# 摘要

# ABSTRACT

目录

# 引言

## 1.1 研究背景及意义

自动售货机是一种继超市、便利店之后的又一种全新的商业零售模式。像阿里推出的无人超市一般，自动售货机是更加小型、便捷的无人超市，是自助购物的终端，商家经过调研与考察将其单独放置在客流量较大的场所，如商场、地铁站和学校等地。自动售货机与传统便利店、无人超市、网上购物等各自代表着不同角色的销售模式。20世纪70年代起，自动售货机零售业在欧美国家和日本等地发展起来，因其不受地域的限制，占地面积较小，无需专人盯守、省人工、省成本且24小时不停机服务，而被称为24小时营业的微型超市。自动售货机在很多发达国家的发展非常迅猛，在日本一些自动售货机甚至普及到田间地头等客流量不大的地区。

我国自引入自动售货机以来，这种新型的销售方式就得到了广大消费者的喜好。19世纪末20世纪初，我国经济逐步发展，人口大量的涌入城市，能够随时随地的购买到当即需求的产品，正满足人们当下的需求。随着消费者需求的增长以及售货机行业的发展，自动售货机中商品的销售种类也逐渐多元化，由传统概念的“饮料机”发展为可以售卖零食、咖啡、面膜、药品等众多商品的“小百货机器”。

然而自动售货机引入我国之后并未像互联网、电子商务一样得到迅猛发展，多年来一直处于一种不温不火的状态，以至于很多人提到自动售货机便把它归结为传统行业。造成这种现象的原因有很多：首先，人力物力成本的提高，让售货机零售业的投放成本和运维成本逐渐增高，利润越来越微薄；其次，信息社会的迅猛发展，让已习惯高度信息化的生活的消费者难以再适应传统的售货机销售出渠道；接着，自动售货机售卖模式多年来并未得到很大提升，传统的投币式已让很多没有足够零钱却急需商品的消费者头疼不已；最后，很多传统自动售货机并未实现实时数据同步，管理员并不能实时的清楚售货机货道内的信息，以及每种商品的销售情况，这也给售货机的发展带来阻碍。

面对自动售货机的发展的众多问题，自动售货行业并未因此萎靡不振，而是另辟蹊径，将自动售货机和这一传统行业和新时期的新兴技术、新型销售方式进行结合，自动售货机行业的发展逐渐进入井喷期。随着互联网技术的发展，传统售货机的交互模式已经显得尤为落后，因而自动售货机将传统的仅仅可以使用投币和刷卡的购物方式增加为可以使用微信、支付宝、银联等移动方式进行支付。多元的支付方式不仅方便了消费者，还省去了每天清点现金的麻烦，更为节省人力。同时在售货机上新增一个显示器，用于显示优惠信息、商品支付信息、广告信息等信息，不仅增强了购物的人机交互性，还增加了一项新的赢利点。这些新型的售货机模式结合了互联网的优势，将售货机行业再次拉回销售业的大竞争之中。

截至目前，已经有多家自动售货机厂商使用这种模式，将售货机和互联网相结合，构造新的赢利点和优势。然而对于一些中小型售货机厂商和售货运营商来说，公司规模较小，没有规模的软件开发和软件运维团队，要想使用新型的售货机就要通过外包公司或者其他渠道购买自动售货机管理系统。后期对于系统的运维需要购置机器和人工，这也是一项不小的开支。因而开发一套能够将售货机行业商家既能使用新型的软件系统，又能节省开销的软件系统，就显得尤为重要。

SaaS(Software-as-a-Service)是互联网技术的一种全新的软件应用模式，是软件即服务的简称，这种模式下将软件部署在服务提供方的服务器，同时软件服务商提供软件的开发、软件运维、数据管理等服务。SaaS允许用户按需支付软件的租用费用，而不需要关注软件的开发和运维，用户通过互联网使用软件，多个用户共享一套软件。这种模式极大地提高了软件的复用率、减少了软件开发的成本，对自动售货机行业来说，能够减少软件开发的成本和开销，能够帮助售货机更好的面对成本、效率的挑战。

## 1.2 研究现状

随着国内人力物力和劳动成本的增加，城市化发展、自动售货机的改良以及支付方式的发展，自动售货机迎来了一次发展的井喷期，整个行业都在改善售货机的发展状态，售货机制造商逐步优化硬件，售货机系统开发逐步更新，整个自动售货的行业都在大步的发展。而造成这一现象的原因离不开国内互联网技术的发展。

自动售货机作为一种共享性质的销售媒介，在欧美和日本等国家和地区的发展相对成熟，我国自引入自动售货机以来，售货机保有量已从2004年的3万台增长为2015年的50.5万台。初期的自动售货机仅仅是引入国外的运营方式、售卖品类、软硬件技术，独立创新的东西甚少，这也造成我国自动售货机行业的发展被各方面掣肘。随着国内技术的逐渐成熟，自动售货机行业集中进行新型自动售货机的开发，首先提供一套供自动售货机运营商使用的后台管理系统，该系统用于运营商的用户管理、机器管理、货道管理、订单管理等信息；其次在售货机终端提供一套可操作的显示屏，用以提供支付宝、微信、银联等移动支付界面。后台管理系统和售货机终端系统的结合使用，一方面提高了消费者够买商品的便利性，另一方面也方便了运营商对售货机及商品的管理。

新型自动售货机与互联网和4G网络相结合，改变了产品的新颖度和支付手段，同时售货机空闲时可在显示屏上播放广告，增加额外赢利点，这次的改造是售货机行业的一个时代性的伟大变革。仅仅几年的时间，现今，地铁站、学校、机场等地随处可见这种新型的自动售货机，旧式售货机已慢慢退出投放。体验新型售货机后会发现，一种自动售货机使用全屏的方式，提供一个包含位置的二维码，用户扫描码后在手机端进行选货，之后进行微信支付，售货机自动出货，但其取消了原先的投币式，仅仅提供微信支付接口，支付类型单一；另一种自动售货机使用小屏幕显示商品列表，用户选货后进行支付购买。各种新型售货机系统如雨后春笋迅速产生和成长，在增进竞争力的同时，也造成了自动售货机行业发张良莠不齐的现状。

无论是哪种新型售货机，都给消费者带来了新的体验和便利，然而对于售货机商家来说，在保证消费者体验最优化的前提之下还要保障商家的盈利最大化。投放一台自动售货机，初期要考虑售货机的购置或生产资金、占地租金、人工雇佣等非技术成本，现今又要考虑新型软件的购置成本。软件的开发、部署和维护需要聘请专业的技术团队，而对于中小型的公司来说，这无疑是一项额外的大笔开销，虽然新型软件能够吸引消费者，增强在同行业中的竞争优势，但开销也令一些公司望而却步。

## 1.3 主要研究内容

基于现今自动售货机行业发展良莠不齐、支付手段单一、新型软件价格高昂等问题，提出基于SaaS模式的自动售货机云平台的新型方案。基于SaaS模式的自动售货机平台将软件部署在软件提供商的服务器和平台，有软件提供商提供软件的开发和运维。软件提供商搜集、调研自动售货机行业的软件需求，根据对其需求的分析设计出一套能覆盖自动售货机需求和功能的软件运行云平台。该平台采用一对多的交付方式，一套软件可供多家商户使用，软件使用者可以像租户一样，根据一定的规则缴纳租金，租用所需求的软件服务。这种多用户共享的软件服务极大地降低了软件开发和软件维护的成本，用户将其使用的功能完全托付给开发商，不需要考虑软件的开发、维护和升级，这种模式的软件服务大大降低了自动售货机公司的运营成本，同时也省去了公司聘请专业软件人员的费用，对于亟待升级售货机系统的公司来说意义深远。

## 1.4 本文结构

本文分为以下几个章节进行介绍：

1. 引言。主要对论文的研究背景以意义进行介绍，对论文的研究现状和主要研究内容进行分析。
2. 相关技术概括。主要介绍了J2EE体系结构、SSM框架、SaaS、Android和其他相关技术。
3. 需求分析。主要介绍该系统的需求分析、设计和数据库设计。需求分析从两个方面进行，一个为SaaS平台端，一个为Android端。画出数据库E-R图及用例图。
4. 系统设计与实现。系统设计与实现分为两个部分，一个部分为SaaS模式平台端，一个为安卓端。主要介绍了系统的整体架构，对系统的各个层次进行阐述，并画出时序图。
5. 测试与评估。对系统进行测试，以保证软件质量。
6. 总结与展望。介绍系统的功能和方法，提出进一步的展望。

# 相关技术概括

本章主要对文中涉及的技术和概念进行阐述，主要介绍了SaaS模式，与SaaS模式相关的多租户技术，多租户技术中使用的数据库存储模式，Spring技术和SpringMVC技术， Mybatis框架，Mybatis框架和Hibernate框架的比较，以及Android技术等内容。

## 2.1 SaaS模式介绍

SaaS是Software-as-a-Service的简称，是IBM提出的一种新型软件交付服务。SaaS服务往往和多租户技术同行，多租户技术的核心思想为SaaS软件服务提供数据隔离的思路和方法。

### 2.1.1 多租户技术

多租户技术（multi-tenancy technology）是一种软件架构技术，是旨在实现一种多个企业用户公用一套软件服务，与此同时能保证各个用户之间的数据实现隔离。多租户技术是SaaS模式开发的一项重要技术，是SaaS软件服务区别去其他服务的一个显著性标识。是SaaS软件服务数据层的重要技术，是实现多用户软件的设计的成功基础技术之一。

传统的软件技术中软件使用者和软件是严格的一对一模式，而多租户架构下的软件模式给不同的租户分配相应的权限，每个租户仅可在相应的权限范围内使用软件提供的服务。

多租户架构在数据存储方面一般分为三种：独立数据库、共享数据库，独立schema、共享数据库，共享schema。多组户数据存储的类型一般根据用户的特性进行选择，具体的类型选择要通过实时考察，以使资源利用最大化。对于数据安全性和隔离性要求极为严格，且能够负担起高昂成本性质的用户，一般采用第一种类型的存储结构，使用独立的数据库，完成数据的绝对隔离。对于数据的安全性及隔离性有要求但并没有要求绝对隔离时，可使用第二种共享数据库，其使用各自的schema进行数据的隔离和区分。对于数据信息敏感性不高，且用户想通过最小的开销完成数据的管理，可使用第三种模式的共享数据库，这种共享数据库使用相同的数据库、schema和数据库表，表中通过特定的标识对租户进行区分来实现数据的隔离。中小型企业且数据敏感度不高的用户往往采用第三种数据库存储模式，用最少的资金和最便利的开发模式实现信息化。

