

网上酒店预订系统

详细设计文档 V0.4



2016-10-29

猫尾巴

高翔、曾欢、黄韵斐、黄泽鹏

**更新历史**

|  |  |  |  |
| --- | --- | --- | --- |
| 修改人员 | 日期 | 变更原因 | 版本号 |
| 高翔 | 2016-10-29 | 建立整体框架，完成部分文档 | V0.1 |
| 黄泽鹏 | 2016-10-29 | hotelui模块文档 | V0.2 |
| 曾欢 | 2016-10-29 | data模块完成、网站管理人员模块完成 | V0.3 |
| 黄韵斐 | 2016-10-29 | 完成promotionbl模块 | V0.4 |

目录

[1 引言 1](#_Toc465542408)

[**1.1编制目的** 1](#_Toc465542409)

[**1.2词汇表** 1](#_Toc465542410)

[**1.3参考资料** 2](#_Toc465542411)

[2 产品概述 2](#_Toc465542412)

[3 体系结构设计概述 2](#_Toc465542413)

[4 结构视角 2](#_Toc465542414)

[**4.1 业务逻辑层的分解** 2](#_Toc465542415)

[**4.1.1 loginbl模块** 2](#_Toc465542416)

[**4.1.2 NetManager模块** 7](#_Toc465542417)

[**4.1.3** promotionbl模块 10](#_Toc465542418)

[**4.1.4 userBl模块** 15](#_Toc465542419)

[4.2 展示层的分解 20](#_Toc465542420)

[4.2.1 hotelui模块 20](#_Toc465542421)

[4.3 数据层的分解 27](#_Toc465542422)

[5. 依赖视角 34](#_Toc465542423)

# 1 引言

**1****.1编制目的**

本报告详细完成对网上酒店预订系统的概要设计，达到指导详细设计和开放的目的，同时实现和测试人员及用户的沟通

**1.****2词汇表**

|  |  |
| --- | --- |
| 缩写或单词 | 解释 |
| OHRS | 网上酒店预订系统 |
| Od | 订单 |
| FillOd | 成功订单 |
| FailOd | 异常订单 |
| IngOd | 正在进行订单 |
| CancleOd | 撤销订单 |
| Credit | 信用值 |
| Hotel | 酒店 |
| Market | 网站营销 |
| Control | 管理员 |
| Infor | 信息 |
| Impl | 接口 |
| Prom | 促销政策 |

**1.****3参考资料**

1. IEEE Std 1016-1998 《IEEE Recommended Practice for Software Design Descriptions》

2.《网上酒店预定系统需求规格说明书》

3.丁二玉，刘钦.计算与软件工程

# 2 产品概述

参考网上酒店预订系统用例文档中对产品的概括性描述。

# 3 体系结构设计概述

参考网上酒店预订系统体系文档中关于体系结构设计的概述

# 4 结构视角

**4.1 业务逻辑层的分解**

**4.1.1 loginbl模块**

（1）模块概述

Loginbl模块承担了客户、酒店工作人员、网站营销人员的登陆需求（非功能性）。

Loginbl 模块的职责是区分用户登录信息，其接口参见软件系统结构描述文档表。

(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，我们添加了接口。比如展示层与业务逻辑层之间，我们添加了businessLogicServer.logicServe接口。业务逻辑层与业务数据层之间添加了businessLogicServe.dataServe接口。为了隔离业务逻辑与业务控制职责，我们增加了logicController，这样loginController将会将用户登录验证的业务逻辑职责和逻辑控制委托给loginBl对象。UserVO是做为用户登录信息的持久化对象被添加到设计模型中的。

**loginbl模块的设计如图1所示**

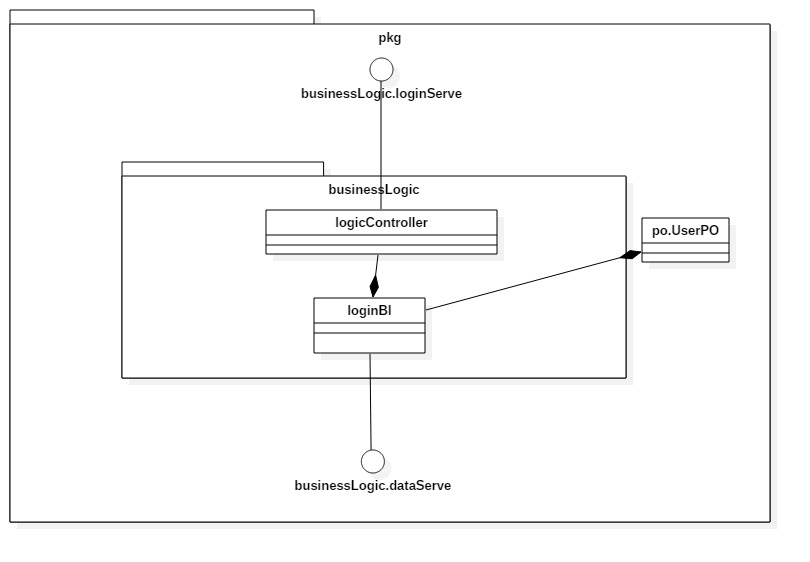


图 1 loginbl模块各个类的设计

loginbl模块中各个类的职责如表所示

**loginbl模块各个类的职责**

|  |  |
| --- | --- |
| 模块名 | 职责 |
| LoginController | 负责实现对应于登录界面所需要的服务 |
| LoginBL | 系统登录的业务逻辑对象 |

1. 模块内部类的接口规范

LoginController和LoginBL的接口规范如表所示。

**LoginController的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| LoginController.login | 语法 | public boolean login(String name,String password); |
| 前置条件 | 账号、密码符合输入规则 |
| 后置条件 | 根据已有信息创建user对象 |
| LoginController.getRole | 语法 | public String getRole(user VO) |
| 前置条件 | 对象初始化成功 |
| 后置条件 | 把user信息更新到loginbl中 |
| LoginController.removeCurrentUser | 语法 | Public void removeCurrentUser(String id) |
| 前置条件 | 用户要求退出系统 |
| 后置条件 | 系统在服务端清除用户登录信息 |
| LoginController.record | 语法 | public void record(UserVO vo) |
| 前置条件 | 用户成功登录 |
| 后置条件 | 记录当前用户到本地文档内 |
| 需要的服务（需接口） | | |
| 服务名 | **服务** | |
| LoginBL.login | 对user基本信息初始化 | |
| LoginBL.getRole | 对user资格进行核对，并确定用户身份 | |
| LoginBL.removeCurrentUser | 把当前用户设置为空 | |
| LoginBL.record | 记录用户信息到本地系统内 | |

**LoginBL的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| LoginBL.login | 语法 | public boolean login(UserVO vo); |
| 前置条件 | 账号、密码符合输入规则 |
| 后置条件 | 根据已有信息创建user对象 |
| LoginBL.getRole | 语法 | public String getRole() |
| 前置条件 | 对象初始化成功 |
| 后置条件 | 返回密码验证结果，并获得用户类型信息 |
| LoginBL.removeCurrentUser | 语法 | Public void removeCurrentUser(String id) |
| 前置条件 | 用户要求退出系统 |
| 后置条件 | 系统在服务端清除用户登录信息 |
| LoginBL.record | 语法 | public void record(UserVO vo) |
| 前置条件 | 用户成功登录 |
| 后置条件 | 记录当前用户到本地文档内 |
| 需要的服务（需接口） | | |
| 服务名 | **服务** | |
| IdentityDataService.getAll() | 对user基本信息初始化 | |
| IdentityDataService.record（String ID） | 对user资格进行核对，并确定用户身份 | |
| IdentityDataService.getCurrentLoginUsers() | 把用户信息记录到本地系统内 | |
| IdentityDataService.updateLoginUsers(ArrayList<String> string) | 把账号退出信息同步到服务端内 | |

