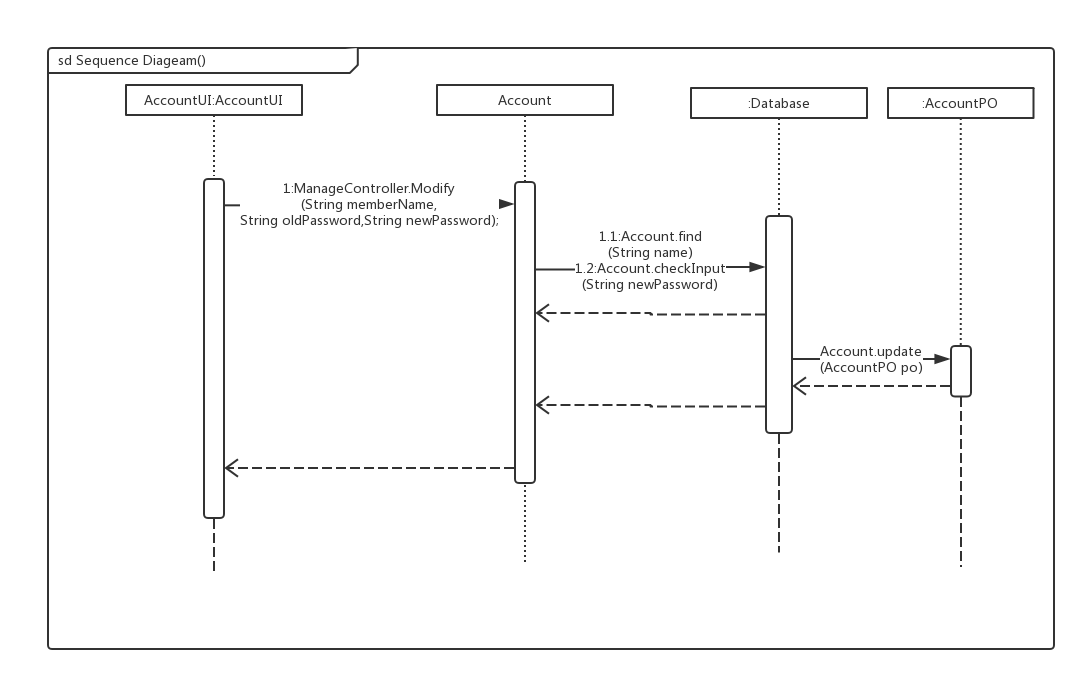
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Account.checkInput | 语法 | | Public ResultMessage check(String id) |
| 前置条件 | | 无 |
| 后置条件 | | 根据输入是否合法，判断请求是否响应 |
| Account.Insert | 语法 | | Public ResultMessage insert(AccountPO po) |
| 前置条件 | | 无 |
| 后置条件 | | 增加新的账户，持久化更新涉及的对象的数据 |
| Account.Update | 语法 | | Public ResultMessage update(AccountPO po) |
| 前置条件 | | 无 |
| 后置条件 | | 更新账户，持久化更新涉及的对象的数据 |
| Account.Delete | 语法 | | Public ResultMessage delete(AccountPO po) |
| 前置条件 | | 无 |
| 后置条件 | | 删除账户，持久化更新涉及的对象的数据 |
| Account.Login | 语法 | | Public ResultMessage login(String memberName,String password); |
| 前置条件 | | 已经创建一个Account领域对象，并且客户端尚未登录账号 |
| 后置条件 | | 将客户端置为已登录账号状态，持久化更新涉及的账号对象的数据 |
| Account.Logout | 语法 | | Public ResultMessage logout(String memberName); |
| 前置条件 | | 客户端处于已登录账号的状态 |
| 后置条件 | | 账号登出，将客户端置为未登录账号的状态 |
| Account.find | 语法 | | Public ResultMessage find(String id) |
| 前置条件 | | id符合输入规则 |
| 后置条件 | | 查找id是否存在，据此返回查找结果，若存在，返回该账号的详细信息 |
| Account.Register | 语法 | | public ResultMessage register(String memberName,String password); |
| 前置条件 | | name和password符合输入规则 |
| 后置条件 | | 查找name是否已存在，据此返回注册结果 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| AccountDataService.insert(AccountPO po) | | 在数据库中插入一个单一持久化对象 | |
| AccountDataService.delete(AccountPO po) | | 在数据库中删除单一持久化对象 | |
| AccountDataService.update(AccountPO po) | | 在数据库中更新单一持久化对象 | |
| AccountDataService.find(String name) | | 根据用户名查找单一持久化对象 | |
| AccountDataService.getNewID(AccountPO po) | | 创建一个数据库中的单一持久化对象的ID | |

