中国科学技术大学

专业硕士学位论文

（专业学位类型）



University of Science and Technology of China

**基于Restful架构的供应链电子发票系统的设计与实现**

作者姓名： 杨堃

专业领域： 软件工程

校内导师： 李春杰教授

企业导师： 谢弘高级工程师

完成时间： 2017年07月09日

A dissertation for master’s degree

（Professional degree type）



**Design and implementation of electronic Invoicing system for supply chain based on Restul architecture**

Author：

Speciality： Software Engineering

Supervisor：

Advisor：

Finished time:

中国科学技术大学学位论文原创性声明

本人声明所呈交的学位论文,是本人在导师指导下进行研究工作所取得的成果。除已特别加以标注和致谢的地方外，论文中不包含任何他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的贡献均已在论文中作了明确的说明。

作者签名：\_\_\_\_\_\_\_\_\_\_\_ 签字日期：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

中国科学技术大学学位论文授权使用声明

作为申请学位的条件之一，学位论文著作权拥有者授权中国科学技术大学拥有学位论文的部分使用权，即：学校有权按有关规定向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅，可以将学位论文编入《中国学位论文全文数据库》等有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。本人提交的电子文档的内容和纸质论文的内容相一致。

保密的学位论文在解密后也遵守此规定。

□公开 □保密（\_\_\_\_年）

作者签名：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 签字日期：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 摘 要

进入二十一世纪以来，全球经济一体化体系越来越成熟，各个企业都逐渐拥有了稳定的产品供应商群体。随着采购收货开票业务量的迅速增长，传统的企业管理系统已经渐渐不能胜任处理这些数据关联性极强的业务需求。 以传统的处理方式，通常会通过人工的方式对采购-收货-支付流程中产生的收货单、发票进行数据核对。这样的方式在处理大量发票信息的时候，不免会增加人力、时力上的消耗。因此，企业对于电子发票开票前数据协商和自动化匹配的需求也将显著上升。

本系统是基于具有 Restful 架构的Jersey平台开发，采用 B/S 系统架构，在企业用户接入网络的情况下，用户可以随时登录系统网页，并方便的完成电子发票相关的操作。使用本系统，能够及时就电子发票的各个字段、数据与供应商沟通修改，而且还可以自动以收货单为标准，核对电子发票中每件商品在价格和数量上是否正确，极大的节约了企业用户在人力和时力上的损耗。弥补了目前电子发票领域中在这一块的功能缺失。

本文着重介绍一套基于Restful 架构的供应链电子发票系统的设计与实现，该系统将基于B/S结构，采用 Java和Postgres+Elasticsearch编程来实现对文档数据的存储和操作。在后端开发模型中，遵照 MVC 模式进行开发。网络数据交互则遵照 HTTPS协议来实现与服务器端的通信。在后台开发环境与运行环境上选用 Postgres+Elasticsearch作为后台数据库服务器，采用 Jetty作为服务器端容器，最终完成具有基础功能的电子发票系统，满足用户在开票前的业务需求。

**关键词**：电子发票 Universal Business Language Restful Jersey

# ABSTRACT

Since twenty-first Century, the global economic integration system has become more and more mature, and each enterprise has gradually stable product supplier groups. With the rapid growth of the amount of invoicing and invoicing, the traditional enterprise management system has been gradually unable to cope with the strong business needs of these data. In a traditional way, data is usually manually checked for receipts and invoices generated in the purchase receipt payment process. This way when dealing with a large number of invoice information, will inevitably increase the consumption of manpower and time. As a result, the demand for data consultation and automated matching of electronic invoices before Invoicing will increase significantly.

This system is developed with Jersey platform based on Restful architecture, using B/S architecture, in the enterprise user access network, users can log on to the system ", and facilitate the completion of electronic invoices related operations. Using this system, each data field, timely communication with suppliers of electronic invoice modification, but also can automatically with the receipt for the standard, check the electronic invoice in each item is in the price and quantity on the right, the loss greatly saves manpower and time business users in the force. Make up for the current lack of functionality in this area of electronic invoicing.

This paper introduces a design and implementation of Supply Chain Based on electronic Invoicing system Restful architecture, the system will be based on the B/S structure, using Java、Postgres and Elasticsearch programming to achieve the storage and operation of the document data. In the backend development model, development is carried out in accordance with the MVC schema. Network data interaction follows the HTTPS protocol to communicate with the server. In the background of development environment using Postgres and Elasticsearch as the back-end database server, using Jetty as the server container, the final completion of electronic invoice system has the basic functions, meet the users in the ballot before the business needs.

**Key Words**: Electronic Invoice, Universal Business Language, Restful, Jersey

**目 录**

[摘 要 I](#_Toc488507165)

[ABSTRACT II](#_Toc488507166)

[第1章 绪论 1](#_Toc488507167)

[1.1 系统研究背景 1](#_Toc488507168)

[1.2 系统研究意义 2](#_Toc488507169)

[1.3 国内外研究现状 2](#_Toc488507170)

[1.3.1 国外研究现状 3](#_Toc488507171)

[1.3.2 国内研究现状 3](#_Toc488507172)

[1.4 本文主要研究内容和本人工作 4](#_Toc488507173)

[1.4.1 本文主要研究内容 4](#_Toc488507174)

[1.4.2 本人工作 5](#_Toc488507175)

[1.5 本文的组织结构 5](#_Toc488507176)

[1.6 本章小结 6](#_Toc488507177)

[第2章 相关技术简介 7](#_Toc488507178)

[2.1 编程语言介绍 7](#_Toc488507179)

[2.2 数据格式JSON介绍 7](#_Toc488507180)

[2.3 数据格式XML介绍 8](#_Toc488507181)

[2.4 设计模式MVC介绍 9](#_Toc488507182)

[2.5 本章小结 10](#_Toc488507183)

[第3章 电子发票系统需求分析 11](#_Toc488507184)

[3.1 业务需求 11](#_Toc488507185)

[3.2 功能性需求 12](#_Toc488507186)

[3.2.1 总体需求 13](#_Toc488507187)

[3.2.2 对账单模块需求 14](#_Toc488507188)

[3.2.3 对帐单行详情模块需求 15](#_Toc488507189)

[3.2.4 模拟发票模块需求 16](#_Toc488507190)

[3.2.5 预制发票模块需求 17](#_Toc488507191)

[3.2.6 消息转发模块需求 18](#_Toc488507192)

[3.3 非功能性需求 19](#_Toc488507193)

[3.3.1 健壮性 19](#_Toc488507194)

[3.3.2 可维护性 19](#_Toc488507195)

[3.3.3 可靠性 19](#_Toc488507196)

[3.4 本章小结 20](#_Toc488507197)

[第4章 个性印刷系统概要设计 21](#_Toc488507198)

[4.1 系统整体结构 21](#_Toc488507199)

[4.2 模块设计 22](#_Toc488507200)

[4.2.1 总体处理流程 22](#_Toc488507201)

[4.2.2 注册模块的设计 23](#_Toc488507202)

[4.2.3 登录模块的设计 24](#_Toc488507203)

[4.2.4 分类模块的设计 24](#_Toc488507204)

[4.2.5 在线制作模块的设计 26](#_Toc488507205)

[4.2.6 订单管理模块的设计 29](#_Toc488507206)

[4.2.7 购物车模块的设计 29](#_Toc488507207)

[4.2.8 个人信息模块的设计 30](#_Toc488507208)

[4.3 非功能设计 31](#_Toc488507209)

[4.4 数据库简要设计 32](#_Toc488507210)

[4.5 本章小结 36](#_Toc488507211)

[第5章 个性印刷系统的详细设计与实现 37](#_Toc488507212)

[5.1 系统整体设计与实现 37](#_Toc488507213)

[5.2 注册模块的设计与实现 38](#_Toc488507214)

[5.3 登录模块的设计与实现 41](#_Toc488507215)

[5.4 分类模块的设计与实现 43](#_Toc488507216)

[5.4.1 分类页面的设计与实现 43](#_Toc488507217)

[5.4.2 模板详情页面的设计与实现 46](#_Toc488507218)

[5.5 在线制作模块的设计与实现 48](#_Toc488507219)

[5.6 订单模块设计与实现 52](#_Toc488507220)

[5.7 购物车模块设计与实现 53](#_Toc488507221)

[5.7.1 购物车页面的设计与实现 53](#_Toc488507222)

[5.7.2 支付页面的设计与实现 56](#_Toc488507223)

[5.8 个人信息模块设计与实现 56](#_Toc488507224)

[5.9 本章小结 59](#_Toc488507225)

[第6章 个性印刷系统测试 60](#_Toc488507226)

[6.1 系统实现环境 60](#_Toc488507227)

[6.2 系统测试 60](#_Toc488507228)

[6.2.1 系统测试概述 60](#_Toc488507229)

[6.2.2 系统测试环境 61](#_Toc488507230)

[6.2.3 系统测试内容 61](#_Toc488507231)

[6.2.4 测试结果分析 68](#_Toc488507232)

[6.3 本章小结 68](#_Toc488507233)

[第7章 总结与展望 69](#_Toc488507234)

[7.1 总结 69](#_Toc488507235)

[7.2 展望 69](#_Toc488507236)

[参考文献 82](#_Toc488507237)

[致 谢 84](#_Toc488507238)

# 绪论

随着国家经济建设的稳步发展，国内各企业的供需网络在企业业务产线多样化、复杂化的过程中逐渐日益庞大。在每一次的供需交易过程中，交易所产生的票据、供需数量、供需类型等数据也随之大量增长。令许多企业的财务和供需管理在人力物力上都面临着巨大的挑战。对于拥有大量供需交易活动的企业，如何合理高效的管理本企业的供需交易流程，如何便捷准确的处理交易流程中的各类数据已经成为了企业提高业务产线生产力，减少人力物力成本开销的关键因素。通过引入电子发票管理系统，结合企业自身的管理系统，设计并开发专业高效的电子发票处理流程，以此实现企业快速准确管理供需交易过程中的各类交易数据，有效提升了企业生产力和交易管理组织能力。

## 1.1 系统研究背景

发票是单位和个人所开具和收取的业务凭证，也是审计机关、税务机关执法检查的重要依据。电子发票是指在购销商品、提供或者接受服务以及从事其他经营活动中，使用税务机关确定的标准开票软件开具、按照主管部门要求格式和方式存储和使用的电子收付款凭证。

随着经济多样化的快速发展，国内各企业的供需交易也越发复杂和频繁。在每一次的供需采购交易过程中，常常伴随着大量发票数据的产生。很多企业把更多的资源和注意力放在如何促进企业生产力的提升方面，但会忽视对交易过程中票据的有效管理和分析。随着供需交易的不断发生，发票等交易数据会迅速积压在企业财务和供需管理部门的工作量中，如果没有一种合适的方法或系统来处理这些采购供应数据，则会严重影响企业对运营资金的管理和跟踪。

通过调研可知，在国内，虽然有电子发票这一概念，但在电子发票的推广和使用上上还与国外有一定的差距。一方面，因为我国当前税收体制与其它国家有一定的区别，另一方面，电子发票系统在国内还没有形成一个完整稳定的生态系统。对于不同企业针对纸质发票的不同需求，在电子发票的设计与开票过程中，都有着不同的规范要求。因此，如何根据不同企业用户之间的规范要求来进一步约束保证纸质开票流程的准确无误是开发电子发票系统的一个关键点。这也是本文所需要解决的问题。

设计和开发电子发票系统意在有效管理企业的每一次采购供应过程。将交易过程中的各类商业文档电子化、虚拟化。通过合理的电子发票创建、修改、导出流程来有效的监控每一笔供需过程。整个系统由浏览器客户端、服务器端和数据持久层三部分组成。笔者主要从事服务器端和数据持久层这两部分的设计和开发工作。使用具有Restful架构的Jersey框架来设计并开发电子发票系统的各类逻辑业务，从而满足企业用户的需要，解决当前企业面临的供需数据繁杂、冗余、不易清晰管理的问题。

## 1.2 系统研究意义

电子发票是我国商务体系信息化网络化建设中的关键一环。电子发票平台是商务现代化建设的一项重大的基础建设工作。因此，设计开发并普及一个良好的电子发票处理系统对于推进我国电子商务现代化建设有重大的促进意义。基于Restful架构的电子发票系统在推广使用后能够有效的推进电子商务现代化建设的步伐。

对于环境，推广该系统在进一步普及电子发票的同时，也能够大大的促进环保、绿色、可持续发展的社会环境的建立；与传统纸质发票相比，电子发票具有“无纸化、网络化、自动化”的特点，能够有效推动电子商务绿色、环保、可持续发展。

据不完全统计，伴随着中国电子商务的高速增长，中国电商规模已经增长到30万亿，采用电子发票，相较于传统纸质发票可以节约发票开具方和接收方50%的成本。根据调研统计分析，在2017年，国内增值税电子普票的开票量将达到552亿张。

对于企业，积极结合电子发票系统与企业自身管理系统，能够提升企业财务和采购供应管理部门的工作效率。该系统在设计中将发票开票流程中，将企业用户对发票格式、字段的规范要求放在首要的考虑位置，通过及时的信息传递以及可修改的电子发票版式进一步加强了在纸质发票开票流程前，企业用户之间的沟通协商能力，保证了最终纸质发票开具的准确无误。针对每一笔交易过程，通过上传收货单、勾选收货详情行开票、使用平台消息系统协商调整票面数据、根据开票上限合理拆分电子发票等流程能够及时准确的监控每一笔供需交易的数据状态。减轻相关工作人员的工作压力，加快企业运营资金的管理和跟踪流程。降低在交易过程中的人力财力成本的开销。

## 1.3 国内外研究现状

在我国当前的税收体制下，不同于欧美等发达国家的税收政策，在发票类型、发票职能、发票字段方面的独特要求都要做特殊考虑。因此在设计和开发电子发票系统的过程中，需要随时结合我国税收政策。在国外，具有领导能力的电子商务解决方案提供商是SAP公司，其在电子发票系统方面的设计具有值得思考和借鉴的地方。在国内，诸如金蝶、百望、航信等公司，都在电子发票方面有自己的设计和考虑。下面主要针对这些公司的电子发票系统进行简略的分析和比较。

### 1.3.1 国外研究现状

在电子商务行业内，SAP 公司是全球最大的企业管理和协同化电子商务解决方案供应商。目前，SAP在120多个国家和地区拥有18,800多家客户、56,000多个系统安装点以及1000万多名最终用户，世界500强中80%以上的公司都在使用SAP的管理解决方案。因此，调研国外电子发票发展状况以 SAP 公司为主具有一定的代表性和全面性。

经深入调研分析，国外的SAP思爱普公司针对发票管理解决方案开发了一款产品，该产品主要由采购订单协同、发票协同、发票工作流、审批管理等几个功能组成。企业通过该产品可以有效控制开票流程，降低成本。通过电子单据集成到发票流程中，可以提高发票处理速度和准确性，并确保发票准时送达指定目的地。功能上相对来说是比较完善的，能够提供丰富的发票管理方面的功能（自动验证、匹配发票与采购单和收货单），最大程度的满足企业用户的工作需求。

但是针对中国企业的业务需求， SAP 目前的产品也存在一定的不足：1. SAP 的产品还没有着重考虑到开票前企业与供应商很可能会就发票信息进行频繁协商的情况。2. SAP 生态系统的互相关联导致其在第三方平台的移植、兼容性方面表现不友好。3. 基于中国发票特有的限额拆分特点，SAP 尚未针对这一特点提出有效的解决方案和相关产品。4. SAP 尚未设计出基于中国发票的电子发票格式。5. 该产品基于 C/S 架构，在部署、跨区域方面灵活性较差。

对比国外主流电子商务公司 SAP 的产品，本文所提出的供应链电子发票系统针对以上不足，设计出符合中国发票字段要求的电子发票样式，并提出了可行的解决方案。不仅在设计中考虑到企业与供应商就电子发票信息协作沟通的业务需求，而且基于 B/S 架构能够做到及时快速部署、友好兼容企业与供应商自带的 ERP 系统。此外，针对中国发票特有的限额拆分特点，在模拟电子发票模块实现了详细的拆分法则。

### 1.3.2 国内研究现状

目前，随着电子商务的快速发展，国内从事电子发票业务相关的企业也逐渐增多，比较典型的有航天信息股份有限公司、百望股份有限公司、汉特等等。

经调研分析，航天信息股份游戏那公司的发票管理系统以金税盘和电子发票平台为中心，以电子发票开具为基础，涵盖了发票开具、查验、推送等功能，但是由于基于 C/S 架构，在系统部署、配置、跨平台方面存在一定的约束条件，无法满足企业多平台、快速部署的需求。百望股份有限公司电子发票管理系统也基本覆盖了发票管理的所有功能，此外，因为该系统基于 B/S 架构，在平台兼容性、部署方面具有一定优势，但由于技术陈旧，目前也逐渐不能适应企业的新需求扩展。汉特公司开发的发票管理系统通过建立管理平台和数据接口，利用从 ERP 中导出的数据和扫描发票获得的数据，直接实现自动开票，消除发票流程中的手工劳动，保证开票的及时性和准确性。帮助企业建立统一的电子发票管理系统平台，将发票类型统一为增值税电子发票，实现电子发票自动化开具、集中存储管理、多渠道发票递送等功能。具有批量开具、查验下载、信息回传 ERP 系统等特点，并且在技术选型上，较为超前，但该产品对于设备的要求较高，系统稳定性较差，因此在企业客户中并没有得到很好的普及。

因此，通过深入调研分析，可以总结出目前国内电子发票管理系统具有如下的特点与不足：1. 绝大多数主流产品适配 Windows 系统。2. 产品技术架构陈旧、结构冗余，系统稳定性较差。3. 基于 C/S 的系统架构，产品具有复杂的安装步骤，在安装阶段、需要企业用户配置过多的用户信息。4. 移植性较差，大多数的产品系统未能集成企业已有的 ERP 系统。5. 国内产品主要考虑了开票和电子发票管理功能，尚未针对开票前企业用户与供应商用户沟通协作以及电子发票限额拆分做出可行的解决方案。

而本文结合国内产品的优点与不足，在系统适配方面，以 B/S 的轻量级系统 架构能够很好的适配 Windows、MacOS 两种操作系统。免去了繁琐的应用安装和配置过程，用户可以通过导入配置的方式做到快速配置。系统以基于 Restful 架构的 Jersey 轻量级框架作为后端业务处理的主要框架结构，保证了业务流程的稳定性。结合 Postgres+Elasticsearch 的独特优势，提升了数据的处理以及搜索能力。此外，在具体的业务场景中，着重考虑企业用户整个采购-收货-支付流程中在开票前，对于如何开具电子发票、开具后发票争议字段的协商、如何自动化核对电子发票数据以及发票限额拆分场景进行详细的讨论并提出有效的解决方案，弥补了国内电子发票管理系统在这方面的功能空白。

## 1.4 本文主要研究内容和本人工作

### 1.4.1 本文主要研究内容

本系统结合国内外发展现状，基于供应链管理场景，设计并实现该场景下一种可行的电子发票整体解决方案，弥补目前供应链管理中电子发票流程缺失的开票前发票协商、自动化核对流程。以浏览器作为客户端，可以适配大多数的企业办公环境，并且对外暴露有集成企业 ERP 系统的接口，更方便企业用户直接导入商业数据，快速部署和维护，具有一定的先进性。

使用本系统，可以简化企业在采购收货付款开票流程中的人力时力损耗。并且针对收货单、纸质发票格式、字段、商品价格、商品数量等数字信息可以基于该系统与供应商进行及时的沟通和修改，从而保证最后开票和付款的准确无误，具有一定的实用性。

供应链电子发票系统是一个基于 UBL 商业文档的，企业用户和供应商用户共同操作的发票系统。整个流程中主要的操作是针对 UBL 商业文档的操作，因此设计具有中国风格的发票电子文档格式是本系统开发的前提。

### 1.4.2 本人工作

上述是系统本周期需要实现的主要功能，在该系统中，我的工作量主要是：（涉及收货单、模拟发票、消息队列三个模块）。在本文中阐述的主要内容有：系统包括哪些功能、如何实现以及完成测试。主要研究内容包含有以下几块：

1. 设计收货单 UBL 文档、模拟发票 UBL 文档。
2. 基于设计好的收货单、模拟发票 UBL文档类型，设计并开发收货单模块、模拟发票模块相关的业务逻辑（相关文档的创建、修改、替换、删除等，以及针对文档数据的计算）。
3. 在两种用户之间的数据交换需要用到JMS+HornetQ作为消息队列来时刻转发各文档的状态变化。因此，基于JMS+HornetQ搭建系统消息转发监听服务也是我的一部分工作。
4. 服务端数据库设计。

## 1.5 本文的组织结构

本文主要分析了现有电子发票系统存在的问题和不足，针对这些方面，提出一种可行的电子发票系统解决方案。结合目前国内外的一些研究现状问题和市场调查，设计与开发一套服务于我国特有税收政策的电子发票系统，为企业用户在财务和采购供应管理方面提供便捷操作。本文的组织结构及相关安排如下：

1. ：绪论。本章节主要是对基于Restful架构的电子发票系统的研究背景、意义、国内外概括进行简要的介绍，通过分析国内外现有公司电子发票系统的特点和不足，提出一种可行的电子发票系统解决方案。此外，对笔者的工作量、论文章节结构进行简略介绍。
2. ：相关技术简介。本章节主要介绍与本文所述系统密切相关的基本技术。涉及Restful、Jersey、Postgres、El asticsearch、JMS、HornetQ、Json、XML、UBL等若干技术概念介绍。

第三章：电子发票系统的需求分析。本章节主要针对企业用户的使用场景、对该系统的各类需求进行明确的分析和介绍。

第四章：电子发票系统的概要设计。基于前一章节的需求分析，对系统的各类功能进行模块化划分，明确各模块之间的层次结构以及调用关系。在这章节中中，采用UML用例图等图表进一步说明各模块的组织结构。此外结合前端数据的调用需要，完成后端数据库的简单设计。

第五章：电子发票系统的详细系统设计与实现。本章节主要是对电子发票系统的整体架构、模块类图、系统业务逻辑层、接口调用层等进行详细的描述和设计。同时展示实现的效果图，并对此作一定的说明。

第六章：电子发票系统的测试。主要是从软件工程的测试角度对系统对系统各个模块的业务功能进行单元测试和集成测试。

第七章：总结与展望。在本章，笔者对该篇论文进行简要总结，同时分析该系统存在的不足之处，结合未来的发展对电子发票系统进行了展望。

## 1.6 本章小结

本章主要阐述电子发票系统的研究背景、意义以及国内外的现状，进一步分析了该系统在企业用户中的工作流程。在最后阐述了笔者的研究内容和工作，同时对论文的组织结构进行了描述，使读者对论文的结构一目了然。

# 第2章 相关技术简介

本章简单介绍与本文紧密相关的基本技术，了解这几种技术之后才能进行相关的开发操作。首先是开发语言Objective-C的介绍，其次对数据交换格式JSON的介绍，再次对XML 存储数据技术的介绍，最后是对iOS开发中常用的设计模式进行介绍。

## 2.1 编程语言介绍

由于本系统是基于iOS[9]的移动应用系统，则在开发时使用的开发语言有两种选择，一种是Swift语言，能够开发iOS程序[10][11]，但其使用范围并不广泛；另一种为OC(Objective-C[12])，它是开发iOS应用的主流语言，故选择其作为开发本系统的语言[13]。

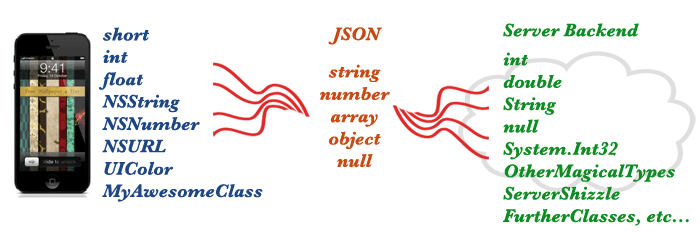
在进行Mac OS和iOS开发时，最常用的程序语言是OC。OC是C程序语言的超集，它具有面向对象的功能，同时也是动态运行时语言。OC具有C的语法、基本数据类型和分支语句，并且增加了定义类和函数的语法[14]。它也增加了对图像和文字管理，由于是动态运行时语言，很多处理都推迟到运行去做，这样大大减少了编译时使用的时间[15]。

## 2.2 数据格式JSON介绍

在与服务器进行数据交换时需要一种轻量级的、便于人阅读与编写的数据格式，这种格式就是常见的JSON数据格式[16]，在移动开发中，JSON被广泛用作前端与后台进行数据通信的格式。

JSON[17] 作为广泛使用的数据通信格式，它有两种格式，简单说就是javascript中的对象和数组，只需要使用这两种结构就能够表达出各种复杂的数据结构。首先是对象在javascript中的表示，使用“{}”包含数据内容，数据结构为{key：value,key：value,...}，在Objective-C中成为字典，在面向对象的开发语言中，key为属性，value则为相应的值，因此使用点方式获取属性的值。另一种就是数组在javascript中的表示，使用“[]”包含数据内容，和其他语言中数组的取值方式相同，直接使用索引[18]。

使用JSON的以上两种结构，可以在相应结构中进行嵌套，组合成更复杂的结构。JSON的值只有string、number、array、object及null。如图2.1所示[19]。

**图2.1 Objective-C与JSON数据转换图**

iOS中使用第三方库JSONModel来解析服务器返回的JSON数据，在本地建立一个数据模型类继承JSONModel，来接收JSON数据，JSONModel会自动解析JSON格式的数据，将其中的值直接赋值给模型中的属性。

## 2.3 数据格式XML介绍

XML是W3C推出的用于新一代网络数据表示、传递和互换的标准语言，它能够标识很多信息，无论数据是何种类型，都能够在它里面被定义，它是树形结构的纯文本文档[20]，目前作为主要的技术手段表示网络中信息的存储和交换。

XML是一种开放性非常强的元标记语言，它使用多种具有现实意义的标签对数据元素进行标识，同时能够多层嵌套。可以使用自定义的方式分配标签，以此来充分表达数据内容，这使得程序的维护工作更加容易。采用树型形式的XML可以清楚的呈现元素间的层次和前后顺序，同时对于描述结构化的实体更加方便和容易理解，XML被广泛的使用使得它的通用标准更广。

