# 摘要

目前，互联网行业和电子商务飞速发展，淘宝、天猫、京东等互联网公司的崛起，无不在挤压着零售业的市场。自动售货机行业作为一个“老牌”零售业也在发生着日新月异的变化。

针对当前互联网的发展趋势，本文提出一个基于SaaS服务的自动售货机云平台的设计方案，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，并且配置相应的Android终端和最新的移动支付方式。该SaaS云平台有两种租户类型，分别是售货机厂商和售货机运营商，按功能又可以分为三个方面，分别是管理员、厂商和售货机模块，每个功能模块下还包括多个功能点。与此同时，本系统还提供基于Android的运营商端的操作员客户端以及自动售货机的销售终端。售货机终端提供广告视频的循环播放服务、商品的选货服务、赠品服务和订单的移动支付（支付宝、微信、银联）服务等。

自动售货机云平台基于SaaS模式（Software-as-a-Service）进行设计，数据库隔离基于多租户思想进行设计，使用JavaWeb分层思想进行云平台的开发。系统后台运用SSM(Spring+SpringMVC+Mybatis)框架进行搭建，具有很强的重用性和扩展性；数据库使用Mysql进行实现；系统界面使用JSP、BootStrap、JQuery、Ajax等前端技术进行设计和开发，符合当前UI设计的发展趋势；客户终端使用miniUSBFT1D串口转换线和设备主控板进行连接，采用Android和串口技术进行开发，能够有效的降低产品成本。该平台提供基于Web的访问入口，租户通过访问Web页面进行注册和系统的使用，无需在本地进行系统的安装和维护。

系统使用分层设计模式，租户种类和用户角色层级嵌套，实现了基于SaaS模式的云平台系统。系统的部署环境和运维最终由软件提供商负责，租户根据需求进行系统的租用，减少了资源的浪费和信息化的成本。

**关键词：**SaaS平台、多租户框架、SSM框架、自动售货机、Android、移动支付

# ABSTRACT

At present, the rapid development of the Internet industry and e-commerce, Taobao, Tmall, Jingdong and other Internet Co's rise, all in the extrusion of the retail market. The vending machine industry as a "veteran" retail industry is also experiencing change rapidly changes.

In view of the current development trend of the Internet, this paper proposes a design scheme based on SaaS service automatic vending machine cloud platform, the mode will change the traditional application system just for a company to a new shared platform for multi-tenant , and configures the corresponding Android terminal and the latest mobile payment mode. There are two types of tenants of the SaaS cloud platform, respectively is the operator of the vending machine and vending machine manufacturers, According to the function, it can be divided into three parts, namely administrator, vendor and vending machine module, each module also includes a plurality of function points. At the same time, the system also provides Android client based on operators, client operators, and vending machines for sale terminals. The vending machine terminal to provide advertising video playback, selection goods, gifts and mobile payment (Alipay, WeChat, China UnionPay) services etc..

The cloud platform of vending machine is based on SaaS model (Software-as-a-Service) for the design, database isolation is based on the idea of multi tenancy, and the development of cloud platform is implemented by using the idea of JavaWeb hierarchy. System background using SSM(Spring+SpringMVC+Mybatis) framework to build, has strong reusability and expansibility. Implement the database used by Mysql， system UI using JSP, BootStrap, JQuery, Ajax and other front-end technology in design and development, flow the trend of UI design; client terminal use miniUSBFT1D serial port line to communite with the equipment master control board, it is developed by Android , can effectively reduce the cost of products. The platform provides access to the portal by web, which allows the tenant to register and use the system by accessing the Web page without the installation and maintenance of the system locally.

The system uses hierarchical design pattern, tenant type and user role hierarchy nesting, and implements the cloud platform system based on SaaS mode. The deployment environment, operation and maintenance of the system are ultimately carried out by the software provider. The tenants rent the system in accordance with the requirements, thus reducing the waste of resources and the cost of information.

**Keywords:** SaaS platform, multi-tenant framework, SSM framework, vending machines, Android, mobile payment

目录

[摘要 I](#_Toc494055399)

[ABSTRACT II](#_Toc494055400)

[目录 IV](#_Toc494055401)

[第1章 引言 1](#_Toc494055402)

[1.1 研究背景及意义 1](#_Toc494055403)

[1.2 研究现状 2](#_Toc494055404)

[1.3 主要研究内容 3](#_Toc494055405)

[1.4 本文结构 3](#_Toc494055406)

[第2章 相关技术概述 5](#_Toc494055407)

[2.1 共享平台设计方法的分析 5](#_Toc494055408)

[2.2 三大框架的选择 7](#_Toc494055409)

[2.3 Android的使用 10](#_Toc494055410)

[第3章 需求分析 13](#_Toc494055411)

[3.1 课题研究目标 13](#_Toc494055412)

[3.2 可行性分析 13](#_Toc494055413)

[3.3 需求分析 14](#_Toc494055414)

[3.3.1 自动售货机SaaS云平台 14](#_Toc494055415)

[3.3.2 Android终端 20](#_Toc494055416)

[第4章 系统设计 23](#_Toc494055417)

[4.1 云平台结构的分析 23](#_Toc494055418)

[4.2 三层架构实现方式 23](#_Toc494055419)

[4.3 售货机云平台系统设计 26](#_Toc494055420)

[4.3.1 系统管理员模块 26](#_Toc494055421)

[4.3.2 运营商模块 26](#_Toc494055422)

[4.3.3 厂商模块 27](#_Toc494055423)

[4.4 Android终端系统设计 28](#_Toc494055424)

[4.5 数据库设计 28](#_Toc494055425)

[第5章 云平台系统的实现 34](#_Toc494055426)

[5.1 租金模块实现 34](#_Toc494055427)

[5.3 系统权限实现 37](#_Toc494055428)

[5.4 厂商模块实现 38](#_Toc494055429)

[5.5 运营商模块实现 40](#_Toc494055430)

[5.6 文件服务器 43](#_Toc494055431)

[第6章 终端系统的实现 45](#_Toc494055432)

[6.1 数据传输加密实现 45](#_Toc494055433)

[6.2 操作员客户端实现 45](#_Toc494055434)

[6.3 售货机终端实现 47](#_Toc494055435)

[第7章 总结与展望 58](#_Toc494055436)

[参考文献 60](#_Toc494055437)

# 第1章 引言

## 1.1 研究背景及意义

自动售货机是继超市、便利店之后的一种全新商业零售模式，类似于阿里推出的无人超市，自动售货机是更加小型、便捷的无人超市。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20世纪70年代起，自动售货机零售业在西方发达地区发展起来，因其不受地域的限制，占地面积较小，节省人力物力资源且24小时不停机服务，而被称为24小时营业的微型超市[[[1]](#endnote-2)]。在日本甚至分布到田间地头等客流量相对较少的地区。上世纪初，我国经济逐步发展，城市人口大幅增加，即时购买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[[[2]](#endnote-3)]，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”。

自动售货机与传统便利店、无人超市、网上购物等在各个销售领域发挥职能，相互配合市场的运营，又相互竞争市场占有量。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利润越来越微薄，投放成本和运维成本也水涨船高；其次，信息社会的迅猛发展，让已习惯高度信息化生活的消费者难以再适应传统的售货机销售渠道，传统投币式已跟不上无现金支付的潮流；最后，很多传统自动售货机并未实现实时数据同步，管理员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，这也给运营工作带来阻碍。

面对发展的众多问题，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，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落后，因此自动售货机将传统仅使用投币和刷卡的支付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[[[3]](#endnote-4)]。多元的支付方式不仅方便了消费者，还省去了清点现金的麻烦，更加节省人力。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，这种方式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环境之中。

## 1.2 研究现状

随着我国互联网技术的发展，整个自动售货机行业都在改善其发展态势[[[4]](#endnote-5)]，售货机制造商的硬件逐步优化、系统开发逐步更新、支付方式逐步多元化。

自动售货机作为一种共享性质的销售载体，在日本和其他发达国家发展相对成熟。我国引入自动售货初期仅仅是引入国外的运营方式、售卖品类和软硬件技术，独立创新的东西甚少，这也造成其发展被各方面掣肘。随着国内技术的逐渐成熟，售货机行业开始开展新型系统的开发：一方面，提供一套供商家使用的云平台管理系统，用于运营商的用户管理、机器管理、货道管理、订单管理、库存管理、商家管理等；另一方面，在售货机终端配备可操作的显示屏，用以提供支付宝、微信、银联等移动支付页面[[[5]](#endnote-6)]。后台管理系统和售货机终端系统的结合使用，在提高消费者购买商品便利性的同时也方便了商家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，增加了产品的新颖度和支付手段[[[6]](#endnote-7)]，同时售货机空闲时可在显示屏上播放广告，又增加额外赢利点。目前出现了两种类型的新型系统，一种是在机器上提供一个包含位置的二维码，通过扫码获取商品信息[[[7]](#endnote-8)]；另一种在终端提供显示器，显示商品列表，用户选货后通过投币、移动支付等多种方式进行支付，用户处支付外，不需要其他的交互。

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，投放一台自动售货机，初期要考虑售货机的购置资本、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。新型软件能够吸引消费者，在增强在同行业中的竞争优势的同时，昂贵的开销也令一些公司望而却步。

## 1.3 主要研究内容

本文描述了基于SaaS(Software-as-a-Service)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。自动售货机行业结合互联网科技和SaaS服务将普通web应用升级为共享应用。近些年来，共享的概念逐渐走进大众的视野，比如北森一体化人才管理平台和包含在线Office的[Google](https://baike.baidu.com/item/Google)Apps，开发商仅仅开发一套平台系统应用，却可供多数人重复使用。不仅如此，共享平台的使用价格较之传统平台也低廉的多。

SaaS 是互联网技术的一种全新的应用模式，是软件即服务的简称，它将软件部署在服务提供方的服务器上，服务商负责软件的开发、运维和数据管理等。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。这种模式极大地增强了软件的复用率、减少了成本，对自动售货机行业来说，能够降低软件开发的成本和开销，帮助售货机行业应对成本和效率的挑战。

软件提供商根据需求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。这种多租户共享的软件服务降低了软件开发和软件维护的成本，用户将功能实现完全托付给开发商，不需要考虑软件的开发、维护和升级，降低自动售货机公司的运营成本的同时，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及意义进行介绍，对论文的研究现状和主要研究内容进行分析。
2. 相关技术概述。主要介绍了J2EE体系结构、SSM框架、SaaS服务、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和用例图。从两个方面进行，一个为SaaS平台端，一个为Android端。
4. 系统设计。主要介绍系统的基本结构、系统设计和数据库设计，并且设计出数据库E-R图。
5. 云平台系统的实现。主要介绍了租户模块的分析和实现、系统权限的设计和实现、厂商模块和运营商模块的分析实现，并做出相应的时序图。
6. 客户端的实现。主要介绍了客户端的分析实现，首先介绍了操作员客户端的各个模块功能和实现，做出对应的时序图。然后介绍了售货机终端的设计与实现，包括购物流程、串口通信、移动支付和版本更新等模块，并做出相应的时序图和流程图。
7. 总结与展望。介绍系统的功能和方法，并提出进一步的展望。

# 第2章 相关技术概述

本章介绍相关的技术和概念，主要从三个方面阐述：首先，简单地介绍了SaaS模式、多租户框架以及数据库隔离模式；然后，系统地阐述了SSM框架和其它框架的对比；最后，详细地描述了Android技术和所使用的相关组件。

## 2.1 共享平台设计方法的分析

1. 多租户架构

多租户技术（multi-tenancy technology）是一种软件架构技术，为实现一套可供多个企业用户公用的软件服务体系[[[8]](#endnote-9)]。多租户技术是SaaS模式的重要支持技术，是SaaS服务区别于其他服务的一个显著特性，它为SaaS软件服务数据层提供了开发思路和技术支持。传统系统中软件使用者和软件系统是严格的一对一关系，体现为多用户模式；而多租户架构下软件使用者和软件系统可为一对多的关系，体现为多租户模式，每个租户在相应的权限范围内使用系统提供的软件服务，租户内部又表现为多用户模式[[[9]](#endnote-10)]。

多租户架构在数据隔离方面体现为三种设计方式：独立数据库、共享数据库但使用独立schema、共享数据库同时共享schema[[[10]](#endnote-11)]。多租户数据的隔离模型一般根据商户的特性进行选择，若系统对于数据安全性和隔离性要求极为严格，一般采用第一种类型的存储结构。若仅要求一定程度的数据安全性和隔离性，可使用第二种隔离模式。若对于数据信息敏感性不高，同时要缩减开销，可使用第三种隔离模式。

1. SaaS服务的分析

SaaS是一种可通过Web访问的软件模式平台[[[11]](#endnote-12)]，是IBM提出的一种新型软件交付服务，软件部署在第三方服务器上，租户根据自己的实际需求订购软件服务，无需购买、安装、维护或升级任何软件产品，能够在享有完善服务的同时，耗费最少的信息化成本。SaaS服务往往和多租户技术同行，多租户技术的核心思想给SaaS软件服务提供了数据隔离的思路和方法[[[12]](#endnote-13)]。

