|  |  |
| --- | --- |
| 学校代码： | 10269 |
| 学 号： | 51151201090 |

2018届研究生硕士学位论文



**自动售货机云平台的设计与实现**

|  |  |
| --- | --- |
| 院 系： | 计算机科学与软件工程学院计算机系 |
| 专 业： | 计算机技术 |
| 领 域： | 复杂信息处理 |
| 指导教师： | 章炯民 副教授 |
| 论文作者： | 任南南 |

2017年10月完成

Dissertation for Master Degree, 2018 School Code: 10269

No: 51151201090

**East China**

**Normal University**

**Design and implementation of vending machine**

**cloud platform**

|  |  |
| --- | --- |
| Department: | Information Science and Technology |
| Major: | Computer Technology |
| Research Area: | Complex Information Processing |
| Supervisor: | Associate Prof. Jiongmin Zhang |
| Student Name: | Nannan Ren |

October, 2017

**华东师范大学学位论文原创性声明**

郑重声明：本人呈交的学位论文《自动售货机云平台的设计与实现》，是在华东师范大学攻读硕士/博士（请勾选）学位期间，在导师的指导下进行的研究工作及取得的研究成果。除文中已经注明引用的内容外，本论文不包含其他个人已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中作了明确说明并表示谢意。

**作者签名： 日期：** 年 月 日

**华东师范大学学位论文著作权使用声明**

《自动售货机云平台的设计与实现》系本人在华东师范大学攻读学位期间在导师指导下完成的硕士/博士（请勾选）学位论文，本论文的研究成果归华东师范大学所有。本人同意华东师范大学根据相关规定保留和使用此学位论文，并向主管部门和学校指定的相关机构送交学位论文的印刷版和电子版；允许学位论文进入华东师范大学图书馆及数据库被查阅、借阅；同意学校将学位论文加入全国博士、硕士学位论文共建单位数据库进行检索，将学位论文的标题和摘要汇编出版，采用影印、缩印或者其它方式合理复制学位论文。

本学位论文属于（请勾选）

（ ）1.经华东师范大学相关部门审查核定的“内部”或“涉密”学位论文\*，

于 年 月 日解密，解密后适用上述授权。

（ ）2.不保密，适用上述授权。

导师签名 本人签名

年 月 日

\* “涉密”学位论文应是已经华东师范大学学位评定委员会办公室或保密委员会审定过的学位论文（需附获批的《华东师范大学研究生申请学位论文“涉密”审批表》方为有效），未经上述部门审定的学位论文均为公开学位论文。此声明栏不填写的，默认为公开学位论文，均适用上述授权）。

**硕士学位论文答辩委员会成员名单**

|  |  |  |  |
| --- | --- | --- | --- |
| 姓 名 | 职 称 | 单 位 | 备 注 |
| 孙蕾 | 副教授 | 华东师范大学计算机系 | 主席 |
| 钱莹 | 副教授 | 华东师范大学计算机系 | / |
| 周建武 | 高级工程师 | 中安电子信息科技有限公司 | / |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 摘要

（云的问题写的较少，可在系统设计上进行完善，尤其是云。还有些可在将来进行完善。云平台多点观点和讨论。从云的角度去写，功能的设计和实现淡化，主要写出云平台和SaaS的实现。

文章的组织，文句不同，说法不合理，句子之间的关联性，段和段之间的联系，有跳跃，应该加上过渡语句。软件测试可写可不写，除非是用了某些测试工具，比如压力测试，可稍微提一下。

实际碰到的问题可以写一下，安卓程序运行有时莫名中断和跳出，重启，要加上原因。自动重启解决的问题就是安卓程序莫名跳出、崩溃，因为远程，可能用户触摸力较大导致震动，连接线会松动。）

（避免重复、遗漏）

早期，国内人力成本低廉、售货机价格高昂且系统落后，这些因素导致售货机的发展举步维艰。近些年来，劳动力成本大幅上升，自动售货机成本逐渐下降。据中国产业信息网统计，自2004年起，中国劳动力成本每年增长13%。高昂的人力成本给售货机行业的运营和发展带来了巨大的负担，为减少人力资源的使用，运营商可将售货机连网，实现售货机的信息化管理和售货终端。然而，目前有60%的售货机运营商为中小型企业，其并无足够的财力和物力搭建一套完善的售货机管理系统。因此，本文拟采用基于SaaS（Software-as-a-Service）的自动售货机云平台的设计方案，解决或改善中小型运营商的以上问题。平台由生产厂商统一管理，供运营商使用。厂商为运营商提供低成本服务，提高商家的管理效率，同时也增强了自身的竞争力。

该平台面向两类用户使用：厂商和运营商。厂商对用户权限、运营商、售货机和租赁等进行基本的管理操作；运营商对运营商内部用户、售货机、商品和库存等进行基本的管理操作。此外，该平台还提供对应的操作员APP和售货APP，操作员APP供运营商操作员使用，售货APP由普通终端用户操作。操作员可通过操作员APP现场更新售货机数据，消费者可通过售货APP进行商品选购并完成移动支付。

自动售货机云平台基于SaaS模式，结合多租户框架，采用JavaWeb分层思想设计实现。云平台后台系统使用SSM (Spring+ SpringMVC+Mybatis)框架，前台界面使用JSP、Ajax等技术。操作员APP和售货APP基于Android系统开发，其中，售货APP通过miniUSBFT1D串口转换线与设备主控板通信，实现串口通信功能。

**关键词：**自动售货机、SaaS、多租户、SSM框架、Android

# ABSTRACT

In the early stage, the domestic labor cost was low, while the vending machine price was high and the system was backward, which led to the difficult development of vending machines. In recent years, the cost of labor has risen sharply, and the cost of vending machines has gradually declined. According to the China industrial information network statistics, China's labor costs increased by 13% per year since 2004. High labor costs brought huge burden to the operation and development of vending machine industry. In order to reduce the use of human resources, business operators can connect vending machines to the internet, and implement the vending machine management and sales terminals. However, the small and medium-sized vending machine business operators account for more than 60% of the market, but they does not have enough financial and material resources to build a complete vending machine management system. Therefore, this paper adopts the design scheme of vending machine cloud platform based on SaaS (Software-as-a-Service) to solve or improve the above problems of small and medium business operators. The platform is managed by the manufacturer and used by the operators. The manufacturer provides low cost service for the operators, and improves the efficiency of the management, and enhances the competitiveness of the business.

The platform is designed for two types of users: manufacturers and business operators. Manufacturers can manage user permission, business operators, vending machines, leases and other basic operations. Business operators can manage users, vending machines, commodities, inventory and other basic management. In addition, the platform also provides corresponding Operator APP and Sale APP, Operator APP is used by operators, and Sale APP is operated by ordinary end users. The operator can update the vending machine data through the Operator APP. The consumers can choose goods through the Sale APP and complete mobile payment.

