**《乐淘电子商务系统》**

**详细分析与设计**

**V2.0**

**目 录**

[第一部分 引言 6](#_Toc323127521)

[一、编写目的 6](#_Toc323127522)

[二、项目背景 6](#_Toc323127523)

[三、定义 7](#_Toc323127524)

[1、技术类 7](#_Toc323127525)

[2、业务类 9](#_Toc323127526)

[四、参考资料 9](#_Toc323127527)

[第二部分 项目概述 10](#_Toc323127528)

[一、项目描述 10](#_Toc323128040)

[二、项目功能描述 10](#_Toc323128041)

[n 用户购物业务功能描述 10](#_Toc323128042)

[n 用户消费业务功能描述 11](#_Toc323128043)

[n 商品管理业务流程描述 13](#_Toc323128044)

[n 订单管理 14](#_Toc323128046)

[n 账户管理 16](#_Toc323128048)

[第三部分 总体设计 18](#_Toc323127529)

[一、技术架构设计 18](#_Toc323127530)

[二、核心控制流程 19](#_Toc323127531)

[1、核心控制流程图 19](#_Toc323127532)

[2、核心控制流程说明 19](#_Toc323127533)

[第四部分 界面设计 20](#_Toc323127539)

[一、界面设计 20](#_Toc323127540)

[1、界面框架设计 20](#_Toc323127541)

[2、PC界面设计 20](#_Toc323127542)

[2.1、登录界面设计 20](#_Toc323127543)

[2.2、主体界面设计 21](#_Toc323127544)

[2.3、商品详情界面设计 22](#_Toc323127545)

[2.4、购物车界面设计 22](#_Toc323127546)

[2.5、结算界面设计 22](#_Toc323127547)

[2.6、订单中心界面设计 23](#_Toc323127548)

[第五部分 单元模块设计 23](#_Toc323127564)

[一、数据访问层设计 24](#_Toc323128049)

[1、类图设计 24](#_Toc323127566)

[2、类的详细设计描述 24](#_Toc323127567)

[2.1 IUserGroupDao接口设计 24](#_Toc323127584)

[二、业务逻辑层设计 25](#_Toc323127587)

[1、类图设计 25](#_Toc323127588)

[2、类的详细设计描述 26](#_Toc323127589)

[2.1 AdminService接口设计 26](#_Toc323127590)

[2.2 ConsumerService接口设计 26](#_Toc323127591)

[2.3 StoreService接口设计 27](#_Toc323127592)

[2.4 CommodityService接口设计 28](#_Toc323127593)

[2.5 CommoditySpecsService接口设计 28](#_Toc323127594)

[2.6 AttributeKeyService接口设计 29](#_Toc323127595)

[2.7 AttributeValueService接口设计 29](#_Toc323127596)

[2.8 CategoryService接口设计 29](#_Toc323127597)

[2.9 CartService接口设计 30](#_Toc323127598)

[2.10 LtOrderService接口设计 30](#_Toc323127599)

[2.11 OrderitemService接口设计 30](#_Toc323127600)

[2.12 CommentService接口设计 31](#_Toc323127601)

[2.13 RecommodityService接口设计 31](#_Toc323128120)

[2.14 OrderAddressService接口设计 31](#_Toc323128121)

[2.15 TokenService接口设计 31](#_Toc323128122)

[2.16 业务接口实现类汇总 32](#_Toc323127610)

[三、控制层设计 32](#_Toc323128050)

[1、类图设计 32](#_Toc323128052)

[四、后台管理 35](#_Toc323127611)

[1、商品信息管理 35](#_Toc323127612)

[1.1、功能汇总 35](#_Toc323127613)

[1.2、类图设计 36](#_Toc323127614)

[1.2.1、类关系结构图 36](#_Toc323127615)

[1.2.2、时序图 36](#_Toc323127616)

[1.3、类的详细设计描述 37](#_Toc323127617)

[1.3.1、CommodityManagement.html描述（待添加） 37](#_Toc323127618)

[1.3.2、ACommodityController描述 37](#_Toc323127619)

[1.4、功能实现 37](#_Toc323127620)

[2.门店信息管理 38](#_Toc323128054)

[2.1、功能汇总 38](#_Toc323128055)

[2.2、类图设计 38](#_Toc323128056)

[2.2.1、类关系结构图 38](#_Toc323128057)

[2.2.2、时序图 39](#_Toc323128058)

[2.3、类的详细设计描述 39](#_Toc323128059)

[2.3.1、StoreManagement.html描述（代添加） 39](#_Toc323128060)

[2.3.2、StoreController描述 40](#_Toc323128061)

[2.4、功能实现 40](#_Toc323128062)

[3.商品分类管理 41](#_Toc323128063)

[3.1、功能汇总 41](#_Toc323128064)

[3.2、类图设计 41](#_Toc323128065)

[3.2.1、类关系结构图 41](#_Toc323128066)

[3.2.2、时序图 41](#_Toc323128067)

[3.3、类的详细设计描述 42](#_Toc323128068)

[3.3.1、CategoryManagement.html描述述 42](#_Toc323128069)

[3.3.2、CategoryController描述 43](#_Toc323128070)

[3.4、功能实现 43](#_Toc323128071)

[4.订单管理 43](#_Toc323128072)

[4.1、功能汇总 43](#_Toc323128073)

[4.2、类图设计 44](#_Toc323128074)

[4.2.1、类关系结构图 44](#_Toc323128075)

[4.2.2、时序图 44](#_Toc323128076)

[4.3、类的详细设计描述 44](#_Toc323128077)

[4.3.1、OrderManagement.html描述 44](#_Toc323128078)

[4.3.2、OrderController描述 45](#_Toc323128079)

[4.4、功能实现](#_Toc323128080) 45

[五、用户功能 46](#_Toc323127699)

[1.商品浏览 46](#_Toc323128081)

[1.1、功能汇总 46](#_Toc323128082)

[1.2、类图设计 46](#_Toc323128083)

[1.2.1、类关系结构图 46](#_Toc323128084)

[1.2.2、时序图 46](#_Toc323128085)

[1.3、类的详细设计描述 47](#_Toc323128086)

[1.3.1、CommodityManagement.html描述 47](#_Toc323128087)

[1.3.2、CommodityController描述 47](#_Toc323128088)

[1.4、功能实现 48](#_Toc323128089)

[2.商品购买 48](#_Toc323128090)

[2.1、功能汇总 48](#_Toc323128091)

[2.2、类图设计 49](#_Toc323128092)

[2.2.1、类关系结构图 49](#_Toc323128093)

[2.2.2、时序图 50](#_Toc323128094)

[2.3、类的详细设计描述 51](#_Toc323128095)

[2.3.1、CartManagement.html描述 51](#_Toc323128096)

[2.3.2、CartController描述 51](#_Toc323128097)

[2 .4、功能实现 51](#_Toc323128098)

[3.用户账号管理 52](#_Toc323128099)

[3.1、功能汇总 52](#_Toc323128100)

[3.2、类图设计 52](#_Toc323128101)

[3.2.1、类关系结构图 52](#_Toc323128102)

[3.2.2、时序图 52](#_Toc323128103)

[3.3、类的详细设计描述 53](#_Toc323128104)

[3.3.1、UserManagement.html描述 53](#_Toc323128105)

[2.3.2、UserController描述 53](#_Toc323128106)

[3.4、功能实现 53](#_Toc323128107)

[第六部分 数据库设计 54](#_Toc323128109)

[一、 数据库整体结构图 55](#_Toc308601754)

[二、数据库表格清单 55](#_Toc323128110)

[1、store表结构 55](#_Toc323128004)

[2、admin表结构 56](#_Toc323128005)

[3、commodity表结构 56](#_Toc323128006)

[4、commodity\_specs表结构 56](#_Toc323128007)

[5、attribute\_key表结构 57](#_Toc323128008)

[6、attribute\_value表结构 57](#_Toc323128111)

[7、category表结构 57](#_Toc323128009)

[8、cart表结构 57](#_Toc323128010)

[9、lt\_order表结构 57](#_Toc323128011)

[10、orderitem表结构 58](#_Toc323128112)

[11、user表结构 58](#_Toc323128113)

[12、comment表结构 58](#_Toc323128114)

[13、collection表结构 59](#_Toc323128123)

[14、order\_address表结构 59](#_Toc323128124)

[15、系统管理外键清单 59](#_Toc323128012)

[第七部分 补充设计和说明 60](#_Toc323128036)

[一、编译运行环境设计 60](#_Toc323128037)

[二、包路径与WEB目录结构设计 60](#_Toc323128038)

[1、包路径设计 60](#_Toc323128039)

# 第一部分 引言

### 一、编写目的

编写本设计的目的是为了准确阐述的具体实现思路和方法，即系统的详细架构和实现逻辑，主要包括程序系统的结构以及各层次中每个程序的设计考虑。预期读者为项目全体成员，包括运行维护和测试人员。

### 二、项目背景

1、公司开发此项目背景

国内电商市场不断扩大，互联网正在引发一场消费时代的变革，B2C的网上销售将成电商发展趋势。通过电子商务系统，企业可以节约很多成本，并且利用有限资源，跨过时间、地域限制获得更多利益。在网络上，拥有充足的空间和市场进行品牌宣传、产品推广；电子商务系统还可以和消费者进行直接接触，减少中间环节；可以掌控价格市场，避免价格战及不良竞争模式；可以开发各种渠道及代理，扩大销售市场。

现今B2C模式的生活消费类电子商务平台普及度较高，成为人们日常生活购物的主要渠道之一，由于其品类繁多、可一站式购物、物美价廉、可自取货物或送货到家等业务为人们的日常生活带来了极大的便利，成为了人们购买商品时的首选。但B2C模式依然存在许多问题，比如资金周转困难，商品定位不准确，网上支付体系不健全，信用机制和电子商务立法不健全。

乐淘电子商务系统将会为消费者提供一个可靠、便利的购物平台，并根据用户的喜好制定专属的商品推荐，提供自取或送货上门等人性化服务，简化商家的管理，拓宽市场和销售渠道，改变人们的消费方式和商户的经营模式。

2、社会背景

* 系统名称：乐淘电子商务系统
* 任务提出者：略。
* 开发者：略。
* 用户和运行该程序系统的计算中心：略。

### 三、定义

详细设计：支持研发团队

研发团队：参考详细设计

真正技术名词、框架名词

#### 1、技术类

**SSM**：

Java SSM框架即指Spring+[SpringMVC](https://so.csdn.net/so/search?q=SpringMVC&spm=1001.2101.3001.7020)+MyBatis的简称，框架集由Spring、MyBatis两个开源框架整合而成（SpringMVC是Spring中的部分内容）,常作为数据源较简单的web项目的框架。

相比于之前的[SSH](https://so.csdn.net/so/search?q=SSH&spm=1001.2101.3001.7020)（Spring+Struts+Hibernate），SSM更加轻量化和灵活，是目前业界主流的Java Web开发框架。

Spring

Spring是一个开源[框架](https://so.csdn.net/so/search?q=%E6%A1%86%E6%9E%B6&spm=1001.2101.3001.7020)，Spring是于2003 年兴起的一个轻量级的Java开发框架，由Rod Johnson 在其著作Expert One-On-One J2EE Development and Design中阐述的部分理念和原型衍生而来。它是为了解决企业应用开发的复杂性而创建的。Spring使用基本的JavaBean来完成以前只可能由EJB完成的事情。然而，Spring的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何Java应用都可以从Spring中受益。 简单来说，Spring是一个轻量级的控制反转（IoC）和面向切面（AOP）的容器框架。

     Spring就像是整个项目中装配JavaBean的大工厂，在配置文件中可以指定使用特定的参数去调用实体类的构造方法来实例化对象。也可以称之为项目中的粘合剂。Spring的核心思想是IoC（控制反转），即不再需要程序员去显式地new一个对象，而是让Spring框架帮你来完成这一切；而重要功能Aop（面向切面编程）能够将那些与业务无关，却为业务模块所共同调用的逻辑或责任（例如事务处理、日志管理、权限控制等）封装起来，在系统的业务流程代码中需要的位置切入这些通用功能代码，以便于减少系统的重复代码，有利于项目未来的可拓展性和可维护性。

SpringMVC

      Spring MVC属于SpringFrameWork的后续产品，已经融合在Spring Web Flow里面。SpringMVC把传统的模型层被拆分为了业务层(Service)和数据访问层（DAO,Data Access Object）。在业务层中可以通过 Spring 的声明式事务操作数据访问层，而在业务层上还允许我们访问 NoSQL ，这样就能够满足异军突起的 NoSQL 的使用了，它可以大大提高互联网系统的性能。

     SpringMVC在项目中拦截用户请求，它的核心Servlet即DispatcherServlet承担中介或是前台这样的职责，将用户请求通过HandlerMapping去匹配Controller，Controller就是具体对应请求所执行的操作，最终把Model层执行结果返回给用户。SpringMVC相当于SSH框架中[struts](https://so.csdn.net/so/search?q=struts&spm=1001.2101.3001.7020)。

MyBatis

MyBatis本是apache的一个开源项目iBatis, 2010年这个项目由apache software foundation 迁移到了google code，并且改名为MyBatis ，2013年11月迁移到Github。MyBatis 是一款优秀的Java持久层框架，它支持定制化 SQL、存储过程以及高级映射。MyBatis 消除了几乎所有的JDBC代码和参数的手工设置以及结果集的检索。MyBatis 使用简单的 XML配置文件或注解，将接口和 Java 的POJOs（Plain Old Java Objects，普通的 Java对象）映射成数据库中的记录。

    MyBatis是对JDBC的封装，它让数据库底层操作变的透明，属于半自动的ORM(对象关系映射)框架。mybatis的操作都是围绕一个sqlSessionFactory实例展开的。mybatis通过配置文件关联到各实体类的Mapper文件，Mapper文件中配置了每个类对数据库所需进行的sql语句映射。在每次与数据库交互时，通过sqlSessionFactory拿到一个sqlSession，再执行sql命令。

Mybatis是半自动ORM映射工具，可以自定义高效的SQL语句提升数据库访问的效率。Mybatis是半自动的原因是因为我们要在Mapper文件中手写SQL语句，全自动的ORM（如[Hibernate](https://so.csdn.net/so/search?q=Hibernate&spm=1001.2101.3001.7020)）是不需要手写SQL的。MyBatis可以自定义高效的SQL语句提升数据库访问的效率；而且MyBatis容易掌握，而Hibernate门槛较高。所以我们用Mybatis代替传统的Hibernate。

