2018 届研究生硕士学位论文

学校代码： 10269

学　　号： 51151201090

華東師範大學

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

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:	Computer Science and Software Engineering
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

现代社会人力成本不断攀升，消费方式升级换代，信息技术及其应用快速发展和深入，传统依靠人工运营和管理的自动售货机已跟不上时代发展的潮流。为提高运营效率、降低成本，自动售货机“触网”，实现管理体系信息化和售货机终端智能化是必由之路。然而，我国有 60%以上的售货机运营商为中小型企业，基本不具备独立承担开发和运维联网自动售货机管理平台的能力，极大限制了该行业的发展。因此，本文设计了一种可共享的自动售货机云平台系统（简称“VMCloudPlatform 系统”），旨在解决或改善这一问题。VMCloudPlatform 系统将由生产厂商统一管理，供众多运营商共享使用，既为运营商降低了成本、提高了管理效率，又增加了自动售货机本身的附加价值，增强了厂商的竞争力。

VMCloudPlatform 系统由三部分构成：VMCloudPlatform 管理系统、运营管理 APP 和终端售货 APP。VMCloudPlatform 管理系统是整个系统的核心和基础，主要实现用户权限、运营商、售货机、租赁、库存等基本的管理功能，并响应来自 APP 的请求，实现数据的采集、存储和查询；运营管理 APP 配置于运营商营业员的移动设备上，方便营业员现场实时更新数据；终端售货 APP 配置于售货机，供普通消费者选择商品并完成移动支付，其具备自动重启和远程更新的功能，从而具有较好的容错能力和较高的更新效率，以适应自动售货机分布点多、地域范围广的特点。

VMCloudPlatform 管理系统基于 SaaS 服务，结合多租户技术，采用 JavaWeb 分层思想设计实现；后台管理系统采用了 SSM (Spring+ SpringMVC+Mybatis)框架，前台界面采用了 JSP、Ajax 和 JQuery 等技术。运营管理 APP 和终端售货 APP 基于 Android 系统开发，其中终端售货 APP 使用 FT312D 芯片和自动售货机相连，实现 Android 的 USB 接口转串口功能，控制售货工作流程。

关键词：自动售货机、SaaS、多租户、SSM 框架、Android

ABSTRACT

Nowadays, the human cost is rising and the consumption mode is updating. Meanwhile, information technology and related applications are rapidly developing and deepening. These lead to the traditional vending machines that rely on manual operation and management have been unable to keep up with the trend of the times. In order to improve the operation efficiency and reduce the cost, the only way that vending machines can do is that connect the network and implement the information management and intelligent vending machine terminal. However, more than 60% of our vending machine operators are small and medium-sized enterprises which greatly limits the development of the vending machine industry. Therefore, this paper designs a sharable vending machine cloud platform system (VMCloudPlatform system for short), aimed at solving or improving these problems. VMCloudPlatform system is managed by the manufacturers and used by multiple operators. It reduces the cost for operators and improves the management efficiency. At the same time, it increases the additional value of vending machines and enhances the competitiveness of manufacturers.

VMCloudPlatform system consists of three parts: VMCloudPlatform management system, VMManage APP and VMSale APP. VMCloudPlatform management system is the core and foundation of the whole system. It mainly realizes the basic management functions of user rights, operators, vending machines, leasing, inventory, and response to requests from APP to achieve data collection, storage and query. VMManage APP is installed on the mobile equipment of operator assistants, which is convenient for operator assistants to update data on the spot. VMSale APP is installed on vending machines for ordinary consumers to select goods and complete mobile payment. It has the functions of automatic restart and remote update, so it has better fault tolerance and higher update efficiency. Automatic restart and remote update adapt to the widely

distribution features of vending machines.

Design and implement of VMCloudPlatform system is based on SaaS mode, multi-tenant framework and JavaWeb layering idea. VMCloudPlatform management system uses SSM(Sping+SpringMVC+Mybatis) and the front end adopts JSP, Ajax, JQuery and other technologies. VMManage APP and VMSale APP are based on the Android system, in which the FT312D chip connects VMSale APP with vending machine, which converts the USB interface of Android into serial port and controls the workflow of the sales.

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

目录

第 1 章 引言

1.1 自动售货机行业概况

自 20 世纪 70 年代起，自动售货机因不受地域的限制，占地面积较小，节省人力物力资源且 24 小时不停机服务[1]，在国外发达国家和地区发展起来。作为世界上最大的自动售货机市场，日本的售货机行业发展相对成熟，仅售货机机型就有 2000 多种。而在欧美地区，欧洲平均每 60 人拥有一台售货机，美国每 40 人拥有一台售货机。但是，中国即使是一线城市的北京也只能达到平均 1000 人拥有一台售货机，与日本、美国等发达国家相比存在很大的差距。

我国自引入自动售货机以来，多年来一直没有得到大幅的发展和广泛的应用。主要有以下几个方面的原因：首先，20 世纪 90 年代，我国人力成本较低[2]，城市商铺租赁和商品营业用房价格低廉，推动了国内早期传统零售产业的发展。而且大多数消费者习惯于传统的交易方式，自动售货机作为舶来品，人们对其安全性有所顾虑。正因如此，此时的自动售货机在国内的发展毫无优势可言。然后，我国人均硬币保有量较低，而自动售货机需使用硬币进行交易。最后，早期国内自动售货机多从国外购买，价格高昂，生产技术不成熟[3]。

随着国内产业结构从劳动密集型向技术密集型转变、人们消费观念的转变、以及人力成本和商铺租赁成本不断上升，自动售货机行业得到发展的良机，产业规模不断扩大。进入 21 世纪，计算机技术和网络不断渗透到各行各业，人们对信息系统的依赖不断增强，国内市场对售货机信息化管理也有着越来越迫切的需求。然而，在管理技术方面，我国仍面临以下几个问题：

- 自动售货机运营商多使用传统的人工管理模式，机器款式老旧，技术不够成熟，很难对分散的售货机进行高效管理。运营商也无法第一时间获取售货机内的商品剩余情况，致使补货的延误，影响经济效益。
- 当今社会 80 后和 90 后的新生代消费者成为消费的主力军，微信、支付宝、银联等移动支付[4]兴起，消费者消费观念逐渐改变[5]，单一的投币支

付已跟不上无现金支付的潮流，导致了新生代消费者的流失。

- 传统自动售货机扩展性较差，无法在自动售货机终端提供电子广告等增值服务。
- 部分售货机实现了信息化的管理平台和智能终端软件，但信息化成本较高，所占比例较大的中小型运营商技术落后[6]，并无足够的财力物力负担信息化软件的开发或机器的购买，影响了整个售货机行业的发展。

1.2 自动售货机的发展现状

目前，自动售货机在日本、欧美等发达国家发展较为成熟。美国在物联网的系统架构、标准定义以及安全管理等方面的研究投入了大笔的资金，使自动售货机基于物联网的系统得以成长[7]，售货机运营商与多家大型饮料商、食品商合作，具有大量的融资和开发运营的经验。而日本很早就采用联机技术，通过电话线路将自动售货机内的库存信息传送至管理系统，实现售货机的远程监控。日本售货机运营商在和大型供应商合作的过程中，在售货机上展示电子广告[8~9]，发掘新的赢利点。

得益于 IT 技术的成熟，很多企业借助 IT 技术实现低成本的信息化系统。国内少数大型售货机商家将自动售货机与互联网相结合，提供自动售货机的后台管理系统和智能终端软件：首先，将自动售货机联网，提供显示屏供用户选货和播放电子广告，并实时更新存货量；然后，营业员通过后台管理系统对终端库存进行监控，并根据库存信息及时补货[10]；最后，提供多种支付方式。这种信息化的发展趋势，适应了当下人们的消费习惯，降低了运营和管理的成本，还拓展了自动售货机的应用领域。

目前，新型联网管理的自动售货机信息化类型逐渐多样化，如：提供智能语音，通过语音和售货机进行交互；提供趣味游戏，在终端设置互动游戏，通过游戏分值换购商品等。联网的自动售货机不断拓展增值业务，以国内首屈一指的自动售货机商家“友宝”为例[11]，2016 年就实现了 2.9 亿元的广告陈列收入，这无疑是一笔很可观的收益。

提供信息化的管理机制、智能化的销售终端和便捷的移动支付，已成为影响自动售货机行业发展的关键因素。国内大型自动售货机商家，如友宝、大连富士山冰、顶顶等都加入了信息化的阵营，并且取得了不俗的成果。但是，研发和更换新款自动售货机依然价格不菲，大批的中小型运营商只能使用传统的人工管理运营方式和老式售货机，导致了新型信息化自动售货机在我国的应用不够广泛。据 2017 年中国自动售货机行业发展概况分析可知，国内自动售货机有 50 余万台，实现信息化的新型机有 5 万余台，仅占 10%左右。因此，如何改造当前传统售货机、帮助中小型运营商实现自动售货机管理信息化、减少资源浪费、增加商家行业竞争力成为当前亟待解决的问题。

1.3 VMCloudPlatform 系统的目标和意义

目前我国的自动售货机运营商多数为中小型企业，售货机机型比较落后，其信息化程度低、管理手段落后、人工成本高。老式售货机销售终端无法联网管理、支付方式单一、功能老旧、难以在此基础上进行增值功能的拓展。这些因素极大地限制了国内自动售货机业务的推广。基于国内中小型企业不具备独立承担开发和运维联网自动售货机的能力，以及无经济实力更换新型售货机的等一系列问题，本文提出了通过外接 Android Pad 的方式改造老式自动售货机，将分散的售货机联网，并实现可共享的管理云平台的信息化设计方案。

本课题的目的是设计一个自动售货机云平台系统（以下简称“VMCloudPlatform 系统”），该系统包括自动售货机云平台后台管理系统（以下简称“VMCloudPlatform 管理系统”）、相应的终端智能售货 APP（以下简称“VMSale APP”），以及方便自动售货机运营商营业员使用的运营管理 APP（以下简称“VMManage APP”）。旨在以较低的成本帮助自动售货机中小型企业实现信息化管理和智能化的售货终端，增强自动售货机厂商（以下简称“厂商”）和自动售货机运营商（以下简称“运营商”）在行业内的竞争力。

VMCloudPlatform 系统由厂商进行管理，为多个中小型运营商提供低价的管理服务。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，从整体上减少软件开发和运维的成本。VMCloudPlatform 管理系统为运营商提供用户管理、售货机管理、商品管理、货道管理和订单管理等功能，将传统的人工管理模式信息化，从而提高员工的办事效率、减少人员的浪费、降低出错率。有了管理系统，营业员可实时监控自动售货机的库存和运营状况，更有计划的进行补货和设备维护，从而降低运营成本。售货机上搭载的 Android Pad 除了通过 VMSale APP 实现售货机的日常售货，还具有可观的广告价值。此外，移动支付的实现，极大改善了用户的购买体验，进一步降低了运营成本。VMManage APP 实现自动售货机营业员在上货现场进行售货机信息的实时更新，提高营业员的工作效率。

1.4 论文的组织结构

本文从以下几个章节进行介绍：

第 1 章：引言。主要对系统的研究背景进行介绍，对国内外同类产品与技术的全面、深入综述与分析。分析国内售货机行业中小型运营商面临的问题，提出基于 SaaS 的 VMCloudPlatform 系统和基于 4G 网络的 VMSale APP 和 VMManage APP，并对主要研究任务进行了阐述。

第 2 章：VMCloudPlatform 系统的相关技术分析。主要介绍了系统开发涉及的 SSM 框架、SaaS 服务、多租户技术、Android 和其他相关技术。

第 3 章：VMCloudPlatform 系统的需求分析。介绍 VMCloudPlatform 系统的需求分析和用例图。从 VMCloudPlatform 系统和相应的 Android 终端软件两个方面详细分析。

第 4 章：VMCloudPlatform 系统的设计。主要介绍自动售货机的网络结构、系统的总体架构、软件的体系架构、自动售货机的硬件构成、系统的实现难点和数据库设计等，并且做出数据库 E-R 图。

第 5 章：VMCloudPlatform 系统的实现与测试。主要介绍关键技术和难点的具体实现，以及厂商、运营商管理系统和终端软件的实现，并给出对应的时序图、部分截图和实现代码。同时还对系统的单元测试、集成测试、功能测试和非功能

测试方法进行了简要介绍。

第 6 章：总结与展望。总结 VMCloudPlatform 系统的功能和实现方法，并提出进一步的展望。

第 2 章 VMCloudPlatform 系统的相关技术分析

VMCloudPlatform 系统使用 SaaS 的设计思想、多租户技术进行软件体系架构的设计。系统后台使用 SSM 框架分层设计，旨在提高软件的开发效率，降低代码的耦合性。使用面向对象和分层开发的方法增强系统代码的可维护性和重用性。系统终端软件使用 Android 技术开发，通过 Android 串口通信技术解决了 Android 设备和自动售货机的串口通信问题。

2.1 云计算技术和 SaaS 服务

2.1.1 云计算技术和 SaaS 服务概述

1. 云计算技术的基本概念

云计算（Cloud Computing）技术是一种新型的理念，这一概念在 2007 年出现，并在 Google、Amazon、IBM、阿里云等 IT 公司迅速发展。云计算是对内部基础设施的一种抽象和虚拟化的技术，能够使用户简易的获得设备的高性能计算和存储能力[12]。它通过网络获取计算机资源，为用户提供弹性伸缩、按需使用的云服务[13]。

目前较为主流的云计算技术有 SaaS（Software-as-a-Service）、PaaS（Platform-as-a-Service）和 IaaS（Infrastructure-as-a-Service），从本质上讲，SaaS 是一种 web 应用程序，PaaS 是一种软件开发平台，为 SaaS 提供基础服务平台，IaaS 是计算机基础设施资源，为 PaaS 提供基础设施服务。SaaS 是当前较为流行的云计算技术之一，软件开发者可将软件系统部署在第三方平台上，供多个租户租用，从而有效降低企业的信息化成本和实现进程，本文使用 SaaS 服务的思想进行系统的设计和实现。

2. 多租户技术和数据隔离分析

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[14]。以服务的形式将系统的软硬件资源、运维管理资源等提供给多个企业租户复用，从而有效降低 SaaS 应用的成本，为 SaaS 软件

服务提供了开发思路和技术支持。多租户是 SaaS 服务和传统软件本质的区别，核心是解决租户间的数据隔离问题。传统软件只针对一个类型的用户进行开发，不存在用户数据混乱的问题，而 SaaS 模式下的软件系统必须进行数据的隔离来区分租户间的数据信息[15]。

多租户架构在数据隔离方面有三种设计模式：独立数据库、共享数据库，单独模式、共享数据库，共享模式[16]。

表 2-1 多租户隔离级别

	独立数据库	共享数据库，单独模式	共享数据库，共享模式
隔离级别	高	中	低
共享级别	低	中	高
安全性	高	中	低
成本	高	中	低
可配置性	高	中	低

“独立数据库”为每个租户创建一个数据库，这种模式的数据实现方式具有最高的安全性和隔离性，同时也耗费最高程度的成本。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。

“共享数据库、单独模式”中多个租户共享同一个数据库，使用 Schema 进行隔离，因数据库表的限制，仅适合租户较少的应用。

“共享数据库，共享模式”中租户共用同一个数据库，所有租户放在同一个数据库中，其安全性和隔离性最低，共享性最高，需重点考虑数据的安全性和隔离性。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。

3. SaaS 服务的基本概念

SaaS 是 IBM 提出的一种新型软件交付服务，服务可供多个租户共享使用。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[17~18]。其主要特征体现在应用代码所处的位置、部署和存取代码的方式，将传统一次性买断模式转换为集中软件租用模式，从而将软件商业模式从产品供需转为服务供需。

SaaS 软件提供商提供服务器、软件服务和共享数据库，并负责平台的开发、

测试，最终将软件部署到自己的服务器上。客户根据自己的需求，通过互联网向提供商购买软件服务。共享的软件服务将单个用户进行软件开发和运维的成本分摊到各个租户的租金中，使软件使用费用变的低廉。这种模式非常适用于软件技术部门不完善、经济能力有限的中小型企业，是其实现信息化的捷径。

2.1.2 SaaS 服务成熟度模型及优势

1. SaaS 服务的成熟度模型

SaaS 服务具有“软件即服务”的理念，其成熟度模型可分为四级，用来表示该模式是否可配置、高性能以及可伸缩性[19]。图 2-1 为 SaaS 的四级成熟度模型。

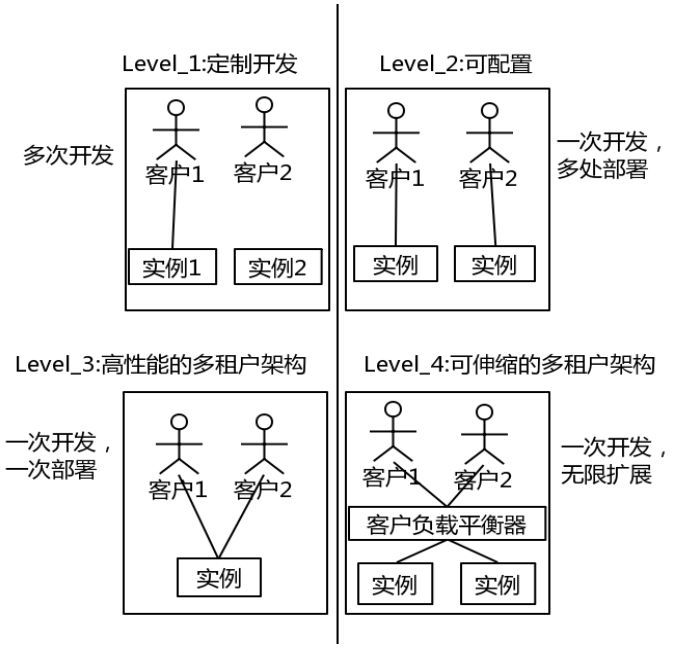

图 2-1 SaaS 成熟度模型

- 成熟度模型 1：每个客户使用一套独立的数据库系统、独立的 web 站点和独立的软件，单独运行一个实例，软件提供商根据客户的需求进行定制化设计和修改，每个客户之间完全独立。从技术上讲，不同客户之间除少量可重用代码外，与传统系统开发并无太大区别。
- 成熟度模型 2：是对第 1 级别的改进，不同客户运行具有相同代码的实例，根据每个客户的不同信息进行系统配置，通过一次开发，多次部署，实现软件的扩展性，降低开发成本。所有客户使用一套软件，当代码库

进行修改时，将作用于每一个客户。

- 成熟度模型 3：第 3 级模型是对前两级模型的进一步优化，从实际技术层面实现了 SaaS 模式。软件系统仅进行一次开发，一次部署，所有客户共用一套软件应用实例、数据库系统和硬件系统，其资源利用率大大超过成熟度模型 2。同时该级别使用多租户的概念，对各个客户进行数据隔离，通过授权保证其数据的安全性。
- 成熟度模型 4：所有客户运行相同的实例，通过中间的客户负载平衡器将客户分配到各个运行的实例上，来分摊大量的访问，是 SaaS 应用的最理想级别。

通过以上分析，成熟度模型 4 是 SaaS 服务最理想的架构模型，但在实际的应用开发中，对于 SaaS 成熟度模型的选择往往需要考虑客户的实际需求和能力等多方面的因素。

2. SaaS 服务的优势分析

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。基于 SaaS 的软件将软件服务部署到提供商的硬件设备上，租户只需通过互联网访问系统，每个租户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[20]。因此，基于 SaaS 服务开发的软件系统将拥有极大的竞争优势。传统软件与 SaaS 软件的“用户”层次对比如图 2-2。