（4）业务逻辑层的动态模型

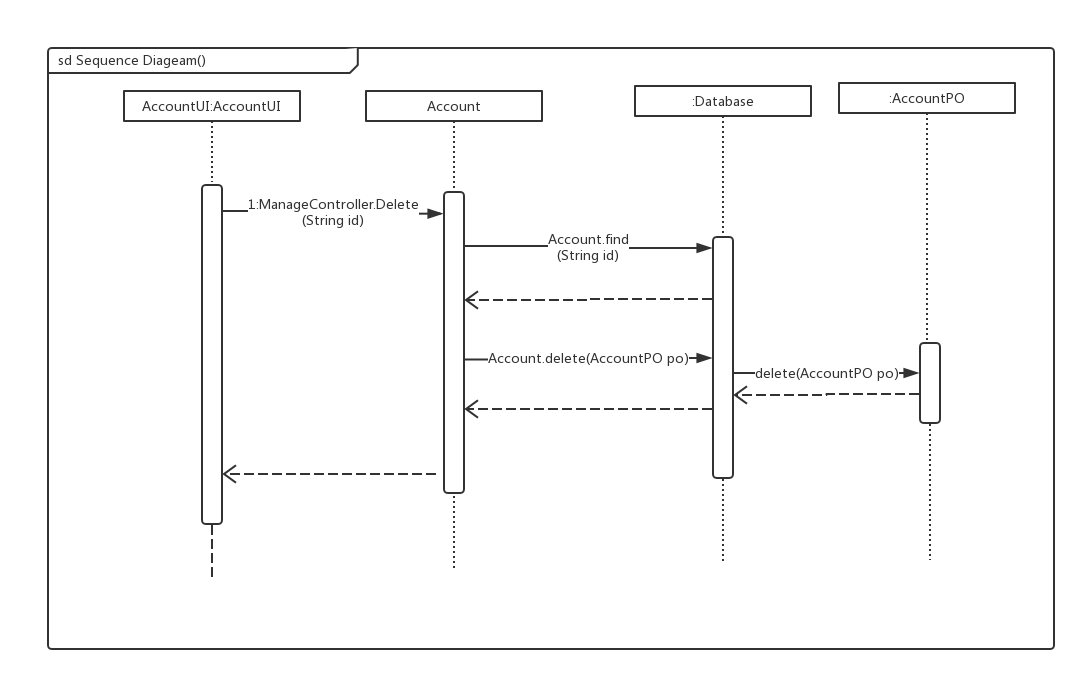
如图表明了当用户进行登入登出账号，账号业务逻辑处理的相关对象之间的协作。



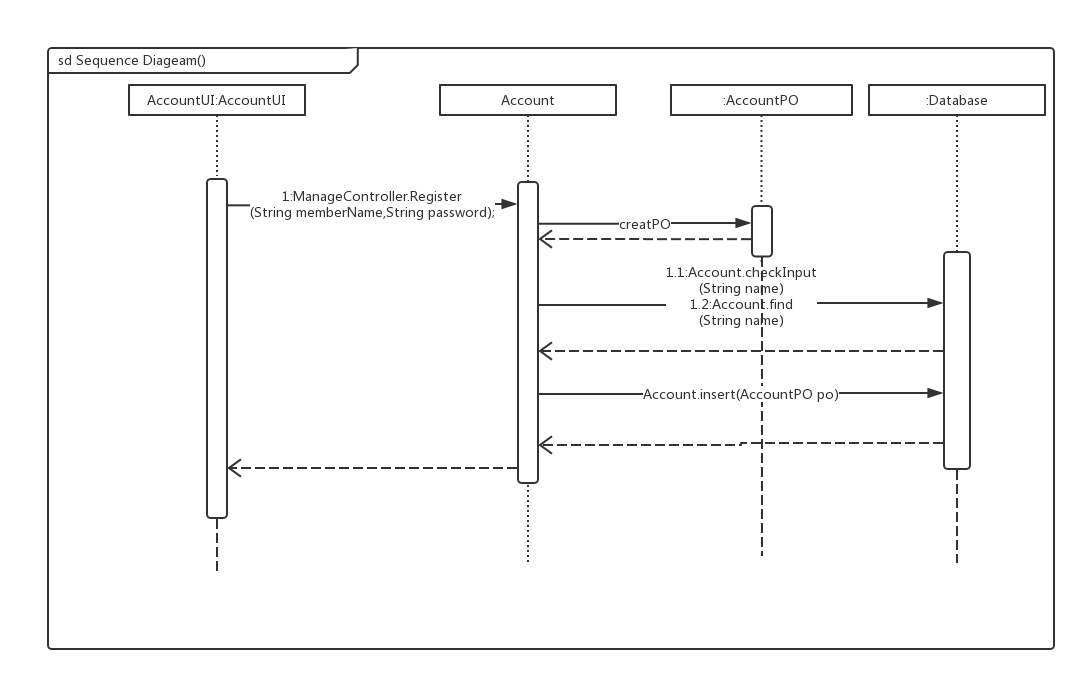
如图表明了当用户想要进行密码修改，账号业务逻辑处理的相关对象之间的协作。



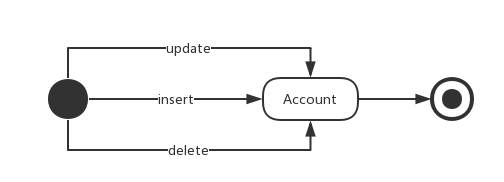
如图表明了当网站管理人员想要对账号进行管理（包括查找、修改、增加和删除账号）时，账号业务逻辑处理的相关对象之间的协作。



