目录

[更新历史 2](#_Toc465547719)

[一、 引言 3](#_Toc465547720)

[(一) 编制目的 3](#_Toc465547721)

[(二) 词汇表 3](#_Toc465547722)

[(三) 参考资料 4](#_Toc465547723)

[二、 产品概述 5](#_Toc465547724)

[三、 体系结构设计概述 6](#_Toc465547725)

[四、 结构视角 7](#_Toc465547726)

[(一) 展示层的分解 7](#_Toc465547727)

[1. UI层设计概要 7](#_Toc465547728)

[2. UI层内部包的职责与划分 8](#_Toc465547729)

[(二) 业务逻辑层的分解 8](#_Toc465547730)

[1. logIn模块 8](#_Toc465547731)

[2. order模块 12](#_Toc465547732)

[3. hotel模块 19](#_Toc465547733)

[4. promotion模块 29](#_Toc465547734)

[5. credit模块 32](#_Toc465547735)

[6. member模块 36](#_Toc465547736)

[7. user模块 41](#_Toc465547737)

[8. market模块 52](#_Toc465547738)

[(三) 数据层的分解 55](#_Toc465547739)

[1. 模块 56](#_Toc465547740)

[2. 模块 56](#_Toc465547741)

[五、 依赖视角 56](#_Toc465547742)

# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **修改日期** | **修改原因** | **版本号** |
| 全体人员 | 2016/10/26 | 最初草稿 |  |
| 全体人员 | 2016/10/28 | 第一次评审后修改 |  |
| 冯俊杰 | 2016/10/4 | 整合、排版 | V1.0 |

# 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 | 系统返回是否成功添加客户；是则进入系统的客户主界面，否则提示添加失败 |
| 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 | 系统返回是否成功添加客户；是则进入系统的客户主界面，否则提示添加失败 |
| 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ID对应的user信息 | |
| User.add(UserPO newUserPO) | 新增该用户并持久化保存该用户的数据 | |
| User.getLogInfo(String userID, UserType userType) | 返回与该ID对应的user密码 | |

1. 业务逻辑层的动态模型
2. 业务逻辑层的设计原理

### order模块

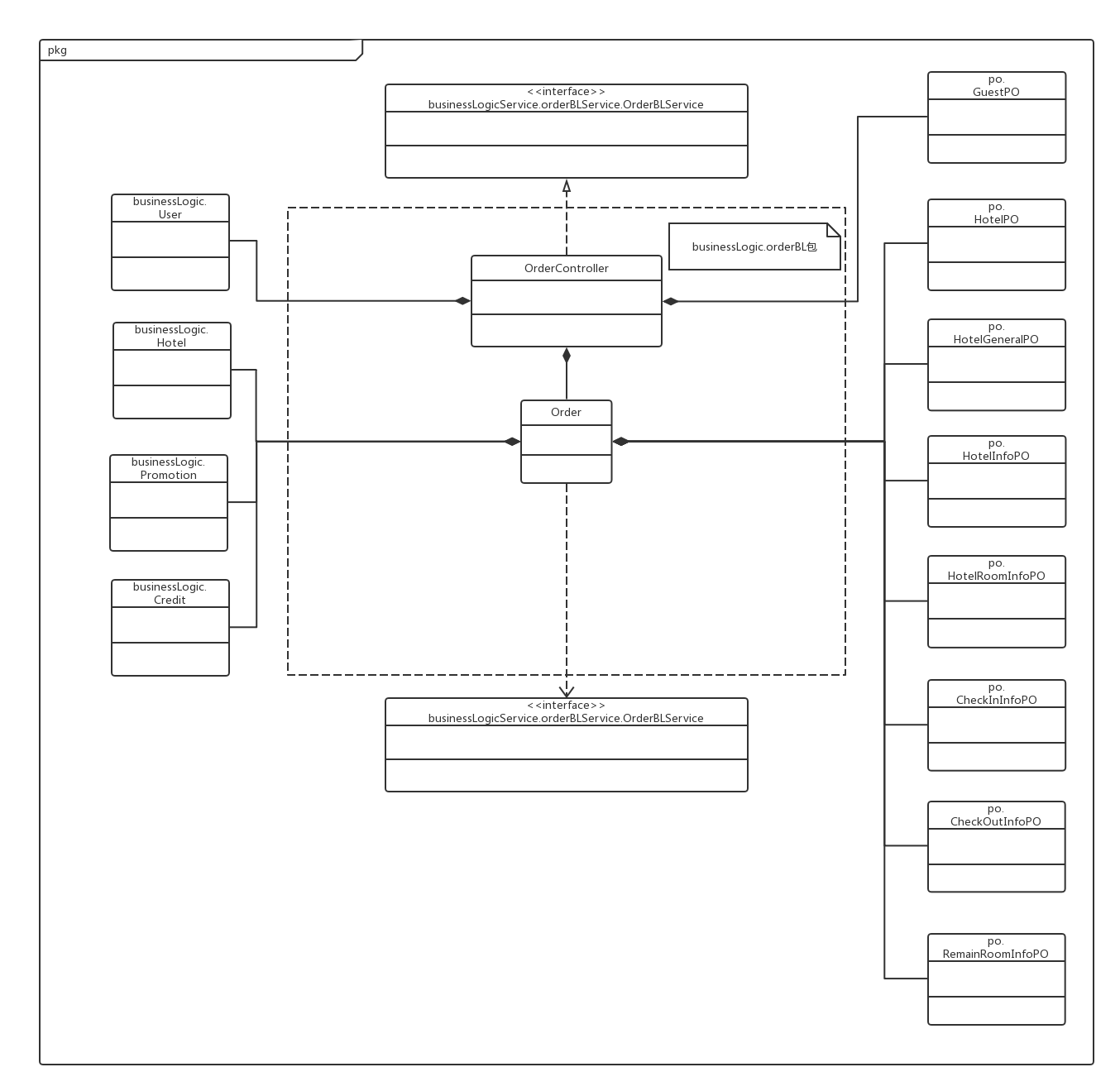
1. 模块概述

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

|  |  |  |
| --- | --- | --- |
| 1. 提供的服务（供接口） | | |
| OrderController.createOrder | 语法 | Public ResultMessage createOrder(orderVO OrderVO) |
| 前置条件 | 客户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 | 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 | 订单状态为异常 |
| 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.undoNormalOrder | 语法 | Public ResultMessage undoNormalOrder(String orderID) |
| 前置条件 | 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.getOrderDetail | 语法 | Public OrderVO getOrderDetail(String orderID) |
| 前置条件 | 知道订单号，需要知道订单内容 |
| 后置条件 | 系统返回该订单号的订单 |
| OrderController.getAllGuestOrderGeneral | 语法 | Public List<OrderGeneralVO> getAllGuestOrderGeneral (String guestID) |
| 前置条件 | 已经登录，需要知道的订单号 |
| 后置条件 | 系统返回查找到的订单结果 |
| OrderController.getAllHotelOrderGeneral | 语法 | Public List<OrderGeneralVO> getAllHotelOrderGeneral (String hotelID) |
| 前置条件 | 酒店工作人员已经登录 |
| 后置条件 | 系统返回该酒店所有订单 |
| OrderController.getBookedHotel | 语法 | Public List<String> getBookedHotel (String guestID) |
| 前置条件 | 客户需要知道所有住过酒店的信息 |
| 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.getBookedHotel(String guestID) | 查看客户预定过的酒店 | |
| User.getSingle(String userID,UserType userType) | 查看客户详情 | |