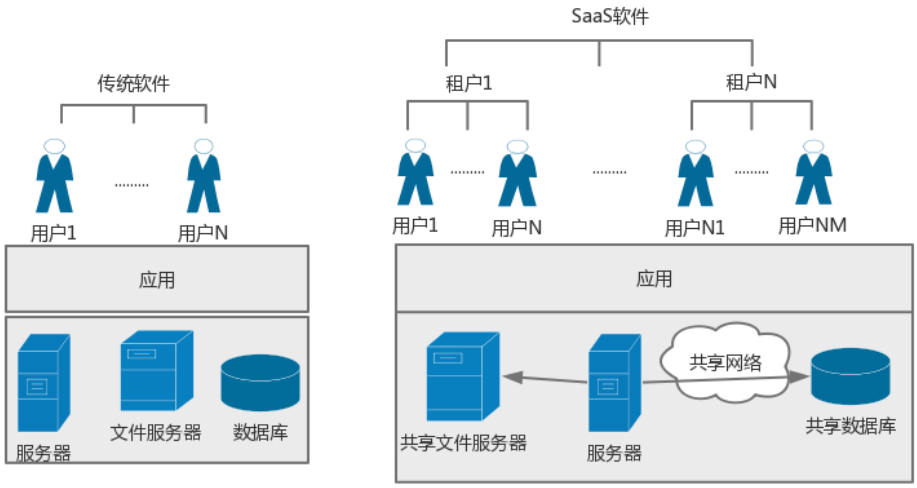

图 2-2 传统软件与 SaaS 软件层次对比

本文面对的目标用户群体为中小型售货机运营商，对于中小型企业而言，选

用 SaaS 提供软件服务和传统软件对比有以下几点优势：

- 支持多租户：软件支持多租户，规模化的使用将开发费用平摊到各个租户中，降低租户信息化的资金压力。
- 所有权转移：软件的所有权将从普通用户转向软件的提供商，软件实施过程较快，不需租用系统的用户做任何操作。
- 软件开发和维护职能转移：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，租户不用培养专业的 IT 团队，不需要参与产品升级和后期维护，以及承担任何软件升级和后期运维的费用。
- 操作方式：租户的操作接口便利，使用门槛较低，客户端使用浏览器登入，在任何能上网的地方都能使用。
- 资金投入：租户资金投入较少，不需要投入大笔资金一次性买入，只需“按需租用，按需付费”，租用价格低廉。
- 适应类型：传统软件提供商难以为中小型规模企业提供低廉的服务，而 SaaS 服务非常实用中小型企业，能有效降低租户的软件成本[21]。

2.2 SSM 框架及应用分析

SSM 框架即 Spring + SpringMVC + MyBatis 框架的整合[22]。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，MyBatis 管理持久层，SpringMVC 管理表现层。

1. Spring 框架和 SpringMVC

Spring 是一种轻量级的容器框架，以 IoC 和 AOP 为内核，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，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 不但方便了解耦，控制了对象间的依赖关系，而且提供了声明式事务的支持，灵活的控制事务管理。还能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对持久层、业务层和展示层的所有支持。Spring 的核心容器有四个，有 Beans、Core、Context 和 SpEL。核心容器构建了 Spring 的基础架构、提供了框架的基础功能。Spring 框架接口如图 2-3 所示。

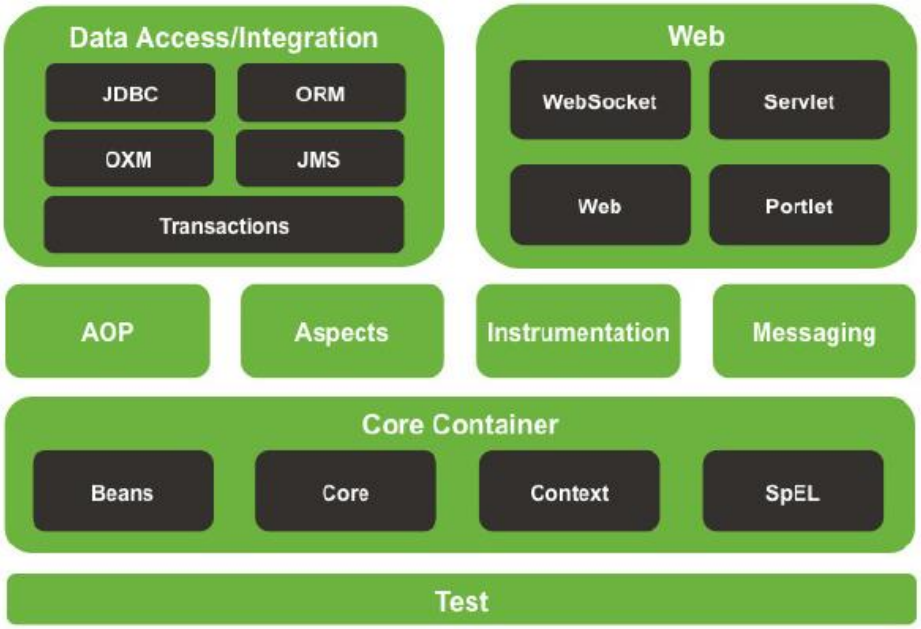

图 2-3 Spring 框架组件图[23]

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，将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。

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 是一种设计思想和对面向对象编程的一种补充，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。

SpringMVC 是一种基于 MVC 三层模式的 web 框架，它围绕 DispatcherServlet 展开，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 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

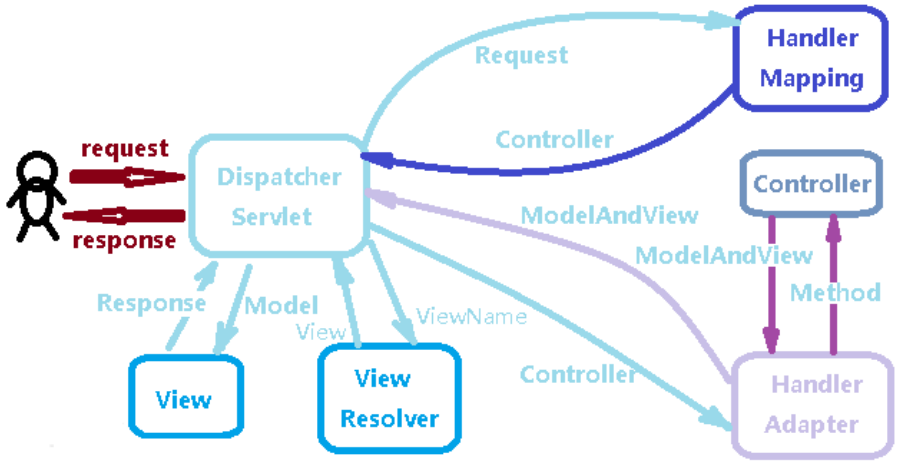

图 2-4 SpringMVC 框架模型图[24]

2. 数据持久层框架对比及选择

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。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。

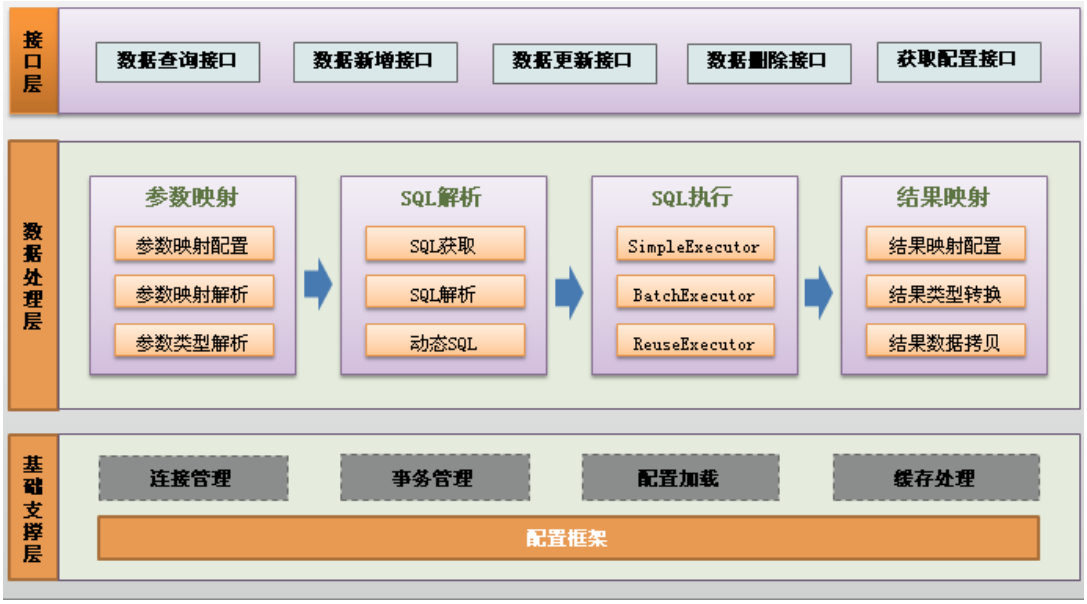

图 2-5 MyBatis 架构图[25]

MyBatis 框架分为三层：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，使用 Mapper 接口满足面向接口编程的需要，据简单的 API 对数据库进行增删改查。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。框架支撑层包括事务的管理、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[26]。

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，和数据库的交互变得简单起来。MyBatis 和 Hibernate 这两大框架在适用类型和成本方面的差异主要有以下几点：

- Hibernate 采用全自动对象关系型映射，Mybatis 采用半自动的映射。MyBatis 可通过配置文件手动编写 SQL 或使用注解进行 SQL 的动态构建，这种定制化的 SQL 构建方法根据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。

VMCloudPlatform 系统选用能够动态定制 SQL 语句的 MyBatis 框架，并将其整合 Spring、SpringMVC 进行整个系统的开发[27]。

2.3 Android 应用技术

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[28]。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

任何配置就可以使用代码打印日志。

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[29]。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，还有自定义开发的应用等。

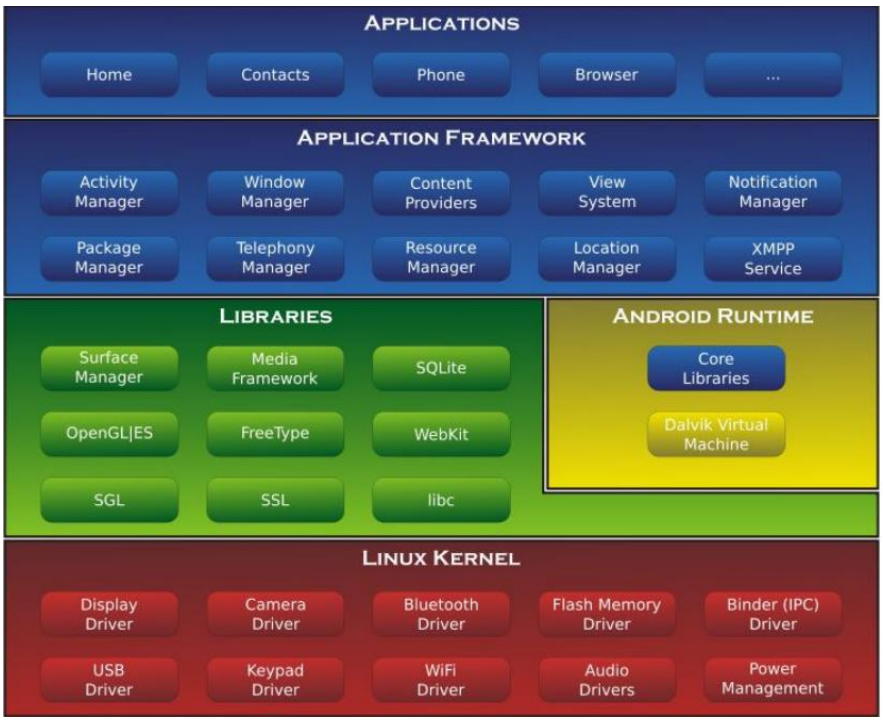

图 2-6 Android 系统体系结构[30]

Android 有四大组件[31]，分别是活动（Activity）、服务（Service）、广播接收器（Broadcast Receiver）和内容提供器（Content Provider）。

Activity 是 Android 开发的基础，所有用户界面和组件都运行在 Activity 之中，它提供一个窗口，用户通过窗口进行交互。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。

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[32]，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。开启 Service 有两种方式：startService() 和 bindService()。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。

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 可以实现程序间的信息共享，用于对外共享数据[33]。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。

2.4 Android 串口通信

串口通信技术是计算机和与其相连的设备之间的一种通信方式，仅通过一根传输数据线和信号线来实现传输过程[34]。串口通信有几个重要参数：波特率、数据位、停止位、奇偶校验等信息，在通信的过程中，每个串口指令都必须按照这些参数进行配置使用。利用串口通信的双方都是一次只接收或只发送一个数据位的信息，虽然和计算机内部数据的并行处理相比速度较慢，但是其通信成本低，能够实现远距离通信，因而在很多领域得以应用。

Android 系统的大部分产品只含有 USB 接口，很少含有串口，难以在工业控制和其他领域有所应用[35]。Android 设备可以通过串口线和外设相连，使用 Open Accessory 协议实现 Android 设备 USB 接口转串口，实现串口通信功能。Android Open Accessory 是 Google 在 2015 年推出的 Android 配件标准协议，是实现 Android 设备与外设通信的 USB 协议。外设作为 USB 主机，Android 作为 USB 设备，由外设为 Android 设备供电，并实现 Android 设备和外设的双向通信。

2.5 4G 无线通信

4G 无线通信技术是在前三代移动通信技术上发展起来的，集成了 3G 和 WLAN 的优点，具有高速、抗干扰和更兼容的优势。它建立在无线通信网络之上，用户可以在任何时间和地点使用 4G 网络，快速高效的传输音频、视频和图片等网络信息。在高速环境下，4G 网络能以 2~100Mbit/s 的数据传输，满足用户的日常信息传输的需求。使用 4G 实现无线网络的通信技术，能够摆脱传统网络平台的束缚。4G 无线通信技术的终端形式多样化，不仅可以使用在普通的手机之上，还可以使用在具有物联网功能的设备上[36]，能够增强设备的交互性。

传统自动售货机远程通信有人工管理、有线监控和无线网络监控三种方式[37]。人工管理主要靠手工摘录和对售货机逐一巡视的方式获取自动售货机终端信息，雇佣成本高、效率低、可靠性差；有线监控方式通过电话线、ADSL 或 485 总线将售货机连成一个相对集中的售货机群，管理范围较小、布线成本高；无线监控方式多通过 GSM 或者 GPRS 技术利用无线网络接入互联网，传输成本较高，传输量小。目前，目前 4G 网络技术发展成熟、数据传输速度快、与物联网技术结合紧密，可以代替 GMS 或者 GPRS 实现无线通信。自动售货机售货终端 VMSale APP 和 VMManage APP 通过 4G 无线通信技术将售货机联网，将孤立的售货机连接起来，把售货机信息及时传送到 VMCloudPlatform 管理系统[38~39]。管理员通过 VMCloudPlatform 管理系统实时监控终端售货机。

第 3 章 VMCloudPlatform 系统的需求分析

基于 SaaS 模式的 VMCloudPlatform 系统包括 VMCloudPlatform 管理系统和终端售货软件 VMSale APP，还有便于运营商营业员使用的 VMManage APP 应用程序。VMCloudPlatform 系统由厂商运营管理，供众多运营商使用。

3.1 VMCloudPlatform 系统的总体目标

基于 SaaS 服务和多租户的思想，使用 SSM 框架设计并实现一套能够供多个租户使用的自动售货机管理云平台；基于 Android 实现终端售货的 VMSale APP 和便于营业员使用的 VMManage APP。通过 4G 无线网络通信将售货机联网，将自动售货机终端信息同步到 VMCloudPlatform 管理系统，以实现对售货机的监控。系统完成应达成以下目标：

1. 完整性和可用性

系统设计应满足租户的全部功能需求和非功能需求，确保租户使用系统进行正常的日常管理，这是系统验收的首要标准。

2. 通用性和重用性

系统模块应具有通用性和重用性，能够适应各个运营商；系统设计过程中必须充分考虑各个模块之间的接口，添加必要的代码注释，保证各个模块之间功能清晰、交互正常，能够进行必要的扩展和更新。

3. 可移植性和健壮性

应在不同版本的 Android 外接设备上稳定运行，当异常退出时，能够进行自我处理机制，这是售货机终端 VMSale APP 应用应该具备的容错性功能。

结合 SaaS 服务、多租户技术、SSM 框架和 Android 技术的使用，VMCloudPlatform 系统的具体实现目标如下：

1. 实现多租户

基于 SaaS 服务进行开发，实现多租户共享平台，降低企业的信息化成本以及缩短信息化周期，解决中小型企业的信息化问题。

2. 改造老旧售货机

使用外接 Android Pad 控制自动售货机的工作流程，消费者在终端通过触摸屏选货，并使用移动支付方式完成商品支付。VMSale APP 通过 4G 网络传送售货机终端商品销售信息。运营行管理员从 VMCloudPlatform 管理系统对售货机库存和状态进行监控。

3. 终端软件智能化

实现支付宝、微信及银联移动支付功能，适应用户消费习惯的转变，减少新生代用户的流失。

4. 运营商营业员小程序

实现供运营商营业员使用的售货机管理软件，帮助营业员在上货现场更新自动售货机信息和个人库存。

3.2 VMCloudPlatform 系统需求分析

3.2.1 VMCloudPlatform 管理系统需求分析

VMCloudPlatform 管理系统由厂商开发和管理，运营商作为租户使用。厂商除作为系统管理员外，还有内部的业务管理模块。厂商包括两种用户角色：系统管理员和厂商管理员，进行系统的权限管理、用户管理、租赁管理、定价管理、售货机管理和货道管理等。运营商包括五种用户角色：运营商管理员、用户组管理员、营业员、库存管理员和财务管理员，进行用户管理、售货机管理、货道管理、商品管理和售货机维护等。运营商租用平台系统时，根据实际需求选择要租用的功能单元，为每个新建用户分配权限，使其操作不同的功能单元。运营商通过 VMCloudPlatform 管理系统对自动售货机终端进行实时监控，及时获取售货机销售状况，以保证售货机终端正常运行。

1. 自动售货机厂商需求

VMCloudPlatform 系统的开发和搭建工作由厂商负责，供自动售货机运营商使用，运营商通过 web 页面进行系统的租用。其中，厂商为一级用户，包括系统

管理员和厂商管理员两种用户角色。

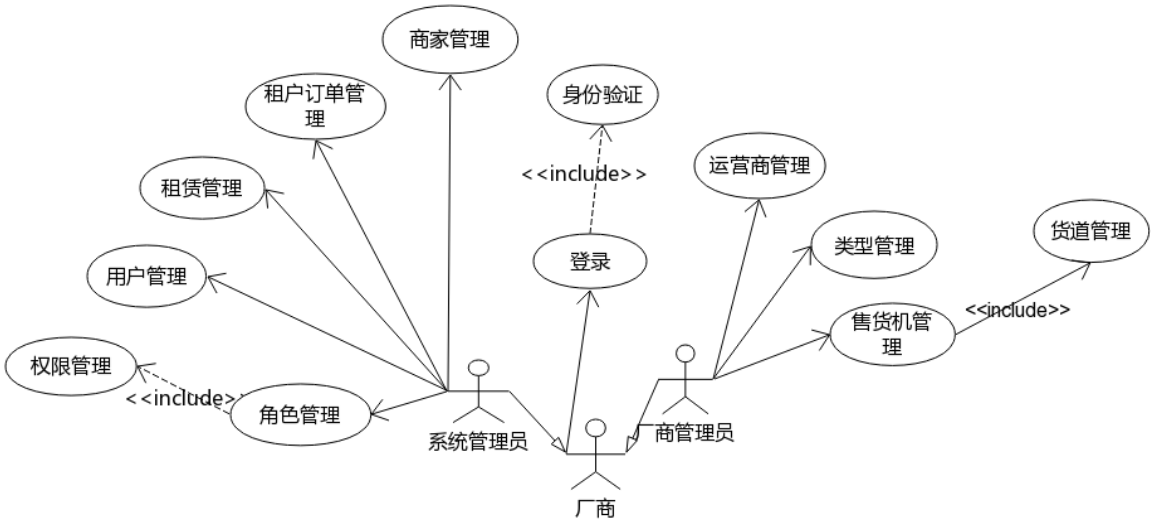

图 3-1 厂商模块系统用例图

平台注册/登录：平台仅有一家厂商账号，厂商是平台的管理者。

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，每个用户角色包含一到多个权限。权限所属类型分为两种：厂商和运营商。

