**酒店预订管理系统EasyHotel**

**软件详细设计描述文档**

**V0.99 初稿**

**EasyHotel小组**

**王涛 王昕尧 李玉莹 雷雨佳 方雅婧**

**2016-11-06**

# 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 |

目录

[更新历史 2](#_Toc465526523)

[1. 引言 3](#_Toc465526524)

[1.1 编制目的 3](#_Toc465526525)

[1.2 词汇表 3](#_Toc465526526)

[1.3 参考资料 3](#_Toc465526527)

[2. 产品概述 3](#_Toc465526528)

[3. 体系结构设计概述 4](#_Toc465526529)

[4. 结构视角 4](#_Toc465526530)

[4.1 业务逻辑层的分解 4](#_Toc465526531)

[4.1.1 hotel模块 4](#_Toc465526532)

[4.1.2 Promotion模块 8](#_Toc465526533)

[5. 依赖视角 12](#_Toc465526534)

# 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

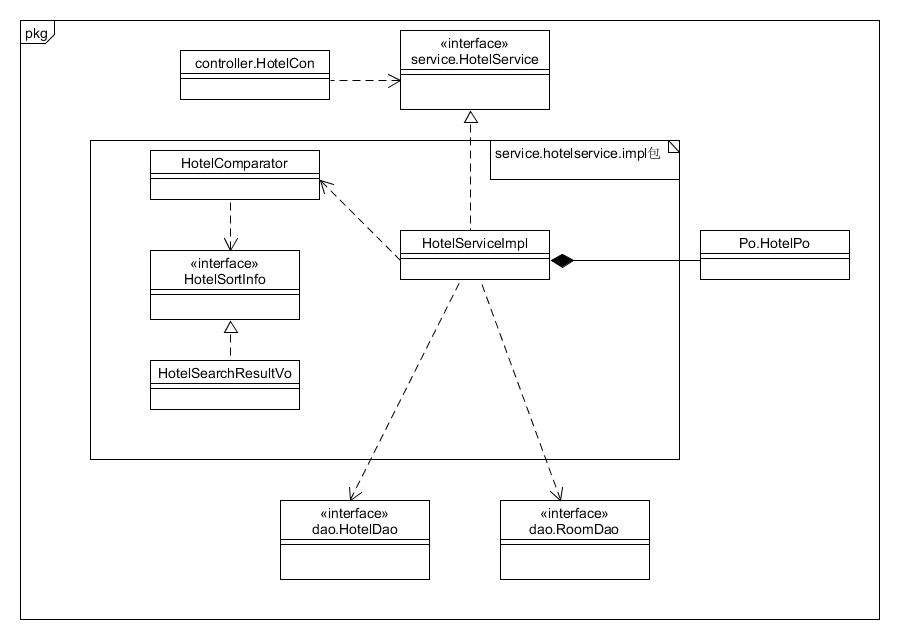