1. 业务逻辑层的动态模型

图2表明了在网上酒店预订系统中，当用户登录时，登录业务逻辑处理的相关对象之间的协作。

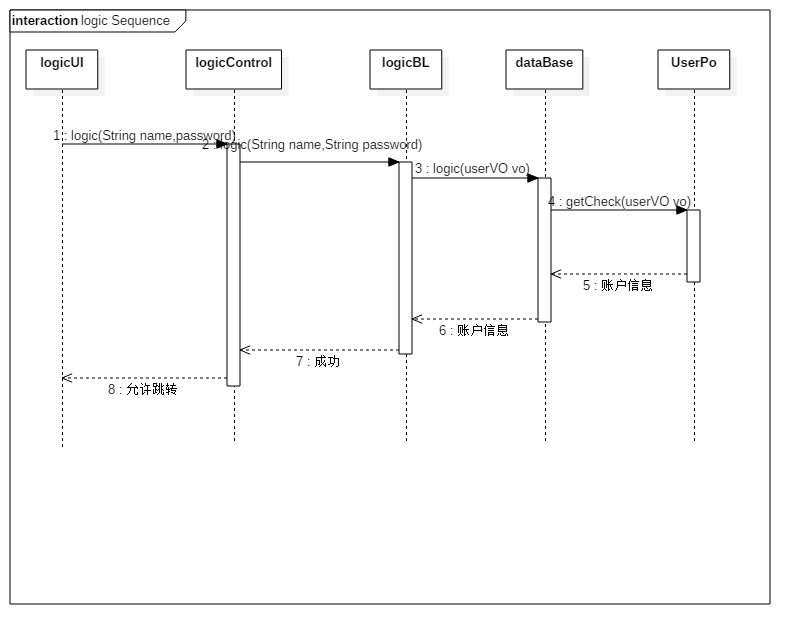


图2 用户登录的顺序图

图3 表明了在网上酒店预订系统中，当用户注销账户时，登录业务逻辑处理的相关对象之间的协作。

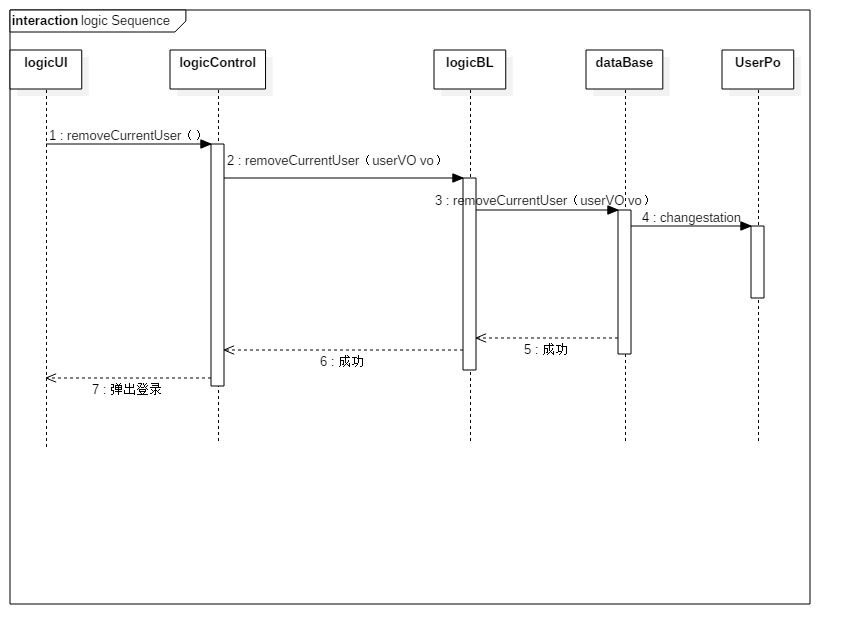


图3：用户注销的顺序图

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。

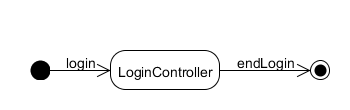


图4：login对象状态图

4.1.2 NetManager模块

（1）模块概述

NetManager模块承担了网站管理人员的登入需求NetManager模块的职责是

网站管理人员对客户，酒店，网站营销人员信息的管理

（2）整体结构由系统由ServiceRunner启动，serviceImp提供数据支持和操作

设计图

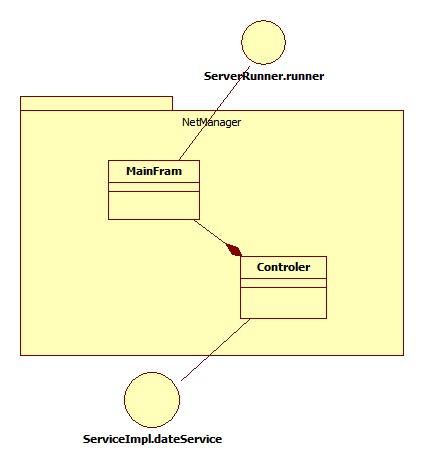


图1 NetManager的设计图

Netmanager里每个类的职责

|  |  |
| --- | --- |
| 模块 | 功能 |
| MainFrame | 界面的显示与布局 |
| Controler | 控制界面的跳转 |

1. 接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| MainFrame.start | 语法 | Public void start() |
| 前置条件 | 系统启动 |
| 后置条件 | 系统正常启动 |
| controler.jumb | 语法 | Public void jumb(int type) |
| 前置条件 | 执行功能 |
| 后置条件 | 跳转界面 |
| Controler.back | 语法 | Public void back(int type) |
| 前置条件 | 执行结束 |
| 后置条件 | 返回上一级界面 |
| Controler.update | 语法 | Public void update（int id) |
| 前置条件 | 修改信息 |
| 后置条件 | 保存并返回上一级 |
| Controler.search | 语法 | Public Information search（int id) |
| 前置条件 | 搜索用户 |
| 后置条件 | 跳转至查看界面或提示用户不存在 |

1. 动态模型

图2表明了网站管理人员查看顾客信息时类之间的合作

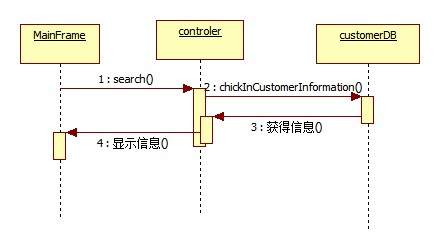


图2 查看客户信息的顺序图

1. Netmanager的设计原理图

采用的是委托式控制的风格。将不同的事件处理委托给该领域的对象

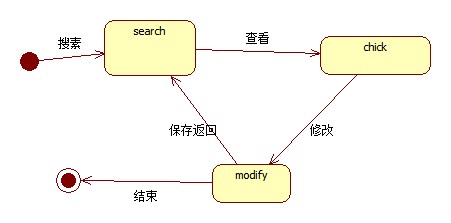


图3NetManager的状态图

**4.1.3** promotionbl模块

（1）模块概述

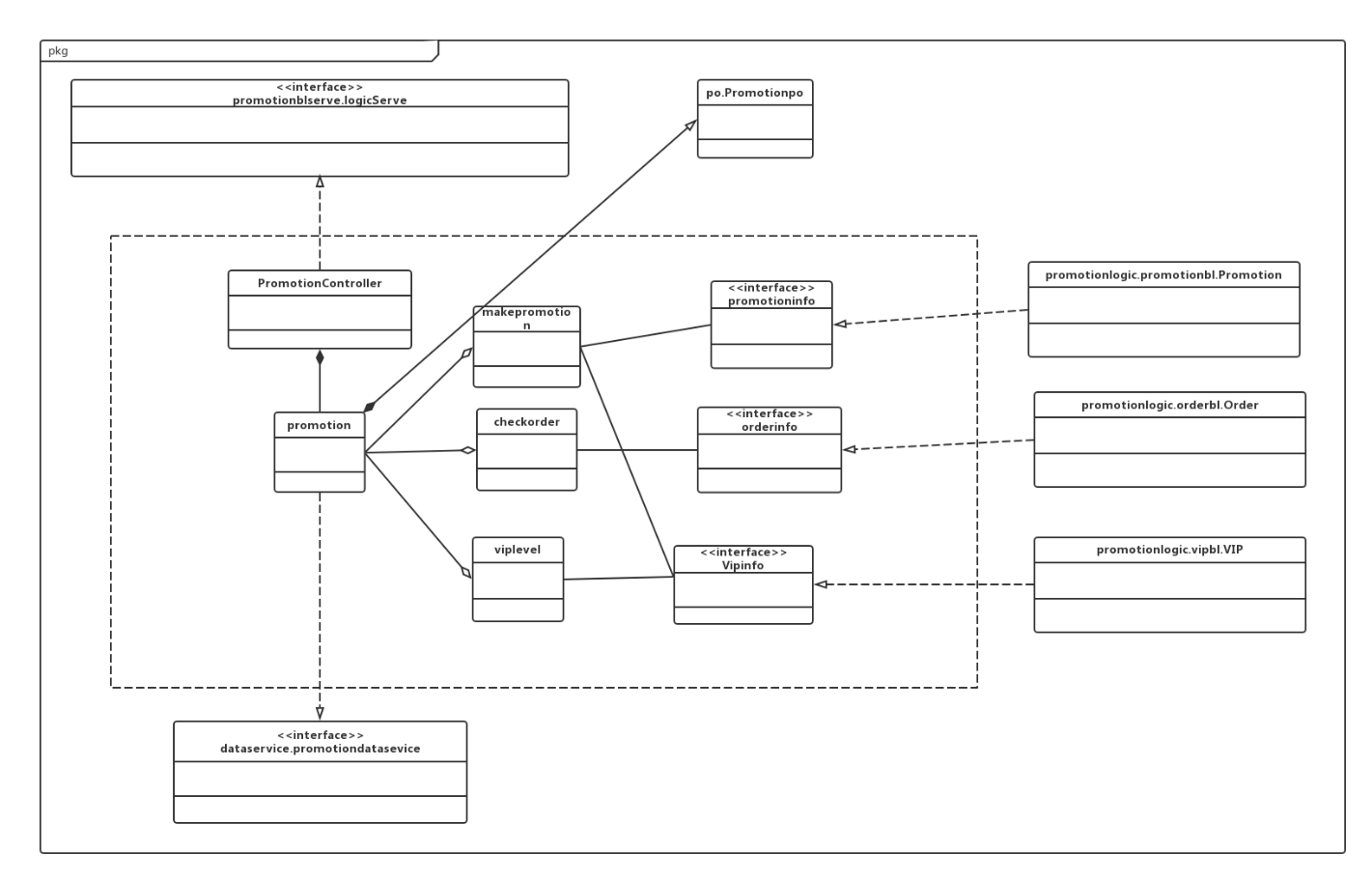
promotionbl模块承担的需求参见需求规格说明功能文档需求及相关非功能需求。

