首都经济贸易大学
Capital University of Economics and Business

专业硕士学位论文

基于微信小程序的网上购物系统的设计与实现

培养单位：信息学院

专业名称：软件工程

作者姓名：程子珍

指导教师：牛东来 教授

首都经济贸易大学
CAPITAL UNIVERSITY OF ECONOMICS AND BUSINESS

专业硕士学位论文

基于微信小程序的网上购物系统的设计与实现

培养单位：首都经济贸易大学

学科专业：软件工程

作者姓名：程子珍

指导教师：牛东来 教授

二〇一八年六月

The Design and Implementation of Online Shopping System Based on WeChat Mini Program

Candidate：Cheng ZiZhen

Supervisor：Prof.Niu DongLai

Capital University of Economics and Business, Beijing, China

摘要

随着信息技术的发展、用户消费升级，传统商超、电子商务在线上推广和购物体验等方面遇到了瓶颈。无人超市在消费者购物行为上加入了高科技科技体验，但伴随这炫酷的黑科技体验，带来的还有高额的店铺成本投入和维护，若投入社会普遍复制这种模式，技术和成本门槛较高。开发一款对于传统商家通用、对于消费者方便快捷的网上购物系统的应用范围更广。本着“触手可及，即用即走”的微信小程序非常适合为人们生活中的重要又低频的需求服务，相对于原生态的 APP 更加切合线下快速推广的这种需求。论文以传统社区类便利店的购物方式为出发点，结合微信小程序技术，采用面向对象的开发方法，开发一种可以方便商家线下推广、消费者线上购物的方便快捷的微信小程序购物系统。

本系统主要由微信小程序客户端、服务器、数据管理端构成，服务器采用了 ThinkPHP5 框架技术，客户端采用了微信小程序的 MINA 框架，数据管理端采用 CMS 框架。客户端的页面实现通过网络与服务器 REST API 接口通信获取 MySQL 数据。本人重点参与了网上购物系统客户端、服务器以及数据库的设计、开发、测试工作。

在系统的设计与实现过程中，对客户端的代码进行了全局的 MVC 模式设计控制，采用 template 模板增加代码的复用性，并创建客户端用户的 token 管理机制进行用户身份验证与权限分级。本文重点阐述了小程序购物系统客户端的商品信息展示、商品分类、购物车、下单支付，个人信息管理及数据库设计六大模块的设计和实现过程。

关键词：微信小程序；购物系统；O2O

Abstract

With the development of information technology and the upgrading of consumer spending, traditional supermarkets, e-commerce have encountered obstacle in online promotion and shopping experience. Unmanned supermarkets add high-tech technology experiences to consumers' shopping behavior, but along with this cool black technology experience, there is still a high cost of shop investment and maintenance. If the society is universally copied this model, technology And the cost threshold is too high. The development of an online shopping system that is universal for traditional businesses and convenient for consumers has a wider range of applications. The WeChat Mini Program that is based on “get it at your fingertips, and is ready for use” is well-suited to serve the important and low-frequency needs of people's lives. Compared with the original eco-app, WeChat Mini Program is more in line with the needs of quickly promote offline. The paper starts with the traditional shopping methods of community convenience stores, uses WeChat Mini Program technology, and adopts an object-oriented development method to develop a convenient and convenient WeChat Mini Program shopping system that can facilitate offline promotion by merchants and online shopping for consumers.

This system is mainly composed of the WeChat Mini Program client, server and data management. The server adopts ThinkPHP5 framework technology, the client adopts the MINA framework of the WeChat Mini Program, and the data management adopts the CMS framework. The client's page implementation communicates with the server REST API interface to obtain MySQL data. I have participated in the design, development, and testing of online shopping system clients, servers, and databases.

In the process of designing and implementing the system, the client's code is subjected to the global MVC pattern design control, the template is used to increase the reusability of the code, and the client user's token management mechanism is created for user authentication and permission grading. This article focuses on the design and implementation of the six modules of display product information, product classification, shopping cart, order payment, personal information management and database design.

Keyword: WeChat Mini Program;Shopping system; O2O

目录

第 1 章　引言

基于微信小程序客户端的网上购物系统是 O2O、电子商务与微信小程序相结合的共同产出物，它集合了微信小程序开发、ThinkPHP5 服务器开发、数据库、网络等多种当下的流行技术于一体，消费者可以通过扫一扫小程序码快速获取店铺和商品信息，方便了用户可以随时随地的购买的商品，小程序购物客户端的加入为电子商务的发展添加了新的生机。

1.1　项目研究的背景和意义

随着近年来我国网络基础设施的不断完善、信息技术的不断发展和第三方支付工具的广泛普及，网络购物已经成为了人们生活中的一项重要内容。基于 PC 平台的购物系统方便了人们足不出户就能享受到购物的乐趣，而基于手机端的在线购物则发挥了智能终端的便携性，能使大众随时随地购买自己喜欢的商品。面向社区的购物模式来源于 O2O 概念, 它是将线下的商务机会与互联网结合，让互联网成为线下交易的平台。此购物系统就是以互联网为基础，社区为基本单位，充分利用社区的信息化基础设施，将社区的小型实体店与网络商城结合起来运营的一种新型电子商务模式，是传统电子商务的一次新的突破。其线上更加注重的是满足消费者便利的需求和会员的互动营销以及为实体店导流，线下则更多的是更好、更便捷的为用户提供现代化的智能购物体验服务。

与此同时在信息技术、消费升级、竞争态势等多因素驱动下，中国零售业正迎来新的转变时机，即“线上+线下+物流”深度融合的“新零售”。其致力于推动线上与线下的一体化进程，促成价格消费时代向价值消费时代的全面转型，技术核心是实现以消费者为中心的会员、支付、库存、服务等方面数据的全面打通，推动消费购物体验的升级，推进消费购物方式的变革。

新零售发展的核心是满足特定购物场景的顾客需求。在顾客已经信息化、技术化、移动互联网化的趋势下，快购、快送、快结的购物方式更被用户所青睐。但对于传统零售来说，为线下店铺提供一种方便快捷的线上购物渠道是增强顾客体验、提高零售效率的重要途径。为此，近年来国内外零售商都已经普遍提供线上与线下相结合的社区类购物服务。纵观当下主流的线上购物系统主要有 PC 端购物系统，APP 购物系统，依赖于第三方应用的购物入口等。2017 年初微信小程序的正式上线进一步活跃了微信的生态圈，微信小程序与原生 APP 相比最大的特点就是用户无需下载安装，通过扫码或搜索就可以获得小程序，由此可见基于微信和即用即走的轻量级小程序平台可以更好的连接线下的服务。

随着“新零售”模式和线上线下结合购物的广泛应用，消费者购物对方便快捷的购物方式新需求成为亟待解决的问题，然而开发、维护适用于多移动操作系统的多个的购物应用系统对于当下亟待转型的传统便利店来讲，这无疑将会增加他们的店铺运营成本，除此之外，繁杂的购物 APP 还会给消费者的手机带来存储压力，不利于获取和留住用户。综合来看，设计出一套可以相对低成本的开发维护的跨平台的购物应用系统是传统便利店在“新零售”下完成转型亟需解决的问题，同时对于消费者来说更看重的是购物系统提供给他们的方便快捷、轻负载即用即走的新型购物体验服务。腾讯公司微信小程序的上线使得设计开发一款可跨平台的为用户提供线上线下购物服务的轻应用成为可能。这就是本论文的研究背景和特色意义。

1.2　国内研究现状分析

近几年来，随着互联网的高速发展，移动互联网以及手机支付的普及给人们的生活带来了翻天覆地的变化。电子商务伴随着互联网的快速发展也在更新改变，从电脑 PC 端到手机移动端的智能化应用普及，使得人们的生活更加依赖于手机，基于淘宝、京东等的大型 APP 购物系统在人们生活中的使用频率越来越高，人们在这些大型的购物 APP 上可以买到全国各地乃至全球各地的商品，方便了人们生活，加速了商品的流通效率。但随着网络购物模式和范围的不断扩大，传统的电子商务的瓶颈也日益显现，比如物流时间的相对滞后和“最后一公里”的配送问题，用户收到实物与网上产品介绍相差很大的问题，售后维权顾客要自己付高额邮费等问题。这些问题降低了用户的购物体验。

新零售的出现将网上购物的范围缩小，着眼于线上线下购物相结合的方式，为用户打造更可靠、高品质的购物服务，这为线下的大型商超的发展带来了新的活力与机遇。伴随着移动互联网的发展，购物方式变得更加移动化、便捷化，注重线上购物与线下购物的结合的社交性，借助于社交平台的购物模式应运而生。微信作为国内最大的社交平台，目前微信的月活跃用户量达十亿，微信已变成人们生活和交往的重要组成部分。微信于 2012 年 8 月推出的微信公众平台，发展至今不仅形成了获取用户流量的新媒体渠道，也促使了基于微信公众号的生活服务平台的成长，其中“美丽说”、“京东商城团购”等通过微信平台运营大大增加了用户量[赵敬，2013]。这些案例的成功表明了微信借助其社交性质可以很好的联通其他生活服务。2017 年 1 月 9 日微信正式发布了微信小程序，张小龙将小程序定位为一款即用即走的轻量级应用。小程序作为一款区别于 PC 端和 APP 端的新物种，借助于微信的海量用户，更适合于连接线下用户。对于中小型的传统线下零售行业，在新零售电子商务的第二次冲击下，小程序定位为其参与到这场变革提供了重要的技术驱动力［张翔，2017]。

1.3 论文的主要研究工作

本文所研究的购物系统主要是一个面向社区居民的线上购物系统，结合社区小型实体商店的新零售化，实现其覆盖周边的线上线下相结合的销售模式，扩充其销售渠道。本系统由后台服务器、微信小程序购物客户端、数据管理系统三部分组成。由于系统的系统模块较多，本文的主要工作为设计及实现微信小程序客户端和后台服务的交互及数据库的设计。

结合微信小程序技术的购物系统主旨在于联通社区商店的线上线下相结合的购物方式。用户可以到店购买商品，亲自查看商品的品相和质量，满意后购买离。若用户在家不便到店购物，用户也可以获取商店的微信小程序，通过小程序购物系统客户端在线上购物，完成浏览商品、添加商品到购物车、下单结算的整个购物流程，同时用户还可以在小程序购物系统的个人信息管理页进行账号信息、收货地址的修改，查看订单信息等功能。

根据以上的需求，整个微信小程序购物系统项目采用 B2C 的架构，开发方法采用服务端、客户端、数据管理三端分离的架构模式。其中服务端采用 ThinkPHP5 和 MySQL 构建 REST API；客户端采用微信小程序来实现，通过向服务器 API 发送 http 通信请求获取数据，来完成自身的功能行为逻辑；数据管理系统作为管理或者运营人员的数据管理后台，同样通过向服务端请求数据，实现商品信息、订单数据和状态的管理。

其中论文的主要工作是在微信小程序购物客户端系统实现获取用户信息，商品信息（包括广告商品、专题商品、商品上新等首页商品）展示，商品分类检索，购物车功能，下单支付功能，个人信息管理功能（电话、地址管理等）、订单管理功能等。本文重点描述了商品信息展示模块、商品分类检索模块、购物车模块、下单支付模块、个人信息管理模块及数据库设计六大模块的设计与实现。

1.4 论文的特色

主要有以下几点：

（1）网上购物的设计相对高效低成本的缓解了线下零售行业中的社区商店缺少线上购物的痛点，基于微信的线上店铺更容易被线下用户接受，扩充了线下中小型社区商店的销售渠道。

（2）基于微信小程序通过扫一扫就可以获取到店铺的一款轻应用实现了原生 APP 才能实现的功能，并且微信小程序的系统运行不受手机移动端版本和类型的影响，更有利于获取和留住用户。

（3）本系统设计实现采用面向对象的开发方法，小程序客户端采用了全局的 MVC 模式设计控制，页面实现采用 template 模板增加模块的复用性，客户端通过向服务器 REST API 发送 http 请求获取数据。

（4）在微信支付的设计与实现中，客户端构建 Token 管理机制，实现小程序客户端进行用户身份验证与权限分级，在客户端与开发者服务器和微信服务器三方敏感接口通信过程中进行严格的 Token 检测机制，保障了系统和用户信息的安全。

1.5　论文结构

第 1 章：引言。从本文主要研究内容出发，阐述了项目的背景及意义，结合分析国内外的现状以及目前零售业发展面临的机遇和挑战，给出项目概述以及本文的主要工作和本文组织结构。

第 2 章：系统架构和关键技术。主要介绍了项目总的系统架构和开发环境，并对开发过程中所需的关键技术做了简要介绍。

第 3 章：微信小程序网上购物系统的需求分析。进行了系统的业务及数据流分析，将系统分成多个不同的功能模块进行 UML 分析，并描述每一模块需要实现的功能。

第 4 章：微信小程序网上购物系统的详细设计与功能实现。介绍了项目的总体架构以及开发环境，数据库的设计与实现，购物系统开发过程中客户端与服务器的交互原理、客户端的 MVC 模式设计以及各个模块设计与实现的详细介绍。

第 5 章：购物系统软件测试与分析。结合 postman 测试工具和微信开发者工具对系统各个模块的进行测试，使系统的功能和性能达到运行要求。

第 6 章：总结与展望。结合本论文已完成的各项工作进行总结，并提出需要注意的问题。

1.6 本章小结

本章为引言部分。1.1 节中主要阐述了微信小程序购物系统的项目背景及其意义；1.2 小节研究国内相关系统的研究现状，体现了微信小程序购物系统面向中小型商店对比传统线上购物系统的优势；1.3 小节中明确了本文在项目中的主要研究工作；1.4 小节总结了论文的主要特色；1.5 小节列出了论文的撰写组织结构，进一步明确了本文的组织框架和系统模块实现设计计划。

第 2 章　系统相关技术研究

本章主要安介绍了本微信小程序购物系统在其开发过程中所用到的关键技术，其中包括微信小程序的 MINA 框架、ThinkPHP5 框架、REST API 的构造、微信支付技术、MySQL 数据库五部分。本课题的设计与实现主要就是建立在这些理论知识的掌握和应用之上。

2.1　微信小程序

微信小程序是一种全新的连接用户与服务的方式。小程序为用户提供了触手可及、用完即走的轻便服务，主要体现在用户获取小程序、使用小程序方便快捷的服务理念上。用户可以在线下扫一扫或者公众号关联小程序码获取小程序应用，或者可以在微信的发现页面搜索小程序，由于小程序的整个文件占用内存空间很小，用户下载安装小程序的过程消耗几乎感知不到，所以从用户获取小程序到点击进入开始使用小程序应用的速度非常快。小程序多入口获取和触手可及用完即走的使用体验降低了器传播和获取用户的成本，并且可以实现原生 APP 的基本功能，更加适合线下的生活服务类的商铺及非刚需类低频应用。

MINA 框架是微信团队为小程序开发提供的框架名字，MINA 框架通过封装微信客户端提供的文件系统、网络通信、任务管理、数据安全等基础功能，由此对上层提供一整套 JavaScript API，让开发者能够非常方便地使用微信客户端提供的各种基础功能与能力，快速构建一个应用。简单来讲，MINA 其实就是一组便于开发微信小程序的工具的集合。

框架提供了自己的视图层特有的描述语言 WXML 、WXSS，以及基于 JavaScript 的逻辑层，并在视图层与逻辑层间提供了数据传输和事件机制。这个数据传输和事件机制是一个响应用户交互的数据绑定系统，它让数据与视图非常简单地保持同步。当需要更新页面数据时，只需要调用逻辑层的相应方法返回结果到视图层即可。

通过下面的 MINA 框架图我们可以看到三大部分：逻辑层、视图层和系统层。如图 2. 1。

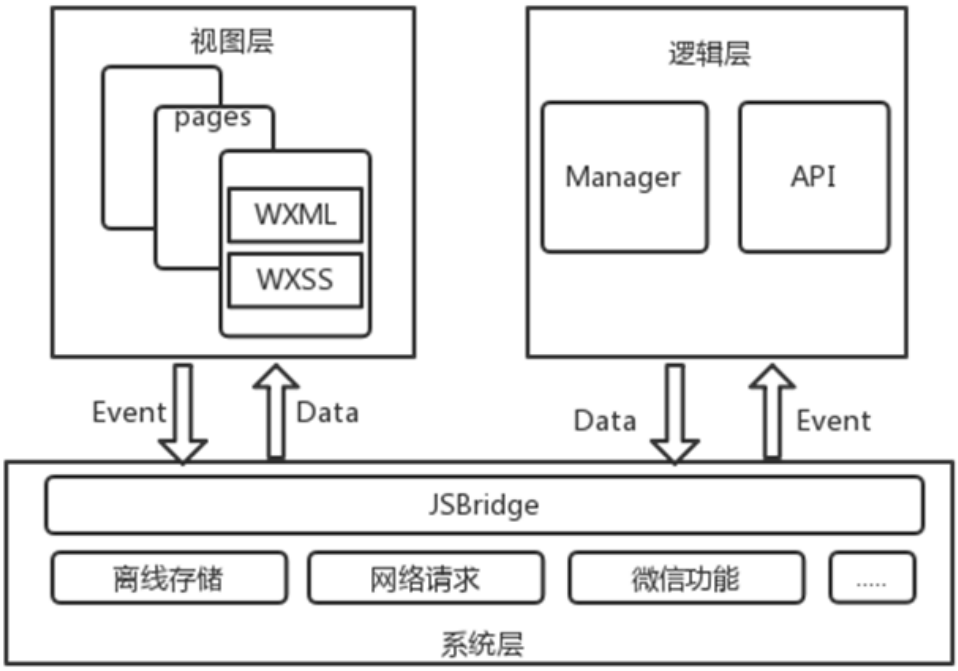

图 2.1　微信 MINA 框架架构图

（1）　逻辑层

逻辑层是 MINA 的服务中心，由微信客户端启用异步线程单独加载运行。页面渲染所需的数据、页面交互处理逻辑都在逻辑层中实现。小程序中的各个页面可以通过逻辑层的函数实现数据管理、网络通信、应用生命周期管理和页面路由。其中，小程序中的每个页面都有与其业务逻辑相对应的生命周期，这个由页面逻辑层的 page()函数实现，生命周期控制中主要包括以下方法：onLoad 方法监听页面加载，onShow 方法监听页面的显示，onReady 方法监听页面初次渲染完成，onHide 方法监听页面的隐藏，onUnload 方法监听页面卸载，不同的业务要在不同的生命周期方法函数内实现。

微信小程序开发框架的逻辑层是由 JavaScript 实现。在 JavaScript 的基础上，微信团队针对小程序项目做了一些适当的修改，以便提高开发小程序的效率。主要修改包括：增加 app 和 page 方法，进行程序和页面的注册；提供丰富的 API, 如扫一扫、支付等微信特有的能力；每个页面有独立的作用域，并提供模块化能力等。

逻辑层的实现就由各个页面的.js 脚本文件负责，微信小程序的逻辑层响应视图层的事件处理，并将处理结果返回到视图层。但由于小程序并非运行在浏览器中，所以 JavaScript 在 Web 中的一些能力无法使用，如 document、window 等。

（2）　视图层

视图层提供了一套类似 HTML 标签的语言以及一系列基础组件。开发者使用 WXML 文件来搭建页面的基础视图结构，使用 WXSS 文件来控制页面的展现样式。视图层就是页面.wxml 文件与.wxss 文件的集合，由组件来进行设计展示。微信小程序在逻

辑层将数据进行处理后发送给视图层展现出来，同时监听视图层的事件。数据在视图层借助组件的设计展现，组件是视图的基本组成单元。

（3） 系统层

系统层主要包括临时数据或缓存、文件存储、网络存储与调用。例如页面临时数据缓存需要在 page()中使用 setData 函数将数据从逻辑层发送到视图层。文件存储和网络存储与调用则需要调用相应的微信 API 接口来实现，如 wx.request 接口可以进行 http 网络请求，通过定义请求的 url、请求参数、请求方法以 json 配置，将请求结果分为 success 和 fail 两种方式处理。除此之外，项目根目录下的三个 app.wxss、app.js、app.json 文件是全局性的系统文件，负责小程序项目的全局公共样式、小程序公共逻辑和小程序公共配置的实现。

2.2 ThinkPHP 5 框架

ThinkPHP 是一个开源的、面向对象的轻量级 PHP 开发框架，适用于敏捷 WEB 应用开发和简化企业应用开发，并且遵循 Apache2 开源协议发布。ThinkPHP 一直秉承简洁实用的设计原则，具有出色的性能和至简的代码，同时也注重易用性。开源模式使得其拥有众多原创功能和特性，兼备国外许多优秀的框架和模式，例如：使用面向对象的开发结构和 MVC 模式，融合了 Struts 的思想和 TagLib（标签库）、RoR 的 ORM 映射和 ActiveRecord 模式。在开源社区团队的积极贡献下，它在开发的易用性、拓展性和效率方面不断优化和改进。

