酒店管理系统

Management System of Hotels

软件体系结构描述文档

V1.0

南京大学软件学院 Octopus小组

成员：周沁涵、桑田、钱柯宇、潘潇睿

2016-10-09

目录

[更新历史 3](#_Toc463945263)

[1. 引言 4](#_Toc463945264)

[1.1 编制目的 4](#_Toc463945265)

[1.2 词汇表 4](#_Toc463945266)

[1.3 参考资料 4](#_Toc463945267)

[2. 产品概述 5](#_Toc463945268)

[3. 逻辑视角 6](#_Toc463945269)

[3. 组合视角 7](#_Toc463945270)

[4.1 开发包图 9](#_Toc463945271)

[4.2 运行时进程 13](#_Toc463945272)

[4.3 物理部署 14](#_Toc463945273)

[5. 接口视角 14](#_Toc463945274)

[5.1 模块的职责 14](#_Toc463945275)

[5.2 用户界面层的分解 18](#_Toc463945276)

[5.2.1 用户界面层模块的职责 18](#_Toc463945277)

[5.2.2 用户界面层模块的接口规范 18](#_Toc463945278)

[5.2.3 用户界面模块设计原理 19](#_Toc463945279)

[5.3 业务逻辑层的分解 19](#_Toc463945280)

[5.3.1 业务逻辑层模块的职责 19](#_Toc463945281)

[5.3.2 业务逻辑层模块的接口规范 20](#_Toc463945282)

[5.4 数据层的分解 20](#_Toc463945283)

[5.4.1 数据层模块的职责 20](#_Toc463945284)

[5.4.2 数据层模块的接口规范 21](#_Toc463945285)

[6. 信息视角 21](#_Toc463945286)

[6.1 数据持久化对象 21](#_Toc463945287)

[6.2 数据库表 21](#_Toc463945288)

# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **日期** | **变更原因** | **版本号** |
| 全体成员 | 2016-10-9 | 最初草稿 | V0.0.0 草稿 |
| 桑田 | 2016-10-10 | 修正排版 | V1.0.0 |
| 周沁涵 | 2016-10-11 | 修改“逻辑视角”部分内容 | V1.0.1 |