**SpringBoot：**

SpringBoot是一种全新的框架，目的是为了简化Spring应用的初始搭建以及开发过程。该框架使用特定的方式(集成starter，约定优于配置)来进行配置，从而使开发人员不需要再定义样板化的配置。SpringBoot提供了一种新的编程范式，可以更加快速便捷地开发Spring项目，在开发过程当中可以专注于应用程序本身的功能开发，而无需在Spring配置上花太大的工夫。

     SpringBoot基于Sring4进行设计，继承了原有Spring框架的优秀基因。SpringBoot并不是一个框架，而是一些类库的集合。maven或者gradle项目导入相应依赖即可使用SpringBoot，而无需自行管理这些类库的版本。

#### 2、业务类

**RFID托盘：**托盘上安装了RFID标记的特殊托盘，进入系统使用的托盘必须按照RFID标签成为RFID托盘，RFID托盘是系统最小管理单元；

**货主IC卡：**每个货主都持有IC卡，对应货主的唯一身份标识。

### 四、参考资料

列出有关的参考资料，如：

1. 本项目的经核准的计划任务书；

《03项目库文档@乐淘电子商务系统》

1. 属于本项目的其他已发表的文件；

《需求说明》

《概要设计》

1. 本文件中各处引用到的文件资料，包括所要用到的软件开发标准。

《Java语言编码规范(Java Code Conventions)》

# 第二部分 项目概述

### 一、项目描述

国内电商市场不断扩大，互联网正在引发一场消费时代的变革，B2C的网上销售将成电商发展趋势。通过电子商务系统，企业可以节约很多成本，并且利用有限资源，跨过时间、地域限制获得更多利益。在网络上，拥有充足的空间和市场进行品牌宣传、产品推广；电子商务系统还可以和消费者进行直接接触，减少中间环节；可以掌控价格市场，避免价格战及不良竞争模式；可以开发各种渠道及代理，扩大销售市场。

### 二、项目功能描述

##### 用户购物业务功能描述

1. 商品浏览

客户可以在网页顶端的幻灯片切换界面浏览感兴趣的商品（根据购物记录或者个人信息推荐热门产品），且可以在该页面内根据商品的不同分类浏览商品的图片以及简略信息，用户可以通过浏览这些商品来选择需要购买的商品。

主要显示商品简要信息，包括:

商品名称

商品图片

商品价格

1. 商品搜索

客户可以在购物界面搜索栏通过输入关键字搜索心仪的产品，系统会根据关键字展示出符合条件的商品，且在搜索框内部会有推荐的热门搜索信息，用户可以直接点击进行搜索，且在点击搜索栏后会有用户的搜索记录。

主要通过关键字查询，查询条件包括以下两种：

1）商品类别

2）商品关键字（名称）

1. 商品信息查询

当用户在浏览商品或者搜索商品后选择到了自己心仪的产品，可以直接点击商品图片或者商品名称，查看商品的详细信息：商品销量，商品评论，商品的型号、颜色，对商品全方位展示的视频和图片。

主要显示商品详细信息,包括：

商品销量

商品品牌

商品评论

商品的规格

商品图片

商品产地

……

##### 用户消费业务功能描述

1. 购买商品

购买商品是客户在商品详情界面点击购买，选择商品款式与数量，确认无误后结算付款。

购买详情主要包括：

商品款式

商品数量

其中，商品款式一项根据商品的种类不同可能有多个选择或为无，如衣服类商品的款式包括了颜色与尺码，而书籍类商品可能没有款式；商品数量注意不能大于库存或是商家设置的限购数额。

1. 添加购物车

添加购物车是客户在“商品详情”界面将商品加入购物车中，并可以在“购物车”界面中查看自己购物车的商品信息。

购物车商品信息主要包括：

商品id

商品数量

商品款式

商品价格

用户可以对购物车中商品数量、款式进行修改，或是删除某项商品。购物车底端会显示选中商品总金额。客户选择好商品并确认无误后结算付款。

1. 支付订单

支付订单指在付款页面，客户选择提货方式与付款方式，确认所有信息无误后付款，后台数据库同时对商品库存数量进行更新。

支付明细包括：

商品id

商品数量

商品款式

商品价格

送货方式

支付方式

其中，送货方式包括送货上门与店面自取，选择“店面自取”后，用户需打开地图选择取货门店与取货时间；选择“送货上门”后，用户需要填写自己的收货地址。付款方式包括支付宝、微信等。

1. 查看订单

付款成功后，客户在“我的订单”界面中查询自己的订单相关信息。

订单信息包括：

商品名称

商品数量

商品总价

物流信息

订单号

下单时间

1. 确认收货

客户在收到货物后，打开“我的订单”界面，点击“确认收货”，支付平台将货款转给商家，同时订单状态转变为“已完成”。

##### 商品管理业务流程描述

1. 商品类别管理

商店管理员通过系统按照商品属性添加到商店列表中所属大类型，生成商店的该类别下的商品。

商品类别详情：

类别序号

类别名称

父类别序号

类别层级

1. 商品信息管理

商店管理员通过系统按照商品详细信息添加到商品商品信息下，生成商品的详细信息。

商品信息详情:

商品名称

商品价格

商品库存

商品描述

商品图片

商品类别

1. 门店信息管理

商店管理员通过系统按照商店实际信息添加到商店信息管理页面中，生成商店详细信息。 商店详细信息：

商店名称

商店地址

商店电话

...

##### 

##### 订单管理

1. 检索订单

用户支付完成订单过后，系统会自动将其加入管理员的所有已完成支付的订单列表内，管理员可通过已完成支付订单列表进行查看具体订单信息，可以通过订单状态（“全部”、“已发货”、“已收到”、“已取货”）检索订单。

顾客/管理员检索订单主要通过关键词查询。内容包括

1）商品信息（主要商品名字和商品类别和品牌名）（管理员无这项）

2）店铺信息（主要店铺名字）

3）订单时间（下单时间划定）

4）订单状态

顾客/管理员查询订单主要显示订单详细信息，内容包括

商店名字（管理员无）

订单号（顾客无）

订单状态

商品名字

商品数量

商品价格

订单价格

顾客增删订单信息主要基于订单信息页面上按钮，内容包括

加入购物车

查看物流

延长物流

退款操作

追加评论

删除订单

商户管理员增删订单信息主要基于后台订单管理界面，内容包括

选择物流方式

查看商品物流

退款操作

处理订单

1. 统计销售数据

系统会分别将各个周、各个月、各个季度内商品销售的种类，数量，资金总额统计到后台数据库，管理员可通过统计数据查询获取相应的数据资料，同时也可将数据生成柱状图、饼图等以便直观的了解整个销售情况。

商户管理员查询销售数据主要通过关键词查询，内容包括

1）商品名字

2）商品种类

3）售卖时间

商户管理员查询销售数据主要显示内容为

商品名字

商品种类

商品数量

销售总额

时间段内限定上述数据

###### 控制订单发货

门店已将货物送出过后，管理员需要将订单号等具体信息输入系统，系统会自动将订单状态信息修改为“已发货”。当用户确认收到商品后，系统会将订单状态信息修改为“已收货”。或者当用户亲自上门取货后，系统会将订单状态信息修改为“已取货”。

商户管理员控制订单主要通过订单信息传递扫描，内容包括

1）系统自动根据订单信息生成条形码或二维码

2）商户打印订单（确认商户发货）

3）物流扫描订单条形码或二维码信息（确认物流发货）

顾客亲自取货后由商户修改订单状态（确认已取货）

4）终端物流在顾客前来取货时扫描条形码或二维码信息

顾客自行开存放商品柜自动确认订单状态（确认已收货）

##### 账户管理

1. 用户注册、登录

消费者登录乐淘网，点击用户注册，进入新用户注册界面，按提示要求设置用户名、密码、头像、性别、年龄、收件地址、联系方式等信息，点击注册。完成新用户注册后点击用户登录，输入用户名及密码，点击登录，待页面显示登录成功后即可进行商品选购。

登录主要通过账号进行查询，内容包括：

账号密码

购物车信息

评论信息

订单信息

注册主要进行用户信息的插入，内容包括：

用户姓名

用户地址

用户邮箱

用户地址

用户头像

……

1. 用户信息管理

用户在登录后可以修改个人信息，点击设置，可以修改用户名、密码、头像、性别、年龄、收件地址、联系方式等信息，保存后即可成功修改、维护个人信息。

主要通过账号查询修改，内容包括：

用户地址

用户邮箱

用户地址

用户头像

……

# 第三部分 总体设计

### 一、技术架构设计



1. Web前端：vue；

2. Java EE应用框架：Spring Boot、MyBatis-plus；

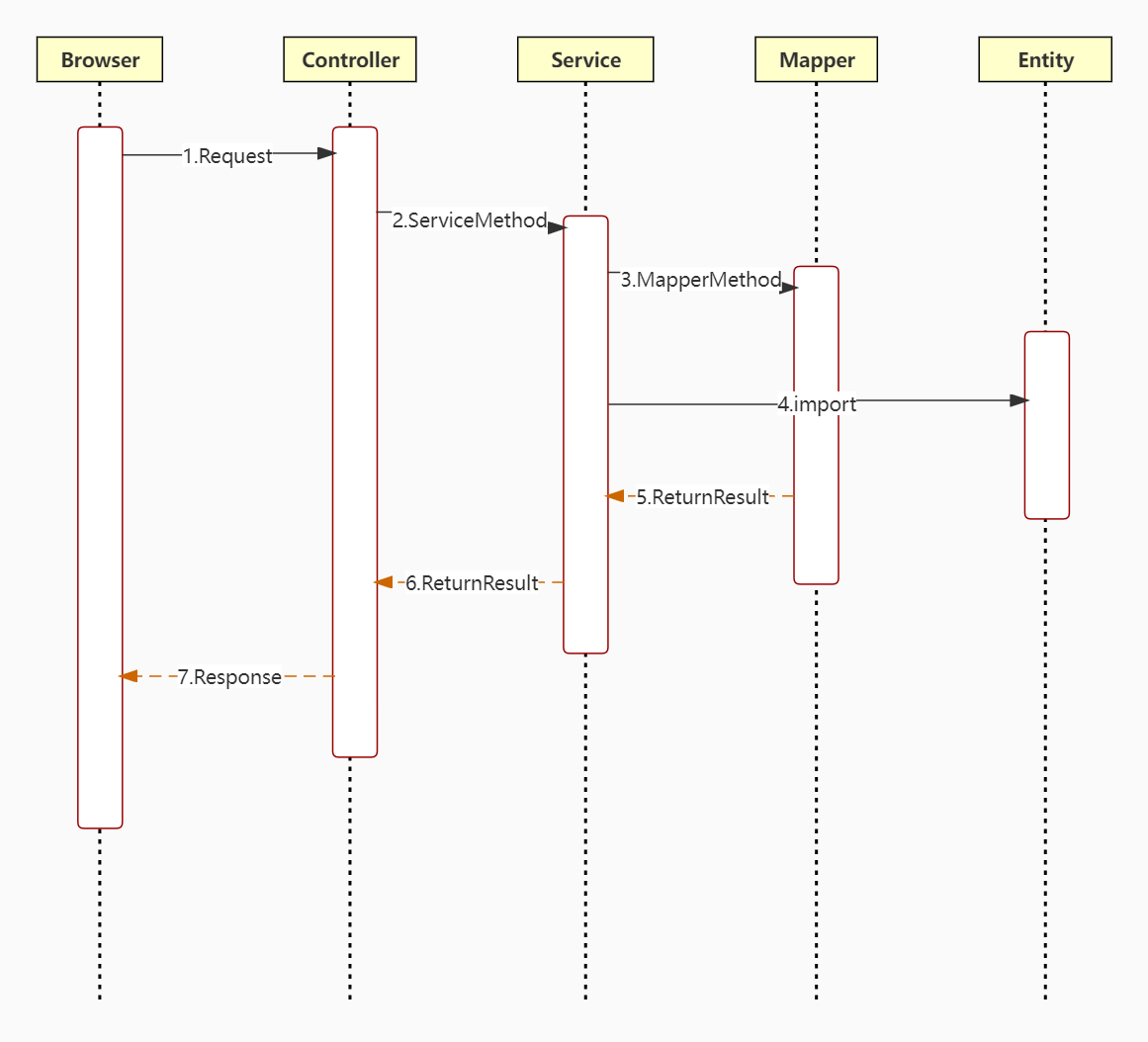
3. 数据库：MySQL；

4. 开发语言：Java，python，

5、大数据 Hadoop框架

### 二、核心控制流程

#### 1、核心控制流程图



#### 2、核心控制流程说明

结合上图，针对每一个消息请求，按照消息请求/回复顺序，作如下说明。

|  |  |  |
| --- | --- | --- |
| **编号** | **消息名称** | **消息说明** |
| 1 | HttpServletRequest | 浏览器发来的HttpServletRequest 请求 |
| 2 | ServiceMethod | 调用service层业务逻辑 |
| 3 | MapperMethod | 调用mapper层业务逻辑 |
| 4 | Import | 使用pojo进行数据操作 |
| 5 | ReturnResult | 返回Mapper执行结果，给Service |
| 6 | ReturnResult | 返回Service执行结果，给Controller |
| 7 | HttpServletResponse | 返回处理结果给浏览器 |

