一室软件设计文档

目录

1、引言

1.1、编写的目的和范围

本详细设计说明书编写的目的是说明程序模块的设计考虑，包括软件描述、业务逻辑、前后端架构设计和流程逻辑等，为软件编程和系统维护提供基础。本说明书的预期读者为系统设计人员、软件开发人员、软件测试人员和项目评审人员。

1.2、术语表

序号	术语或简略语	说明性定义
1	PM	Project Manager，项目经理
2	FE	FrontEnd，项目前端
3	SE	ServiceEnd，项目服务端

1.3、参考资料

资料名称	作者	文件编号、版本	首页	github
vue	尤雨溪	^2.2.2	https://vuejs.org/	https://github.com/vuejs/vue
iview	TalkingData	^2.0.0-rc.9	https://www.iviewui.com	https://github.com/iview/iview

1.4、使用的文字处理和绘图工具

文字处理软件：石墨（https://shimo.im）

绘图工具：Umlet

2、全局数据结构说明

2.1、常量

前端使用统一主题色：#2c3e50

部分背景色：#48392e

2.2、变量

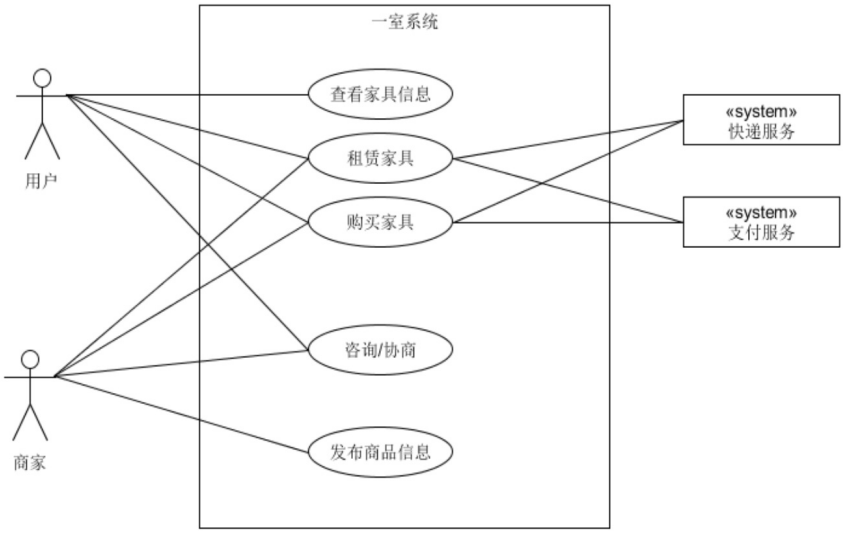
2.3、数据结构

用到的常见数据结构：

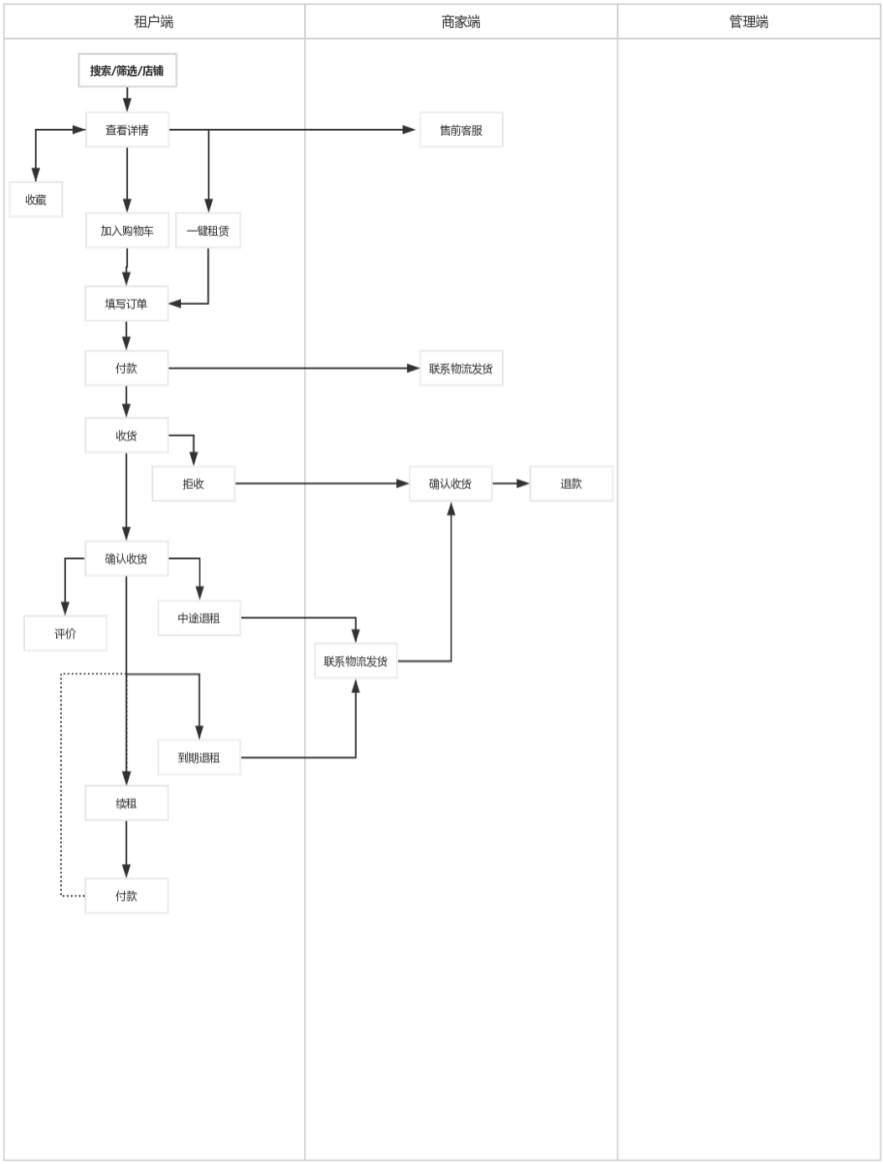
- Quese
- List
- Stack
- Persistent data structure

3、模块设计

3.1、用例图



3.2、流程图



3.3、功能设计说明

3.3.1、用户服务