1. Domain类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Order.createOrder | 语法 | Public ResultMessage createOrder(orderPO OrderPO) |
| 前置条件 | 客户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 | 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 | 订单状态为异常 |
| 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.undoNormalOrder | 语法 | Public ResultMessage undoNormalOrder(String orderID) |
| 前置条件 | 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.getOrderDetail | 语法 | Public OrderVO getOrderDetail(String orderID) |
| 前置条件 | 知道订单号，需要知道订单内容 |
| 后置条件 | 系统返回该订单号的订单 |
| Order.getAllGuestOrderGeneral | 语法 | Public List<OrderGeneralVO> getAllGuestOrderGeneral (String guestID) |
| 前置条件 | 已经登录，需要知道的订单号 |
| 后置条件 | 系统返回查找到的订单结果 |
| Order.getAllHotelOrderGeneral | 语法 | Public List<OrderGeneralVO> getAllHotelOrderGeneral (String hotelID) |
| 前置条件 | 酒店工作人员已经登录 |
| 后置条件 | 系统返回该酒店所有订单 |
| Order.getBookedHotel | 语法 | Public List<String> getBookedHotel (String guestID) |
| 前置条件 | 客户需要知道所有住过酒店的信息 |
| 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(orderVO OrderVO) | 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.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
2. 业务逻辑层的设计原理

### 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。



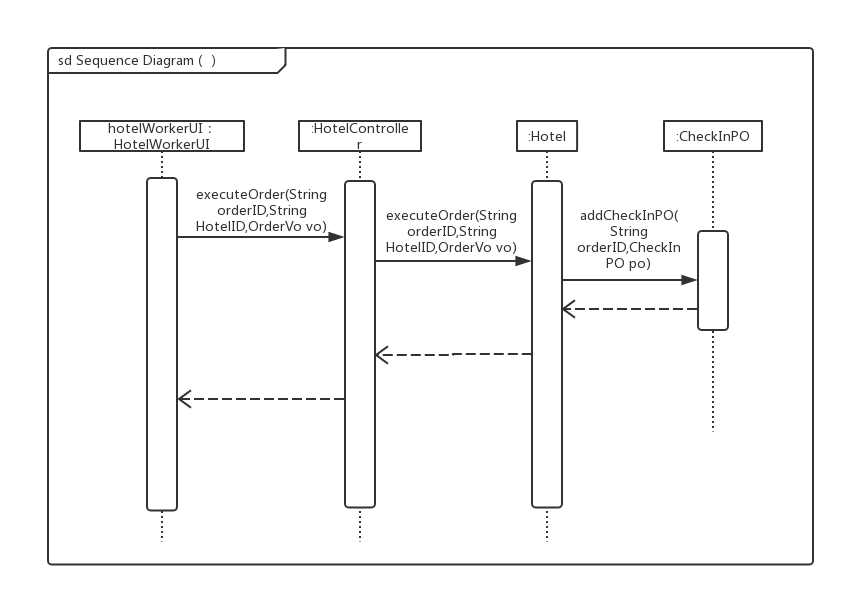
1. 模块内部类的接口规范
2. Controller类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| HotelController.getHotelInfo | 语法 | Public HotelVO getHotelInfo(String userID) |
| 前置条件 | 已知酒店工作人员的编号，需要维护酒店基本信息 |
| 后置条件 | 系统返回该酒店工作人员对应的酒店基本信息 |
| HotelController.updateHotelInfo | 语法 | Public ResultMessage updateHotelInfo (HotelVO hotelVO) |
| 前置条件 | 酒店工作人员确认修改酒店基本信息，已知修改后的酒店基本面信息 |
| 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保存是否成功 |
| HotelController.getHotelRoomInfo | 语法 | Public RoomInfoVO getHotelRoomInfo (String userID) |
| 前置条件 | 已知酒店工作人员的编号，需要维护客房信息 |
| 后置条件 | 系统返回该酒店工作人员管理的客房信息 |
| HotelController.updateHotelRoomInfo | 语法 | Public ResultMessage updateHotelRoomInfo (RoomInfoVO RoomInfoVO) |
| 前置条件 | 酒店工作人员确认修改客房信息，已知修改后的客房信息 |
| 后置条件 | 系统更新并持久化保存客房信息，并提示酒店工作人员保存是否成功 |
| HotelController.updateCheckIn | 语法 | Public ResultMessage updateCheckIn (List<String> roomNum,Calendar inTime,Calendar predictedOutTime) |
| 前置条件 | 酒店工作人员已执行订单，并已知入住房间号的列表，入住时间以及预计离开时间 |
| 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是否更新成功 |
| HotelController.updateCheckOut | 语法 | Public ResultMessage updateCheckOut (Calendar outTime) |
| 前置条件 | 酒店工作人员对订单办理退房，已知退房时间 |
| 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是否更新成功 |
| HotelController.getRemainRoomInfo | 语法 | Public RemainRoomInfoVO getRemainRoomInfo (String userID) |
| 前置条件 | 已知酒店工作人员的编号，需要对线下客户的入住和退房进行处理或需要查看剩余客房信息 |
| 后置条件 | 系统返回该酒店工作人员所对应酒店的剩余客房信息 |
| 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.getRoomInfo | 得到酒店客房信息 | |
| Hotel.getRemainRoomInfo | 得到酒店的剩余客房信息 | |
| Hotel.updateHotelInfo | 更新酒店基本信息 | |
| Hotel.updateRoomInfo | 更新酒店客房信息 | |
| ExecutedOrderInfo.getExecutedOrder | 得到已执行的订单 | |
| ExecutedOrderInfo.checkOut | 已执行的订单退房 | |
| UnExecutedOrderInfo.getUnexecutedOrder | 得到未执行的订单 | |
| UnExecutedOrderInfo.execute | 执行未执行的订单 | |
| AbnormalOrderInfo.getAbnornalOrder | 得到异常的订单 | |
| UnExecutedOrderInfo.execute | 执行异常订单 | |

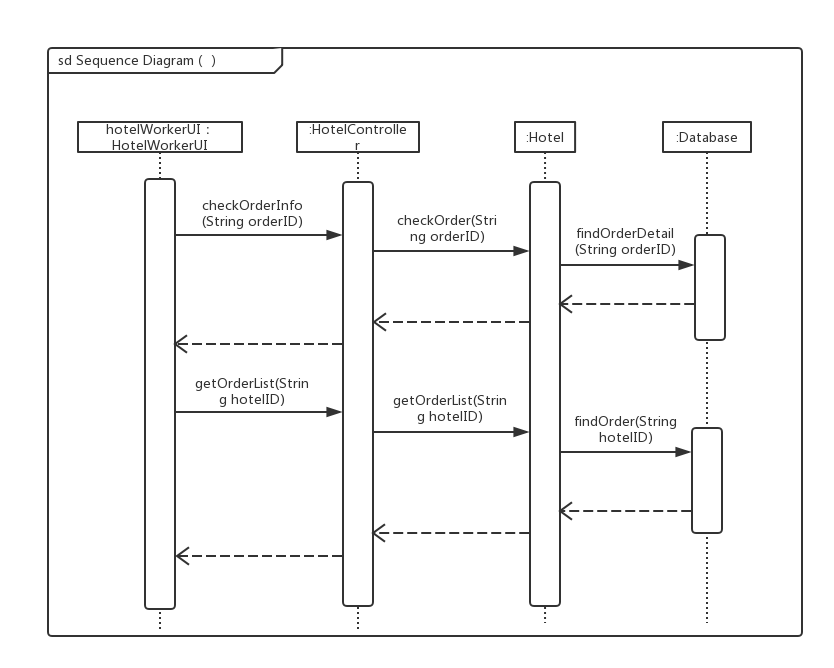
1. Domain类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Hotel.getHotelInfo | | 得到酒店基本信息 |
| Hotel.getRoomInfo | | 得到酒店客房信息 |
| Hotel.getRemainRoomInfo | | 得到酒店的剩余客房信息 |
| Hotel.updateHotelInfo | | 更新酒店基本信息 |
| Hotel.updateRoomInfo | | 更新酒店客房信息 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| HotelDataService.getHotelList(Address VO addressVO) | 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 | |

