图片包含 游戏机, 赌场, 房间

描述已自动生成鲜果购买配送一站式解决方案

**面向对象分析及设计文档**

|  |  |
| --- | --- |
| 作者 | 孙嘉祎 202000800154  宋昕霓 202000800092  沐思遥 202000800109  黄欣灵 202000800082 |
| 单位 | 山东大学 |
| 指导教师 | 王文玉 |
| 完成日期 | 2022年5月20日 |

目录

CONTENTS

[**1.面向对象需求分析** 5](#_Toc104136274)

[1.1用例建模 5](#_Toc104136275)

[1.1.1子系统划分用例建模 5](#_Toc104136276)

[1.1.2库存管理子系统 7](#_Toc104136277)

[1.1.3销售管理子系统 8](#_Toc104136278)

[1.1.4用户管理子系统 9](#_Toc104136279)

[1.2初始类图 10](#_Toc104136280)

[1.2.1用到的外部实体 (参与者) 10](#_Toc104136281)

[1.2.2识别分析可能存在的类 11](#_Toc104136282)

[1.2.3初始类图 11](#_Toc104136283)

[1.3子系统状态图建模 14](#_Toc104136284)

[1.3.1库存管理子系统状态图 14](#_Toc104136285)

[1.3.2销售管理子系统状态图 15](#_Toc104136286)

[1.3.3用户管理子系统状态图 16](#_Toc104136287)

[1.4顺序图用例描述 17](#_Toc104136288)

[1.4.1库存管理顺序图 17](#_Toc104136289)

[1.4.2销售管理顺序图 20](#_Toc104136290)

[1.4.3用户管理顺序图 24](#_Toc104136291)

[**2.面向对象软件设计** 26](#_Toc104136292)

[2.1构建部署说明 26](#_Toc104136293)

[2.1.1技术架构 26](#_Toc104136294)

[2.1.2部署图 26](#_Toc104136295)

[2.1.3软件功能模块图 27](#_Toc104136296)

[2.1.4包图 30](#_Toc104136297)

[2.2详细类图 31](#_Toc104136298)

[2.2.1 详细类设计 32](#_Toc104136299)

[2.2.2完整类图（含包关系） 33](#_Toc104136300)

[2.2.3大类图一览 38](#_Toc104136301)

[2.3数据表设计 40](#_Toc104136302)

[2.3.1数据表设计 40](#_Toc104136303)

[2.3.2物理文件设计 44](#_Toc104136304)

[2.3.3约束 45](#_Toc104136305)

[2.3.4索引设计 46](#_Toc104136306)

[2.3.5建表SQL语句 47](#_Toc104136307)

[2.4接口设计描述 53](#_Toc104136308)

[2.4.1持久层接口 53](#_Toc104136309)

[2.4.2服务层接口 58](#_Toc104136310)

[2.4.3web层接口 61](#_Toc104136311)

[2.4.4前端API接口 65](#_Toc104136312)

[2.4.5界面类，外观类设计 73](#_Toc104136313)

[2.5活动图 78](#_Toc104136314)

[2.5.1配送业务逻辑 78](#_Toc104136315)

[2.5.2注册业务逻辑 79](#_Toc104136316)

[2.5.3登录业务逻辑 80](#_Toc104136317)

[2.5.4删除过期水果业务逻辑 82](#_Toc104136318)

[2.5.5出货业务逻辑 82](#_Toc104136319)

[2.5.6退货业务逻辑 83](#_Toc104136320)

**1.面向对象需求分析**

开发一款关于线上销售、配送水果的软件可以为买家和卖家的交易提供便利，节省排队时间，利于协助疫情管控。同时，作为计算机专业的学生，我们需要考虑软件在顾客使用时的易操作性和功能性以及商家管理时功能的全面性。我们小组需要这样一个富有意义的项目更深入地学习如何开发一款软件，将思维趋向软件工程师的思维方法。

在先前结构化分析设计当中，我们发现：传统的结构化方法存在描述复杂，不便于表达复杂的依赖关系等问题。面对日益复杂的软件架构，需要新的方法理念支撑软件设计工作。

面向对象的需求分析方法的核心是利用面向对象的概念和方法为软件需求建造模型。它包含面向对象风格的图形语言机制和用于指导需求分析的面向对象方法学。

因此我们将按照面向对象的方法，从需求分析到软件设计对整个系统进行重新的分析设计，以使得系统更加规范合理，简化开发复杂度。

1.1用例建模

用例模型是系统既定功能及系统环境的模型，它可以作为客户和开发人员之间的契约。用例是贯穿整个系统开发的一条主线。同一个用例模型即为需求工作流程的结果，可当作分析设计工作流程以及测试工作流程的输入使用。

用例模型最重要的作用是将系统行为传达给客户或最终用户。因此，模型必须易于理解。

用例图是从需求分析报告到软件系统设计的第一步，也是系统整个分析过程中最重要的图，它的改变将影响到其它图，用例建模贯穿整个软件开发的过程。

1.1.1子系统划分用例建模

* 分析确定系统的执行者

系统的主要功能是：包括注册登录，销售管理和库存管理三大管理功能。通过分析确定系统的执行者(角色)我们确定了用户，管理员，供应商，配送员四类实体。

* 确定用例

我们设计了库存管理，销售管理，注册登录三大主要的子系统。其中库存管理负责控制各类货物出库入库，销售管理处理具体的财务，注册登录则是完成用户账户体系。

* 用例分解

本着自顶向下，逐步细化，提高抽象层次的几大原则，我们先给出顶层用例，也就是子系统的划分，接下来将通过用例分解，并分配给不同的角色完成详细的用例建模。

下面是总体用例图：



下面是主要用例概述表：

| 参与者 | 用例 | 说明 |
| --- | --- | --- |
| 用户 | 登录注册 | 为实现精准服务，任何使用者都需要注册登录 |
| 购买商品 |  |
| 退货 | 一定条件下允许退货 |
| 评论 | 评价每一次购买 |
| 查看购买记录 | 便于用户记录喜欢的商品方便下次购买，记录消费信息便于用户合理掌握消费习惯 |
| 管理员 | 进货 | 向仓库补充内容，确保商品是最新的 |
| 设置供应商 | 管理供应链，确保获得新鲜的产品 |
| 设置配送 | 调度人员，将水果送到用户手中 |
| 查看评论 | 查看用户留下的想法 |
| 删除过期食品 | 及时清理过期滞销的商品，减少亏损并避免意外事件发生 |
| 缺货提醒 | 随时查看哪些商品需要注意 |
| 销量查询 | 掌握财务动向和用户习惯，便于精准调整营销策略，吸引更多用户 |

1.1.2库存管理子系统



1.1.3销售管理子系统



1.1.4用户管理子系统



1.2初始类图

1.2.1用到的外部实体 (参与者)

外部实体，又称参与者，指与系统直接或间接进行交互的人，事，物甚至子系统，它包括：

* 使用系统的用户
* 系统与既有系统的链接
* 系统与硬件设备的交互
* 系统与第三方平台（如政府机构，网上银行）的交互

考虑到本程序是一个单体应用，自成生态闭环，且由于条件受限，并没有第三方参与，因此参与者有一下几类

* 用户：注册登录使用系统，选购水果并购买下单要求配送，一定条件下可以要求退货退单，并获得相应的积分补偿，积分在平台内可按照一定规则在购买水果时换取折扣，用户购买水果时也会随机获取少量积分作为活动礼品（取决于管理员的设置）。用户可随时对系统给出反馈评价
* 管理员：管理整个系统的运行，统计分析用户行为以改进服务质量，负责记录管理从供应商进货，处理过期滞销水果，查询管理账目记录，安排配送任务，处理退换货请求和用户纠纷，查看分析用户反馈。
* 供应商：为系统运用方供应商品，通过与管理员签约在系统中登记注册，并在每次供货活动中履行被登记的义务。

