目录

[更新历史 2](#_Toc463988925)

[一、 引言 3](#_Toc463988926)

[(一) 编制目的 3](#_Toc463988927)

[(二) 词汇表 3](#_Toc463988928)

[(三) 参考资料 4](#_Toc463988929)

[二、 产品描述 5](#_Toc463988930)

[三、 逻辑视角 6](#_Toc463988931)

[四、 组合视角 8](#_Toc463988932)

[(一) 开发包图 8](#_Toc463988933)

[(二) 运行时进程 13](#_Toc463988934)

[(三) 物理部署 14](#_Toc463988935)

[五、 接口视角 15](#_Toc463988936)

[(一) 模块的职责 15](#_Toc463988937)

[1. 客户端模块视图 15](#_Toc463988938)

[2. 服务器端模块视图 15](#_Toc463988939)

[3. 客户端各层的职责 15](#_Toc463988940)

[4. 服务器端各层的职责 16](#_Toc463988941)

[5. 各层联系 16](#_Toc463988942)

[(二) 用户界面层的分解 17](#_Toc463988943)

[1. 用户界面层模块的职责 18](#_Toc463988944)

[2. 用户界面模块的接口规范 18](#_Toc463988945)

[3. 用户界面模块设计原理 19](#_Toc463988946)

[(三) 业务逻辑层的分解 19](#_Toc463988947)

[1. 业务逻辑层模块的职责 19](#_Toc463988948)

[2. 业务逻辑模块的接口模范 19](#_Toc463988949)

[(四) 数据层的分解 19](#_Toc463988950)

[1. 数据层模块的职责 20](#_Toc463988951)

[2. 数据层模块的接口规范 20](#_Toc463988952)

[六、 信息视角 21](#_Toc463988953)

[(一) 数据持久化对象 21](#_Toc463988954)

[(二) 文件格式 21](#_Toc463988955)

# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **修改日期** | **修改原因** | **版本号** |
| 冯俊杰 | 2016/10/4 | 最初草稿 |  |
| 全体成员 | 2016/10/10 | 新增逻辑分包 |  |
| 全体成员 | 2016/10/11 | 新增物理分包 |  |
| 冯俊杰 | 2016/10/11 | 排版、美化 | V0.1 |

# 引言

## 编制目的

本报告详细完成对互联网酒店预订系统的概要设计，达到指导详细设计和开发的目的，同时实现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. 丁二玉，刘钦.计算与软件工程（卷二）[M]机械工业出版2012：134—182
4. IEEE std 1471-2000