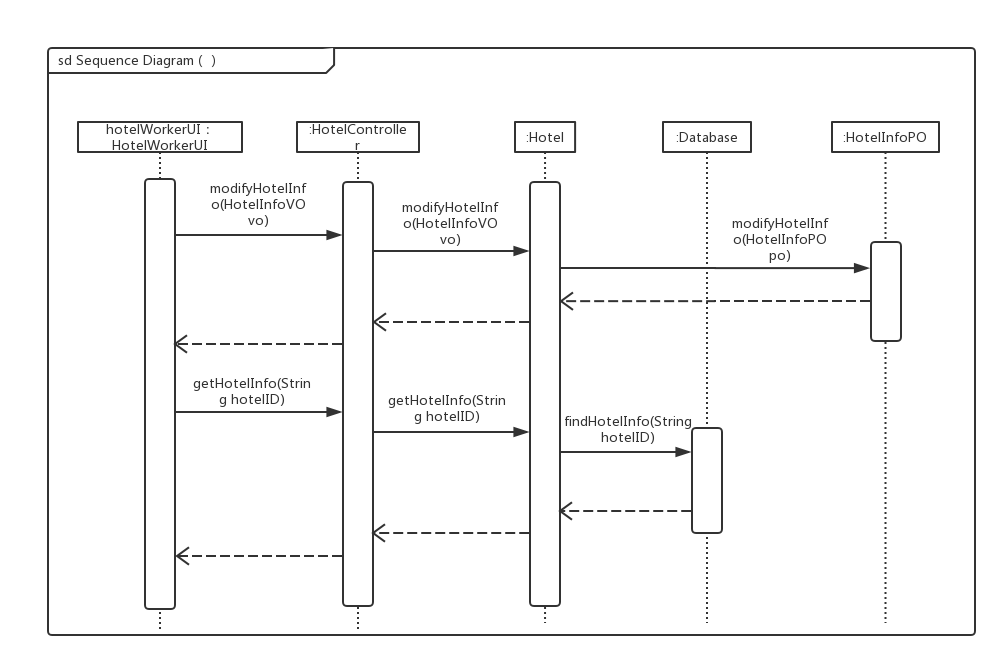
1. 业务逻辑层的动态模型



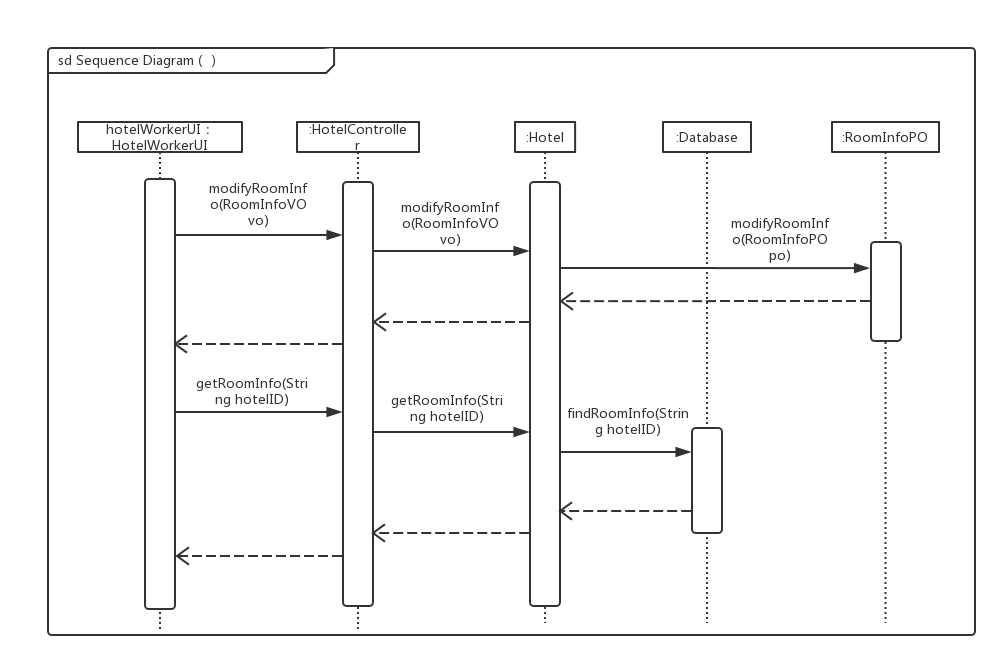
订单执行及更新入住信息顺序图



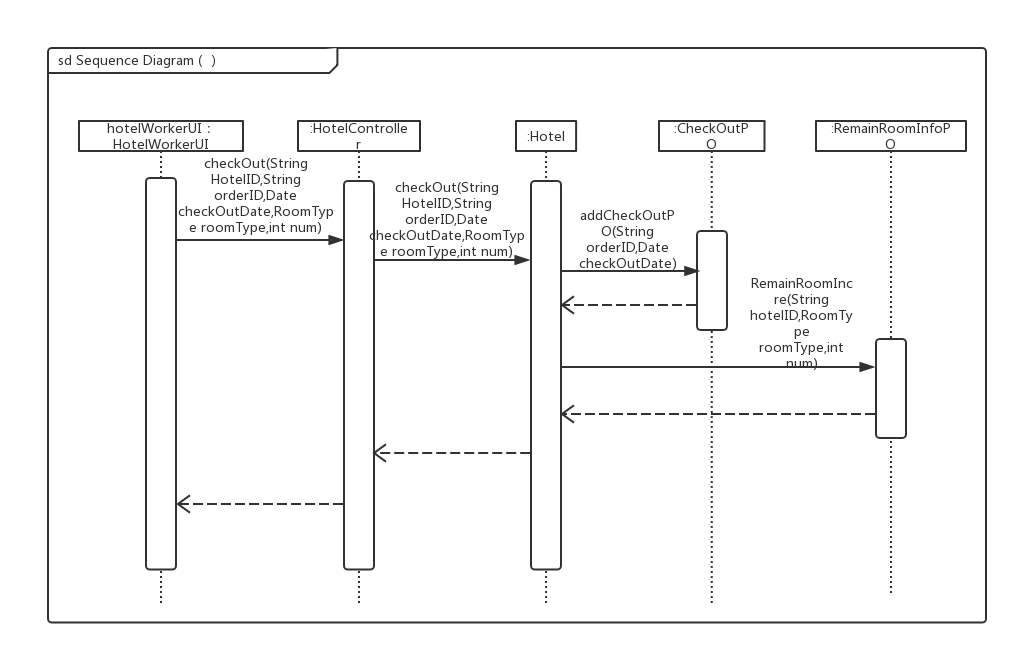
酒店订单浏览顺序图



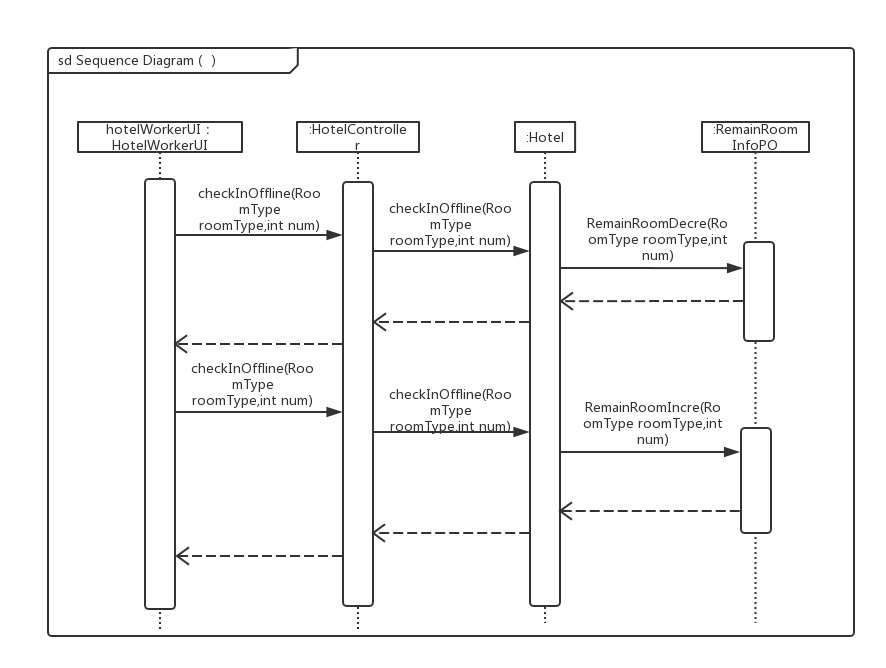
酒店基本信息维护顺序图



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

### 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。



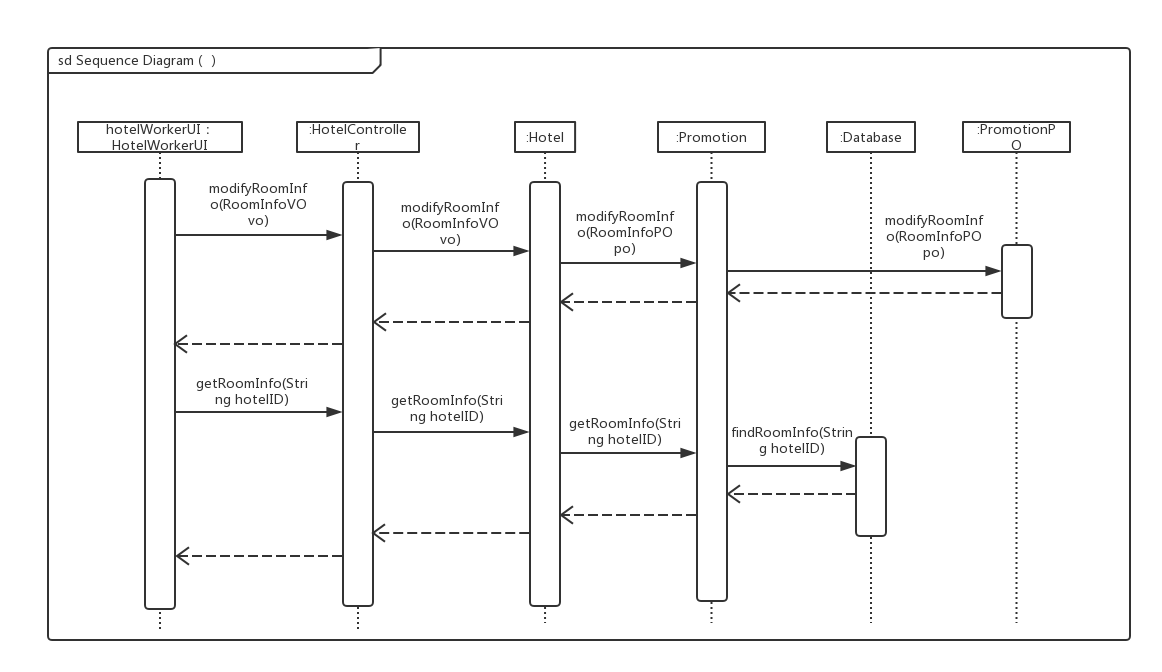
1. 模块内部类的接口规范
2. Controller类的接口规范

|  |  |  |
| --- | --- | --- |
| 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

