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参见软件体系结构文档

4.1.1 ClientBL模块

(1) 模块概述

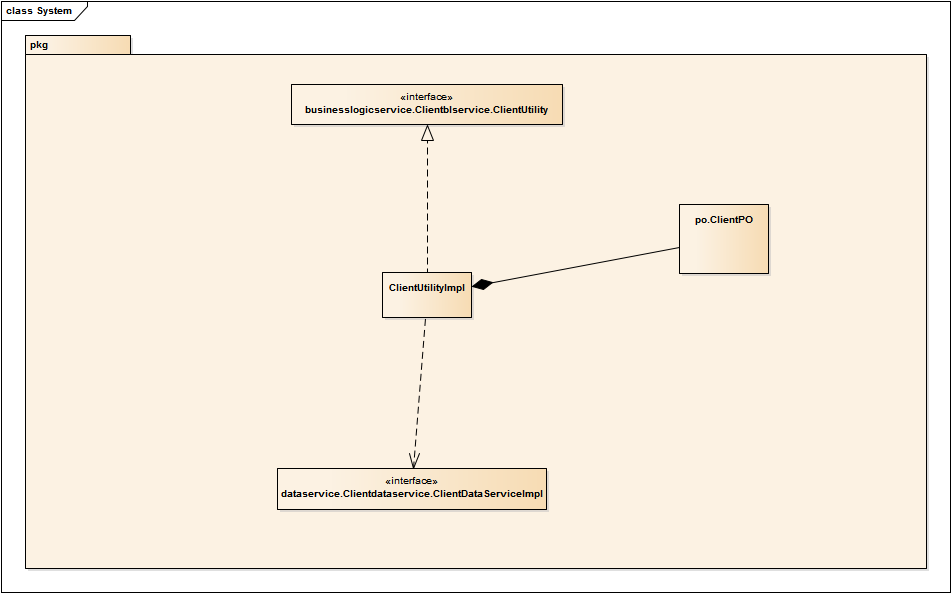
ClientBL模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

ClientBL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，我们将系统分为展示层、业务逻辑层和数据层。每一层之间为了增加灵活性，我们会添加接口。比如展示层和业务逻辑层之间，我们将添加businesslogicservice.Clientblservice.ClientUtility接口。业务逻辑层和数据层之间添加dataservice.Clientdataservice.ClientDataService接口。ClientPO是作为客户信息的持久化对象被添加到设计模型中去的。

模块设计：



ClientBL模块各个类的职责：

|  |  |
| --- | --- |
| 模块 | 职责 |
| LoginController | 负责实现对应于登陆界面所需要的服务 |
| ClientUtility | 负责实现客户管理界面所需要的服务 |
| User | 系统用户的领域模型对象，拥有用户数据的账号和密码，可以解决登陆问题 |

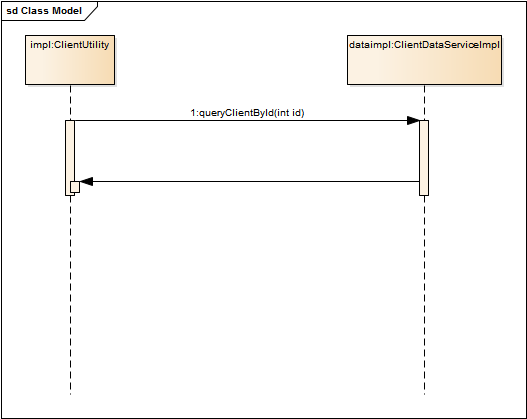
(3) 模块内部类的接口规范

ClientUtility的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ClientUtility.addClient | 语法 | public void addClient(ClientPO clientPO) throws Exception |
| 前置条件 | 输入符合规范 |
| 后置条件 | 调用ClientDataServiceImpl的addClient方法 |
| ClientUtility. deleteClient | 语法 | public void deleteClient(int id) throws Exception |
| 前置条件 | 输入符合规范 |
| 后置条件 | 调用ClientDataServiceImpl的deleteClient方法 |
| ClientUtility. modifyClient | 语法 | public void modifyClient(ClientPO newClientPO) throws Exception |
| 前置条件 | 输入符合规范 |
| 后置条件 | 调用ClientDataServiceImpl的modifyClient方法 |
| ClientUtility. queryClientById | 语法 | public ClientPO queryClientById(int id) throws Exception |
| 前置条件 | 输入符合规范 |
| 后置条件 | 调用ClientDataServiceImpl的queryClientById方法 |
| ClientUtility. queryClient | 语法 | public Vector<ClientPO> queryClient(Vector<String> filters) throws Exception |
| 前置条件 | 输入符合规范 |
| 后置条件 | 调用ClientDataServiceImpl的queryClient方法 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 功能 |
| ClientDataService. addClient(ClientPO clientPO) | 增加一个客户 |
| ClientDataService. deleteClient(int id) | 删除一个客户 |
| ClientDataService.modifyClient(ClientPO newClientPO) | 修改一个客户 |
| ClientDataService. ClientPO queryClientById(int id) | 通过客户编号查询一个客户 |
| ClientDataService.queryClient(Vector<String> filters) | 通过过滤条件查询客户 |