用户管理：用户信息的管理，系统管理员添加、修改、查询和删除用户信息。

租赁管理：系统管理员定制 VMCloudPlatform 管理系统的租金规则。

租户管理：系统管理员对已注册运营商进行查询、添加、删除等管理操作。

类型管理：对售货机基本类型进行定义，查询、更新和删除，将售货机信息和类型信息进行关联。

售货机管理：对厂商售货机进行添加、修改、删除管理，初始化售货机基本信息和售货机类型，将售货机分配给运营商。

2. 自动售货机运营商需求

售货机运营商是 VMCloudPlatform 管理系统的主要使用者，通过租用界面进行平台的可配置租用。软件租用成功后运营商管理员添加下级管理员并配置用户权限。运营商使用系统，信息化的进行用户管理、售货机管理、货道管理、商品管理、财务管理和库存管理等功能。其内部模块和其他运营商相互隔离，互不干扰。

平台租用：运营商通过 web 访问入口进行平台的租用和管理员的初始化，租

用页面中，运营商根据企业内部的实际需求选择需要的功能单元，实现“按需租用，按需付费”。平台租用成功后，数据库存储租用的开始、截止日期、功能组，并将租户可用状态置为可用。

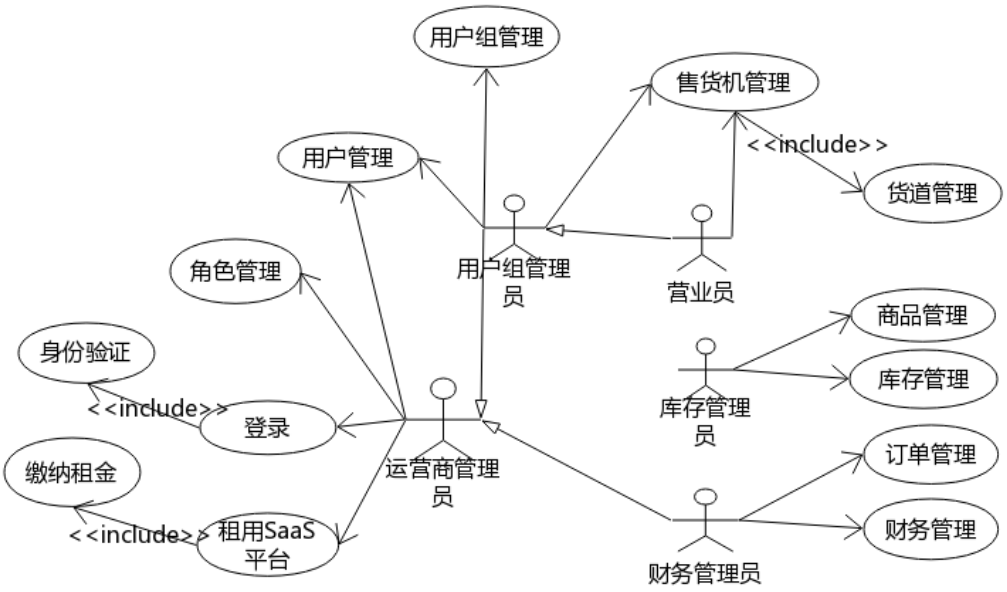

图 3-2 运营商模块用例图

用户管理：运营商管理员管理用户信息、用户组信息、分配用户角色信息。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。为方便管理，管理员创建用户组，为每个用户组分配一个组管理员，组管理员拥有管理下级营业员的权限。

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。管理员对售货机进行基本的添加、修改、删除、修改操作，并将机器分配给营业员，管理并分配机器组给营业员。管理员添加售货机同时完善货道信息，货道信息存储了该货道的容货量、当前库存量、在售商品等。对于销售相同商品的货道可根据需求将其添加至一个货道组，以便于批量操作。

商品管理：商品管理员对商品进行添加、修改、删除、修改操作。

库存管理：库存管理员包括出库信息查询、仓库出货、商品调拨、清单打印等模块。营业员填写调拨单，请求分配商品；仓库管理员查看调拨单，查看出库信息。

订单/财务管理：VMSale APP 工作过程中，每完成一笔订单就将订单信息进

行封装，发送到 VMCloudPlatform 管理系统，管理员进入管理页面查看自动售货机订单信息。财务管理员通过订单信息对每月的交易金额进行统计。

3.2.2 VMManage APP 需求分析

VMManage APP 是为方便运营商营业员运营管理、提高其工作效率、减少人工使用而开发的一套移动端应用。它能够取代传统人工抄记的管理模式，通过 4G 网络更新售货机数据和货物的统计。营业员通过 VMManage APP 查看所管理的机器列表、货道存货量、查看个人信息、查看个人库存和检测更新等操作。

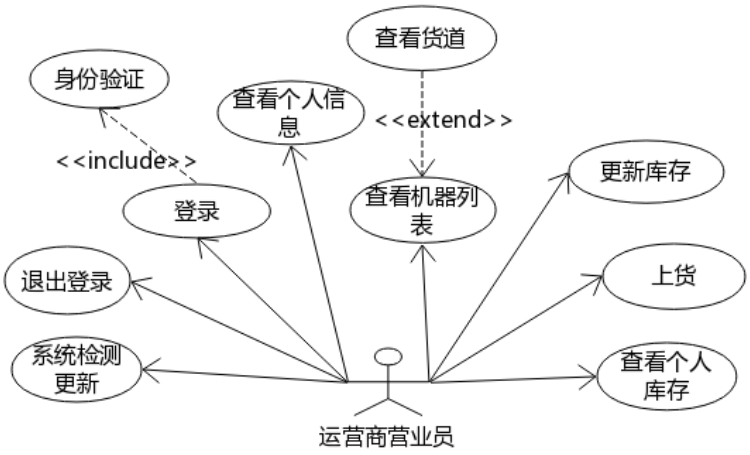

图 3-3 VMManage APP 用例图

用户登录：营业员及其上级用户能够使用 VMManage APP 软件，用户输入商家编号、用户名和密码，信息发送到应用服务器，验证通过后登陆成功。

售货机查询：提供查询售货机详情、货道列表、货道库存量、在售商品和销售价格的功能。

货道库存更新：营业员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，更新当前售货机的库存。

版本更新：VMManage APP 提供版本更新选项，系统检测到新版本 APK 文件时进行下载更新。

3.2.3 VMSale APP 需求分析

老式自动售货机由物理按钮、投币模块、计价模块和出货模块构成，消费者

通过物理按钮选货，投入硬币，然后售货机投币模块和计价模块判断是否出货，最后出货模块使用驱动电机控制商品出货。老式售货机使用按键选货和单一的投币式支付方式，没有先进的通信条件，很难对当前功能进行拓展，售货机的运营和管理完全依赖人工，造成大量的人力成本的耗费。

目前，市面上出现了一些基于 Windows 或者 Linux 平台的新型自动售货机，它们提供了可视化的选货界面、多元的支付方式、增值的电子广告和信息化的后台管理系统，并取得了良好的成效。但是研发一套信息化的系统或更换新型自动售货机的成本较高，一搬中小型企业并无力承担。因此，VMCloudPlatform 系统使用较为廉价的 Android Pad 作为外接设备和自动售货机相连，实现自动售货机的信息化管理和运营。

VMSale APP 基于 Android 开发，运行在 Android Pad 中，为消费者提供选货和移动支付，将分散的售货机连网，与 VMCloudPlatform 管理系统进行及时交互。VMSale APP 通过串口和自动售货机通信，控制售货机的售货流程，使用 4G 网络和 VMCloudPlatform 管理系统进行通信，实时更新数据。

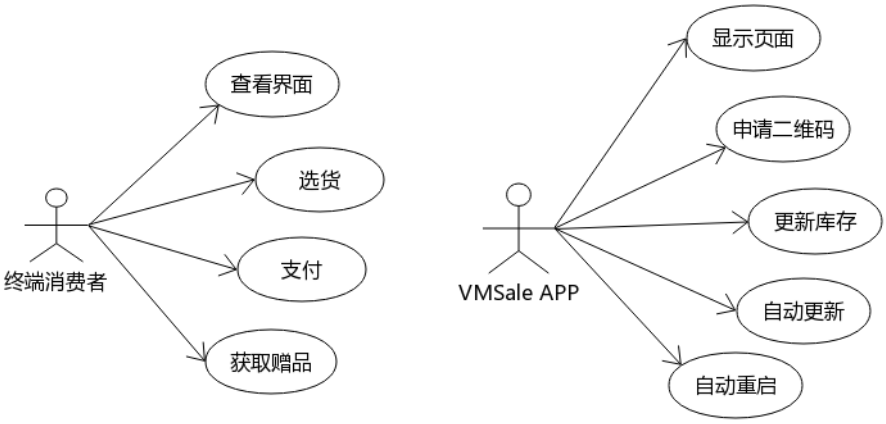

图 3-4 售货终端用例图

选货交易：在显示屏中展示选货页面，并提供支付宝、微信、银联移动支付方式，支付成功后进行出货操作。交易成功后，将交易信息封装发送到后台系统中，并更新相应库存。

赠品服务：支付赠送商品后系统产生一个赠品码，输入赠品码在任意一台售货机换赠同等价位的商品。

电子广告：无人操作时播放商家提供的广告。

货道监控：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，在页面中网格形式显示。

自动更新：自动售货机的发展越来越快，消费者的消费习惯不断变化，终端 VMSale APP 软件也随之进行更新。VMSale APP 安装在远端，需要定期检测文件服务器上 APK 版本，自动下载和静默安装。

自动重启：因软件本身或者设备的问题，终端 APP 使用的过程中可能出现异常退出的情况，需要提供一个自动重启软件的功能。

第 4 章 VMCloudPlatform 系统的设计

VMCloudPlatform 系统包括 VMCloudPlatform 管理系统和终端售货软件 VMSale APP，此外还附带一个方便营业员的 VMManage APP 小程序。本章将从系统的逻辑架构、技术架构对 VMCloudPlatform 系统进行详细的分析，并给出数据库的设计。

4.1 VMCloudPlatform 系统的逻辑架构

4.1.1 VMCloudPlatform 系统的功能架构

VMCloudPlatform 管理系统的功能组织结构如图 4-1 所示，图中描述了系统的功能模块和组织结构，为方便获取系统的主要功能，该图对于细小的功能模块并未穷尽列出。VMSale APP 和 VMManage APP 的组织结构如图 4-2 所示。

VMCloudPlatform 管理系统分为两个模块，分别是厂商模块和运营商模块。各模块实现的功能如下：

1）厂商是系统的开发者和管理者，该模块主要进行系统初始化、预定义和管理的功能，同时还提供了厂商内部运营所需的管理功能。包括系统的用户管理、租金信息管理、权限管理、角色管理、租户管理、售货机管理、类型管理、订单管理和运营商管理。租户管理是对使用系统租户的基本信息进行存储和查询，实现租户间的数据隔离，保证租户间的数据安全。角色管理定义用户的权限，设定角色和权限之间的组合关系为用户授权。

2）运营商模块为系统的主要模块，供运营商使用。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、库存管理等功能。其中用终端监控模块维护自动售货机终端的库存和 VMCloudPlatform 管理系统的后台数据库信息保持一致。运营商管理员及时获取售货机终端的销售信息，查询缺货售货机的 ID、地理位置等，使过营业员不去实地勘测也能及时对缺货售货机补货。

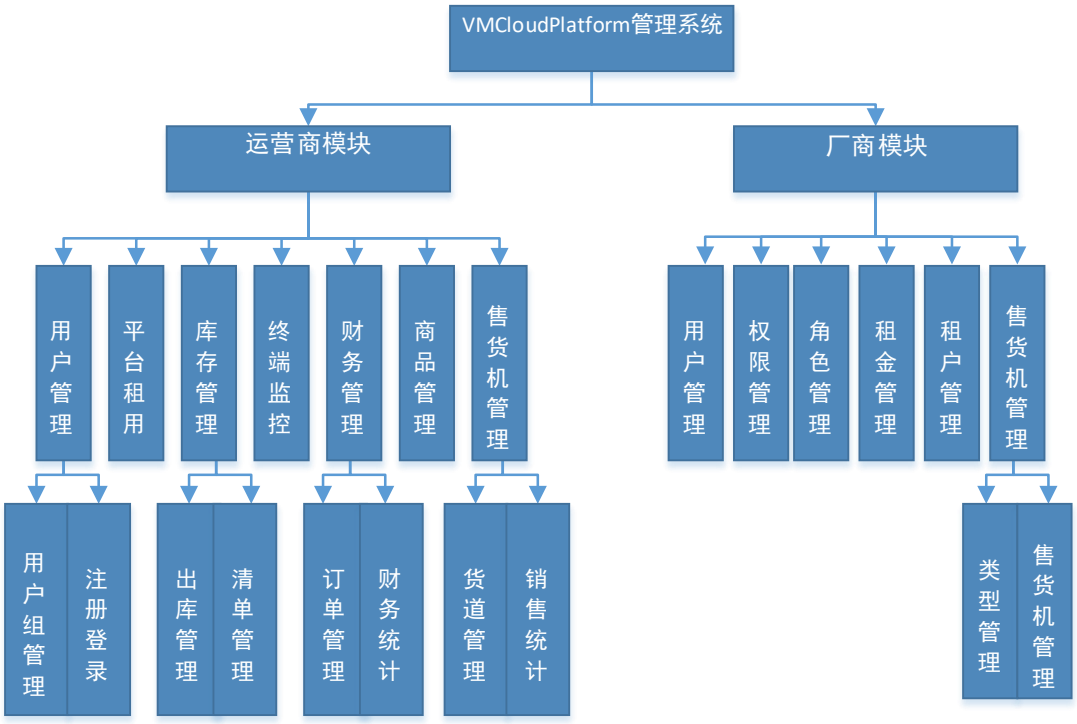

图 4-1 VMCloudPlatform 管理系统功能结构图

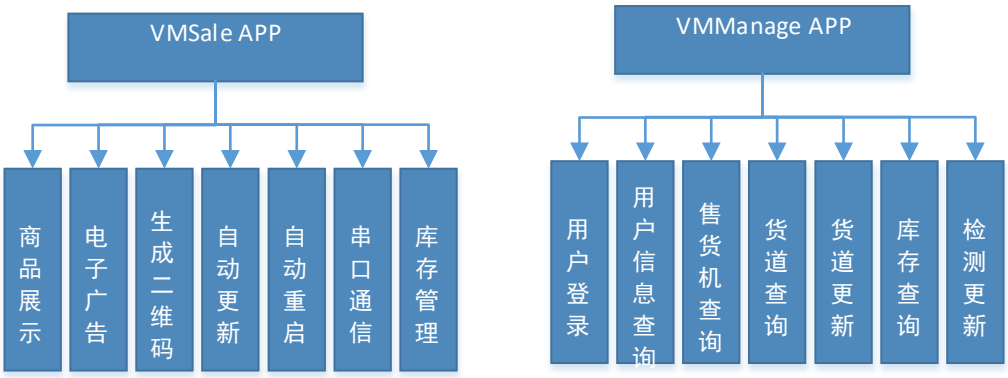

图 4-2 VMSale APP 和 VMManage APP 的功能结构图

VMSale APP 是自动售货机的终端售货软件，主要为终端消费者服务。它包括商品展示、电子广告、生成移动支付的二维码、自动更新软件、自动重启软件、串口通信和库存更新的功能。

VMManage APP 是运营商营业员使用的软件，旨在提高营业员的工作效率、减少人力成本的使用。主要包括用户登录、用户信息查询、售货机查询、货道查询、库存查询和检测更新功能。

4.1.2 VMCloudPlatform 管理系统的逻辑分层

VMCloudPlatform 管理系统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，是一个单实例的数据模型。系统使用多个租户注册和模拟多租户的功能实现，租户登录时根据登录表单信息进行租户的角色和权限判定，进入相应的租户空间。VMCloudPlatform 管理系统通过 web 浏览器为系统管理员、租户和自动售货机终端提供访问权限，终端系统使用 4G 网络发送数据，调用平台系统接口，和后台系统进行交互。系统总体逻辑架构图如图 4-3，通过该图对系统的整体业务逻辑进行简单的介绍。

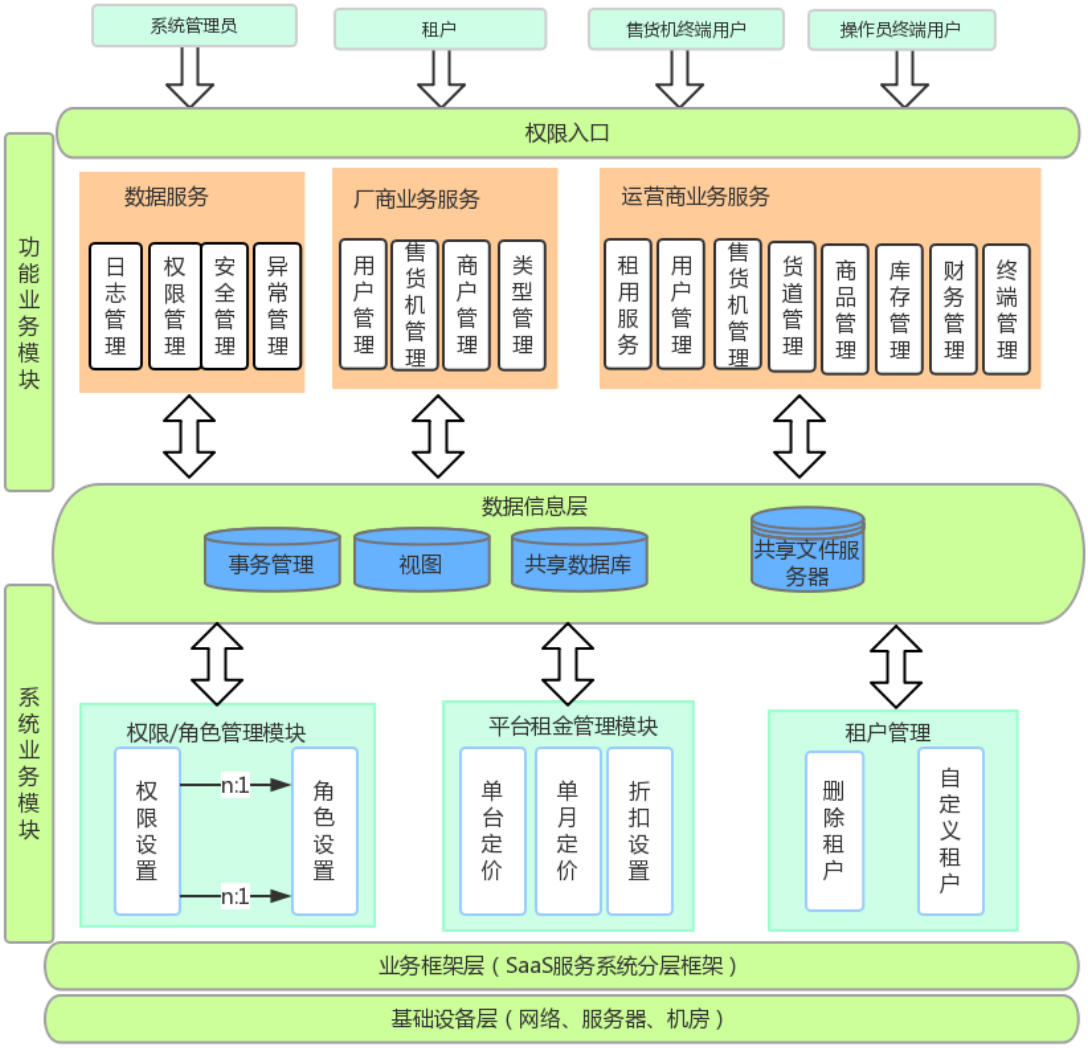

图 4-3 VMCloudPlatform 管理系统总体逻辑架构

整个平台系统分为多层：最底层为基础设备层和业务逻辑层，包括系统的硬件、网络和系统配置。平台的业务模块分为两类，一类是系统业务、另一类是功

能业务。权限管理、平台租金管理是 VMCloudPlatform 管理系统的系统业务模块，为所有用户共用，整个运营商模块都依托该层的功能；厂商业务服务、运营商服务和数据服务为功能业务，分别管理厂商、运营商的日常运营工作和系统的公共数据维护；平台上层为权限控制入口，供平台使用者进入，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[40]。