如图表明了当用户想要注册新账号的时候，账号业务逻辑处理的相关对象之间的协作。



如图所示的状态图描述了Account对象的生存期间的状态序列、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领域对象。

4.2.5 Promotionbl模块

（1）模块概述

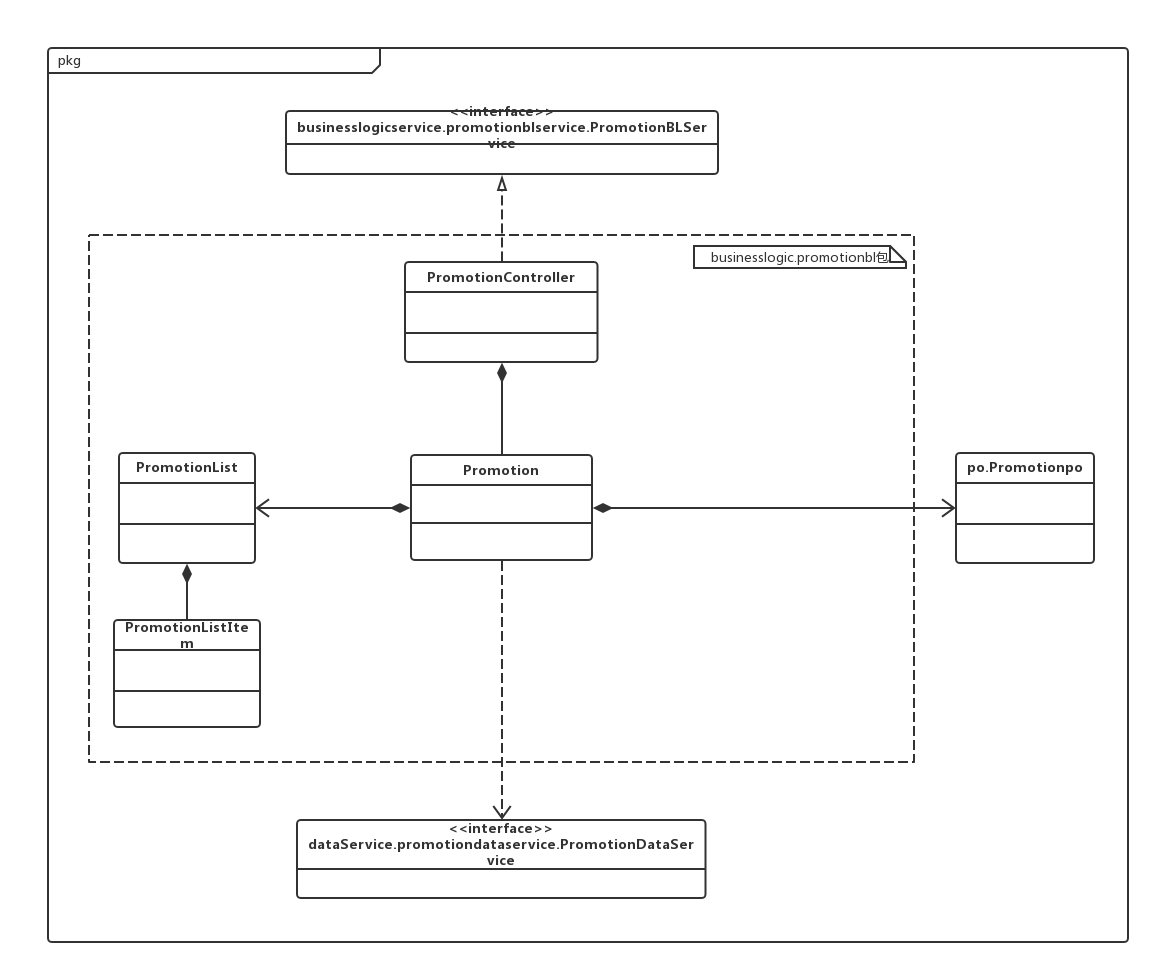
Promotionbl模块承担编辑并实行促销策略的需求，由酒店工作人员和网站营销人员维护，需要允许以上人员编辑促销策略信息（包括名称、开始日期、结束日期、折扣额度、面向的用户群、最小折扣房间数、描述、面向的商圈），并制定不同会员等级所需要的信用值。具体功能需求和非功能需求参见软件需求规格说明文档。