2017 年 2 月发布的 ThinkPHP5.0 版本是一个颠覆和重构版本，采用全新的架构思想，引入了更多的 PHP 新特性，对核心进行了优化，去除了不必要的依赖，惰性加载性能更加明显，支持 composer，并针对 API 开发进行了大量的优化，包括路由、日志、异常、模型、数据库、模板引擎和验证等模块都已经重构。

ThinkPHP5 主要特性包括：

1、遵循 PSR-2、PSR-4 规范。

2、支持 Composer。

3、支持单元测试。

4、安全机制，详细的日志能帮你轻轻松松的做到问题定位。

5、减少核心依赖，增加了扩展的灵活性，支持命令行指令扩展。

6、具备优秀的性能和 REST 支持，支持远程调试，优化了 API 开发工作。

7、惰性加载机制。

8、路由、配置和自动加载的缓存机制。

Think PHP 在项目中的配置，一个项目下可以容纳多个子系统，例如前端系统和后端管理系统等等。在项目中包括公共函数文件夹 Common、系统配置文件夹 Conf（其

中包含该项目的所有配置信息，包括数据库和系统常量以及模板选择和缓存、调试、日志等等)、源代码储存文件夹 Lib（分数据库表达与处理的 Model 文件夹和程序源代码的 Action 文件夹)、支持多种语言的 Lang 文件夹、文件缓存的 Runtime 文件夹以及网页模板的 Tpl 文件夹。此外 index.php 是 Think PHP 的入口文件夹。

2.3 RESTFul API

REST（Representational State Transfer，简称 REST）描述了一个架构样式的网络系统。它在 Roy Fielding 的博士论文中被首次提出，Roy Fielding 的博士是 HTTP 规范的主要编写者之一。REST 作为一种 Web 服务交互方案，注重于用简单轻量的方法设计和实现。简单来讲，REST 没有明确的标准，像是一种设计的风格。

REST 指的是一组架构约束条件和原则。满足这些约束条件和原则的应用程序或设计就是 RESTful。RESTful 的特点是：它是一种软件架构风格、设计风格，而不是标准，向外提供了一组设计原则和约束条件。在服务于客户端和服务器交互类的软件时。这种风格可以更简洁，更有层次，更易于实现缓存等机制。在服务器中符合 RESTful 设计标准的 API，即 RESTful API。

REST 将互联网上的一切实体定义为资源，每一个资源至少对应一个 URL，每个 URL 代表一类操作，使得 Web 资源、服务具有可寻址性。客户端与服务器间的交互通过标准的 HTTP 方法(GET、POST、PUT、DELETE 等) 实现网络资源的处理。设计 REST 风格的 API 需要遵循以下原则:

（1） 唯一专有名词表示原则

在 REST 风格的 URL 设计中，资源是一个实体，统一使用名词表示，且该名词应保证在本 API 系统中的唯一性。

（2）分层独立性原则

每一个符合 REST 风格的 API 都是分层独立的，分层是指每个 API 都是内部封装的，其交互范围限制在单个层。独立性包含两个方面，首先是功能的独立性，即一个 API 提供的功能尽量不依赖其他 API 的执行结果；其次是部署的独立性，API 应尽量部署到专有域名和服务器上，保证能够长期使用。

（3）安全性原则

当调用者在执行对数据库的操作指令时，系统检测该数据库是否提供该操作，对于风险性操作执行操作限制、备份保护、日志记录、风险提示等。

（4）简单化原则

简单化原则是指构造的 API URL 应尽量简短，去除冗余信息。

（5）无状态原则

指服务器端不保存客户端请求的状态(如 session、cookie 等) ，即客户端每次发起的请求是即时独立的。

6) 版本兼容原则

API 的更新应尽量保持与原版本的兼容，并以不同的版本标识，版本号应放入 URL，易于标识和调用。

7) 执行结果一致性原则

此原则针对 API 返回结果的信息一致性进行设计，即服务器执行 API 后的返回数据中请求参数信息和请求结果。

8) 缓存原则

REST 使用缓存设计原则，对执行频率较高的 API 进行缓存以提升加载速度。

2.4 微信支付技术

微信支付主要分为主要分 2 大类，普通模式和服务商模式。其中最常规的普通模式，适用于有自己开发团队或外包开发商的直连商户收款。开发者申请自己的 appid 和 mch_id，两者需具备绑定关系，以此来使用微信支付提供的开放接口，对用户提供服务。本文主要介绍的是最常规的普通模式。

申请微信支付的必须是已经被认证的非个人小程序号，并且在微信支付模块中已申请微信支付功能。申请微信支付成功的微信平台会收到用于微信支付的商户号，商户平台用的密码等重要信息。除此之外，小程序的 APPId、APPSecret 也是用于开发微信支付流程中的重要参数。小程序的前端是使用微信提供的框架开发，后端使用的是开发者自己部署的服务器。

微信支付有多种支付方式，包括刷卡支付，公众号支付，扫码支付，APP 支付等，其中小程序的微信支付是在微信里调起的，属于公众号支付的范畴。在微信小程序支付的代码开发过程中中主要调用了一个微信支付功能的 SDK，主要包括 WxPay.Api.php、WxPay.Config.php、WxPay.Data.php、WxPay.Exception.php、WxPay.Notify.php 五个文件，其中 WxPay.Api.php 是微信支付 SDK 的接口，WxPay.Config.php 是配置文件，WxPay.Data.php 中定义了一组在微信支付中需要用的基本参数对象，其余两个文件分别是错误异常类和接受回调通知的处理类。完成微信小程序的支付需要将微信支付的 SDK 嵌入到自己的支付业务代码中去。微信官方的 API 文档中给出的微信小程序支付的业务流程时序图，如图 2.2。

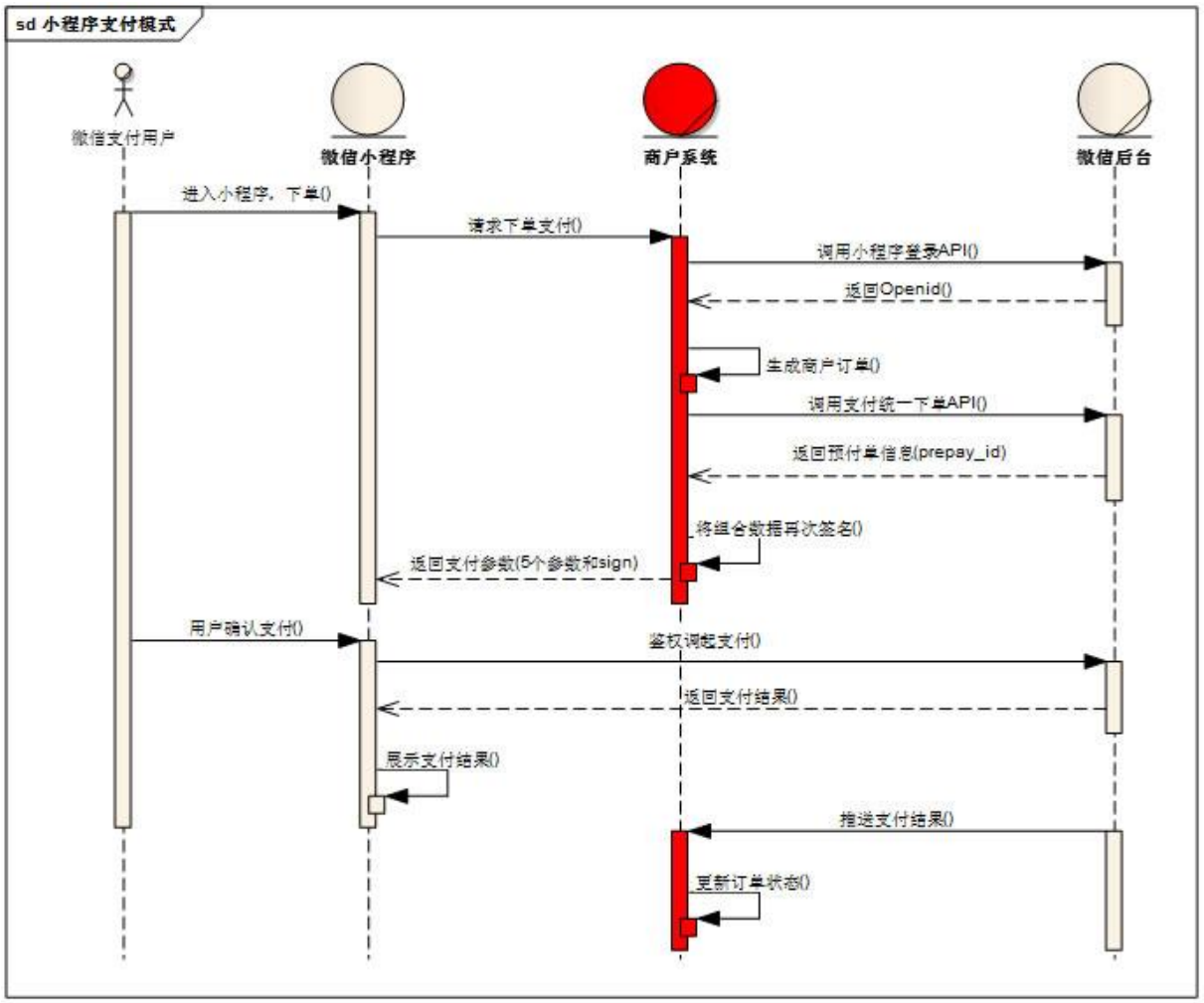

图 2.2　微信小程序支付业务流程时序图

从上图可以看到微信小程序的支付流程比较复杂，主要包括以下几步：当用户下单完成发起请求支付的信息后，小程序客户端会首先调用自己的服务器，将用户的订单信息发送到服务器，比如商品 id、数量等信息；当自己的服务器接收到用户的购买商品的订单信息后，会调用微信服务器的统一下单 API，生成一个预付订单，并将预付订单信息返回自己的服务器，在此过程中，统一下单 API 接口需要用户的 openId，如果自己的服务器中保存了用户的 openId 就不需要再获取用户的 openId，否则需要在第一步中携带 code 换取用户的 openId；在自己的服务器中需要对从微信服务器中返回的预付订单信息进行签名，并将预付订单参数信息和签名一起返回小程序客户端；小程序客户端在收到预付订单信息和签名后，再调用 wx.requestPayment，将预付订单信息和签名一起提交到微信服务器；微信服务器会先验证这些预付订单信息和签名，如果验证通过，则会在小程序拉起支付界面（在微信开发者工具中会跳出一个支付二维码，开发者可以扫描这个二维码在微信中拉起支付）；支付完成后，微信会返回给客户端支付结果，同时将支付的结果推送到服务器中，更新订单的状态。在微信返回微信支付结果到客户端的过

程中，需要开发者用自己的外网服务器与微信服务器通信，如果是本地的，则无法推送通知。值得注意的是，在完成微信小程序支付的过程中，开发者服务器生成了一个预支付订单信息，并携带了签名，这个机制增强了支付过程的安全性。

2.5 MySQL 数据库

MySQL 是一个小型关系型数据库管理系统，开发者是瑞典 MySQL AB 公司，该公司在 2008 年被 Sun 公司收购，2009 年 Sun 公司又被 Oracle 公司收购。目前 Internet 上的中小型网站的数据库大多是 MySQL。MySQL 具有体积小、速度快、总体拥有成本低、开放源代码、性能快捷、优化 SQL 语言、容易使用、多线程和可靠性、多用户支持、可移植性和开放源代码、遵守国际标准和国际化支持、为多种编程语言提供 API 等特点。因为其体积小，速度快，成本低，并且 MySQL 属于开源产品，这为许多中小型网站的开发和运维工作降低了成本。

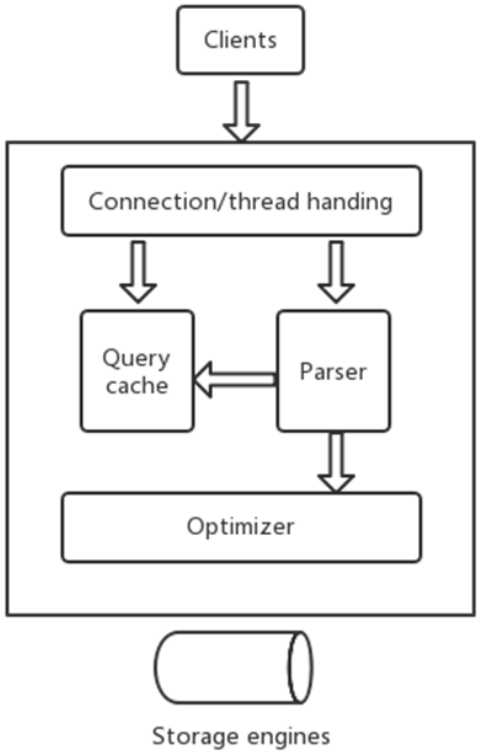

图 2.3 MySQL 系统架构图

从上面的逻辑图中可以看出，如图 2.3，MySQL 内部大致分为三层：

（1）最上层是大部分基于网络的 C/S 服务都有的部分，比如连接处理、授权认证、安全等；

（2）中间层主要是 MySQL 的核心服务功能，包括查询解析、分析、优化、缓存以及内置函数（例如，日期、时间、数学和加密函数），跨存储引擎的功能都在这一层实现，如存储过程、触发器、视图等。

（3）底层是存储引擎，主要负责数据的存储和提取，是数据库中非常重要非常核心的部分，也是 MySQL 区别与其他数据库的一个重要特性。

MySQL 采用的是客户/服务器体系结构，因此实际使用时，有两个程序：

1. MySQL 服务器程序，运行在数据库服务器上，负责在网络上监听并处理来自客户端的服务请求，然后把请求结果返回给用户；

2. MySQL 客户端程序，负责连接数据库服务器，并将信息返回给服务器。

2.6　本章小结

在本章中，主要研究了本文工作中主要用到的关键技术，包括微信小程序框架，服务器端开发中的 RESTFul 风格的 API 接口，在小程序中的微信支付关键技术的流程介绍，最后阐述了选择 MySQL 作为项目的数据库的优势以及其系统架构介绍。这些都是本文在项目中所做工作的关键理论基础。

第 3 章　系统分析

软件工程活动是“生产一个最终满足需求且达到工程目标的软件产品所需要的步骤”。软件工程包括了需求、设计、编码实现、测试等内容，需求分析作为软件工程中重要的一步，主要作用是对用户想要解决的问题和想法进行详细的分析，分层次的分析问题，从输入、问题处理到结果的输出，准确的表述系统必须完成那些工作，对系统的的分析形成完整具体的要求。本章正是通过系统需求分析和 UML 用例图模型两方面研究了微信小程序网上购物系统的设计。

3.1　系统业务流程分析

本文在项目系统中的主要工作是完成小程序购物系统客户端、服务器和数据库的开发和设计。在微信小程序购物系统客户端部分，消费者用户可以进入到微信小程序购物系统客户端，进行商品浏览、商品分类检索、添加购物车，管理购物车和个人信息，下单等主要网上购物操作。在数据库部分，通过合理的设计，使各个数据表之间拥有可靠的关联关系，并具备一定的数据库可扩展性。

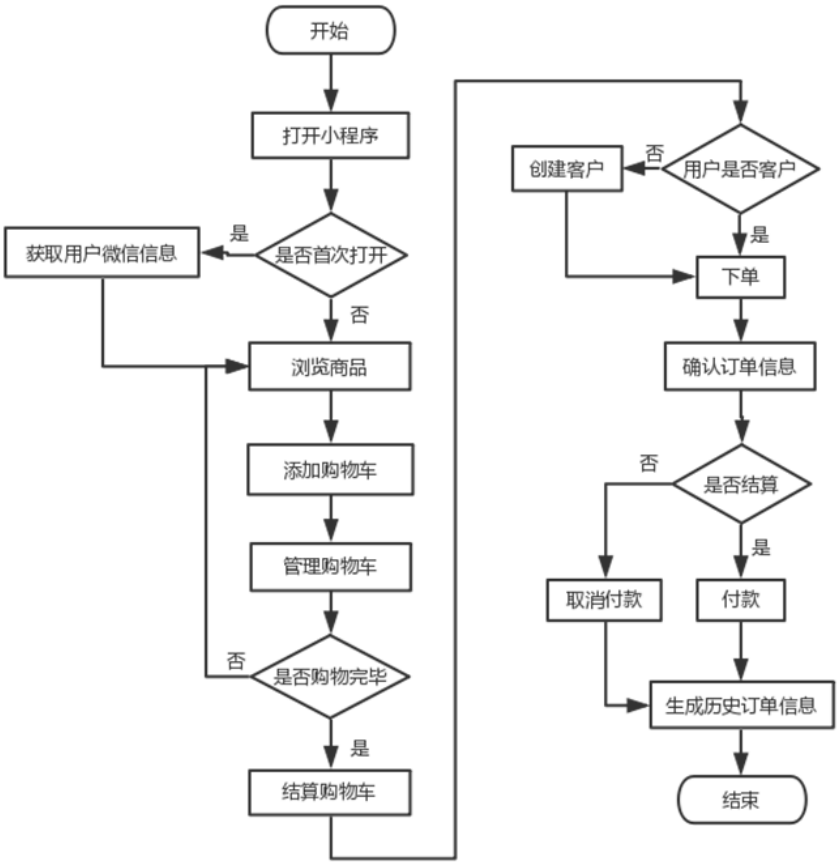

图 3. 1 小程序客户端系统的业务流程图

微信小程序客户端系统的购物流程图 3.1 所示，当用户获取到小程序后，可以选择获取是否允许获取个人信息，也可以浏览小程序的各个页面或者商品。当用户在浏览和分类检索商品的过程中，遇到符合自己心意的商品时，则可以选择把商品加到购物车，当用户结束挑选商品后，可以进入到购物车页面进行结算商品，结算商品时，用户若为未登录状态，需要返回第一步允许微信获取个人信息，并补充具体的个人信息（如地址、电话等），此时，则可以进行下单，进入到支付界面进行订单结算，根据不同的支付结果生成不同的历史订单信息。至此购物流程结束。

3.2　系统数据流分析

本微信小程序网上购物系统项目的数据主要包括客户、产品、订单数据。用户在小程序客户端界面发生的交互动作是业务的数据来源，根据用户行为和购物系统的架构设计分析，业务系统的数据流处理主要包括客户端与本地缓存、客户端与系统服务器、客户端与微信服务器三部分。

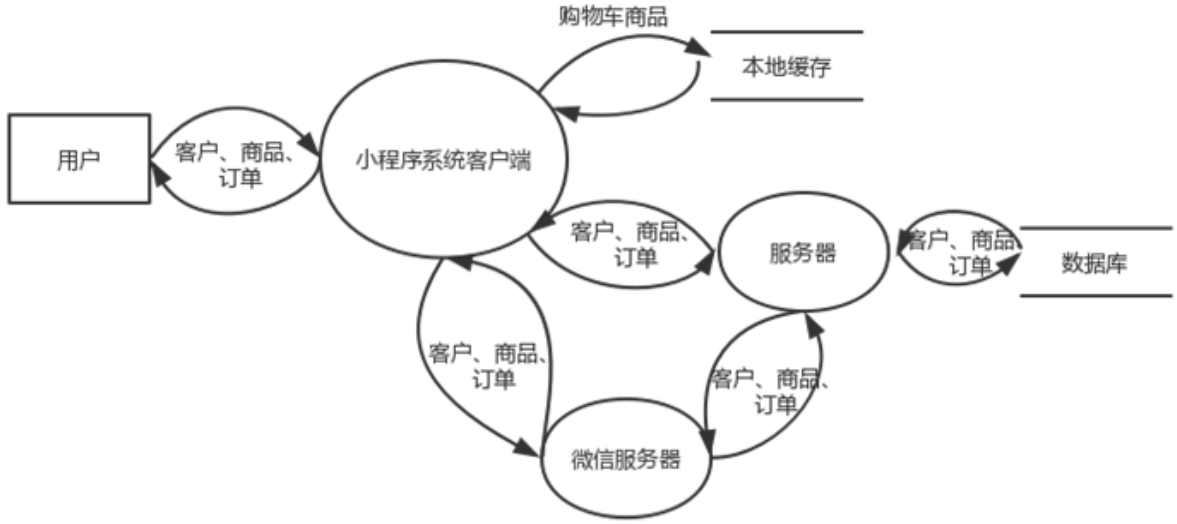

图 3.2 系统业务数据流图

如上图 3.2 统业务数据流图所示，用户可以通过微信小程序网上购物系统客户端携带个人信息请求商品、订单等信息。小程序客户端接收用户的行为请求，在控制层进行相应的业务逻辑处理，并将处理结果以页面展示的方式返回给用户，如果用户请求的数据涉及到数据库的查询和更新，客户端会将用户需求的请求参数通过模型层与服务器通信，在服务器部分进行数据的校验和处理，如果用户的请求合法则会在数据库中进行相应的响应操作，服务器处理完毕后会将结果返回给客户端。在系统设计中，由于购物车的数据量较小，与购物车相关的数据存放在小程序本地缓存中，可以为用户提供更快速的数据反馈效果。用户行为中与微信支付有关的数据请求还会涉及到微信服务器，需要向微信服务器请求微信预订单等参数信息，并返回结果。

