目录

[更新历史 2](#_Toc466238755)

[一、 引言 4](#_Toc466238756)

[(一) 编制目的 4](#_Toc466238757)

[(二) 词汇表 4](#_Toc466238758)

[(三) 参考资料 5](#_Toc466238759)

[二、 产品概述 6](#_Toc466238760)

[三、 体系结构设计概述 7](#_Toc466238761)

[四、 结构视角 8](#_Toc466238762)

[(一) 展示层的分解 8](#_Toc466238763)

[1. UI层设计概要 8](#_Toc466238764)

[2. UI层内部包的职责与划分 9](#_Toc466238765)

[(二) 业务逻辑层的分解 9](#_Toc466238766)

[1. logIn模块 9](#_Toc466238767)

[2. order模块 14](#_Toc466238768)

[3. hotel模块 25](#_Toc466238769)

[4. promotion模块 40](#_Toc466238770)

[5. credit模块 43](#_Toc466238771)

[6. member模块 48](#_Toc466238772)

[7. user模块 54](#_Toc466238773)

[8. market模块 65](#_Toc466238774)

[(三) 数据层的分解 70](#_Toc466238775)

[1. orderData模块 70](#_Toc466238776)

[2. hotelData模块 71](#_Toc466238777)

[3. promotionData模块 75](#_Toc466238778)

[4. marketData模块 76](#_Toc466238779)

[5. guestData模块 76](#_Toc466238780)

[6. hotelWorkerData模块 78](#_Toc466238781)

[7. webMarketerData模块 79](#_Toc466238782)

[8. webManagerData模块 80](#_Toc466238783)

[五、 依赖视角 82](#_Toc466238784)

# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **修改日期** | **修改原因** | **版本号** |
| 董金玉 | 2016/10/26 | 新增User、Member部分模型初稿 |  |
| 龚尘淼 | 2016/10/27 | 新增Hotel、Promotion部分模型初稿 |  |
| 董金玉 | 2016/10/27 | 据第一次评审修改顺序图、状态图 |  |
| 高源 | 2016/10/28 | 新增Credit、LogIn、Market部分模型初稿 |  |
| 董金玉 | 2016/10/28 | 据第一次评审修改顺序图、状态图 |  |
| 冯俊杰 | 2016/10/28 | 新增Order类图 |  |
| 龚尘淼 | 2016/10/28 | 据第一次评审修改顺序图、状态图 |  |
| 全体成员 | 2016/10/28 | 确定规范 |  |
| 董金玉 | 2016/10/28 | 据第二次评审修改顺序图、状态图 |  |
| 董金玉 | 2016/10/28 | 新增getAll()接口 |  |
| 龚尘淼 | 2016/10/29 | 据第二次评审修改顺序图、状态图 |  |
| 高源 | 2016/10/29 | 据第二次评审修改顺序图、状态图 |  |
| 冯俊杰 | 2016/10/29 | 新增模块填写模版 |  |
| 冯俊杰 | 2016/10/29 | 新增Order模块 |  |
| 董金玉 | 2016/10/29 | 新增Member、User模块 |  |
| 高源 | 2016/10/29 | 据第二次评审修改顺序图、状态图 |  |
| 高源 | 2016/10/29 | 新增Credit、LogIn、Market模块 |  |
| 冯俊杰 | 2016/10/29 | 整合、排版 | V1.0 |
| 冯俊杰 | 2016/11/2 | 新增Hotel接口 |  |
| 董金玉 | 2016/11/2 | 修改memberController和userController规范 |  |
| 冯俊杰 | 2016/11/2 | 新增Order模块顺序图 |  |
| 董金玉 | 2016/11/3 | 据第三次评审修改顺序图 |  |
| 董金玉 | 2016/11/5 | 修改部分错误，添加接口 |  |
| 高源 | 2016/11/5 | 添加界面初版 |  |
| 冯俊杰 | 2016/11/6 | 更新物理分包，添加getAllCreditDetail()接口 |  |
| 高源 | 2016/11/6 | 修改模块图 |  |
| 董金玉 | 2016/11/6 | 据第四次评审修改顺序图 |  |
| 冯俊杰 | 2016/11/6 | 更新LogIn接口 |  |
| 冯俊杰 | 2016/11/6 | 修正模块 |  |
| 董金玉 | 2016/11/6 | 添加数据层分解 |  |
| 冯俊杰 | 2016/11/6 | 整合、排版 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 引言

## 编制目的

本报告详细完成对互联网酒店预订系统的详细设计，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编写，是了解系统的导航。

## 词汇表

|  |  |
| --- | --- |
| **缩写或单词** | **解释** |
| HRS | 互联网酒店预订系统 |
| order | 订单 |
| hotel | 酒店 |
| guest | 客户 |
| hotelWorker | 酒店工作人员 |
| webMarketer | 网站营销人员 |
| webManager | 网站管理人员 |
| signUp | 注册 |
| logIn | 登录 |
| UI | 界面 |
| BL | 逻辑 |
| Data | 数据 |
| promotion | 促销策略 |
| credit | 信用值 |
| member | 会员 |
| market | 营销 |
|  |  |
|  |  |
|  |  |
|  |  |

## 参考资料

1. 互联网酒店预订系统用例文档
2. 互联网酒店预订系统需求规格说明文档
3. 互联网酒店预订系统体系结构设计文档
4. 丁二玉，刘钦.计算与软件工程（卷二）[M]机械工业出版2012：134—182
5. IEEE std 1471-2000

# 产品概述

参考《互联网酒店预订系统用例文档》和《互联网酒店预订系统需求规格说明文档》中对产品的概括描述。互联网酒店预订系统主要功能见用例图如下。



# 体系结构设计概述

参考互联网酒店预订系统体系结构设计文档的描述。

主要采用分层的架构方式，采用分布式的部署方式，将数据存储在服务器端，客户端。通过RMI方式调用获取数据。

# 结构视角

## 展示层的分解

### UI层设计概要

根据需求，系统存在38个用户界面（加粗为主界面）： **登录界面**、**注册界面**、**客户主界面**、个人基本信息界面、个人基本信息维护界面、会员注册界面、未评价订单界面、订单评价界面、个人订单概况列表界面、个人订单详情界面、个人订单撤销界面、商圈地址选定界面、酒店概况列表界面、订单生成界面、酒店搜索界面、酒店详情界面、会员注册界面、**酒店工作人员界面**、酒店信息界面、酒店信息维护界面、客房信息维护界面、酒店促销策略界面、订单列表界面、订单详情界面、剩余客房信息界面、线下客户入住和退房处理界面、**网站营销人员界面**、用户编号录入界面、信用充值界面、异常订单概况列表界面、异常订单详情界面、异常订单撤销界面、网站促销策略界面、会员等级制定界面、**网站管理人员界面**、会员信息界面、会员信息维护界面、酒店信息添加界面。

界面跳转如下图所示。



### UI层内部包的职责与划分

## 业务逻辑层的分解

业务逻辑层的开发包图请查看体系结构设计文档。

### logIn模块

1. 模块概述

logIn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。

logIn模块的职责和接口参见软件系统结构描述文档内对该模块的描述。

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。在展示层和业务逻辑层之间添加logInBLService接口。为了隔离业务逻辑职责和逻辑控制职责，添加logInController，这样logInController会将对用户登录和客户注册的业务逻辑处理委托给logIn对象。guestPO、hotelWorkerPO、webMarketer、webManager是作为账户信息的持久性对象被添加到设计模型中去的。



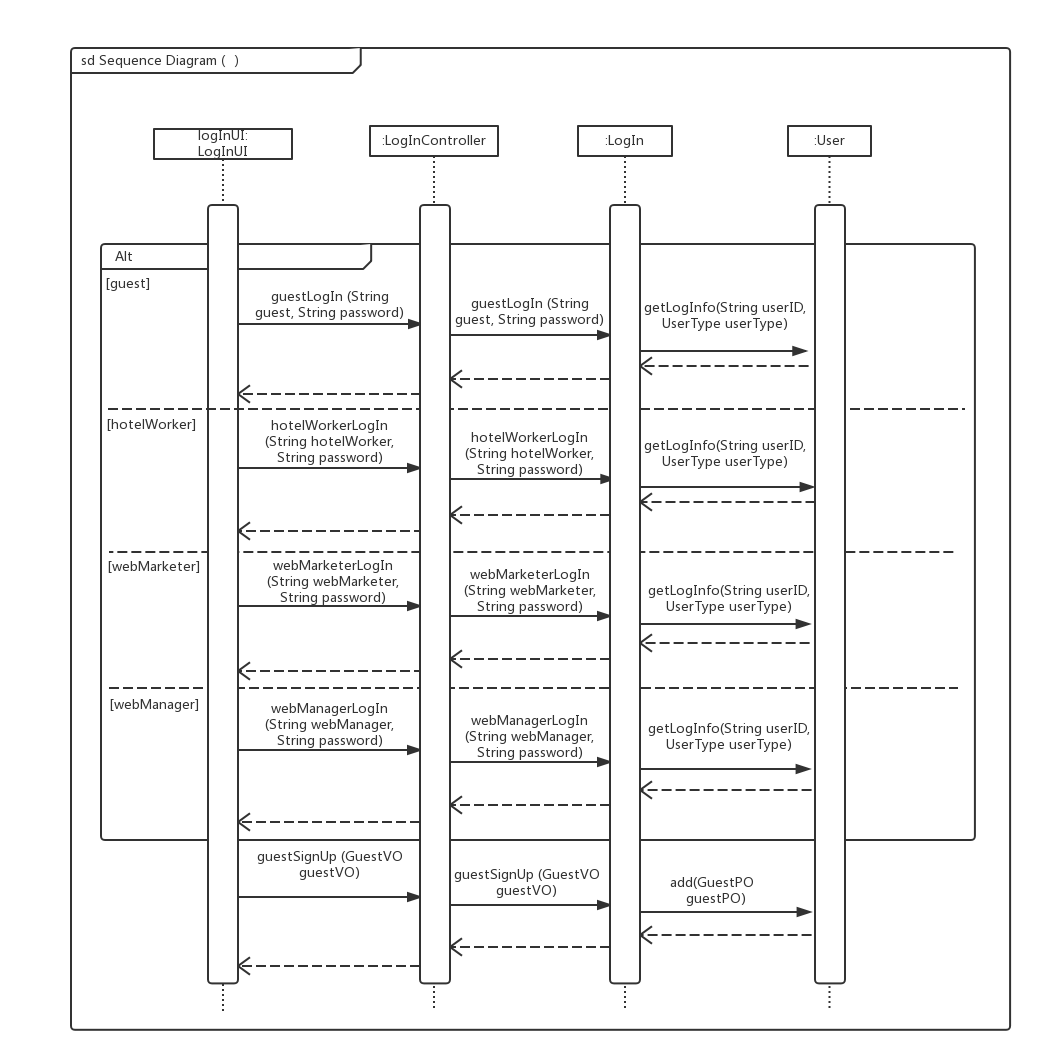
1. 模块内部类的接口规范
2. Controller类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| LogInController.guestLogIn | 语法 | public ResultMessage guestLogIn (String guest, String password) |
| 前置条件 | 客户输入客户编号和密码 |
| 后置条件 | 客户被认证并进入系统客户主界面 |
| LogInController.hotelWorkerLogIn | 语法 | public ResultMessage hotelWorkerLogIn (String hotelWorker, String password) |
| 前置条件 | 酒店工作人员输入编号和密码 |
| 后置条件 | 酒店工作人员被认证并进入系统酒店工作人员主界面 |
| LogInController.webMarketerLogIn | 语法 | public ResultMessage webMarketerLogIn (String webMarketer, String password) |
| 前置条件 | 网站营销人员输入编号和密码 |
| 后置条件 | 网站营销人员被认证并进入系统网站营销人员主界面 |
| LogInController.webManagerLogIn | 语法 | public ResultMessage webManagerLogIn (String webManager, String password) |
| 前置条件 | 网站管理人员输入编号和密码 |
| 后置条件 | 网站管理人员被认证并进入系统网站管理人员主界面 |
| LogInController.guestSignUp | 语法 | public ResultMessage guestSignUp (GuestVO guestVO) |
| 前置条件 | 客户信息符合规范要求 |
| 后置条件 | 系统添加并持续化该客户对并进入系统客户主界面，或者提示添加失败 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| LogIn.guestLogIn | 客户登录系统 | |
| LogIn.hotelWorkerLogIn | 酒店工作人员登录系统 | |
| LogIn.webMarketerLogIn | 网站营销人员登录系统 | |
| LogIn.webManagerLogIn | 网站管理人员登录系统 | |
| LogIn.guestSignUp | 客户注册 | |

1. Domain类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| LogIn.guestLogIn | 语法 | public ResultMessage guestLogIn (String guest, String password) |
| 前置条件 | 客户输入客户编号和密码 |
| 后置条件 | 客户被认证并进入系统客户主界面 |
| LogIn.hotelWorkerLogIn | 语法 | public ResultMessage hotelWorkerLogIn (String hotelWorker, String password) |
| 前置条件 | 酒店工作人员输入编号和密码 |
| 后置条件 | 酒店工作人员被认证并进入系统酒店工作人员主界面 |
| LogIn.webMarketerLogIn | 语法 | public ResultMessage webMarketerLogIn (String webMarketer, String password) |
| 前置条件 | 网站营销人员输入编号和密码 |
| 后置条件 | 网站营销人员被认证并进入系统网站营销人员主界面 |
| LogIn.webManagerLogIn | 语法 | public ResultMessage webManagerLogIn (String webManager, String password) |
| 前置条件 | 网站管理人员输入编号和密码 |
| 后置条件 | 网站管理人员被认证并进入系统网站管理人员主界面 |
| LogIn.guestSignUp | 语法 | public ResultMessage guestSignUp (GuestVO guestVO) |
| 前置条件 | 客户信息符合规范要求 |
| 后置条件 | 系统添加并持续化该客户对并进入系统客户主界面，或者提示添加失败 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| User.add(UserPO newUserPO) | 新增该客户并持久化保存该客户的数据 | |
| User.getLogInInfo(String userID, UserType userType) | 得到与该用户编号相对应的密码 | |