4.2 VMCloudPlatform 系统的技术架构

4.2.1 自动售货机的硬件构成及关键技术

改进的自动售货机终端外接 Android Pad，使用串口线和售货机相连，通过串口信息的收发，控制自动售货机的工作流程。智能化终端软件 VMSale APP 可以搭载在 Android Pad 上，由此实现自动售货机的低成本改造。

1. 自动售货机终端硬件构成

图 4-4 为改造后的自动售货机硬件结构图，主要是在传统硬件结构[41]上添加 Android 设备。各个模块之间的功能如下：1）MDB 总线通信接口用于协调 VMC（自动售货机主控制器）与多个外设之间的通信，负责现金或非现金交易，是自动售卖系统的内部总线协议；2）MCU（中央处理单元）负责信息的加工处理，是自动售货机的核心部分，控制自动售货机的整个销售流程；3）电机驱动模块配合 MCU 操作，根据所选商品的货道号，主控板驱动相应的电机运转，将货物推送到客户手中；4）RS232 是自动售货机的扩展异步串行接口，使用 UART 串口和自动售货机内的 MCU 连接，自动售货机的外接 Android 设备通过 USB 接口转串口的 FT312D 芯片和 RS232 接口连接，通过串口消息收发控制整个售货机的售货活动 5）信息采集模块，主要包括门开关、温度检测、制冷等。

2. VMSale APP 实现的关键技术

在自动售货机终端软件设计的过程中面临三个方面的问题。首先，售货机终端使用外接 Android 设备的方式进行改造，需要考虑 Android 设备的供电问题以及 Android 设备和自动售货机的串口通信功能；然后，自动售货机终端的 Android

设备放置在各个售货机网点，难免因为一些无法预知的问题导致软件的错误，需提供相应的应对措施；最后，VMSale APP 根据自动售货机的销售情况和当前技术发展的趋势不断更新，当文件服务器上有新的 APP 版本出现时，需要对零散分布的自动售货机终端进行软件系统的自动更新。总体来说，VMSale APP 涉及以下几个关键技术：自动售货机终端 Android 设备和售货机的串口通信，Android 设备的持续供电；终端应用异常退出后不通过人工操作自动重启；VMSale APP 更新时，不通过人工操作的方式在自动售货机终端进行自动更新。

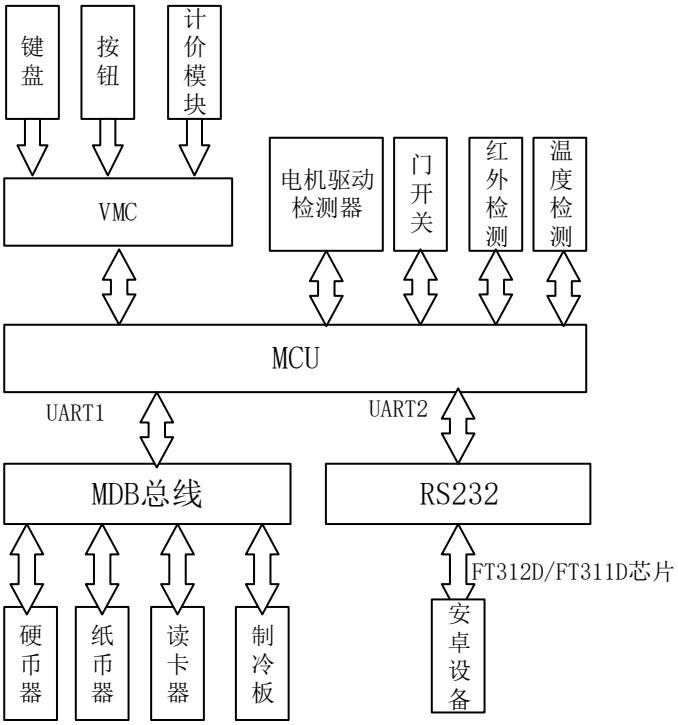

图 4-4 自动售货机硬件结构图

4.2.2 联网自动售货机的网络架构

联网自动售货机的网络结构如图 4-5 所示，系统由自动售货机销售终端 VMSale APP、VMCloudPlatform 管理系统和 VMManage APP 三个部分构成。VMSale APP 通过串口和自动售货机通信，通过无线网络和管理系统交互。系统还涉及三种支付平台：银联支付、微信支付、支付宝支付。

自动售货机销售终端完成售货机的正常销售活动，并使用 4G 网络实现无线传输。消费者通过 VMSale APP 选货，并使用 4G 网络完成移动支付。销售信息

使用 4G 模块进行编码，发送给 VMCloudPlatform 管理系统。数据验证通过后，在后台数据库中保存，同时自动售货机终端也会保存一份数据，并标记发送状态，以便于实现数据的失败重传。后台数据更新成功后发送一个确认信号给终端系统进行反馈[42]，表示已正确接收数据，和终端系统交互成功。

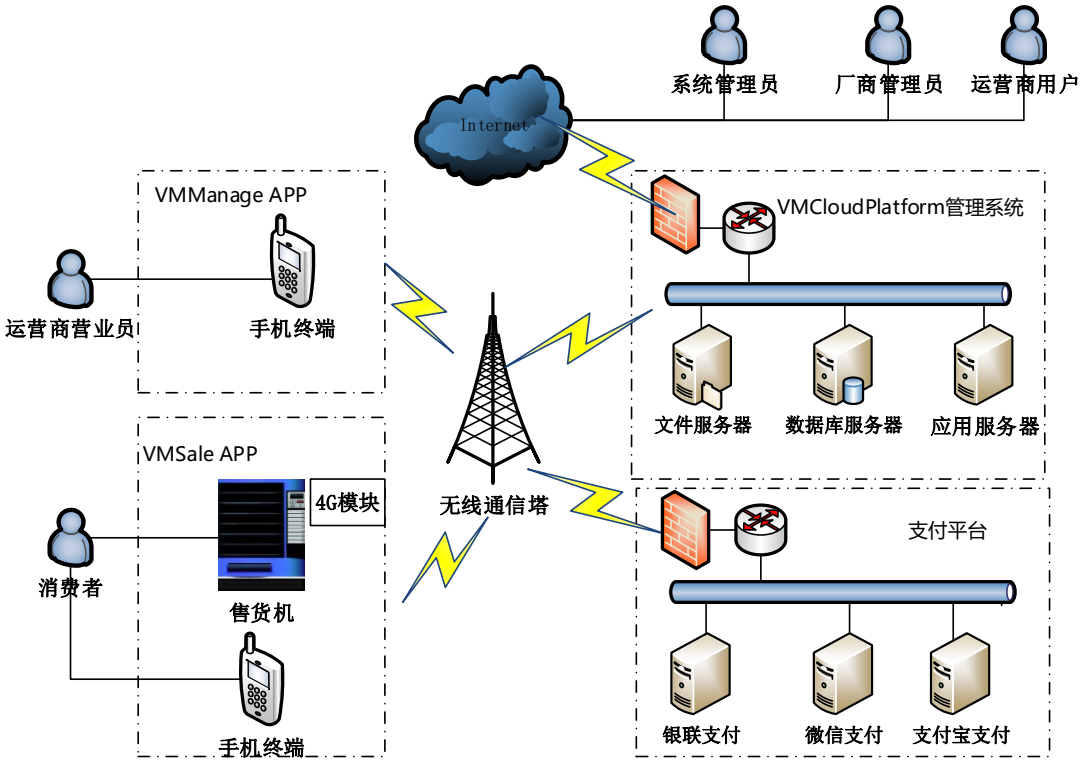

图 4-5 自动售货机网络拓扑结构图

自动售货机运营商可在 PC 管理终端通过 Internet 进入系统，查询各个网点自动售货机内库存、历史信息、订单信息，对其进行全面监控，在需要补货时直接定位到售货机的所处位置集中补货。

4.2.3 VMCloudPlatform 管理系统的软件体系架构

1. 分层的软件体系架构

现代的软件体系架构采用分层开发的设计思想，一个健壮的系统分层包括配置层、数据层、业务逻辑层和表示层，如图 4-6 所示。

软件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；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。

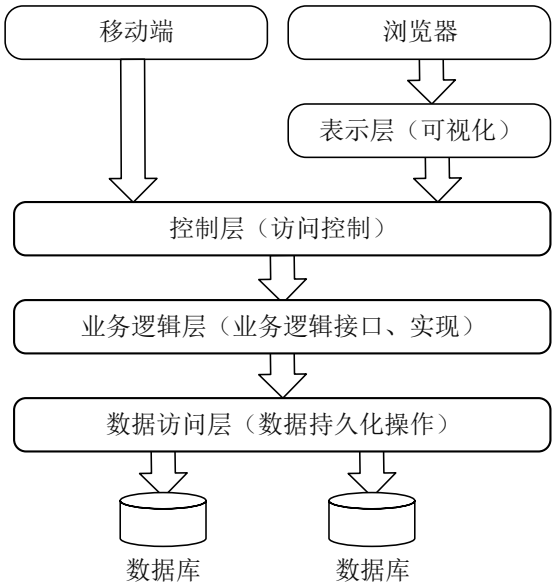

图 4-6 分层软件体系架构

- 数据库层：位于底层，主要任务是存放商家、用户的基本信息和的日常数据资源。
- 数据访问层：位于业务逻辑和数据库之间，用于访问和更新数据，并为上层 Service 提供调用接口。
- 业务逻辑层：系统的核心部分，是业务逻辑的具体实现，用于接收控制层的请求。主要定义各个数据模块的操作规则和统一接口，封装业务逻辑，方便系统的解耦，提高代码的利用率。
- 控制层：是表示层和业务逻辑层交互的桥梁，用于接收表示层或客户端的请求，并进行相应的处理和转发。
- 表示层：负责前台页面的展示，将客户端请求转发到控制层，控制层返回数据反馈给客户端，进行渲染，与 Controller 层结合紧密、协作工作。

2. 基于 SaaS 服务的分层软件体系架构

采用分层思想进行系统开发能够使软件开发过程耦合性降低、可维护性增强。VMCloudPlatform 管理系统将传统的分层体系架构和 SaaS 服务相结合，开发支持多租户的软件服务平台，使其满足多重租赁、易扩展的要求。

如 2.1.1 节所述，多租户有三种隔离级别，分别是“独立数据库”、“共享数

据库，单独模式”、“共享数据库，共享模式”。清晰的表述了三种隔离级别的隔离性、安全性、成本和可配置性。“独立数据库”中每个租户都有单独的数据库，可以根据用户的特性进行个性化配置，是最理想化多租户隔离级别。“共享数据库，共享模式”所有租户共用一个数据库甚至是一张表结构，是共享性最高的多租户隔离级别。VMCloudPlatform 管理系统选用“共享数据库，共享模式”作为数据库的多租户开发模型，为运营商提供尽可能低廉的服务价格。该隔离级别能够最大限度的提高数据库的利用率，具有最低的硬件成本和系统维护的成本，当租户数量足够多时，其优势愈加得以体现。

如 2.1.2 节所述，SaaS 成熟度模型分为四级，达到成熟度模型 3 及以上级别才真正实现了 SaaS 模式，其中最理想的成熟度模型为第 4 级。但是 SaaS 成熟度模型的选则要根据系统设计的目的、主要用户群体、用户真正需求和开发成本等诸多因素进行分析，而不是直接采用最高级别的设计模式。VMCloudPlatform 系统所采用的是 SaaS 的第 3 级成熟度模型，如图 4-7 所示。

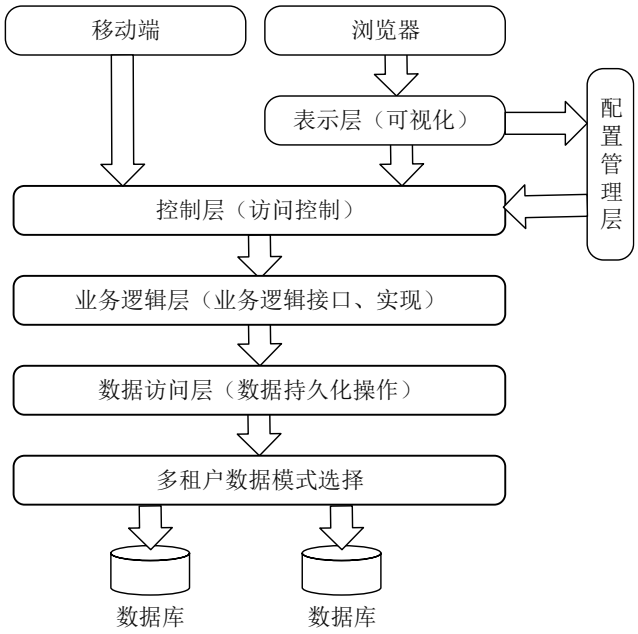

图 4-7 基于 SaaS 服务的分层体系架构

与一般的软件分层架构相比，基于 SaaS 服务的分层体系架构在表示层和控制层之间添加了配置管理层，在数据访问层和数据库之间添加多租户模式选择层[43]。配置管理层为租户提供个性化的配置管理功能，通过预留扩展字段实现在单

个应用实例中的功能控制，有效减少传统数据库资源的浪费和数据表的冗杂。传统软件系统由用户一次性买入，软件开发商获取用户的详细需求，针对用户进行软件的定制化开发，开发具有一定的成本要求。而基于成熟度模型 3 的 SaaS 系统架构根据不同租户可能需要的特定功能，主动在数据库中为系统提供预留字段，使系统具有一定的灵活性和可配置性。多租户模式选择层主要根据多租户模式的选择和数据库的设计，减少代码的重复性操作，对租户查询的信息进行过滤，保证用户不会访问到其他租户的数据。增加扩展字段，增强共享系统的扩展性、减少代码的大幅修改，降低开发的维护成本。

4.3 VMCloudPlatform 管理系统的数据库设计

1. 数据库概念结构设计

VMCloudPlatform 管理系统使用“共享数据库，共享模式”的多租户隔离级别，其核心是解决数据隔离的问题和实现多租户。在 SaaS 服务中建立多租户模型时，需要在每个业务表中增加 TENANTID 字段或者 FIRMID 字段，用来区分各个租户的信息，实现数据隔离，保证各租户见的数据隐私，如表 4-1 为一个简单的多租户表。这是在传统的软件模式中没有的字段，在 SQL 语句中使用 TENANTID=?进行租户数据的区分，防止读到其他数据。

表 4-1 多租户业务表示例

USER_ID	USER_NAME	USER_PASS	FIRM_ID
001	ZhangSan	123456	00101
002	LiSi	123456	00102

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）。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。VMCloudPlatform 管理系统的 E-R 图如图 4-8。

2. 数据库逻辑结构设计

- 商家（商家 Id，商家编号，功能 Id，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）

- 扩展表（数据 Id，基本表表名，配置表 Id，扩展字段值）

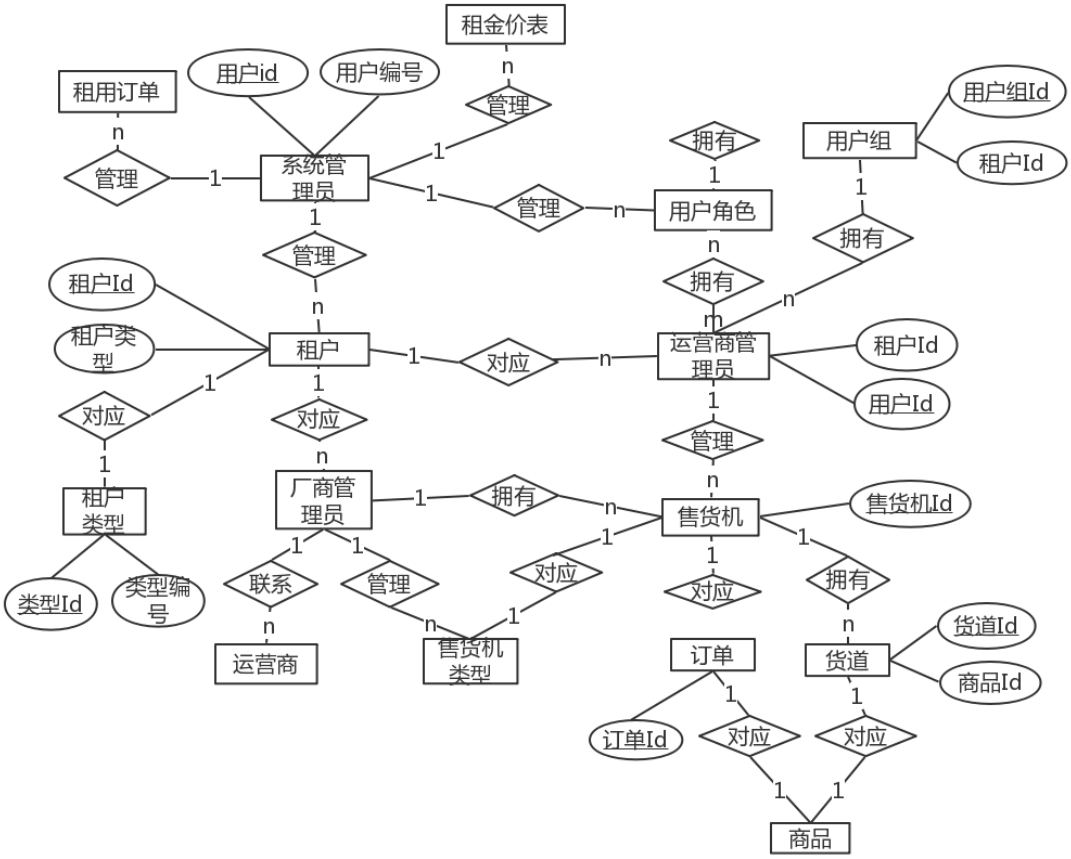

图 4-8 VMCloudPlatform 管理系统 E-R 图

- 扩展表配置表（配置表 Id，租户编号，基本表表名，扩展字段类型，扩展字段名）
- 功能表（功能 Id，基本模块 Id）
- 基本模块表（基本模块 Id，模块名称，模块描述）
- 租户功能表（租户编号，基本模块 Id）
- 平台租金（定价名，对应价格，折扣量）
- 基本模块租金表（基本模块 Id，模块名称，模块租金）
- 数据库加密密钥（密钥 Id，密钥）
- 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）
- 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，营业员 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，营业员，运营商 Id，上缴时间）