（1）模块概述

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

（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 |

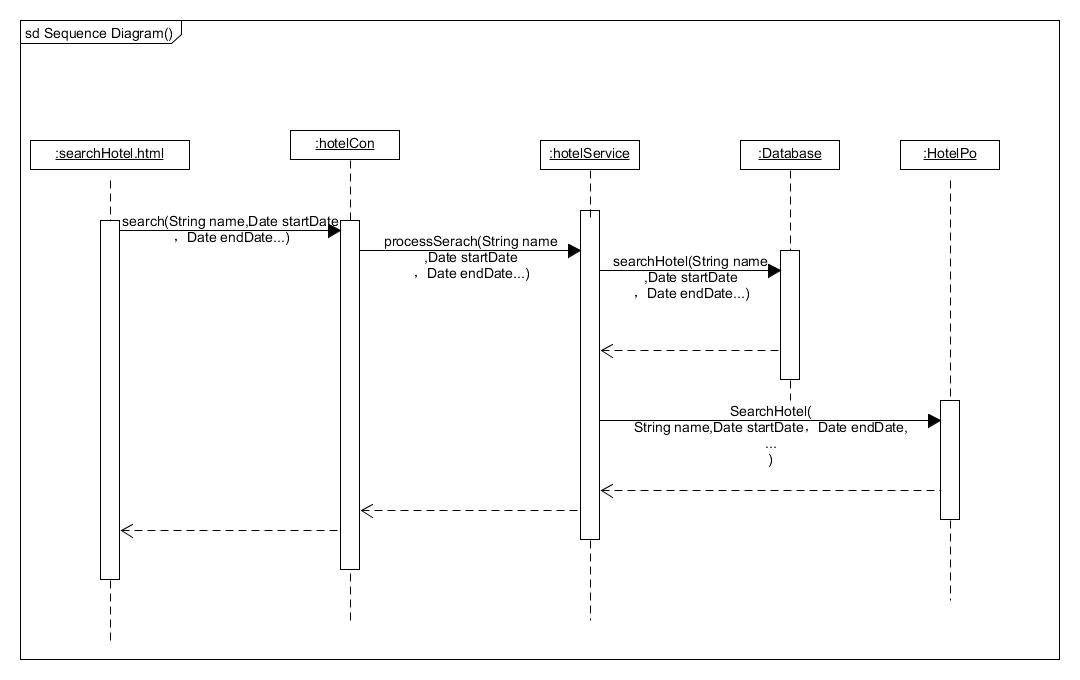
（3）模块内部类的接口规范

HotelServiceImpl类的接口规范

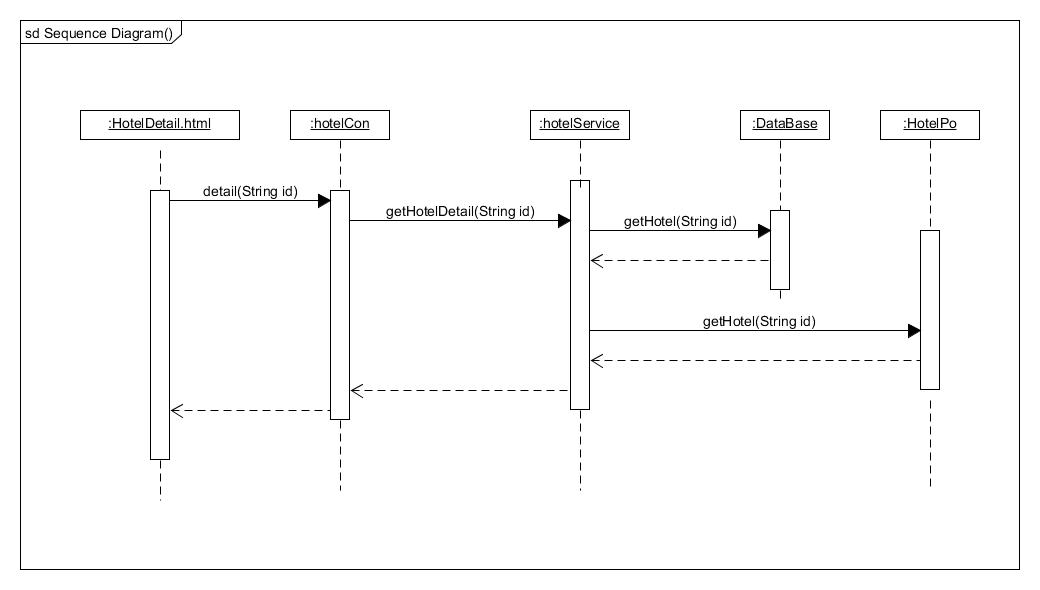
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| HotelServiceImpl.getHotelDetai | 语法 | | public HotelVo getHotel(String hotelId) |
| 前置条件 | | 酒店ID正确且符合规则 |
| 后置条件 | | 根据酒店ID查找酒店，返回其详细信息 |
| HotelServiceImpl.getAllHotels | 语法 | | public List<HotelVo>getAllHotels() |
| 前置条件 | | 点击查询所有酒店 |
| 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