promotionbl模块的职责是制定网站促销政策，其接口参见软件系统结构描述文档表。

（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，我们添加了接口。比如展示层与业务逻辑层之间，我们添加了promotionlogicserver.logicServe接口。业务逻辑层与业务数据层之间添加了promotionlogicserve.dataServe接口。为了隔离业务逻辑与业务控制职责，我们增加了promotionController，这样promotionController将会将制定促销策略的业务逻辑职责和逻辑控制委托给promotionBl对象。promotionPO是作为制定促销策略的持久化对象被添加到设计模型中的。Promotioninfo和orderinfo和vipinfo都是根据依赖倒置原则，为了消除循环依赖产生的接口

Promotionbl模块的设计如图所示：



（3）promotionbl模块中各个类的职责如表所示

**promotionbl模块各个类的职责**

|  |  |
| --- | --- |
| 模块 | 职责 |
| PromotionController | 负责实现对应于制定促销政策界面所需要的服务 |
| Promotion | 促销政策的领域模型对象 |
| Makepromotion | 制定促销政策的领域模型对象，拥有促销策略数据 |
| Checkorder | 查看异常订单的领域模型对象，拥有异常订单信息，可以帮助完成查看异常订单所需要的服务 |
| Viplevel | VIP等级制定的领域模型对象，拥有VIP信息，可以帮助完成制定VIP等级制度所需要的服务。 |

PromotionController的接口规范

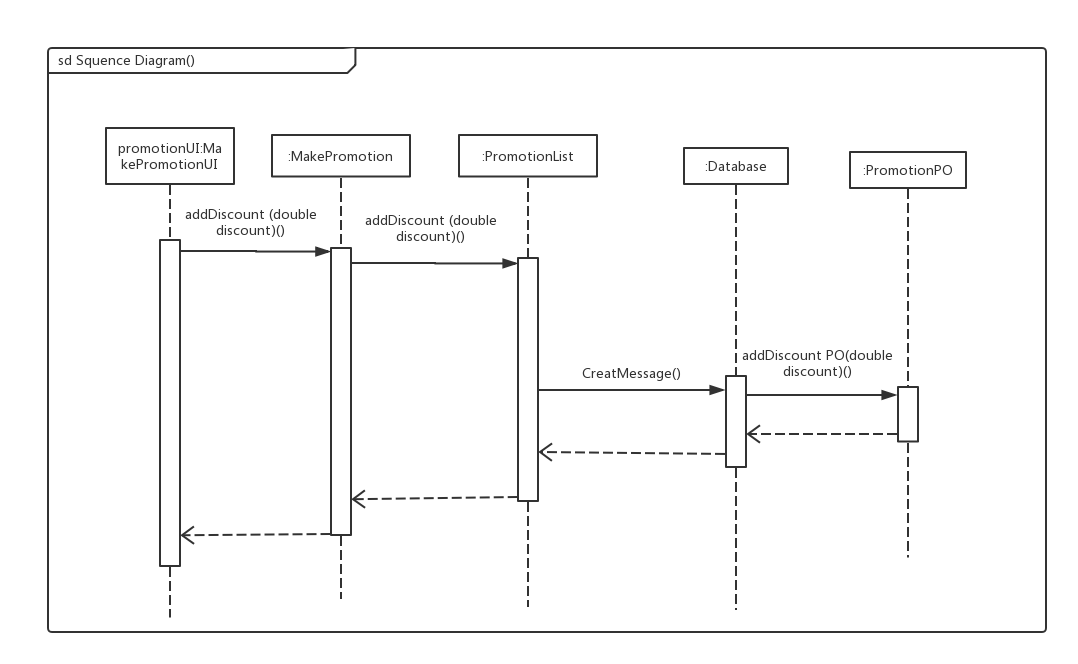
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PromotionController.addDiscount | 语法 | public void addDiscount (double discount) |
| 前置条件 | 已创建出一个MakePromotion领域对象，并且输入折扣符合规则 |
| 后置条件 | 调用MakePromotion领域对象的addDiscount方法 |
| PromotionController.addTime | 语法 | public void addTime(double starttime,double endtime) |
| 前置条件 | 已创建出一个MakePromotion领域对象，输入开始结束时间符合规则 |
| 后置条件 | 调用MakePromotion领域对象的addTime方法 |
| PromotionController.addVIPpromotion | 语法 | public void addVIPpromotion(Shangquan SQ,int level,double discount) |
| 前置条件 | 已创建出一个MakePromotion领域对象，并且输入商圈和会员等级符合规则 |
| 后置条件 | 调用MakePromotion领域对象的addVIPpromotion方法 |
| PromotionController.addPriceMinus | 语法 | Public void addPriceMinus（double price，double minus） |
| 前置条件 | 已创建出一个MakePromotion领域对象，并且输入的满减符合规则 |
| 后置条件 | 调用MakePromotion领域对象的addPriceMinus方法 |
| PromotionController.end | 语法 | Public void end() |
| 前置条件 | 已创建出一个MakePromotion领域对象 |
| 后置条件 | 调用MakePromotion领域对象的end方法 |
| 需要的服务（需接口） | | |
| 服务名 | **服务** | |
| Makepromotion.addDiscount(double discount) | 加入新的折扣 | |
| Makepromotion.addTime(double starttime,double endtime) | 加入新的节假日折扣起始时间 | |
| MakePromotion.addVIPpromotion Shangquan SQ,int level,double discount) | 加入新的VIP在特定商圈的优惠政策 | |
| Makepromotion.addPriceMinus double price，double minus） | 加入新的满减促销政策 | |