该服务提供了对用户的基本操作，用户包括买家用户、商家用户、管理用户等，根据不同的用户区分系统提供不同的功能，其中买家用户拥有租赁商品、意见反馈、通讯评论等功能，而商家具有商品管理、反馈管理、订单管理等功能，管理用户拥有对商品的审核、对店家的审核等功能，与电商平台的用户设计有类似之处。

3.3.2、商品服务

该服务提供了原子商品的基本操作，包括商品基本信息、商品类别的增删改查，统一商品原子类，为更复杂的组合商品提供更好的数据基础，这个服务是整个系统中最重要的一环。

3.3.3、组合商品服务

该服务提供了商品的组合套餐等需求，通过组合原子商品，为商品提供了更多元化的选择，通过这个服务，可以针对不同的场景提供不同的组合商品，为商家提供了更好的服务。

3.3.4、订单服务

该服务为用户和商家提供了商品的租赁服务，通过对商品的选择以及下单，商家可以为用户提供所协商的商品，同时这个订单为用户和商家提供了凭据，通过历史记录我们可以为用户或者商家提供维权服务。

3.3.5、通讯服务

该服务为用户和商家提供了在线交流的平台，聊天记录保存在云端，可以随时调取记录；另外也为管理端提供了发送系统消息的平台，可以及时地为用户提供信息服务；除此之外还提供了商家发送商品推送的功能，能更有利于商家发布商品时用户能及时地接收到信息。

3.3.6、物流服务

该服务为用户提供物流服务，当用户下单后，商家或者物流平台可以在任何时间为用户提供该商品的现状信息，用户通过物流服务查询订单情况，可以随时掌握商品信息。

3.3.7、信用和支付服务

该服务是用户和商家个人信用的凭据，系统根据既定的规则给每位用户和每个商家的信用进行评级，通过信用体制，用户和商家都可以更好地了解到对方，以便对订单做出合适的回应，另外支付服务通过和外支付平台合作，为用户提供更友好的支付体验。