· 配送员：完全被管理员控制，被安排配送任务，并将完成情况汇报给管理员便于记录。

1.2.2识别分析可能存在的类

在软件分析阶段，我们仅对实体类进行讨论，有以下几种：

| 类名 | 对应于 |
| --- | --- |
| User | 用户 |
| Manager | 管理员 |
| Category | 商品种类 |
| Good | 商品（水果） |
| Supplier | 供应商 |
| DeliveryMan | 配送员 |
| Order | 订单 |
| OrderDetail | 订单细节 |
| OrderHsitory | 订单历史记录 |
| OrderDetailHistory | 历史订单细节记录 |
| Delivery | 配送记录 |
| ReturnGood | 退货记录 |
| Supply | 供货记录 |
| Comment | 用户反馈 |

1.2.3初始类图

由于控制类服务类涉及到具体的软件设计，因此这里仅体现实体类及其关系。

**备注：由于访问器，修改器，构造器，toString等方法使用极其频繁且内容基本单调重复，因此在本图及之后的图中一律省略不写。并不意味着没有这些方法！**

请将屏幕横过来查看下图：

文本

低可信度描述已自动生成

1.3子系统状态图建模

状态图(Statechart Diagram)是描述一个实体基于事件反应的[动态](https://baike.baidu.com/item/%E5%8A%A8%E6%80%81/585005)行为，显示了该实体如何根据当前所处的状态对不同的事件做出反应。为了描述较为复杂的用例，我们对子系统给出了相应的状态图。

我们的状态图案子系统划分包括出货状态图、退货状态图、配送状态图、下单状态图、登陆状态图和注册状态图。

1.3.1库存管理子系统状态图

1.3.1.1出货



1.3.1.2退货



1.3.2销售管理子系统状态图

1.3.2.1配送



1.3.2.2下单



1.3.3用户管理子系统状态图





1.4顺序图用例描述

顺序图是将交互关系表示为一个二维图。纵向是时间轴，时间沿竖线向下延伸。横向轴代表了在协作中各独立对象的类元角色。顺序图由类图衍生而来，描述具体的工作流程。

顺序图依旧是按照子系统划分，功能有查询库存商品信息、修改库存进货出货、缺货提醒、下单、退货、管理购物车、评论反馈、登陆注册几个功能。

1.4.1库存管理顺序图

1.4.1.1查询库存，商品信息



1.4.1.2修改库存，进货出货



1.4.1.3缺货提醒功能



1.4.2销售管理顺序图

1.4.2.1下单



1.4.2.2退货



1.4.2.3管理购物车



1.4.2.4评价反馈



1.4.3用户管理顺序图



**2.面向对象软件设计**

面向对象方法是当今程序设计最常用的方法，指导开发活动，是建立在“对象”概念基础上的方法学，简称OO( Object-Oriented)方法。面向对象方法的本质是主张参照人们认识一个现实系统的方法，完成分析、设计与实现一个软件系统，提倡用人类在现实生活中常用的思维方法来认识和理解描述客观事物，强调最终建立的系统能映射问题域，使得系统中的对象，以及对象之间的关系能够如实地反映问题域中固有的事物及其关系。

我们将围绕着问题域，数据管理，任务管理，人机交互四大主题进行分析，确定类结构以及类之间的关系。完成软件设计方案

2.1构建部署说明

2.1.1技术架构

本项目以网站形式呈现，采用前后端分离和模板引擎渲染技术相结合的开发方式完成。

前端采用广泛使用的vue.js界面框架，具有强大的自适应排版能力，后期还可以根据需要轻松打包为Android应用程序，使网站可触及的用户进一步增加。前后端交互方面采用基于Ajax技术的Axios框架，对常用功能很好的封装。

后端以最著名的Spring Boot为核心框架，高效率封装web底层基础设施，使开发者专注于业务逻辑本身的实现，辅以MyBatis简化持久层开发，实现后端和数据库分离开发。同时针对后台管理部分的特点，使用Spring Boot自带的后端模板渲染引擎Thymeleaf，简化后台增删改查页面的开发。

数据库软件系统采用免费开源，性能优异，稳定可靠的MySQL数据库软件，这是中小型网站甚至某些大型互联网项目最常用的经典数据库软件系统。

2.1.2部署图

本着“移动为先云为先”的战略思想，我们所有的东西都将基于云服务，优先适配移动设备，优化手机体验。



图示, 示意图

描述已自动生成

2.1.3软件功能模块图

为了便于理解，我们依旧给出了结构化的软件功能模块图，如图所示：





2.1.4包图

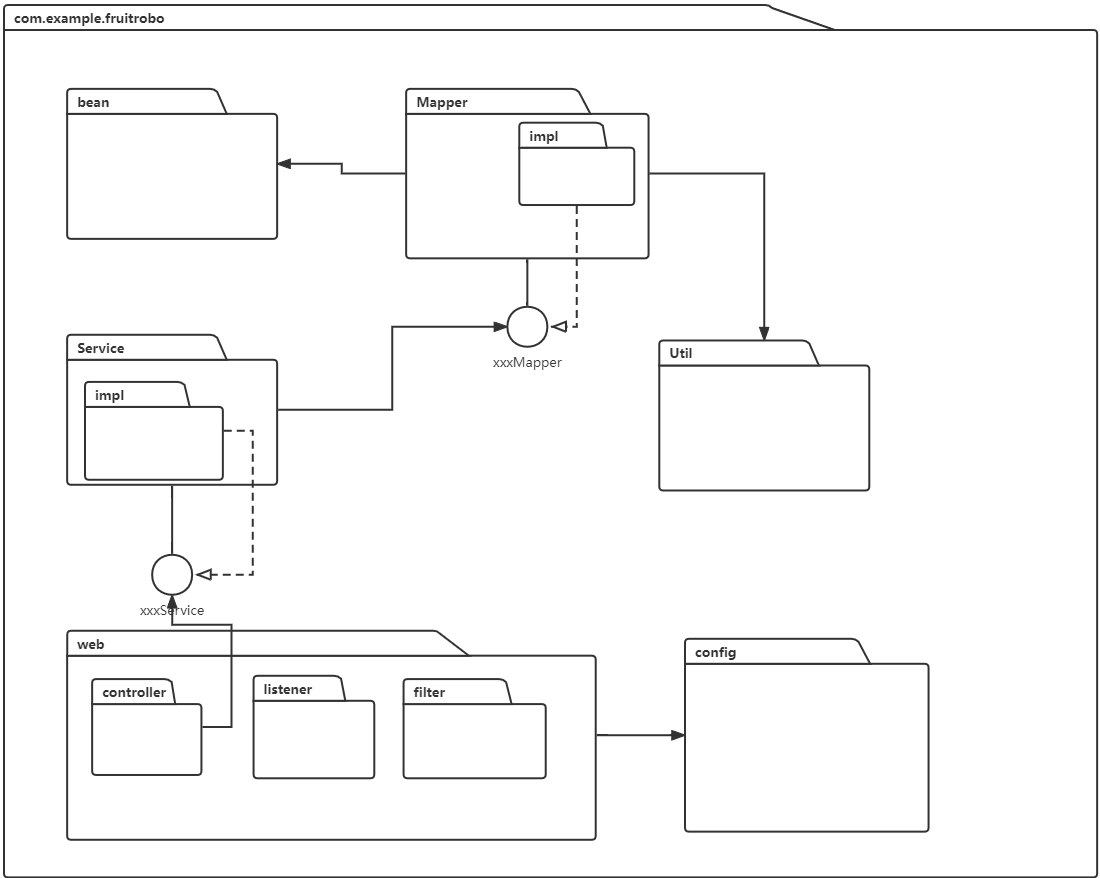
本项目遵循搭建网站常见的三层架构模式：即将网站后台程序分为三大部分——负责驱动数据库，直接操作数据的持久层；处理具体业务逻辑，将现实生活中的工作流表达为代码的服务层；和与前台交互，传输数据呈现内容的表示层（web层）。同时将常用的重复次数多的公共代码提取出来，设计成实用工具程序放在单独一个工具层以提高抽象层次和代码复用程度。

