**酒店预订管理系统EasyHotel**

**软件详细设计描述文档**

**V1.0 正式版**

**EasyHotel小组**

**王涛 王昕尧 李玉莹 雷雨佳 方雅婧**

**2016-11-13**

# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **日期** | **变更原因** | **版本号** |
| 王涛 | 2016-10-29 | 最初草稿 | V0.9 草稿 |
| 王涛 | 2016-11-06 | 修正草稿 | V0.99 初稿 |
| 王昕尧 | 2016-11-13 | 评审后的正式版 | V1.0正式版 |

目录

[更新历史 2](#_Toc466842072)

[1. 引言 4](#_Toc466842073)

[1.1 编制目的 4](#_Toc466842074)

[1.2 词汇表 4](#_Toc466842075)

[1.3 参考资料 4](#_Toc466842076)

[2. 产品概述 4](#_Toc466842077)

[3. 体系结构设计概述 5](#_Toc466842078)

[4. 结构视角 5](#_Toc466842079)

[4.1 业务逻辑层的分解 5](#_Toc466842080)

[4.1.1 Hotel模块 5](#_Toc466842081)

[4.1.2 Order模块 9](#_Toc466842082)

[4.1.3 Room模块 15](#_Toc466842083)

[4.1.4 Promotion模块 19](#_Toc466842084)

[4.1.5 User模块 23](#_Toc466842085)

[5. 依赖视角 28](#_Toc466842086)

# 1. 引言

## 1.1 编制目的

本报告详细完成对酒店预订管理系统EasyHotel的详细设计，达到指导后续软件构造的目的，同时实现和测试人员及用户的沟通。

本报告面向开发人员、测试人员及最终用户而编写，是了解系统的导航。

## 1.2 词汇表

|  |  |  |
| --- | --- | --- |
| **词汇名称** | **词汇含义** | **备注** |
| EasyHotel | 酒店预订管理系统 |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 1.3 参考资料

1. 骆斌 丁二玉 刘钦. 软件工程与计算. 卷2, 软件开发的技术基础[M]. 机械工业出版社, 2013.

2. IEEE标准

3. 酒店预订管理系统EasyHotel用例文档

4. 酒店预订管理系统EasyHotel需求规格说明文档

5. 酒店预订管理系统EasyHotel体系结构设计文档

# 2. 产品概述

随着工作和生活节奏的加快，人们的差旅出行需要迅速提升，但是传统上的线下酒店预订模式有着很多不方便之处：

因为目的地往往是外地，无法全面了解当地的酒店情况，选择范围大大受限，难以选择到最适合的酒店；

提前预订的双方都有风险，酒店担心恶意预订，顾客担心酒店不符合预期；

酒店要及时掌握预订顾客的行程变化也会比较麻烦，需要频繁的长途联系；

虽然过去的体验能够给未来带来参考，但是时间一长很多信息就会被遗忘掉，过去的参考作用就受到限制。（事实上，过去的出行体验会大大影响人们未来的选择）

酒店预订管理系统EasyHotel就是为了方便人们出行中的酒店预订未开发的。它提供一个预订平台，酒店人员注册后可以维护酒店信息，顾客注册后进行酒店预订，网站营销人员和管理人员则能够对网站进行管理和信息维护。

# 3. 体系结构设计概述

酒店预订管理系统是典型的信息系统，MVC和分层是常见的两种选择，又因为开发人员不熟悉Java Swing以及RMI技术，对Web技术较了解，因此选择MVC作为系统的体系结构风格。MVC风格将系统分为3部分：模型Model、视图View、控制器Controller。模型封装应用数据，处理业务逻辑、视图负责呈现给用户的界面、控制器接受用户动作，并对应作出适当的处理。

详见体系结构设计文档。

# 4. 结构视角

## 4.1 业务逻辑层的分解

业务逻辑层的开发包图详见软件体系结构文档。

### 4.1.1 Hotel模块

（1）模块概述

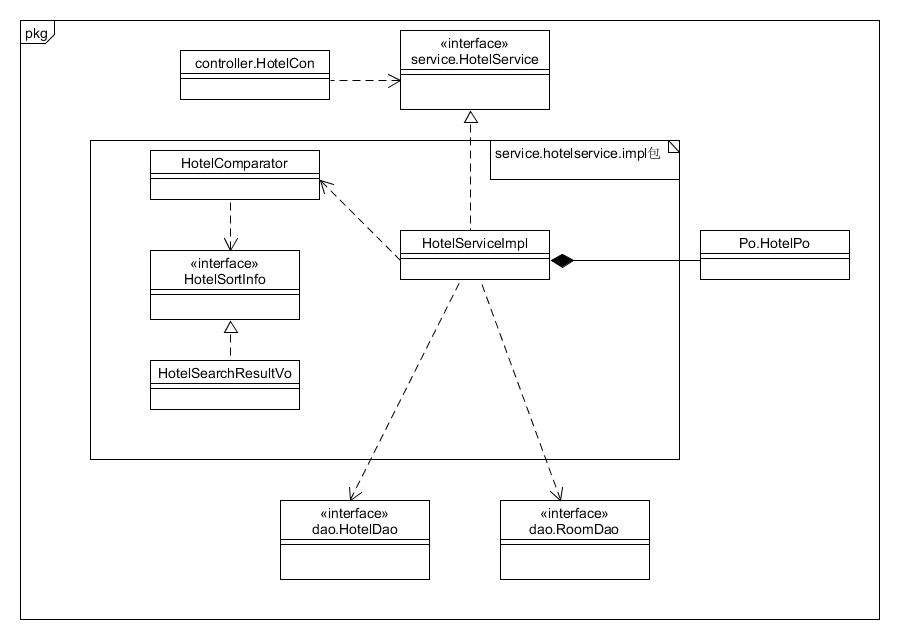
Hotel模块承担的需求参加需求规格说明文档功能需求及相关非功能需求。

Hotel模块的职责及接口参见软件系统结构描述文档。

（2）整体结构

根据体系结构的设计，我们将系统分为视图层、控制器层、模型层，模型层又细分为Service业务逻辑层和Dao数据访问层。每一层之间为了增加灵活性，我们会添加接口。控制器层和Service层之间添加了service.HotelService接口。Service层和Dao层之间添加了dao.HotelDao接口。HotelPo是作为酒店记录的持久化对象被添加到设计模型中去的。HotelComparator类的添加是为了处理酒店搜索的排序逻辑，内部通过实现java的Comparator接口完成对酒店搜索结果的排序，HotelSortInfo接口定义了要进行排序的对象应该遵循的规范，使得排序逻辑可以被其他模块复用。