4、接口设计

4.1、内部接口

前端的内部接口主要由以下几个方面组成：组件、页面、统一数据模型、页面路由，其中所有的页面都由多个组件组成，页面与组件之间的通信通过统一数据模型进行沟通，同时统一数据模型和后端进行直接的交流，而页面路由控制多个页面之间的跳转逻辑，控制页面切换。

4.2、外部接口

4.2.1、接口说明

此接口说明阐述如何设计接口，以满足快速前后端对接的需求，在这里我们主要说明的是前端与后端进行交流的接口设计部分。

4.2.2、调用方式

接口的设计采用restful的设计理念，其中GET指代资源的获取，POST指代资源的添加，PUT指代资源的更新，DELETE指代资源的删除，其中GET、PUT、DELETE三个方法是幂等的。全局统一使用 ^api/**/* 来标明接口数据来自后端动态资源，区别于前端静态资源。除此之外，我们使用JSON格式的数据进行交流，前端使用Promise封装统一接口，每一个接口对应的方法都需要完成明确的功能。

5、 数据库设计

5.1 产品业务属性

基础属性

指设计在商品表的一些基础字段。

其中可选的设计点有：

- 副名称：由于商品名称经常要加上一些促销信息，如本商品参与什么活动之类。但经常改动主名称容易导致出错，所以增加此字段来专门管理促销信息。显示时连接到主名称后即可。
- 产品描述：产品描述建议另设计一表存放，对提高产品搜索、产品列表显示有帮助。

- 状态：常见的状态有草稿、未发布、发布、下架等，如果是逻辑删除的，还有已删除状态。

价格

系统支持产品SKU，实际价格是在产品SKU实体中管理的。
促销价格不在这里管理，在营销管理模块统一管理。

SEO相关

集中管理各类SEO相关的信息。

商品媒体

- 主图：由于显示频繁，会直接设计在产品表中（或是冗余）。
- 多图（即附图）：开发中会提供多种查看方式。

扩展属性

是设计最困难的部分，也是商品是否可以灵活扩展的关键。

库存

库存这里是常见的开发迭代点。在研发早期，一般这里直接设计成支持零库存和单一数值库存。在其它功能完成后，才会对这里做扩展，开发内嵌的库存子模块或者整合外部系统。

外部关联

商品的外部关联非常的多，随着系统的扩展，肯定会有新的外部关联实体。所以商品模块的开发，需要提供大量的外部接口或者Tag封装（如商品选取器等。）

5.2 产品SKU以及SPU模型

SKU=Stock Keeping Unit(库存量单位)

同一型号的产品，或者说是同一个产品项目（产品条形码是针对企业的产品项目来进行定义的），因为产品与产品之间有某些属性不同，用以区别开这些不同产品的属性即产品变异属性，又称作SKU属性，因为它决定了SKU的绝对数量。