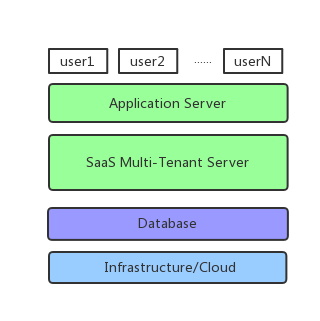
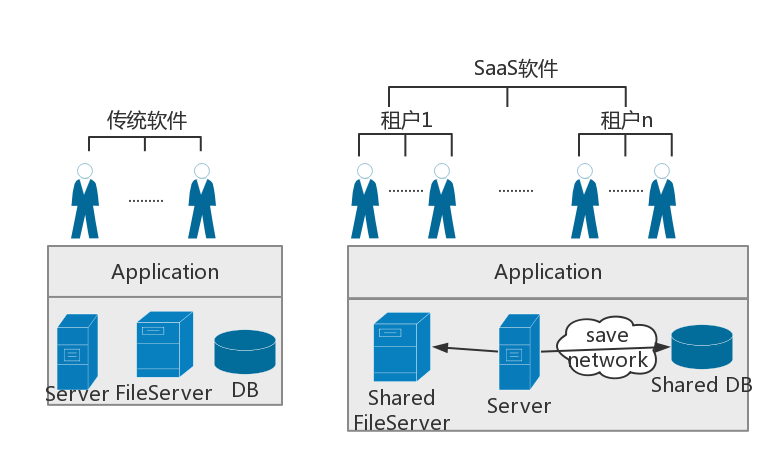
SaaS主要通过采集租户需求，将该行业内的软件需求进行抽象和封装，形成一套完整的开发体系。软件提供商提供服务器、软件服务和共享数据库，负责平台的开发、测试，最终统一部署到自己的服务器上。类似于较为常见的云服务器的使用模式，用户可以根据自己的需求，按照一定的收费规则向软件服务商购买SaaS软件服务。SaaS的软件服务采用一对多的运营模式，租户通过web入口直接访问使用。租户租金往往覆盖了软件的开发费用、运维费用以及软件的许可证费用等。共享的软件服务将单个用户进行软件开发和运维的成本分摊到各个租户的租金中，让软件开发费用变得低廉。这种模式非常适用于软件技术部门不完善的中小型企业，是其实现信息化的捷径。如图2-1为SaaS多租户软件服务的架构。

图2-1 SaaS多租户软件服务的架构

SaaS多租户软件服务架构由下至上第一层为基础层，该层一般为普通服务器或云服务器；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；最高层为各个租户实体[[[13]](#endnote-14)]。SaaS服务商进行软件开发和数据的维护，并为租户提供初始用户名和密码，初始租户管理员具有该租户内部的最高权限，用户根据权限使用系统的不同服务，不同租户间相互隔离，互不冲突。服务提供方管理员仅对租户的最高权限管理员进行管理，并不干涉租户内部的管理和层次结构。

传统软件与SaaS软件的对比如图2-2。

图2-2 传统软件与SaaS软件对比

## 2.2 三大框架的选择

SSM框架即Spring + SpringMVC + MyBatis框架的整合[[[14]](#endnote-15)]。通过xml配置文件配置数据库连接池、事务管理和注解等内容。使用Spring管理业务逻辑层，使用MyBatis管理持久层，使用SpringMVC管理表现层。

1. Spring框架和SpringMVC框架的关系

Spring以IoC和AOP为内核，是一种轻量级的容器框架，能很好的解决JavaEE企业级用用程序开发的复杂性。它将传统的开发和配置变得简洁，可以使开发人员集中更多的精力到业务逻辑上去，从一定程度上缩短了开发时间。

Spring不仅方便了解耦，还可以控制对象间的依赖关系。它提供了用于测试的模块，可通过简单的注解实现单元测试。Spring也提供声明式事务的支持，能够灵活的控制事务管理。同时，还可与其他优秀框架进行良好的集成，如Struts2、Hibernate和MyBatis框架等。

Spring框架由七个模块组成，提供了企业级开发中包括持久层、业务层和展示层的所有支持。Spring构建在核心容器上，其主要分布如图2-3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的基础架构并提供框架的基础功能。它实现了IoC模式， IoC是Inverse of Control的缩写，表示控制反转。程序之间的关系完全交由容器控制，实现类的选择控制权移交给第三方裁决，用户不用关心对象的创建和销毁，降低了类与类之间的依赖关系和代码的耦合度。

图2-3 Spring框架组件图[[[15]](#endnote-16)]

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。它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、异常管理等方面会产生大量不利于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，有效降低代码的耦合性[[[16]](#endnote-17)]。

Spring还提供了基于Junit4的Spring测试框架，在测试类上使用@ContextConfiguration(locations= "classpath:\*\*.xml")注解指定bean注入的配置文件，使用@RunWith(SpringJUnit4ClassRunner.class)指明集成Spring单元测试的的测试运行器[[[17]](#endnote-18)]。

SpringMVC是一种基于MVC三层模式的web框架，它使用注解通过简单的POJO类作为控制器进行请求的处理和转发。SpringMVC的框架模型如图2-4所示。SpringMVC围绕DispatcherServlet展开，它负责将请求发送到相应的后台处理程序。使用@Controller注解将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格式的的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。SpringMVC使用注解的方式实现方法的注入，不需要手动创建对象，同时能够实现数据的请求转发，尽量减少了xml配置文件的使用。SpringMVC集成了Spring的优点，将企业级web开发变得更加简洁。

图2-4 SpringMVC框架模型图[[[18]](#endnote-19)]

1. 数据持久层框架的对比及选择

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[[[19]](#endnote-20)]。MyBatis支持定制化SQL语句、存储过程以及高级映射，它使用注解或xml配置将接口和POJO类映射成数据库中的记录。首先，MyBatis根据配置文件创建SqlSessionFactory；接着，使用注解或者配置文件来获取SqlSession；然后通过sql语句执行数据库操作，最后关闭SqlSession。MyBatis架构图如图2-5所示。

图2-5 MyBatis架构图[[[20]](#endnote-21)]

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，根据简单的 API对数据库实现增删改查操作，使用Mapper接口满足了面向接口编程的需要。数据处理层是MyBatis的核心，通过传入的参数值动态的构建SQL语句，将返回结果集转换为List<E>列表或对象，使得MyBatis有很强的扩展性和灵活性。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。它可以和Spring框架很好的整合，通过Spring自动扫描装配Dao，使用SQL构建器动态构造SQL语句[[[21]](#endnote-22)]。

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，根据MyBatis和Hibernate这两大框架的适用类型和成本进行考虑和分析，主要分为以下几点：

* Hibernate采用全自动对象关系型映射，Mybatis采用半自动的映射。MyBatis的SQL语句可根据需求进行定制，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构建方法可以按照开发人员的意志操作指定的字段。此外，使用原生的SQL语句进行数据库的操作在性能上更加优化，查询速度更快，拥有更高的灵活性和可控性。
* Hibernate对原生的JDBC进行了封装，而MyBatis是基于原生的JDBC，从运行速度上来说MyBatis更有优势。
* 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项目的实际适用性和成本等多种因素进行选择。所以本文选用了能够动态定制SQL语句的MyBatis框架，并将其整合Spring、SpringMVC进行整个系统的开发[[[22]](#endnote-23)]。

## 2.3 Android的使用

Android是由Andy Rubin开发的可用于手机、平板电脑、电视、手表等设备的一款基于Linux的操作系统 [[[23]](#endnote-24)]。自Android问世以来，便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，Android系统结构图如图2-6所示。

图2-6 Android系统体系结构[[[24]](#endnote-25)]

Android的系统架构分为四层，从下至上分为Linux内核层、系统运行层、应用框架层和应用层[[[25]](#endnote-26)]。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，如开发时常用的Activity Manager、Content Providers等；应用层提供核心应用程序，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)）[[[26]](#endnote-27)]。

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，用户通过窗口进行交互。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的生命周期中包含四种状态，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。Activity的生命周期如图2-7。

Service是Android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，它不需要任何交互界面[[[27]](#endnote-28)]，可以让服务一直后台运行以满足用户需求，是后台默默的执行者。只有当用户进程被Kill时，相应的Service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()。

图2-7 Activity生命周期图解

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，能够通过广播接收器实现各个程序之间信息的交互，在程序之间实现广播的发送和接收。Broadcast Receiver也未提供任何可供交互的用户界面，用户不能进行显式的操作。它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Broadcast Receiver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)可以实现程序间的信息共享，主要用于对外共享数据[[[28]](#endnote-29)]。Content Provider一般提供一个供用户存储或使用数据的接口，它对底层进行了抽象和封装，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。

# 第3章 需求分析

基于SaaS模式的自动售货机云平台系统分为云平台管理系统和终端两大模块。云平台管理系统包括自动售货机厂商和运营商两个模块，终端包括操作员终端和自动售货机终端两个模块。本章将从云平台系统和终端系统中的四个部分进行需求分析。

## 3.1 课题研究目标

主要使用SSM框架、多租户模型和SaaS模式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）使用的管理系统及终端系统。

云平台系统分为两大模块，分别是自动售货机厂商模块和运营商模块。售货机厂商系统主要包含用户管理、运营商管理、售货机类型管理等。运营商系统主要包括用户管理、售货机管理、库存管理、商品管理、财务管理等模块。

系统的Android终端同样分为两大模块，分别是操作员（售货机上货员工）客户端和自动售货机终端。操作员客户端为用户提供一个可随时随地进行售货机查询和货道更新的Android应用，免去了人工记录的麻烦。售货机终端主要完成自动售货机的日常销售工作，为用户提供支付宝、微信以及银联二维码的移动方式进行支付。

## 3.2 可行性分析

基于SaaS模式自动售货机云平台的设计从技术、经济和操作的可行性等方面进行分析[[[29]](#endnote-30)]。

从技术上来说，数据库开发模式使用多租户架构的概念，云平台后台使用SSM框架进行开发，平台前端使用JSP、BootStrap、Ajax、Javascript、JQuery等进行前端开发，终端使用Android技术进行开发。

从经济上来说，系统由第三方开发，租户不用付出人力和物力，通过web入口进入租用页面，按需购买平台使用时间和容量，最大程度的减少开销。

从可操作性上来说，自动售货机厂商为售货机生产者，售货机厂家可对自己生产的售货机进行增删改查的基础管理。售货机运营商维护售货机的日常运营和销售活动。操作员终端主要进行售货机的查询和货道商品的更新操作，Android终端首先通过HTTP请求处理和后台服务器进行交互，获取后台数据信息，然后查询请求的售货机货道的当前信息，操作员根据货道编号和商品数量对当前货道进行更新。售货机终端用于售货机的日常运营活动，并通过HTTP请求访问后台管理系统，将每次售货的结果传回后台系统进行数据的更新。

系统从技术、经济和操作三个方面都具备开发的可行性和必要性。

## 3.3 需求分析

本节将从系统管理员、运营商、厂商和消费者等几个方面描述系统的需求分析和用例图。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[[[30]](#endnote-31)]，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。系统需求与软件功能仅通过自然语言进行描述，这种方法经常造成使用者或开发者理解上的偏差与审核的疲劳。随着UML系统建模的出现，通过可视化的方法描述系统需求逐渐成为软件开发中的一项基本要素。

### 3.3.1 自动售货机SaaS云平台

SaaS云平台供三种类型的租户使用，分别是系统管理员、自动售货机运营商管理员和自动售货机厂商管理员。该平台系统可供多个租户使用，租户通过Web入口输入租户编号、用户名和密码进入系统。平台系统管理员主要进行系统权限管理、用户管理、租赁管理、定价管理等活动。运营商端管理系统主要进行售货机日常运营、售货机维护和后续活动的管理，包括用户管理、售货机管理、货道管理、商品管理、订单管理等，每个管理员角色拥有不同数量的权限，负责相应的管理区间。此外，运营商端管理系统还提供自动售货机销售终端需要的货道信息和商品信息。售货机根据后端提供的信息展示商品列表，进行购物交易，并将交易记录和货道变更信息回存到后台管理系统。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、合作商家管理等内容。

1. 系统管理员

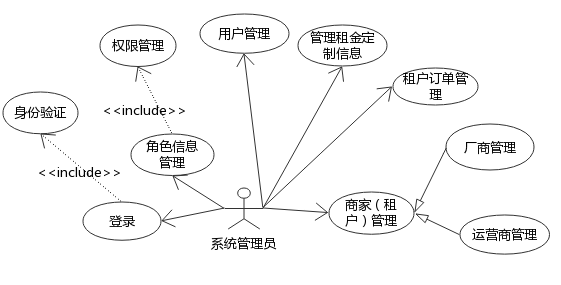
系统管理员是平台的提供商，处于管理的第一层。基于SaaS的售货机云平台是由第三方提供商提供，用户通过访问web页面根据实际需求进行系统空间和时间的租用。系统管理员为一级用户，需要对平台的租用规则进行定义，对处于第二级和二级以下的租户进行管理。系统管理员用例图如图3-1所示。

图3-1 系统管理员用例图

* + 用户角色管理：

用户角色包括权限管理和角色管理两个部分，每个角色可以由多个权限组成，不同的权限对应不同的操作区间。用户权限包括权限名称、权限编号、权限描述、权限所属租户类型等属性。权限所属类型分为三种：系统管理员、运营商和厂商，在数据库中用数字0、1和2进行标识。

* + 租金信息管理：

系统管理员对SaaS平台的租金表进行定制，规则如下：若用户第一次使用系统，可免费试用30天，30天之内能够使用系统所有功能，给用户一个租用前的免费体验期。30天试用期结束后，系统标记租户已行使了试用机会，不能再次免费使用。管理员定制规则，设置每月的租用费用，若租用期达到一年，设置租金折扣为85折；达到两年设置租金折扣为75折；一次性定制三年则设置租金折扣为65折。此外，设置每增加一台租用空间所需的金额，厂商和运营商端可分别设置。根据以上规则用户在系统页面中自行设置租用时间和租用空间。若租户租用期限到达，系统给出租用到期的提示，并将租户设置为不可用状态。

* + 租户管理：

租户是系统管理员所对应的“用户”，租户通过注册页面选择租用时间和租用空间进行租金缴纳和商户的注册。

1. 运营商端管理系统

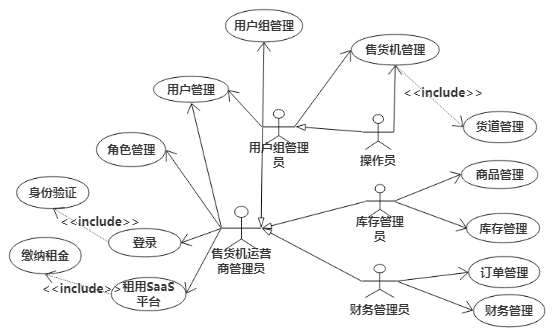
售货机运营商是系统的主要用户之一。软件租用成功后运营商管理员可添加下级管理员并分配给用户权限，不同的用户权责分明，共同管理租户系统。运营商通过租户租用界面进行系统的租用和商户的创建，然后使用该系统。运营商系统用例图如下。

图3-2 运营商系统用例图

* + 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。进行租用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恢复使用。

* + 用户注册/登录：

用户通过浏览器进入登录入口，输入公司编号、账号和密码进行登录，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。新用户由运营商端管理员进行添加，并提供初始用户密码。