# 第四部分 界面设计

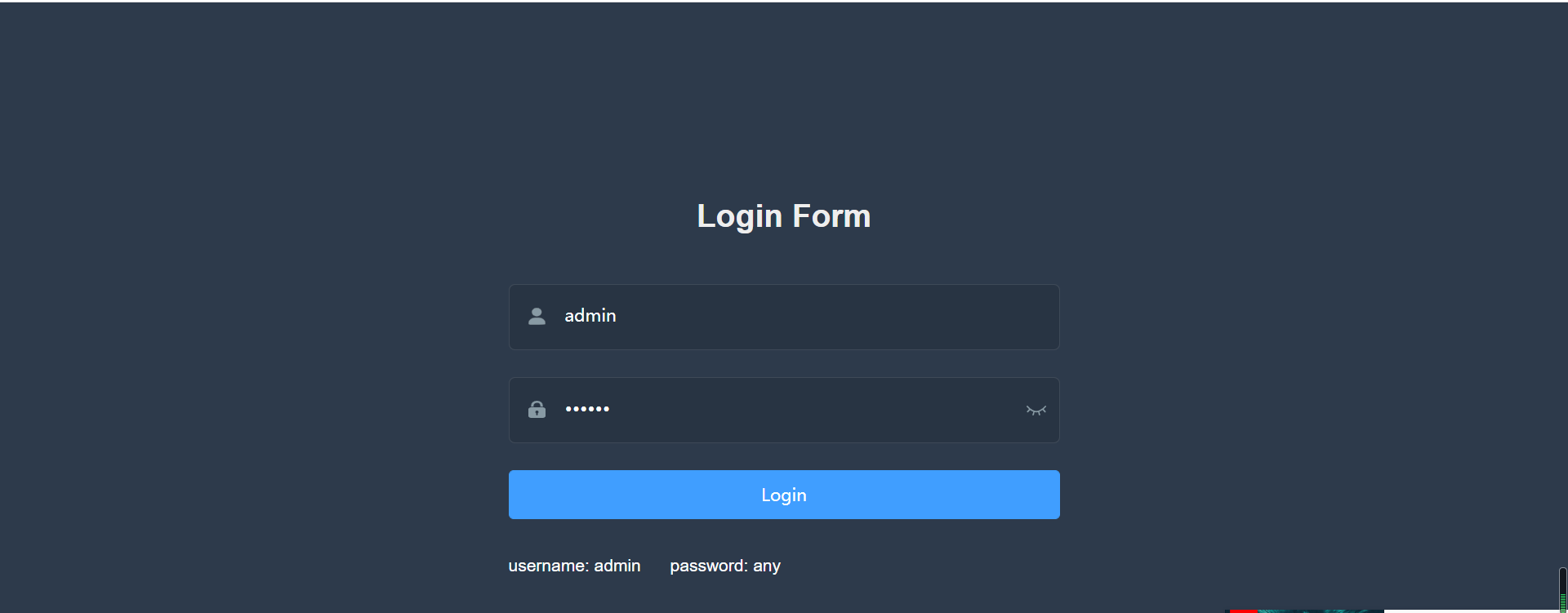
### 一、界面设计

#### 1、界面框架设计

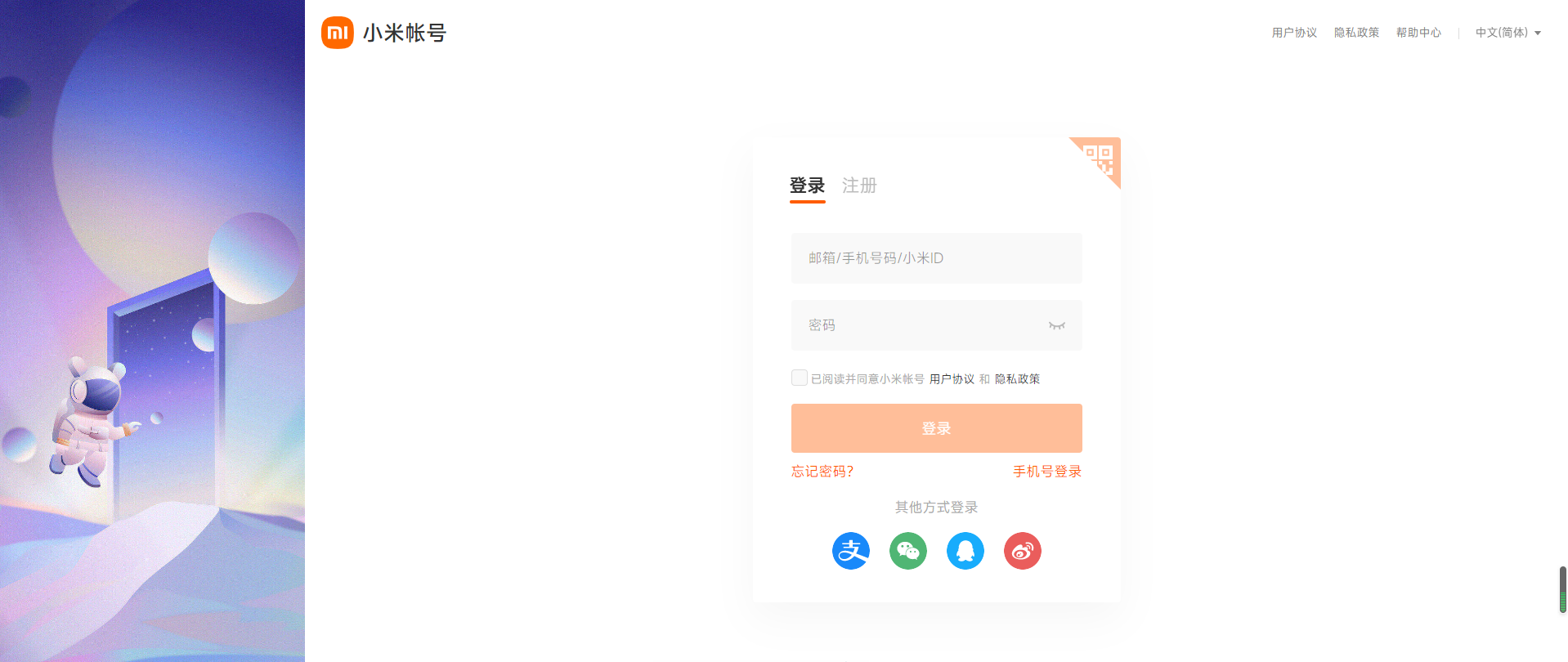
#### 2、PC界面设计

##### 2.1、登录界面设计

后台：

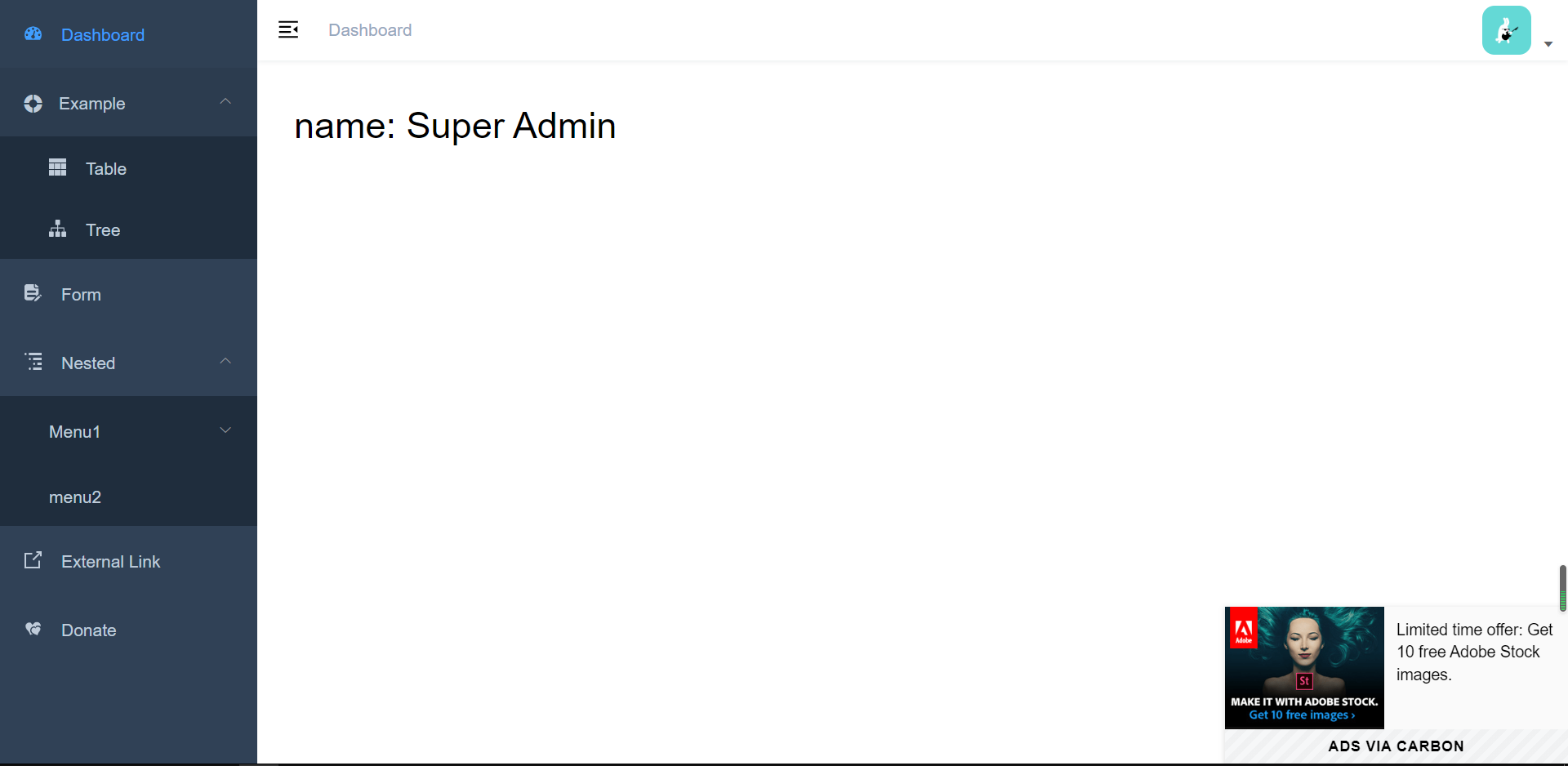


前台：

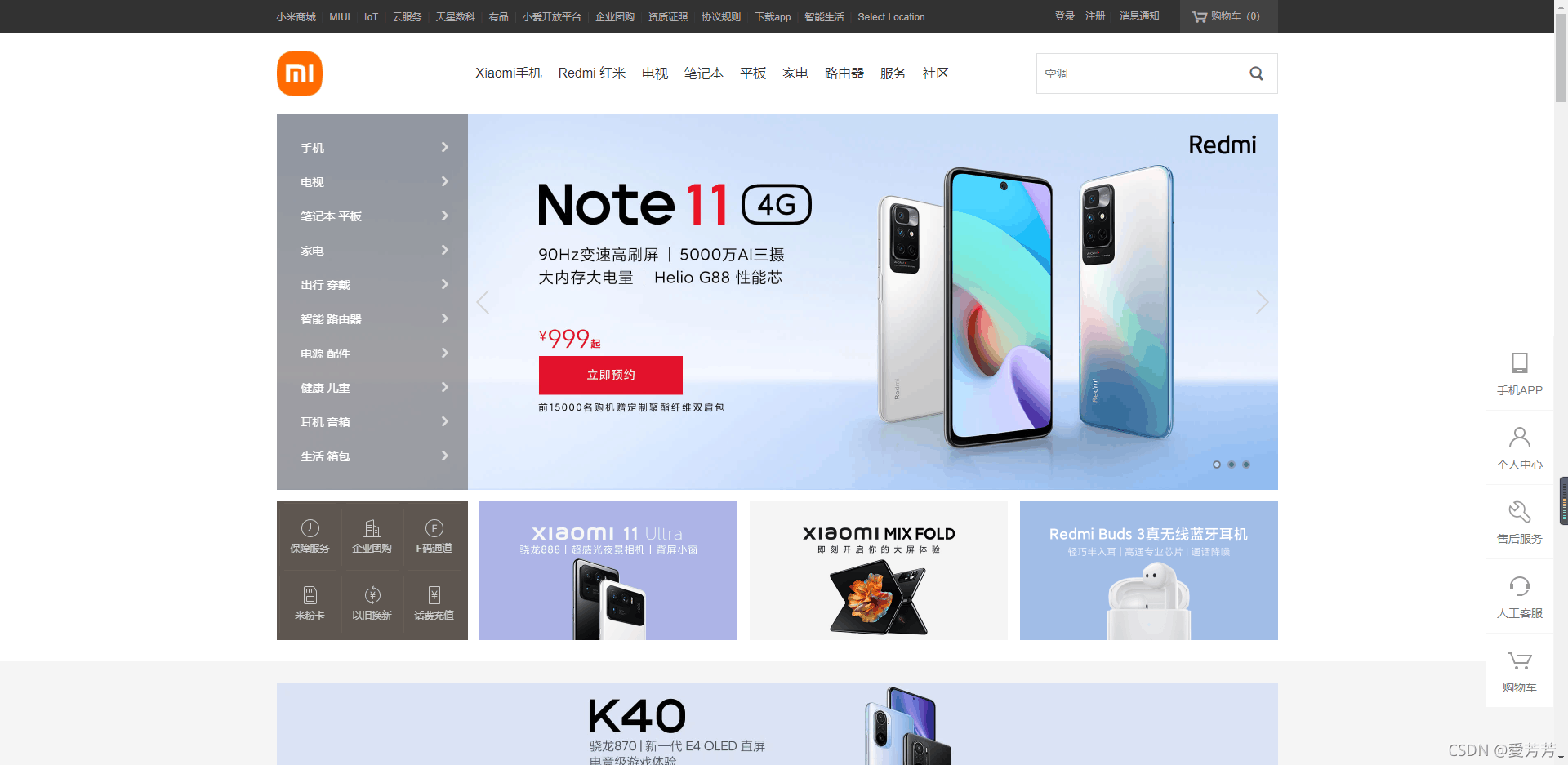


##### 2.2、主体界面设计

后台：



前台：



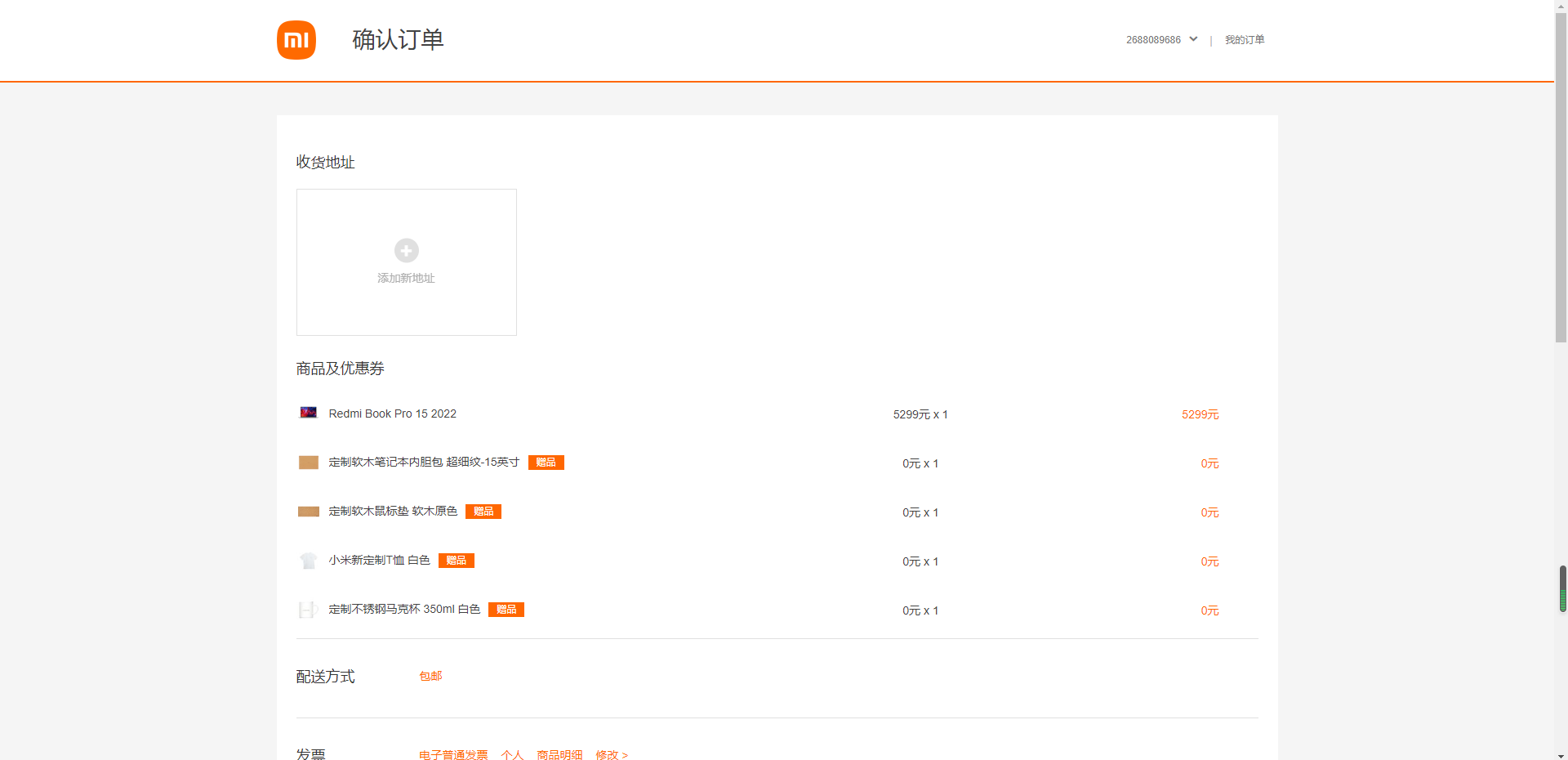
##### 2.3、商品详情界面设计



##### 2.4、购物车界面设计



##### 2.5、结算界面设计



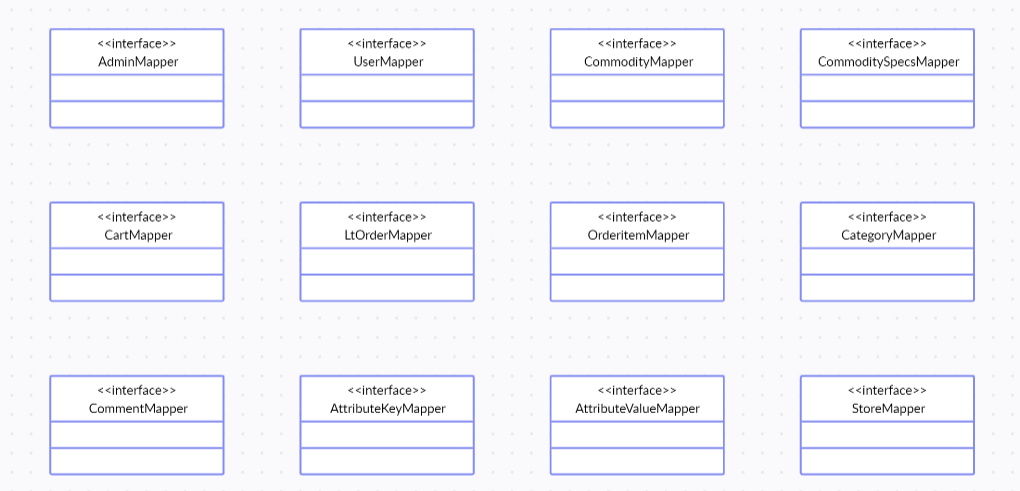
##### 2.6、订单中心界面设计



# 第五部分 单元模块设计

### 一、数据访问层设计

#### 1、类图设计



#### 2、类的详细设计描述

##### 2.1 IUserGroupDao接口设计



详细描述：

|  |
| --- |
| IUserGroupDaoUserGroup 用户组的CRUD操作  事务定义@Transactional(propagation=Propagation.REQUIRED)，声明在接口名上方，表示本接口内所有的方法如果没有定义事务类型，都以该事务类型执行，该事务为：如果当前没有事务，就新建一个事务，如果已经存在一个事务中，加入到这个事务中 |
| public void saveUserGroup(UserGroup ug);  保存用户组 |
| public void updateUserGroup(UserGroup ug);  更新用户组 |
| public void deleteUserGroup(String ugid);  删除用户组 |
| public UserGroup findUserGroupById(String ugid);  通过用户组id查找该用户组  事务配置为：@Transactional(readOnly=true) |
| public List<UserGroup> findUGList(String ugID);  通过用户ID查找其所在组  事务配置为：@Transactional(readOnly=true) |

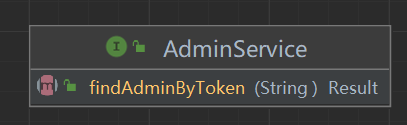
### 二、业务逻辑层设计

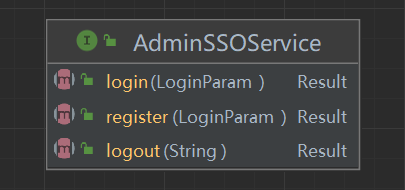
#### 1、类图设计



#### 2、类的详细设计描述

##### 2.1 AdminService接口设计

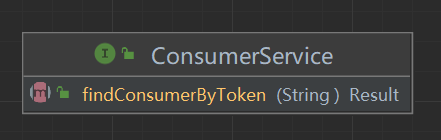


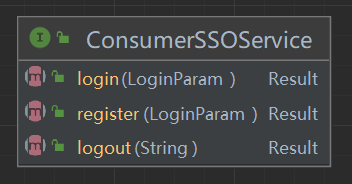


详细描述：

|  |
| --- |
| AdminService管理员登录注册接口  事务定义@Transactional(propagation=Propagation.REQUIRED)，声明在接口名上方，表示本接口内所有的方法如果没有定义事务类型，都以该事务类型执行，该事务为：如果当前没有事务，就新建一个事务，如果已经存在一个事务中，加入到这个事务中 |
| public void saveUser(User user);  保存管理员数据 |

##### 2.2 ConsumerService接口设计

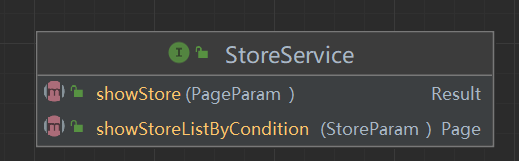




详细描述：

|  |
| --- |
| UserService顾客登录注册接口  事务定义@Transactional(propagation=Propagation.REQUIRED)，声明在接口名上方，表示本接口内所有的方法如果没有定义事务类型，都以该事务类型执行，该事务为：如果当前没有事务，就新建一个事务，如果已经存在一个事务中，加入到这个事务中 |