3.3 系统功能总体功能分析

根据分析用户网购购物的基本需求，对比分析手机购物 APP 和 PC 购物网站上的主要功能模块，本着小程序开发设计的三大原则：在功能方面，小程序应比原生 APP 更单一；设计方面，小程序硬币原生 APP 更简洁；在使用场景方面，小程序应比原生 APP 更明确。本小程序购物系统的主要定位是为线下传统的面向社区的小型商店易于传播和使用的线上购物渠道，基于小程序的主要定位，小程序应结合线下实体店为用户提供更加主题明确的购物服务，尽量减少与购物无关的不必要的功能模块，为用户提供用完即走、方便快捷的使用体验。根据以上分析，本小程序购物系统的业务逻辑模块分为用户登录、商品信息展示、分类检索商品、购物车、下单支付、个人信息管理功能模块，总体的系统功能模块如下图所示，本章将对各个功能模块进行详细的需求分析。

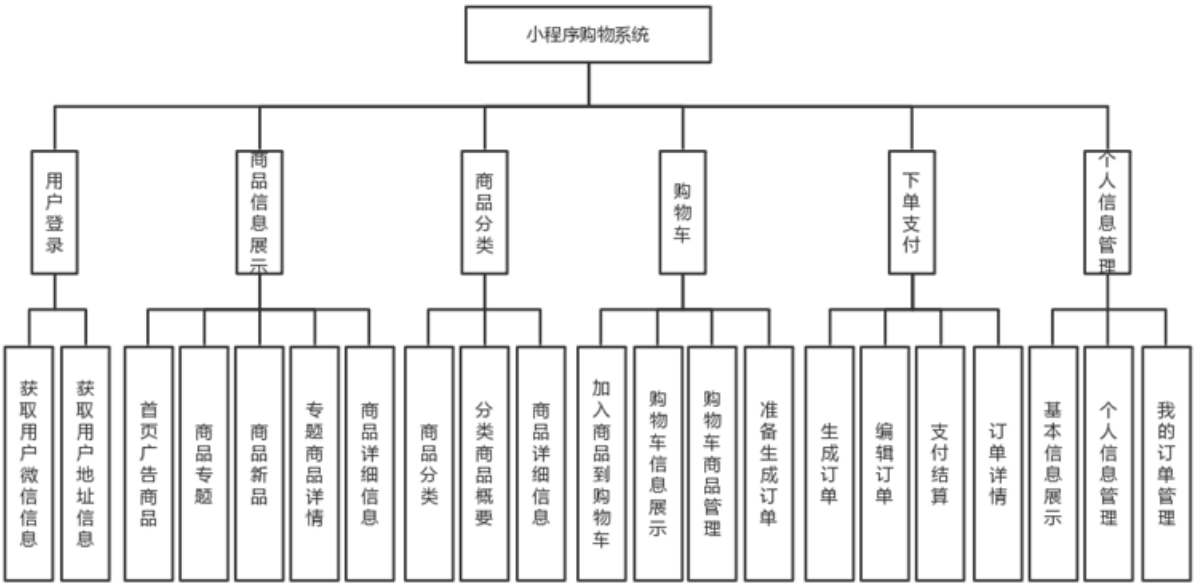

图 3.3 小程序客户端整体功能结构图

从图 3.3 功能结构图中，我们可以大致总结出小程序客户端购物系统的几个主要功能，包括用户查看商品功能（包括浏览广告商品、浏览专题商品、查看商品详情等），用户分类检索商品，购物车功能，查看个人信息功能（包括管理个人信息，修改地址，查看订单等），以及下单微信支付功能五大模块，本文也主要是围绕着这些功能模块的设计和实现而展开的。

3.4 系统功能详细需求分析

根据以上小节的对微信小程序购物系统客户端的业务流程、数据流以及功能模块的分析，根据用户登录小程序时的微信授权状态可以将用户分为游客和客户端软件用户，

其中接受微信授权的用户为客户端软件用户，否则为无登录状态的游客，不能使用客户端的部分功能模块。由此可以得出小程序客户端用户的系统功能用例图，如图 3.4。

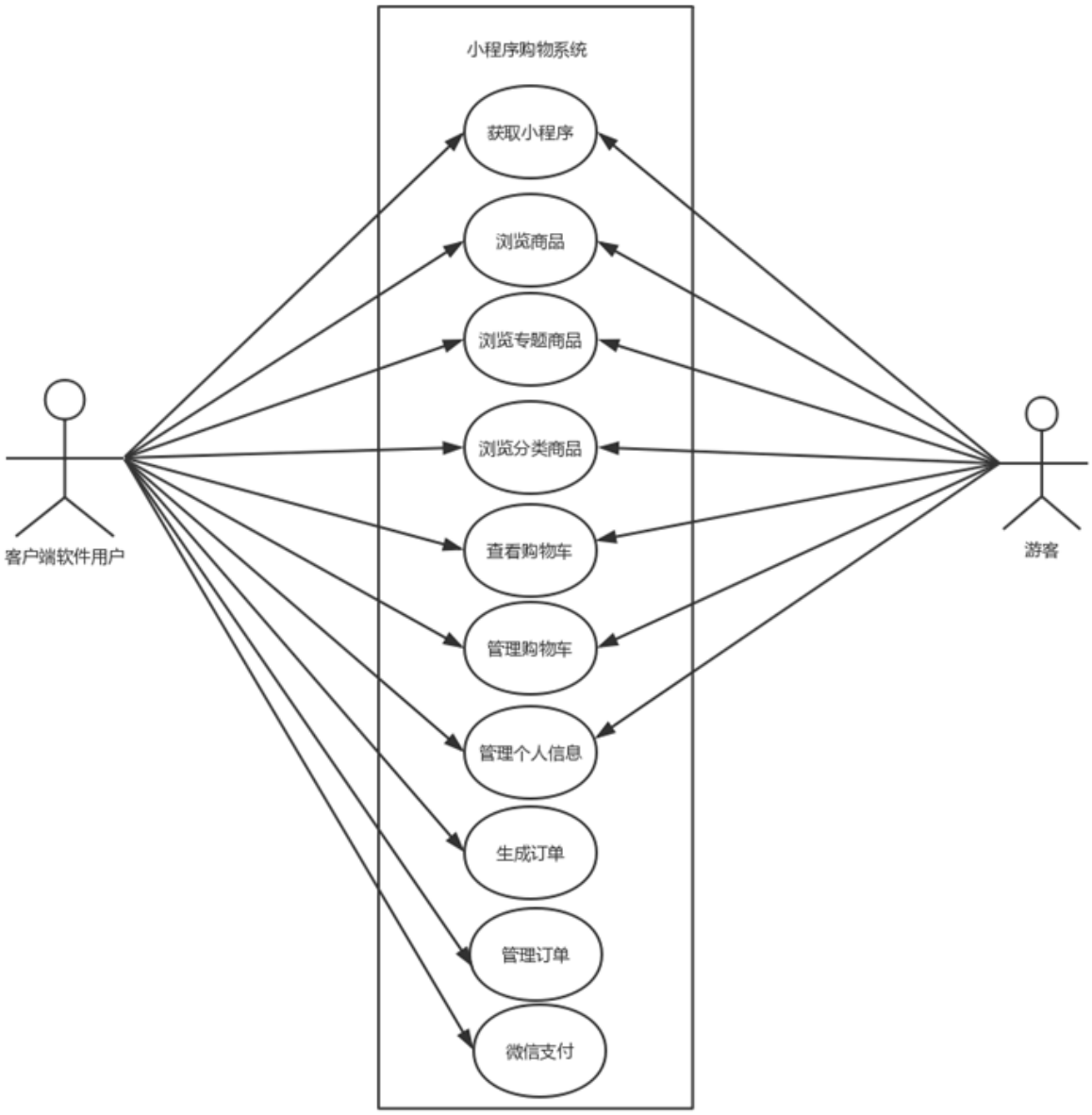

图 3.4　小程序客户端用户的系统功能用例图

3.4.1　商品信息展示

当用户获取到微信小程序时，进入到微信小程序购物系统的客户端，无论用户是否允许被获取微信头像及昵称信息，都可以进入到微信小程序的各个页面进行浏览，主要是当用户浏览小程序的“首页”页面时，就会触发用户浏览商品的功能。主要为用户浏览广告商品、浏览专题商品、浏览上新商品。

小程序购物系统的用户浏览商品用例图如图 3.5。

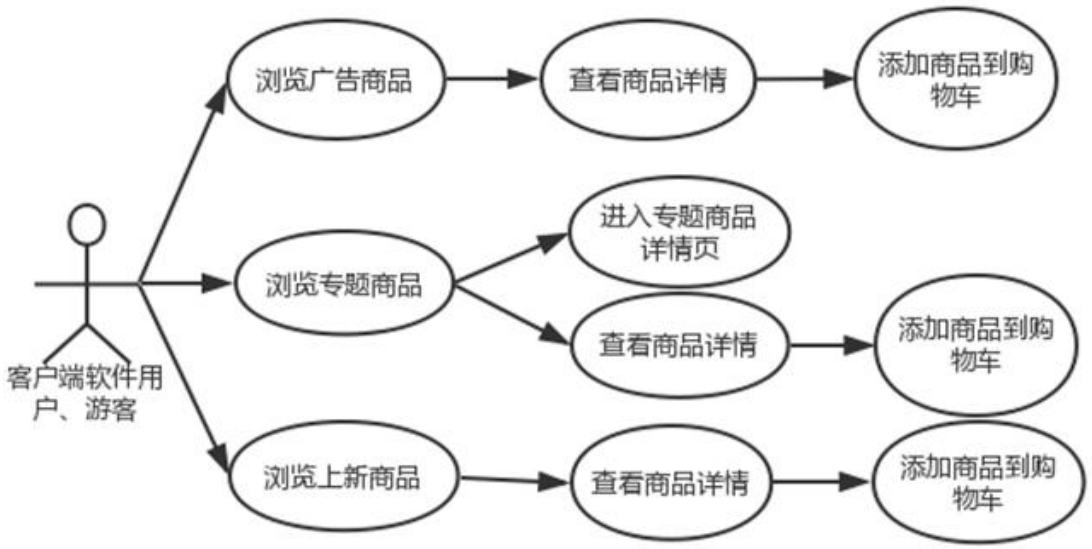

图 3.5　用户浏览商品用例图

用户浏览商品的功能需求分析如下：

（1）　无论用户是否已经被获取微信头像或者昵称信息或完善个人信息，都具有商品浏览的权限。

（2）　当用户进入到微信小程序的“首页”页面时，可以浏览页面头部的广告商品、三个主题商品栏以及最近上新商品。其中主题商品栏可以调转到专题商品详情页面，显示的是同一种主题的一组商品矩阵展示。

（3）　广告商品可以在页面的首部实现自动滚动播放的 banner 广告形式。

（4）　用户在页面可以通过手动下拉动作获取到最新的商品信息。

（5）　商品详情的展示内容包括商品图片、商品名称、商品数量、商品价格、商品库存状态、产品参数、商品详细图片展示等内容，并且用户可以在商品详情页选择商品的数量，将商品加入到购物车。

3.4.2　商品分类

用户在小程序购物系统客户端浏览商品的同时也可以进入到“分类”页面，选择某一类目下的商品进行浏览，这样分类检索商品的功能设计可以让用户更快的找到所需要的商品，同时在分类浏览商品的过程中，也可以从分类商品的角度更加全面的认识本购物系统所售卖商品的范围和特点。主要为用户查看商品分类信息、查看商品详细信息等。

小程序购物系统的商品分类用例图如图 3.6。

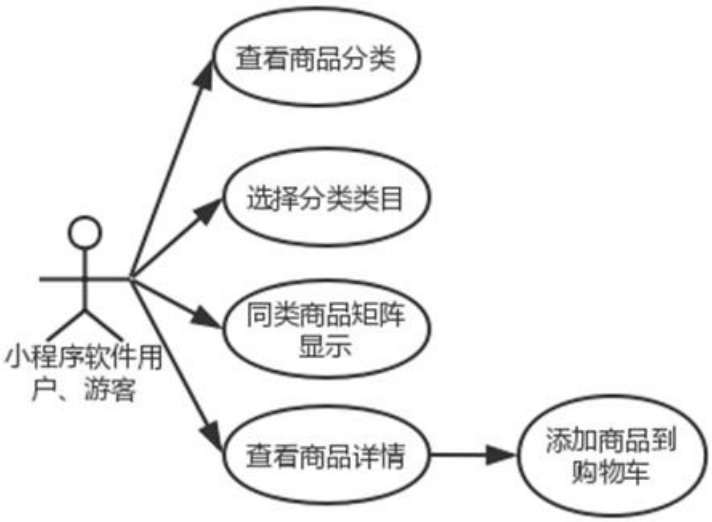

图 3.6　小程序商品分类用例图

分类检索商品的功能需求具体分析如下：

（1）　分类商品主要在“分类”页面进行展示，当用户在导航栏处点击“分类”时即可进入到商品分类页面。

（2）　分类商品根据店铺售卖商品的特点自定义分类，分类栏在页面左半部分，可点击分类词进行分类页面跳转。

（3）　分类商品展示时，用户可以在右侧分类商品矩阵中，点击商品图片，跳转带商品详情页，查看商品详情。

（4）　商品详情页展示商品的详细信息以及用户可以选择商品添加到购物车。用户可以在系统默认设置的[1,10]范围之间选择商品数量加入到购物车，对于库存量为零的商品则会显示无货提示。

3.4.3　购物车

用户在浏览商品的过程中，可以把符合心意的商品加入到购物车中，同时方便用户在下单付款前对所选购的多个商品进行核对检查。用户可以通过导航栏选择“购物车”进入到购物车页面，也可以在商品详情页点击购物车小图标进入到购物车页面。主要为用户查看购物车、编辑购物车商品、提交订单等。

小程序购物系统客户端的购物车用例图如图 3.7。

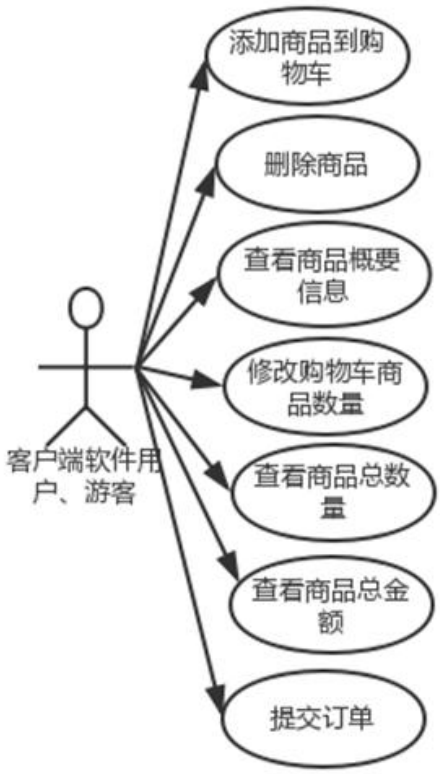

图 3.7　购物车用例图

对于购物车功能的需求分析具体如下：

（1）　用户可以在小程序的底部导航栏处，点击“购物车”进入到购物车页面，也可以从商品详情页点击右上角的购物车标志图片跳转到购物车页面。

（2）　用户只有在已被获取了微信头像昵称等信息下，才可以将选择的商品加入到购物车或者进入购物车检查核对商品。

（3）　当用户选择商品加入到购物车时，用户可以在购物车查看到此商品。其中不同的商品会分条目显示，若某一商品的数量多个，购物车的页面可以准确的显示器具体的数字。

（4）　用户购物车页面的展示设计内容包括商品简要信息（其中包含商品图片，商品名称，商品单价，商品数量，删除按钮，商品单选框），全选按钮，下单结算按钮，选择结算商品的总金额。

（5）　用户可以在购物车页面中对商品的数量进行手动增加或者减少修改，增加的数量不能超过商品的数据库中的库存量，减少的数量不能低于 1。若用户想要删除某一商品，可直接在商品条目处点击删除小标识删除此商品。

（6）　用户提交订单时，可以手动选择商品条目前的复选符号，也可以之=直接点击全选按钮选择全部商品，进行下单结算。

（7）　用户可以直接点击购物车页面中的商品条目，跳转到此商品的商品详情页，方便用户在下单结算前再次快捷的查看商品的详细信息。

（8）　当用户已经提交了订单后，所提交订单中的商品将不会再购物车中显示。

3.4.4　下单支付

用户可以在购物车页面对商品发起下订单支付的需求，也可以在个人信心的订单管理页面发起订单支付的需求，通过微信支付对订单进行付款。

小程序购物系统客户端的用户下单支付用例图如图 3.8。

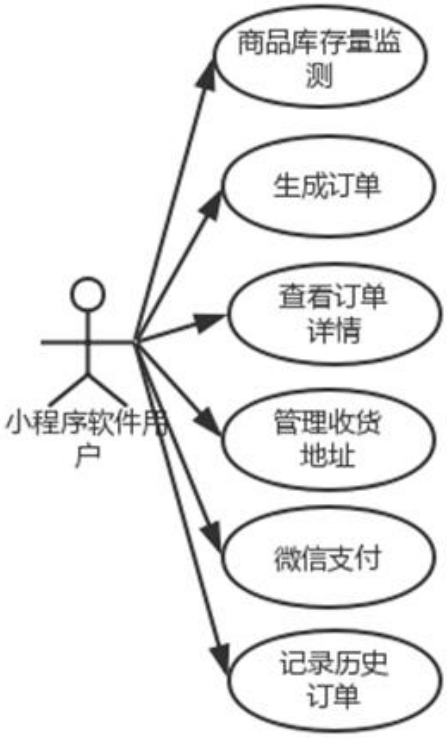

图 3.8　用户下单支付用例图

对于购物系统的下单支付功能的需求分析如下：

（1）　用户可以在购物车页面点击下单按钮进入到订单确认页面。

（2）　用户只有在已经完善了个人信息（收货地址，电话等信息）时，才可以进行订单的付款操作，也可以选择在此编辑修改个人信息；若未完善个人信息处于游客状态，则会在点击付款的时候跳出完善个人信息的提示框。

（3）　当用户的信息是完善状态时，点击下单，若数据库中相应商品的库存量检测通过，则可以顺利生产订单。订单一旦产生，数据库中的订单表则会多出一条记录，同时相应商品的库存量也会减少。

（4）　对生成的订单点击付款，使用微信支付即可，此时也会生成历史订单信息。

3.4.5 个人信息管理

微信小程序购物系统应有一个用户可以查看和管理个人信息的页面，并且应是购物系统主要页面之一，方便用户可以快捷的查看和管理个人信息。主要分为查看和管理个人信息和收货地址信息、查看和管理订单信息。

小程序购物系统客户端的个人信息管理用例图如图 3.9。

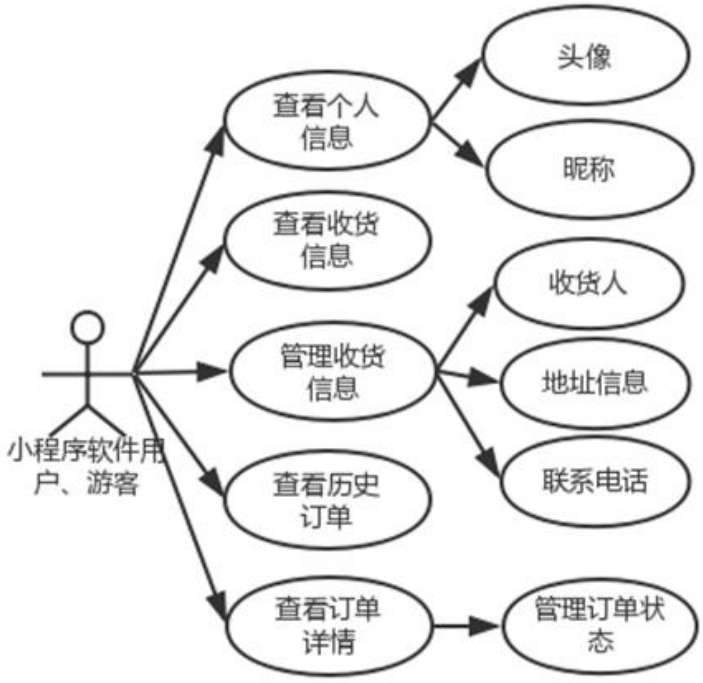

图 3.9 用户个人信息管理用例图

对于用户个人信息管理的功能需求分析如下：

（1） 用户可以在小程序购物系统客户端的导航栏处点击“我的”进入到个人信息管理页面。

（2） 个人信息管理部分主要包括三部分内容：用户微信头像、昵称的显示，用户的收货地址、电话的管理，用户订单的查看和管理。

（3） 在用户地址管理处，用户可以随时修改个人信息，包括姓名、电话、地址等信息，其中电话和地址的填写需要满足一定的填写规范，如果不符合规范则会给出系统提示。

（4） 用户可以在个人信息页面查看历史订单信息及其订单状态的简要信息，点击某一个订单，可进入查看订单详情。

（5） 订单详情页面可显示下单时间、订单编号信息，收货信息以及订单所包含的商品信息。

3.5　本章小结

本章中主要对微信小程序购物系统客户端的功能进行了需求分析，其中 3.1、3.2、3.3 小节是对小程序客户端系统的购物业务流程图和整个项目的数据流和整体的功能结构进行了分析，随后在此基础上，我们对小程序客户端购物系统的主要功能模块进行了功能需求分析，包括商品信息展示、商品分类、购物车、下单支付、个人信息管理的需求分析及 UML 用例图模型分析。这为接下来的小程序客户端的功能设计和实现做好了基础。