* + 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对用户进行基本的增删改查操作，同时修改用户的角色信息。管理员创建用户组，为每个用户组分配一个组管理员，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。管理员可查看所有角色信息，以及每个角色所包含的用户权限，每个类型的租户只能查看同类角色信息。

* + 售货机管理：

售货机管理模块分为机器管理、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基本的增删该查操作，并将机器分配给操作员，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，填写售货机信息表单进行添加，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基本增删改查操作，将机器分组进行分配。点击添加售货机组按钮，弹出模态框，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，则只分配未被分配的售货机；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应完善货道信息，在货道管理界面进行售货机货道的查询，页面列出了售货机铭牌号、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、在售商品等信息。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，对商品进行基本管理操作。进入添加信息页面，填写详细信息表单后点击保存；进入详情页面，查看详细信息；填写更新信息表单，进行商品信息的更新；点击删除，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、清单打印等模块。操作员填写调拨单，请求分配商品；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、备注信息等内容；仓库管理员进入查询页面查看出库信息。

* + 订单和财务管理：

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HTTP请求，将订单信息保存在后台数据库中，管理员进入管理页面查看自动售货机订单信息。售货机完成交易后会向后台发送一条交易信息，其中包含交易金额，财务管理员对每月的交易金额进行统计。

1. 厂商端管理系统

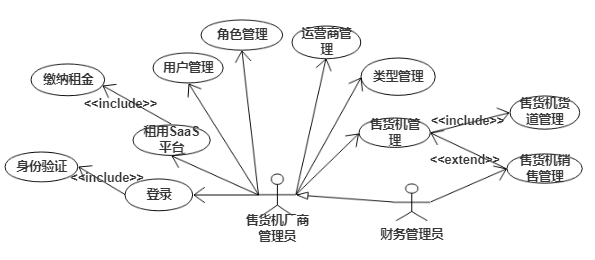
售货机厂商是该平台的用户之一，厂商通过平台的租用页面对软件进行租用，之后独立进行厂商内部的运营和管理。与自动售货机运营商相同，售货机厂商也处于该SaaS云平台的第二级，是系统管理员的下级用户。各个厂商之间相互隔离，分别进行软件的使用和系统的管理。厂商系统用例图如图3-3所示。

图3-3 厂商系统用例图

* 平台租用：

厂商初次使用系统需要对平台进行租用，租户缴纳租金，并完成商家注册，在租户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再次恢复使用。

* 用户注册/登录：

厂商租用平台成功后商家，系统根据商家编号、账号和密码进入相应的页面。厂商管理员可对用户进行基本的增删改查操作。

* 用户管理：

厂商管理员可添加、更新、查询和删除用户，管理员单击添加管理员按钮，弹出用户信息模态框，填写用户信息表单，点击保存用以新增用户；点击详情按钮查看用户详细信息；点击更新按钮，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，弹出还未添加的商家列表，点击列表复选框，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：

类型管理页面主要涉及售货机内的商品存放类型，管理员进入类型管理页面对售货机类型进行基本的增删改查操作。售货机类型独立进行定义，当对售货机进行管理时，将售货机与相应类型进行关联。

* 售货机管理：

管理员对机器信息进行管理，可对售货机进行基本增删改查操作。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表单后点击保存进行提交。售货机详细信息包含售货机类型，页面会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，将向后台发送一个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

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，它们的后台服务都由SaaS云平台管理系统进行提供，每一条数据操作都和后台系统进行交互。

1. 操作员客户端

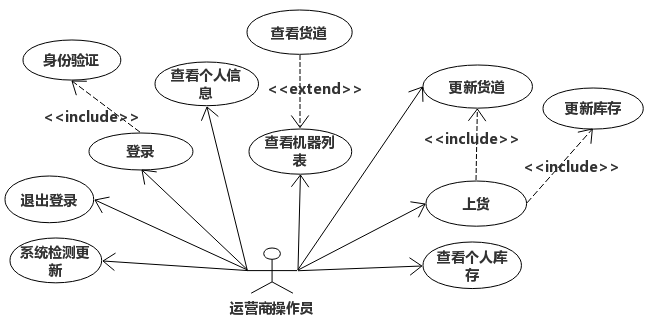
操作员客户端是为自动售货机运营商开发的一套移动端应用，主要为方便操作员上货时货道的更新和货物的统计。操作员通过租户编号、用户名和密码进行登录，系统验证成功后进入个人页面。个人页面包含五个选项菜单，分别是机器列表、更新货道、查看个人信息、查看库存、退出登录和检测更新等。运营商操作员客户端用例图如图3-4。

图3-4 运营商操作员客户端用例图

* 用户登录：

操作员客户端允许登录的用户类型有操作员及其租户空间内上级用户。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：

用户进入售货机页面，点击售货机可查看当前售货机详情；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。

* 货道管理：

用户进入货道管理页面，填写货道更新表单和相应的售货机Id、货道编号、加货量等信息，Android终端向后台服务端发送一条POST请求，将当前售货机对应货道的加货量和现存量进行更新。

* 版本更新：

客户端首页提供版本更新按钮，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处理，否则，下载新的版本文件并进行更新。

1. 自动售货机终端

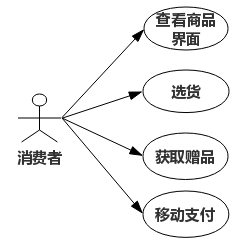
自动售货机终端为消费者提供基本交易渠道。机器终端用例图如图3-5。

图3-5 自动售货机终端用例图

* 多媒体播放：

售货机终端在无人操作时，播放文件服务器上提供的广告视频，视频按照次序轮番循环播放。当有人操作售货机时进入终端主页面。

* 货道管理：

系统通过售货机号访问后台服务器，获取对应机器的货道信息表。信息表中包含货道、售货机信息、商品信息、价格信息、图片路径信息等内容，终端系统使用网格布局在页面中显示商品列表。

* 用户选货：

用户进入选货页面，页面显示商品的图片、价格以及是否有货，点击商品进行选货。

* 商品支付：

用户选择购买商品或者赠送商品，选货成功后，点击确认生成订单。此时终端系统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，后台处理成功后将返回支付的二维码信息，用户可通过银联、支付宝和微信等进行扫码支付。

* 售货机出货：

若上一步为商品选购，商品支付成功后，系统通知自动售货机将被选货道的商品出货。否则，进入赠品页面，输入赠品码，展示商品页面，用户根据赠品码选购相应价格的商品，选购成功后进行出货。

* 赠品服务：

若商品支付时选择赠品服务，则生成一个赠品码。获取赠品码后，用户在任意一台售货机输入赠品码，换赠同等价位的商品。

* 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，记录支付的订单号和支付类型以便于统计，同时对当前售货机的货道余量进行更新。

* 版本更新：

编写单独的应用安装在Android平板上，默认开机自启动。该应用每天自动检测服务器端是否有新的售货机终端的APK文件，若有，则下载后静默安装售货机终端系统并自动重启。

# 第4章 系统设计

## 4.1 云平台结构的分析

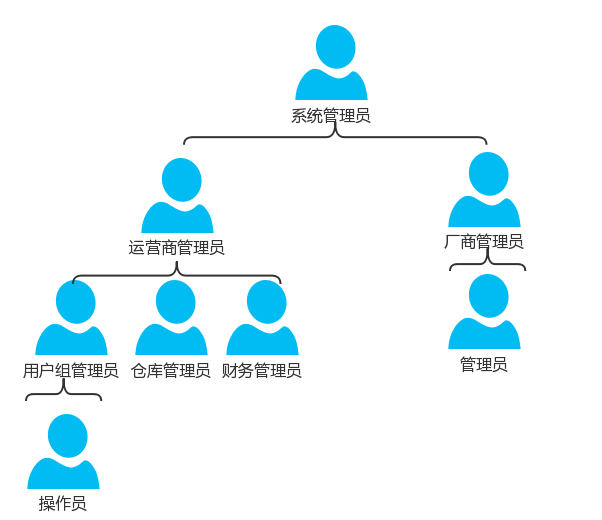
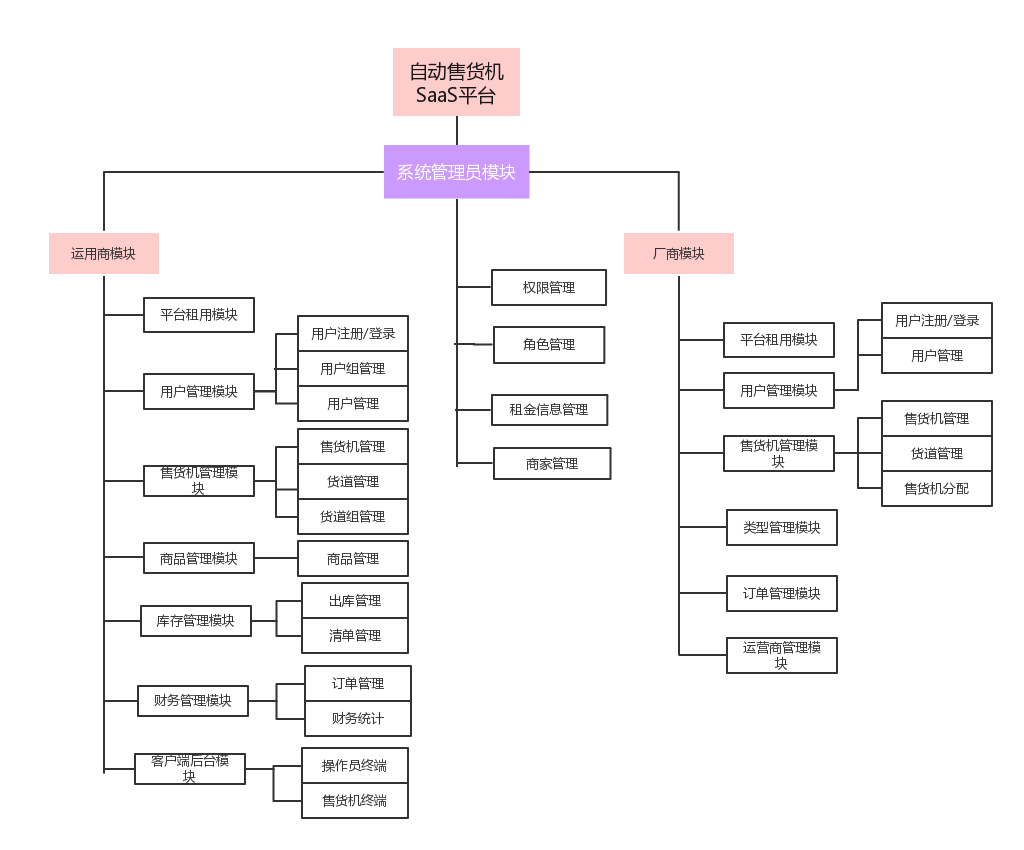
SaaS云平台管理系统用户种类有以下三种：1）系统管理员，具有最高权限，是该系统平台的提供商，提供了基本的租金规范和租用规则，管理厂商和运营商两种租户；2）运营商，该SaaS平台系统的租户之一，对其内部成员和业务进行定制和管理；3）厂商也是该SaaS平台系统的租户，对售货机等信息进行管理。图4-1，为用户层次组织结构图。

图4-1 用户层次组织结构图

根据功能又分为三个模块：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，供运营商使用；厂商模块是售货机提供商所使用。根据运营商和厂商这两种租户类型进行系统的设计。售货机云平台功能结构图如图4-2。

## 4.2 三层架构实现方式

系统采用三层架构进行开发，使用Maven进行版本的控制，所有依赖包在pom.xml文件中进行引入。相关信息在resource文件中列出，包括数据库账号信息、日志信息、框架的配置信息以及数据库连接池等。

图4-2 售货机云平台太功能结构图

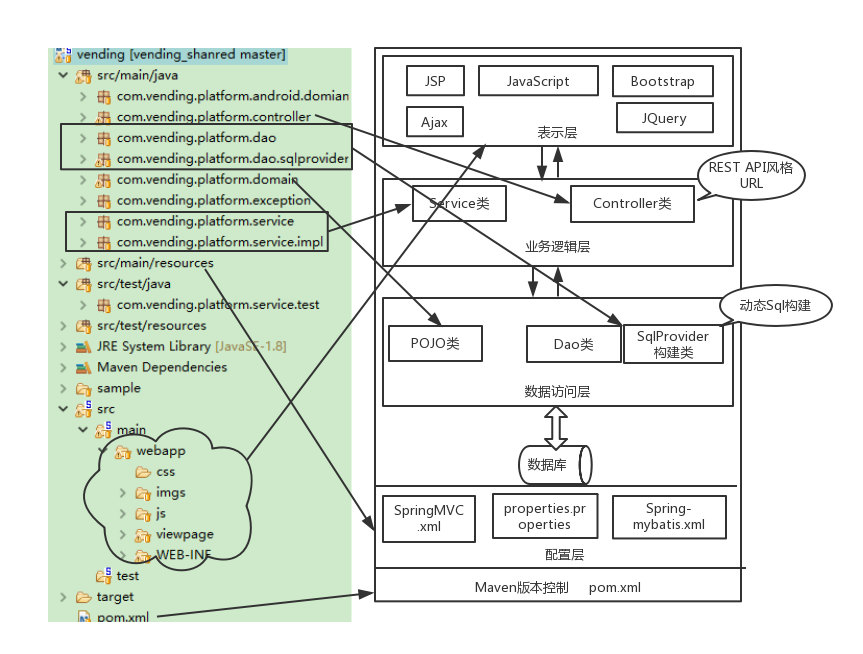
SSM与三层架构中的具体系统设计图如图4-3所示。

图4-3 SSM与三层架构系统设计图

1. 数据库访问层：

数据访问层负责系统和数据库的交互，完成数据的基本操作，系统使用轻量级的Mybatis框架。在pom.xml文件中引入框架所需的依赖，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。

数据访问层由三个部分：POJO类、Dao类和SqlProvider。其中，POJO类罗列了数据库中的所有的字段属性以及数据库中表与表的映射关系。Dao类即MyBatis框架中的Mapper接口，定义了访问数据库的接口，每个接口上使用@SelectProvider注解进行标注，type属性指向SQL构造的类，method属性指向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。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：