1. 业务逻辑层的动态模型



LogIn顺序图

1. 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。

### order模块

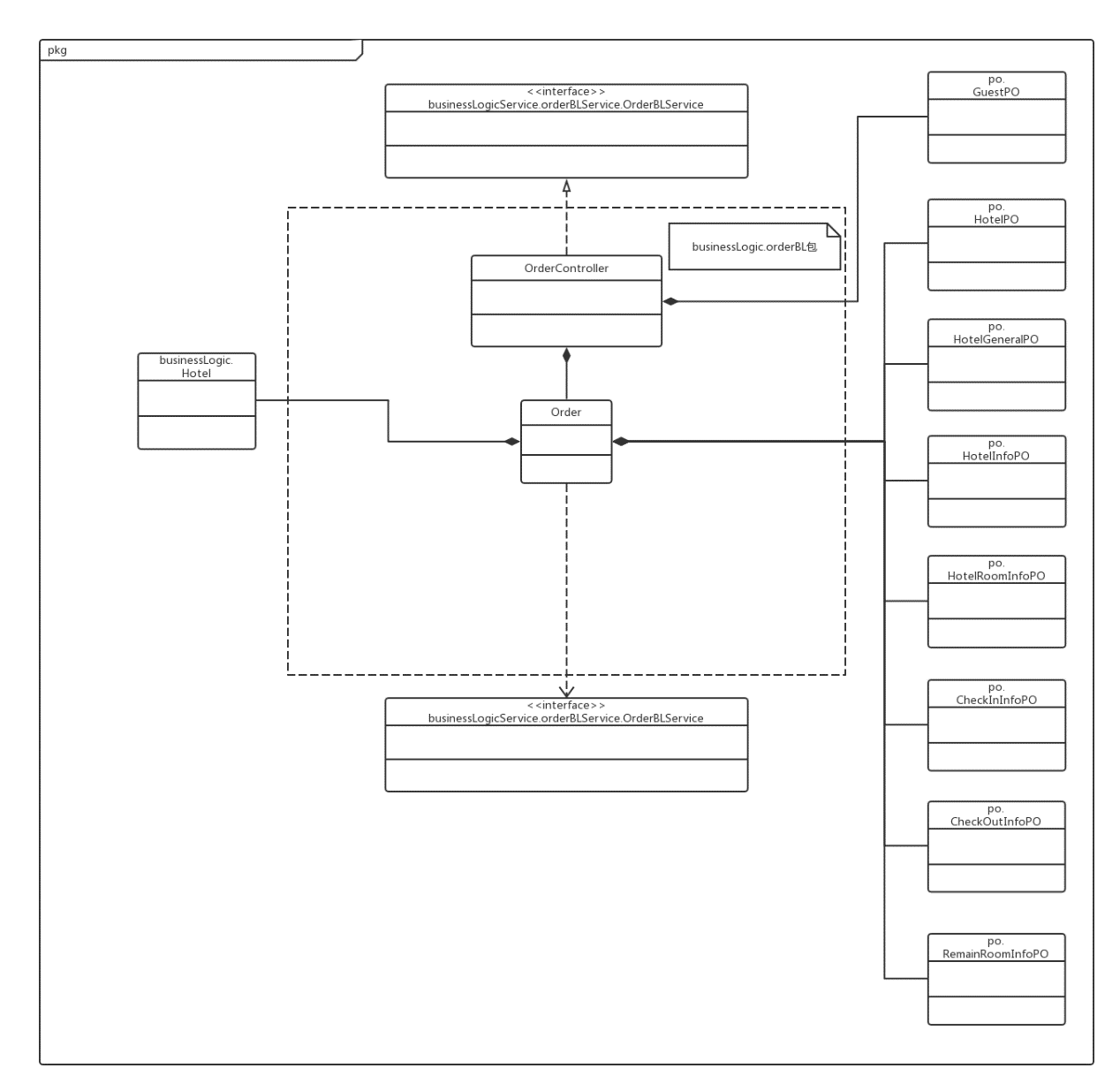
1. 模块概述

order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。

order模块的职责和接口参见软件系统结构描述文档内对该模块的描述。

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。在展示层和业务逻辑层之间添加orderBLService接口，在业务逻辑层和数据层之间添加orderDataService接口。为了隔离业务逻辑职责和逻辑控制职责，添加OrderController，这样OrderController会将对用户登录和客户注册的业务逻辑处理委托给Order对象。GuestPO、HotelPO、HotelGeneralPO、HotelInfoPO、HotelRoomInfoPO、CheckInInfoPO、CheckOutInfoPO、RemainRoomInfoPO是作为账户信息的持久性对象被添加到设计模型中去的。



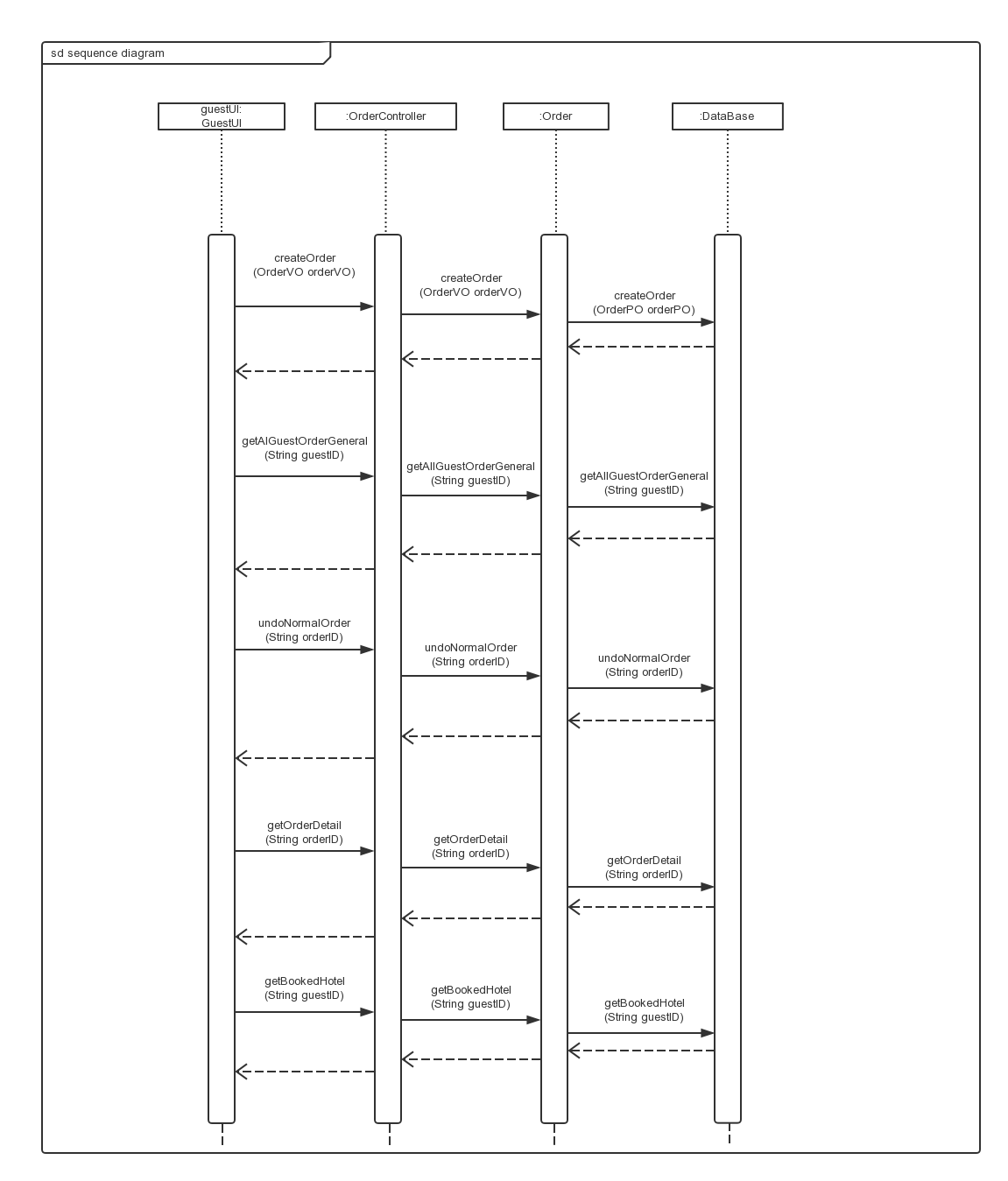
1. 模块内部类的接口规范
2. Controller类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderController.createOrder | 语法 | public ResultMessage createOrder(OrderVO orderVO) |
| 前置条件 | 客户已被认证和授权，填写订单的内容符合规则且完整 |
| 后置条件 | 系统生成未执行订单并持久化增加该客户的订单数据 |
| OrderController.executeOrder | 语法 | public ResultMessage executeOrder (String orderID) |
| 前置条件 | 酒店工作人员已被认证和授权，订单状态为未执行 |
| 后置条件 | 系统更改状态为已执行订单并持久化保存该客户的订单数据 |
| OrderController.undoAbnormalOrder | 语法 | public ResultMessage undoAbnormalOrder(String orderID) |
| 前置条件 | 网站营销人员已被认证和授权 |
| 后置条件 | 系统更改状态为已撤销订单、恢复一定比例的信用值并持久化保存该客户的订单数据 |
| OrderController.undoNormalOrder | 语法 | public ResultMessage undoNormalOrder(String orderID) |
| 前置条件 | 客户已被认证和授权，订单状态为未执行 |
| 后置条件 | 系统更改状态为已撤销订单并持久化保存该客户的订单数据 |
| OrderController.getOrderDetail | 语法 | public OrderVO getOrderDetail(String orderID) |
| 前置条件 | 用户已被认证和授权，已知订单号，需要知道订单内容 |
| 后置条件 | 无 |
| OrderController.getAllGuestOrderGeneral | 语法 | public List<OrderGeneralVO> getAllGuestOrderGeneral (String guestID) |
| 前置条件 | 客户已被认证和授权 |
| 后置条件 | 无 |
| OrderController.getAllHotelOrderGeneral | 语法 | public List<OrderGeneralVO> getAllHotelOrderGeneral (String hotelID) |
| 前置条件 | 酒店工作人员已被认证和授权 |
| 后置条件 | 无 |
| OrderController.getAllAbnormalOrderGeneral | 语法 | public List<OrderGeneralVO> getAllAbnormalOrderGeneral (LocalDate date) |
| 前置条件 | 网站营销人员已被认证和授权 |
| 后置条件 | 无 |
| OrderController.getAllAbnormalOrderGeneral | 语法 | public List<OrderGeneralVO> getAllHotelOrderGeneral () |
| 前置条件 | 网站营销人员已被认证和授权 |
| 后置条件 | 无 |
| OrderController.getBookedHotel | 语法 | public List<String> getBookedHotel (String guestID) |
| 前置条件 | 客户已被认证和授权 |
| 后置条件 | 系统返回所有该客户已住过酒店的ID列表 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| Order.createOrder(orderVO OrderVO) | 订单生成 | |
| Order.executeOrder (String orderID) | 订单执行 | |
| Order.undoAbnormalOrder (String orderID) | 撤销异常订单 | |
| Order.undoNormalOrder | 撤销正常订单 | |
| Order.getOrderDetail (String orderID) | 查看订单概况 | |
| Order.getAllGuestOrderGeneral(String guestID) | 查看所有客户订单概况 | |
| Order.getAllHotelOrderGeneral(String hotelID) | 查看所有酒店订单概况 | |
| Order.getAllAbnormalOrderGeneral(LocalDate date) | 查看指定时间所有异常订单概况 | |
| Order.getAllAbnormalOrderGeneral() | 查看所有异常订单概况 | |
| Order.getBookedHotel(String guestID) | 查看客户预定过的酒店 | |

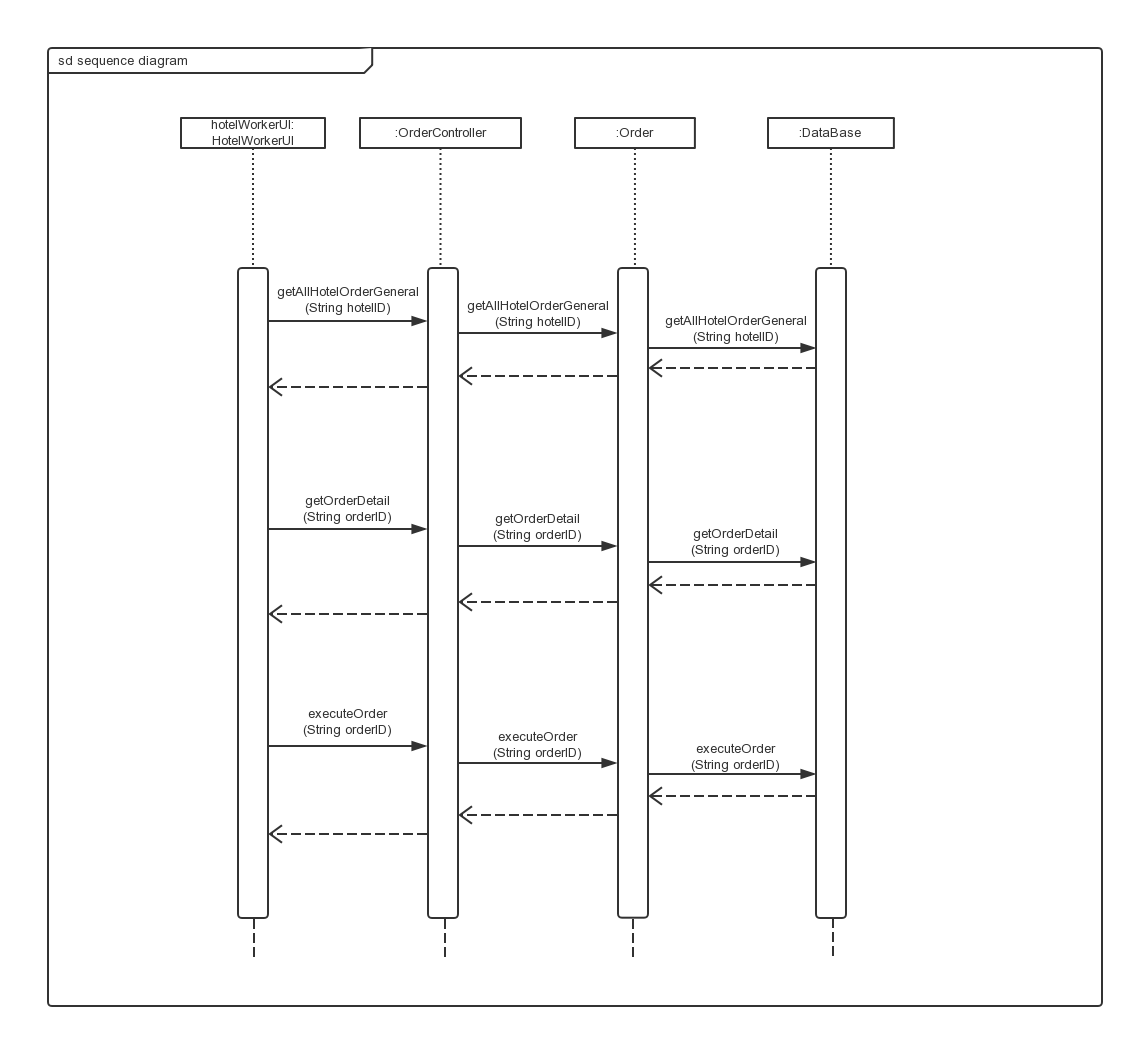
1. Domain类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Order.createOrder | 语法 | public ResultMessage createOrder(OrderVO orderVO) |
| 前置条件 | 客户已被认证和授权，填写订单的内容符合规则且完整 |
| 后置条件 | 系统生成未执行订单并持久化增加该客户的订单数据 |
| Order.executeOrder | 语法 | public ResultMessage executeOrder (String orderID) |
| 前置条件 | 酒店工作人员已被认证和授权，订单状态为未执行 |
| 后置条件 | 系统更改状态为已执行订单并持久化保存该客户的订单数据 |
| Order.undoAbnormalOrder | 语法 | public ResultMessage undoAbnormalOrder(String orderID) |
| 前置条件 | 网站营销人员已被认证和授权 |
| 后置条件 | 系统更改状态为已撤销订单、恢复一定比例的信用值并持久化保存该客户的订单数据 |
| Order.undoNormalOrder | 语法 | public ResultMessage undoNormalOrder(String orderID) |
| 前置条件 | 客户已被认证和授权，订单状态为未执行 |
| 后置条件 | 系统更改状态为已撤销订单并持久化保存该客户的订单数据 |
| Order.getOrderDetail | 语法 | public OrderVO getOrderDetail(String orderID) |
| 前置条件 | 用户已被认证和授权，已知订单号，需要知道订单内容 |
| 后置条件 | 无 |
| Order.getAllGuestOrderGeneral | 语法 | public List<OrderGeneralVO> getAllGuestOrderGeneral (String guestID) |
| 前置条件 | 客户已被认证和授权 |
| 后置条件 | 无 |
| Order.getAllHotelOrderGeneral | 语法 | public List<OrderGeneralVO> getAllHotelOrderGeneral (String hotelID) |
| 前置条件 | 酒店工作人员已被认证和授权 |
| 后置条件 | 无 |
| Order.getAllAbnormalOrderGeneral | 语法 | public List<OrderGeneralVO> getAllAbnormalOrderGeneral (LocalDate date) |
| 前置条件 | 网站营销人员已被认证和授权 |
| 后置条件 | 无 |
| Order.getAllAbnormalOrderGeneral | 语法 | public List<OrderGeneralVO> getAllHotelOrderGeneral () |
| 前置条件 | 网站营销人员已被认证和授权 |
| 后置条件 | 无 |
| Order.getBookedHotel | 语法 | public List<String> getBookedHotel (String guestID) |
| 前置条件 | 客户已被认证和授权 |
| 后置条件 | 系统返回所有该客户已住过酒店的ID列表 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| OrderDataService.createOrder(orderPO OrderPO) | 根据提供的po增加持久化对象 | |
| OrderDataService.executeOrder(String orderID) | 修改单一持久化对象 | |
| OrderDataService.getOrderDetail(String orderID) | 根据ID查找持久化对象 | |
| OrderDataService.getAllHotelOrderGeneral (String hotelID) | 根据ID查找持久化对象 | |
| OrderDataService.getAllGuestOrderGeneral (String guestID) | 根据ID查找持久化对象 | |
| OrderDataService.getAllAbnormalOrderGeneral (Date date) | 根据ID查找持久化对象 | |
| OrderDataService.getAllAbnormalOrderGeneral () | 根据ID查找持久化对象 | |
| OrderDataService.undoNormalOrder(String orderID) | 修改单一持久化对象 | |
| OrderDataService.UndoAbnormalOrder(String orderID) | 修改单一持久化对象 | |
| Hotel.getHotelInfo (String userID) | 获得单一对象 | |
| Promotion.getDiscount (PreOrder preOrder) | 获得单一对象 | |