各层之间遵循接口与实现分离的原则，通过抽象公共接口实现包之间的通信。

实体类采用EJB规范，可配合Spring容器，Lombok实用工具程序简化开发，减少重复枯燥代码的编写。

包名、类名均遵循主流的习惯命名法，尽可能做到见名知意。简化同学之间协作理解代码的难度。

具体的包图如图所示：



2.2详细类图

本程序的类设计遵循以下规范：

· 类名遵循首字母大写命名法

· 属性名，方法名，对象名，变量名遵循驼峰命名法

· 包名遵循小写字母命名法和域名倒置命名法。

· 面向接口编程：dao层和service层采取接口与实现分离的原则，dao层所有接口在对应的资源目录下有XML文件实现，service接口在impl子包实现

· 数据库部分用常见的设计模式如工厂模式，进一步规范代码

· 不允许无任何说明的常量字面值出现，所有常量，枚举都预先进行规定。

2.2.1 详细类设计

2.2.1.1业务逻辑处理类

业务逻辑处理类直接与数据库打交道，实现最基础的增删改查。

有以下几种：

（1）持久层

| 类名 | 说明 |
| --- | --- |
| UerMapper | MyBatis框架要求的提供Mapper接口，作为对数据库表格读写的途径 |
| ManagerMapper |
| OrderMapper |
| GoodMapper |
| SupplyMapper |
| ReturnGoodMapper |
| CommentMapper |
| DeliveryMapper |

（2）服务层

| 类名 | 说明 |
| --- | --- |
| UserService | 注册登录，找回密码，修改信息 |
| ManagerService |
| GoodService | 添加修改货物种类，管理进货 |
| SupplyService |
| SaleService | 负责卖货退货，账目统计 |
| ReturnService |
| DeliveryService | 负责处理配送任务 |
| CommentService | 处理用户反馈 |

2.2.1.2控制类

由于网站没有界面类，因此需要合适的控制器负责连接网页和程序代码

（1）控制器

| 类名 | 说明 |
| --- | --- |
| UserLoginController | 接管注册登录的前台请求 |
| ManagerLoginController |
| UserMainController | 接管主页面 |
| ManagerMainController |
| SaleController | 接管销售业务 |
| CommentController | 接管用户反馈 |
| GoodMapper | 接管库存管理业务 |
| DeliveryMapper | 接管配送业务 |

（2）拦截器

| 类名 | 说明 |
| --- | --- |
| UserLoginInterceptor | 系统的看门狗，负责鉴权把关，防止未经授权的意外访问 |
| ManagerLoginInterceptor |

2.2.2完整类图（含包关系）

2.2.2.1 bean包

图示, 示意图

描述已自动生成

2.2.2.2mapper包

图示, 示意图

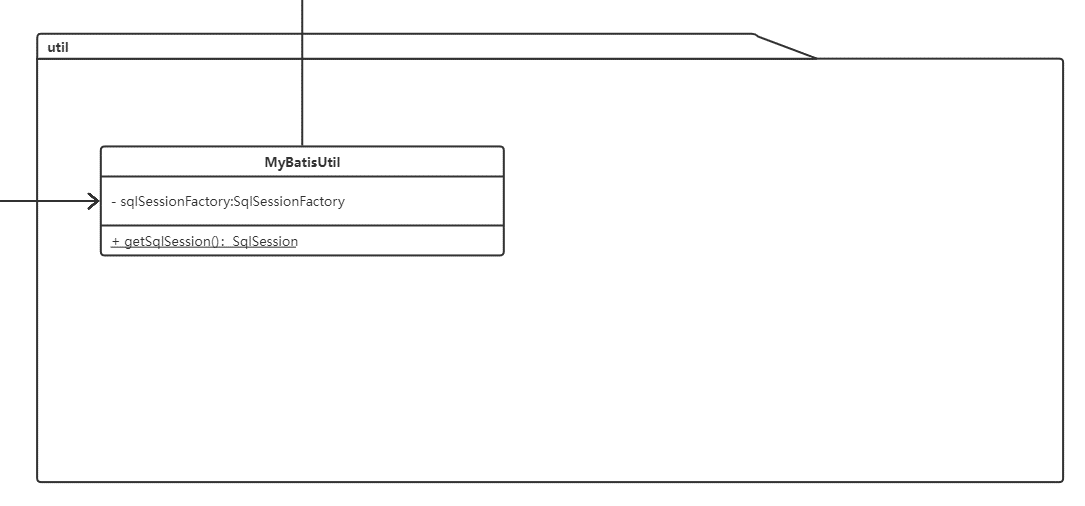
描述已自动生成

2.2.2.3service包

图示, 示意图

描述已自动生成

2.2.2.4util包



2.2.2.5web包

图示

描述已自动生成

2.2.3大类图一览

点击下面这个图标可以打开高清附件（实在不好意思图太大了）



图示

低可信度描述已自动生成

2.3数据表设计

本数据库遵循以下规范：

* 不允许使用物理外键，一切外键概念必须在应用层解决。 因为每次级联delete或update的时候，都要级联操作相关的外键表，不论有没有这个必要，由其在高并发的场景下，这会导致性能瓶颈。因此后面的约束仅仅是逻辑上的，在应用层实现而不是写入数据库。
* 数据库命名遵循下划线规范，这和驼峰命名法是一脉相承的。
* 对于和关键字重名的标识符，加适当的后缀区分。

2.3.1数据表设计

2.3.1.1用户信息表

| 主键 | 外键 | 字段名称 | 数据类型 | 允许NULL值 |
| --- | --- | --- | --- | --- |
| √ |  | 用户ID | Int | 否 |
|  |  | 用户昵称 | Nvarchar(10) | 是 |
|  |  | 邮箱 | Nvarchar(20) | 否 |
|  |  | 密码 | Nvarchar(10) | 否 |
|  |  | 积分 | Int | 是 |

2.3.1.2管理员信息表

| 主键 | 外键 | 字段名称 | 数据类型 | 允许NULL值 |
| --- | --- | --- | --- | --- |
| √ |  | 管理员ID | Int | 否 |
|  |  | 管理员昵称 | Nvarchar(10) | 是 |
|  |  | 邮箱 | Nvarchar(20) | 否 |
|  |  | 密码 | Nvarchar(10) | 否 |
|  |  | 权限等级 | Smallint | 否 |