业务逻辑层实现系统的业务逻辑，能够控制用户访问Dao层。开发过程主要体现在两个方面， Service和Controller。业务逻辑层的开发需在在配置文件中启动注解，配置扫描包的路径。Service使用面向接口编程的方式进行开发，并注入数据层操作，使用@Service注解对Service类进行标注。

Controller主要用于和用户进行交互，Controller类中使用@Controller注解进行注册。Controller类中包括多个实现方法,使用@RequestMapping设置访问的URL路径，这种Rest API风格的开发模式也是SpringMVC的特色之一。Controller的返回值类型可以是ModelAndView，使用这种返回模式返回一个ModelAndView对象，其中包括了跳转的页面和需要传输的值。可以在spring-mvc.xml配置文件中配置跳转文件的前缀和后缀，在view层中自动拼接前缀和后缀，找到对应位置的JSP页面。

1. 表示层：

表示层使用JSP、Ajax、JavaScript、Bootstrap等技术进行开发。JSP用于显示页面，并使用标准标签库简化页面代码，Ajax用于发送异步请求，BootStrap用于进行页面的设计及风格统一。

## 4.3 售货机云平台系统设计

### 4.3.1 系统管理员模块

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。角色由权限构成，一个角色由多个权限构成，呈现一对多的数据关系。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1个月、1年、2年、3年和试用等几个时间单位，软件租用后系统会允许租户免费试用30天。商家在租用页面中选择要租用的时间和空间，系统使用Ajax异步生成租金总额，然后向后台发送一条租户订单信息。自动售货机厂商和运营商都通过该平台进行软件租用。

### 4.3.2 运营商模块

售货机运营商是该SaaS平台的用户种类之一，主要进行售货机业务和运营的管理。一个运营商的用户角色分为五种，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。运营商管理员为运营商商家的一级用户，可以管理系统内部的所有操作；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。

运营商管理员包括多个模块，如SaaS平台租用、缴纳租金、角色管理、用户管理、售货机管理、库存管理、财务管理等。售货机运营商首先进入租用页面进行软件系统的租用，根据不同的租用时间和空间支付相应金额的租金。租用成功后，运营商管理员使用商家编号、用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为其分配角色。为方便管理，管理员可创建用户组，为每个用户组分配组管理员，每个用户组下包含多个普通操作员，负责自动售货机的日常管理操作。管理员可进行售货机的查询、添加和删除，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，方便统一管理，减少工作量。

操作员主要包含以下几种功能：机器管理、货道管理、上货、上缴营业额等。运营商操作员承担着售货机各个货道的上货、更新和管理等操作。操作员登录平台查看机器列列表和详细信息。点击售货机进入货道的列表页面，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进行货道查询。

用户组管理员是操作员的上级用户，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，对组员进行查询、添加和移除等操作。

库存管理员包含以下内容：一是商品管理、二是库存管理。库存管理员进入商品页面对商品进行相应的增删改查操作。库存管理包含多个选项：出库信息查询，对商品的出库状况进行查询和搜索；仓库出货，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实时更新；出库清单，由系统统计一定时段的商品出库信息，生成出库清单表。

财务管理员对运营商的营业额进行管理，用于对订单记录进行查询以及对营业额进行统计和清算。

### 4.3.3 厂商模块

售货机厂商是售货机的提供者，售货机厂商通过首页的租用入口进行软件平台使用权的租用。售货机厂商一般分为厂商管理员和财务管理员两种。厂商管理员是厂商端的一级用户，具有厂商端的最高权限。厂商管理员可根据需求创建用户，并为用户分配角色，负责不同的权限区间。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查看当前合作的运营商，点击添加运营商按钮可以通过复选框添加新的合作运营商。管理员对售货机贩卖类型进行定义，售货机贩卖类型一般包括饮料机、食品机、咖啡机等类型，可对贩卖类型进行增删改查操作。管理员需要进行售货机的信息维护，对售货机进行添加、修改、删除和查询操作，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实货道数量相匹配，同时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，对售货机售出或租用订单进行统计和查询。

## 4.4 Android终端系统设计

除SaaS云平台外，与之配合使用的还有Android终端系统。终端系统分为两类：一类是操作员客户端，供运营商操作员使用，操作员可随时随地进行终端的登录和使用；另一类是售货机Android终端，供消费者使用，消费者通过售货机终端进行商品的购买操作。

1. 操作员客户端

运营商操作员终端的主要服务对象为操作员，操作员维护售货机的日常上货和管理操作，他们查询当前售货机，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，用户通过检测更新选项对当前终端系统版本进行下载更新。

1. 售货机客户端

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，该系统作为自动售货机运营的技术支持，为售货机的正常运营而服务。终端系统为消费者提供选货交互界面，消费者在页面中进行查询和选货，选货成功后生成商品的支付订单。支付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，在商品支付页面选择购买或者赠送，若为商品赠送则生成一个赠品码，凭借赠品码在任意一台售货机上选取同等价格的商品。终端的可视化界面、交互性、易操作性以及方便性更能够适应当前用户的消费习惯。

## 4.5 数据库设计

数据库设是软件开发的基础，与平台开发的好坏息息相关。该平台数据库的分析和设计如下：首先，进行SaaS平台多租户模式的分析和设计；然后，进行数据库的概念结构设计；最后，进行数据库的逻辑设计和数据库表的设计。

1. 数据库多租户设计模式的实现

本文在实现多个租户共享一套系统的同时将各租户间的业务进行隔离，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多个用户共同使用系统；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添加众多用户，租户的内部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、隔离性和操作难易度等多个方面进行考虑。

多租户数据层的实现方式分析如下：

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、安全性和开销较高。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服务提供方，这种模式的数据实现方式虽然有最高的安全性和隔离性，但是却大大提高了成本。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，使用Schema进行个隔离。这种数据库的设计模式使每个数据库可以支持更多的租户，却不适合租户之间的数据统计。

最后，是完全共享的模式。数据库被共用，在表中增加相应的租户Id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。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：自动售货机行业的业务相对较为统一，而软件系统仅作为一个辅助手段希望最大限度的减少成本。通过各个方面的分析，该系统选用第三种模式进行数据库的设计开发。

1. 数据库概念结构设计

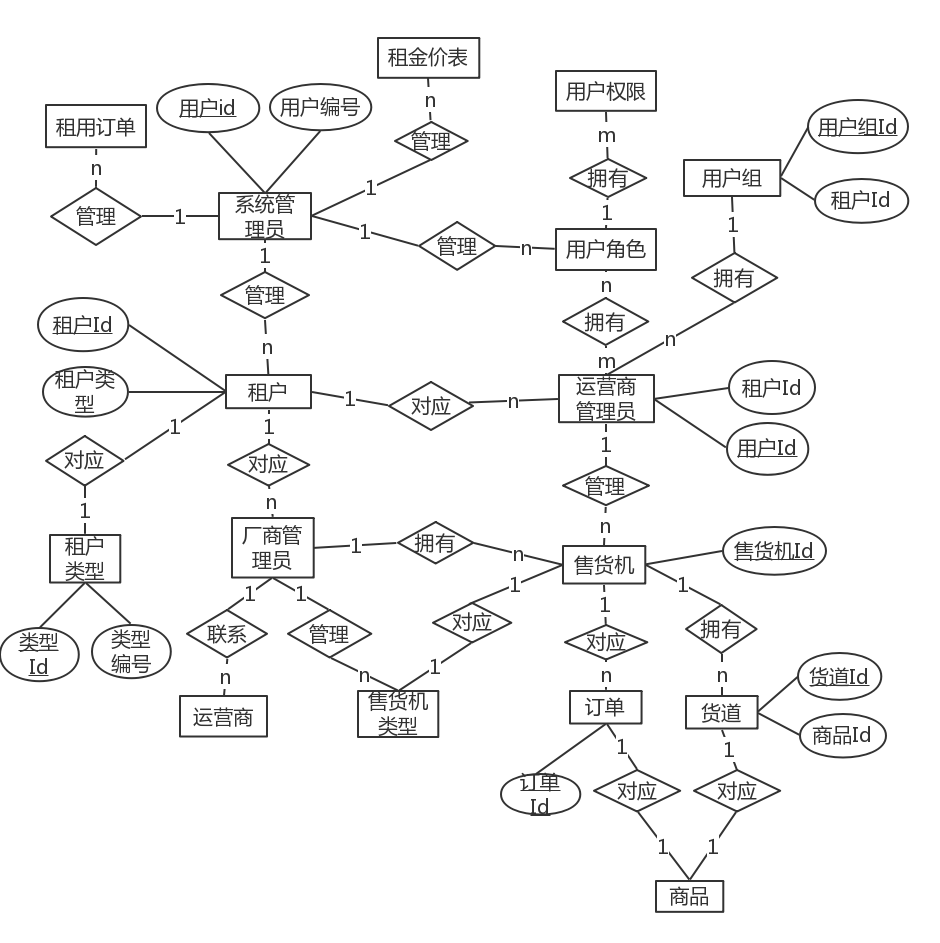
在确定了系统使用的数据库多租户架构模式，以及对系统的需求和功能的分析的提前下，本节提出了一个描述数据实体关系的数据模型，即E-R图（Entity-Relationship Diagram）[[[31]](#endnote-32)]。顾名思义，E-R图描述实体之间的关系以及实体的属性。E-R图实体之间的关系分为1对1、1对多、多对多三种，通过这三种方式能够图形化的描述各个实体之间的关系。图4-4，为自动售货机云平台系统的E-R图。

图4-4 售货机云平台系统的E-R图

1. 数据库逻辑结构设计

该SaaS云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，已用台数，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（Id，编号，名称，描述，所属类型，操作者，操作时间）
  + 用户角色（Id，名称，所属类型，操作者，操作时间）
  + 角色权限表（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（Id，用户Id，角色Id，操作者，操作时间）
  + 分组（分组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（Id，名称，主板号，厂商Id，售货机价格，支持类型，是否售出，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（Id，名称，主板号，类型名称，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（Id，名称，编号，规格，单位，进价，售价，描述，图片，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，交易单号，过期时间）
  + 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）

1. 数据库维护

除使用数据库自身的加密算法保障数据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。一旦出现故障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：

* 敏感数据使用数据库MD5、ENCODE等加密方式进行加密[[[32]](#endnote-33)]。为保证数据库安全，在用户信息存储的过程对用户密码使用MD5算法，MD5加密为不可逆加密，能有效保障用户的账号安全。即使用户密码被获取到，也不能进行解密，增加了安全性。对用户隐私信息，如手机号码，账号信息等使用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，使用DECODE(字段名称,‘加密密文’)进行解密。
* 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

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

# 第5章 云平台系统的实现

本章介绍了基于SaaS的自动售货机云平台管理系统的设计与实现。云平台系统使用SSM（Spring+SpringMVC+Mybatis）框架和Maven进行开发，并使用MySQL数据库和多租户的思想进行数据库的开发和存储。主要使用SaaS软件即服务的思想，结合多租户架构实现了自动售货机厂商和运营商平台的管理模块。

## 5.1 租金模块实现

系统初始化时需对租金规则进行定义，单位租金定制分别为1个月、1年、2年、3年和免费使用几种。其中1年、2年和3年分别设置相应的折扣率，当租用期限少于一年时不提供折扣，当租用期限达到一年及以上可为用户提供相应的折扣。系统管理员对租金的定价方式和折扣进行查看和更新。除按时限定义租金外，还需按照售货机台数进行定义，定义每租用一台售货机的价格。

租金计算公式可定义如下：

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;初次试用时租金为0。

若租户进行续租，应先计算出当前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：

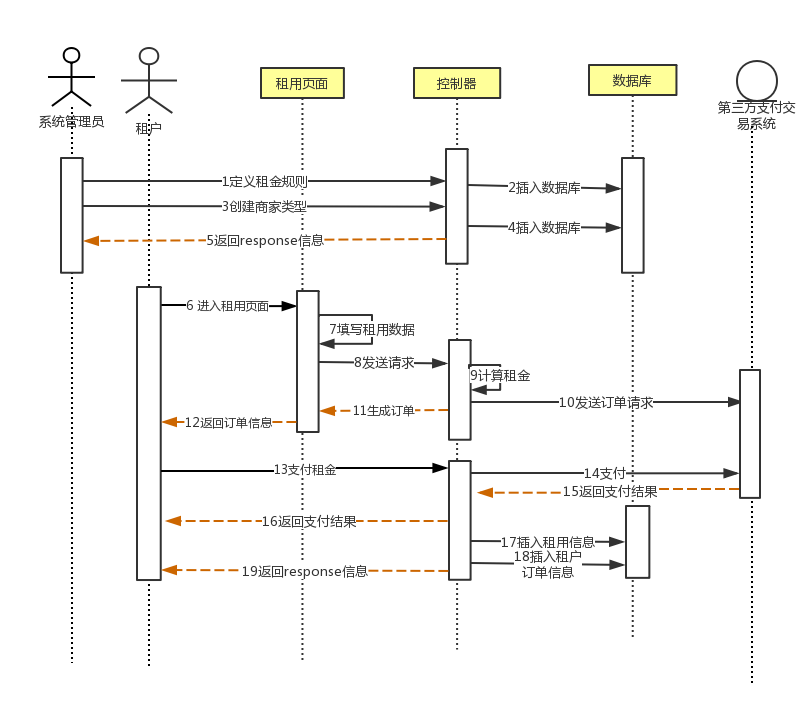
公式（2）为用于计算已支付但未使用的租用期限内的租金金额，若为试用期或者租金到期时间小于当前时间，则剩余租金为0；若租用开始时间到租用截止时间小于12个月，则使用P\*(edTime-now)\*N\*ρ计算剩余租金；若租用开始时间到租用截止时间大于或等于12个月，则使用P\*(edTime-now)\*N\*D\*ρ计算剩余租金，其中增加了一项折扣信息。通过公式（1）和公式（2）可得新的租金计算公式为：

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，随后填写商家名称、商家编号，并初始化管理员密码从而完成用户的创建及租用。租户选择成功后会生成一个订单，在支付成功后即可通过Web访问该平台系统。

数据库加密实现如下：

|  |
| --- |
| 1、加密  INSERT INTO userinfo(userName, password,phone…)VALUE  ('user1', MD5('password'),ENCODE('xxx'，'encode'));  2、查找或解密  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()+"')");  } |