Promotionbl模块的职责及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，我们将系统分为用户界面层、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。比如用户界面层和业务逻辑层之间，我们添加businesslogicservice.promotionblservice.PromotionBLService接口。业务逻辑层和数据层之间添加dataService.promotiondataservice.PromotionDataService接口，为了隔离业务逻辑职责和逻辑控制职责，我们增加了PromotionController，这样PromotionController会将对促销策略的业务逻辑处理委托给Promotion对象。PromotionPO是作为促销策略的持久化对象被添加到设计模型中去的。

Promotionbl模块的设计如图所示。



Promotionbl模块各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| PromotionController | 负责实现制定促销策略所需要的服务 |
| Promotion | 促销策略的领域模型对象，拥有促销策略的ID、名称、开始时间、结束时间、折扣额度、最低折扣房间数、描述等信息，可以帮助完成制定促销策略界面所需要的服务。 |

（3）模块内部类的接口规范

PromotionController和Promotion的接口规范如下表所示。

**PromotionController**的接口规范

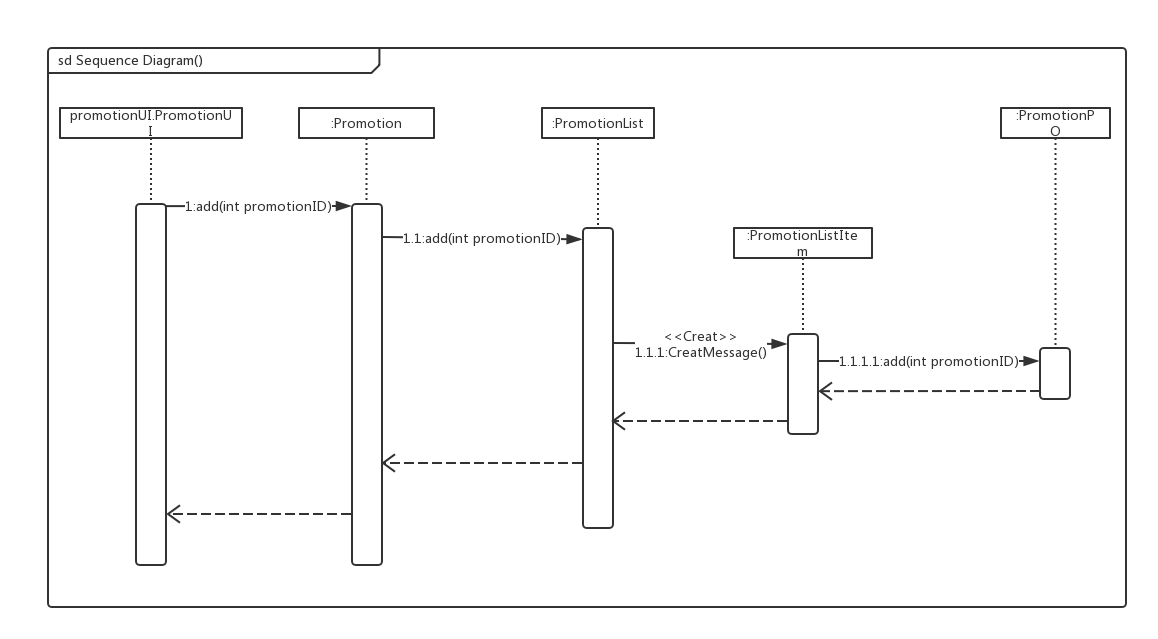
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PromotionController. add | 语法 | Public ResultMessage add(int promotionId) |
| 前置条件 | 已创建一个Promotion领域对象 |
| 后置条件 | 调用Promotion领域对象的add方法 |
| PromotionController. delete | 语法 | Public ResultMessage delete(int promotionId) |
| 前置条件 | 已创建一个Promotion领域对象 |
| 后置条件 | 调用Promotion领域对象的delete方法 |
| PromotionController. update | 语法 | Public ResultMessage update(int promotionId) |
| 前置条件 | 已创建一个Promotion领域对象 |
| 后置条件 | 调用Promotion领域对象的update方法 |
| PromotionController. editStartDate | 语法 | Public ResultMessage editStartDate(Date date) |
| 前置条件 | 已创建一个Promotion领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Promotion领域对象的editStartDate方法 |
| PromotionController. editEndDate | 语法 | Public ResultMessage editEndDate(Date date) |
| 前置条件 | 已创建一个Promotion领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Promotion领域对象的editEndDate方法 |
| PromotionController. editRoomNum | 语法 | Public ResultMessage editRoomNum(int number) |
| 前置条件 | 已创建一个Promotion领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Promotion领域对象的editRoomNum方法 |
| PromotionController. editDiscount | 语法 | Public ResultMessage editDiscount(int amount)  Public ResultMessage editDiscount(double rate) |
| 前置条件 | 已创建一个Promotion领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Promotion领域对象的editDiscount方法 |
| PromotionController. editDescription | 语法 | Public ResultMessage editDescription(String description) |
| 前置条件 | 已创建一个Promotion领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Promotion领域对象的editDescription方法 |
| PromotionController. editLevelCredit | 语法 | Public ResultMessage editLevelCredit(int credit) |
| 前置条件 | 已创建一个Promotion领域对象，并且输入符合输入规则 |
| 后置条件 | 调用Promotion领域对象的editLevelCredit方法 |
| 需要的服务（需接口） | | |
| 服务名 | | 服务 |
| Promotion. add(int promotionId) | | 添加一个促销策略 |
| Promotion. delete(int promotionId) | | 删除一个促销策略 |
| Promotion. update(int promotionId) | | 更新促销策略信息 |
| Promotion. editStartDate(Date date) | | 编辑促销策略开始时间 |
| Promotion. editEndDate(Date date) | | 编辑促销策略结束时间 |
| Promotion. editRoomNum(int number) | | 编辑提供折扣的最小房间数 |
| Promotion. editDiscount(int amount)  Promotion. editDiscount(double rate) | | 编辑促销策略的折扣额度 |
| Promotion. editDescription(String description) | | 编辑促销策略的描述 |
| Promotion. editLevelCredit(int credit) | | 编辑某一会员等级所需的信用值 |