Hotel模块的设计如图所示。



hotel模块各个类的设计

hotel模块各个类的职责如下表所示：

|  |  |
| --- | --- |
| 模块 | 职责 |
| HotelServiceImpl | 负责实现与酒店相关的业务逻辑 |

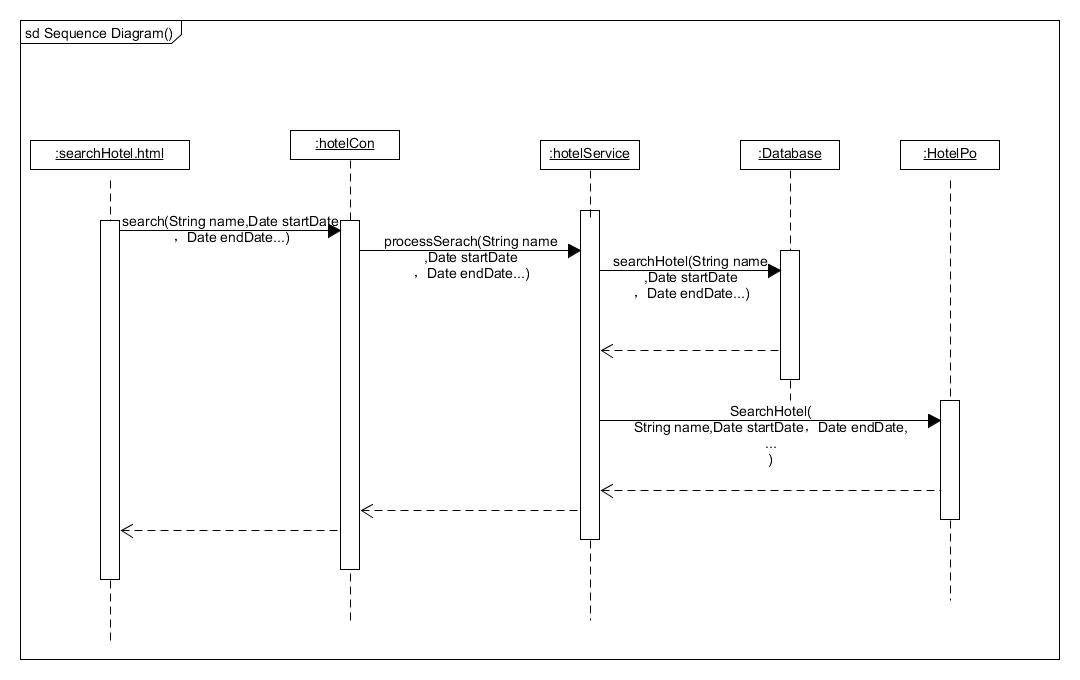
（3）模块内部类的接口规范

HotelServiceImpl类的接口规范

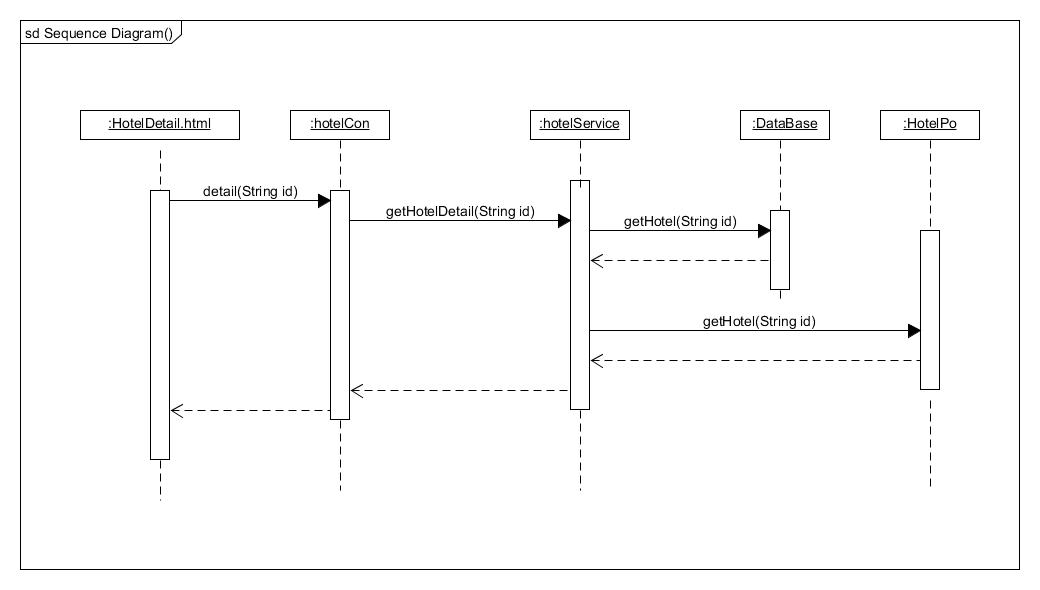
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| HotelServiceImpl.getHotelDetai | 语法 | | public HotelVo getHotel(String hotelId) |
| 前置条件 | | 酒店ID正确且符合规则 |
| 后置条件 | | 根据酒店ID查找酒店，返回其详细信息 |
| HotelServiceImpl.getAllHotels | 语法 | | public List<HotelVo>getAllHotels() |
| 前置条件 | | 点击查询所有酒店 |
| 后置条件 | | 返回包含所有酒店的列表 |
| HotelServiceImpl.searchHotels | 语法 | | public List<HotelVo> searchHotels(SearchCondition con) |
| 前置条件 | | 搜索条件输入且选择正确 |
| 后置条件 | | 根据条件查询酒店，并返回包含符合条件的酒店列表 |
| HotelServiceImpl.sortHotels | 语法 | | public List<HotelVo> sortHotels(List<HotelVo> hotels ,SortCondition con) |
| 前置条件 | | 选择排序关键字 |
| 后置条件 | | 根据排序关键字对酒店列表中的酒店排序，返回排序后的酒店列表 |
| HotelServiceImpl.addHotel | 语法 | | public opMessage addHotel(HotelPo hotel) |
| 前置条件 | | 酒店对象正确 |
| 后置条件 | | 根据输入的酒店信息增加酒店，返回添加结果 |
| HotelServiceImpl.deleteHotel | 语法 | | public opMessage deleteHotel(String hotelId) |
| 前置条件 | | 酒店id正确且符合格式 |
| 后置条件 | | 根据酒店id删除酒店，返回删除结果 |
| HotelServiceImpl.ModifyHotel | 语法 | | public opMessage modifyHotel(HotelPo hotel) |
| 前置条件 | | 酒店对象正确 |
| 后置条件 | | 修改该酒店的信息，返回修改结果 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| hotelDao.insert | | 在数据库中插入新的酒店信息 | |
| hotelDao.update | | 在数据库中更新指定酒店的酒店信息 | |
| hotelDao.delete | | 根据酒店id在数据库中删除该酒店信息 | |
| hotelDao.getHotelById | | 根据酒店id查找酒店 | |
| hotelDao.getAllHotels | | 查找所有酒店 | |
| hotelDao.query | | 根据查询条件查找酒店 | |
| roomDao.getallRoomsByHotel | | 根据酒店id得到其所有房间 | |

（4）业务逻辑的动态模型

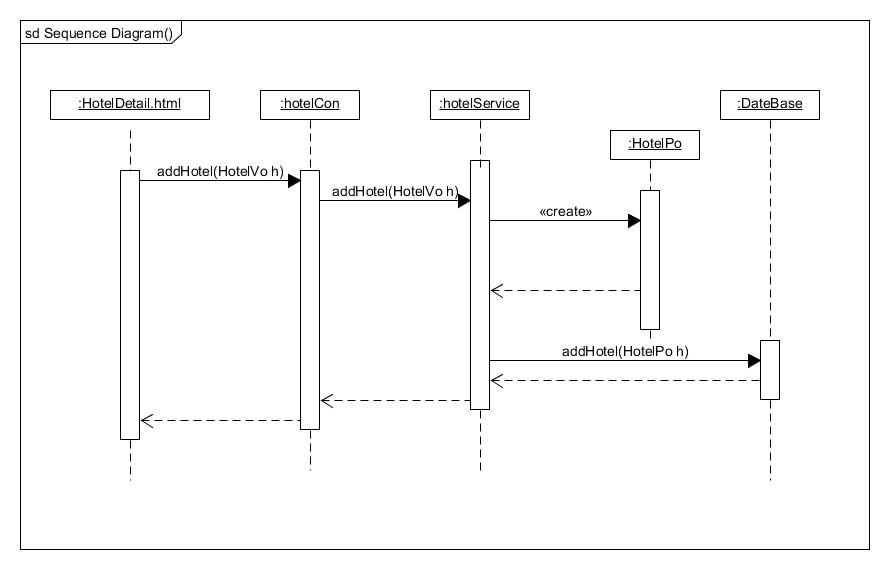
搜索酒店的顺序图



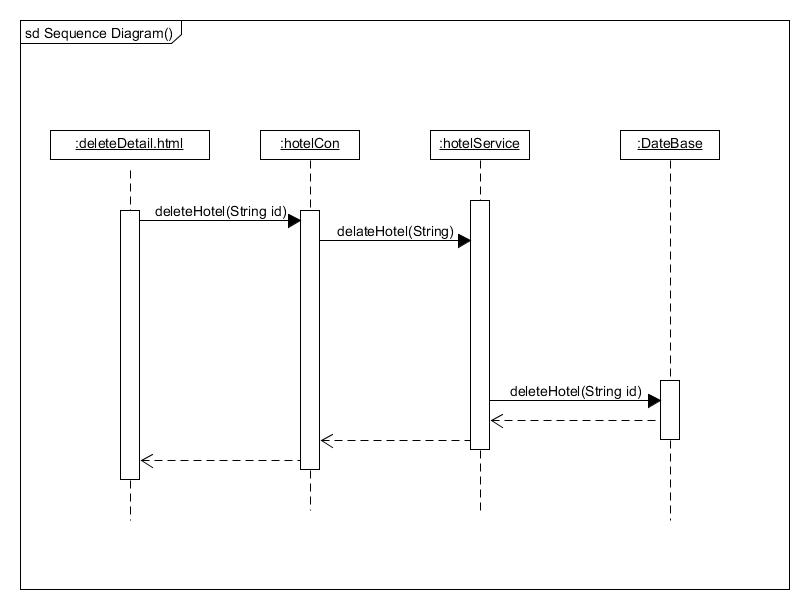
查看酒店细节信息顺序图



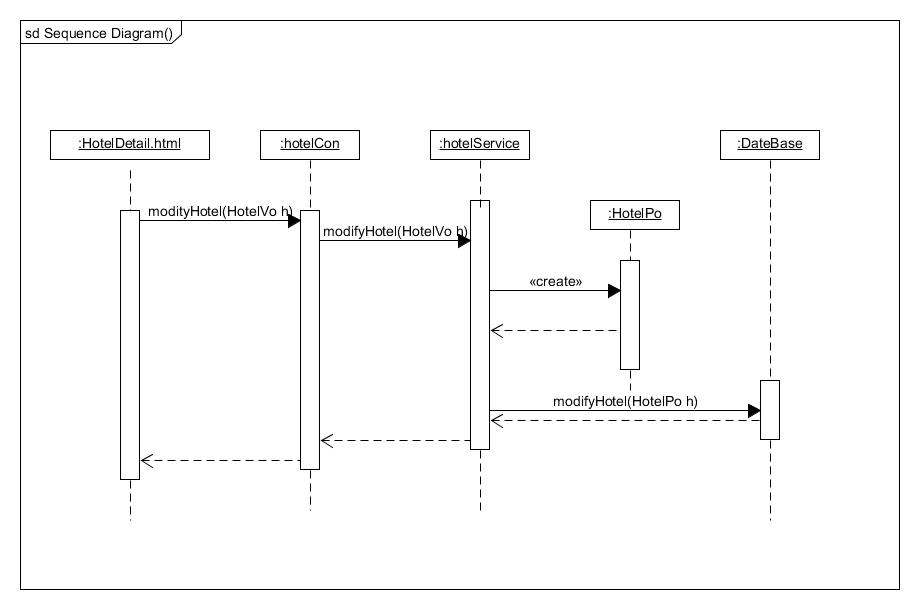
添加酒店顺序图



删除酒店顺序图



修改酒店顺序图



### 4.1.2 Order模块

（1）模块概述

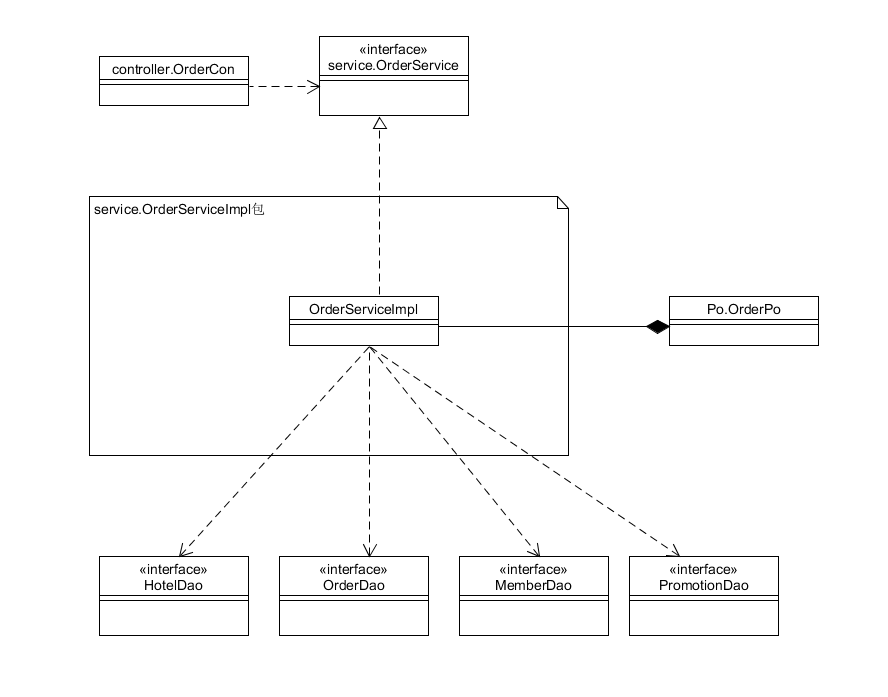
Order模块承担的需求参加需求规格说明文档功能需求及相关非功能需求。

Order模块的职责及接口参见软件系统结构描述文档。

（2）整体结构

根据体系结构的设计，我们将系统分为视图层、控制器层、模型层，模型层又细分为Service业务逻辑层和Dao数据访问层。每一层之间为了增加灵活性，我们会添加接口。控制器层和Service层之间添加了service.OrderService接口。Service层和Dao层之间添加了dao.OrderDao接口。OrderPo是作为酒店记录的持久化对象被添加到设计模型中去的。