### 2.1.2 SaaS概述

SaaS是一种基于互联网的新型软件模式，其主要思想是软件服务商进行通过采集用户需求进行软件的开发、测试，开发方将软件统一部署到自己的服务器上，为用户提供服务器，软件服务和共享数据库。除此之外，软件服务商还提供软件和数据层的维护。用户可以根据自己的需求，按照一定的收费规则（规则可以为服务的多少或者服务的时限等）向软件服务商购买SaaS软件服务，并通过Internet网络使用所购买的软件服务。这种软件服务方式的运营类似于云服务器，用户只需缴纳一定空间和一定年限的租金便可以使用服务器，而不需再购买昂贵的服务器。

SaaS的软件服务对租户采用一对多的运营模式，软件服务商通过采集用户的需求，进行系统的设计和分析，系统开发完善后可通过直接访问互联网的方式供多家用户使用。用户缴纳的SaaS软件服务的费用往往包含了软件的开发费用，软件的运维费用以及软件的许可证等。共享的软件服务将单个用户进行软件开发和运维的成本分摊到各个租户的租金中，使原本让公司负重累累的软件开发费用变得低廉。这种多重租赁的软件模式非常适用于软件技术部门不完善的中小型企业，是其实现价格低廉的信息化的一种重要的途径。

如图，为SaaS多组户软件服务的架构。

SaaS多租户软件服务架构第一层为基础层，一般为服务器或云服务器，第二层为数据库层，第三层为SaaS多租户服务，第四层为应用层，其上为各个租户。SaaS服务商进行软件的开发和数据的维护，并为租户提供初始的用户名和密码，SaaS服务商系统管理员对各个租户进行管理。租户租用服务后具有其公司或组织的最高权限，在内部，租户最高管理员可以添加自己的子管理用户和分组，租户对租用的服务有绝对的最高权限，不同租户相互独立，互不干扰。SaaS服务商系统管理员仅仅对租户最高权限管理员进行管理，并不干涉系统使用方内部的管理和层次结构。

## 2.2 SSM框架

SSM框架即Spring + SpringMVC + MyBatis框架的整合。框架整合了Spring、SpringMVC和MyBatis，通过xml配置文件进行数据库连接池配置，事务管理和注解的配置等内容。通过Spring管理业务逻辑层，通过MyBatis管理持久层，SpringMVC实现表现层。

### 2.2.1 Spring框架和SpringMVC

Spring框架是为解决Java SE/EE企业级应用程序复杂性而创建的一种开源框架，以IoC（控制反转）和AOP（面向切面编程）为内核，是一种轻量级的容器框架。Spring使用基本的JavaBean来代替重量级的EJB框架，使用Spring后，项目中的EJB仍可继续使用。Spring的缔造者Rod Johnson摈弃了Java EE框架复杂、低效的缺陷，赋予了Spring框架更加灵巧、易于开发、测试和部署的特性。Spring框架给开发人员带来了很大的便利，众多优点使其得到广泛应用。

Spring框架方便了解耦，可控制对象间的依赖关系，避免程序的过度耦合。提供了更加简易的开发，Spring提供了用于测试的模块，可通过简单的注解实现单元测试等其他测试，用户可以随时进行测试，儿不用担心启动整个庞大的程序。Spring也提供声明式事务的支持，能够灵活的控制事务管理。Spring框架也可以和其他优秀框架进行良好的集成，如Struts2、Hibernate和MyBatis框架等。

Spring框架为Java开大提供了综合、广泛的平台，帮助开发者更简洁、专注的开发应用程序。现今已经集成了20多个模块，由七个模块组成，其每个模块都可以单独存在，也可以多个联合实现，这七个模块提供了企业级开发的包括持久层、业务层和展示层的所有支持。Spring构建在核心容器上，其主要分布如下图所示：

Spring的核心容器有四个，包括Beans、Core、Context和SpEL，核心容器构建了Spring的基础架构，提供了框架的基础功能。它实现了IoC模式， IoC是Inverse of Control的缩写，在框架中表示控制反转，它将类之间的依赖关系从代码中脱离开来，将某一接口的具体实现类的选择控制权从调用类移除，移交给第三方裁决，降低了类与类之间的以来关系，促进了松耦合。控制反转将程序之间的关系完全交由容器控制，Java的垃圾回收机制使用户不用考虑对象的销毁，IoC使用户不用关心对象的创建和销毁，想要更换一个对象时只需修改极少的内容，实现了对象的热插拔。

Spring的另一个重要特性是AOP（面向切面编程），它是Aspect-Oriented Programming的简称，AOP是一种设计思想，任何符合AOP思想的技术实现都可以被称作AOP。AOP是面向对象编程的一种补充，面向对象编程在类似事务处理、日志管理、异常管理等方面的会产生大量不利维护且复用效果差的代码。AOP把软件分为核心关注点和横切关注点两个部分，将技术的实现代码和业务进行分离，能够有效降低代码的耦合性，利于维护和复用。

DAO模块将数据库处理的逻辑代码和数据库连接相分离，使得数据访问层的结构清晰，耦合度低。

此外Spring还提供了基于Junit4的Spring测试框架，通过@RunWith注解执行Spring的测试运行器。

SpringMVC是Spring框架的后续产品，是一种基于MVC三层模式的web框架。SpringMVC使用注解通过简单的POJO类作为控制器进行请求的处理，下图为SpringMVC的框架模型。

SpringMVC围绕DispatcherServlet展开，他负责将请求发送到和是的处理程序，使用视图返回最终的结果。它通过@Controller将普通POJO类作为请求转发和处理的Action类，在方法上使用@RequestMapping注解实现类似于REST的URL请求。这种通过注解的方式实现类似于REST的URL请求的功能是SpringMVC区别与其他框架的优势之一，也是本文摈弃Struts2而选用SpringMVC的原因之一。SpringMVC使用注解的方式实现方法的注入，不需要手动去创建对象，使用注解的方式实现数据的转发，尽量减少了xml配置文件的使用。SpringMVC集成了所有Spring的优点，将企业级开发变得更加简洁。

### 2.2.2 Mybatis框架

MyBatis是apache的一个开源项目，其前身是iBATIS，是一个数据访问层框架。MyBatis支持定制化SQL语句、存储过程以及高级映射，可使用注解或xml配置文件进行配置，将接口和POJO类映射成数据库中的记录。MyBatis根据配置文件创建SqlSessionFactory，SqlSessionFactory根据注解或者配置文件中的配置来获取SqlSession，根据其中包含的sql执行所需要的方法，然后进行数据的sql操作等，最后关闭SqlSession。

MyBatis架构图如下：

MyBatis框架可分为三层：接口层、数据处理层、基础支撑层。

接口层提供和数据库的交互方式，MyBatis可使用传统的MyBatis提供的API或者使用Mapper接口两种方式和数据库进行交互。MyBatis提供了非常简单且方便的API，能够提供给用户实现增删改查的数据库操作，使用Mapper接口这种调用方式满足了面向接口编程的需要。

数据处理层是MyBatis的核心，通过传入参数构建动态SQL语句，MyBatis通过传入的参数值进行动态的构建SQL语句，使得MyBatis有很抢的拓展性和灵活性。SQL语句生成之后，MyBatis执行SQL语句，并肩返回的结果集转换为List<E>列表。

框架支撑层负责事务的管理、连接池的管理、缓存以及SQL语句的配置。

MyBatis和Spring框架可以很好的整合，通过Spring自动扫描装配dao，使用SQL构建器动态构造SQL语句。

### 2.2.3 Mybatis与ibernate框架比较

Hibernate也是当前较为流行的开源ORM框架之一，提供了对JDBC的封装。Hibernate框架可以使用在任何需要使用JDBC的场景，能够通过配置文件或注解实现持久化类和数据库之间的映射。Hibernate框架对JDBC操作进行封装，使开发人员从负责的数据库编码中解放出来，更加注重业务逻辑的编写，进而提高代码的开发效率。

MyBatis和Hibernate都是优秀的关系型映射框架，都对数据库的操作做了更加简化的封装，将数据库操作变得简单起来。而本文将选用MyBatis框架作为数据操作层的框架，主要是根据MyBatis和Hibernate这两大框架的适用类型和成本进行分析，主要分为以下几点：

* 1. MyBatis相对于Hibernate的全自动对象关系型映射，采用半自动的映射。MyBatis的SQL可根据需求进行定制，可通过配置文件手动编写SQL或使用注解@SelectProvider在普通POJO类中进行SQL的动态构建。MyBatis的这种定制化的SQL构建方法可以按照需求操作指定的字段，同时使用原生的SQL语句进行操作数据库较之Hibernate的HQL语言性能上更加优化，查询速度更快。MyBatis的自动构建的SQL语句灵活性和可控性更好。
  2. Hibernate对原生的JDBC进行了封装，而MyBatis是基于原生的JDBC，运行速度上MyBatis更有优势。
  3. MyBatis框架较之Hibernate框架入门简单，更易学习，学习成本也相对较低，同时开发速度更快。

虽然Hibernate框架也有很多的功能和特性的优点，但开发一项项目要根据实际情况和项目的实际适用性和成本进行选择，所以本文选用了能够动态定制SQL语句的MyBatis框架进行开发，将其整合Spring、SpringMVC进行整个系统的开发框架。

## 2.3 Android介绍

Android是有Andy Rubin开发的一款可用于手机、平板电脑、电视、手表等设备的一款基于Linux的操作系统，后被谷歌收购。自Android问世以来，以一种迅猛的势头发展，2011年，Android首次超越了当时占比最高的Symbian系统，2013年，全国使用Android系统的设备已有10亿台之多。Android系统不仅提供了丰富的系统控件和SQLite这一自带的数据库，还提供了系统自带的日志工具类Log，不需任何配置就可以在代码中使用Log.\*\*()进行日志的打印。

首先介绍Android的体系结构：

从图中可以看出，Android的系统架构分为四层，从下到上分为Linux内核层、系统运行层、应用框架层和应用层。Linux内核层为Android设备提供了底层驱动，如USB Driver、WIFI Driver等；系统运行层提供了Android开发过程中的主要特性支持，如该层中提供了Android的内置数据库SQLite、Android的2D图形引擎SGL等，同时该层还提供一些核心的运行时的库，这允许用户使用Java语言进行开发；应用框架层提供大量的API，比如开发时常用的Activity Manager、Content Providers等；应用层提供核心应用程序，我们所有开发的应用都要安装在这一层，如系统自带的联系人、日历，还有我们自己开发的应用等。

Android开发有四大组件，分别是活动（Activity）、服务（Service）、广播接收器（Broadcast Receiver）和内容提供器（[Content Provider](http://baike.baidu.com/item/Content%20Provider)）。

Activity是Android开发的基础，所有用户界面和组件都运行在活动之中，它提供一个窗口，提供给用户进行和应用进行交互，Activity的效果类似于web应用中的网页，一个Android应用由多个Activity组成。一个Activity的生命周期中有四个状态，其中Activity类中有7个回调方法，覆盖了其生命周期的每一个环节。