Design and implement of vending machine cloud platform is based on SaaS mode, multi-tenant framework and JavaWeb layering idea. The cloud platform background system uses SSM(Sping+SpringMVC+Mybatis) and the front end adopts JSP, Ajax and other technologies. Operator APP and Sale APP are based on the Android system, in which the Sale APP communicates with the Main board through the miniUSBFT1D serial port conversion line, and realizes the serial communication function.

**Keywords:** vending machine, SaaS, multi-tenant, SSM framework, Android

目录

[摘要 I](#_Toc497837707)

[ABSTRACT II](#_Toc497837708)

[目录 IV](#_Toc497837709)

[第1章 引言 1](#_Toc497837710)

[1.1 研究背景及意义（每小节标题，目录具体化，不要用模板） 1](#_Toc497837711)

[1.2 研究现状 2](#_Toc497837712)

[1.3 主要研究内容 3](#_Toc497837713)

[1.4 论文的组织结构（可使用模板） 3](#_Toc497837714)

[第2章 相关技术概述 5](#_Toc497837715)

[2.1 多租户和SaaS服务 5](#_Toc497837716)

[2.2 SSM框架 8](#_Toc497837717)

[2.3 Android应用 11](#_Toc497837718)

[第3章 （\*\*系统的）需求分析 14](#_Toc497837719)

[3.1 系统目标 14](#_Toc497837720)

[3.2 可行性分析 14](#_Toc497837721)

[3.3 需求分析 15](#_Toc497837722)

[3.3.1 自动售货机云平台 15](#_Toc497837723)

[3.3.2 Android终端 20](#_Toc497837724)

[第4章 云平台系统设计 24](#_Toc497837725)

[4.1 系统总体架构 24](#_Toc497837726)

[4.2 系统详细设计 25](#_Toc497837727)

[4.3 数据库设计 28](#_Toc497837728)

[第5章 云平台系统的实现 35](#_Toc497837729)

[5.1 租金模块实现 35](#_Toc497837730)

[5.3 系统权限实现 38](#_Toc497837731)

[5.4 厂商模块实现 38](#_Toc497837732)

[5.5 运营商模块实现 40](#_Toc497837733)

[5.6 文件服务器 44](#_Toc497837734)

[第6章 终端系统的实现 45](#_Toc497837735)

[6.1 数据传输加密实现 45](#_Toc497837736)

[6.2 操作员APP实现 45](#_Toc497837737)

[6.3 售货APP实现 47](#_Toc497837738)

[第7章 总结与展望 58](#_Toc497837739)

[致谢 60](#_Toc497837740)

[参考文献 61](#_Toc497837741)

# 第1章 引言

本章介绍了论文研究背景及内容，主要从研究背景及意义、研究现状和主要研究内容等几个方面进行介绍。（删除，或者变成第一章的摘要）

## 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。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。

随着消费者需求的变化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)]。然而自动售货机引入我国之后并未得到迅猛发展，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，传统投币式支付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，管理员不能实时的获取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，自动售货行业并未萎靡不振，而是另辟蹊径，将这一传统行业和新兴技术、新型销售方式进行结合，将售货机连入互联网，将传统投币支付的售货机改变为可以使用微信、支付宝、银联等移动方式进行支付[[[3]](#endnote-4)]的设备。多元的支付方式不仅方便了消费者，还省去了清点现金的繁琐，更加节省人力。同时在售货机上新增一个显示器，用于显示商品支付信息、广告信息等，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。售货机技术的发展，将售货机行业再次拉回销售业的竞争环境之中。

## 1.2 研究现状

我国引入自动售货机初期仅仅是引入国外的运营方式、售卖品类和软硬件技术，独立的创新点甚少，这也造成其发展被各方面掣肘。随着互联网技术的发展，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。

现今，售货机行业中供运营商使用的管理系统主要有以下几种类型：1）最常见的是售货机厂商提供的管理系统，这种系统往往功能简单，不能满足运营商的功能需求。2）运营商自行开发的管理系统，这类系统能够满足运营商的日常统计、管理等功能，但是开发成本较高。3）运营商不使用管理系统，仅仅通过人工清算售货机的日常销售概况。

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APP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。

新型售货机系统由厂商端提供，供运营商使用，售货APP与互联网和4G网络相结合，增加了产品功能的多样性[[[6]](#endnote-7)]。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二维码，通过扫码获取商品购物列表[[[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。

新型售货机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，不仅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，软件的开发、部署和维护还需要聘请专业的技术团队。而多数售货机运营商为中小型企业，并无财力和物力去搭建一套完善的新型售货机管理平台。新型软件虽然能够吸引消费者，增强同行业中的竞争优势，但昂贵的开销也令一些公司望而却步。

## 1.3 主要研究内容

SaaS 是互联网技术的一种全新的应用模式，是软件即服务的缩写，它将软件部署在服务提供方的服务器上，提供商负责软件的开发、运维和数据管理等[[[8]](#endnote-9)]。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相对低廉。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，该平台是为解决或改善当前售货机运营商无法提供足够的财力和物力去搭建完善的新型系统这一问题而设计。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。

售货机厂商根据需求分析设计出一套能覆盖自动售货机运营需求和功能的云管理平台，为运营商提供服务。平台采用一对多的交付方式，一套软件可供多家商户使用，运营商只需缴纳少量租金就可以正常使用平台系统。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，运营商将功能实现完全托付给厂商，不需要考虑软件的开发、维护和升级，降低运营成本的同时，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商家意义深远。

## 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。从两个方面介绍，一个为SaaS平台端，一个为Android端。
4. 云平台系统设计。主要介绍系统的总体架构、详细设计和数据库设计，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. 终端系统的实现。首先介绍了操作员APP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APP的设计与实现，并做出相应的时序图和流程图。
7. 总结与展望。总结系统的功能和方法，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 多租户和SaaS服务

1. 多租户技术

多租户技术（multi-tenancy technology）是一种软件架构技术，是实现多企业共享服务的一套软件服务体系[[[9]](#endnote-10)]。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[[[10]](#endnote-11)]。

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，隔离数据模型、共享数据库，共享数据模型 [[[11]](#endnote-12)]。

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。每新增一个租户就需要构建一个新的数据库，从本质上讲各个租户相当于传统的多用户模式，并未真正实现多租户的理念。

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进行隔离。这种模式提供了一定程度的逻辑隔离，但又不是完全的隔离，每个数据库可以支持更多的租户数量，节约了服务器的资源成本。但是若出现故障，数据恢复较为困难，同时因数据库表的限制，仅适合租户较少的应用。