**Promotion**的接口规范

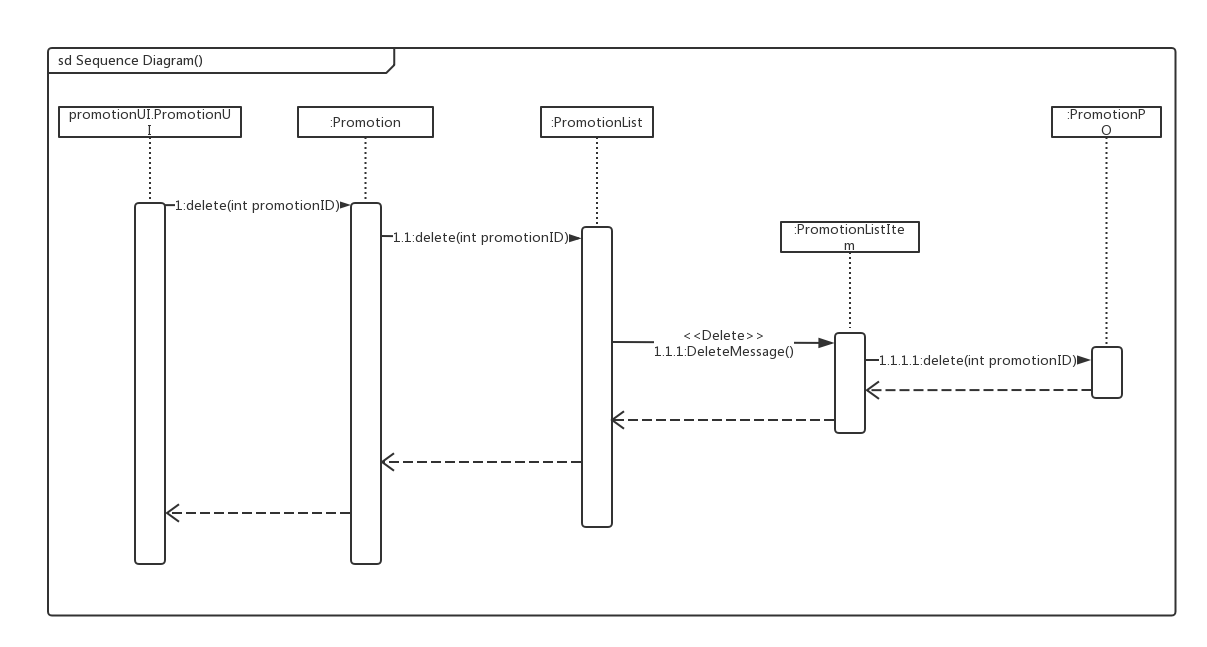
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Promotion. add | 语法 | Public ResultMessage add(int promotionId) | |
| 前置条件 | 无 | |
| 后置条件 | 系统在促销策略列表里添加一项，标记为new | |
| Promotion. delete | 语法 | Public ResultMessage delete(int promotionId) | |
| 前置条件 | 无 | |
| 后置条件 | 系统删除对应的促销策略 | |
| Promotion. update | 语法 | Public ResultMessage update(int promotionId) | |
| 前置条件 | 促销策略信息有更新 | |
| 后置条件 | 系统更新促销策略信息 | |
| Promotion. editStartDate | 语法 | Public ResultMessage editStartDate(Date date) | |
| 前置条件 | 开始时间合法 | |
| 后置条件 | 暂存促销策略的开始时间 | |
| Promotion. editEndDate | 语法 | Public ResultMessage editEndDate(Date date) | |
| 前置条件 | 结束时间合法 | |
| 后置条件 | 暂存促销策略的结束时间 | |
| Promotion. editRoomNum | 语法 | Public ResultMessage editRoomNum(int number) | |
| 前置条件 | 房间数合法 | |
| 后置条件 | 暂存最低优惠房间数 | |
| Promotion. editDiscount | 语法 | Public ResultMessage editDiscount(int amount)  Public ResultMessage editDiscount(double rate) | |
| 前置条件 | 折扣额度合法 | |
| 后置条件 | 暂存折扣额度 | |
| Promotion. editDescription | 语法 | Public ResultMessage editDescription(String description) | |
| 前置条件 | 促销策略描述合法 | |
| 后置条件 | 暂存促销策略的描述 | |
| Promotion. editLevelCredit | 语法 | Public ResultMessage editLevelCredit(int credit) | |
| 前置条件 | 信用值合法 | |
| 后置条件 | 暂存某会员等级所需的信用值 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| PromotionDataService. find(String id) | | | 根据ID进行查找单一持久化对象 |
| PromotionDataService. find(String field, String value) | | | 根据字段名和值进行查找多个持久化对象 |
| PromotionDataService.insert(PromotionPO po) | | | 插入单一持久化对象 |
| PromotionDataService. delete(PromotionPO po) | | | 删除单一持久化对象 |
| PromotionDataService. update(PromotionPO po) | | | 更新单一持久化对象 |
| DatabaseFactory. getPromotionDatabase | | | 得到Promotion数据库的服务的引用 |
| DataBaseFactory.insert(PromotionPO po) | | | 在数据库中插入PromotionPO对象 |
| DataBaseFactory.delete (PromotionPO po) | | | 在数据库中删除PromotionPO对象 |
| DataBaseFactory.delete(PromotionPO po) | | | 在数据库中更新单一持久化对象 |