# 产品描述

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。

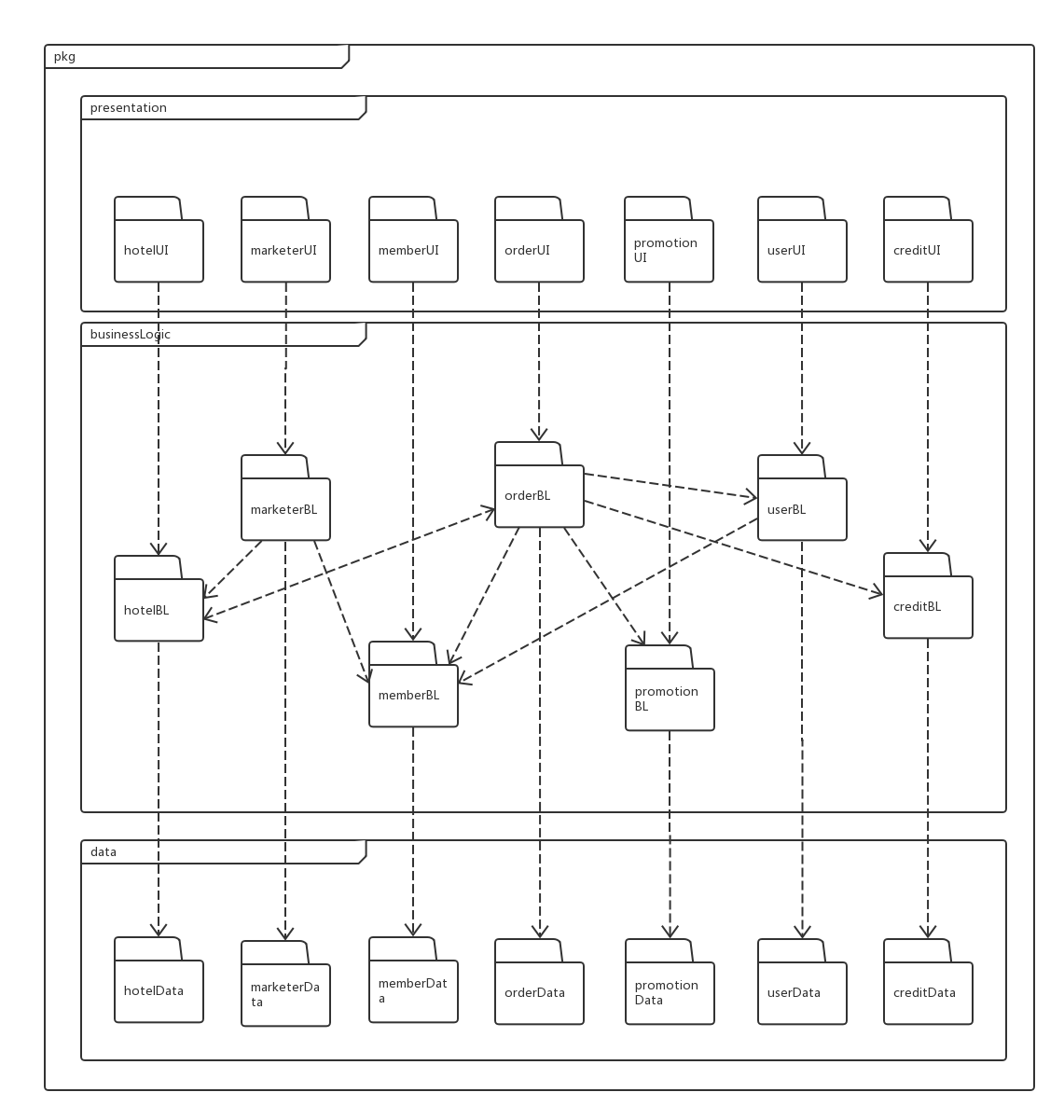


# 逻辑视角

互联网酒店预订系统中，选择了分层体系结构的风格，将系统分为3层（展示层、业务逻辑层、数据层）能够很好的示意整个高层抽象。展示层包括GUI页面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。分层体系结构的逻辑视角和逻辑设计方案如图1和图2所示。