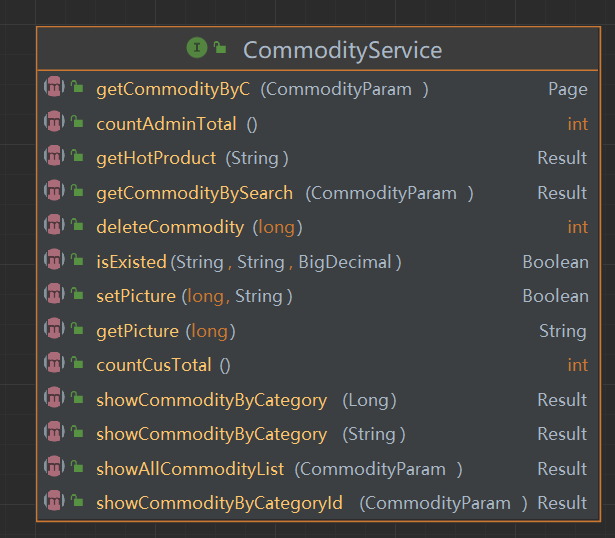
##### 2.3 StoreService接口设计



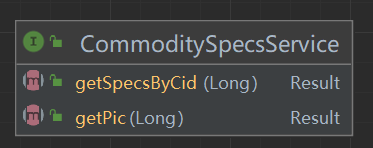
详细描述：

|  |
| --- |
| StoreService 门店管理接口  事务定义@Transactional(propagation=Propagation.REQUIRED)，声明在接口名上方，表示本接口内所有的方法如果没有定义事务类型，都以该事务类型执行，该事务为：如果当前没有事务，就新建一个事务，如果已经存在一个事务中，加入到这个事务中 |

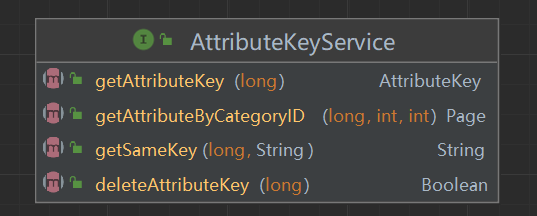
##### 2.4 CommodityService接口设计



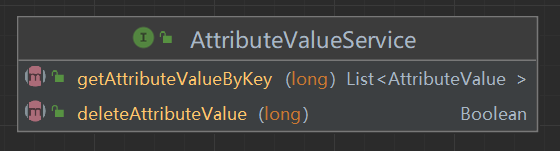
##### 2.5 CommoditySpecsService接口设计



##### 2.6 AttributeKeyService接口设计



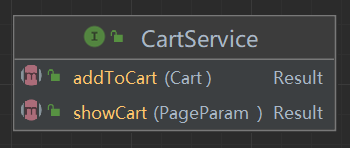
##### 2.7 AttributeValueService接口设计



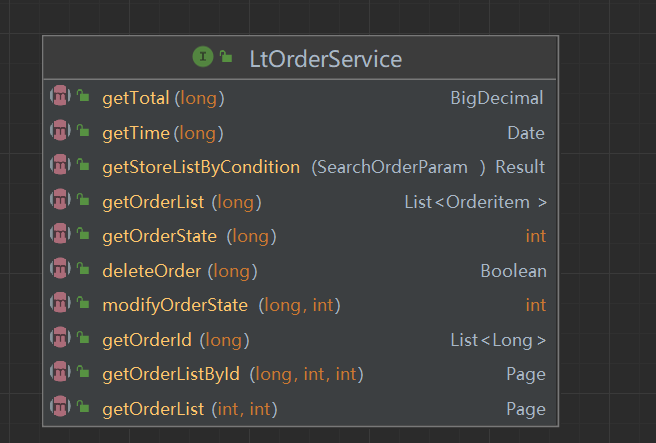
##### 2.8 CategoryService接口设计



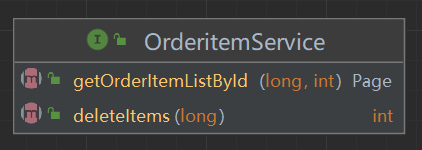
##### 2.9 CartService接口设计



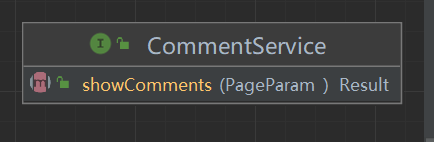
##### 2.10 LtOrderService接口设计



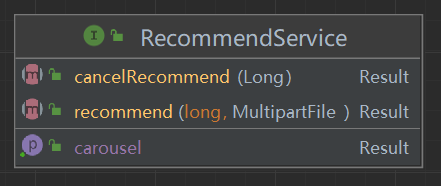
##### 2.11 OrderitemService接口设计



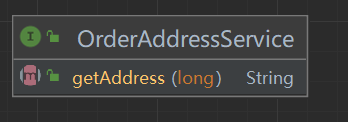
##### 2.12 CommentService接口设计



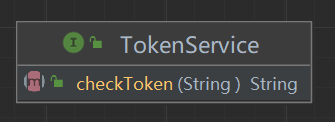
##### 2.13 RecommodityService接口设计



##### **2.14 OrderAddressService接口设计**



##### **2.15 TokenService接口设计**



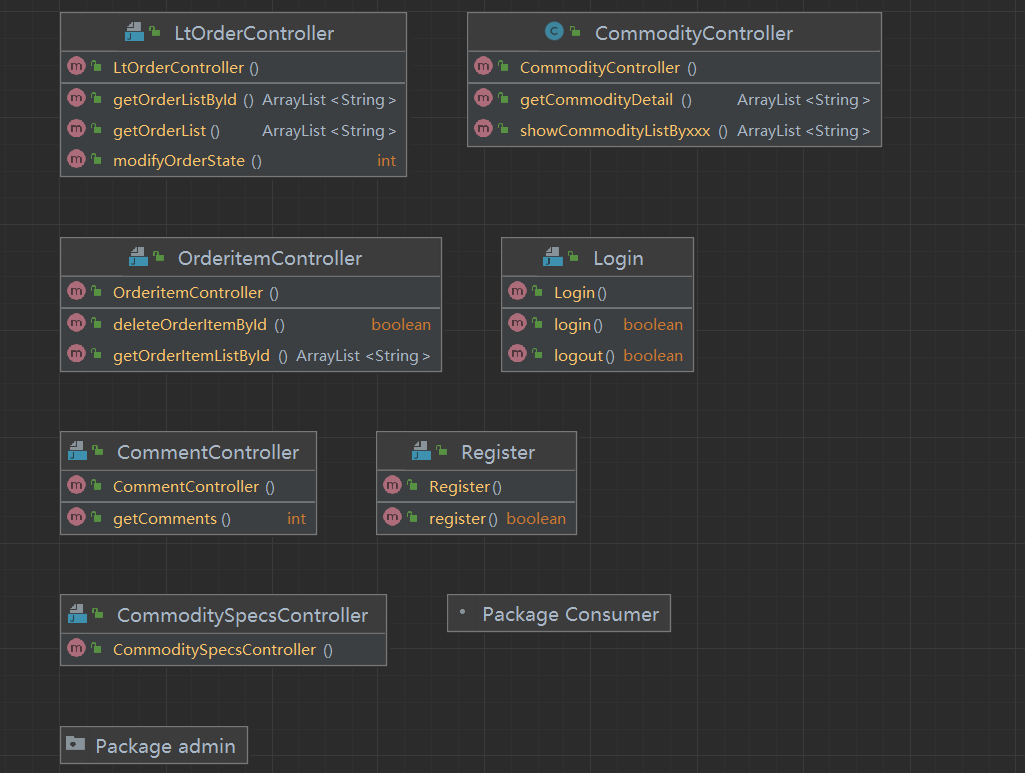
##### 2.16 业务接口实现类汇总

|  |  |
| --- | --- |
| 接口 | 实现类 |
| AdminService | AdminServiceImpl |
| UserService | UserServiceImpl |
| StoreService | StoreServiceIml |
| CommodityService | CommodityServiceImpl |
| CommoditySpecsService | CommoditySpecsServiceImpl |
| AttributeKeyService | AttributeKeyServiceImpl |
| CategoryService | CategoryServiceImpl |
| CartService | CartServiceImpl |
| LtOrderService | LtOrderServiceImpl |
| OrderitemService | OrderitemServiceImpl |
| CommentService | CommentServiceImpl |
| RecommendService | RecommendServiceimpl |
| OrderAddressService | OrderAddressServiceImpl |
| TokenService | TokenServiceImpl |