第 5 章 VMCloudPlatform 系统的实现与测试

VMCloudPlatform 系统的实现涉及后台管理系统和终端应用软件，以及自动售货机的改造。本章从系统实现的关键技术、自动售货机厂商和运营商管理系统以及 VMSale APP、VMManage APP 软件的实现进行介绍。

5.1 系统开发环境和开发工具

系统的整体设计流程基于软件工程的设计思路，VMCloudPlatform 管理系统基于 SaaS 服务和多租户技术的思想，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。VMSale APP 和 VMManage APP 基于 Android 进行开发，VMSale APP 通过 Android 串口通信实现和自动售货机的交互。

表 5-1 开发环境及工具

开发环境	工具及版本
操作系统	Windows10、Android
数据库	MySQL5.5、SQLite
服务器	Tomcat7.0.73
浏览器	Google、Firefox、IE 9
运行时环境	JDK1.8
开发语言	Java、Android
集成开发环境	Spring Tool Suite、Android Studio
项目管理工具	Maven3.3.9
代码管理	GitHub
UML 系统建模	Microsoft Visio 2013
串口调试	serial port utility

5.2 关键技术和难点的实现

5.2.1 多租户数据模型

1. 多租户数据模型考虑的问题

VMCloudPlatform 管理系统使用“共享数据库，共享模式”的租户隔离级别。“共享数据库，共享模式”除拥有较高的共享性和较低的开发成本外，还需承担一定的风险。现需要重点考虑以下几个方面的问题：

- 数据扩展性：不同的租户在使用系统时，有不同的数据扩展需求。传统数据扩展方式在多租户模式下易造成数据资源的浪费，与系统的开发初衷相悖，需要实现 SaaS 多租户的数据扩展技术。
- 可配置性：传统软件实现方式或者独立数据库的隔离模式能够个性化的定制系统的需求，而 SaaS 作为一种共享服务强调的是“按需付费”，因此对于系统的可配置性就有了一定的要求。
- 数据的安全性：共享数据库中所有租户的信息都保存在一张表中，租户对于数据没有管理权限，因此需在数据的安全控制上进行设计，以确保租户的信息不被泄露，保证租户在使用系统的过程中不会访问到其他租户的信息。

2. 多租户数据模型关键问题的实现

（1） 数据扩展性实现

多租户平台系统中，每个租户对于存储内容会有差异性需求，可通过数据库的扩展技术，在满足租户需求的同时不影响其他租户的使用[44]。

普通软件采用定制列的方式在数据库业务表中实现数据的扩展，如表 5-2 所示。这种扩展方式实现简单，在传统应用中较为常见，但是在多租户实现环境中极易造成数据资源的浪费，导致基本业务表数据列急剧增加，严重破坏了表结构。比如，极端情况下，每个租户都有一个定制字段，而这一字段对于其他用户毫无意义。

表 5-2 定制列实例

USER_ID	FRIM_ID	USER_NAME	AGE	MARRIED	...	ADDRESS
001	00101	ZhangSan	25		…	
002	00202	LiSi			…	Shanghai

[45]中提出了一种使用预分配字段和预分配字段配置表的方式解决数据的扩展问题，数据库表包含租户 ID 字段、若干预留扩展字段和 XML 字段，能有效

减少资源的浪费，提高了字段的拓展性，给 VMCloudPlatform 管理系统数据扩展的实现提供了思路。

VMCloudPlatform 管理系统使用预分配字段表、预分配字段配置表对数据扩展方法进行改进。当业务表中需要扩展字段时，在扩展字段表中添加一个字段值，并在配置表中对扩展字段的基本信息进行定义，不会造成数据空间的浪费。

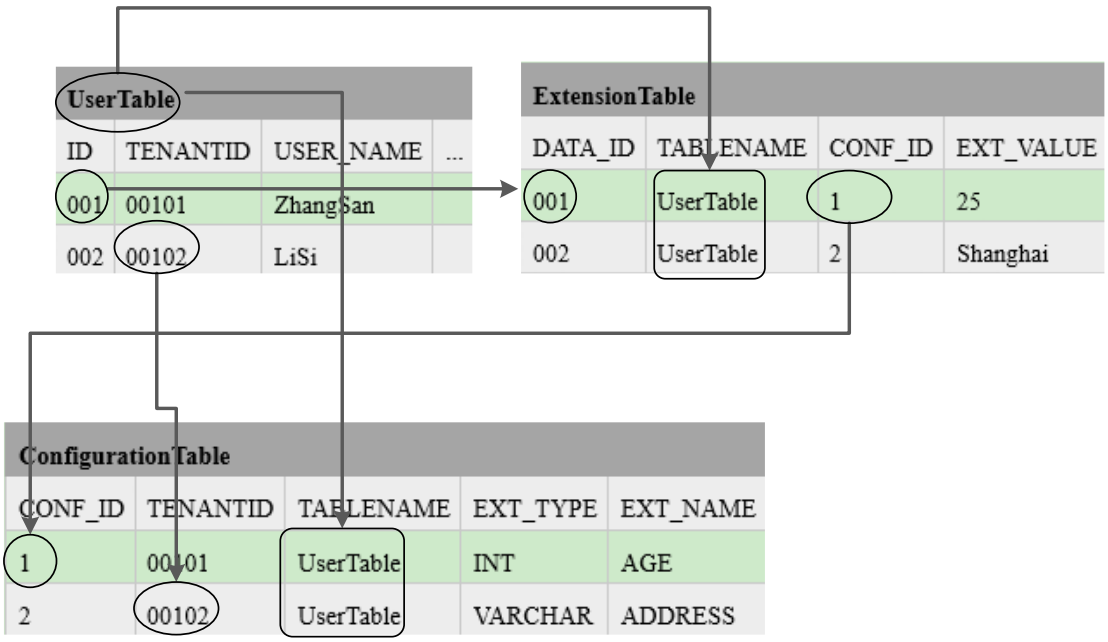

图 5-1 数据扩展模式

图 5-1 表示三表之间的对应关系，基本业务表主要存储租户的 ID 和通用信息，扩展表的 DATA_ID 对应基本表数据的 ID，EXT_VALUE 表示扩展字段的数据。配置表中配置 ID 和扩展表中 CONF_ID 进行关联，并在配置表中存储扩展字段的类型及名称。扩展表和配置表指明所扩展的基本表表名和租户的 ID。

（2） 可配置性的实现

SaaS 应用区别于普通应用还有一个显著的特征，就是“按需租用，按需付费”。系统开发的过程中将软件的功能结构进行拆分，拆成一个个独立的互不重叠的功能模块，租户在使用时可根据实际的使用需求和使用场景，定制化的选择功能集合。

通过基本模块表、功能表和租户功能表的结合使用实现系统可配置性的功能。如图 5-2 所示，租户基本信息表主要保存租户的基本信息，如租户 ID、租户编

号、租户名称和功能 ID 等，供登录时进行身份验证。基本模块表保存系统的基本子功能，功能之间互不重叠，MENU_ID 为基本模块表的 ID，MENU_NAME 表示模块的菜单项名称，MENU_DESC 是对模块的简述。功能表是连接租户基本表和基本模块表的中间桥梁，包括 FUNCTION_ID 和 MENU_ID，每一个 FUNCTION_ID 对应一到多个 MENU_ID，组成租户的基本功能模块，并将 FUNCTION_ID 作为唯一标识存入租户基本表中。租户功能表有两个字段 TENANTID 和 MENU_ID，分别表示租户的编号和租户对应的直接子功能模块。租户功能表的设定是为了减少租户登录时对功能的查询次数，可直接通过该表获取当前用户的所有功能，减少功能表、租户基本信息表和基本模块表之间的连接查询。

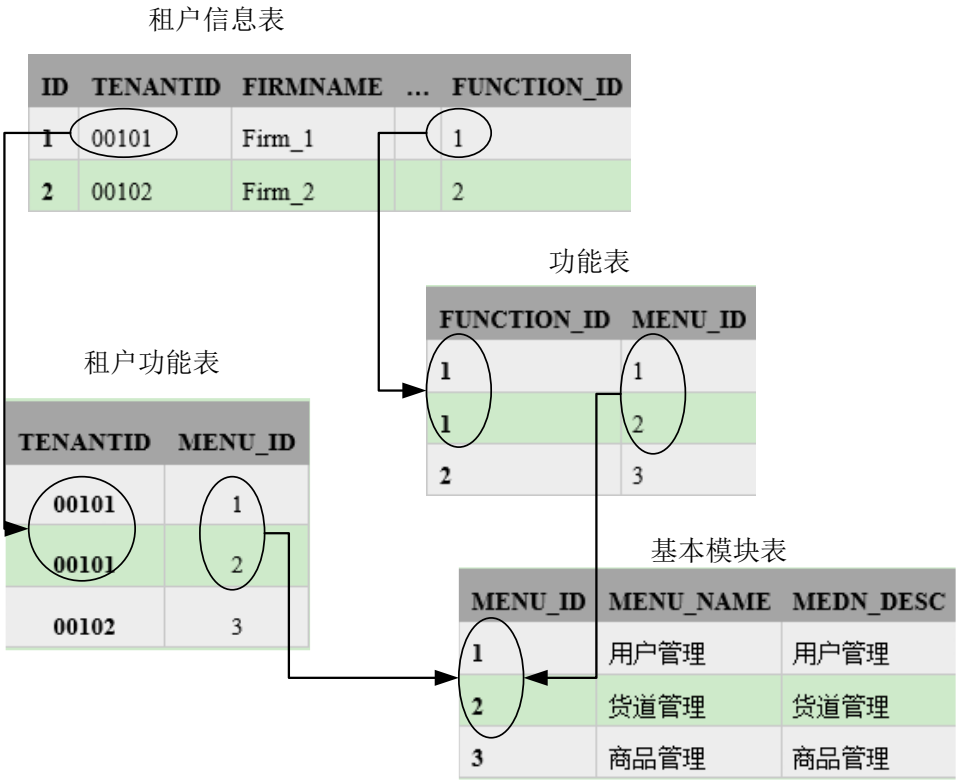

图 5-2 可配置租户功能关系表

（3） 数据库安全性实现

自动售货机云平台基于 SaaS 服务的思想设计实现，租户数据都存储在软件提供商的数据库服务器中，在使用的过程中对数据无控制的权限。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。系统使用数据过滤和数据加密两个方面保证数据的安全。

- 使用视图过滤租户信息

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[46]。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信息的访问，进而提高系统的安全性[47]。

```
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;
```

通过验证当前登录的商家 Id，建立了视图 ChannelView，查询结果只包含当前商家货道信息的部分行，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。

- 数据加密设计

数据库加密粒度可分为三种，分别是表级加密、记录级加密和字段加密[48]。表级加密是以数据库表为单位进行加密，这种加密方式效率低，对空间消耗大，因而并不常用。记录级加密以记录为单位进行加密，每次对一条记录进行读写时就进行一次加密解密。但是对一条记录而言，并不是每个字段都需要加密处理，因此该方法的加解密也较为麻烦。字段为单位的加密方法加粒度较小，在存取时只对需要加密的信息进行加解密处理，如用户密码字段，这种方式较为灵活。

在数据库端中使用 MD5、ENCODE 等加密方式进行加密。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，也无法解密，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');
```

5.2.2 Android 串口通信

1. USB 接口转串口方案

因为自动售货机终端使用 RS232 串口作为串行通信的总线标准，Android 设备只提供了一个 Mirco USB 接口，如何实现 Android Mirco USB 接口和售货机串口连接和对 Android 设备的持续性供电成为售货机改造的关键问题。目前 Android 通过 USB 转串口控制外设有三种方式[49]：1）驱动模式：Android 设备作为 USB 主机，使用 Java 的 JNI（Java Native Interface）机制和 Android 的 NDK（Native Development Kit）直接在 Linux 文件系统对外设进行操作[50]；2）主机模式：外设提供 USB 接口，Android 设备作为 USB 主机，使用 USB HOST API 直接控制 Android 的 USB 接口；3）配件模式：使用 AOA（Android Open Accessory）协议，通过特殊芯片的辅助控制外设。其中前两种方式使由 Android 设备为串口总线供电，最后一种方式由外设为 Android 设备供电。Android 设备 USB 接口转串口模式如图 5-3 所示。

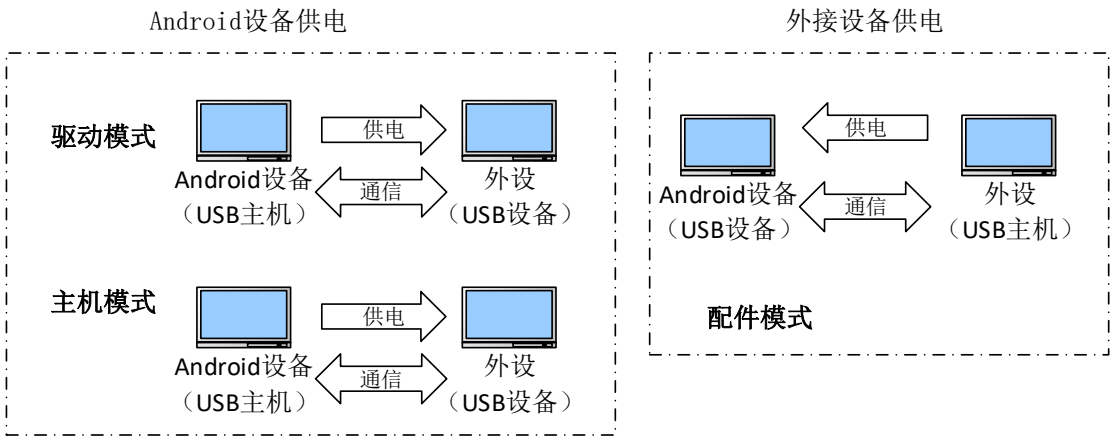

图 5-3 Android 设备 USB 接口转串口模式

驱动模式使用 Java 的 JNI 调用串口设备文件，需要开发者自行开发驱动程序且需要 root 权限。因 Linux 内核的源码各不相同，该方法兼容性并不高。

主机模式是通过 Android 系统的 android.hardware.usb 直接控制 Android 设备的 USB 接口，然后控制外设。可使用英商菲特蒂亚有限公司（FIDI）的 FT232 芯片，将自动售货机的 RS232 串口转换为 USB 接口。然后通过 OTG（On-The-Go）转换线将 Android 设备的 Mirco USB 接口转为 USB 母口，与 RS232 转换后的 USB 接口相连。该方式能够和售货机进行连接，不需要驱动程序的开发和 root 权限。但是 Android 设备只有一个 Mirco USB 接口，无多余接口为 Android 设备供电，待电量耗尽将无法为自动售货机提供持续服务。

配件模式使用了 AOA 协议，提供了 Android 设备和外设通信的 USB 协议，允许 Android 设备通过 USB 连接外设。实际实现过程中，可使用支持 AOA 协议的 FT312D 芯片将自动售货机 RS232 转换为 Mirco USB 接口，直接和 Android 设备相连。该模式不需要驱动程序的开发和 root 权限，也不需要额外的 OTG 转换线，外设作为 USB 主机，Android 设备作为 USB 设备，自动售货机作为外设为 Android 设备进行持续供电。

配件模式能够高效、简易的实现自动售货机终端和 Android 设备的通信，同时能够为 Android 设备持续供电。因此，自动售货机终端采用该模式实现 Android 串口通信，连接示意图如图 5-4 所示。

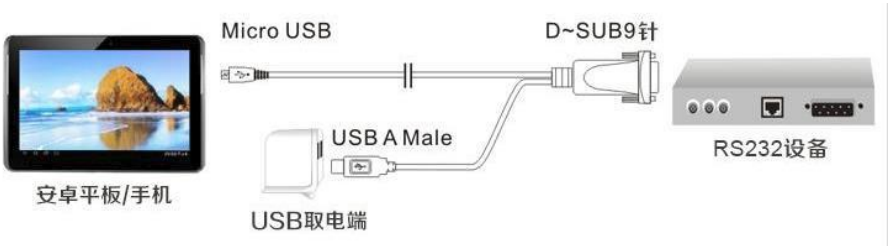

图 5-4 安卓设备和自动售货机串口通信连接图

2. 串口通信技术关键实现代码

使用配置模式实现 Android 串口通信，通信过程如下：首先，应保证自动售货机和 Android 外接设备以及 USB 转串口芯片都遵循 AOA 协议；接着，自动售货机连接 Android 设备，不断循环检测连接状态；然后，判断当前 Android 设备

是否支持 AOA 协议，若不支持直接结束，否则，判断是否处于 AOA 模式，若不在该模式，则发送请求尝试开启 Android 设备的 AOA 模式；最后，若 Android 设备处于 AOA，则建立通信。实现流程图如图 5-5 所示。

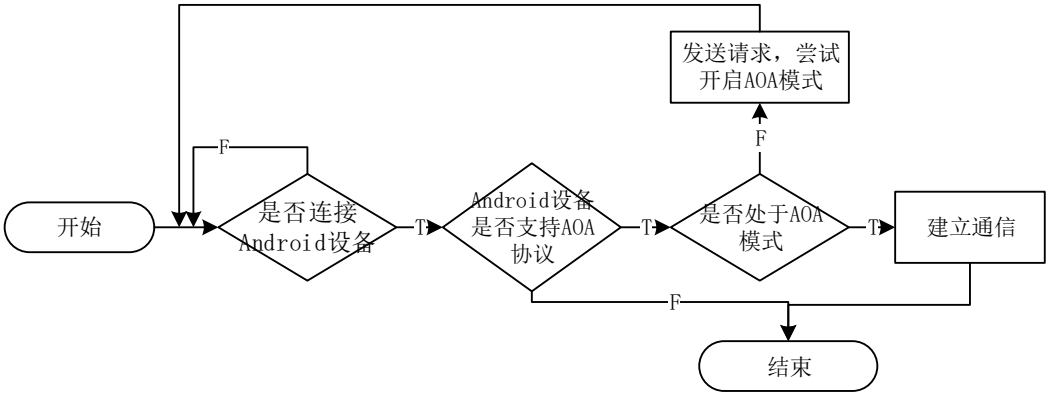

图 5-5 Android 设备 USB 接口转串口的实现流程图

实现代码片段如下：

```
//manifest.xml 配置 USB 权限
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.USB_PERMISSION" />
<uses-feature android:name="android.hardware.usb.accessory"/>
usbmanager = (UsbManager) context.getSystemService(Context.USB_SERVICE);
mPermissionIntent = PendingIntent.getBroadcast(context, 0, new Intent(ACTION_USB_PERMISSION),
0);
IntentFilter filter = new IntentFilter(ACTION_USB_PERMISSION);
filter.addAction(UsbManager.ACTION_USB_ACCESSORY_DETACHED);
context.registerReceiver(mUsbReceiver, filter);
//一旦检测到 Android 设备与自动售货机相连就打开 Android 的 AOA 操作
if (usbmanager.hasPermission(accessory)) {
    OpenAccessory(accessory);
}
public void OpenAccessory(UsbAccessory accessory)
{
    filedescriptor = usbmanager.openAccessory(accessory);
    if(filedescriptor != null){…
        inputstream = new FileInputStream(fd);//创建接收数据流
        outputstream = new FileOutputStream(fd);//创建发送数据流
        ……
    }
}
//创建和注册广播接收器来监听匹配特定过滤标准的 USB 事件尝试与自动售货机连接
```