**图1：分层结构**



**图2：逻辑设计方案**

# 组合视角

## 开发包图

[软工2 166]与抽象的逻辑设计相比，实现物理设计要考虑更多的实现细节，这些细节有：

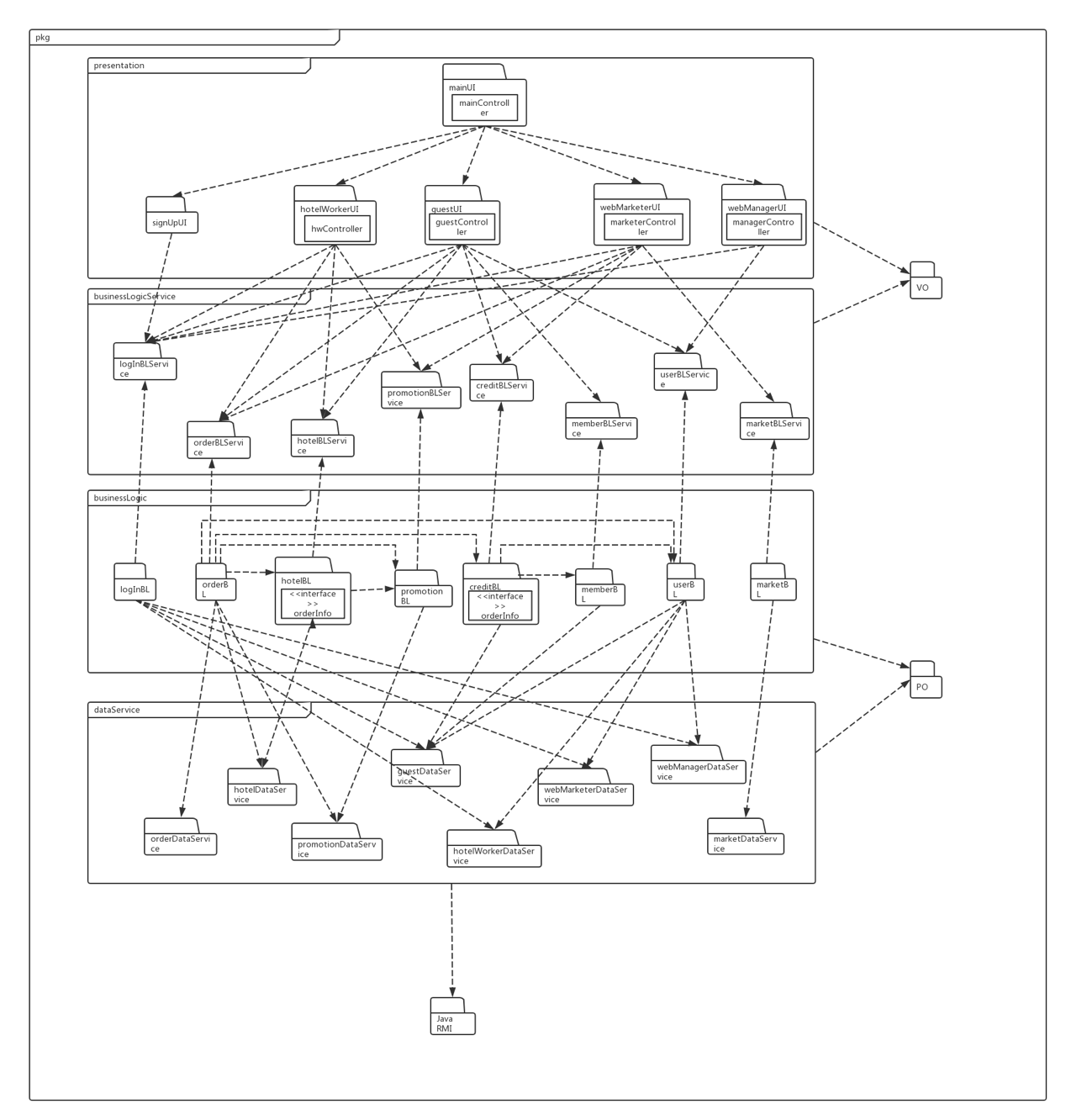
1. 所有的presentation层开发包都需要使用图形类型建立界面，都要依赖于图形界面类库包。
2. 在presentation层实现时，由mainUI包负责不同参与者在整个页面之间的跳转逻辑。其他各包负责各自页面自身的功能，并由各自的controller负责自身内部页面之间的跳转逻辑。
3. presentation层与businessLogic层被置于客户端，data层被置于服务器端，那么businessLogic层的开发包不可能依赖于data层的开发包。使用RMI技术，将data层开发包分解为置于客户端的dataService接口包和置于服务器端的data层开发包。这样一来，businessLogic层开发包依赖于dataService包，dataService包依赖于RMI类库包。
4. 所有的data层开发包都需要进行数据持久化（例如读写数据库、读写文件等），所以它们会有一些重复代码，可以将重复代码独立为新的开发包，然后所有的data层开发包都依赖于dataBaseUtility。dataBaseUtility会依赖于JDBC类库包。
5. 在分层风格的典型设计中，不希望高层直接依赖于低层，而是为低层建立接口包，实现依赖倒置原则，所以应该调整为：各presentation层开发包（调用）依赖于logic层接口包businessLogicService包，businessLogic层开发包也依赖于（实现了）businessLogic层接口包businessLogicService包。
6. 在分层风格的典型设计中，presentation层与logic层之间、logic层与data层之间可能会传递复杂数据对象，那么相邻两层都需要使用数据对象声明，所以需要将数据对象声明独立为开发包——VO包和PO包。其中VO包负责在presentation层与logic层之间传递，PO包负责在logic层与data层之间传递。
7. 使用依赖倒置原则消除包的循环依赖现象，将循环依赖变为单向依赖：
8. orderBL和hotelBL:将部分orderBL类抽象接口orderInfo置入hotelBL包，这样orderBL单向依赖于hotelBL（实现接口+调用）
9. orderBL和creditBL:将部分orderBL类抽象接口orderInfo置入creditBL包，这样orderBL单向依赖于hotelBL（实现接口+调用）