XML有以下术语[21]：

1. 元素（Elements）：标识XML文档中的每一个组件。
2. 属性（Attributes）：紧跟在元素之后，有名和值，标识元素的其他特征。
3. 统一编码（Unicode）：它是可以容纳世界上所有文字和符号的字符编码方案，由国际组织制定。
4. 实体：“文本片”，在XML中使用文本来表示实体对象，是XML文本的基本单位。
5. 标记（Markup）：像标记（TAG）、实体引用（Entity reference）和声明这样的构件。
6. 序言（Prolog）：关于文档实例表现形式的信息。

所有XML文档都必须符合下列规则[21]：

1. XML文档必须有一个根，其中的第一个元素就是根，并且有且只能有一个根，一定具有一个单独的标记来定义，所有其他元素都必须成对的在根元素中嵌套；
2. 所有的XML元素必须要正确嵌套，不允许不正确的嵌套；
3. XML标记是区分大写与小写的，这与HTML不一样，它的标记是大写与小写敏感的。在XML文档中开始和终止的标记必须保持一致；
4. 全部的XML文档一定要有一个终止的标记，在XML文档中，忽略终止标记是不符合规定的。

## 2.4 设计模式MVC介绍

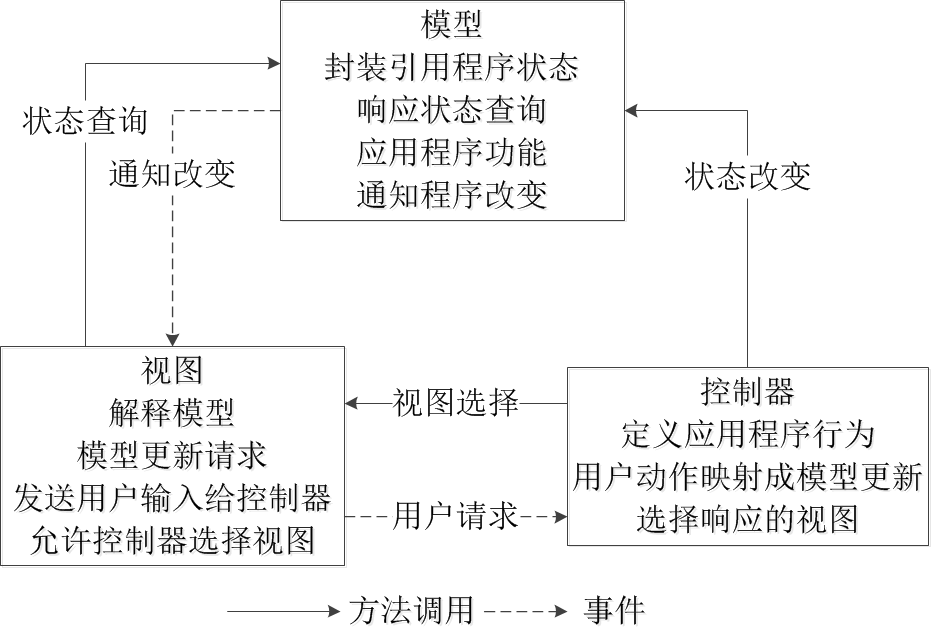
本系统使用MVC[22]模式进行开发，它是设计中常见的一种设计模式，也是我们客户端的总体设计结构。使用MVC模式将应用系统程序分成三层，每层都有自己的工作，各层之间分离，做到开发中常说的高内聚、低耦合。这三层分别为：模型层（Model），视图层（View），控制器层（Controller）。

模型层：处理应用程序数据逻辑部分，通常负责数据库的存取操作，一个model能为多个view供应数据，由于高复用性，所以会减少代码的重复性。

视图层：视图层负责展示界面效果供用户进行人机交互，显示模型数据。

控制器层：控制器部分处理用户的人机交互操作，读取用户在视图层输入的数据，同时也可进行控制视图层的输入，将输入的数据发送给数据模型层。

这三层之间的关系和功能如图2.2所示。

**图2.2** **MVC三层关系和功能**

由图2.2可知，一个功能模块在实际开发中可以分成三个部分，Model是开发中的数据模型，用来保存网络请求的数据，View则是展示给用户看到的界面，同时能够响应用户的操作，而Controller就是控制器，它的作用是负责Model与View的通信，负责业务逻辑。

视图控制器Controller连接数据模型Model与视图View，而视图上响应使用者操作的行为是使用代理或者Target-Action的方式实现的，响应操作后将事件传递给Controller，而Controller做出什么样的响应取决于用户操作的行为。当需要进行更新操作时，APP需要进行一次网络请求，将请求的新数据保存在对应的数据模型Model中，视图控制器Controller检查到数据模型有更新时，然后再使用请求到的数据更新视图上的显示数据，使视图展示的数据为最新的。因此，在这个数据交互更新中，所有的操作都是通过控制器进行协调的。

MVC分层的好处：

1. UIViewController代码量骤降，易于维护

视图控制器被拆分之后，它的主要职能就是进行逻辑控制，响应用户操作，对于其他的网络请求，数据保存以及视图显示的功能不在其中实现，而是被拆分到数据模型与视图中去实现，这样使控制器的负荷大大降低。

1. 复用性提高

当美工设计人员对展示的界面进行修改的时候，尤其是在项目初期，经常会修改之前已做的东西，这时开发人员就会轻松的多，界面的修改只需要修改View即可，其他地方几乎不需要重写。当然如果界面完全不同，则可以再写一个新的视图，只要业务逻辑不变，模型与控制器就不会变，这样就大大的减少了时间及人力，由于改动小也会给测试人员带来很大便利。

## 2.5 本章小结

本章对课题中使用的JSON数据格式，XML标记语言以及MVC设计模式进行了简单的介绍，同时介绍它们的优势。通过本章的介绍，快速的了解实现系统的关键技术，为设计实现基于iOS的个性印刷系统打下极好的技术基础。

# 第3章 电子发票系统需求分析

本章主要内容是对基于Restful架构的电子发票系统进行业务和功能上的需求分析。从用户的角度出发，考虑系统的具体使用场景。以此为基础，划分出明确的系统业务需求和功能需求。使用UML建图模型来绘制该系统的用例图、功能模块图、流程图和架构图，从而进一步将系统的各个功能点独立化，设计并调整功能之间的依赖关系，使之满足后续开发的开发流程。

## 3.1 业务需求

世界经济发展越来越成熟，对于每一个成熟的企业，都有若干的供应商与其对应，保持着稳定的采购供应往来。随着企业业务规模的不断扩大，企业的采购需求也随之增加，在企业和供应商之间大批量的商业货物交易，在极大地推对企业发展的同时，也带来了一系列的问题。大批量的交易往来导致短期内收货发货信息的急剧增加。面对这样的问题，大多数的企业还是依靠人力或者是简单的表格系统来处理繁杂的货物信息，这样的处理方式不仅花费较多的时间，而且对于货物信息中的时间、价格等关键信息的互相检查、核对有着很大的疏忽遗漏风险。这是数据核对方面的问题，此外，在供应商发货后导出开票的过程中，由于供应商与企业缺乏有效的沟通机制，常常不能及时地就发票字段、折扣、货物数量、金额等字段进行商定，从而导致最后纸质发票开票数据不吻合的情况。

随着互联网技术的发展，B2B的软件架构已经越来越成为主流。通过集成企业与供应商自己的信息管理系统，将需要管理和比对的数据放在单独的模拟电子发票系统中进行处理是一种可行的解决方案。通过供应链电子发票系统，企业和供应商可以随时跟进每一份收货单、模拟电子发票和纸质发票的状态。此外，供应链电子发票系统提供了快捷的沟通服务，能够基于当前需要修改的文档，及时发送修改信息给对方，待双方确认无误后可以正确规范纸质发票的各字段格式和数值，从而在整个采购到供应的流程中规范发票流程，极大地便捷了企业和供应商的操作，减少了双方在这方面的资源消耗。[6]

通过以上分析，结合企业供应链采购供应流程，可以总结出以下几点业务需求方向：

1. 系统界面设计合理，符合财务、采购管理部门的操作习惯；
2. 系统涵盖采购供应管理过程中需要的所有商业文档的电子格式，并且有与传统纸质文档相当的填写功能；
3. 设计合理的文档转换流程，实现文档之间数据迁移、填充自动化，保证整个文档转换流程中数据一致性；
4. 针对各类型文档，提供条件查询，满足用户对某一类文档的针对性处理；
5. 提供采购方用户和供应商用户之间文档的准确传输功能。

目前，国内外有若干电子发票系统产品，它们有各自的特点，但也普遍存在一些问题，诸如安装使用繁琐、技术选型落后、文档格式设计不规范统一、产品功能单一、与企业自身管理系统集成困难等缺点。因此，结合以上业务需求，并分析现有相关产品的特点与不足，采用基于Restful架构的电子发票系统主要有以下几个优势：

1. 基于浏览器客户端的B/S架构

基于浏览器客户端的产品设计，能够在满足用户所有业务需求的同时，可以免去用户单独安装产品系统，独立于各类操作系统和硬件安装要求。可以使用户可以在任何有网络的地方能够随时通过各类型主流浏览器登录系统操作相关业务。

2. 基于具有Restful架构的Jersey框架

使用Jersey框架，通过合理的功能划分，将各个功能模块独立化。并且清晰各个模块之间的层次结构和调用关系。相对于传统的设计架构，能够使开发人员更明确各个功能的覆盖范围和调用流程，从而保证了开发过程中功能实现的合理性和正确性。

3. 使用可扩展的UBL（ Universal Business Language）商业通用文档

UBL（ Universal Business Language），全称是商业通用语言，是在国际上广泛通用的商业文档格式。它使用XML标签来定义各个文档字段。开发人员可以结合开发需要，通过XML来扩展相应的商业文档类型。使用该文档，可以根据中国电子发票的特有格式以及企业用户的特有需要来设计满足条件的各类文档字段。

4. 兼容各类第三方服务，提供更多功能性服务

Jersey作为一款开源的Restful框架，能够兼容各类的主流第三方插件服务。通过使用这些插件，能够提供更好更全面的功能性服务。

以上是针对电子发票系统的业务需求分析。下面针对上述提出的各类业务需求方向，分析具体的业务场景，明确出合理的功能性需求，以及对系统的流程、架构进行分析。

## 3.2 功能性需求

业务需求是通过用户的角度来简要的整理出用户操作的时候需要的各项功能点。功能需求意在通过分析用户的业务需求和使用场景，从开发人员的角度出发，结合实际开发的技术、成本等方面的考虑，详细规划出合理的各项功能以及它们之间的关联依赖关系。

### 3.2.1 总体需求

通过上述的业务需求分析，可以初步确定电子发票系统的功能覆盖点。在接下来的功能需求分析中，将进一步针对每一个模块，结合用户的操作流程以及开发过程中需要采用的技术实现，详细划分该模块应该具有的特定功能以及与其它模块的相互关系。

供应链电子发票系统中包括了企业用户、供应商用户和系统管理员三种用户。其中，企业用户和供应商用户是该系统的主要服务对象，他们的职能存在于整个电子发票工作流程中。系统管理员主要存在于系统部署初期，根据用户给的权限控制信息，来预先配置对应企业相关用户的职能范围和权限防火墙。根据其不同的角色以及用户权限和操作流程，对系统有不同的需求。使用用例图可以清晰的描述电子发票系统各类用户的功能范围、各功能下属子功能以及各功能间相互关系。下图即是电子发票系统企业用户和供应商用户的完整用例图模型。



图3-1 电子发票系统整体用例图

在该系统中，主要的服务用户是企业用户和供应商用户。主要的商业文档类型包含对账单文档、模拟发票文档、预制发票文档以及由对账单衍生出来的的对账单行类型。在完整的工作流程中，企业用户通过对帐单管理以FTP的方式上传对帐单列表。并将需要批次的若干对帐单通过该系统发送给供应商用户，让后者根据具体收货信息进行勾选开票。在对帐单行管理中，主要由供应商用户通过查看所有对帐单的行详情，然后根据筛选条件，勾选若干行或一键全选创建模拟发票（在系统设计中，所有的对帐单在上传和发送后，都会由后台提取其所有行信息，存储在后台对帐单行数据库中，并在对帐单行管理模块中集中操作和处理）。供应商通过勾选创建或者手动创建均可以进入模拟发票管理模块，在该模块中，供应商用户主要负责创建模拟发票，发送模拟发票，以及针对企业用户的消息修改模拟发票。同样的，通过模拟发票管理，企业用户针对供应商用户发送来的模拟发票可以执行接收、拒绝或发出修改信息等。在双方沟通并确定模拟发票信息后，供应商用户通过预制发票管理，根据预先自定义配置的开票限额，拆分模拟发票为一张或多张预制发票（预制发票在版式、字段、内容数据均与最后的纸质发票相同）并调用当地税务局系统进行开票。

系统管理员主要是对整个系统的配置信息进行管理。系统管理员可以根据企业用户的要求来配置各个公司用户的个性化配置。

本需求着重描述供应链电子发票系统的设计与实现。此系统主要涉及收货单模块、行详情模块、模拟发票模块、预制发票模块、消息队列模块等。

### 3.2.2 对账单模块需求

对帐单管理模块的用例图如图3-2所示。

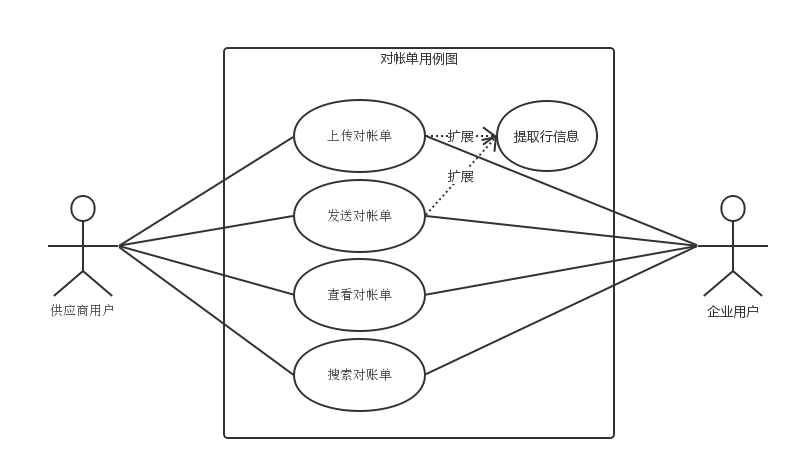


图3-2 对帐单用例图

对帐单模块主要负责商业文档类型为对帐单的文档处理。供应链电子发票系统主要服务对象是企业用户和供应商用户。对于企业用户，在该模块中可以通过FTP上传接口将需要处理的收货单批量上传到供应链电子发票系统。企业用户可以在该系统中通过筛选，选择需要发送的收货单，在点击发送后，系统可以自动根据每一份收货单中的公司信息来找到正确的供应商公司，将对应的收货单发送给正确的供应商。考虑到具体业务的实现方式和用户操作流程，用户需要检查并通过对帐单的每一行行详情勾选创建模拟发票。因此在系统开发中，后台在用户上传和发送对帐单后，自动将每一份对帐单拆分并保存所有的行信息，这次行信息会在对帐单行管理模块中统一管理。供应商用户可以接收到企业用户发送来的收货单信息，并在该系统中筛选查看。可以根据需要勾选对应的收货单来直接导出模拟电子发票，其中的商品数据会从收货单中直接提取，从而保证了模拟电子发票在创建的时候就保持与收货单商品数据的正确对应。企业用户和供应商用户对帐单用例图如图3-2 所示。

### 3.2.3 对帐单行详情模块需求

对帐单行管理模块用例图如图3-3所示。

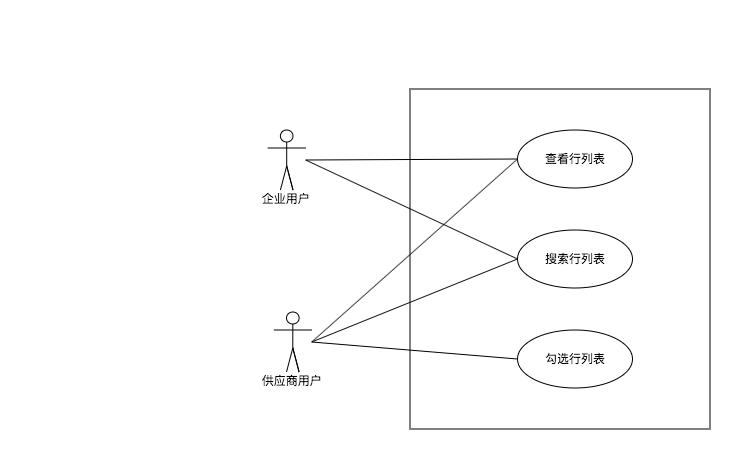


图3-3 对帐单行详情用例图

对帐单行详情管理模块是根据用户操作流程和业务场景而从对帐单管理模块中衍生出来的功能模块。该模块主要在供应商用户方面操作。在企业用户上传和发送对帐单后，后台会将对帐单的行信息单独提取出来存储在后台行数据库中，并在该模块中统一操作管理。供应商用户在该模块中可以默认查看到所有对帐单行的列表信息，或者通过筛选功能，以日期、公司名称等过滤出特定的行信息。在找到需要的行信息后，供应商用户可以通过勾选开票来创建模拟发票。如果一次创建需要当前筛选的全部行信息，可以通过一键开票来免去多次勾选的冗余操作，加快开票流程。企业用户和供应商用户对帐单行用例图如图3-3所示。

### 3.2.4 模拟发票模块需求

模拟发票管理模块的用例图如图3-4所示。

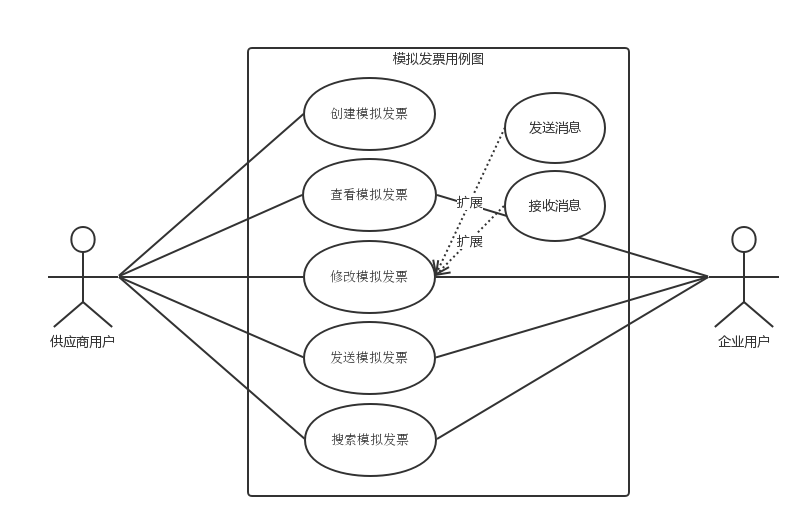


图3-4 模拟发票用例图

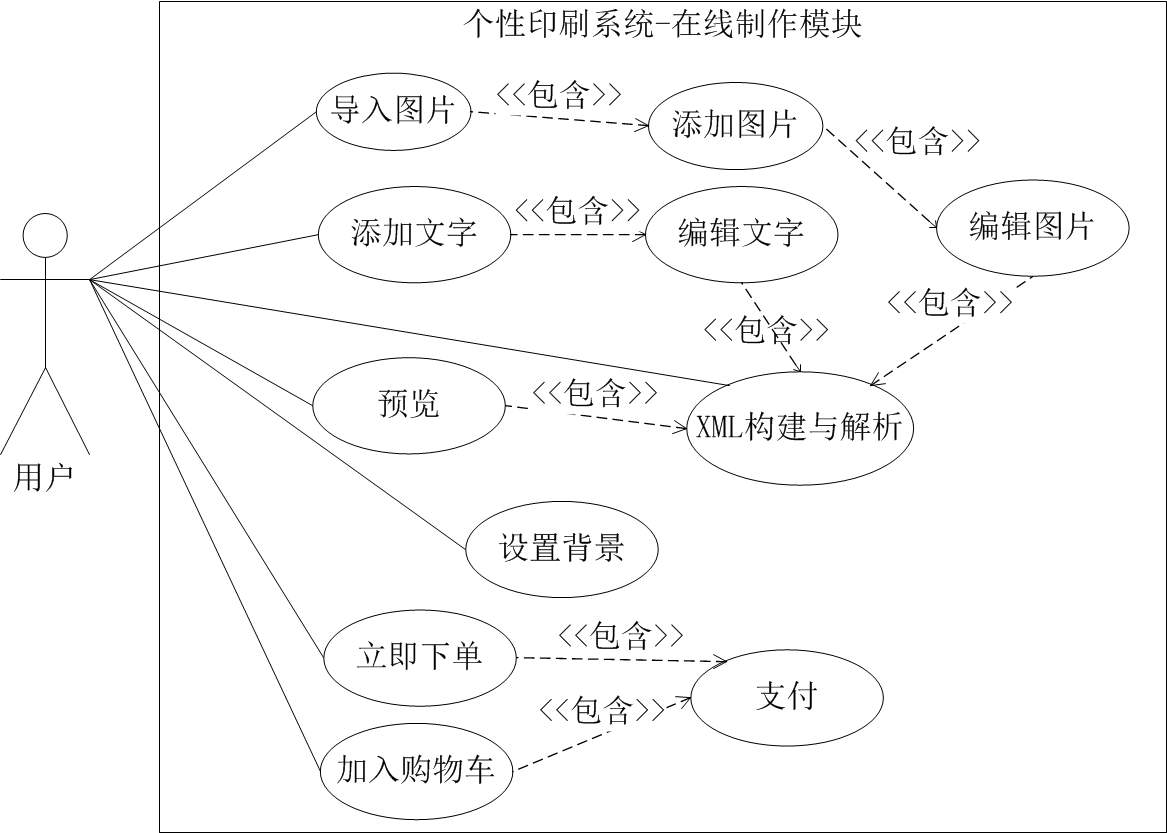
模拟发票是供应链电子发票系统的主要数据文档，所有与纸质发票相关联的信息都在模拟发票中一一对应，任何一个针对模拟发票的操作在最后导出开票时，均会影响到纸质发票的票面信息。供应商用户在勾选若干对帐单行自动创建模拟发票或者手动创建空白模拟发票后，系统会根据这些收货单中的信息或者用户的输入信息创建一份模拟发票，并同时更改对应收货单的状态，并且可以通过筛选选择对应的模拟发票发送给企业用户。双方可以将这份模拟电子发票视为电子版的纸质发票，在样式、字段方面均与纸质发票保持一致。对于该模拟发票中的差异信息，企业用户和供应商用户可以通过该系统的信息沟通服务来随时发送修改信息以征求对方的确定。这种及时的沟通方式极大的方便了双方的信息交换，简化了时间和人力的花费。企业用户和供应商用户模拟发票用例图如图3-2 所示。

在创建模拟发票并发送之后，供应商用户等待企业用户操作结果，不同的操作结果会影响供应商用户不同的后续操作流程，具体的几种操作如下：

1. 接受，企业用户在收到供应商用户发来的模拟发票后，如果发票各类信息准确无误，企业用户通过点击接受按钮认可该模拟发票。后台在收到企业用户的接受信息后，会自动更新供应商用户一方模拟发票中对应的模拟发票状态。供应商用户可以在这基础上针对这份已被接受的模拟发票执行拆分成为预制发票的操作。
2. 拒绝，状态变更与接受操作类似，在企业用户拒绝这份模拟发票后，供应商用户收到拒绝信息，此时这份模拟发票属于无法操作状态，供应商用户无法针对这份模拟发票做任何操作。
3. 修改，企业用户如果给出修改信息，后台会自动针对需要修改的地方，重新临时生成一份修改后的模拟发票（这份临时的模拟发票除了被修改的地方存在信息更新外，其它字段与原有的模拟发票完全一致），企业用户将这份临时模拟发票发送给供应商用户。供应商用户会在模拟发票管理模块中发现这份临时的模拟发票修订版，在查看后，如果同意修改，则通过点击确定按钮，后台会自动使用这份修改后的临时模拟发票覆盖原有的模拟发票；如果不同意修改，则与其用户操作类似，可以再针对修改意见发出新的临时修改后的模拟发票，供企业用户操作。

