**面向对象的软件规格说明书**

编写人员：杨顺杰、魏泽弘、高时玉、赵任生、龚云基、杨笑千

指导老师：张大林

编写日期：2022.5.15

本说明书版权与一切解释权归蓝瞳团队所有

目录

[1 概述 3](#_Toc20972)

[1.1 系统简述 3](#_Toc12802)

[1.2 软件设计目标 3](#_Toc3900)

[1.2.1 功能要求 3](#_Toc13067)

[1.2.2 性能要求 3](#_Toc14682)

[1.2.3 输入要求 4](#_Toc9379)

[1.2.4 输出要求 4](#_Toc11735)

[1.2.5 安全与保密要求 4](#_Toc8191)

[1.3 参考资料 4](#_Toc4973)

[1.4 修订版本记录 4](#_Toc16862)

[2 术语表 5](#_Toc10506)

[3 用例 7](#_Toc13821)

[3.1 用户信息模块 7](#_Toc14830)

[3.2 交易模块 10](#_Toc5929)

[3.3 展示模块 13](#_Toc18432)

[3.4 社区模块 15](#_Toc4306)

[3.5 活动模块 17](#_Toc17801)

[4 概要设计 19](#_Toc22095)

[4.1 简述 19](#_Toc16877)

[4.2 系统结构设计 19](#_Toc25910)

[4.2.1 系统功能架构 19](#_Toc18723)

[4.2.2 系统软件架构 21](#_Toc29552)

[4.2.3 系统类对象图 23](#_Toc29691)

[4.2.4 系统包图 28](#_Toc12019)

[4.2.5 系统构件（模块，组件）图 30](#_Toc22382)

[4.3 约束和假定 31](#_Toc25673)

[5 对象（类）描述 32](#_Toc13192)

[5.1 子系统1【构件1】中的对象 32](#_Toc2634)

[5.1.1 对象：GoodsCarBean 32](#_Toc1722)

[5.1.1.1 属性描述： 32](#_Toc25781)

[5.1.1.2 方法描述： 33](#_Toc4238)

[5.1.2 对象：ShopContextBean 35](#_Toc15506)

[5.1.2.1 属性描述： 35](#_Toc26382)

[5.1.2.2 方法描述： 36](#_Toc25716)

[5.1.3 对象：ShopInformationBean 37](#_Toc7266)

[5.1.3.1 属性描述： 37](#_Toc12322)

[5.1.3.2 方法描述： 38](#_Toc7799)

[5.1.4 对象：UserWantBean 40](#_Toc556)

[5.1.4.1 属性描述： 40](#_Toc2210)

[5.1.4.2 方法描述： 41](#_Toc26481)

[5.1.5 对象：ForgetController 43](#_Toc8096)

[5.1.5.1 属性描述： 43](#_Toc14253)

[5.1.5.2 方法描述： 43](#_Toc12904)

[5.1.6 对象：GoodsController 44](#_Toc11132)

[5.1.6.1 属性描述： 44](#_Toc11939)

[5.1.6.2 方法描述： 44](#_Toc7033)

[5.1.7 对象：RegisterController 45](#_Toc28100)

[5.1.7.1 属性描述： 46](#_Toc31080)

[5.1.6.2 方法描述： 46](#_Toc10271)

[5.1.7 对象：BaseResponse 46](#_Toc23428)

[5.1.4.1 属性描述： 46](#_Toc9578)

[5.1.4.2 方法描述： 47](#_Toc29545)

[5.2 推荐系统模块【算法的原理流程描述、各种接口类】 49](#_Toc16011)

[6 动态模型 50](#_Toc21816)

[6.1 场景（Scenarios） 50](#_Toc8871)

[6.1.1 登陆注册场景 50](#_Toc22754)

[6.1.2 商品模糊查询 51](#_Toc8649)

[6.1.2用户信息修改 52](#_Toc28348)

[6.1.2交易商品 53](#_Toc4506)

[6.2 状态图 53](#_Toc18894)

[6.2.1 状态图1： 54](#_Toc17662)

[6.2.2 状态图2： 54](#_Toc5494)

[6.2.3 状态图3： 55](#_Toc5536)

[6.2.4 状态图4： 55](#_Toc22878)

[6.2.5 状态图5： 55](#_Toc13986)

[7 数据库设计 56](#_Toc1487)

[7.1 数据库说明 56](#_Toc528)

[7.2 数据库结构 56](#_Toc13283)

[7.3 表结构 57](#_Toc16083)

[8 非功能性设计 61](#_Toc16438)

[8.1 性能需求 61](#_Toc2857)

[8.1.1 处理能力 61](#_Toc23535)

[8.1.2 响应时间 61](#_Toc14319)

[8.2 安全性需求 61](#_Toc20327)

[9 辅助文档 62](#_Toc840)

[10 词汇索引 62](#_Toc17079)

# 1 概述

## 1.1 系统简述

**项目名称**：蓝瞳校园二手物品交易系统

**项目目的和意义**：支持勤俭节约；倡导互惠互利；开展爱心义卖；践行低碳环保；做到物尽其用；支持以物换物；实现智能管理（机器学习、人工智能）；达到安全平稳。

**本《软件规格说明书》的预期读者是：**

* 二手交易平台平台开发经理
* 项目组所有人员
* 测试组人员
* 开发团队授权调阅本文档的其他人员

## 1.2 软件设计目标

## 1.2.1 功能要求

此系统需要完成的功能主要有：用户个人信息管理、二手交易平台（发布信息，修改信息，删除信息，浏览信息，搜索信息等）、用户交流功能（发表留言，查看留言，删除留言等）、用户指南（添加二手指南，查看二手指南，修改二手指南，删除二手指南）、智能推荐。

## 1.2.2 性能要求

为了能够为用户提供充足的物品信息和快捷的处理手段，用户使用时系统要保证运行稳定、效率高，能够快速做出响应，迅速处理各项数据、信息，显示出所需信息，所以要有一定的可扩展性和灵活性，当毕业生处理闲置物品高峰期等大量人员操作时，做到节省录入时间、提高交易效率和安全性，系统要保证操作简便、快捷。

## 1.2.3 输入要求

对于卖家，需要上传物品的类型、价格、图片、完好程度等信息；对于买家，可以在物品或者卖家评论区发布自己的意见和建议。

## 1.2.4 输出要求

卖家上传的物品信息会在系统展示出来供用户挑选。界面做到简洁明了，易于操作。

## 1.2.5 安全与保密要求

由于校园二手交易系统涉及到学生利益问题，所以系统的安全性一定是不容忽视的，系统要做到用户交易过程公平公正，并且用户的购买信息、浏览记录和个人信息等隐私可以自由选择保密，不对任何人公布。

## 1.3 参考资料

　 [1]高美珍，洪家平.高校二手物品交易系统的设计与实现[J].湖北师范大学学报（自然科学版），2018，38（04）：65-69.

[2]曲蕴慧.校园二手交易平台的构建与实现[J].电子设计工程，2014，22（06）：70-72.

[3]庄彦，未培.基于校园网的二手交易平台构建与实践[J].集宁师范学院学报，2016，38（02）：40-43.

[4]蓝瞳校园二手交易平台需求规格说明书

## 1.4 修订版本记录

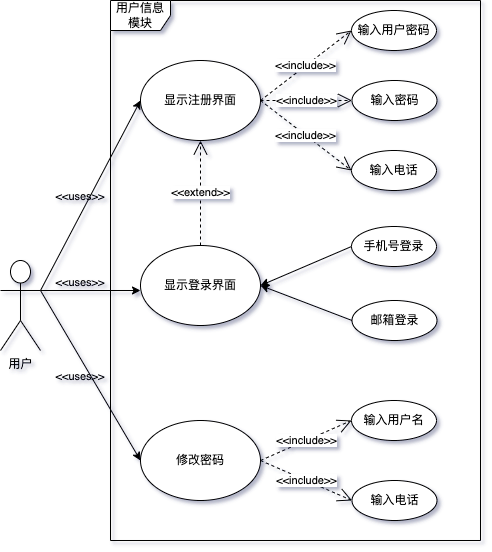
|  |  |  |
| --- | --- | --- |
| 时期 | 修改人 | 修改内容 |
| 2022.5.19 | 杨顺杰 | 系统概述 |
| 2022.5.19 | 魏泽弘 | 构件图，类图 |
| 2022.5.19 | 高时玉 | 架构图，通信图 |
| 2022.5.19 | 龚云基 | 用例图，活动图 |
| 2022.5.19 | 赵任生 | 架构图，通信图 |
| 2022.5.19 | 杨笑千 | 包图，状态图 |

# 2 术语表

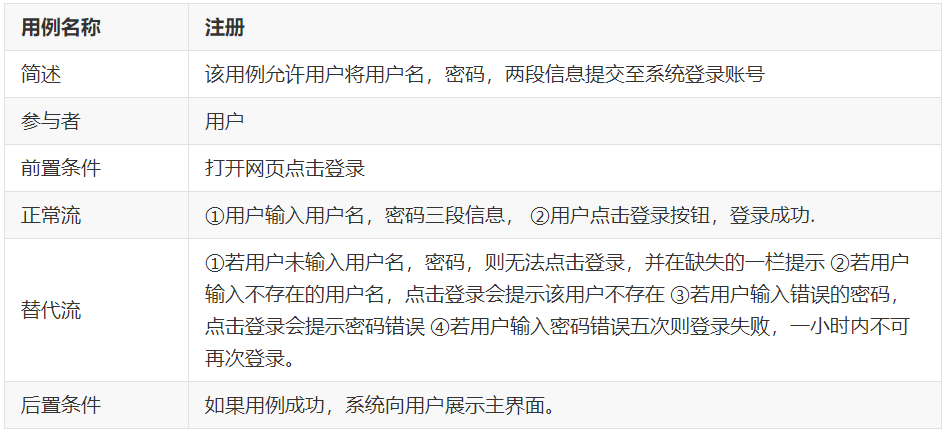
|  |  |
| --- | --- |
| **概念** | **具体定义** |
| 活动图 | 活动图是UML用于对系统的动态行为建模的另一种常用工具，它描述活动的顺序，展现从一个活动到另一个活动的控制流。活动图在本质上是一种流程图。虽然活动图与状态图都是状态机的表现形式，但是两者还是有本质区别：活动图着重表现从一个活动到另一个活动的控制流，是内部处理驱动的流程；而状态图着重描述从一个状态到另一个状态的流程，主要有外部事件的参与。 |
| 用例图 | 用例图（英语：use case diagram）是用户与系统交互的最简表示形式，展现了用户和与他相关的用例之间的关系。通过用例图，人们可以获知系统不同种类的用户和用例。用例图也经常和其他图表配合使用。 |
| 时序图 | 时序图（Sequence Diagram），又名序列图、循序图，是一种UML交互图。它通过描述对象之间发送消息的时间顺序显示多个对象之间的动态协作。它可以表示用例的行为顺序，当执行一个用例行为时，其中的每条消息对应一个类操作或状态机中引起转换的触发事件。 |
| 类图 | 类图(Class diagram)由许多（静态）说明性的模型元素（例如类、包和它们之间的关系，这些元素和它们的内容互相连接）组成。类图可以组织在（并且属于）包中，仅显示特定包中的相关内容。类图(Class diagram)是最常用的UML图，显示出类、接口以及它们之间的静态结构和关系；它用于描述系统的结构化设计。类图(Class diagram)最基本的元素是类或者接口。类图主要用在面向对象软件开发的分析和设计阶段，描述系统的静态结构。类图图示了所构建系统的所有实体、实体的内部结构以及实体之间的关系。即．类图中包含从用户的客观世界模型中抽象出来的类、类的内部结构和类与类之间的关系。它是构建其他设计模型的基础，没有类图，就没有对象图、状态图、协作图等其他UMI．动态模型图．也就无法表示系统的动态行为。类图也是面向对象编程的起点和依据。类图用于描述系统中所包含的类以及它们之间的相互关系，帮助人们简化对系统的理解，它是系统分析和设计阶段的重要产物，也是系统编码和测试的重要模型依据。 |
| 状态图 | 状态图(Statechart Diagram)是描述一个实体基于事件反应的动态行为，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。通常我们创建一个UML状态图是为了以下的研究目的：研究类、角色、子系统、或组件的复杂行为。UML通过使用状态图来支持基于事件的模型，状态图用来来描述一个类对象在不同用例间状态的迁移。当一个用例或某个事件发生时，类对象的状态就会发生迁移，状态图有助于分析人员审核业务逻辑，以及完善静态模型。 |
| 数据流图 | 数据流图或数据流程图（Data Flow Diagram），缩写为DFD。数据流图DFD是描述系统中数据流程的一种图形工具，它标志了一个系统的逻辑输入和逻辑输出，以及把逻辑输入转换逻辑输出所需的加工处理。DFD显示系统将输入和输出什么样的信息，数据如何通过系统前进以及数据将被存储在何处。它不显示关于进程计时的信息，也不显示关于进程将按顺序还是并行运行的信息，而不像传统的关注控制流的结构化流程图，或者UML活动工作流程图，它将控制流和数据流作为一个统一的模型。数据流图从数据传递和加工的角度，以图形的方式刻画数据流从输入到输出的移动变换过程。 |
| 包图 | 包图（package Diagram）是由若干个包以及包之间的关系组成，在 UML 中用类似于[文件夹](https://baike.baidu.com/item/%E6%96%87%E4%BB%B6%E5%A4%B9/7031091)的符号表示的模型元素的组合。系统中的每个元素都只能为一个包所有，一个包可嵌套在另一个包中。使用包图可以将相关元素归入一个系统。一个包中可包含附属包、图表或单个元素。包是一种分组机制，将同类的类、对象、模型元素放在一起,形成高内聚，低耦合的类集合，可以说，一个包相当于一个子系统。 |
| 构件 | 在UML2.0中，构件被认为是在一个系统或子系统中独立的封装单位，它通过一系列的接口对外界提供功能。也就是说，在UML2.0中，构件被认为是独立的，是呈现事物的更大的设计单元，这些事物一般将使用可更换的构件来实现。构件提供一个或多个接口。在书中，更倾向于把构件定义为：在软件系统中遵从并实现一组接口的物理的、可替换的软件模块。在这个构件定义中，强调了构件的两个重点—接口和可替换（或称为重用）。  构件的实施细节应被隐藏，它使用一系列的供接口（Provided Interface）提供它的功能，使用需接口（Required Interface）接收其他构件提供的功能。通过使用接口，可以避免系统中各个构件之间发生直接依赖关系，有利于新构件的替换。如果构件间的依赖关系与接口有关，那么构件可以被具有同样接口的其他构件替代。 |
| 软件系统架构图 | 从本质上来看，软件系统架构图是属于一种系统草图。在软件系统架构图所描述的对象就是直接的进行系统抽象组件构成。连接系统的各个组件之间就是做到把组件之间所存在的通讯比较明确与相对细致的实施描述。处于相应的系统实现环节，那么就会使得细化这些抽象组件成为现实的组件，比如可以是具体的某个类或者是对象。从面向对象领域进行分析，那么各个组件之前实施的连接实现往往是接口。 |