平台租用使用Ajax发送异步请求，当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，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，Ajax发送异步请求，后台查询平台的租金定制表，根据租金的计算公式（1）对当前数量的售货机和租用时限进行计算。租户模块序列图如图5-1。

图5-1 租用过程序列图

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();  } |

如图5-2，为最终实现效果图。

图5-2 租用页面效果图

用户点击提交后，控制层生成租用订单，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，并为商家分配初始管理员账号。每次登录时系统会调用接口进行商家可用性判断，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，返回验证失败提示，提醒用户商家租用期限已到期。

## 5.3 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权限管理。用户的权限管理和角色管理由系统管理员负责，主要负责定义、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，并设置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范围，类型用于区分当前权限是运营商权限或者厂商权限。权限类型在数据库中用数字0、1和2进行表示，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。

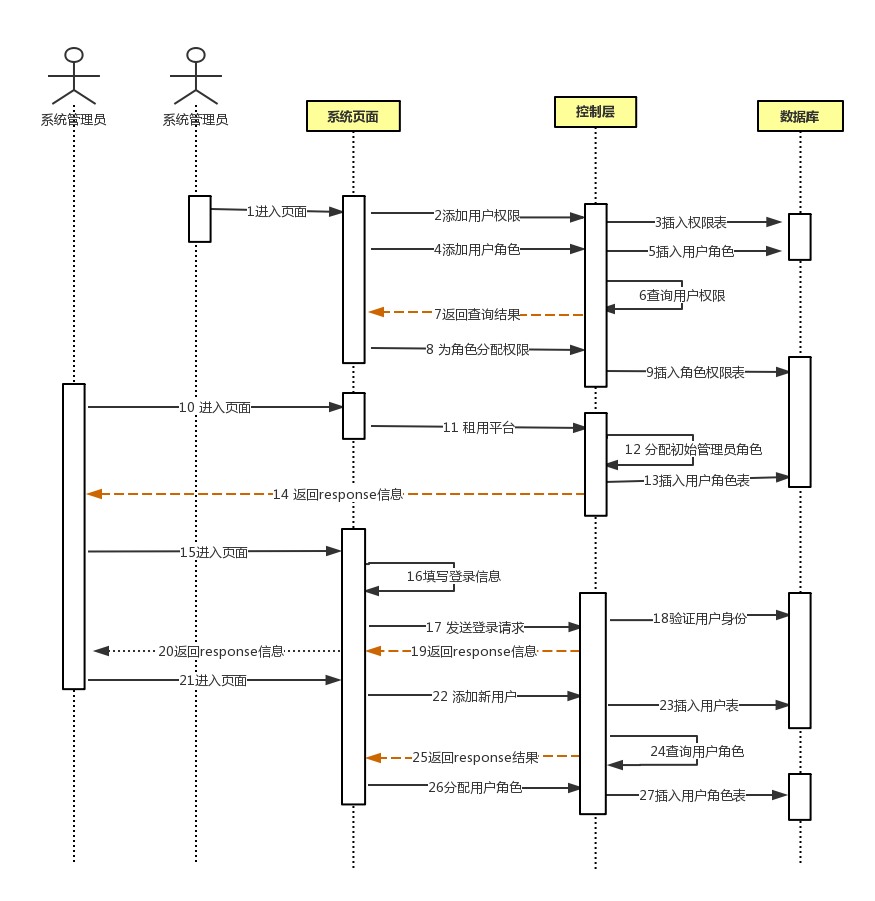
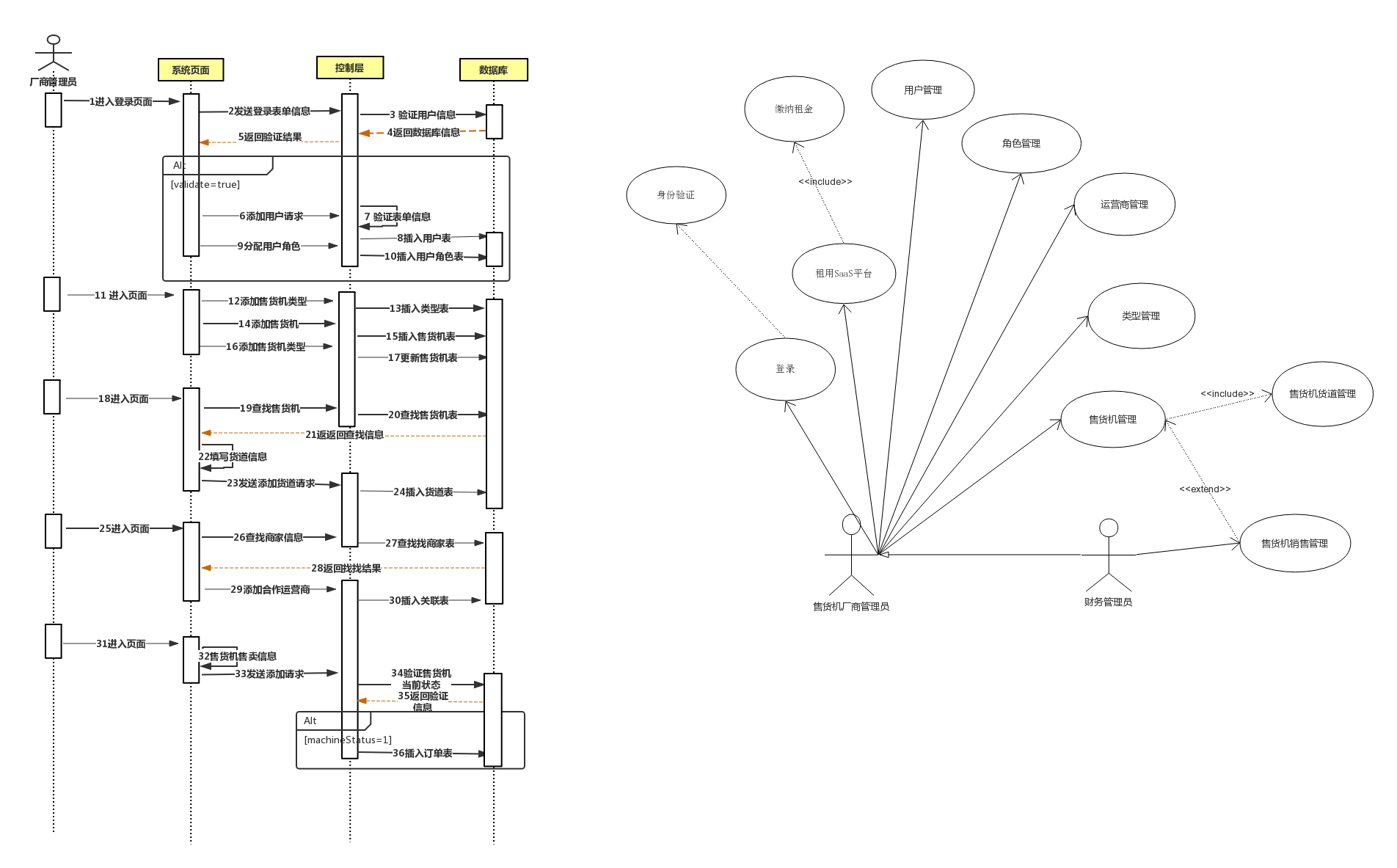
每个用户角色可以拥有多个权限，如运营商超级管理员拥有所有运营商类型的权限，操作员仅拥有操作员权限。系统初始化初期，当系统管理员对用户权限初始化后，可对用户角色进行管理，其中包括用户角色的创建、权限的分配、用户角色删除和修改等操作。用户角色属性包括名称、描述等内容。用户角色权限表中包含了角色Id和拥有的权限Id，当查找用户所拥有的权限时，获取该表中所有用户角色Id对应权限的集合即可。如图5-3，为权限管理模块序列图。

图5-3 权限管理模块时序图

## 5.4 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。登录成功后使用@SessionAttributes("user")注解将用户详细信息放在session中，在其他需要使用的方法中通过@ModelAttribute("user") UserInfo userInfo将user对象传入。

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前缀进行拼接。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，也可以返回一个String类型的view串，还可对返回结果进行redirect重定向。厂商模块时序图如图5-4。

图5-4 厂商模块时序图

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，使用session存储用户权限编码，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类型的Id将售货机和类型进行关联。管理员添加售货机，每个售货机添加对应的商家Id用于数据的隔离，售货机信息被封装成MachineInfo对象，然后使用Ajax发送HTTP请求，调用后台系统的addMachine(MachineInfo machineInfo)方法进行信息的添加。售货机分配时调用getManuMachineStatus()方法查看分配状态，一旦处于被分配状态将予以提示不能被重复售出。

## 5.5 运营商模块实现

运营商功能块包括用户管理、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。

* 运营商超级管理员：

运营商超级管理员拥有租户内的最高权限，能够对用户进行基本数据操作，为用户分配不同的角色，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。管理员创建用户时需对当前用户进行验证，使用boolean repeat = userManagerService.alreadyUser(userInfo)获取验证结果。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。管理员对自动售货机进行添加、修改、删除、查询，管理售货机货道信息，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图5-5。

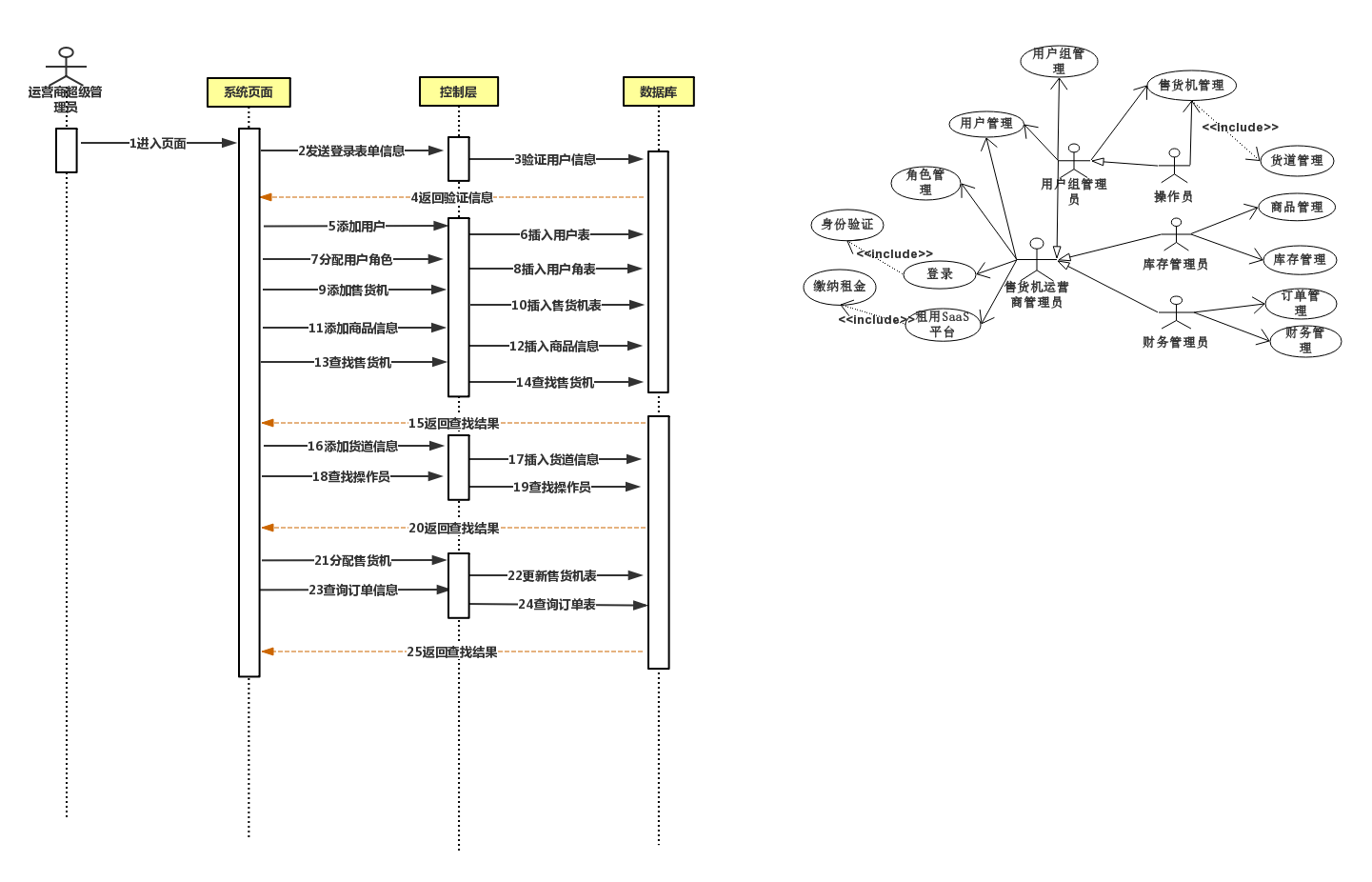
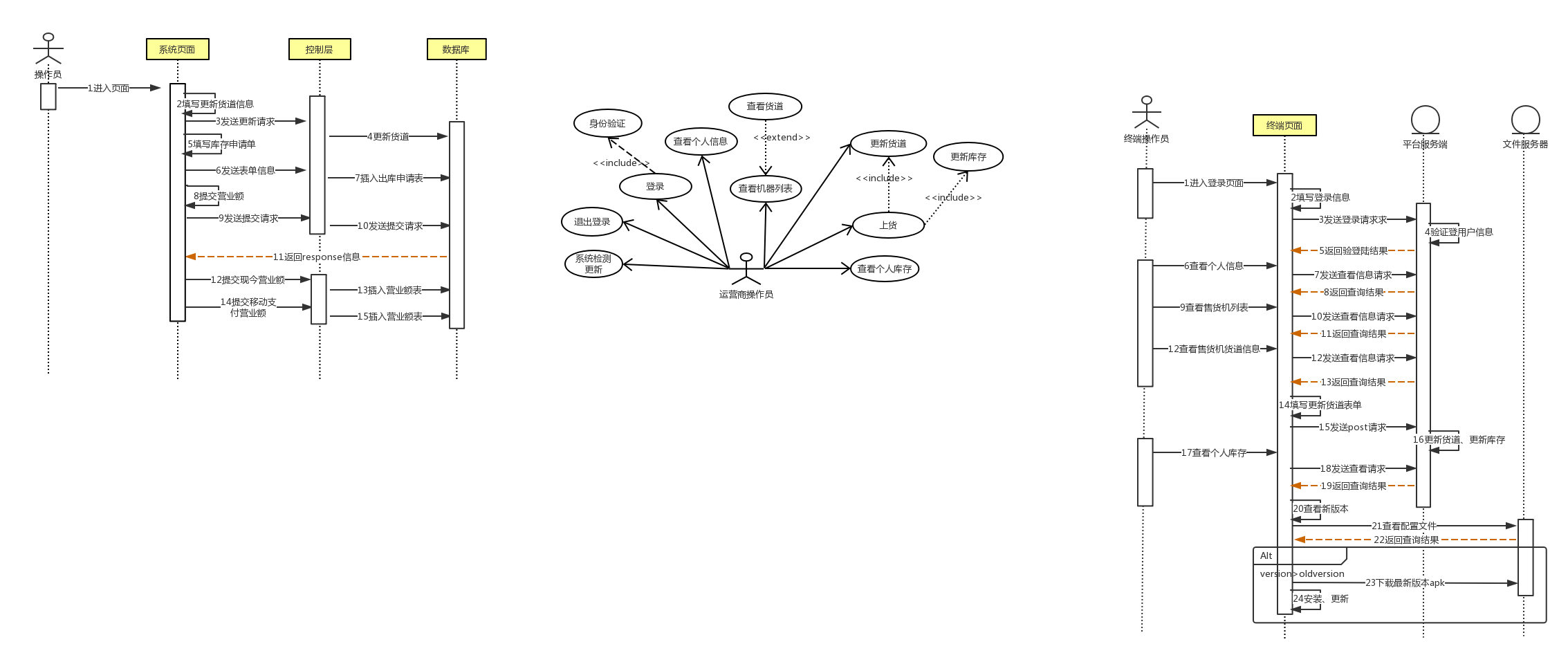
操作员输入登录信息后调用userManagerService.checkFirmStatus(firmInfo)检查当前所属的租户是否可用以及进行用户身份的判定，登录成功后进入租户的内部平台页面。根据当前用户拥有的系统权限，系统将售货机查询、货道管理、库存申请、个人信息查询、更改密码等菜单选项显示出来。在页面中使用if (auths.contains("0010101"))作为操作员判定条件，用于确定要显示的菜单选项。售货机初始化时调用Ajax的addChannelInfo()方法进行货道的添加，此时添加的仅仅为货道的基础信息。当需要将货道和商品关联时调用Ajax的addChannelWare()方法将货道和商品的对应信息插入到货道商品表中，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。管理员更新货道信息时，货道上增加的商品数量，在个人库存中会相应的减少。