第 4 章　系统设计与实现

4.1　项目总体架构

本项目在总体架构上主要分为微信小程序客户端、服务器、数据管理三大部分。其中微信小程序客户端部分主要利用微信小程序的“MINA”框架（WXML，WXSS，JavaScript）以及其提供的丰富组件，采用 MVC 的设计模式来完成客户端界面和功能设计与实现，客户端可与消费者用户直接交互，提供流畅的购物体验。服务器端主要采用 ThinkPHP5 和 MySQL 构建 REST API 为客户端提供数据接口，其中 ThinkPHP5 作为服务器开发的一个外部框架，可在消费者用户使用微信小程序购物系统浏览商品、加入购物车、下单等的业务逻辑中，实现客户端与数据库数据请求以及数据处理的关键部分。除了本地服务器的支持之外，微信小程序的客户端实现的部分功能还需要服务器编写业务逻辑调用微信服务器的 API 接口来实现。MySQL 主要担任数据库的角色，可实现数据存储、数据表设计等功能， MySQL 数据库在与 TP5 服务器配合下向客户端提供可访问的数据接口。数据管理作为管理或者运营人员的数据管理平台，通过调用 API 从服务器的数据库中获取商品信息、订单数据等。系统总体架构图如图 4.1。

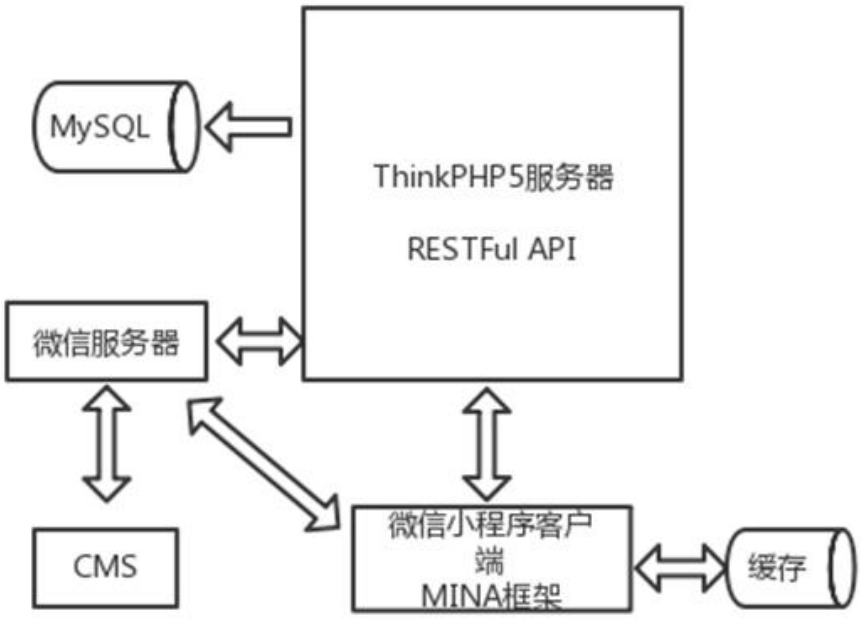

图 4.1　系统总体架构图

4.2　项目开发方法及环境部署

项目客户端的开发过程采用面向对象的开发方法，充分利用面向对象方法中的封装性、继承性和多态性的特点，将系统划分为相对独立的多个模块，每个模块具有自己的功能设计和实现要求，在开发过程中可以分模块集中开发实现，然后最后再将各个模块组织和集成，这样使得整个系统的结构层次分明，也便于开发工作的管理。

面向对象的软件开发方法是解决软件设计过程中所出现基础性问题的的解决方案,避免在开发中做重复的工作。进行可重复使用性设计,在软件的总体框架中可以将同一个功能模块的不同业务部分的应用通过面向对象的设计,以及模块的复用来实现重复利用,减少开发得重复工作,提高开发效率。例如分析系统中通用的处理方法，将其封装设计成相对独立可重复使用的代码作为基类，当功能在系统中多个模块应用时,直接调用基类就可以。这样会减少因为重复功能的代码的复制修改而带来的代码质量问题，同时也避免了代码的冗余。从系统的角度进行分析,利用对象的单位作为基本构造单元,可以使软件系统变得模块化、可复用性能好、易于维护，同时便于优化软件结构和质量。

项目的基础环境基于 XAMPP（Apache+MySQL+PHP+PERL）一个功能强大的建站集成软件包,本项目采用的是 XAMPP 的 7.1.12 版本 xampp-win32-7.1.12-0-VC14-installer 安装包，选择安装的组件有 PHP、MySQL 和 Apache 服务器。项目选择 ThinkPHP 5 作为 web 开发框架，在 Git 仓库中下载 ThinkPHP 5 5.0.7 版本的应用项目和核心框架，与 XAMPP 集成作为主要的业务逻辑和 API 的开发语言和框架。在开发工具选择上，购物系统客户端采用的是微信 web 开发者工具来编写微信小程序购物系统的主界面，采用 PHPstorm 来开发服务器的 PHP API 代码，使用 PostMan 作为服务器接口的测试工具，选用 Navicat 连接数据库作为 MySQL 的可视化管理工具。

4.3　微信小程序购物系统的数据库设计

数据库的设计是软件系统开发和建设中最基础和最核心的部分，良好的数据库设计可以让系统具有更快更好的运行速度。目前关系型数据库是我国中小型系统的主流选择。关系型数据库的设计要有严格的设计规范，对软件系统中的各类数据进行有效的组织存储和维护，为软件系统运行中的数据需求提供高效准确的处理速度。在实现这个目标的过程中数据库的规范设计就显得尤为重要，保持数据存储的一致性和完整性是提高数据存储和处理效率的重要保障之一。在数据库设计时，不仅需要满足数据库的设计准则，还需要结合具体的业务知识分析出系统中的不同实体之间的关系，并确定这些实体之间的关系类型。

本文根据小程序购物系统的功能需求分析，按照规范设计的要求，将对数据库设计的概念结构设计、逻辑结构设计、数据库配置与实施进行详细分析与设计。

4.3.1 数据库概念结构设计

数据库的概念结构设计的过程是将用户的需求综合抽象为一个信息结构的过程，概念结构是各类数据模型的共同基础。E-R 图是描述概念模型的有效工具。E-R 图由实体、属性、联系三个要素组成，例如本系统中商品信息就是一个实体，它包含描述商品的众多属性，联系表示实体之间的关系，实体与实体之间的联系可以分为三种：一对一、一对多、多对多。

本文从系统的业务和功能模块分析确定系统需要的实体集，以及实体与实体之间的联系。小程序购物系统中关键的实体信息如下：

（1）购物系统的商品信息实体图的属性包含商品 id、商品名称、单价、库存量、商品类别等，如图 4.2 所示：

图 4.2　商品信息实体图

（2）购物系统的用户信息实体图的属性主要包含用户 id，openid，用户昵称等信息，如图 4.3 所示：

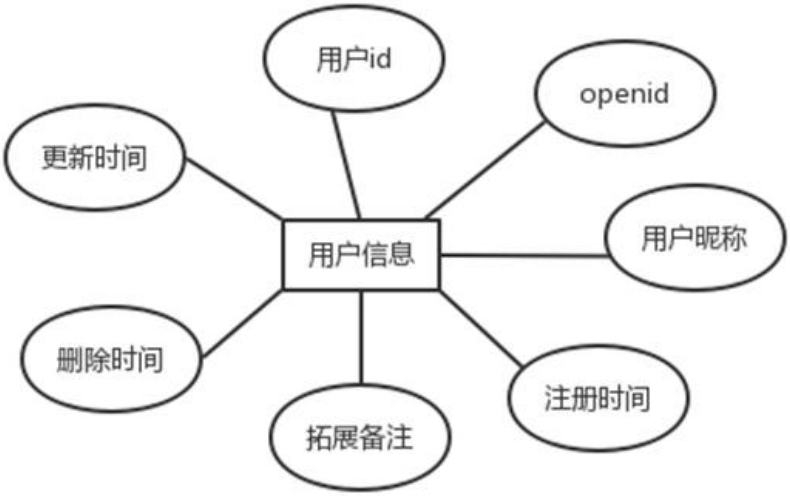

图 4.3　用户信息实体图

（3）购物系统的订单信息实体图的属性主要包含订单 id、订单号、用户 id、订单创建时间、订单金额、订单状态、快照订单等信息，如图 4.4 所示：

图 4.4　订单信息实体图

（4）购物系统的用户地址信息实体图的主要属性包含主键 id、用户 id、姓名、电话、详细地址等信息，如图 4.5 所示：

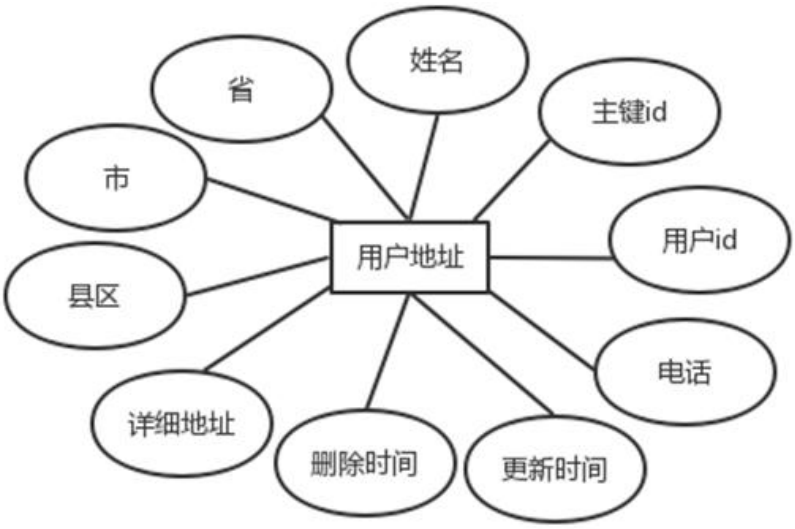

图 4.5　用户地址信息实体图

（5）购物系统的订单商品信息实体图的主要属性包含商品 id、订单 id、数量等信息，如图 4.6 所示：

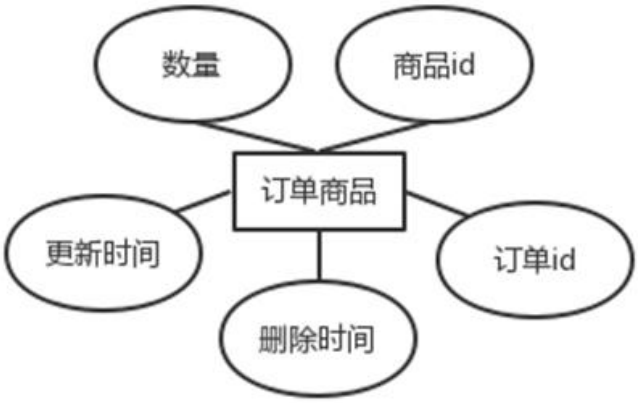

图 4.6　订单商品信息实体图

（6）购物系统的专题信息实体图的主要属性包含专题 id、专题名称、描述信息等信息，如图 4.7 所示：

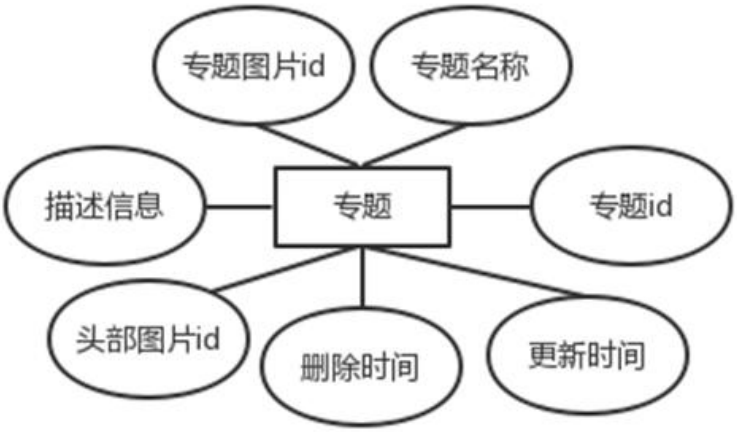

图 4.7　专题信息实体图

（7）购物系统的商品分类信息实体图的主要属性包含类目 id、类目名称、描述信息等信息，如图 4.8 所示：

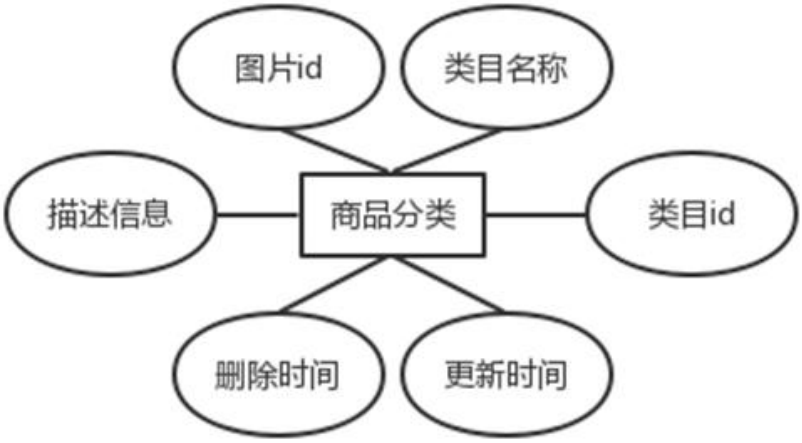

图 4.8　商品分类信息实体图

由系统的实体以及实体与实体之间的联系可得出系统 E-R 图，小程序购物系统的总体 E-R 图如图 4.9 所示：

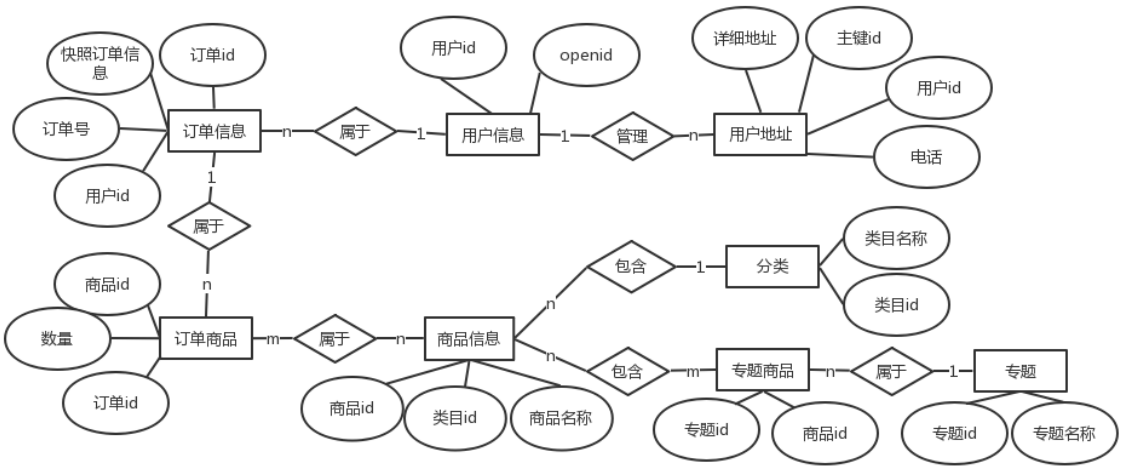

图 4.9　系统总体 E-R 图

在概念模型的设计中，采用 E-R 图来表示系统需求所抽象成的信息结构，其中用矩形表示 E-R 图中的实体，椭圆形表示属性，菱形表示联系。用户表与用户地址表之间是一对多的关系，用户表与订单表之间是一对多的关系，订单表和商品表通过订单商品连接，订单表中的一条记录对应订单商品表中的多条记录，订单商品与商品表之间是多对多的关系，分类表中的一种分类类目对应商品表中的多个商品，专题表和商品表之间通过专题商品表连接，商品表与专题商品表之间是多对多的关系，一个商品可能根据业务需求可能会属于不同的专题。

4.3.2　数据库逻辑结构设计

根据概念结构设计中的 E-R 图进行数据库的规则转换，将 E-R 图实体和实体间的联系转换为关系模式，并确定关系模式中的属性和码。实体中的属性与关系模式中的属性相对应，实体中的码与关系中的码相对应。本系统最后的数据模型以数据表的形式展现，具体说明了数据表中属性名称、主键、含义、约束条件，是否为空等信息，系统中主要的数据库表设计详细如下：

（1）专题商品信息表 theme_product 基本数据表，如表 4.1。

表 4.1　专题商品信息表（theme_product）

字段名	类型	大小	是否为空	字段描述
theme_id	int	11	否	主题外键
product_id	int	11	否	商品外键

（2）专题信息表 theme 基本数据表，如表 4.2。

表 4.2　专题信息表（theme）

字段名	类型	大小	是否为空	字段描述
id	int	11	否	主键 id
name	varchar	50	否	专题名称
description	varchar	255	否	专题描述
topic_img_id	int	11	否	主题图，外键
delete_time	int	11		删除时间
head_img_id	int	11	否	专题列表页，头图
update_time	int	11		更新时间

（3）商品信息表 product 基本数据表，如表 4.3。

表 4.3　商品信息表（product）

字段名	类型	大小	是否为空	字段描述
id	int	11	否	主键
name	varchar	80	否	商品名称
price	decimal	11	否	价格,单位：分
stock	int	11	否	库存量
delete_time	int	11		删除时间
category_id	int	11	否	商品类别
main_img_url	varchar	255	否	主图 ID 号
from	tinyint	4	否	图片来源
create_time	int	11		创建时间
update_time	int	11		更新时间
summary	varchar	50		摘要
img_id	int	11	否	图片外键

（4）用户信息表 user 基本数据表，如表 4.4。

表 4.4　用户信息表（user）

字段名	类型	大小	是否为空	字段描述
id	int	11	否	主键
openid	varchar	50	否	openid
nickname	varchar	50	否	昵称
extend	varchar	255		扩展备注
delete_time	int	11		删除时间
create_time	int	11		注册时间
update_time	int	11		更新时间

（5）订单信息表 order 基本数据表，如表 4.5。

表 4.5　订单信息表（order）

字段名	类型	大小	是否为空	字段描述
id	int	11	否	主键
order_no	varchar	20	否	订单号
user_id	int	11	否	外键，用户 id
delete_time	int	11		删除时间
create_time	int	11		创建时间
total_price	decimal	11	否	订单总额
status	tinyint	4	否	订单状态
snap_img	varchar	255		订单快照图片
snap_name	varchar	80		订单快照名称
total_count	int	11	否	总数量
update_time	int	11		更新时间
snap_items	text	0		订单其他信息快照
snap_address	varchar	500		地址快照
prepay_id	varchar	100		订单微信支付的预订单 id

（6）用户地址信息表 user_address 基本数据表，如表 4.6。

表 4.6　用户地址信息表（user_address）

字段名	类型	大小	是否为空	字段描述
id	int	11	否	主键 id
name	varchar	30	否	收获人姓名
mobile	varchar	20	否	手机号
province	varchar	20	否	省
city	varchar	20	否	市
country	varchar	20	否	区
detail	varchar	100	否	详细地址
delete_time	int	11		删除时间
user_id	int	11	否	外键
update_time	int	11		更新时间

（7）订单商品信息表 order_product 基本数据表，如表 4.7。

表 4.7　订单商品信息表（order_product）

字段名	类型	大小	是否为空	字段描述
order_id	int	11	否	联合主键，订单 id
product_id	int	11	否	联合主键，商品 id
count	int	11	否	商品数量
delete_time	int	11		删除时间
update_time	int	11		更新时间