2.3.1.3商品种类信息表

| 主键 | 外键 | 字段名称 | 数据类型 | 允许NULL值 |
| --- | --- | --- | --- | --- |
| √ |  | 种类ID | Int | 否 |
|  |  | 种类名称 | Nvachar(10) | 是 |
|  |  | 备注 | Nvarchar(30) | 是 |

2.3.1.4商品信息表

| 主键 | 外键 | 字段名称 | 数据类型 | 允许NULL值 |
| --- | --- | --- | --- | --- |
| √ |  | 商品ID | Int | 否 |
|  | √ | 种类ID | Int | 是 |
|  |  | 库存数量 | Int | 是 |
|  |  | 品名 | Nvarchar(10) | 否 |
|  |  | 保质期 | Time | 是 |
|  |  | 单价 | float | 否 |
|  |  | 备注 | Nvarchar(30) | 是 |

2.3.1.5供应商信息表

| 主键 | 外键 | 字段名称 | 数据类型 | 允许NULL值 |
| --- | --- | --- | --- | --- |
| √ |  | 供应商ID | Int | 否 |
|  |  | 供应商名称 | Nvarchar(10) | 否 |
|  |  | 联系方式 | Nvarchar(20) | 否 |
|  |  | 负责人姓名 | Nvarchar(10) | 是 |

2.3.1.6配送员信息表

| 主键 | 外键 | 字段名称 | 数据类型 | 允许NULL值 |
| --- | --- | --- | --- | --- |
| √ |  | 配送员ID | Int | 否 |
|  |  | 配送员姓名 | Nvarchar(10) | 是 |
|  |  | 联系方式 | Nvarchar(20) | 否 |

2.3.1.7订单信息表

| 主键 | 外键 | 字段名称 | 数据类型 | 允许NULL值 |
| --- | --- | --- | --- | --- |
| √ |  | 订单ID | Int | 否 |
|  | √ | 用户ID | Int | 否 |
|  | √ | 配送员ID | Int | 否 |
|  |  | 下单时间 | Datetime(7) | 否 |
|  |  | 折扣比例 | Float | 是 |
|  |  | 支付方式 | Nvarchar(10) | 是 |
|  |  | 支付单号 | 尚不明确 | 是 |
|  |  | 收货地址 | Nvarchar(30) | 否 |

2.3.1.8订单细节信息表

| 主键 | 外键 | 字段名称 | 数据类型 | 允许NULL值 |
| --- | --- | --- | --- | --- |
| √ | √ | 订单ID | Int | 否 |
| √ | √ | 商品ID | Int | 否 |
|  |  | 数量 | Int | 否 |

2.3.1.9订单历史记录表

| 主键 | 外键 | 字段名称 | 数据类型 | 允许NULL值 |
| --- | --- | --- | --- | --- |
| √ |  | 订单ID | Int | 否 |
|  | √ | 用户ID | Int | 否 |
|  | √ | 配送员ID | Int | 否 |
|  |  | 下单时间 | Datetime(7) | 否 |
|  |  | 折扣比例 | Float | 是 |
|  |  | 支付方式 | Nvarchar(5) | 是 |
|  |  | 支付单号 | 尚不明确 | 是 |
|  |  | 收货地址 | Nvarchar(30) | 否 |
|  |  | 配送用时 | Time | 否 |

2.3.1.10订单细节历史记录表

| 主键 | 外键 | 字段名称 | 数据类型 | 允许NULL值 |
| --- | --- | --- | --- | --- |
| √ | √ | 订单ID | Int | 否 |
| √ | √ | 商品ID | Int | 否 |
|  |  | 数量 | Int | 否 |

2.3.1.11退货记录表

| 主键 | 外键 | 字段名称 | 数据类型 | 允许NULL值 |
| --- | --- | --- | --- | --- |
| √ |  | 流水号 | Int | 否 |
|  | √ | 用户ID | Int | 否 |
|  | √ | 订单ID | Int | 否 |
|  | √ | 商品ID | Int | 否 |
|  |  | 退货数量 | Int | 否 |
|  |  | 处理时间 | Datetime(7) | 否 |
|  |  | 补偿积分值 | Int | 是 |

2.3.1.12供应商供货记录表

| 主键 | 外键 | 字段名称 | 数据类型 | 允许NULL值 |
| --- | --- | --- | --- | --- |
| √ | √ | 商品ID | Int | 否 |
| √ | √ | 供应商ID | Int | 否 |
| √ |  | 运抵时间 | Datetime(7) | 否 |
|  |  | 发货时间 | Datetime(7) | 否 |
|  |  | 运输列车车次 | Nvarchar(10) | 否 |
|  |  | 备注 | Nvarchar(64) | 否 |

2.3.1.13用户评价记录表

| 主键 | 外键 | 字段名称 | 数据类型 | 允许NULL值 |
| --- | --- | --- | --- | --- |
| √ | √ | 用户ID | Int | 否 |
| √ | √ | 订单ID | Int | 否 |
|  |  | 评价时间 | Datetime(7) | 否 |
|  |  | 评价星级 | Smallint | 否 |
|  |  | 评价内容 | Nvarchar(255) | 是 |

2.3.1.14销量表

| 主键 | 外键 | 字段名称 | 数据类型 | 允许NULL值 |
| --- | --- | --- | --- | --- |
| √ | √ | 商品ID | Int | 否 |
|  |  | 商品名称 | Nvarchar(10) | 否 |
|  |  | 销量 | Int | 否 |

2.3.2物理文件设计

| 属性 | 值 |
| --- | --- |
| 名称 | FruitRobo |
| 文件组数量 | 1 |
| 编码 | UTF-8 |
| 数据库引擎 | MySQL InnoDB |
| 主文件逻辑名称 | FruitRobo |
| 初始大小 | 10MB |
| 最大大小 | 1GB |
| 增长速度 | 1MB |
| 日志文件逻辑名称 | FruitRobo\_Log |
| 初始大小 | 1MB |
| 最大大小 | 10MB |
| 增长速度 | 512KB |

2.3.3约束

注意：此类约束均为逻辑概念，不在数据库物理设计当中体现。

2.3.3.1外键约束

| 名称 | 表 | 列 | 被参照表 | 被参照列 |
| --- | --- | --- | --- | --- |
| FK\_PRODUCT\_1 | 商品单信息表 | 种类ID | 商品种类信息表 | 用户ID |
| FK\_ORDER\_1 | 订单信息表 | 用户ID | 用户信息表 | 用户ID |
| FK\_ORDER\_2 | 订单信息表 | 配送员ID | 配送员信息表 | 配送员ID |
| FK\_ORDERDETAIL\_1 | 订单细节信息表 | 订单ID | 订单信息表 | 订单ID |
| FK\_ORDERDETAIL\_2 | 订单细节信息表 | 商品ID | 商品信息表 | 商品ID |
| FK\_HISORDER\_1 | 订单信息表 | 用户ID | 用户信息表 | 用户ID |
| FK\_HISORDER\_2 | 订单信息表 | 配送员ID | 配送员信息表 | 配送员ID |
| FK\_HISDETAIL\_1 | 订单细节信息表 | 订单ID | 订单信息表 | 订单ID |
| FK\_HISDETAIL\_2 | 订单细节信息表 | 商品ID | 商品信息表 | 商品ID |
| FK\_TUIHUO\_1 | 退货记录表 | 用户ID | 用户信息表 | 用户ID |
| FK\_TUIHUO\_2 | 退货记录表 | 订单ID | 订单信息表 | 订单ID |
| FK\_TUIHUO\_3 | 退货记录表 | 商品ID | 商品信息表 | 商品ID |
| FK\_SONGHUO\_1 | 供应商供货信息表 | 商品ID | 商品信息表 | 商品ID |
| FK\_SONGHUO\_2 | 供应商供货信息表 | 供应商ID | 供应商信息表 | 供应商ID |
| FK\_COMMENT\_1 | 用户评价记录表 | 用户ID | 用户信息表 | 用户ID |
| FK\_COMMENT\_2 | 用户评价记录表 | 订单ID | 订单记录表 | 订单ID |