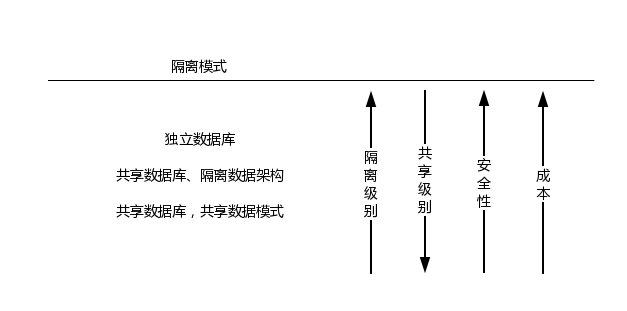
最后，共享数据库，共享数据模式。租户共用同一个数据库，在表中增加相应的TenantId进行数据的隔离。所有租户放在同一个数据库甚至是同一张表中，从本质上说，这是一种安全性、隔离性最低，共享性最高的方案，实现了效率的最大化。当系统需要支持的租户越来越多时，共享数据库低成本、高效的特性就显现出来了。但是这种方案的数据隔离在开发时需要提供更高的技术手段加强数据的安全。租户隔离级别分析如图2-1。

图2-1 租户隔离级别对比

1. SaaS服务模式

SaaS是一种可通过Web访问的软件模式平台，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。用户可以根据自己的需求向软件服务商购买SaaS软件服务。SaaS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2-2为SaaS多租户软件服务的架构。

图2-2 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设备层，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。提供方管理员仅对租户进行管理，并不干涉租户内部的管理和层次结构。

传统软件部署在商家的硬件设备上，除了软件开发的开销还有大量的硬件和服务器的花费。而SaaS提供商将软件服务部署到提供商的硬件设备上，使用者只需通过互联网访问系统，每个商家都节省了硬件设施的消耗，潜在的资源节省超出了传统软件的很多倍。在这种情况下，基于SaaS服务进行开发的提供商将拥有绝对的竞争优势，使用低价高质量的服务吸引客户。即使通过收取租户租金来支持开发成本，仍然有着很大的优势。传统软件与SaaS软件的对比如图2-3。

使用SaaS提供软件服务和传统软件有很大的区别：首先，软件的所有权将从用户转向软件的提供商；然后，软件的基础设施和维护由用户转向软件提供商，包括系统的开发，硬件设施和技术服务；最后，相比传统软件，SaaS软件服务价格低廉。

图2-3 传统软件与SaaS软件对比

## 2.2 SSM框架

SSM框架即Spring + SpringMVC + MyBatis框架的整合[[[14]](#endnote-15)]。通过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企业级应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，也提供声明式事务的支持，能够灵活的控制事务管理。此外，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。Spring构建在核心容器上，其主要分布如图2-4所示。

Spring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2-4 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，框架模型如图2-5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。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。@Controller注解将POJO类作为请求转发和处理的Action类， @RequestMapping注解实现了类似于REST格式的的URL请求，这也是SpringMVC区别与其他框架的优势之一。SpringMVC集成了Spring的优点，将企业级web开发变得更加简洁。

图2-5 SpringMVC框架模型图[[[18]](#endnote-19)]

1. 数据持久层框架对比

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，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2-6所示。

图2-6 MyBatis架构图[[[19]](#endnote-20)]

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，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[[[20]](#endnote-21)]。

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。MyBatis和Hibernate这两大框架的适用类型和成本对比主要分为以下几点：

* 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，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。

根据以上分析，本文选用能够动态定制SQL语句的MyBatis框架，并将其整合Spring、SpringMVC进行整个系统的开发[[[21]](#endnote-22)]。

## 2.3 Android应用

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 [[[22]](#endnote-23)]。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的系统架构，Android系统结构图如图2-7所示。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[[[23]](#endnote-24)]。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。

图2-7 Android系统体系结构[[[24]](#endnote-25)]

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)）[[[25]](#endnote-26)]。

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2-8。

图2-8 Activity生命周期图解

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[[[26]](#endnote-27)]，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。开启Service有两种方式，一种是startService()，另一种是bindService()。

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[[[27]](#endnote-28)]。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厂商模块和运营商模块两个部分，终端包括操作员APP自动售货机售货APP两个部分。本章将从云平台系统和终端系统两个方面进行需求分析。

## 3.1 系统目标

本系统主要使用SSM框架、多租户模型和SaaS模式设计并实现一套能够供多个租户使用的管理平台及终端。

云平台系统分为两大模块，分别是厂商模块和运营商模块。厂商模块包括租金定义、租赁管理、权限管理、角色管理、用户管理、运营商管理、售货机类型管理等。运营商模块包括用户管理、售货机管理、库存管理、商品管理、财务管理等模块。

系统的Android终端分为两大个部分，分别是操作员（售货机上货员工）APP和售货APP。操作员APP为用户提供一个可随时随地进行售货机查询和售货机信息更新的Android应用，免去了人工记录的麻烦。售货APP主要完成自动售货机的日常销售工作，为用户提供支付宝、微信以及银联的二维码扫码支付方式。

## 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[[[28]](#endnote-29)]。

从技术角度分析，数据库开发模式使用多租户架构的概念，云平台后台系统使用SSM框架，平台前端使用JSP、BootStrap、Ajax、Javascript、JQuery等技术，终端使用Android技术。

从经济角度分析，系统由售货机厂商开发，售货机运营商不用付出软件开发、部署的人力和物力，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。

从可操作性上分析，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APP主要进行售货机的查询和货道商品的更新操作，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APP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

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[[[29]](#endnote-30)]，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。本节将从厂商、运营商和消费者等几个方面描述系统的需求分析和用例图。

### 3.3.1 自动售货机云平台

SaaS云平台供两种类型的租户使用，分别是自动售货机厂商和自动售货机运营商。厂商管理员为软件平台的提供者，运营商为平台的主要租户，运营商通过浏览器使用系统。售货机厂商模块包括两种用户角色，分别是系统管理员和厂商管理员，主要进行系统权限管理、用户管理、租赁管理、定价管理、售货机管理、货道管理等活动。运营商端管理模块包括五种用户角色，分别是运营商管理员、用户组管理员、操作员、库存管理员和财务管理员，主要进行售货机日常运营、售货机维护和后续活动的管理。运营商模块包括用户管理、售货机管理、货道管理、商品管理、订单管理等，每个管理员拥有不同的权限，负责相应的管理区间。此外，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。

1. 厂商模块

基于SaaS的售货机云平台是由厂商提供，租户通过web页面进行系统的租用。平台管理员为一级用户，包括系统管理员和厂商管理员两种用户角色。厂商模块系统用例图如图3-1所示。

图3-1 厂商系统用例图

* + 平台注册/登录：

平台仅有一家厂商账号，厂商管理员属于平台管理者用户角色，由系统管理员创建，进行用户名和密码的初始化。管理员填写登录页面表单，系统根据商家编号、账号和密码进入相应的页面。

* + 角色/权限管理：

用户角色包括权限管理和角色管理两个部分，每个角色可以拥有多个权限，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。权限所属类型分为两种：厂商和运营商，在数据库中用数字0、1进行标识。

* + 用户管理

系统管理员可添加、更新、查询和删除用户，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，点击提交新增用户；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；点击删除按钮，删除选中用户。

* + 租金管理：

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。同时设置租用单台售货机的价格。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。