图5-5 运营商超级管理员模块时序图

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。

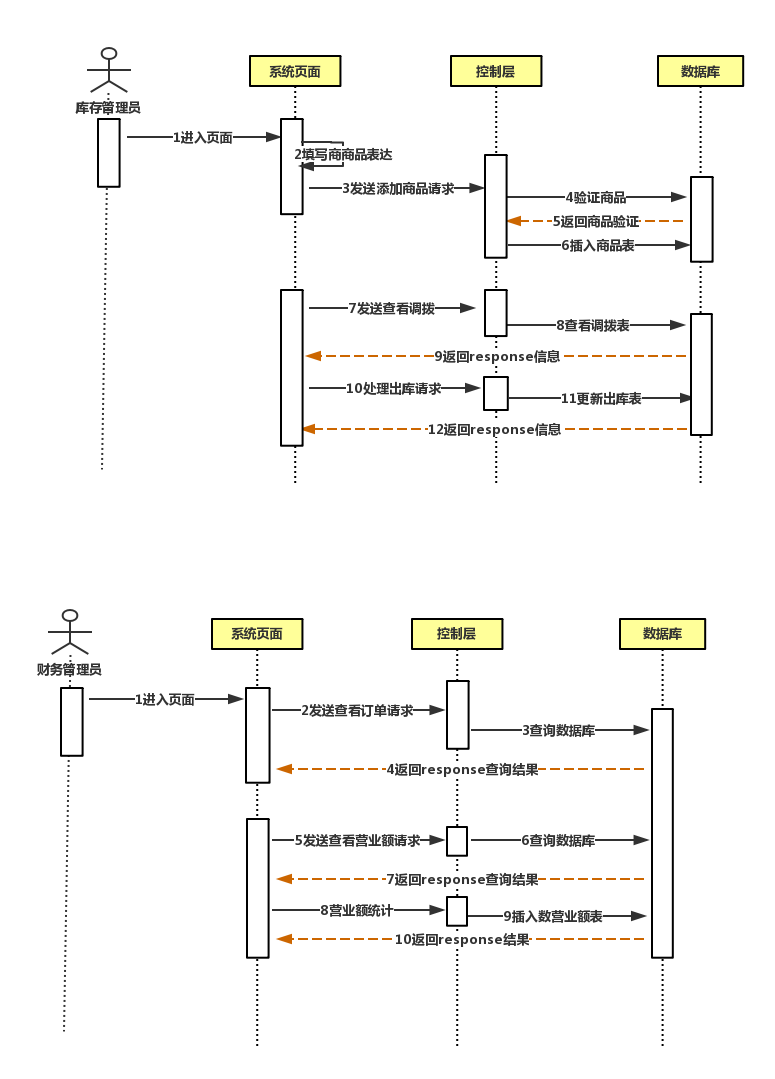
操作员申请库存首先调用getAllWareInfosByFirm()方法获取商家所有商品信息，操作员填写商品信息和商品数量提交表单，通过调用addShipToUser()接口进行个人库存的申请。操作员时序图如图5-6。

图5-6 操作员模块时序图

* 库存管理员：

库存管理员分为商品管理和库存管理两个功能。商品管理模块的功能是对该商家售卖的商品进行基本数据操作。商品管理和商品库存管理模块在控制层代码编写过程中设定的的根请求路径为@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，为操作员新增库存量。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，可将出库处理标记更新为0，此时表示出货成功，对应库存表实时更新。库存管理模块时序图如图5-7。

* 财务管理员：

财务管理员主要负责订单管理和财务管理两项内容。订单管理模块中，财务管理员可调用getAllOrders()接口查看所有订单信息。根据操作员提交的营业额数据，财务管理员可对某台售货机或某个营业员一定期限内的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图5-8。

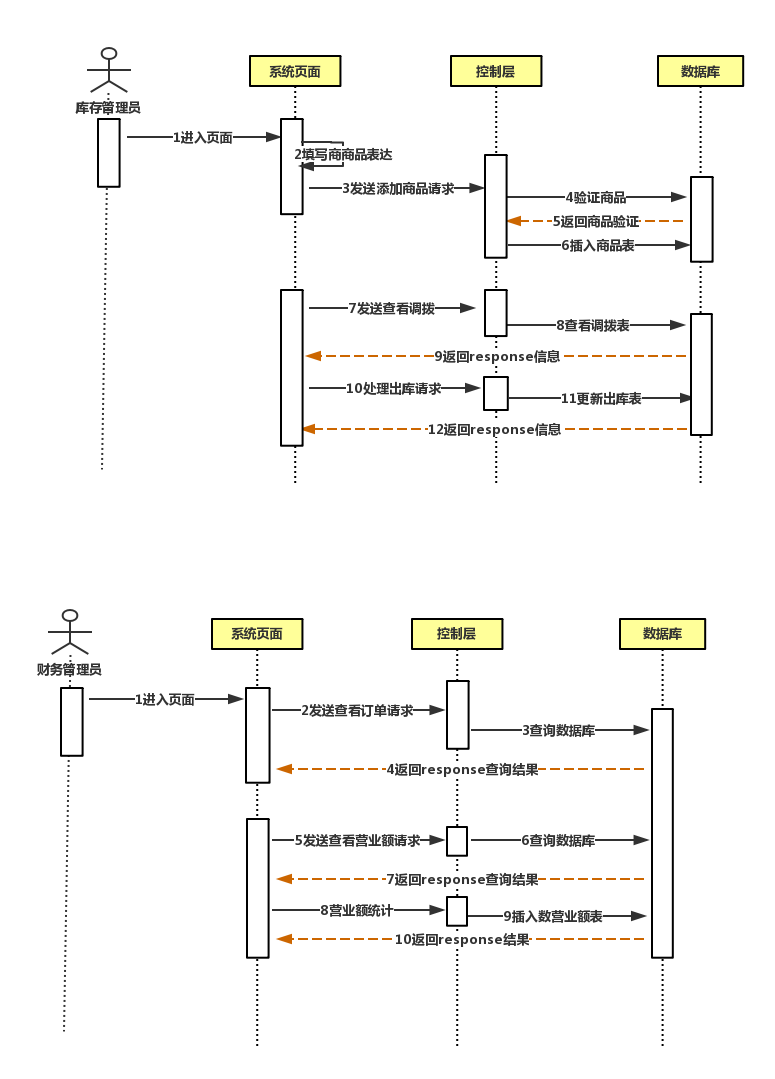
图5-7 库存管理模块时序图

图5-8 财务管理模块时序图

## 5.6 文件服务器

管理系统除提供一般的后台管理之外还需提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文件，用于配置文件服务器使用的目录。配置文件内容如下：

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

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

</Context>

商品图片和终端版本信息存储在文件服务器的目录下，当客户端访问用户图片或视频时，根据获取的资源地址访问文件服务器下的内容。

# 第6章 终端系统的实现

本章主要介绍自动售货机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，另一种为供自动售货机使用的销售终端。终端系统使用Android进行开发，后台管理系统为Android终端提供服务支持。

## 6.1 数据传输加密实现

Android终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，以保证信息安全。数据加密主要使用在用户的信息、密码、支付信息等敏感信息。

系统主要采用了了MD5算法、RSA算法、DES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[[[33]](#endnote-34)]。因为系统不需要知道用户的密码，所以将用户密码进行MD5算法处理后进行存储，查找时只需匹配MD5值即可。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[[[34]](#endnote-35)]。DES为对称的加密算法，使用密钥进行加密解密。

系统综合使用了以上的三种加密算法来保证信息传递的安全。

## 6.2 操作员客户端实现

操作员客户端和后台服务之间通过HTTP协议实现数据的访问和交互，使用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/client的控制接口，请求结束后返回一个通过URLEncode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，获取返回的数据信息。普通的多线程方法无法返回一个实体的对象，但是系统需要通过发送一次耗时请求获取相应的数据，这时FutureTask就可以发挥作用了。使用FutureTask异步执行和Callable结合使用，然后通过FutureTask的get方法调用最终的处理结果，并将其返回[[[35]](#endnote-36)]。

调用HttpURLConnection的postByResponse接口向后台服务器发送POST请求验证用户信息表单，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解析后获取当前用户信息。

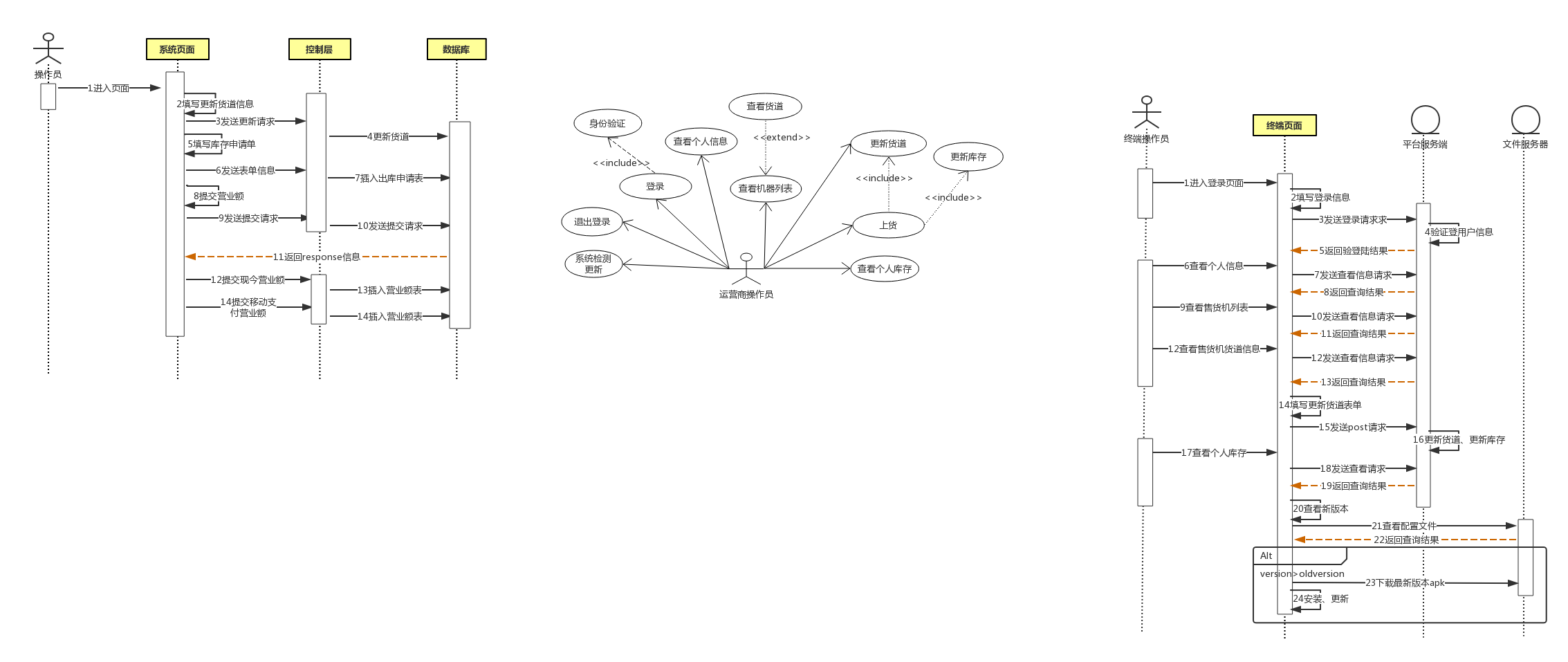
用户登陆成功后使用SharedPreferences将信息存储在本地，存储时设定SharedPreferences的name和mode，然后通过设定的值随时查看自己的个人信息。