图为Activity的生命周期：

Service是安卓四大组件中的一个重要的内容，主要用于实现不需要和用户交互的长期运行的工作，一般为后台默默的执行者，它不需要任何交互界面，可以让服务一直后台运行，以满足用户需求。只有当用户进程被Kill时，相应的服务才会中止运行。Service可以应用在多个场景，如后台运行的音乐播放器，后台地图导航等都是使用了Service的特性。开启Service有两种方式，一种是startService()，另一种是bindService()。

Broadcast Receiver应用在程序之间，可在各个程序中间和程序之间传播信息，可以使用广播接收器这一组件实现各个程序之间信息的交互。Android可以在程序之间实现广播的发送和接收。广播接收器也未提供任何可供交互的用户界面，用户不能对其进行显式的操作。广播接收器是一个全局的接收器，能够对系统全局的广播消息进行监听。应用程序发出广播的Intent后，任何匹配该Intent的广播接收器都可以被启动，这也是能在程序间传播消息的重要原因。

[Content Provider](http://baike.baidu.com/item/Content%20Provider" \t "_blank)可以实现程序间的共享，主要用于对外共享数据。Content Provider一般提供一个接口供用户存储或使用数据，它对底层进行了抽象和封装个，即使在开发过程中更换数据库，Content Provider仍然不会影响上层的使用。

# 需求分析与系统设计

基于SaaS模式的自动售货机云平台系统分为后台云管理系统和终端系统两大模块。后台管理系统分为自动售货机厂商和运营商模块，终端系统分为操作员终端和自动售货机终端两大模块。本章将从这两大模块，四个部分进行需求分析和总体设计。

## 3.1 课题研究目标

基于SaaS模式的自动售货机云平台主要研究目标是运用SSM框架和SaaS模式的思想，设计和实现一套能够供多租户（多个自动售货机运营商或厂商）通过直接访问互联网的方式就能够使用的管理系统。

系统后台管理系统分为两大模块，分别是自动售货机厂商和自动售货机运营商，自动售货机厂商为售货机生产者，售货机厂家可将自己生产的售货机进行增删改查的基础管理，也可将售货机卖给合作的售货机运营商。售货机中还包含用户管理、运营商管理、售货机类型管理等。售货机运营商为自动售货机的购买者或者租户，运营商主要对售货机进行使用，通过售货机售卖商品和进行其他销售性活动。运营商主要有用户管理、售货机管理、库存管理、商品管理、财务管理等模块。

系统的Android终端分为两大模块，分别是操作员（售货机上货员工）客户端和自动售货机终端。操作员终端主要进行售货机的查询和售货机货道商品的更新操作，操作员客户端通过和后台服务器上的信息进行交互获取要查询的售货机的货道状况。旨在为操作员提供一个可随时随地都可以进行售货机查询和货道更新的Android客户端，免去了人工记录的麻烦。售货机终端主要是完成自动售货机的售货工作，可以为用户提供支付宝、微信以及银联二维码的支付方式进行支付。

## 3.2 用例分析

用例图是用来描述系统某个功能单元的一种UML模型描述图，用于描述基本流程之间的“使用者”关系，进而可视化的表示系统的需求。本节将从系统管理员、运营商及厂商等几个方面描述系统的参与者及用例之间动态关系。

### 3.2.1 SaaS云平台用例分析

SaaS云平台管理系统根据用户种类可以分为系统管理员、运营商和厂商三种，各个种类的用户中有包含各自的角色和权限。系统管理员为该SaaS系统的提供商，具有最高权限，可以管理厂商和运营商；运营商是SaaS系统的租用者，可以对其内部进行定制和管理；厂商也是SaaS系统的租用者，可以对售货机等信息进行管理。

#### 3.2.1.1 系统管理员

系统管理员是该平台的提供商，拥有系统的最高权限。基于SaaS的售货机云平台是由第三方提供商提供，用户通过互联网根据需求进行租用。系统管理员为系统的一级用户，需要对系统的租用规则进行定义和管理，对处于第二级的商户进行管理。如图，为第一级系统管理员的用例图。

系统管理员为本系统提供商具有的用户属性，系统管理员主要负责角色信息管理、权限管理、用户管理、租金定制、租户订单管理、商家管理等业务。系统管理员是系统开发时就需要进行初始化的用户，包含账户名和密码，以及角色所属类别。系统管理员角色管理包含了权限管理这一业务，每个权限代码代表拥有该权限的用户的活动范围。角色由权限构成，一个角色可以包含一到多个权限，角色和权限的关系为一对多，系统管理员可对用户角色进行管理。系统管理员还需对该SaaS平台的租用规则进行定制，租用规则由租用时限和租用售货机台数决定，按照时限可以分为15天、1个月、3个月和1年等几个时间单位，软件租用后系统会允许租户免费试用30天。商家进行定制购买后，会在后台生成一条租户订单信息，系统管理员可对租户订单信息进行管理。自动售货机厂商和运营商都通过该平台进行软件租用，其属于系统管理员的下级用户，须遵守规约，受其管制。

#### 3.2.1.2 运营商用例分析

售货机运营商是系统的主要用户之一，首先用户进入平台主页，平台主页下方提供购买软件入口，用户进入软件的租用页面，租用时会根据用户提供的密码初始化该运营商的管理员账户和密码。软件租用成功后运营商管理员可添加自己的下级管理员并分配给不同的用户权限，不同的用户权责分明，共同管理租户系统。

如图，为运营商商家的用例图。

售货机运营商是该SaaS平台的用户种类之一，主要进行售货机售货过程中的各个业务的管理。从图中可以看出，一个运营商内的用户种类分为五种，分别为：售货机运营商、用户组管理员、操作员、库存管理员和财务管理员。售货机运营商管理员为运营商商家的一级用户，可以管理系统的所有操作，用户管理员、操作员、库存管理员和财务管理员为运营商的二级用户，每个类型的用户有其对应的权限，用户需根据自己的权限管理系统。

售货机运营商管理员主要有以下几种功能：SaaS平台租用、缴纳租金、角色管理、用户管理和售货机管理等。售货机运营商首先通过页面进行软件系统的租用，根据不同的时限支付相应金额的租金，之后运营商管理员根据初始化的账户名和密码登录进软件系统，不同的运营商之间相互隔离。运营商管理员可以查看系统支持的用户角色，以及每个角色所包含的用户权限，当管理员添加下级用户时，可根据需求为用户分配用户角色。为方便管理，管理员可创建用户组，为每个用户组分配用户组管理员，每个用户组下包含多个普通操作员，负责自动售货机的管理。管理员可进行售货机的查询、添加和删除，可按售货机地址将售货机进行分组，同时还可将具有相同商品和容量的货道进行分组，进而统一管理，减少工作量。

运营商操作员主要有以下几种功能：售货机管理、售货机货道管理、售货机上货、上缴营业额等。运营商操作员是售货机运营的重要用户角色之一，它承担着售货机各个货道的上货、更新和管理等操作。运营商操作员可登陆该平台系统查看所管理的售货机列表并查看售货机详细信息，点击某售货机可查看售货机中货道的列表，每个货道列表列出了当前货架上的商品、商品数量和额定存货量等货道信息。除此种查看方式外，该平台还提供了单独的售货机货道信息管理页面，操作员在查询条件中输入售货机信息，点击查询按钮便可列出该售货机的所有货道信息。运营商操作员上货时可进行上货量的提交，这将在下文操作员客户端的需求分析中进行描述。

用户组管理员是操作员的上级用户，它拥有操作员的所有权限。用户组管理员是为方便将众多管理员分组管理而设置，每个用户组管理员可以管理一定数量的用户，可以进行组员的查询、添加和移除等操作。

库存管理员的职责主要分为两块时：一是商品管理、二是库存管理。库存管理员进入商品管理页面可对运营商商家的商品进行商品添加、商品更新、商品删除和商品查询等工作。库存管理分为以下几个主要菜单选项：出库信息查询，主要对商品的出库信息进行查询和搜索；仓库出货，主要为操作员提供货物支持和出货记录工作；商品入库，当商品库存发生变化时需要管理员进行实时更新；清单打印，由系统根据商品的出库情况统计出一定时段的商品出库信息表，管理员可对该页面的出库信息进行打印。

财务管理员主要对运营商的营业额进行管理，用于查询订单记录以及对营业额进行统计和清算。

#### 3.2.1.3 厂商用例图分析

该SaaS云平台不仅为售货机运营商提供了使用接口，还未售货机厂商提供了软件使用的服务。售货机厂商是该平台的用户之一，厂商通过平台的购买页面对软件进行租用，之后独立进行厂商内部的运营和管理。与自动售货机运营商一样，售货机厂商也处于该SaaS云平台的第二级，是系统管理员的下级用户。各个厂商之间相互隔离，分别进行软件的使用和系统的管理。

如图，为厂商商家用例图。

售货机厂商是售货机的提供者，和运营商一样，售货机厂商需要通过首页的租用入口进行软件平台使用权的租用，软件租用成功后，厂商管理员可在添加其他管理员一同进行系统的管理。

售货机厂商因业务复杂度低，所以其用户角色复杂度较低，一般分为厂商管理员和财务管理员两种。厂商管理员拥有厂商商家内部系统的最高权限，是厂商的一级用户，可进行用户的角色查询、用户管理。厂商管理员可根据需求添加用户，并为用户分配相应的用户角色，不同的角色负责相应的权限范围。首先，每个厂商都有其合作的运营商，管理员进入运营商管理菜单可查看当前合作的运营商，点击添加运营商按钮可以添加新的合作运营商。然后，管理员需对售货机贩卖类型进行定义，售货机贩卖类型一般分为饮料机、食品机等类型，管理员可对贩卖类型进行增删改查。最后，管理员需要进行售货机的信息维护，可对售货机进行添加、修改、删除和查询，添加售货机时还需对其货道进行定义，将其与自动售货机真是货道数量相匹配，维护售货机售价及租用价格。厂商管理员还设有二级用户财务管理员，可对售货机售出或租用订单进行统计和查询。

### 3.2.2 Android终端用例分析

除自动售货机SaaS云平台外，与之配合使用的还有自动售货机的Android终端系统。终端系统分为两类，一类是移动终端，供运营商操作员使用，操作员可随时随地进行终端的的登录和使用；另一类是售货机Android终端，供消费者使用，消费者可通过售货机终端进行商品的购买操作。

#### 3.2.2.1 操作员客户端

操作员客户端供运营商操作员及其运营商系统内的上级管理员使用，操作员通过用户名和密码进行登录，系统验证成功后进入个人页面。个人页面有五个选项菜单，分别是机器列表、更新货道、查看个人信息、查看库存、退出登录和检测更新等。

如图，为运营商操作员客户端用例图。

运营商操作员终端主要服务对象为操作员，当操作员对某售货机货道进行上货后，可以查询当前售货机，并定位到上货的货道，根据具体的货道信息、上货商品编号、上货商品数量进行货道的更新，更新货道后该操作员的个人库存也相应进行更新。此外，用户可通过检测更新选项对当前终端系统版本进行下载更新。

