



廣西科技師範學院

Guangxi Science & Technology Normal University

2021 届本科毕业设计

题 目 基于 MongoDB 和 Flask 的网上书店系统的设计与实现

学 院 名 称 数学与计算机科学学院

专 业 名 称 计算机科学与技术（大数据方向）

学 号 1704010135

学 生 姓 名 黄彩思

指导教师姓名（职称） 蒋林利（教授）

教 务 处 制

二〇二一年三

基于 MongoDB 和 Flask 的网上书店系统的设计与实现

计算机科学与技术（大数据方向） 黄彩思

摘 要

【摘 要】 随着互联网大数据时代的到来，越来越多的人使用计算机网络进行一些活动，进行最多的网络活动有购物、社交、办公等。网络和信息化时代还在飞速发展，其改善了人们的生活条件也改变了人们的生活方式。

本文将对购物类型的网络活动进行研究，使用网上书店作为实例。通过对比国内外的研究现状，综合其前沿技术实现一个网上书店系统。本系统是基于 Python 的轻量级 Web 框架 Flask 和非关系型数据库 MongoDB 来设计和实现，最终系统的部署方式采用简单、便捷的 Docker 容器技术，将系统和数据库，以及反向代理服务器 Nginx 都进行容器化管理。

本文通过详细的分析，确定系统需要实现的功能。本系统的基本功能包括用户的登录与注册、图书展示、图书搜索、购物车、用户订单管理、用户评论、后台订单管理、后台图书管理、后台用户管理等。根据系统的需求讲解了其非关系型数据库设计的典型案例，以及 Docker 容器的设计。然后详细的讨论了系统实现的过程，并给出了系统功能的实现截图，用到的关键的技术和一些核心的代码。最后阐述了对于本系统的一些功能测试和性能测试方案。

网上书店的基本功能都已实现，并通过测试。但是因为个人的技术局限性，还存在着许多需要优化和完善的地方。

【关键词】 Flask；MongoDB；网上书店；Docker。

Design and implementation of online bookstore system based on mongodb and flask

Computer science and Technology (big data) HUANG cai-si

Abstract

Abstract: With the advent of the era of Internet big data, more and more people use the computer network to carry out some activities, the most network activities are shopping, social networking, office and so on. The Internet and information age is still developing rapidly, which has improved people's living conditions and changed people's way of life.

This paper will study the online activities of shopping types, using the online bookstore as an example. By comparing the research status at home and abroad, and integrating its cutting-edge technology, an online bookstore system is realized. The system is designed and implemented based on Python lightweight web framework flask and non relational database mongodb. The final deployment of the system adopts the simple and convenient docker container technology, and the system and database, as well as the reverse proxy server nginx are containerized.

Through detailed analysis, this paper determines the functions of the system. The basic functions of the system include user login and registration, book display, book search, shopping cart, user order management, user comments, backstage order management, backstage book management, backstage user management, etc. According to the requirements of the system, the typical cases of non relational database design and the design of docker container are explained. Then it discusses the process of the system implementation in detail, and gives the screenshot of the system function, the key technology and some core code. At last, some functional testing and performance testing schemes of the system are described.

The basic functions of online bookstore have been realized and passed the test. However, due to personal technical limitations, there are still many areas that need to be optimized and improved.

Key words: Flask; MongoDB; Online Bookstore; Docker.

目 录

摘 要.....	1
Abstract.....	2
1 绪论.....	1
1.1 研究背景和意义.....	1
1.2 国内外研究现状.....	1
1.3 论文的组织结构.....	2
1.4 本章小节.....	2
2 相关技术介绍.....	3
2.1 Flask 开发框架.....	3
2.2 MongoDB 数据库.....	3
2.3 Bootstrap 开发框架.....	3
2.4 Nginx 服务器.....	4
2.5 uWSGI 服务器.....	4
2.6 Docker 容器技术.....	4
2.7 本章小节.....	5
3 系统需求分析.....	6
3.1 系统功能性需求分析.....	6
3.1.1 前台功能需求.....	6
3.1.2 后台管理功能需求.....	8
3.2 系统非功能性需求分析.....	9
3.3 本章小结.....	9
4 系统设计.....	10
4.1 系统架构设计.....	10
4.2 功能设计.....	11
4.2.1 商品订单流程设计.....	11

4.2.2 后台订单管理功能设计.....	13
4.2.3 图书管理和用户管理功能设计.....	14
4.2.4 系统其他功能设计.....	14
4.3 数据库设计.....	14
4.3.1 图书集合设计.....	15
4.3.2 订单集合设计.....	16
4.4 基于 Docker 的一键部署设计.....	19
4.5 本章小结.....	20
5 系统的实现.....	21
5.1 环境搭建.....	21
5.1.1 开发环境搭建.....	21
5.1.2 部署环境搭建.....	21
5.2 Flask 项目搭建和 Blueprint.....	21
5.3 系统前台实现.....	23
5.3.1 index 主页实现.....	23
5.3.2 商品详情页实现.....	24
5.3.3 购物车实现.....	25
5.3.4 结算页实现.....	26
5.3.5 用户订单页实现.....	27
5.3.6 订单详情页实现.....	28
5.4 系统后台管理实现.....	29
5.4.1 后台管理主页.....	29
5.4.2 订单管理.....	29
5.4.3 图书管理.....	30
5.4.4 用户管理.....	30
5.5 本章小结.....	31
6 系统测试.....	32