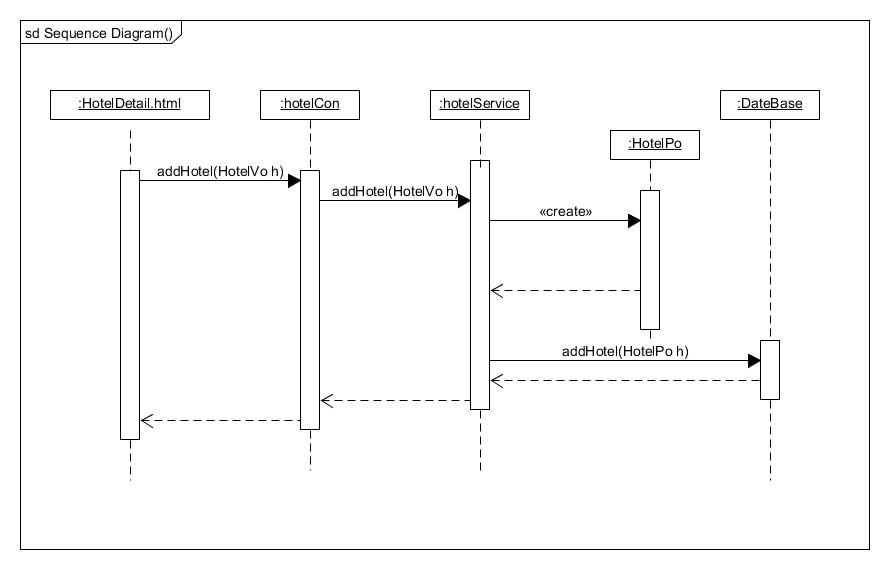
搜索酒店的顺序图



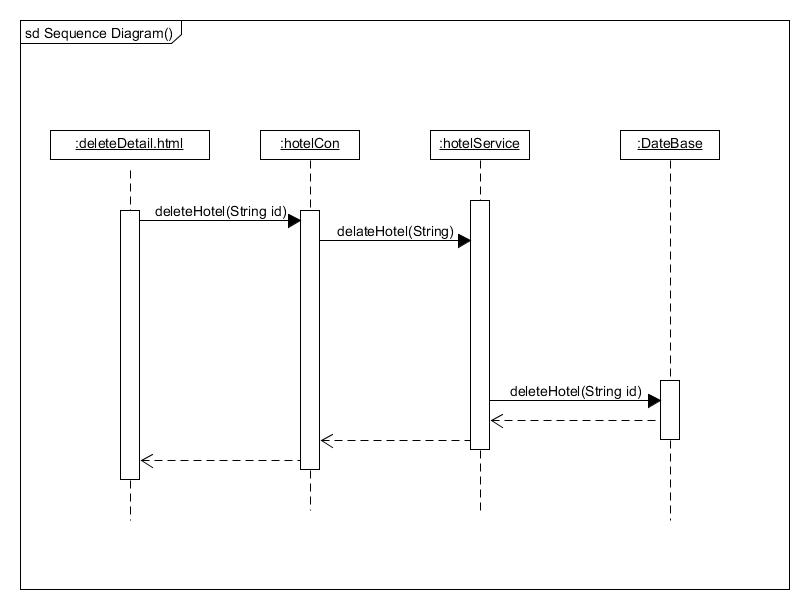
查看酒店细节信息顺序图



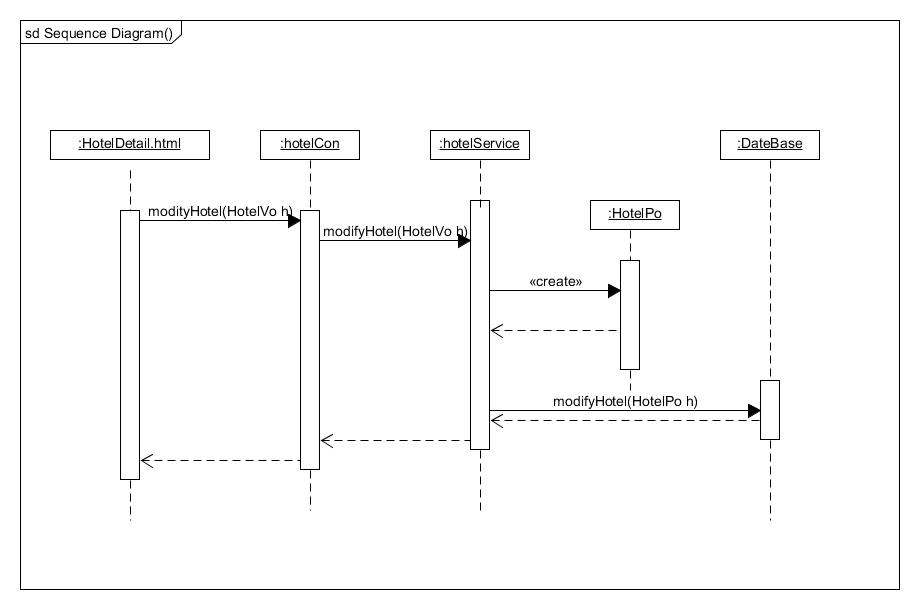
添加酒店顺序图



删除酒店顺序图



修改酒店顺序图



### 4.1.2 Promotion模块

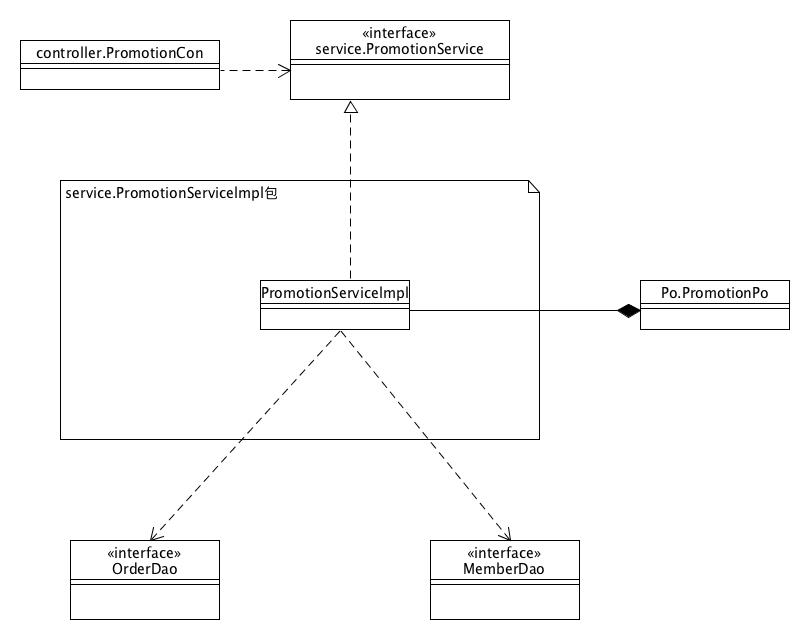
（1）模块概述

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