8）在logic层中，初始化和业务逻辑层上下文的工作被分配到utility包中。

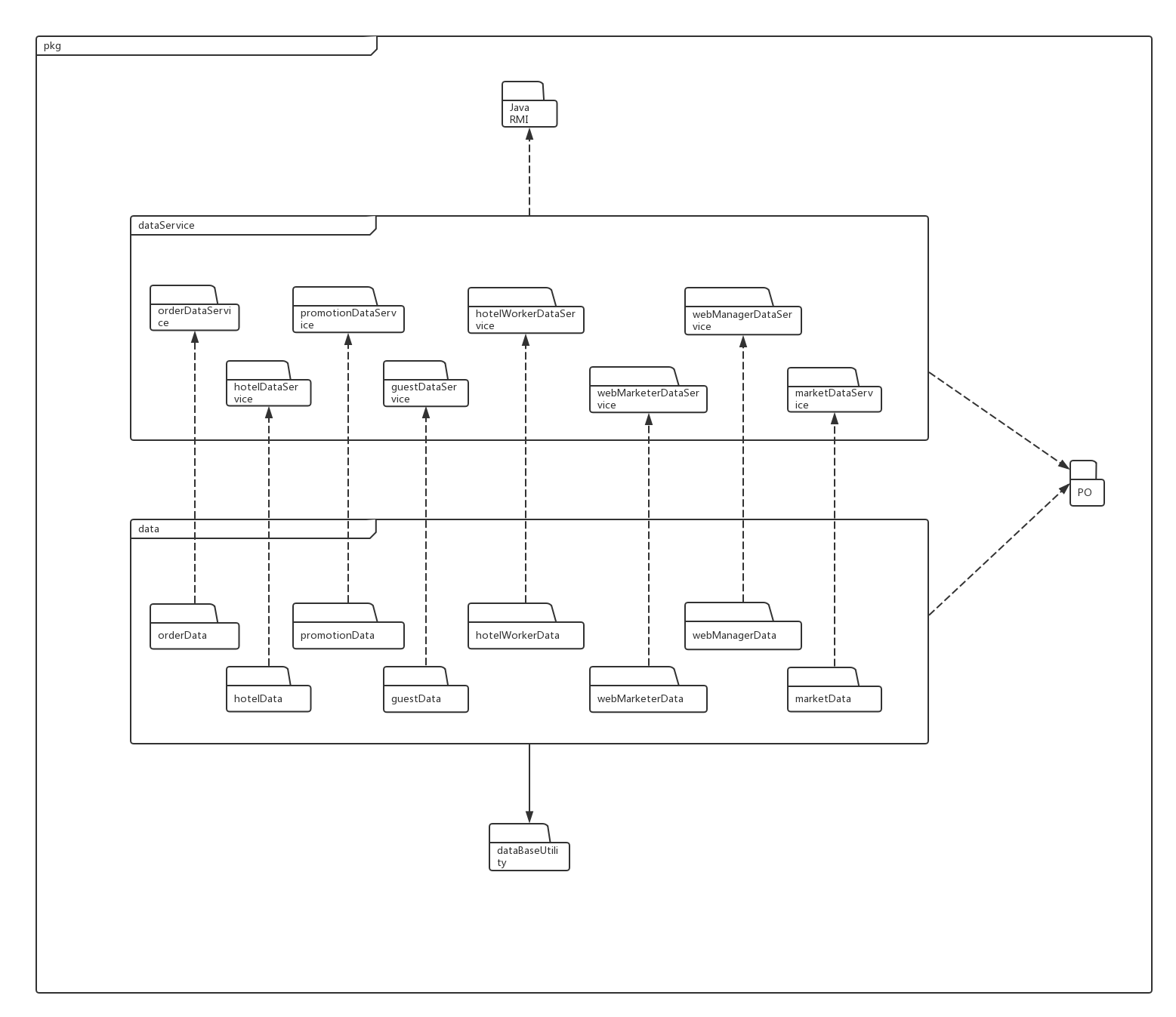
界面层依据参与者分为main、注册界面和4个主界面；逻辑层主要依据逻辑分包的结果，并添加logInBL，负责用户的登录和客户的注册；数据层进行整合合并，依照参与者、订单、酒店的实际情况进行分类。