Hotel模块的设计如图所示。



Order模块各个类的职责如下表所示：

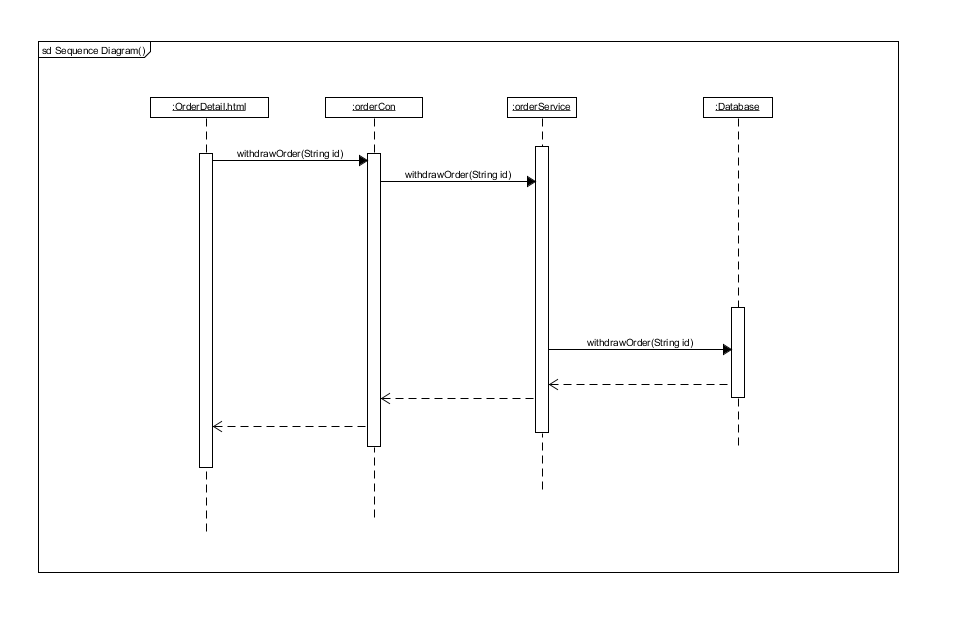
|  |  |
| --- | --- |
| 模块 | 职责 |
| OrderServiceImpl | 负责实现与订单相关的业务逻辑 |

（3）模块内部类的接口规范

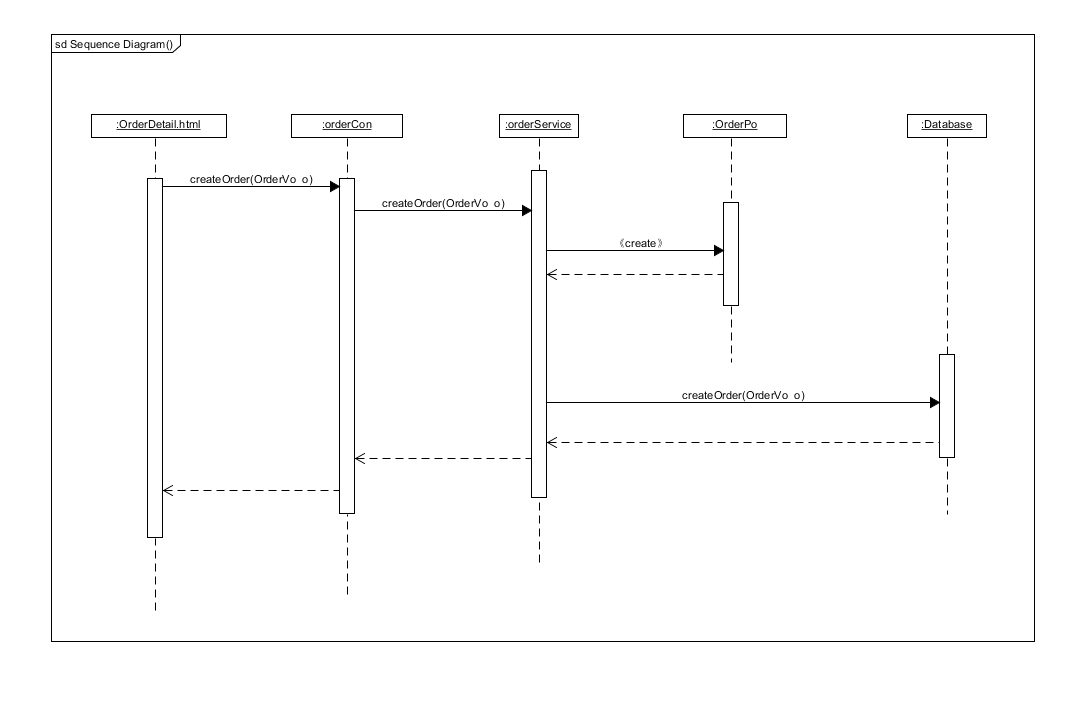
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| orderService.getAllOrdersByCustomer | 语法 | public List<OrderVo> getAllOrdersByCustomer (String customer Id) |
| 前置条件 | 客户ID正确且符合规范 |
| 后置条件 | 根据客户id查找相应的用户，返回其所有订单 |
| orderService.getAllOrdersByHotel | 语法 | public List<OrderVo> getAllOrdersByHotel(String hotelId) |
| 前置条件 | 酒店ID正确且符合规范 |
| 后置条件 | 根据酒店id查账相应的酒店，返回其所有订单 |
| orderService.getAllCustomerOrdersByState | 语法 | public List<OrderVo> getAllCustomerOrdersByState (String customerId,String state) |
| 前置条件 | 客户ID及订单状态正确且符合规范 |
| 后置条件 | 根据客户的id查找用户，返回其指定状态的订单 |
| orderService.getAllHotelOrdersByState | 语法 | public List<OrderVo> getAllHotelOrdersByState (String hotelId,String state) |
| 前置条件 | 酒店ID及订单状态正确且符合规范 |
| 后置条件 | 根据酒店的id查找酒店，返回其指定状态的订单 |
| orderService.getOrder | 语法 | public OrderVo getOrder(String orderId) |
| 前置条件 | 订单ID正确且符合规范 |
| 后置条件 | 根据订单id找到订单并将订单信息返回 |
| orderService.placeOrder | 语法 | public OpMessage placeOrder(OrderPo order,String memberId,String hotelId,String promotionId) |
| 前置条件 | 会员ID、酒店ID、促销ID正确且符合规范 |
| 后置条件 | 根据用户输入的内容生成订单，并将生成结果返回 |
| orderService.revokeOrder | 语法 | public OpMessage revokeOrder(String OrderId) |
| 前置条件 | 订单ID正确且符合规范 |
| 后置条件 | 根据订单id将此订单撤销，并返回撤销结果 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| memberService.getMember | 根据会员id返回该会员 | |
| hotelService.getHotel | 根据酒店id返回该酒店 | |
| promotionService.getPromotion | 根据促销信息id返回该促销信息 | |
| orderDao.insert | 在数据库中增加该订单信息 | |
| orderDao.update | 在数据库中更新该订单信息 | |
| orderDao.delete | 在数据库中删除该订单信息 | |
| orderDao.getAllOrdersByCustomer | 根据会员id返回该会员所有订单 | |
| orderDao.getAllOrdersByHotel | 根据酒店id返回该酒店所有订单 | |
| orderDao.getAllCustomerOrdersByState | 根据会员id和订单状态返回该会员相关订单 | |
| orderDao.getAllHotelOrdersByState | 根据酒店id和订单状态返回该酒店相关订单 | |
| orderDao.getOrder | 根据订单id返回该订单信息 | |