#### 3.2.2.2 售货机终端

自动售货机终端提供了一种基于Android平板开发的新型终端系统，该终端系统作为自动售货机运营的技术支持，为自动售货机的正常运行而服务。终端系统为消费者提供选货的交互界面，消费者在页面中进行选货，选货成功后生成支付的订单，消费者进行订单的支付，支付成功后，系统通知售货机出货。消费者除选择购物外，还可进行商品的赠送。在商品支付页面可选则购买或者赠送，若为商品赠送则生成一个赠品码，凭借赠品码可在任意一台售货机上选取同等价格的商品。消费者使用自动售货机终端完成一次购买的行为，终端的优良的人机交互、易操作性以及方便性是给消费者带来良好体验的几大因素。

如图，为自动售货机终端消费者用例图：

## 3.3 需求分析

### 3.3.1 SaaS云平台需求分析

SaaS云平台有三部分组成，分别是系统管理员、自动售货机运营商管理平台和自动售货机厂商平台。SaaS云平台提供后台管理系统，可供多个商家使用，用户可通过网页访问的方式使用系统。系统管理员主要进行系统权限管理、用户管理、租赁管理、定价管理等。运营商端管理系统主要进行售货机日常销售以及维护售货机销售和后续活动的管理，包括用户管理、售货机管理、货道管理、商品管理、订单管理等，各个管理员角色分别拥有不同的权限，负责其相应的管理区域。运营商端管理系统提供自动售货机终端需要的货道信息和商品信息，售货机根据后端信息进行商品交易，并将交易记录存回后台管理系统。厂商端管理系统主要进行售货机的销售及管理，售货机信息管理、货道管理、合作商家管理等。

SaaS平台的用户分为系统管理员、运营商管理员、厂商管理员几种，为更好的描述各个管理员之间的关系，下图给出用户的组织结构图：

#### 3.3.1.1 系统管理员需求分析

* + 用户角色管理：

用户角色包括用户的权限管理和用户角色管理，不同的权限对应不同的操作范围，每个用户角色可以包含多个用户权限。用户权限包括权限名称、权限编号、权限描述、权限所属类型，权限所属类型分为三种：系统管理员、运营商、厂商，在数据库中分别用0、1、2进行标识。系统管理员可进行权限的查询，对用户角色进行增删改查操作，添加、修改、删除用户角色中包含的权限。

* + 租金信息管理：

系统管理员需对SaaS系统的租用规则进行定制，管理员定制规则如下：若用户第一次使用系统，可免费使用30天，30天之内可以使用系统所有功能，用于给用户一个租用前的免费体验期；30天试用期结束后，将在商家的数据库表中做标记，标识已使用过试用机会，不能再次免费使用；管理员制定规则，给出每月的租用费用，若租用期达到一年，则将给出85折优惠，达到两年给出75折优惠，达到三年给出65折优惠，用户在系统页面中可成倍购买租用时间。租用成功后系统将租用订单存储到数据库中，系统管理员可进行查看。若商户租用期限到达，将给出租用到期的提示。

* + 商家管理：

商家是系统管理员所对应的“用户”，商家通过注册页面选择合适的租用期限和租用台数，进行商户的注册和租金缴纳。系统管理员可查看已注册租用的商户。

#### 3.3.1.2 运营商端管理系统需求分析

运营商通过商家租用界面进行系统的租用和商户的创建进行使用该系统，主要功能如下：

* + 平台租用：

运营商初次使用系统，首先进行系统的租赁和租金的缴纳，进行该操作时会给商家分配一个初始的管理员账号和密码，租赁成功后跳转登录界面进行登录。第一次使用系统可免费试用30天，商家信息表中记录租用的开始时间和结束时间，租用到期后需续费才能再次使用。

* + 用户登录/注册：

用户通过浏览器进入登录入口，输入公司编号、用户名和密码进行登录，三项缺一不可。系统根据用户输入的信息进行匹配，验证商家租用时间是否到期，验证通过后，进入其对应的商家及权限范围内的页面。新的用户由管理员进行添加注册，并提供初始密码。

* + 用户管理：

用户管理分为用户信息管理、用户组管理、角色管理三项。系统管理员和用户组管理员可添加新的用户、查看组内用户的用户信息、修改用户信息、修改用户角色等。管理员可创建用户组，为每个用户组设置一个组管理员，由组管理员对组内成员进行管理。此外，普通操作员不具有用户管理的权限。管理员可查看平台提供的角色信息，以及每个角色所包含的用户权限，因自动售货机运营商所需管理员角色相对固定，所以系统管理员对用户角色进行管理，并为运营商提供固定的角色分类供管理员选择。

* + 售货机管理：

售货机管理分为售货机管理、分组管理、货道组管理、货道管理四个部分。

管理员可对售货机进行添加、查询、更新、删除、分配操作员等操作，管理员可通过输入售货机铭牌号、设备主板、是否分配、售货机类型等信息查询匹配的售货机列表。管理员点击添加售货机按钮后弹出添加信息的模态框，填写售货机信息进行添加，并添加售货机对应的货道；点击详情按钮，查看售货机详细信息及货道列表，点击更新按钮，更新售货机信息，点击分配按钮，为售货机分配操作员。

管理员可对售货机组进行添加、修改、更新、分配和删除操作。点击添加售货机组按钮，弹出模态框，管理员填写售货机组信息后进行提交，添加一个新的分组；点击详情按钮进行售货机组信息查看；点击更新信息按钮可对售货机组信息进行更新；点击分配按钮，可将售货机内所有售货机分配给选定的操作员，若组内售货机已被分配，则只分配未分配的售货机；点击删除按钮，则删除该售货机组。

管理员添加售货机的同时也完善售货机内的货道信息，管理员可在货道管理界面进行售货机货道的查询，页面提供了售货机铭牌号、货道编号、商品编号等查询条件。

每个货道对应着一个售货机，货道信息存储了该货道的容货量、当前库存量、销售的商品等信息。对于销售相同商品的货道可以将其添加至一个货道组，方便了管理员进行批量操作。

* + 商品管理：

商品管理员进入商品管理页面，可对商品进行添加、修改、删除和查询。管理员点击添加商品按钮，弹出添加商品模态框，输入商品详细信息后点击保存即可；点击商品详情按钮，可产看商品的详细信息；点击更新商品按钮，进行商品信息的更新；点击删除商品，进行商品的删除。

* + 库存管理：

库存管理包括出库信息查询、仓库出货、商品调拨、清单打印等四个功能模块。操作员填写调拨单，请求仓库管理员为自己分配商品；仓库管理员查看调拨单，在出库页面中填写营业员编号、所选商品、出库数量、备注信息等进行商品的出库；仓库管理员进入出库信息查询页面查看出库信息。

* + 订单/财务管理：

自动售货机终端工作过程中，每完成一笔订单就向后台发送一个post请求，将订单信息保存在数据库中，管理员可通过页面查看自动售货机订单信息。售货机完成交易信息后会向后台发送一条订单信息，订单信息中包含交易金额，财务管理员可统计每月的交易金额。

#### 3.3.1.3 厂商端管理系统需求分析

* 平台租用：

厂商初次使用系统需要对平台进行租用，商家交纳租金，并完成注册，在商家表中初始化起始时间，第一次使用系统同样可以获得30天的免费试用时间，租用日期到期后进行续费可继续使用。

* 用户登录/注册：

厂商租用平台成功后可进入登录入口进行系统的登录，系统会根据输入的商家编号以及用户名和密码判断当前登录的商家类型，使其进入相应的菜单页面。厂商管理员可以新增用户、修改用户信息、查询用户信息、删除用户等。

* 用户管理：

厂商管理员可添加、更新、查询、删除用户，管理员单击添加管理员按钮，弹出用户信息模态框，管理员填写新的用户信息，点击保存用以新增用户；点击详情按钮可查看用户详细信息；点击更新按钮，可进行用户的信息更新。

* 运营商管理：

每个售货机厂商都有一系列的合作运营商，点击添加商家按钮，弹出一系列的商家列表，管理员可点击列表前的复选框，选择要添加的商家。除此之外，厂商管理员还可查看和删除相应的商家，表示不再将该商家的信息显示在合作商家列表中。

* 类型管理：

类型管理页面主要对售货机的售货类型进行管理，每一台售货机生产时就已确定该售货机的特性，以及该售货机可以售卖的商品类型。管理员进入类型管理页面可对售货机类型进行添加、修改、查询和删除等操作。售货机类型独立进行定义，当对售货机进行管理时，可将售货机与相应类型进行关联。

* 售货机管理：

管理员对售货机信息进行管理，可对售货机进行添加、修改、删除、查询、售出等操作。管理员点击添加售货机按钮，弹出添加售货机模态框，填写售货机详细信息后点击保存进行提交，售货机详细信息包含售货机类型，页面将给出一个下拉列表，进行售货机类型的选择；售货机添加成功，点击详细信息按钮查看售货机详细信息，同时添加包括售货机货道编号、货道容量等内容的货道信息；点击更新按钮，进行售货机信息的更新；点击分配按钮，弹出运营商列表，选择某运营商将已完成交易的售货机信息分配给相应的商家；点击删除按钮进行售货机的删除。

* 订单管理：

每台售货机分配给相应的运营商时，将在后台发送一个post请求，将售货机销售信息进行存储，管理员可进入订单管理页面查看售货机销售订单。

### 3.3.2 Android终端需求分析

Android终端主要包含两个模块，分别是运营商操作员客户端和自动售货机售货客户端，它们的后台服务都在SaaS云平台中，每一条操作都和后台系统进行连接。

#### 3.3.2.1 操作员客户端需求分析

操作员客户端是为自动售货机运营商开发的一套移动端应用，主要为方便操作员上货时进行货道更新和货物的统计。

* 用户登录：

操作员客户端允许登录的用户类型有运营商管理员、组管理员和运营商操作员三种，不具有操作员角色的用户不能登录该系统。用户输入商家编号、用户名和密码，客户端将信息发送到服务端，服务端进行用户身份验证后跳转至客户端主页面。

* 售货机查询：

用户点击机器列表按钮，进入售货机列表页面；点击售货机可查看售货机详细信息；点击售货机货道信息可查看售货机货道列表，查看当前货道库存量、销售商品、销售价格等信息。

* 货道管理：

用户进入货道管理页面，用户可填写页面上的货道更新表单，填写售货机Id、货道编号、加货量等信息，安卓客户端向后台服务端发送一条post请求，将对应售货机的对应货道的加货量和现存量进行更新。

* 版本更新：

操作员客户端首页提供版本更新的按钮，每当新版本完成就在文件服务器上存放新的apk包和存有版本信息的配置文件。用户点击更新按钮，系统将自动检测是否存在新版本的apk文件，若已是最新版本则不进行更新，否则，下载新的版本文件并进行更新。