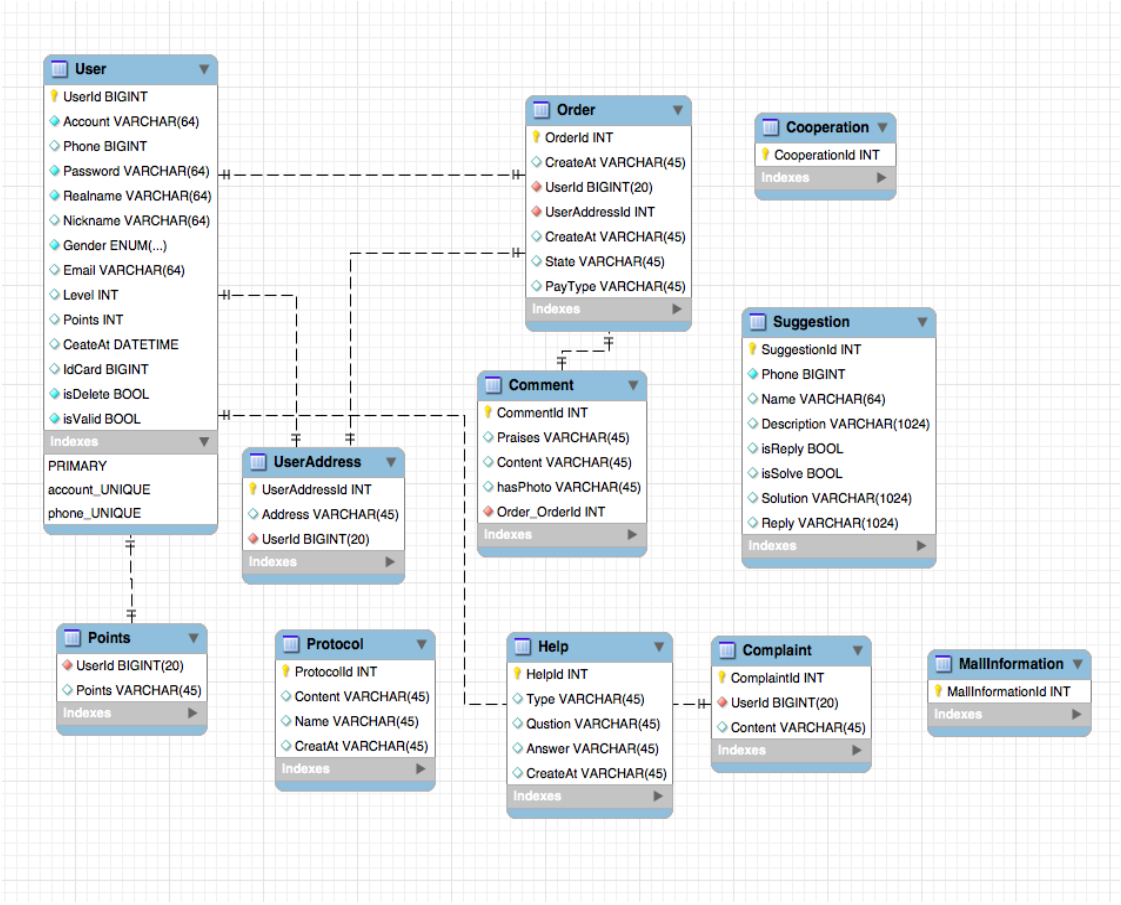
SPU = Standard Product Unit （标准产品单位）

SPU是商品信息聚合的最小单位，是一组可复用、易检索的标准化信息的集合，该集合描述了一个产品的特性。

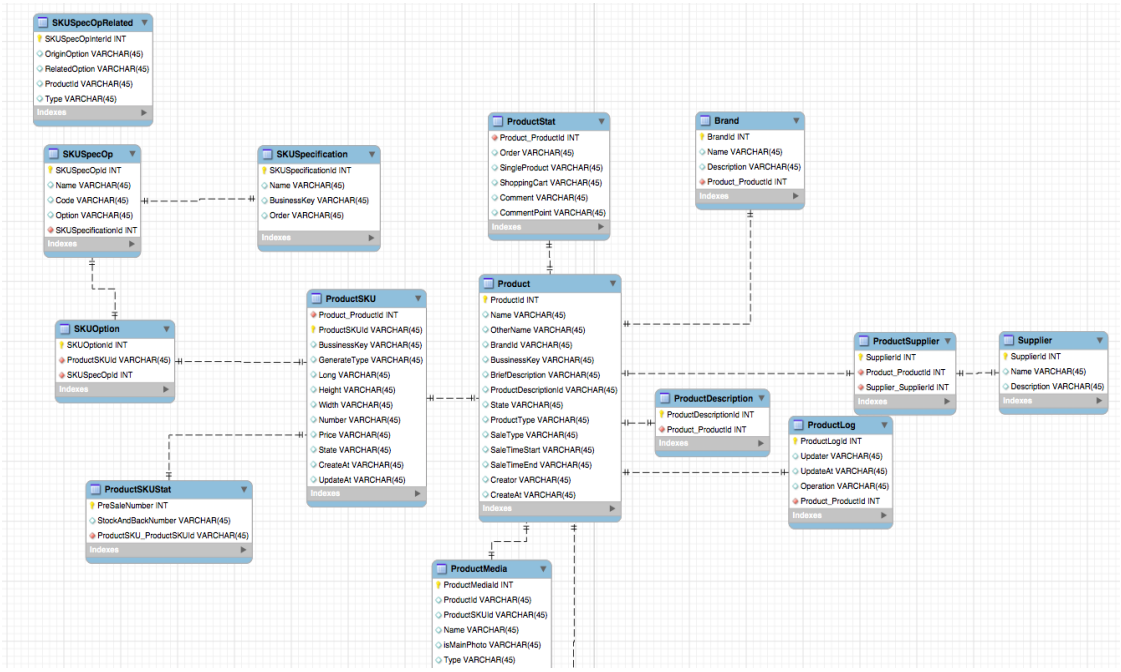
通俗点讲，属性值、特性相同的商品就可以称为一个SPU。

5.3 数据库 EER Diagram

整体（不含Product部分）



Product



6、系统安全保密设计

6.1、说明

在电子商务中系统的安全性和保密性是必不可少的，也是系统中保证用户权益和经济安全最重要的一环，为了防止恶意攻击和信息泄漏等风险，拥有一个具有高度保密性和安全性的架

构设计是必不可少的，同时也必须保障数据的安全性和可恢复性。

6.2、设计

6.2.1、SSL协议

SSL是一个传输层的安全协议，用于在Internet上传送机密文件。它由SSL记录协议、SSL握手协议和SSL警报协议组成。SSL协议主要提供三方面的服务：

1. 用户和服务器的合法性认证。
2. 加密数据以隐藏被传送的数据。
3. 保护数据的完整性。

6.2.2、SET（Secure Electronic Transaction，安全电子交易）

SET协议向基于信用卡进行电子交易的应用提供了实现安全措施的规则。SET支付系统主要由持卡人、商家、发卡行、收单行、支付网关和认证中心6个部分组成。

6.2.3、认证中心