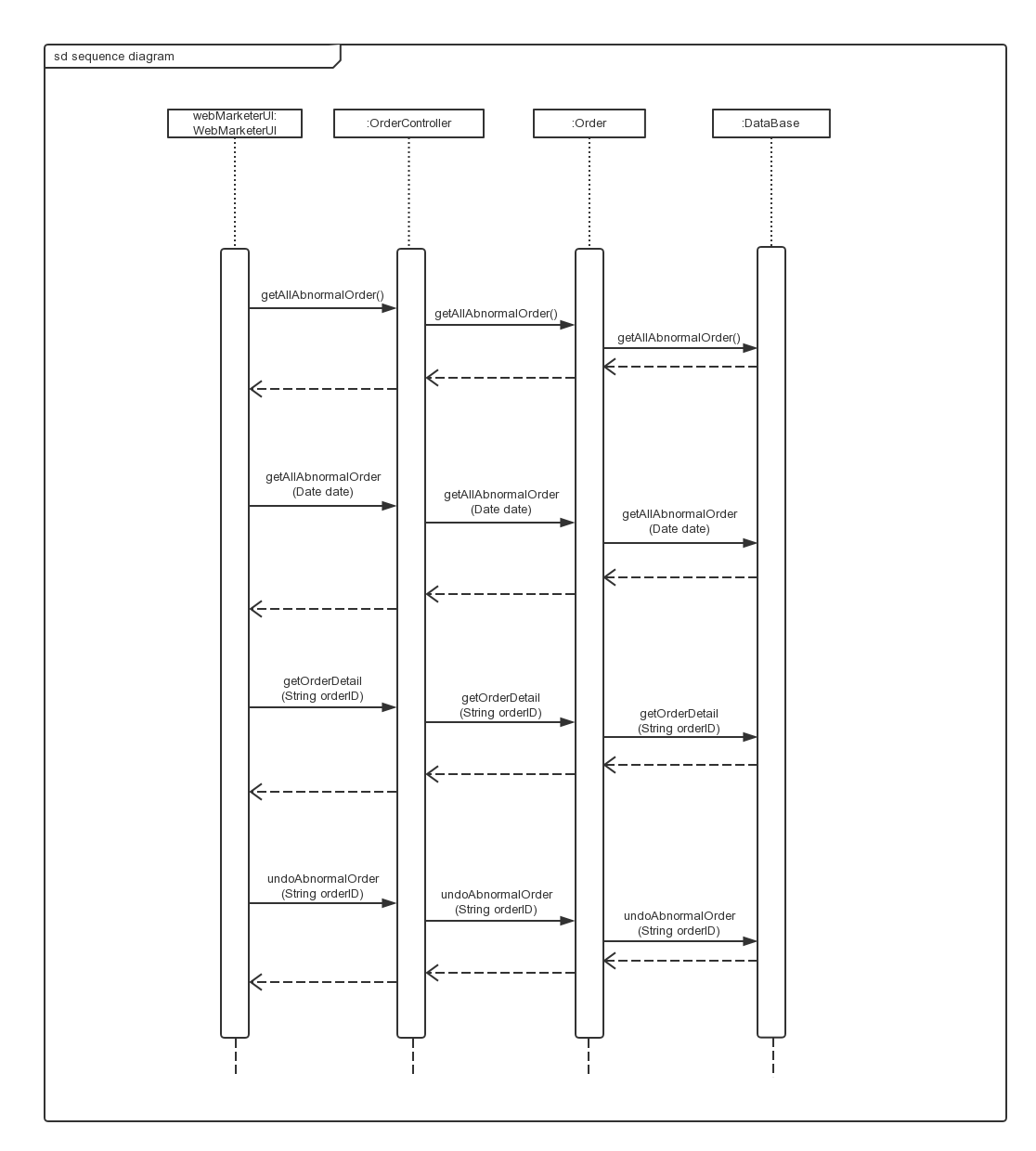
1. 业务逻辑层的动态模型



Order面向客户的顺序图



Order面向酒店工作人员的顺序图

 Order面向网站营销人员的顺序图

1. 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。

### hotel模块

1. 模块概述

hotel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。

hotel模块的职责和接口参见软件系统结构描述文档内对该模块的描述。

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。在展示层和业务逻辑层之间添加hotelBLService接口。为了隔离业务逻辑职责和逻辑控制职责，添加hotelBLService，这样hotelBLService会依赖于ExecutedOrderInfo、UnexecutedOrderInfo、AbnormalOrderInfo三个接口实现对订单执行以及退房的业务逻辑，同时hotelBLService将对酒店基本信息维护、客房信息维护、剩余客房信息查看、酒店促销策略维护、线下处理的业务逻辑委托给hotel对象。HotelPO、HotelWorkerPO、RoomInfo、RemainRoomInfo是作为酒店信息的持久性对象被添加到设计模型中去的。



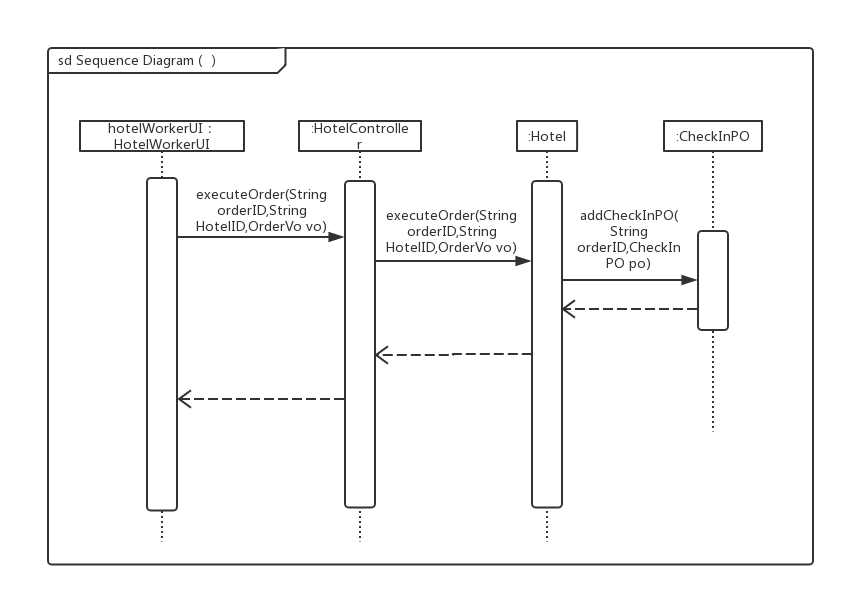
1. 模块内部类的接口规范
2. Controller类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HotelController.getHotelInfo | 语法 | public HotelVO getHotelInfo(String hotelWorkerID) |
| 前置条件 | 客户或酒店工作人员已被识别和授权 |
| 后置条件 | 无 |
| HotelController.updateHotelInfo | 语法 | public ResultMessage updateHotelInfo (HotelVO hotelVO) |
| 前置条件 | 酒店工作人员已被识别和授权 |
| 后置条件 | 系统更新并持久化保存酒店基本信息并提示酒店工作人员修改成功，否则提示错误 |
| HotelController.getHotelRoomInfo | 语法 | public RoomInfoVO getHotelRoomInfo (String hotelWorkerID) |
| 前置条件 | 酒店工作人员已被识别和授权 |
| 后置条件 | 无 |
| HotelController.updateHotelRoomInfo | 语法 | public ResultMessage updateHotelRoomInfo (RoomInfoVO RoomInfoVO) |
| 前置条件 | 酒店工作人员已被识别和授权 |
| 后置条件 | 系统更新并持久化保存酒店基本信息并提示酒店工作人员修改成功，否则提示错误 |
| HotelController.updateCheckIn | 语法 | public ResultMessage updateCheckIn (String orderID, List<String> roomNum, LocalDateTime inTime, LocalDateTime expectedLeaveTime) |
| 前置条件 | 酒店工作人员已被识别和授权 |
| 后置条件 | 系统更新并持久化保存入住信息并提示酒店工作人员更新成功，否则提示失败 |
| HotelController.updateCheckOut | 语法 | public ResultMessage updateCheckOut (String orderID, LocalDateTime outTime) |
| 前置条件 | 酒店工作人员已被识别和授权 |
| 后置条件 | 系统更新并持久化保存退房信息并提示酒店工作人员更新成功，否则提示失败 |
| HotelController.getRemainRoomInfo | 语法 | public List<RoomInfoVO> getRemainRoomInfo (String hotelWorkerID) |
| 前置条件 | 酒店工作人员已被识别和授权 |
| 后置条件 | 无 |
| HotelController.updateRemainRoomInfo | 语法 | public ResultMessage updateRemainRoomInfo (Operation operation, Map<RoomType, Integer> roomInfo) |
| 前置条件 | 酒店工作人员已被识别和授权 |
| 后置条件 | 系统更新并持久化保存剩余客房信息并提示酒店工作人员更新成功，否则提示失败 |
| HotelController.add | 语法 | public ResultMessage add (HotelVO HotelVO) |
| 前置条件 | 网站管理人员已被识别和授权 |
| 后置条件 | 系统添加并持久化保存酒店信息并提示添加成功，否则提示错误 |
| HotelController.getHotelList | 语法 | public List<HotelGeneralVO> getHotelList (AddressVO addressVO) |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 无 |
| HotelController.getSortedHotels | 语法 | public List<HotelGeneralVO> getSortedHotels (SortStrategy sortStrategy); |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 无 |
| HotelController.getBookedHotels | 语法 | public List<HotelGeneralVO> getBookedHotels (String userID) |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 系统更新并持久化保存剩余客房信息并提示酒店工作人员更新成功，否则提示失败 |
| HotelController.getHotelDetail | 语法 | public HotelVO getHotelDetail (String hotelID) |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 无 |
| HotelController. getUncommentedHotels | 语法 | public List<HotelGeneralVO> getUncommentedHotels (String userID) |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 无 |
| HotelController.updateEvaluation | 语法 | public ResultMessage updateEvaluation (EvaluationVO evaluationVO) |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 系统更新订单中酒店评分评价并提示评价更新更新成功，否则提示失败 |
| HotelController.getPromotions | 语法 | public List<PromotionVO> getPromotions (String hotelID) |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 无 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| Hotel.getHotelInfo | 得到酒店基本信息 | |
| Hotel.updateHotelInfo | 更新酒店基本信息 | |
| Hotel.getHotelRoomInfo | 得到酒店客房信息 | |
| Hotel.updateHotelRoomInfo | 更新酒店客房信息 | |
| Hotel.updateCheckIn | 更新入住信息 | |
| Hotel.updateCheckOut | 更新退房信息 | |
| Hotel.  getRemainRoomInfo | 得到酒店的剩余客房信息 | |
| Hotel.updateRemainRoomInfo | 更新酒店的剩余客房信息 | |
| Hotel.add | 添加酒店 | |
| Hotel.getHotelList | 得到指定地址的酒店列表 | |
| Hotel.getSortedHotels | 得到指定排序方式后的酒店列表 | |
| Hotel.getBookedHotels | 得到客户已经预定的酒店列表 | |
| Hotel.getHotelDetail | 得到酒店详情 | |
| Hotel.getUncommentedHotels | 更新客户未评价的酒店列表 | |
| Hotel.updateEvaluation | 更新订单评价及酒店评分评价 | |
| Hotel.getPromotions | 得到指定酒店的促销策略 | |