2.3.3.2用户自定义完整性约束

| 名称 | 表 | 列 | 描述 |
| --- | --- | --- | --- |
| C1 | 用户信息表 | 用户昵称 | 长度>=5，仅包含字母数字 |
| C2 | 用户信息表 | 邮箱 | 包含@字符的邮箱地址 |
| C3 | 用户信息表 | 密码 | 包含大小写字母和数字，长度>=5 |
| C4 | 商品信息表 | 单价 | 非负数 |
| C5 | 商品信息表 | 库存 | 非负数 |
| C6 | 供应商信息表 | 联系方式 | QQ号或包含@字符的邮箱地址 |
| C7 | 配送员信息表 | 联系方式 | QQ号或包含@字符的邮箱地址 |
| C8 | 订单信息表 | 折扣比例 | >=0且<1的浮点数 |
| C9 | 订单信息表 | 支付方式 | “微信”，“支付宝”，“银行卡”，“PyaPal”，“Github上点赞代替付款”的其中一个值 |
| C10 | 订单细节记录表 | 数量 | 正整数 |
| C11 | 订单历史记录表 | 折扣比例 | >=0且<1的浮点数 |
| C12 | 订单历史记录表 | 支付方式 | “微信”，“支付宝”，“银行卡”，“PyaPal”，“Github上点赞代替付款”的其中一个值 |
| C13 | 订单细节历史记录表 | 数量 | 正整数 |
| C14 | 订单细节历史记录表 | 折扣比例 | >=0且<1的浮点数 |
| C15 | 退货记录表 | 退货数量 | <=相应订单细节数量 |
| C16 | 退货记录表 | 补偿积分 | 非负整数 |
| C17 | 供应商供货记录表 | 发货时间 | <=运抵时间 |
| C18 | 供应商供货记录表 | 运输列车车次 | 1-5位数字，可选以K，T，Z，L，Y一个字母开头 |
| C19 | 管理员信息表 | 权限等级 | 1-3的正整数 |

2.3.4索引设计

| 名称 | 表 | 列 | 升序/降序 |
| --- | --- | --- | --- |
| L1 | 订单信息表 | 下单时间 | 降序 |
| L2 | 订单历史记录表 | 下单时间 | 降序 |
| L3 | 退货记录表 | 处理时间 | 降序 |
| L4 | 供应商供货记录 | 发货时间 | 降序 |
| L5 | 供应商供货记录 | 运抵时间 | 降序 |
| L6 | 用户评价记录 | 评价时间 | 降序 |
| L7 | 商品种类 | 保质期 | 升序 |

2.3.5建表SQL语句

|  |  |
| --- | --- |
| fruit\_robo.sql | SQL |
| /\*   Navicat Premium Data Transfer   Source Server         : myconn   Source Server Type    : MySQL   Source Server Version : 80028   Source Host           : localhost:3306   Source Schema         : fruit\_robo   Target Server Type    : MySQL   Target Server Version : 80028   File Encoding         : 65001   Date: 16/05/2022 20:50:41  \*/  SET NAMES utf8mb4;  SET FOREIGN\_KEY\_CHECKS = 0;  -- ----------------------------  -- Table structure for category  -- ----------------------------  DROP TABLE IF EXISTS `category`;  CREATE TABLE `category`  (    `caid` int NOT NULL AUTO\_INCREMENT,    `caname` varchar(10) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    `canote` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    PRIMARY KEY (`caid`) USING BTREE  ) ENGINE = InnoDB AUTO\_INCREMENT = 16 CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci ROW\_FORMAT = Dynamic;  -- ----------------------------  -- Table structure for comment\_f  -- ----------------------------  DROP TABLE IF EXISTS `comment\_f`;  CREATE TABLE `comment\_f`  (    `uid` int NOT NULL,    `oid` int NOT NULL,    `comdate` datetime NULL DEFAULT NULL,    `comlevel` varchar(5) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    `comcontent` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    PRIMARY KEY (`uid`, `oid`) USING BTREE  ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci ROW\_FORMAT = Dynamic;  -- ----------------------------  -- Table structure for deliveryman  -- ----------------------------  DROP TABLE IF EXISTS `deliveryman`;  CREATE TABLE `deliveryman`  (    `dlid` int NOT NULL AUTO\_INCREMENT,    `dlname` varchar(10) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    `dlcontact` varchar(20) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    PRIMARY KEY (`dlid`) USING BTREE  ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci ROW\_FORMAT = Dynamic;  -- ----------------------------  -- Table structure for good  -- ----------------------------  DROP TABLE IF EXISTS `good`;  CREATE TABLE `good`  (    `gid` int NOT NULL AUTO\_INCREMENT,    `caid` int NULL DEFAULT NULL,    `ginventory` int UNSIGNED NULL DEFAULT NULL,    `gname` varchar(10) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    `glast` int NULL DEFAULT NULL,    PRIMARY KEY (`gid`) USING BTREE,    INDEX `c1`(`caid` ASC) USING BTREE  ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci ROW\_FORMAT = Dynamic;  -- ----------------------------  -- Table structure for manager  -- ----------------------------  DROP TABLE IF EXISTS `manager`;  CREATE TABLE `manager`  (    `mid` int NOT NULL AUTO\_INCREMENT,    `mname` varchar(10) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    `memail` varchar(20) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    `mpassword` varchar(10) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    `mprem` int NULL DEFAULT NULL,    PRIMARY KEY (`mid`) USING BTREE  ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci ROW\_FORMAT = Dynamic;  -- ----------------------------  -- Table structure for order\_f  -- ----------------------------  DROP TABLE IF EXISTS `order\_f`;  CREATE TABLE `order\_f`  (    `oid` int NOT NULL AUTO\_INCREMENT,    `uid` int NULL DEFAULT NULL,    `otime` datetime NULL DEFAULT NULL,    `odiscount` float NULL DEFAULT NULL,    `opaymethod` varchar(10) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    `oaddress` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    PRIMARY KEY (`oid`) USING BTREE,    INDEX `c2`(`uid` ASC) USING BTREE  ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci ROW\_FORMAT = Dynamic;  -- ----------------------------  -- Table structure for order\_f\_history  -- ----------------------------  DROP TABLE IF EXISTS `order\_f\_history`;  CREATE TABLE `order\_f\_history`  (    `hoid` int NOT NULL,    `huid` int NULL DEFAULT NULL,    `hotime` datetime NULL DEFAULT NULL,    `hodiscount` float NULL DEFAULT NULL,    `hopaymethod` varchar(10) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    `hoaddress` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    `hodltime` time NULL DEFAULT NULL,    PRIMARY KEY (`hoid`) USING BTREE  ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci ROW\_FORMAT = Dynamic;  -- ----------------------------  -- Table structure for orderdetail  -- ----------------------------  DROP TABLE IF EXISTS `orderdetail`;  CREATE TABLE `orderdetail`  (    `oid` int NOT NULL,    `gid` int NOT NULL,    `odquantity` int NULL DEFAULT NULL,    PRIMARY KEY (`oid`, `gid`) USING BTREE,    INDEX `c5`(`gid` ASC) USING BTREE  ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci ROW\_FORMAT = Dynamic;  -- ----------------------------  -- Table structure for orderdetail\_history  -- ----------------------------  DROP TABLE IF EXISTS `orderdetail\_history`;  CREATE TABLE `orderdetail\_history`  (    `hoid` int NOT NULL,    `hgid` int NOT NULL,    `hodquantity` int NULL DEFAULT NULL,    PRIMARY KEY (`hoid`, `hgid`) USING BTREE  ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci ROW\_FORMAT = Dynamic;  -- ----------------------------  -- Table structure for purchase  -- ----------------------------  DROP TABLE IF EXISTS `purchase`;  CREATE TABLE `purchase`  (    `gid` int NOT NULL,    `suid` int NOT NULL,    `pusendtime` datetime NOT NULL,    `puarrivetime` datetime NULL DEFAULT NULL,    `putrainno` varchar(10) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    `pucomment` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    PRIMARY KEY (`gid`, `suid`, `pusendtime`) USING BTREE  ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci ROW\_FORMAT = Dynamic;  -- ----------------------------  -- Table structure for salesreturn  -- ----------------------------  DROP TABLE IF EXISTS `salesreturn`;  CREATE TABLE `salesreturn`  (    `srtid` int NOT NULL AUTO\_INCREMENT,    `uid` int NULL DEFAULT NULL,    `oid` int NULL DEFAULT NULL,    `gid` int NULL DEFAULT NULL,    `stttime` time NULL DEFAULT NULL,    `srtpoint` int NULL DEFAULT NULL,    PRIMARY KEY (`srtid`) USING BTREE  ) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci ROW\_FORMAT = Dynamic;  -- ----------------------------  -- Table structure for supplier  -- ----------------------------  DROP TABLE IF EXISTS `supplier`;  CREATE TABLE `supplier`  (    `suid` int NOT NULL AUTO\_INCREMENT,    `suname` varchar(10) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    `suofficer` varchar(20) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    `sucontact` varchar(20) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    PRIMARY KEY (`suid`) USING BTREE  ) ENGINE = InnoDB AUTO\_INCREMENT = 10 CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci ROW\_FORMAT = Dynamic;  -- ----------------------------  -- Table structure for user\_f  -- ----------------------------  DROP TABLE IF EXISTS `user\_f`;  CREATE TABLE `user\_f`  (    `uid` int NOT NULL AUTO\_INCREMENT,    `uname` varchar(10) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    `uemail` varchar(20) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    `upassword` varchar(10) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL DEFAULT NULL,    `upoint` int NULL DEFAULT NULL,    PRIMARY KEY (`uid`) USING BTREE  ) ENGINE = InnoDB AUTO\_INCREMENT = 1 CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_0900\_ai\_ci ROW\_FORMAT = Dynamic;  SET FOREIGN\_KEY\_CHECKS = 1; | |