### 3.2.5 预制发票模块需求

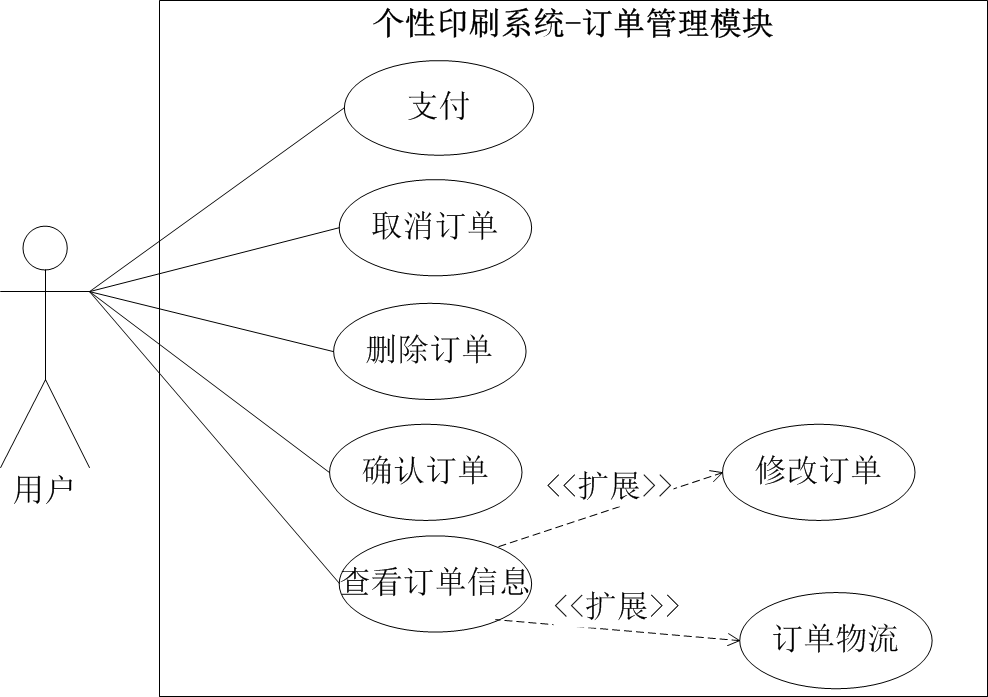
预制发票管理模块用例图如3.5所示。

图3-5 预制发票例图

预制发票管理模块依赖于模拟发票管理模块，主要负责模拟发票的拆分和最后开票工作。企业用户与供应商用户在确定接受模拟发票后，这份模拟发票可以进入预制发票管理模块。供应商用户点击该模拟发票界面的创建预制发票按钮，后台会根据当前用户预先配置的开票限额规则以最优的方式来拆分当前模拟发票；拆分后，当前的模拟发票会划分为若干个预制发票。在拆分过程中，会将该模拟发票与拆分出的预制发票相互关联，在预制发票的备注信息中会保存原始模拟发票的发票号码。这样便能时刻跟踪对应的模拟发票和它的所有预制发票。此外，这些预制发票的总金额相加即是模拟发票的总金额，其它字段信息与模拟发票完全一致。供应商用户在拆分出预制发票后，点击导出开票，系统会提取每一份预制发票的字段信息，发送给当地税务局系统，调用相关接口开具纸质发票。在整个开票流程中，所有字段数据都是在双方用户查看沟通后确定的，在各文档的转换中，数据也时刻保持一致，因此在最后的纸质发票开具时，该系统可以保证若干张纸质发票的字段名称、数据金额在汇总后与对应的模拟发票信息完全一致。企业用户与供应商用户的预制发票用例图如图3-5所示。

### 3.2.6 消息转发模块需求

消息转发模块用例图如图3-6所示。

图3-6 消息转发用例图

消息转发模块主要负责企业用户和供应商用户之间的信息传递。需要传递的信息包括：各类型商业文档、用户针对文档的临时修改文档以及文档的状态变更。在该系统中，文档的传递始终是以传递副本的形式进行；当前用户创建的文档在传递时，后台会复制一份与原始文档一致的文档发送给接收方，接收方始终保留着原始文档传递后的副本，并不是直接传递原始文档。这样可以保证创建者对于原始文档有足够的跟踪权限，维持了文档稳定的生命周期。对于临时修改信息文档和文档的状态变更，也类似于上述的传递方式，只不过此时的副本变成了对应的临时修改信息副本和状态副本。企业用户和供应商用户的消息转发用例图如图3-6所示。

## 3.3 非功能性需求

在软件系统的需求分析中，包含功能性需求和非功能性需求。其中非功能性需求主要是指系统在满足用户业务场景的各类功能性需求之外，还具有的其它方面的特性，例如系统性能、可靠性、可维护性等特性。

### 3.3.1 健壮性

健壮性主要描述了系统处理异常错误的能力。在用户输入非正确内容时，系统能够针对这些非正常输入内容能够做出正确的处理，使系统保持正确的运行，如果系统无法处理这些异常，将会导致系统运行期间出错，甚至引发系统强制关闭。

在该系统中，考虑到具体的业务场景，用户会输入诸如税率、单价、折扣等数字方面的信息。在输入这些信息的时候，如果不能在输入阶段检查用户输入格式是否有误，则在后续的计算过程中，就会因为读取了错误的因子而出现计算方面的异常。因此，在用户手动输入阶段，系统通过检测，及时定位异常形式的数字或其它内容，并给出相应的提示性信息，那么就能够在输入阶段保证用户输入的正确性，从而保证后续的计算过程准确无误。

### 3.3.2 可维护性

软件的可维护指的是在后期的系统维护中，相关人员可方便的对系统产生的错误进行纠正、校对，保证系统的正常运行或满足新的需求。系统需要进行维护的原因主要有以下两种[24]：

1. 改正性维护。改正性维护是指软件在使用时发现了一些潜在的漏洞（bug）之后，为了诊断和改正这些bug而修改软件的源码、配置文件等进行的活动。
2. 完善性维护。完善性维护是指为了完善现有系统的功能或提高系统的性能及用户体验，而进行的修改系统的活动。由于系统采用的是软件工程中的迭代开发，第一个开发周期只完成简单的基础功能，系统后续会继续进行迭代开发，这就要求本系统在开发时应该设计好系统架构，编码时保证一些方法和类能够共用便于后期扩展。

### 3.3.3 可靠性

可靠性主要描述了系统在可靠的输入条件以及实际运行期间，能够稳定提供给定服务的性能。因此，为了确保该系统的可靠性，针对用户的需求，需要考虑以下几方面的开发考虑：

1）浏览器系统：系统需要一定的浏览器版本支持，使系统在企业和供应商中都能稳定应用，目前支持 IE8、Chrome 45及以上版本、Safari、Firefox 等主流浏览器。

2）界面需求：对于企业用户和供应商用户，因为要处理大量的文档数据，界面应该设计友好，结构清晰，操作方便，有良好的用户体验；而对于管理员用户，界面需要能方便访问，对管理项目清晰明了，操作简单。

3）通信网络：根据系统要求，能在公共互联网上通信，访问系统服务器，实现数据的传送和接收。并且在数据传输过程中采用HTTPS加密的形式传输，保证数据的安全性。

4）数据存储：系统需要对所有的数据进行安全的存储，方便各种用户对数据的操作。在大量用户同时访问时，保证数据提取的准确性和高效性。

## 3.4 本章小结

本章的主要内容是针对供应链电子发票系统通过UML建模进行了详细的需求分析。通过前期调研和与用户进行沟通，对用户的业务需求进行了整理，明确了用户的业务操作场景以及针对哪些文档具有哪些操作等。功能需求分析主要是在业务需求分析的基础上，从开发角度，以模块化的思想划分业务场景，针对模块进行功能点的确定和划分以及合理设计各模块间的耦合、依赖关系，使之满足实际的开发需要。最后，针对系统的非功能性需求，从健壮性、可维护性、可靠性三方面分析了系统在满足用户业务场景的前提下，对异常处理、操作环境、响应速度等因素的要求。

# 第4章 个性印刷系统概要设计

本章主要是对设计的个性印刷系统的移动端和服务端进行总体的概述，通过对系统总体结构、功能结构以及数据库进行概要设计，为第五章详细设计与实现提供良好的思路。

## 4.1 系统整体结构

基于iOS的个性印刷系统构建的是一种为移动用户带来巨大方便的系统，不断的提高个性印刷系统方面的工作效率问题，同时提供良好的界面以便用户进行相应的操作与处理。从系统的整体设计实现的角度考虑，个性印刷系统的整体架构为如图4.1所示。用户通过移动端或者Web网页登录系统，登录成功后进行印刷品的制作、在线下单付款操作，完成后向服务器端发送印刷品数据。后台服务器对数据进行处理最终将成品送到用户手中。

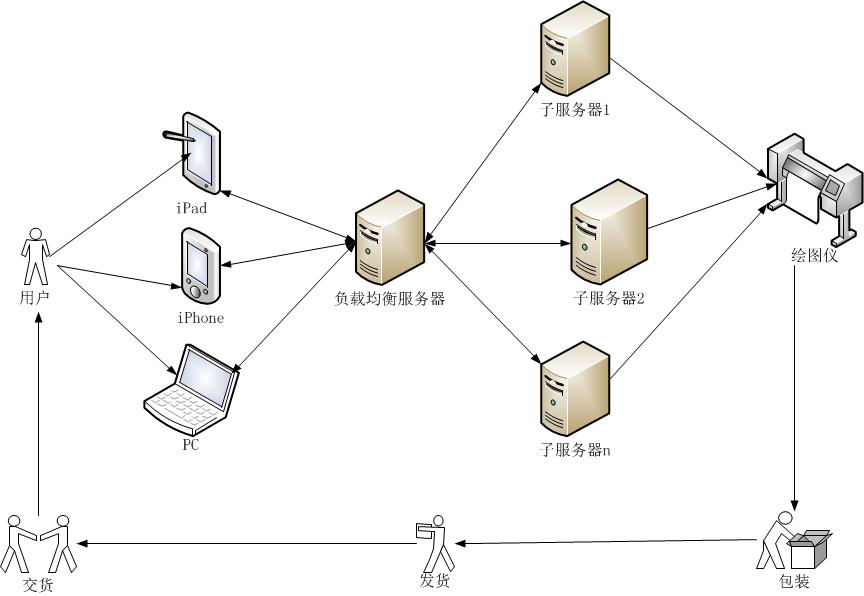
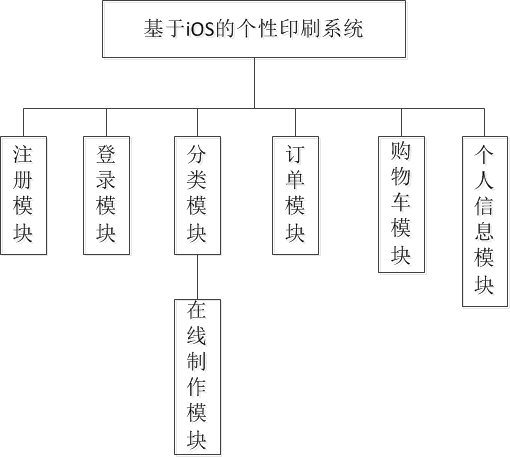
**图4.1** **系统整体架构**

图4.1描述了系统的整体架构图，使前端与后台的工作显得更加分明。当前端向服务器发起请求时，后台对请求进行处理，最后给出响应。

本文研究的是基于iOS的个性印刷系统，其架构为C/S架构模式[25]。首先，移动端能够接收使用者的操作，进行交互，且提供良好的展示效果；其次，与服务器进行数据交换时，服务器能够给移动及Web端提供一些数据交互的接口。客户端与后台建立连接后可以进行增、删、改、查操作，然后服务器再去操作相应的数据库。

****基于iOS的个性印刷APP若按模块划分，则主要分为注册、登录、分类、在线制作、订单管理、购物车以及个人信息模块。模块的划分是以移动端页面布局来划分，每个模块下都有丰富的功能，这些功能在业务上可能会相互调用，例如购物车模块在进行结算的时候会调用个人信息模块下的地址管理功能，获取收货联系人信息。图4.2为系统模块结构图。

**图4.2** **模块结构图**

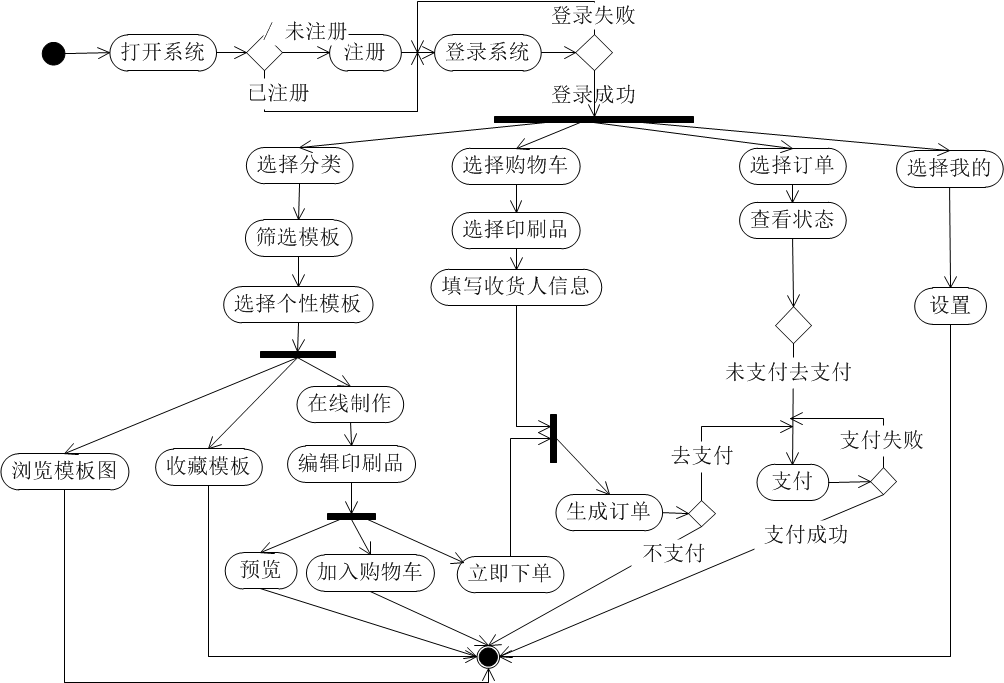
图4.2为系统的总体模块结构图，在分类模块下选择模板后进入在线制作模块，在该模块下进行制作印刷品等一系列的操作。在线制作模块中有立即下单和加入购物车等功能。

## 4.2 模块设计

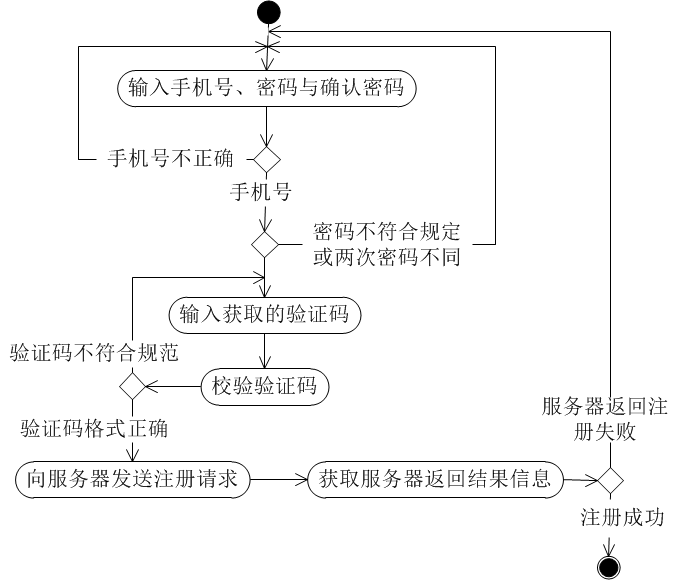
### 4.2.1 总体处理流程

在进行模块设计之前必须清楚总体的联系与处理流程，移动端APP各个功能模之间的联系如图4.3活动图所示。

图4.3体现了各个模块之间的关系以及处理流程。使用者必须先登录，如果没有账户则需先注册，成功注册并登录后，可以看见四个模块，模块之间进行切换时可以查看个性印刷品分类、订单信息、购物车信息以及个人基本信息。之后几节主要描述注册登录、产品分类模块、在线制作和订单管理、购物车以及个人信息模块。

**图4.3** **功能模块活动关系图**

### 4.2.2 注册模块的设计

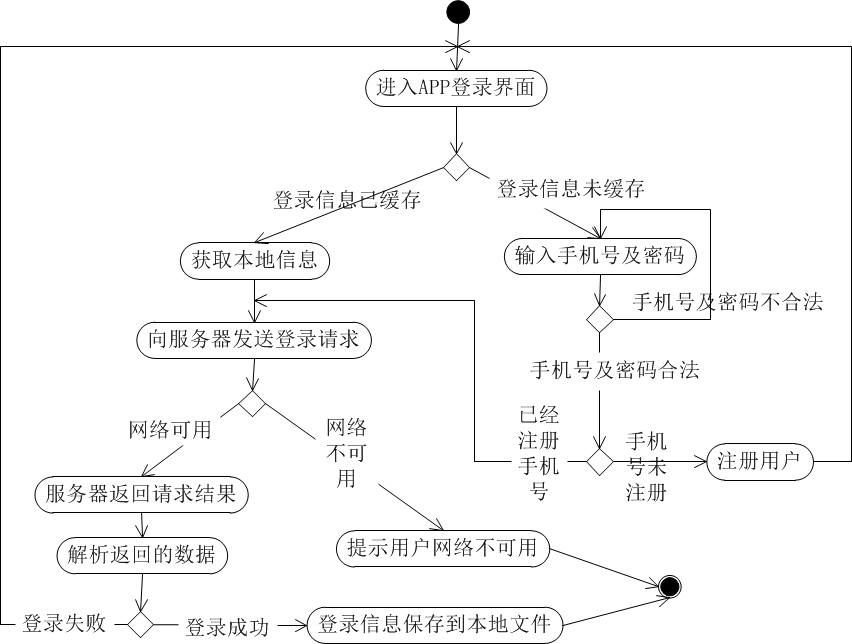
****用户进行注册时，需要输入手机号，密码以及短信验证码，为了保护用户信息需要对注册密码进行加密处理。后台数据库中的用户表的设计是以手机号码为主键，所以注册时只能使用未被注册过的手机号码，其他输入都应该提示错误，同时注册按钮无法点击。因此在客户端需要对输入的手机号进行验证，密码的输入也应满足一定的规范。如图4.4注册的活动图所示。

**图4.4** **注册活动图**

图4.4为注册活动图，整个APP的注册过程遵照图4.4的流程进行。注册时需要填写手机号、密码和验证码，客户端需要对填写的手机号进行正确性验证，同时，输入的密码和验证码需要符合一定的规范才允许注册。向服务端发起网络请求之前需要对注册信息进行加密处理，双方使用RSA加密技术[26]进行处理，并需要服务端提供公钥保存在客户端，注册成功后返回到登录界面进行登录操作。

### 4.2.3 登录模块的设计

登录模块需要用户输入手机号和密码，完成输入后点击登录按钮客户端向服务端发起网络请求。在发起网络请求前先对手机号及密码进行验证，同时需要对请求的数据进行加密处理，服务端再使用私钥进行解密。客户端的登录设计分两种情况，分别为自动和手动输入登录。第一次打开系统时本地没有保存用户的任何信息，需要手动输入，再次打开直接登录。

****图4.5为登录模块的总体活动图设计。

**图4.5** **登录活动图**

图4.5为系统登录活动图，APP启动时会进行自动登录，从本地获取用户的登录信息，若本地存在用户登录信息则会自动发起网络请求进行登录，若不存在则需要用户手动输入。输入手机号和密码后，客户端判断手机号及密码是否合法，合法后发起网络请求，否则重新输入。登录成功后将手机号与密码信息写入到本地文件进行保存，同时进入系统，登录失败则重新登录。

### 4.2.4 分类模块的设计

产品分类模块按产品的不同进行分类，界面显示则分左右两栏列表，左侧列表显示所有模板的分类，右侧列表则显示对应分类的所有模板信息。用户进入分类模块时会向服务器发起一个网络请求，从服务器获取所有分类信息进行展示，数据以JSON格式返回。选择一种喜欢的模板分类后，会再次向服务端发起网络请求，根据点击的分类编号从服务器获取该分类的所有模板信息，将模板信息显示在右边的分栏中。

图4.6为分类模块的整体活动图。

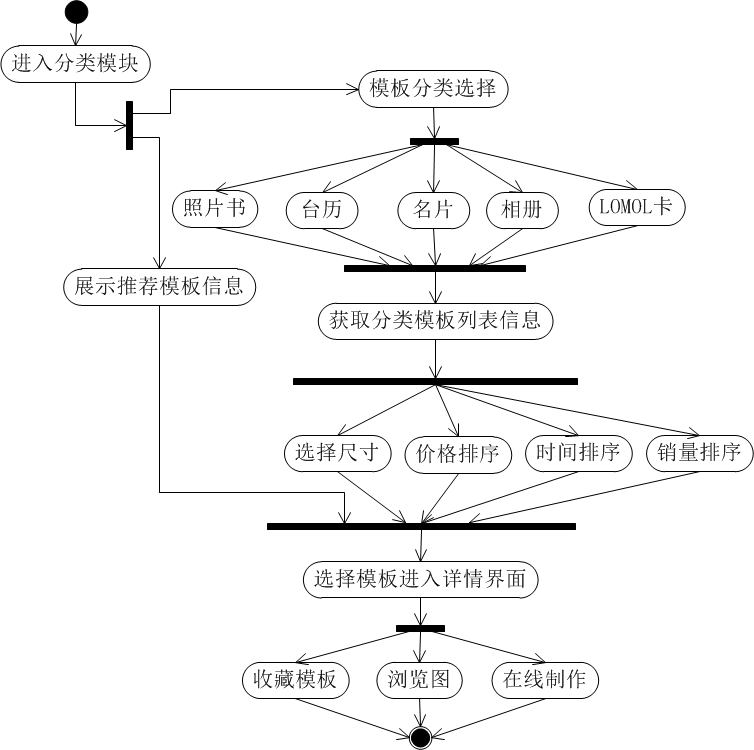
**图4.6** **分类模块活动图**

图4.6展示了分类模块的操作流程，在客户端开发中，分类信息应该做到动态改变，并非固定的几种分类存储在客户端本地。本系统中用户看到的所有数据几乎都是从服务端数据库中获取得到，这样设计的优势为更新分类信息而不用发布客户端版本。模板分类的视图显示使用UIKit中的一组基本组件UITableView进行展示，而右侧的模板使用自定义的UICollectionView来展示。在每一项选择下用户还可以根据尺寸筛选，按上架时间、价格及销量进行排序从而筛选出自己想要的模板。分类模块涉及到的几个功能使用以下几种方式进行设计：

1. 向服务器请求分类数据，动态展示分类种类信息，进入分类模块后，前端向服务端发起网络请求，前端可以分页请求，即每次获取多少条数据，服务器查询数据库，将查询结果组织成JSON格式传回前端，当然分类信息并不算多，在这里没有必要进行分页请求。客户端使用JSONModel解析服务端传回的JSON数据，最后将解析的结果存储在数组中。
2. 展示分类信息，请求的分类数据信息需要展示在客户端，可以从数组中得知共有多少条分类数据，使用UITableView来展示所有的分类数据。UITableView是Objective-C中的自带的系统类，使用它时需要设置代理，并实现它的几个代理方法。
3. 选择一种分类，将分类的编号作为参数向服务器进行分页请求，同样对数据进行解析后存放在另外一个数组中，这些数据就是模板数据。使用自定义的UICollectionView展示模板信息，使用方法同UITableView。展示的数据包括模板名、价格、尺寸以及模板缩略图等，这些数据信息都从后台数据库中获取。
4. 选择模板，点击模板进入详情界面，查看模板的详情描述，在详情页可以收藏模板、浏览模板图片，也可以进入在线制作模块进行个性印刷品的制作。

### 4.2.5 在线制作模块的设计

在线制作模块是本系统设计的核心部分，我们收藏、筛选模板的最终目的就是为了根据自己需求设计出个性的产品。该模块包括功能较多，逻辑相对复杂，如导入图片、添加图片框、添加文字框、预览、设置背景、立即下单、加入购物车等。用户进入在线制作模块时，客户端会从服务器获取对应模板的XML文件，然后解析文件中的内容并展示在界面上。用户编辑模板时就是对XML文件进行读写操作，下单后会将XML文件中的数据以字符串形式发送给服务器进行处理。在线制作模块的结构图如图4.7所示。

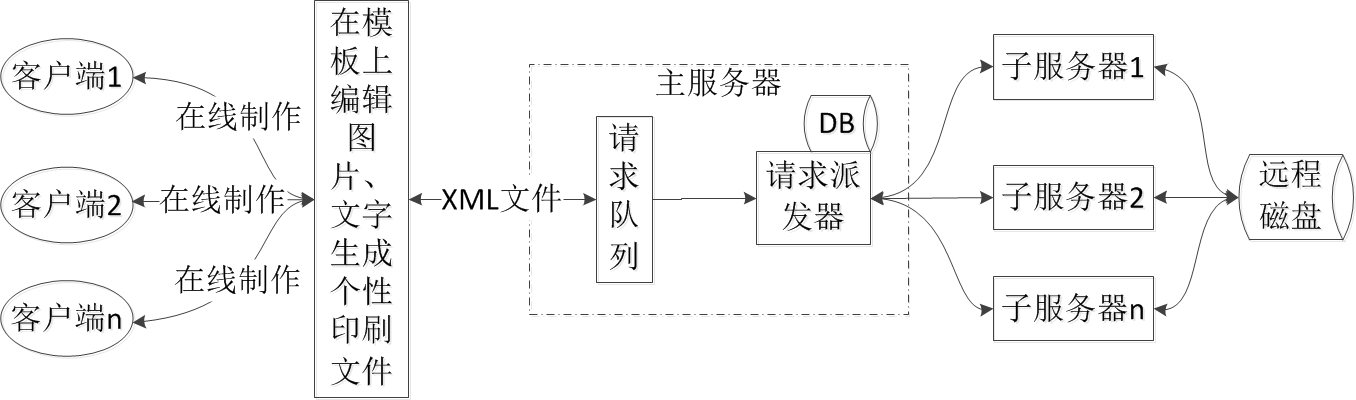
**图4.7** **在线制作模块结构**

图4.7描述个性印刷的工作结构，客户端完成所有编辑后读取印刷品文件信息，转化成字符串后发送给后台服务器，后台将所有请求存放在一个队列中，然后根据子服务器的当前情况进行分发处理。

用户在客户端编辑个性印刷品时，首先需要把要使用的图片导入到系统中，展示在在线制作页面的左侧，编辑时可以对图片进行滤镜渲染、剪切处理，然后生成新的图片，同时可以为印刷品添加文字。图片与文字的内容及位置只要发生变动都会更新XML文件中对应的数据，由于XML是可标记语言，它能够记录制作的印刷品的布局。编辑好的个性印刷品会生成XML文件，用户下单付款后就会将XML文件中的数据发送给服务器，服务器解析数据，并插入到数据库中。