CA是电子商务体系中的核心环节，是电子交易中信赖的基础。CA的功能有证书发放、证书更新、证书撤销和证书验证。

6.2.4、安全体系

措施包括防火墙、入侵检测、病毒和木马扫描、安全扫描、日志审计系统等。安全审计包括两个方面：采用网络监控与入侵防范系统，识别网络中各种违规操作与攻击行为，即时响应并进行阻断；对信息内容和业务流程的审计，可以防止内部机密或敏感信息的非法泄漏和单位资产的流失。安全审计功能的6个部分有：安全审计自动响应、安全审计数据生成、安全审计分析、安全审计浏览、安全审计事件选择和安全审计时间存储。

6.2.5、系统备份与恢复

业务持续规划

1. 建立一个恢复规划的实施所需要的工程组以及相应的支撑基础设施。
2. 实施对攻击行为以及风险的管理评估，从而识别其是否是恢复规划所需解决的问题。
3. 实施业务影响分析，用来判定业务的时间急迫性以及确定最大可忍受停工期。
4. 恢复规划的保存和实施。
5. 建立并采用一种可实施的测试和维护策略。

灾难恢复规划

1. 与日常生产及运行息息相关的关键性系统。
2. 部分机构的重心系统，采用类似的架构。
3. 在另一地区设计规模较小但架构相同的系统。
4. 利用备份工具，包括磁带、磁盘和光盘等。

7、系统性能设计

7.1、缓存以及缓存层

在数据层和应用层之间增加数据缓存层，提供全局数据服务。可以大大减少数据库往返次数。与读取数据库和读取大文件（如XML文件）比，读取内存的速度无疑要快的多。所以对经常要访问的数据进行缓存是非常好的实践方法。因为现在系统往往内存很大，可以充分利用大内存，而共享内存更能实现数据并发访问。