```
private final BroadcastReceiver mUsbReceiver = new BroadcastReceiver(){…};
```

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。

3. 串口通信协议

配件模式实现 Android 设备和售货机的连接，随后就可进行串口消息的收发。串口消息报文格式如下：

表 5-3 串口消息报文格式

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

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 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。

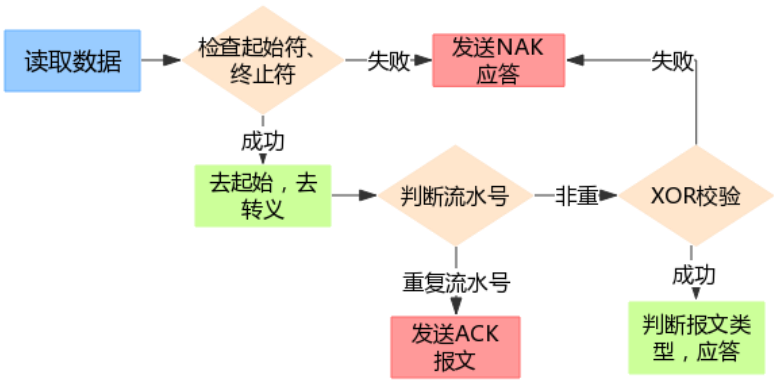

图 5-6 串口通信流程图

图5-6为串口的处理流程。串口消息处理过程中需要对串口进行封装和解析：1）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；2）将起始符和终止符去除，并删除转义符，再进行之后的验证；3）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；4）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；5）解析报文类型。

5.2.3 VMSale APP 自动更新/重启

1. VMSale APP 自动更新

自动售货机终端设备分布在不同的售货点，如机场、地铁站、学校、工厂等地，根据不同的客户群体，往往需要调整 VMSale APP 的功能模块。互联网技术不断发展，各种新型科技不断涌现，VMSale APP 也紧跟技术发展的趋势不断更新。这两个原因致使自动售货机终端软件应用需要不断的更新。传统 Android 设备使用者更新软件时直接下载新版本 APK，然后根据页面提示步骤逐步进行软件的更新和重启。但是自动售货机分散分布在各个站点，若由人工手动更新，其工作量大、耗费时间长、更新不及时、效率低下、人力成本较高，因此提出了自动更新 VMSale APP 的需求。VMSale APP 的自动更新分为两个步骤：APK 版本自动检测和 APK 静默安装。

自动检测是指软件在运行的过程中，定时检测文件服务器中 APK 配置文件，查看当前版本和服务器中 APK 的版本。若当前版本小于服务器上 APK 的版本，则解析配置文件，找到 APK 的下载地址，将最新的 APK 文件下载到 Android 设备上。

静默安装是指软件安装过程中无需用户的干预，直接按默认设计进行安装。Android 系统并未给出静默安装的方法和接口，普通的 Android 应用也不需要静默安装这一功能，但是自动售货机终端软件因其分散性和持续性服务的特殊性需要提供此功能。[51]提出编制模拟鼠标键盘操作的脚本实现软件的静默安装，受此启发，本文通过模拟人工点击 Android 触摸屏的 APK 应用，操作软件安装过

程中弹出的界面，实现 VMSale APP 的静默安装。

（1） APK 自动检测的实现过程

首先，在文件服务器放置最新版本的 APK 文件和保存 APK 信息的 JSON 文件。JSON 文件中保存 APK 的版本号、APK 名称、下载地址和更新内容等。具体信息如下：

```
{
   "download_path":"http://mileyrenpathsample/auto-update-version/vmsale-v1.0.apk",
   "apkName": "vmsale-v1.0.apk",
   "versionCode":1,
   "updateMessage":"[1]新增移动支付功能<br/>"
}
```

然后，解析 JSON 文件，将本地软件版本和服务器端 APK 版本进行对比，若有更新则通过异步 AsyncTask 下载新版本 APK 文件。实现代码片段如下：

```
//mainfest 配置权限
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
private void parseJson(String result) {
            int apkCode =    new JSONObject(result).getInt("versionCode");
            int versionCode = getCurrentVersionCode();
            if (apkCode > versionCode) {goToDownloadApk(apkUrl);}
}
```

（2） APK 静默安装的实现过程

模拟人工点击 Android 提示框的操作需要 Android 设备具备无障碍功能，该功能需事先在 Android 设备上开启。其实现原理是后台下载应用后，调用系统的 PackageInstaller 监听程序和安装页面，获得软件安装界面的按钮位置，然后通过 Android Accessibility（辅助功能）提供的模拟用户点击功能操作页面，代替用户执行安装步骤，完成软件无人干预的安装。具体实现步骤如下：

首先，在 Mainfest 中添加声明，配置 Android 的无障碍服务；接着，在 res 文件夹下创建 accessibility_service_config.xml 文件指向要监听的程序接口。

```
//Mainfest 配置
<uses-permission android:name="android.permission.BIND_ACCESSIBILITY_SERVICE"/>
<service    android:name=".update.MyAccessibilityService"      android:label="智能安装"
            android:permission="android.permission.BIND_ACCESSIBILITY_SERVICE"
```

```
        android:enabled="true" android:exported="true">
        <intent-filter>
          <action android:name="android.accessibilityservice.AccessibilityService" />
        </intent-filter>
        <meta-data    android:name="android.accessibilityservice"
                      android:resource="@xml/accessibility_service_config" />
</service>
```

然后，创建 AccessibilityService 获取安装界面的 UI 元素，模拟用户点击。

```
private boolean iterateNodesAndHandle(AccessibilityNodeInfo nodeInfo) {
        if (nodeInfo != null) {
            int childCount = nodeInfo.getChildCount();
            if ("android.widget.Button".equals(nodeInfo.getClassName())) {
                String nodeContent = nodeInfo.getText().toString();
                if ("安装".equals(nodeContent) || "打开".equals(nodeContent)) {
                    nodeInfo.performAction(AccessibilityNodeInfo.ACTION_CLICK);
                    return true;
                }
            } else if ("android.widget.ScrollView".equals(nodeInfo.getClassName()))
             nodeInfo.performAction(AccessibilityNodeInfo.ACTION_SCROLL_FORWARD);
                for (int i = 0; i < childCount; i++) {
                  AccessibilityNodeInfo childNodeInfo = nodeInfo.getChild(i);
                  if (iterateNodesAndHandle(childNodeInfo)) return true;
                }
        }
        return false;
    }
```

2. VMSale APP 自动重启

VMSale APP 安装在自动售货机的终端，由普通消费者使用。和普通应用相比，VMSale APP 不能进行实时的人工维护，难免会碰到一些不可预计的故障，导致异常退出。比如：消费者在使用 Android Pad 时出现操作不当的现象，导致自动售货机和 Android 设备之间的连接线松动，致使 VMSale APP 异常退出；Android 应用的一些不易察觉的问题未被穷尽测出，潜在 Bug 导致程序异常崩溃；Android 设备出现问题，导致程序的异常退出等。自动售货机提供 24 小时不停机的服务，任何一种异常退出都会影响自动售货机的正常销售，带来不可预估的损失。因此，需要进行必要的处理和重新启动。

首先，使用 Android 的 UncaughtExceptionHandler 接口捕获运行时异常，并

进行处理；然后，使用 android.os.Process.killProcess 退出进程；最后，使用 PendingIntent 重启应用。处理过程中把所有 Activity 添加至列表，当系统出现异常，就将 Activity 列表中的所有 Activity 实例关闭，然后 Kill 当前进程。正常应用使用 startActivity(intent)启动，但该方法在遇到异常时，若不能关闭异常进程，应用就需关闭两次，导致 startActivity 方法不能正常工作。本文使用 AlarmManager 启动 PendingIntent，在关闭开启的所有 Activity 实例和异常进程后就能实现应用的重启。实现代码片段入下：