(4) 业务逻辑层的动态模型



(5) 业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问别的业务逻辑由各自的控制器委托给不同的对象。

4.1.2 SaleBL模块

(1) 模块概述

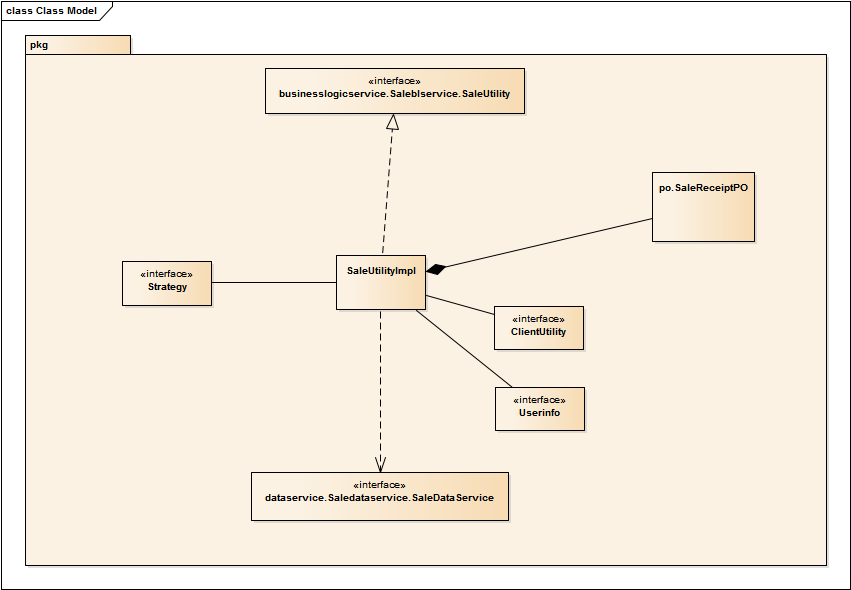
SaleBL模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

SaleBL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，我们会添加接口。比如展示层和业务逻辑层之间，我们添加businesslogicservice.Saleblservice.SaleUtility接口。业务逻辑层和数据层之间添加dataservice.Saledataservice.SaleDataService接口。SaleReceiptPO是作为销售退货单据的持久化对象被添加到设计模型中去的。

模块设计：



SaleBL模块各个类的职责：

|  |  |
| --- | --- |
| 模块 | 职责 |
| LoginController | 负责实现对应于登陆界面所需要的服务 |
| SaleUtility | 负责实现销售退货管理管理界面所需要的服务 |
| User | 系统用户的领域模型对象，拥有用户数据的账号和密码，可以解决登陆问题 |

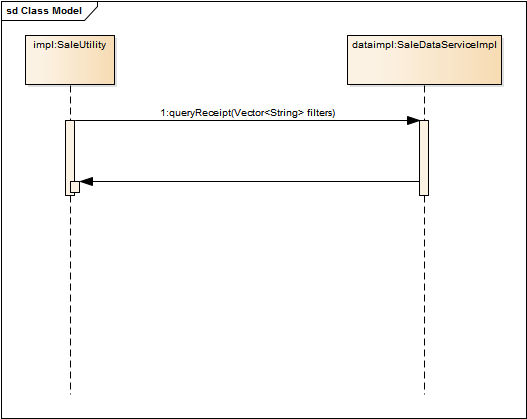
(3) 模块内部类的接口规范

SaleUtility的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| SaleUtility. makeReceipt | 语法 | public void makeReceipt(SaleReceiptPO receipt) throws Exception |
| 前置条件 | 输入符合规范 |
| 后置条件 | 调用SaleDataServiceImpl的makeReceipt方法 |
| SaleUtility. queryReceipt | 语法 | public Vector<SaleReceiptPO> queryReceipt(Vector<String> filters) throws Exception |
| 前置条件 | 输入符合规范 |
| 后置条件 | 调用SaleDataServiceImpl的queryReceipt方法 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 功能 |
| SaleDataService.makeReceipt(SaleReceiptPO receipt) | 增加一条销售退货记录 |
| SaleDataService.queryReceipt(Vector<String> filters) | 查询销售退货记录 |