（4）业务逻辑层的动态模型

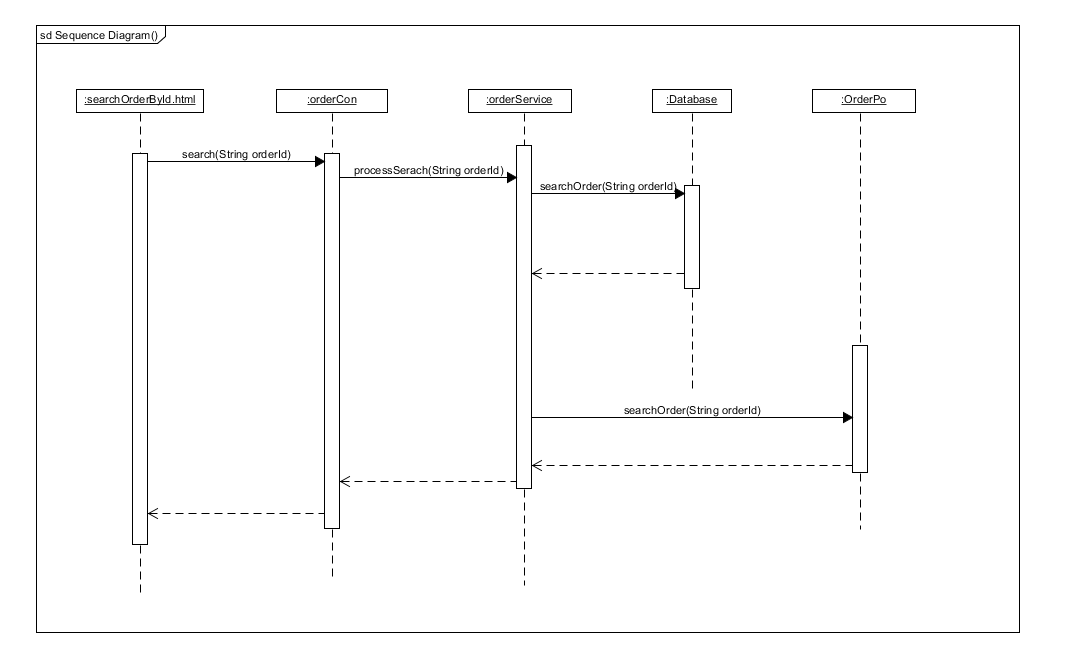
撤销订单



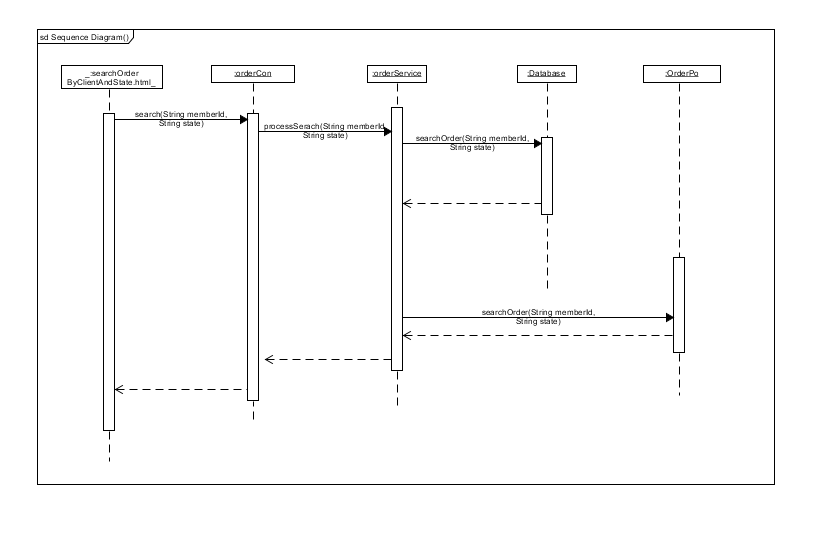
生成订单



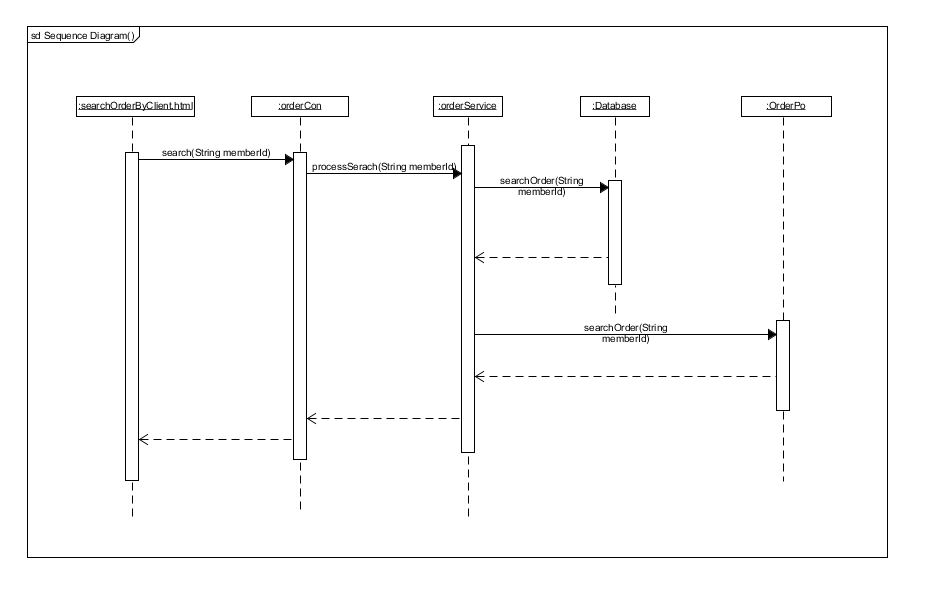
通过订单编号搜索订单



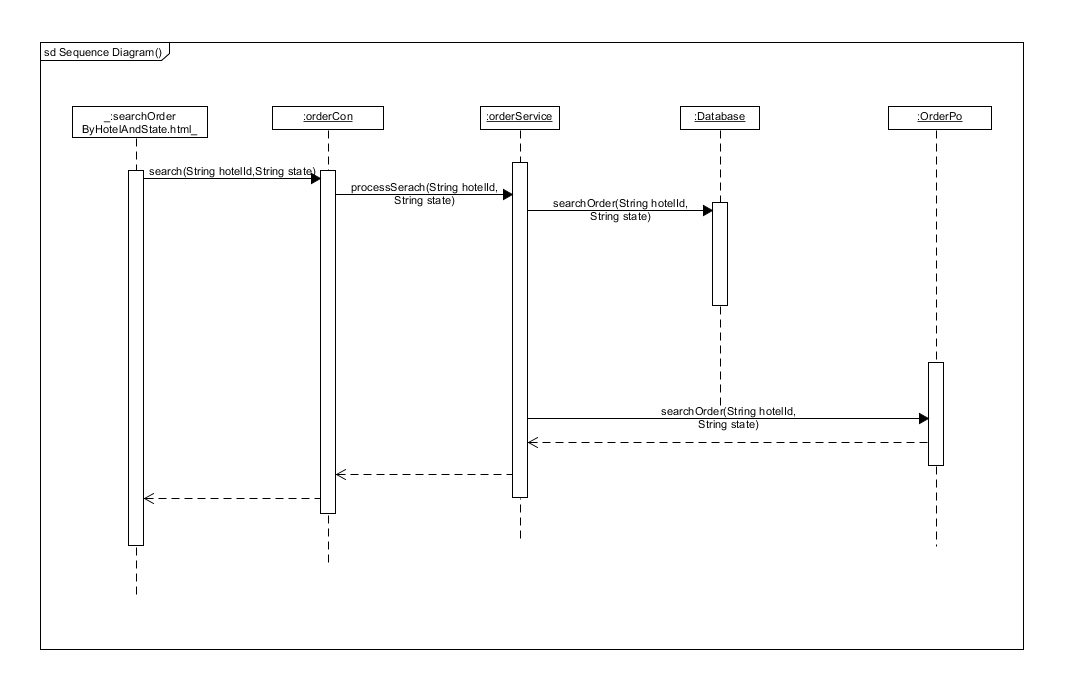
通过会员编号及订单状态搜索订单



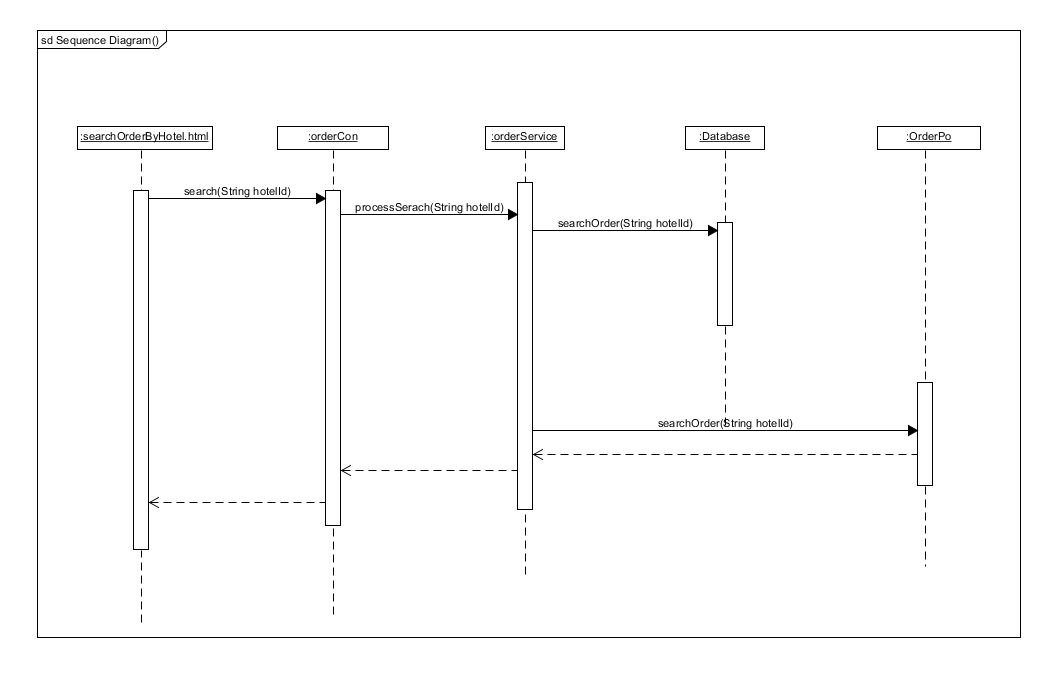
通过会员编号搜索订单



通过酒店编号及订单状态搜索订单



通过酒店编号搜索订单



### 4.1.3 Room模块

（1）模块概述

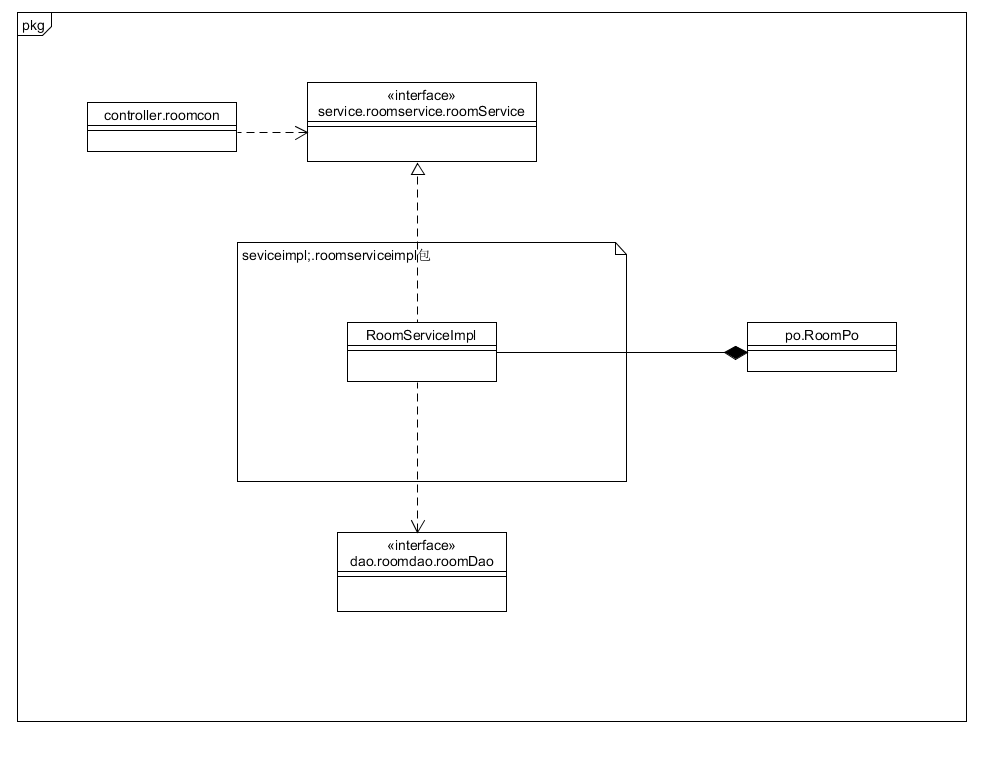
Room模块承担的需求参加需求规格说明文档功能需求及相关非功能需求。

Room模块的职责及接口参见软件系统结构描述文档。

（2）整体结构

根据体系结构的设计，我们将系统分为视图层、控制器层、模型层，模型层又细分为Service业务逻辑层和Dao数据访问层。每一层之间为了增加灵活性，我们会添加接口。控制器层和Service层之间添加了service.RoomService接口。Service层和Dao层之间添加了dao.RoomDao接口。Roompo是作为酒店记录的持久化对象被添加到设计模型中去的。