在该模块中用户的操作有多种，如导入图片、编辑、预览、加入购物车或者下单付款，在线制作的总体活动图如图4.8所示。

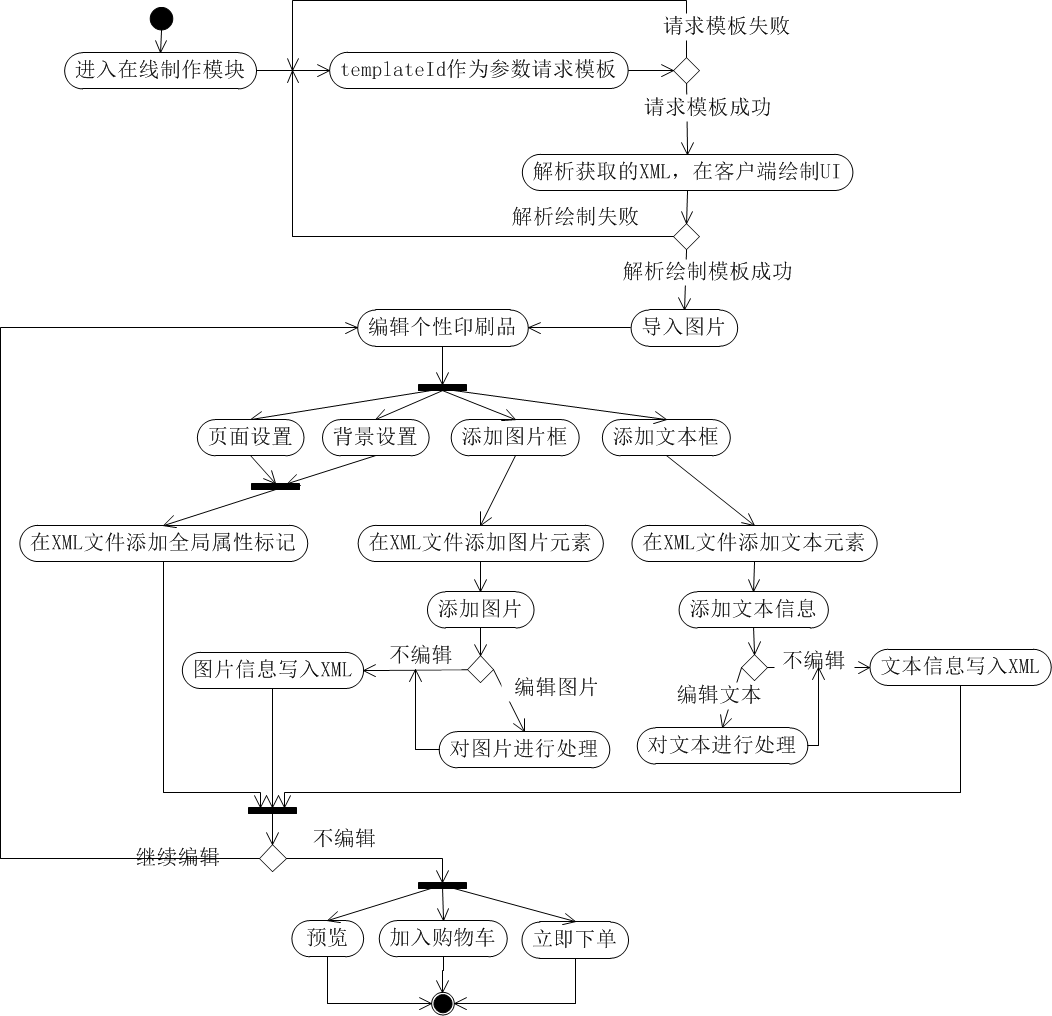
**图4.8** **在线制作模块活动图**

图4.8为在线制作总体活动流程，主要有以下几个操作：

1. 导入，在线制作界面的左侧是图片列表，点击“+”标志会打开相册，用户选择图片导入到列表中，列表使用UITableView实现。
2. 编辑，编辑区域即是解析模板文件后的展示区，布局如何显示取决于XML文件中的数据。每添加一张图片就会在XML文件中添加一个元素及属性，记录这张照片。添加文字也是同样的道理。点击该区域中的一张图时，在布局上方会出现相关操作，通过这些操作可以对该图进行编辑。
3. 预览，完成个性印刷品的编辑后，点击预览按钮，客户端会将之前编辑好的个性印刷品从XML文件中读取出来显示给用户，让用户直观的看到最终打印出来的效果。
4. 加入购物车，完成编辑后生成XML文件，将该文件加入到购物车，可以继续进行其他操作。加入购物车的个性印刷品会有唯一的编号标记它，同时标记它所属分类，尺寸以及价格。完成所有操作后进入购物车进行结账付款。
5. 下单付款，完成编辑后，用户可以直接点击提交按钮，填写信息后生成订单，进行结算时调用第三方支付完成这次交易，完成的交易可以在订单模块中查看到。

在线制作的过程中会遇到如下两个问题，编辑印刷品以及图片处理。

1. 个性印刷品的编辑，所有的数据模型都需要保存在文件中，然后传递给服务器。服务器对该文件数据进行解析，最后生成工业印刷文件进行打印。建立XML模板需要构建个性印刷品的数据模型，通过XML文档来记录模板布局信息。XML是一个树形结构的文件，记录的各元素之间的关系结构图如图4.9。

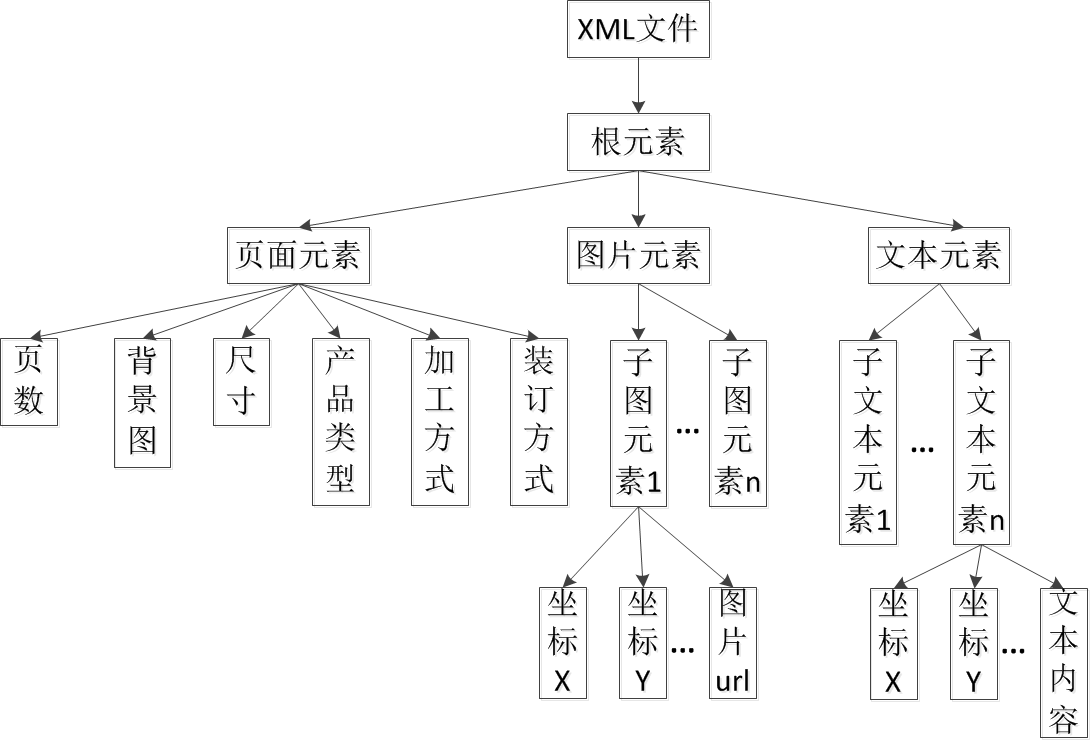
**图4.9** **XML模板文件结构图**

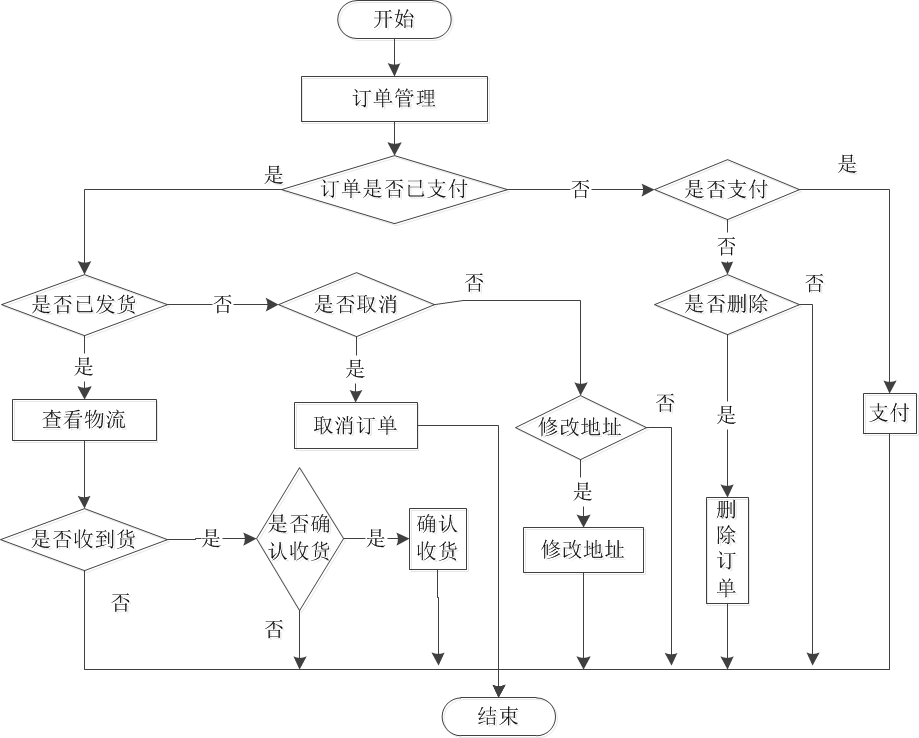
图4.9 为XML文件记录的各元素之间的关系结构图。根元素下只有三种类型子元素，页面元素记录整个模板的信息，包括页数，背景图等子元素；图片元素下就是模板中包含的所有子图元素，每个图片也有他们自己的子元素，记录图片的属性。客户端有一个UI生成类GenerateUI，它的作用为获取个性模板XML文件，然后解析该文档中的数据，完成代码到页面的映射。

在实际应用中表3.1列举出的元素只是其中一部分，本文篇幅有限只列举部分关键元素。通过记录这些元素信息，前端可以更好的解析文档数据将其映射到图形界面上展示给用户。

1. 处理图片时，可以设置Quartz2D中的context的transform属性，为该属性赋值不同的值使其能够偏移、缩放、旋转的功能[27]。context中我们的操作都是基于它的坐标原点，因此在context中需要旋转图片时，最少需要两个步骤：旋转和平移。Quartz2D也可实现给图片添加水印效果，首先要开启一个基于位图的上下文，将图片和文字都写入到上下文中，最后从位图上下文中获取添加水印的图片。滤镜效果及图像实时呈现是个技术难点，解决这个问题是使用GPUImage实现滤镜效果[28]。GPUImage是一个基于GPU图像和视频处理的框架，由于使用GPU处理图像和视频[29]，所以速度非常快，处理速度是使用CPU来处理的100倍。应用层将图片文件传递给OpenGL层进行一系列的处理[30]，完成后直接显示到屏幕上。

### 4.2.6 订单管理模块的设计

客户端订单功能模块是管理用户订单数据的关键服务。在订单模块中，用户可以查询所有的订单信息及订单状态，订单的状态可以提示用户下一步操作，在该模块中用户也可删除未支付的订单。订单管理模块以列表的形式显示所有订单信息，视图单元继承UITableViewCell定制符合系统需求的视图单元。进入该模块就会向服务端请求订单列表信息，服务器通过分页的方式返回数据，有效的避免时间的浪费以及流量消耗。客户端请求时可以指定请求参数pageIndex和pageSize的值，说明每次请求第几页多少条数据，每次上拉加载更多时pageIndex就加1再次请求数据。在订单列表界面展示订单的基本信息，包括模板单价、印刷产品名称、订单状态、产品数量、总价格、市场原价格及个性模板预览图。

****订单管理模块的工作流程如图4.10所示。

**图4.10** **订单模块工作流程图**

图4.10订单模块工作流程图，进入订单详情后，可以查看物流、地址和创建时间等信息，客户端可以根据服务端返回的订单状态码来控制操作逻辑。同一界面上根据状态码的不同控制界面如何显示，逻辑上相对复杂。订单是否可删除以及能否查看物流信息，编辑联系人信息都取决于服务端返回的状态码。

### 4.2.7 购物车模块的设计

购物车模块在APP中体现了现实购物中购物车的功能，用户将个性印刷品加入购物车中，完成所有操作后进行统一付款。图4.11为购物车的总体活动图。

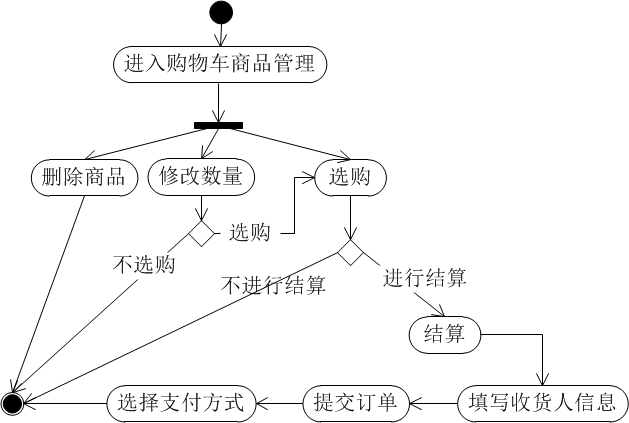
**图4.11** **购物车活动图**

图4.11为购物车模块活动图，该模块总体设计包括购物车中商品的简要信息展示，以及修改商品数量和结算功能，同时可以将某一商品从购物车中删除。

1. 在商品列表中用户可以通过点击“+”和“-”标志来修改购买商品数量，下限为1，无上限。修改数量后，界面上总价和数量要实时更新，在这里使用block来实现。
2. 删除商品，用户不想购买的商品，可以从商品列表中删除，具体做法是向左滑动商品单元格，出现删除按钮，点击删除按钮即可。
3. 选择购买的商品，用户选择购买的商品后，这些商品的数据信息会加入到一个数组中保存，然后进入订单信息填写界面。填写收货人信息时有两种方式，一种是创建一个新的联系人，填写所有信息，另一种则是从已有的收货人地址信息列表中选择一个。所有信息填写完成后进行付款操作，在本系统中暂时只支持三种付款方式，分别是支付宝、微信和银联支付，在系统中只需要集成他们的SDK即可。付款成功后这些商品将从购物车中移除，并且出现在订单模块中。

### 4.2.8 个人信息模块的设计

个人信息模块设计的目的是管理用户的个人信息，在实际APP页面中体现在“我的”一栏下。个人信息模块总体设计包括的功能有修改头像、查看收藏的模板、地址管理、意见反馈、客服帮助以及设置等，个人模块的总体活动图设计如图4.12所示。

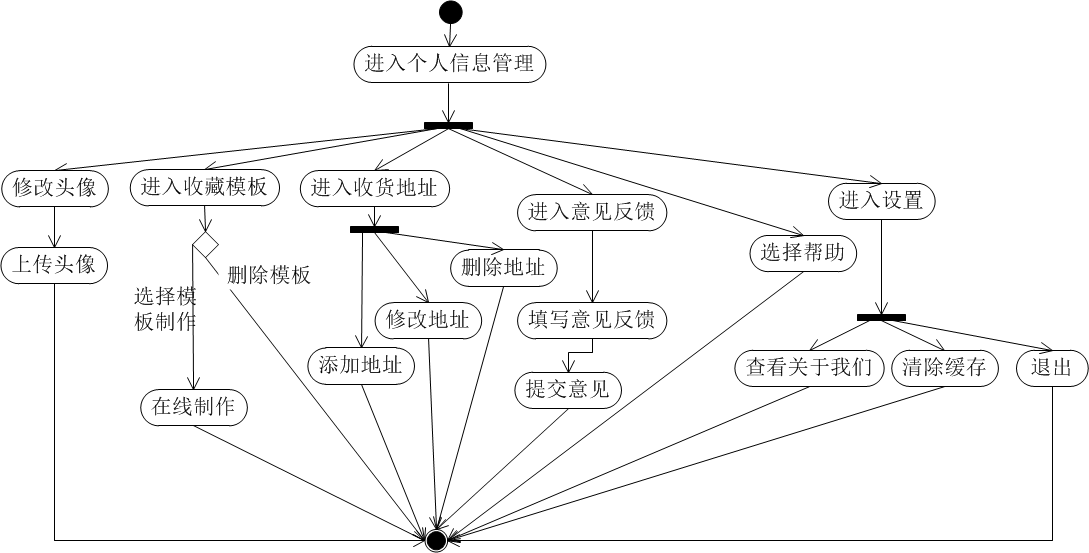
**图4.12** **个人信息模块活动图**

图4.12为个人信息管理模块的活动图，该模块的展示分上下两部分，上半部分为用户的头像及手机号，下半部分为各个功能列表。该部分使用Objective-C中的UITableView来实现页面的展示，上半部分为UITableView的头，下半部分为分组的列表。该模块操作主要是向服务器发起增删改查的动作，使用AFNetWorking进行网络请求，系统中所有图片的加载都使用SDWebImage进行懒加载。

## 4.3 非功能设计

该系统在研究国内外相关个性印刷系统及其相应流程的基础上，分析一部分国内外非常优秀的类似软件产品，针对移动用户构建的一个简单实用、界面良好的个性印刷系统，为用户的个性印刷带去便捷。

不同尺寸屏幕的自适应即是系统健壮性的一种体现，在进行该系统设计的过程中，进行页面的UI设计时，开发者需要注意不同尺寸屏幕的自适应，做好自动布局[31]。在iPhone6上开发的产品在iPhone6plus，iPhone5s，iPhone4s上运行时都能够正常的显示，不会出现布局混乱的现象。iPhone3GS以及之前iPhone的屏幕尺寸是320\*480，即宽320，高是480，没有更多的尺寸，开发非常方便，不需要做自适应，iPhone4S显示屏为Retina，但是屏幕尺寸的大小并没有改变，仍旧是320\*480 point，不需要做特殊处理。但在出现iPhone5S后，它的尺寸变为了320\*568，即高度变大了，为了适应不同屏幕，可以使用iOS系统中提供的autoresizingMask做一些简单的适配，以达到自适应的要求，可以继续开发满足这几个屏幕。但是出现iPhone6，6S及plus后，它的屏幕尺寸变为375\*667和414\*736，宽与高都发生了变化，再使用autoresizingMask就不能满足要求了。使用iOS提供的autolayout或VFL实现布局，可以完成，但是使用起来相对麻烦，本系统的开发使用的是第三方开源库Masonry，Masonry是封装了iOS的VFL实现约束的。Masonry使用起来非常的方便、简单，紧需几行代码就能实现自动布局，有些图标的大小也可按不同屏幕尺寸的比例进行控制。

## 4.4 数据库简要设计

前文功能模块设计时需要获取的数据非常多，客户端几乎不存储任何数据，数据存储在后台数据库中，客户端可以随时获取，也便于更新数据。系统服务器端的数据库采用MySQL[32]，在设计数据库之前需先分析了解用户的需求，系统中会有哪些实体，这些实体分别有哪些属性。本文对服务端数据库的构建进行了简单的分析与设计，同时使用E-R图描述实体关系，E-R分为局部E-R图和综合E-R图。数据库设计不是客户端开发的主要部分，本文只对涉及到的局部E-R图进行分析，如图4.13所示为系统数据库的局部E-R图。

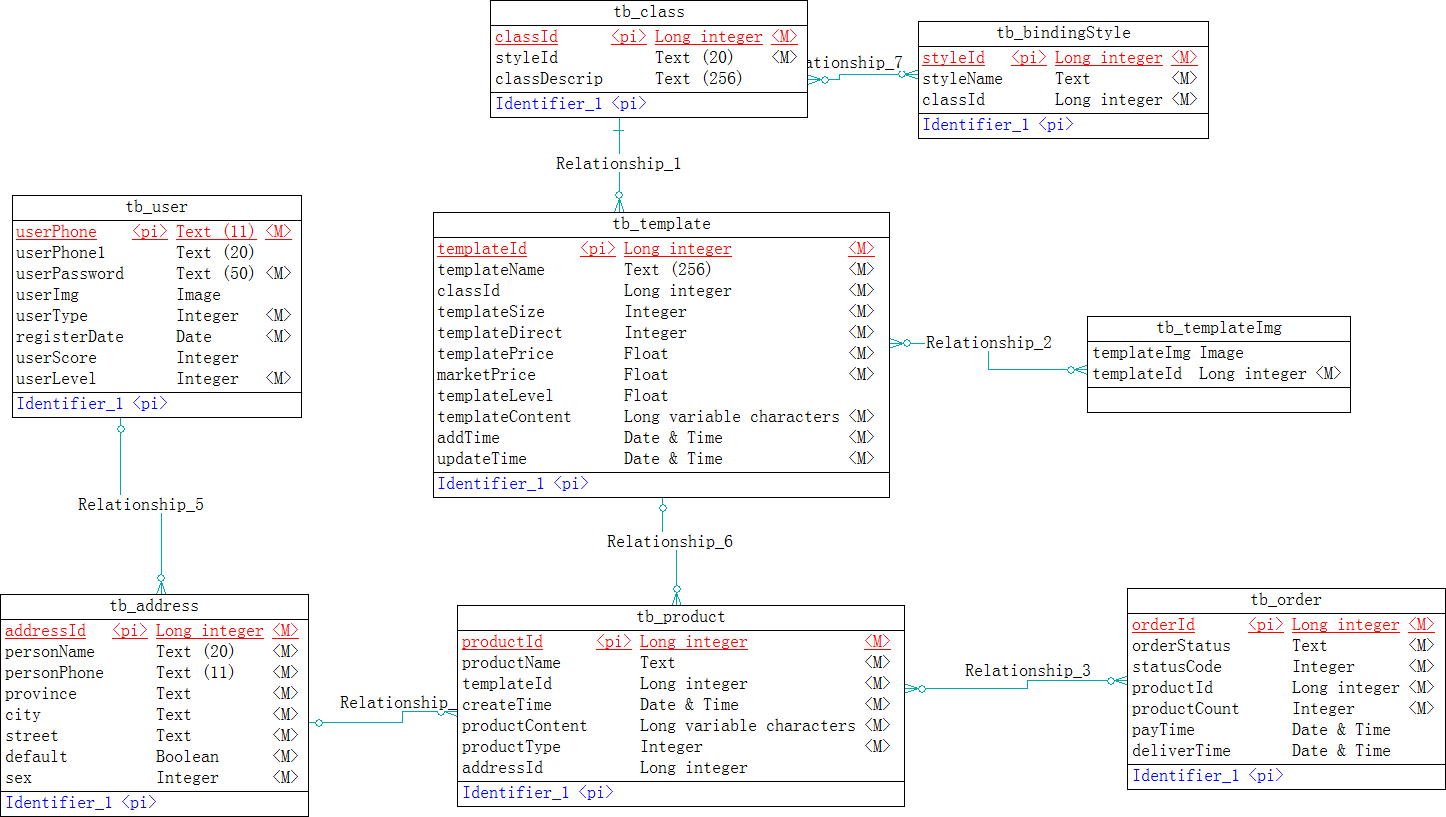
**图4.13** **局部E-R图**

图4.13的E-R图只是进行数据库设计的第一步，之后还要将E-R图转化为逻辑结构，关系数据表。各个表的信息如下：

1. 用户信息表

用户信息表是保存用户信息的表，表结构包括多个字段，包括手机号、用户名、密码、头像、用户类型、注册日期、用户积分以及等级个人信息。在本文中只提到表中userPhone与userPassword两个字段，用户信息表的表结构如表4.1所示。

**表4.1 用户信息表**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 默认 | 主键 | 非空 | 自增 | 备注 |
| userPhone | text | 11 |  | 是 | 是 |  | 手机号码 |
| userName | text | 20 |  |  |  |  | 用户名 |
| userPassword | text | 50 |  |  |  |  | 密码 |
| userImg | longblob |  |  |  |  |  | 用户头像 |
| userType | int |  | 0 |  | 是 |  | 0:用户，1:管理员 |
| registerDate | date |  |  |  | 是 |  | 注册时间 |
| userScore | int |  |  |  |  |  | 积分 |
| userLevel | int |  | 0 |  | 是 |  | 等级 |

1. 分类信息表

分类信息表保存的是所有分类的表，表字段包括分类编号、分类名描述以及模板的装订方式等信息。当有新的分类时，管理员可以向表中插入分类数据。该表的结构如表4.2分类信息表所示。

**表4.2 分类信息表**

| 字段名 | 数据类型 | 长度 | 默认 | 主键 | 非空 | 自增 | 备注 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| classId | bigint |  |  | 是 | 是 | 是 | 分类编号 |
| styleId | bigint |  |  |  |  |  | 装订方式，外键 |
| classDescript | text | 256 |  |  |  |  | 分类名描述 |

1. 模板表

模板表保存的是分类模板信息，该表和分类表、模板图片表以及个性印刷产品表有关联关系。表结构包括的字段有模板编号、模板名称、分类编号、模板尺寸、模板方向、模板价格、模板市场价格、模板评论等级、模板内容以及模板上架时间和更新时间，表结构如表4.3模板表所示。

**表4.3 模板表**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 默认 | 主键 | 非空 | 自增 | 备注 |
| templateId | bigint |  |  | 是 | 是 | 是 | 模板编号 |
| templateName | text | 256 |  |  | 是 |  | 模板名称 |
| classId | bigint |  |  |  | 是 |  | 分类编号 |

**续表**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 默认 | 主键 | 非空 | 自增 | 备注 |
| templateSize | int |  |  |  | 是 |  | 模板尺寸 |
| templateDirect | int |  | 0 |  | 是 |  | 0:横向，1:竖向 |
| templatePrice | float |  |  |  | 是 |  | 本系统价格 |
| marketPrice | float |  |  |  | 是 |  | 市场价格 |
| templateLevel | float |  |  |  |  |  | 模板等级 |
| templateContent | longtext |  |  |  | 是 |  | 模板内容 |
| addTime | datetime |  |  |  | 是 |  | 上架时间 |
| updateTime | datetime |  |  |  | 是 |  | 更新时间 |