2.4接口设计描述

此部分从编程语言接口和API接口两个层面说明分析本程序当中用到的接口以及其功能和特性。

后端接口遵循常见的命名规范。尽可能使用简单变量和类对象传递数据，减少耦合度较高的场景。

前端API接口尽可能遵循RESTFUL风格，这是微服务理念下的标准API风格，通过路径变量和请求方式区分不同的服务，有利于一致性和统一处理。

2.4.1持久层接口

接口层次关系概述图：

图示

描述已自动生成

接口名称：UserMapper

接口所含方法：

| 修饰符 | 方法名 | 参数类型 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | addUser | User | int |  |
| public | updateUser | User | int |  |
| public | deleteUser | int | int |  |
| public | findUserById | int | User |  |
| public | getUserList | 无 | List<User> |  |

接口名称：ManagerMapper

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | addManager | Manager | int |  |
| public | modifyManager | Manager | Manager |  |
| public | deleteManager | int | Manager |  |
| public | findUserBy | String | List<User> |  |

接口名称：CategoryMapper

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | addCategory | Category | int |  |
| public | updateCategory | Category | int |  |
| public | findCategoryById | int | Category |  |
| public | deleteCategory | int | int |  |

接口名称：GoodMapper

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | addGood | Good | int |  |
| public | updateGood | Good | int |  |
| public | findGoodById | int | Good |  |
| public | deleteGood | int | int |  |

接口名称：SupplierMapper

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | addSupplier | Supplier | int |  |
| public | updateSupplier | Supplier | int |  |
| public | findSupplierById | int | Supplier |  |
| public | deleteSupplier | int | int |  |

接口名称：OrderMapper

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | addOrder | Order | int |  |
| public | updateOrder | Order | int |  |
| public | findOrderById | int | Order |  |
| public | deleteOrder | int | int |  |

接口名称：OrderDetailMapper

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | addOrderDetail | OrderDetail | int |  |
| public | updateOrderDetail | OrderDetail | int |  |
| public | findOrderDetailById | int | OrderDetail |  |
| public | deleteOrderDetail | int | int |  |

接口名称：ReturnMapper

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | addReturn | Return | int |  |
| public | updateReturn | Return | int |  |
| public | findReturnById | int | Return |  |
| public | deleteReturn | int | int |  |

接口名称：DeliveryManMapper

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | addDeliveryMan | DeliveryMan | int |  |
| public | updateDeliveryMan | DeliveryMan | int |  |
| public | findDeliveryManById | int | DeliveryMan |  |
| public | deleteDeliveryMan | int | int |  |

接口名称：PurchaseMapper

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | addPurchase | Purchase | int |  |
| public | updatePurchase | Purchase | int |  |
| public | findPurchaseById | int | Purchase |  |
| public | deletePurchase | int | int |  |

2.4.2服务层接口

接口层次关系概述图

图片包含 图形用户界面

描述已自动生成

接口名称：UserService

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | login | AbstractUser | AbstractUser |  |
| public | Register | AbstractUser | void |  |
| public | ChangePassword | AbstractUser | void |  |

接口名称：goodsService

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | addCategory | Category | void |  |
| public | deleteCategory | String | void |  |
| public | ChangeGoodQuantity | Integer,Integer | void |  |
| public | ChangeGoodPrice | Integer,float | void |  |
| public | ChangeGoodBzq | Integer,Time | void |  |
| public | getGoodsOutOfDate | Integer | Time |  |

接口名称：DeliveryManService

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | addDeliveryMan | DeliveryMan | void |  |
| public | deleteDeliveryMan | Integer | void |  |
| public | changedeliveryMethod | Integer,String | void |  |

接口名称：CommentService

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | deleteComment | Integer | void |  |
| public | addComment | String | void |  |

接口名称：OrderService

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | makeOrder | Order,OrderDetail | Order |  |
| public | confirmRecieve | Integer | void |  |
| public | evaluateReturnGoods | returnGoods | void |  |