room模块的设计如图所示



room模块的各个类的职责如下表所示。

|  |  |
| --- | --- |
| 模块 | 职责 |
| RoomServiceImpl | 负责实现与客房相关的业务逻辑 |

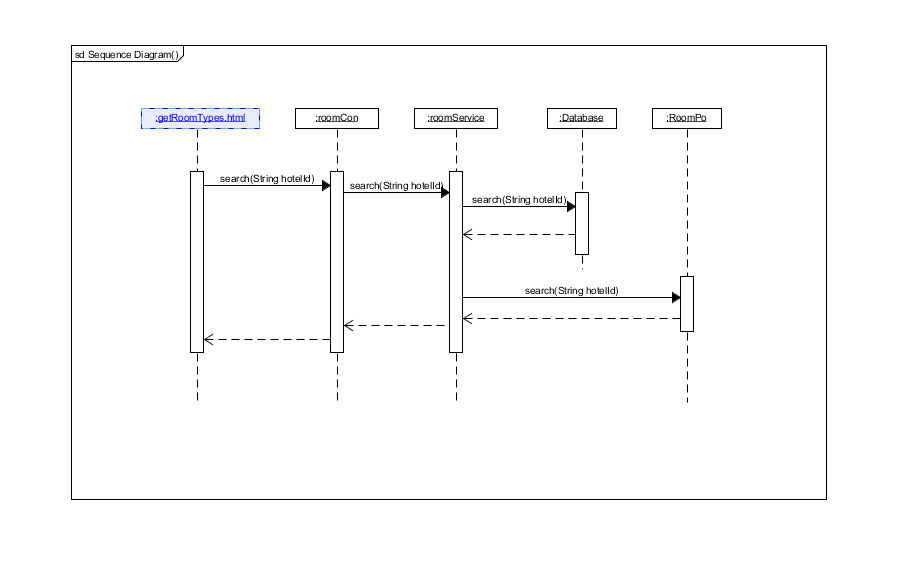
（3）模块内部类的接口规范

RoomServiceImpl的接口规范如下表所示。

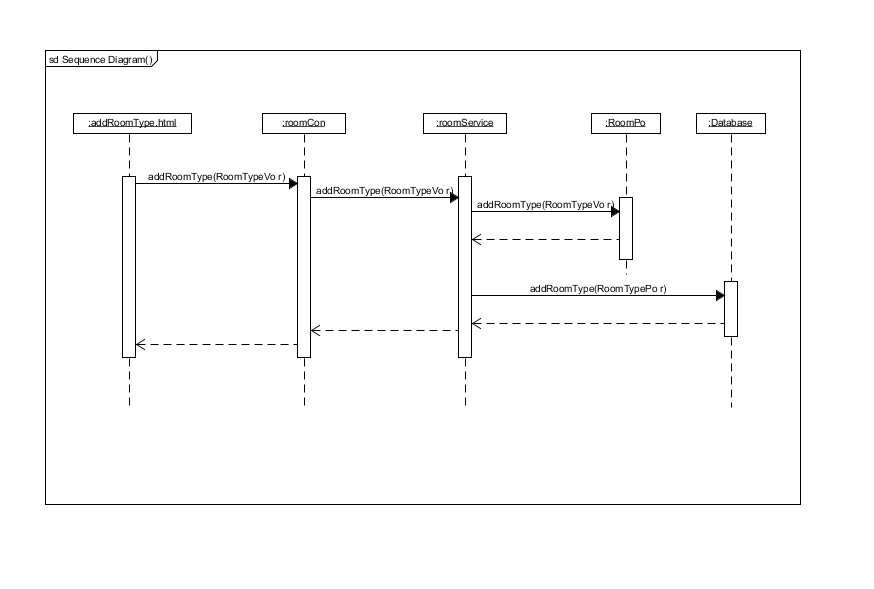
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| RoomServiceImpl.get getRoomsByHotel | 语法 | public List<RoomVo> getRoomsByHotel (String HotelId) |
| 前置条件 | 酒店id正确且符合标准 |
| 后置条件 | 根据酒店id查找所有房间，并返回包含所有房间的列表 |
| RoomServiceImpl.deleteRoom | 语法 | public opMessage deleteRoom(String roomId) |
| 前置条件 | 房间id正确且符合标准 |
| 后置条件 | 根据房间id删除房间，并返回删除结果 |
| RoomServiceImpl.modifyRoom | 语法 | public opMessage modifyRoom(RoomPo room) |
| 前置条件 | 房间对象正确 |
| 后置条件 | 修改指定房间的信息 |
| RoomServiceImpl.addRoom | 语法 | public opMessage addRoom(RoomPo room) |
| 前置条件 | 房间对象正确 |
| 后置条件 | 根据信息增加房间信息，返回添加结果 |
| RoomServiceImpl. updateRoomNumber(boolean isCheckin) | 语法 | public opMessage updateRoomNumber(boolean isCheckin) |
| 前置条件 | 布尔类型正确 |
| 后置条件 | 更新可用房间数量，返回结果 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| roomDao.insert | 在数据库中插入一条房间信息 | |
| roomDao.update | 在数据库中修改一条房间信息 | |
| roomDao.delete | 在数据库中删除一条房间信息 | |
| roomDao. updateRoomNumber | 在数据库中更改可用房间数量 | |
| roomDao.getRoomsByHotel | 根据酒店id在数据库中查找该酒店所有的房间 | |

（4）业务逻辑层的动态模型

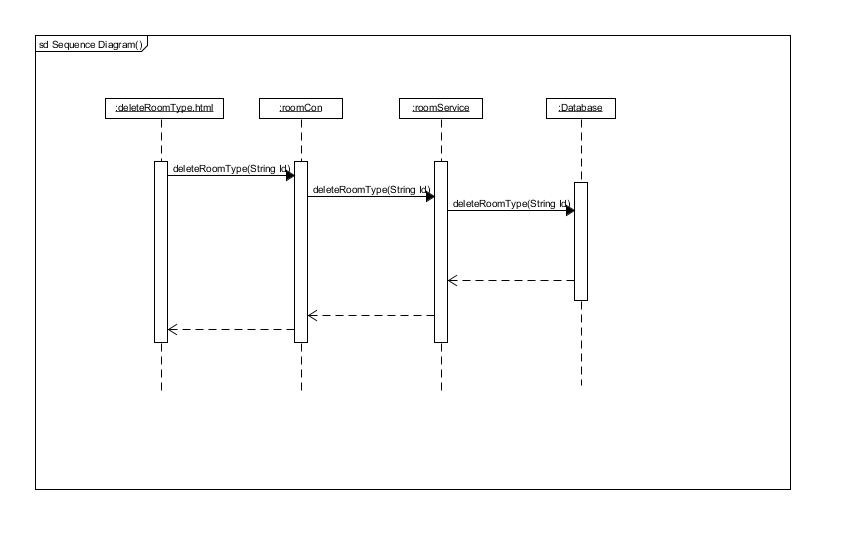
通过酒店查找客房信息



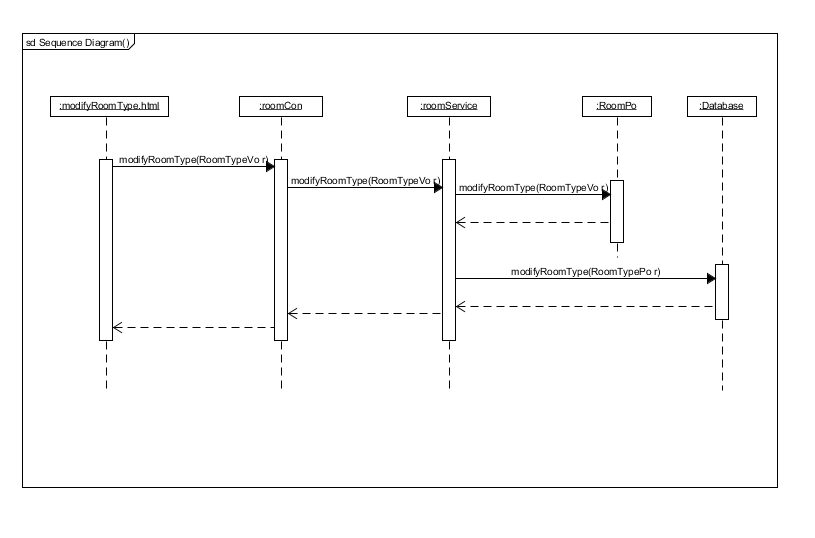
添加房间



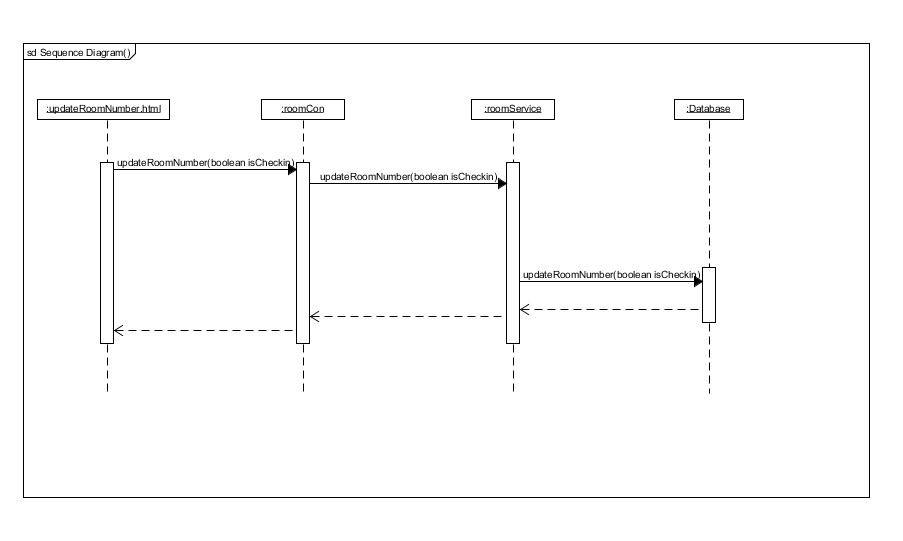
删除房间



修改房间



更新可用房间数量



### 4.1.4 Promotion模块

（1）模块概述

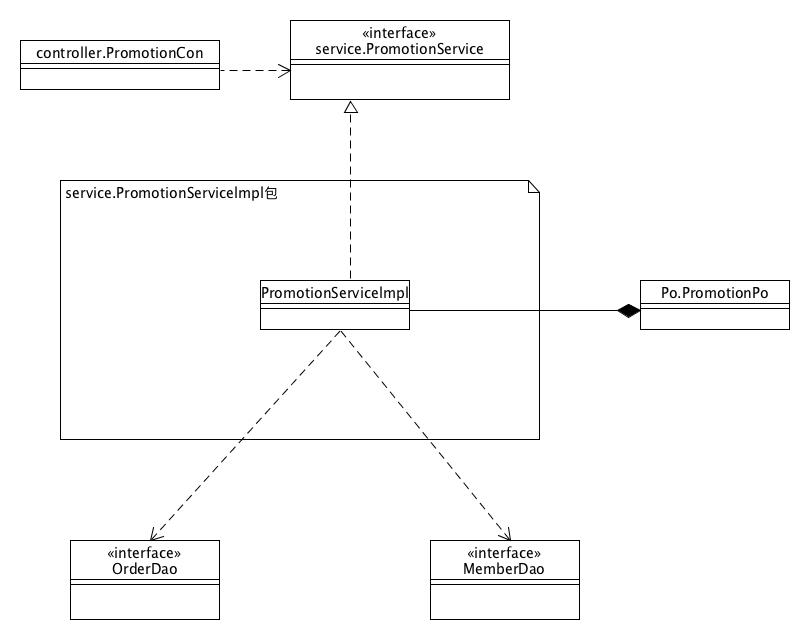
Promotion模块承担的需求参加需求规格说明文档功能需求及相关非功能需求。

Promotion模块的职责及接口参见软件系统结构描述文档。

（2）整体结构

根据体系结构的设计，我们将系统分为视图层、控制器层、模型层，模型层又细分为Service业务逻辑层和Dao数据访问层。每一层之间为了增加灵活性，我们会添加接口。控制器层和Service层之间添加了service.PromotionService接口。Service层和Dao层之间添加了dao.ServiceDao接口。PromotionPo是作为促销记录的持久化对象被添加到设计模型中去的。