1. 产品信息表

产品信息表保存的是用户根据模板定制的个性印刷产品信息，当用户设计完成个性印刷产品后，用户进行下一步操作前都要先向该表中插入设计好的产品信息。该表包含的字段有产品编号、产品名称、模板编号、用户手机号、创建时间、产品内容、产品类型以及地址编号等数据。表结构如4.4产品信息表。

**表4.4 产品信息表**

| 字段名 | 数据类型 | 长度 | 默认 | 主键 | 非空 | 自增 | 备注 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| productId | bigint |  |  | 是 | 是 | 是 | 产品编号 |
| productName | text |  |  |  | 是 |  | 产品名 |
| classId | bigint |  |  |  | 是 |  | 分类编号，外键 |
| userPhone | text | 11 |  |  | 是 |  | 用户手机号，外键 |
| createTime | datetime |  |  |  | 是 |  | 创建时间 |
| productContent | longtext |  |  |  | 是 |  | 产品内容 |
| productType | int |  |  |  | 是 |  | 0:保存，1:订单，2:购物车 |
| addressId | bigint |  |  |  |  |  | 地址编号，外键 |

1. 订单信息表

该表是保存全部用户设计完成并产生订单的数据表，表中包含订单的基本信息，如订单编号、用户手机号、订单状态、状态码、产品编号、购买产品数量、支付的时间以及发货的时间等信息。通过外键产品编号查询产品相关数据，再通过产品表中地址编号能够获取订单的相关数据。订单信息表关联几张其他表，其结构由图4.13可知。表结构如4.5订单信息表所示。

**表4.5 订单信息表**

| 字段名 | 数据类型 | 长度 | 默认 | 主键 | 非空 | 自增 | 备注 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| orderId | bigint |  |  | 是 | 是 | 是 | 订单编号 |
| userPhone | text | 11 |  |  | 是 |  | 手机号 |
| orderStatus | text |  |  |  | 是 |  | 订单状态 |
| statusCode | int |  |  |  | 是 |  | 0:待付，1:待发，2:待收货 |
| productId | bigint |  |  |  | 是 |  | 产品编号，外键 |
| productCount | int |  |  |  | 是 |  | 产品数量 |
| payTime | datetime |  |  |  |  |  | 支付时间 |
| deliverTime | datetime |  |  |  |  |  | 发货时间 |

1. 地址管理信息表

地址管理信息表存储用户添加的地址及联系人信息，用户在下单时会调用后台服务器的接口去查询该表的信息供用户选择，用户能够在客户端对该表进行增、删、改、查操作。地址管理信息表中包括多个字段，包括地址编号、用户手机、联系人、联系人电话、省份、市、街道、邮编以及默认地址字段等信息。该表的表结构如表4.6地址管理信息表所示。

**表4.6 地址管理信息表**

| 字段名 | 数据类型 | 长度 | 默认 | 主键 | 非空 | 自增 | 备注 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| addressId | bigint |  |  | 是 | 是 | 是 | 地址编号 |
| userPhone | text | 11 |  |  |  |  | 手机号，外键 |
| personName | text | 20 |  |  | 是 |  | 联系人姓名 |
| personPhone | int | 11 |  |  | 是 |  | 联系人电话 |
| province | text |  |  |  | 是 |  | 省份 |
| city | text |  |  |  | 是 |  | 市 |
| street | text |  |  |  | 是 |  | 街道 |
| sex | int |  | 0 |  | 是 |  | 0：女，1：男 |
| default | bool |  | false |  | 是 |  | 是否为默认地址 |

1. 装订方式表

装订方式表存储印刷装订方式，与分类表关联，这两个表是多对多的关系，每个分类可以有多种装订方式，每种装订方式可以被多个分类使用。在装订方式表中，存储的字段有装订方式编号、装订方式名及分类编号等信息。该表的表结构如表4.7装订方式表所示。

**表4.7 装订方式表**

| 字段名 | 数据类型 | 长度 | 默认 | 主键 | 非空 | 自增 | 备注 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| styleId | bigint |  |  | 是 | 是 | 是 | 装订方式编号 |
| styleName | text |  |  |  | 是 |  | 装订方式名 |
| classId | bigint |  |  |  | 是 |  | 分类编号，外键 |

1. 模板图片表

模板图片表存储所有模板的预览图片，用户选择一个模板后可以查看模板的效果图，通过模板表中的模板编号查询模板图片表格，从而获取模板的所有预览图。该表中存放的字段为模板图片和对应模板的编号等信息，该表的表结构如表4.8所示。

**表4.8 模板图片表**

| 字段名 | 数据类型 | 长度 | 默认 | 主键 | 非空 | 自增 | 备注 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| templateImg | longblob |  |  |  |  |  | 模板图片 |
| templateId | bigint |  |  |  | 是 |  | 模板编号 |

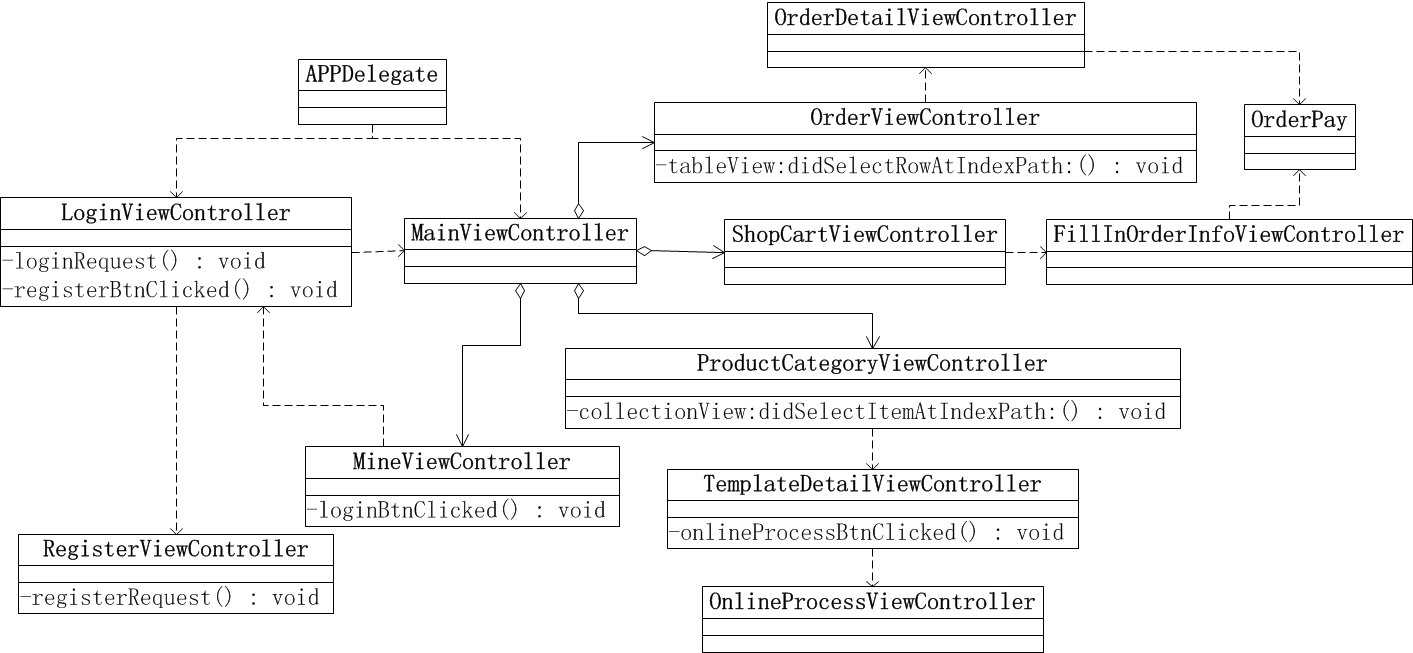
## 4.5 本章小结

本章主要描述该系统的概要设计，划分出移动端的各个模块，同时描述各模块之间的联系。通过列出系统的整体架构图和实现各模块的活动图与流程图，展现出系统整体的结构与流程实现，为后续详细的设计奠定基础。在本章最后对客户端请求服务器涉及到的数据库的几张表进行了简单的设计，保证能够满足前端数据请求。

# 第5章 个性印刷系统的详细设计与实现

移动端个性印刷系统是C/S架构，本章将在第四章的基础上进行更为详细的设计与实现，主要将从两个方面进行介绍，一方面将从代码实现角度详细描述个性印刷系统的设计与实现过程，另一方面简要展示个性印刷系统实现的部分效果。

## 5.1 系统整体设计与实现

本系统主要是为用户提供移动端个性印刷，移动客户端主要分成注册、登录、分类、在线制作、订单管理、购物车、个人信息管理几大模块，各个模块之间的关系如类图5.1所示。

**图5.1** **各模块关系类图**

图5.1为客户端各个模块之间的关系类图，展示客户端的总体类图设计，通过图5.1可以清楚各个模块之间的关联关系与调用关系。各类的作用及调用关系如下：

1. AppDelegate类为系统启动的入口类，该类依赖LoginViewController类与MainViewController两个类。手动登录时会跳转到登录界面，自动登录成功时则直接进入APP的主界面。
2. MainViewController类为系统主界面的视图控制器类，由四个类组成系统的主界面，构成分类模块、订单模块、购物车模块和个人信息模块。这几个模块的主要类为ProductCategoryViewController、OrderViewController、ShopCartViewController和MineViewController，它们与MainViewController类是部分与整体的聚合关系。点击主界面底部各标签按钮进入各自模块。
3. LoginViewController类为登录模块的视图控制器类，调用loginRequest方法登录成功则跳转到系统主界面，点击界面注册按钮时调用registerBtnClicked方法进入注册界面。
4. ProductCategoryViewController类为分类控制器类，展示分类及模板信息，点击模板时调用collectionView:didSelectItemAtIndexPath：方法进入模板详情TemplateDetailViewController控制器类。
5. OnlineProcessViewController类为在线制作控制器类，在该类中完成在线编辑相关操作。调用TemplateDetailViewController中的onlineProcessBtnClicked方法会跳转到对应模板的在线制作模块。
6. OrderViewController类为订单管理控制器类，该类关联视图类展示订单信息。点击一个订单时调用tableView:didSelectRowAtIndexPath：方法进入订单详情控制器类OrderDetailViewController中。
7. ShopCartViewController类为购物车控制器类，该类管理未购买的商品。
8. MineViewController类为个人信息管理控制器类，关联视图类展示收藏模板、收货地址等选项。用户未登陆时点击头像调用loginBtnClicked方法跳转到LoginViewController控制器类中，展示登录界面。

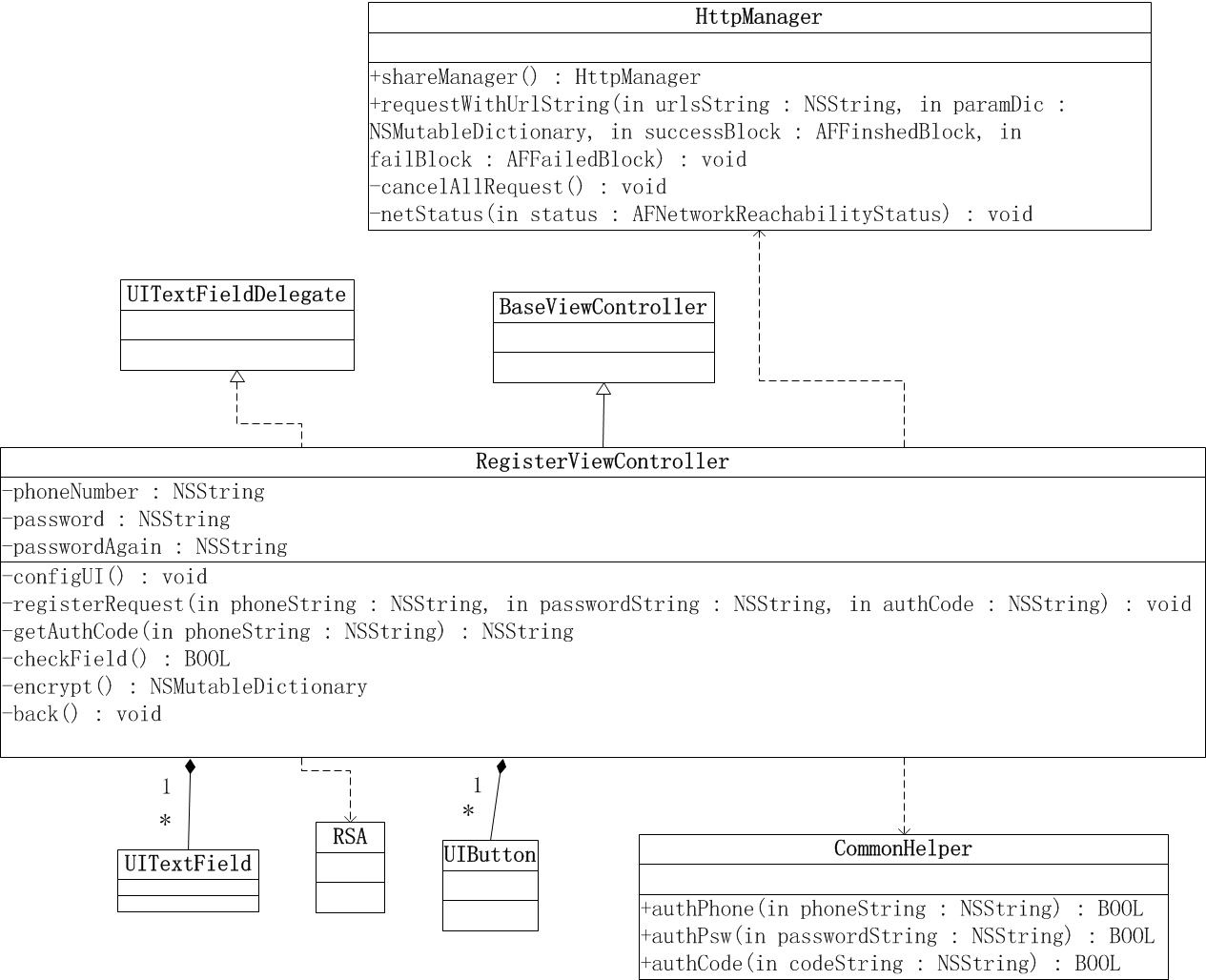
以上为该系统各个模块的主要控制器类，各个模块的详细实现在后续章节进行叙述。

## 5.2 注册模块的设计与实现

注册功能的实现几乎是每个应用都应具备的，在该界面包括的信息主要有手机号、验证码、密码等，在注册时需要用户输入两次密码进行确认。如图5.2所示为注册界面。

由图5.2得知，用户在注册时需要输入手机号、密码以及验证码，输入完所有信息后再验证注册信息的合法性。前端需要控制手机号码的输入只能为数字且为有效的11位手机号码，手机号码合法后，再对输入的两次密码进行验证，密码必须相同，且不能过于简单。若两次密码不相同，则需要重新输入密码，直到验证成功。手机号输入完成后，可以点击发送验证码获取短信验证码，将短信验证码也输入到验证码框中才能进行下一步操作。发起网络请求之前需要对注册的密码进行加密处理，以免被别人盗取。服务端接收到注册请求后向数据库的用户表中插入一条新数据，并向客户端返回注册成功的结果。

**图5.2** **注册界面**

****根据注册界面效果图能够快速了解实现该模块的类，注册模块的类图关系如图5.3所示。

**图5.3** **注册模块类图**

根据图5.3界面及上述描述，可知该模块的核心类为RegisterViewController注册控制器，该类需要继承公共父类完成导航栏的返回等功能。界面由四个输入框，四个按钮及顶部一个图片组成，故UITextField、UIButton及UIImage的对象与注册控制器类是部分与整体的组合关系，同时遵守UITextFieldDelegate协议实现它的代理方法。验证输入的信息时调用公共类CommonHelper中的方法，加密处理需要调用RSA类中的方法，最后发起网络请求时调用公共网络请求类HttpManager中的方法，注册控制器类和这三个类是依赖关系。

1. BaseViewController是通用父类，所有的视图控制器类都继承该类，包括对通用UI的处理以及相关监听器。
2. HttpManager是进行网络请求的通用类，进行注册时需要使用该类的网络请求方法。+(HttpManager \*)shareManager是个类方法，该方法返回HttpManager类的单例对象，保证整个系统中只有一个HttpManager对象实例。requestWithUrlString:(NSString\*)urlString parms:(NSMutableDictionary\*)paramDic successBlock:(AFFinshedBlock)successBlock failBlock:(AFFailedBlock)failBlock方法就是进行网络请求的公共方法，该方法也是类方法，其中urlString就是进行网络请求的地址，paramDic是网络请求的参数，该参数是以一个字典的形式传入，而successBlock与failBlock是网络请求成功与失败的回调块。
3. UITextFieldDelegate是UITextField的协议，类RegisterViewController需要遵守该协议，实现UITextFieldDelegate的中的几个方法，对输入文本进行控制。
4. UITextField是系统的输入文本框类。
5. UIButton类，定义界面上的注册按钮。
6. RSA，RSA为第三方库，该库主要作用是进行加密处理。
7. CommonHelper类是一个封装了很多公共方法的类，很多需要统一处理的地方都在该类中进行了封装。+(BOOL)authPhone:(NSString \*)phoneString是对输入的手机号进行验证的方法，phoneString就是输入的手机号，验证的内容包括输入的长度，手机号的格式，采用Objective-C中的正则表达式进行验证。同理，+(BOOL)authPsw:(NSString \*)passwordString是验证密码的方法，+(BOOL)authCode:(NSString \*)codeString，都是采用正则表达式进行验证。
8. RegisterViewController类是注册控制器，继承BaseViewController，该类封装了注册相关的所有方法。

进入到注册模块后，系统自动调用-(void)configUI方法进行配置注册界面的UI，设置各种代理方法，当输入手机号时会触发UITextField的代理方法，然后调用+(BOOL)checkField方法检查输入的手机号是否正确，在checkField中调用的校验方法是CommonHelper类中的authPhone。输入手机号后点击发送验证码按钮会触发getAuthCode方法，该方法以手机号作为参数向服务器发起请求。输入完所有的信息后点击注册按钮调用registerRequest方法，在该方法中调用checkField方法对两次密码进行比较。若密码相同，则调用RSA类中的encryptString:(NSString \*) publicKey:(NSString \*)方法对用户的信息进行加密处理，将返回的值作为请求参数再去调用公共网络请求管理类HttpManager中的requestWithUrlString:(NSString\*)urlString parms:(NSMutableDictionary\*)paramDic successBlock:(AFFinshedBlock)successBlock failBlock:(AFFailedBlock)failBlock方法向服务器发起网络请求，根据返回值提示用户。

## **iOS Simulator Screen Shot Oct 22, 2016, 12.47.02 AM**5.3 登录模块的设计与实现

根据需求分析，只有先注册才能使用登录功能。实现登录界面如图5.4所示。

**图5.4** **登录界面**

在登录界面只有两个输入框，即用户手机号输入框与密码输入框，输入手机号与密码后客户端先对这两个字段做一个简单的前台验证。与注册时相同，验证手机号的合法性，以及密码的复杂度，若这两个有一个不满足用户都是无法向服务端发起网络请求。输入成功后，再调用本地加密方法对密码加密，然后发起网络请求。由图5.4可知，该模块实现的主要功能就是输入已注册的用户信息，登录进入系统。

在界面上有两个输入框使用UITextField完成，同时具有三个UIButton按钮，忘记密码、立即注册为透明按钮，登录按钮具有黄色边框，它们组成登录界面。登录模块的核心类为LoginViewController，与注册模块类似，需要继承公共父类BaseViewController，实现协议UITextFieldDelegate的相关方法，同时依赖RSA、CommonHelper及HttpManager类进行数据加密、验证以及网络请求。在该模块的实现中，我们不仅实现登录界面，同时也要完成登录功能的实现，包括加密处理，网络请求以及信息保存等。构造得到该模块的类图如图5.5所示。

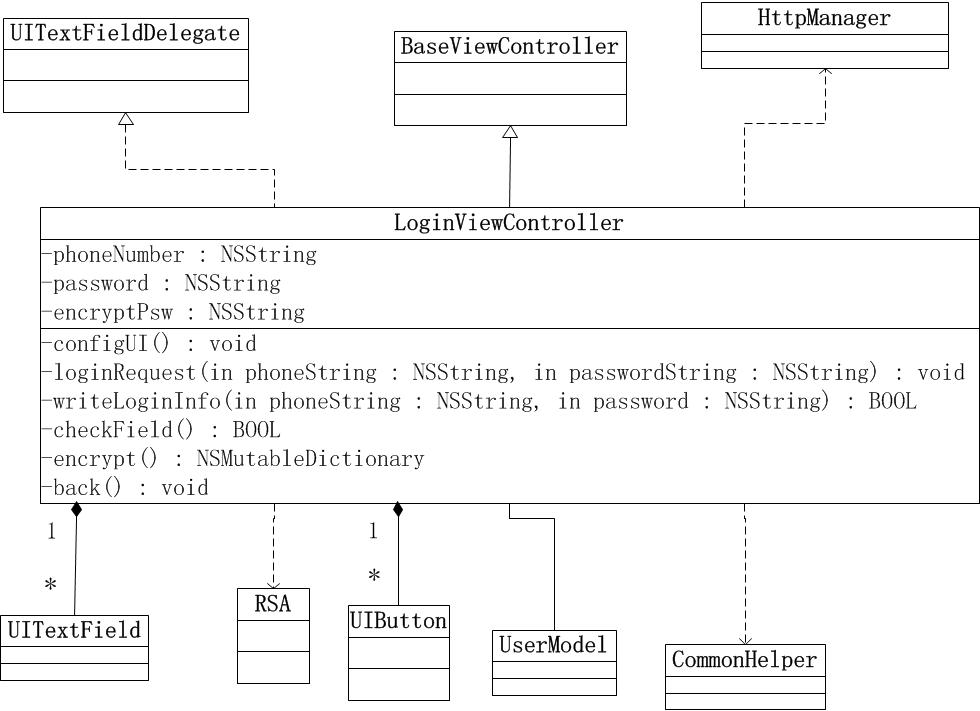
**图5.5** **登录类图**

图5.5是根据实现登录功能需要的类及类之间的关系构造的类图，在LoginViewController类中首先调用configUI方法配置登录界面的UI，同时初始化变量。输入手机号与密码后都会调用checkField的方法验证合法性，点击登录按钮时触发loginRequest方法，该方法与服务器交互，向服务器发送登录请求。在方法内部会调用checkField方法验证手机号与密码的合法性，然后再去调用encrypt方法对登录数据进行加密处理，待所有操作完成后再去调用HttpManager中的requestWithUrlString: parms: successBlock: failBlock:方法进行网络请求。loginRequest方法会对请求回来的数据进行解析，将返回的用户数据保存在UserModel类型的变量中，判断登录成功还是失败，若登录成功则再调用方法writeLoginInfo方法将登录数据存储在本地文件中，再打开APP时会自动登录。

encrypt方法调用RSA的方法对密码进行加密，iOS端使用第三方库RSAUtil进行加密，该库在github上非常受欢迎[33]。在进行加密前，前端需要获取公钥，公钥由服务端生成之后保存在前端。在encrypt方法中直接调用RSAUtil库中的RSA类的encryptString:(NSString \*) publicKey:(NSString \*)方法进行加密，传入要加密的字符串和公钥，该方法返回一个加密的字符串。注册和登录时对密码进行加密都是同样的方法。

## 5.4 分类模块的设计与实现

### 5.4.1 分类页面的设计与实现

该模块主要显示个性印刷系统支持的哪些分类产品，每一种分类对应一些个性印刷模板，通过缩略图可以浏览喜欢的模板信息。

分类模块主要分成三个部分，左部为分类信息的展示，用户可以根据需求选择喜欢的分类；右部为相应分类对应的所有个性印刷模板，展示模板的相关信息；上部为筛选和排序选项，用户可以通过这几个功能快速的找到自己想要的模板。分类模块的界面如图5.6所示。

**图5.6** **分类界面**

由图5.6可知，该模块主要由三个部分组成，左半部分为分类列表信息，使用UITableView实现，顶部则是筛选和排序功能，使用UIButton实现，而右半部分则显示每种分类对应的模板，主要使用UICollectionView实现。这三个系统类与产品分类控制器ProductCategoryViewController的关系为部分与整体的组合关系，同时依赖TemplateCell显示模板信息，该类为继承UICollectionViewCell的视图单元。该类实现UITableView与UICollectionView的代理类方法完成数据的展示，依赖HttpManger请求数据，得到数据存放在产品分类模型ProductCategoryModel和产品模板模型ProductTemplateModel的对象中，ProductCategoryViewController依赖这两个类，他们之间为关联关系。由上图界面效果及各类之间的关系得到该模块的类图5.7所示。

