### 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

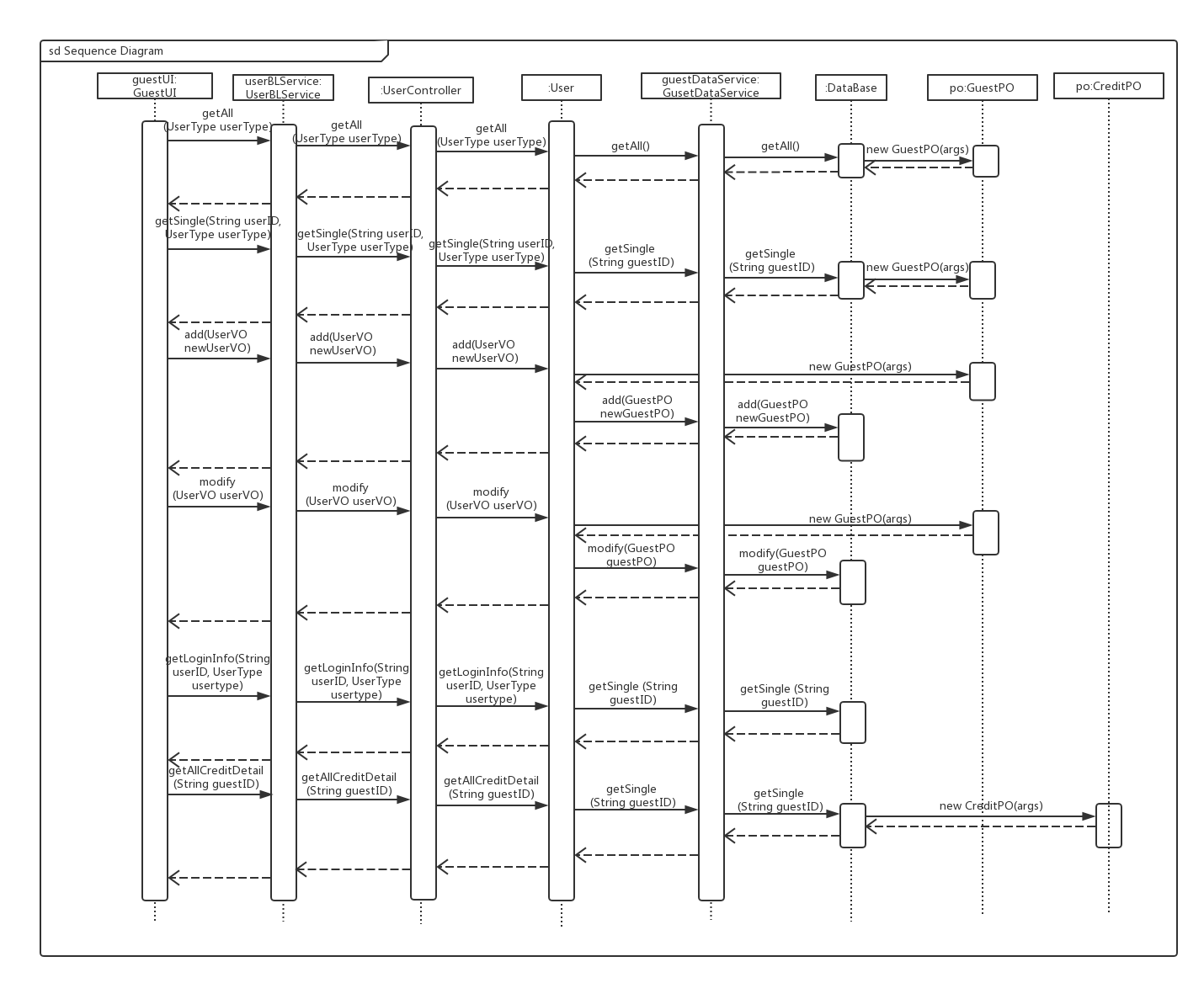
(1) Controller类的接口规范

|  |  |  |
| --- | --- | --- |
| 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 userVO) |
| 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 ) |
| 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) |
| 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 ,double 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.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** | |
| User.add | 系统添加用户信息 | |
| User.modify | 系统修改用户信息 | |
| User.getSingle | 系统返回单一用户的信息 | |
| User.addHotel | 系统添加酒店信息 | |
| User.modifyCredit | 系统修改指定客户的信用值 | |
| User.getAll | 系统返回指定用户类型的所有用户的信息 | |
| User.getLoginInfo | 系统返回指定用户相关的登录信息 | |