Promotion模块的设计如图所示。



Promotion模块各个类的设计

Promotion模块各个类的职责如下表所示：

|  |  |
| --- | --- |
| 模块 | 职责 |
| PromotionServiceImpl | 负责实现与促销活动相关的业务逻辑 |

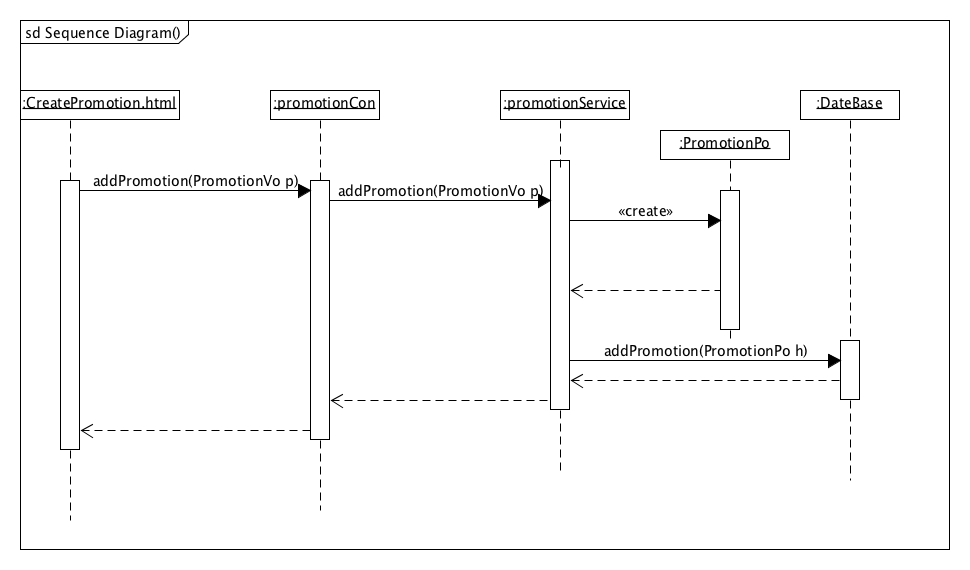
（3）模块内部类的接口规范

PromotionServiceImpl类的接口规范

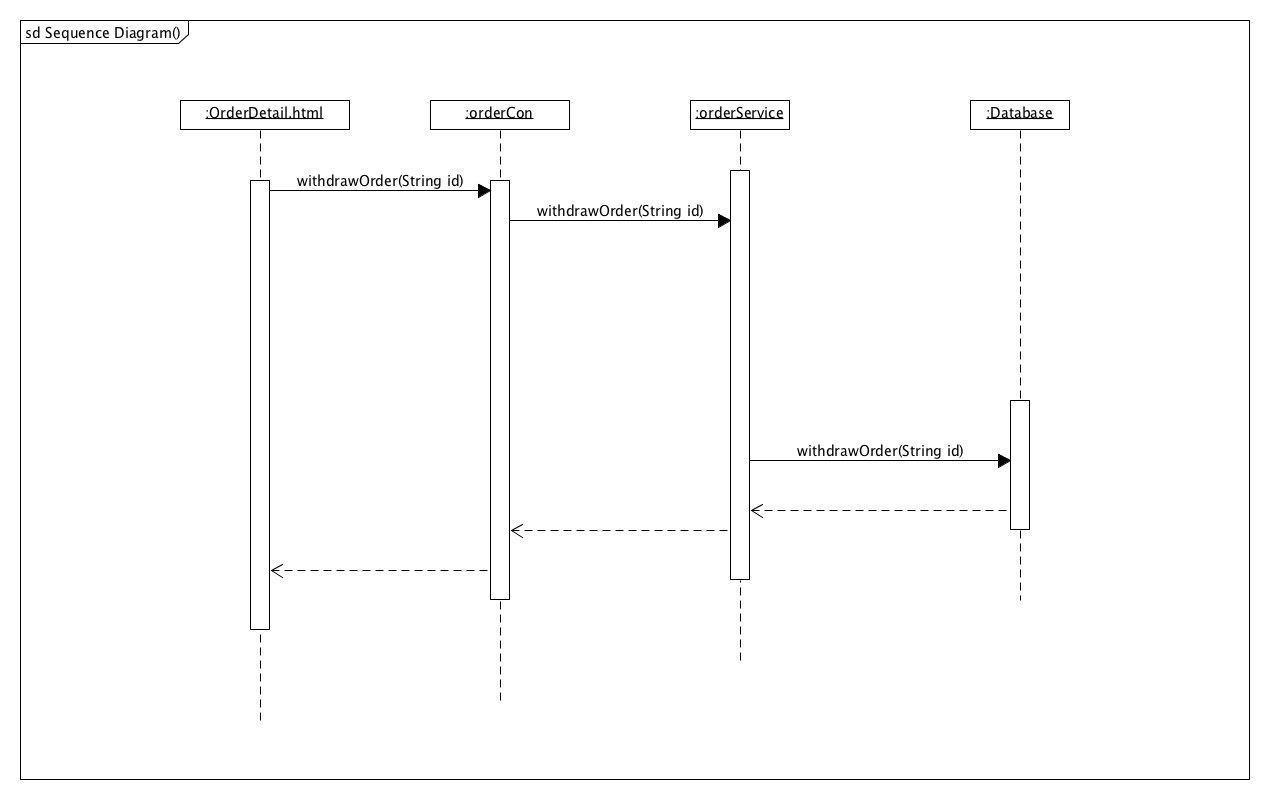
|  |  |  |  |
| --- | --- | --- | --- |
| promotionService.getPromotion | | 语法 | public promotionVo getPromotion (String promotionId) |
| 前置条件 | 促销信息id正确且符合标准 |
| 后置条件 | 根据促销信息id查找促销信息并返回 |
| promotionService.getAllPromotionsByHotel | | 语法 | public List<PromotionVo> getHotelPromotions(String hotelId) |
| 前置条件 | 酒店id正确且符合格式 |
| 后置条件 | 根据酒店id查找所有促销信息，并返回包含所有促销信息的列表 |
| promotionService.getAllSitePromotions | | 语法 | public List<PromotionVo> getSitePromotions() |
| 前置条件 | 点击搜索网站促销信息 |
| 后置条件 | 返回包含所有网站促销信息的列表 |
| promotionService.addPromotion | | 语法 | public opMessage addPromotion (PromotionPo promotion) |
| 前置条件 | 促销信息对象正确 |
| 后置条件 | 添加促销，并返回添加结果 |
| promotionService.modifyPromotion | | 语法 | public opMessage modifyPromotion (PromotionPo promotion) |
| 前置条件 | 促销信息对象正确 |
| 后置条件 | 添加促销，并返回添加结果 |
| promotionService.deletePromotion | | 语法 | public opMessage deletePromotion (String promotionId) |
| 前置条件 | 促销信息id正确且符合标准 |
| 后置条件 | 根据促销信息id删除该促销，返回删除结果 |
| 需要的服务（需接口） | | | |
| 服务名 | 服务 | | |
| promotionDao.getPromotionById | 根据促销信息id查找促销信息 | | |
| promotionDao.insert | 在数据库中增加一条促销信息 | | |
| promotionDao.update | 在数据库中修改一条促销信息 | | |
| promotionDao.delete | 在数据库中删除一条促销信息 | | |
| promotionDao.getAllSitePromotions | 返回所有网站促销信息 | | |
| promotionDao.getAllPromotionsByHotel | 根据酒店id查找所有促销信息 | | |