(4) 业务逻辑层的动态模型



(5) 业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问别的业务逻辑由各自的控制器委托给不同的对象。

4.1.3 StockBL模块

(1) 模块概述

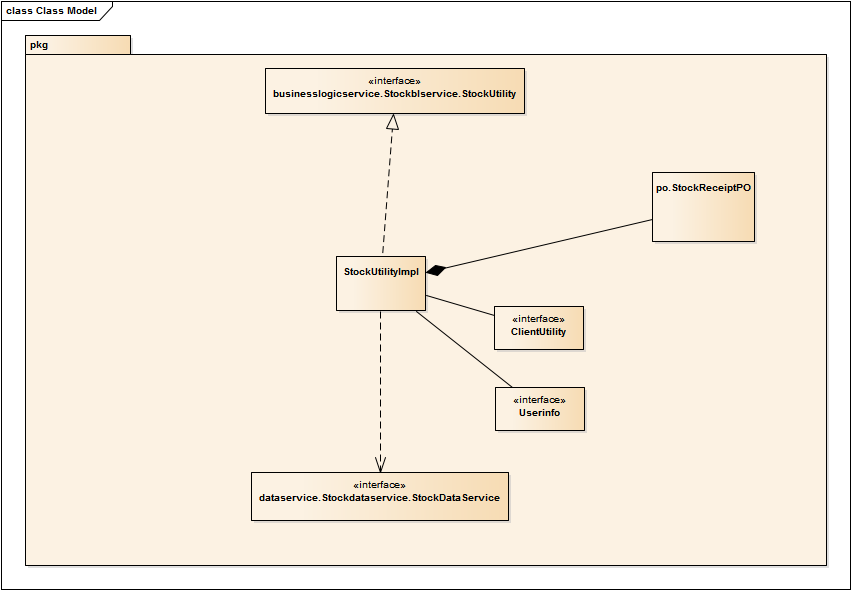
StockBL模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

StockBL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，我们会添加接口。比如展示层和业务逻辑层之间，我们添加businesslogicservice.Stockblservice.StockUtility接口。业务逻辑层和数据层之间添加dataservice.Stockdataservice.StockDataService接口。StockReceiptPO是作为进货退货单据的持久化对象被添加到设计模型中去的。

模块设计：



StockBL模块各个类的职责：

|  |  |
| --- | --- |
| 模块 | 职责 |
| LoginController | 负责实现对应于登陆界面所需要的服务 |
| StockUtility | 负责实现销售退货管理管理界面所需要的服务 |
| User | 系统用户的领域模型对象，拥有用户数据的账号和密码，可以解决登陆问题 |

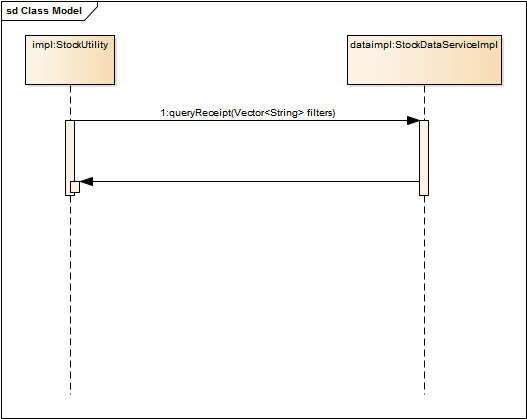
(3) 模块内部类的接口规范

StockUtility的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockUtility. makeReceipt | 语法 | public void makeReceipt(StockReceiptPO receipt) throws Exception |
| 前置条件 | 输入符合规范 |
| 后置条件 | 调用StockDataServiceImpl的makeReceipt方法 |
| StockUtility. queryReceipt | 语法 | public Vector<StockReceiptPO> queryReceipt(Vector<String> filters) throws Exception |
| 前置条件 | 输入符合规范 |
| 后置条件 | 调用StockDataServiceImpl的queryReceipt方法 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 功能 |
| StockDataService.makeReceipt(StockReceiptPO receipt) | 增加一条进货退货记录 |
| StockDataService.queryReceipt(Vector<String> filters) | 查询进货退货记录 |

(4) 业务逻辑层的动态模型



(5) 业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问别的业务逻辑由各自的控制器委托给不同的对象。