MakePromotion的接口规范

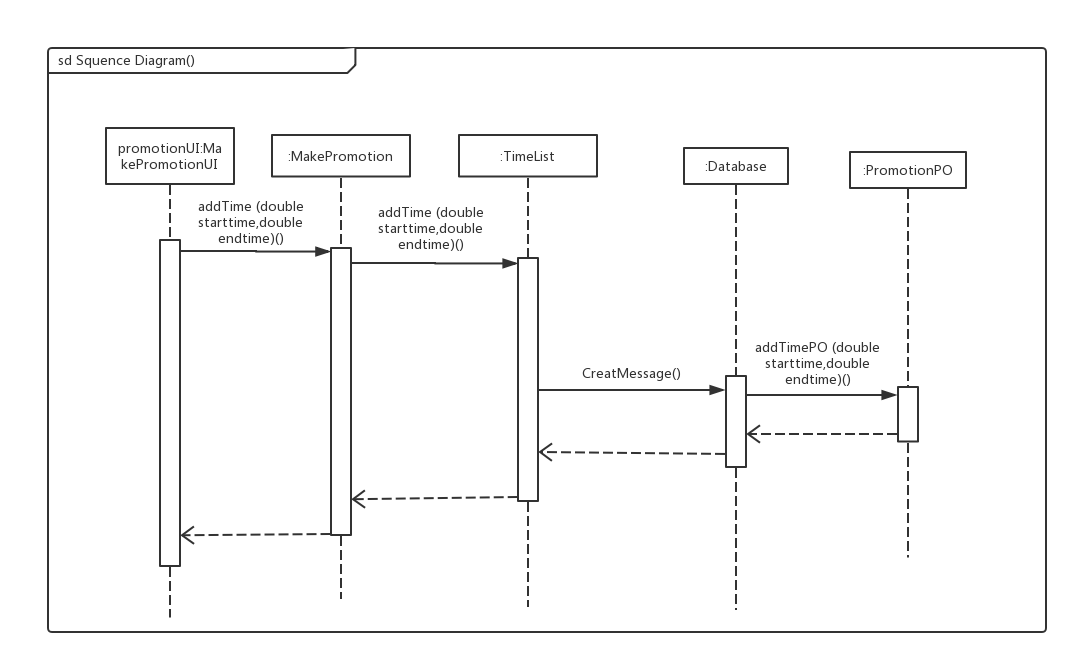
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Makepromotion.addDiscount | 语法 | public void addDiscount (double discount) |
| 前置条件 | 网站营销人员输入打折比例 |
| 后置条件 | 根据打折比例计算出价格 |
| Makepromotion.addTime | 语法 | public void addTime(double starttime,double endtime) |
| 前置条件 | 网站营销人员点击输入制定节日促销政策 |
| 后置条件 | 将折扣的起始时间记录 |
| MakePromotion.addVIPpromotion | 语法 | public void addVIPpromotion(Shangquan SQ,int level,double discount) |
| 前置条件 | 网站营销人员制定VIP在特定商圈的优惠政策 |
| 后置条件 | 将VIP在特定商圈的优惠政策记录 |
| Makepromotion.addPriceMinus | 语法 | Public void addPriceMinus（double price，double minus） |
| 前置条件 | 网站营销人员输入满减促销政策 |
| 后置条件 | 将满减促销政策记录 |
| MakePromotion.end | 语法 | public void end() |
| 前置条件 | 完成促销政策制定 |
| 后置条件 | 结束此次制定 |
| 需要的服务（需接口） | | |
| 服务名 | **服务** | |
| PromotionDataService.add(double discount) | 增加新的折扣记录 | |
| PromotionDataService.delete(double discount) | 删除原有的折扣记录 | |
| PromotionDataService.update | 将折扣更新并记录 | |
| PromotionDataService. getPromotionDatabase | 得到促销政策数据库的服务的引用 | |

1. 业务逻辑层的动态模型

图2表明了在网上酒店预订系统中，促销人员制定促销政策是，促销政策业务逻辑处理的相关对象之间的协作。



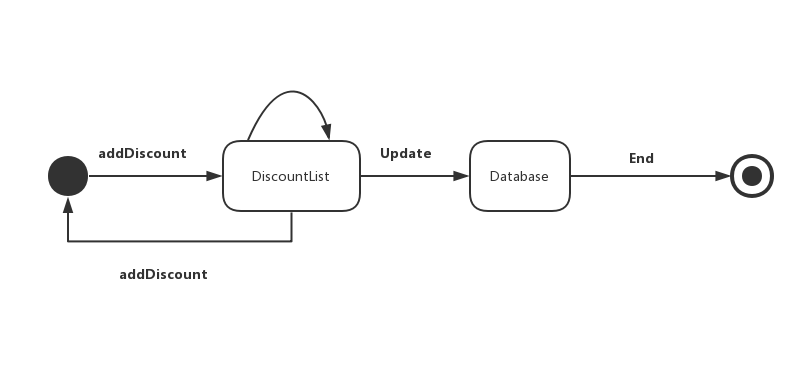
添加促销政策顺序图



添加节假日促销起始时间顺序图

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。



**4.1.4 userBl模块**

（1）模块概述

userBl模块承担了客户需求有关的功能：查看基本信息、帐单管理、查看酒店、订单生成等功能性需求。

useBl 模块的职责是完成用户的所有操作的逻辑处理，接口详见系统体系结构文档。

(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，我们添加了接口。比如展示层与业务逻辑层之间，我们添加了userBl.userServer.userblserver接口。业务逻辑层与业务数据层之间添加了userBl.userServe.userDataserver接口。为了隔离业务逻辑与业务控制职责，我们增加了userController，这样userController将会将用户功能的业务逻辑职责和逻辑控制委托给users对象。UserPO是做为用户个人信息、订单信息的持久化对象被添加到设计模型中的。而userInformation 、userHotal与 userOd的添加是userInfo的容器类。userInformation保有用户的基本信息；userOd封装了该用户的订单信息，userHotal封装了用户所检索的目的地酒店信息。

**userbl模块的设计如图1所示**

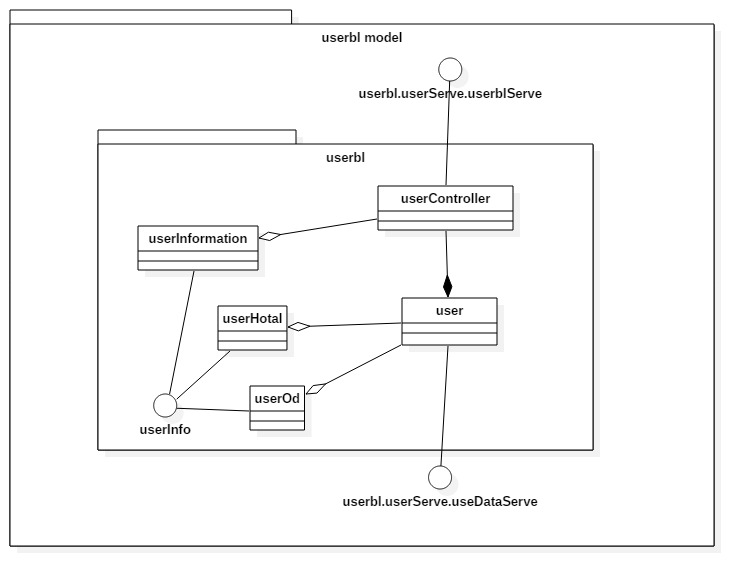


图 1 userbl模块各个类的设计

userbl模块中各个类的职责如表所示

**loginbl模块各个类的职责**

|  |  |
| --- | --- |
| 模块名 | 职责 |
| userController | 负责实现对应于登录界面所需要的服务 |
| LoginBL | 系统登录的业务逻辑对象 |

1. 模块内部类的接口规范

LoginController和LoginBL的接口规范如表所示。

**LoginBL的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| LoginBL.login | 语法 | public boolean login(UserVO vo); |
| 前置条件 | 账号、密码符合输入规则 |
| 后置条件 | 根据已有信息创建user对象 |
| LoginBL.getRole | 语法 | public String getRole() |
| 前置条件 | 对象初始化成功 |
| 后置条件 | 返回密码验证结果，并获得用户类型信息 |
| LoginBL.removeCurrentUser | 语法 | Public void removeCurrentUser(String id) |
| 前置条件 | 用户要求退出系统 |
| 后置条件 | 系统在服务端清除用户登录信息 |
| LoginBL.record | 语法 | public void record(UserVO vo) |
| 前置条件 | 用户成功登录 |
| 后置条件 | 记录当前用户到本地文档内 |
| 需要的服务（需接口） | | |
| 服务名 | **服务** | |
| IdentityDataService.getAll() | 对user基本信息初始化 | |
| IdentityDataService.record（String ID） | 对user资格进行核对，并确定用户身份 | |
| IdentityDataService.getCurrentLoginUsers() | 把用户信息记录到本地系统内 | |
| IdentityDataService.updateLoginUsers(ArrayList<String> string) | 把账号退出信息同步到服务端内 | |