（4）业务逻辑层的动态模型

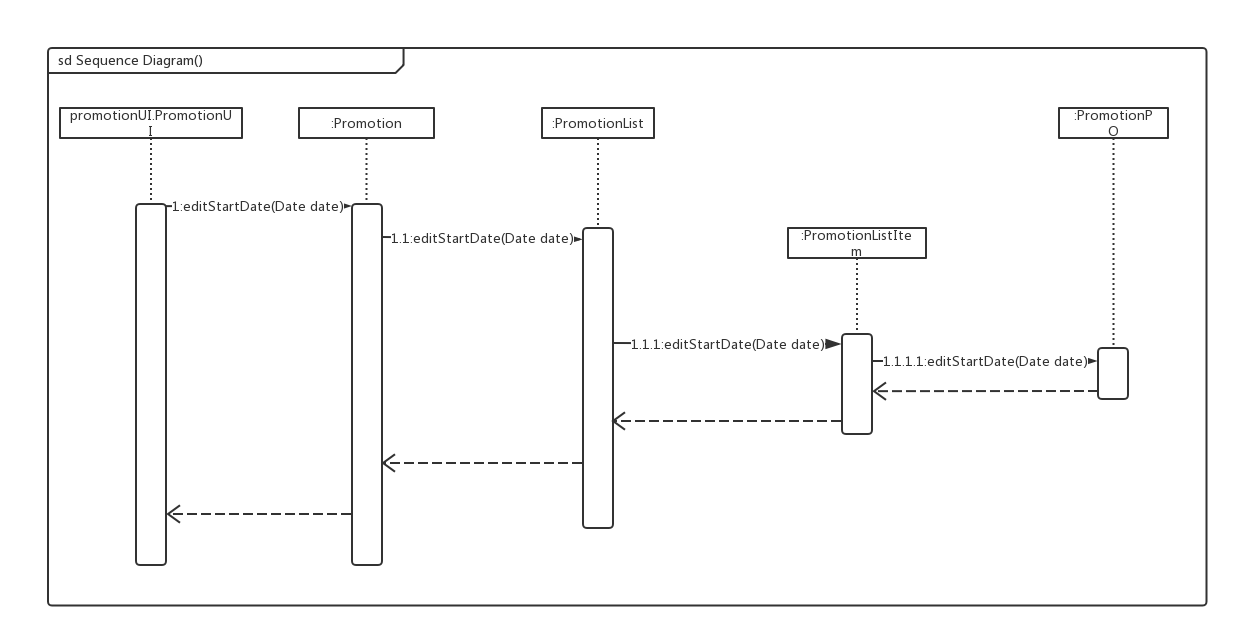
如图表明了互联网酒店管理系统中，当酒店工作人员或网站营销人员添加一个促销策略时，促销策略业务逻辑处理的相关对象之间的协作。