# 3 用例

### 3.1 用户信息模块

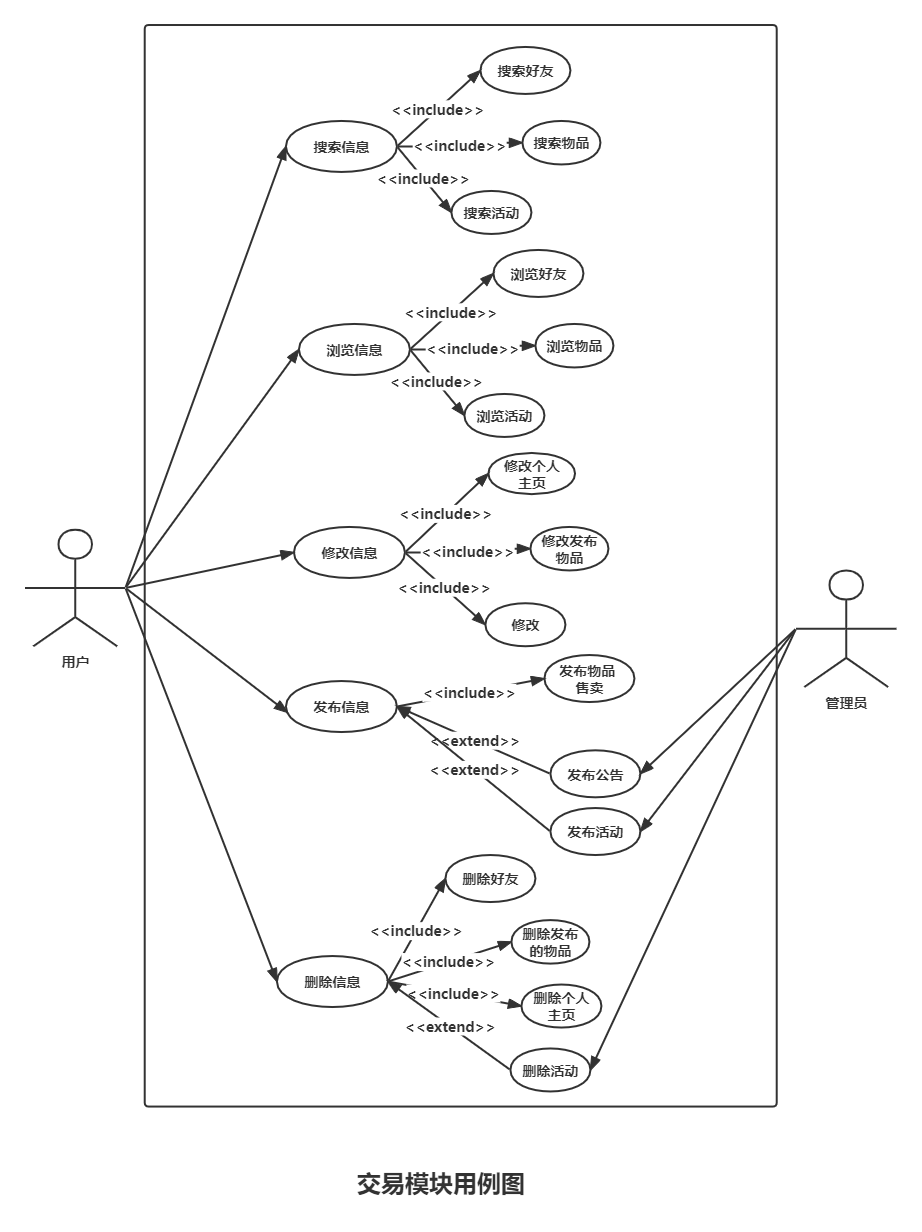


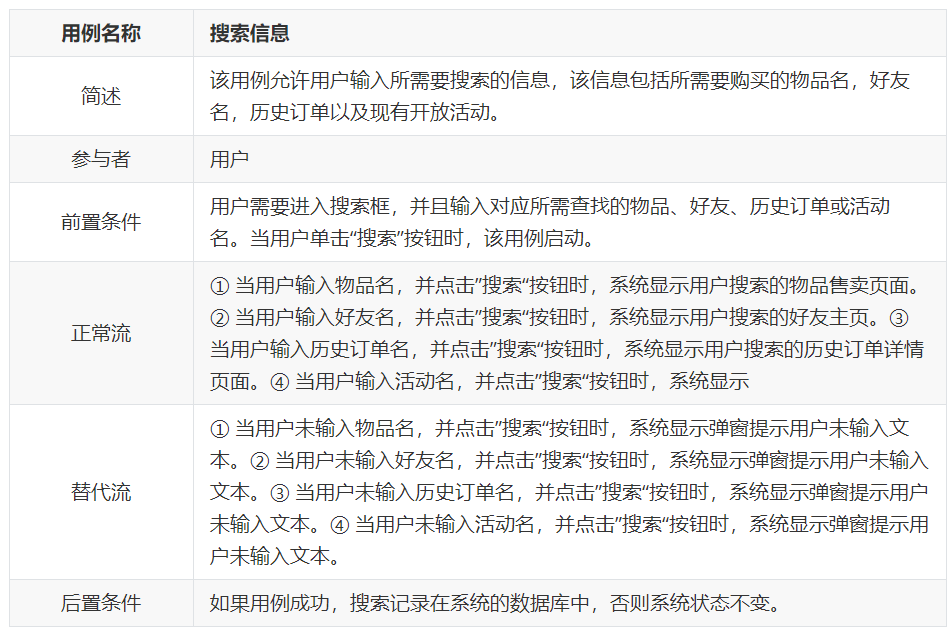




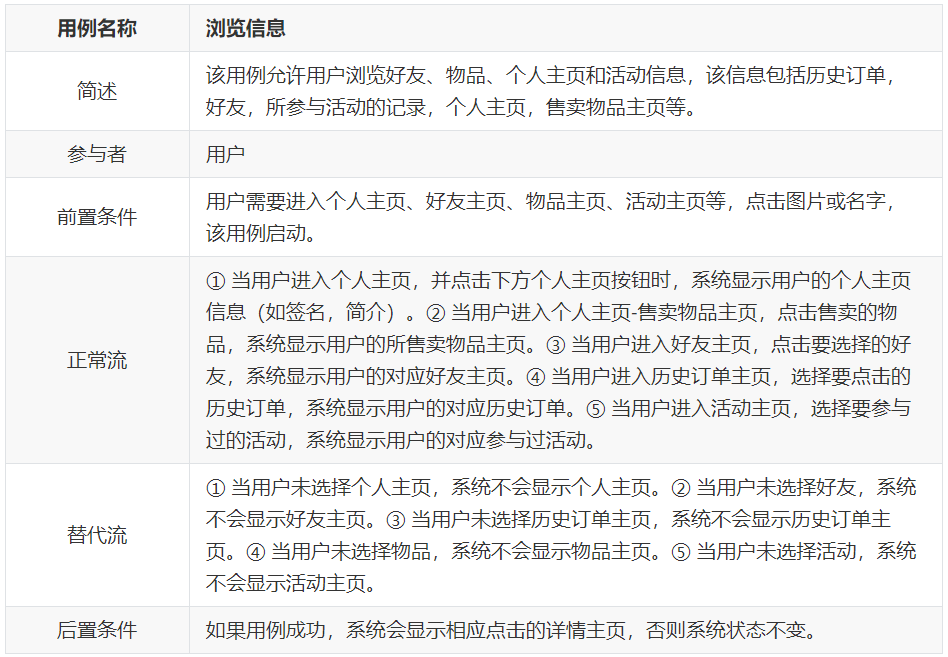


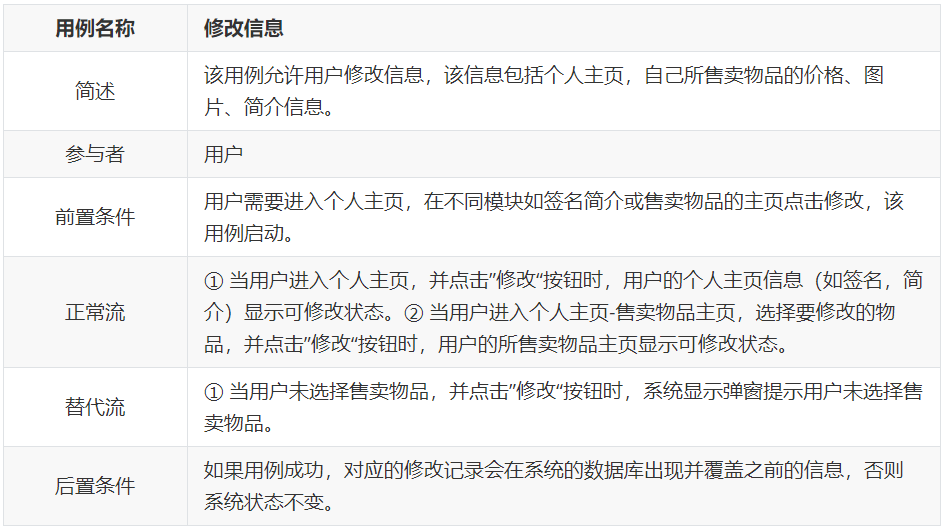
### 3.2 交易模块





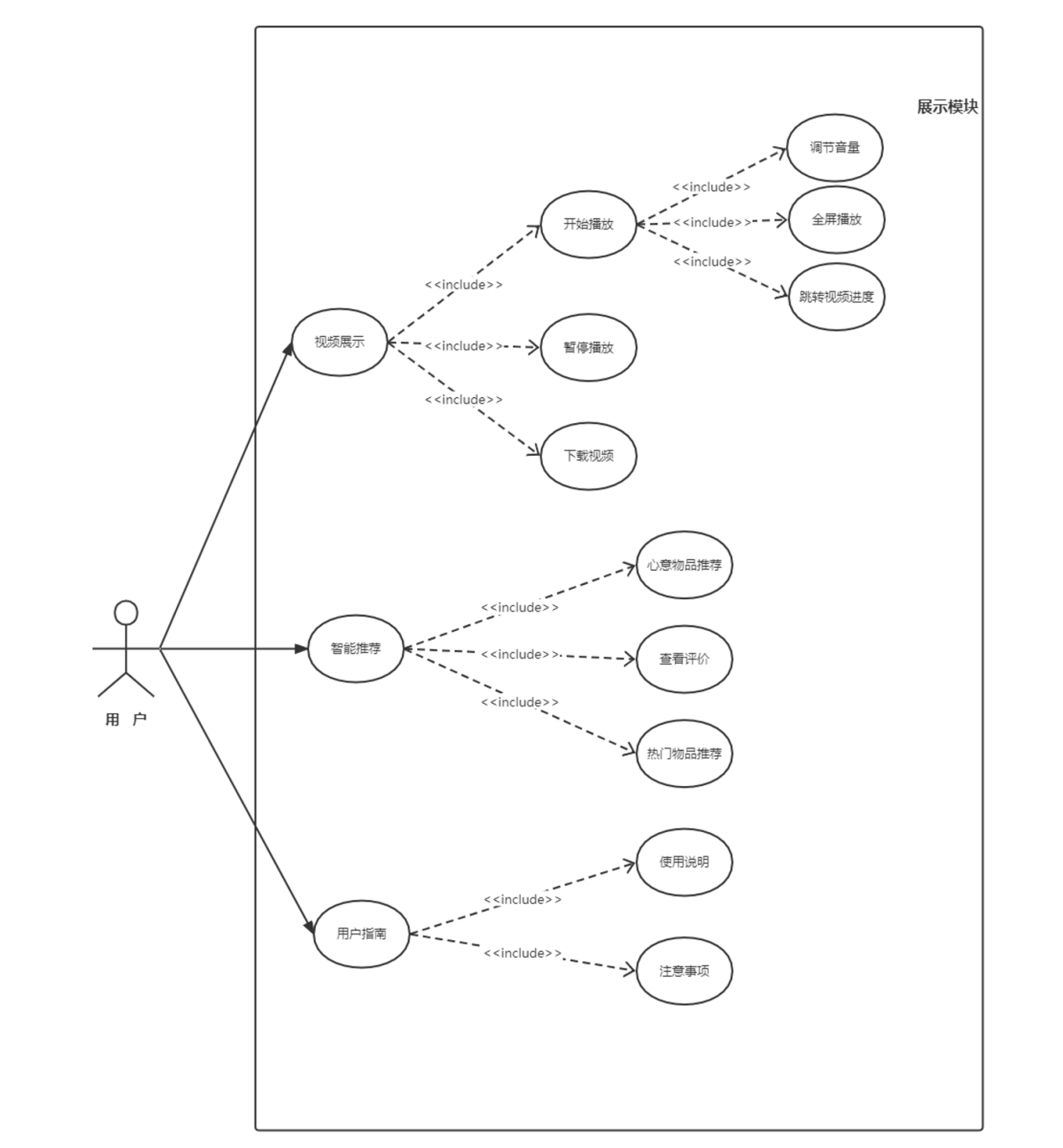




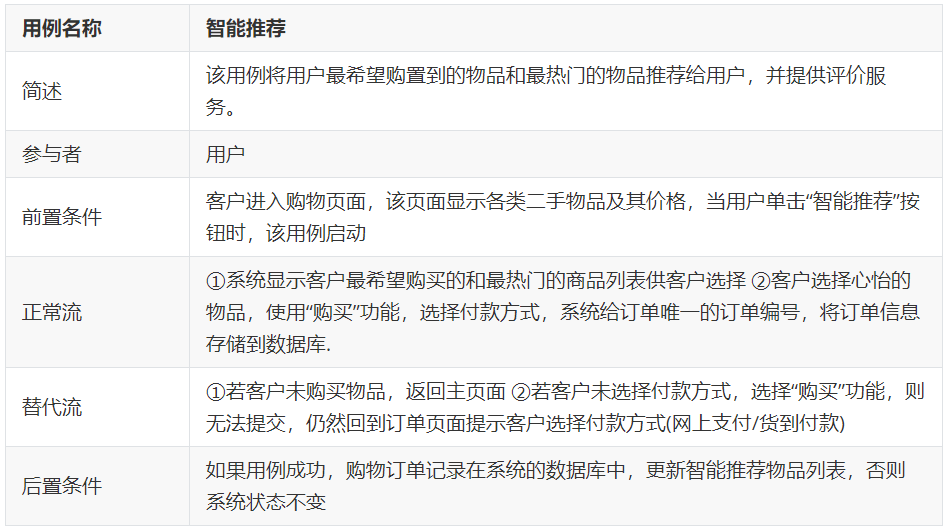




### 3.3 展示模块

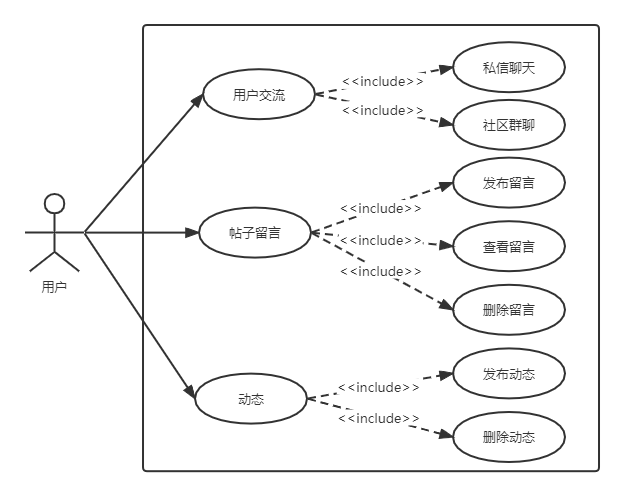


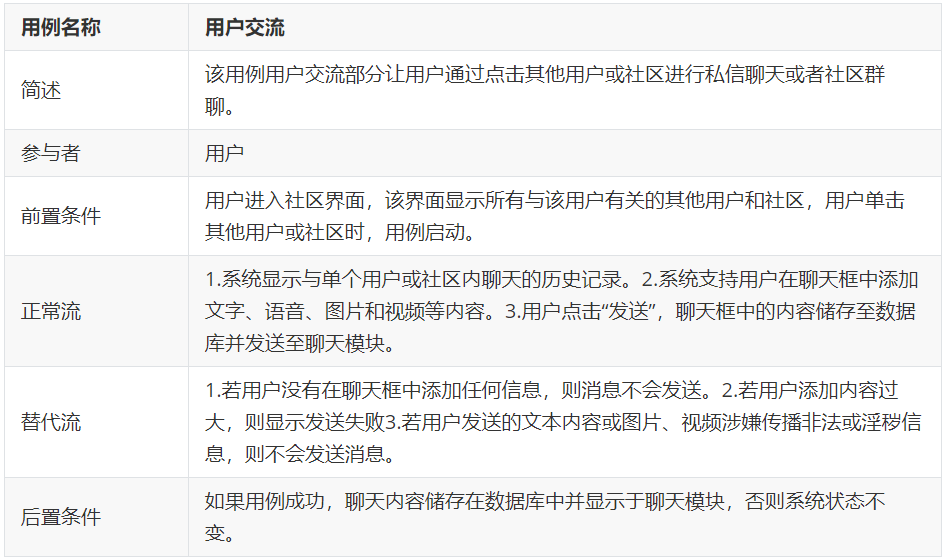




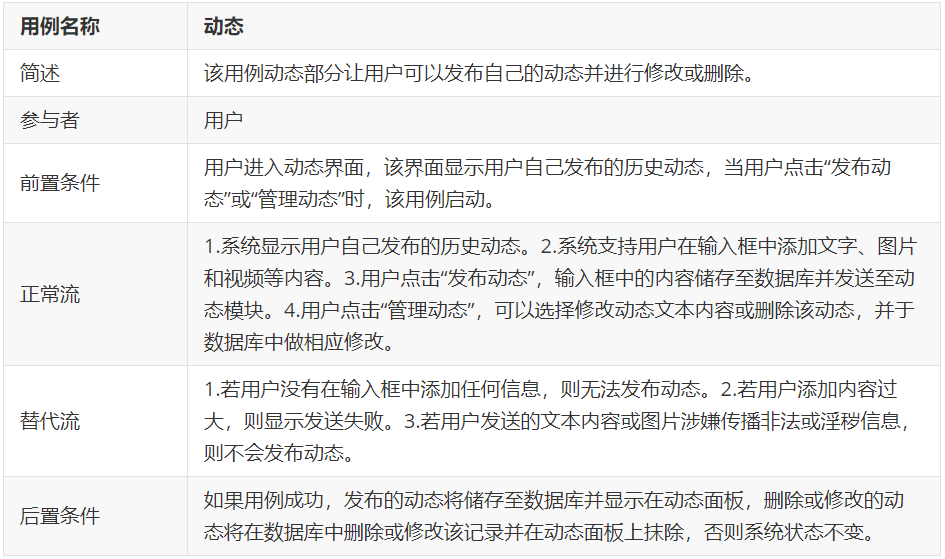


### 3.4 社区模块

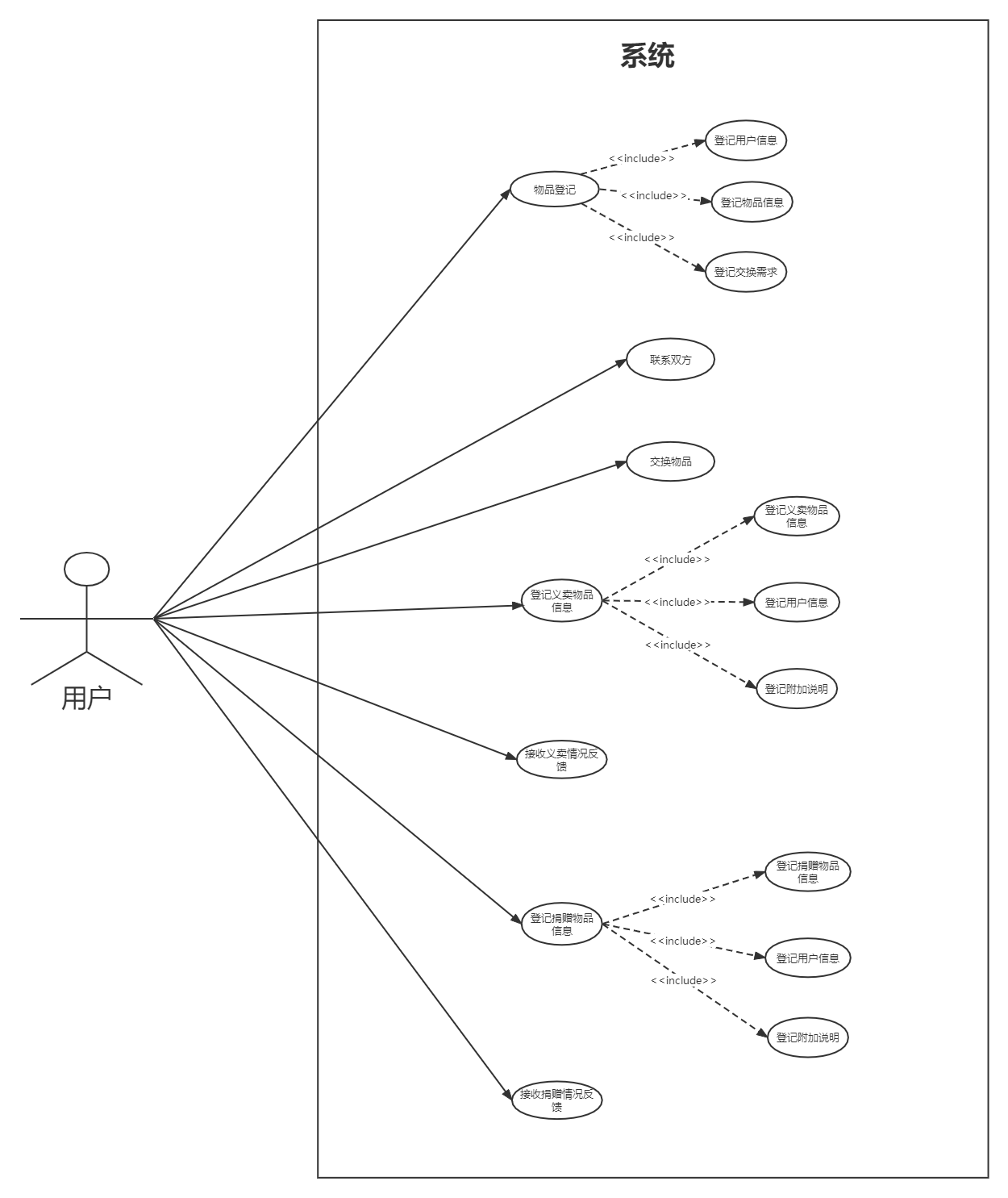






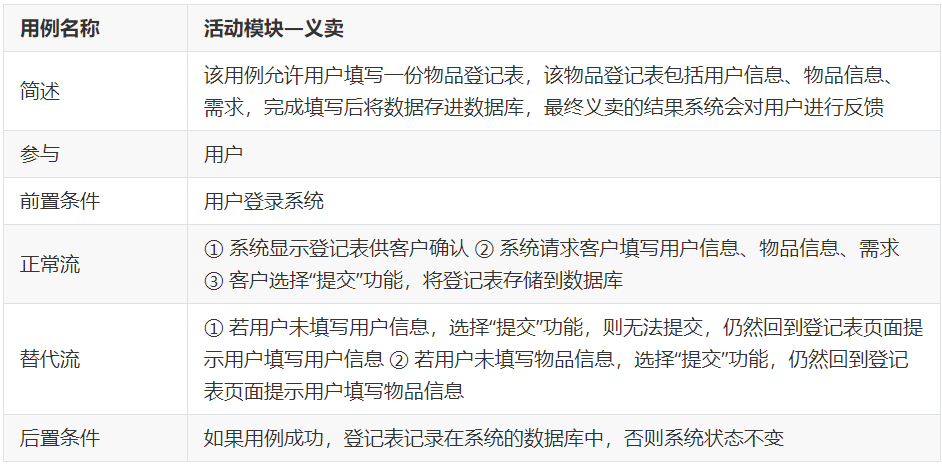


### 3.5 活动模块









　　此处要求**系统**用用例图表述（UML），对每个用例（正常处理的情况）要有中文叙述。【**注意用例关系：包含、扩展、泛化**】更加详细的系统用例。【在需求文档，uml、活动图【业务流程图】】【有可能，用例图粗，设计补充，细化】说明：在需求文档细化和说明，此处略

# 4 概要设计

## 4.1 简述

　　这部分要求突出整个设计所采用的方法（是面向对象设计还是结构化设计）、系统的体系结构（例如客户/服务器结构）以及使用到的相应技术和工具（例如OMT、Rose）

## 4.2 系统结构设计

　　这部分要求提供高层系统结构的描述，使用方框图来显示主要的组件及组件间的交互。**把逻辑结构同物理结构分离，分别进行描述**。别忘了说明图中用到的俗语和符号。