7.2、多线程

现在基本上大部分软件实现多线程或多进程，多线程对单CPU系统还只是顺序利用CPU时间和改善用户体验，多CPU系统才是真正的并行。要注意的是多线程不要争抢访问同一资源而导致部分串行操作，要做到真正的并行操作多线程并不容易。另外，在多线程间同步一个庞大的资源，过多创建线程又没有实现线程池也会导致系统性能下降。

7.3、负载平衡

物理上增加地位对等的集群服务器（Cluster），通过负载分配算法分配相应服务器来相应客户端请求。很多系统支持负载均衡，Windows server2003 IIS就支持负载均衡服务，其他如WebLogic, WebSphere也有集群版本支持负载均衡。当然你也可以自己实现负载分配算法。

7.4、数据库优化

如果应用程序使用了数据库，可以采取许多步骤来消除访问和写入数据时的瓶颈：

1. 标识潜在的索引，但不要创建过多的索引。
2. 如果使用 SQL Server，则使用 SQL Server 的事件探查器和索引优化向导。
3. 监视处理器的使用；理想范围是：75-80% 处理器时间。
4. 使用查询分析器分析查询计划以优化查询。
5. 使用存储过程优化性能。
6. 标准化写入的大量数据 —写入较少的数据。
7. 取消标准化读取的大量数据 —读取较少的数据。

7.5、文件系统优化

有时候系统性能不好，但当你关闭写log的功能，性能一下子提高很多。因为频繁的打开关闭大log文件时I/O开销非常大，同样记录log到数据库也一样。所以，release版尽量减少写log，或干脆移到裸设备上。

频繁打开关闭文件对系统性能下降程度是惊人的，可以通过一些变通办法来减少文件的频繁操作。

例如，原来的缓存持久化实现是保存在XML文件，每次要获得一个配置项，都打开XML文件，通过XPath拿到这个配置项的值，这样效率不高，而且容易把这个XML文件lock住；改进的方法是：通过比较XML文件的修改时间(System.IO.File.GetLastWriteTime)判断是否要再次打开文件，大大提高了效率；另一个可以改进的方法是：启动时读取所有配置到一个静态的HashTable，每次要获得一个配置项都从内存HashTable获取，在最后或适当的时候持久化到XML。

7.6、代码性能设计

在编程实现上，代码性能设计也很重要，一些昂贵的操作会占用大量的资源和CPU时间。例如，字符串相加没用StringBuilder, 频繁创建对象，差劲的排序或递归算法，过多的装箱拆箱，过多的使用反射（Reflection），频繁new HashTable或大的数组，用异常（Catch Exception）用做正常的逻辑，使用复杂的正则表达式，等等。

7.7、应用层

比如应用层和数据库的API，在.Net中就有就有DataReader、DataSet和IList等的选择以及转换等，这个根据具体情况而定；还有就是大家常采用的数据的格式化和压缩，以及采用分页，减少传输的数据量；是否可以把一部分处理逻辑放在客户端呢，减少服务端的工作量。界面端也是有很多针对性能优化的考虑，例如绘图，控件重绘都是非常耗资源的，各控件的数据

加载和数据绑定性能也各不相同，尽量采用惰性加载，异步加载；初始化和启动速度等都是需要考虑和优化的。

8、系统出错处理

8.1、出错信息

采用错误提示窗口向用户提示错误，并友好地处理错误。例如，用户登陆失败时，出现错误提醒。

8.2、补救措施

- 定期建立数据库备份，一旦服务器数据库被破坏，可以使用最近的一份数据库副本进行还原。
- 为防止服务器故障，预备另外一台服务器，只要主服务器出现故障，可以迅速启动预备服务器运行系统。

8.3、系统维护设计

- 基础数据维护：对于一些基础数据，安排管理员进行维护。
- 数据库备份和恢复：利用数据库自身提供的备份和恢复功能实现。
- 系统升级维护：根据用户使用效果调查表，筛选用户提出的功能要求，对于合理的要求予以采纳，并安排人员对系统进行修改和完善。