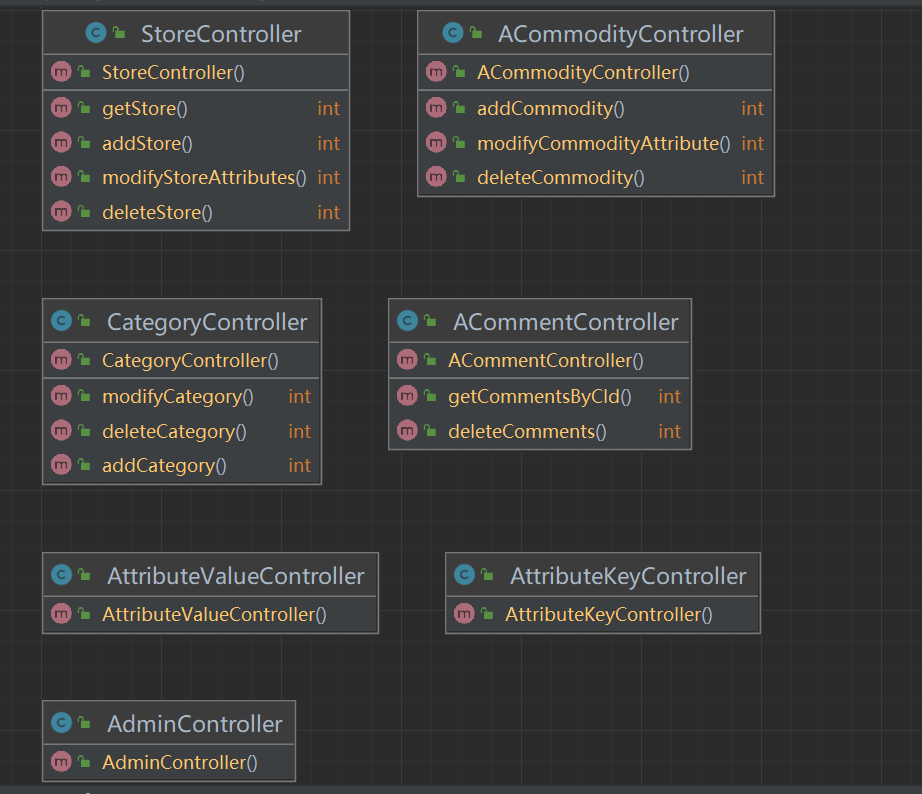
### 三、控制层设计

#### 1、类图设计

**com.letao.mall.controller**—这个包的类封装管理员和顾客通用的方法



**com.letao.mall.controller.admin**—这个包里的类封装专属管理员的方法



**com.letao.mall.controller.consumer**—这个包里的类封装专属顾客的方法

#### 

### 四、后台管理

#### 1、商品信息管理

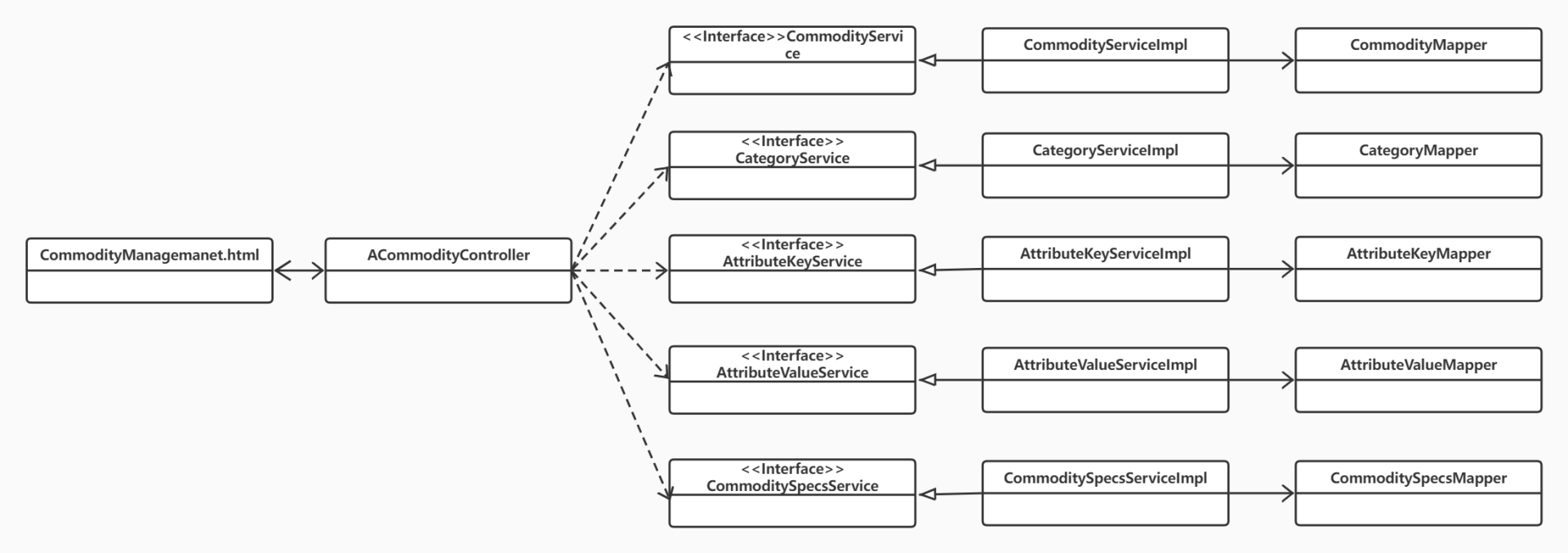
##### 1.1、功能汇总

管理系统中的所有商品的信息。

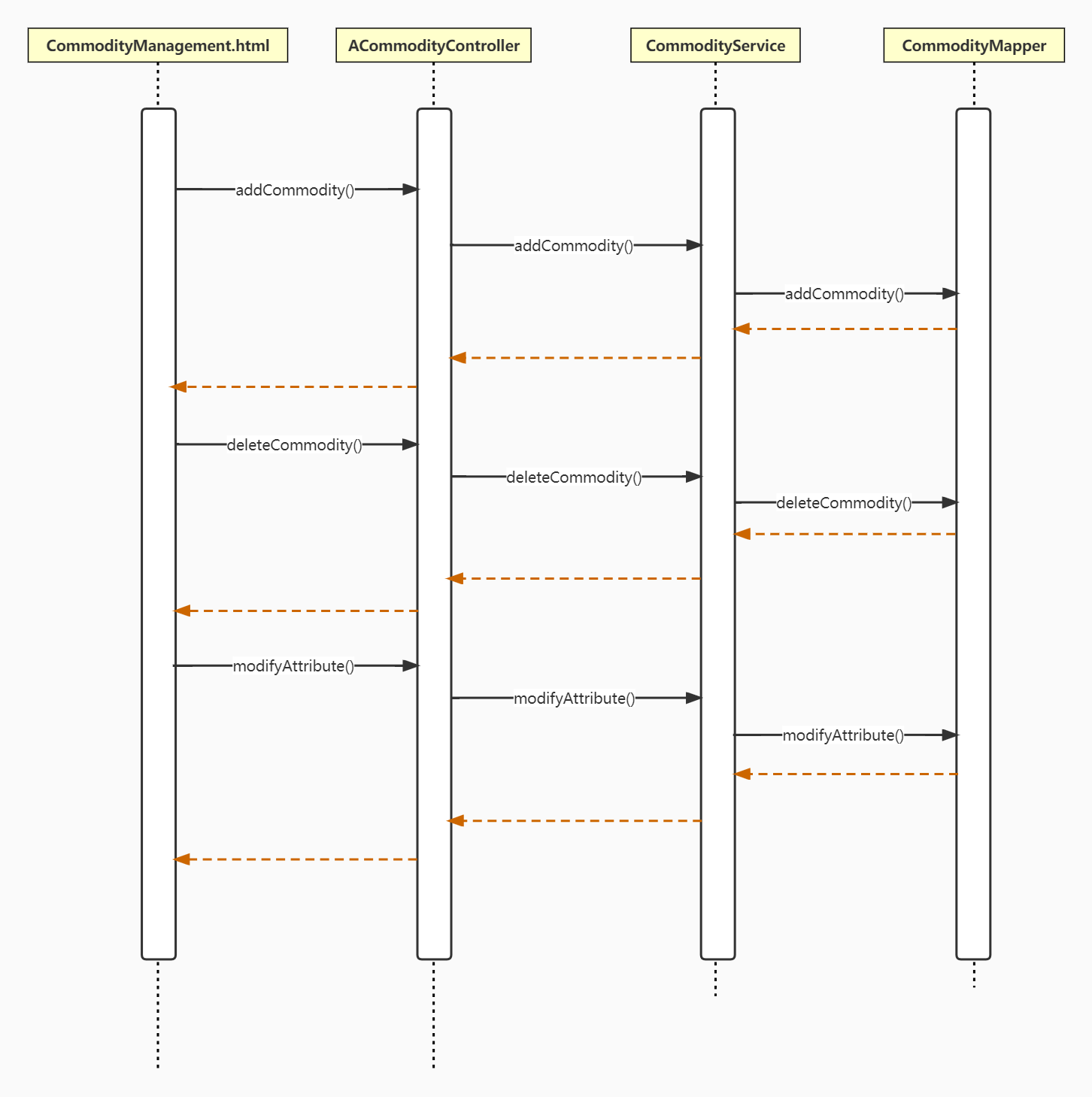
|  |  |
| --- | --- |
| 功能 | 描述 |
| 新增 | 新增商品 |
| 修改 | 修改商品信息 |
| 删除 | 删除商品(将商品置为失效状态，但不删除数据库记录) |
| 查询 | 根据现有查询条件，查询商品信息 |

##### 1.2、类图设计

###### 1.2.1、类关系结构图



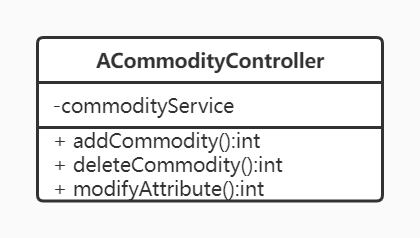
###### 1.2.2、时序图



##### 1.3、类的详细设计描述

###### 1.3.1、CommodityManagement.html描述（待添加）

###### 1.3.2、ACommodityController描述



详细描述：

|  |
| --- |
| ACommodityController商品管理Controller  包含响应商品信息增加修改查询等请求  实现【后台管理】中【商品管理】页面的所有功能 |
| 1. 成员变量   CommodityService commodityService 业务处理接口 |
| 1. public int addCommodity() 添加商品 |
| 1. public int deleteCommodity() 删除商品 |
| 1. public int modifyAttribute() 修改商品属性 |

##### 1.4、功能实现

ACommodityController类根据用户的不同请求调用不同的业务方法，CommodityService商品业务接口，其实现类负责商品信息的数据库操作。CategoryService分类管理业务接口，与AttributeKeyService，AttributeValueService，AttributeValueService，CommoditySpecsService共同组合实现对商品属性的修改。