通过用户Id和所属的商家Id，使用HttpURLConnection类请求查看管理的售货机及相应的售货机货道信息。操作员将加货量、货道编号、商品Id和售货机Id等填入表单后由系统封装成Map对象，然后发送一个包含Map信息的POST请求给服务器端就能进行货道的更新，同时也将对应的个人库存进行更新。

用户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类的方法请求数据，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APK的版本名和版本号。从文件服务器上下载该APK的版本配置文件信息，使用JSONObject获取响应数据的各个字段值并和本地信息对比，返回比对结果。若配置文件中标识的版本号高于当前本地APK的版本号，则解析出文件中的APK文件路径，调用downloadTask方法下载最新版本的APK文件，然后进行软件安装和覆盖；否则，不作任何处理。

如图6-1，终端操作员模块时序图。

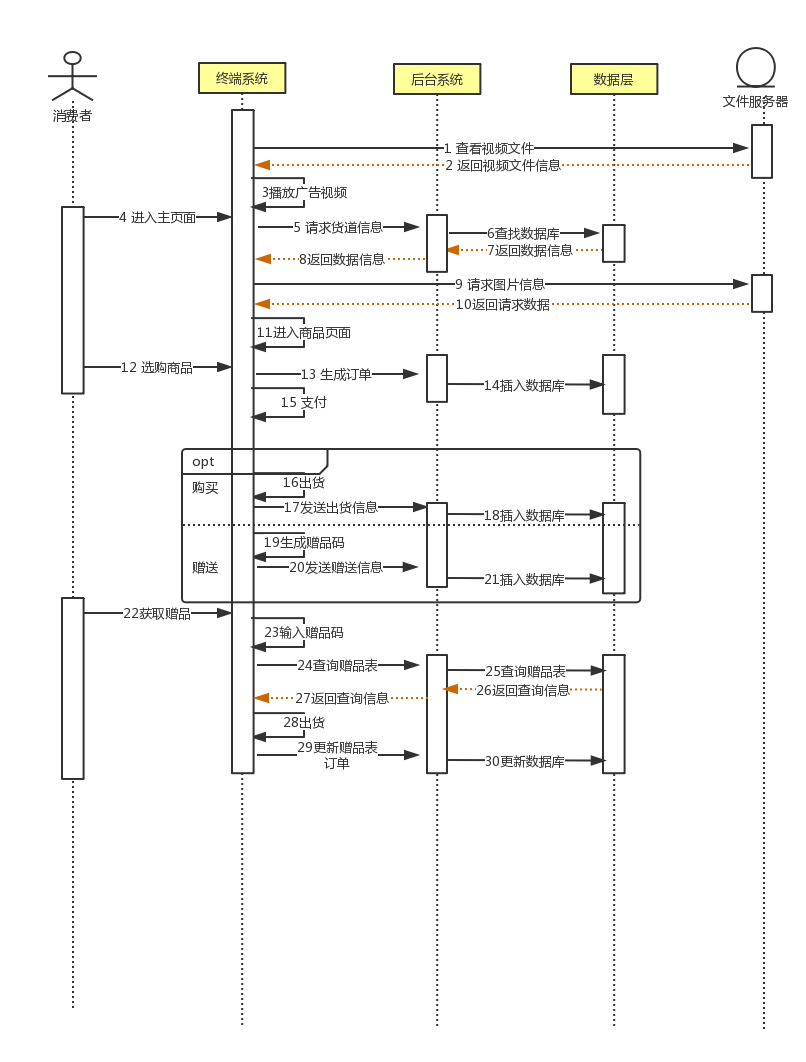
图6-1 终端操作员时序图

## 6.3 售货机终端实现

1. 购物流程的实现

自动售货机终端主要完成购物流程，给消费者展示选购界面，使用Android的GridView展示商品信息。无人使用时，售货机界面轮番播放广告视频，当界面被触摸时进入选货界面。界面中每个货道的商品都陈列在页面中，若商品缺货，则在图片上覆盖红色“缺货”的印戳，若消费者点击缺货的商品图片，终端系统将提示此商品缺货，并继续维持当前页面。选货成功后系统经过HTTPS请求生成支付二维码[[[36]](#endnote-37)]，用户使用手机APP扫描二维码信息进行支付。交易完成后通知售货机出货，并将订单信息发送给后台服务器进行存储，同时更新对应的货道信息。

如图6-2，自动售货机终端时序图。

图6-2 自动售货机终端时序图

* 商品展示页面

商品展示使用Android的 GridView网格视图的功能，activity\_ware.xml布局文件中声明使用GridView进行网格分布，ware\_grid.xml布局文件中设定每个网格中要显示内容的布局和组合，布局中设置GridView的显示列数以及各个分格之间的间隔。

Adapter使用继承自BaseAdapter的方式来实现，让开发者自定义要显示的内容。首先Activity检测本地货道表，之后通过获取GridView布局，并调用Adapter构造函数进行Adapter的创建。Adapter接收Activity传入的JSON格式的本地货道表，将该JSON对象进行解析，获取ware\_grid.xml内容，重写getView()方法，并在相应的位置上放置解析出的商品内容。

描述本地货道的JSON数据中包含了货道中商品的名称、价格、描述和图片路径等信息，该信息放置在上文提到的Apache tomcat文件服务器下，如[http://VendingConfig.IP:VendingConfig.PORT/vendingfile/drinkImages/\*\*.png](http://VendingConfig.IP:VendingConfig.PORT/vendingfile/drinkImages/**.png)为一个图片的路径。

Adapter中重写getView()方法对网格布局中每一个网格显示的内容进行设置。首先，获取gride\_view.xml文件中对应图片位置的ImageView。然后，通过解析JSON对象得到商品图片的路径信息，调用获取图片的BitmapHelper.getBitmapTask(String ImagePath)方法，获取到一个Bitmap对象。最后，使用ImageView.setImageBitmap(Bitmap bitmap)设置当前图片的显示值。获取货道信息后，可得到当前货道内的存货量，若存货量为0，表示当前货道无货，需要将商品展示信息设置为“无货”，同时，货道对应的商品图片上覆盖“无货”标志的红色标签。当所选商品存货量为0时，页面使用Toast提示“此货道商品已售罄，请选择其他货道商品”。

* 多媒体播放

自动售货机终端除进行商品的销售外还可以提供广告位租用，当终端系统处于空闲状态时，可循环播放广告信息。Android中使用VideoView组件实现视频的播放，VideoView组件可以播放本地视频和网络视频。播放本地视频时，把视频源文件放置在Android设备的SD卡中；播放网络视频时，使用setVideoURI(Uri uri)加载当前网络视频资源。因广告资源放置在文件服务器上，属于网络资源，该系统采用第二种视频加载方法播放广告商视频。

服务器端将广告商提供的广告视频文件放置在Apache Tomcat服务器的video目录下，售货机终端先访问该文件服务器的目录，找到该文件路径下的全部视频文件，将视频文件的指向放置在List表中。

首先，在layout配置文件中，使用<VideoView/>标签设置视频播放的布局。

在Activity中设置初始播放文件的下标为0，获取到List中表示的第0个视频路径。然后，使用URI uri = URI.parse(list.get(currentVideo))加载该视频文件。最后，使用VideoView实现视频在页面中播放。为VideoView设置一个监听，当某一视频播放完毕后，调用nextVideo()方法将index检索加一，若已到达最后一个视频，将index设置为0，表示从头播放视频信息，Activity使用index值获取下一个视频的播放地址。

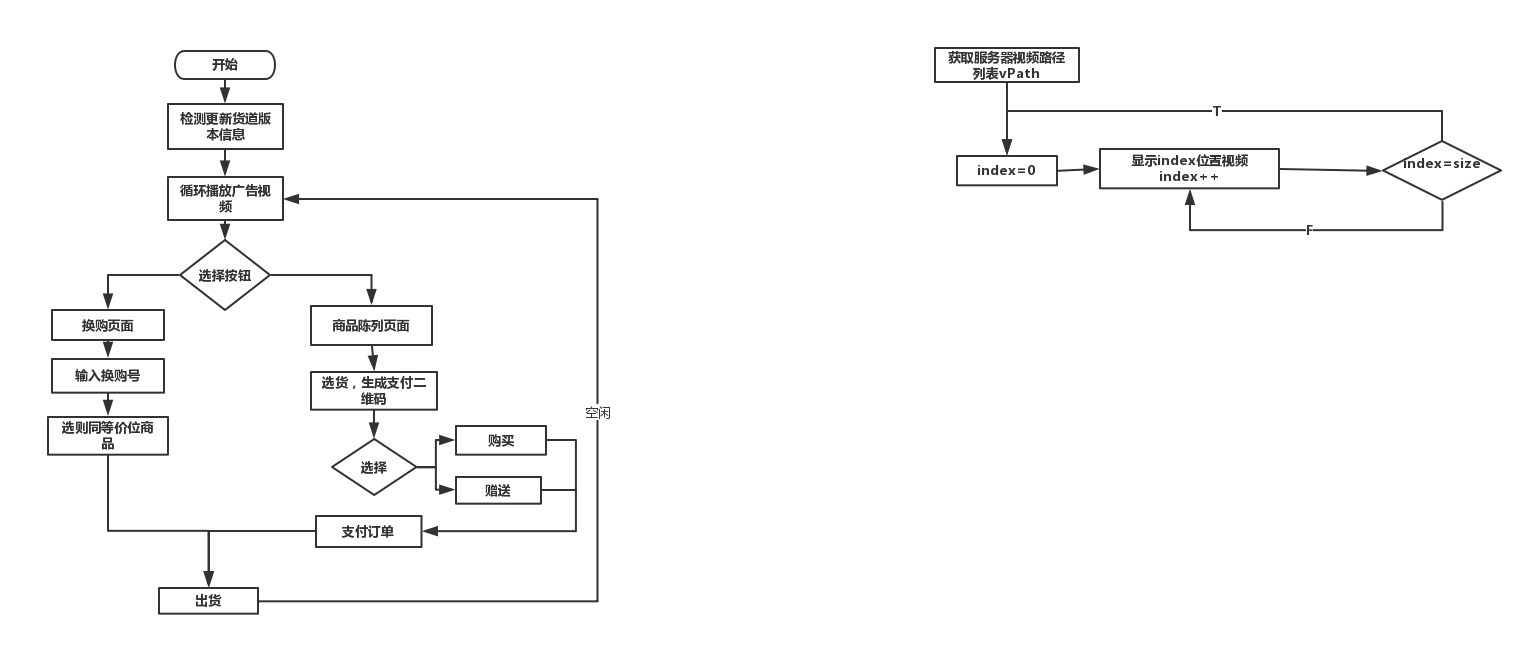
如图6-3，为广告视频播放一般流程。

图6-3 视频播放流程图

* 购物流程

售货机终端系统的主要功能是为消费者提供一个选购和交易的页面，交易完成后控制被选购的商品出货。

系统页面中提供两种购物方式，一种为换购、一种为选购。当选择“选购”按钮时，用户进入选购页面，可选择一件商品并确认当前商品是赠送还是直接购买。若为赠送，则生成一个换购码，进入支付页面进行支付，输入被赠送者的手机号码，将换购码和换购信息以短信形式发送到该手机，不进行出货，然后返回主页；若为“购物”则直接进入支付页面，在用户支付完成后，进行出货。当选择“换购”按钮时，输入换购号，进行出货。

1. 串口通信实现

售货机终端和售货机控制盒之间使用FT321D芯片作为串口转换的芯片进而通过串口信息的传递实现二者通信。终端和售货机控制盒通过串口消息的收发和消息的处理来通知售货机工作。售货机控制盒控制着商品的出货操作，终端系统和控制盒之间使用串行通信方式控制着售货机的运营。串行通信实现了全双工通信，主控板和终端之间可以互相收发信息。

Android设备和售货机之间使用miniUSBFT21D串口转换线进行连接，该串口线提供了供电接口，售货机可为Android设备和串口设备进行持续供电。Android设备使用Open Accessory模式进行开发，Android终端和售货机直接进行数据的交互。Android设备和售货机之间通信需要特定的通信协议和串口指令，对接收的指令进行解析，并将要发送的信息进行封装之后再进行传输，以此实现二者之间的信息传递。

新版自动售货机抛弃原始的物理选货按键，售货机内货物结构可不再向用户展示，原始的按钮盒窗格被替换。商家可将售货机窗口封闭、在原先的窗口中张贴宣传海报，或者嵌入显示屏。此时售货机的销售工作交由Android设备，Android设备联接4G网络，Android设备主页中展示各个货道中售卖的商品、价格以及名称等信息。消费者通过Android设备进行选货，根据选货结果生成支付二维码，消费者扫描终端生成的二维码进行交易。交易结束后，终端系统将订单信息、价格信息、货道信息等封装成串口指令进行发送。控制盒读取串口信息，解析信息确定当前需要出货的货道，进行出货。省略了物理按钮的售货机，从成本和开发两个方面考虑都更加简洁、便于维护，同时屏幕显示购物列表更符合当前大众的购物习惯，实现售货机终端的进一步变革。

Android在AndroidManifest中设置系统对USB的读取权限，获取UsbManager对像用于管理设备。使用ComService类初始化串口，首先设置串口的读取的参数，设置串口参数如下：波特率：9600；数据位：8位；停止位：1位；校验位：无。消息报文格式如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 描述 | 类型 | 长度 |
| STX | 帧起始符 0x06 | **Byte** | 1 |
| 报文头 |  | **Byte** | 1 |
| 报文内容（Data） | 消息报文正文 | **Block** | 可变 |
| XOR校验码 | 校验内容包括报文头和报文内容的异或值 | **Byte** | 2 |
| ETX | 帧结束符 0x07 | **Byte** | 1 |

在消息报文中，为避免混乱和产生歧义，除帧起始符和帧结束符之外，其他报文中不可以出现STX或ETX保留字。传输过程中使用转义字符0x10进行转义，除起始符和结束符外，如碰到0x06则自动替换成0x10，0x06；如碰到0x07则替换为0x10，0x07。同样，当收到报文时需对收到的消息进行去转义，如碰到0x10，0x06，则将其替换成0x06，碰到0x10，0x07则替换成0x07。消息报文包括STX、报文头、报文内容、XOR校验码、ETX等五个部分的内容，其中STX和ETX为开始和结束的标志，报文内容为主要发送的信息，报文内容包括4Byte的会话流水号和可变长度的数据位。通过读取串口信息，对信息进行校验，解析指令，进行处理后，发送相应的应答。订单支付成功后，发送串口信息，进入出货界面。串口通信过程流程图如图6-5.