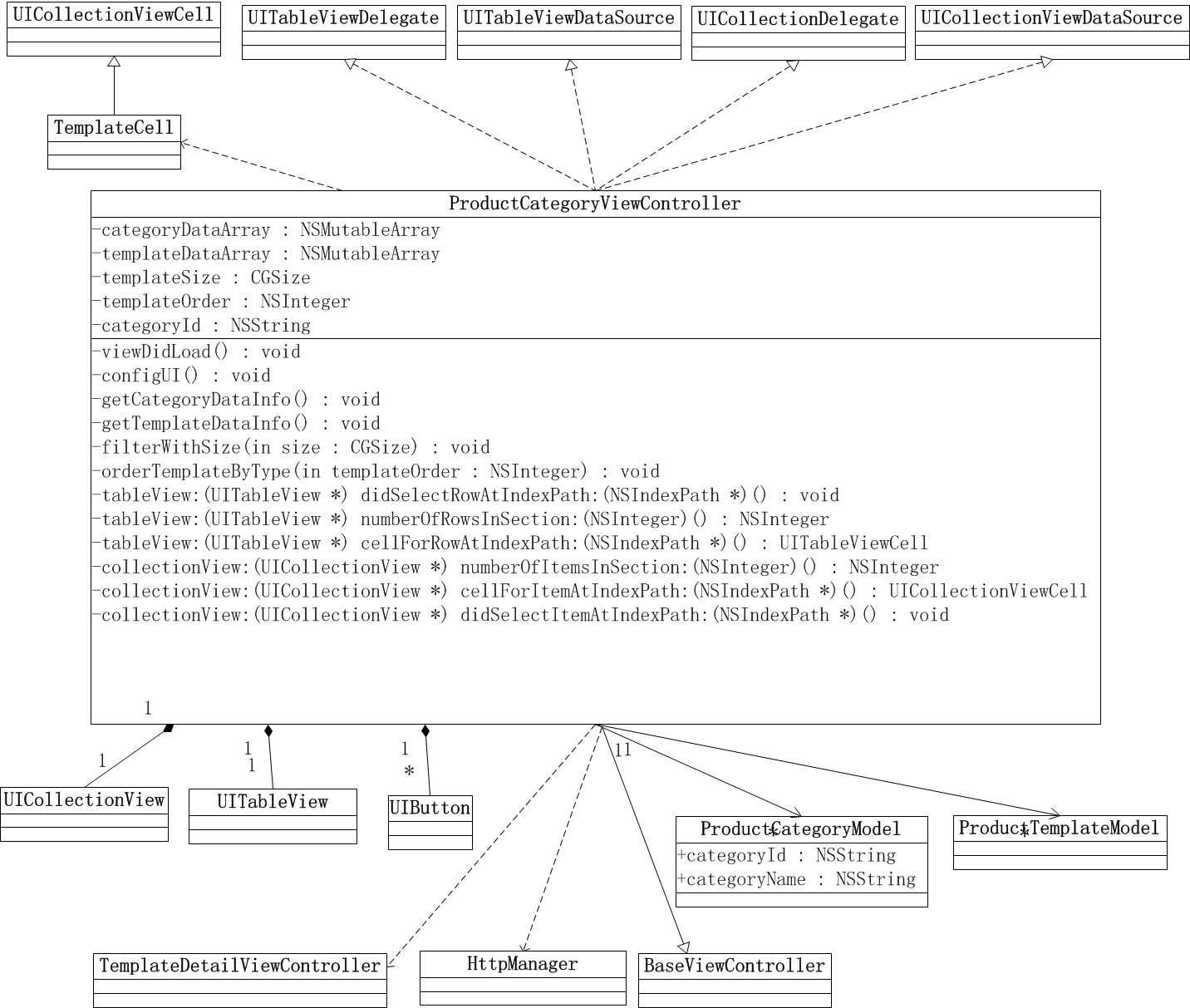
**图5.7** **分类模块类图**

图5.7为分类模块主要类之间的关系图，这个模块中主要包括请求数据、数据模型、数据展示这几块。ProductCategoryModel和ProductTemplateModel是两个数据模型，分别存放从服务端查询的分类数据和模板信息数据。

该模块最重要的一个类就是ProductCategoryViewController，该类就是MVC模式中的对应的控制器部分，连接着数据模型和视图显示。用户点击客户端底部分类标签时就开始对该页面进行初始化，调用该类中系统自带的viewDidLoad方法，其他调用方法依次如下：

1. 加载分类栏的UI界面，configUI方法主要实现分类模块UI界面展示效果，同时对类中将要使用到的UITableView与UICollectionView代理进行设置，初始化变量categoryDataArray与templateDataArray两个数组，注册UITableView与UICollectionView需要显示的Cell。
2. 获取产品分类信息，getCategoryDataInfo方法，该方法在进行初始化操作时就被调用，便于户进来就能看见本系统有哪些分类。getCategoryDataInfo方法内部会调用HttpManager中的网络请求方法去从服务器获取数据，服务端的数据以JSON格式返回给客户端，解析返回的数据对应本地已建立的数据模型ProductCategoryModel。一次请求返回的数据条数取决于设置的参数，将解析的ProductCategoryModel对象存储到categoryDataArray数组中，数据请求结束调用系统方法reloadData更新UI展示数据。
3. 产品分类信息的展示，调用reloadData方法后，系统自动调用代理方tableView:(UITableView \*)tableView numberOfRowsInSection:(NSInteger)section与tableView:(UITableView \*)tableView cellForRowAtIndexPath:(NSIndexPath \*)indexPath。前一个方法返回categoryDataArray数组的个数，即有系统有多少分类，后一个方法负责将categoryDataArray数组中对应的ProductCategoryModel数据模型中的数据赋值给TableView对应的Cell，最终显示在页面上。用户点击显示出来的产品分类时系统会调用UITableView的另外一个代理方法tableView:(UITableView \*)tableView didSelectRowAtIndexPath:(NSIndexPath \*)indexPath，在该代理方法中根据indexPath从categoryDataArray中获取对应的数据模型，将该模型中的categoryId字段作为参数向服务端再次进行数据请求服务获取模板信息。
4. 模板信息展示，从(3)中获得了categoryId后继续调用getTemplateDataInfo方法，该方法将categoryId作为参数请求服务器获取对应的的模板数据。同getCategoryDataInfo相似，请求回来的数据解析成对应的ProductTemplateModel数据模型，将所有的ProductTemplateModel对象存放在templateDataArray数组中，然后去更新UI。方法collectionView:(UICollectionView\*)collectionView numberOfItemsInSection:(NSInteger) section返回templateDataArray数组中模板数据的个数，collectionView:(UICollectionView \*)collectionView cellForItemAtIndexPath:(NSIndexPath \*)indexPath方法根据数组templateDataArray中的模型对象实例化TemplateCell，显示在UI界面上。TemplateCell是继承UICollectionViewCell的自定义类。获取模板信息的接口时分页查询的，用户指定pageIndex和pageSize的值，向服务器请求数据，返回pageIndex页的pageSize条数据。每次上拉CollectionView时pageIndex加1再向服务器请求数据，加载更多数据。collectionView:(UICollectionView \*)collectionView didSelectItemAtIndexPath:(NSIndexPath \*)indexPath是选择模板方法，当用户点击个性印刷模板时就会调用该方法，在该方法中会获取对应模板的编号templateId，编号作为变量传递，然后跳转到模板详情界面。
5. 筛选和排序，用户也可根据模板的尺寸大小进行筛选，调用filterWithSize:(CGSize)方法，传入需要的尺寸，请求后台获取符合尺寸的模板。同时用户可以调用orderTemplateByType:(NSInteger)方法，按价格、上架时间及更新时间进行排序。服务器返回的数据存放在客户端的数组中，通过UICollectionViewCell展示所有的信息，包括模板名称、方向、分类名及价格等信息。

### 5.4.2 模板详情页面的设计与实现

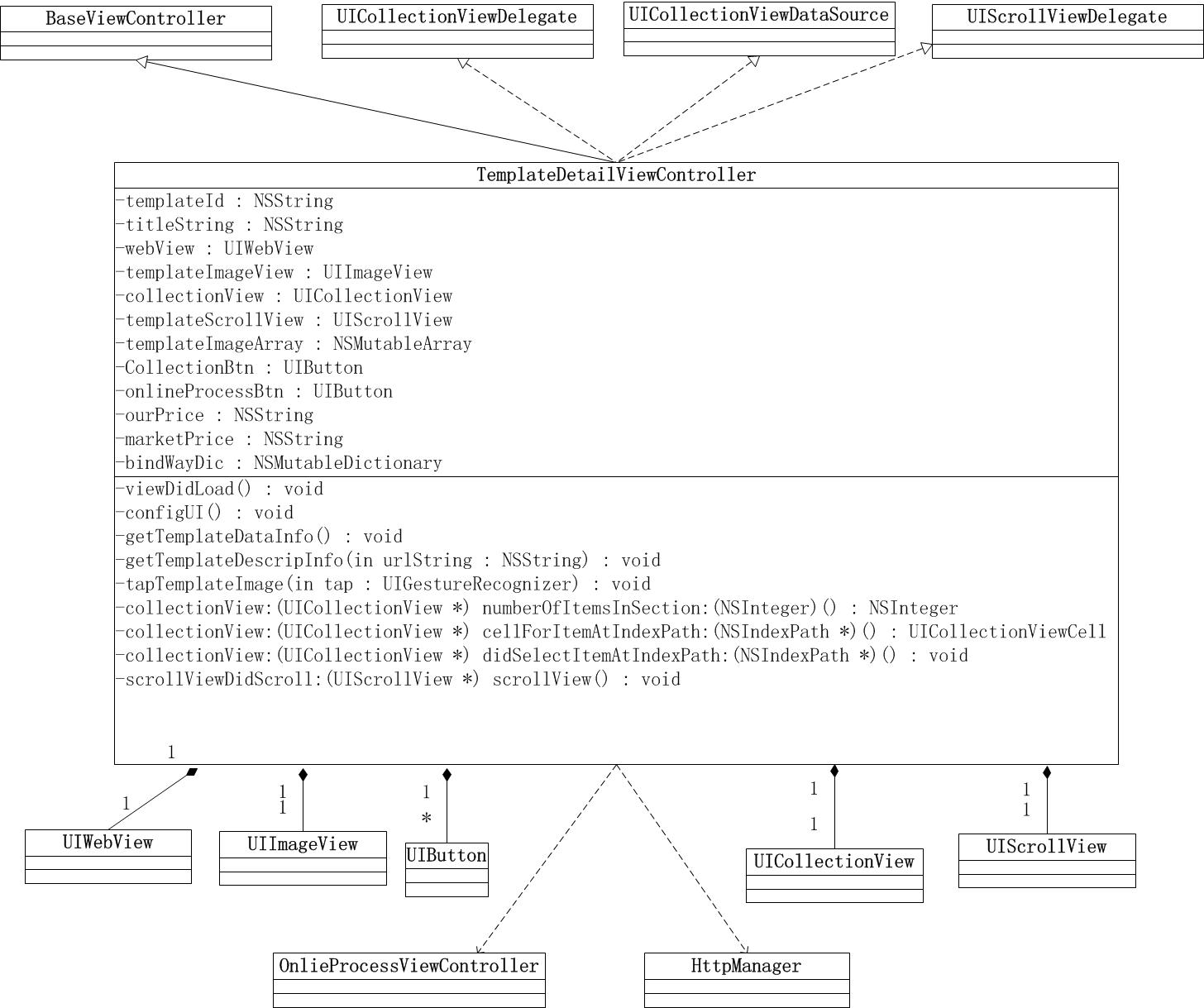
模板详情界面主要展示模板的描述信息，价格以及模板图片，在该界面可以进行交互，用户可以收藏模板，浏览模板图片，同时可以进入到在线制作界面进行制作。

该模块的核心实现类为TemplateDetailViewController，同理，该类需要继承公共父类完成导航栏等功能，该类由UIImageView、UIWebView、UIButton、UICollectionVIew及UIScrollView组成，它们与TemplateDetailViewController组合关系，这几个类的主要功能是展示模板预览图、模板详情描述及收藏和在线制作。同时需要依赖HttpManger类的网络请求方法，向服务器请求模板详情信息，最后实现UICollectionView和UIScrollView的代理方法完成模板预览图的展示。从模板详情界面能够跳转到在线制作界面，故TemplateDetailViewController类与在线制作类OnlineProcessViewController之间为依赖关系。

该界面主要分四个部分，上半部分有左右两块组成，左边为模板的图片展示区，底部是一个CollectionView展示模板的小图可以预览，右边是模板的信息，包括价格、名称、星级、装订方式、收藏以及在线制作按钮。下半部分是使用WebView展示的模板描述信息。由这些类之间的关系得到的类图如图5.8所示。

图5.8类图中只列举出该类涉及到的主要属性和方法。templateId是前一页传递过来的模板编号，TemplateDetailViewController类进行初始化加载视图时就会调用getTemplateDetailInfo方法向服务器发送网络请求获取模板基本信息，其中templateId就是作为网络请求的参数。TemplateDetailViewController需要实现UICollectionViewDelegate、UICollectionViewDataSource及UIScrollViewDelegate几个协议中的方法完成一些特殊功能。TemplateDetailViewController中的方法与属性作用分别如下：

1. 网络请求方法getTemplateDetailInfo，该方法内部会根据传入的templateId向服务端发送请求，请求返回的数据为JSON格式。内部对数据进行解析，分别将模板预览图放在templateImageArray数组中，使用UICollection展示，价格存放在ourPrice与marketPrice变量中，而对应的装订方式存放在bindWayDic字典中。待该网络请求数据成功后，会获取JSON数据中的一个key，descripUrl，这个key就是关于商品描述的url。然后会使用这个url作为参数调用getTemplateDescripInfo方法加载数据，和前面的不同是这部分直接使用webView，不再绘制描述的UI，这样既简单又方便。

**图5.8** **模板详情模块类图**

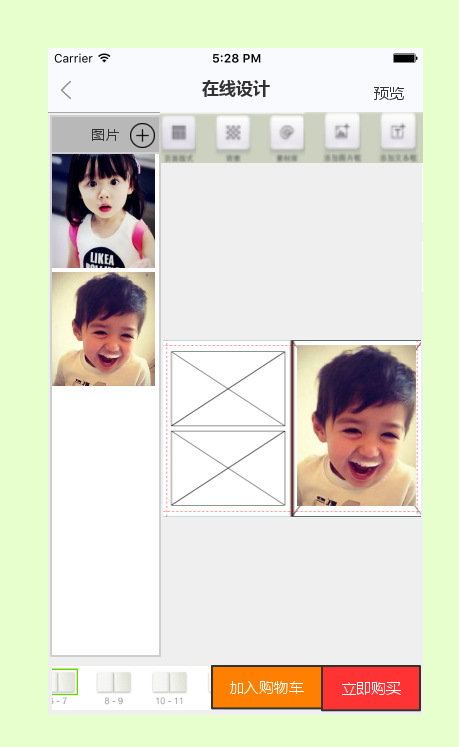
1. 视图滚动方法scrollViewDidScroll，该方法是UIScrollView的代理方法。请求得到的模板预览图需要放在templateScrollView上进行放大展示，当用户去滚动这个视图上的图片时就会调用该方法。
2. 点击缩略图放大与缩小，滚动视图上每一张照片都添加了手势，用户用手指点击缩略时就会调用tapTemplateImage方法，该方法主要作用是放大与缩小。当点击图片时方法内部实例化一个新的size更大的视图对象，用来显示图片，再次点击时该对象销毁。

收藏按钮和在线制作按钮的响应方法相同，根据按钮的tag值来区分。点击收藏按钮时会将该模板的templateId插入到后台数据库的收藏表中，供以后查询使用。点击在线制作按钮时将templateId赋值给OnlineProcessViewController对象的对应的属，再调用系统方法pushViewController:animated方法跳转到在线制作页面。

## 5.5 在线制作模块的设计与实现

在线设计模块从模板详情页进入，该页面布局分为上下左右四个部分，上部分为编辑选项，下部分为印刷品页面快速选择区，左部为导入的图片展示区，右部为编辑区域。

该模块实现相对复杂，不仅在界面上需要实现，在功能上更加复杂，需要实现的功能有在线制作，图片处理，文字处理，模板布局XML的设计。同时需要导入相册中图片以及完成下单或者加入购物车。在线制作界面如图5.9所示。

**图5.9** **在线制作界面**

该界面的实现涉及到UICollectionView、UIScrollView、UIButton以及UIImage等类，该模块的核心类为OnlineProcessViewController。从界面的构建可知，该界面由UICollectionVIew、UIScrollView、UIView、UIButton和UIImageView组成，这些系统类与OnlineProcessViewController类是部分与整体的聚合关系，因为这些类还被其他整体共享，并不随着整体的破坏而破坏。若完成UICollectionView与UIScrollView的功能OnlineProcessViewController类需要实现这两个类相应协议的方法，所以该类与UICollectionViewDelegate、UICollectionViewDataSource及UIScrollViewDelegate的关系是实现关系。除了界面的实现，更主要的是功能，在进行编辑时涉及到XMLLayout、ImageProcess与TextProcess类，它们分别为封装解析XML文件、编辑图片与文字的类，可以独立存在，与在线制作控制器类的关系为聚合关系。

根据以上对该模块各类的分析，构造出的类图如图5.10所示。

**图5.10** **在线制作类图之间关系**

在线制作部分关联了太多的类，图5.10只列出关键部分的相关类，其中OnlineProcessViewController与其余三个类ImageProcess、XMLLayout和TextProcess是一种聚合关系，这三部分可以单独存在，不受OnlineProcessViewController影响，当点击预览的时候XMLLayout的对象会作为PreviewProduct的参数传递过去，便于查看定制的印刷品，故预览类PreviewProduct是依赖XMLLayout的。

OnlineProcessViewController加载视图时会调用viewDidLoad方法，在该方法中进行数据的初始化及调用相关方法加载数据绘制界面。

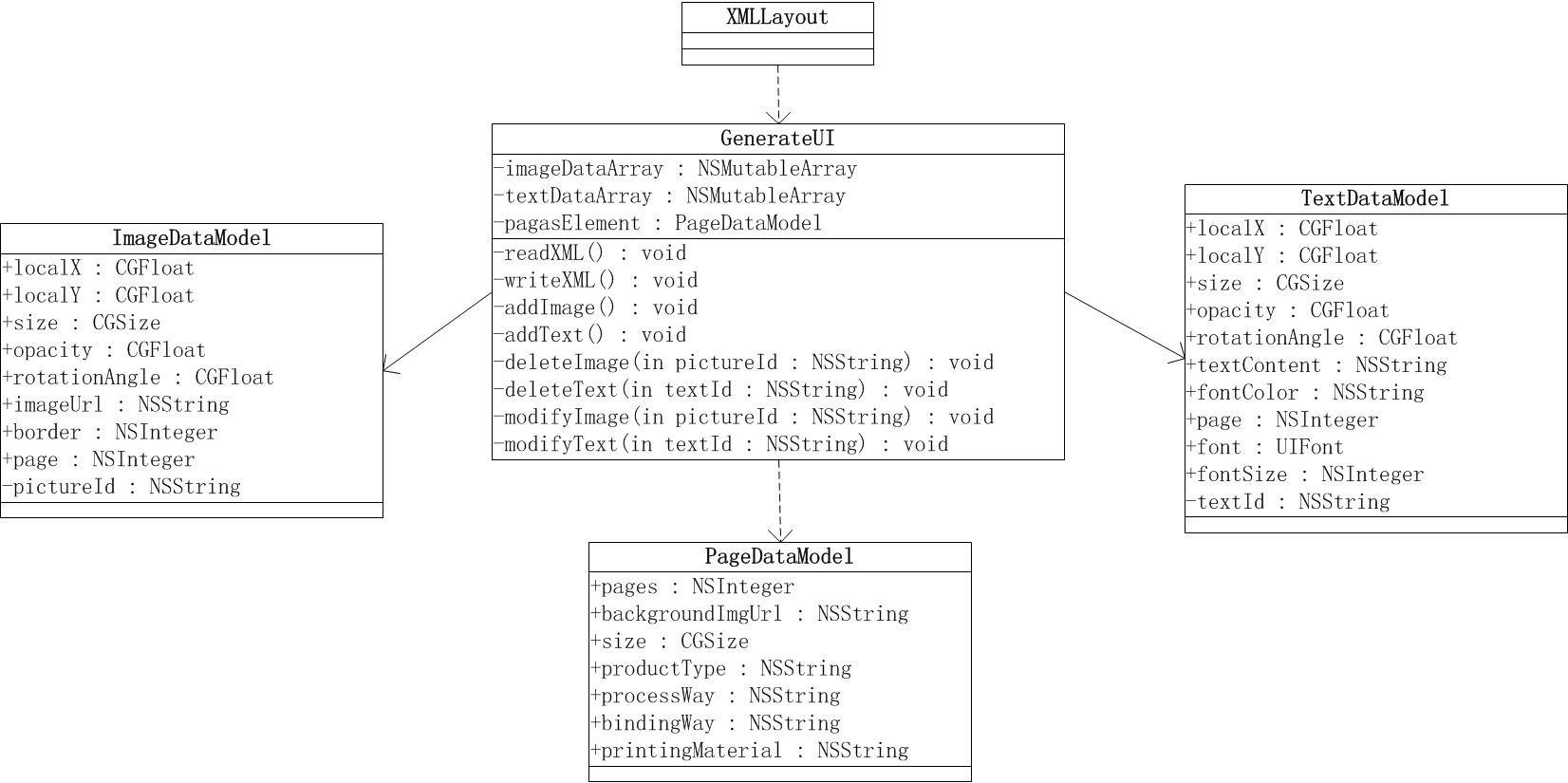
1. 导入图片的设计与实现

进入该模块后，通过点击“+”按钮调用OnlineProcessViewController类中的importImages方法打开系统相册。导入图片功能使用了Photos和AssetsLibrary框架，在importImages中使用GCD方法加载系统的所有相册。调用ALAssetsLibrary类中的[library enumerateGroupsWithTypes:ALAssetsGroupAll usingBlock:assetGroupEnumerator failureBlock:assetGroupEnumberatorFailure]方法读取相册，因为相册中的图片具有位置信息，所以在访问系统图片时，系统会弹出询问框“是否允许访问位置信息”？如果选择否，就会回调执行assetGroupEnumberatorFailure块中的代码，同时告知用户失败。ALAssetsGroupAll这个值是表示获取全部相册，把每个相册列出来，然后再去选择某个相册中的图片。

可以对ELCAssetTablePicker中的assetGroup设置过滤，如下设置为只读取图片，如[picker.assetGroup setAssetsFilter:[ALAssetsFilter allPhotos]]。然后调用preparePhotos读取相册中的所有图片，然后就是读取照片信息，调用ALAsset类中的thumbnail方法，thumbnail是图片的缩略图，这样加载的照片是缩略图，而不能是巨大的原图，可优化内存。

1. 编辑印刷品的设计与实现

客户端根据事先与服务端约定好的规则解析XML文件，根据文件中的元素、属性以及从属关系在视图上添加图片框和文本框，展示给用户。解析XML 文件使用谷歌的开源框架GData，该框架使用非常方便快捷，直接在工程中引入该框架，或者使用Cocoapods进行自动管理。解析完模板并展示出布局后可以开始编辑操作。图5.11为布局界面生成与存储的简单类图关系。

**图5.11** **布局界面生成与存储类图**

XML文档结构是一种树形结构，能够清晰的表达客户端的界面构成，XML 有且只有一个根节点，它记录着该文档的一些信息。进入到在线制作模块后系统会读取XML数据解析模板，XMLLayout依赖GenerateUI类进行界面和XML数据的转换。方法readXML读取XML模板进行解析，先定位到文档的根节点，然后按照深度优先遍历的顺序遍历所有的子节点[34]，客户端遍历是根据节点的类型映射到iOS界面，遍历过程中遇到节点名称为pagesGlobal时将对应的页面设置信息转换成PageDataModel对象，pictureType类型为图片信息，转换成ImageDataModel对象存储在imageDataArray数组中，textType为文本信息，转换成TextDataModel对象存储在textDataArray数组中。然后根据这些信息实例化图片和文本对象添加到界面上展示给用户。完成所有的操作后再调用writeXML方法将数组中的数据写入到XML文件中。

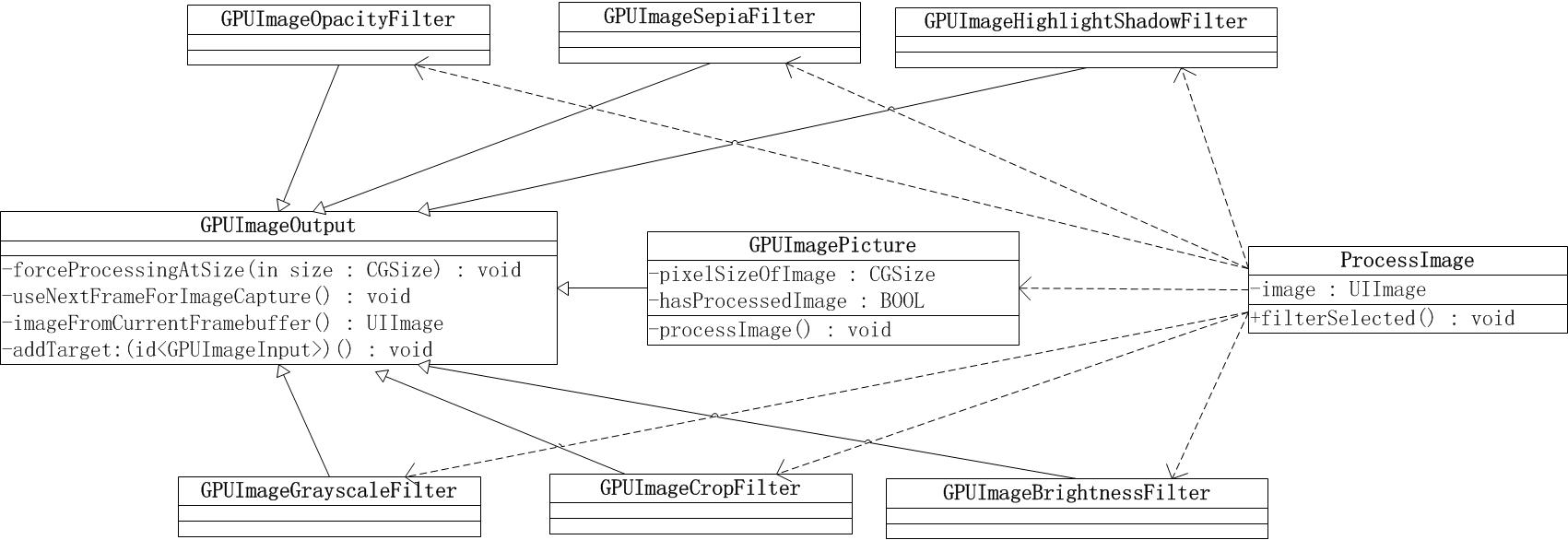
1. 方法addImage向界面中添加一个图片，当添加新图片时，调用该方法创建一个新的ImageDataModel对象及ImageView对象。ImageDataModel对象添加进imageDataArray数组中，同时调用writeXML方法将图片信息写入到XML文件中。同理，addText方法向界面中添加一个文本。
2. 方法deleteImage根据传入的pictureId将ImageDataModel对象从imageDataArray数组中删除，同时将XML文档中的该节点删除。同理，deleteText删除一个文本。
3. 方法modifyImage根据传入的pictureId对imageDataArray数组中ImageDataModel对象进行修改，同时修改XML文档中该节点信息。同理，modifyText为修改文本信息的方法。图5.12为空白模板文档。