接口名称：purchaseService

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | addPurchase | purchase | void |  |
| public | deletePurchase | Integer | void |  |
| public | changeQua | Integer,Integer | void |  |
| public | changeTrans | Integer,String | void |  |

接口名称：SupplierService

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | deleteSupplier | Integer | void |  |
| public | addSupplier | Supplier | void |  |
| public | changeName | Integer,String | void |  |
| public | changeConnetMethod | Integer,String | void |  |
| public | changeAddrs | Integer,String | void |  |

2.4.3web层接口

接口名称：LoginPage

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | LoginPage | 无 | String |  |
| public | mainPage | Manager,Model,HttpSession | String |  |
| public | avoidRefresh | Model,HttpSession | String |  |
| public | logOut | HttpSession | String |  |

接口名称：RegisterPage

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | RegisterPage | 无 | String |  |
| public | mainPage | Manager,Model,HttpSession | String |  |
| public | avoidRefresh | Model,HttpSession | String |  |
| public | JumpLogin | HttpSession | String |  |

接口名称：BuyGoodPage

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | BuyGoodPage | 无 | String |  |
| public | getGoods | Model,HttpSession | List<Good> |  |
| public | Buy | HttpSession | String |  |

接口名称：PayPage

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | PayPage | 无 | String |  |
| public | confirmPayment | Model | String |  |
| public | avoidRefresh | Model,HttpSession | String |  |
| public | SubmitOrder | HttpSession | String |  |

接口名称：DeliveryProccessPage\_

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | DeliveryProccessPage | 无 | String |  |
| public | viewStatus | 无 | String |  |
| public | mainPage | Manager,Model,HttpSession | String |  |
| public | avoidRefresh | Model,HttpSession | String |  |
| public | SubmitOrderHistory | HttpSession | String |  |
| public | ReturnGoods | HttpSession | String |  |

接口名称：AboutUsPage

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | AboutUsPage | 无 | String |  |
| public | refreshComment | Model | List<Comment> |  |
| public | avoidRefresh | Model,HttpSession | String |  |
| public | AddContent | HttpSession | String |  |

接口名称：SupplierManagePage\_

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | SupplierManagePage | 无 | String |  |
| public | avoidRefresh | Model,HttpSession | String |  |
| public | ModifySupplier | HttpSession | String |  |

接口名称：SellsPage

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | SellsPage | 无 | String |  |
| public | avoidRefresh | Model,HttpSession | String |  |
| public | outOfDateList | Model | List<Good> |  |

接口名称：SendGoodsManagePage

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | SendGoodsManagePage | 无 | String |  |
| public | avoidRefresh | Model,HttpSession | String |  |
| public | SendGoods | HttpSession | String |  |

接口名称：ReturnGoodsManagePage

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | ReturnGoodsManagePage | 无 | String |  |
| public | mainPage | Manager,Model,HttpSession | String |  |
| public | avoidRefresh | Model,HttpSession | String |  |

接口名称：SendStateManegePage

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | SendStateManegePage | 无 | String |  |
| public | mainPage | Manager,Model,HttpSession | String |  |
| public | avoidRefresh | Model,HttpSession | String |  |
| public | ChoseDeliveryman | HttpSession | String |  |

接口名称：DeliverymanManegePage

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | DeliverymanManegePage | 无 | String |  |
| public | mainPage | Manager,Model,HttpSession | String |  |
| public | avoidRefresh | Model,HttpSession | String |  |
| public | ModifyDeliveryman | HttpSession | String |  |

接口名称：MessagePage

接口所含方法：

| 修饰符 | 方法名 | 参数 | 返回值 | 备注 |
| --- | --- | --- | --- | --- |
| public | MessagePage | 无 | String |  |
| public | mainPage | Manager,Model,HttpSession | String |  |
| public | avoidRefresh | Model,HttpSession | String |  |
| public | ReadMesage | HttpSession | String |  |

2.4.4前端API接口

2.4.4.1 用户注册接口

路径：/users/{name}/{password}/{email}/{verifycode}

请求方式：POST

参数：

|  |  |
| --- | --- |
| 名称 | 说明 |
| Name | 用户昵称 |
| Password | 密码 |
| Email | 用户邮箱 |
| Verifycode | 验证码 |

返回内容：

{id:用户获得的ID}

备注：如果为空则表明注册失败，如果成功会重定向到登录页面。

2.4.4.2用户登录接口

路径：users/{name}/{password}

请求方式：GET

参数：

|  |  |
| --- | --- |
| 名称 | 说明 |
| Name | 用户昵称 |
| Password | 密码 |

返回内容：

{success:是否成功}

备注：成功返回true并跳转到用户主页，失败返回false

2.4.4.3管理员登录接口

路径：admins/{name}/{password}

请求方式：GET

参数：

|  |  |
| --- | --- |
| 名称 | 说明 |
| Name | 管理员账户名称 |
| Password | 密码 |

返回内容：

{success:是否成功}

备注：成功返回true并跳转到管理员主页，失败返回false

2.4.4.4管理员注册接口

路径：/admin/{name}/{email}/{password}

请求方式：POST

参数：

|  |  |
| --- | --- |
| 名称 | 说明 |
| Name | 管理员名称 |
| Password | 密码 |
| Email | 邮箱 |

返回内容：

{id:管理员获得的ID}

备注：如果为空则表明注册失败，如果成功会重定向到登录页面。

2.4.4.5 商品种类查询接口

路径：/goods/{category}

请求方式：GET

参数：

|  |  |
| --- | --- |
| 名称 | 说明 |
| Category | 商品种类名称 |

返回值

{data:商品,[…]}

商品：

{

productid:商品ID

name:商品名称

count:库存量

price:单价

keepdate:保质期

}

备注：1 查询失败，则data为null

2 category参数可选，如果不填，则随机返回一条商品信息

2.4.4.6 商品名称查询接口

路径：/goods/{product}

请求方式：GET

参数：

|  |  |
| --- | --- |
| 名称 | 说明 |
| Category | 商品种类名称 |

返回值

{

productid:商品ID

name:商品名称

count:库存量

price:单价

keepdate:保质期

}

2.4.4.7下订单

路径：orders/{userid}/

请求方式：POST

参数：

|  |  |
| --- | --- |
| 名称 | 说明 |
| userid | 用户ID |
| Zffs | 支付方式 |
| Productid | 商品ID |
| Count | 订购数量 |

返回值

{

Orderid:订单ID

{

备注：此接口通过表单提交，商品ID和数量可以不止一个

2.4.4.8查看未派送订单

路径：/orders

请求方式：GET

参数：无

返回值

{

data:{

ordered:订单id

status:配送状态

addr:收获地址

time:下单时间

zffs:支付方式

psyname:配送员昵称

psycontact:配送员联系方式

}

[…]

}

备注：返回所有未配送订单

2.4.4.9查看订单细节

路径：orderdetail/{orderid}

请求方式：GET

参数：

|  |  |
| --- | --- |
| 名称 | 说明 |
| Orderid | 订单ID |

返回值：

{

data:{

{

ordered:订单ID

productid:商品名称

count:数量

price:价格

}[…]

}

}

2.4.4.10退货

路径：orders/{orderid}

请求方式：DELETE

参数：

|  |  |
| --- | --- |
| 名称 | 说明 |
| orderid | 订单ID |

返回值

{success:是否成功}

2.4.4.11设置配送

路径：/order/{orderid}/{psyid}

请求方式：PUT

参数：

|  |  |
| --- | --- |
| 名称 | 说明 |
| orderid | 订单ID |
| Psyid | 配送员ID |

返回值：

{success:是否成功}

2.4.4.12查销量

路径：/xiaoliang

请求方式：GET

参数：无

返回值：

{

data:

{

productid:商品ID ,

productname:品名,

count:销量

}

}

2.4.4.13进货

路径：

/gongying/{productid}/{supplierid}/{count}/{method}

请求方式：PUT

参数

|  |  |
| --- | --- |
| 名称 | 说明 |
| productid | 产品ID |
| supplierid | 供应商ID |
| count | 供应数量 |
| Method | 运输列车车次 |

返回值：

重定向到库存查询页面

2.4.4.14查供应商

路径：

/supplier

请求方式：GET

参数：无

返回：供应商信息列表

2.4.4.15 添加供应商

路径：/supplier

请求方式：POST

参数：

|  |  |
| --- | --- |
| 名称 | 说明 |
| supplierid | 供应商ID |
| Suppliername | 供应商名称 |
| adminname | 负责人名称 |
| contact | 联系方式 |

返回值：是否添加成功

备注：重定向到查询供应商页面

2.4.4.16 删除供应商

路径：/supplier/{id}

请求方式：DELETE

参数：

|  |  |
| --- | --- |
| 名称 | 说明 |
| supplierid | 供应商ID |

返回值：是否成功

备注：重定向到查询供应商页面

2.4.5界面类，外观类设计

我们对界面进行了进一步修改，现在所有的界面已经被现代设计元素刷新，我们大量使用了圆角设计，旨在让你以熟悉的方式更接近你所喜爱的东西，这些新的体验将继续使你更容易工作、玩耍、创造和联系，无论你在哪里或使用什么设备。

我们简化了界面设计和用户操作，在赋能生产力的同时，激发更多创造力。全新 UI更加现代化，整洁且美观，令人耳目一新。每一种声音、字体和图标，这一切都经过精心设计，让您可以自由掌控，为您带来更加平静和轻松的感觉。全新一代WebUI，与你所爱，更近一步。

2.4.5.1首页

图形用户界面, 应用程序

描述已自动生成

2.4.5.2注册页面

图形用户界面, 网站

描述已自动生成

2.4.5.3购物页面

图形用户界面, 网站

描述已自动生成

2.4.5.4结算页面

图形用户界面, 应用程序

描述已自动生成

2.4.5.5查看配送/退货

图形用户界面, 应用程序, 网站

描述已自动生成

2.4.5.6关于我们/评论

“关于我们”页面，可以查看用户评论，并且了解开发团队的详细信息。

文本

中度可信度描述已自动生成

2.4.5.7供应商操作页面

表格

描述已自动生成

2.4.5.8查询销售情况页面

表格

描述已自动生成

2.4.5.9设置配送任务页面

表格

描述已自动生成

2.4.5.10管理员通知中心页面

应用程序

低可信度描述已自动生成

2.5活动图

我们对比较复杂的业务逻辑给出了活动图进一步说明，为了使将来开发有利可据。

我们认为比较重要的活动有以下几个部分：

* 注册登录
* 购买下单
* 退货
* 配送

下面将一一给出具体的活动图描述。

2.5.1配送业务逻辑

图示, 示意图

描述已自动生成

2.5.2注册业务逻辑

图示

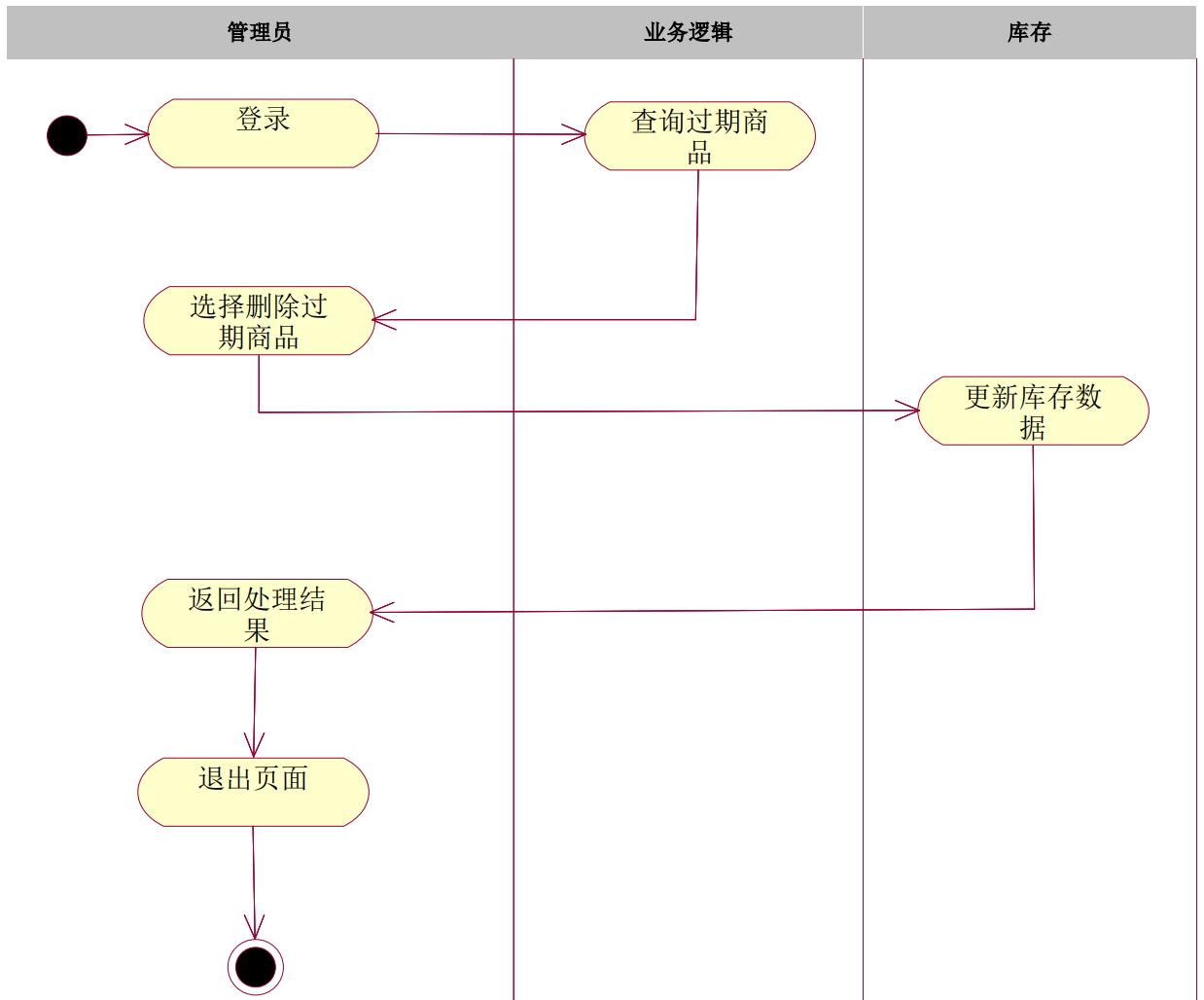
描述已自动生成

2.5.3登录业务逻辑

图示

描述已自动生成

2.5.4删除过期水果业务逻辑



2.5.5出货业务逻辑

图示

描述已自动生成

2.5.6退货业务逻辑

图示

描述已自动生成