|  |  |
| --- | --- |
| 开发（物理）包 | 依赖的其他包 |
| mainUI | signUpUI, guestUI, hotelWorkerUI, webMarketerUI, webManagerUI, 界面类库包, VO |
| signUpUI | logInBLService, 界面类库包, VO |
| guestUI | logInBLService, orderBLService, hotelBLService, creditBLService, memberBLService, UserBLService, 界面类库包, VO |
| hotelWorkerUI | logInBLService, orderBLService, hotelBLService, promotionBLService, 界面类库包,VO |
| webMarketerUI | logInBLService, orderBLService, promotionBLService, creditBLService, marketBLService, 界面类库包,VO |
| webManagerUI | logInBLService, UserBLService, 界面类库包,VO |
| logInBLService | VO |
| orderBLService | VO |
| hotelBLService | VO |
| promotionBLService | VO |
| creditBLService | VO |
| memberBLService | VO |
| marketBLService | VO |
| userBLService | VO |
| logInBL | logInBLService, guestDataService, hotelWorkerDataService, webMarketerDataService, webManagerDataService, PO |
| orderBL | orderBLService, orderDataService, hotelBL, promotionBL, creditBL, userBL, PO |
| hotelBL | hotelBLService, hotelDataService, orderBL, promotionBL, PO |
| promotionBL | promotionBLService, promotionDataService, PO |
| creditBL | creditBLService, guestDataService, orderBL, memberBL, userBL, PO |
| memberBL | memberBLService, guestDataService, PO |
| marketBL | marketBLService, marketDataService, PO |
| userBL | userBLService, guestDataService, hotelWorkerDataService, webMarketerDataService, webManagerDataService, PO |
| orderDataService | Java RMI, PO |
| hotelDataService | Java RMI, PO |
| guestDataService | Java RMI, PO |
| hotelWorkerDataService | Java RMI, PO |
| webMarketerDataService | Java RMI, PO |
| webManagerDataService | Java RMI, PO |
| promotionDataService | Java RMI, PO |
| marketDataService | Java RMI, PO |
| orderData | orderDataService, PO |
| hotelData | hotelDataService, PO |
| guestData | guestDataService, PO |
| hotelWorkerData | hotelWorkerDataService, PO |
| webMarketerData | webMarketerDataService, PO |
| webManagerData | webManagerDataService, PO |
| promotionData | promotionDataService, PO |
| marketData | marketDataService, PO |
| VO |  |
| PO |  |
| dataBaseUtility | java.JDBC |
| 界面类库包 | marketDataService, PO |
| JAVA RMI |  |
| java.swing,java.awt,java.2D,java.3D |  |

下图为互联网酒店预订系统客户端开发包图：

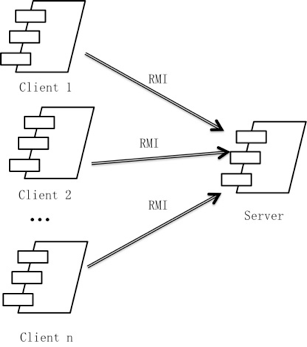


下图为互联网酒店预订系统服务端开发包图：



## 运行时进程

在互联网酒店预订系统中，会有多个客户端进程和一个服务器端进程，其进程图如图5所示。结合部署图，客户端进程是在客户端机器上运行，服务器端进程是在服务器端机器上运行。