### 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

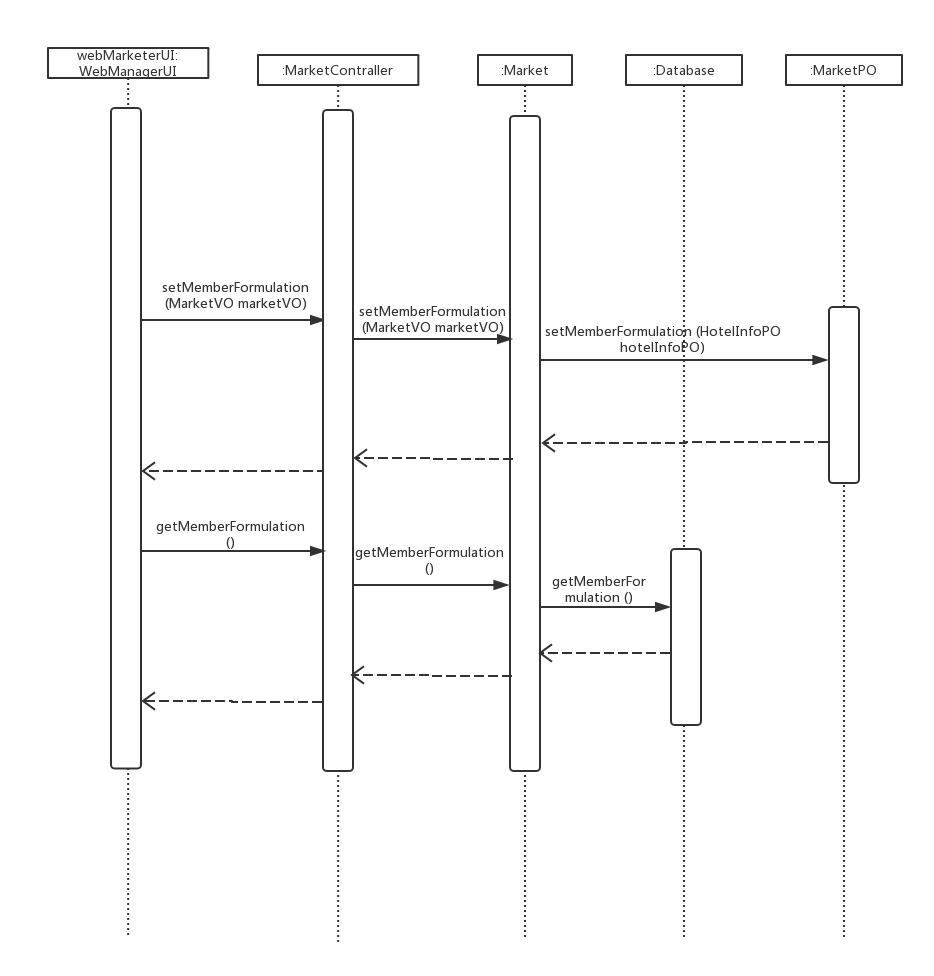
(1) Controller类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| creditController.charge | 语法 | Public ResultMessage charge(int chargeNum) |
| 前置条件 | 网站营销人员为客户信用充值 |
| 后置条件 | 系统为客户信用值充值并返回是否成功 |
| creditController.getBasicInfo | 语法 | Public CreditVO getBasicInfo (String ID) |
| 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 | 系统返回与该ID对应的包含信用值及信用记录的客户个人基本信息 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| credit.charge | 充值信用值 | |
| credit.getBasicInfo | 返回信用值信息 | |