（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 |

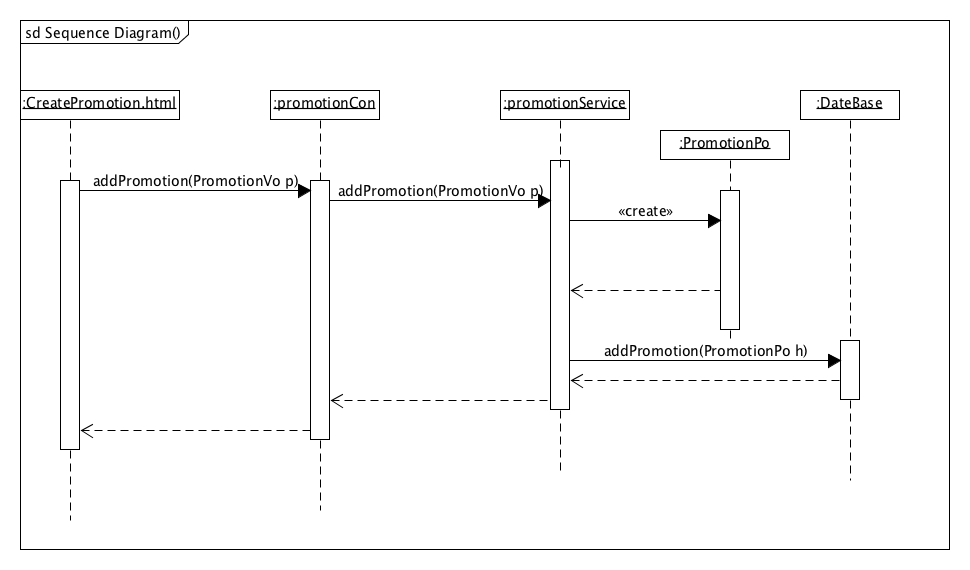
（3）模块内部类的接口规范

PromotionServiceImpl类的接口规范

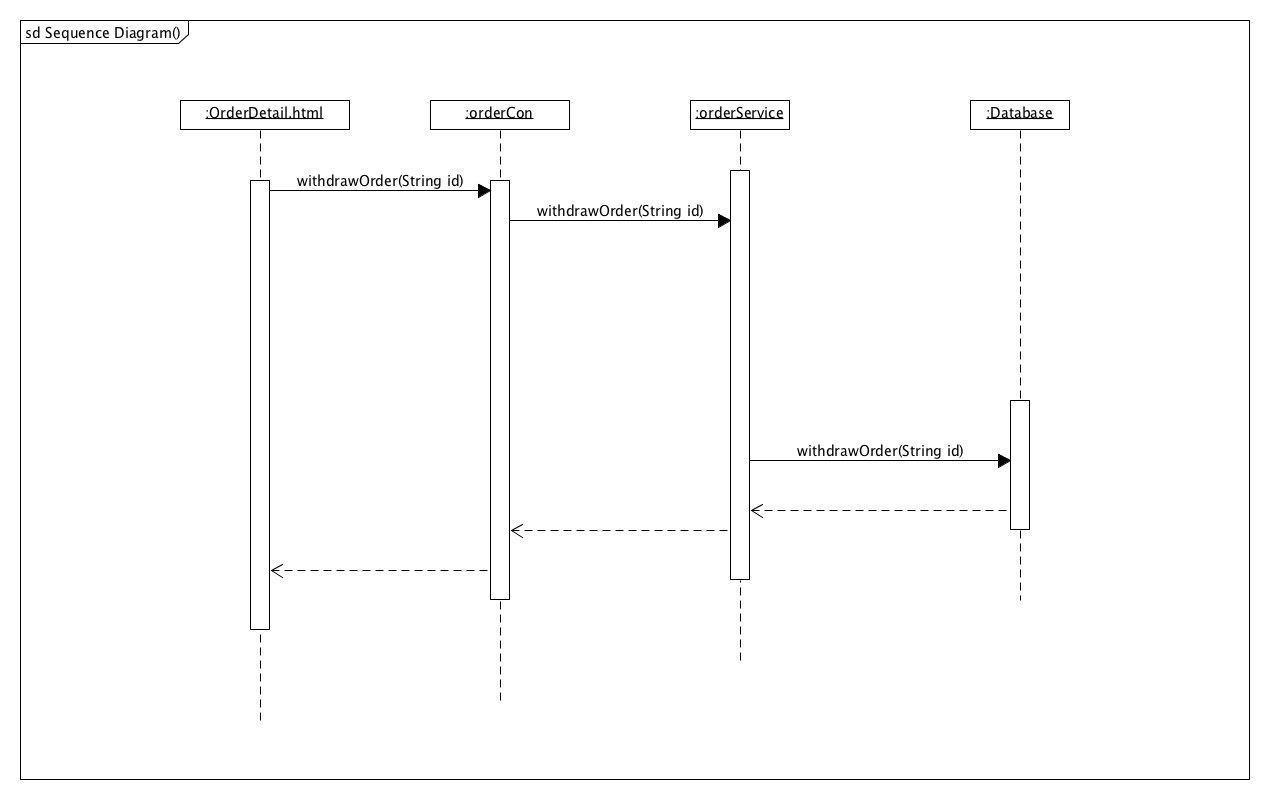
|  |  |  |  |
| --- | --- | --- | --- |
| promotionService.getPromotion | | 语法 | public promotionVo getPromotion (String promotionId) |
| 前置条件 | 促销信息id正确且符合标准 |
| 后置条件 | 根据促销信息id查找促销信息并返回 |
| promotionService.getAllPromotionsByHotel | | 语法 | public List<PromotionVo> getHotelPromotions(String hotelId) |
| 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