### 4.2.1 系统功能架构



### 4.2.2 系统软件架构

**概念**

表现层（JSP）采用MVC模式。M称为模型，也就是实体类。用于数据的封装和数据的传输。V为视图，也就是GUI组件，用于数据的展示。C为控制，也就是事件，用于流程的控制。

业务层（逻辑层、service层）采用事务脚本模式。将一个业务中所有的操作封装成一个方法，同时保证方法中所有的数据库更新操作，即保证同时成功或同时失败。避免部分成功部分失败引起的数据混乱操作。

持久层（DAO）采用DAO模式，建立实体类和数据库表映射（ORM映射）。也就是哪个类对应哪个表，哪个属性对应哪个列。持久层的目的就是完成对象数据和关系数据的转换。

**描述**

软件系统架构图中体现出SSM框架的三层架构，其中表示层对应软件系统架构图中的展示层，SSM框架的业务层对应软件系统架构图中的业务层和基础服务层，DAO层即对应架构图中的DAO层。

在表示层中，模板引擎可以实现业务逻辑代码和用户界面代码分离，还可以可以实现数据分离（动态数据与静态数据）等功能，降低代码的耦合度。网页通过Ajax交互来展示内容，浏览器通过 js 异步发起请求，局部更新页面，浏览器地址栏不会发生变化，局部更新不会舍弃原来页面的内容。

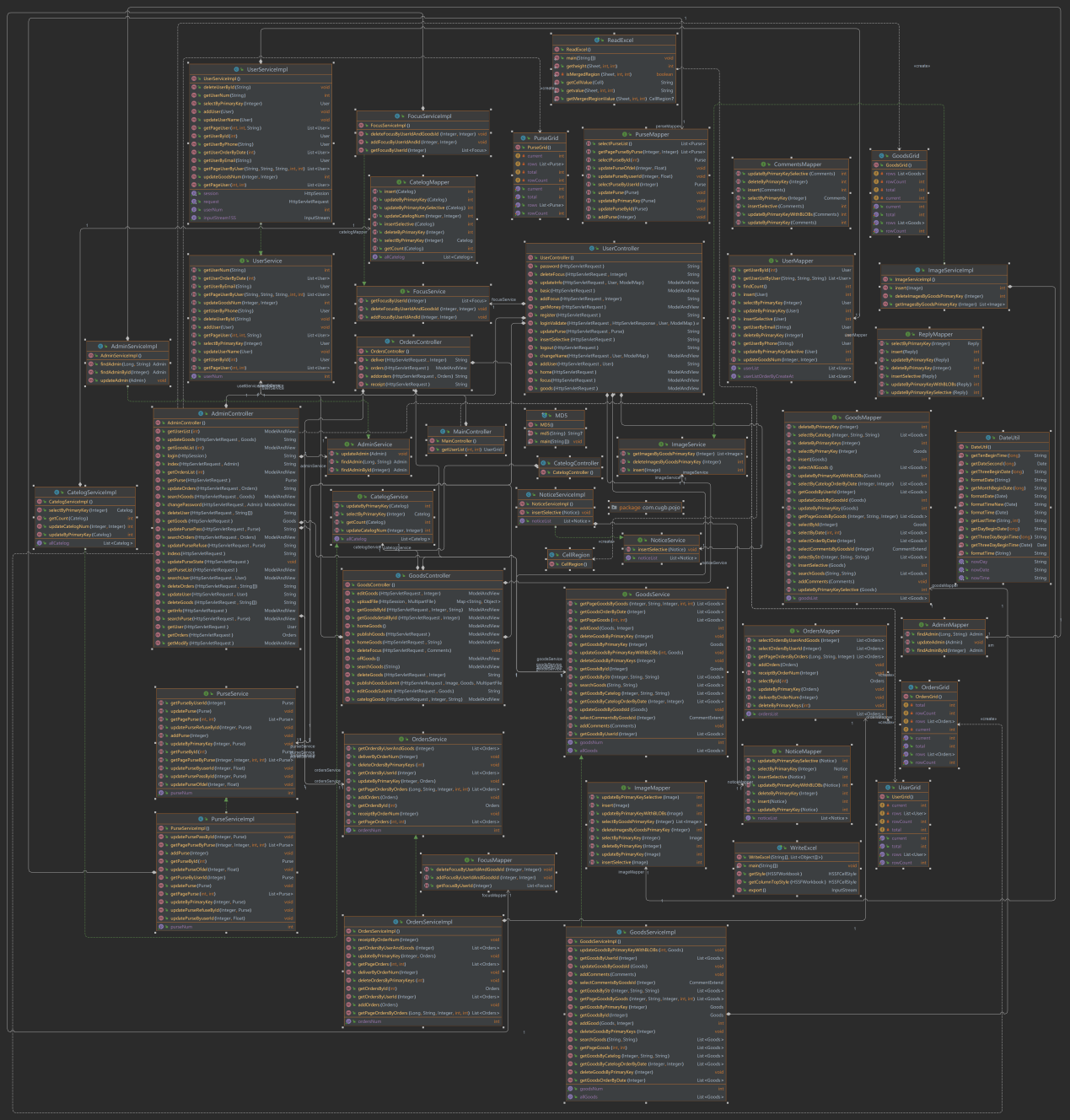
在业务层中，我们主要通过六个模块来处理核心的业务逻辑，用户信息模块主要负责管理用户的账号和用户的个人消息，包括订单信息，个人信息等，交易模块主要负责买卖双方的交易，活动模块主要负责义卖，以物易物等平台推出的活动，展示模块主要负责智能推荐和用户指南，社区模块负责用户发布动态，查看动态以及私信等功能，管理员模块负责管理系统后台所有用户的信息，管理用户包括买家卖家的账户信息。在架构图的业务层中，这些业务都通过调用基础服务层的基础服务来实现具体功能，比如用户信息模块用到了基础服务层中的身份认证和短信服务，管理员模块用到了权限管理。