1. 业务逻辑层的动态模型

图2表明了在网上酒店预订系统中，当用户登录时，登录业务逻辑处理的相关对象之间的协作。

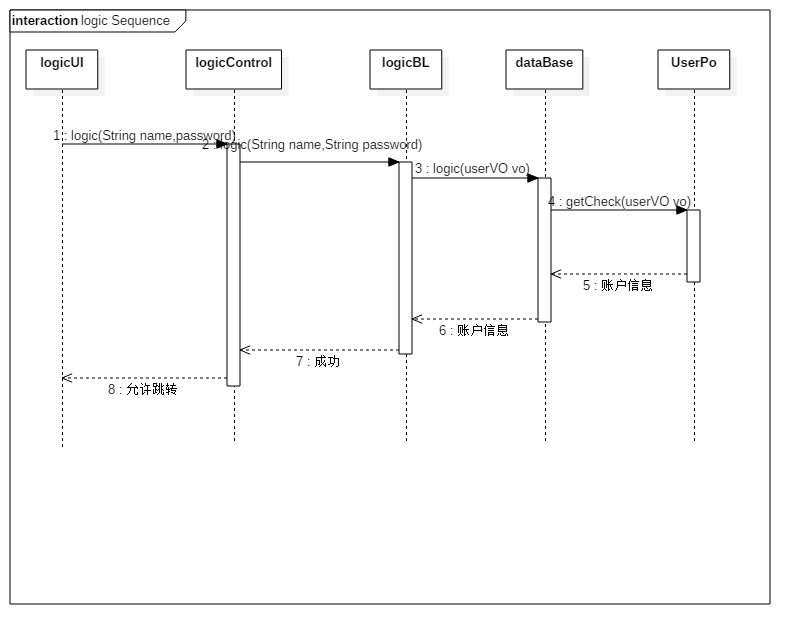


图2 用户登录的顺序图

图3 表明了在网上酒店预订系统中，当用户注销账户时，登录业务逻辑处理的相关对象之间的协作。

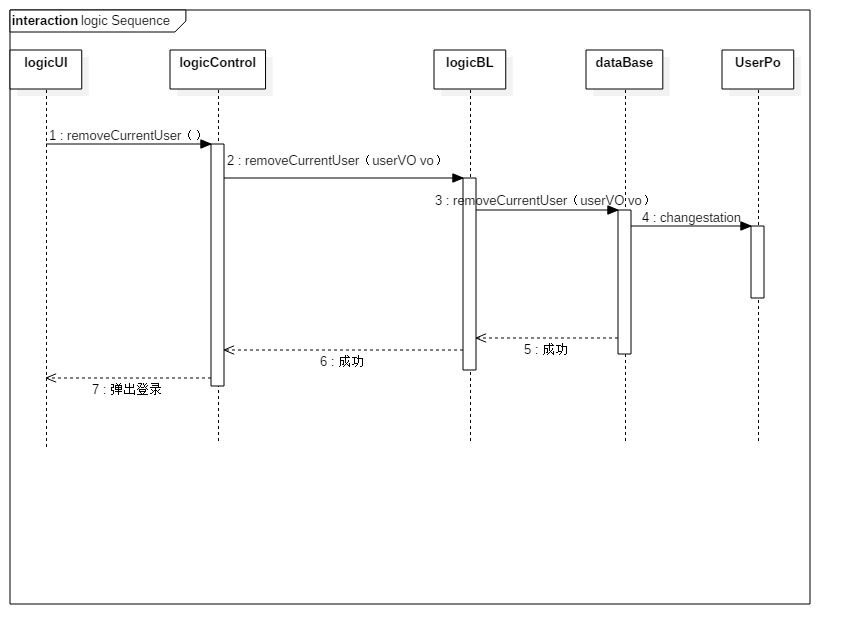


图3：用户注销的顺序图

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。

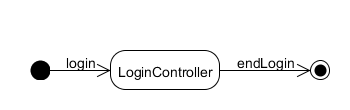


图4：login对象状态图

## 4.2 展示层的分解

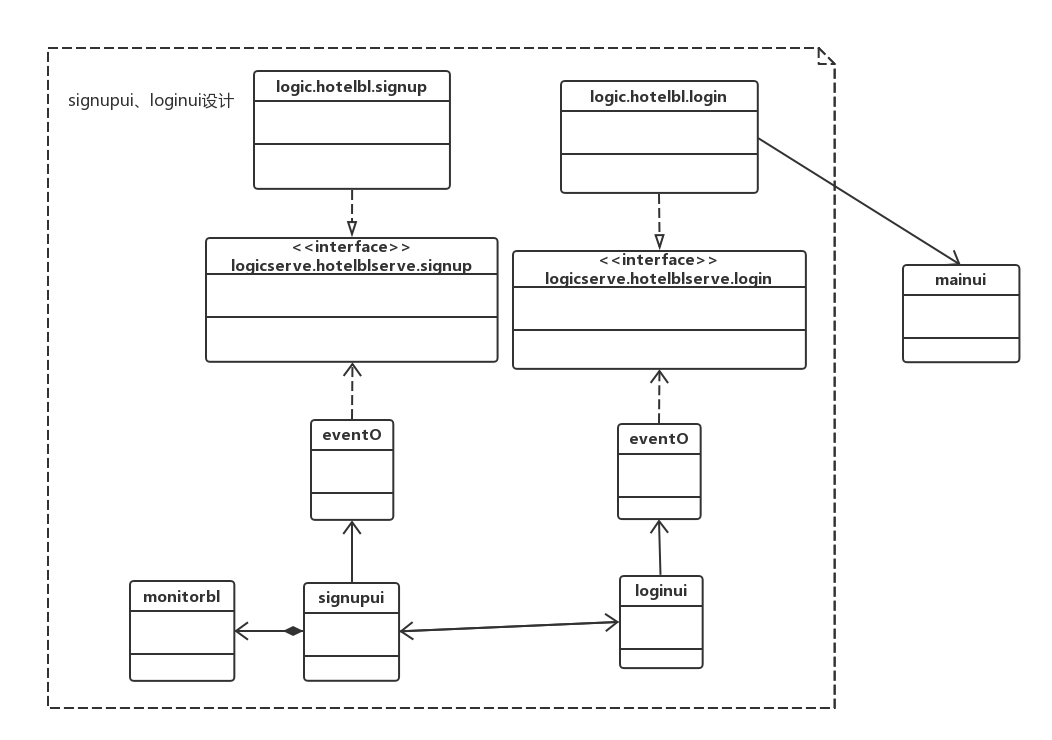
### 4.2.1 hotelui模块

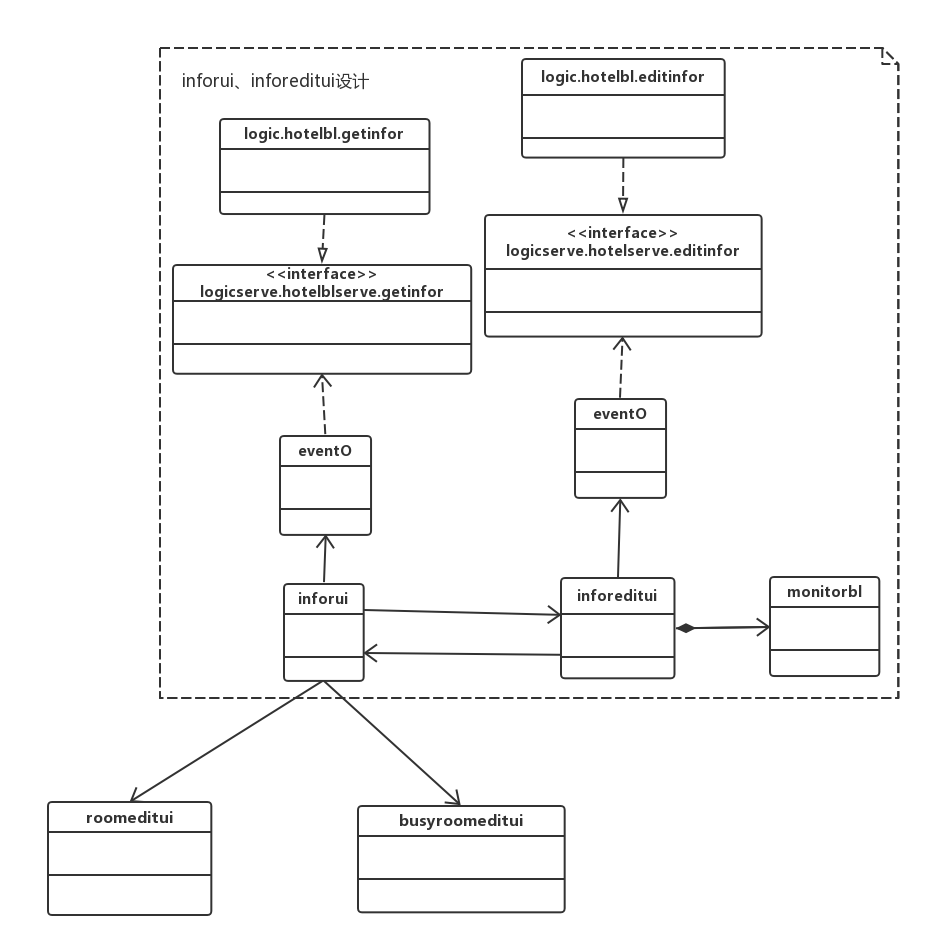
#### （1）模块概述

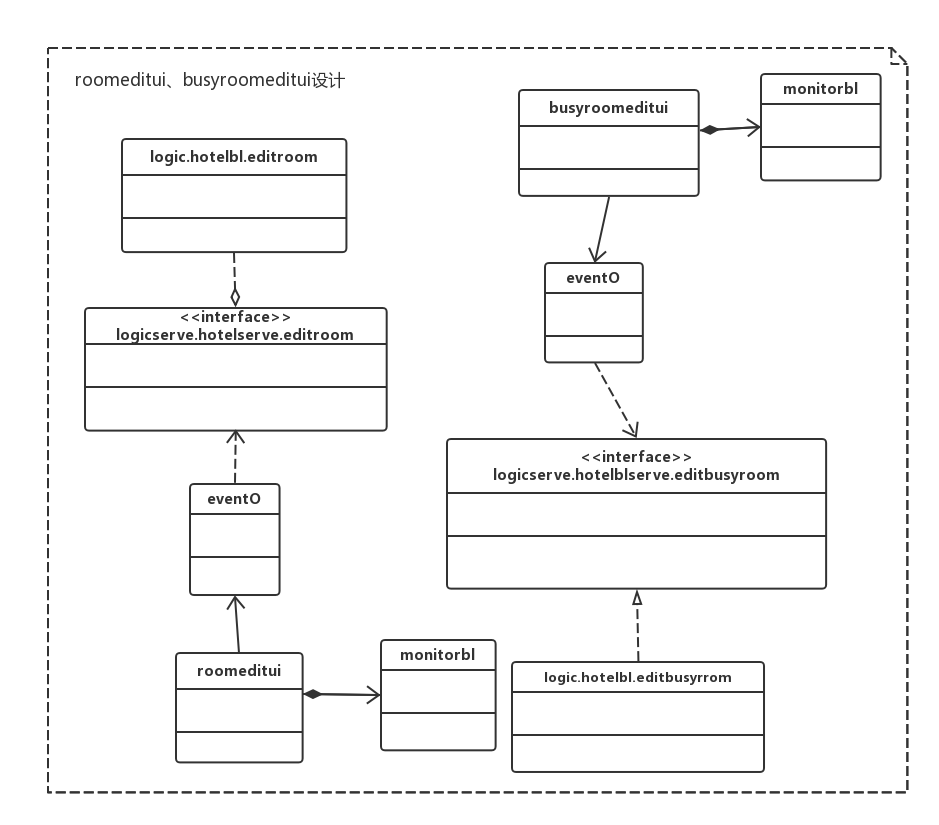
hotelui模块承担的需求参见用例文档和需求模型功能需求及相关非公能需求。

hotelui模块的职责及接口参见体系结构设计模型与文档。

#### （2）整体结构

hotelui模块的设计图





hotelui模块各个类的职责如下所示

**hotelui模块各个类的职责**

|  |  |
| --- | --- |
| 类名/模块名 | 职责 |
| signupui | 负责酒店注册界面的排版 |
| loginui | 负责酒店登陆界面的排版 |
| inforui | 负责酒店基本信息界面的排版 |
| inforeditui | 负责酒店基本信息编辑界面的排版 |
| roomeditui | 负责酒店客房信息编辑界面的排版 |
| busyroomeditui | 负责酒店入住客房数量修改界面的排版 |
| eventO | 负责接受展示层发起的请求，调用逻辑层实现 |
| monitorbl | 负责监视输入的正确性 |
| searchui | 负责酒店搜索界面的排版 |
| hotellistui | 负责商圈内酒店列表界面的排版 |
| hotelinforui | 负责酒店基本信息界面的排版 |

#### （3）模块内部类的接口规范

**signupui的接口**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| signupui.initui | **语法** | Public void initui() |
| **前置条件** | 用户点击酒店注册 |
| **后置条件** | 调用init()方法初始化酒店注册界面 |
| 需要的服务（需接口） | | |
| 服务名 | **服务** | |
| hotelbl.signup(HotelInfor infor) | 注册酒店信息 | |
| monitorbl.check(HotelInfor infor) | 调用本地输入检测来反馈输入的酒店信息格式的正确性 | |

**loginui的接口**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| loginui.initui | **语法** | Public void initui() |
| **前置条件** | 用户选择“酒店登陆“ |
| **后置条件** | 调用init()方法初始化酒店登陆界面 |
| 需要的服务（需接口） | | |
| 服务名 | **服务** | |
| hotelbl.login(String username, String password) | 酒店工作人员登陆 | |

**inforui的接口**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| inforui.initui | **语法** | Public void initui() |
| **前置条件** | 酒店工作人员成功登陆 |
| **后置条件** | 调用init()方法初始化酒店信息界面 |
| 需要的服务（需接口） | | |
| 服务名 | **服务** | |
| hotelbl.getinfor(String hotalid) | 返回该酒店的基本信息 | |

**inforeditui的接口**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| inforedit.initui | **语法** | Public void initui() |
| **前置条件** | 用户点击酒店信息编辑 |