读取串口信息时，一般分为以下流程进行处理：

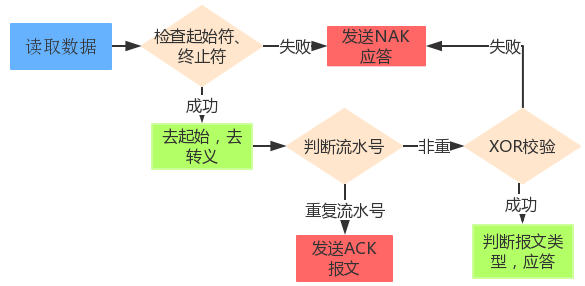
* 检查信息的起始符和终止符，若出错，则退出当前函数，否则进入下一步。
* 将起始符和终止符去除，并删除转义符，再进行之后的验证。
* 获取报文类型和流水号，若为重复报文，则直接发送ACK报文，否则进入下一步。
* 对信息进行XOR校验，检验包括报文头和报文内容的异或值，若校验失败，则发送NAK报文，否则进入下一步。
* 解析报文类型。

图6-5 串口通信流程图

1. 移动支付功能实现

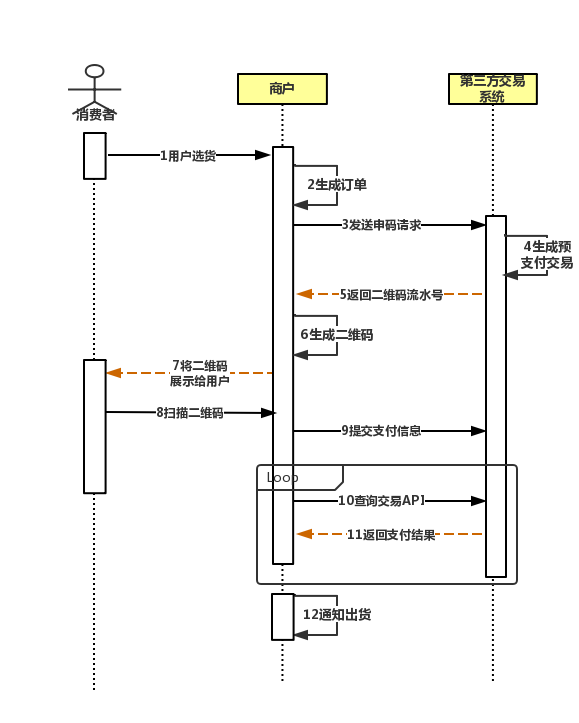
新型自动售货机新增二维码移动支付功能。二维码支付非常适用于消费者和商家之间的这种小额支付，它分为“主扫”和“被扫”模式。商家根据选购商品信息生成支付二维码，属于“被扫”方；消费者使用手机APP扫描商户提供的二维码，属于“主扫”方。该系统采用了银联二维码支付、支付宝二维码支付和微信二维码支付，大体实现过程时序图如图6-6。

图6-6 移动支付时序图

* 银联二维码支付

银联二维码支付流程有以下几步：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的规则对版本号、编码方式、签名方法、交易方式、交易类型等进行配置。由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，并将订单信息、价格信息、订单发送时间、超时时间和数据元一起封装成一个Map对象。对该Map对象进行签名，通过HttpURLConnection类向银联全渠道交易系统发送请求，同步返回请求结果。然后，请求结果为一个包含二维码信息的JSON对象，将该JSON对象进行解析生成一个Map<String,String>对象，对该对象进行签名的验证，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。

支付二维码动态生成，每个二维码都设定一个固定的超时时间，超过设定的超时时间后该二维码失效，不能再进行支付；被扫二维码不能进行重复交易，保证订单的唯一性。

银联向商家提供公钥和私钥证书，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，才予以继续下面的步骤。签名过程如下：首先，Map字段中除key为signature的字段之外的所有其他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摘要，使用私钥证书中的私钥对字符串进行签名。最后，将签名后的字符串进行Base64编码放置到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。验签过程和签名过程类似，不同的是验签时使用公钥证书。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"  } |

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。请求过程如下：首先，对请求字段进行签名。然后，对获取结果进行验签。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。

* 支付宝二维码支付

支付宝二维码支付和银联支付相类似，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。

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、二维码失效时间等进行过配置，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规则对其他参数进行相应的配置，组成一个POST请求数据包。请求参数使用key/value模式进行存储。其中包含一个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，将字符串使用私钥和RSA算法进行签名，并使用Base64进行编码，签名结果放置在sign字段，该请求字符串发送给支付宝服务。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。签名使用Base64进行解码，使用RSA算法和支付宝提供的公钥进行签名的验证，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。售货机终端系统再使用二维码生成类将qrCode转换成二维码显示在页面中。用户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和其他数据元信息，封装成一个key/value的数据包发送。支付宝端返回一个JSON格式的字符串，分析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。

* 微信二维码支付

微信支付采用HTTPS的传输模式，使用POST发送请求，它采用XML格式进行请求的提交和返回。此外，微信使用MD5和SHA等算法对请求对象进行签名。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、二维码失效时间、随机字符串nonce\_str和其他数据元进行设置。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的进行表示，每一对数据使用&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请求串。该信息使用MD5算法进行运算，运算结果全部转换为大写字符，并将最终得到的签名结果放置在sign字段的value位置上。然后，将String串转换为XML格式，并以POST方式将该请求信息送给微信端。最后，微信端返回一个XML的响应数据，当return\_code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。使用二维码生成类，将生成的二维码显示在页面中。

商家查询交易状态需要传入微信订单号和其他基本信息配置。首先，按照申码的请求数据封装过程将请求字段进行MD5签名，将请求域转换为XML格式，以POST方式发送给微信端。然后，服务端接收请求参数进行数据处理，并将结果信息以XML格式返回。最后，解析响应数据，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。

1. 应用版本更新

售货机客户端的软件更新往往没有人为的手动点击和干涉，因售货机终端分布在不同的区域，每次新版本软件开发完成后不可能由操作员逐个手动检测软件版本和更新软件，因此售货机终端的更新检测采用静默的更新和安装。

管理员在文件服务器中放置软件的最新版本和包含软件新版本信息的JSON文件。使用另一个Android小应用对售货机终端进行检测更新。系统设置一个定点的时间，使用Service后台服务每天按时进行软件版本的检测和新版本的下载，下载完成后调用系统的安装程序完成安装。

更新应用使用Android提供的无障碍服务进行智能安装，Android使用AccessibilityService来模拟用户的点击操作。在accessibility\_service\_config.xml文件中设置要监听的程序和安装页面，并在AndroidManifest.xml文件中配置无障碍服务。使用AccessibilityService捕获安装页面，并模拟点击“安装”按钮进行软件的安装。这样的一个安装过程并不需要用户的手动操作，仅仅在每天的定时时间通知系统服务进行检测，然后模拟用户行为进行更新和重启。

# 第7章 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。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。

云平台系统采用Spring、SpringMVC、MyBatis框架，Maven构建工具、Mysql数据库和Bootstrap、JQuery、Ajax和三层架构的思想实现分层开发系统使用共享数据库，对数据库采用多种加密方法，增强数据安全性。

终端系统采用Android进行开发，分为两个类型：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。系统提供二维码扫码支付，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。

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，用户可直接通过网页入口访问平台系统。过程便捷，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，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，数据库中对用户或其他敏感信息进行加密操作，防止信息的泄露。
* 经济性：系统由软件提供商进行开发、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，一次开发可供多个租户使用，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，操作员不需手动手动记录和誊抄货道的更新状况，直接使用手机实时更新货道变更信息。

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，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。系统使用共享数据库模式，共享程度高，因此对安全性的要求也很高。此外，还需对该行业进行进一步的研究，根据新的调查和对新技术的研究对系统进行更新和完善。总体来看，该系统满足当前售货机行业的需求，能够以最低廉的成本和最少的耗时完成行业内部的信息化。

# 参考文献

1. []Brown G. Automated vending machine system for recorded goods: US, US5445295[P]. 1995. [↑](#endnote-ref-2)
2. []白丽. 自动售货机:第三次零售业革命[J]. 电子商务, 2005(3):64-66. [↑](#endnote-ref-3)
3. []马涛. 我国移动支付业务发展分析[J]. 金融科技时代, 2005, 13(3):9-10. [↑](#endnote-ref-4)
4. []余世明, 晁岳磊, 缪仁将. 自动售货机研究现状及展望[J]. 中国工程科学, 2008, 10(7):51-56. [↑](#endnote-ref-5)
5. []詹昌平, 金瓯. 基于移动支付的自动售货机[J]. 现代电子技术, 2004, 27(17):38-40. [↑](#endnote-ref-6)
6. []张书利. 4G环境下的移动电子商务模式研究和创新[D]. 山东师范大学, 2014. [↑](#endnote-ref-7)
7. []陈荆花, 王洁. 浅析手机二维码在物联网中的应用及发展[J]. 电信科学, 2010, 26(4):39-43. [↑](#endnote-ref-8)
8. []李森. 浅析基于SaaS架构的多租户技术[J]. 电子设计工程, 2013, 21(20):41-44. [↑](#endnote-ref-9)
9. []刘国萍, 刘建峰, 谭国权. 多租户SaaS服务安全技术研究[J]. 电信科学, 2011(S1):11-15. [↑](#endnote-ref-10)
10. []何海棠, 朱晓辉, 陈苏蓉. SaaS模式下多租户数据库的研究[J]. 郑州铁路职业技术学院学报, 2012(3):31-33. [↑](#endnote-ref-11)
11. [] Cusumano, Michael. Cloud computing and SaaS as new computing platforms[J]. Communications of the Acm, 2010, 53(4):27-29. [↑](#endnote-ref-12)
12. [] Bezemer C P, Zaidman A. Multi-tenant SaaS applications:maintenance dream or nightmare?[C]// Joint Ercim Workshop on Software Evolution. DBLP, 2010:88-92. [↑](#endnote-ref-13)
13. [] Aulbach S, Grust T, Jacobs D, et al. Multi-tenant databases for software as a service:schema-mapping techniques[C]// ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, Bc, Canada, June. DBLP, 2008:1195-1206. [↑](#endnote-ref-14)
14. []Zhang D, Wei Z, Yang Y. Research on Lightweight MVC Framework Based on Spring MVC and Mybatis[C]// Sixth International Symposium on Computational Intelligence and Design. IEEE Computer Society, 2013:350-353. [↑](#endnote-ref-15)
15. []王建国, 王建英. Struts+Spring+Hibernate框架及应用开发[M]. 清华大学出版社, 2011. [↑](#endnote-ref-16)
16. []Yuan X F. AOP Based on Spring Framework[J]. Computer & Modernization, 2006. [↑](#endnote-ref-17)
17. []徐彩云. 用TestCase对Spring的Bean进行单元测试[J]. 电脑知识与技术, 2011, 07(16):3881-3883. [↑](#endnote-ref-18)
18. []陈雄华, 林开雄. Spring 3.x企业应用开发实战[M]. 电子工业出版社, 2012. [↑](#endnote-ref-19)
19. []Mudunuri S. Mybatis in Practice: A Step by Step Approach for Learning Mybatis Framework[J]. 2013. [↑](#endnote-ref-20)
20. [] Mudunuri S. Mybatis in Practice: A Step by Step Approach for Learning Mybatis Framework[J]. 2013. [↑](#endnote-ref-21)
21. []Zhang D, Wei Z, Yang Y. Research on Lightweight MVC Framework Based on Spring MVC and Mybatis[C]// Sixth International Symposium on Computational Intelligence and Design. IEEE, 2014:350-353. [↑](#endnote-ref-22)
22. []宫志方, 程林, 杨培强. 一种基于Spring和MyBatis的MVC框架:, CN 105843609 A[P]. 2016. [↑](#endnote-ref-23)
23. []杨潇亮. 基于安卓操作系统的应用软件开发[J]. 电子制作, 2014(19):45-46. [↑](#endnote-ref-24)
24. []郭霖. 第一行代码[M]. 人民邮电出版社, 2014. [↑](#endnote-ref-25)
25. []曾健平, 邵艳洁. Android系统架构及应用程序开发研究[J]. 微计算机信息, 2011(9):1-3. [↑](#endnote-ref-26)
26. []董晓刚. 浅析Android系统的四大基本组件[J]. 中国电子商务, 2013(1):39-39. [↑](#endnote-ref-27)
27. []纪晓阳. 线程在Android开发中的应用[J]. 软件, 2013(8):24-26. [↑](#endnote-ref-28)
28. []尹京花, 王华军. 基于Android开发的数据存储[J]. 数字通信, 2012, 39(6):79-81. [↑](#endnote-ref-29)
29. []赖均. 软件工程[M]. 清华大学出版社, 2016. [↑](#endnote-ref-30)
30. []胡荷芬. UML系统建模基础教程[M]. 清华大学出版社, 2014. [↑](#endnote-ref-31)
31. []王珊, 萨师煊. 数据库系统概论.第4版[M]. 高等教育出版社, 2006. [↑](#endnote-ref-32)
32. []王晓峰, 王尚平. 数据库加密方法研究[J]. 西安理工大学学报, 2002, 18(03):263-268. [↑](#endnote-ref-33)
33. []魏晓玲. MD5加密算法的研究及应用[J]. 信息技术, 2010(7):145-147. [↑](#endnote-ref-34)
34. []刘传领, 范建华. RSA非对称加密算法在数字签名中的应用研究[J]. 通信技术, 2009, 42(03):192-193. [↑](#endnote-ref-35)
35. []方腾飞, 魏鹏, 程晓明. Java并发编程的艺术[M]. 机械工业出版社, 2015. [↑](#endnote-ref-36)
36. []崔莹. 手机二维码支付应用技术和发展概述[J]. 电脑知识与技术, 2013(4):945-947. [↑](#endnote-ref-37)