# 1. 引言

## 1.1 编制目的

本报表详细完成对酒店管理系统的概要设计，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。

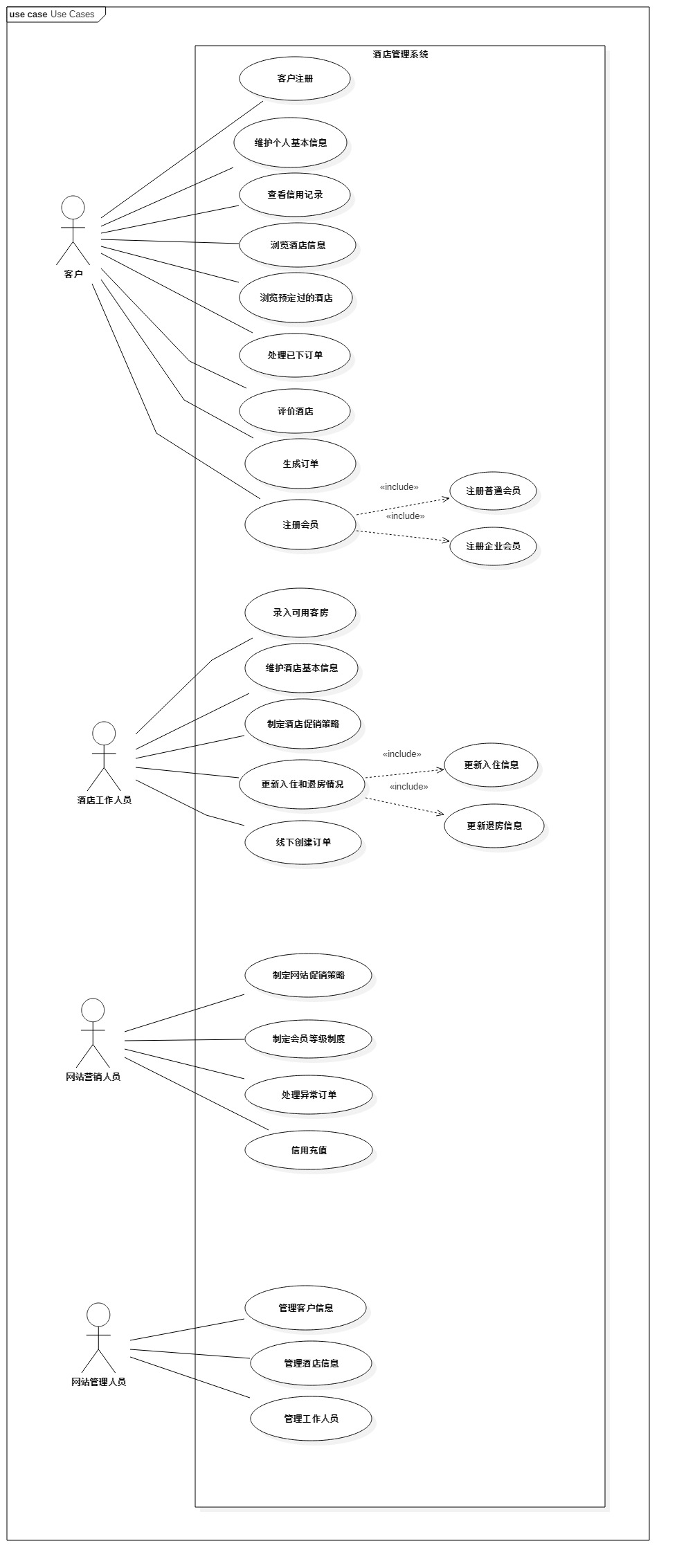
## 1.2 词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| \_ui | 表示某展示层 |  |
| \_bl | 表示某逻辑层 |  |
| \_data | 表示某数据层 |  |
| RMI | 表示远程方法调用 |  |
| utility\_bl | 表示初始化和业务逻辑上下文的工作 |  |
| datafactory | 表示调用其他数据库的方法 |  |

## 1.3 参考资料

1. IEEE std 1471-2000
2. 丁二玉，刘钦.计算与软件工程（卷二）[M]机械工业出版2012：134—182

# 2. 产品概述

参考酒店管理系统用例文档和酒店管理系统软件需求规格说明文档中队产品的概括描述。酒店管理系统主要是应用于建立线上酒店预订系统，并添加营销策略与管理,主要功能见用例图如下。