| **后置条件** | 调用init()方法初始化酒店信息编辑 |
| 需要的服务（需接口） | | |
| 服务名 | **服务** | |
| hotelbl.getinfor(String hotelid) | 返回该酒店的基本信息 | |
| hotelbl.editinfor(HotelInfor info) | 传入修改后的酒店信息，返回一个boolean表示修改是否成功 | |
| monitorbl.check(HotelInfor infor) | 调用本地输入检测来反馈输入的酒店信息格式的正确性 | |

**roomeditui的接口**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| roomeditui.initui | **语法** | Public void initui() |
| **前置条件** | 用户点击酒店客房信息编辑 |
| **后置条件** | 调用init()方法初始化酒店客房信息编辑界面 |
| 需要的服务（需接口） | | |
| 服务名 | **服务** | |
| hotelbl.getroominfor(String accredit) | 返回酒店客房信息 | |
| monitorbl.check(int[] num) | 调用本地输入检测来反馈输入的酒店客房各类型数量的正确性 | |

**busyroomeditui的接口**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| busyroomeditui.initui | **语法** | Public void initui() |
| **前置条件** | 用户点击入住客房信息编辑 |
| **后置条件** | 调用init()方法初始化入住客房信息编辑界面 |
| 需要的服务（需接口） | | |
| 服务名 | **服务** | |
| hotelbl.getbusyroominfor(String accredit) | 返回酒店入住客房信息 | |
| monitorbl.check(int change) | 调用本地输入检测来反馈输入的酒店入住客房数量大小的正确性 | |

**monitorbl的接口**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| monitorbl.check | **语法** | Public boolean[] check(HotelInfor infor) |
| **前置条件** | 获取用户输入的酒店信息 |
| **后置条件** | 对用户输入信息进行检查后返回一个boolean数组来表示各项正确性 |
| monitorbl.check | **语法** | Public boolean[] check(int[] num) |
| **前置条件** | 获取用户输入的酒店客房各类型数量 |
| **后置条件** | 对用户输入信息进行检查后返回一个boolean数组来表示各项正确性 |
| monitorbl.check | **语法** | Public boolean check(int change) |
| **前置条件** | 获取用户输入的酒店客房各类型数量 |
| **后置条件** | 对用户输入数量进行检查后返回一个boolean来表示各项正确性 |

**searchui的接口**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| searchui.initui | **语法** | Public void initui() |
| **前置条件** | 用户点击酒店搜索 |
| **后置条件** | 调用init()方法初始化酒店搜索界面 |
| 需要的服务（需接口） | | |
| 服务名 | **服务** | |
| hotelbl.searchGrogshop(String tradearea) | 搜索商圈内酒店 | |

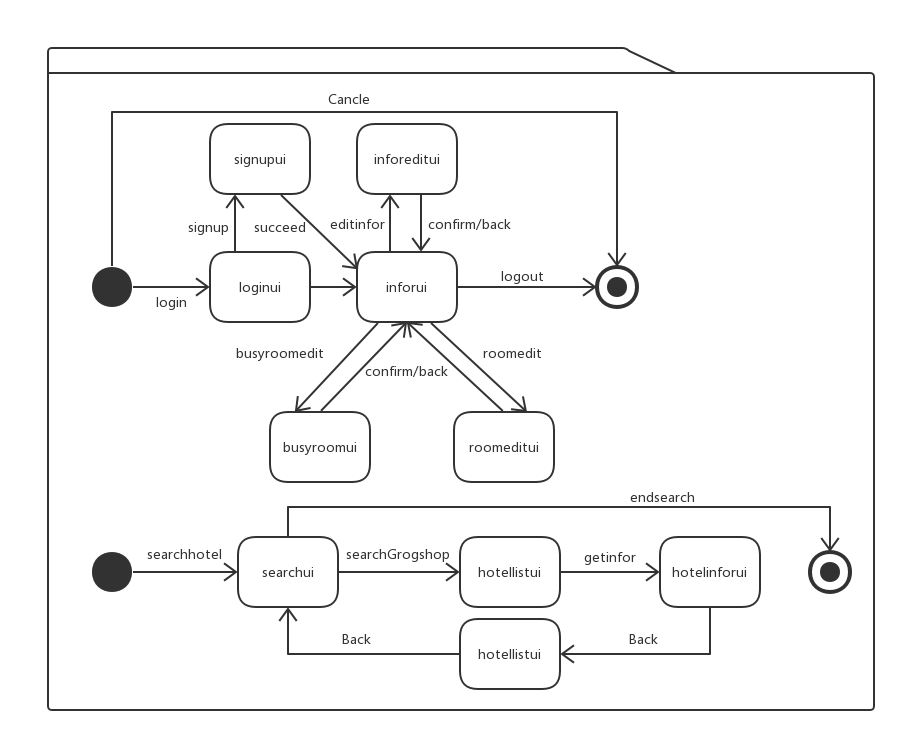
**hotellistui的接口**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| hotellistui.initui | **语法** | Public void initui() |
| **前置条件** | 用户搜索商圈内酒店 |
| **后置条件** | 调用init()方法初始化酒店列表界面 |
| 需要的服务（需接口） | | |
| 服务名 | **服务** | |
| hotelbl.searchGrogshop(String tradearea) | 搜索商圈内酒店 | |