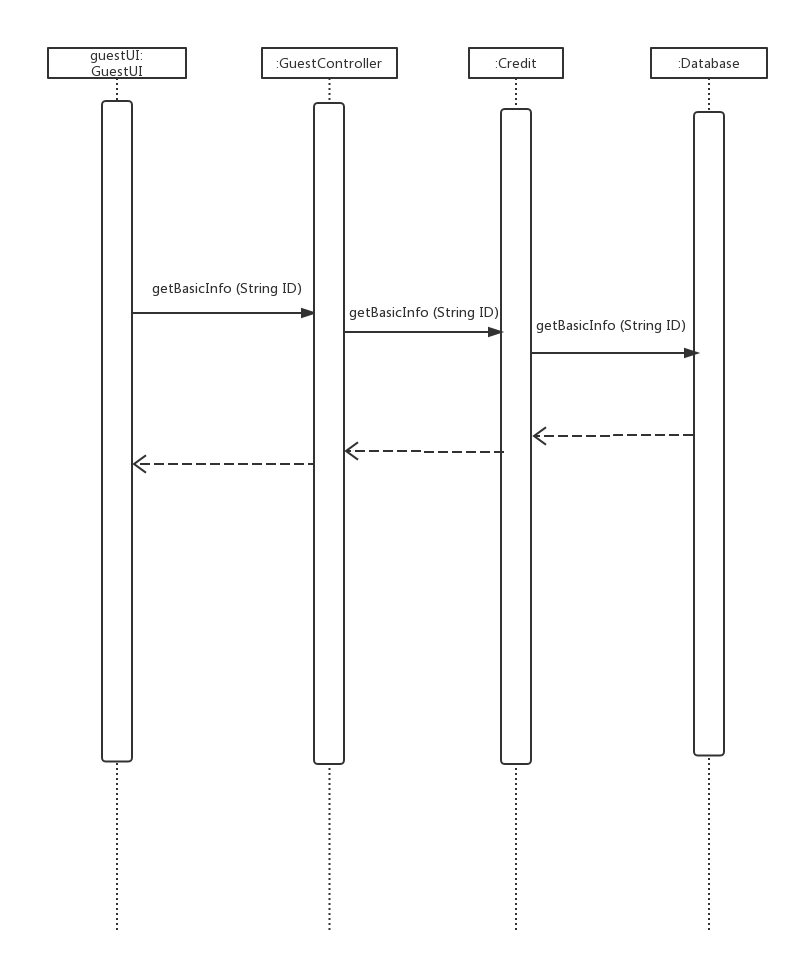
(2)Domain类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| credit.charge | 语法 | Public ResultMessage charge(int chargeNum) |
| 前置条件 | 网站营销人员为客户信用充值 |
| 后置条件 | 系统为客户信用值充值并返回是否成功 |
| credit.getBasicInfo | 语法 | Public CreditVO getBasicInfo (String ID) |
| 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 | 系统返回与该ID对应的包含信用值及信用记录的客户个人基本信息 |
| 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ID对应的用户信息 | |
| User.modifyCredit(float creditNum) | 返回客户信用值是否修改成功 | |
| market.getMemberFormulation() | 返回现有的会员等级制度信息 | |
| order.getAllGuestOrderGeneral (String guestID) | 返回与该ID对应的客户所有订单概况 | |

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

### 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。



图1—member模块设计类图

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.memberSignUp | 语法 | Public ResultMessage memberSignUp (MemberVO memberVO) |
| 前置条件 | 客户输入的会员信息符合信息规范 |
| 后置条件 | 系统返回是否成功注册会员；是则提示添加成功，否则提示注册失败原因 |
| memberController.modifyMemberInfo | 语法 | Public ResultMessage modifyMemberInfo (String guestID, MemberVO memberVO) |
| 前置条件 | 用户输入客户编号，并且输入的信息符合信息规范 |
| 后置条件 | 系统返回是否成功修改会员；是则提示修改成功，否则提示修改失败原因 |
| 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