进入DAO层，DAO层主要是负责数据库的访问和对数据库的操作，业务层的功能需要访问数据库的数据或者对数据库进行操作，就必须要通过DAO层中的自定义函数来实现来实现，比如通过SELECT语句来查询数据库，UPDATE语句来修改数据库等。

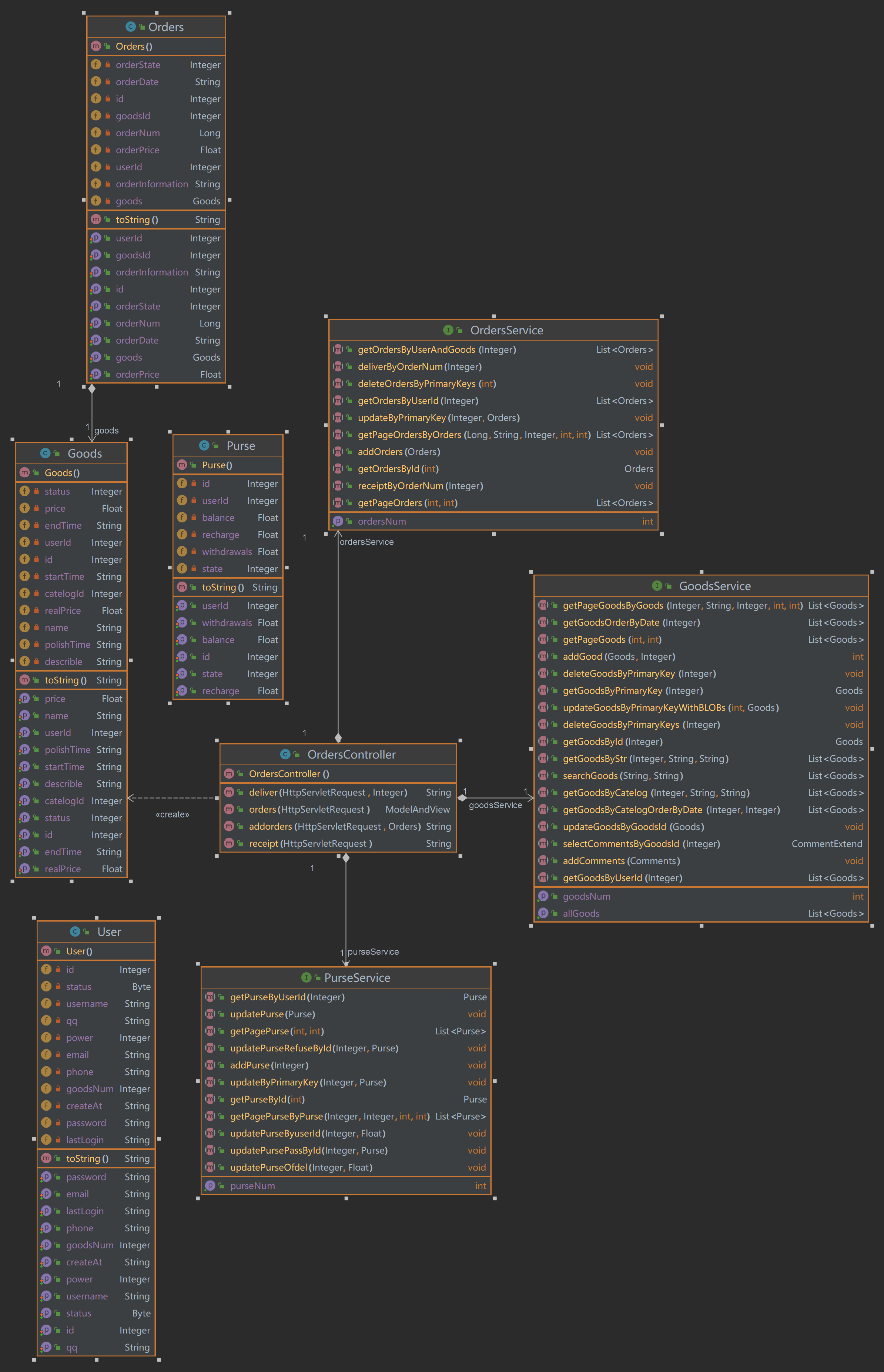
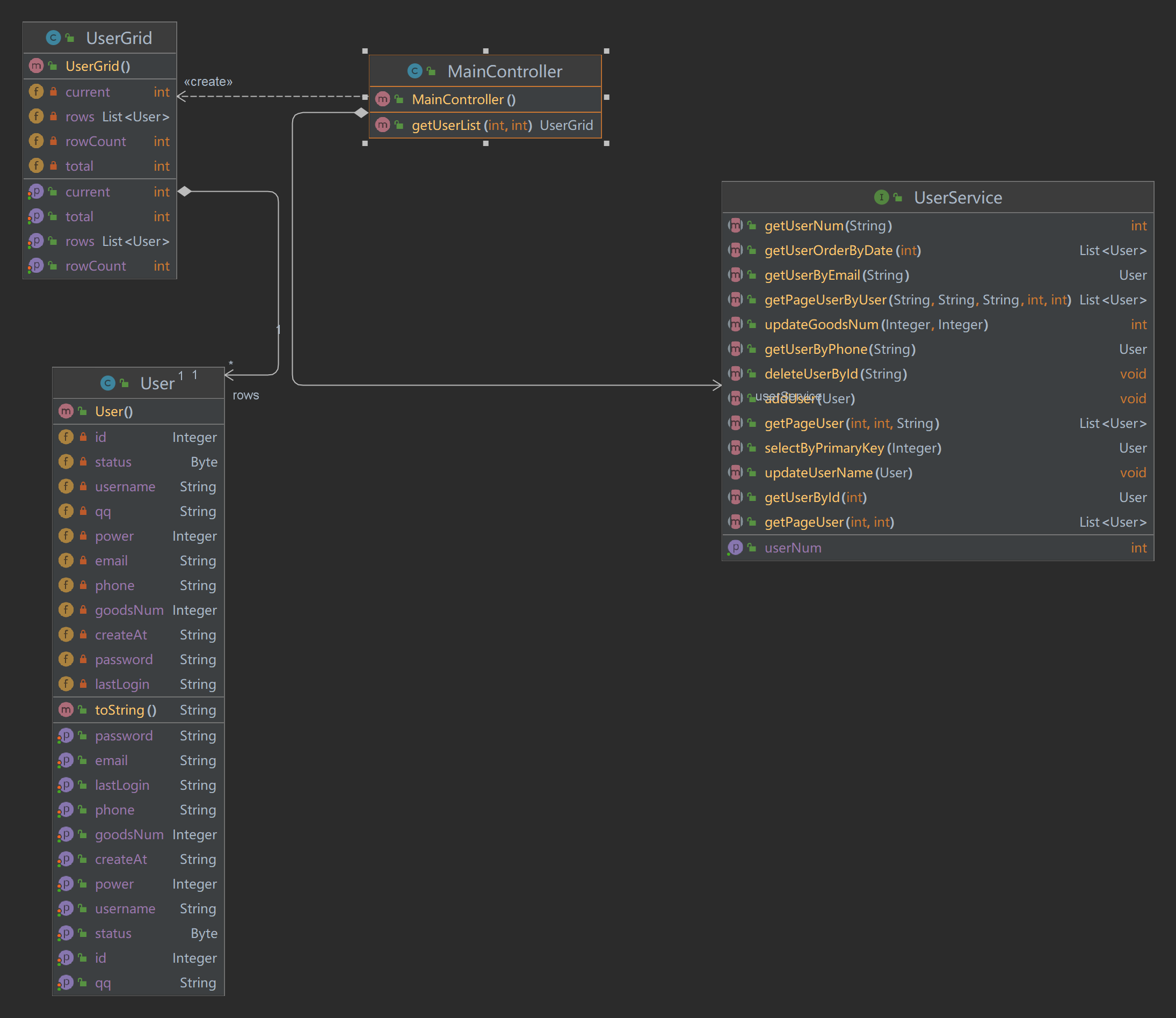
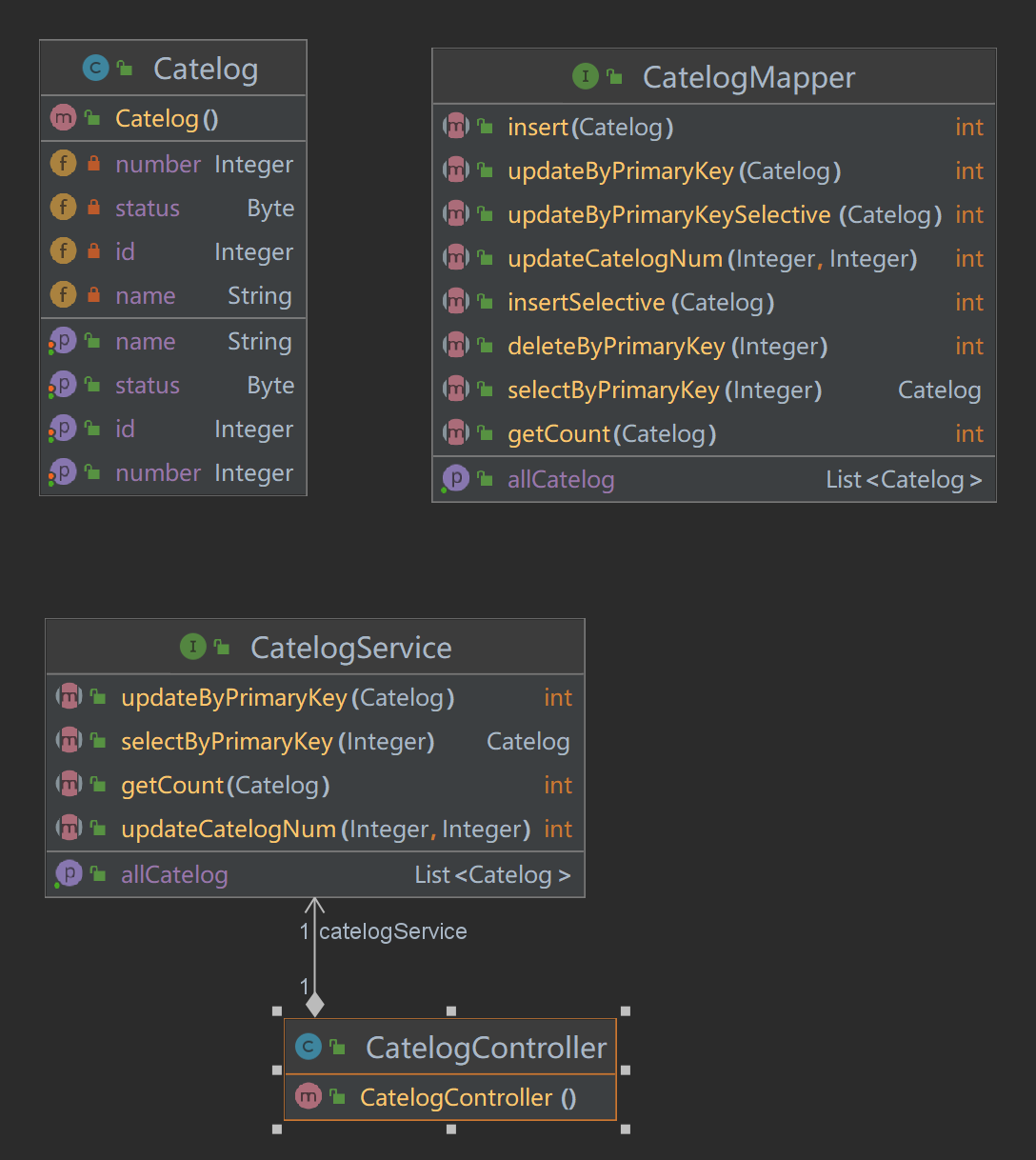
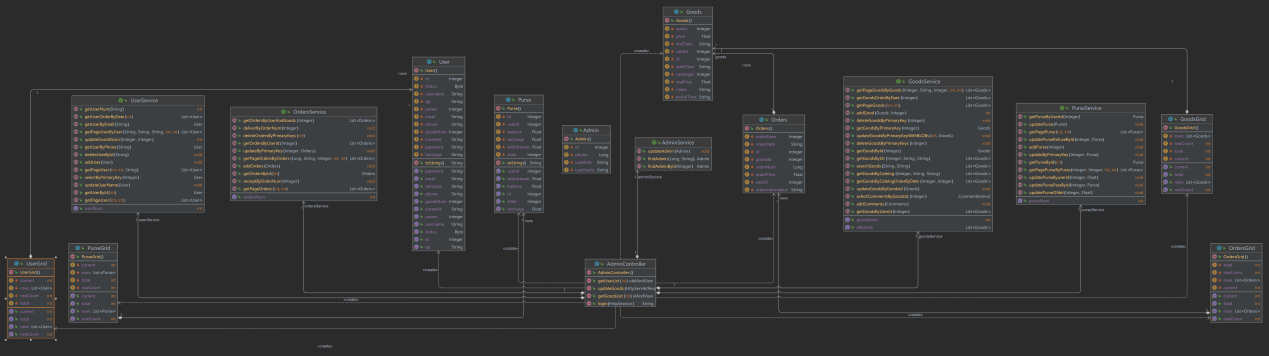
用户在展示层网页点击访问历史订单按钮，展示层通过Ajax传达指令给业务层，这部分是业务层的用户信息模块来负责，而用户信息模块通过基础服务层来调用DAO层的自定义函数，在自定义函数中，DAO层通过 SELECT 语句来访问数据库，从而得到该用户的历史订单信息，再传达到用户面前，这样各个层次各司其职，构成了一个完整的软件系统架构。