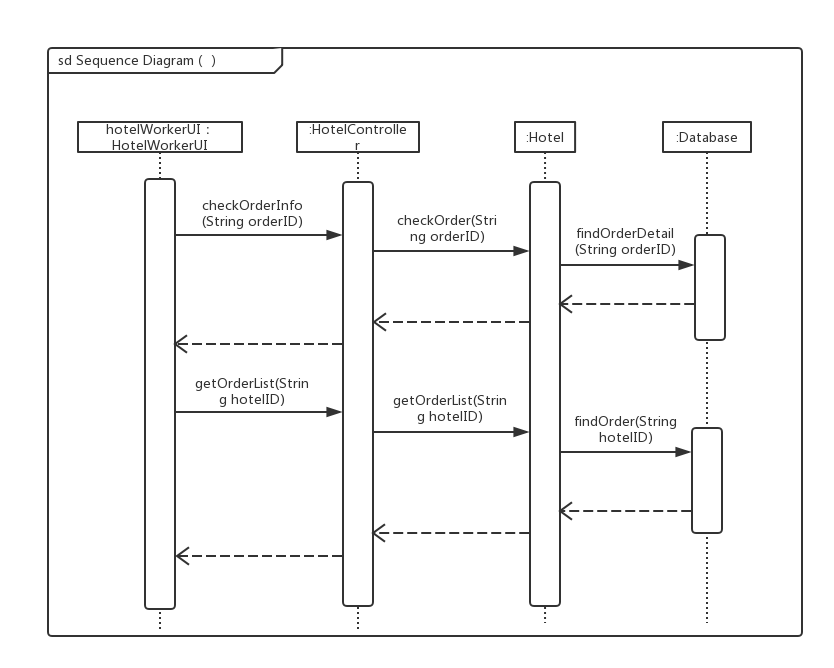
1. Domain类的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 提供的服务（供接口） | | | | |
| Hotel.getHotelInfo | | | 语法 | public HotelVO getHotelInfo(String hotelWorkerID) |
| 前置条件 | 客户或酒店工作人员已被识别和授权 |
| 后置条件 | 无 |
| Hotel.updateHotelInfo | | | 语法 | public ResultMessage updateHotelInfo (HotelVO hotelVO) |
| 前置条件 | 酒店工作人员已被识别和授权 |
| 后置条件 | 系统更新并持久化保存酒店基本信息并提示酒店工作人员修改成功，否则提示错误 |
| Hotel.getHotelRoomInfo | | | 语法 | public RoomInfoVO getHotelRoomInfo (String hotelWorkerID) |
| 前置条件 | 酒店工作人员已被识别和授权 |
| 后置条件 | 无 |
| Hotel.updateHotelRoomInfo | | | 语法 | public ResultMessage updateHotelRoomInfo (RoomInfoVO RoomInfoVO) |
| 前置条件 | 酒店工作人员已被识别和授权 |
| 后置条件 | 系统更新并持久化保存酒店基本信息并提示酒店工作人员修改成功，否则提示错误 |
| Hotel.updateCheckIn | | | 语法 | public ResultMessage updateCheckIn (String orderID, List<String> roomNum, LocalDateTime inTime, LocalDateTime expectedLeaveTime) |
| 前置条件 | 酒店工作人员已被识别和授权 |
| 后置条件 | 系统更新并持久化保存入住信息并提示酒店工作人员更新成功，否则提示失败 |
| Hotel.updateCheckOut | | | 语法 | public ResultMessage updateCheckOut (String orderID, LocalDateTime outTime) |
| 前置条件 | 酒店工作人员已被识别和授权 |
| 后置条件 | 系统更新并持久化保存退房信息并提示酒店工作人员更新成功，否则提示失败 |
| Hotel.getRemainRoomInfo | | | 语法 | public List<RoomInfoVO> getRemainRoomInfo (String hotelWorkerID) |
| 前置条件 | 酒店工作人员已被识别和授权 |
| 后置条件 | 无 |
| Hotel.updateRemainRoomInfo | | | 语法 | public ResultMessage updateRemainRoomInfo (Operation operation, Map<RoomType, Integer> roomInfo) |
| 前置条件 | 酒店工作人员已被识别和授权 |
| 后置条件 | 系统更新并持久化保存剩余客房信息并提示酒店工作人员更新成功，否则提示失败 |
| Hotel.add | | | 语法 | public ResultMessage add (HotelVO HotelVO) |
| 前置条件 | 网站管理人员已被识别和授权 |
| 后置条件 | 系统添加并持久化保存酒店信息并提示添加成功，否则提示错误 |
| Hotel.getHotelList | | | 语法 | public List<HotelGeneralVO> getHotelList (AddressVO addressVO) |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 无 |
| Hotel.getSortedHotels | | | 语法 | public List<HotelGeneralVO> getSortedHotels (SortStrategy sortStrategy); |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 无 |
| Hotel.getBookedHotels | | | 语法 | public List<HotelGeneralVO> getBookedHotels (String userID) |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 系统更新并持久化保存剩余客房信息并提示酒店工作人员更新成功，否则提示失败 |
| Hotel.getHotelDetail | | | 语法 | public HotelVO getHotelDetail (String hotelID) |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 无 |
| Hotel.getUncommentedHotels | | | 语法 | public List<HotelGeneralVO> getUncommentedHotels (String userID) |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 无 |
| Hotel.updateEvaluation | | | 语法 | public ResultMessage updateEvaluation (EvaluationVO evaluationVO) |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 系统更新订单中酒店评分评价并提示评价更新更新成功，否则提示失败 |
| Hotel.getPromotions | | | 语法 | public List<PromotionVO> getPromotions (String hotelID) |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 无 |
| 需要的接口（需接口） | | | | |
| 服务名 | **服务内容** | | | |
| HotelDataService.getHotelList(AddressPO addressPO) | 得到数据库里的当前商圈、地址信息里的所有酒店列表 | | | |
| HotelDataService.getHotelDetail(String hotelID) | | 得到数据库里的对应当前酒店编号的详细信息 | | |
| HotelDataService.addEvaluation(EvaluationPO evaluationPO) | | 在数据库里给对应当前酒店编号的酒店添加评论 | | |
| HotelDataService.getHotelInfo(String hotelID) | | 从数据库里获得本酒店的酒店基本信息 | | |
| HotelDataService.setHotelInfo(HotelInfoPO  hotelInfoPO) | | 修改数据库里本酒店酒店基本信息 | | |
| HotelDataService.getHotelRoomInfo(String hotelID) | | 从数据库里获得本酒店的客房信息 | | |
| HotelDataService.setHotelRoomInfo (HotelRoomInfoPO hotelRoomInfoPO) | | 修改数据库里本酒店的客房信息 | | |
| HotelDataService.setCheckInInfo(CheckInInfo checkInInfo) | | 修改数据库里订单的入住信息 | | |
| HotelDataService.setCheckOutInfo(CheckOutInfo checkOutInfo) | | 修改数据库里该订单的退房时间 | | |
| HotelDataService.getRemainRoomInfo(String hotelID) | | 获取数据库里该酒店的剩余房间信息 | | |
| HotelDataService.setRemainRoomInfo  (RemainRoomInfoPO remainRoomInfoPO) | | 修改数据库里该酒店的剩余房间信息 | | |
| HotelDataService.add(HotelInfoPO hotelInfoPO) | | 在数据库里添加一条就酒店的信息 | | |
| Promotion.getPromotions(String userID) | | 依据酒店工作人员编号得到数据库里当前酒店的所有策略信息列表 | | |
| Order.getBookedHotels (String guestID) | 从所有订单中得到该用户预定过的酒店 | | | |

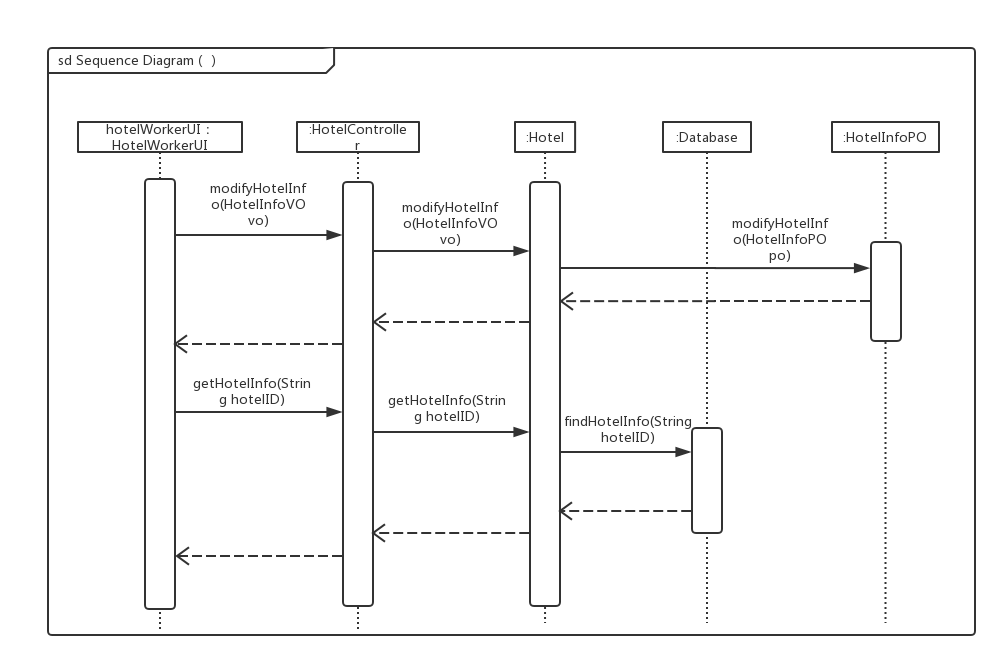
1. 业务逻辑层的动态模型



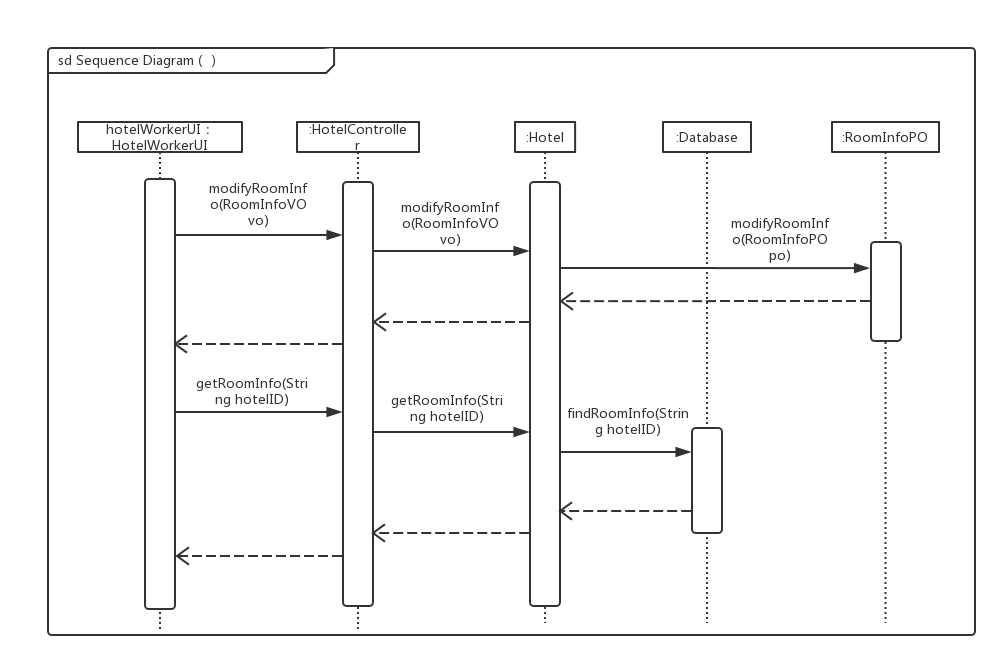
订单执行及更新入住信息顺序图



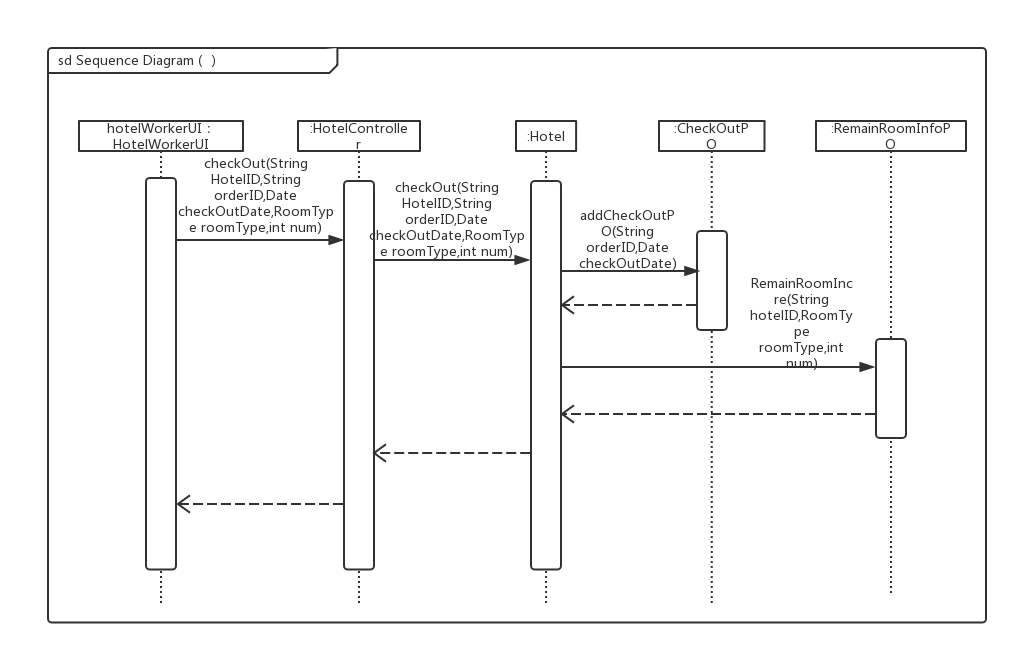
酒店订单浏览顺序图



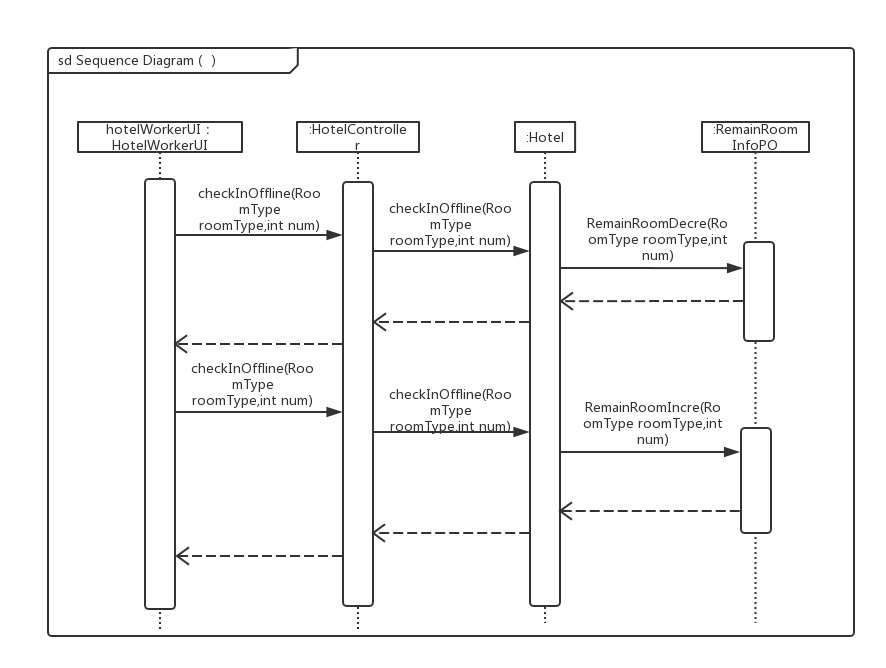
酒店基本信息维护顺序图



客房信息维护顺序图



退房及更新退房信息、剩余客房信息



线下客户入住和退房处理顺序图

1. 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。

### promotion模块

1. 模块概述

promotion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。

promotion模块的职责和接口参见软件系统结构描述文档内对该模块的描述。

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。在展示层和业务逻辑层之间添加promotionBLService接口。为了隔离业务逻辑职责和逻辑控制职责，添加promotionController，这样promotionController会将对查看及维护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性对象被添加到设计模型中去的。