# 3. 逻辑视角

酒店管理系统中，选择了分层体系结构风格，将系统分为3层(展示层、

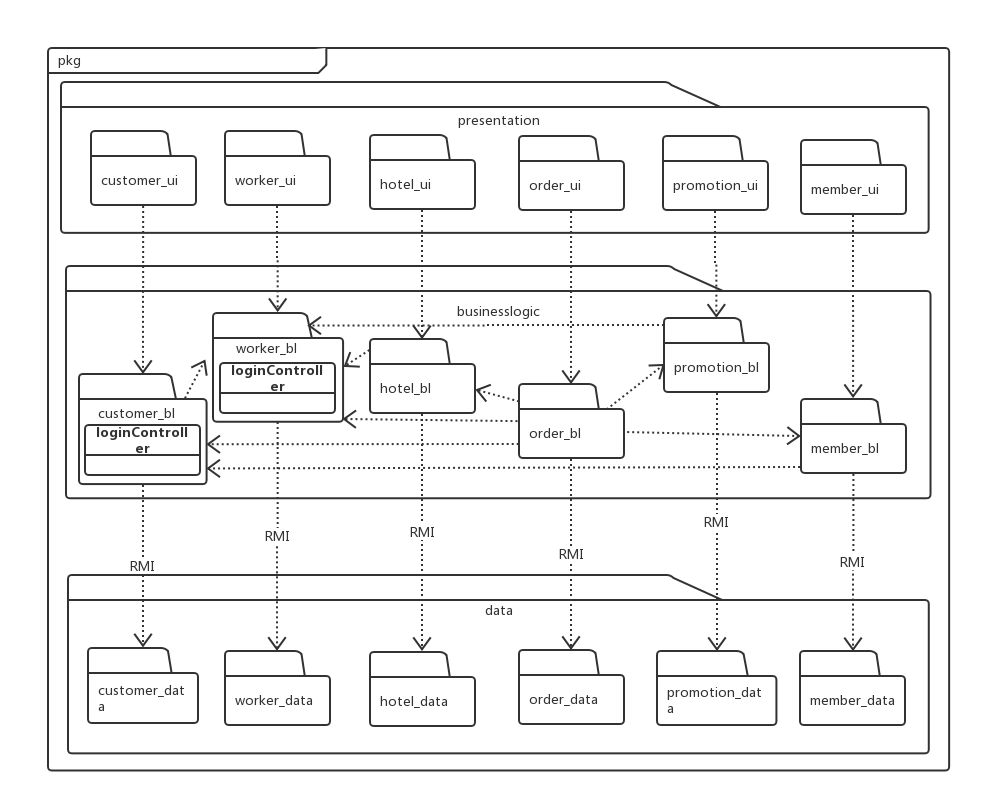
业务逻辑层、数据层)能够很好地示意整个高层抽象。展示层包含GUI页面

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一和图二所示。



图一 参照体系结构风格的包图表达逻辑视角



图二 软件体系结构逻辑设计方案

# 组合视角

[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. presentation层与logic层被置于客户端，data层被置于服务器端，那么logic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data\_service接口包和置于服务器端的data层开发包。这样一来，logic层开发包依赖于data\_service包，data\_service包和data层的开发包都依赖于RMI类库包。
2. 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。datavaseutility会依赖于JDBC类库包。
3. 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。
4. 此外，presentation层实现时，由main\_ui包负责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。
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，logic层开发包也依赖于（实现了）logic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）。
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：

* Order和Promotion:将部分Order类抽象接口置入Promotion包，这样Order单向依赖于Promotion（实现接口+调用）
* Hotel和Order:将部分Order类抽象接口orderInfoService置入Hotel包，这样Order单向依赖于Hotel（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\_bl包中。