### 4.2.3 系统类对象图

**整体类图**



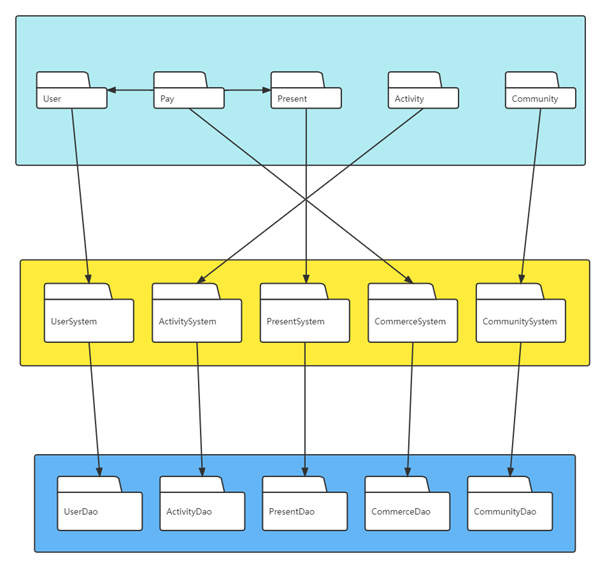
**局部类图**



### 4.2.4 系统包图

**概念**

包图（package Diagram）由若干个包以及包之间的关系组成。包是一种分组机制，将同类的类、对象、模型元素放在一起,形成高内聚，低耦合的类集合，可以说，一个包相当于一个子系统。



**描述**

User包中存放用户模块相关页面，包括用户登陆注册、用户信息等；

Pay包中存放交易模块相关页面，包括购物车、支付等；

Present包中存放展示模块相关页面，包括展示页面、视频介绍页面等；

Activity包中存放活动模块相关页面，包括各种活动有关页面；

Community包中存放社区模块相关页面；

五个System包里存放控制类，接收用户的请求，负责页面的跳转，以及存放实体类，负责数据的存储和数据的传递。

五个Dao包里存放实现业务逻辑处理和数据访问的类。

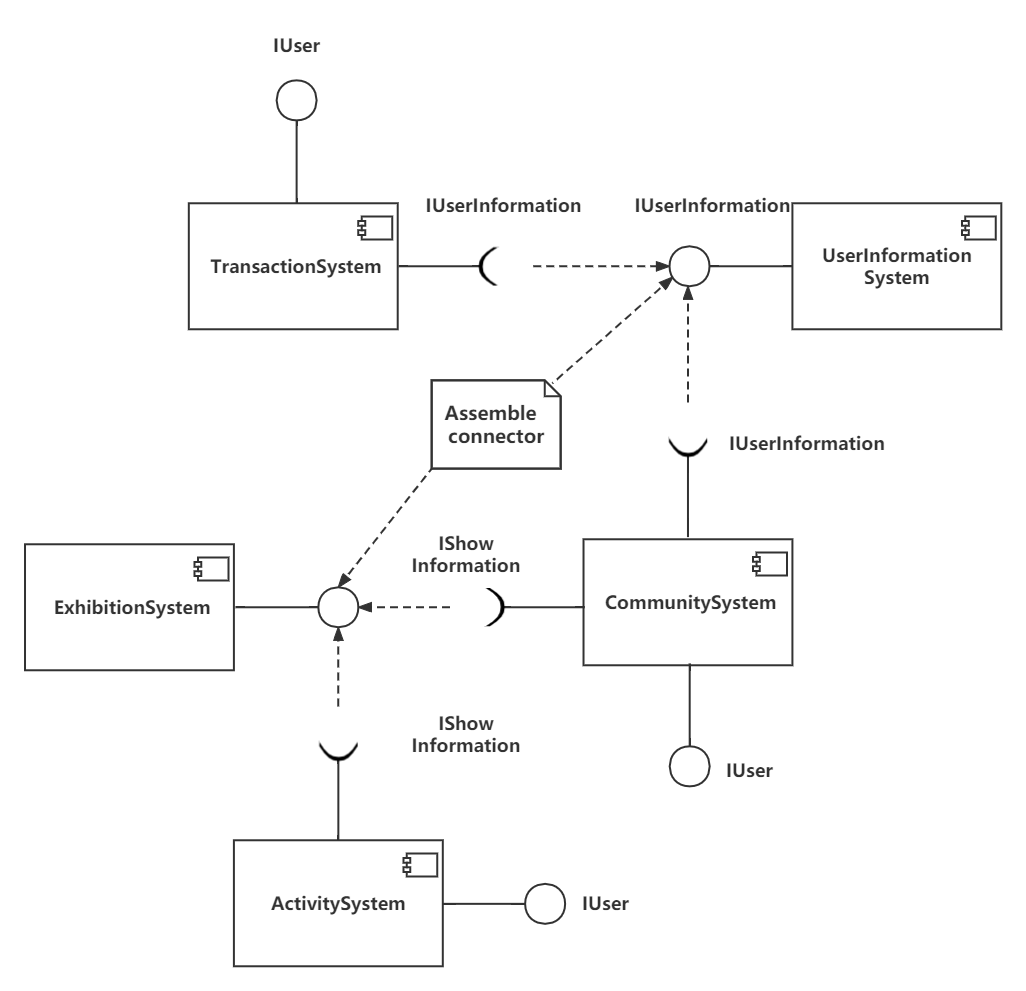
五个模块，即用户模块，交易模块，展示模块，活动模块，社区模块分别对应一个包，其中交易模块包依赖于用户模块包和展示模块包，五个模块包分别依赖于对应模块的system包，五个功能模块的system包分别依赖于对应的dao包 。

### 4.2.5 系统构件（模块，组件）图

图中一共用五个构件，分别是**UserInformationSystem**， **TransactionSystem** ， **CommunitySystem** ， **ExhibitionSystem** ， **ActivitySystem** 。

其中**UserInformationSystem**构件伸出两个供接口，分别提供给用户，**TransactionSystem** 和 **CommunitySystem** ，同时该两个构件伸出需接口，从**UserInformationSystem**接收功能服务。

同理 **ExhibitionSystem** 提供IShowInformation给 **CommunitySystem** ， **ActivitySystem** ，而每个构件的都有IUser接口为用户提供服务



## 4.3 约束和假定

1. **软件环境约束**
2. 开发软件：IDEA
3. 配置环境：MAVEN，JAVA11
4. 应用服务器：TOMCAT
5. 数据库：MYSQL
6. **硬件环境约束**
7. windows系统
8. 硬盘空间：16G以上
9. 处理器：64位处理器
10. **接口/协议的约束**
11. 采用HTTP和UDP协议
12. **软件质量的约束**
13. 性能：需要满足3000人以上同时登录二手交易网站，并且满足多人同时在线操作，要求延迟不能超过50ms。
14. 可用性：网页各个按钮，输入框等操作需要正常运行，当出现用户错误性操作时（如输入非法数字），系统需要及时的提示用户
15. 可靠性：当网页出现错误的时候，需要维持最基本的网页登录以及数据的交互与存储。
16. 健壮性：系统要有一定的承载压力的能力。
17. 安全性：系统向合法用户提供服务的同时能够阻止非授权用户使用的企图或拒绝服务的能力
18. 可修改性：系统能够快速地以较高的性能价格比对系统进行变更的能力
19. 易用性：衡量用户使用一个软件产品完成指定任务的难易程度，系统UI不能设计的过于难用，要让用户简明地抓住操作。

（9）可测试性：当网页发现故障，应具有隔离、定位其故障的能力特性，以及在一定的时间和成本前提下，进行测试设计、测试执行的能力。

（10）功能性：系统所能完成所期望工作的能力。

（11）互操作性：指系统与外界或系统与系统之间的相互作用能力。

1. **额外约束**

（1）在线交易平台必须在1s内完成客户的交易请求。该要求主要对应性能，可以采用的架构设计策略有增加计算资源、改善资源需求（减少计算复杂度等）、资源管理（并发、数据复制等）和资源调度（先进先出队列、优先级队列等）。

（2）该平台必须严格保证客户个人信息和交易信息的保密性和安全性。该要求主要对应安全性，可以采用的架构设计策略有抵御攻击（授权、认证和限制访问等）、攻击检测（入侵检测等）、从攻击中恢复（部分可用性策略）和信息审计等。

（3）当发生故障时，该平台的平均故障恢复时间必须小于l0s。该要求主要对应可用性，可以采用的架构设计策略有Ping/Echo、心跳、异常和主动冗余等。

（4）由于企业业务发展较快，需要经常为该平台添加新功能或进行硬件升级。添加新功能或进行平台升级必须在6小时内完成。该要求主要对应可修改性，可以采用的架构设计策略有软件模块泛化、限制模块之间通信、使用中介和延迟绑定等。

# 5 对象（类）描述

## 5.1 子系统1【构件1】中的对象

### **5.1.1 对象：GoodsCarBean**

　　用途：购物车序列化到前端

　　约束：

　　持久性：

#### **5.1.1.1 属性描述：**

　　1. 属性：id

　　类型：int

　　描述：商品id

　　约束：

　　2. 属性：name

类型：String

　　描述：商品名称

　　约束：

3. 属性：price

类型：double

　　描述：商品价格

约束：

4. 属性：sort

类型：String

　　描述：商品分类

约束：

5. 属性：quantity

类型：int

　　描述：商品数量

约束：

#### **5.1.1.2 方法描述：**

　　1. 方法：getId()

　　返回类型：int

　　参数：

　　返回值：id

　　Pre-Condition：

　　Post-Condition：

　　读取/修改的属性：

　　调用的方法：

2. 方法：getName()

　　返回类型：String

　　参数：

　　返回值：name

　　Pre-Condition：

　　Post-Condition：

　　读取/修改的属性：

　　调用的方法：

3. 方法：getPrice()

　　返回类型：double

　　参数：

　　返回值：price

　　Pre-Condition：

　　Post-Condition：

　　读取/修改的属性：

　　调用的方法：

4. 方法：getSort()

　　返回类型：String

　　参数：

　　返回值：sort

　　Pre-Condition：

　　Post-Condition：

　　读取/修改的属性：

　　调用的方法：

5. 方法：getQuantity()

　　返回类型：int

　　参数：

　　返回值：quantity

　　Pre-Condition：

　　Post-Condition：

　　读取/修改的属性：

　　调用的方法：

6. 方法：setId(int id)

　　返回类型：void

　　参数：id

　　返回值：

　　Pre-Condition：input id

　　Post-Condition：

　　读取/修改的属性：id

　　调用的方法：

7. 方法：setName(String name)

　　返回类型：void

　　参数：name

　　返回值：

　　Pre-Condition：input name

　　Post-Condition：

　　读取/修改的属性：name

　　调用的方法：

8. 方法：setPrice(double price)

　　返回类型：void

　　参数：price

　　返回值：

　　Pre-Condition：input price

　　Post-Condition：

　　读取/修改的属性：price

　　调用的方法：

9. 方法：setSort(String sort)

　　返回类型：void

　　参数：sort

　　返回值：

　　Pre-Condition：input sort

　　Post-Condition：

　　读取/修改的属性：sort

　　调用的方法：

10. 方法：setQuantity(int quantity)

　　返回类型：void

　　参数：quantity

　　返回值：

　　Pre-Condition：input quantity

　　Post-Condition：

　　读取/修改的属性：quantity

　　调用的方法：

### **5.1.2 对象：ShopContextBean**

　　用途：发布的物品信息序列化到前端

　　约束：

　　持久性：

#### **5.1.2.1 属性描述：**

　　1. 属性：id

　　类型：int

　　描述：商品id

　　约束：

　　2. 属性：context

类型：String

　　描述：商品描述

　　约束：

3. 属性：username

类型：String

　　描述：商品发布人

　　约束：

#### **5.1.2.2 方法描述：**

　　1. 方法：getId()

　　返回类型：int

　　参数：

　　返回值：id

　　Pre-Condition：

　　Post-Condition：

　　读取/修改的属性：

　　调用的方法：

2. 方法：getContext()

　　返回类型：String

　　参数：

　　返回值：context

　　Pre-Condition：

　　Post-Condition：

　　读取/修改的属性：

调用的方法：

3. 方法：getUsername()

　　返回类型：String

　　参数：

　　返回值：username

　　Pre-Condition：

　　Post-Condition：

　　读取/修改的属性：

调用的方法：

4. 方法：setId(int id)

　　返回类型：void

　　参数：id

　　返回值：

　　Pre-Condition：input id

　　Post-Condition：

　　读取/修改的属性：id

调用的方法：

5. 方法：setContext(String context)

　　返回类型：void

　　参数：context

　　返回值：

　　Pre-Condition：input context

　　Post-Condition：

　　读取/修改的属性：context

调用的方法：

6. 方法：setUsername(String username)

　　返回类型：void

　　参数：username

　　返回值：

　　Pre-Condition：input username

　　Post-Condition：

　　读取/修改的属性：username

调用的方法：

### **5.1.3 对象：ShopInformationBean**

　　用途：商品信息序列化到前端

　　约束：

　　持久性：

#### **5.1.3.1 属性描述：**

　　1. 属性：id

　　类型：int

　　描述：商品id

　　约束：

　　2. 属性：name

类型：String

　　描述：商品名称

　　约束：

3. 属性：price

类型：double

　　描述：商品价格

约束：

4. 属性：sort

类型：String

　　描述：商品分类

约束：

5. 属性：quantity

类型：int

　　描述：商品数量

约束：

#### **5.1.3.2 方法描述：**

　　1. 方法：getId()

　　返回类型：int

　　参数：

　　返回值：id

　　Pre-Condition：

　　Post-Condition：

　　读取/修改的属性：

　　调用的方法：

2. 方法：getName()

　　返回类型：String

　　参数：

　　返回值：name

　　Pre-Condition：

　　Post-Condition：

　　读取/修改的属性：

　　调用的方法：

3. 方法：getPrice()

　　返回类型：double

　　参数：

　　返回值：price

　　Pre-Condition：

　　Post-Condition：

　　读取/修改的属性：

　　调用的方法：

4. 方法：getSort()

　　返回类型：String

　　参数：

　　返回值：sort

　　Pre-Condition：

　　Post-Condition：

　　读取/修改的属性：

　　调用的方法：

5. 方法：getQuantity()

　　返回类型：int

　　参数：

　　返回值：quantity

　　Pre-Condition：

　　Post-Condition：

　　读取/修改的属性：

　　调用的方法：

6. 方法：setId(int id)

　　返回类型：void

　　参数：id

　　返回值：

　　Pre-Condition：input id

　　Post-Condition：

　　读取/修改的属性：id

　　调用的方法：

7. 方法：setName(String name)

　　返回类型：void

　　参数：name

　　返回值：

　　Pre-Condition：input name

　　Post-Condition：

　　读取/修改的属性：name

　　调用的方法：

8. 方法：setPrice(double price)

　　返回类型：void

　　参数：price

　　返回值：

　　Pre-Condition：input price

　　Post-Condition：

　　读取/修改的属性：price

　　调用的方法：

9. 方法：setSort(String sort)

　　返回类型：void

　　参数：sort

　　返回值：

　　Pre-Condition：input sort

　　Post-Condition：

　　读取/修改的属性：sort

　　调用的方法：

10. 方法：setQuantity(int quantity)