（8）商品图片信息表 product_image 基本数据表，如表 4.8。

表 4.8　商品图片信息表（product_image）

字段名	类型	大小	是否为空	字段描述
id	int	11	否	主键
img_id	int	11	否	图片 id，外键
delete_time	int	11		删除时间
order	int	11	否	图片排序序号
product_id	int	11	否	商品 id，外键

（9）图片信息表 image 基本数据表，如表 4.9。

表 4.9　图片信息表（image）

字段名	类型	大小	是否为空	字段描述
id	int	11	否	主键 id
url	varchar	255	否	图片路径
from	tinyint	4	否	图片来源
delete_time	int	11		删除时间
update_time	int	11		更新时间

（10）分类信息表 category 基本数据表，如表 4.10。

表 4.10　分类信息表（category）

字段名	类型	大小	是否为空	字段描述
id	int	11	否	主键 id
name	varchar	50	否	分类名称
topic_img_id	int	11	否	外键，关联 image 表
delete_time	int	11		删除时间
description	varchar	100		描述
update_time	int	11		更新时间

4.3.3　数据库的配置与实施

本文结合本微信小程序购物系统业务的实际情况，在项目开发环境中建立 MySQL 数据库，并使用 Navicat 图形化管理数据库，开源数据库 MySQL 作为关系型数据库系统可实现网络化，满足本系统的实际需要，并且 PHP 中所有函数对 MySQL 数据库给予支持。

由数据库的概念结构和逻辑结构分析，根据数据表设计进行 MySQL 数据库的设计与实现。建立数据库为 zerg，以 root 为用户名，在本数据库中共建立了 14 张表，创建与数据表相对应的 SQL 语句，并输入了相应表的测试数据。在项目服务器中建立服务器连接数据库的配置文件 database.php，其中主要的配置参数如下：

```
'type'          => 'mysql',  //数据库类型
'hostname'      => '127.0.0.1',  // 服务器地址
'database'      => 'zerg',  // 数据库名
'username'      => 'root',  // 用户名
'password'      => '123456',  // 密码
```

```
'hostport'        => '3306',   // 端口
'charset'         => 'utf8',   // 数据库编码默认采用 utf8
'debug'           => true,   // 数据库调试模式
```

通过以上配置实现服务器与数据库的通信，并在测试环境中中开启 SQL 日志记录，便于查看数据库的性能表现和测试记录。服务器访问数据库的数据则采用 ORM（Object Relational Mapping）对象-关系映射的方式实现。ORM 是为了解决面向对象开发方法与关系型数据库的映射而产生的，实现了对象型数据和关系型数据的相互转化，具有更高的开发灵活性。相比原生 SQL 简单的增删改查操作，ORM 利用模型对象可以实现更复杂的业务逻辑，将访问数据库的代码封装到模型类中，模型类的命名与数据库中的表名一致，然后在服务器的控制器层调用模型类来支持控制器层中的业务逻辑。例如在控制器层的获取首页广告数据示例如下：

```
use app\api\model\Banner as BannerModel;
$Banner = BannerModel::getBannerByID($id);
```

4.4 微信小程序购物系统设计与实现

用户在小程序购物系统客户端来完成网上浏览商品、加入购物车、结算支付等行为。这也是本文所要实现的主要功能。在用户完成这一系类的行为中，除了要有微信小程序中 MINA 框架编写美观大方的界面设计外，还需要微信小程序购物系统客户端与本地服务器和微信服务器中 API 逻辑接口的支持，将 API 设计嵌入到界面的编写中。此外用户在微信小程序购物系统客户端购买商品过程中产生的数据还需要通过服务器与 MySQL 数据库交互。在小程序客户端购物系统的代码实现中，遵循 MVC 设计模式思想，.js 文件充当控制器的角色，model.js 文件是模型层的角色，在此文件中会编写调用本地服务器接口的代码，最后的.wxml 文件和.wxss 文件相当于 MVC 中的视图层。基于此结构编写微信小程序购物系统客户端的界面和功能模块。

本小节以实现客户端为目标综合介绍了客户端以及和服务器交互的代码实现原理，重点完成了五大模块的详细设计和实现，这五大模块分别为商品信息展示模块、商品分类模块、购物车模块、下单支付模块、个人信息管理模块。在设计和实现过程中，充分利用于原型图、时序图、流程图和代码分析来表述设计实现的过程，并展示了程序的运行效果图。

4.4.1 客户端与服务器的交互设计

小程序购物系统客户端的页面效果实现需要请求服务器的数据，除此之外，客户端页面接收用户的动作时，也需要请求服务器的的数据更新页面数据，这个过程需要客户端的.wxml 文件、.js 文件和服务器 AP 三者的配合实现，其具体的实现原理如图 4.10。

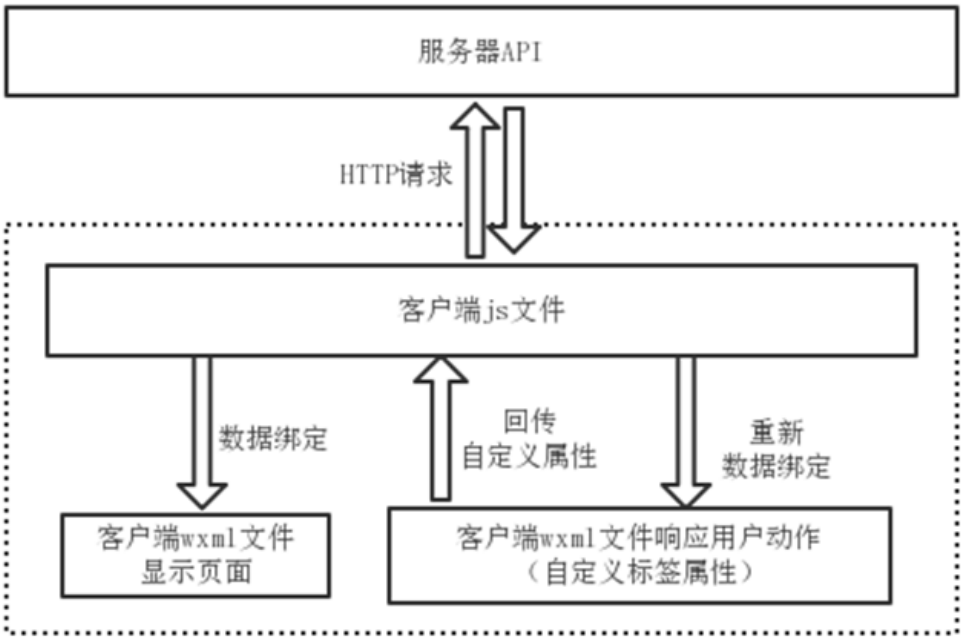

图 4.10　客户端与服务器交互原理图

当小程序客户端页面显示时需要请求服务器的数据，此时，客户端的 js 类文件（包括小程序模型层、控制层以及基类 base.js 文件）会发送 HTTP 请求到服务器 API 请求获取服务器和数据库的数据，服务器接到请求后会调用相关的方法类返回处理结果，此时服务器会先将结果返回到 js 文件的请求函数中，在 js 文件中通过 setData 数据绑定的方式将数据返回到 wxml 文件，此时视图层才能将获取到数据的显示到页面。

上述的是静态的数据显示的服务器请求过程，当用户在客户端页面发生交互动作时，则需要控制器层和服务器的代码作出相应的业务逻辑判断再返回结果。这个客户端页面动态的请求服务器数据的过程主要分为两步：获取用户动作和响应用户动作。获取到用户的动作需要 wxml 文件标签自定义属性数据，当监听到用户的动作时，比如点击事件，视图层会通过 event 事件的方式将标签自定义属性数据回传到控制层和模型层。接下来由 js 文件响应处理接收到的事件请求，通过标签自定义属性数据调用相应的方法完成业务逻辑的处理，需要请求服务器数据的，则需要发送 http 请求到服务器 API，当服务器处理完成后返回处理结果到 js 文件，最后重新数据绑定服务器数据到视图层更新显示页面。

4.4.2　客户端的 MVC 模式设计

MVC（Model View Controller）是模型(model)－视图(view)－控制器(controller)三层架构的缩写，是一种将应用程序的数据、界面显示、业务逻辑分离的软件代码设计方法。MVC 的分层大致体现在模型、视图、控制器三个核心部件，他们各自处理自己的任务。其中：模型是处理应用程序数据逻辑的部分。一般模型对象负责在数据库的数据存取和

处理。视图用于处理数据显示的部分。通常视图是依据模型数据创建的。控制器负责程序调度控制，用于处理应用程序中用户交互的部分，负责读取从视图层的数据，控制用户输入，并向模型发送数据请求。

在小程序客户端的程序开发中，每个页面的实现都需要四种文件类型的支持，分别是.wxml 后缀的模板文件、.wxss 后缀的样式文件、.js 后缀的逻辑文件、.json 后缀的配置文件，其中.wxml 后缀的模板文件、.wxss 后缀的样式文件用来完成页面的结构和样式设计，.json 后缀的配置文件用于当前页面的配置，.js 后缀的逻辑文件用于页面展示过程中的用户操作的逻辑实现。对应 MVC 的设计模式思想，本文把.wxml 后缀的模板文件作为视图层，将.js 后缀的逻辑文件分为.js 和 model.js 两种类型的文件，分别用于控制器层和模型层。在本文设计的小程序 MVC 架构中，.wxml 后缀的模板文件作为视图层用于展示页面和数据；.js 后缀的文件作为控制器层用于页面逻辑的控制，以及用户在视图层所发生交互动作的事件方法定义；model.js 后缀的文件作为模型层用于数据的读取和处理，通过发送 http 请求与服务器通信，获取相应的数据并返回给控制器。

小程序客户端的 MVC 模式设计原理图如图 4.11。

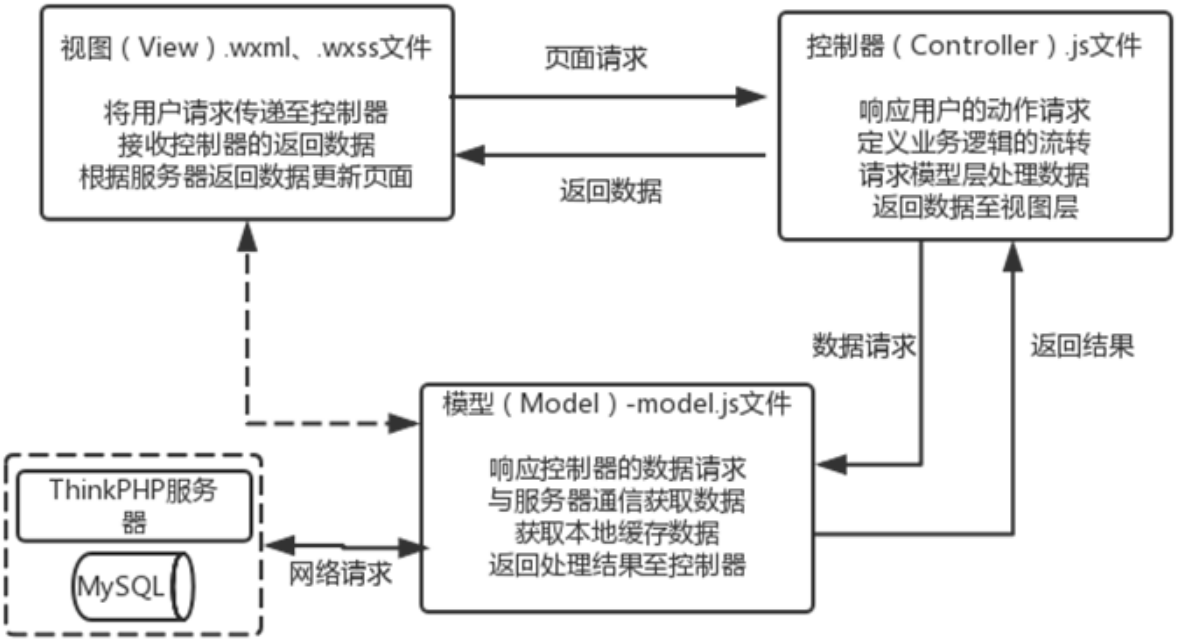

图 4.11　小程序客户端的 MVC 模式设计原理图

由上图可知本文小程序的 MVC 模式设计的基本处理方法，当用户在视图层与小程序页面触发事件请求时，用户产生的请求会调用至控制器的相应方法，在控制器的相应方法中完成用户所触发事件的业务逻辑判断与处理，返回业务的处理结果至视图层，当需要获取或者更新数据库中的数据时，控制器层会调用模型层 model.js 文件中的相应方法与服务器 API 通信，获取或更新数据库的数据，完成后将请求结果返回至控制器，控制器将业务逻辑的处理结果返回相应的视图层更新页面。视图层不直接与模型层请求数

据，而是由控制器处理业务逻辑并向模型层请求处理用户在与界面交互过程中所需要的数据需求。这样将界面展示、业务逻辑处理和数据处理分离的设计方法，降低了客户端系统各个功能模块的耦合度，结构清晰，增加了功能组建的复用性，使得客户端系统的设计与实现变得更加灵活可控。

4.4.3　商品信息展示模块的设计实现

用户浏览商品的模块的功能设计和实现主要在微信小程序客户端的系统的“首页”页面中。“首页”中的商品包括头部的 banner 广告商品和精选主题的商品和底部的最近新品商品矩阵。

4.4.3.1　banner 广告商品

广告商品的展示实现了商品图片的自动轮播效果，设置固定的时间使图片自动循环切换，同时指示器（图片上的小圆点）也随之切换。其实现过程分为布局设计，加载广告商品图片，循环切换广告商品图片三步。

使用简要的原型图来表示我们实现的 banner 广告图片循环播放的效果，如图 4.12。

图 4.12　banner 广告原型图

接下来，我们根据功能需求和原型图的设计来实现 banner 广告商品的显示。首先我们使用 swiper 类来加载可以自动切换的 banner 广告图片，自动切换图片主要是参数 autoplay="true"的作用。循环的广告图片由 bannerArr 数组中的图片来控制，并显式显示 indicator-dots 也跟随切换。Banner 广告循环轮播显示的代码在 home.wxml 文件中实现，并在 home.wxss 中控制其图片的样式，当点击商品图片跳转到商品详情页面时，主要有由 bindtap="onProductsItemTap" 参数设置来实现，商品详情页的展示由调用 onProductsItemTap 方法来实现，此方法在 home.js 中定义的，onProductsItemTap 方法内的 wx.navigateTo 可以实现跳转到商品详情页的功能。简要的代码如下。

```
<swiper indicator-dots="true" autoplay="true" class="swiper">
  <block wx:for="{{bannerArr}}">
    <swiper-item class="banner-item"
      data-id="{{item.key_word}}" bindtap="onProductsItemTap">
```

```
            <image mode="aspectFill" class="item-image"
                src="{{item.img.url}}">
            </image>
        </swiper-item>
    </block>
</swiper>
```

其中从数据库中加载数据到客户端的实现主要由数据绑定和箭头函数的功能代码负责。数据绑定的功能在 home. js 控制器中实现，调用箭头函数得到从服务器异步获取的数据，并在_loadData 模块中通过调用 this.setData 微信数据绑定 API 从数据库中获取 bannerArr 数组的 res 数据到视图页面。

```
home.getBannerData(id, (res) => {
        this.setData({
          'bannerArr': res
        });
    });
```

小程序从服务器请求调用数据库数据的功能主要有 base.js 文件中封装的 request 基类来实现的，具体到 banner 广告数据的调用，则是在 home 中的 home-model.js 模型文件中，定义了请求 banner 广告图片的 request 方法所需要的参数信息来实现的。其中的 url 则实现了小程序客户端与开发者服务器的通信，调用 callback 异步回调函数采用异步方式获得服务器的参数。在服务器部分，通过服务器的 banner 控制器中的 getBanner 方法来获取数据库中 banner 表以及 banner_item 表的数据信息。其中 banner 广告模型层的代码如下：

```
getBannerData(id, callback){
    var params = {
      url: 'banner/'+id,
      sCallback:function(res){
        callback && callback(res.items);
      }
    }
    this.request(params);
}
```

其中发起 http 请求的 request 方法实现采用了面向对象的继承法。由于在小程序客户端的很多地方都需要发起 http 请求的需求，为了增加代码的复用性和整洁性，将这个方法封装到了 base.js 文件的 Base 基类中。在 Base 基类中定义 request 方法，封装了微

信的 wx.request()方法实现发起 http 请求的功能，这样在 home-model.js 模型文件中通过继承即可。

用户浏览 Banner 广告商品的程序时序图，如图 4.13：

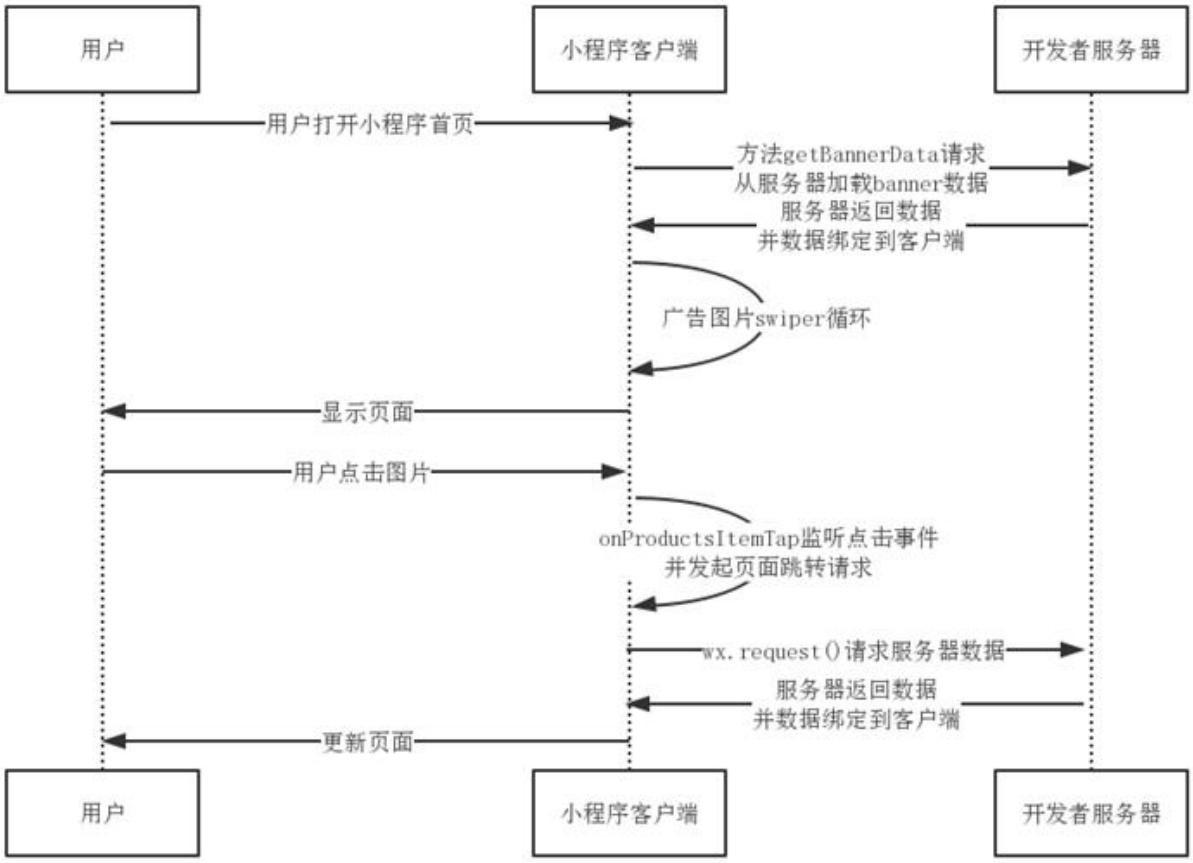

图 4.13　浏览首页 banner 广告商品程序时序图

以上的代码设计就可以完成小程序客户端通过 http 请求与服务器通信，通过服务器路由 API 信息调用的相关的方法获取数据库数据并通过小程序客户端的视图层完美的展现到界面上的完整流程了。

在小程序客户端的运行效果图如图 4.14。

图 4.14　首页 banner 广告运行效果图

4.4.3.2　主题展示

精选主题商品的展示部分同样需要小程序通过服务器调用数据库数据，这一层的功能需要在模型层 home-model.js 文件调用 base 文件中的 request 基类绑定精选主题数据的请求参数，并在控制器层 home.js 文件中进行请求数据的 this.setData 数据绑定，在视图层的 home.wxml 文件中进行界面的设计与展示，其中精选主题部分分为图片大小不同上下两栏，在视图设计中采用了微信的 wx:if 、wx:else 判断方法来统一实现精选主题部分的界面设计。简要代码如下：

```
<view class="home-main-header">精选主题</view>
    <view class="theme-box">
      <block wx:for="{{themeArr}}">
          <view wx:if="{{index==2}}" class="theme-item big"
        data-id="{{item.id}}" data-name="{{item.name}}"
        bindtap="onThemesItemTap">
          <image src="{{item.topic_img.url}}"></image>
        </view>
        <view wx:else class="theme-item" data-id="{{item.id}}"
        data-name="{{item.name}}" bindtap="onThemesItemTap">
          <image src="{{item.topic_img.url}}"></image>
        </view>
      </block>
```

```
</view>
```