#### 3.3.2.2 自动售货机终端需求分析

自动售货机终端为消费者提供购买的页面和购买服务。

* 用户选货：

用户进入选货页面，页面提供了商品的图片、价格以及是否有货，用户点击商品进行选货。

* 商品支付：

用户选货成功后，点击确认，会向第三方支付服务器发送商品支付信息，包括时间、价格、货道、交易号等内容，后台处理成功后将支付的路径，用户可通过银联、支付宝和微信等进行支付。

* 售货机出货：

商品支付成功后，系统通知自动售货机被选商品的货道进行出货。

* 赠品服务：

若支付是选择是赠品服务，则不通知售货机出货，生成一个赠品码。获取赠品码后，用户在任意一台售货机输入赠品码，即可获取同等价位的商品。

* 更新信息：

每次交易成功后，都需向后台发送一条交易记录，将交易记录插入售货机销售的订单表中，同时记录支付的订单号和支付类型以便于统计。对当前售货机的货道余量进行更新。

## 3.4 系统功能结构图

基于SaaS的自动售货机云平台分为三个模块：系统管理员模块、运营商模块、厂商模块，其中系统管理员模块为系统的整体设计和基础模块，主要进行初始化、预定义和管理的功能。运营商模块为系统的主要模块之一，运营商租用后可进行使用；厂商模块是售货机提供商所使用。根据其运营商和厂商这两种租户类型的不同进行系统的设计，下图为该平台的功能结构图：

## 3.5 数据库设计

详细的需求分析之后需要对系统的数据库进行设计，数据库设计的好坏关系到平台最终的实现的好坏，是软件开发的基础。该平台的数据库将从以下几个方面进行分析和设计，首先进行SaaS平台多租户模式的分析和设计，接着进行数据库的概念结构设计，然后进行数据库的逻辑设计和数据库表的设计。

### 3.5.1 数据库多租户设计模式的实现

本文的SaaS云平台在实现多个商家共享一套系统的同时将各商家间的业务进行隔离，主要得益于多租户技术的软件架构技术。多租户和经常说到的多用户概念有所不同，多用户是在一套系统上可以建立多个用户，供用户使用系统；多租户是许多组织或商家使用一套平台系统，每个组织或商家（即租户）有其独立的业务空间，在各自的空间内可以创建多个用户，租户内的用户仅可访问该组织或商家的数据。

多租户系统的实现方式有多种，在选择使用哪种方式之前，要先从成本、安全性、隔离性、操作难易度和等多个方面进行考虑。多租户数据层的实现方式有以下几种：

首先是最高级别的独立数据库，从本质上分析，独立数据库为每个租户创建一个数据库，数据的隔离性和安全性极好，但是成本却极高。独立数据库与传统的软件系统的数据库设计方式几乎一样，仅仅将软件部署在SaaS服务提供商，这种模式的数据实现方式虽然有极高的安全性和隔离性，但是却大大提高了开发和租用的成本，一般来说，SaaS云平台服务对象大多为中小型企业，这种高昂的成本的数据库设计方式显然被弃用。

其次是共享数据库、隔离数据架构，多个租户共享同一个数据库，每个租户独有一个Schema。这种模式的数据库设计使每个数据库可以支持更多的租户，但是若要跨租户统计数据则存在一定的困难。

最后是共享数据库、共享数据架构的模式，所有租户使用同一个数据库、同一个Schema，在表中增加相应的TenantID进行数据的隔离。从本质上说，这是一种安全性、隔离性最低，共享性最高的方案，但是却有最低的开发成本和最简便的操作方式。当系统需要支持的租户越来越多时，共享数据库低成本、高效的特性就显现出来了。

综合上述三种多租户数据的设计模式，结合自动售货机行业的特性进行分析：首先、动售货机行业的业务相对较为统一，需要独立定制化的内容并不多；其次、自动售货机行业一般更注重业务的发展和商品的销量，而软件系统仅作为一个辅助手段；最后、软件的开发希望最大限度的减少成本。通过各个方面的分析，该系统选用第三种多租户模式——共享数据库、共享数据架构模式。

### 3.5.2 数据库概念结构设计

确定了系统使用的数据库多租户架构，以及前面对系统的需求和功能的分析，提出了一个描述数据实体关系的数据模型，即E-R图（Entity-Relationship Diagram）。顾名思义，E-R图就是描述实体之间的关系，以及实体的属性。E-R图实体之间的关系分为1对1、1对多、多对对三种，通过这三种方式能够图形化的描述各个属性之间的关系。

图中描述了自动售货机 SaaS平台的E-R图，并给出了实体的部分属性和实体与实体之间的关系。

### 3.5.3 数据库逻辑结构设计

上文中E-R中给出了系统管理员、运营商管理员、厂商管理员、租金定价、与用户角色、租户、售货机、订单等多个实体。通过其对数据库的概念设计，我们分析各个实体以及其间关系可以得出数据库的逻辑设计列表。

将E-R图装换为关系模型，确定关系模式的属性和码，将实体的属性和实体之间的关系转换为关系模式，SaaS平台的逻辑结构设计如下。

* + 商家（商家Id，商家编号，商家名，商家类型，商家描述，商家状态，开始租用时间，租用到期时间，租用台数，是否试用过，操作人，操作时间）
  + 平台租金（定价名，对应价格，折扣量）
  + 售货机租金（租用类型，每台价格）
  + 租户订单（订单Id，租户Id，租户类型，租金总价，租用售货机台数，开始租用时间，结束租用时间）
  + 用户权限（权限Id，权限编号，权限名称，权限描述，权限所属类型，操作者，操作时间）
  + 用户角色（角色Id，角色名称，角色所属类型，操作者，操作时间）
  + 角色权限表（角色权限Id，角色Id，权限Id）
  + 用户（用户Id，用户编号，用户名，密码，手机号码，email，是否为小组管理员，用户组Id，可用状态，用户所属商家Id，操作者，操作时间）
  + 用户角色表（用户角色Id，用户Id，角色Id，操作者，操作时间）
  + 分组（分组组Id，组名，分组类型，分组描述，所属商家Id，操作者，操作时间）
  + 售货机类型（类型Id，类型名称，所属商家Id，操作者，操作时间）
  + 厂商端售货机（售货机Id，售货机名，售货机主板号，厂商Id，售货机价格，售货机类型，是否售出，售出运营商，操作者，操作时间）
  + 售货机销售表（出售Id，厂商Id，运营商Id，出售时间）
  + 运营商（运营商管理Id，厂商Id，运营商Id）
  + 运营商端售货机（售货机Id，售货机名称，售货机主板号，售货机类型股，是否分配，用户Id，售货机地址，售货机组Id，运营商Id，操作者，操作时间）
  + 货道（货道Id，货道编号，额定存货量，当前存货量，新增存货量，货道组Id，售货机Id，商家Id，操作者，操作时间）
  + 货道历史记录表（历史Id，售货机名称，货道编号，商品名称，运营商Id，新增库存，操作者，操作时间）
  + 货道组（货道组Id，货道组名，商家Id，商品Id，商品价格，是否折扣，操作人，操作时间）
  + 货道商品表（货道Id，商品Id，商品售卖价格，是否特价，售货机Id）
  + 商品（商品Id，商品名，商品编号，商品规格，商品单位，商品进价，商品售价，商品描述，运营商Id，操作者，操作时间）
  + 用户库存（库存Id，用户Id，商品Id，商品库存，更新时间）
  + 出货（出货Id，商品Id，订单编号，出货数量，营业员Id，出货类型，是否结清，备注，运营商Id，操作员Id，操作时间）
  + 赠品表（赠品编码，商品价格，商品数量，赠品编码，是否提现，商家Id，交易时间）
  + 订单（订单Id，订单号，商品Id，售价，售货机Id，货道Id，商家Id，数量，售出时间，交易结果）
  + 营业额（营业额Id，售货机Id，金额，类型，操作员，运营商Id，上缴时间）

### 3.5.4 数据库维护

除使用数据库自身的加密算法保障数据库库字段信息的安全外，还要注意其他方面的隐患。数据库存放着用户和商家的关键信息，若出现以下任意一种状况都将造成不可挽回的后果。比如：机器的损坏、工作人员不小心删除、电源的故障、人为破坏等。若没有数据库的备份，一旦出现故障将不能再将数据库恢复，给开发和生产生活带来巨大的损失。

因此，对于数据库的维护可以通过以下几点实现：1、敏感数据使用数据库MD5、ENCODE等加密方式进行加密；2、每天对数据库信息进行定时的固定备份，并定期将这些数据备份到光盘、软盘中。备份脚本如下：

@echo off

set "Ymd=%date:~,4%%date:~5,2%%date:~8,2%"

D:\"WorkSoftware"\"MySQL Server 5.1"\bin\mysqldump --opt -u root --password=密码 -h127.0.0.1 vending > D:\testbackupdb\_%Ymd%.sql

@echo on

@pause

3、加强机房安全，确保设备能够安全正常的使用和运行；4、加强内部员工素质培训，确保数据不被泄露和破坏。

# 管理系统的设计与实现

本章介绍了基于SaaS的自动售货机平台的设计与实现以及客户端的设计与实现，管理系统使用SSM（Spring+SpringMVC+Mybatis）框架和Maven进行开发，系统基于MySQL数据库并使用多租户模式进行数据库的开发和存储。系统采用三层架构进行开发，分别从数据层、业务逻辑层、表示层三层进行详细的系统设计。系统代码主要实现自动售货机厂商和运营商平台的管理模块，使用SaaS软件即服务的思想，结合多租户架构进行设计和开发。

## 4.1三层架构

平台系统采用三层架构进行开发，分别是数据层，业务逻辑层和表示层，开发过程中，各层之间进行分离，权责明晰。系统使用Maven进行构建，所有依赖包在pom.xml文件中进行引入。所有的配置信息在resource文件中列出，包括数据库库信息、日志信息、框架的配置信息以及数据库连接池等。

1. 数据库访问层：

数据访问层负责系统和数据库的交互，完成数据的查找、更新、添加和删除等操作，是程序和数据库之间沟通的桥梁。传统数据库访问代码冗杂，维护较为困难，因此这里采用了轻量级的Mybtais框架来实现持久化。在pom.xml文件中引入框架所需的依赖，并将数据库配置信息卸载properties文件中，方便维护。为简化数据库的操作，在配置文件spring-mybatis.xml中启用注解，并进行JNDI数据源和Spring事务管理的配置。配置文件通过完整的文件路径扫描mapper文件的，通过面向接口编程的方式完成数据库的操作。

数据访问层由三个重要部分：POJO类、Dao类、SqlProvider。其中，POJO类是数据库中字段的映射，罗列了数据库中的所有的字段属性以及数据库库中表与表的映射关系。Dao类即MyBatis框架中的Mapper接口，在DAO类中定义访问数据库的接口，接口中提供访问数据库所需的参数，每个接口上使用@SelectProvider注解标注，type指向sql构造的类，method指向sql构造的具体方法。SqlProvider类是Mybatis的动态SQL构造类，它使用接口的方式进行定义。使用动态的SQL构造能够减少xml文件带来的冗余性，减少额外的配置文件，使代码更为简洁。