　　返回类型：void

　　参数：quantity

　　返回值：

　　Pre-Condition：input quantity

　　Post-Condition：

　　读取/修改的属性：quantity

　　调用的方法：

### **5.1.4 对象：UserWantBean**

　　用途：用户求购信息序列化到前端

　　约束：

　　持久性：

#### 5.1.4.1 属性描述：

　　1. 属性：id

　　类型：int

　　描述：商品id

　　约束：

　　2. 属性：name

类型：String

　　描述：商品名称

　　约束：

3. 属性：price

类型：double

　　描述：商品价格

约束：

4. 属性：sort

类型：String

　　描述：商品分类

约束：

5. 属性：quantity

类型：int

　　描述：商品数量

约束：

#### **5.1.4.2 方法描述：**

　　1. 方法：getId()

　　返回类型：int

　　参数：

　　返回值：id

　　Pre-Condition：

　　Post-Condition：

　　读取/修改的属性：

　　调用的方法：

2. 方法：getName()

　　返回类型：String

　　参数：

　　返回值：name

　　Pre-Condition：

　　Post-Condition：

　　读取/修改的属性：

　　调用的方法：

3. 方法：getPrice()

　　返回类型：double

　　参数：

　　返回值：price

　　Pre-Condition：

　　Post-Condition：

　　读取/修改的属性：

　　调用的方法：

4. 方法：getSort()

　　返回类型：String

　　参数：

　　返回值：sort

　　Pre-Condition：

　　Post-Condition：

　　读取/修改的属性：

　　调用的方法：

5. 方法：getQuantity()

　　返回类型：int

　　参数：

　　返回值：quantity

　　Pre-Condition：

　　Post-Condition：

　　读取/修改的属性：

　　调用的方法：

6. 方法：setId(int id)

　　返回类型：void

　　参数：id

　　返回值：

　　Pre-Condition：input id

　　Post-Condition：

　　读取/修改的属性：id

　　调用的方法：

7. 方法：setName(String name)

　　返回类型：void

　　参数：name

　　返回值：

　　Pre-Condition：input name

　　Post-Condition：

　　读取/修改的属性：name

　　调用的方法：

8. 方法：setPrice(double price)

　　返回类型：void

　　参数：price

　　返回值：

　　Pre-Condition：input price

　　Post-Condition：

　　读取/修改的属性：price

　　调用的方法：

9. 方法：setSort(String sort)

　　返回类型：void

　　参数：sort

　　返回值：

　　Pre-Condition：input sort

　　Post-Condition：

　　读取/修改的属性：sort

　　调用的方法：

10. 方法：setQuantity(int quantity)

　　返回类型：void

　　参数：quantity

　　返回值：

　　Pre-Condition：input quantity

　　Post-Condition：

　　读取/修改的属性：quantity

　　调用的方法：

### **5.1.5 对象：ForgetController**

　　用途：对用户忘记密码或者用户名进行重置操作

　　约束：用户有忘记密码的操作时

　　持久性：用户进行忘记密码或者用户名操作时

#### **5.1.5.1 属性描述：**

1.属性：map

　　 类型：Map<String, Integer>

　　 描述：验证结果（结果字段 , 结果代码）

　　 约束：插入得满足<String, Integer>格式

#### **5.1.5.2 方法描述：**

1.方法：checkPhone

　　返回类型：Map

　　参数：HttpServletRequest request , Model model , String code , String token

　　Pre-Condition：无

　　Post-Condition：无

　　读取/修改的属性：request ， code

　　调用的方法：相关的数据库读写方法

　　处理逻辑：使用request ， code参数调用该方法，根据request ， code 的参数判断对

于手机号和验证码是否验证正确，返回一个验证结果的map

2.方法：updatePassword

　　返回类型：BaseResponse

　　参数：HttpServletRequest request , Model model , String password, String token

　　Pre-Condition：无

　　Post-Condition：无

　　读取/修改的属性：request ， password , token

　　调用的方法：相关的数据库读写方法

　　处理逻辑：使用request ， password , token参数调用该方法，根据request ， password , token的参数进行密码重置， 返回密码的重置是否成功

### **5.1.6 对象：GoodsController**

　　用途：用户对商品的查看和查询操作

　　约束：用户已登录

　　持久性：整个用户使用期间

#### **5.1.6.1 属性描述：**

1.属性：userInformation

　　 类型：UserInformation

　　 描述：用户信息

　　 约束：无

2.属性：shopInformations

　　 类型：List<ShopInformation>

　　 描述：商品信息列表

　　 约束：无

3.属性：shopInformationBean

　　 类型：ShopInformationBean

　　 描述：商品页面信息

　　 约束：无

4.属性：shopContexts

　　 类型：List<ShopContext>

　　 描述：商品目录列表

　　 约束：无

5.属性：shopContextBeans

　　 类型：List<ShopContextBean>

　　 描述：商品目录展示页面列表

　　 约束：无

#### **5.1.6.2 方法描述：**

1.方法：publish

　　返回类型：String

　　参数：HttpServletRequest request, Model model

　　Pre-Condition：无

　　Post-Condition：无

　　读取/修改的属性：request ， model

　　调用的方法：相关的数据库读写方法

　　处理逻辑：使用request ， code参数调用该方法，判断用户是否登录了，在用户登录的情况 下就可以进入发 布商品页面，返回一个进入成功或者需要验证的字符串

2.方法：findByName

　　返回类型：String

　　参数：HttpServletRequest request, Model model ， String name

　　Pre-Condition：无

　　Post-Condition：无

　　读取/修改的属性：request ， model ， name

　　调用的方法：相关的数据库读写方法

　　处理逻辑：使用request ， model ， name参数调用该方法，根据商品的名字进行模糊查 询，遍历商品后选出符合的商品进行展示，返回一个查询成功或者失败的字符串

3.方法：selectById

　　返回类型：String

　　参数：HttpServletRequest request, Model model ， int id

　　Pre-Condition：无

　　Post-Condition：无

　　读取/修改的属性：request ， model ， id

　　调用的方法：相关的数据库读写方法，商品的信息搜寻等方法

　　处理逻辑：使用request ， model ， id参数调用该方法，根据商品的id进行查询，遍历商品 后选出符合的商品进行商品详情展示，返回一个查询成功或者失败的字符串

4.方法：selectBySort

　　返回类型：List<ShopInformation>

　　参数：int sort

　　Pre-Condition：无

　　Post-Condition：无

　　读取/修改的属性：sort

　　调用的方法：无

　　处理逻辑：使用sort参数调用该方法，根据商品的类别进行查询，遍历商品后选出符合的商 品进行商品详情展示，返回一个查询成功或者失败的字符串

### **5.1.7 对象：RegisterController**

　　用途：用户的注册操作

　　约束：无

　　持久性：用户注册期间

#### **5.1.7.1 属性描述：**

1.属性：realPhone

　　 类型：String

　　 描述：用户手机号

　　 约束：满足手机号的格式（11位）

2.属性：result

　　 类型：int

　　 描述：注册结果

　　 约束：1 为成功

3.属性：userPassword

　　 类型：UserPassword

　　 描述：用户密码

　　 约束：满足密码格式

#### **5.1.6.2 方法描述：**

1.方法：insertUser

　　返回类型：BaseResponse

　　参数：HttpServletRequest request, String token ， String password

　　Pre-Condition：无

　　Post-Condition：无

　　读取/修改的属性：request ， token ， password

　　调用的方法：相关的数据库读写方法

　　处理逻辑：使用request ， token ， password参数调用该方法，让用户填写信息进行注册， 返回是否注册成功的应答

### **5.1.7 对象：BaseResponse**

　　用途：对请求的基础应答

　　约束：有请求动作

　　持久性：整个网页的运作时期

#### **5.1.4.1 属性描述：**

1.属性：result

　　 类型：int

　　 描述：对于网页请求的应答代码

　　 约束：满足int型字符的条件

2. 属性：msg

　　 类型：String

　　 描述：对于网页请求的应答字符信息

　　 约束：满足String型字符的条件

#### **5.1.4.2 方法描述：**

1.方法：BaseResponse

　　返回类型：无

　　参数：int result, String msg

　　返回值：无

　　Pre-Condition：无

　　Post-Condition：无

　　读取/修改的属性：result， msg

　　调用的方法：无

　　处理逻辑：用result， msg参数调用该构造方法，构造一个对象

2.方法：BaseResponse

　　返回类型：无

　　参数：int result

　　返回值：无

　　Pre-Condition：无

　　Post-Condition：无

　　读取/修改的属性：result

　　调用的方法：无

　　处理逻辑：用result参数调用该构造方法，构造一个对象

3.方法：fail

　　返回类型：BaseResponse

　　参数：无

　　返回值：new BaseResponse

　　Pre-Condition：无

　　Post-Condition：无

　　读取/修改的属性：无

　　调用的方法：无

　　处理逻辑：不使用任何参数调用该方法，返回一个新的BaseResponse对象

4.方法：fail

　　返回类型：BaseResponse

　　参数：String msg

　　返回值：new BaseResponse

　　Pre-Condition：无

　　Post-Condition：无

　　读取/修改的属性：无

　　调用的方法：无

　　处理逻辑：使用msg参数调用该方法，返回一个新的BaseResponse对象

5.方法：fail

　　返回类型：BaseResponse

　　参数：int result

　　返回值：new BaseResponse

　　Pre-Condition：无

　　Post-Condition：无

　　读取/修改的属性：无

　　调用的方法：无

　　处理逻辑：使用result参数调用该方法，返回一个新的BaseResponse对象

6.方法：success

　　返回类型：BaseResponse

　　参数：无

　　返回值：new BaseResponse

　　Pre-Condition：无

　　Post-Condition：无

　　读取/修改的属性：无

　　调用的方法：无

　　处理逻辑：不使用任何参数调用该方法，返回一个新的BaseResponse对象

7.方法：success

　　返回类型：BaseResponse

　　参数：String msg

　　返回值：new BaseResponse

　　Pre-Condition：无

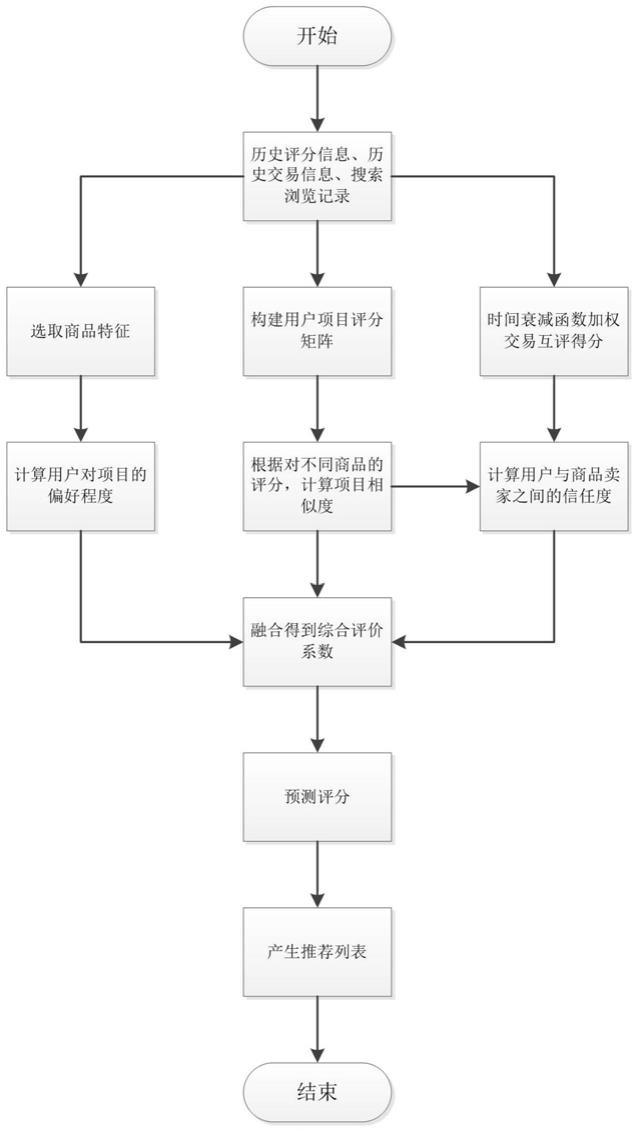
　　Post-Condition：无

　　读取/修改的属性：无

　　调用的方法：无

　　处理逻辑：使用msg参数调用该方法，返回一个新的BaseResponse对象

## 5.2 推荐系统模块【算法的原理流程描述、各种接口类】



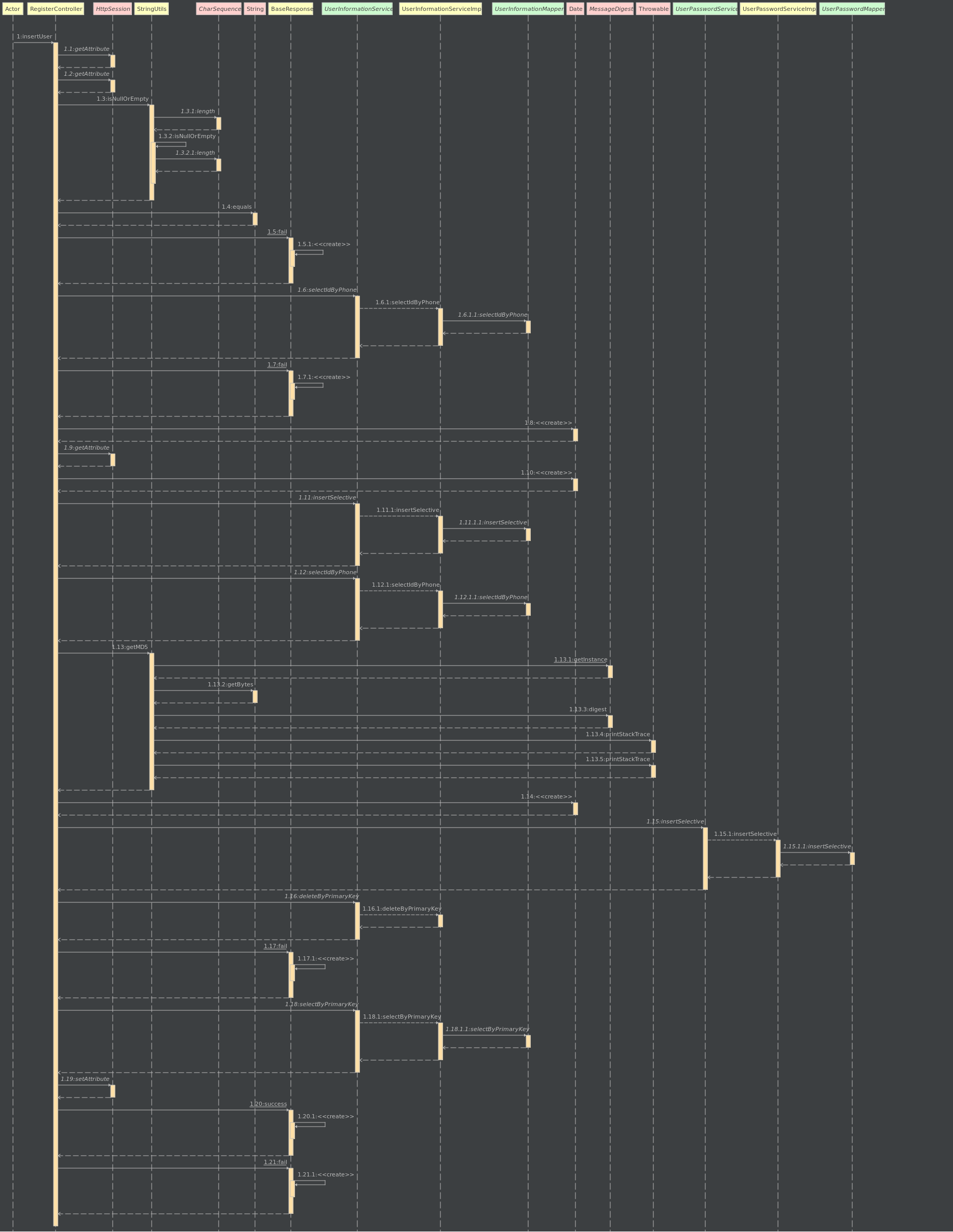
# 6 动态模型

## 6.1 场景（Scenarios）

时序图（Sequence Diagram），又名序列图、循序图，是一种UML[交互图](https://baike.baidu.com/item/%E4%BA%A4%E4%BA%92%E5%9B%BE/6812905)。它通过描述对象之间发送消息的时间顺序显示多个对象之间的动态协作。它可以表示[用例](https://baike.baidu.com/item/%E7%94%A8%E4%BE%8B/163511)的行为顺序，当执行一个用例行为时，其中的每条消息对应一个类操作或[状态机](https://baike.baidu.com/item/%E7%8A%B6%E6%80%81%E6%9C%BA/6548513)中引起转换的触发事件。