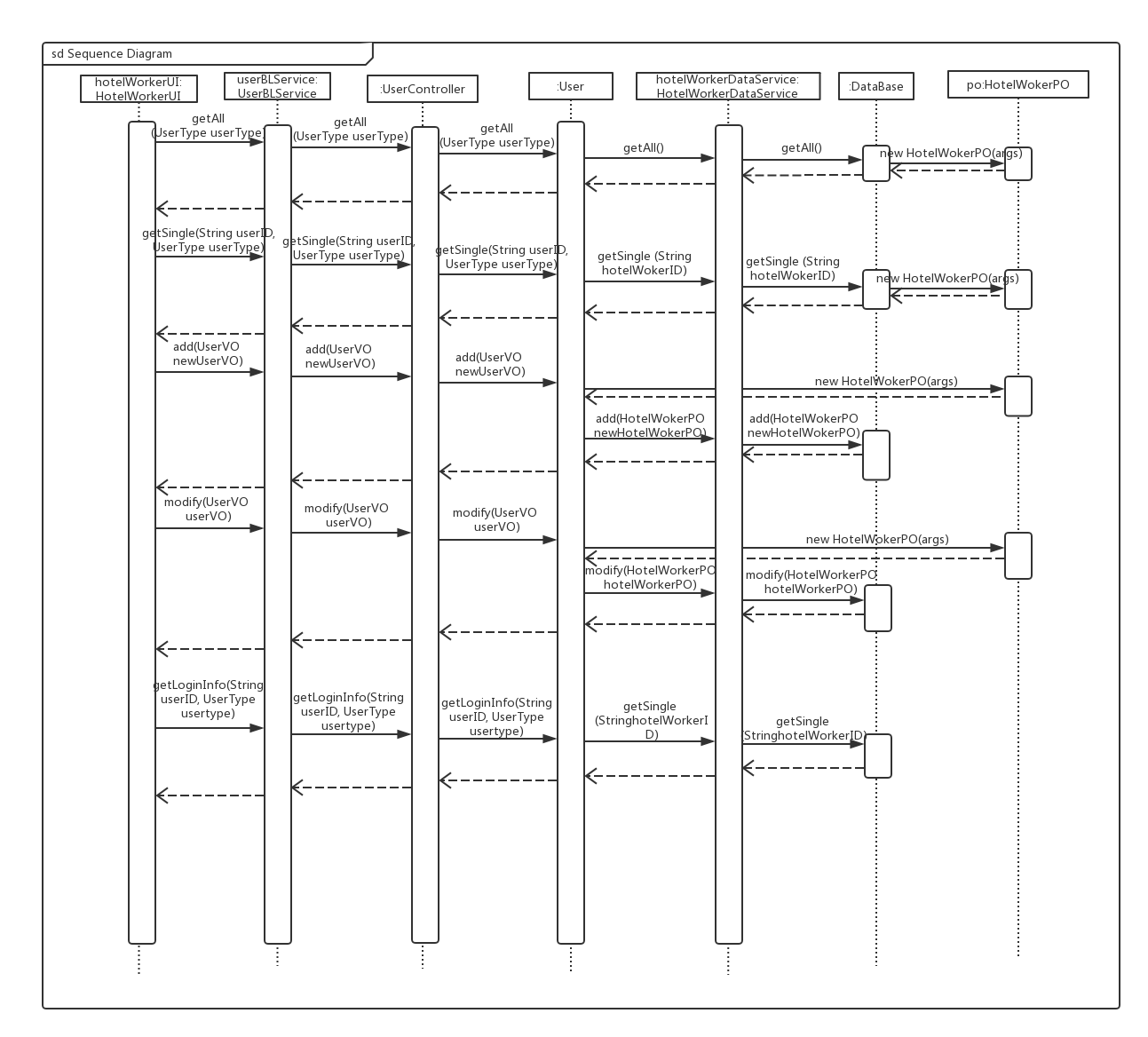
(2)Domain类的接口规范

|  |  |  |
| --- | --- | --- |
| 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 userVO) |
| 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) |
| 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) |
| 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, double 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.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.getSingleGuest(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.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.getSingleHotelWorker(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(HotelWorkerPO hotelWorkerPO) | 修改单一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.  getSingleWebMarketer(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 | |
| webManagerDataService.add(WebManagerPO newWebManagerPO) | 根据提供的newWebManagerPO增加网站管理人员持久化对象 | |
| webManagerDataService.getSingleWebManager(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 | |
| Hotel.add | 新增该酒店并持久化保存该酒店的数据 | |