1. 业务逻辑层：

业务逻辑层是用户和数据访问层之间沟通的桥梁，用于实现系统的业务逻辑。开发过程中主要体现在两个方面，一个是Service，一个是Controller，在配置文件中启动注解，配置要扫描的包路径之后就开始进行业务逻辑的开发。业务逻辑层中Service使用面向接口编程的方式进行开发，Service中方法调用访问层接口实现与数据库的交互，Service将较为负责却又可重用的代码写在一个接口中供Controller调用。Service类中使用@Service注解进行标注。

Controller主要用于和用户进行交互。每个Controller类中使用@Controller注解进行标注，使用@Autowired注解进行Service方法的注入。Controller方法上使用@RequestMapping设置访问的URL路径，使用这种Rest API风格的开发模式也是SpringMVC的特色之一。Controller的返回值类型可以ModelAndView，使用这种返回模式返回一个ModelAndView对象，其中包括了转向的页面和传输的值。在spring-mvc.xml配置文件配置跳转文件的前缀和后缀，可自动在返回的view中拼接前缀和后缀，找到对应位置的jsp。

1. 表示层：

表示层使用Jsp、Ajax、JavaScript、Bootstrap等技术进行开发。使用Jsp进行页面的开发，在Jsp中使用标准标签库用于简化页面代码，使用Ajax进行发送异步请求，使用BootStrap进行页面的设计及风格统一。

## 4.2 租户模块实现

经过上文需求分析及数据库设计之后确定该系统的采用共享数据库，共享数据架构的多租户架构。系统中所有的用户共享同一个数据库，利用租户Id进行数据的隔离。系统初始化时应对系统租金进行定义，分别为试用一个月、一个月、一年、两年、三年进行定价，其中一年、两年和三年分别对应着相应的折扣，按月进行租用不提供折扣，当租用期限达到按年计算可为用户提供相应的折扣。系统管理员可对定价和折扣进行查看和更新。除按年限定义租金外，还需按照售货机台数进行定义，定义每台售货机的价格，每增加一台售货机便将总价增加单台售货机的一倍。

租金计算公式可定义为：

其中SUM为租金总价，P表示每个月的租金金额，ρ表示每增加一台售货机的金额，N表示售货机的数量，T表示租用的月份，Y代表租用的年限，D代表租金的折扣。当租用时间不到一年时，租金总额为P\*T\*N\*ρ;当租用时间大于等于一年时，租金总额为P\*Y\*12\*D\*N\*ρ;使用时租金为0。

若租户租进行续租或临时改变租用条件，可先计算出总租用金额和已支付的未用期限的金额，两者相减即得新增的租用金额。公式如下：

公式（2）为用于计算已支付但未使用的租用期限内的租金金额，若为试用期或者租金到期时间小于当前时间，则剩余租金为0；若租用开始时间到租用截止时间小于12个月，则使用P\*(edTime-now)\*N\*ρ计算剩余租金；若租用开始时间到租用截止时间大于或等于12个月，则使用P\*(edTime-now)\*N\*D\*ρ计算剩余租金，其中增加了一项折扣信息。通过公式（1）和公式（2）可得新的租金计算公式为：

租户进入租用界面，选择租户类型、租用时长以及售货机台数，填写商家名称、商家编号以及初始管理员密码进行用户的创建及租用。租户选择成功后会生成一个订单，用户支付成功后即可通过网页使用过该平台。租金的支付采用银联、微信和支付宝的支付方式进行支付，其实现将在终端的设计中进行详细的介绍。为保证数据库安全，在用户创建的过程对用户密码进行MD5加密存储，MD5加密为不可逆加密，能有效保障用户的账号安全，对用户进行验证时，可将用户输入的密码进行MD5加密后在数据库中查询。对用户隐私信息，如手机号码等进行ENCODE加密存储，ENCODE加密为双向加密方式，加密时使用ENCODE(‘存储信息’,‘加密密文’)对要存储的数据进行加密，可使用DECODE(字段名称,‘加密密文’)行信息的解密。

|  |
| --- |
| 1、Mysql的加密方式，对用户密码和敏感信息进行加密。  INSERT INTO userinfo(userName, password,phone…)VALUES  ('user1', MD5('password'),ENCODE('xxx'，'encode'));  2、使用MD5和ENCODE对数据进行加密后到数据库中进行匹配查找。  SELECT (username,password,DECODE(phone, 'encode') as phone) from userinfo where password= MD5('password');  3、对应代码实现用户信息的加密算法。  if (StringUtils.isNotBlank(userInfo.getPassword())) {  VALUES("password","'"+MD5("+userInfo.getPassword()+"')");  }  if (StringUtils.isNotBlank(userInfo.getPassword())) {  WHERE("a.password=MD5("+userInfo.getPassword()+"')");  } |

租户模块序列图如下：

平台租用使用Ajax发送异步请求，输入商家名称和商家编号时，会向数据库进行查询，若已存在同样名称或编号的商家，则控制器向页面发送提示，在页面上提示该名称或编号已占用。用户选择售货机的租用台数和租用时间，通过Ajax发送异步请求，后台查询平台的租金定制表，根据租金的计算算法公式（1）对当前数量的售货机和租用时限进行计算。其中当租用时限达到以年为单位时，平台将给与不平程度的折扣。后台实现的部分码如下：

|  |
| --- |
| double money = 0;  if (rentTime == 0) {// 免费试用  money = 0;  } else if (rentTime < 12) {// 租用期限为一年内  money = monthMoney \* rentTime \* machineNum \* sNumPrice.getNumPrice();  } else {// 租用期限为年的倍数、找到租金折扣  int discount = 1;  for (SaasPrice saasPrice : saasPrices) {  if (rentTime == 12 && saasPrice.getPriceName().equals("1year")) {  discount = saasPrice.getDiscount();  } else if (rentTime == 24 && saasPrice.getPriceName().equals("2year")){  discount = saasPrice.getDiscount();  } else if (rentTime == 36 && saasPrice.getPriceName().equals("3year")){  discount = saasPrice.getDiscount();  }  }  money = monthMoney \* rentTime \* (discount / 100) \* machineNum \* sNumPrice.getNumPrice();  } |

最终实现效果如图所示：

用户点击提交后，后台控制层生成租用订单，用户进行进而的支付，支付成功后后台将新的租用信息添加到租户订单表中，同时创建新的商家，并为商家分配初始超级管理员。最后，系统跳转到商家登录页面。用户输入商家编号、用户名和密码进行登录，每次执行登录操作之前系统进行过商家可用性判断，若当前时间在商家租用截止时间之后，则判定商家不可用，并将商家可用状态置为0，返回验证失败，提醒用户商家租用期限已到期。

## 4.3 系统权限实现

系统权限管理包括不同租户之间的权限隔离和同一租户内部的多个用户之间的权限。平台系统权限管理分为多个模块，包括权限管理、用户角色管理、用户注册和用户管理等。用户的权限管理和角色管理有系统管理员负责，系统管理员负责定义、管理和分配，再由租户进行内部管理。

系统初始化时由系统管理员创建多个用户权限，并给与每个用户权限的编码和类型，权限编码代表拥有该权限的用户的访问区域，权限类型代表该权限为运营商权限或者厂商权限。权限类型在数据库中用0、1和2进行区分，0代表系统管理员，1代表运营商，2代表厂商；权限编码的设定也有一定的规则，运营商编码为001XX，厂商编码为002XX。

每个用户角色可以拥有多个用户权限，如运营商超级管理员可拥有所有运营商类型的权限，操作员仅拥有操作员的权限。系统初始化初期，当系统管理员对用户权限完成初始化后，可对用户角色进行管理，其中包括用户角色创建、用户角色权限的分配、用户角色删除、修改等操作。用户角色包括角色名称、角色描述等内容，在数据库中，为使用户角色和权限的设置更加独立，将用户角色和权限表单独进行设置，将用户角色和权限的对应关系放置在用户角色权限表中。在用户角色权限表中包含了用户角色Id和对应的用户权限Id，当查找某个用户所拥有的权限时，查找该表中所有用户角色的Id对应的权限Id即可。

当租户创建时，控制层为租户超级管理员分配最高权限的用户角色，租户管理员可查看同类型的用户角色，并在添加新用户时将用户角色分配给用户。

用户权限序列图如下：

## 4.4 厂商模块实现

厂商用户通过输入商家编号、用户名和密码登录系统，系统控制层高通过查询userrole用户角色表判断当前用户的角色以及拥有的权限。登陆成功后将用户信息放在session中@SessionAttributes("user")，在其他需要使用的方法中通过@ModelAttribute("user") UserInfo userInfo将user对象传入。在Jsp页面中配合使用JSTL表达式和EL表达式，可用于编写各种动态页面，简化Jsp页面中的操作，减少在Jsp页面中java代码的使用。

厂商模块传入的Session为SessionAttributes({ "user", "machineTypes", "operMgrs" })，分别为用户、机器类型和厂商运营商表。请求URL的根路径为@RequestMapping("/manu")，在每个方法前都有一个对应的请求映射地址，最终地址都要加上/manu进行拼接。控制层中方法的返回值有多种，可以返回一个封装了model和view信息的ModelAndView对象，或者返回一个String类型的view串，还可对返回结果进行redirect重定向。

厂商时序图如下：

厂商模块包括用户管理、售货机类型管理、售货机管理、运营商管理和订单管理几个模块。厂商超级管理员登录系统，系统后台对公司编号和用户权限进行判定，调用getAllRoleAuthInfos(RoleAuthInfo info)接口获取用户所有权限，将该用户的权限编码放置在session中，在菜单Jsp页面中通过EL表达式和JSTL判断当前用户可见的菜单项。

售货机类型单独存储在一张表中，只需在售货机信息的字段中添加和类型的Id即可将售货机和类型进行关联。管理员添加售货机，每个售货机后需添加对应的商家Id用于隔离，售货机信息被封装成MachineInfo对象之后，使用ajax封装http，使用addMachine(MachineInfo machineInfo)接口进行添加。售货机分配时调用getManuMachineStatus()查看分配状态，一旦处于被分配状态将予以提示不能被重复售出。

## 4.5 运营商模块实现

运营商功能模块包括用户管理、售货机管理、货道管理、商品管理、财务管理、库存管理和订单管理等模块。

* 运营商超级管理员：

运营商超级管理员拥有系统的最高权限，能够对用户进行添加、修改、删除和查找，为用户分配不同的用户角色，用户角色包括用户组管理员、操作员、库存管理员和财务管理员。用户角色由系统管理员进行管理，商家调用assignRoleToUser(Integer userId,Integer[] roleIds)接口进行角色的分配。每个用户可以拥有多个角色，超级管理员拥有全部的用户角色。管理员对自动售货机进行添加、修改、删除、查询，管理售货机货道等，将售货机分配给操作员进行管理。