### 6.1.1 登陆注册场景

用户分别输入手机号、密码与确认密码后点击注册按钮完成注册。前端部分判断用户是否在以上三个部分输入了信息，接着判断邮箱的格式是否正确、密码与确认密码两个部分的内容是否相同等用户输入信息。当用户内容输入正确后，客户端会将用户所输入的内容传给服务器。服务器得到信息后会再一次判断信息的格式等是否符合要求，无误后将得到的信息与数据库中的信息作比较，如有重复则提示前端邮箱已被注册，无重复则将信息存入数据库并提示前端注册成功。 用户分别输入手机号、密码后点击登录按钮进行登录操作。前端判断用户是否在以上两个部分输入了信息，接着判断手机号的格式是否正确。当用户内容输入正确后，客户端会将用户所输入的内容传给服务器。服务器得到信息后会再一次判断信息的格式等是否符合要求，无误后将得到的信息与数据库中的信息作比较。如果在数据库中能找到邮箱和密码与传入信息相同，则服务器返回客户端，前端得到登录成功的信息就跳转到登录成功的页面；如果不同客户端提示用户邮箱与密码不匹配、登录失败。



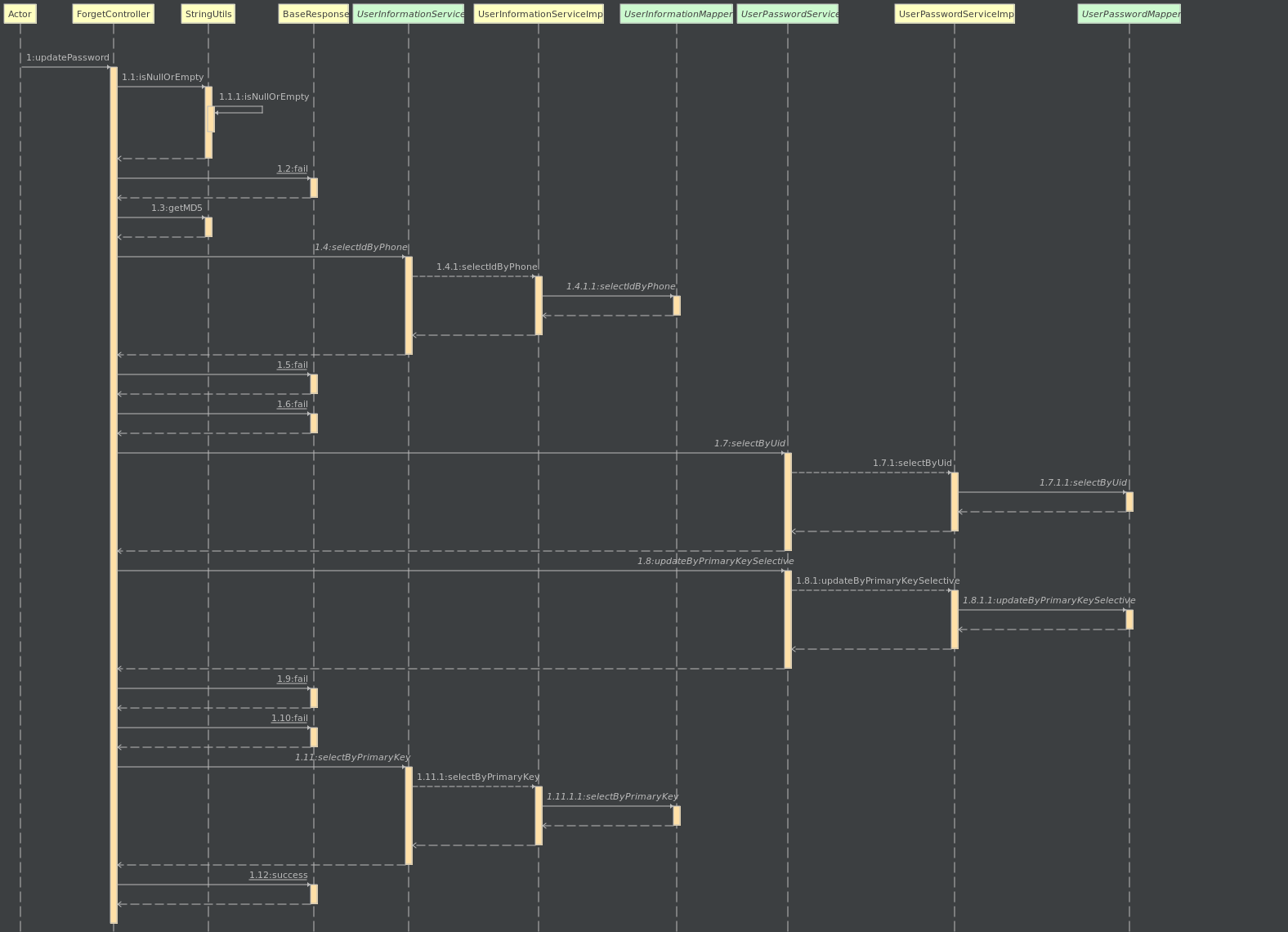
### 6.1.2 商品模糊查询

在用户使用搜索功能的时候会用到商品模糊查询，用户首先根据商品名输入大概的搜索关键词，然后商品控制层调用ShopInformation服务层使用相关算法来模糊查询商品，其中查询商品信息是调用了数据库查询信息，如果查询到符合要求的商品则返回给前台，展示给用户。



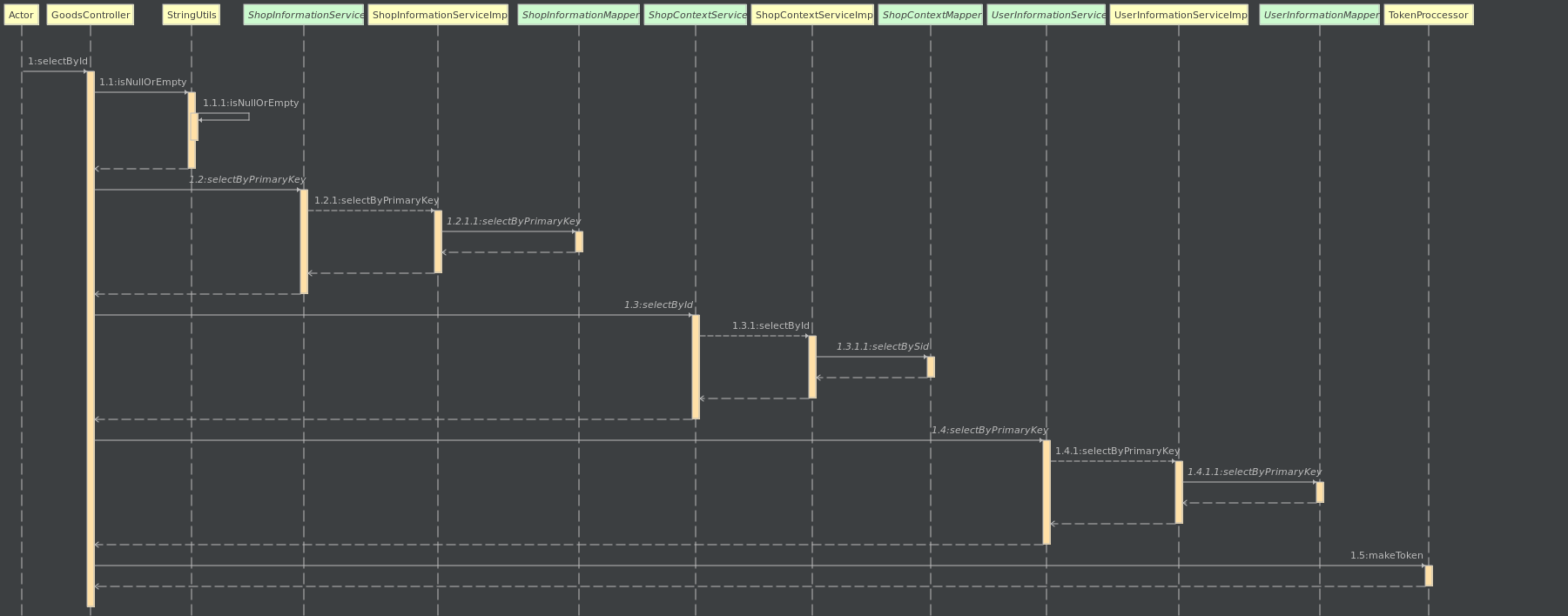
### 6.1.2用户信息修改

用户在成功登录之后，需要打开修改个人信息的面板，系统调用数据库返回给用户他的个人信息，用户选择编辑，选择要修改的内容重新填写，点击保存按钮，系统检验数据的合理性，如果成功那么前台提示用户信息已经成功修改，否则提示用户重新修改。在成功修改过后，数据库中的用户信息也得到改变。



### 6.1.2交易商品

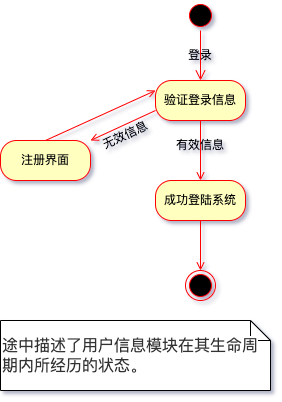
用户可以通过商品详情页直接购买商品，也可以打开购物车界面来购买，用户选中购买的商品侯点击支付，系统让用户选择支付方式，用户选择并成功支付过后，系统检测支付是否成功，如果支付成功那么前台提示用户成功支付，否则提示用户继续支付，如果交易完成，那么在用户的个人信息里面添加一笔订单，并且商品被下架。



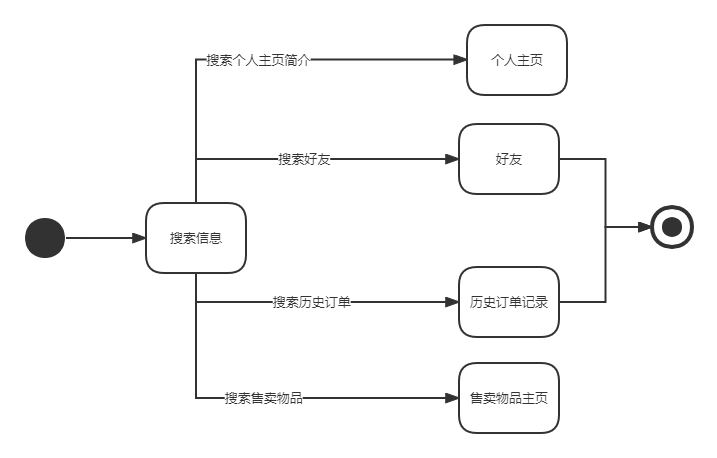
### 6.2 状态图

这部分的内容包括系统动态模型重要的部分的状态图。可能你想为每个对象画一个状态图，但事实上会导致太多不期望的细节信息，只需要确定系统中一些重要的对象并为之提供状态图即可。