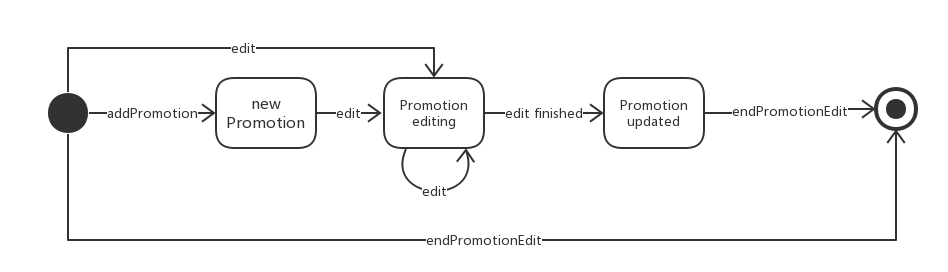
如图表明了互联网酒店管理系统中，当酒店工作人员或网站营销人员删除一个促销策略时，促销策略业务逻辑处理的相关对象之间的协作。



如图表明了互联网酒店管理系统中，当酒店工作人员或网站营销人员编辑一个促销策略时，促销策略业务逻辑处理的相关对象之间的协作。



如图所示的状态图描述了Promotion对象的生存期间的状态序列、引起转移的事件，以及因状态转移而伴随的动作。