运营商超级管理员时序图如下：

* 操作员模块：

操作员主要提供给对售货机的管理，用于对售货机的货道进行上货，上货后的货道更新，商品的申请以及营业额的上缴等管理活动。

操作员登录系统后由系统进行用户身份的判定，登陆成功后操作员进入租户的内部平台，根据其拥有的权限，系统将售货机查询、货道管理、库存申请、个人信息查询、更改密码等菜单显示出来。在页面中if (auths.contains("0010101"))作为操作员判定条件，用于确定该要显示的菜单。售货机初始化时调用ajax的addChannelInfo()接口进行货道的添加，此时添加的仅仅为货道的基础信息，当需要将货道和商品关联时调用ajax的addChannelWare()接口将货道和商品的对应信息插入到货道商品表，此时的货道和商品仍然分布在在两张表中，仅仅将货道和商品的对应信息添加到另一张独立的表中用于区分。更新货道信息时，货道上增加的商品数量在个人库存中相应的减少。

售货机终端新增了移动支付，操作员上缴营业额时需要将通过移动支付的订单信息进行封装提交到营业额表中，以便于财务管理员统计。

操作员申请库存首先调用getAllWareInfosByFirm()获取商家所有商品信息，操作员填写商品信息和商品数量提交表单，通过调用addShipToUser()接口进行个人库存的申请。操作员时序图如下：

* 库存管理员：

库存管理员有两项功能模块，分别为商品管理模块和库存管理模块。商品管理模块主要对该商家售卖的商品进行添加、修改、删除和查询等管理操作。商品管理操作和商品库存管理的根请求路径为@RequestMapping("/ware")，使用insertWareInfo()接口将商品信息封装为一个WareInfo对象进行商品的添加。库存管理模块主要对商品的库存进行管理，处理操作员的申请库存请求，为操作员添加库存量。首先库存管理员调用getAllShipments()接口查询数据库中所有未被处理的出库请求记录，获取记录后，操作员可将出库处理标记更新为0，表示进行出货成功。此时，相应的操作员库存表中对应操作员的对应商品数量进行更新。

库存管理模块时序图如下：

* 财务管理员：

财务管理员主要负责订单管理和财务管理两大模块。订单管理模块中，财务管理员可调用查看订单的接口getAllOrders()接口查看所有订单信息。根据操作员提交的营业额数据，财务管理员可对已一定期限内的某台售货机或某个营业员的总营业额进行统计。财务管理员还可查看售货机的销售记录，每条销售记录下都记录了商品、交易金额、售货机Id、货道Id、商家Id，售卖时间等内容。售货机端每销售一件商品都会将销售信息进行封装，发送一条http请求给平台服务端，服务端接收销售信息后将其插入到订单表中。财务管理模块时序图如下：

## 4.6 客户端控制

管理系统除提供一般的后台管理之外还需提供供安卓端调用的后台接口，其根请求url为@RequestMapping(value = "client")。客户端从服务端请求数据，服务端将图片、配置等信息放置在Apache Tomcat搭建的Http文件服务器上。

在tomcat安装目录下创建xml配置文件，用于配置文件服务器使用的文件目录。

<?xml version="1.0" encoding="UTF-8"?>

<Context path="/file" docBase="E://vending" crossContext="true">

</Context>

商品图片和客户端版本信息存储在文件服务器的目录下，当客户端访问服务器端用户图片时，根据数据库中图片的地址访问文件服务器下内容。

# 客户端的设计与实现

本章主要介绍自动售货机平台客户端的设计与实现，客户端分为两种，一种为供运营商使用的操作员客户端，一种为供自动售货机使用的销售终端。客户端使用Android进行开发，后台管理系统为Android终端提供后台服务的支持。

## 5.1 数据传输加密实现

Android客户端和服务器进行数据交互时需要对传输的数据进行加密处理，以保证数据安全。数据加密主要对用户的信息、密码、支付信息等敏感信息进行加密传输。

系统主要采用了了MD5加密算法、RSA加密算法、DES加密算法等算法。MD5加密算法为单向加密，是一个将字符串进行不可逆转换的算法，主要对用户的密码进行加密，因系统不用知道用户的密码，因此将用户密码的MD5值进行存储，查找时通过MD5值进行查询即可。RSA是一种非对称的可逆性加密，在客户端存放公钥，将信息用公钥加密后传回服务端，服务端使用私钥进行解密。DES为对称的加密算法，使用密钥进行加密解密，其运算速度较快。

## 5.2 操作员客户端实现

操作员客户端和服务器之间通过Http协议实现数据的访问和交互，使用Http的get请求获取服务端的数据，使用post请求将封装好的json数据传输至后台，进行数据的更新。服务端提供请求根url为/android的控制接口，请求结束后方法返回一个通过URL编码的String字符串或者String格式的Json串。Android端提供一个用于访问服务端的HttpURLConnection的连接Util类，Android中所有的耗时操作都放置在线程中，防止程序崩溃。首先，用户和终端页面进行交互，点击屏幕触发请求，将请求参数封装成key/value类型的Map对象，并对该请求对象进行进一步封装，将其封装成以&符号连接的请求串。接着，使用outputStream.write(data)向服务器写入数据。最后，获取服务器返回的结果数据。普通的多线程方法无法返回一个实体的对象，FutureTask却可以返回一个对象。使用FutureTask异步执行并和Callable结合使用，然后通过FutureTask的的get方法调用最终的处理结果，并将其返回。

用户进入操作员终端登录页面，输入商家编号、用户名和密码，并将请求信息放在map中，调用HttpURLConnection的postByResponse接口向后台服务器发送post请求，用户身份信息验证通过后，后台服务器方法调用pojo类中的toString方法，向终端返回一个json格式的字符串，使用JsonObject解析后将用户信息存储在终端。

用户登陆成功后使用SharedPreferences将信息存储在本地，用户可使用存储时设定的SharedPreferences的name和mode获取其存储记录，随时查看自己的个人信息。

通过用户当前的用户Id和所属的商家Id，可通过HttpURLConnection发送post请求，查看当前用户管理的售货机及售货机货道。当操作员对售货机进行操作时，只需将加货量、货道编号、商品Id和售货机Id等封装成map对象，将其封装成一个post请求发送给服务器，就能进行货道的更新，同时也进行了个人库存的更新。

用户可以通过操作员终端页面查看个人的当前库存，使用当前Id、所属商家Id，调用HttpURLConnection类的方法向服务器发送请求，服务端返回一个json格式的String对象。这里的库存仅仅用于查看，操作员不具有手动操作的权限，当操作员更新售货机货道时，后台服务器会根据加货量和商品号自动进行用户库存的更新。

Android终端使用PackageManager packageManager= getPackageManager()获取一个PackageManager对象，并调用getPackageInfo使用PackageInfo packInfo = packageManager.getPackageInfo(getPackageName(),0)获取一个PackageInfo对象，接下来通过packInfo.versionName和packInfo.versionCode获取当前已安装版本的版本名称版本号。接下来使用上文提到的HttpURLConnection类发送getByResponse请求，获取文件服务器上放置的终端apk版本的json文件，文件包括版本名、版本号和新版本apk地址等信息，内容如下：

{

"AppName": "operagterApp.apk",

"versionName": "test1.0",

"versionCode": 2,

"url": "http://baseurl/newOperaterApp.apk"

}

获取版本文件后使用JsonObject将获取的内容进行解析并和本地版本信息进行对比，并返回比对结果，版本检测操作使用FutureTask和Callable共同完成。若有新的版本出现，则调用downloadTask方法下载json文件中url指向地址的最新版本apk文件，并进行最新软件的覆盖安装。

终端操作员模块时序图如下：

## 售货机终端实现

### 5.3.1 功能的实现

### 5.3.2 串口通信实现

### 5.3.3 移动支付功能实现

新型自动售货机新增移动支付功能，这里使用二维码移动支付。二维码支付非常适用于消费者和商家之间的小额支付，二维码支付分为“主扫”和“被扫”模式。商家根据选购商品信息生成支付二维码，属于“被扫”方；消费者使用手机APP扫描商户提供的二维码，属于“主扫”方。使用移动支付便利简洁，给消费者更好的使用体验。

银联支付、支付宝支付和微信支付的大体实现过程的时序图如下。 其中消费者为“主扫”方，商户为“被扫”方，第三方交易系统为银联、微信、支付宝的服务系统。

#### 5.3.3.1 银联二维码支付

2016年12月，银联商务推出了二维码支付，其作为银联的一个全新支付方式正式得以推广。银联二维码支付有两种支付模式，一种是主扫模式，一种是被扫模式。主扫模式是由商家发起收款流程，在商家端同步生成支付的二维码，此时的二维码是动态的，每发起一次收款行为就会产生不同的二维码信息，消费者持手机扫描此二维码进行支付。被扫模式是由消费者发起支付行为，消费者端生成支付的二维码，收款端扫描二维码完成收款。这里使用主扫模式进行实现二维码的支付。

一般银联二维码支付放入流程有以下几步。首先，商户向银联申请入网，银联方为商户提供入网后交易所需的证书以及证书的设置方式。接着，根据银联二维码支付交易规范中数据元和申请二维码请求报文对版本号、编码方式、签名方法、交易方式、交易类型等进行配置。根据商户入网的基本信息设置商户号码、终端号、接入类型、支付超时时间等信息。用户进入售货页面进行选货，选货成功后，生成订单，并将订单信息、价格信息、订单发送时间、超时时间和之前设置的数据元一起封装成一个Map对象。并将该Map对象进行签名之后，通过HttpURLConnection向银联全渠道交易系统发送请求，同步返回请求结果。然后，请求结果返回一个包含二维码信息的Json对象，将该Json对象进行解析拼装成一个Map<String,String>对象，对该对象验签成功后，判定respCode字段的值，若为00，则表示二维码流水号qrCode获取成功，并将该二维码字段qrCode返回给安卓终端。若二维码获取失败，则将qrCode的值设置为“Request Error”，并将其返回安卓终端。最后，使用二维码生成类Create2DCode将二维码流水号qrCode信息转换为二维码，并展示在页面中。消费者打开付款客户端扫描二维码，对二维码进行解析，并在支付App上完成付款。

支付二维码动态生成，相应的二维码也动态生成，当超过设定的超时时间后，该二维码失效，不能进行支付；用户扫描二维码进行支付后，该二维码即刻失效，不能进行重复交易。 这种设置失效时间的二维码能防止用户长时间不支付和重复支付的错误状况。