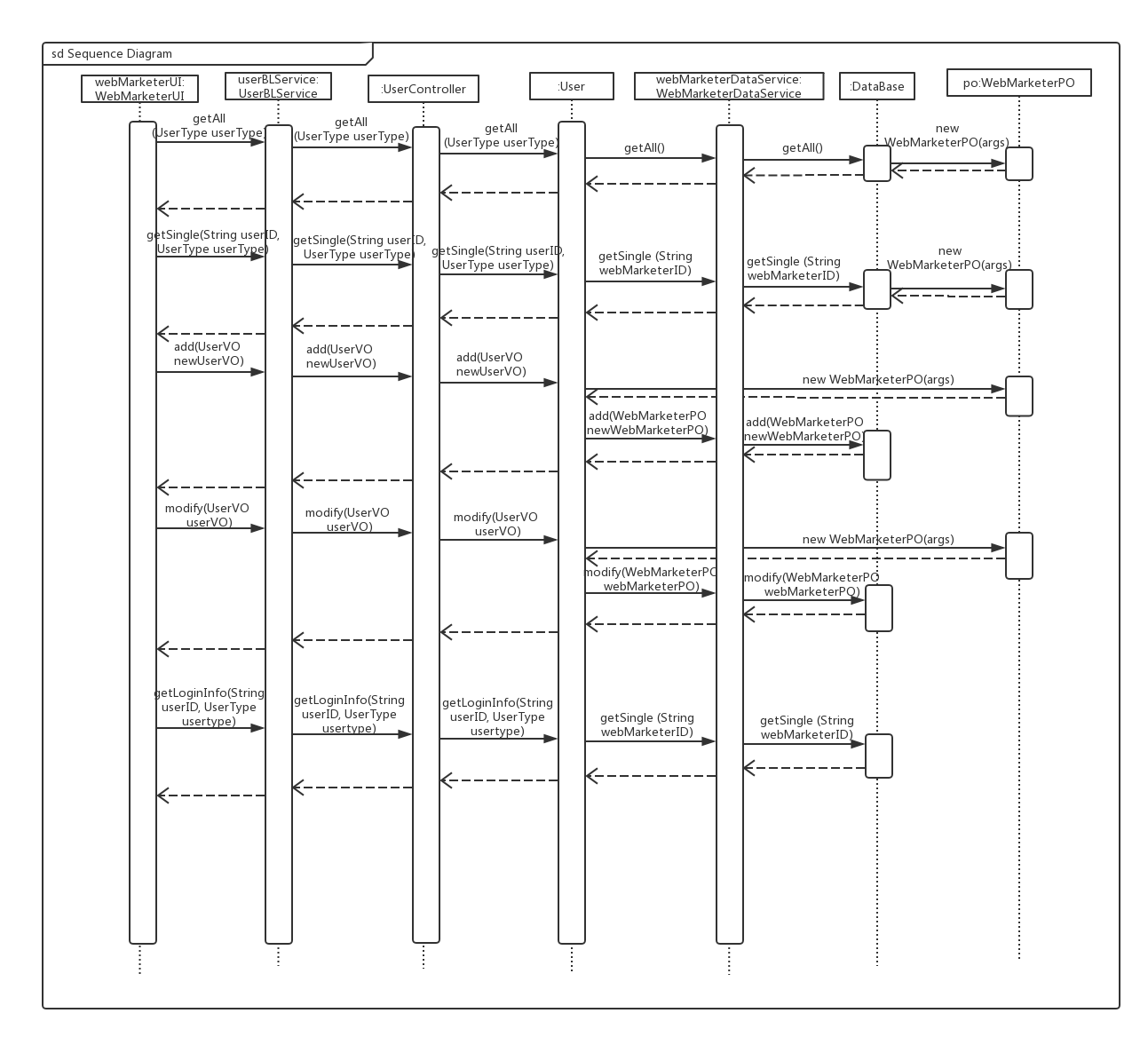
1. 业务逻辑层的动态模型



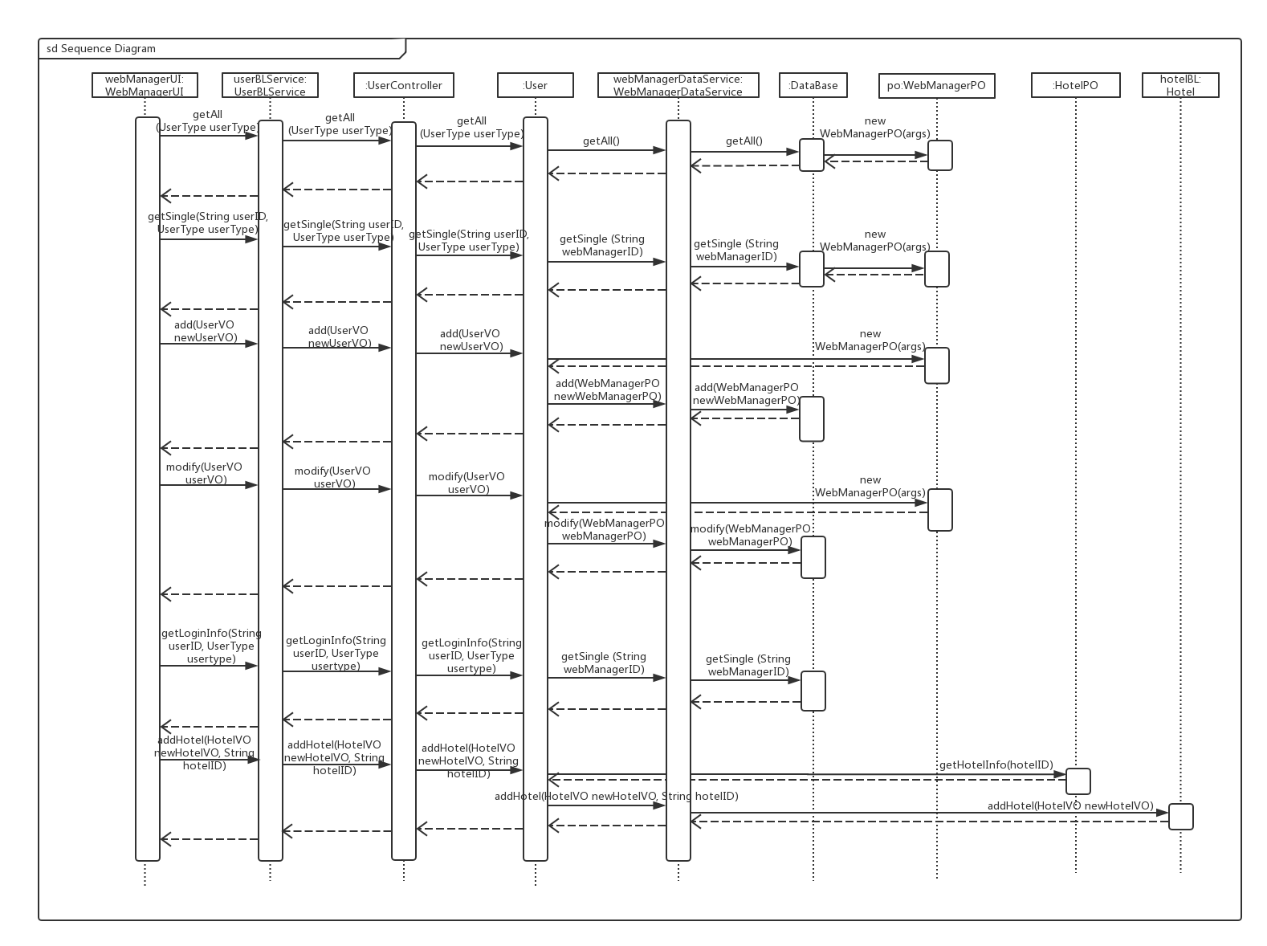
user面向客户的顺序图



user面向酒店工作人员的顺序图



user面向网站营销人员的顺序图



user面向网站管理人员的顺序图



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

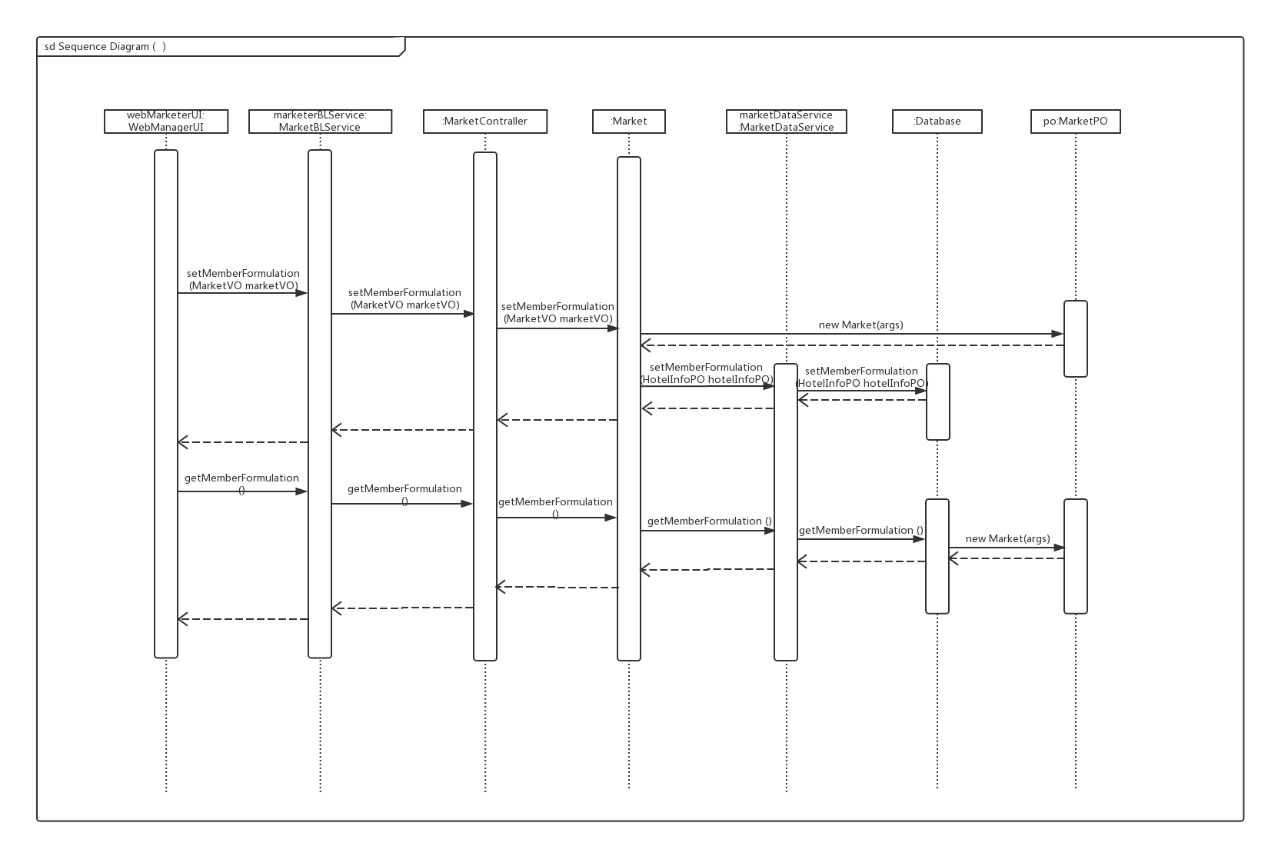
(1) Controller类的接口规范

|  |  |  |
| --- | --- | --- |
| 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 |
| marketController.getLevel | 语法 | Public int getLevel （String userID) |
| 前置条件 | userID.Length=对应用户类型的ID长度 |
| 后置条件 | 系统根据客户当前信用值计算并返回当前会员等级 |