* + 商家管理/运营商管理：

运营商是系统管理员所对应的“用户”，运营商通过注册页面选择租用时间和售货机台数进行租金缴纳和商户的注册。系统管理员也可自行创建运营商账户，单独设置租用价格。

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，弹出未添加的商家列表，点击列表复选框选择要添加的商家。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，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，填写包含售货机类型的售货机详细信息表单后点击保存进行提交。信息添加成功后，点击详细情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。

1. 运营商模块

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用户权限，不同的用户权责分明，共同管理租户系统。运营商系统用例图如图3-2。

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进行注册，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，验证用户是否可用，验证通过后，进入其对应的权限范围页面。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。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，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，普通操作员员不具有用户管理的权限。

* + 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改查操作，并将机器分配给操作员，通过售货机铭牌号、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填写售货机表单信息进行添加；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，提交后添加一个新的分组；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；仓库管理员可进入查询页面查看出库信息。

* + 订单/财务管理：

自动售货机售货APP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，管理员可进入管理页面查看自动售货机订单信息。财务管理员可通过订单信息对每月的交易金额进行统计。

### 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APP和自动售货机售货客户端，它们的后台服务都由SaaS云平台管理系统提供，数据操作和后台系统进行交互。

1. 操作员APP

操作员APP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现场更新售货机数据、进行货物的统计。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。个人页面包含多个菜单选项，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APP用例图如图3-3。

图3-3 运营商操作员APP用例图

* 用户登录：

操作员APP允许登录的用户类型有操作员及其上级用户。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，服务端对用户身份验证成功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。

* 货道更新：

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：

新版本终端系统开发完成后会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。客户端首页提供版本更新按钮，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. 售货APP

售货APP为消费者提供基本交易界面。售货APP用例图如图3-4。

图3-4 自动售货机售货APP用例图

* 多媒体播放：

售货APP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移动支付二维码信息，用户可通过银联、支付宝和微信等移动应用进行扫码支付。

* 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。该应用定期自动检测文件服务器上是否有新的售货APP的APK文件，若有，则下载后静默安装售货APP并自动重启。

# 第4章 云平台系统设计

## 4.1 系统总体架构

（缺乏网络架构、安全性问题的考虑。参考论文3.2.1中的网络拓补图，不写外网，外网直接通过防火墙。一个个APP终端，管理员替换一般用户。一部分通过Internet，另一部分通过移动终端访问的。

软件架构参考图3-2。技术架构图+逻辑架构图。

因为是云平台，要考虑安全问题，可参考宝山区第五章的图。对图有文字的描述和说明。

技术架构，将其具体化

）

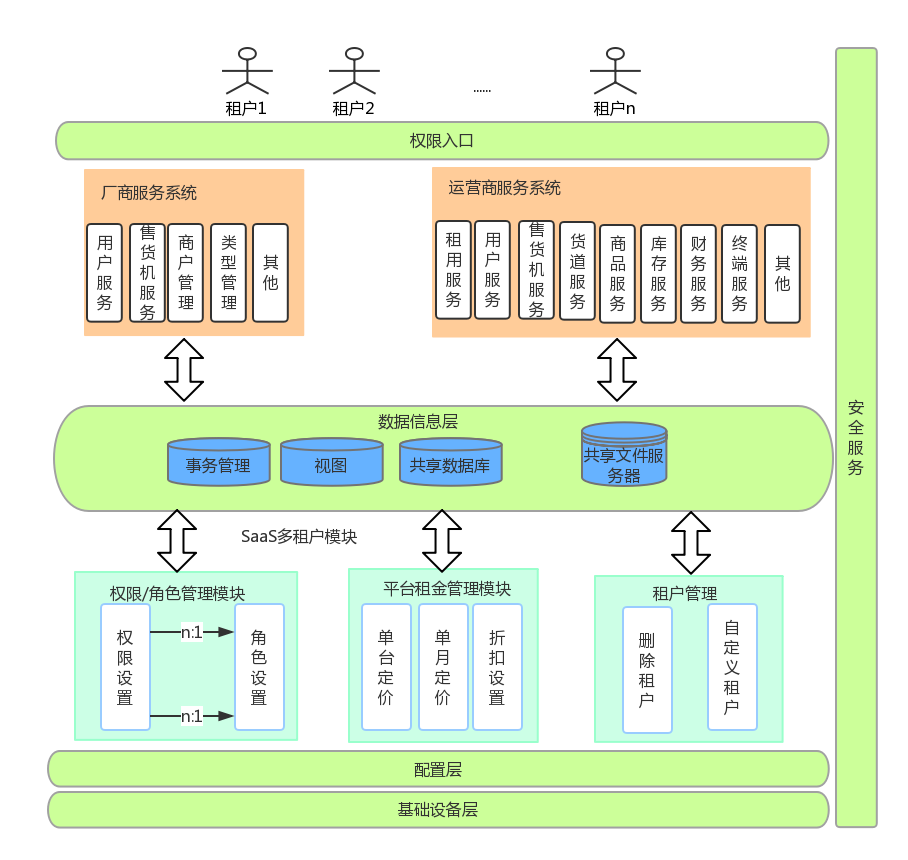
基于SaaS的售货机云平台底层使用多租户的数据模型，结合SaaS服务的思想进行数据库的设计。租户登录时会根据登录表单信息进行角色和权限的判定，判定成功后用户进入相应的管理页面。平台系统总体逻辑架构图如图4-1。

图4-1 平台总体逻辑架构

整个平台系统分为多层：最底层为基础设备层和配置层，主要包括系统的硬件、网络和系统配置；权限管理、平台租金管理为系统的公用层，整个运营商模块都依托该层次的功能进行实现；厂商内部管理服务和运营商服务层处于同一级别，分别管理厂商和运营商的日常运营工作；最上层为权限入口，平台系统供一家厂商和多家运营商使用，运营商是概念上的租户，通过权限入口使用系统[[[30]](#endnote-31)]。