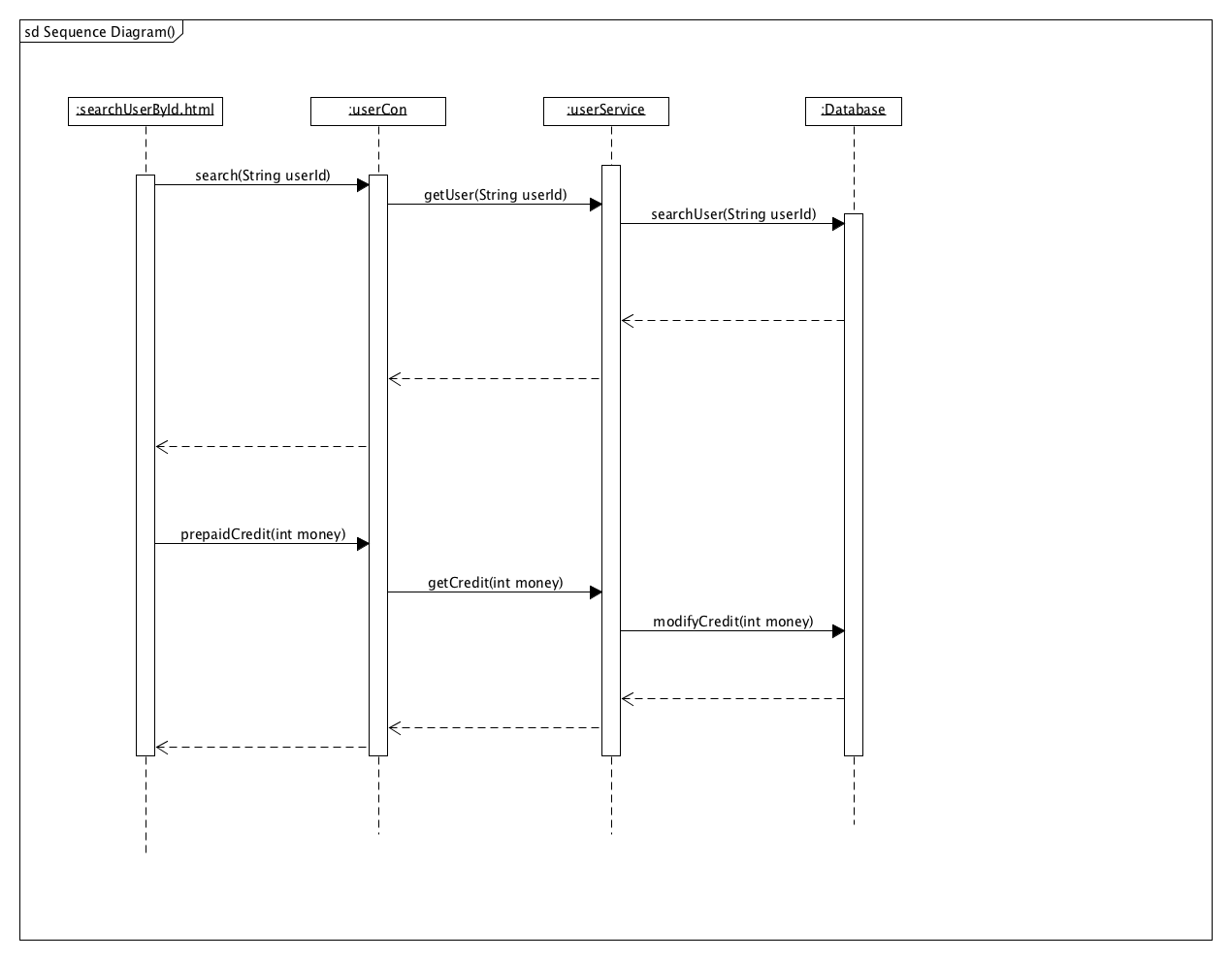
制定促销策略的顺序图



处理异常订单顺序图



信用充值顺序图



# 5. 依赖视角

待补充。