**图5.12 XML个性印刷模实例图**

图5.12是一个简单的个性印刷空白模板文档，该文档记录了模板如何布局，记录文档中只有图片和文字，根元素下只有三种类型元素，分别表示全局页面设置、图片和文字。图片和文字两个元素下还有子元素，这些子元素按顺序存储，他们就是模板中图片和文字的布局信息。

1. 图片处理模块的设计与实现

该模块主要实现对图片进行编辑的功能，即本文摘要中提到的对图片做滤镜处理。本课题对图像的处理主要使用第三方GPUImage框架，该框架封装了丰富的处理方法，本系统并不是专业处理图片的软件，故该框架中的处理方法对本系统来说已经搓搓有余。进行图像处理的类为ProcessImage，该类依赖GPUImage框架中的几个类，类图关系如图5.13所示。

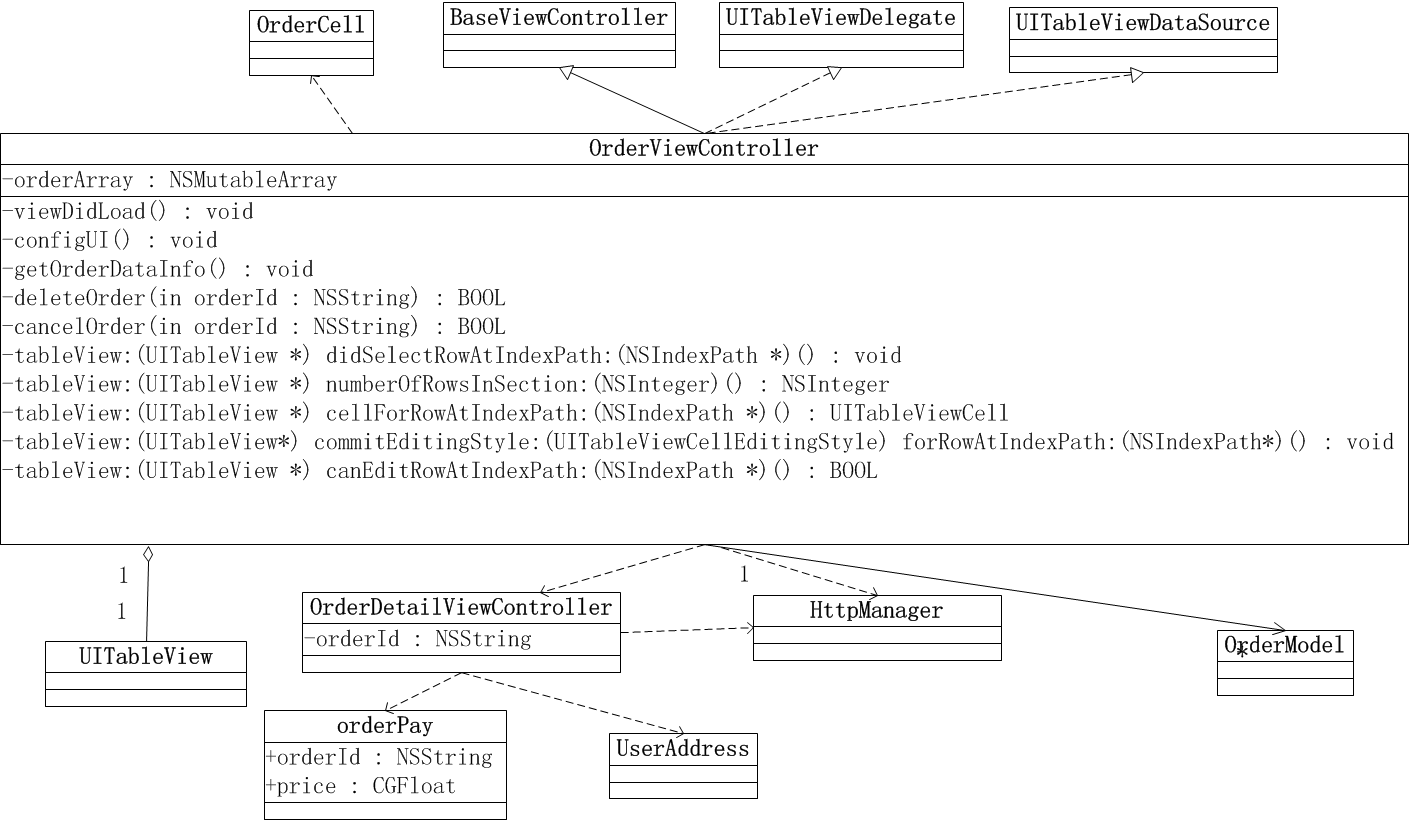
**图5.13** **图像处理类图关系**

该类图实现图像的编辑功能，以图像增加亮度GPUImageBrightnessFilter为例，解释ProcessImage类是如何使用GPUImage中处理图片的相关类。用户点击图片后选择一个滤镜进行处理，则调用该类中的filterSelected方法，在该方法中会声明一个GPUImageBrightnessFilter的对象brightnessFilter，然后设置该brightnessFilter的亮度值，再调用父类中的forceProcessingAtSize和useNextFrameForImageCapture方法设置渲染区域。之后再使用点击的图片作为参数实例化GPUImagePicture的一个对象，该对象再调用addTarget:方法添加滤镜，将brightnessFilter作为参数传入，完成上述所有步骤后调用processImage方法进行渲染。渲染成功后调用brightnessFilter的imageFromCurrentFramebuffer方法获取渲染后的图片。GPUImageCropFilter类是对图片进行剪切的类。

## 5.6 订单模块设计与实现

订单模块主要作用是管理用户的订单信息，在该模块中用户可以查看订单的状态信息，对订单进行各种操作。包括查看订单支付状态，若未支付，则可以进行支付、删除订单，修改联系人信息以及对订单的地址信息进行增、删、改、查操作。该模块的界面主要使用UITableView实现，UITableView可以被多个整体共享，不受整体限制，与订单控制器OrderViewController关系为聚合关系，订单单元格需要继承UITableViewCell类进行自定义，订单模块的类图如图5.14所示。

图5.14为订单模块的简要类图，用户根据订单的状态进行相应的操作，订单未完成支付时，用户可以进入订单详情点击界面上的支付按钮，这时会触发OrderDetailViewController类中的payOrderWithId:price方法，将订单编号与价格传递给OrderPay中对应的方法，在该方法中调用第三方支付软件进行支付。未完成的订单，可以修改地址信息，调用类中的addNewAddress、deleteAddress、modifyAddress、及searchAddress方法来修改地址信息。可以编辑购买的数量或者继续编辑个性印刷产品，获取订单对应的产品编号跳转到在线制作页面。对已完成支付的订单，可以查看订单的物流，使用UITableView来动态展示。

**图5.14** **订单模块类图**

## 5.7 购物车模块设计与实现

### 5.7.1 购物车页面的设计与实现

购物车模块用于统一管理消费者选购的所有个性印刷品，图5.15所示。

**图5.15** **购物车界面**

购物车模块界面的实现主要使用自定义的UITableViewCell以及UIView和UIButton实现的结算界面UITableView与UIButton类与购物车视图控制器是部分与整体的聚合关系，同时该类UITableViewCell的子类ShopCartCell类实现商品的展示。与其他控制器类相同，该类使用UITableView，故需要实现该类的代理方法，根据以上分析构造得到购物车主要类图如图5.16所示。

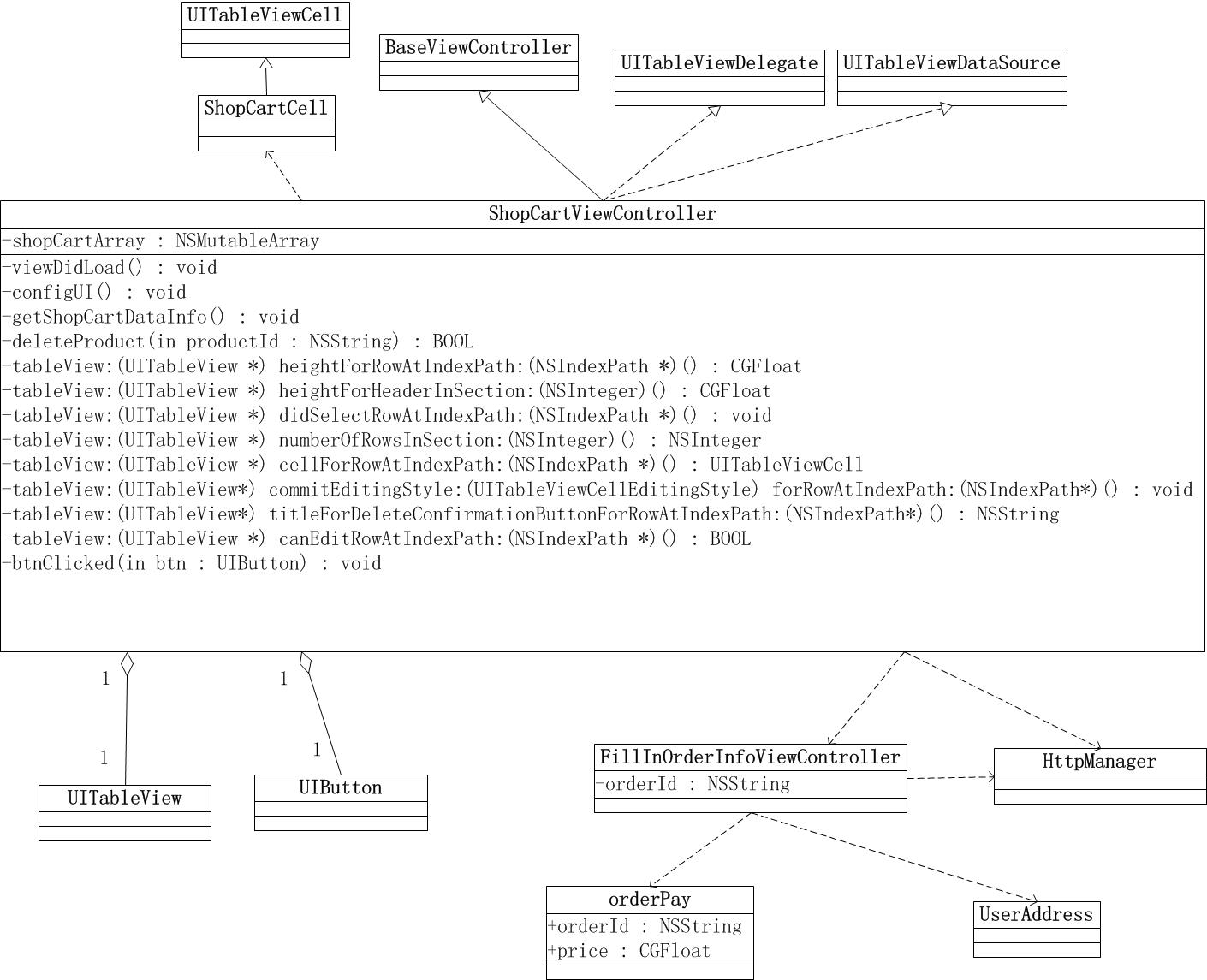
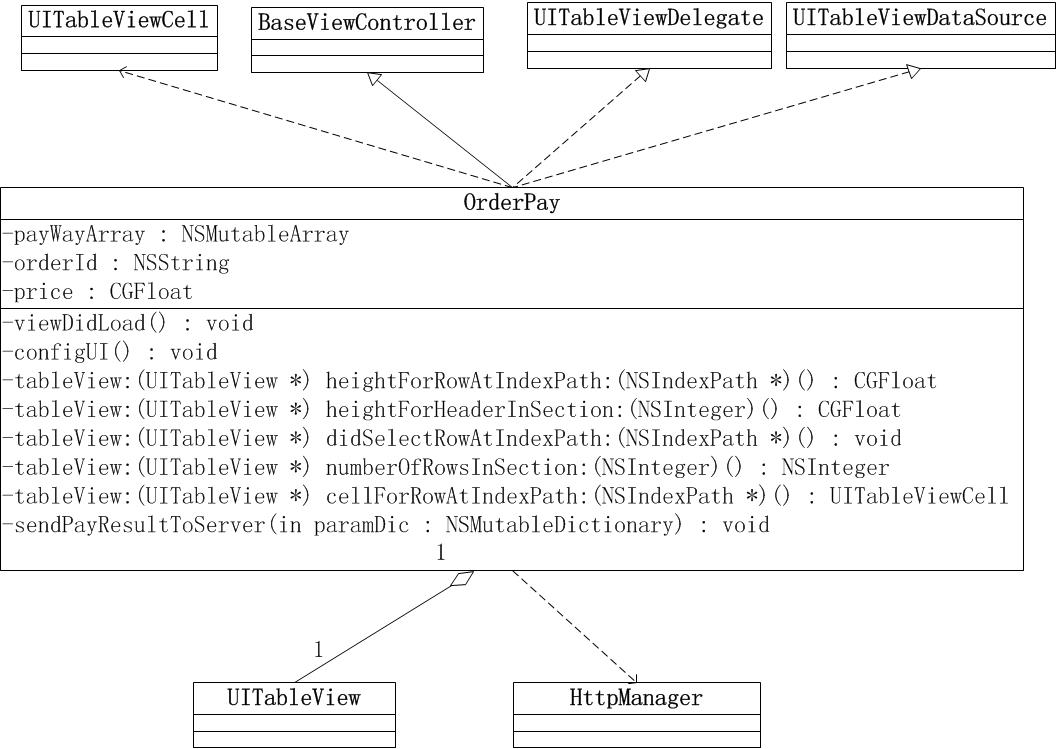
**图5.16** **购物车类图**

图5.16是实现购物车模块的界面与功能的类图，这些类的设计如下：

1. ShopCartViewController依赖ShopCartCell类来实现商品的展示，是实现购物车模块的关键类。方法configUI用来绘制购物车界面，然后调用getShopCartDataInfo方法从服务器获取购物车信息，在该方法中需要使用HttpManager的网络请求方法，获取到的数据存入到shopCartArray数组中，然后展示在ShopCartCell上。
2. ShopCartCell继承UITableViewCell类，自定义满足要求的商品展示单元。点击单元格上的“+”和“-”按钮时会回调ShopCartCell中的block，在ShopCartViewController中实现这个block，操作商品数量的增减。
3. UIButton类为系统类，定义一个结算按钮，点击该按钮会调用btnClicked方法跳入填写订单信息界面。
4. FillInOrderInfoViewCotroller类实现填写订单信息界面，在该界面输入收货人信息以及备注信息，填写完成，将信息传递给后台服务器，后台服务器将各种数据拼接后生成一个签名后的字符串，然后调用orderPay类中的相关方法进入结算页面。

### 5.7.2 支付页面的设计与实现

支付页面相对简单，在该页面主要展示三种支付方式，分别为支付宝、微信及银联。该界面的实现主要使用UITableView，与OrderPay是聚合关系，进行支付是需要依赖网络请求管理类中的方法向服务器发送请求。类图关系如图5.17所示。

**图5.17** **支付模块类图**

在OrderPay类中，payWayArray数组用于存放支付方式的名称，展示在UITableViewCell上。同时在该类中集成了三种支付方式的SDK，点击某一种支付，若手机上已经安装支付APP，将签名作为参数传递打开该APP进行支付。支付完成后，将返回的支付结果显示在客户端，同时调用sendPayResultToServer方法将结果发送给服务器更新数据。

## 5.8 个人信息模块设计与实现

个人信息模块主要用于管理用户信息，根据系统需求实现该模块的功能。个人信息模块的界面如图5.18所示。

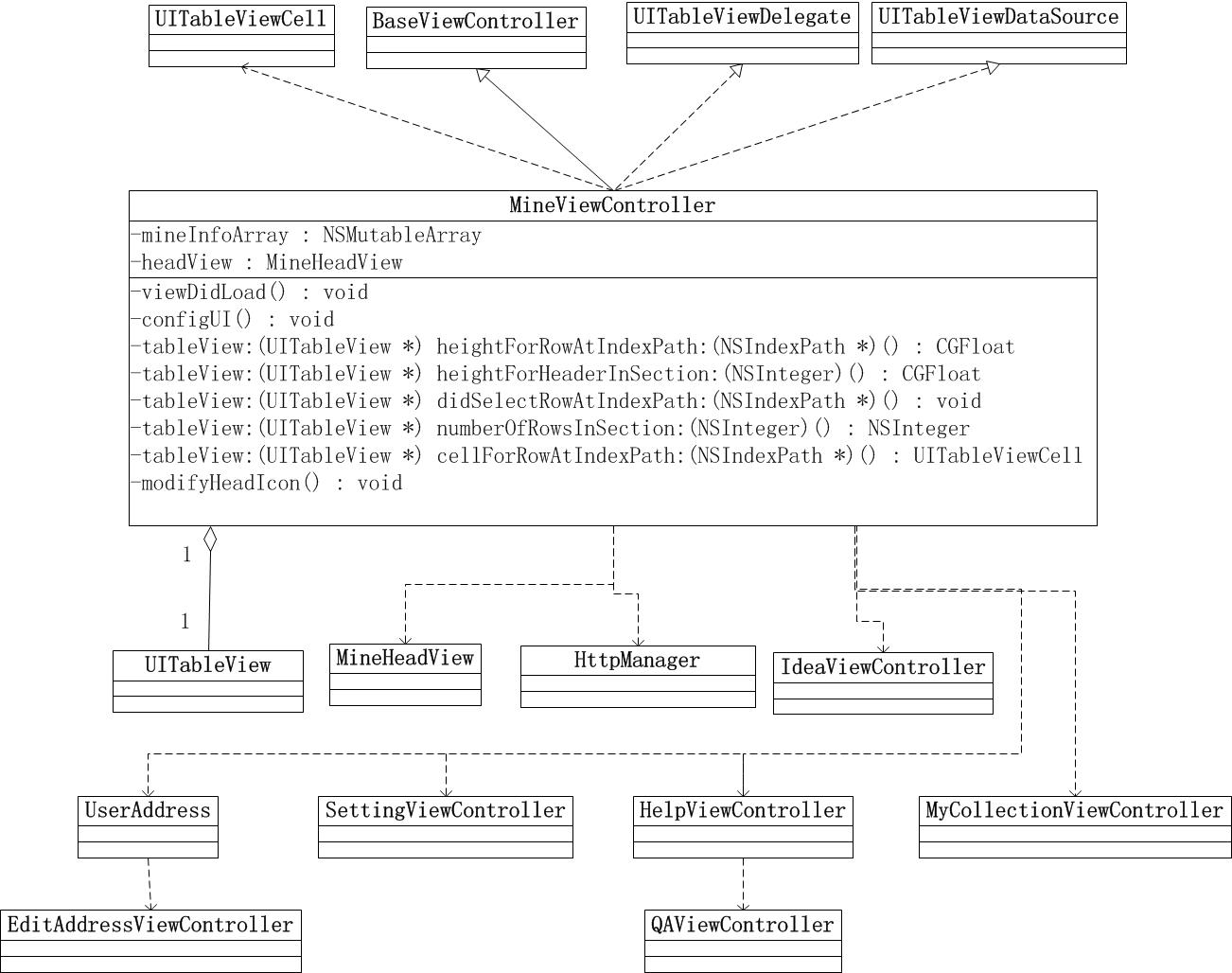
**图5.18** **我的界面**

个人信息界面的实现主要使用UITableView实现，主要分两部分，上部分显示用户头像和手机号，下半部分为选项列表，包括地址、收藏、客服帮助、意见反馈以及设置。

个人信息视图控制器MineViewController依赖UserAddress、MyCollectionViewController、HelpViewController、 IdeaViewController以及SettingViewController五个控制器类，点击列表单元时会跳转到对应的视图控制器。实现该界面以及功能的主要类图如图5.19所示。

图5.19展示了个人信息模块的简单类图关系，各类的功能作用如下：

1. MineViewController类主要使用UITableView实现个人信息界面的展示，MineHeadView对象作为UITableView的头，点击MineHeadView上的头像时会调用modifyHeadIcon方法打开相册，上传头像等一系列操作。其他选项信息存储在mineInfoArray数组中展示在UITableViewCell上。
2. UserAddress类管理用户地址，同理，该类使用UITableView显示所有的地址信息，自定义UITableViewCell可以进行编辑。底部有“新增地址”按钮，点击该按钮则调用该类中的addressBtnClicked方法跳转到EditAddressViewController界面，在该界面可以对地址进行增、删、改操作，编辑完成点击保存则会将地址信息发送给服务器。

**图5.19** **个人信息模块类图**

1. MyCollectionViewController使用UICollectionView展示用户收藏的所有模板，直接使用ProductCategoryViewController中的TemplateCell显示分类模块中相同的信息。
2. HelpViewController同样使用UITableView展示信息，头部使用View显示帮助电话，底部列表显示常见问题选项，点击该项会调用代理方法跳转到QAViewController界面。在该类中会向服务器发送网络请求，将请求结果直接用UIWebView展示，常见问题存储在服务端便于修改。
3. IdeaViewController类用来编辑用户意见，方便用户反馈信息。在该类中主要涉及到TextView及UIButton，TextView显示用户的编辑区域，点击提交按钮则向服务器发起post请求，将反馈信息提交给服务器。
4. SettingViewController使用UITableView显示了三种信息，“关于我们”、“清理缓存”及“退出当前账号”。点击“关于我们”则会发起网络请求获取一个Web网页的公司信息。清理缓存主要是清理通过SDWebImage加载的缓存，通过[[SDWebImageCache sharedImageCache] checkTmpSize]获取缓存大小，然后调用[[SDWebImageCache sharedImageCache] clearDisk]方法清理图片缓存。点击“退出当前账号”则调用logout方法，清理本地该账号信息，同时向服务器发送退出请求。

## 5.9 本章小结

本章介绍系统主要模块的详细设计与实现，从系统的注册登录、分类、在线设计、订单管理、购物车以及个人信息方面进行设计与实现。主要以各个模块的类图为主，再根据类图设计详细介绍内部如何实现，同时展示系统的部分效果图。

# 第6章 个性印刷系统测试

系统在实现完成之后，需要利用测试对系统中实现的功能进行验证。对系统测试来说，就是将已经实现的系统中的各个功能模块组成在一起，采用一种特定的方法针对这种系统进行组装模块及功能确认方面的测试。在进行测试的过程中，测试的范围是针对整个软件产品，验证整个开发的软件产品能够有效的满足系统用户的需求，找出系统中存在的问题，然后采取针对性的措施对其进行完善，通常在软件系统的测试过程中，有两种常见测试方法，分别是白盒与黑盒测试。

## 6.1 系统实现环境

移动端个性印刷系统的实现环境有硬件和软件两种，做iOS开发需要使用Mac系统，所需硬件如表6.1所示。

**表6.1 硬件条件**

|  |  |
| --- | --- |
| 说明 | 型号 |
| 处理器 | 2.39GHz Intel Core i3 |
| 内存 | 8GB |

而软件需求如表6.2所示：

**表6.2 软件条件**

|  |  |
| --- | --- |
| 平台说明 | 型号 |
| 操作系统 | Yosemite10.10.5 |
| 开发工具 | Xcode |
| 开发语言 | Objective-C |
| Web 服务器 | Tomcat 6.0 |

## 6.2 系统测试

### 6.2.1 系统测试概述

黑盒测试[35]，又被称为功能测试，这种测试方法主要是采用观察方法等来测试系统中已经开发完成的应用系统功能，在采用这种方法对系统进行测试的过程中，首先需要对这个软件产品的功能有一个详细的了解，知道系统中有哪些功能。但是不需要对软件系统内容的实现结构进行了解，这么做主要是为了尽最大可能降低系统的内容结构对系统的测试方面所带来的影响，不断的加快系统测试的实现效率。同时，黑盒测试，这种方法对于测试人员的基本要求也比较的低。

白盒测试[36]是一种与黑盒测试完全不同的方法，这种方法对于测试人员自身的要求比较高，要求测试人员必须对开发的软件产品内部的结构有一个清晰的认识，根据系统开发工程师所设计的条件信息、分支信息、顺序结构信息、循环结构等信息，发现系统的不足，帮助系统不断改进，本系统主要使用黑盒测试。

### 6.2.2 系统测试环境

本节中，主要是按照前面的需求对个性印刷系统进行测试：移动客户端测试环境如表6.3所示。

**表6.3 测试硬件环境**

|  |  |  |
| --- | --- | --- |
| 硬件资源 | 名称/类型 | 备注 |
| iPhone4 | CPU | 苹果 A7芯片 M7运动协处理器 1.3GHZ 双核 |
| 内存 | 1GB |
| 硬盘 | 16GB |
| 操作系统 | iOS 7.0 |
| iOS 8.0，iOS9.0 |
| iPhone5s、iPhone6s、iPhone6 plus | CPU | 苹果 A7芯片 M7运动协处理器 1.3GHZ 双核 |
| 内存 | 1GB |
| 硬盘 | 16GB |
| 操作系统 | iOS7.0 |
| iOS 8.0，iOS9.0 |

表6.3所示的内容为系统测试的硬件需求，测试的目标是在不同的硬件设备与不同的操作系统下，系统是否能够正常的运行。

### 6.2.3 系统测试内容

1. 自适应能力测试

自适应测试是开发中需要考虑的一个主要部分，如今苹果有多款不同尺寸的手机，我们的个性印刷系统不能只适配一种尺寸，所有机型都应该适配，这样用户在使用时才会有更好的体验。测试的机型与结果如表6.4所示。

根据表6.4列出的机型对其进行适配能力的测试，测试结果表明适配良好，没有出现严重的界面布局混乱问题。如图6.1为iPhone4在线制作图，图6.2为iPhone6在线制作图。

**表6.4 测试机型**

|  |  |
| --- | --- |
| 机型 | 兼容性 |
| iPhone4 | 兼容 |
| iPhone5s | 兼容 |
| iPhone6 | 兼容 |
| iPhone6 plus | 兼容 |