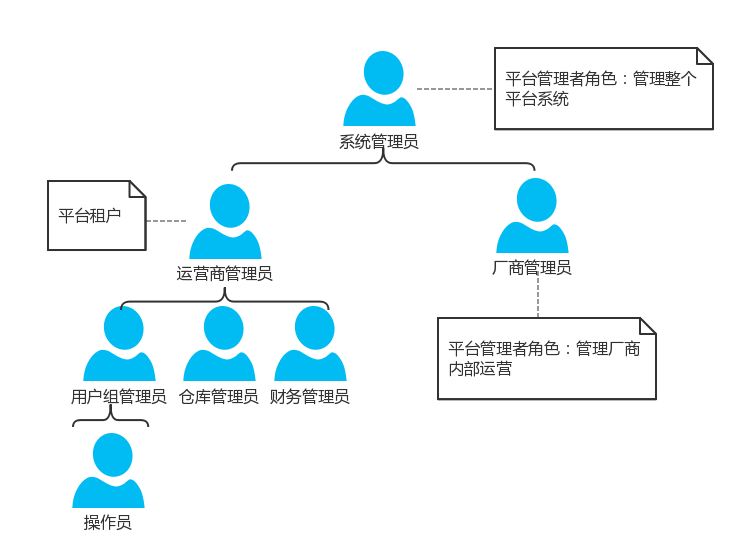
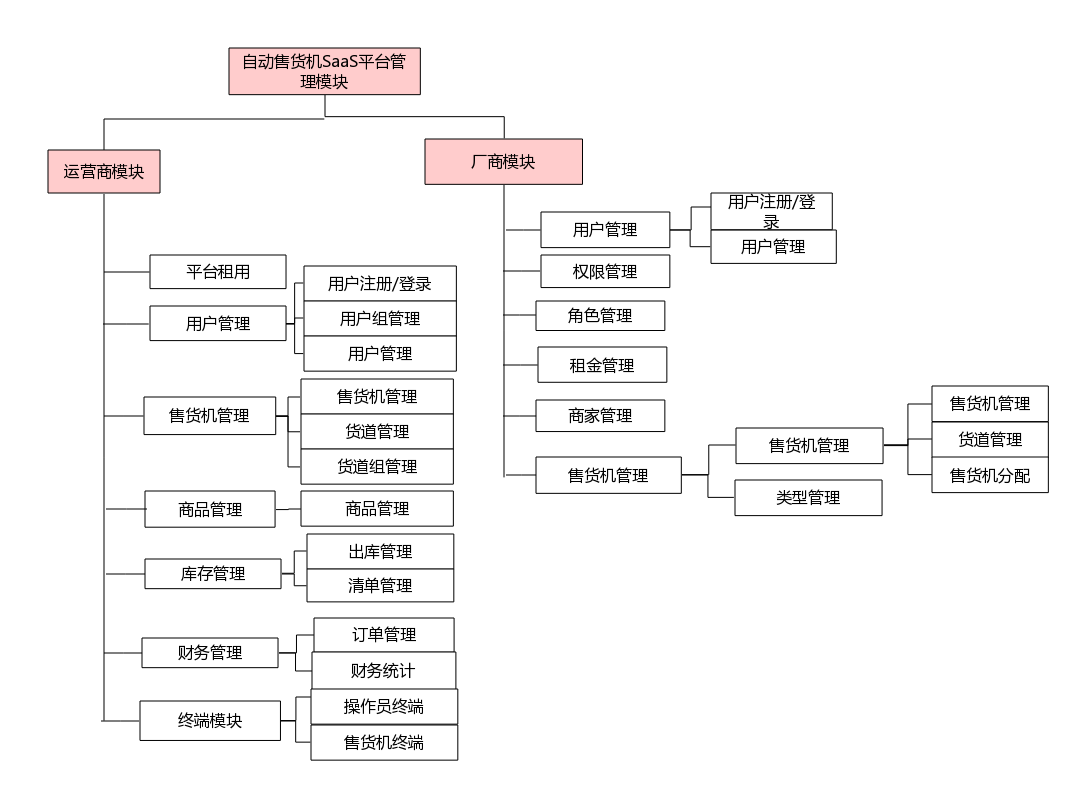
根据系统分析，该平台系统用户角色分为以下几种：1）平台管理者，包括系统管理员和厂商管理员两种用户角色。系统管理员具有最高权限，提供了基本的租金规范和租用规则，管理厂商和运营商两种类型的用户。厂商管理员，对厂商端售货机等信息进行管理。2）运营商管理员，层次在系统管理员之下，是该SaaS平台系统的租户，内部又包括用户组管理员、仓库管理员、财务管理员和操作员角色，对其运营商业务进行管理。图4-2，为用户角色层次结构图。

图4-2 用户角色层次结构图

## 4.2 系统详细设计

1. 功能结构

根据系统功能，自动售货机云管理平台又可分为两个模块：厂商模块和运营商模块。其中，厂商模块是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，同时还提供了厂商内部运营所所需的管理功能。运营商模块为系统的主要模块之一，是该平台租户的主要功能模块，供运营商使用。售货机云平台功能结构图如图4-3。

图4-3 售货机云平台功能结构图

* 厂商模块包括系统的用户管理、租金信息管理、权限管理、角色管理、商家管理、售货机管理、类型管理、订单管理和运营商管理等基本功能。
* 运营商模块主要包括用户管理、售货机管理、货道管理、订单管理、商品管理、库存管理等基本功能。在数据库上进行数据的隔离，使用加密算法保证数据安全，避免各个租户间数据操作的影响。

1. 系统架构

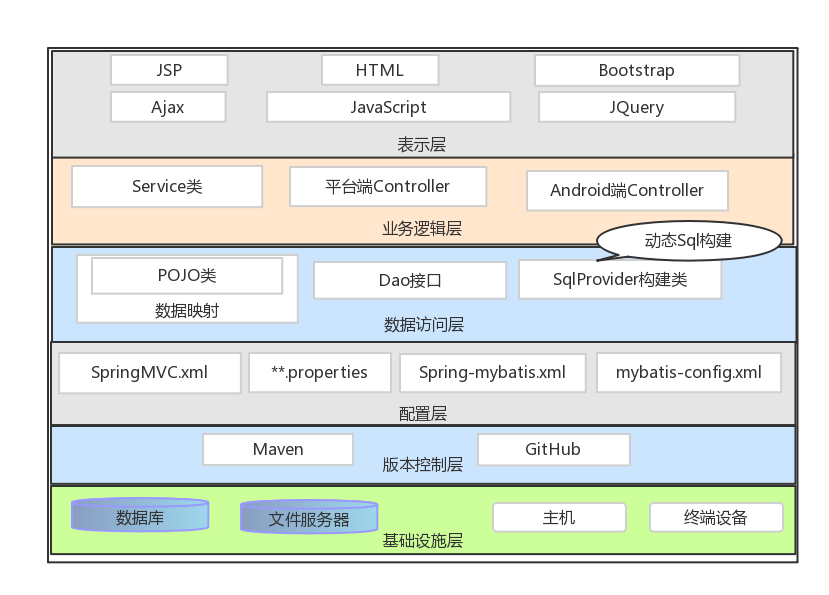
云平台系统的整体技术框架综合考虑厂商和运营商的需求和系统的长远发展，以适应将来业务的持续增长、服务的延伸和平台服务的可持续发展。云平台系统采用分层思想进行开发，使用J2EE框架进行业务逻辑开发，包括配置层、数据访问层、业务逻辑层和表示层[[[31]](#endnote-32)]；使用Maven进行版本的控制，所有依赖包在pom.xml文件中引入；使用GitHub进行代码的管理。相关信息在resource文件中列出，包括数据库账号信息、日志信息、框架的配置信息以及数据库连接池等。云平台系统技术分层架构如图4-4所示。

图4-4 云平台系统技术分层架构图

* 基础设施层：

包括数据库、文件服务器、基础主机系统和终端设备等部分。

* 版本控制层：

版本控制层使用Maven和Github进行项目的构建和代码版本的控制。系统使用Maven，在pom.xml文件中使用几行代码就可以构建简单的项目，具有较高的重用性和一致性。代码使用GitHub进行托管，实现代码版本的控制。