商户发送二维码请求时，银联交易系统需对请求信息进行验签，商户收到应答后，也需对应答消息进行验签，只有双方验签都成功的情况下，才予以进行下面的步骤。商户向银联发送Map数据之前需要对当前信息进行签名，签名所需的证书放置在assets下。首先，Map请求域中含有一个key为signature的字段，将该字段除外的多有其他key/value值用&连接成一个字符串，并按照名称进行排序。然后，对生成的字符串使用SHA-256进行摘要，使用assets中的私钥证书中的私钥对字符串进行签名，签名时使用的算法常量为SHA256withRSA。最后，将签名后的字符串使用Base64进行编码，之后将编码后的字符串放置到signature字段和其他字段一起组合成一个请求体，一起发送给银联后台进行处理。银联后台对请求串进行处理后，返回一个Json对象，将Json对象转换为Map进行签名的验证。签名的验证和签名类似，首先和签名的第一步一样，将签名域用&连接成一个字符串。然后，使用SHA-256算法对该字符串进行摘要，和签名不同的是，这里使用assets下的公钥证书中的公钥对字符串进行验证。签名验证成功后，才允许将请求结果返回给客户端。对返回的对象进行解析，找到qrCode字段的value值，即为所求得的二维码流水号。一条成功的申码返回信息中应该至少包含以下内容，其中respCode和respMsg表示返回结果是否成功，qrCode表示二维码流水号。

|  |
| --- |
| {  "qrCode":"https://qr.95516.com/00010001/62211432508676126225962129622718",  "respMsg":"成功[0000000]",  "txnTime":"20170829095153",  "respCode":"00"  } |

用户扫码支付成功后，可对当前支付结果进行查询。交易查询除需要基本的商户信息和其他配置外，还需终端提供交易的订单号和交易的时间。将所有信息封装成功一个Map对象后，向银联发送查询请求。首先，根据银联申码过程中的签名方法对请求字段进行签名。然后，银联端返回查询结果，终端对该结果进行签名的验证。验证成功后，查看respCode字段是否和返回正确信息时出现的value值相匹配，若匹配则解析出origRespCode字段进行返回。最后，判断origRespCode的value值判断交易是否成功。

银联二维码支付中，银联向商家提供需要的证书，证书中包含加密使用的公钥和私钥，每次发送请求时都会对请求域进行签名和加密，请求结果返回时，也会对结果进行签名的验证，这种双重的防护极大保障了交易的安全性。

#### 5.3.3.2 支付宝二维码支付

支付宝二维码支付和银联支付相类似，都需要对请求消息使用私钥加密，使用公钥验证，支付宝移动支付使用的是RSA安全签名机制。由商家生成二维码信息，消费者使用支付宝终端打开扫一扫进行扫码支付。支付完成后，由终端系统查询支付的结果。

首先，需要获取支付宝支付的公钥和私钥，在代码中对公钥、私钥、请求URL、商户APPID等进行过配置，解析选货后的商品价格和货道信息，并根据支付宝API中提供的公共请求参数对其他参数进行相应的配置，组成一个POST请求内容。请求参数使用key/value模式进行存储，并设置二维码失效时间。请求参数中包含一个key为sign的字段，将除此字段的其他键值对用&符号连接，并按名称排序。然后，将字符串进行使用私钥和RSA算法进行签名，并使用Base64进行编码，将签名结果放置在sign字段。之后将该请求字符串发送给支付宝服务器。最后，支付宝端返回一个Json格式的结果，对alipay\_trade\_pay\_response部分进行验签，对其使用Base64进行解码，之后使用RSA算法进行签名的验证，验证时使用的是支付宝提供的公钥，验证成功后将结果进行返回。判断alipay\_trade\_pay\_response中key为code的字段对应的值是否为10000，请求成功时code为10000，msg为Success，并生成交易号和qrCode，此时的Json串中的qrCode即为需要的二维码流水号，将qrCode从Json串中解析出来，返回售货机终端系统。售货机终端系统再使用二维码生成类即可将qrCode转换成二维码，显示在页面中。

支付宝申码过程中会设置二维码的支付超时时间，二维码生成后，超过这个时间还没有用户对其进行支付的情况下，系统将该二维码信息设为失效，不能再进行扫描支付。每当用户对一个收款二维码进行支付后，系统同样将该二维码设为失效，以防止重复支付。订单号在移动支付的过程中唯一，不能有同样的订单号产生。

用户扫码支付成功后，终端系统会向支付宝服务端查询该交易是否成功。首先，商户需要提供交易的订单号，并将其他配置信息进行设置，封装成一个key/value的形式向支付宝服务端URL发送POST查询请求。支付宝端返回一个JSON格式的字符串，查询alipay\_trade\_query\_response对应的value值中code或trade\_status的值的状态，当code值为10000同时trase\_status值为TRADE\_SUCCESS时，此时的交易状态成功，查询到正确的交易记录。

支付宝二维码支发送请求时返回Json对象，使用GSON包将Json字符串进行解析，传输过程中使用HTTPS协议进行传输。商户作为被扫方，向支付宝进行申码，申码成功后在售货机终端页面中显示付款二维码，消费者打开支付宝APP进行扫码支付，支付成功后，订单立即创建，二维码失效，不能被重复支付。

#### 5.3.3.3 微信二维码支付

微信二维码支付采用“扫码支付”的支付模式，消费者打开微信扫一扫功能扫描二维码支付订单。同银联支付和支付宝支付相类似，微信支付采用HTTPS的传输模式，使用POST发送请求，不同的是银联和支付宝都使用JSON格式的对象或字符串进行请求的提交和数据的返回，微信采用XML格式进行请求的提交和返回。此外，微信采用MD5和SHA等算法对请求对象进行签名。微信二维码二维码支付有两种模式，一种为商户生成固定的二维码信息，由用户扫描并输入信息进行支付，另一种是由商户生成动态的二维码信息，一个订单对应一个二维码，用户支付后二维码失效。结合这两种模式和自动售货机的交易类型，这里选用第二种开发模式，由商户生成带有价格且具有失效控制的二维码信息。

首先，商家通过微信公众平台或开放平台进行支付账号的申请，按照申请步骤进行申请后可获得微信公众账号appid、微信支付商户号mch\_id、秘钥key和接口密码secret等信息。将获取的商户账号信息和其他必备参数进行设置，添加二维码失效时间字段，添加随机字符串nonce\_str字段。接着，将请求字段封装成key/value格式的Map对象，对map对象进行签名。将该请求Map对象中的每一个除sign字段的非空值使用key=value的字符串进行表示，每一对中使用&符号进行连接，并将其按照字母顺序进行排列，最终连接成一个待验签的字符串。将该字符串使用MD5算法进行运算，运算得到的字符串全部转换为大写字符，并将最终得到的签名字符串放置在sign字段的value位置上。然后，将String对象转换为XML格式的字符串，以Post方式将该请求字符串发送给微信端。最后，微信端返回一个XML格式的字符串，解析字符串，当return\_code的和result\_code值为SUCCESS时，表示申码成功，解析出XML中的code\_url字段的值，将其进行返回。并使用二维码生成类，将生成的二维码显示在页面中。

二维码生成之后，消费者使用手机微信APP打开扫一扫，进行扫码支付。每个二维码设有失效时间，一旦超时或者已被扫描支付便不可重复扫描和交易。交易结束后，商家可根据微信订单号进行交易信息的查询。

商家查询交易状态时需要传入微信订单号和其他基本信息配置。首先，按照申码过程中的请求数据封装过程将请求字段进行MD5签名，之后将请求域转换为XML格式的字符串，以Post方式发送给微信服务端。然后，服务端接收请求参数进行数据处理，并将处理结果以XML格式进行返回。最后，将返回结果进行解析，判断return\_code和trade\_state的值是否为SUCCESS，这里return\_code为通信成功的标识，trade\_state为交易成功的标识，只有二者都为SUCCESS才能判断当前请求结果是否正确，该订单的交易是否正确完成。

# 总结与展望

本文描述了基于SaaS模式的自动售货机云平台的设计与实现，该平台是为适应当今自动售货机厂商和运营商发展现状的一个新概念的设计。平台采用了SaaS模式的设计思想，结合多租户设计框架，将传统的仅供一家公司使用的单个应用系统转变为可供多个租户使用的共享平台。近些年来，共享的概念逐渐走进大众的视野，比如共享单车、共享KTV、共享汽车等概念，开发商仅仅投入一件商品，却可供多数人重复使用，不仅如此，共享商品的价格较之传统商品也低廉的多。本文描述的平台系统也采用共享的思路，开发商使用多租户框架和SaaS思想开发一套多租户的自动售货机平台，将其部署到开发商提供的服务器上进行过发布，同时给商家提供租用系统的界面，商家租用成功后直接通过网页访问进行使用。

平台系统采用Spring、SpringMVC、MyBtais框架，Maven构建工具、Mysql数据库和Bootstrap、JQuery、Ajax等技术进行开发。系统采用三层架构的思想，将数据层、业务层和控制层进行分层开发。

系统开发了Android端的终端系统，分为两个模块：一为供操作员使用的手机端应用；二为供售货机使用的终端售卖系统。终端系统使用Http协议和后台进行交互，进而实现访问和操作数据库的功能，终端和后台进行交互时使用加密传输保证传输数据的安全性。终端系统使用Http传输，使用数据加密方式进行数据传输。售货机终端系统提供售货机商品的货道展示，终端系统使用Adapter和网格方式显示商品列表，用户选购商品后可通过扫描二维码的方式进行商品的支付，移动支付方式提供了微信支付、支付宝支付和银联支付三种方式。

该平台系统主要有以下几个特点：

* 易用性：系统采用SaaS模式进行开发，由开发商端进行系统的部署和发布，用户单独在本地安装软件，可直接通过网页入口访问平台系统，使用简单便捷，系统简便易学，适合各个层次的人员使用。
* 通用性：系统针对自动售货机厂商和开发商进行开发，面向用户类型明确，开发范围基本符合自动售货机厂商和运营商的使用范围。
* 隔离性：系统采用多租户模式，各个租户之间数据进行隔离，数据库中对用用户或其他敏感信息进行加密操作。
* 经济性：系统有开发商进行开发、部署和维护，不需要使用者提供技术的支持和维护工作。第一方面，租户商家节省了雇佣技术人员的额外开销和人员配置；第二方面，租户商家不需要对系统进行运维支持，减少了工作量；第三方面，该系统为一个共享的多租户系统，因一次开发多个租户使用，其租金较为低廉，从长远来看，非常适合想要减少开销的租户。此外，终端系统使用价格低廉的Android设备进行开发。
* 便捷性：Android终端提供了移动端的货道更新操作和二维码的移动扫码支付。

然而，该系统还存在一些缺点。目前自动售货机商家的发展进程良莠不齐，各个商家使用的售货机种类不已，终端系统的通用性还未做大规模的测试，想要真正统一推广还需加大工作力度和用户调研，以保证能够将系统进一步优化，以供不同种类的商家使用。此外系统采用统一开发和共享数据库、共享数据架构的多租户模式，共享程度高。因为自动售货机行业发展现状不同，行业内业务形式较为类似和统一，该平台系统只提供了大多数公司都能使用的业务功能，暂时不支持用户功能的自定义定制。要真正达到这种要求，这还需要对该行业进行进一步的调研，根据新的调查和对新技术的研究对系统进行更新和完善。

# 参考文献

# 致谢