```
// 创建服务用于捕获崩溃异常
    private class MyUncaughtExceptionHandler implements UncaughtExceptionHandler {
        @Override
        public void uncaughtException(Thread thread, Throwable ex) {
         Intent intent = new Intent(application.getApplicationContext(), MainActivity.class);
         PendingIntent restartIntent = PendingIntent.getActivity(
                 application.getApplicationContext(), 0, intent,
                 Intent.FLAG_ACTIVITY_NEW_TASK);
         AlarmManager mgr = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
         mgr.set(AlarmManager.RTC, System.currentTimeMillis() + 1000, restartIntent);
        finishAllActivity(); //结束所有 Activity
         android.os.Process.killProcess(android.os.Process.myPid()); //退出当前应用
        }
    };
```

5.3 VMCloudPlatform 系统的主要功能实现

5.3.1 自动售货机厂商功能

1. VMCloudPlatform 管理系统租户注册

VMCloudPlatform 管理系统由厂商进行开发和运维，供不同的运营商租用，运营商在该功能模块实现注册和初始化。平台注册功能使用了 Ajax 技术异步发送信息，对运营商输入的名称和编号进行合法性判定，并根据运营商选择的功能模块和租用时间计算租用所需的金额。

用户提交合法表单信息后，系统将运营商信息添加到租户基本信息表中，同时创建运营商账户和初始管理员账号。一个完整的租户基本信息表记录了租户的

基本信息、功能模块 Id、租用起止时间、租户可用状态等。具体实现页面如图 5-7 所示。

图 5-7 自动售货机云平台租户租用界面

2. 用户角色与用户权限

用户角色和用户权限的管理包括不同租户之间的权限隔离、同一租户内部的权限管理和用户角色的权限组合。用户的权限管理和角色管理由系统管理员负责，分为定义、管理和分配。运营商在用户创建的过程中，给用户分配角色。用户角色层级关系如图 5-8 所示。

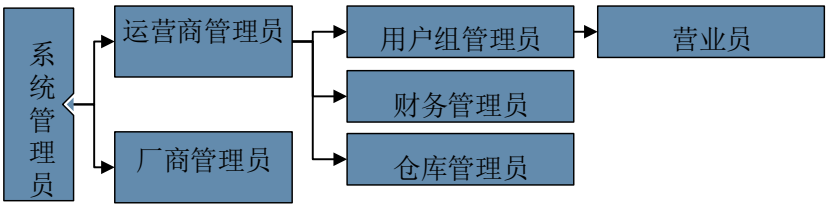

图 5-8 VMCloudPlatform 系统用户层级关系表

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。

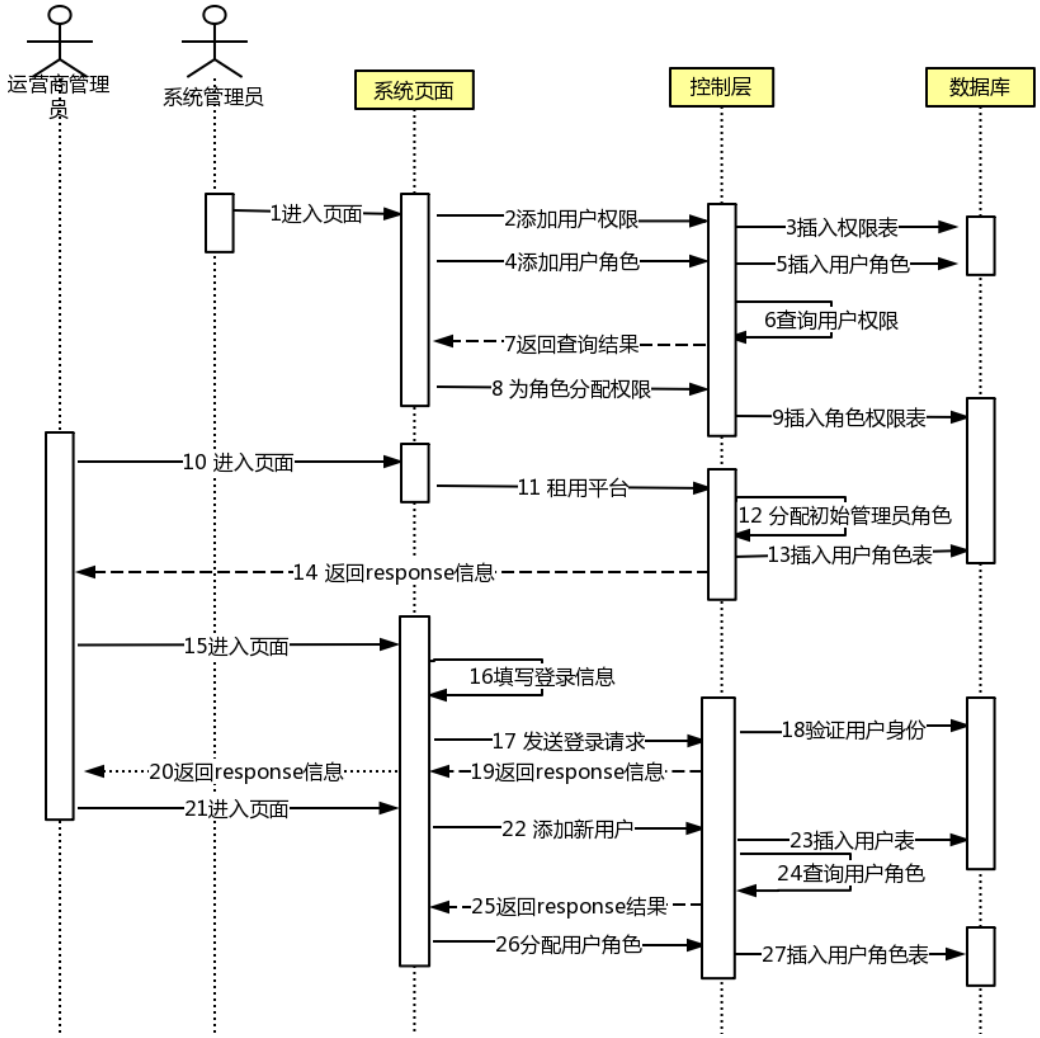

图 5-9 权限管理模块序列图

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的运营商权限，营业员仅拥有运营管理的权限。系统初始化时，对用户角色管理，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 5-9，为权限管理模块序列图。

厂商除进行系统管理外，还具有用户管理、售货机类型管理、售货机管理、运营商管理和订单管理功能。厂商用户输入商家编号、用户名和密码登录系统，JSP 页面通过 EL 表达式和 JSTL 显示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存放在 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。

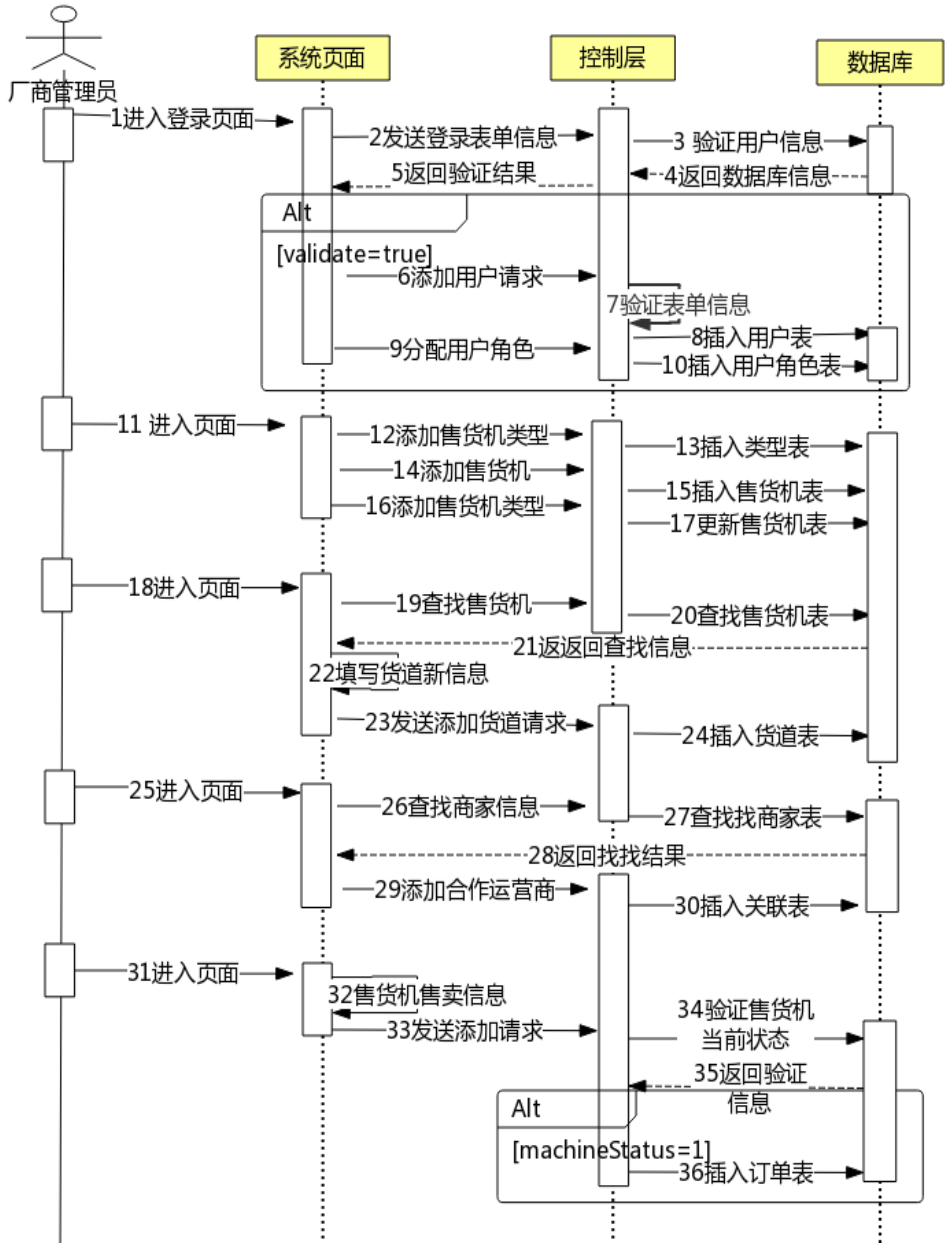

图 5-10 厂商模块时序图

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，最终地址都要与/manu 前缀拼接。

图 5-11 售货机信息列表

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。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)方法添加。售货机分配时调用 getManuMachineStatus()方法查看分配状态。

5.3.2 自动售货机运营商功能

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

1. 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管理，包括用户组管理员、营业员、库存管理员和财务管理员。创建用户时使用 userManagerService.alreadyUser(userInfo)对当前用户进行判重验证，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，将售货机分配给营业员进行管理。运营商超级管理员时序图如图 5-12。

2. 运营商营业员

营业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页面，菜单包括售货机查询、货道管理、库存申请、个人信息查询、更改密码。售货机初始化时调用 Ajax 的 addChannelInfo()添加货道，调用 Ajax 的 addChannelWare()方法将货道与商品

的对应信息插入到货道商品表中。货道和商品信息存储在两张表中，对应信息添加到另一张独立的表中以便管理。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。VMSale APP 新增了移动支付功能，营业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，财务管理员用于财务的统计。

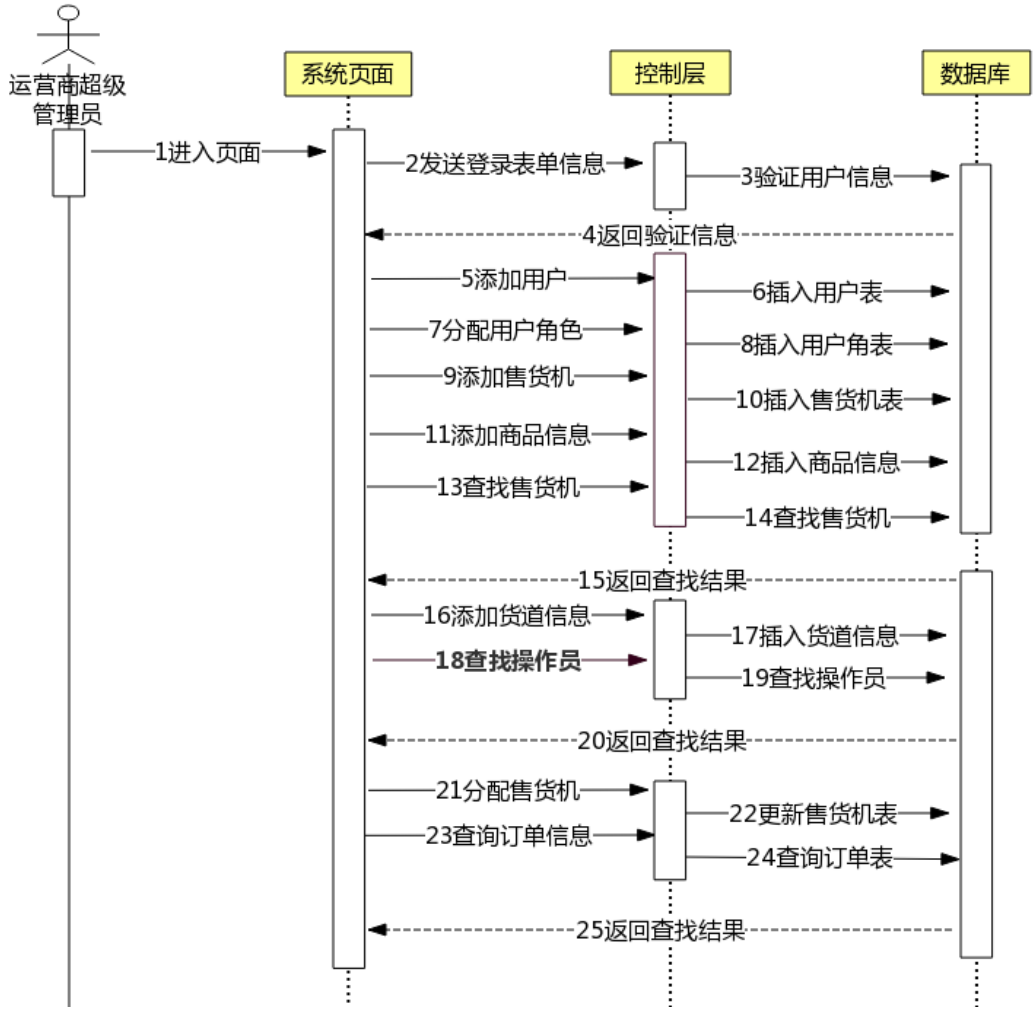

图 5-12 运营商超级管理员模块时序图

营业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。营业员实现页面示例图如图 5-13，5-14 所示。

3. 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，处理营业员的申请库存请求，为营业员新增库存量。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，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 5-15 所示。

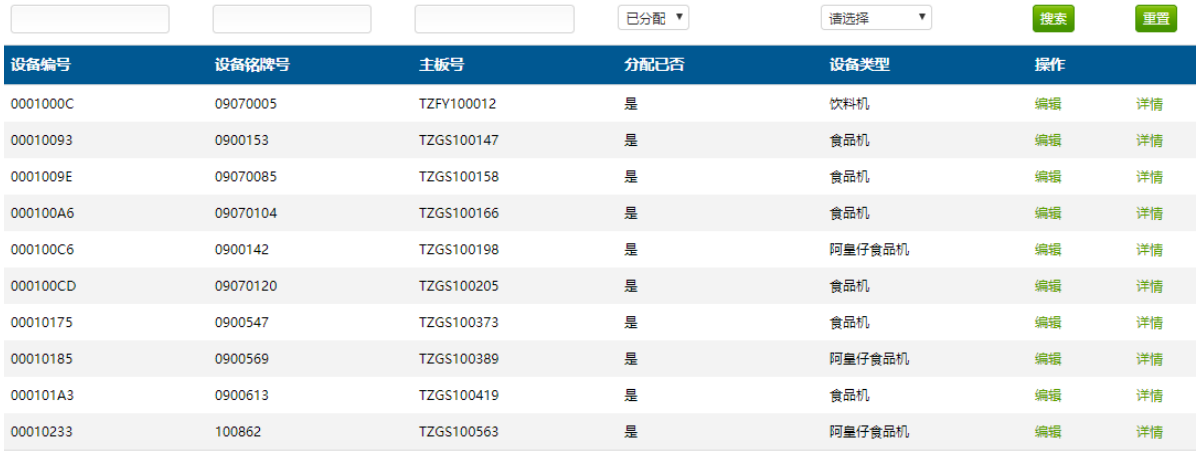

图 5-13 售货机查询页面

图 5-14 库存申请页面

4. 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。根据营业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。售货机终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 5-16。

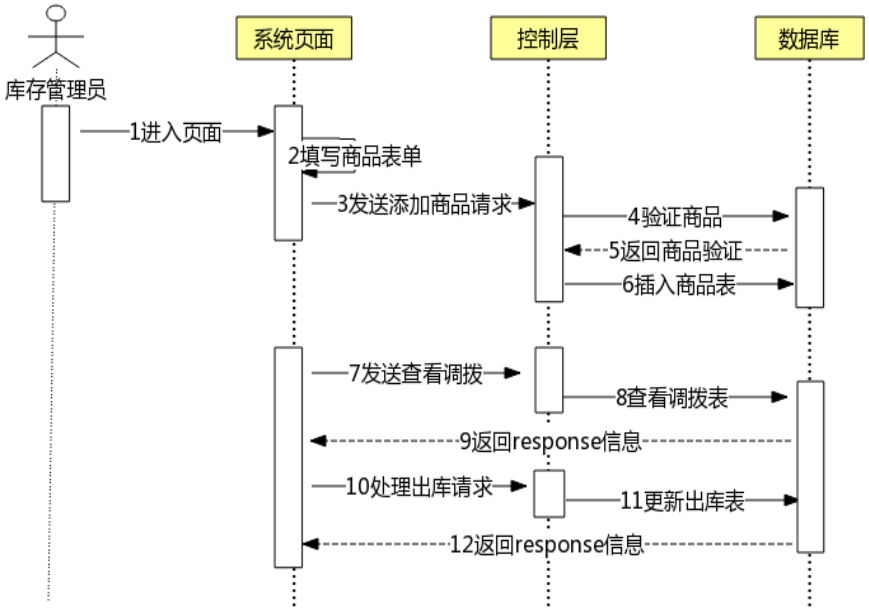

图 5-15 库存管理模块时序图

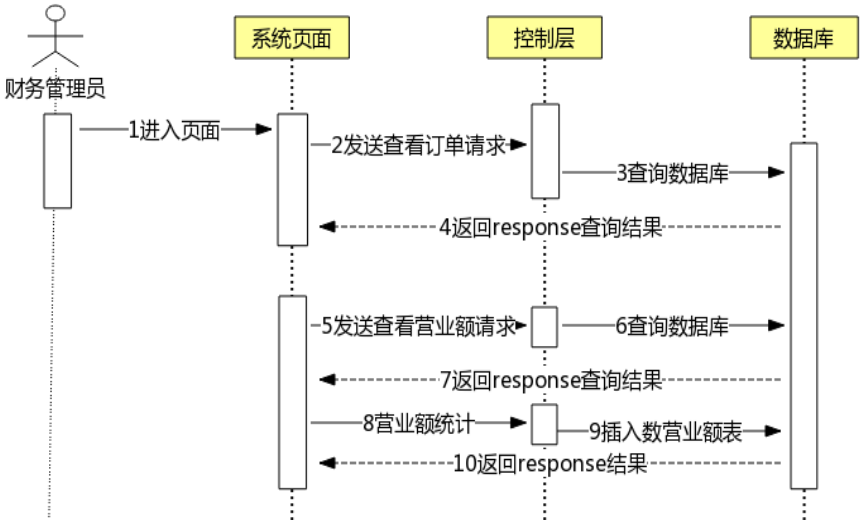

图 5-16 财务管理模块时序图

5.3.3 VMManage APP

VMManage APP 开发的目的是供自动售货机运营商营业员使用，营业员在手机上安装该软件，通过软件实现机器信息查询和售货机信息更新，减少重复性工作、有效减少人工出错、提高工作效率、降低人力成本的消耗。

VMManage 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 数据传输至后台服务器，进行数据的更新。服务器为提供 url 为/client 的控制接口，处理结束后返回一个 URLEncode 编码的 String 字符串或者 String 格式的 JSON 串。Android 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返回。

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。营业员通过用户 Id 和所属的商家 Id，使用 HttpURLConnection 类请求查看售货机及相应的货道信息。营业员将加货量、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 VMManage 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，营业员不具有手动操作的权限。当营业员更新售货机库存时，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。

运营商营业员 VMManage APP 时序图如图 5-17 所示。

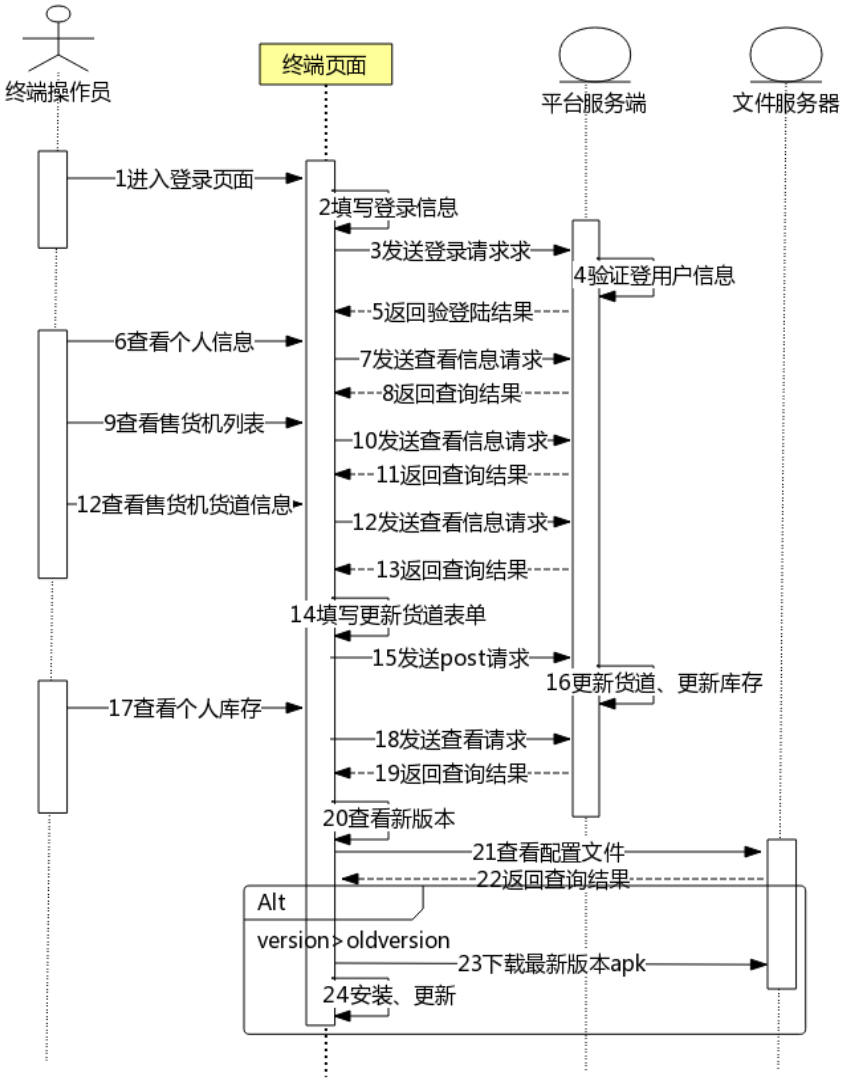

图 5-17 运营商营业员终端时序图

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 APK 的版本名和版本号。系统从文件服务器上下载最新 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。

VMManage APP 使用流程如 5-18 所示。

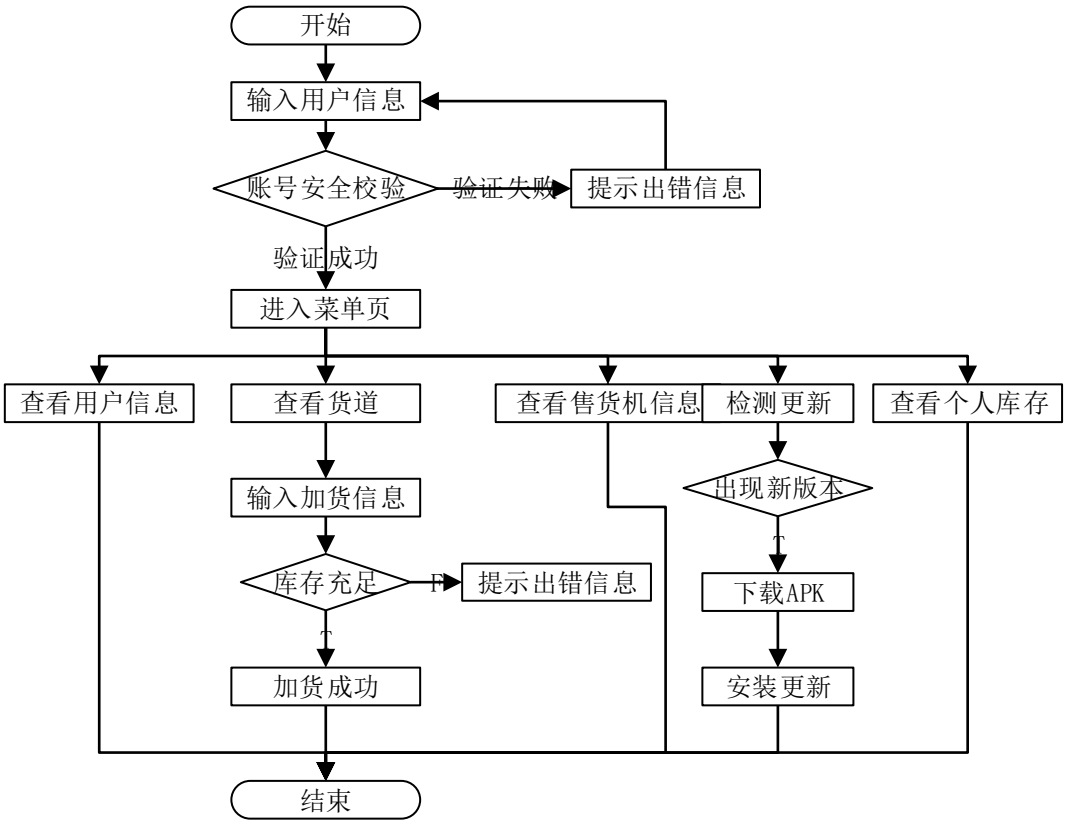

图 5-18 终端营业员使用流程图

5.3.4 VMSale APP

1. 购物流程的实现

VMSale APP 用于完成购物流程，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[52]，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。图 5-19 为一次完整的购物流程。

（1） 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 为一个图片的路径。

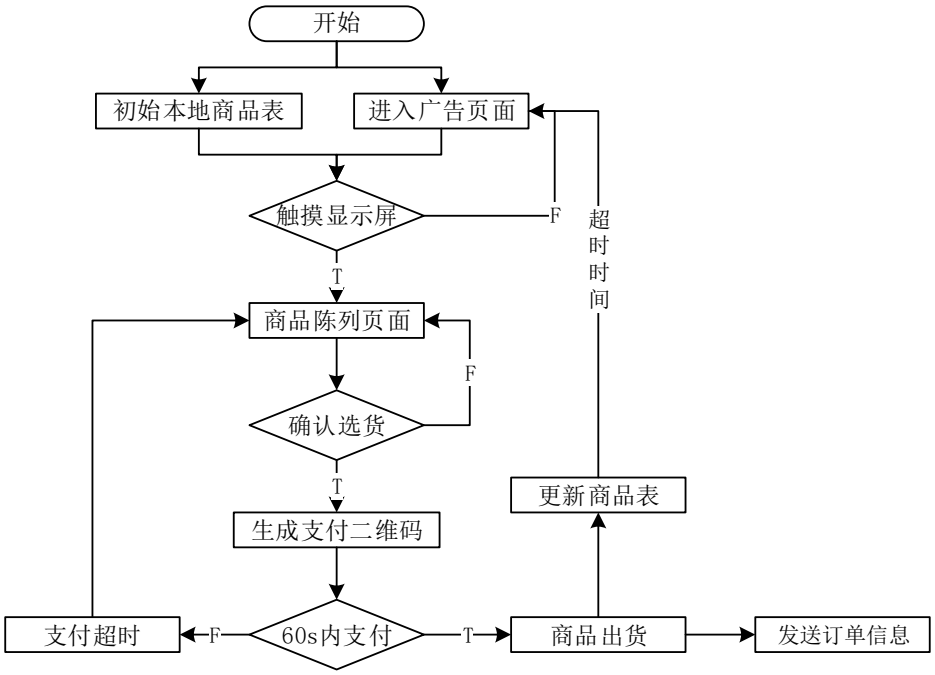

图 5-19 自动售货机 VMSale APP 购物流程图

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(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”。

（2） 多媒体播放

VMSale 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，VMSale 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 5-20。

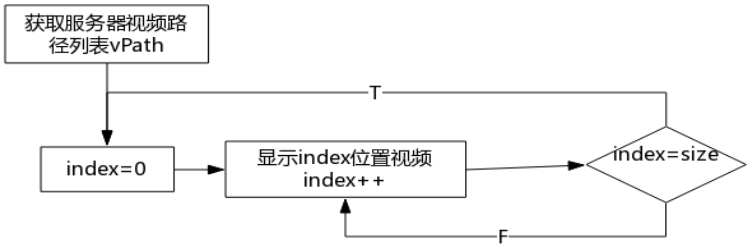

图 5-20 视频播放流程图

2. 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。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 5-21。

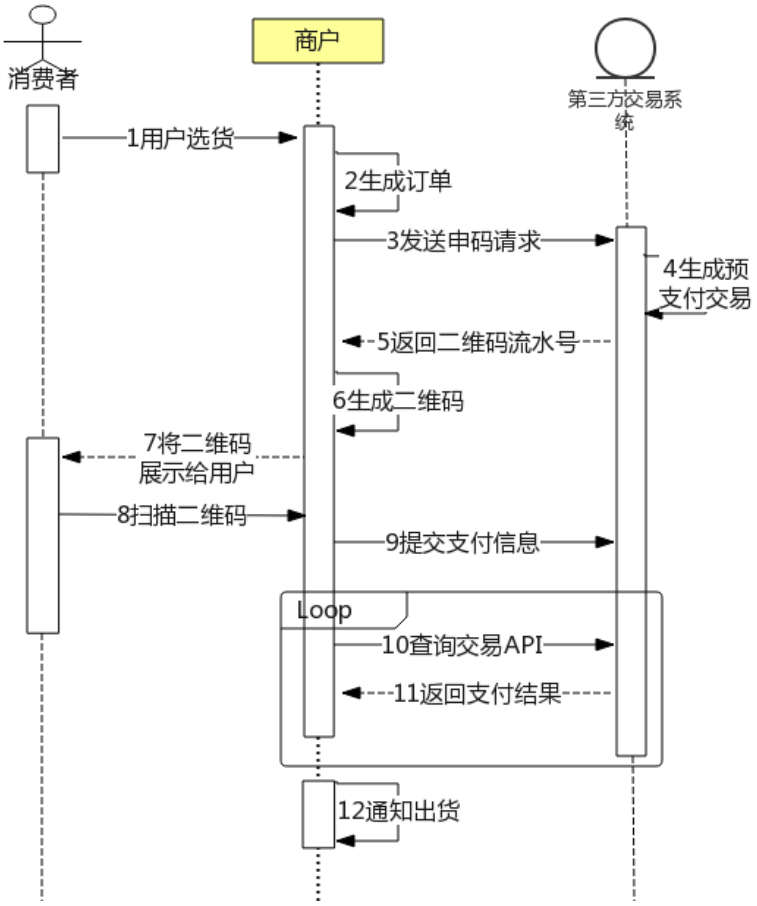

图 5-21 移动支付时序图

（1） 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。然后，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。同步返回一个包含二维码信息的 JSON 串并将其解析成 Map 对象验签。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”并返回。最后，使用二维码生成类 Create2DCode 转 qrCode 信息为二维码，展示在页面中。

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。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。对象签名部分代码如下：