**hotelinforui的接口**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| hotelinforui.initui | **语法** | Public void initui() |
| **前置条件** | 用户选择查看酒店 |
| **后置条件** | 调用init()方法初始化酒店信息界面 |
| 需要的服务（需接口） | | |
| 服务名 | **服务** | |
| hotelbl.getinfor(String hotelid) | 返回酒店信息 | |

#### （4）hotelui模块的动态模型



# 4.3 数据层的分解

4.3.1 DateBase模块

1. 模块概述

DateBase模块承担了系统对数据库的操作。职责是对数据库的管理

1. 整体结构

DBlinke提供了对数据的唯一连接，所有对数据库的连接都必须通过DBllinke获得一个Statement 对象才能对数据库进行操作IDManager 对数据库ID进行管理，避免ID冲突。Register记录已连接并登入的客户端，若没有在Register里记录则不能操作

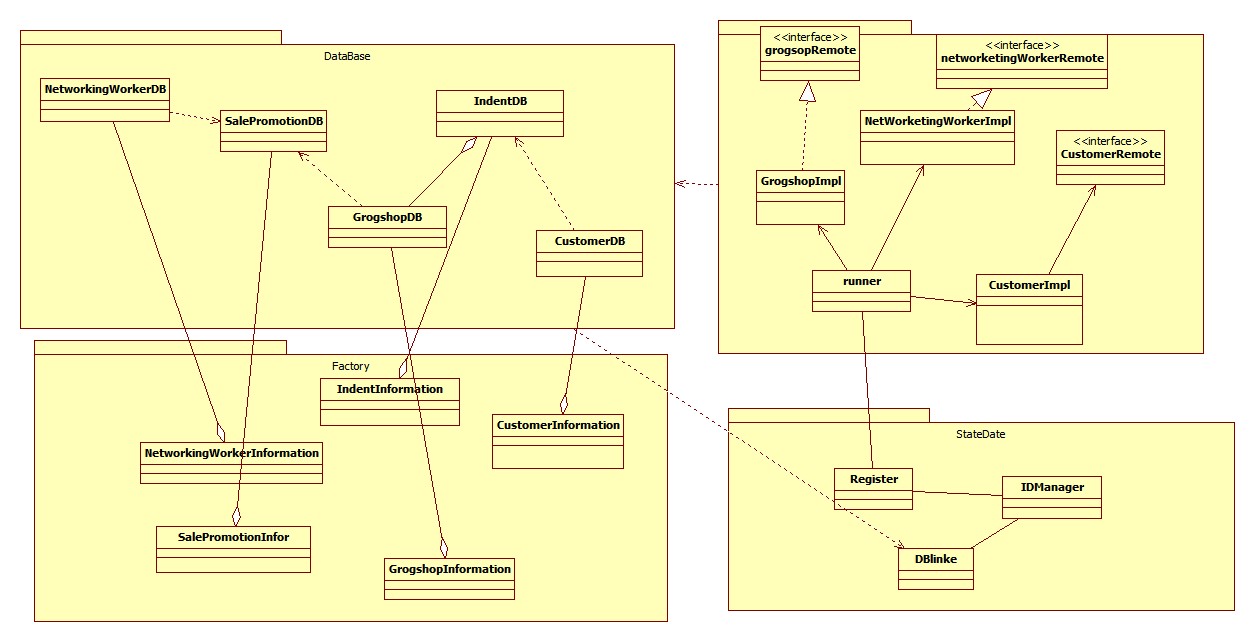


图4 DateBase模块的设计图

（3）每个类的职责

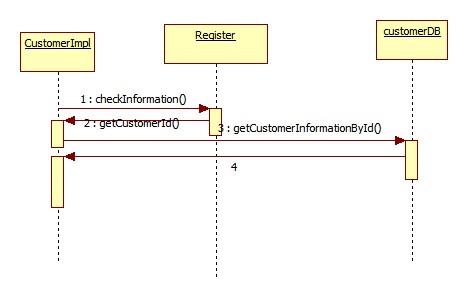
|  |  |
| --- | --- |
| 类 | 职责 |
| CustomerDB | 对客户表的数据库操作与维护 |
| GrogshopDB | 对酒店信息的数据的数据库操作 |
| NetWorkingWorkerDB | 对网站管理人员信息的数据库操作 |
| SalePromotionDB | 对营销策略信息的数据库操作 |
| IndentDB | 对订单信息的数据库操作 |
| GrogshopImpl | 对外的酒店远程调用 |
| CustomerImpl | 对外的客户远程调用 |
| NetworkingWorkerImpl | 对外的网站营销人员远程调用 |
| DBlinke | 对数据库的唯一连接 |
| Register | 记录登入的用户 |
| IDManger | 对数据库账户的管理 |

（4）datebase模块的接口规范

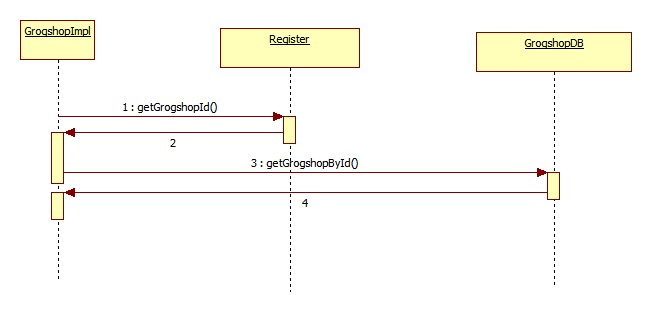
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| customerDB.getCustomerById() | 语法 | **public** CustomerInformation getCustomerById(**int** customerId) |
| 前置条件 | 该Id存在并且调用方登入和授权 |
| 后置条件 | 返回客户信息。 |
| customerDB.createNewCustomer | 语法 | **Public int c**eateNewCustomer (CustomerInformation cusInfor) |
| 前置条件 | 接受用于创建用户且的信息对象此对象的id属性无效 |
| 后置条件 | 创建成功返回一个账号Id值否则返回0 |
| customerDB.updateCustomerInformation | 语法 | **Public boolean** updateCustomerInformation(CustomerInformation cusInfor,**int** customerId) |
| 前置条件 | 要修改的Id存在且不会修改Id，密码，信用值 |
| 后置条件 | 成功修改返回true否则返回false |
| customerDB.modifyPassword | 语法 | Public boolean modifyPassword(int customerId,int password) |
| 前置条件 | Id存在 |
| 后置条件 | 密码修改成功返回true，否则返回false |
| customerDB.delete | 语法 | Public boolean delete(int customerId) |
| 前置条件 | Id存在 |
| 后置条件 | 删除账户 |
| GrogshopDB.getAllGrogshpInAArea | 语法 | Public ArrayList<GrogshopInformation> getAllGrogshopInAArea(String businessArea) |
| 前置条件 | 有需要搜索的商圈的请求 |
| 后置条件 | 返回一个酒店信息列表 |
| GrogshopDB.getGrogshopById | 语法 | Public GrogshopInformation getGrogshopById(int grogshopId |
| 前置条件 | 酒店需要查看酒店信息 |
| 后置条件 | 返回酒店信息 |
| GrogshopDB.insertNewGroghshop | 语法 | Public boolean insertNewGrogshop(GrogshopInformation grogInfor) |
| 前置条件 | 获得增加新酒店的信息 |
| 后置条件 | 增加新酒店 |
| GrogshopDB.update | 语法 | Public boolean update(int grogshopId,GrogshopInformation grogIInfor) |
| 前置条件 | 需要更新酒店信息 |
| 后置条件 | 酒店信息更新 |
| networkingWorkerDB.createNewWorker | 语法 | Public int createNewWorker(networkingWorkerInformation netInfor) |
| 前置条件 | 需要增加网站营销人员 |
| 后置条件 | 添加成功 |
| networkingWorkerDB.update | 语法 | Public boolean update(networkingWorkerInformation netInfor) |
| 前置条件 | 更新网站营销人员信息 |
| 后置条件 | 更新成功 |
| IndentDB.getIndentByCustomerId | 语法 | Public public ArrayList<IndentInformation> getIndentByCustomerId(int customerId,int type) |
| 前置条件 | 账号Id存在 |
| 后置条件 | 获得用户的订单信息 |
| IndentDB.getIndentDyGrogshoId | 语法 | public ArrayList<IndentInformation> getIndentByGrogshopId(int grogshopId) |
| 前置条件 | 酒店账号存在 |
| 后置条件 | 返回酒店所有的订单信息 |
| IndentDB.getIndentByIndentId | 语法 | public IndentInformation getIndentByIndentId(int indentId) |
| 前置条件 | 订单id存在 |
| 后置条件 | 返回该订单信息 |
| IndentDB.insertNewIndent | 语法 | Public int insertNewIndent(IndentInformation inInfor) |
| 前置条件 | 该订单信息正确 |
| 后置条件 | 返回该订单Id |
| IndentDB.updateIndnetStatus | 语法 | public boolean updateIndentStatus(int indentId,int status) |
| 前置条件 | 订单状态发生改变 |
| 后置条件 | 返会是否成功 |
| SalePromotion.getSalePromotion | 语法 | public SalePromotionInformation getSalePromotion(int Id) |
| 前置条件 | Id合理且存在 |
| 后置条件 | 返回促销策略信息 |
| SalePromotion.insertNewSalePromotion | 语法 | protected boolean insertNewSalePromotion(SalePromotionInformation saleInfor) |
| 前置条件 | 在酒店新建时 |
| 后置条件 | 自动创建一个促销策略 |
| SalePromotion.updateSalePromotion | 语法 | public boolean UpdateSalePromotion(SalePromotionInformation saleInfor) |
| 前置条件 | 促销策略发生改变 |
| 后置条件 | 更新促销策略 |
| IDManager.register | 语法 | **synchronized** **public** **static** String registe(**int** id ) |
| 前置条件 | 客户端连接并登入 |
| 后置条件 | 记录客户端的连接 |
| IDManager.unregister | 语法 | **synchronized** **public** **static** String unregiste(String accredit ) |
| 前置条件 | 客户端失去连接 |
| 后置条件 | 抹去客户端的连接记录 |
| DBlinke.createStatement | 语法 | **protected** **static** Statement createStatement() |
| 前置条件 | 需要对数据库进行操作 |
| 后置条件 | 创建对数据库操作的对象 |
| IDManager.createIndent | 语法 | **synchronized public static int createIndentId()** |
| 前置条件 | 新订单创建 |
| 后置条件 | 生成一个订单Id |
| IDManager.createCustomerId | 语法 | **synchronized public static int createCustomerId()** |
| 前置条件 | 新客户创建 |
| 后置条件 | 生成一个客户Id |
| IDManager.createGrogshop | 语法 | **synchronized public static int createGrogshopId()** |
| 前置条件 | 新酒店创建 |
| 后置条件 | 生成一个酒店工作人员Id |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | **服务** |
| DBlinke.createSatement | 提供对数据库的Statement对象 |