其中，与 banner 广告图片的展示类似的是，当点击专题图片时，可以跳转到的有一组商品矩阵的专题商品，而这组专题商品页需要通过调用 onThemesItemTap 方法来实现。在这个方法中定义了 wx.navigateTo 到另一个页面的功能，在这个页面跳转的过程中，通过 getDataSet（）方法来进行传递和接受页面所需要的参数。

```
onThemesItemTap: function (event) {
  var id = home.getDataSet(event, 'id');
  var name = home.getDataSet(event, 'name');
  wx.navigateTo({
    url: '../theme/theme?id=' + id + '&name=' + name
  })
},
```

点击精选主题图片，会跳转到相应主题的商品矩阵页。精选商品的商品矩阵页面实现方式与 home 首页的架构相同，商品展示方式的设计与实现与最近新品部分类似，引用了 product template 模板来定义商品矩阵界面。

在小程序客户端的运行效果图依次如图 4.15。

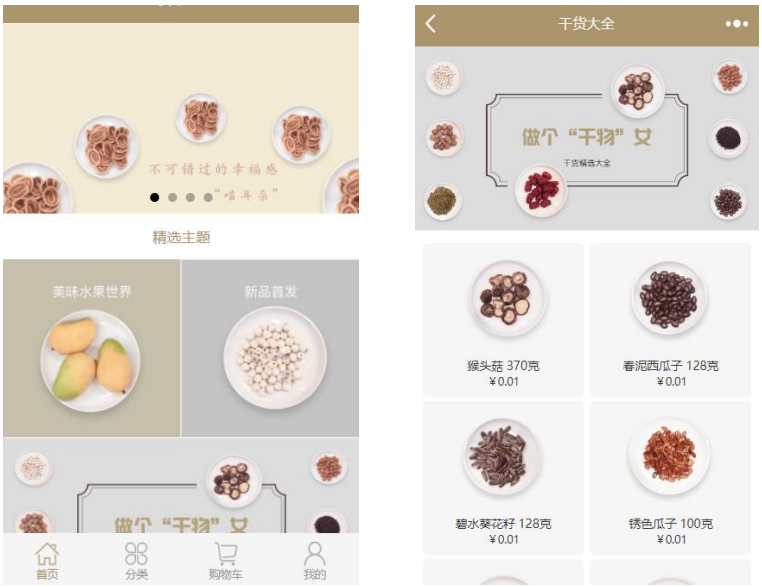

图 4.15　用户浏览主题商品运行效果图

4.4.3.3　最近新品 template 模板

最近新品的内容主要是将商品的上架时间倒序排序，选取 15 个商品展示到首页，以此来增加用户对新品的关注度。这部分的主要内容是将商品以矩阵形式展示出来，方便用户可以知晓上新商品的概要信息，当用户点击商品时可以跳转到相应商品的详情页。由此可见，在这个商品矩阵中，每一个商品的 UI 设计是相同的，但是商品数据不同。通过视图层的 home.wxml 文件就可以实现基本样式，但是考虑到商品矩阵的功能会

在之后的代码编写如专题商品的展示中多次使用，所以在这部分使用 template 模板来完成商品矩阵列表功能，这无疑增加了代码的复用性，使代码更加精简美观。调用 template 模板的最新商品展示的视图层简要代码如下：

```
<import src="../tpls/products/products-tpl.wxml" />
<view class="home-main-products">
        <view class="home-main-header">最近新品</view>
        <template is="products" data="{{products:productsArr}}">
        </template>
  </view>
```

从代码中可以看出，引用了../tpls/products/products-tpl.wxml 文件下的 product template 模板，并将从服务器请求到的数据通过 date 关键字传递到了 template 模板中。由于product的wxss和wxml的视图设计会在多个模块用到，所以在将其封装成了product template 模板，以增加代码的复用性和简洁性。

模板文件 product template 放在项目的 tpls 文件下，将最近新品展示的样式设计代码 products-box class 放在<template>标签下，与页面 wxml 文件不同的是由于 template 模板不支持 js 文件，所以模板的 wxml 文件中不可以直接加载数据，而需要在页面通过数据绑定和异步回调函数将得到的服务器数据传递到 template 模板中。Template 模板的代码设计实现如下：

```
<template name="products">
    <view class="products-box">
        <block wx:for="{{products}}">
            <view class="products-item"
             bindtap="onProductsItemTap" data-id="{{item.id}}">
                <image class="products-image"
                src="{{item.main_img_url}}" mode="aspectFill"></image>
                <view class="products-item-bottom">
                    <text class="name">{{item.name}}</text>
                    <view class="price">￥{{item.price}}</view>
                </view>
            </view>
        </block>
    </view>
</template>
```

至此，我们已经设计并实现了用户浏览商品模块的主要功能，并主要介绍了其中用到的小程序 MVC 思想，数据绑定，template 模板在实现小程序购物系统客户端中的应用。在小程序客户端的运行效果图依次如图 4.16。

图 4.16　最近新品运行效果图

4.4.4　商品分类模块的设计实现

商品分类页面可以实现商品种类的快速导航功能，方便用户快速找到意向商品。用户可以在左侧的商品类目栏选择一种商品种类，相应的在右侧即可看到对应的商品，点击商品的图片和文字即可进入商品详情页面。根据以上功能需求具体实现过程如下：

在界面设置上，在 category.wxml 文件 container 容器类中的 category-box 包含左右两个 box，分别为 left-box 展示商品类目信息和 right-box 展示相应类目下的所有商品信息。left-box 中循环遍历读取从 category 的控制器和模型中获得的数据展示到界面，并监听在商品类目上的点击动作，当用户点击商品类目时，客户端会立刻调用服务器中的的 getCategoryType 方法获取分类类目信息和 getProductsByCategory 方法获得相应分类的商品信息，并展示到页面中。

为了使页面的加载速度更快，避免反复向服务器 API 接口请求数据，可以客户端从服务器第一次请求到的商品分类数据放到小程序客户端缓存中。当用户切换分类页面时，会调用 category.js 文件中的 changeCategory 方法判断当前分类下的商品数据是否已经被加载过，如果已经加载过当前分类的商品数据，则可以直接从小程序的本地缓存中读取相应的分类商品数据即可。这样做也有存在一定的问题，即客户端的分类页面数据不是实时从服务器加载而来的，但是对于分类页面的商品数据变化频率较小，用户打开小程序初次请求的数据就可以满足当前浏览分类商品的需求。避免频繁的向服务器发送这类非紧要的请求，既可以提供给用户良好交互体验，也可以给服务器减少不必要的数

据请求压力。综合考虑利弊，这样的处理方式可以使用户与小程序客户端的交互更加友好。

4.4.4.1 分类模块

在商品分类页面的 right-box 部分，用户可以点击商品类目时就可以切换到相应的分类商品的数据页面，由于一种商品类目就会有一个商品分类数据页面，页面分为广告图片、标题、分类商品矩阵三部分，不同的商品类目的商品分类数据页面框架形式相同，但是其中的商品类目和商品信息在变化，所以此处我们再次使用了 template 模板来定义商品分类数据页面。通过 onProductsItemTap 方法绑定商品的 id，当点击商品图片时，就可以调用 categorydetail template 模板中的 onProductsItemTap 方法来完成点击商品图片时跳转到相同 id 号的商品详情页。调用 template 的简要代码如下：

```
<import src="../tpls/category/category-tpl.wxml"/>
<view class="right-box">
  <view class="foods-type-box">
    <template is="categorydetail" data="{{categoryInfo:categoryProducts}}">
    </template>
  </view>
</view>
```

分类检索商品功能模块的商品分类页面运行效果图如图 4.17。

图 4.17 商品分类页面运行效果图

4.4.4.2 商品详情模块

当用户点击商品图片时，则会在商品分类数据页面触发 onProductsItemTap 事件方法 navigateTo 跳转到商品详情页。点击分类商品转到的商品详情的交互流程由 category template 模板中的 bindtap="onProductsItemTap"事件响应，当监听到点击商品时，则会调用 category.js 文件中的 onProductsItemTap 方法跳转到商品详情页。商品详情页的设计分为头部和底部两部分。其中头部主要展示商品的概要信息，并且是商品详情页与用户产生交互交互动作的主要区域，如选择商品数量，加入购物车等；底部主要展示商品的详细信息。在用户选择商品数量前，首先对商品的库存量进行了监测，当商品库存量为 0 时，商品数量选择按钮为灰色不可选中状态，并且显示“无货”，选择商品数量部分主要使用了 picker 组件，简要代码如下：

```
<picker class="{{product.stock==0?'disabled':''}}"
 range="{{countsArray}}" bindchange="bindPickerChange" value="1">
    <view>
      <text class="counts-tips">数量</text>
      <text class="counts-data">{{productCount}}</text>
      <image class="counts-icon" src="../../imgs/icon/arrow@down.png">
      </image>
    </view>
 </picker>
```

Picker 组件主要有两个属性：range 和 value。range 数组用来定义可选择的数量范围，此处定义了数组的取值范围为[1,10]，value 用来定义用户选择的第几个数值。当用户点击数量选择按钮时调用 bindPickerChange 方法，获取用户选择的商品数量，采用数据绑定方式将 productCount 从小程序的控制器层传回视图层。

商品详情页的底部主要展示商品的详细信息，分为三个 tab 页：商品详情、产品参数，售后保障。商品详情部分以图片形式展示，产品参数和售后保障部分以文字形式展示。简要实现代码如下：

```
<block wx:for="{{['商品详情','产品参数','售后保障']}}">
    <view class="tabs-item {{currentTabsIndex==index?'selected':''}}"
    bindtap="onTabsItemTap" data-index="{{index}}">
      {{item}}
    </view>
</block>
……
<view class="product-detail-imgs" hidden="{{currentTabsIndex!=0}}">
```

对 tab 页数组进行循环遍历，当点击 tab 页标题时，调用 onTabsItemTap 方法将相应的 tab 页进行显式展示,并对未选中的 tab 页隐藏。商品的详细信息均是从服务器的数据库中请求获取的数据。

分类检索商品功能模块的商品详情页面展示图如图 4.18。

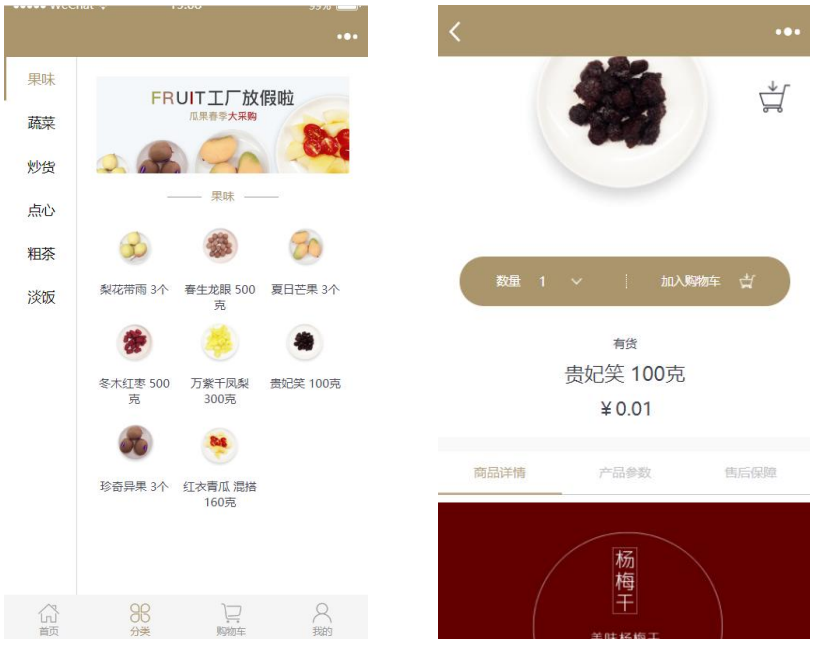

图 4.18　商品详情页面运行效果图

4.4.5　购物车模块的设计实现

网上商城购物车与现实超市的购物车类似，用于用户存放选购商品。同时，用户还可以对购物车中商品进行管理操作。本系统中可以通过商品详情页点击购物车标志 wx.switchTab 跳转进入购物车，还可以在小程序下方的导航栏点击购物车，即可显示购物车页面。进入到购物车页面后，可以点击 cart-item 部分的三个 button 对订单的数量进行更改。点击下单按钮即可跳转到付款页面。购物车模块主要包括购物车页面设计与功能逻辑的实现两部分。

购物车的页面设计及样式控制主要在视图层 cart.wxml、cart.wxss 文件中实现。当购物有数据时，显示购物车页面；当购物车没有数据时，显示“您还没有添加任何商品”页面。购物车页面主要分为 class="cart-box"和 class="footer-account-box all-accounts-box" 两个 view 组件。在 cart-box view 组件展示购物车商品的概要信息，用户对商品的数量和种类的操作区。在 footer-account-box all-accounts-box view 组件展示购物车的选择商品总金额，以及用户对购物车商品进行全选、下单的操作区。以数据绑定的方式将静态数据加载到购物车页面，采用事件监听回调的方式处理用户与购物车的交互动作。

我们采用面向对象的方式来编写购物车业务逻辑部分。首先购物车的数据来自以小程序客户端本地商品详情页中添加商品到购物车的数据，所以购物车的数据存储在小程

序的本地缓存中。购物车的功能实现总体规划为：在控制器 cart.js 完成购物车页面的逻辑设计；在模型 cart-model.js 中完成购物车数据的获取及处理。购物车模块功能实现的业务流程图如图 4.19：

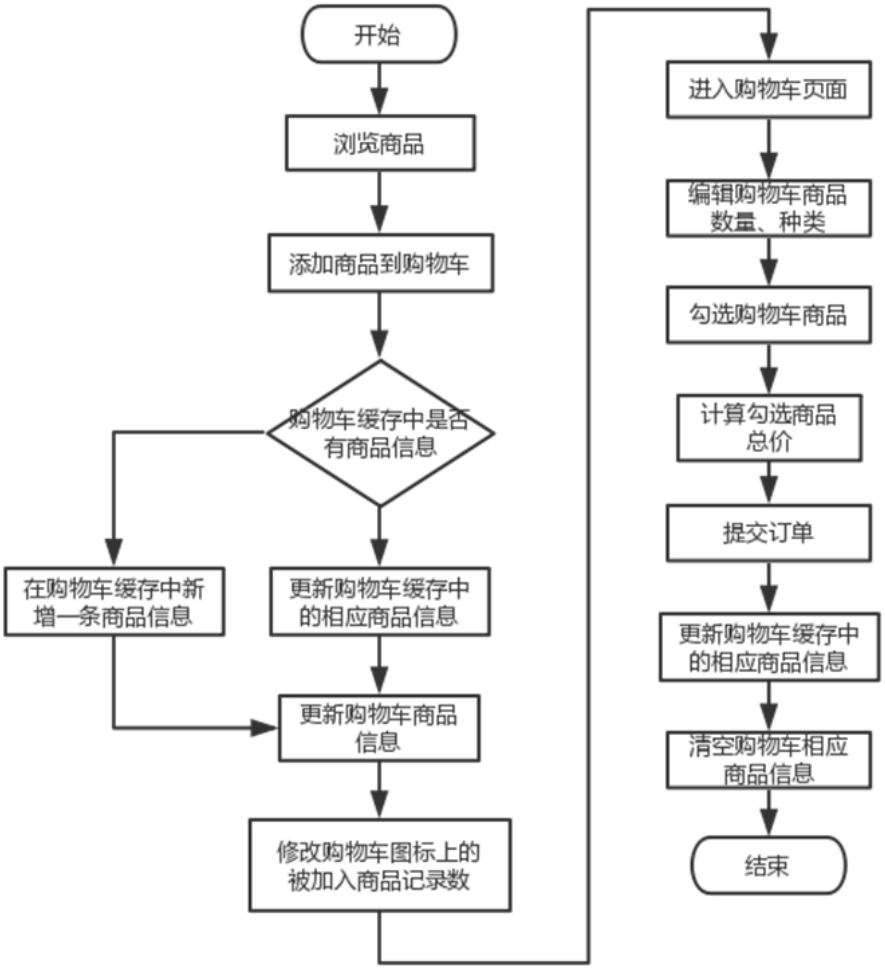

图 4.19　购物车模块程序流程图

从购物车数据产生到用户对购物车商品的编辑和处理等业务逻辑的实现如下：

（1）添加商品到购物车

购物车的数据从商品详情页产生。当用户在商品详情页点击“加入购物车”时，会调用 product.js 文件中的 onAddingToCartTap 事件回调函数，在此函数中会首先调用 addToCart 方法获取当前商品的 id、name、图片 url、price 属性信息，然后引用 cart-model.js 文件中的 add 方法执行添加商品到购物车并更新保存本地缓存的购物车数据功能。与此同时，在商品详情页面右上角的购物车标识会更新显示当前购物车中的商品数量总和，在事件回调函数 onAddingToCartTap 中会调用 cart-model.js 模型类中的 getCartTotalCounts 方法获取购物车缓存中的商品总数量，更新商品详情页中当前购物车中的商品总数量 cartTotalCounts，并以数据绑定的方式将显示到商品详情页的视图层。在商品详情页的购物车交互功能实现逻辑代码简要如下：

```
import { Cart } from '../cart/cart-model.js';
var product = new Product();
var cart = new Cart();
  onAddingToCartTap:function(event){
    this.addToCart();
    var counts =
      this.data.cartTotalCount + this.data.productCount;
    this.setData({
      cartTotalCounts: cart.getCartTotalCounts()
    });
  },
  addToCart:function(){
      var tempObj = {};
      var keys = ['id', 'name', 'main_img_url', 'price'];
      ......
      cart.add(tempObj, this.data.productCount);
 },
```

模型层 cart-model.js 文件中的 add 方法实现添加商品到购物车的主要业务逻辑。在商品详情页的事件回调函数调用 onAddingToCartTap 方法添加到购物车之前需要通过 getCartDataFromLocal 方法读取购物车的缓存数据，并调用_isHasThatOne 方法判断商品是否已经被添加到购物车中，并返回这个商品的数据信息，若商品不存在购物车中，则向购物车中新增一条商品条目，否则对购物车中相应商品的数量进行更改。最后对本地缓存中的购物车数据进行更新和保存。购物车模型层添加商品到购物车的简要代码如下：

```
  add(item, counts) {
    var cartData = this.getCartDataFromLocal();
    var isHasInfo = this._isHasThatOne(item.id, cartData);
    if (isHasInfo.index == -1) {
      item.counts = counts;
      item.selectStatus = true;
      cartData.push(item);
    }
    else {
      cartData[isHasInfo.index].counts += counts;
    }
```

```
        wx.setStorageSync(this._storageKeyName, cartData);
    }
```

控制器层 cart.js 主要与 cart 的视图层配合完成购物车页面的业务逻辑控制。通过 cart.js 控制器中的 onshow 类来加载购物车界面。当用户进入购物车页面时，可以获取当前缓存中的最新的购物车页面数据。简要页面监听函数代码如下：

```
import { Cart } from 'cart-model.js';
var cart = new Cart();
onShow: function () {
    var cartData = cart.getCartDataFromLocal();
    var cal = this._calcTotalAccountAndCounts(cartData);
    this.setData({
      selectedCounts: cal.selectedCounts,
      selectedTypeCounts: cal.selectedTypeCounts,
      account: cal.account,
      cartData: cartData
    });
},
```

其中，调用 cart.getCartDataFromLocal()方法用于获取小程序本地缓存中全部的购物车数据，_calcTotalAccountAndCounts()方法计算购物车的选中的商品的信息如商品总数量、商品的种类总数、当前所选商品的总金额。

用户进入购物车页面时，onshow 类中的数据会以数据绑定的方式将这些信息初始化展示在视图层。当用户在购物车页面对商品种类进行操作，如复选框和全选框的选择或取消选择操作，此时会触发 toggleSelect 和 toggleSelectAll 事件监听函数更改缓存中商品的选中状态，并调用_resetCartData 函数重新计算购物车中所选商品的总数量、总种类数、总金额，同时更新视图层中数据绑定的 selectedCounts、selectedTypeCounts、account、cartData 四个参数的值。

此外模型层中的一些其他方法对用户在购物车页面的交互动作实现起了关键的支持作用。例如：_changeCounts 方法：修改购物车中商品的数量；cutCounts 方法：通过商品 id 对购物车中的商品进行删除操作；execSetStorageSync 方法：对本地缓存中的购物车数据进行更新和保存等。