```
data.put(SDKConstants.param_certId, CertUtil.getSignCertId());// 设置签名证书序列号
// 将 Map 信息转换成 key1=value1&key2=value2 的形式
String stringData = coverMap2String(data);
byte[] byteSign = null;
String stringSign = null;
try {
    // 通过 SHA256 进行摘要并转 16 进制
    byte[] signDigest = SecureUtil
            .sha256X16(stringData, encoding);
    byteSign = SecureUtil.base64Encode(SecureUtil.signBySoft256(
            CertUtil.getSignCertPrivateKey(), signDigest));
    stringSign = new String(byteSign);
    data.put(SDKConstants.param_signature, stringSign); // 设置签名域值
    return true;
} catch (Exception e) {
    Log.w("SDKUtil","签名错误", e);
    return false;
}
```

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 字段和 origRespCode 的字段信息。最后，判断 origRespCode 的 value 值判断交易是否成功。

（2） 支付宝二维码支付

支付宝二维码支付和银联支付相类似，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，使用私钥和 RSA 算法进行签名，并使用 Base64 编码，签名结果放置在 sign 字段，然后发送给支付宝服务器。最后，支付宝返回一个 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 JSON 串中的 qrCode 即为需要的二维码流水号。VMSale 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。

一条成功的支付宝申码返回结果如下，其中 qr_code 字段为二维码信息，sign 为签名后的信息。

{"alipay_trade_precreate_response":{"code":"10000","msg":"Success","out_trade_no":"20171119234800","qr_code":"https:\/\/qr.alipay.com\/bax06073uad3eslmdczy20f3"},"sign":"ow8PhXpErf7etC0NqSoQTttxuIPw0RxHRsEtup22ftEuzjQGvPNUrHGhJ7IXykQFvnYHiFNMbGviNeo7q0osfKPaLo9a1pbOGXT1Tv3q1+AMLGPHcjEo9FHisYzhKcsjtObAZ3bmq/PA7DNxXZYaCXSfQIThqo4N2POCADL1m4Q="}

（3） 微信二维码支付

微信支付使用 HTTPS 传输，它采用 XML 格式进行请求的提交和返回。微信支付使用 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、密钥 key 和接口密码 secret 等信息。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。Map 对象中的每一个除 sign 字段的非空值使用 key=value 的形式表示，每一对数据使用&符号连接，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，发送至微信服务器。最后，微信返回一个 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，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。

5.4 VMCloudPlatform 系统的测试与分析

VMCloudPlatform 系统的测试贯穿开发的整个流程，包括软件的单元测试、集成测试、接口测试、软件的功能测试和非功能测试，以保证整个系统的可用性和稳定性。由于篇幅限制，本节只展示部分测试内容。

5.4.1 单元测试和集成测试

单元测试使用了 Junit，便于一边编程，一边测试，可以尽早的纠正开发早期遇到的问题。Spring 提供了基于 Junit4 的 Spring 测试框架，在测试类上使用@ContextConfiguration(locations= "classpath:**.xml")注解指定注入的配置文件，使用@RunWith(SpringJUnit4ClassRunner.class)指明集成 Spring 单元测试的测试运行器。示例代码如下：

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = "classpath:spring-mybatis.xml")
public class AuthorityinfoTest {
    @Autowired
    private IUserManagerDao aDao;
    @Test
    public void getUserById() {
        UserInfo userInfo = aDao.getUserById(1);
        assertNotNull(userInfo);
    }
}
```

使用 Mockito 工具生成模拟对象，模拟单元测试中的驱动模块和桩模块，可以更专注于该模块的测试，减少和其他模块的交互，示例代码如下：

```
IUserManagerDao dao = Mockito.mock(IUserManagerDao.class);
when(dao.getAuthorityInfoById(authId)).thenReturn(user);
```

集成测试是在单元测试的基础上，将已经测试通过的各个模块组合起来测试，检查单元之间的接口是否存在问题。这里采用了自底向上的测试方法，从最小的“原子”模块开始，不断向上集成。

5.4.2 功能测试和非功能测试

系统除了单元测试、集成测试和接口测试之外，还需要必要的功能测试和非功能测试。

1. 功能测试

功能测试不需要知道软件内部的编码和工作流程，只需要输入测试用例，并验证测试结果即可。设计测试用例的过程中，尽可能全面的覆盖所有可能出现的

输入，以保证测试各种可能的场景。

（1） 平台租用功能测试

平台租用功能模块主要涉及到租户名称、租户编号、初始密码、功能选择、租用时间等信息的判定。平台租用功能使用等价类划分的方式测试，如表 5-4 所示。

表 5-4 平台租用功能测试

输入	有效等价类	无效等价类
租户名称	(1)25 位以内的中文、英文、下划线字符	(2)空字符 (3)特殊字符，如￥、%、# (4)超过 25 位字符
租户编号	(5)6 位数字字符	(6)有非数字 (7)非 6 位数字
密码	(8)25 位以内英文、数字、下划线字符	(9)空字符 (10)特殊字符，如￥、%、# (11)超过 25 位字符
功能选择	(12)至少选择一个功能	(13)不选择
售货机台数	(14)整数	(15)非整数
租用时间	(16)选择时间	(17)不选择

根据表 5-4 所列的有效等价类和无效等价类进行随机组合，能有效测试租用界面的表单判定和租用功能的实现，当输入无效等价类时，页面能进行正确合理的错误提示，测试结果较好。

表 5-5 平台租用部分测试结果

编号	测试用例	测试结果	预期结果
1	正确的输入用例	创建租用账号成功，数据库更新，创建成功	创建租用账号成功，数据库存储功能信息和租用起止日期
2	输入已存在的租户名称	页面提示名称已占用，创建失败	页面提示名称已占用，创建失败
3	输入非数字租户编号	页面提示租户编号输入不合法，创建失败	页面提示租户输入不合法，创建失败
4	输入含有特殊字符的密码	页面提示密码是字母、数字、下划线，请重新输入，创建失败	页面提示密码是字母、数字、下划线，请重新输入，创建失败
5	不选择功能框	页面提示请至少选择一个功能，创建失败	页面提示请至少选择一个功能，创建失败

（2） 售货机上货功能测试

售货机上货的功能测试要验证两个方面的内容：售货机内部库存的更新和营业员库存更新。如表 5-6 所示，其中 T 表示输入正确的条件，F 表示输入错误的条件，‘-’表示该条件无论是 True 或者 False 都不影响结果，‘√’表示测试结果正确。

表 5-6 售货机上货功能测试判定表

		编号	名称	1	2	3	4	5	6
条件桩		①	营业员编号	T	F	-	-	-	-
		②	售货机 Id	T	T	F	-	-	-
		③	货道编号	T	T	-	F	-	-
		④	上货数量	T	T	-	-	F	-
		⑤	商品 Id	T	T	-	-	-	F
动作桩	中间结果	⑥	售货机库存更新	T	T	F	F	F	F
		⑦	营业员个人库存更新	T	F	F	F	F	F
	结果	⑧	更新成功	√					

根据判定表给出的条件桩和动作桩，只有在条件 1 的情况下，所有条件都输入且输入正确，才返回正确的测试结果，其他任何输入组合都返回错误的结果。测试结果良好，能够验证错误的测试用例，并正确处理。

（3） 终端购物流程测试

对自动售货机终端购物流程的测试，主要测试以下功能：

- 进入选货页面，查看选货界面是否能够正常滚动、翻页
- 是否正常显示缺货信息
- 点击缺货商品是否正常提示缺货信息
- 确定选货后是否正常生成二维码
- 二维码超时时间内是否还能够正常支付
- 支付成功后能否正常出货。

终端购物 APP 的串口信息收发过程使用 serial port utility 串口调试工具进行测试，用来模拟自动售货机的实际操作。serial port utility 串口调试工具和 Android Pad 相连接，Android Pad 执行操作，并发送串口指令，serial port utility 接收指令，并返回处理结果。

2. 非功能测试

非功能测试主要是压力测试和性能测试，是在系统开发完成之后，对整个系统的整体运行质量进行评估。在压力测试时，编写 SQL 在数据库中生成大量测试数据，通过 JMeter 软件进行 Http 请求的压力测试，使用 Jmeter 模拟多个用户并发负载的过程，如图 5-22 所示。

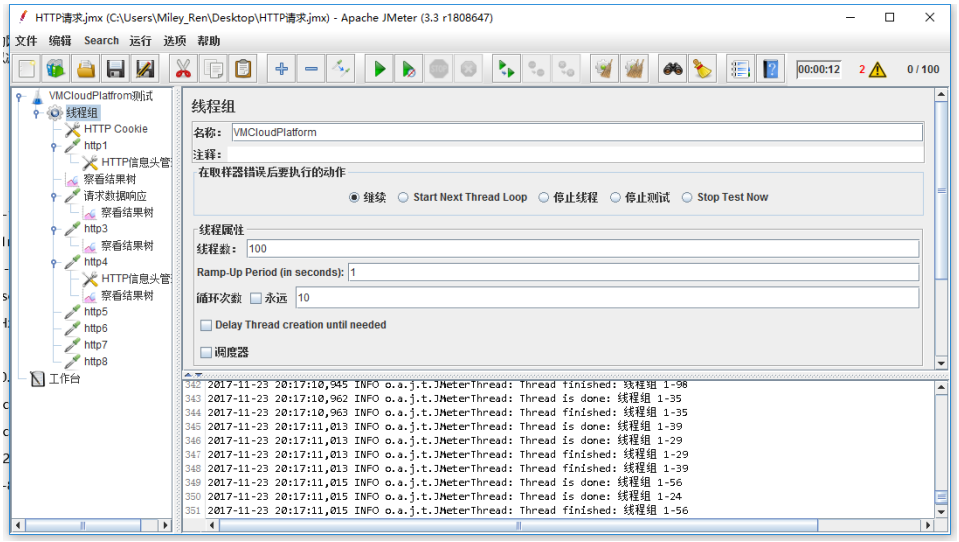

图 5-22 使用 JMeter 测试

经测试，系统的响应时间可以控制在 2 秒之内，请求录入时间可以控制在 2 秒以内，请求查询响应时间在 2 秒之内。

经过单元测试、集成测试、功能测试和非功能测试，测试结果表明，系统具有较好的易用性，能够满足用户的基本要求，无明显延迟，响应时间在合理范围之内。网页布局合理、界面美观、操作简便，具有较好的交互性。

第 6 章 总结与展望

6.1 总结

自动售货机联网、管理信息化和终端智能化是自动售货机行业的发展趋势。然而，我国大部分运营商为中小型企业，根本不具备独立开发或购买一套新型软件的能力。基于这个原因，本文开发了一套可共享的自动售货机云平台系统——VMCloudPlatform 系统，旨在帮助中小型运营商以较低的成本、较快的速度跟上行业发展的潮流。VMCloudPlatform 系统包括后台 VMCloudPlatform 管理系统和相应的售货终端软件 VMSale APP，以及便于营业员使用的 VMManage APP。

VMCloudPlatform 管理系统是整个系统的核心，实现了系统的用户权限、运营商的租用，以及运营商内部自动售货机的管理和监控功能。运营商可以通过该系统管理售货机，远程的实施监控，从而有效地提高了管理效率，降低了人力成本。VMCloudPlatform 管理系统基于 SaaS 服务和多租户技术，并采用了 J2EE 的分层设计思想。VMCloudPlatform 管理系统数据字段的可扩展性和功能的可配置性在供多个租户共享使用的同时，提供了“按需租用，按需付费”的特性。

传统自动售货机通过加装 Android Pad 并部署 VMSale APP 实现智能化，Android Pad 通过 FT312D 芯片和售货机控制盒连接，实现串口通信。VMSale APP 不仅将销售数据实时传送到后台，使运营商能够及时准确地掌握每台售货机的销售状况，还满足了消费者移动支付的需求。

VMManage APP 是一个方便运营商营业员使用的 Android 应用程序，它安装在营业员的移动设备中，可以帮助营业员实时地更新自动售货机库存信息，提高工作效率。

VMCloudPlatform 系统主要有以下几个特点：

- 易用性：系统基于 SaaS 进行开发，售货机厂商对系统部署和发布，售货机运营商直接通过浏览器访问，使用便捷、简单易学。
- 通用性：系统面向自动售货机厂商和运营商，用户类型明确，满足自动售货机厂商和运营商的需求。

- 隔离性：系统采用多租户模式，各个租户之间数据进行隔离，防止信息的泄露。
- 经济性：1）运营商不需要提供硬件、网络、和运维，省去了雇佣技术人员的雇佣成本；3）一次开发多次使用，租金低廉，帮助厂商吸引客户，增强竞争力；4）终端使用价格低廉的 Android 设备改造；5）信息化的管理，提高工作效率、降低人力成本。
- 便捷性：VMSaleAPP 提供了货道更新操作和移动支付，营业员使用手机实时更新售货机库存信息。

6.2 展望

当今自动售货机行业发展非常快，新机型、新应用层出不穷，VMCloudPlatform 系统也必须与时俱进，要根据需求的变化不断地更新、完善和扩展，VMCloudPlatform 系统的设计已对此作了充分地考虑。近年来，信息安全问题越来越受到重视，未来运营商对数据的安全隔离要求也必定会越来越高，进一步探究 SaaS 隔离级别和多租户模式的优化将是十分必要的。随着平台的持续运营，其中所积累的销售记录逐渐增多，这些数据将是一笔宝贵的财富，如何充分利用海量数据进行二次开发，从中挖掘出有价值的信息也将是今后必须研究的课题。

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: U.S. Patent 5,445,295[P]. 1995-8-29.

[2] 魏浩，郭也．中国制造业单位劳动力成本及其国际比较研究[J]．统计研究，2013, 30(8):102-110.

[3] 宁安毅，张文娣，赵伟,等．自动售货机的兴起与发展[J]．黑龙江科技信息，2015(9):63.

[4] 马涛．我国移动支付业务发展分析[J]．华南金融电脑, 2005, 13(3): 9-10.

[5] 倪春洪．新生代消费心理趋势研究[J]．艺术与设计:理论版, 2011(2X):255-256.

[6] 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.

[7] 李向文．欧、美、日韩及我国的物联网发展战略——物联网的全球发展行动[J]．射频世界, 2010(3):49-53.

[8] 王黎楠．我国自动售货行业的困境和建议[J]．卷宗, 2016(7).

[9] 王萌皎．自动售货机的盈利模式[J]．中国市场, 2017(19):97-97.

[10] 周婷婷，王宁诚．自动售货机销售监测反馈系统[J]．电脑知识与技术, 2016 (4X): 275-277.

[11] 朱先妮.友宝在线：小售货机里的大乾坤[N].上海证券报，2016-09-20(007).

[12] 赵明阳，杨晓妮．云计算及其关键技术[J]．软件(教育现代化)（电子版），2013(5):115-116.

[13] 王磊，陈刚，陆忠华．基于云计算的高效科学计算应用软件框架[J]．华中科技大学学报(自然科学版), 2011, 39(s1):166-169.

[14] 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.

[15] 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.

[16] 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.

[17] Bezemer C P, Zaidman A. Multi-tenant SaaS applications: maintenance dream or nightmare?[C]//Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE). ACM, 2010: 88-92.

[18] Cusumano M. Cloud computing and SaaS as new computing platforms[J]. Communications of the ACM, 2010, 53(4): 27-29.

[19] Chong F, Carraro G. Architecture strategies for catching the long tail[J]. MSDN Library, Microsoft Corporation, 2006: 9-10.

[20] 耿冰, 于修理. SaaS 与传统软件的比较研究[J]. 沈阳师范大学学报(自然科学版), 2009, 27(1):84-86.

[21] 陈鹏, 薛恒新. 面向中小企业信息化的 SaaS 应用研究[J]. 机械设计与制造工程, 2008, 37(1):10-13.

[22] Zhang D, Wei Z, Yang Y. Research on lightweight MVC framework based on spring MVC and mybatis[C]//Computational Intelligence and Design (ISCID), 2013 Sixth International Symposium on. IEEE, 2013, 1: 350-353.

[23] 王建国, 王建英. Struts+Spring+Hibernate 框架及应用开发[M]. 清华大学出版社, 2011:293-357.

[24] 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.

[25] Mudunuri S. Mybatis in Practice: A Step by Step Approach for Learning Mybatis Framework[J]. 2013.

[26] Zhang D, Wei Z, Yang Y. Research on lightweight MVC framework based on spring MVC and mybatis[C]//Computational Intelligence and Design (ISCID), 2013 Sixth International Symposium on. IEEE, 2013, 1: 350-353.

[27] 宫志方, 程林, 杨培强. 一种基于 Spring 和 MyBatis 的 MVC 框架:, CN

105843609 A[P]. 2016.

[28] 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.

[29] 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.

[30] 郭霖. 第一行代码[M]. 人民邮电出版社, 2014.

[31] 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.

[32] 纪晓阳. 线程在 Android 开发中的应用[J]. 软件, 2013(8):24-26.

[33] 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.

[34] 夏伟.基于 3G 网络的自动售餐机系统设计与实现[D].中南大学,2013.

[35] 赵春亭, 左小五. 基于 Android 系统的 USB 转串口的研究[J]. 工业控制计算机, 2014, 27(1):83-84.

[36] 李炜键, 孙飞. 基于 4G 通信技术的无线网络安全通信分析[J]. 电力信息与通信技术, 2014, 12(1):127-131.

[37] 徐述书. 基于 GPRS 和 GIS 的自动售货机监控管理系统[D]. 东南大学, 2012.

[38] 景东男, 韩建民, 王爱华. 基于物联网的自动售货机及远程监控系统[J]. 计算机技术与发展, 2013(5):228-230.

[39] 舒新峰. 无线网络自动售货机系统设计[J]. 西安邮电大学学报, 2009, 14(3):92-94.

[40] 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.

[41] 陈萌, 叶桦, 达飞鹏. 自动售货机主控制器及执行机构的设计与实现[J]. 东南大学学报(自然科学版), 2007, 37(s1):24-28.

[42] 马艳丽, 杨奎河. 基于 SSH 框架的自动售货机远程监控管理系统的设计与实现[J]. 数字通信世界, 2017, 8: 135.

[43] 彭荣. SaaS 模式下多租户系统架构及关键技术研究[D]. 大连海事大学, 2010.

[44] 邓伟华. SAAS 应用的数据模型研究与设计[J]. 电脑编程技巧与维护, 2009(8):5-6.

[45] 陈国庆, 郑洋洋. 云计算多租户架构中数据处理系统及处理方法:, CN102930027A[P]. 2013.

[46] 刘腾飞. SECloud 系统多租户数据库的研究与实现[D]. 重庆大学, 2014.

[47] 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.

[48] 王正飞. 数据库加密技术及其应用研究[D]. 复旦大学, 2005.

[49] 汤滢江, 何铁军, 贾通,等. Android 环境下 USB 扩展串口方法研究[J]. 金陵科技学院学报, 2013(4):9-14.

[50] 高海彬. JNI 在 Android 系统下串口控制的应用[J]. 信息技术, 2013(10):173-176.

[51] 周新宇, 王印玺. 浅谈软件静默安装技术在系统维护中的应用[J]. 中国科技信息, 2012(16):104-104.

[52] 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.