（4）业务逻辑的动态模型

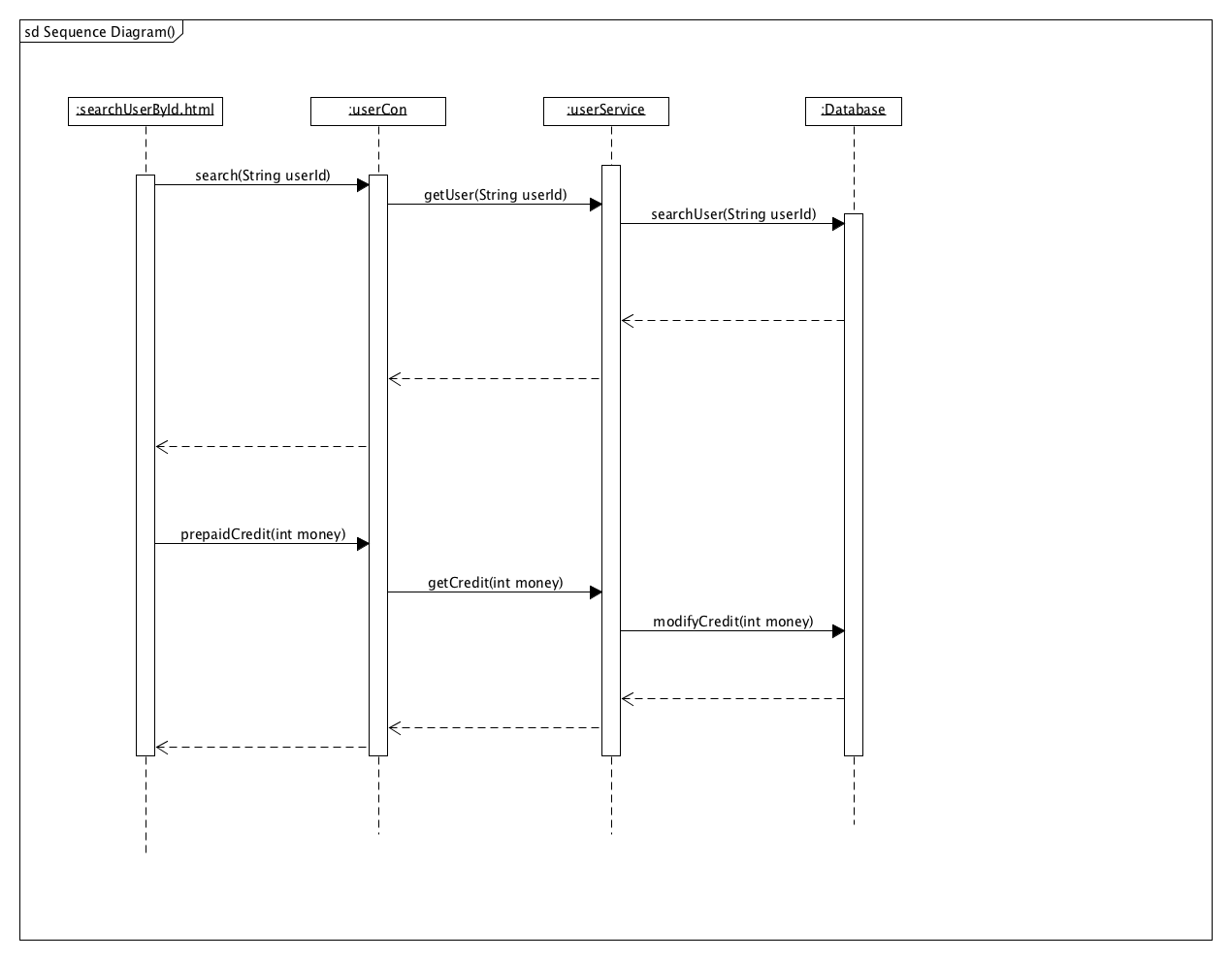
制定促销策略的顺序图



处理异常订单顺序图



信用充值顺序图



### 4.1.5 User模块

（1）模块概述

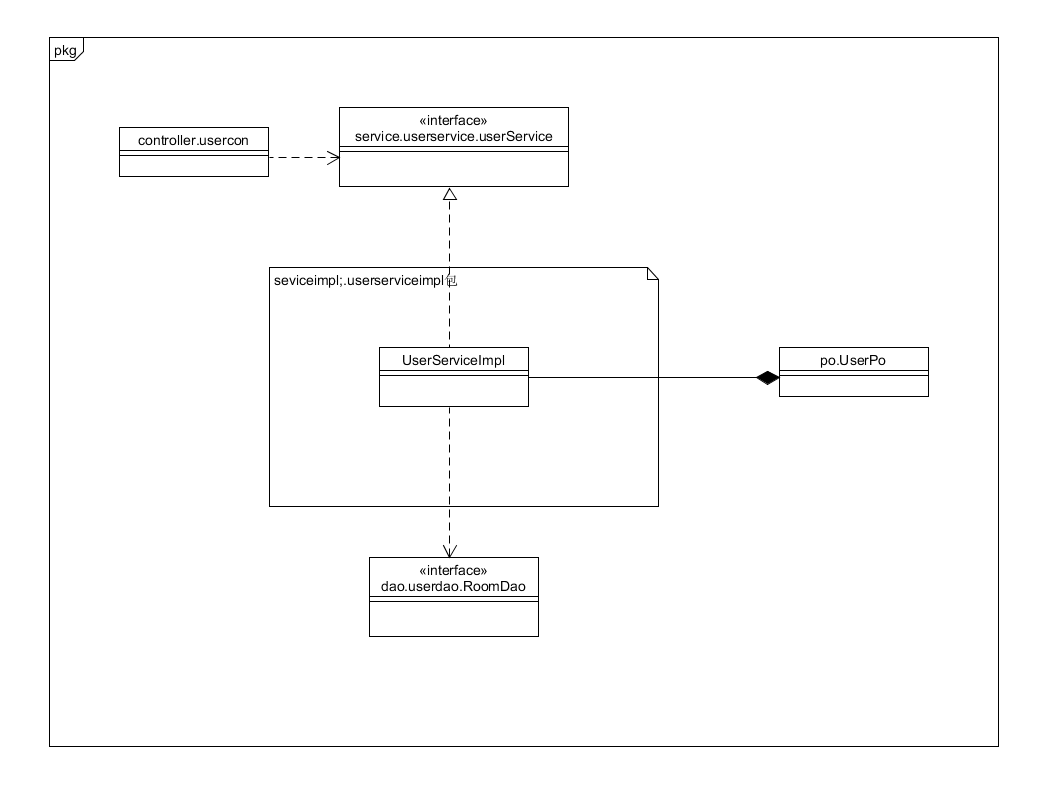
User模块承担的需求参加需求规格说明文档功能需求及相关非功能需求。

User模块的职责及接口参见软件系统结构描述文档。

（2）整体结构

根据体系结构的设计，我们将系统分为视图层、控制器层、模型层，模型层又细分为Service业务逻辑层和Dao数据访问层。每一层之间为了增加灵活性，我们会添加接口。控制器层和Service层之间添加了service.User Service接口。Service层和Dao层之间添加了dao.UserDao接口。userPo是作为促销记录的持久化对象被添加到设计模型中去的。

User模块的设计如图所示。



User模块各个类的职责如下表所示：

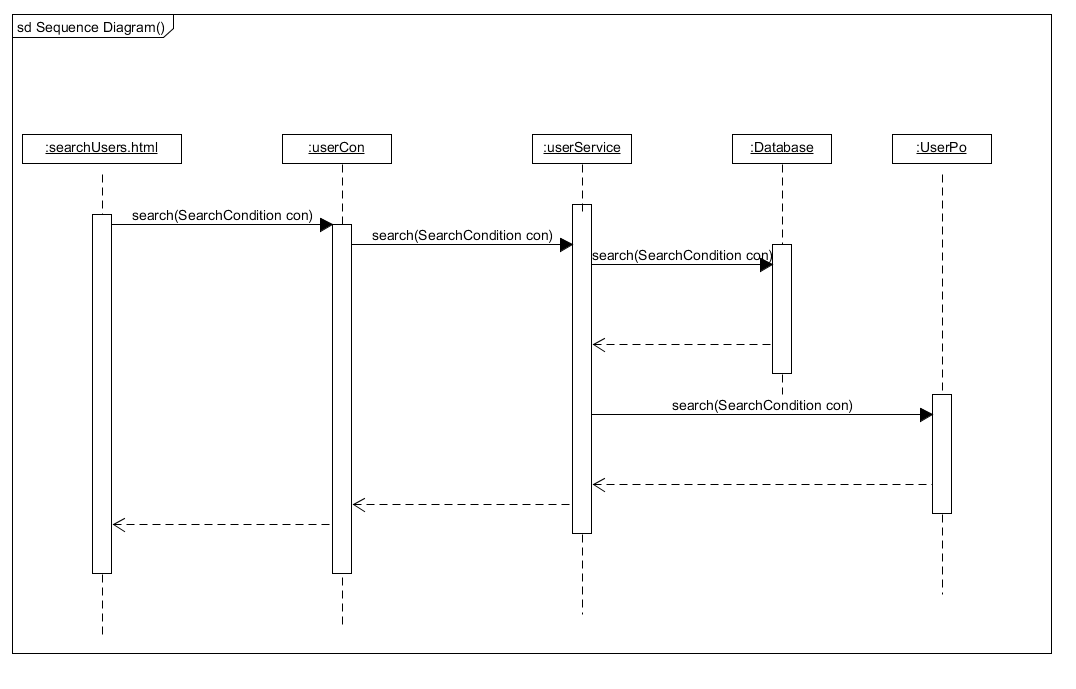
|  |  |
| --- | --- |
| 模块 | 职责 |
| UserServiceImpl | 负责实现与用户相关的业务逻辑 |

（3）模块内部类的接口规范

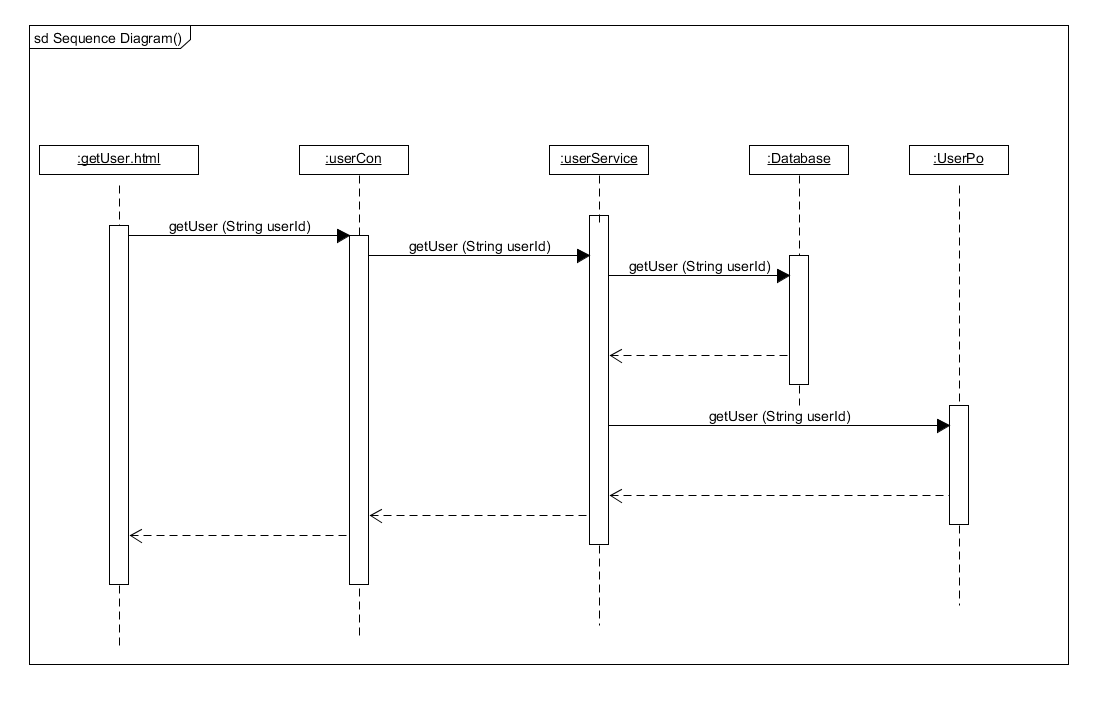
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| userService.searchUsers | | 语法 | public List<UserVo> searchUsers(SearchCondition con) |
| 前置条件 | 搜索条件正确 |
| 后置条件 | 根据搜索条件查找相关用户，返回包括相关用户的列表 |
| userService.getUser | | 语法 | public UserVo getUser (String userId) |
| 前置条件 | 用户id正确且符合格式 |
| 后置条件 | 根据用户id查找用户，并返回其信息 |
| userrService.getAllUsers | | 语法 | public List<UserVo> getUsers() |
| 前置条件 | 点击查找所有用户 |
| 后置条件 | 查找系统内所有用户，返回包含所有用户的列表 |
| userService.addUser | | 语法 | public opMessage addUser (UserPo user) |
| 前置条件 | 用户对象正确 |
| 后置条件 | 添加一条用户信息，返回添加结果 |
| userService.modifyUser | | 语法 | public opMessage modifyUser (UserPo user) |
| 前置条件 | 用户对象正确 |
| 后置条件 | 修改指定用户信息，返回修改结果 |
| userService.deleteUser | | 语法 | public opMessage deleteUser (String userId) |
| 前置条件 | 用户id正确且符合格式 |
| 后置条件 | 根据用户id删除该用户，返回删除结果 |
| userService.memberLogin | | 语法 | public opMessage memberLogin(MemberVo member) |
| 前置条件 | 会员对象正确 |
| 后置条件 | 返回登录结果 |
| userService.userLogin | | 语法 | public opMessage userLogin(UserVo user) |
| 前置条件 | 用户对象正确 |
| 后置条件 | 返回登录结果 |
| userService.memberRegister | | 语法 | public opMessage memberRegister(MemberVo member) |
| 前置条件 | 会员对象正确 |
| 后置条件 | 返回注册结果 |
| 需要的服务（需接口） | | | |
| 服务名 | 服务 | | |
| memberService.getMember | 根据会员id查找会员信息 | | |
| userDao.insert | 在数据库中添加一条用户信息 | | |
| userDao.delete | 在数据库中删除一条用户信息 | | |
| userDao.update | 在数据库中更新一条用户信息 | | |
| userDao.getUser | 根据用户id查找用户信息 | | |
| userDao.getAllUsers | 查找数据库中所有用户 | | |
| userDao.query | 根据搜索条件查找相关用户 | | |