* 配置层：

配置层为SSM框架的公用部分，系统的开发流程都以配置层信息为依据，包括数据库信息配置、Spring、SpringMVC和Mybatis的整合等。

* 数据访问层：

数据访问层是系统访问数据的通用层，负责系统和数据库的交互，完成数据的基本操作，并为上层Service提供调用接口。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。

* 业务逻辑层：

业务逻辑层是系统的核心部分，用于接收请求，主要关注各个页面和API的控制和跳转，分为Service层和Controller层。

Service层主要负责模块的逻辑应用设计，使用面向接口编程的方式进行开发，并注入数据层操作。Service层用于连接Dao层和Controller层的交互，在业务逻辑控制中编写用户身份校验、商户租用期验证等信息。在Service中封装业务逻辑能够方便系统的解耦，提高代码的利用率，也令开发更为简洁。

Controller层负责业务流程的控制，Controller类中使用@Controller注解进行注册，实现业务的调度处理。Controller接收请求信息，调用Service层接口进行业务流程的控制，请求处理完毕后返回一个ModelAndView对象，其中包括了跳转的页面和需要传输的值。

* 表示层：

表示层负责前台页面的展示，与Controller层结合紧密，两者结合进行协作工作，页面中通过表单的提交、链接和按钮的点击等操作将请求信息传入Controller层，通知相应的请求和转发操作。

## 4.3 数据库设计

数据库是软件开发的基础，与平台开发的好坏息息相关。本节将会从以下几个方面进行数据库的分析和设计：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系统的同时将各租户进行业务隔离，主要得益于多租户软件技术架构。多租户和经常说到的多用户概念有所不同，由许多组织或商家共用一套平台系统，每个组织或商家（即租户）有其独立的业务空间，在各自的空间内添加用户，租户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三种，分为独立数据库、共享数据库，隔离数据架构、共享数据库，共享数据架构。多租户数据隔离模型一般根据商户的特性进行选择，从成本、安全性、隔离性和操作难易度等多个方面进行考虑：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。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，本文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

在确定系统多租户架构模式、需求和功能分析的提前下，本节给出一个描述数据实体关系的数据模型，即E-R图（Entity-Relationship Diagram）[[[32]](#endnote-33)]。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图实体之间的关系包括一对一、一对多、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数据对应关系。自动售货机云平台系统的E-R图如图4-5。

1. 数据库逻辑结构设计
   * 商家（商家Id，商家编号，商家名，类型，描述，状态，开始时间，到期时间，租用台数，已用台数，已试用，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（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，开始时间，到期时间）
   * 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，商家Id，操作者，操作时间）

图4-5 售货机云平台系统的E-R图

* + 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，商家Id，操作者，操作时间）
  + 售货机类型（类型Id，名称，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，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，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. 多租户数据库安全

在多租户系统中保护数据库安全被视为首要关注点，租户将信息交给提供商托管，提供商必须给出一套让租户足够信赖的安全体系，保证租户的数据安全。数据库存放着用户和商家的关键信息，比如用户的信息、编号等，若出现以下任意一种状况都将造成不可挽回的后果，比如数据泄露、机器损坏、工作人员不小心删除、电源的故障、人为破坏等，一旦出现故障将给开发和生产生活带来巨大的损失。因此，需对数据提供多个层次和多方面的安全保障，抵挡来自网络或外部的威胁。

数据库安全技术可以从以下几个方面来实现：

* 视图过滤租户信息：

视图是从一个或几个基本表中导出的虚表，就像是一个窗口，可以通过视图查看当前的表信息，限制用户访问特定的信息。在多租户共享数据库模式下，使用视图对数据库表中的某一数据进行授权，就可以阻止对该表中其他租户的信息进行访问，进而提高系统的安全性[[[33]](#endnote-34)]。

视图定义好之后，租户就可以像查询普通表一样查询视图了，同时也可以在一个视图上提交数据，用于添加表信息。如下，在售货机货道表上创建一个视图ChannelView，该视图只允许租户访问自己所属公司对应的货道信息，而无法获取其他租户的信息。

1、创建视图：

CREATE VIEW ChannelView AS

SELECT \* FROM channelinfo WHERE firmId=Login\_Firm\_id();

2、在视图上查询信息：

SELECT \* FROM ChannelView;

这段SQL语句中，通过当前登录的商家Id，建立了视图ChannelView，该视图只包含当前商家的售货机货道信息的部分行。对于商家内部的用户，及获取ChannelView视图的权限，而不是整张表的访问权限。

使用视图机制，就可以在设计数据库系统时，对不同的商家定义不同的视图，使其他租户的信息不会出现在该商家的视图中。这样的机制就自动对不同的商家信息进行了隔离，防止信息的交叉泄露。

* 数据库加密：

和传统系统相比，基于SaaS的多租户系统将数据放在提供商的服务器中，但是对于租户来讲，提供商并不完全可信，对于租户的一些敏感信息，如公司财务和员工信息等，并不想让提供商或其他租户看到，这时就不能将数据明文存储在数据库中了。为保证数据库的信息安全，可以使用数据库加密算法和业务加密算法对数据信息进行加密。

为保证数据库安全，在数据库端中可以使用MD5、ENCODE等加密方式进行加密[[[34]](#endnote-35)]。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。

数据库加密实现如下：

1、使用MD5和ENCODE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');

在业务逻辑中可以使用对称加密和非对称加密算法进行数据信息的编码。对称加密中，客户端和服务器都使用相同的密钥进行加密解密。非对称加密中使用公钥进行加密，私钥进行解密，传输过程中，即使攻击者获取了传输秘文和公钥也不能对其解密，因为只有与之对应的私钥才能够解密密文。加密的数据不能被恶意的检索到，这也从一个方面增强了数据的安全性。

* 数据库备份：

系统每天对数据库信息进行定时备份，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

* 非技术安全：

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。一般情况下，租户和提供商需签订一项保密协议，为数据的安全提供法律保障。

# 第5章 云平台系统的实现

本章介绍基于SaaS的自动售货机云平台管理系统的设计与实现，系统使用SaaS软件即服务的思想和多租户架构的设计方法。云平台管理系统使用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，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

系统初始化时系统管理员需对租赁规则进行定义，单位租金定制分别为1个月、1年、2年、3年和免费试用几种。其中对1年、2年和3年的租用期设置相应的折扣率，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时租户可获得相应的折扣。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：

其中为租金总价，表示每个月的租金金额，表示每增加一台售货机的金额，表示售货机的数量，表示租用的月份，代表租用的年限，代表租金的折扣。当租用时间不到一年时，租金总额为;当租用时间大于等于一年时，租金总额为;试用租金为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金额，两者相减即得新增的租用金额。计算未使用租金的公式如下：