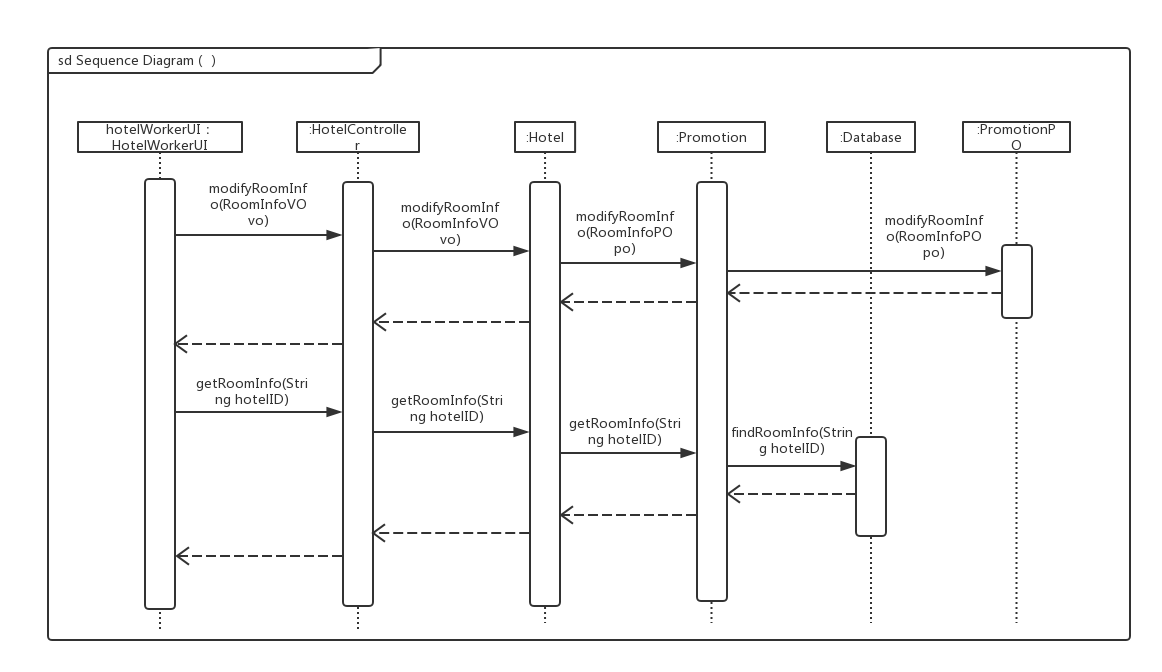
1. 模块内部类的接口规范
2. Controller类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PromotionController.getPromotions | 语法 | Public List<promotionVO> getPromotions (String userID) |
| 前置条件 | 已知用户编号（可以是酒店工作人员，也可以是网站营销人员），需要维护促销策略 |
| 后置条件 | 系统返回当前用户所管理的促销策略的列表 |
| PromotionController.update | 语法 | Public ResultMessage update (List<PromotionVO> list) |
| 前置条件 | 用户（酒店工作人员或网站营销人员）确认更新促销策略，已知当前的促销策略的列表 |
| 后置条件 | 更新并持久化保存当前促销策略，并提示用户更新是否成功 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| Promotion.getPromotions | 得到当前用户所管理的促销策略列表 | |
| Promotion.update | 更新促销策略 | |

1. Domain类的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 提供的服务（供接口） | | | | |
| Promotion.getPromotions | 语法 | | Public List<promotionVO> getPromotions (String userID) | |
| 前置条件 | | 已知酒店工作人员或网站营销人员的编号 | |
| 后置条件 | | 返回促销策略列表 | |
| Promotion.update | 语法 | | Public boolean update (String userID) | |
| 前置条件 | | 已知酒店工作人员或网站营销人员的编号 | |
| 后置条件 | | 返回更新是否成功的布尔值 | |
| 需要的接口（需接口） | | | | |
| 服务名 | **服务内容** | | | |
| PromotionDataService.  getPromotions(String userID) | | 根据提供的用户编号，找到并返回数据库中对应用户所管理的策略列表 | |
| PromotionDataService.  setPromotions(String userID,List<promotionPO> list) | | 根据持有一系列promotionPO的list对象修改数据库中对应用户所管理的策略列表 | |

1. 业务逻辑层的动态模型



酒店促销策略维护顺序图

1. 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。

### credit模块

1)模块概述

credit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。

credit模块的职责和接口参见软件系统结构描述文档内对该模块的描述。

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。在展示层和业务逻辑层之间添加creditBLService接口。为了隔离业务逻辑职责和逻辑控制职责，添加creditController，这样creditController会将对用户登录和客户注册的业务逻辑处理委托给credit对象。creditPO是作为账户信息的持久性对象被添加到设计模型中去的。



3)模块内部类的接口规范

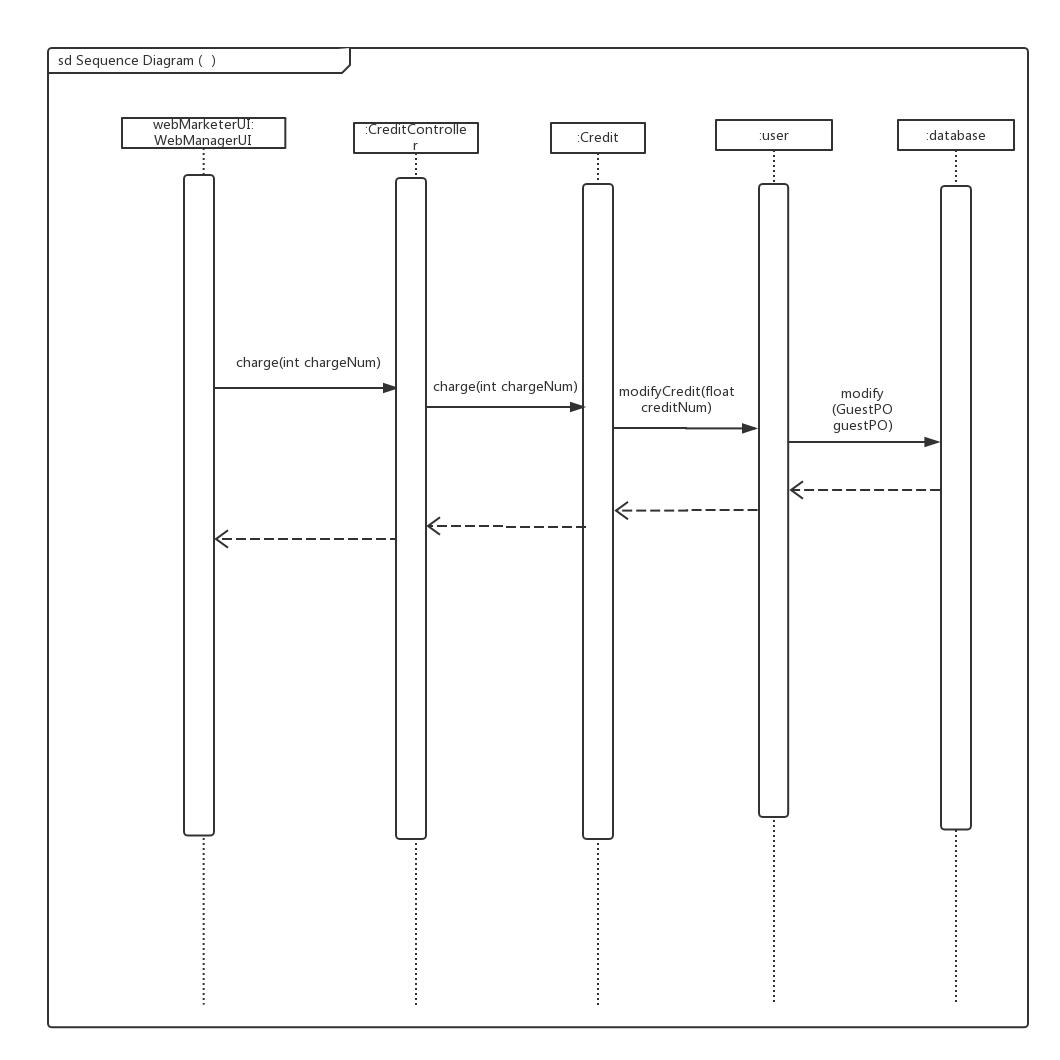
(1) Controller类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CreditController.charge | 语法 | public ResultMessage charge(String guestID, int chargeNum) |
| 前置条件 | 网站营销人员被识别和授权，并已知客户编号和欲充值的的信用值 |
| 后置条件 | 系统为客户信用值充值 |
| CreditController.getBasicInfo | 语法 | public BasicInfoVO getBasicInfo (String ID) |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 无 |
| CreditController.getAllCreditDetail | 语法 | public List<CreditVO> getAllCreditDetail (String guestID) |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 无 |
| CreditController.getMemberFormulation | 语法 | public List<MarketVO> getMemberFormulation () |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 无 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| Credit.charge | 为用户充值信用值 | |
| Credit.getBasicInfo | 得到客户的基本信息 | |
| Credit.getAllCreditDetail | 得到客户信用值变化详情的列表 | |
| Credit.getMemberFormulation | 得到会员等级制度信息 | |

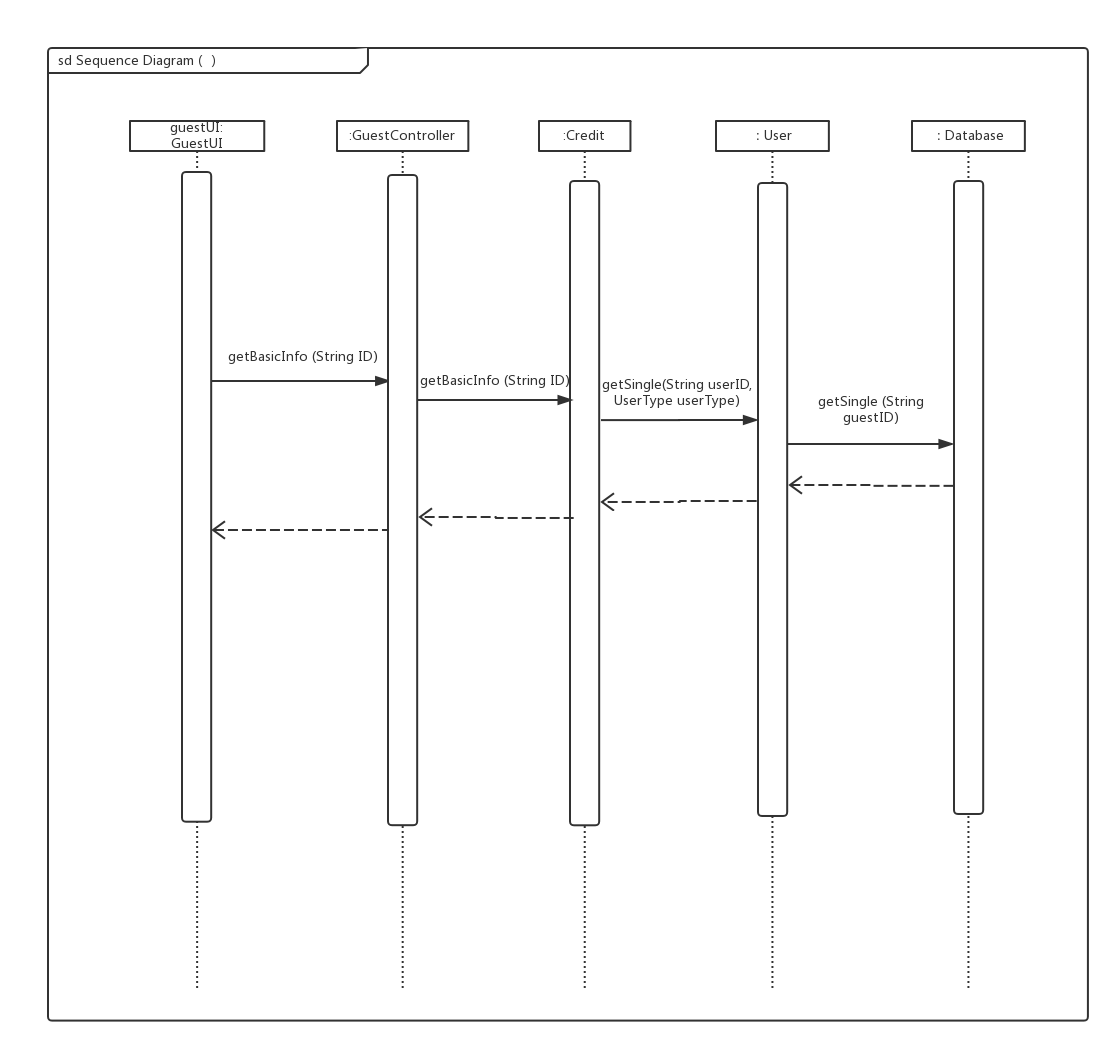
(2)Domain类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Credit.charge | 语法 | public ResultMessage charge(String guestID, int chargeNum) |
| 前置条件 | 网站营销人员被识别和授权，并已知客户编号和欲充值的的信用值 |
| 后置条件 | 系统为客户信用值充值 |
| Credit.getBasicInfo | 语法 | public BasicInfoVO getBasicInfo (String ID) |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 无 |
| Credit.getAllCreditDetail | 语法 | public List<CreditVO> getAllCreditDetail (String guestID) |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 无 |
| Credit.getMemberFormulation | 语法 | public List<MarketVO> getMemberFormulation () |
| 前置条件 | 客户已被识别和授权 |
| 后置条件 | 无 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| User.getSingle(String userID, UserType userType) | 返回与该编号对应的用户信息 | |
| User.getAllCreditDetail(String guestID) | 返回与该ID对应的信用值变化详情 | |
| User.modifyCredit(String guestID, float creditNum) | 为客户充值信用值 | |
| Market.getMemberFormulation() | 返回现有的会员等级制度信息 | |

4) 业务逻辑层的动态模型



信用充值顺序图



信用查看顺序图

1. 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。

### member模块

1. 模块概述

member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。

member模块的职责和接口参见软件系统结构描述文档内对该模块的描述。

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。在展示层和业务逻辑层之间添加memberBLService接口。为了隔离业务逻辑职责和逻辑控制职责，添加memberController，这样memberController会将对会员注册的业务逻辑处理委托给member对象。memberPO是作为会员信息的持久性对象被添加到设计模型中去的。