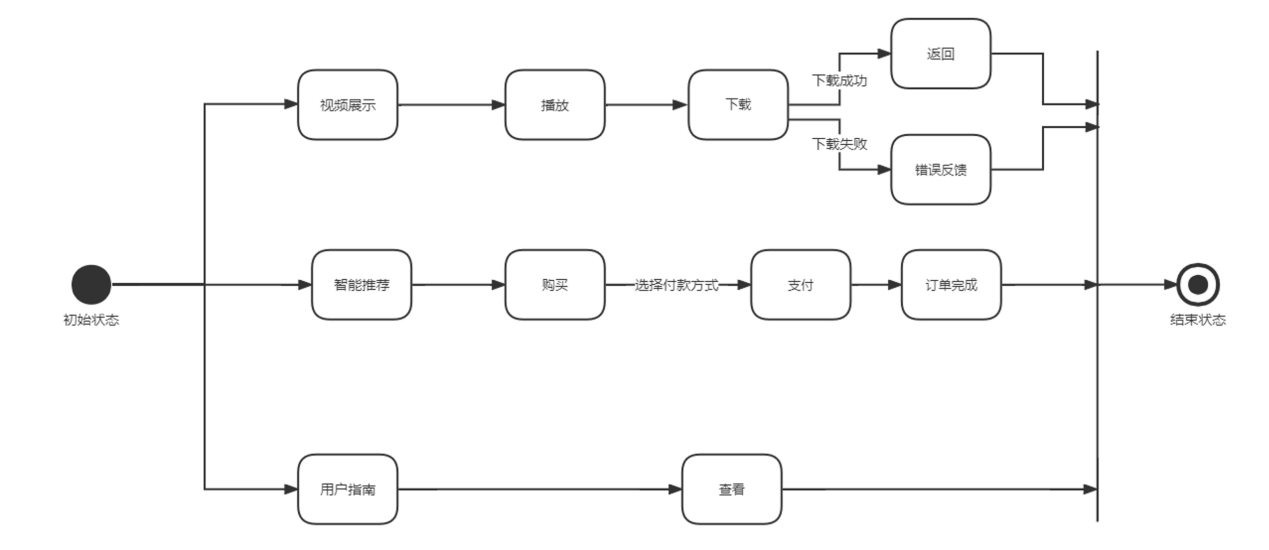
### 6.2.1 状态图1：



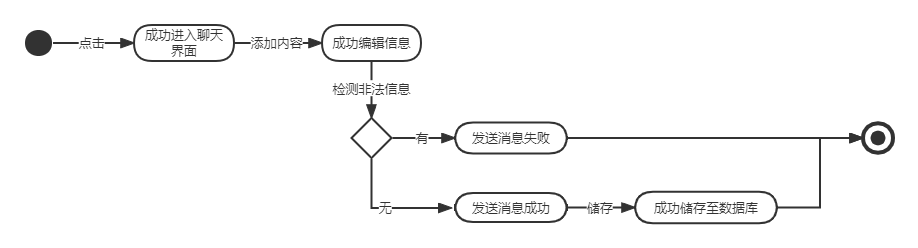
### 6.2.2 状态图2：



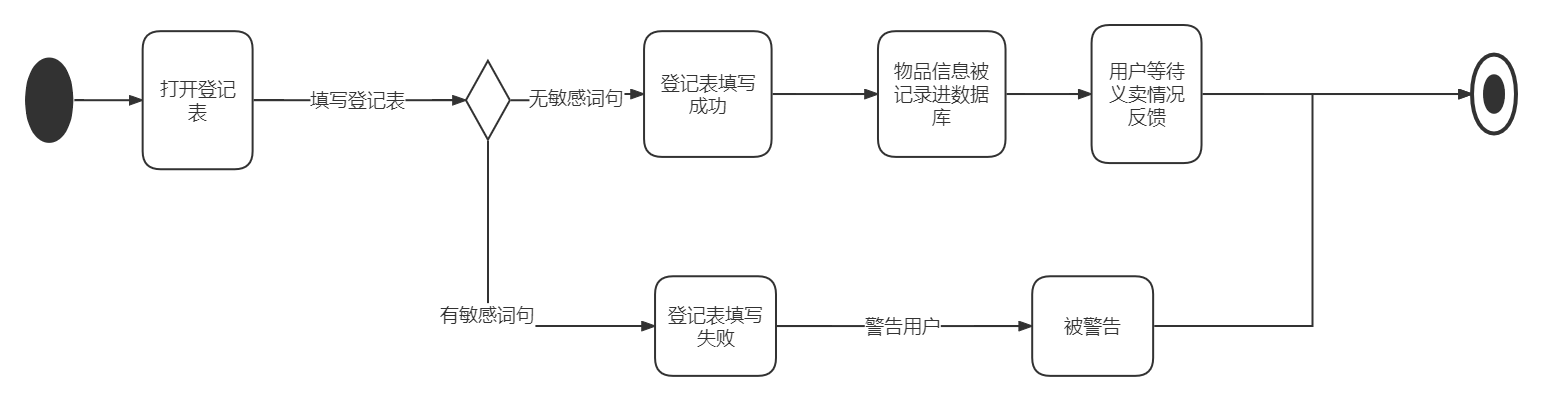
### 6.2.3 状态图3：



### 6.2.4 状态图4：



### 6.2.5 状态图5：



# 7 数据库设计

## 7.1 数据库说明

我们使用MYSQL数据库进行所有用户和商品等信息的存储。

## 7.2 数据库结构

首页需要的数据是热门商品的信息，商品信息需要建立一个Bean对象，存储内容是商品的id、商品名称、商品单价、商品类别、商品数量、商品详细信息、商品成色、商品附带的图片地址、热门程度、发表用户的id、留言的集合，当用户发表之后，保存数据到Bean对象中，并将它存储到数据库。首页请求数据之后，根据热门度返回多个数据，将数据存储到list集合中，并且将它转换为JSON格式数据，返回到页面，页面解析数据并且显示。

　　商场则负责显示所有的商品数据，实现侧边栏点击指定的类别之后，显示对应类别的数据，需要的是类别的id，因此类别Bean对象的数据是类别id、类别名称，点击之后，通过类别id获取到商品详情集合，并且将它转为JSON格式的数据，返回到前端页面，显示数据。

　　点击商品之后，需要显示商品详细信息，通过商品的id返回指定商品的Bean对象，转化为JSON格式的数据，返回到前端页面显示，需要显示商品的留言信息，因此需要留言的Bean对象，包括数据留言id、留言的商品id、留言内容、留言时间，结合商品的详情返回JSON格式数据，前端解析之后显示。

　　求购商城，负责显示所有的求购商品，需要求购商品的Bean，内容包括求购商品id、求购商品名称、求购商品数量、求购商品单价、求购商品详细信息、求购商品留言集合。页面显示时从后台返回对象集合转变的JSON数据，前端解析数据并且显示出来。

　　登陆界面需要验证用户信息，用户信息包括内容用户id、用户名、用户的手机号码、用户密码、用户真实姓名、用户地址、用户email、用户性别、用户学号。验证的时候只需要判断手机号码和密码是否符合，符合则验证通过，返回验证结果，用JSON存储数据。失败则返回失败信息。

　　主要用到的数据结构是ArrayList负责组合各种信息，并且转换为JSON格式数据返回。数据需要排序的时候，用到的是快速排序，负责排列热门商品等的信息，并且返回数据到前端页面进行显示。

## 7.3 表结构

用户信息表

| **字段名** | **字段类型** | **是否可为空** | **备注** |
| --- | --- | --- | --- |
| Id | Int(11) | 否 | 主键 |
| Modify | Datetime | 是 | 修改时间 |
| Username | Varchar(50) | 否 | 用户昵称 |
| Phone | Char(11) | 否 | 用户手机号码 |
| realName | Varchar(20) | 是 | 用户真实姓名 |
| Clazz | Varchar(20) | 是 | 用户所在班级 |
| Sno | Char(12) | 是 | 用户学号 |
| Dormitory | Varchar(20) | 是 | 宿舍号 |
| Gender | Char（2） | 是 | 性别 |
| Createtime | Datetime | 是 | 创建时间 |
| Avatar | Varchar(200) | 是 |  |

用户密码表

| **字段名** | **字段类型** | **是否可为空** | **备注** |
| --- | --- | --- | --- |
| Id | Int | 否 | 主键 |
| Modify | Datetime | 是 | 修改时间 |
| Password | Varchar(24) | 否 | 用户密码 |
| Uid | Int | 否 | 用户id |

商品表

| **字段名** | **字段类型** | **是否可为空** | **备注** |
| --- | --- | --- | --- |
| Id | Int(11) | 否 | 主键 |
| Modify | Datetime | 是 | 修改时间 |
| Name | Varchar(50) | 否 | 商品名称 |
| Level | Int | 否 | 商品成色 |
| Remark | Varchar(255) | 是 | 商品详细信息 |
| Price | Decimal(0.00) | 否 | 商品价格 |
| Sort | Int | 否 | 商品类别 |
| Count | Int | 否 | 商品数量 |
| Display | Int | 否 | 商品是否被下架 |
| Transaction | Int | 否 | 交易方式 |
| Sales | Int | 否 | 商品销量 |
| Uid | Int | 否 | 发表的用户id |
| Image | Varchar(255) | 否 | 商品的图片地址 |

商品留言

| **字段名** | **字段类型** | **是否可为空** | **备注** |
| --- | --- | --- | --- |
| Id | Int(11) | 否 | 主键 |
| Modify | Datetime | 是 | 修改时间 |
| Sid | Int | 否 | 商品的id |
| Content | Varchar(122) | 否 | 留言内容 |
| Display | Int | 否 | 是否可见 |
| Uid | Int | 否 | 评论的用户id |

我的收藏

| **字段名** | **字段类型** | **是否可为空** | **备注** |
| --- | --- | --- | --- |
| Id | Int(11) | 否 | 主键 |
| Modify | Datetime | 是 | 修改时间 |
| Sid | Int | 否 | 商品的id |
| Display | Int | 否 | 是否可见 |
| Uid | Int | 否 | 评论的用户id |

求购商品

| **字段名** | **字段类型** | **是否可为空** | **备注** |
| --- | --- | --- | --- |
| Id | Int(11) | 否 | 主键 |
| Modify | Datetime | 是 | 修改时间 |
| Name | Varchar(50) | 否 | 求购商品名称 |
| Remark | Varchar(255) | 是 | 求购商品详情 |
| Price | Decimal(0.00) | 否 | 求购商品价格 |
| Sort | Int | 否 | 求购商品类别 |
| Count | Int | 否 | 求购商品数量 |
| Display | Int | 否 | 信息是否被删除 |
| Transaction | Int | 否 | 交易方式 |
| Uid | Int | 否 | 发表的用户id |
| Image | Varchar(255) | 否 | 商品的图片地址 |

求购商品留言

| **字段名** | **字段类型** | **是否可为空** | **备注** |
| --- | --- | --- | --- |
| Id | Int(11) | 否 | 主键 |
| Modify | Datetime | 是 | 修改时间 |
| Qid | Int | 否 | 求购商品的id |
| Content | Varchar(122) | 否 | 留言内容 |
| Display | Int | 否 | 是否可见 |
| Uid | Int | 否 | 评论的用户id |

我发布的商品

| **字段名** | **字段类型** | **是否可为空** | **备注** |
| --- | --- | --- | --- |
| Id | Int(11) | 否 | 主键 |
| Modify | Datetime | 是 | 修改时间 |
| Display | Int | 否 | 是否被删除 |
| Uid | Int | 否 | 用户id |
| Sid | Int | 否 | 对应商品id |

我求购的商品

| **字段名** | **字段类型** | **是否可为空** | **备注** |
| --- | --- | --- | --- |
| Id | Int(11) | 否 | 主键 |
| Modify | Datetime | 是 | 修改时间 |
| Display | Int | 否 | 是否被删除 |
| Uid | Int | 否 | 用户id |
| Qid | Int | 否 | 求购商品id |

已购买的商品

| **字段名** | **字段类型** | **是否可为空** | **备注** |
| --- | --- | --- | --- |
| Id | Int(11) | 否 | 主键 |
| Modify | Datetime | 是 | 修改时间 |
| State | Int | 否 | 商品当前的状态 |
| Uid | Int | 否 | 用户id |
| Sid | Int | 否 | 商品id |
| Quantity | Int | 否 | 商品数量 |

购物车

| **字段名** | **字段类型** | **是否可为空** | **备注** |
| --- | --- | --- | --- |
| Id | Int(11) | 否 | 主键 |
| Modify | Datetime | 是 | 修改时间 |
| Display | Int | 否 | 商品是否被删除 |
| Uid | Int | 否 | 用户id |
| Sid | Int | 否 | 商品id |
| Quantity | Int | 否 | 商品数量 |

# 8 非功能性设计

在这个部分，必须说明如何处理需求文档中指定的非功能性需求。尽可能客观地评估系统应付每一个非功能性的需求的能力程度。如果某些非功能性需求没有完全在设计的系统中实现，请务必在此说明。另外，你也需要对系统将来的进化作一个估计并描述本设计如何使系统能够适应这些可预见的变化。

（1）评估详细设计 满足安全性需求

（2）框架配置性工作 满足，做出详细说明，例如，安全性，使用框架提供

## 8.1 性能需求

### 8.1.1 处理能力

由于是在线二手交易平台，其处理能力主要考虑系统能承载的最大并发用户数，按照实际情况的规划，系统至少能承载的最大并发用户数要求达到全校学生总人数\*φ，φ为0至1的常数，随服务器容量而定。

### 8.1.2 响应时间

为了能够快捷地提供在线测评服务，系统应该能够快速地响应在线测评请求。用户最终得到结果的响应时间除了与系统响应速度有关外，还与网络状况有关。因此对Web服务器端需要较高的要求。

## 8.2 安全性需求

传输的数据都采用高强度的加密算法加密 (DES)，使得数据即使泄漏、被截获后，也无法识别相关的数据内容，确保数据安全。对于客户端与服务器交互的数据，使用安全套接子层 (SSL,SSL 加密传输主要是针对 WEB的数据传输，基于重要信息的传输安全考虑而设计的) 进行信息交换，并在客户移动终端和服务器之间重要的信息的交换。

# 9 辅助文档

* 可行性分析报告
* 项目需求规格说明书

# 10 词汇索引

| **概念** | **具体定义** |
| --- | --- |
| 活动图 | 活动图是UML用于对系统的动态行为建模的另一种常用工具，它描述活动的顺序，展现从一个活动到另一个活动的控制流。活动图在本质上是一种流程图。虽然活动图与状态图都是状态机的表现形式，但是两者还是有本质区别：活动图着重表现从一个活动到另一个活动的控制流，是内部处理驱动的流程；而状态图着重描述从一个状态到另一个状态的流程，主要有外部事件的参与。 |
| 用例图 | 用例图（英语：use case diagram）是用户与系统交互的最简表示形式，展现了用户和与他相关的用例之间的关系。通过用例图，人们可以获知系统不同种类的用户和用例。用例图也经常和其他图表配合使用。 |
| 时序图 | 时序图（Sequence Diagram），又名序列图、循序图，是一种UML交互图。它通过描述对象之间发送消息的时间顺序显示多个对象之间的动态协作。它可以表示用例的行为顺序，当执行一个用例行为时，其中的每条消息对应一个类操作或状态机中引起转换的触发事件。 |
| 类图 | 类图(Class diagram)由许多（静态）说明性的模型元素（例如类、包和它们之间的关系，这些元素和它们的内容互相连接）组成。类图可以组织在（并且属于）包中，仅显示特定包中的相关内容。类图(Class diagram)是最常用的UML图，显示出类、接口以及它们之间的静态结构和关系；它用于描述系统的结构化设计。类图(Class diagram)最基本的元素是类或者接口。 类图主要用在面向对象软件开发的分析和设计阶段，描述系统的静态结构。类图图示了所构建系统的所有实体、实体的内部结构以及实体之间的关系。即．类图中包含从用户的客观世界模型中抽象出来的类、类的内部结构和类与类之间的关系。它是构建其他设计模型的基础，没有类图，就没有对象图、状态图、协作图等其他UMI．动态模型图．也就无法表示系统的动态行为。类图也是面向对象编程的起点和依据。 类图用于描述系统中所包含的类以及它们之间的相互关系，帮助人们简化对系统的理解，它是系统分析和设计阶段的重要产物，也是系统编码和测试的重要模型依据。 |
| 状态图 | 状态图(Statechart Diagram)是描述一个实体基于事件反应的动态行为，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。通常我们创建一个UML状态图是为了以下的研究目的：研究类、角色、子系统、或组件的复杂行为。UML通过使用状态图来支持基于事件的模型，状态图用来来描述一个类对象在不同用例间状态的迁移。当一个用例或某个事件发生时，类对象的状态就会发生迁移，状态图有助于分析人员审核业务逻辑，以及完善静态模型。 |
| 数据流图 | 数据流图或数据流程图（Data Flow Diagram），缩写为DFD。数据流图DFD是描述系统中数据流程的一种图形工具，它标志了一个系统的逻辑输入和逻辑输出，以及把逻辑输入转换逻辑输出所需的加工处理。DFD显示系统将输入和输出什么样的信息，数据如何通过系统前进以及数据将被存储在何处。它不显示关于进程计时的信息，也不显示关于进程将按顺序还是并行运行的信息，而不像传统的关注控制流的结构化流程图，或者UML活动工作流程图，它将控制流和数据流作为一个统一的模型。 数据流图从数据传递和加工的角度，以图形的方式刻画数据流从输入到输出的移动变换过程。 |