（2）编辑购物车

当用户在商品条目处点击“-”或“+”对商品的商量进行更改时，则会触发 cart.js 控制器文件中的 changeCounts()事件监听函数，在 changeCounts()函数中通过视图层获取事件的商品 id 和操作类型 type，进而判断是增加商品数量还是减少商品数量。增加商品数量和减少商品数量的功能通过调用 cart-model.js 模型文件中的 addCounts 方法和

cutCounts 方法实现。最后更新缓存中的购物车数据，并调用_resetCartData 方法重新计算所选商品的总金额和总数量更新到视图层。如上分析，购物车商品数量的更改的主要业务逻辑代码如下：

```
changeCounts: function (event) {
    var id = cart.getDataSet(event, 'id'),
        type = cart.getDataSet(event, 'type'),
        index = this._getProductIndexById(id),
        counts = 1;
    if (type == 'add') {
        cart.addCounts(id);
    } else {
        counts = -1;
        cart.cutCounts(id);
    }
    this.data.cartData[index].counts += counts;
    this._resetCartData();
},
```

商品删除功能与商品数量的增加、减少功能类似。当点击删除商品按钮时会触发控制器 cart.js 文件中的删除商品事件监听函数 delete()，把相应的商品从购物车中删除，然后调用_resetCartData 方法重新计算所选商品的总金额和总数量更新到视图层，最后调用模型层 cart-model.js 文件中的 delete 方法对小程序本地缓存中的购物车数据进行更新。

购物车功能模块的页面展示图如图 4.20。

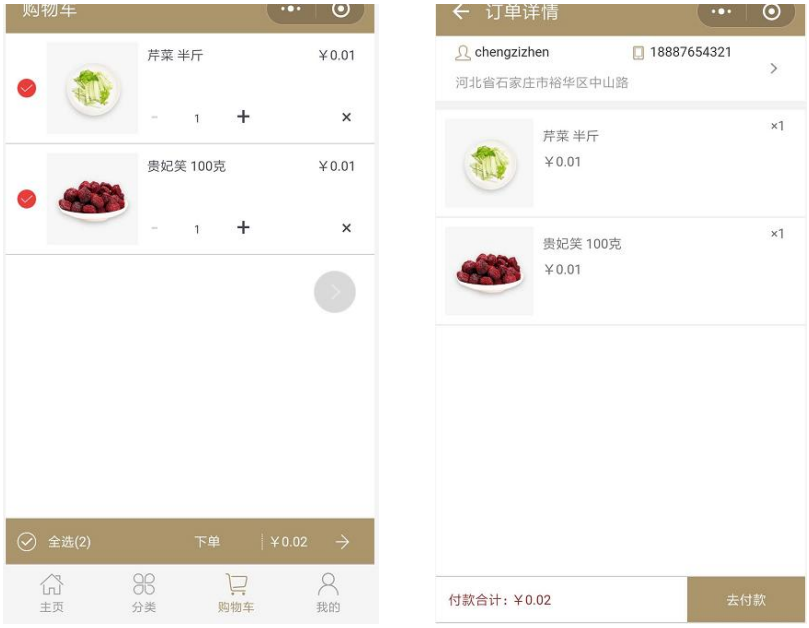

图 4.20　购物车页面及订单请求页面运行效果图

4.4.6　下单支付模块的设计实现

用户在购物车页面可以发起生成订单请求，并在订单详情页编辑订单完成微信支付功能，最后微信返回给用户一个支付结果通知。总体来看，这个过程主要分为三部分：用户下单、微信支付、微信返回客户端支付结果。在完成这个除了要具备微信支付的开发资格外，还需要在实现下单支付的过程中对发起微信支付的小程序用户进行 token 令牌检验，以及数据库的商品库存量检测等。本模块的实现过程主要对小程序客户端令牌验证和下单支付两部分来实现下单支付模块的页面和功能设计。

4.4.6.1　客户端用户 Token 管理机制

对比传统购物网站或者原生 APP 的用户购物过程，一般用户需要在已登录的情况下发起下单、支付或者编辑个人信息的请求。同样的在小程序客户端也需要类似的机制对系统的某些敏感接口做权限控制，在小程序中与传统的网站或原生 APP 不同的是：小程序用户没有注册和登陆的过程，或者说小程序用户没有用户名和密码来标识用户的唯一性。但是小程序有自己特有的用户管理机制，通过微信身份认证体系来实现用户免密登陆，使用 token 令牌来代替传统 web 开发中的 cookie 身份验证和权限控制。其中本系统中设计的小程序用户登陆获取 token 令牌的过程如图 4.21。

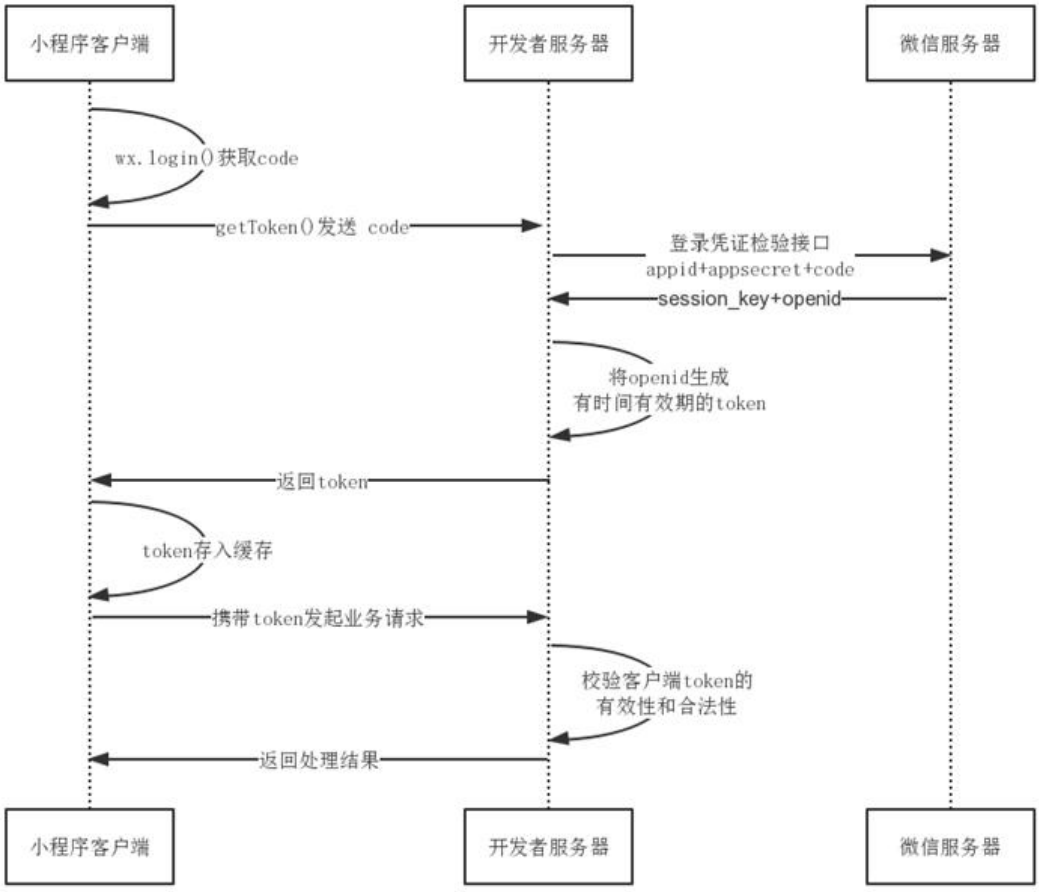

图 4.21　用户获取 token 令牌时序图

由时序图分析可以得出用户登陆小程序获取 token 令牌的过程。用户打开小程序时，微信的 wx.login()方法获会为每一个初次打开小程序的用户生成一个唯一的 code 码，当用户携带 code 码访问服务器接口时，开发者服务器通过 getToken()接口将用户 code 码和小程序信息发送到微信服务器，当微信服务器接收到请求后会返回给开发者服务器 openid 和 session_key 两个参数，此处的 openid 就是用户在本小程序内的用户唯一标识，相当于 user_id。开发者服务器会将 openid 存储到数据库作为用户唯一标识，同时开发者服务器会将处理后 openid 返回给小程序用户。但是由于 openid 作为用户的敏感信息，需要在开发者服务器处理加工成具有时间有效期的 token 令牌返回到小程序客户端缓存中，便于做用户访问服务器接口的身份验证。当用户访问服务器接口前，在开发者服务器处会先对用户的 token 令牌与数据库中的 openid 做有效性和合法性的验证，检验通过后才能返回业务数据。

由于在小程序系统中多个业务接口的数据请求都需要进行用户的 token 验证，如客户端调用用户地址接口、下单接口、支付接口等，此时根据面型对象的开发思想，需要建立一专门的基类来统一的全局性的管理客户端 token 请求。

我们将客户端不同业务的 token 请求封装成了一个客户端 token 管理机制，获取和管理 token 的功能由客户端的 token.js 基类中的 verify()方法实现。在不同的业务请求中，相应业务的 model 类会调用 token 基类携带令牌请求服务器数据。其中 verify()方法的简要代码如下：

```
//检验令牌是否合法或有效
  verify() {
    var token = wx.getStorageSync('token');    //从缓存中读取 token
    if (!token) {
      this.getTokenFromServer();      //调用从服务器获取 token 方法
    }
    else {
      this._veirfyFromServer(token);   //调用携带令牌去服务器校验令牌方法
    }
  }
```

本系统中在客户端设计客户端 token 管理机制基本流程如图 4.22。

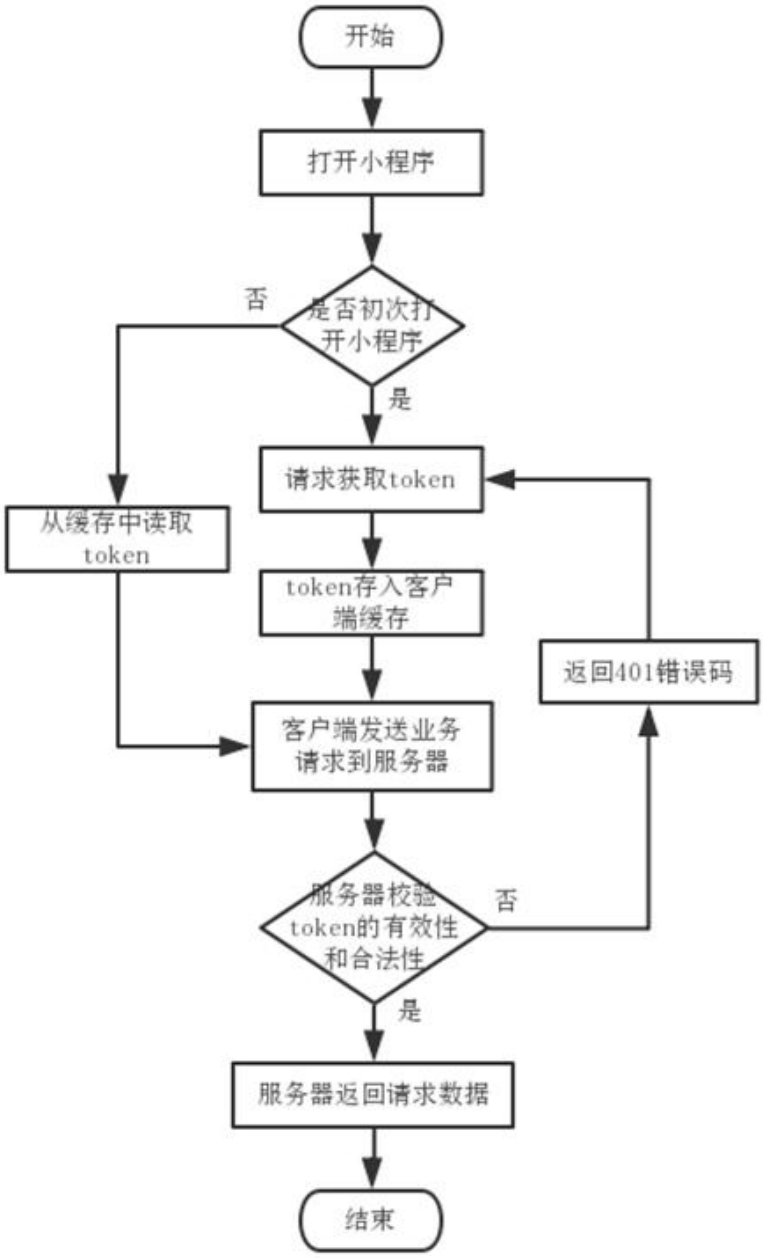

图 4.22　客户端 token 管理机制流程图

4.4.6.2　下单支付模块

当用户在购物车的订单页面和个人信息页面对订单点击付款时，客户端就向开发者服务器发起了生成订单请求，若用户的 token 检测通过则会成功生成订单，用户进入到订单详情页面完成微信支付，否则生成订单失败。下单支付页面主要分为三部分：编辑收货地址、订单详情、微信支付。其中编辑收货地址与个人信息页面的实现方法类似。在本小节主要对生成订单到微信支付功能设计与实现进行详细介绍。

小程序客户端、开发者服务器、微信服务器下单支付的业务逻辑图如图 4.23。

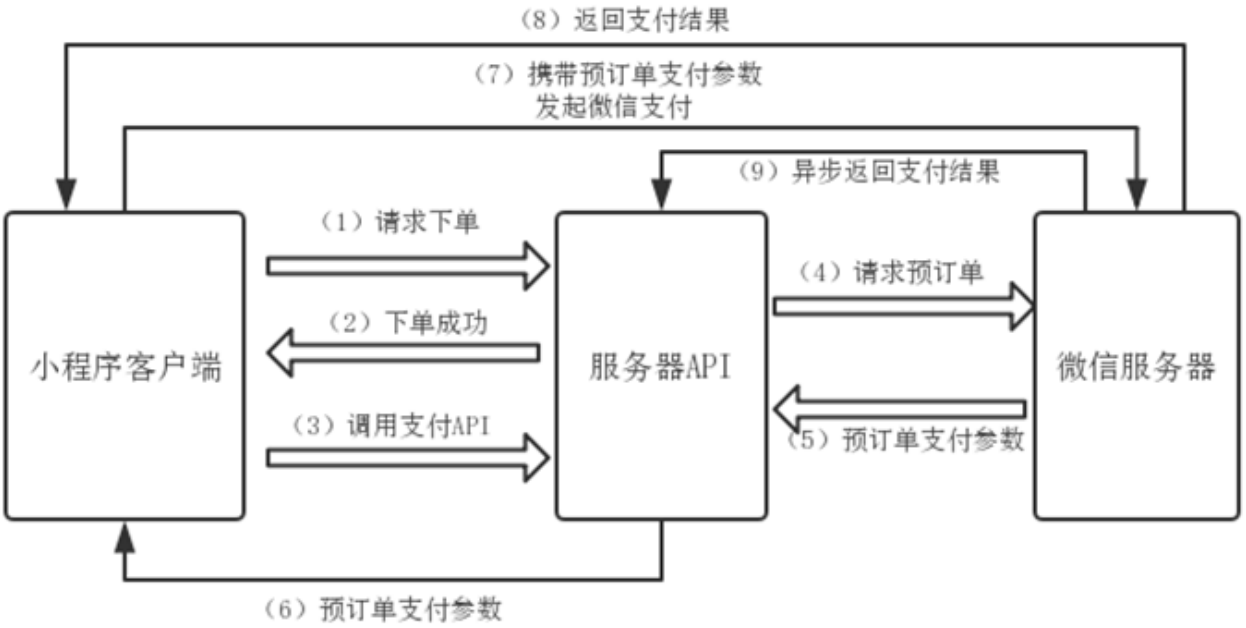

图 4.23　系统下单支付业务逻辑图

在客户端部分，当页面监听到用户的“去付款”操作时，用户就向小程序客户端发起了下单支付的请求，我们在业务代码逻辑处理的时候讲这两种方式的支付行为都在 order.js 控制器中实现。在处理过程上，将支付处理分成第一次支付订单生成和第二次支付订单支付两阶段，这样就可以将对订单进行最后数据库的库存量检测，确保了支付的准确性。客户端部分的主要逻辑代码设计如下：

```
//支付分两步，第一步是生成订单号，然后根据订单号支付
 order.doOrder(orderInfo, (data) => {      //调用下单 API，进行库存量监测
      if (data.pass) {          //订单生成成功
      var id = data.order_id;    //更新订单状态
      that.data.id = id;
      that._execPay(id);   //进行库存量检测通过，调用服务器预订单接口
    } else {
      that._orderFail(data); //  下单失败返回结果  } });
```

Order.js 中的 order.doOrder 方法中用于生成订单获取到订单号，然后在订单状态的判断，如果订单状态通过，则生成订单并更新订单状态；在_execPay()方法中调用模型 order-model.js 中的 execPay 方法进行微信预订单支付参数的请求（这些请求参数与服务中的支付处理参数一致，包括预订单中给的订单签名的信息），微信预订单的由开发者服务器的 pre_order API 发起请求，然后将请求结果返回到客户端模型层，在客户端携带这些预订单参数调用 wx.requestPayment()方法发起微信支付。最后微信将微信支付结果返回到客户端，返回参数的可能值有 0 表示商品缺货等原因导致订单不能支付，1 表示

支付失败或者支付取消，2 表示支付成功。如果返回回调参数非 0，则进行已下单的商品从购物车中删除，并携带返回的参数 wx.navigateTo 到 pay-result 支付结果页面。如果订单没有成功生成，则通过微信的 wx.showModal 窗口返回订单生成失败的原因，一般订单生成失败的原因是订单商品未通过库存量监测。

下单支付的功能模块的运行效果图如图 4.24。

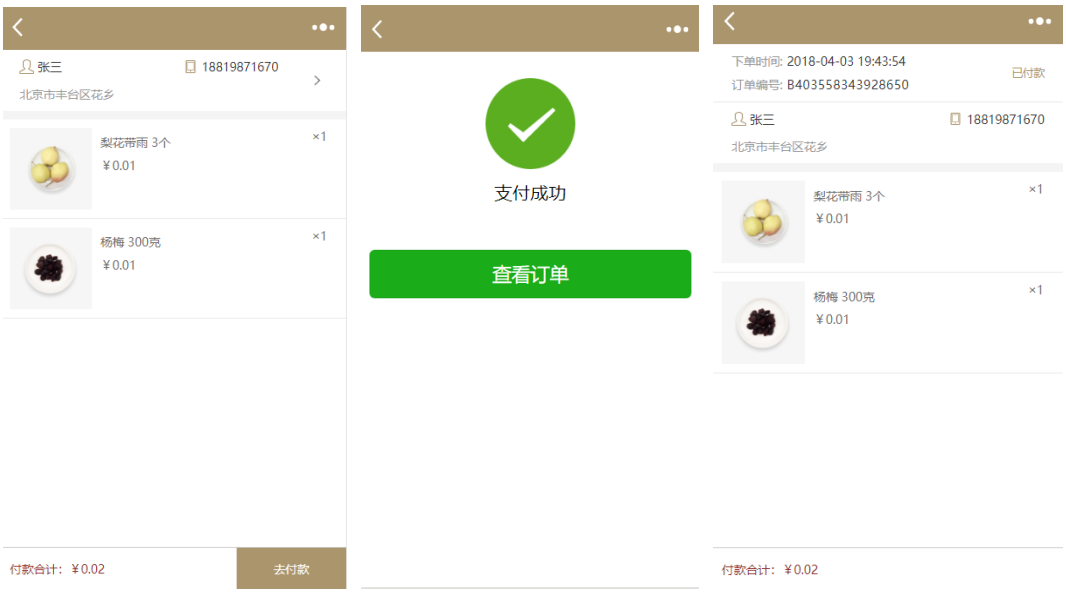

图 4.24　下单支付功能模块运行效果图

4.4.7　个人信息管理模块的设计实现

个人信息管理部分主要包括三部分内容：用户基本信息部分显示用户微信头像、昵称；地址管理部分用于用户的收货地址、电话的管理；我的订单部分用于查看和管理历史订单。这三部分在视图层 my.wxml 文件中由三个 view 模块来定义。

4.4.7.1　用户信息管理

（1）用户基本信息

通过用户登录小程序接受微信授权来获取用户的基本信息。这个过程需要先调用微信的微信登录 wx.login() API 接口来获取用户的临时登陆凭证 code 码，在小程序中 code 码可以作为用户本小程序中的唯一标识。然后再调用微信的获取用户信息的 wx.getUserInfo() API 接口来获取用户的基本信息，此时若用户接受微信登陆时的授权请求，则会在 wx.getUserInfo() API 接口中返回用的基本信息，主要包括：用户昵称、用户头像、用户性别、用户的所在国家、省份、城市以及所用的语言等信息。此时在我的信息页面就会显示用户的微信头像和昵称信息。否则 wx.getUserInfo()接口会执行 fail 方法，在 fail 方法内，购物系统为不接受微信授权的用户准备了默认的用户昵称和头像信息。简要的实现代码如下：

```
wx.login({
  success: function () {
    wx.getUserInfo({
      success: function (res) {
        typeof cb == "function" && cb(res.userInfo); },
      fail: function (res) {
        typeof cb == "function" && cb({
          avatarUrl: '../../imgs/icon/user@default.png',
          nickName: '便利猫'}); } });}, })
```