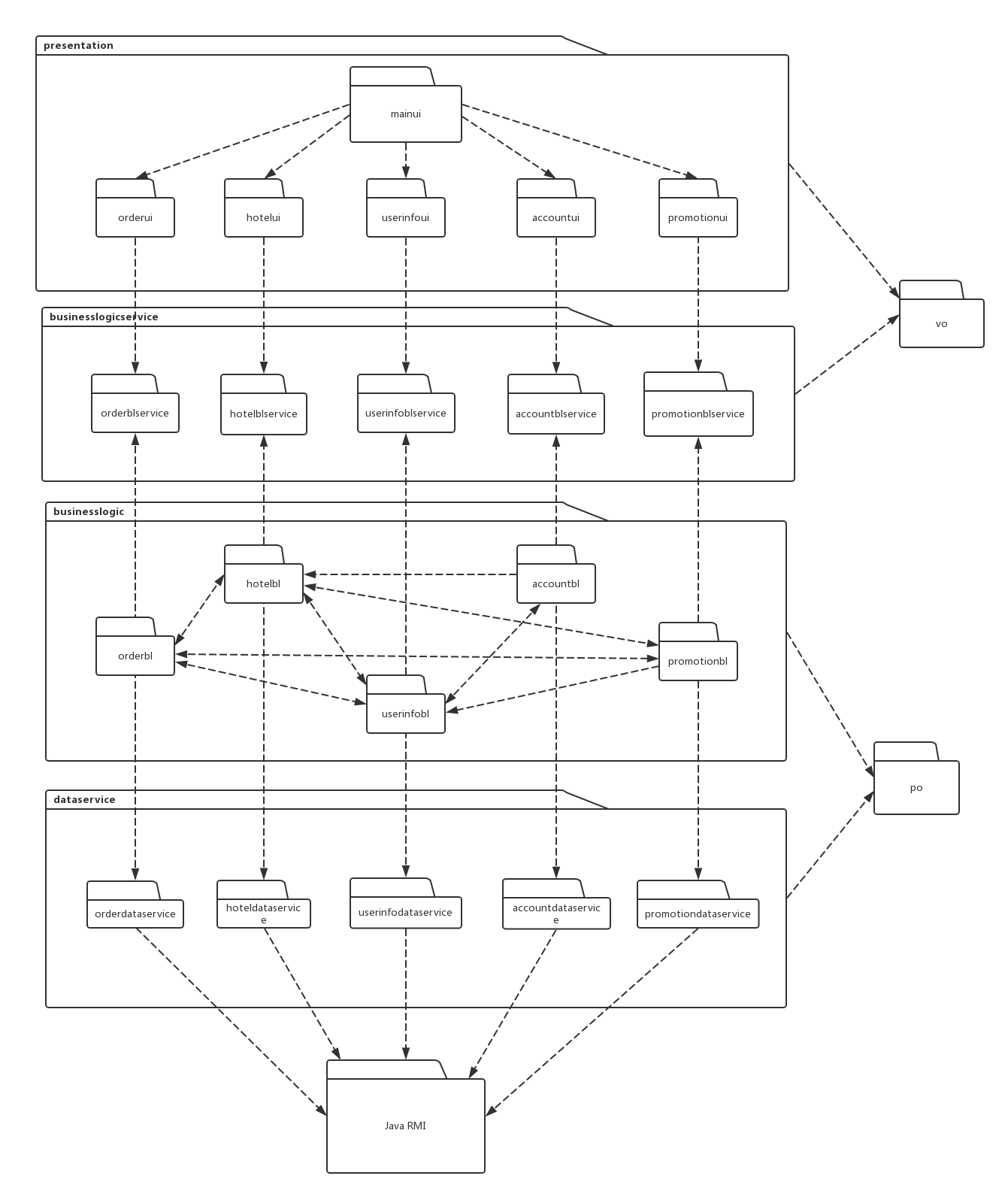


（5）业务逻辑层的设计原理

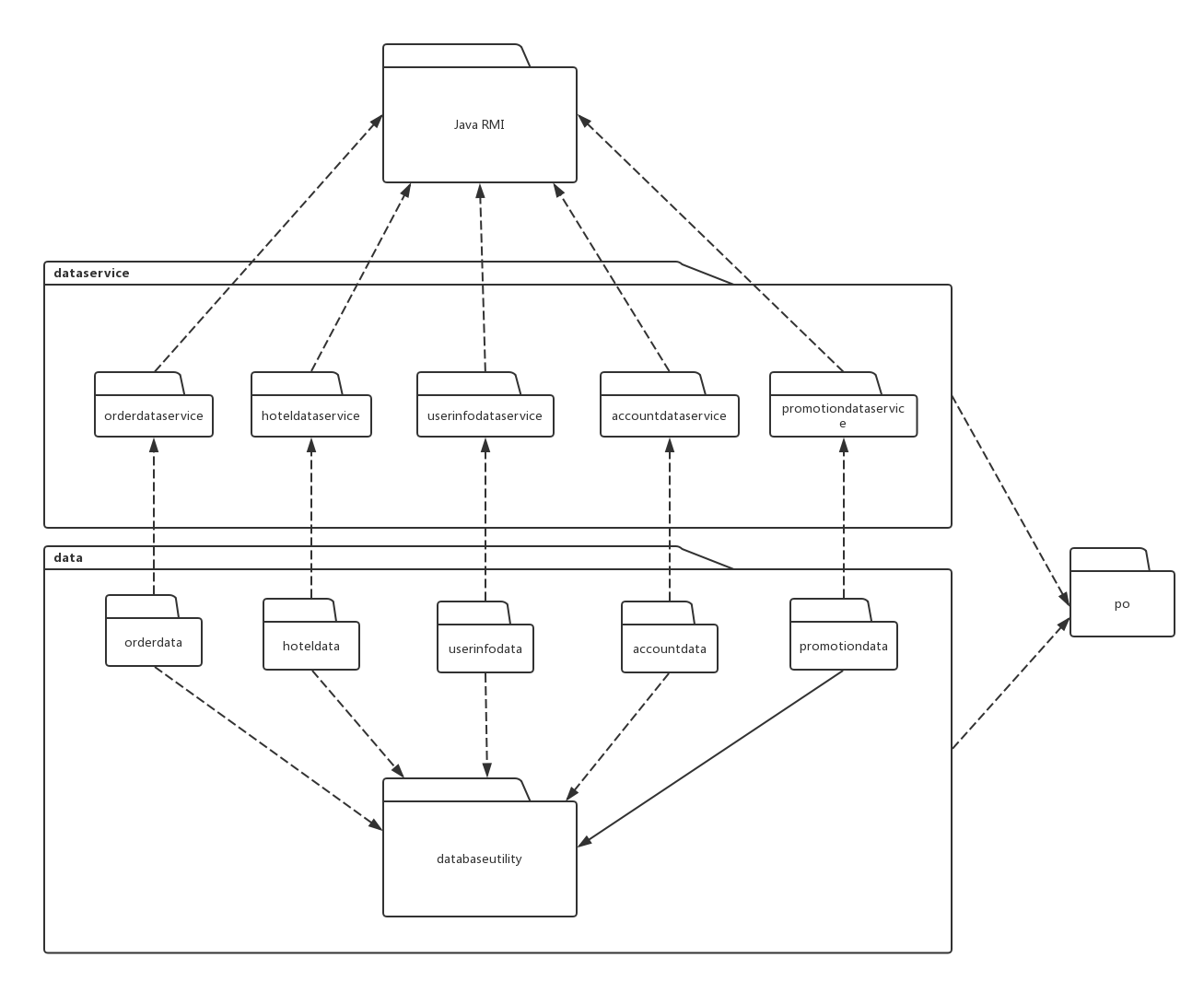
利用委托式控制风格，每个界面需要访问的业务逻辑由各自的控制器委托给不同领域对象。

4.3数据层的分解

5.依赖视角



客户端包图



服务器端包图