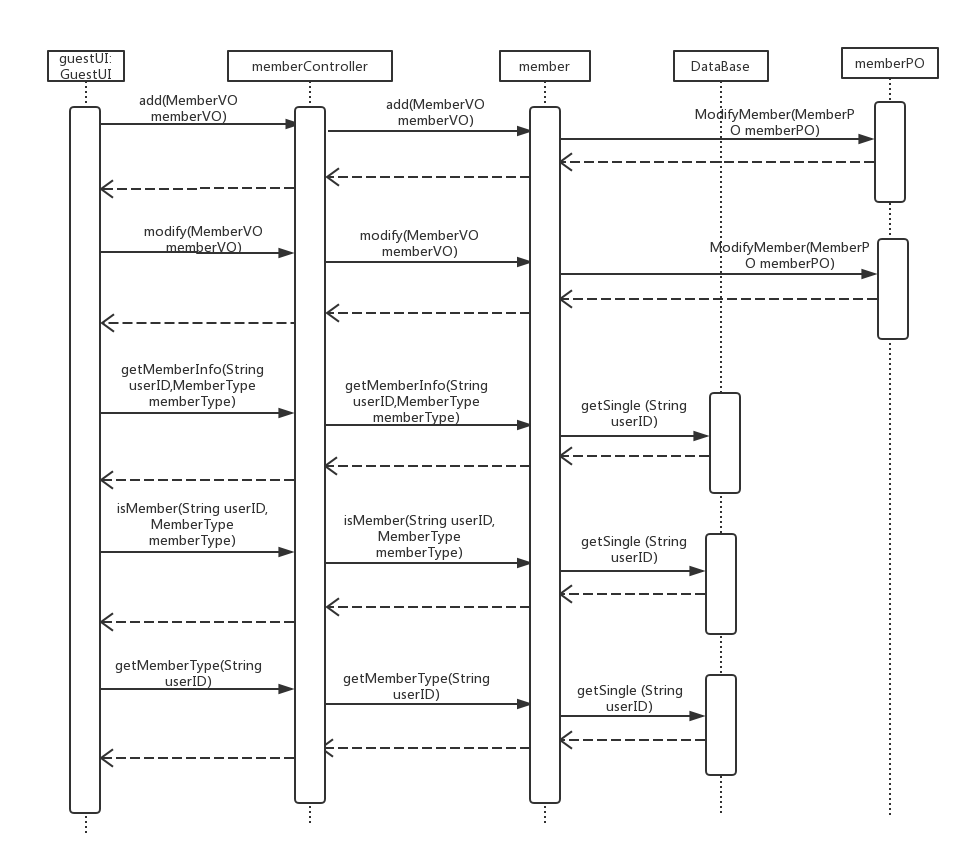


图2—member模块设计顺序图



图3—member模块设计状态图

1. 业务逻辑层的设计原理

### 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。



图1—user模块设计类图

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 (String userID,UserVO userVO) |
| 前置条件 | 用户输入用户编号， 输入的用户信息符合信息规范 |
| 后置条件 | 系统返回是否成功添加用户；是则提示添加成功，否则提示添加失败原因 |
| UserController.modify | 语法 | Public ResultMessage modify (String userID,UserVO userVO) |
| 前置条件 | 用户输入用户编号， 输入的用户信息符合信息规范 |
| 后置条件 | 系统返回是否成功修改用户信息；是则提示修改成功，否则提示修改失败原因 |
| UserController.getSingle | 语法 | Public UserVO getSingle (String userID) |
| 前置条件 | 用户输入用户编号 |
| 后置条件 | 系统返回与用户编号对应的用户信息 |
| UserController.addHotel | 语法 | Public ResultMessage addHotel(String hotalID, HotelVO hotelVO) |
| 前置条件 | 网站管理人员输入酒店编号和符合信息规范的酒店信息 |
| 后置条件 | 系统返回是否成功添加酒店信息；是则提示添加成功，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 | 网站营销人员输入客户编号和更改后的信用值 |
| 后置条件 | 系统返回是否成功修改信用值；是则提示修改成功，否则提示修改失败原因 |
| UserController.getAll | 语法 | Public List<UserVO> getAll(UserType userType) |
| 前置条件 | 用户输入用户类型 |
| 后置条件 | 系统返回该用户类型的所有用户的信息 |
| 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.getAll | 系统返回指定用户类型的所有用户的信息 | |
| UserController.modifyCredit | 系统修改指定客户的信用值 | |

(2)Domain类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| User.getAll | 语法 | Public List<UserVO> getAll(UserType userType) |
| 前置条件 | 无 |
| 后置条件 | 系统返回指定用户类型的所有用户信息 |
| User.getSingle | 语法 | Public UserVO getSingle(String userID，UserType userType) |
| 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ID对应的user信息 |
| User.add | 语法 | Public ResultMessage add(UserVO newUserVO) |
| 前置条件 | 输入的userID不存在，password和所有信息符合规范 |
| 后置条件 | 系统新增该用户并持久化保存该用户的数据 |
| User.modify | 语法 | Public ResultMessage modify(UserVO userVO) |
| 前置条件 | 用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 |
| User.addHotel | 语法 | Public ResultMessage addHotel(HotelVO newHotelVO, String hotelID) |
| 前置条件 | 输入的hotelID不存在,所有酒店信息符合规范 |
| 后置条件 | 系统新增该酒店并持久化保存该酒店的数据 |
| User.modifyCredit | 语法 | Public ResultMessage modifyCredit (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.getLoginInfo | 语法 | Public ResultMessage getLoginInfo(String userID, UserType userType) |
| 前置条件 | userID.Length=对应用户类型的ID长度 |
| 后置条件 | 系统返回与该用户ID对应的密码  如果ID不存在，则系统返回该ID对应用户不存在 |
| 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userID查找客户持久化对象 | |
| 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userID查找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userID查找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userID查找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顺序图