1. 模块内部类的接口规范

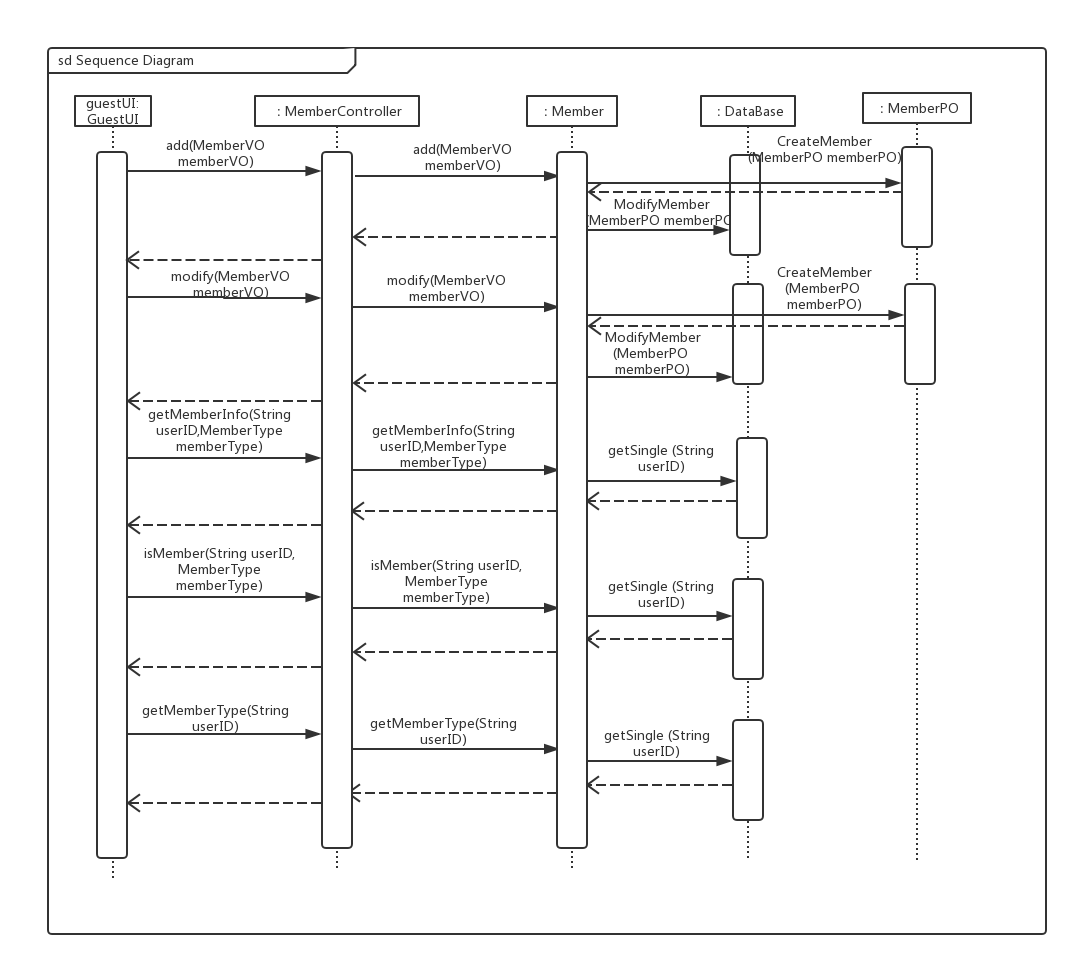
(1) Controller类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| MemberController.add | 语法 | public ResultMessage add(MemberVO memberVO) |
| 前置条件 | 客户已被识别和授权，输入的会员信息符合信息规范 |
| 后置条件 | 添加并持久化会员对象并提示添加成功，否则提示添加失败原因 |
| MemberController.modify | 语法 | public ResultMessage modify(MemberVO memberVO) |
| 前置条件 | 客户已被识别和授权，输入的会员信息符合信息规范 |
| 后置条件 | 系统修改客户会员信息并提示修改成功，否则提示修改失败原因 |
| MemberController. getMemberInfo | 语法 | public MemberVO getMemberInfo(String userID,MemberType memberType) |
| 前置条件 | 用户输入用户编号和会员类型 |
| 后置条件 | 系统返回与该userID和会员类型相对应的会员信息 |
| MemberController.isMember | 语法 | public boolean isMember(String userID, MemberType memberType) |
| 前置条件 | 用户已被识别和授权 |
| 后置条件 | 无 |
| MemberController.getMemberType | 语法 | public MemberType getMemberType(String userID, MemberType memberType) |
| 前置条件 | 用户输入用户编号和会员类型 |
| 后置条件 | 系统返回该用户的会员类型 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| Member.add | 系统增加会员信息 | |
| Member.modify | 系统修改会员信息 | |
| Member.getMemberInfo | 系统返回指定客户的会员信息 | |
| Member.isMember | 系统返回指定客户是否为会员 | |
| Member.getMemberType | 系统返回指定客户的会员类型 | |

(2)Domain类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Memeber.add | 语法 | public ResultMessage add(MemberVO memberVO) |
| 前置条件 | 会员信息符合规则且齐全 |
| 后置条件 | 系统根据已指定会员类型持久化为该客户增加会员数据 |
| Member.modify | 语法 | public ResultMessage modify(MemberVO memberVO) |
| 前置条件 | 会员信息符合规则且齐全 |
| 后置条件 | 系统修改该会员并持久化保存该客户的会员数据 |
| Member.getMemberInfo | 语法 | public MemberVO getMemberInfo(String userID,MemberType memberType) |
| 前置条件 | userID.Length=对应用户类型的ID长度 |
| 后置条件 | 系统返回与该userID对应的指定会员类型 |
| Member.isMember | 语法 | public boolean isMember(String userID, MemberType memberType) |
| 前置条件 | userID.Length=对应用户类型的ID长度 |
| 后置条件 | 返回该userID的用户是否为会员 |
| Member.getMemberType | 语法 | public MemberType getMemberType(String userID, MemberType memberType) |
| 前置条件 | userID.Length=对应用户类型的ID长度 |
| 后置条件 | 系统返回该userID的会员类型 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| GuestDataService.ModifyMember(MemberPO memberPO) | 根据提供的memberPO修改单一持久化客户对象的信用值 | |
| GuestDataService.  getSingle (String userID) | 根据ID查找客户持久化对象 | |

1. 业务逻辑层的动态模型



member模块设计顺序图



member模块设计状态图

1. 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。

### user模块

1. 模块概述

user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。

user模块的职责和接口参见软件系统结构描述文档内对该模块的描述。

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。在展示层和业务逻辑层之间添加userBLService接口。为了隔离业务逻辑职责和逻辑控制职责，添加userController，这样userController会将对用户信息的增改查和酒店添加的业务逻辑处理委托给user对象。guestPO、hotelWorkerPO、webMarketerPO、webManagerPO是作为用户信息的持久性对象被添加到设计模型中去的。



1. 模块内部类的接口规范

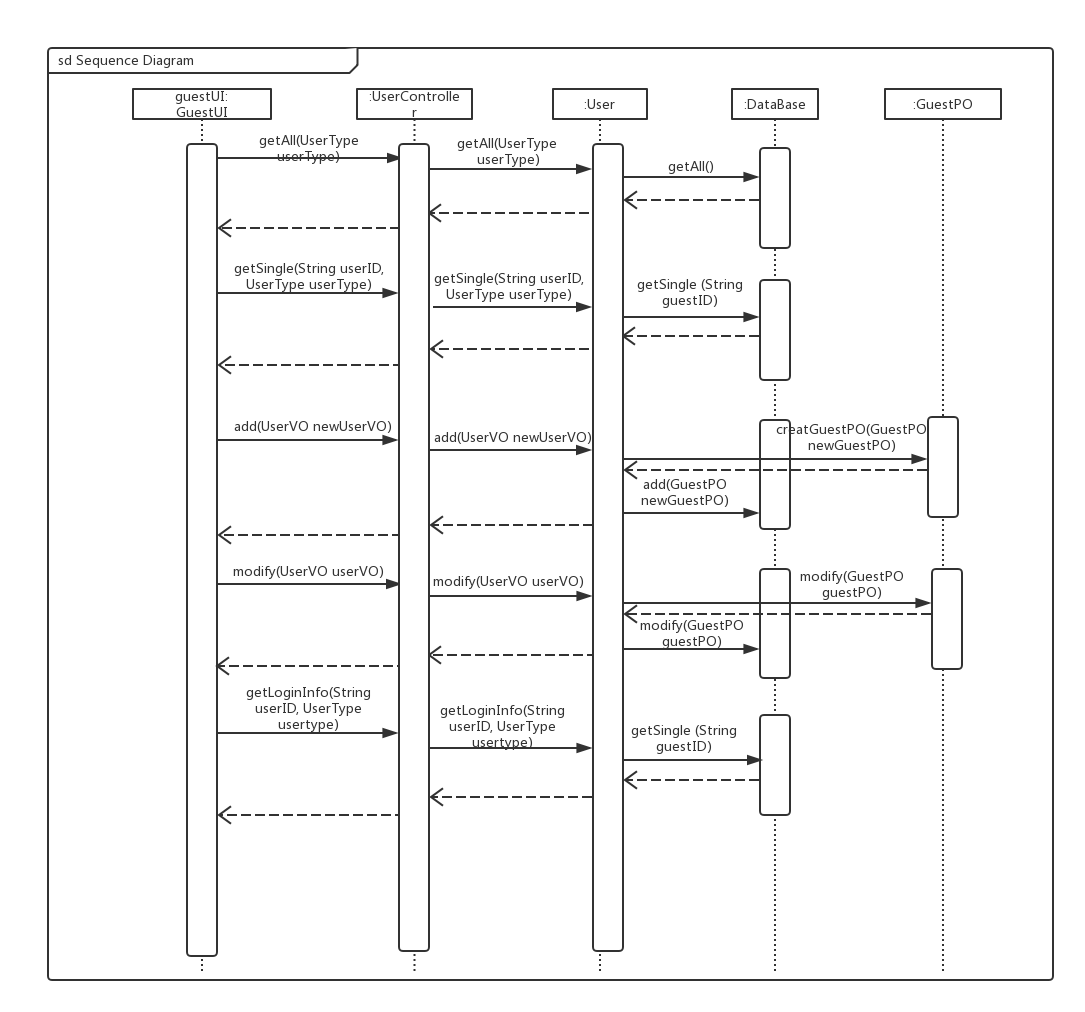
(1) Controller类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserController.add | 语法 | public ResultMessage add(UserVO newUserVO) |
| 前置条件 | 客户或网站管理人员已被识别和授权 |
| 后置条件 | 系统添加并持久化保存此对象，提示添加成功，否则提示添加错误 |
| UserController.modify | 语法 | public ResultMessage modify(UserVO newUserVO) |
| 前置条件 | 用户已被识别和授权 |
| 后置条件 | 系统修改用户信息并提示修改成功，否则提示修改失败原因 |
| UserController.getSingle | 语法 | public UserVO getSingle (String userID , UserType userType) |
| 前置条件 | 用户已被授权和识别 |
| 后置条件 | 无 |
| UserController.addHotel | 语法 | public ResultMessage addHotel(HotelVO hotelVO, String hotalID) |
| 前置条件 | 网站管理人员已被授权和识别 |
| 后置条件 | 系统成功添加酒店信息并提示添加成功，否则提示添加失败原因 |
| UserController.modifyCredit | 语法 | public ResultMessage modifyCredit (String guestID ,float creditNum) |
| 前置条件 | 网站营销人员已被识别和授权，输入的客户编号和更改后的信用值合法 |
| 后置条件 | 系统修改客户信用值并提示修改成功，否则提示修改失败原因 |
| UserController.getAll | 语法 | public List<UserVO> getAll(UserType userType) |
| 前置条件 | 用户已被授权和识别 |
| 后置条件 | 无 |
| UserController.getAllCreditDetail | 语法 | public List<CreditVO> getAllCreditDetail(String guestID) |
| 前置条件 | 用户已被授权和识别 |
| 后置条件 | 无 |
| UserController.getLogInInfo | 语法 | public ResultMessage getLoginInfo(String userID, UserType userType) |
| 前置条件 | 用户已被授权和识别，输入的信息合法 |
| 后置条件 | 无 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| UserController.add | 系统添加用户信息 | |
| UserController.modify | 系统修改用户信息 | |
| UserController.getSingle | 系统返回单一用户的信息 | |
| UserController.addHotel | 系统添加酒店信息 | |
| UserController.modifyCredit | 系统修改指定客户的信用值 | |
| UserController.getAll | 系统返回指定用户类型的所有用户的信息 | |
| UserController.getALLCreditDetail | 系统返回指定客户的所有信用详情 | |
| UserController.getLoginInfo | 系统返回指定用户相关的登录信息 | |

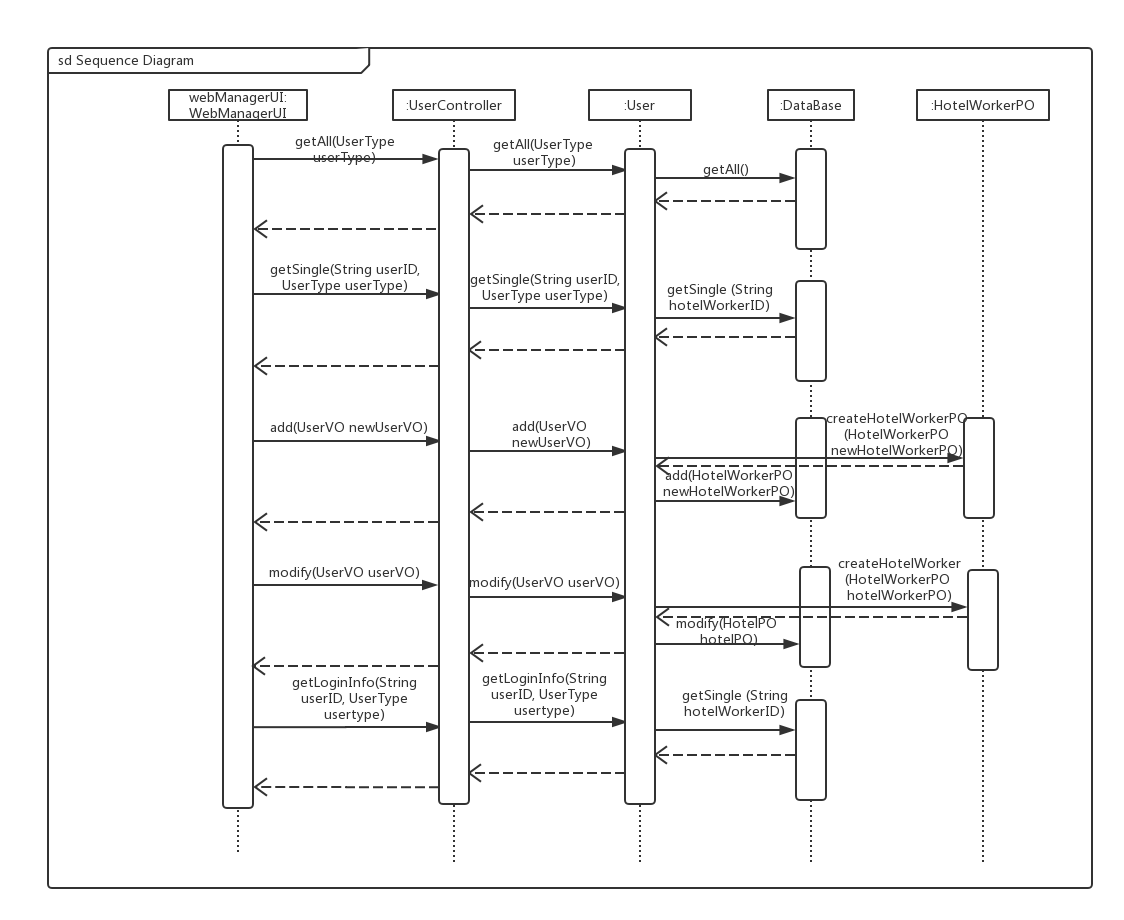
(2)Domain类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| User.add | 语法 | public ResultMessage add(UserVO newUserVO) |
| 前置条件 | 客户或网站管理人员已被识别和授权 |
| 后置条件 | 系统添加并持久化保存此对象，提示添加成功，否则提示添加错误 |
| User.modify | 语法 | public ResultMessage modify(UserVO newUserVO) |
| 前置条件 | 用户已被识别和授权 |
| 后置条件 | 系统修改用户信息并提示修改成功，否则提示修改失败原因 |
| User.getSingle | 语法 | public UserVO getSingle(String userID，UserType userType) |
| 前置条件 | 用户已被识别和授权 |
| 后置条件 | 无 |
| User.addHotel | 语法 | public ResultMessage addHotel(HotelVO newHotelVO, String hotelID) |
| 前置条件 | 网站管理人员已被识别和授权 |
| 后置条件 | 系统成功添加酒店信息并提示添加成功，否则提示添加失败原因 |
| User.modifyCredit | 语法 | public ResultMessage modifyCredit (String guestID, float creditNum) |
| 前置条件 | 网站营销人员为客户信用充值 |
| 后置条件 | 系统修改该客户信用值并持久化保存该用户数据 |
| User.getAll | 语法 | public List<UserVO> getAll(UserType userType) |
| 前置条件 | 用户已被授权和识别 |
| 后置条件 | 无 |
| User.getAllCreditDetail | 语法 | public List<CreditVO> getAllCreditDetail (String guestID) |
| 前置条件 | 用户已被授权和识别，guestID.Length=客户ID长度12 |
| 后置条件 | 无 |
| User.getLoginInfo | 语法 | public ResultMessage getLoginInfo(String userID, UserType userType) |
| 前置条件 | 用户已被授权和识别 |
| 后置条件 | 无 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| guestDataService.add(GuestPO newGuestPO) | 根据提供的newGuestPO增加客户持久化对象 | |
| guestDataService.getSingle (String guestID) | 根据guestID查找客户持久化对象 | |
| guestDataService.getAllGuest () | 获取所有客户持久化对象 | |
| guestDataService.getAllCreditDetail(String guestID) | 获取所有客户信用值的持久化对象 | |
| guestDataService.modify(GuestPO guestPO) | 修改单一客户持久化对象 | |
| hotelWorkerDataService.add(HotelWorkerPO newHotelWorkerPO) | 根据提供的newHotelWorkerPO增加酒店工作人员持久化对象 | |
| hotelWorkerDataService.getSingle (String hotelWorkerID) | 根据hotelWorkerID查找酒店工作人员持久化对象 | |
| hotelWorkerDataService.getAllHotelWorker() | 获取所有酒店工作人员持久化对象 | |
| hotelWorkerDataService.modify(HotelPO hotelPO) | 修改单一酒店工作人员持久化对象 | |
| hotelWorkerDataService.init() | 清空所有酒店工作人员持久化对象 | |
| webMarketerDataService.add(WebMarketerPO newWebMarketerPO) | 根据提供的newWebMarkterPO增加网站营销人员持久化对象 | |
| webMarkterDataService.  getSingle (String webMarketerID) | 根据webMarketerID查找网站营销人员持久化对象 | |
| webMarkterDataService.getAllWebMarketer () | 获取所有网站营销人员持久化对象 | |
| webMarkterDataService.modify(WebMarketerPO webMarketerPO) | 修改单一网站营销人员持久化对象 | |
| webMarketerDataService.init() | 清空所有网站营销人员持久化对象 | |
| webManagerDataService.add(WebManagerPO newWebManagerPO) | 根据提供的newWebManagerPO增加网站管理人员持久化对象 | |
| webManagerDataService.getSingle (String webManagerID) | 根据webManagerID查找网站管理人员持久化对象 | |
| webManagerDataService.getAllWebManager () | 获取所有网站管理人员持久化对象 | |
| webManagerDataService.modify(WebManagerPO webManagerPO) | 修改单一网站管理人员持久化对象 | |
| webManagerDataService.init() | 清空所有网站管理人员持久化对象 | |
| Hotel.add | 新增该酒店并持久化保存该酒店的数据 | |