## 物理部署

互联网酒店预订系统中客户端构件是放在客户端机上，服务器端构件是放在服务器端机器上。在客户端节点上，还要部署RMIStub构件。由于Java RMI 构件属于JDK 1.8的一部分。所以，在系统JDK环境已经设置好的情况下，不需要独立部署。部署图如下图所示。



# 接口视角

## 模块的职责

### 客户端模块视图



### 服务器端模块视图



### 客户端各层的职责

|  |  |
| --- | --- |
| 层 | 职责 |
| 启动模块 | 负责初始化网络通信机制，启动用户界面 |
| 用户界面层 | 基于窗口的互联网预订系统客户端用户界面 |
| 业务逻辑层 | 对于用户界面的输入进行响应并进行业务处理逻辑 |
| 客户端网络模块 | 利用java RMI机制查找RMI服务 |

### 服务器端各层的职责

|  |  |
| --- | --- |
| 层 | 职责 |
| 启动模块 | 负责初始化网络通信机制，启动用户界面 |
| 数据层 | 负责数据的持久化及数据访问接口 |
| 服务器端网络模块 | 利用Java RMI机制开启RMI服务，注册RMI服务 |

### 各层联系

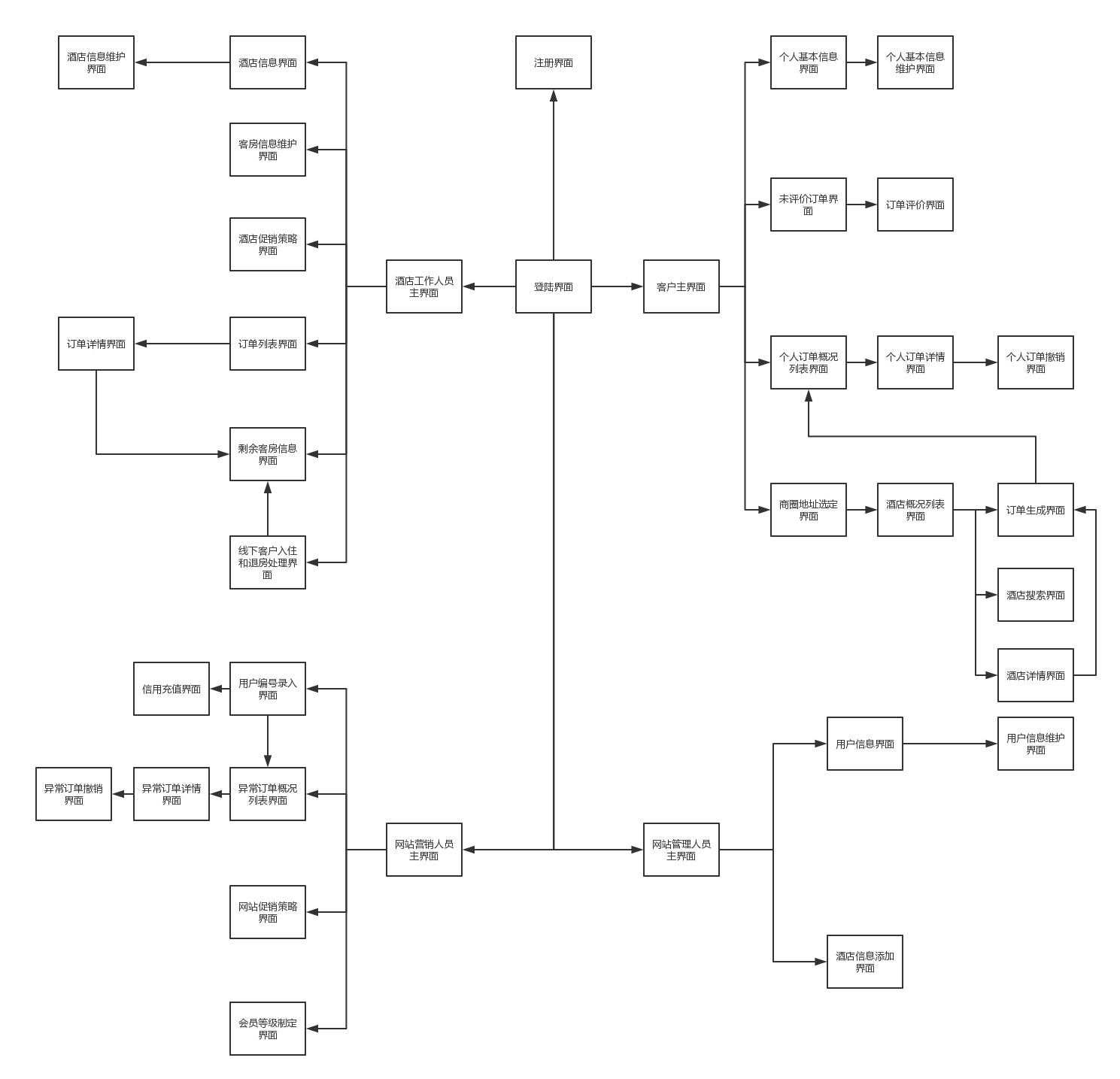
每一层只是使用下方直接接触的层。层与层之间仅仅是通过接口的调用来完成的。层之间调用的接口如下图所示。