1：guestUI

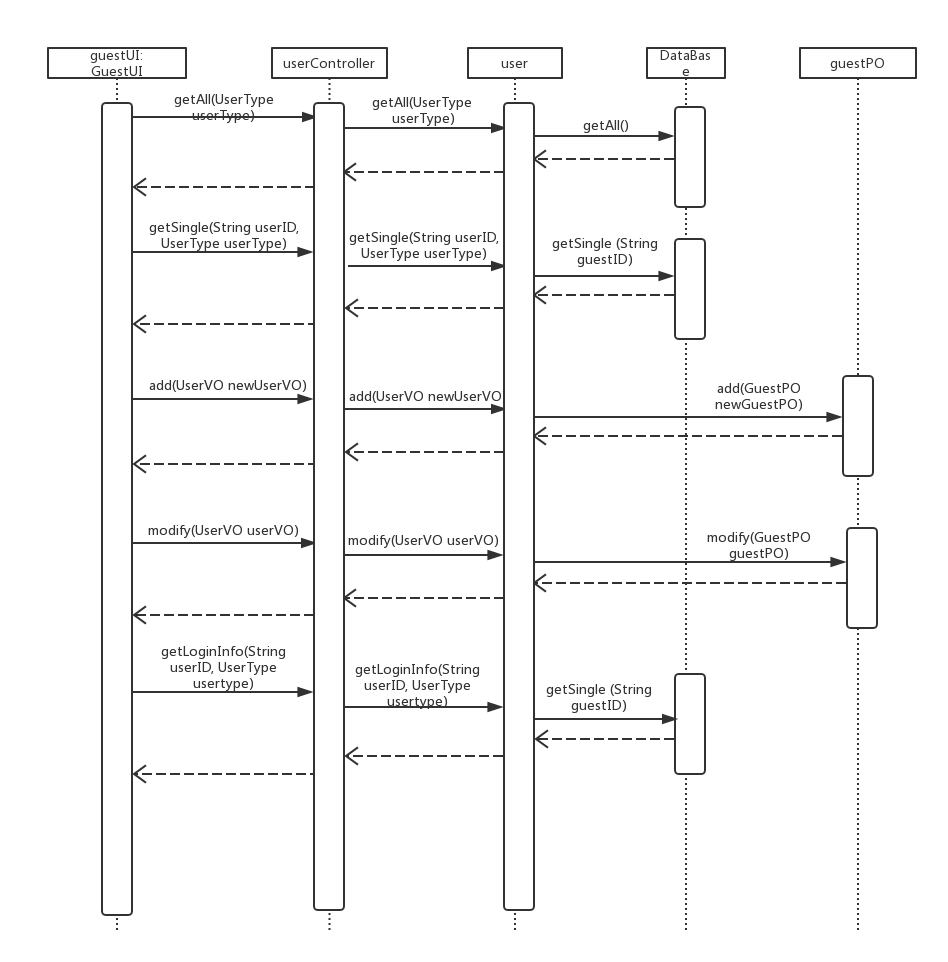


图2—user模块设计顺序图（guest）

2:webManagerUI

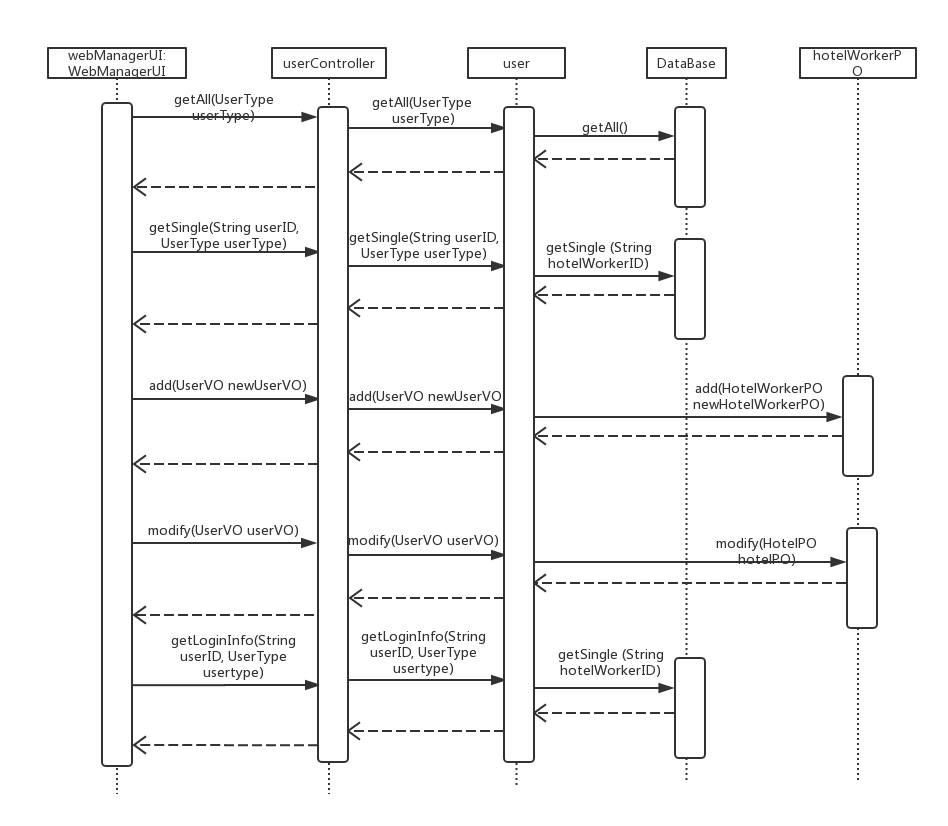


图3—user模块设计顺序图（hotelWorker）

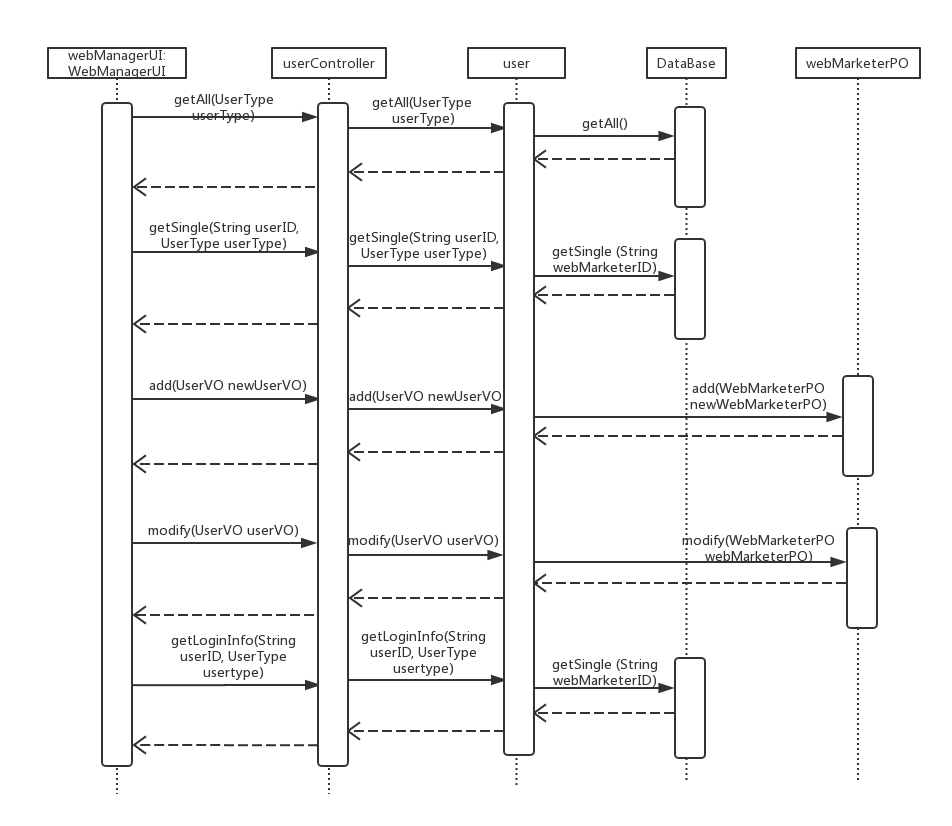


图4—user模块设计顺序图（webMarketer）

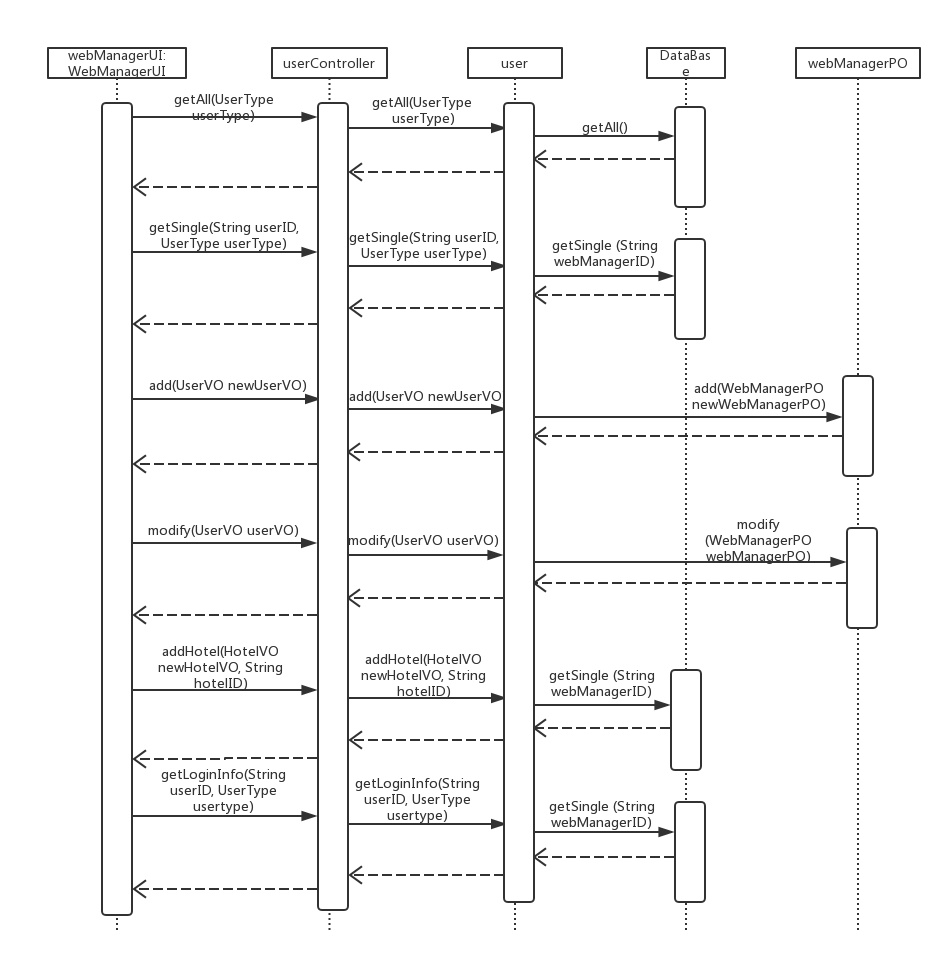


图5—user模块设计顺序图（webManager）



图6—user模块设计状态图

1. 业务逻辑层的设计原理

### 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

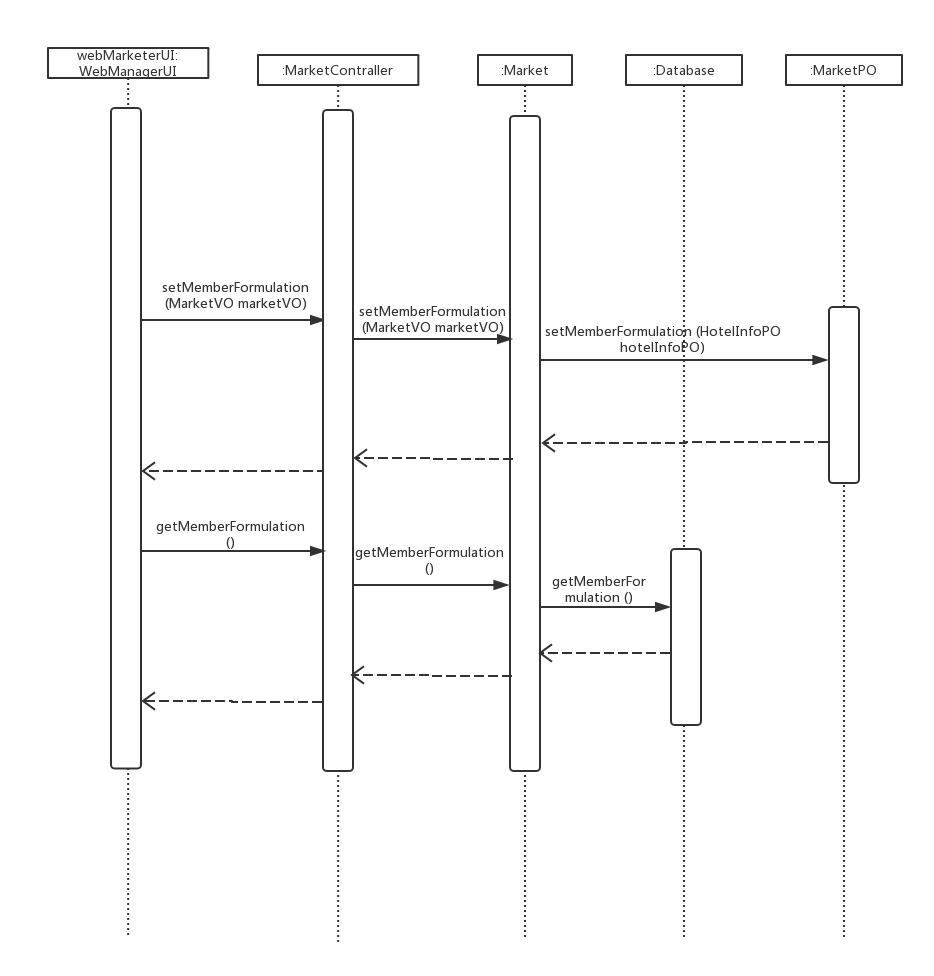
(1) Controller类的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| marketController.getMemberFormulation | 语法 | Public MarketVO getMemberFormulation () |
| 前置条件 | 无 |
| 后置条件 | 返回现有的会员等级制度信息 |