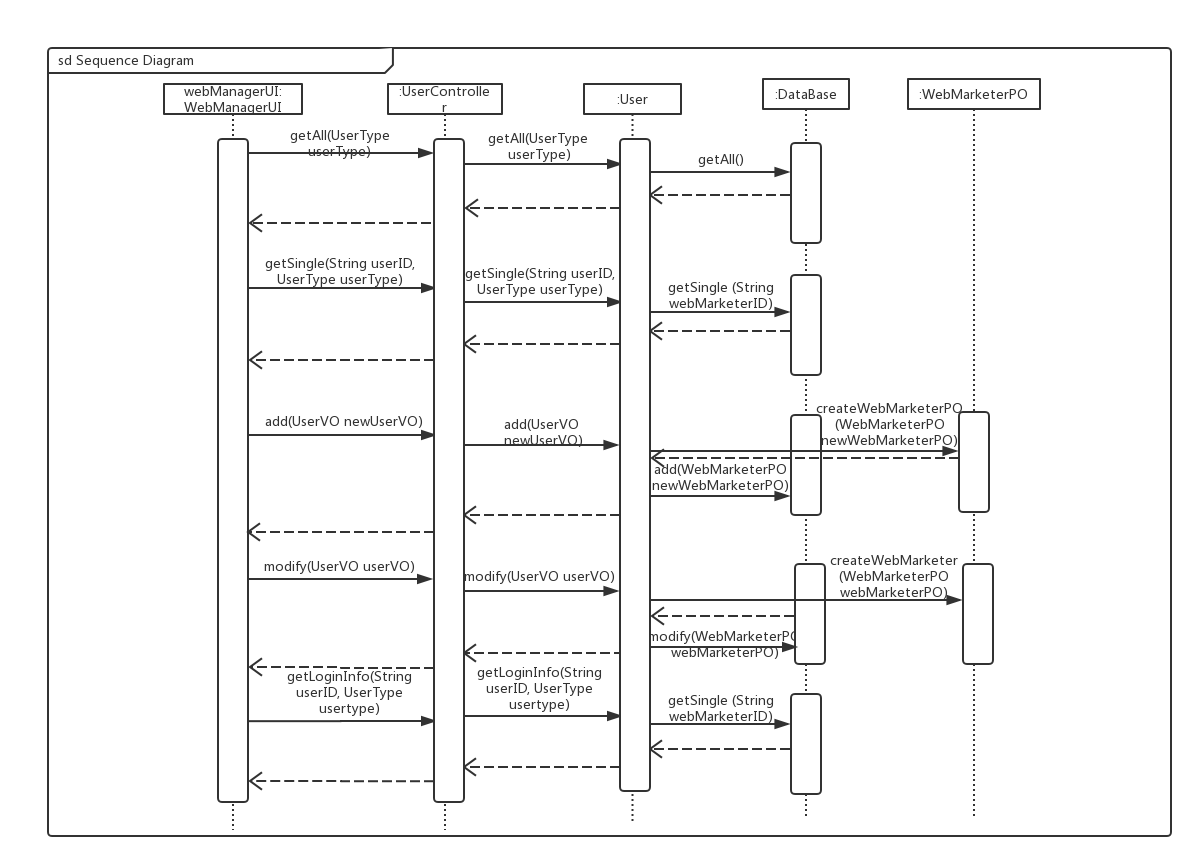
1. 业务逻辑层的动态模型



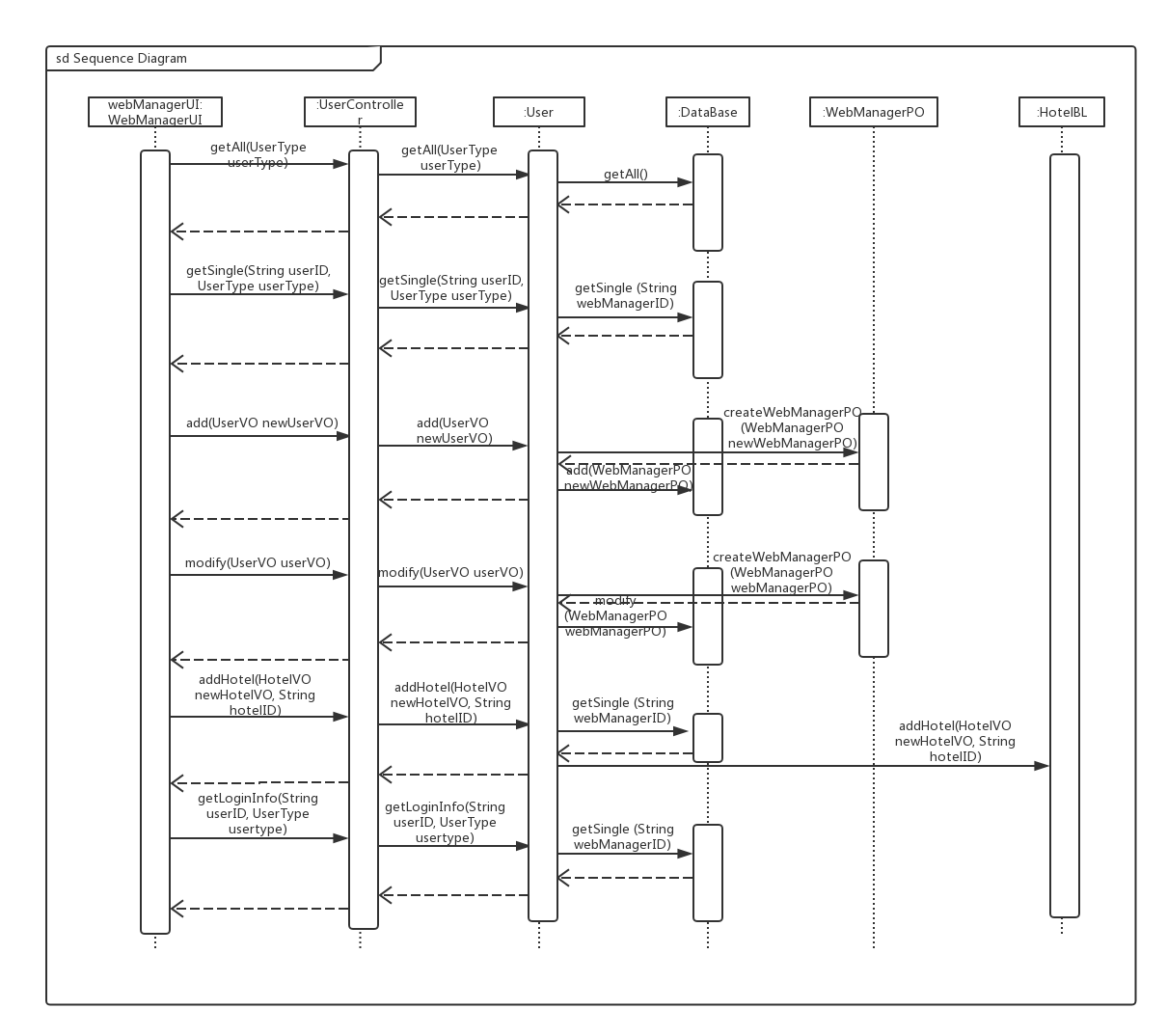
user模块设计顺序图（guest）



user模块设计顺序图（hotelWorker）



user模块设计顺序图（webMarketer）



user模块设计顺序图（webManager）



user模块设计状态图

1. 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。

### market模块

1)模块概述

market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。

market模块的职责和接口参见软件系统结构描述文档内对该模块的描述。

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。在展示层和业务逻辑层之间添加marketBLService接口。为了隔离业务逻辑职责和逻辑控制职责，添加marketController，这样marketController会将对用户登录和客户注册的业务逻辑处理委托给market对象。marketPO是作为账户信息的持久性对象被添加到设计模型中去的。



3)模块内部类的接口规范

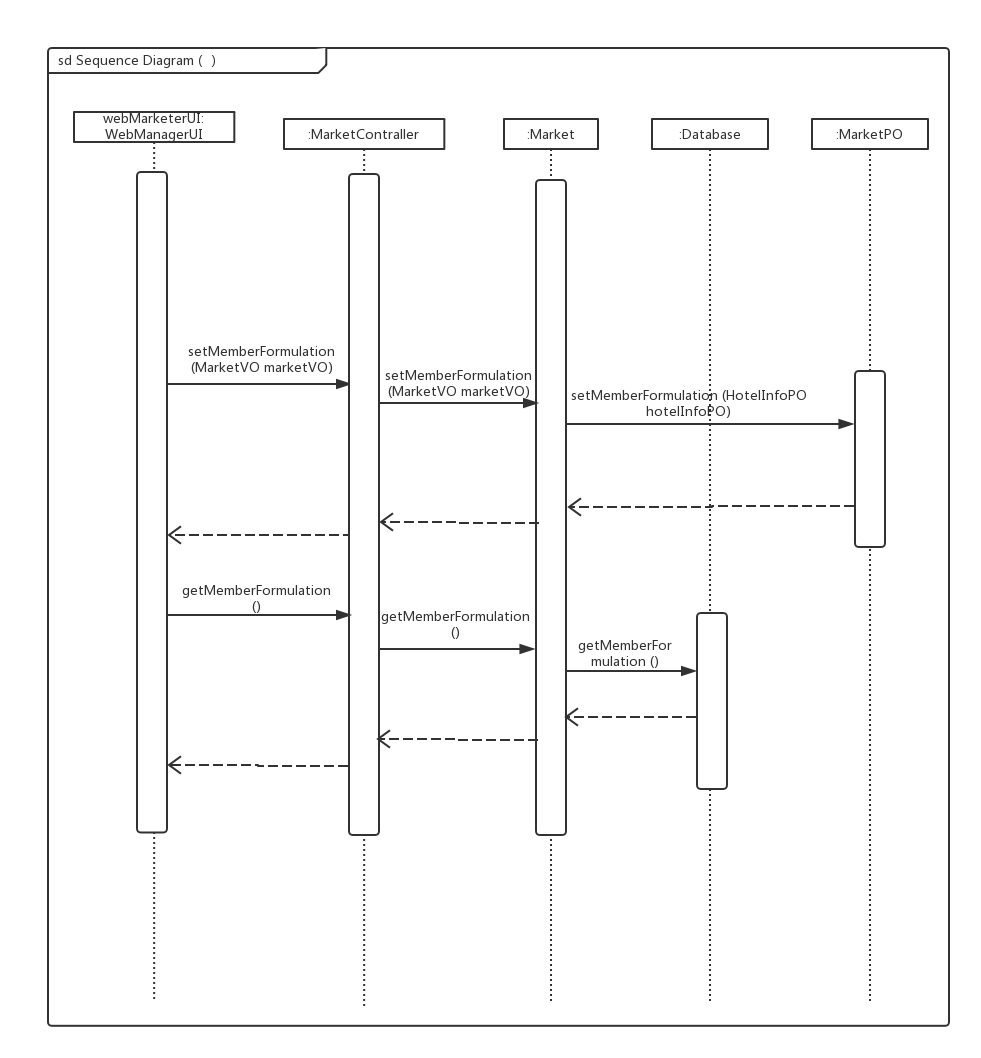
(1) Controller类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| marketController.getMemberFormulation | 语法 | public List<MarketVO> getMemberFormulation () |
| 前置条件 | 用户已被识别和授权 |
| 后置条件 | 无 |
| marketController.setMemberFormulation | 语法 | public ResultMessage setMemberFormulation (List<MarketVO> marketVO) |
| 前置条件 | 网站营销人员已被识别和授权，输入会员等级制度合法 |
| 后置条件 | 系统修改、记录会员等级制度信息并提示修改成功，否则提示错误 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| market.getMemberFormulation | 返回现有的会员等级制度信息 | |
| market.setMemberFormulation | 修改会员等级制度信息 | |

(2)Domain类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| market.getMemberFormulation | 语法 | public List<MarketVO> getMemberFormulation () |
| 前置条件 | 用户已被识别和授权 |
| 后置条件 | 无 |
| market.setMemberFormulation | 语法 | public ResultMessage setMemberFormulation (List<MarketVO> marketVO) |
| 前置条件 | 网站营销人员已被识别和授权，输入会员等级制度合法 |
| 后置条件 | 系统修改、记录会员等级制度信息并提示修改成功，否则提示错误 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| marketDataService.getMemberFormulation | 返回现有的会员等级制度信息 | |
| marketDataService.setMemberFormulation(MarketVO marketVO) | 修改会员等级制度信息 | |