（2）地址管理

用户收货地址部分可显示用户的收货地址，也可以编辑更改或新增收货地址信息。当打开页面时，可以直接显示用户的当前收货地址，在 my.js 文件通过_getAddressInfo()方法调用 address 基类中的获取服务器中用户收货地址的信息，并以箭头函数回调的方式将 address 基类中的函数返回到当前 getAddressInfo()方法中，再以数据绑定的方式将地址信息显示到页面。用户也可以点击“地址管理”，进行更改或新增收货地址。当视图层监听到用户的地址管理的事件请求时，会调用 my.js 控制器文件中的地址管理 editAddress()事件监听函数，用户会进入到地址管理页面。在地址管理页面用户可以新增地址信息，调用 wx.chooseAddress()微信 API 接口完成新增地址操作。用户也可以选择历史收货地址替换当前的收货地址，此时需要请求 address 基类中的方法向开发者服务器数据完成收货地址数据的获取和保存。调用 address 基类中的主要方法有获取用户地址信息的 getAddress()方法、更新保存地址信息到数据库的 submitAddress()方法和_setUpAddress()方法。其中地址管理 editAddress()事件监听函数的简要代码如下：

```
editAddress: function (event) {
    wx.chooseAddress({       //调用微信 API 接口新增收货地址。
      success: function (res) {
        console.log(res);
        var addressInfo = {
          name: res.userName,
          mobile: res.telNumber,
          totalDetail: address.setAddressInfo(res)}
        that._bindAddressInfo(addressInfo); } })}    //绑定地址到页面
```

地址管理页面的运行效果图如图 4.25。

图 4.25　地址管理页面运行效果图

4.4.7.1　用户订单管理

用户在我的订单部分查看到的都是数据库订单表中的订单快照信息。订单快照信息中记录的是在订单生成时数据库中的商品和订单状态信息，一旦订单生成，随着时间的变化订单快照信息不会变化。简而言之，用户可以在我的订单部分查看历史订单信息。

showOrderDetailInfo()事件监听函数当监听到用户点击历史订单时，会请求服务器数据跳转到订单详情页，对于还未完成支付的订单可以继续支付，此时会调用_execPay()函数进行订单的二次支付，支付流程与下单支付的流程相同，调用 order 类中的 execPay()方法发起微信支付，并在完成支付过程后更新订单状态。_execPay()函数的简要实现代码如下：

```
_execPay: function (id, index) {
    var that = this;
    order.execPay(id, (statusCode) => {     //调用 order 类中的 execPay()方法调起微信支付并通过箭头回调函数返回支付结果。
        if (statusCode > 0) {
            var flag = statusCode == 2;
            //更新订单显示状态
            if (flag) {
                that.data.orderArr[index].status = 2;
```

```
            that.setData({
                orderArr: that.data.orderArr      });          }
            //跳转到 成功页面
            wx.navigateTo({
                url: '../pay-result/pay-result?id=' + id + '&flag=' + flag + '&from=my'
            });          }
        else {
            that.showTips('支付失败', '商品已下架或库存不足');          } }); }
```

系统的个人信息管理功能模块的页面运行效果图如图 4.26。

图 4.26　个人信息管理页面运行效果图

4.5　本章小结

本章首先介绍了项目的系统总体架构，系统开发中客户端和服务器所选用的开发方法以及开发环境的部署。其次完成了购物系统的 MySQL 数据库的设计与实现，介绍了客户端与服务器数据交互的原理，以及小程序客户端的 MVC 模式设计控制，最后完成了微信小程序购物系统客户端五大模块的详细设计和实现，分别为用户浏览商品模块、分类检索模块、购物车模块、下单支付模块、个人信息管理模块。重点介绍了在设计和实现过程中所用到的 MVC 模式、template 模板设计和 Token 管理机制关键技术方法与思想。

第 5 章　系统测试

软件测试是为了发现程序中可能潜在的错误而执行程序的过程，本章对微信小程序网上购物系统的主要用例进行了功能和性能测试，通过对用例的测试确保系统的可靠性。

5.1　测试方案

对本网上购物系统的测试按照以下方案进行：

1、在开发过程中选用 postman 接口测试工具，postman 是一个 API 接口测试工具，可以通过发送 get、post 等 http 请求进行客户端和服务器的 API 接口通信。

2、在开发完成后，结合微信开发者工具对小程序购物系统进行全面的验收测试。分别在客户端对各个功能模块进行功能测试，结合小程序页面展示和数据库数据的变更情况进行测评。利用微信开发者工具对小程序进行云真机性能测试，通过对测试数据的分析评估购物系统的运行性能。

本文功能测试用例采用的黑盒测试方法，根据需求对系统各个页面和功能设计测试用例测试。性能测试采用的是微信开发者工具提供的测试工具，通过测试报告数据对小程序系统进行性能分析。

5.2　功能测试

对系统的主要页面和功能点的测试用例如下：

（1） 用户登陆小程序测试用例如表 5.1 所示。

表 5.1　用户登陆小程序测试用例

用例名称	用户登陆小程序
目的	测试用户打开小程序时微信登陆和微信授权的功能。
前提	用户初次打开小程序。
测试流程	1)　获取小程序，进入“我的”页面。 2)　弹出微信授权获取公开信息页面，选择允许或拒绝。
预期结果	当用户选择允许时，个人信息显示用户的微信头像和昵称；当用户拒绝时，显示系统默认的头像和昵称。
实际结果	实际结果与预期结果一致。

（2）商品分类测试用例如表 5.2 所示。

表 5.2　商品分类测试用例

用例名称	商品分类
目的	测试商品分类模块功能。
前提	无
测试流程	1）用户点击分类类目。 2）用户点击分类商品概要信息。
预期结果	页面跳转到对应分类页面，点击商品跳转到商品详情页。
实际结果	实际结果与预期结果一致。

（3）购物车测试用例如表 5.3 所示。

表 5.3　购物车测试用例

用例名称	购物车
目的	测试购物车模块功能。
前提	无
测试流程	1）用户在商品详情页选择商品数量点击加入到购物车。。 2）用户在购物车点击更改商品数量按钮“+”、“-”。
预期结果	若商品库存量不足，商品显示无货，无法加入到购物车；如商品库存量充足，商品加入到购物车，且购物车更新商品数量，用户可以在[1,库存量]范围内更改商品数量。
实际结果	实际结果与预期结果一致。

（4）下单支付测试用例一如表 5.4 所示。

表 5.4　下单支付测试用例

用例名称	下单支付测试用例一
目的	测试下单支付功能
前提	用户 token 合法并有效，全程商品库存量充足
测试流程	1）用户在购物车页面点击“下单”。 2）用户在订单详情页点击“去付款”。
预期结果	跳转到生成订单详情页，付款成功后，跳出支付成功通知页面，并可查看订单详情；取消付款，跳出支付失败页面，并可查看订单详情，购物车删除已生成订单商品，数据库更新订单状态。
实际结果	实际结果与预期结果一致。

（5）下单支付测试用例二如表 5.5 所示。

表 5.5　下单支付测试用例

用例名称	下单支付测试用例二
目的	测试下单支付功能
前提	用户 token 合法并有效，商品库存量在添加购物车时充足，生成订单时不足
测试流程	1）用户在购物车页面点击“下单”。 2）用户在订单详情页点击“去付款”。
预期结果	跳转到生成订单详情页，点击“去付款”，跳出“下单失败，商品缺货”提示框。
实际结果	实际结果与预期结果一致。

（6）下单支付测试用例三如表 5.6 所示。

表 5.6　下单支付测试用例

用例名称	下单支付测试用例三
目的	测试下单支付功能
前提	用户 token 合法并有效，商品库存量在生成订单时充足，付款时不足
测试流程	1）用户在购物车页面点击“下单”。 2）用户在订单详情页“去付款”，然后点击取消付款。 3）用户在我的订单处，对未支付订单继续支付。
预期结果	点击取消付款，跳转到支付失败页面，并生成订单；在我的订单页面对订单再次支付时，跳出“下单失败，商品缺货”提示框。
实际结果	实际结果与预期结果一致。

（7）用户地址管理测试用例如表 5.7 所示。

表 5.7　用户地址管理测试用例

用例名称	下单支付测试用例四
目的	测试地址管理功能
前提	用户 token 合法并有效
测试流程	1）在我的页面点击“地址管理”。 2）点击新增地址。
预期结果	跳转到地址管理页面，可以接受微信授权获取当前卫星定位信息；点击新增地址可按照规范新增收货信息。地址编辑完毕后保存，我的页面刷新显示的当前收货地址信息。
实际结果	实际结果与预期结果一致。

5.3 性能测试

小程序购物系统的性能测试采用的是微信开发者工具提供的开发阶段的小程序测试工具，小程序的测试机器由微信云平台随机分配的云真机完成，云真机是由 WeTest 腾讯质量开发平台部署和维护的真实机器。在微信开发者工具中可每 24 小时申请生成一次关于小程序的 Android 机型测试报告。测试报告主要包括测试机型、小程序错误总数、首屏加载时间、平均 CPU 占比等信息。本文将 20 个 Android 机型测试结果的信息汇总分析如表 5.8 所示。

表 5.8 系统性能测试

测试机型	安卓版本	分辨率	微信版本	错误页面数	首屏平均加载时间（秒）	非首屏平均加载时间（秒）	平均 CPU 占比	平均内存占用（MB）
MI 4LTE	V6.0	1080×1920	6.6.5.1280	0	2.8	0.17	15.52%	113.4
Vivo V3MA	V5.1	720×1280	6.6.5.1280	0	4.61	0.22	14.19%	138.86
R7Plusm	V5.1	1080×1920	6.6.5.1280	0	4.89	0.19	6.31%	126.49
GN9010	V5.1	720×1280	6.6.5.1280	0	5.48	0.27	4.77%	131.55
vivo Y67A	v6.0	720×1280	6.6.5.1280	0	2.15	0.18	14.37%	124.66
NEM_AL10	V6.0	1080×1920	6.6.5.1280	0	4.7	0.16	6.13%	144.83
KIW-AL10	V6.0	1080×1920	6.6.5.1280	0	5.58	0.23	8.37%	143.24
TA-1000	V7.0	1080×1920	6.6.5.1280	0	2.74	0.18	3.16%	137.36
vivo Y31A	V5.1	540×960	6.6.5.1280	0	6.4	0.26	10.40%	120.76
OPPO R9s	V6.0	1080×1920	6.6.5.1280	0	3.17	0.17	9.57%	106.57
M3	V5.1	720×1280	6.6.5.1280	0	5.23	0.28	11.09%	106.47
Vivo Y55A	V6.0	720×1280	6.6.5.1280	0	3.57	0.19	2.13%	102.55
vivo X7	V5.1	1080×1920	6.6.5.1280	0	1.72	0.11	3.85%	99.22
OPPO A53	V5.1	720×1280	6.6.7.1300	0	4.03	0.27	0.57%	121.25
HUAWEI TAG-TL00	V5.1	720×1280	6.6.7.1300	0	4.09	0.2	0.85%	130.62
vivo X7	V5.1	1080×1920	6.6.7.1300	0	2.38	0.18	2.07%	108.44
AM-AL00	V6.0	720×1280	6.6.7.1300	0	5.77	0.2	6.73%	123.56
SM-G9280	V6.0	1440×2560	6.6.7.1300	0	3.15	0.2	9.12%	158.1
vivo V3Max A	V5.1	1080×1920	6.6.7.1300	0	5.88	0.23	3.27%	133.61
平均值				0	4.12	0.20	6.97%	124.82

由以上的小程序测试报告数据的分析可得，本小程序系统在云真机测试系统中未出现程序异常，在适配不同测试机型、安卓版本、像素分辨率、微信版本的测试环境中均运行正常。其中，系统整体的首屏页面平均加载时间为 4.12 秒，非首屏页面的平均加载

时间为 0.2 秒，平均 CPU 占比为 6.97%，平均内存占用为 124.82。小程序的平均首次加载时间、平均 CPU 占比、平均内存占用在不同型号的云真机上行性测试略有不同，这与测试机的内存大小、CPU 处理器的性能、网络状态等因素有关，但当测试机获取到小程序进行首屏加载后，其非首屏的加载时间保持在 0.2 秒左右，可见在小程序的使用过程中交互反应性能良好。整体来看，小程序购物系统的总体性能表现良好。

通过对微信小程序购物系统的各个模块进行测试，用户在进行商品的浏览、分类商品的检索和商品详情的查看，在商品详情页将商品加入到购物车，可以对购物车的商品和地址信息进行编辑后下单支付，根据支付结果数据库中的商品和订单信息产生相应改变。测试证明，本小程序购物客户端系统可以满足用户的购物的功能和性能需求。

5.4　本章小结

本章首先介绍了系统的测试方案和测试工具，然后对系统的主要功能模块进行了详细的测试用例分析，并结合微信开发者工具对小程序的云真机测试报告进行了性能分析。测试表明本系统的各个模块的实现达到了设计要求。

第 6 章　总结与展望

6.1　总结

本文通过分析当下传统中小型商店在新零售下面临的机遇和挑战，完成了基于微信小程序的网上购物系统。在项目的设计与实现过程中，按照面向对象的软件开发思想，完成了系统从需求分析、架构搭建与设计、编码实现购物客户端系统到完成系统的功能和性能测试，在这个过程中我不仅学习到了软件开发的流程，同时对 WXML、WXSS、微信小程序等的基础知识有了更加深入的了解。在开发过程中我的购物系统客户端开发能力和服务器端整体知识应用能力得到了进一步的提高，更加进一步的了解了 MVC 设计模式在小程序和 TP5 中的应用，小程序 template 模板的优势以及 MySQL 数据库的部署构建和使用，从全局角度把握每一个系统和模块在项目中的功能和作用。

作为当前生活中关注热点的微信小程序，建立微信小程序购物系统能够充分体现自由便捷购物的新零售思想。即用即走的微信小程序与购物系统的结合，对于线下商店的线上推广工作有很大助力作用。

在完成论文的过程中，我更加明白了需求分析和全局性规划开发的重要性。对于一个项目而言，在进行开发工作之前，需要对需求进行充分的分析，根据业务需求和对象定义代码的业务逻辑，并对多个模块的功能实现进行全局性的控制，减少重复和低效的编程工作。此外，数据库的概念设计和逻辑设计对于项目的开发也很重要，服务器和客户端的功能实现都需要数据库的支持。

在这次完成毕业设计和论文的过程中，遇到了很多困难，但克服困难的过程也是一次自我提高的机会，实践使我对理论知识有了更深的理解，也提高了解决问题的能力。

6.2　展望

系统也有不足和需要继续改善的地方，需要在接下来的学习中进一步的完善：

（1）对系统功能设计上还可以进一步拓展优化，如物流跟踪等。

（2）系统还可以进一步开发微信小程序的功能，如短信消息通知等。

（3）系统在订单支付方面还可以增加多种支付方式，如支付宝等。

参考文献

[1] 雷磊. 微信小程序开发入门与实践[M]. 北京: 清华大学出版社, 2017

[2] 张翔. 微信小程序：分享微信创业 2.0 时代千亿红利[M]. 北京：清华大学出版社, 2017

[3] 方蓓工作室. 微信公众平台开发最佳实践[M]. 北京：机械工业出版社, 2015.5

[4] 李炜康. 基于微信的网上购物系统的设计与实现[D]. 南京大学, 2015

[5] 谭楷祥. 基于微信公众平台的点餐系统的设计与实现[D]. 南京大学, 2014

[6] 潘凯华,刘中华等编著.PHP 从入门到精通[M]. 清华大学出版社, 2010

[7] Luke Welling,Laura Thomson. PHP 和 MySQLWeb 开发（原书第 4 版）[M]. 北京：机械工业出版社，2009.4

[8] 储广昕. 分析软件开发中数据库设计理论的实践[J]. 信息通信, 2015, 149(5):127-128

[9] 潘凯华,刘中华,杨明,编著.PHP 开发实战 1200 例[M]. 清华大学出版社, 2010

[10] 李尊朝, 苏军等. Web Database Applications with PHP&MySQL [M]. 出版社:中国电力出版社, 2007.23-25

[11] 萨师煊, 王珊等. 数据库系统概论（第三版）[M]. 北京:高等教育出版社, 2005.45-187

[12] 张立科. Mysql 数据库通用模块及典型系统开发实力导航[M]. 北京:人民邮电出版社,2006.10-295

[13] 沈炜, 徐慧,汤倩. Mysql 数据库编程技术与实例[M]. 北京:人民邮电出版社, 2005. 114-226

[14] 冯晓强,程晓昕. 基于 MVC 模式的网上购物系统的设计与实现[J]. 现代计算机:专业版. 2009(7):177-180

[15] 孔秀丽. 基于微信公众平台的大学生移动商城的构建[J]. 科技创业月刊，2015(3):25-26

[16] 于秀山, 于洪敏,著. 软件测试新技术与实践[M]. 电子工业出版社, 2006

[17] 徐兰芳, 彭冰, 编. 数据库设计与实现[M]. 上海交通大学出版社, 2006

[18] 徐其帅. 面向对象软件开发方法的实例分析[D]. 浙江大学, 2008

[19] 孟祥磊. 微信公众平台开发实例教程[M]. 人民邮电出版社, 2017

[20] 李华明. 基于 PHP 和 MySQL 的网上购物系统设计与实现[D]. 电子科技大学, 2014

[21] 李蓉. 基于 MVC 模式的 WEB 应用研究[J]. 计算机应用与软件，2015(10):12-13

[22] 赵敬, 李贝. 微信公众平台发展现状初探, 新闻实践, 2013(8):8-10

[23] 金莹. 基于微信小程序的个人电台系统的设计与实现[D]. 吉林大学, 2017

[24] 朱勇贞. 基于微信的电商平台的设计与实现[D]. 东华大学, 2017

[25] 王勇锋. 基于微信公众平台的医药零售模式的研究和实现[D]. 华南理工大学, 2017

[26] 王权. 基于微信开放平台购物中心的设计与实现[D]. 吉林大学, 2016

[27] 王昭英. 基于 MVC 设计模式的 thinkphp 框架的研究与应用[D]. 西安建筑科技大学, 2010

[28] 杜江. PHP 与 MySQL 高性能应用开发[M]. 机械工业出版社, 2016

[29] 谢晓萍. 微信力量[M]. 机械工业出版社, 2015

致谢

光阴似箭，研究生阶段的珍贵学习时光就要结束了。两年的研究生时光弥足珍惜，我不仅收获了专业的理论知识和技术基础，还有老师们和朋友们的珍贵情义。在此表示发自内心的感谢！

感谢师恩！感谢我的研究生导师牛东来教授和毕业设计校外导师任茂松教授，是您们平时对我学习的专业教导引我进入了学术的殿堂。首先，感谢他们在百忙之中对我学习和项目中遇到的困难进行悉心的指导，在我毕业设计和论文撰写过程中的细心指导和帮助。其次，更要感谢他们不仅传道受业、答疑解惑，更是亦师亦友、如父如母，不仅教会了我实践经验和做事的方法，更是启迪了我批判与质疑的精神。能够得到恩师的谆谆教导是我此生的福气和荣幸。

感谢硕士求学生涯里所有传授我知识、开拓我视野的授课老师们！他们不仅教会了我专业知识与技能，还传授给了我探索与创新的勇气。让我学习了软件工程的系统的理论知识和专业技术知识，还为我毕业论文项目完成和未来职业发展打下了坚实的基础。

感谢母校！感谢首都经济贸易大学给我提供了优秀的学习氛围和求知环境。感谢软件工程班这个温暖、积极的大集体，在两年的生活中和奋进向上的同学们一起学习，一起进步，在遇到困难的时候能集思广益，共同解决学习和生活中遇到的困难，在我迷茫无措时有他们的指路，在我困顿失意的时候有他们的鼓励，在我欢喜幸福的时候有他们的陪伴。能在首都经济贸易大学软件工程专业学习两年的硕士生活，是我一生珍贵的财富，以后不管我在哪里，首都经济贸易大学永远是我最思念的地方。

最后我要感谢我最敬爱的父母和家人们，感谢他们对我的生活无微不至的关怀和鼓励，是他们的爱和支持鼓舞着我不断向前，他们是我在遇到困难时的强大精神支柱，祝福他们永远幸福健康。

首都经济贸易大学
Capital University of Economics and Business

专业硕士学位论文