经过细节改进，最终建立的酒店管理系统开发包设计如表4.1-1，其局部包图如图4.1-1和图4.1-2所示：

## 4.1 开发包图

表4.1-1 开发包设计

|  |  |
| --- | --- |
| 开发（物理）包 | 依赖的其他开发包 |
| main\_ui | login\_ui, customer\_ui, clerk\_ui, marketer\_ui, manager\_ui, search\_ui, hotel\_ui, order\_ui, member\_ui, promotion\_ui, vo |
| login\_ui | login\_bl\_service,vo,界面类库包 |
| login\_bl\_service |  |
| login\_bl | login\_bl\_service, data\_factory,po, utlity\_bl |
| customer\_ui | customer\_bl,vo,界面类库包 |
| cutomer\_bl\_service |  |
| customer\_bl | cutomer\_bl\_service,utlity,po, customer\_data\_service |
| customer\_data\_service | Java RMI,po |
| customer\_data |  |
| clerk\_ui | clerk\_bl\_service,vo,界面类库包 |
| clerk\_bl\_service |  |
| clerk\_bl | clerk\_data\_service, data\_factory,utility\_bl,po |
| clerk\_data\_service | Java RMI,po |
| clerk\_data |  |
| marketer\_ui | marketer\_bl\_service,vo,界面类库包 |
| marketer\_bl\_service |  |
| marketer\_bl | marketer\_bl\_service, data\_factory, utility\_bl,po |
| marketer\_data\_service | Java RMI,po |
| marketer\_data |  |
| manager\_ui | manager\_bl\_service, vo,界面类库包 |
| manager\_bl\_service |  |
| manager\_bl | manager\_data\_service, manager\_bl\_service, utility\_bl,po |
| manager\_data\_service | Java RMI,po |
| manager\_data |  |
| search\_ui | search\_bl\_service,vo,界面类库包 |
| search\_bl\_service |  |
| search\_bl | search\_bl\_service,data\_factory, utility\_bl,po |
| hotel\_ui | hotel\_bl\_service, vo,界面类库包 |
| hotel\_bl\_service |  |
| hotel\_bl | manager\_bl,clerk\_bl,utility,data\_factory, hotel\_data\_service |
| hotel\_data\_service | Java RMI,po |
| hotel\_data |  |
| order\_ui | order\_bl\_service, vo,界面类库包 |
| order\_bl\_service |  |
| order\_bl | order\_bl\_service ,customer\_bl,clerk\_bl,marketer\_bl,hotel\_bl,member\_bl,promoteon\_bl,utility,po,data\_factory, order\_data\_service |
| order\_data\_service | Java RMI,po |
| order\_data |  |
| member\_ui | member\_bl\_service, vo,界面类库包 |
| member\_bl\_service |  |
| member\_bl | member\_bl\_service,customer\_bl,utility,po, member\_data\_service |
| member\_data\_service | Java RMI,po |
| member\_data |  |
| promotion\_ui | promotion\_bl\_service, vo,界面类库包 |
| promotion\_bl\_service |  |
| promotion\_bl | promotion\_bl\_service, promotion\_data\_service |
| promotion\_data\_service | Java RMI,po |
| promotion\_data |  |
| vo |  |
| po |  |
| utility\_bl |  |
| 界面类库包 |  |
| Java RMI |  |
| Datafactory\_service |  |
| databaseutility | JDBC |