4) 业务逻辑层的动态模型



会员等级制定与查看顺序图

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。

## 数据层的分解

数据层模块的分解详见需求规格文档中的相关描述。

### orderData模块

1)模块概述

orderData模块负责保存订单信息，进行增，改，查的操作。

2)模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| orderData.createOrder | 语法 | public ResultMessage createOrder (OrderPO order) |
| 前置条件 | 不存在相同的orderId |
| 后置条件 | 数据库中添加一条订单数据 |
| orderData.getOrderDetails | 语法 | public OrderPO getOrderDetail (String orderID) |
| 前置条件 | 无 |
| 后置条件 | 系统返回对应id的OrderPO,如果id不存在，返回Null |
| orderData.getAllGuestOrderGeneral | 语法 | public List<OrderGeneralPO> getAllGuestOrderGeneral (GuestID GuestID) |
| 前置条件 | 无 |
| 后置条件 | 系统返回对应id的OrderPO,如果id不存在，返回Null |
| orderData. getAllHotelOrderGeneral | 语法 | public List<OrderGeneralPO> getAllHotelOrderGeneral (GuestID GuestID) |
| 前置条件 | 无 |
| 后置条件 | 系统返回对应id的OrderPO,如果id不存在，返回Null |
| orderData. executeOrder | 语法 | public ResultMessage executeOrder (String orderID) |
| 前置条件 | 存在相同的orderId且订单状态为未执行 |
| 后置条件 | 数据库中更改未执行订单的状态为已执行 |
| orderData. undoAbnormalOrder | 语法 | public ResultMessage undoAbnormalOrder (String orderID) |
| 前置条件 | 存在相同的orderId且订单状态为异常 |
| 后置条件 | 数据库中更改并保存订单的状态数据 |
| orderData. undoNormalOrder | 语法 | public ResultMessage undoNormalOrder (String orderID) |
| 前置条件 | 存在相同的orderId且订单状态为未执行 |
| 后置条件 | 数据库中更改并保存订单的状态 |

### hotelData模块

1)模块概述

hotelData模块负责保存酒店信息，进行增，改，查的操作。

2)模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| hotelData.getHotelList | 语法 | public List<HotelGeneralPO> getHotelList(AddressPO addressPO) |
| 前置条件 | 已知商圈和地址信息 |
| 后置条件 | 系统在数据库中查询并返回符合该商圈和地址信息的所有酒店列表 |
| hotelData.getHotelDetail | 语法 | public HotelPO getHotelDetail(String hotelID) |
| 前置条件 | 已知客户点击查看的酒店编号 |
| 后置条件 | 返回保存酒店详细信息的HotelPO对象 |
| hotelData.addEvaluation | 语法 | public boolean addEvaluation (EvaluationPO evaluationPO) |
| 前置条件 | 已知评论信息的对象evaluationPO以及被评论的酒店编号 |
| 后置条件 | 系统向数据库中添加evaluationPO，并返回添加是否成功的布尔值 |
| hotelData.getHotelInfo | 语法 | public HotelInfoPO getHotelInfo (String hotelID) |
| 前置条件 | 已知酒店编号，逻辑层想获取酒店的基本信息 |
| 后置条件 | 返回一个保存有该酒店基本信息的HotelInfoPO对象 |
| hotelData.setHotelInfo | 语法 | public boolean  setHotelInfo(HotelInfoPO hotelInfoPO) |
| 前置条件 | 已知保存有酒店基本信息的HotelInfoPO对象，逻辑层请求修改该酒店的基本信息 |
| 后置条件 | 系统在数据库中更新该HotelInfoPO对象，并返回一个修改是否成功的布尔值 |
| hotelData.getHotelRoomInfo | 语法 | public List<RoomInfoVO> getHotelRoomInfo(String hotelID) |
| 前置条件 | 已知酒店编号，逻辑层请求查看该酒店的客房信息 |
| 后置条件 | 系统返回一个保存有酒店客房信息的HotelRoomInfoPO的对象 |
| hotelData.setHotelRoomInfo | 语法 | public boolean setHotelRoomInfo  (List<RoomInfoVO> list) |
| 前置条件 | 已知一个保存有酒店客房信息的HotelRoomInfoPO对象，逻辑层请求更新数据库的酒店客房信息 |
| 后置条件 | 系统更新数据库里的HotelRoomInfoPO对象，并返回是否修改成功的布尔值 |
| hotelData.setCheckInInfo | 语法 | public boolean setCheckInInfo (CheckInInfo checkInInfo) |
| 前置条件 | 已知保存有入住的信息的对象，该订单已被执行，需要修改数据库中的入住信息 |
| 后置条件 | 系统修改数据库中的入住信息，并返回一个修改是否成功的布尔值 |
| hotelData.setCheckOutInfo | 语法 | public boolean setCheckOutInfo (CheckOutInfo checkOutInfo) |
| 前置条件 | 已知保存有退房信息的对象，该客户已办理退房，需要更新数据库里的退房信息 |
| 后置条件 | 系统更新数据库里的退房信息，并返回更新是否成功的布尔值 |
| hotelData.getRemainRoomInfo | 语法 | public List<RoomInfoVO> getRemainRoomInfo(String hotelID) |
| 前置条件 | 已知酒店的编号，逻辑层请求当前酒店剩余客房的信息 |
| 后置条件 | 系统返回一个保存剩余客房信息的RemainRoomInfoPO对象 |
| hotelData.setRemainRoomInfo | 语法 | public boolean setRemainRoomInfo  (List<RoomInfoVO> list) |
| 前置条件 | 已知保存有已被修改剩余客房信息的RemainRoomInfo的对象以及当前酒店的编号，逻辑层请求修改此酒店的剩余客房信息 |
| 后置条件 | 系统更新数据库里此酒店的剩余客房信息，并返回一个是否修改成功的布尔值 |
| hotelData.add(HotelInfoPO hotelInfoPO) | 语法 | public boolean add (HotelInfoPO hotelInfoPO) |
| 前置条件 | 已知需要被添加酒店的HotelInfoPO对象 |
| 后置条件 | 系统向数据库里添加此HotelInfoPO对象，并返回一个添加是否成功的布尔值 |

### promotionData模块

1)模块概述

promotionData模块负责保存策略信息，进行增，删，改，查的操作。

2)模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| promotionData.getPromotions | 语法 | public List<PromotionPO> getPromotions(String userID) |
| 前置条件 | 已知用户编号，逻辑层请求获得该用户（酒店工作人员、网站营销人员）所管理的促销策略 |
| 后置条件 | 返回该用户所管理的促销策略 |
| promotionData.setPromotions | 语法 | public boolean setPromotions(String userID,List<PromotionPO> list) |
| 前置条件 | 已知用户编号和被修改的促销策略列表，逻辑层请求更新数据库中的策略列表 |
| 后置条件 | 系统更新数据库中的策略列表，并返回一个是否修改成功的布尔值 |

### marketData模块

1)模块概述

marketData模块负责保存会员等级制度信息，进行改，查的操作。

2)模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| marketData.getMemberFormulation | 语法 | public List<MarketVO> getMemberFormulation() |
| 前置条件 | 无 |
| 后置条件 | 返回现有的会员等级制度信息 |
| marketData.setMemberFormulation | 语法 | public MessageResult setMemberFormulation (List<MarketVO> marketVOList) |
| 前置条件 | 网站营销人员输入会员等级制度信息 |
| 后置条件 | 系统记录此会员等级制度信息 |

### guestData模块

1)模块概述

guestData模块负责保存客户信息，进行增，改，查的操作。

2)模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| guestData.modifyMember | 语法 | public ResultMessage modifyMember(MemberPO memberPO) |
| 前置条件 | 无 |
| 后置条件 | 根据提供的memberPO修改单一持久化客户对象的信用值 |
| guestData.getSingle | 语法 | public ResultMessage getSingle (String guestID) |
| 前置条件 | guestID.Length=客户的ID长度10 |
| 后置条件 | 系统返回与该ID对应的客户信息 |
| guestData.add | 语法 | public ResultMessage add(GuestPO newGuestPO) |
| 前置条件 | 输入的guestID不存在，password和所有客户信息符合规范 |
| 后置条件 | 系统新增该客户并持久化保存该客户的数据 |
| guestData.modify | 语法 | public ResultMessage modify  (GuestPO guestPO) |
| 前置条件 | 客户信息符合规则且齐全 |
| 后置条件 | 系统修改该用户信息并持久化保存该用户数据 |
| guestData.init | 语法 | public ResultMessage init() |
| 前置条件 | 无 |
| 后置条件 | 清空所有客户持久化对象 |

### hotelWorkerData模块

1)模块概述

hotelWorkerData模块负责保存酒店工作人员信息，进行增，改，查的操作。

2)模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| hotelWorkerData.getSingle | 语法 | public ResultMessage getSingle (String hotelWorkerID) |
| 前置条件 | hotelWorkerID.Length=酒店工作人员的ID长度8 |
| 后置条件 | 系统返回与该ID对应的酒店工作人员信息 |
| hotelWorkerData.add | 语法 | public ResultMessage add(HotelWorkerPO newHotelWorkerPO) |
| 前置条件 | 输入的hotelWorkerID不存在，password和所有酒店工作人员信息符合规范 |
| 后置条件 | 系统新增该酒店工作人员并持久化保存该酒店工作人员的数据 |
| hotelWorkerData.modify | 语法 | public ResultMessage modify  (HotelWorkerPO hotelWorkerPO) |
| 前置条件 | 酒店工作人员信息符合规则且齐全 |
| 后置条件 | 系统修改该酒店工作人员信息并持久化保存该酒店工作人员数据 |
| hotelWorkerData.init | 语法 | public ResultMessage init() |
| 前置条件 | 无 |
| 后置条件 | 清空所有酒店工作人员持久化对象 |

### webMarketerData模块

1)模块概述

webMarketerData模块负责保存酒店工作人员信息，进行改，查的操作。

2)模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| webMarketerData.getSingle | 语法 | public ResultMessage getSingle (String webMarketID) |
| 前置条件 | webMarketID.Length=网站营销人员的ID长度6 |
| 后置条件 | 系统返回与该ID对应的网站营销人员信息 |
| webMarketerData.add | 语法 | public ResultMessage add(WebMarketerPO newWebMarketerPO) |
| 前置条件 | 输入的webMarketerID不存在，password和所有网站营销人员信息符合规范 |
| 后置条件 | 系统新增该网站营销人员并持久化保存该网站营销人员的数据 |
| webMarketerData.modify | 语法 | public ResultMessage modify  (WebMarketerPO webMarketerPO) |
| 前置条件 | 网站营销人员信息符合规则且齐全 |
| 后置条件 | 系统修改该网站营销人员信息并持久化保存该网站营销人员数据 |
| webMarketerData.init | 语法 | public ResultMessage init() |
| 前置条件 | 无 |
| 后置条件 | 清空所有网站营销人员持久化对象 |

### webManagerData模块

1)模块概述

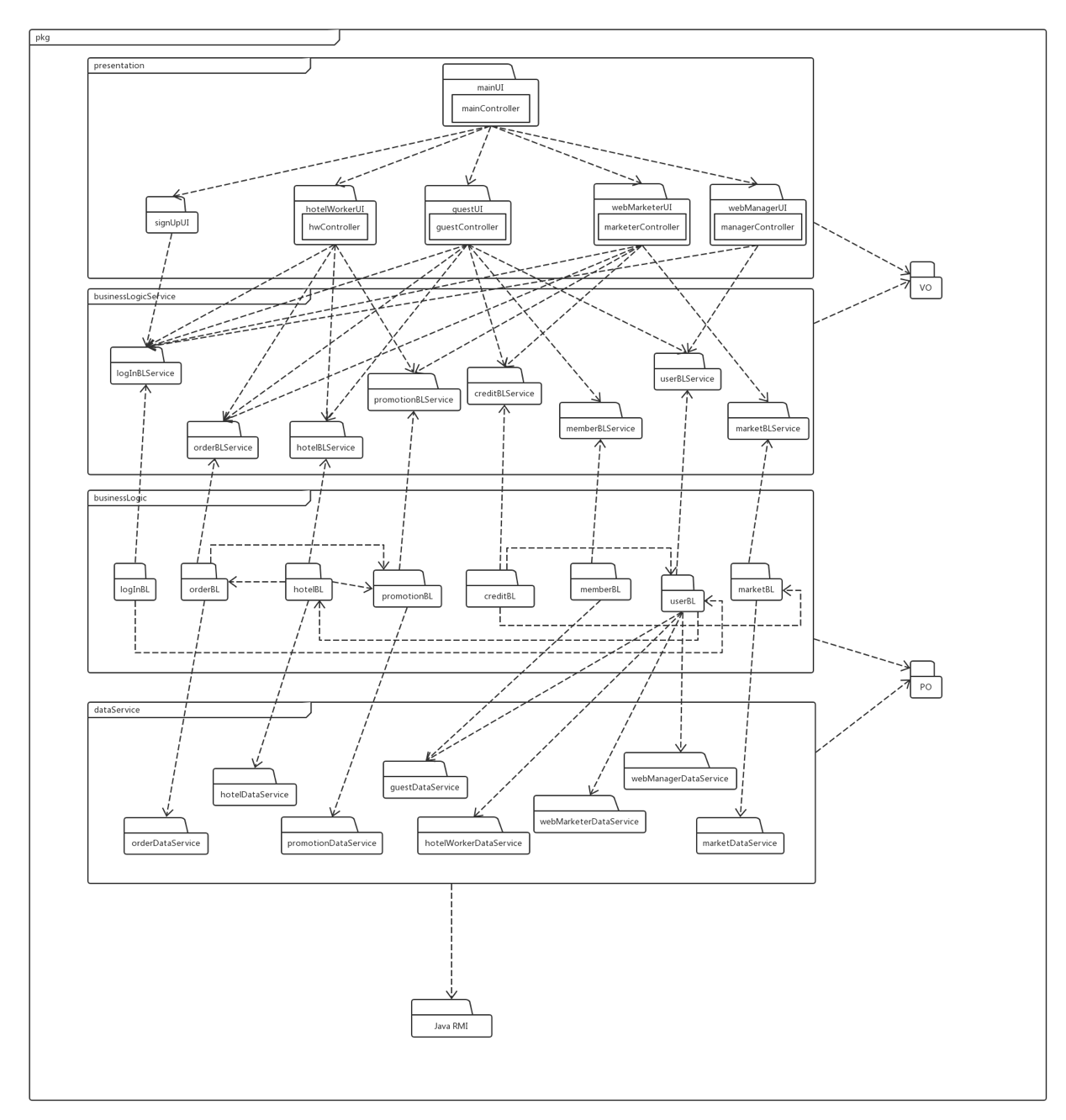
webManagerData模块负责保存酒店工作人员信息，进行改，查的操作。

2)模块内部接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| webManagerData.getSingle | 语法 | public ResultMessage getSingle (String webManagerID) |
| 前置条件 | webManagerID.Length=网站管理人员的ID长度4 |
| 后置条件 | 系统返回与该ID对应的网站管理人员信息 |
| webManagerData.add | 语法 | public ResultMessage add(WebManagerPO newWebManagerPO) |
| 前置条件 | 输入的webManagerID不存在，password和所有网站管理人员信息符合规范 |
| 后置条件 | 系统新增该网站管理人员并持久化保存该网站管理人员的数据 |
| webManagerData.modify | 语法 | public ResultMessage modify  (WebManagerPO webManagerPO) |
| 前置条件 | 网站管理人员信息符合规则且齐全 |
| 后置条件 | 系统修改该网站管理人员信息并持久化保存该网站管理人员数据 |

# 依赖视角

下图为互联网酒店预订系统中客户端各包之间的依赖关系。



下图为互联网酒店预订系统中服务器端各包之间的依赖关系。