|  |  |  |
| --- | --- | --- |
| **接口** | **服务调用方** | **服务提供方** |
| logInBLService  orderBLService  hotelBLService  promotionBLService  creditBLService  memberBLService  marketBLService  userBLService | 客户端  展示层  （presentation） | 客户端  业务逻辑层  （businessLogic） |
| orderDataService  hotelDataService  guestDataService  hotelWorkerDataService  webMarketerDataService  webManagerDataService  promotionDataService  marketDataService | 客户端  业务逻辑层  （businessLogic） | 服务器端  数据层  （data） |

## 用户界面层的分解

根据需求，系统存在36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、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、**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。



### 用户界面层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| MainFrame | 界面frame，负责界面的显示和界面的跳转 |

### 用户界面模块的接口规范

1. **用户界面层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| mainUI | 语法 | init(args:String[]) |
| 前置条件 | 用户触发相应需要显示界面的条件 |
| 后置条件 | 显示Frame并且加载LoginPanel |

1. **用户界面层模块需要的服务接口**

|  |  |
| --- | --- |
| 服务名 | 服务 |
| bl.accountservice | 登录界面的业务逻辑接口 |
| bl.\*blservice | 每个ui模块会有对应的数据逻辑接口 |

### 用户界面模块设计原理

用户界面拟用java的Swing和AWT库来实现。

## 业务逻辑层的分解

业务逻辑层包括多个针对界面及业务逻辑处理对象。例如User对象负责处理登录界面

的业务逻辑。业务逻辑层的设计如下图所示：

### 业务逻辑层模块的职责

### 业务逻辑模块的接口模范

## 数据层的分解

数据层主要给业务逻辑层提供数据访问服务，包括对于持久化数据的增、删、改、查。例如，AccountBusinessService的服务主要由AccountDataService提供。由于持久化数据的保存可能存在多种形式：Txt文件、序列化文件、数据库等，所示抽象了数据服务。以AccountDataService为例，数据层模块的具体描述下图所示：

### 数据层模块的职责

### 数据层模块的接口规范

# 信息视角

## 数据持久化对象

系统的PO类就是对应的相关的实体类，如下所示。本系统共包含23个PO类，在此

不一一赘述，只做部分介绍：

AccountPO类为账户信息，包括ID、密码、账户类型、个人其他信息。

ConstPO类为系统常量，包括城市间距离，交通方式运费，包装费用等等。

DocPO类为所有单据父类，储存了单据ID、生成时间、单据类型、单据状态。

持久化对象ReceivePO类的定义如图所示：

## 文件格式

本系统依据不同情况分别采用数据库、序列化、txt文件持久化保存三种方式保存文件