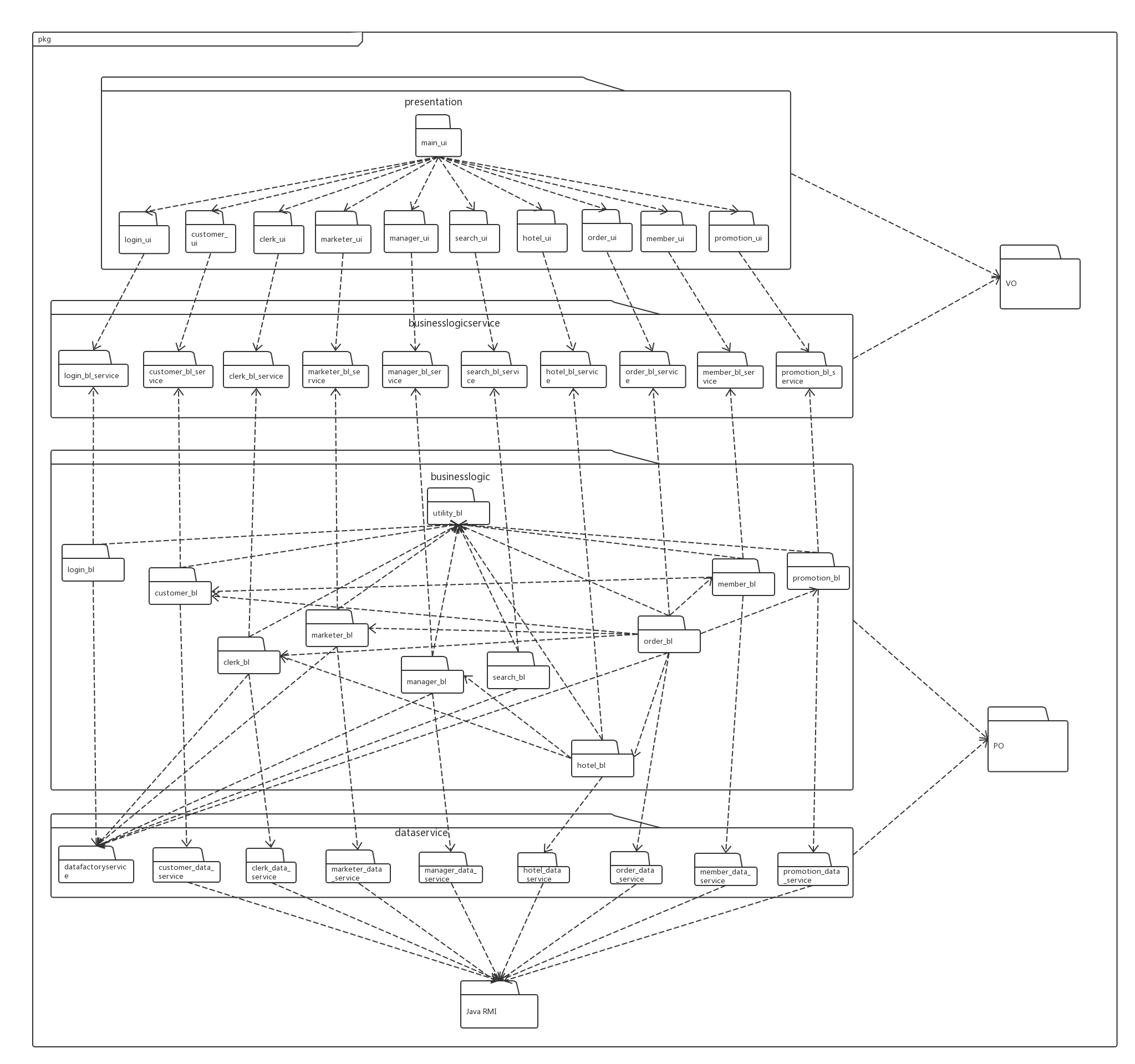


图4.1-1 快递物流系统客户端开发包图

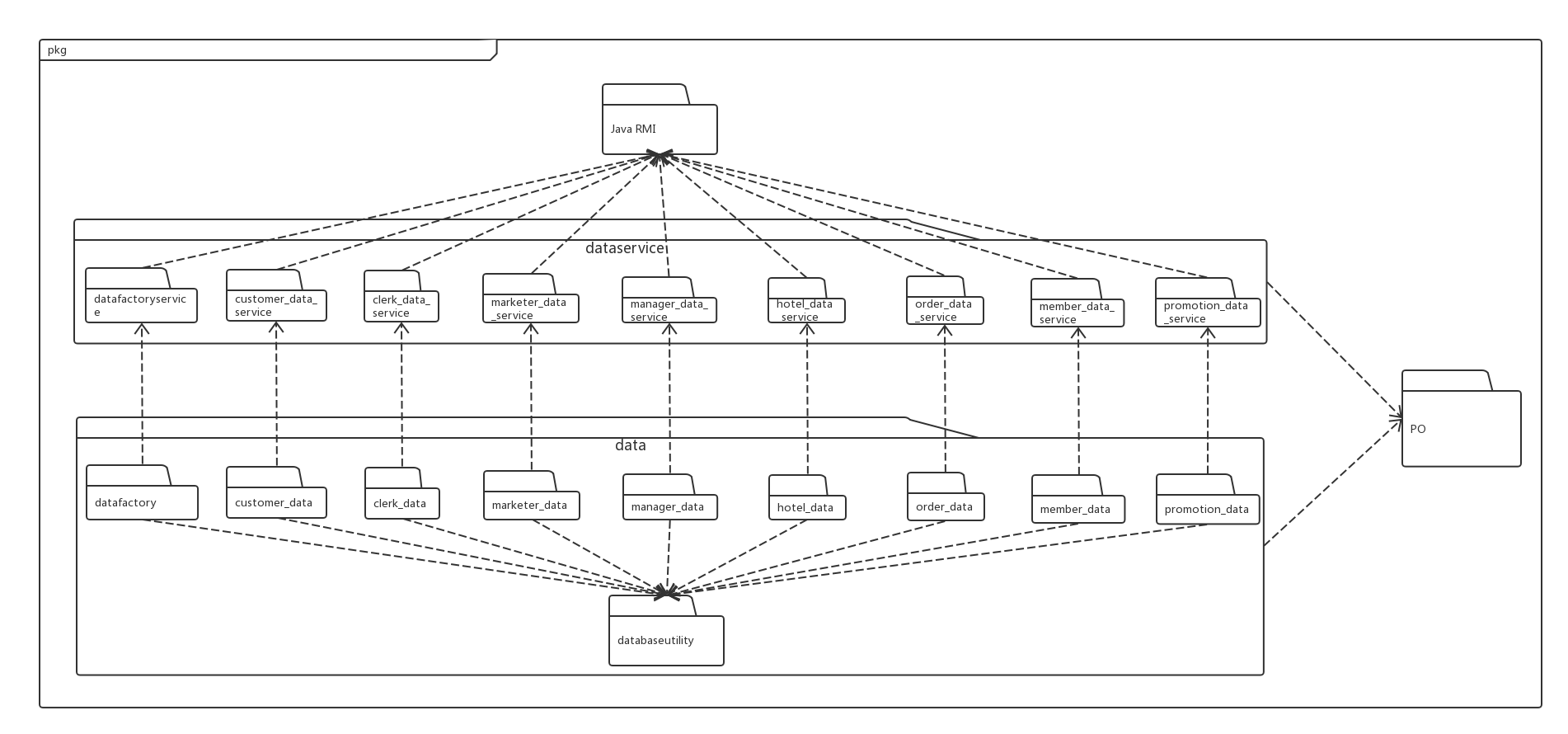


图4.1-2 快递物流系统服务器端开发包图

## 4.2 运行时进程

在酒店管理系统中，会有多个客户端进程和一个服务器端进程，其进程图如图4.2所示。结合部署图，客户端进程实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。

图4.2 进程图

## 4.3 物理部署

酒店管理系统中客户端构件是放在客户端机器上，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JavaRMI构件属于JDK6.0的一部分。所以，在系统JDK环境已经设置好的情况下，不需要再独立部署。部署图如图4.3所示。



图4.3 部署图

# 5. 接口视角

## 5.1 模块的职责

客户端模块和服务器端模块视图分别如图5.1-1和图5.2-2所示。客户端各层和服务器端各层的职责分别如表5.1-1和表5.2-2所示。



图5.1-1 客户端模块视图



图5.1-2 服务器端模块视图

表5.1-1 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 | 基于窗口的酒店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响应和业务处理逻辑。 |
| 客户端网络模块 | 利用Java RMI机制查找RMI服务，检测网络连接状态，进行断线重连。 |

表5.1-2 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。 |
| 服务器端网络模块 | 利用Java RMI机制开启RMI服务，注册RMI服务 |

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表5.1-3所示。

表5.1-3 层之间调用的接口

|  |  |  |
| --- | --- | --- |
| 接口 | 服务调用方 | 服务提供方 |
| login\_bl\_service  customer\_bl\_service  clerk\_bl\_service  marketer\_bl\_service  manager\_bl\_service  search\_bl\_service  hotel\_bl\_service  order\_bl\_service  member\_bl\_service  promotion\_bl\_service | 客户端展示层 | 客户端业务逻辑层 |
| customer\_data\_service  clerk\_data\_service  marketer\_data\_service  manager\_data\_service  hotel\_data\_service  order\_data\_service  member\_data\_service  promotion\_data\_service  datafactoryservice | 客户端业务逻辑层 | 服务器端数据层 |