**图6.1** **iPhone4在线制图** **图6.2** **iPhone6在线制图**

在开发过程中，使用AutoLayout技术对系统进行了布局约束，同时，一部分页面的显示也根据机型的尺寸进行了等比例缩放，以达到布局自适应，用户在不同的机型上看到的UI不会是错乱的。

1. 可靠性测试

可靠性测试[37]是软件可靠性保证过程中非常关键的一步，经过可靠性测试的软件可以降低软件中残存的错误数，使可靠性达到较高的要求。对可靠性的检验可根据安全性、准确性和易恢复性方面考虑。

1. 安全性的测试，通过抓包截取数据，查看关键数据是否加密。使用抓包工具为Fiddler，电脑端安装Fiddler后，在工具栏“Tools->Fiddler Options->Connections”下勾选“Allow remote computers to connect”。手机端网络的HTTP代理选择手动，服务器地址设置为电脑端的IP地址，打开APP进行抓包测试。以登录为例，抓取数据查看用户名及密码是否加密处理，如图6.3所示。

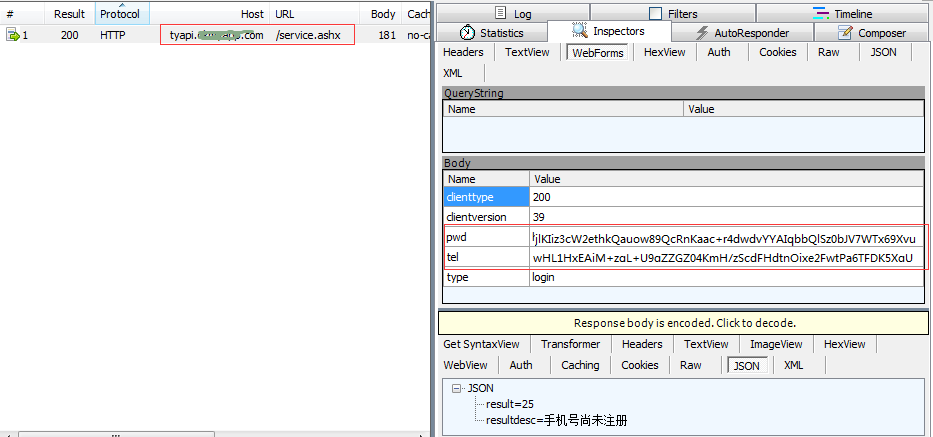
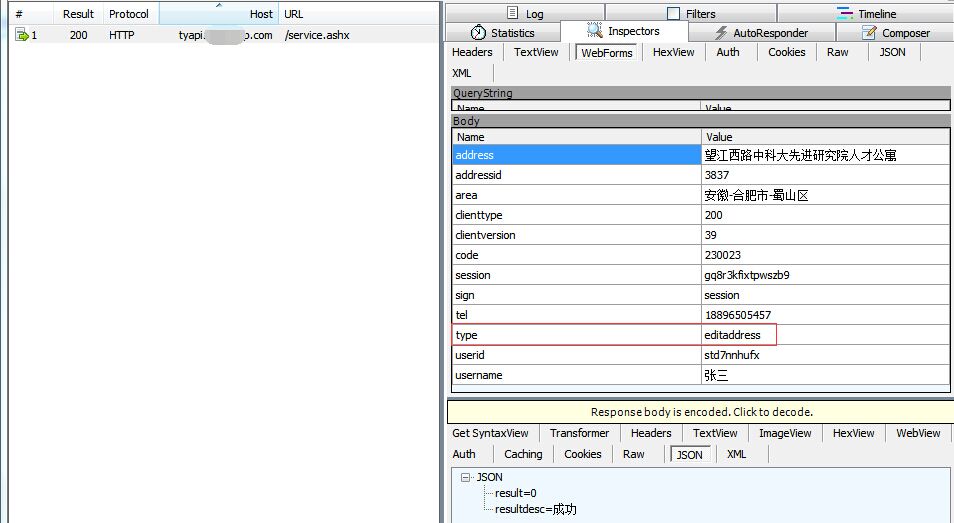
**图6.3** **登录安全性测试**

图6.3为登录时抓取的数据包，通过图6.3可以看到用户的关键信息手机号及密码已经进行了加密处理，同时返回数据表明该手机尚未注册，提示用户注册。通过抓包测试表明，即使数据被外界截取，没有对应的私钥进解密也是一串无用的乱码。系统数据使用RSA的加密方式，公钥放在客户端进行加密处理，私钥放在服务端进行解密，客户端对关键信息加密后服务端是能够进行解密的。同理，注册以及其他敏感数据的操作都使用该方法加密，即使被截取也是一串乱码，从而保证数据的安全可靠。

1. 数据准确性的测试，通过一个例子进行说明，即用户下单时填写的收货地址，测试用户首次填写的地址保存到服务端后，再次获取的地址信息是否和首次的地址相同。和验证安全性方法相同，使用抓包工具抓取客户端与服务端交换的数据。通过请求体中的参数type可知请求类型，type的值为editaddress时是向服务端插入一条数据，type值为getaddress，获取当前用户保存的所有收货地址。

测试数据如图6.4和图6.5所示。

**图6.4** **保存收货地址测试数据**

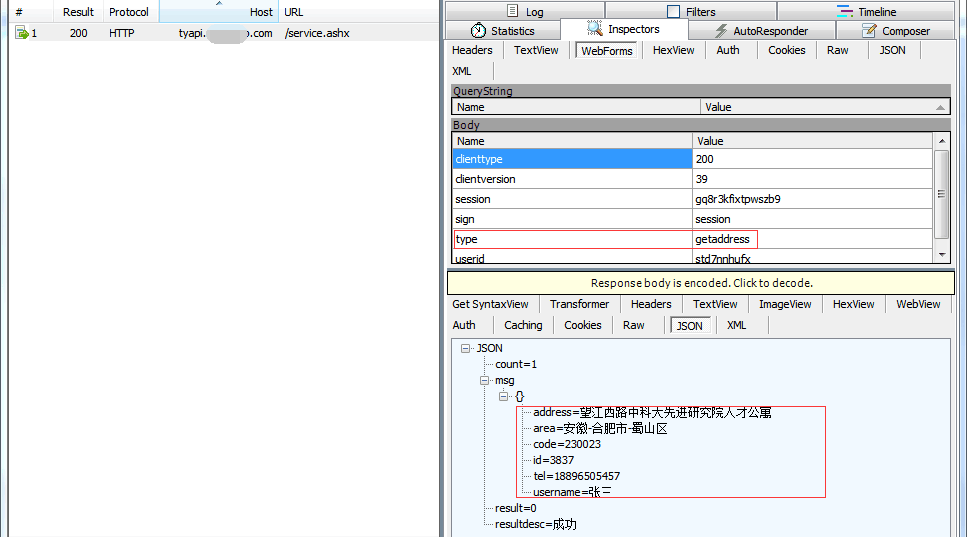
**图6.5** **获取收货地址测试数据**

图6.4为用户编辑收货地址时保存地址向服务器发送的请求，收货地址信息保存在请求体中，服务端插入一条数据。图6.5为请求获取收货地址的数据，用户再次打开收货地址界面时向服务器发送请求，获取当前用户保存的所有收货地址，返回的JSON数据如图6.5。通过对比图6.4和图6.5可发现，用户保存的收货地址与获取到的地址是一致的，从而表明客户端与服务器交换数据的准确性。

1. 易恢复性测试，测试易恢复性时主要测试网络出现错误、网络断开以及没有新数据更新时是否仍保持上次请求的数据。断开网络进行数据刷新时的结果如图6.6及图6.7所示。

**图6.6** **断开网络前数据** **图6.7** **断开网络后数据**

通过对比图6.6与图6.7，发现断开网络后再次更新数据会弹出提示，告诉用户网络异常请检查网络。客户端请求数据的顺序为先发送请求，客户端会根据返回结果判断是否超时、网络异常还是无新数据，只有在获取到新的数据时才会清空之前的数据进行更新，其他情况不进行操作。图6.6与图6.7表明出现网络异常，更新失败时会保持之前的数据，给用户良好体验。

1. 功能测试

功能测试是软件测试过程中非常重要的一个环节，主要是对实现的功能的情况进行基础的测试，通过不断的向服务端发送请求信息，从服务端获取信息，然后验证前端系统的各个功能。该部分测试目标如表6.5所示。

**表6.5 测试目标**

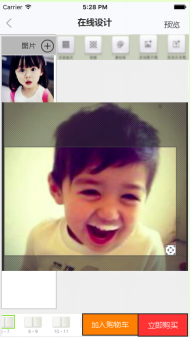
|  |  |
| --- | --- |
| 测试目标 | 个性化印刷系统实现的功能与需求相符 |
| 方法 | 1. 根据需求分析，逐项测试系统功能，看结果是否与需求一致。  2. 覆盖系统中的各个业务流程。 |
| 完成标准 | 1. 系统功能要与需求分析一致。  2. 尽快解决bug问题，对暂时无法解决的问题提出相应的解决方案。 |
| 需要考虑的特殊事项 | 1. 要时刻关注需求的变化，对需求的变化要及时进行验证评测及评估。  2. 注重把需求与现实效果结合起来，确保需求能够实现。 |

本系统涉及到的功能比较多，对应的测试用例更多，不可能在本文中一一列举，下面只以在线制作为例进行测试。测试过程与结果如表6.6所示：

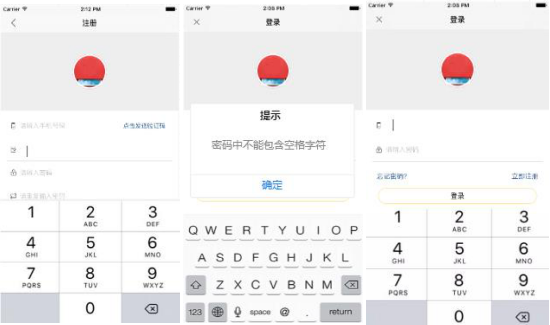
**表6.6 个性印刷系统在线制作测试**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例标识 | UC20 | 项目名称 | | 个性印刷系统 | |
| 开发人员 |  | 模块名称 | | 在线制作 | |
| 用例作者 |  | 参考章节 | | 系统需求分析 | |
| 测试方法 | 黑盒 | 测试日期 | | 2016年3月22日11:32:31 | |
| 用例描述 | 测试在线制作功能是否完善 | | | | |
| 测试项目 | 输入动作 | | 期望结果 | | 实际结果 |
| 添加图片框 | 用户点击页面上方的添加图片框按钮。 | | 点击结束后再当前页显示一个存放图片的框架。 | | 和期望结果相符。 |
| 导入图片 | 点击导入图片按钮，导入相册图片，并将图片添加到相框中。 | | 用户点击导入图片按钮，打开系统相册选择图片，图片出现左侧的图片列表中。 | | 和期望结果相符。 |
| 编辑图片 | 点击相框中的图片或双击图片 | | 点击相框中图片，在屏幕上方出现对应的图片编辑功能，包括各种滤镜和增亮效果，双击时弹出新视图，可对图片进行修剪。 | | 和期望结果相符。 |

选择编辑区域中的一张图片后，选择上方出现的各种处理功能，在测试中展示剪切图片功能，点击上方剪切按钮后的效果图如图6.8所示。

**图6.8** **修剪图片**

1. 健壮性测试

健壮性测试主要针对系统中的输入模块进行测试，通过非规范性输入测试系统是否有能力处理。本文主要对注册与登录模块进行健壮性测试，输入错误的信息观察系统的反应，测试结果如图6.9所示。

**图6.9健壮性测试效果**

图6.9中左边的图表明在进行注册时，手机验证码输入框只能数字，不允许输入其它字符，弹出的键盘为数字键盘。中间这幅图表明在输入密码时，可以输入字母、数字等字符，但不允许输入空格，若存在空格则提示用户。最右侧为登录时输入手机号，和验证码相同，手机号输入框只能输入数字，弹出数字键盘就避免输入其他字符。由测试结果可知满足健壮性要求。

### 6.2.4 测试结果分析

分别从系统的适配、可靠性、功能以及健壮性进行测试。测试发现本系统对各个机型的适配做的很好，无论是小屏的iPhone4还是大屏的iPhone6及iPhone6 plus都能正确的显示。同时通过抓包对系统的可靠性做了测试，测试表明系统在数据安全及准确性方面表现良好，对数据做了加密处理，保存与获取的数据一致。并且在网络异常的情况下仍能维持上次的数据而不是出现错误或空白。而在功能方面，根据第三章的需求分析进行测试，测试结果基本与分析一致，基本都已完成。由于文章篇幅有限，其他的测试过程与结果没有在论文中列出。总体功能能够满足用户的一些日常需求。

## 6.3 本章小结

本章先是列举了iOS的个性印刷系统的测试环境，然后对个性印刷系统的主要功能及非功能进行了测试，包括系统实现的在线制作模块功能、适配能力、可靠性以及健壮性进行了测试，最后对测试的结果进行了分析。

# 第7章 总结与展望

## 7.1 总结

由于移动互联网快速的发展，智能手机用户数量也在不断增长，传统印刷及Web端印刷满足不了移动用户的需求。本文在这些需求之上，设计实现了基于iOS的个性印刷系统，为用户带来了曙光，并提供更好的移动印刷体验。

本文主要工作总结有以下几个方面：

1. 在本文的第一章详细介绍了个性印刷系统的背景，阐述传统印刷及Web端印刷满足不了移动用户的需求，说明实现基于iOS的个性印刷系统的意义并对国内外的研究现状进行分析。
2. 实现基于iOS的个性印刷系统时需要技术的支持，在本文的第二章对系统需要的技术进行简单介绍。其中Objective-C作为开发iOS应用的主要开发语言，JSON则作为移动端与服务端进行数据交换的格式，XML作为记录模板及个性印刷品数据的格式。通过这些内容的介绍，能让读者快速了解系统实现的技术。
3. 在本文的第三、四、五章节对系统的注册、登录、分类、在线制作、订单管理、购物车以及个人信息模块进行分析、设计与实现，采用C/S架构、MVC设计模式以及UIKit框架完成基于iOS的个性印刷系统，并展示部分效果图。通过这几章介绍读者能对基于iOS的个性印刷系统的功能以及实现详情有所了解。
4. 最后对完成的个性印刷系统进行了相关测试，分别对iPhone不同型号手机的适配能力、软件的健壮性、可靠性以及功能进行测试。

通过以上的所有的工作，实现了基于iOS的个性印刷系统，并完成了本文的撰写。

## 7.2 展望

在个性印刷系统设计与实现的过程中，主要是针对系统的需求分析方面、概要设计方面、详细设计与实现方面进行介绍，对照需求分析预期的目标，完成了最初的基础功能。但是由于个人能力及其时间等方面的限制，个性印刷系统在进行处理的过程还有一些不足，主要有：

1. 一些实际的功能模块与真正的实用还存在一定的差距，需要继续完善。
2. 系统无法兼容iOS8.0以下版本，导致一些用户无法使用，下一步工作是完善系统的其他功能，修复系统中一些问题，使其能够兼容iOS8.0以下版本。
3. 对系统进行优化，提高性能。

所以在未来的工作中需要解决以上三个问题，同时丰富系统的功能，为用户带来更好的体验。

# 参考文献

1. 马金涛.个性化数码印刷为印刷企业带来新商机[J].中国印刷,2014(10):47-51
2. [冯良奎](http://www.cnki.net/kcms/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CJFQ&sfield=au&skey=%e5%86%af%e8%89%af%e5%a5%8e&code=).中美个性化印刷的观察与思考[J].数码印刷,2010(04):1-2
3. 昌池.西欧网络印刷呈成长趋势[J].广东印刷,2009(01):2-3
4. 赵建昊.B2B电子商务商业模式比较研究[D].首都经济贸易大学,2013
5. 荆林波.B2B商业模式及其价值评价[J].商业经济文萃,2001(01) : 18-22
6. 候海涛.个性化印刷的特点、应用及在我国的发展情况[J].中国包装,2004(04):79-81
7. 朱奇.网易印像派顾客服务策略研究[J].商,2014(21):66
8. 侯淑英.B /S模式和C /S模式优势比较[J].沈阳教育学院学报,2007(02):98-100
9. [Youcong Ni](http://scholar.cnki.net/result.aspx?q=%e4%bd%9c%e8%80%85:(Youcong+Ni)),[Bei Chen](http://scholar.cnki.net/result.aspx?q=%e4%bd%9c%e8%80%85:(Bei+Chen)),[Peng Ye](http://scholar.cnki.net/result.aspx?q=%e4%bd%9c%e8%80%85:(Peng+Ye)),[Chunyan Wang](http://scholar.cnki.net/result.aspx?q=%e4%bd%9c%e8%80%85:(Chunyan+Wang)) A Framework for iOS Application Development[J].Journal of Software,2014 (02):398-403
10. The Swift Programming Language (Swift 2.2)[OL].https://developer.APPle.com/library/ ios/documentation/Swift/Conceptual/Swift\_Programming\_Language/GuidedTour.html
11. Swift[OL].https://swift.org/
12. Objective-C[OL].https://developer.APPle.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/ObjectiveC.html
13. Paul Krill.APPle's iOS strength prompted Objective-C language's growth in 2011[OL]. http://www.infoworld.com/article/2618274/objective-c/article.html
14. [cainiaozhang](http://www.cnblogs.com/patientAndPersist/).[Programming With Objective-C](http://www.cnblogs.com/patientAndPersist/p/4419994.html)[OL].

http://www.cnblogs.com/patientAndPersist/p/4419994.html

1. [David Chisnall](http://scholar.cnki.net/result.aspx?q=%e4%bd%9c%e8%80%85:(David+Chisnall)).A Modern Objective-C Runtime[J].Journal of Object Technology,2009, Vol.8(01):221-240
2. [王魁生](http://www.cnki.net/KCMS/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CJFQ&sfield=au&skey=%e7%8e%8b%e9%ad%81%e7%94%9f&code=09081672;24239988;),[王晓波](http://www.cnki.net/KCMS/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CJFQ&sfield=au&skey=%e7%8e%8b%e6%99%93%e6%b3%a2&code=09081672;24239988;).利用JSON进行网站客户端与服务器数据交互[J].软件导刊,2010,Vol.9(03):147-149
3. JSON教程[EB/OL].http://www.runoob.com/json/json-tutorial.html
4. 小小程序员.java中Array.List.Map.Object与Json[OL]. http://www.docin.com/p-704478533.html
5. Jack Shin的专栏.[[JSONModel解析数据成Model](http://blog.csdn.net/smking/article/details/40432287)](http://blog.csdn.net/smking/article/details/40432287)[OL].

http://blog.csdn.net/smking/article/details/40432287

1. AkmalB.Chaudhri 等著，刑春晓等译.XML 数据管理:纯 XML和支持XML的数据库系统[M],2006:245~268
2. [章建春](http://www.cnki.net/kcms/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CMFD&sfield=au&skey=%e7%ab%a0%e5%bb%ba%e6%98%a5&code=25352079;).一种基于XML的个性化界面系统的研究与实现[D].电子科技大学,2009
3. 韩怿冰.MVC模式浅析[J].科技信息, 2007(25):351
4. [袁国铭](http://www.cnki.net/KCMS/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CJFQ&sfield=au&skey=%e8%a2%81%e5%9b%bd%e9%93%ad&code=30759821;09643551;06241785;26517832;),[刘瑞](http://www.cnki.net/KCMS/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CJFQ&sfield=au&skey=%e5%88%98%e7%91%9e&code=30759821;09643551;06241785;26517832;),[樊波](http://www.cnki.net/KCMS/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CJFQ&sfield=au&skey=%e6%a8%8a%e6%b3%a2&code=30759821;09643551;06241785;26517832;)等.UML用例图在软件工程中的步骤设计研究[J].微型电脑应用,2014,Vol.30(01):46-48
5. 贾慕星.基于iOS的化妆品电子商务APP的设计与实现[D].北京邮电大学,2015
6. [陈飞](http://www.cnki.net/KCMS/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CMFD&sfield=au&skey=%e9%99%88%e9%a3%9e&code=09011280;).基于三层C/S结构的医院信息系统的研究与实现[D].武汉理工大学,2008
7. [贺令亚](http://www.cnki.net/KCMS/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CMFD&sfield=au&skey=%e8%b4%ba%e4%bb%a4%e4%ba%9a&code=11699196;).RSA加密算法的研究与实现[D].中南大学,2009
8. 记忆里.关于Quartz 2D编程指南[DB/OL].

http://supershll.blog.163.com/blog/static/3707043620128492741525/

1. [BradLarson](https://github.com/BradLarson).GPUImage[DB/OL].https://github.com/shaohua0720/GPUImage
2. [吴恩华](http://www.cnki.net/KCMS/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CJFQ&sfield=au&skey=%e5%90%b4%e6%81%a9%e5%8d%8e&code=05971336;00864671;),[柳有权](http://www.cnki.net/KCMS/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CJFQ&sfield=au&skey=%e6%9f%b3%e6%9c%89%e6%9d%83&code=05971336;00864671;).基于图形处理器(GPU)的通用计算[J].计算机辅助设计与图形学学报,2004,Vol.16(05):602-611
3. [朱奕杰](http://www.cnki.net/KCMS/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CMFD&sfield=au&skey=%e6%9c%b1%e5%a5%95%e6%9d%b0&code=11028543;).基于Linux的OpenGL性能及其跨平台应用程序开发研究[D].哈尔滨工程大学,2007
4. [杨超](http://www.cnki.net/KCMS/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CMFD&sfield=au&skey=%e6%9d%a8%e8%b6%85&code=06420485;).移动互联网终端图形自适应中间性中控件库的开发与应用[D].北京邮电大学,2012
5. 郑岚.[Python访问MySQL数据库](http://www.cnki.net/kcms/detail/detail.aspx?filename=DNBC201006025&dbcode=CJFQ&dbname=CJFD2010&v=)[J].电脑编程技巧与维护.2010(06):59-60
6. ideawu.Objective-C-RSA[OL].https://github.com/ideawu/Objective-C-RSA
7. [Taenam Kim](http://scholar.cnki.net/result.aspx?q=%e4%bd%9c%e8%80%85:(Taenam++Kim))，[Kyungyong Chwa](http://scholar.cnki.net/result.aspx?q=%e4%bd%9c%e8%80%85:(Kyungyong++Chwa)).Parallel algorithms for a depth first search and a breadth first search[J].International Journal of Computer Mathematics,1986,Vol.19 (01):39-54
8. [Boris](http://www.dangdang.com/author/Boris_1) [Beizer](http://www.dangdang.com/author/Beizer_1).Black-Box Testing: Techniques for Functional Testing of Software and Systems[M].Wiley,1982
9. [封亮](http://www.cnki.net/KCMS/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CJFQ&sfield=au&skey=%e5%b0%81%e4%ba%ae&code=20209576;20970474;),[严少清](http://www.cnki.net/KCMS/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CJFQ&sfield=au&skey=%e4%b8%a5%e5%b0%91%e6%b8%85&code=20209576;20970474;).软件白盒测试的方法与实践[J].计算机工程,2000,Vol.26(12):87-90
10. 杨玉丽.[软件可靠性研究现状与展望](http://www.cnki.net/kcms/detail/detail.aspx?filename=DNZS201001053&dbcode=CJFQ&dbname=CJFD2010&v=)[J].电脑知识与技术,2010,Vol.6(01):128-129

# 致 谢

首先，感谢我的母校中国科学技术大学给了我读研深造的机会，并且给了我良好的教育。在读研期间，学校的学习氛围以及严谨的治学态度给我留下深刻的印象，同时学校尽可能的为学生提供学习资料与良好的环境，增长了学生的见识，我以我的母校而自豪。

其次，要感谢我的班主任×××老师，感谢他在读研期间对我的生活和学习上的关心及帮助，×××老师在教学上的认真和对学生负责的态度给我留下深刻印象，同时也是我学习的榜样。最重要的是要感谢×××老师作为我的校内导师，感谢×××老师细心指导我的开题报告及论文的撰写，×××老师无论多忙多晚都会耐心为我解决疑惑，同时给予技术上的支持以及论文修改的建议。若没有×××老师的细心指导，论文的完成非常的困难。在这里非常感谢×××老师的指导，×××老师对学术要求严谨，同时又平易近人，给我留下深刻印象，衷心祝愿她永远健康快乐，工作顺利。

再次，要感谢×××公司。感谢×××给了我实习的机会，使我了解了企业工作和学校学习的区别，让我提前进入工作氛围。同时感谢企业导师×××，感谢她在工作中给予我的指导，为我解决了很多工作中的疑惑，同时为我的论文选题以及工作学习提供很多宝贵的建议，感谢她对我的工作和生活的关心。也要感谢同事们在工作上给予的技术指导，使我的技术得到很大提升。

最后，感谢我的家人和朋友，是他们一直在背后默默的支持和鼓励我，在遇到挫折时他们总会为我加油。感谢父母对我的栽培和付出，希望他们永远健康快乐。

××××年××月

**学位论文修改说明**

**意见1：**英文摘要有错误。

修改说明1：针对英文摘要重新做了翻译，纠正错误。详见英文摘要。

**意见2：**存在错字。

修改说明2：通读文章修改错字，如第3页“在加上公司的...”改为“再加上公司的...”。  
**意见3：**总结内容不准确。

修改说明3：重新撰写总结内容，概括本文所做工作。详见第68页7.1节。

**意见4：**英参考文献格式不对。

修改说明4：针对参考文献格式问题做相应调整，如行间距，字体，期刊引用起止页码。详见第82至83页参考文献。

**意见5：**一些流程图不对。

修改说明5：由于流程图中方框只允许有一个出口，第4章概要设计流程图中并行执行的操作流程不对，故将存在问题的流程图改为活动图，能够并行执行某些操作。具体见第4章图4.3，图4.4，图4.5，图4.6，图4.8，图4.11，图4.12。  
**意见6：**在系统实现中，没有系统模块的关联与调用关系。

修改说明6：针对该问题，在第5章添加5.1节系统总体设计与实现，给出系统各模块之间的类图关系，并描述各类作用及调用关系。详见第37至38页5.1节。