| marketController.setMemberFormulation | 语法 | Public ResultMessage setMemberFormulation (MarketVO marketVO) |
| 前置条件 | 网站营销人员输入会员等级制度 |
| 后置条件 | 系统记录此会员等级制度信息 |
| 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 MarketVO getMemberFormulation () |
| 前置条件 | 无 |
| 后置条件 | 返回现有的会员等级制度信息 |
| market.setMemberFormulation | 语法 | Public ResultMessage setMemberFormulation (MarketVO marketVO) |
| 前置条件 | 网站营销人员输入会员等级制度 |
| 后置条件 | 系统记录此会员等级制度信息 |
| 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持久化对象，系统返回是否成功记录 | |

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

## 数据层的分解

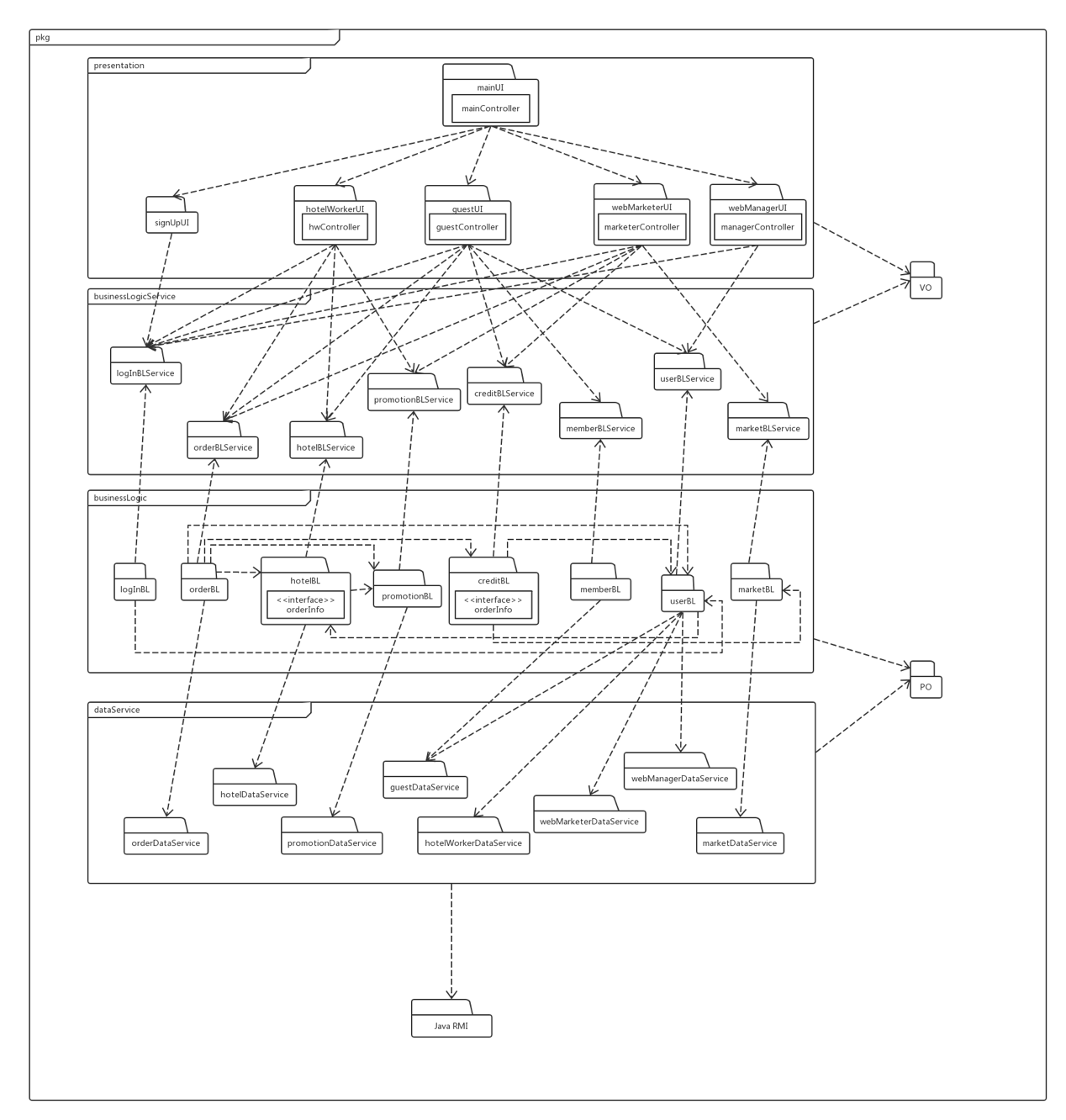
数据层模块的分解详见需求规格文档中的相关描述。

### 模块

### 模块

# 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。