| marketController.getLevelName | 语法 | Public String getLevel （String userID) |
| 前置条件 | userID.Length=对应用户类型的ID长度 |
| 后置条件 | 系统根据客户当前信用值计算并返回当前会员等级的名字 |
| marketController.get MemberDiscout | 语法 | Public double getMemeberDiscout（String userID) |
| 前置条件 | userID.Length=对应用户类型的ID长度 |
| 后置条件 | 系统根据客户当前信用值计算用户当前会员等级，根据等级返回该用户享受的等级折扣 |
| 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 | |
| market.getLevel | 返回用户当前的会员等级 | |
| market.getLevelName | 返回用户当前的会员等级名字 | |
| market.getMemberDiscout | 返回用户当前的可以享受的等级折扣 | |

(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 |
| market.getLevel | 语法 | Public int getLevel （String userID) |
| 前置条件 | userID.Length=对应用户类型的ID长度 |
| 后置条件 | 系统根据客户当前信用值计算并返回当前会员等级 |
| market.getLevelName | 语法 | Public String getLevel （String userID) |
| 前置条件 | userID.Length=对应用户类型的ID长度 |
| 后置条件 | 系统根据客户当前信用值计算并返回当前会员等级的名字 |
| market.getMemberDiscout | 语法 | Public double getMemeberDiscout（String userID) |
| 前置条件 | userID.Length=对应用户类型的ID长度 |
| 后置条件 | 系统根据客户当前信用值计算用户当前会员等级，根据等级返回该用户享受的等级折扣 |
| 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(List<MarketPO> marketPOList | 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。