借用客户维护个人信息用例来说明层之间的调用，如图5.1-3所示。每一层之间都是由上层依赖了一个接口（需接口），而下层实现这个接口（供接口）。customer\_bl\_service提供了customer界面所需要的所有业务逻辑功能customer\_data\_service提供了对数据库的查询、修改等操作。这样的实现就大大降低了层与层之间的耦合。

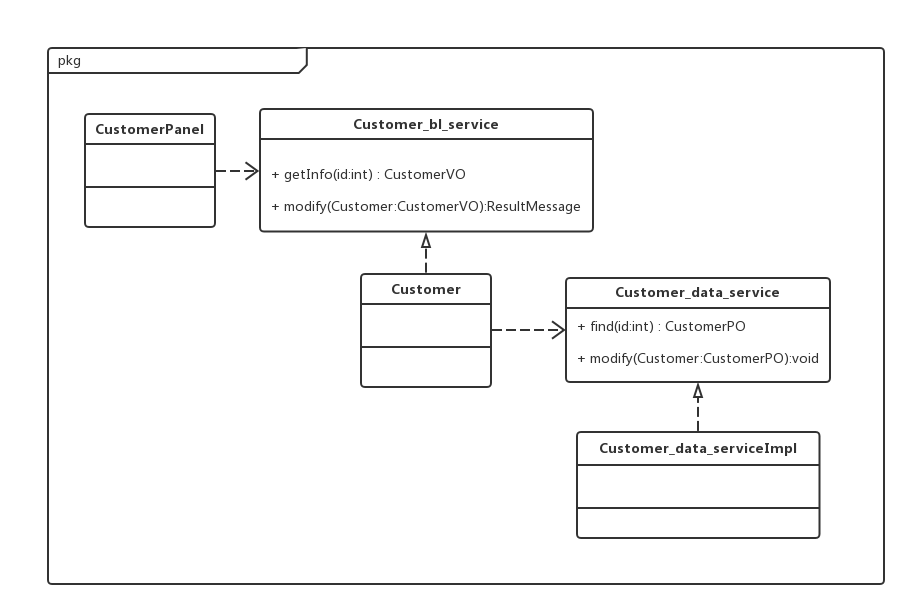


图5.1-3 客户维护个人基本信息用例层之间调用的接口

## 5.2 用户界面层的分解

### 5.2.1 用户界面层模块的职责

如表5所示为用户界面层模块的职责。

表5 用户界面层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
|  |  |
|  |  |

### 5.2.2 用户界面层模块的接口规范

用户界面层模块的接口规范如表6所示。

表6 用户界面层模块的接口规范

|  |  |  |
| --- | --- | --- |
|  | 语法 |  |
| 前置条件 |  |
| 后置条件 |  |

用户界面层模块需要的服务接口如表7所示。

表7 用户界面层模块需要的服务接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
|  |  |
|  |  |

### 5.2.3 用户界面模块设计原理

## 5.3 业务逻辑层的分解

### 5.3.1 业务逻辑层模块的职责

业务逻辑层模块的职责如表8所示。

表5 业务逻辑层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
|  |  |
|  |  |

### 5.3.2 业务逻辑层模块的接口规范

XXX模块的接口规范如表9所示。

表9 XXX模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
|  | 语法 |  |
| 前置条件 |  |
| 后置条件 |  |
|  |  |  |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
|  |  | |

## 5.4 数据层的分解

### 5.4.1 数据层模块的职责

数据层模块的职责如表n所示。

表n 数据层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
|  |  |
|  |  |

### 5.4.2 数据层模块的接口规范

数据层模块的接口规范如表n所示。

表n 数据层模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
|  | 语法 |  |
| 前置条件 |  |
| 后置条件 |  |

# 6. 信息视角

## 6.1 数据持久化对象

## 6.2 数据库表