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，则使用计算剩余租金。通过公式（1）和公式（2）可得续租租金的计算公式为：

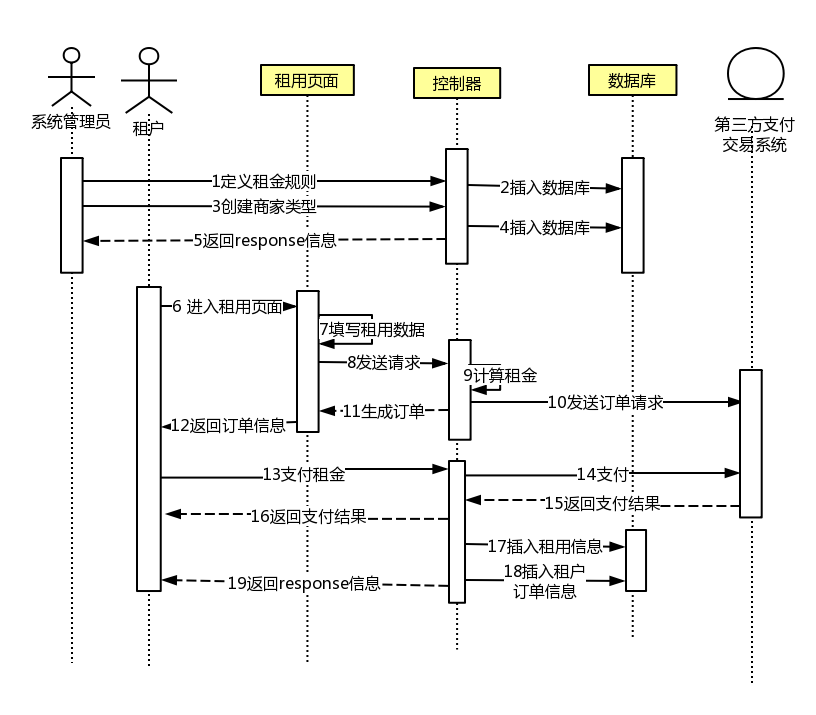
 平台租用功能的实现使用Ajax发送异步请求查询是否存在同名商家，若已存在同样名称或编号的商家，则控制器向页面发送提示信息，提示当前名称或编号已占用。用户选择售货机的租用台数和租用时间，Ajax发送异步请求，后台查询平台的租金定制表，根据租金的计算公式（1）对当前数量的售货机和租用时限进行计算。租户模块序列图如图5-1，实现效果图如图5-2。

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();

}



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，系统将租用信息添加到租户订单表中，同时创建商家账户，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。

## 5.2 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，权限编码代表着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进行表示，0代表厂商，1代表运营商；权限编码的设定也有一定的规则，厂商编码为000XX，运营商编码为001XX。

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。系统初始化时，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。用户角色权限表通过角色Id和权限Id的对应关系获取所有用户角色的权限列表。如图5-3，为权限管理模块序列图。

## 5.3 厂商模块实现

厂商模块除权限管理和租金管理等，还包括用户管理、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。厂商用户输入商家编号、用户名和密码登录系统，系统查询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，JSP页面通过EL表达式和JSTL判断当前用户可见的菜单项。系统使用@SessionAttributes("user")注解将用户详细信息放在session中，在其他需要使用登录信息的方法中通过@ModelAttribute("user") UserInfo userInfo将user对象传入。

图5-3 权限管理模块时序图

厂商模块传入的Session为SessionAttributes({ "user", "machineTypes", "operMgrs" })，分别为用户、机器类型和运营商表。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，最终地址都要陪陪/manu前缀进行拼接。控制层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、一个String对象或进行redirect重定向。厂商模块时序图如图5-4。

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。每个售货机使用商家Id格力数据，售货机信息被封装成MachineInfo对象，管理员添加售货机使用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提示不能被重复售出。

图5-4 厂商模块时序图

## 5.4 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商家系统内的最高权限，能够对用户进行基本数据操作，用户角色由系统管理员进行管理，运营商端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，调用assignRoleToUser(Integer userId,Integer[] roleIds)接口为用户分配角色。管理员对自动售货机进行添加、修改、删除、查询操作，管理售货机货道信息，并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。

图5-5 运营商超级管理员模块时序图

* 运营商操作员

操作员输入登录信息调用userManagerService.checkFirmStatus(firmInfo)对当前租户和用户身份进行判定，登录成功后进入租户的内部平台页面，根据当前用户的权限，页面将售货机查询、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。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，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。售货APP新增了移动支付功能，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统计提交到营业额表中，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，然后填写商品信息和商品数量提交表单，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：

库存管理员包括商品管理和库存管理两个功能。商品管理模块是对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。

图5-7 库存管理模块时序图

* 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，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5-8 财务管理模块时序图

## 5.5 文件服务器

管理系统除提供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APP，另一种为供自动售货机使用的售货APP。终端系统使用Android进行开发，云平台管理系统为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，数据加密主要体现在用户的信息、密码、支付信息等敏感数据。

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，传输过程中即使密文被截获，获取者也不能对信息进行解密，主要对用户的密码进行加密[[[35]](#endnote-36)]。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，将信息使用公钥加密后传回服务端，服务端使用与之匹配的私钥进行解密[[[36]](#endnote-37)]。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APP实现

操作员APP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端提供访问服务端的HttpURLConnection连接Util类，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，将请求参数封装成key/value类型的Map对象，使用&符号将key/value值进行连接；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，这时FutureTask就可以发挥作用了，使用FutureTask异步执行和Callable结合使用，通过get方法调用最终的处理结果，并将其返回[[[37]](#endnote-38)]。

系统调用HttpURLConnection的postByResponse接口向后台服务器发送POST请求验证用户信息表单，并向终端返回一个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，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。操作员通过用户Id和所属的商家Id，使用HttpURLConnection类请求查看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APP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getPackageManager()获取PackageManager对象，并使用packageManager.getPackageInfo(getPackageName(),0)获取到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。系统从文件服务器上下载该APK的版本配置文件信息，配置文件存储为JSON格式，使用JSONObject获取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，不做任何处理。

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APP实现

1. 购物流程的实现

售货APP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，售货机界面轮番循环播放广告视频，当界面被触摸时进入选货界面。选货成功后系统经过HTTPS请求生成支付二维码[[[38]](#endnote-39)]，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，售货APP时序图。

图6-2 售货APP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，在activity\_ware.xml布局文件中声明使用GridView进行网格分布，设定每个网格中要显示内容的布局和组合，设置GridView的显示列数以及各个分格之间的间隔。

Adapter继承BaseAdapter类，让开发者自定义要显示的内容。首先Activity检测本地货道表，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格式的货道信息，将该JSON对象进行解析，获取货道列表。然后获取ware\_grid.xml内容，重写getView()方法，并在相应的位置上显示解析后的商品内容。货道信息中包含商品名称、价格、描述和图片路径等信息，图片信息放置在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，解析JSON对象得到商品图片的路径信息，调用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