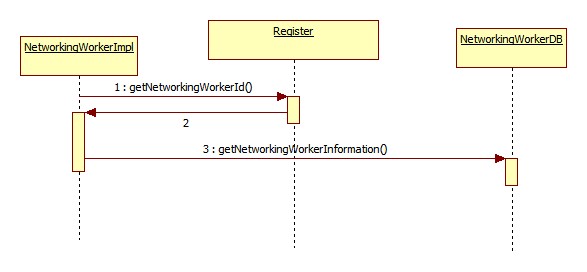
（5）业务逻辑的动态图

A)客户端授权查看客户信息

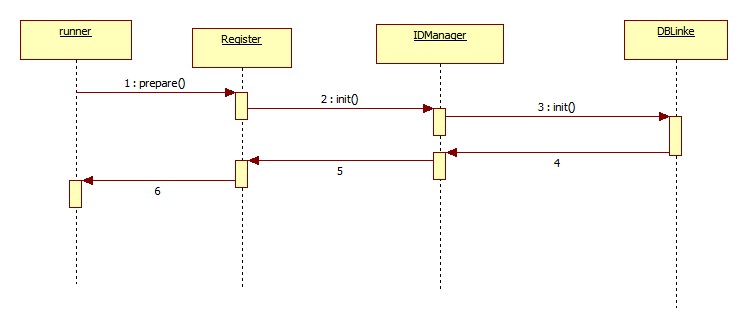
B)酒店端授权查看酒店信息



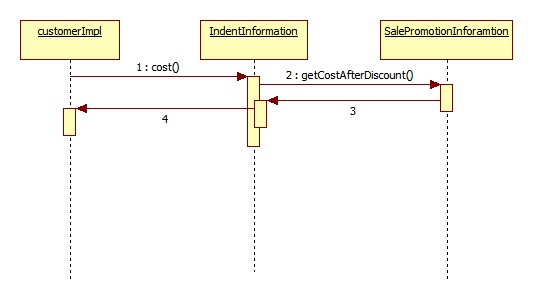
C)网站营销人员授权查看信息



D)系统启动顺序图



E)客户订单计算价格



(7)业务逻辑层的设计原理

采用分散式的控制风格，由客户端发起请求然后处理请求

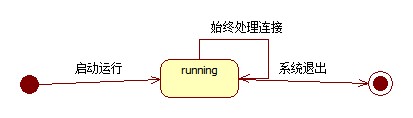
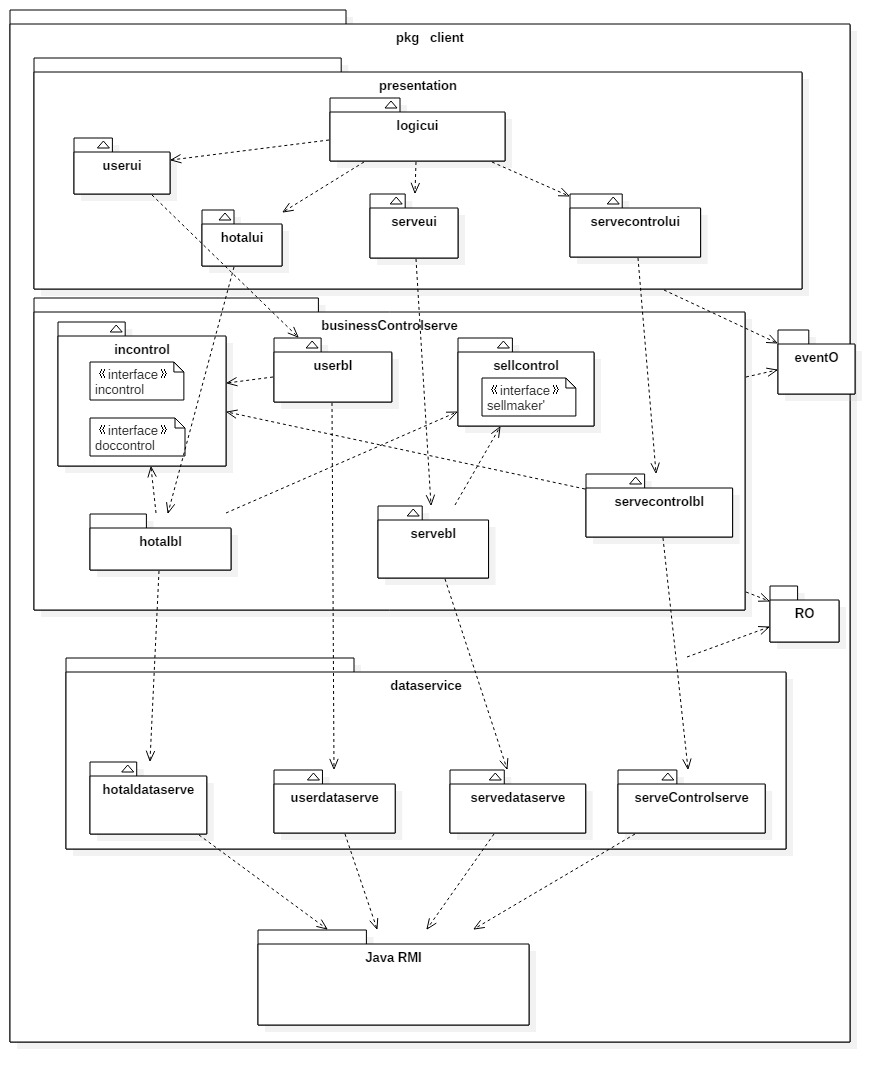


图 5服务器的状态图

# 5. 依赖视角

下图分别是客户端与服务端各自包之间的依赖关系。



（提示：eventO包中是对监听器对象的声明，RO是对网络联通对象的声明）