在商品管理模块中，各个类间的相互方法调用，他们的方法调用顺序详见调用时序图。

#### **2.门店信息管理**

##### 2.1、功能汇总

管理系统中的所有门店的信息。

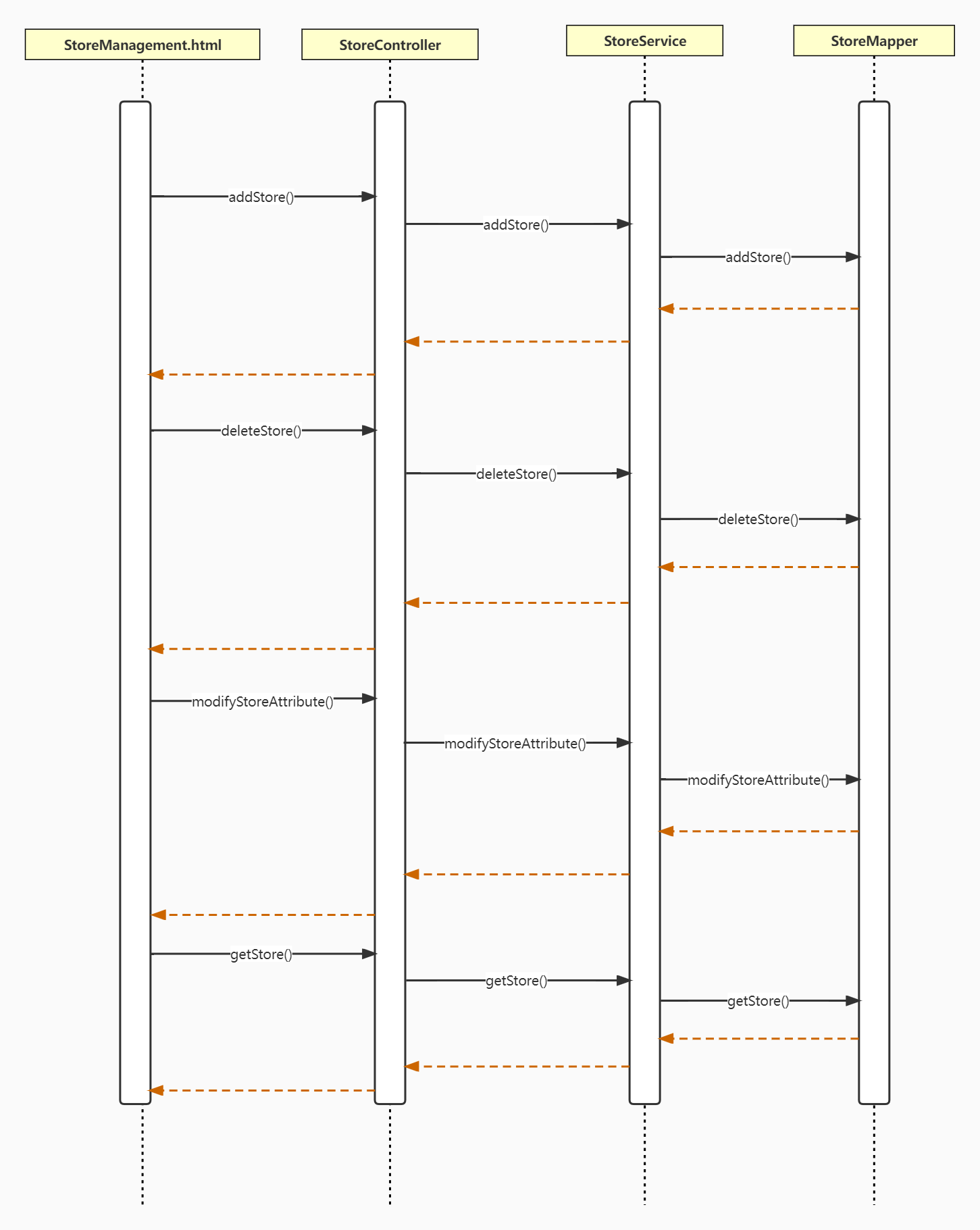
|  |  |
| --- | --- |
| 功能 | 描述 |
| 新增 | 新增门店 |
| 修改 | 修改门店信息 |
| 删除 | 删除门店 |
| 查询 | 根据现有查询条件，查询门店信息 |

##### 2.2、类图设计

###### 2.2.1、类关系结构图



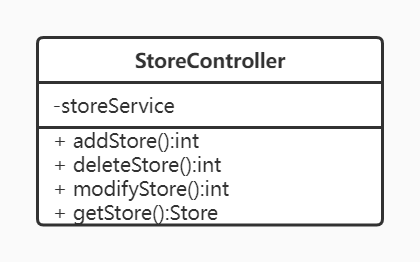
###### 2.2.2、时序图



##### 2.3、类的详细设计描述

###### 2.3.1、StoreManagement.html描述（代添加）

###### 2.3.2、StoreController描述



详细描述：

|  |
| --- |
| StoreController门店管理Controller  包含响应门店信息增加修改删除查询等请求  实现【后台管理】中【门店管理】页面的所有功能 |
| 1. 成员变量   StoreService storeService门店管理业务处理接口 |
| 1. public int addStore()添加门店 |
| 1. public int deleteStore() 删除门店 |
| 1. public int modifyStoreAttribute() 修改门店信息 |
| 1. public Store getStore() 查询门店信息 |

##### 2.4、功能实现

StoreController类根据用户的不同请求调用不同的业务方法，StoreService门店业务接口，其实现类负责门店信息的数据库操作。

在门店管理模块中，各个类间的相互方法调用，他们的方法调用顺序，详见调用时序图。

#### **3.商品分类管理**

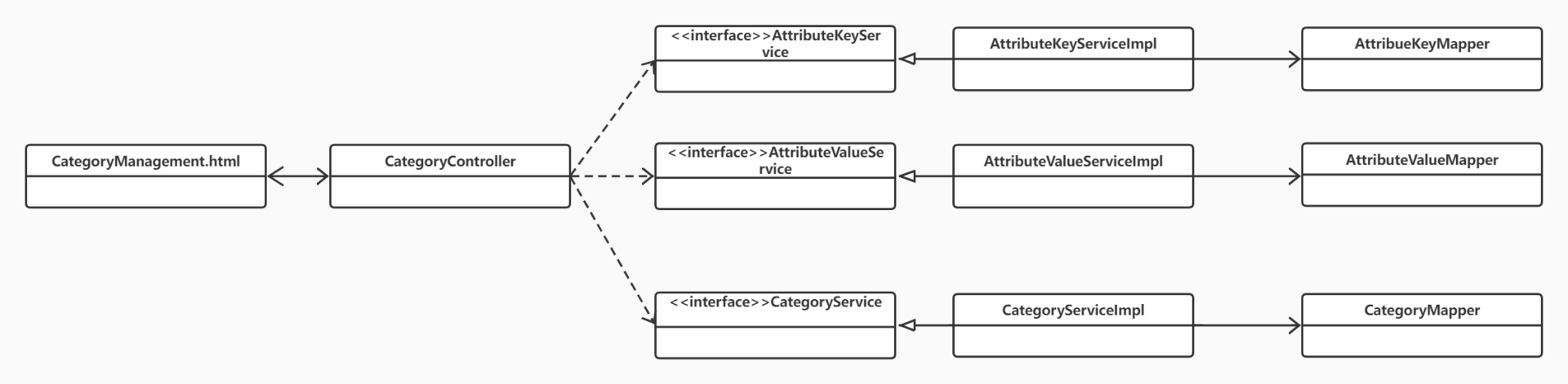
##### 3.1、功能汇总

管理系统中的所有标签信息。

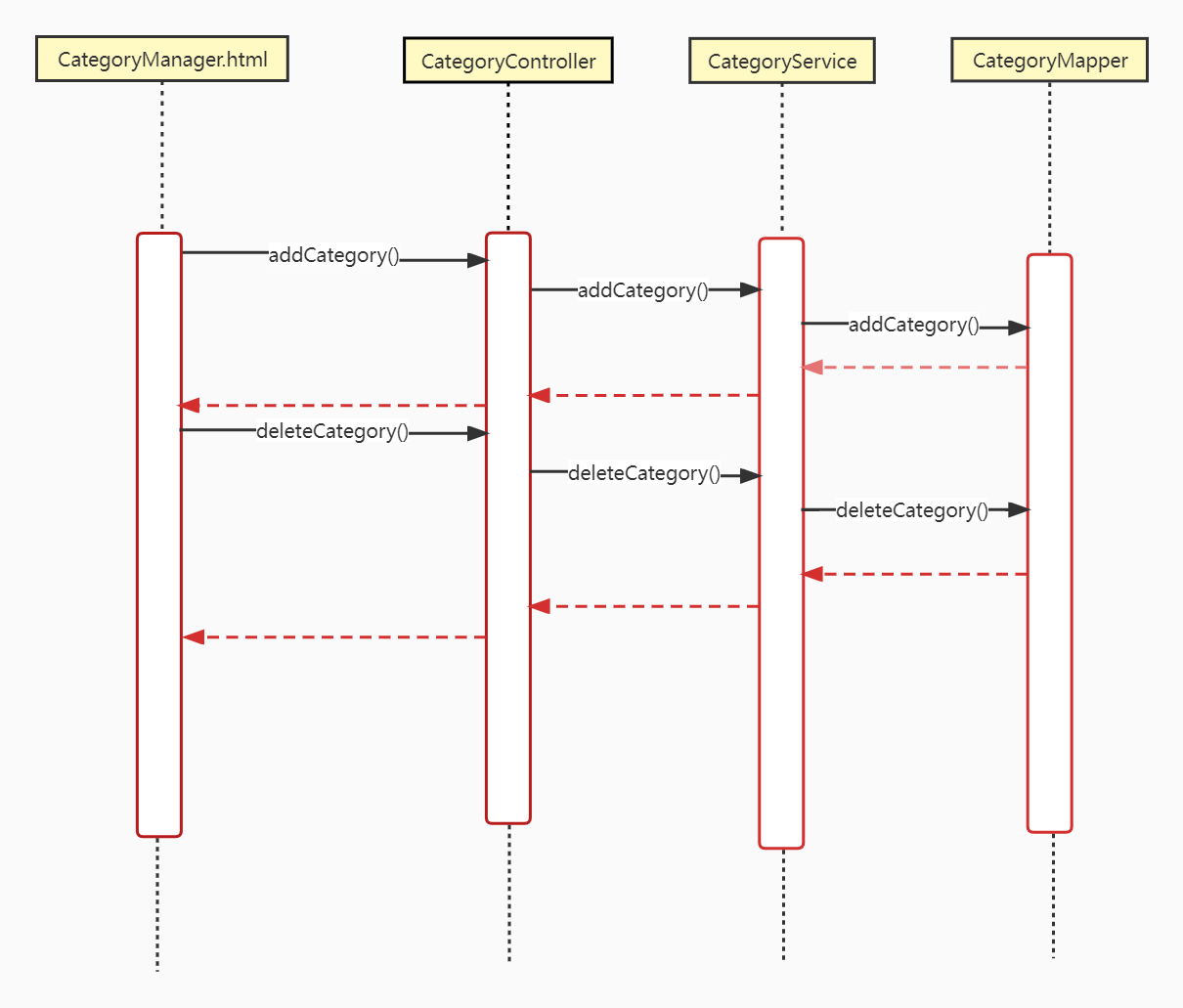
|  |  |
| --- | --- |
| 功能 | 描述 |
| 添加 | 添加一个新的标签 |
| 修改 | 修改标签内容 |
| 删除 | 删除标签 |
| 查询 | 根据现有查询条件，查询标签 |

##### 3.2、类图设计

###### 3.2.1、类关系结构图



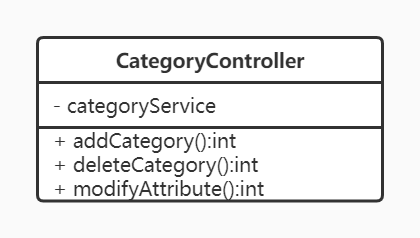
###### 3.2.2、时序图



##### 3.3、类的详细设计描述

###### 3.3.1、CategoryManagement.html描述述

###### 3.3.2、CategoryController描述



详细描述：

|  |
| --- |
| CategotyController管理商品分类标签的添加删除等操作 |
| 业务处理成员接口  CategoryService categoryService |
| 1.public int addCategory()：添加标签 |
| 2.public int deleteCategory() ：删除标签 |
| 3.public int modifyAttribute() ：更改标签 |

##### 3.4、功能实现

CategoryController类根据用户的不同请求调用不同的业务方法，实现用户需求。

#### **4.订单管理**

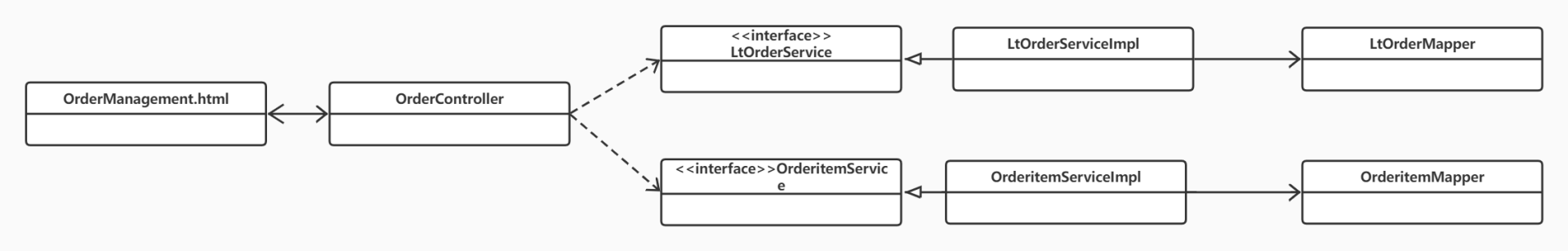
##### 4.1、功能汇总

管理系统中的所有订单信息。

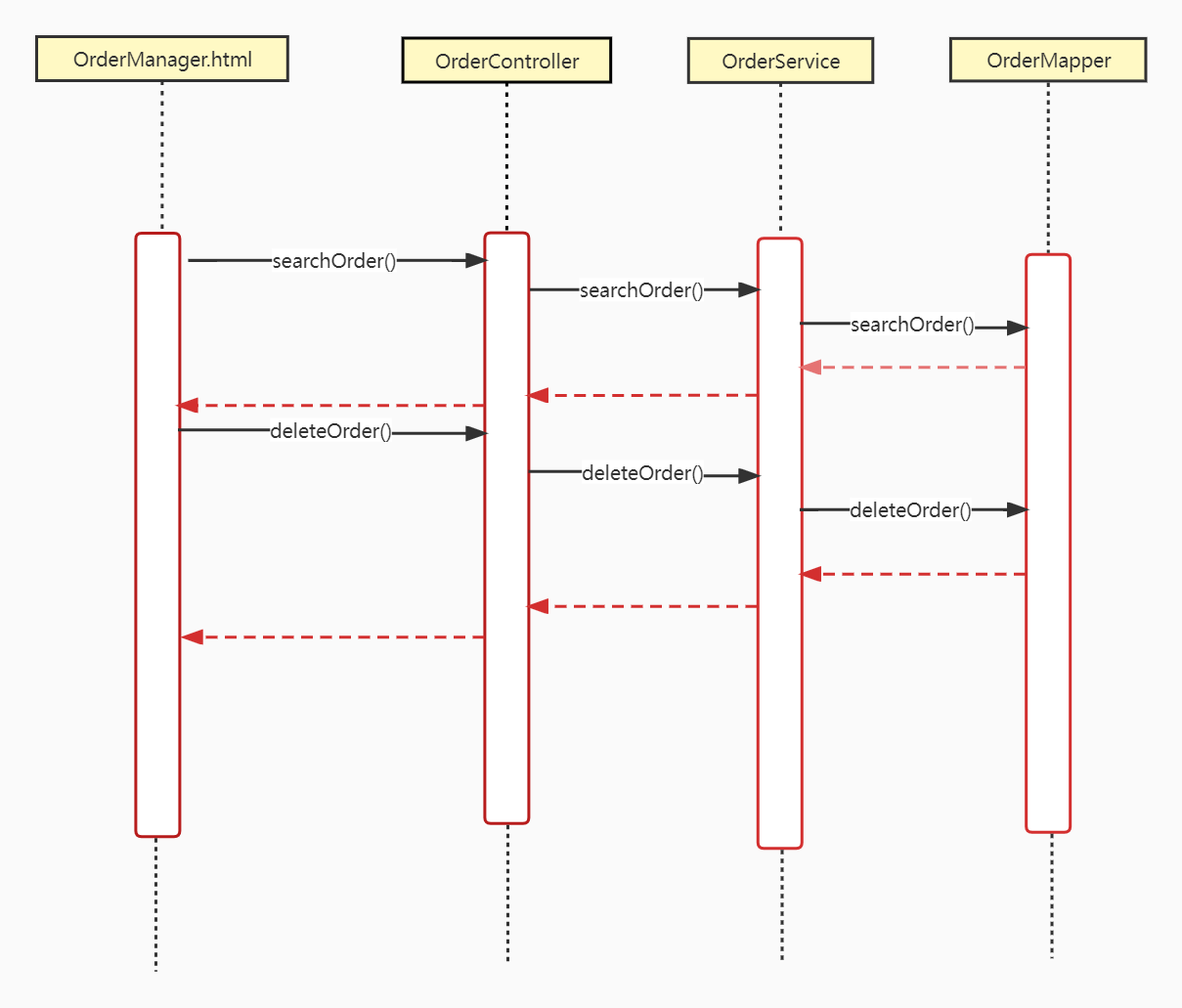
|  |  |
| --- | --- |
| 功能 | 描述 |
| 修改 | 修改订单状态 |
| 删除 | 删除订单 |
| 查询 | 根据现有查询条件，查询订单 |