售货APP除进行商品的销售外还可以提供广告播放，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APP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，将视频文件的路径放置在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。广告视频播放一般流程如图6-3。

图6-3 视频播放流程图

* 购物流程

售货APP主要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，售货机不出货，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，选择同等价位的商品进行出货。

1. 串口通信实现

售货APP和售货机控制盒之间使用FT321D芯片作为串口转换的芯片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，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，系统对接收的指令进行解析，并将要发送的信息封装后进行传输，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，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，确定当前需要出货的货道进行出货。

系统在AndroidManifest中设置对USB的读取权限，获取UsbManager对像用于设备管理。使用ComService类初始化串口，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。

串口消息报文格式如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 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，当接收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。系统通过读取串口信息，对信息进行校验，解析指令处理后，发送相应的应答。订单支付成功，发送串口信息，进入出货界面。串口通信过程流程图如图6-4。

图6-4 串口通信流程图

* 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。

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。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。移动支付时序图如图6-5。

* 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。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，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。然后，同步返回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对象，对其进行签名的验证，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。

图6-5 移动支付时序图

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，才继续下面的步骤。验签过程如下：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时间。将所有信息封装成一个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，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，然后发送给支付宝服务。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，验证成功后进行返回。判断alipay\_trade\_pay\_response中key为code的字段，请求成功时code为10000，msg为Success，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APP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交易的过程如下：首先，将交易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。

支付宝二维码发送请求时返回JSON对象，使用GSON包将JSON字符串进行解析，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，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和其他数据元作为请求字段。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的形式进行表示，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，将请求串转换为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code\_url字段值并返回，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，并转换为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售货APP软件版本和更新软件，因此售货APP的更新检测采用静默更新和安装。

管理员在文件服务器中放置软件的最新版本文件和包含软件新版本信息的JSON文件。使用另一个Android应用程序对售货APP进行检测更新，系统设置一个定点时间，通过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无障碍服务智能安装，并在AndroidManifest.xml文件中进行配置，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。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商家发展现状的一个新概念设计，采用了SaaS模式的设计思想，结合多租户设计框架，将单一的售货机管理系统转变为可共享的售货机管理云平台。

云平台系统采用SSM框架，Maven构建工具、Mysql数据库、Bootstrap、Ajax和分层的设计思想实现系统的开发。系统使用“共享数据库，共享数据模式”实现数据库的共用，并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功能，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，并提供数据的加密传输保证传输信息的安全性。

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，由售货机厂商进行系统的部署和发布，售货机运营商可直接通过网页入口访问平台系统。系统的使用方式便捷，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运营商进行开发，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，数据库对用户和其他敏感信息进行加密操作，防止信息的泄露。
* 经济性：1）运营商不需要提供硬件和网络服务，还省去了雇佣技术人员的额外开销和人员配置；2）运营商不需要对系统进行运维支持，减少了工作量；3）该系统为一个共享的多租户系统，一次开发可供多个运营商使用，租金低廉，能够帮助厂商吸引客户，增强竞争力；4）终端系统使用价格低廉的Android设备进行开发，改造成本较低。
* 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，操作员不需要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。

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。目前自动售货机商家的发展良莠不齐，售货机种类不一，想要大幅推广该平台系统还需加大工作力度和调研，以保证系统的进一步优化，供不同的商家使用。今后的工作中需要进一步研究的内容有以下几点：首先，是多租户模型的改进，在数据隔离和数据安全上可进一步加强；其次，是平台系统的推广，系统不仅需要供当前厂家的合作运营商使用，还可为其他运营商提供服务，加强系统定制的灵活程度。

# 致谢

时光飞逝，两年半的研究生生涯即将结束。借此机会，我要感谢在这个两年半的研究生生活中给予我帮助的老师和同学，感谢华东师范大学为我提供了优良的学习环境，感谢我的父母对于我学业的支持。

首先，感谢我的导师章炯民副教授对我论文选题的帮助和建议。在该课题的研究过程中，是您给与了我研究的方向，帮我理清思路。感谢您和公司沟通，为我提供开发的机器和控制板，让我潜心学习和研究。系统的完善和论文的成文都离不开章老师的悉心指导和帮助，在我思路混乱之际，是您耐心的指导和建议鼓励我继续前行。此外，两年半的研究生生活中，您对我的生活上的关怀、学业上的引导都对我的生活产生极大的帮助和影响。您严谨的治学态度、优良的工作作风和不断学习的科研态度都将对我今后的工作和学习产生巨大的影响，令我受益匪浅。

然后，感谢华师大为我提供了优良的学习环境和学习氛围。我还要感谢一下实验室的同学，感谢张彰师兄给与我的鼓励和支持以及论文的修订，感谢张媛师姐给与我的思路和鼓励，感谢蒋超群师姐耐心解答我的疑惑和问题。感谢师兄周志民、李超，薛杰，师姐贾柯，同学亓麟、贾高胜、孟庆红，师弟何卓立、钱宝健、瞿夏君、吴凯宗，腾宇，师妹朱曼、方雪，厉劲草，对我学习和生活上的陪伴和帮助。

最后，感谢我的父母和家人，是你们的默默支持，以及对我的鼓励和关怀，才使我顺利完成学业，追求更好的未来。

# 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. [] Cusumano M. Cloud computing and SaaS as new computing platforms[M]. ACM, 2010. [↑](#endnote-ref-9)
9. []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-10)
10. []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-11)
11. []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-12)
12. [] Bezemer C P, Zaidman A. Multi-tenant SaaS applications: maintenance dream or nightmare?[J]. Proceedings of the Joint Ercim Workshop on Software Evolution & International Workshop on Principles of Software Evolution, 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. []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-21)
21. []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-22)
22. []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-23)
23. []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-24)
24. []郭霖. 第一行代码[M]. 人民邮电出版社, 2014. [↑](#endnote-ref-25)
25. []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-26)
26. []纪晓阳. 线程在Android开发中的应用[J]. 软件, 2013(8):24-26. [↑](#endnote-ref-27)
27. []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-28)
28. []赖均. 软件工程[M]. 清华大学出版社, 2016. [↑](#endnote-ref-29)
29. []胡荷芬. UML系统建模基础教程[M]. 清华大学出版社, 2014. [↑](#endnote-ref-30)
30. []陈旭, 武振业. 中小企业管理信息系统总体设计研究[J]. 计算机应用研究, 1999(12):11-14. [↑](#endnote-ref-31)
31. []陆荣幸, 郁洲, 阮永良,等. J2EE平台上MVC设计模式的研究与实现[J]. 计算机应用研究, 2003, 20(3):144-146. [↑](#endnote-ref-32)
32. []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-33)
33. []王兰生, 尹湛. 面向对象数据库视图的研究与实现[J]. 南京邮电大学学报(自然科学版), 2000, 20(3):73-76. [↑](#endnote-ref-34)
34. []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-35)
35. []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-36)
36. []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-37)
37. []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-38)
38. []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-39)