（4）业务逻辑层的动态模型

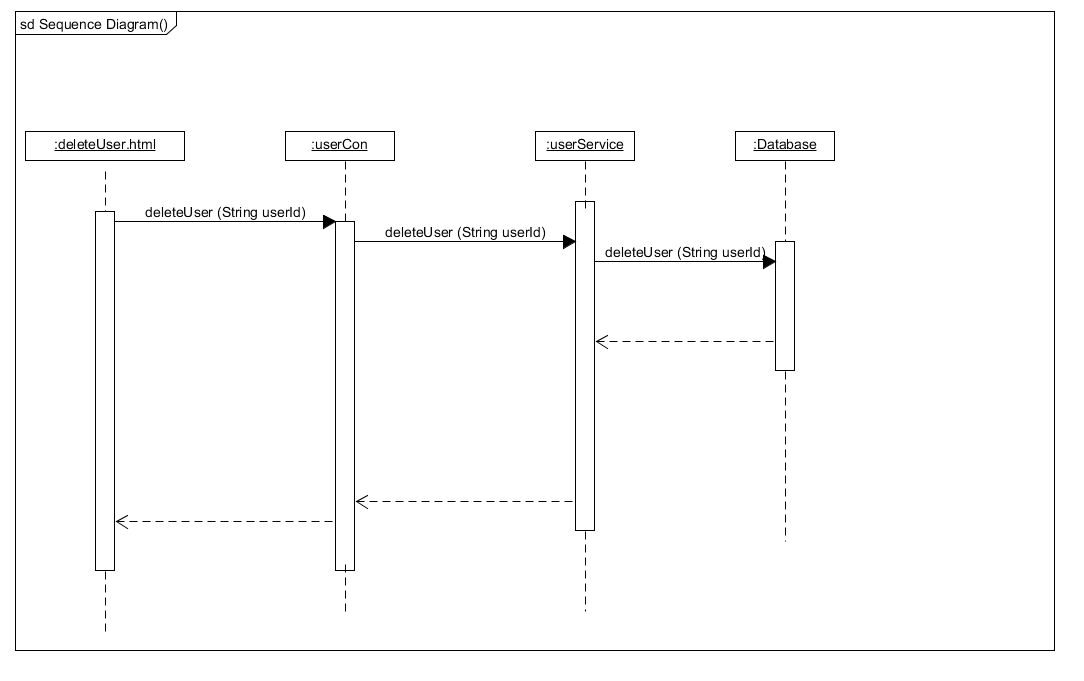
根据搜索条件查找相关用户



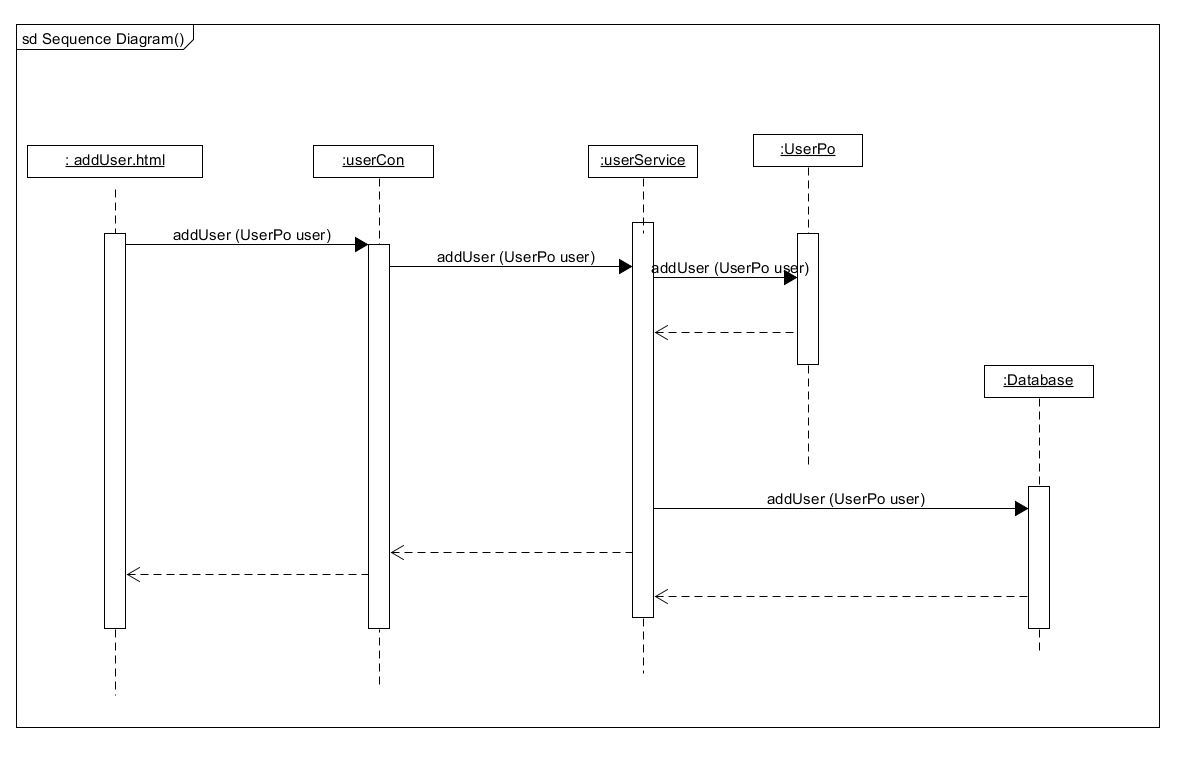
根据用户ID寻找用户



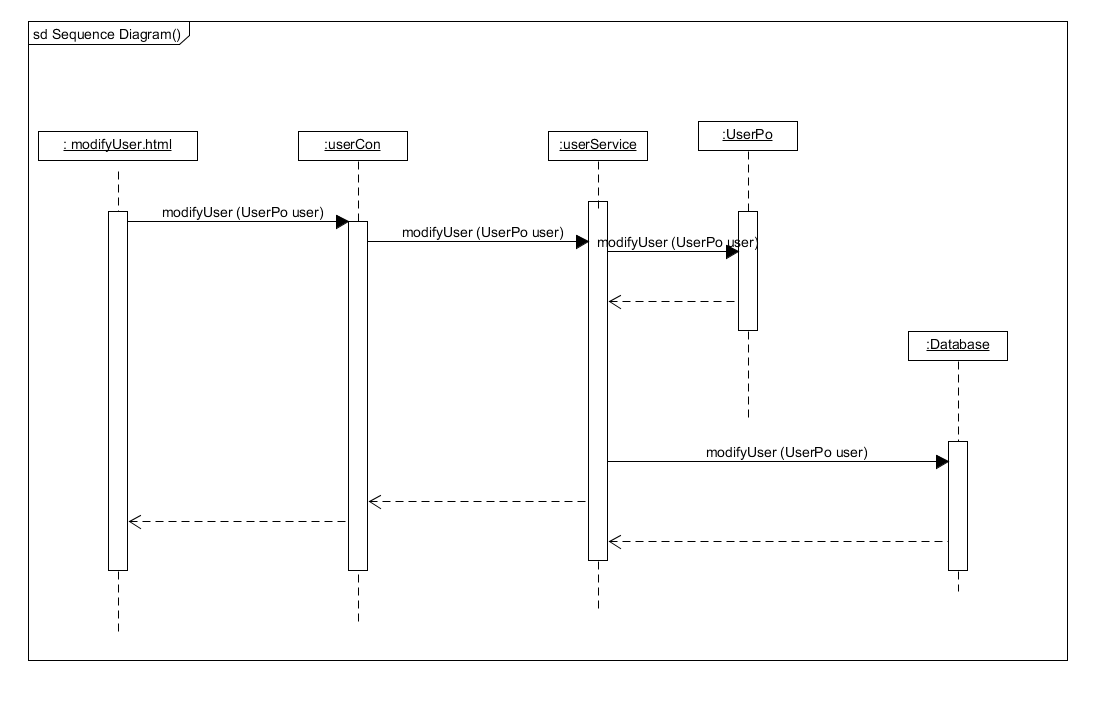
根据用户ID删除用户



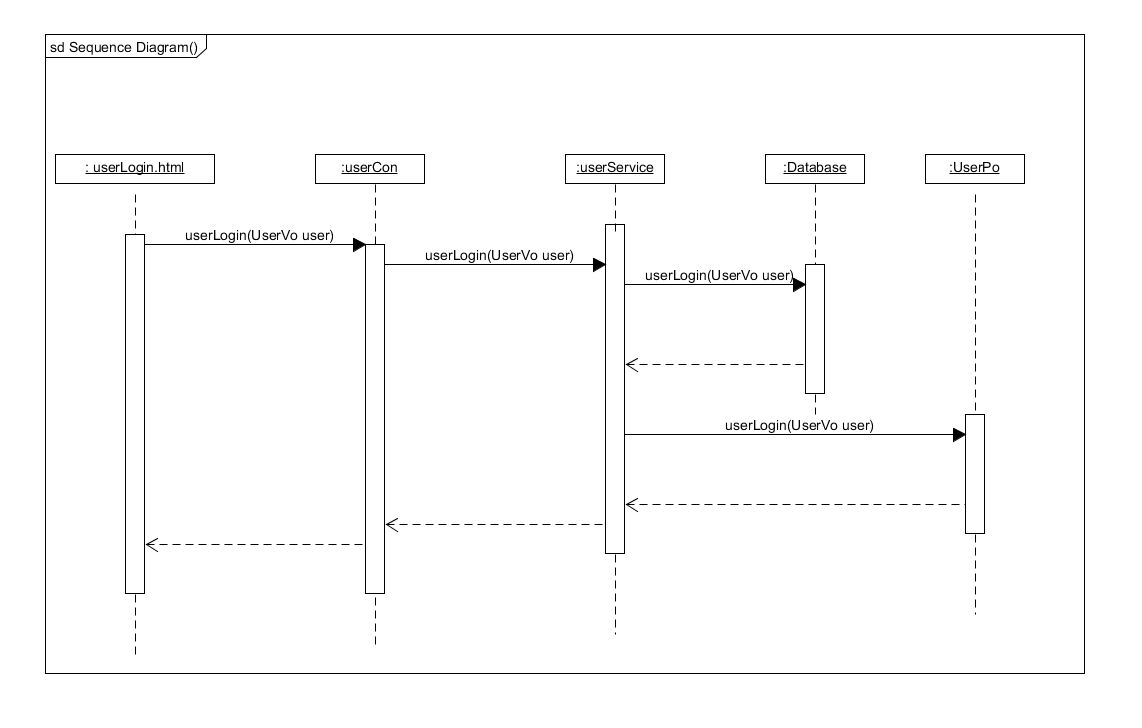
添加用户



修改指定用户信息



会员登陆



# 5. 依赖视角

下图是各个包之间的依赖关系。