##### 4.2、类图设计

###### 4.2.1、类关系结构图



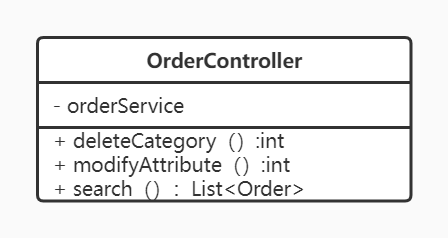
###### 4.2.2、时序图



##### 4.3、类的详细设计描述

###### 4.3.1、OrderManagement.html描述

###### 4.3.2、OrderController描述



详细描述：

|  |
| --- |
| OrderController管理订单的查询删除修改等操作 |
| 成员变量：LtOrderService ltOrderService |
| 1.public int deleteOrder()：删除订单 |
| 2.public List<order> search() ：查询订单列表 |
| 3.public int modifyAttribute() ：修改订单信息 |

##### 4.4、功能实现

OrderController类根据用户的不同请求调用不同的业务方法，实现用户需求。

### 五、用户功能

#### **1.商品浏览**

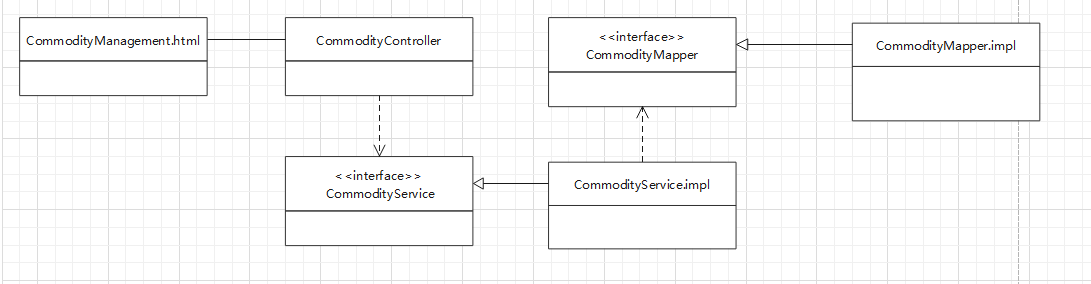
##### 1.1、功能汇总

管理系统中的所有用户购物的信息。

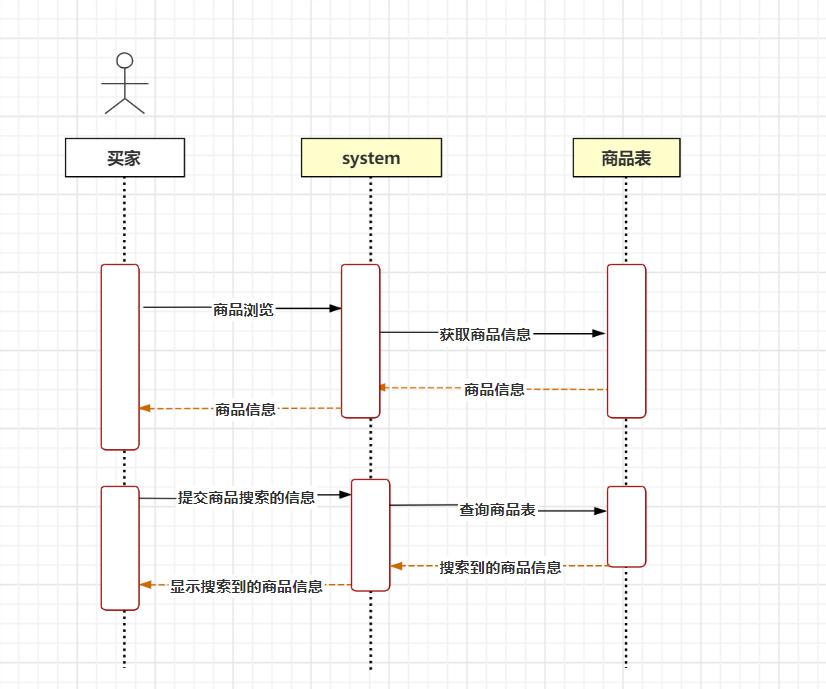
|  |  |
| --- | --- |
| 功能 | 描述 |
| 新增 | 添加商品到购物车 |
| 修改商品 | 修改选定页面中商品信息 |
| 删除商品 | 删除选定页面中商品 |
| 浏览商品 | 根据现有查询条件，查询浏览商品详细信息 |

##### 1.2、类图设计

###### 1.2.1、类关系结构图



###### 1.2.2、时序图



##### 1.3、类的详细设计描述

###### 1.3.1、CommodityManagement.html描述

CommodityManagement.html是顾客管理商品页面，是顾客在用户功能模块中的顾客管理商品页面可以直接看到的用户界面，本页面包含查询列表和商品选定两部分，在该页面用户执行的每一项操作触发的请求都会由服务器转发到CommodityController用户管理Controller该类处理。

具体页面设计如下图所示：

###### 1.3.2、CommodityController描述

详细描述：

|  |
| --- |
| CommodityController用户管理Controller  包含响应用户信息增加修改查询等请求  实现【商品浏览】中【商品搜索】和【商品选定】页面的所有功能 |
| 1. 成员变量 |
| 1. public void validateOptCommodityManagement()添加修改选定商品信息的方法 |
| 1. public String sortCommodityByScore(String category) 将商品按照评分正常排序，最开始展现在顾客面前的顺序 |
| 1. public String sortCommodityBySales() 将商品按照销量分类 |
| 1. public String sortCommodityByPriceHigh() 将商品按照价格从高到低排序 |
| 1. public String sortCommodityByPriceLow（）将商品按照价格从低到高排序 |
| 1. public String addCommodityToCart()添加商品到购物车 |
| 1. public Int setCommodityNum() 修改选定商品时的商品数量 2. public String GetCommodityDetail() 查看选定商品的详细信息 3. public String showCommodityToCustomer() 展示商品给顾客看 |

##### 1.4、功能实现

CommodityController类即业务外观方法根据用户对商品的不同请求调用不同的业务逻辑方法，CommodityService商品业务逻辑接口，其实现类CommodityService.impl负责用户业务逻辑操作。通过调用CommodityMapper商品数据库接口进行，其实现类CommodityMapper.impl负责商品数据库操作

在顾客商品管理模块中，各个类间的相互方法调用，他们的方法调用顺序，在这里为大家提供了时序图，详见调用时序图5.1.1.2.2。

#### **2.商品购买**

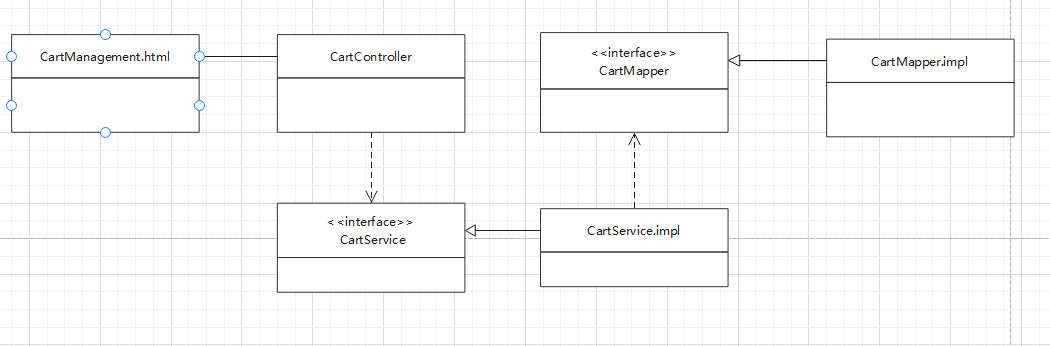
##### 2.1、功能汇总

管理系统中的购物车中商品的信息。

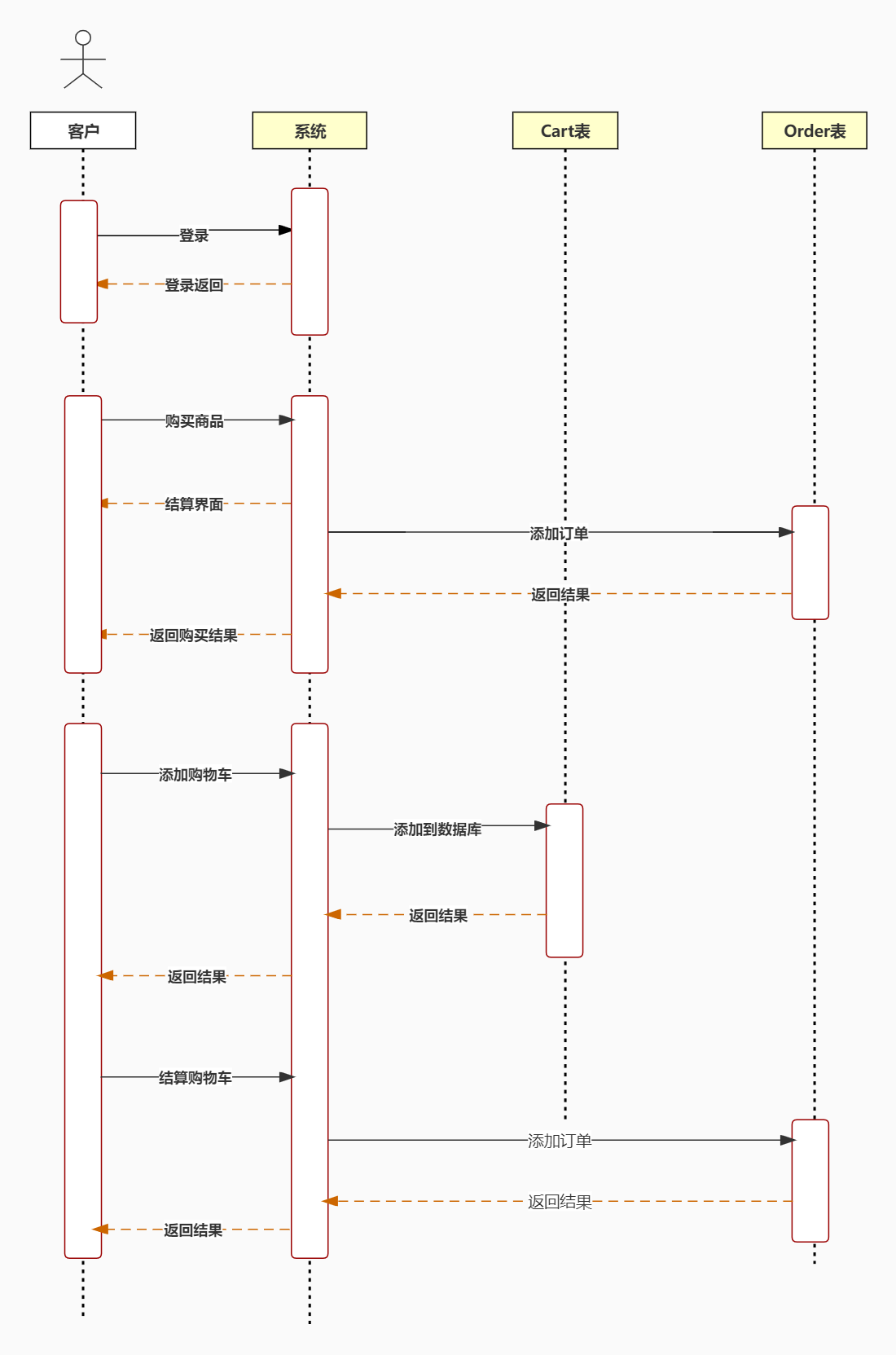
|  |  |
| --- | --- |
| 功能 | 描述 |
| 新增 | 付款将商品加入到订单 |
| 修改 | 修改购物车中商品信息 |
| 删除 | 删除购物车中商品 |
| 查询 | 根据现有查询条件，查询购物车中商品信息 |

##### 2.2、类图设计

###### 2.2.1、类关系结构图



###### 2.2.2、时序图



##### 2.3、类的详细设计描述

###### 2.3.1、CartManagement.html描述

CartManagement.html是购物车管理页面，是顾客在顾客管理模块中的顾客消费管理页面可以直接看到的顾客购物车界面，本页面包含商品删除和商品修改两部分，在该页面用户执行的每一项操作触发的请求都会由服务器转发到CartController购物车管理Controller该类处理。

具体页面设计如下图所示：

###### 2.3.2、CartController描述

详细描述：

|  |
| --- |
| CartController购物车管理Controller  包含修改和删除购物车内商品信息、将购物车内商品付款生成订单等请求  实现【购物车管理】页面的所有功能 |
| 1. 成员变量 |
| 1. public void validateOptCartManagement()添加修改购物车信息，页面输入校验方法 |
| 1. public void deleteCommodity()删除购物车中商品 |
| 1. public Int addNum()增加购物车中商品数量 |
| 1. public Int reduce()减少购物车中商品的数量 |
| 1. public String showCart()展示购物车 |
| 1. public String toOrder()生成订单 |
|  |

##### 2 .4、功能实现

CartController类即业务外观根据用户需求CartManagement.html即网页端调用不同的CartService即业务逻辑类接口，其实现类CartService.impl进行业务逻辑操作。通过调用CartMapper接口，其实现类CartMapper.impl进行数据库操作

在用户管理模块中，各个类间的相互方法调用，他们的方法调用顺序，在这里为大家提供了时序图，详见调用时序图5.2.1.2.2。

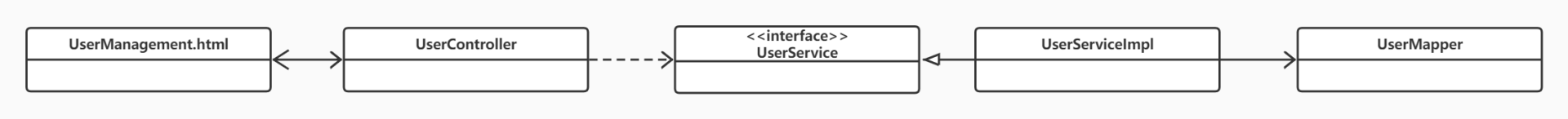
#### **3.用户账号管理**

#### 3.1、功能汇总

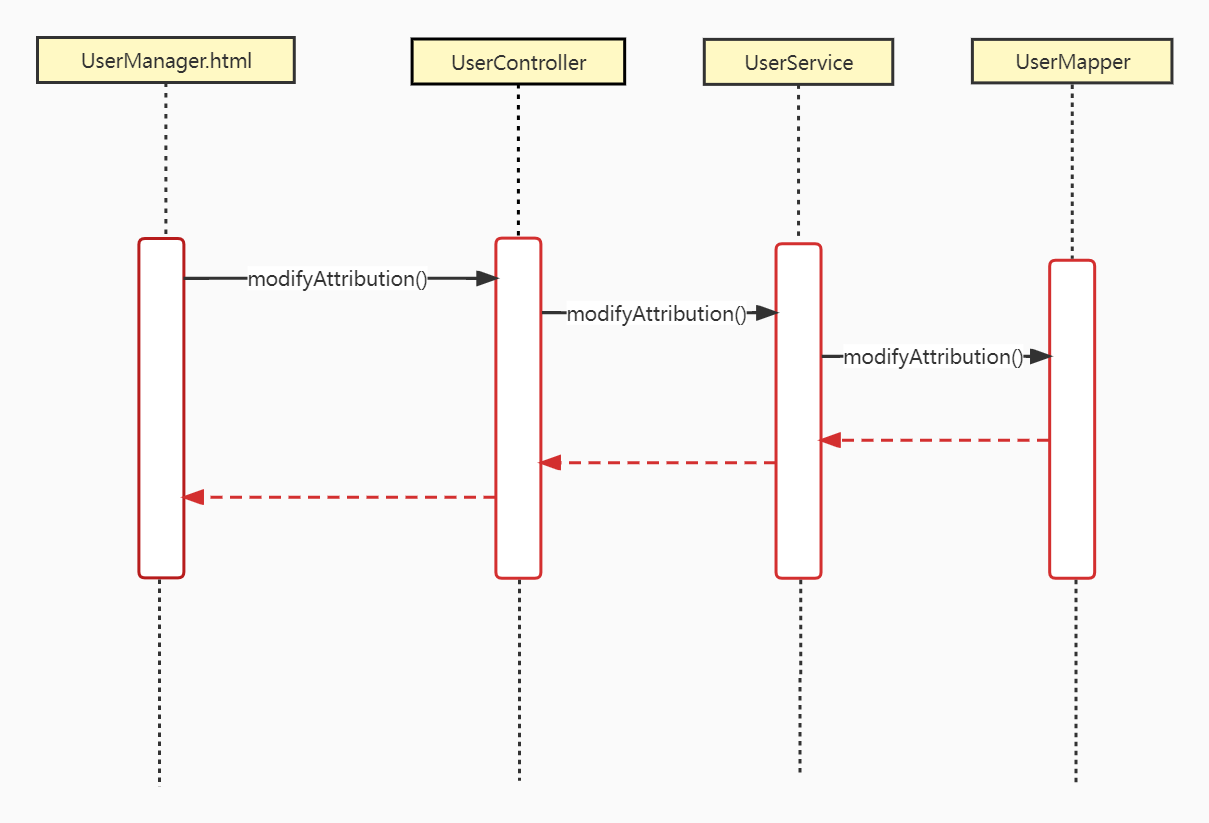
|  |  |
| --- | --- |
| 功能 | 描述 |
| 修改 | 用户修改账号信息 |

##### 3.2、类图设计

###### 3.2.1、类关系结构图



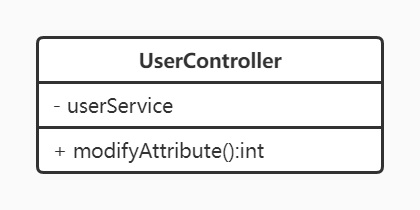
###### 3.2.2、时序图



##### 3.3、类的详细设计描述

###### 3.3.1、UserManagement.html描述

###### 2.3.2、UserController描述



详细描述：

|  |
| --- |
| UserController用户可修改账号信息 |
| 成员变量  UserService userService |
| 1.public int modifyAttribute()：修改用户信息 |

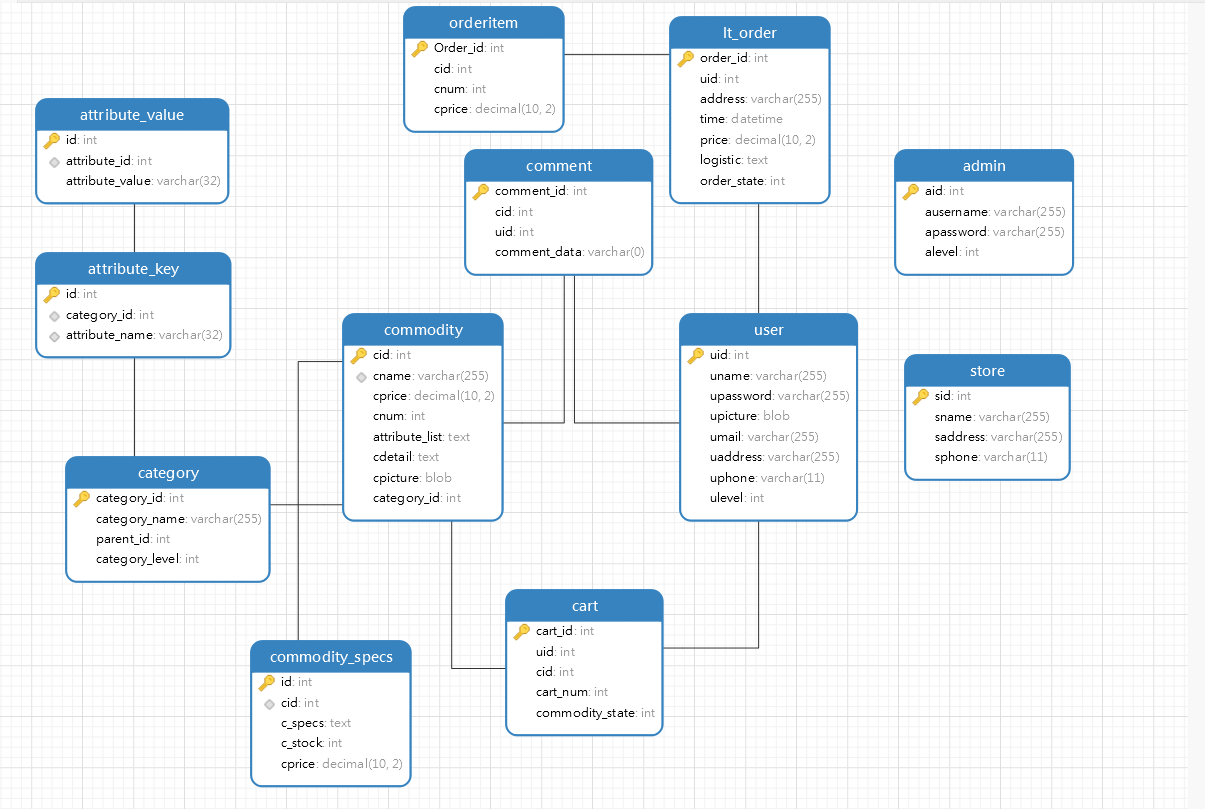
##### 3.4、功能实现

UserController类根据用户的不同请求调用不同的业务方法，实现用户需求。

#### 

# 第六部分 数据库设计

### 数据库整体结构图



### 二、数据库表格清单

|  |  |  |
| --- | --- | --- |
| 序号 | 表名 | 注释 |
| 1 | store | 门店表 |
| 2 | admin | 管理员表 |
| 3 | commodity | 商品表 |
| 4 | commodity\_specs | 商品规格表 |
| 5 | attribute\_key | 属性key表 |
| 6 | attribute\_value | 属性value表 |
| 7 | category | 分类表 |
| 8 | cart | 购物车表 |
| 9 | lt\_order | 订单表 |
| 10 | orderItem | 订单详情表 |
| 11 | user | 顾客表 |
| 12 | comment | 评论表 |

#### 1、store表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | sid | int |  |
| 2 | sname | VARCHAR(255) | 门店名称 |
| 3 | saddress | VARCHAR(255) | 门店地址 |
| 4 | sphone | VARCHAR(11) | 电话 |

#### 2、admin表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | aid | int |  |
| 2 | ausername | VARCHAR(255) | 管理员名 |
| 3 | apassword | VARCHAR(255) | 管理员密码 |
| 4 | alevel | int | 权限级别 |

#### 3、commodity表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | cid | int |  |
| 2 | cname | VARCHAR(255) | 商品名 |
| 3 | cprice | decimal(10,2) | 商品价格 |
| 4 | cnum | int | 库存 |
| 5 | cdetail | text | 商品详情 |
| 6 | cpicture | blob | 商品图片 |
| 7 | attribute\_list | text | 规格列表 |
| 8 | category\_id | int | 商品标签（类型） |
| 9 | csales | int |  |
| 10 | is\_recommend |  |  |

#### 4、commodity\_specs表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | id | int | id |
| 2 | cid | int | 商品Id |
| 3 | c\_specs | text | 商品规格 |
| 4 | c\_stock | int | 商品库存 |
| 5 | cprice | decimal(10,2) | 商品价格 |

#### 5、attribute\_key表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | id | int | 属性Id |
| 2 | category\_id | int | 分类Id |
| 3 | attribute\_name | VARCHAR(32) | 属性名 |

#### 6、attribute\_value表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | id | int | 属性值Id |
| 2 | attribute\_id | int | 属性Id |
| 3 | attribute\_value | VARCHAR(32) | 属性值 |

#### 7、category表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | category\_id | int | 分类id |
| 2 | category\_name | VARCHAR(255) | 分类名称 |
| 3 | parent\_id | int | 父分类的id |
| 4 | category\_level | int | 分类层级 |

#### 8、cart表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | cart\_id | int |  |
| 2 | uid | int | 用户Id |
| 3 | cs\_id | int | 商品Id |
| 4 | cart\_num | int | 购买数量 |
| 5 | commodity\_state | int | "1"表示“可以购买”，“0”表示“商品已失效”） |

#### 9、lt\_order表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | order\_id | int | 订单id |
| 2 | uid | int | 用户id |
| 3 | address | VARCHAR(255) | 订单地址 |
| 4 | price | decimal(10,2) | 订单价格 |
| 5 | time | datetime | 下单时间 |
| 6 | logistic | text | 物流详情 |
| 7 | order\_state | int | 订单状态（是否完成） |

#### 10、orderitem表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | id | long |  |
| 2 | order\_id | long |  |
| 3 | cid | long | 商品Id |
| 4 | cnum | long | 单个商品购买数量 |
| 5 | cprice | decimal(10,2) | 商品单/总价(同一件商品的总价) |
| 6 | cs\_id | long | 商品规格id |
| 7 | cpicture | String | 图片 |

#### 11、user表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | uid | int |  |
| 2 | uname | VARCHAR(255) | 用户名 |
| 3 | upassword | VARCHAR(255) | 用户密码 |
| 4 | upicture | blob | 头像 |
| 5 | umail | VARCHAR(255) | 邮箱 |
| 6 | uaddress | VARCHAR(255) | 收货地址 |
| 7 | uphone | VARCHAR(11) | 电话 |
| 8 | ulevel | int | vip等级（可修改） |

#### 12、comment表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | comment\_id | int |  |
| 2 | cid | int | 商品Id |
| 3 | uid | int | 单个商品购买数量 |
| 4 | comment\_data | varchar(0) | 商品单/总价(同一件商品的总价) |

#### 13、collection表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | id | long |  |
| 2 | cid | int | 商品Id |
| 3 | uid | int | 用户Id |

#### 14、order\_address表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 列名 | 数据类型 | 注释 |
| 1 | id | long |  |
| 2 | uid | long | 用户Id |
| 3 | name | String | 用户姓名 |
| 4 | address | String | 地址 |
| 5 | phone | String | 电话 |

#### 15、系统管理外键清单

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 外键名称 | 父表 | 父键列 | 子表 | 外键列 | 关系 | 说明 |
| category\_id | category | category\_id | commodity | category\_id | 0..n | 一个分类可以有多个商品 |
| cid | commodity | cid | commodity\_specs | cid | 0..n | 一个商品可以有多个规格 |
| category\_id | category | category\_id | attribute\_key | category\_id | 0..n | 一个分类可以有多个属性 |
| attribute\_id | attribute\_key | id | attribute\_value | attribute\_key | 0..n | 一个属性可以有多个属性值 |
| uid | user | uid | cart | uid | 1..1 | 一个用户可以有一个购物车 |
| cid | cart | cid | commodity | cid | 0..n | 一个购物车可以有多个商品 |
| uid | user | uid | lt\_order | uid | 0..n | 一个顾客可以有多个订单 |
| cid | orderitem | cid | commodity | cid | 0..n | 一个订单可以有多个商品 |
| uid | user | uid | comment | uid | 0..n | 一个用户可以有多条评论 |
| cid | commodity | cid | comment | cid | 0..n | 一个商品可以有多条评论 |

# 第七部分 补充设计和说明

### 一、编译运行环境设计

jdk

mysql

vue

### 二、包路径与WEB目录结构设计

#### 1、包路径设计

后端项目：项目目录

前端项目：项目目录