6.1 功能测试.....	32
6.2 性能测试.....	35
6.3 结果分析.....	36
6.4 本章小结.....	36
7 总结与展望.....	37
7.1 总结.....	37
7.2 展望.....	37
参考文献.....	39
致谢.....	40
声 明.....	41

1 绪论

1.1 研究背景和意义

2012 年大数据时代的到来给许多企业带来新的机遇与挑战。面对数据爆炸式的指数级增长,数据的存储、分析等都面临着巨大的挑战。同时,海量数据的结构也变得多样化,结构化、半结构化、非结构化。在现有大数据的存储中结构化数据仅有 20%,其余 80%则是存在于物联网、电子商务、社交网络等领域的半结构化数据和非结构化数据,据统计全球结构化数据增长速度约为 32%,半结构化数据和非结构化数据的增速高达 63%^[1]。而非结构化数据也成为主要的数据成分。这对很多的 web 系统提出了前所未有的挑战。

随着企业的不断扩大,业务的增加,产生的数据量也跟着增长。而如何存储这些海量的数据,如何对这些数据提取有价值的信息成为很多系统急需解决的难题。而目前,许多的企业 web 系统还是使用传统的关系型数据库来存储数据,面对这些海量的、复杂的数据,传统的数据库已经满足不了需求。而 MongoDB 作为介于关系型数据库和非关系型数据库之间的文档类型的数据库,其存储的方式,非常适合 Python 中字典、列表等类型的使用。而 Python 语言作为大数据时代最热门的语言,集成了大量的第三方库,便于用户的使用。Flask 框架就是一个轻量级的 web 框架。虽然是轻量级,但是应有的功能都有,扩展性也很大可以快速的搭建一个 web 服务。

因此,在大数据时代背景下网上图书系统的设计应该要考虑大规模的多样性的数据集合,并且能够为 web 应用提供可扩展的高性能存储解决方案。以及对这些大规模的、松散的、低价值密度的数据进行预测、分析,从而提取有价值的数据,提升竞争力等,成为 web 应用适应大数据环境的关键。

1.2 国内外研究现状

网上虚拟书店,业务环节大为简化,能节约费用开支,降低销售成本^[2]。国外对于线上书店的发展相对于国内来说要早很多,其中最知名的是亚马逊平台了,其在 1995 年就开始发展网上书店业务,经过两年的发展在 1997 年成功上

市。亚马逊发展到今天已经是一个综合的跨国电子商务了。

国内发展的相对于国外的发展来说要晚一些，国内知名的网上书店就是当当网了，其在 1999 年才正式成立的，比国外晚了 4 年，其发展到现在也没有国外的亚马逊出名。

1.3 论文的组织结构

本论文由以下章节构成：

第一章 绪论。介绍了网上书店系统的背景、国内外研究发展的现状、系统设计的目标以及论文的意义。

第二章 相关技术的介绍。阐述论文中参考和使用到的技术，其包括 Python 的 Flask 框架、非关系型数据库 MongoDB 的概念和前端框架 Bootstrap。

第三章 系统需求分析。分析网上书店系统的实际业务需求、功能性需求和非功能性需求。

第四章 系统总体设计。描述了网上书店系统的总体设计方案，分别描述了系统的架构设计、系统功能模块设计和系统数据库设计。

第五章 系统详细实现。描述网上书店系统的详细设计，包括系统的开发环境的搭建、购物车功能、前台用户功能模块和后台管理模块的实现。

第六章 系统测试。对系统所开发的基本功能进行测试，以及系统的稳定性测试和系统的抗压测试。

最后一章 结论。总结系统，阐述了论文的主要工作、存在的问题和对未来的规划。

1.4 本章小节

本章通过查阅现有资料阐述了，本篇研究的意义，结合国内外研究的现状，为今后的需求提供理论支持，最后对本文的结果安排进行简单的说明。

2 相关技术介绍

2.1 Flask 开发框架

Flask 是一个使用 Python 编写的轻量级 Web 应用框架,也被称为“microframework”。Flask 的基本模式为在程序里将一个视图函数分配给一个 URL,每当用户访问这个 URL 时,系统就会执行给该 URL 分配好的视图函数,获取函数的返回值并将其显示到浏览器上^[3]。其 WSGI 工具箱采用 Werkzeug, HTML 模板引擎则是使用 Jinja2。Flask 没有像 Django 框架一样有默认的数据库、admin 后台管理工具、ORM 等,但是保留了对这些工具扩增的弹性,即像插件一样的第三方库,开发者可以按需要有选择的添加这些功能。如对 Bootstrap 集成的 Flask-Bootstrap,像 Django 一样的关系型数据库框架的数据库映射 ORM 的 Flask-SQLAlchemy,也有用于 MongoDB 的 ORM 的 Flask-MongoEngine,以及用于后台管理的 Flask-admin,还有结合了 SQLAlchemy 的 Flask-security 用于权限管理等第三方库和插件。选择 Flask 的一个原因是,开发本系统不需要用到这么多的功能,需要的功能可以使用装饰器对函数进行扩展。装饰器是可调用的对象,其参数是另一个函数(被装饰的函数),装饰器可能会处理被装饰的函数,然后把它返回,或者将其替换成另一个函数或可调用对象^[4]。如使用装饰器判断是普通游客还是已登录的用户。

2.2 MongoDB 数据库

MongoDB 是使用 C++语言开发的非关系型数据库,具有高性能、高可用、可伸缩、易部署、易使用等特点^[5]。它不仅可以用于单台计算机,还可以扩展部署到复杂大型、多数据的计算中心。其存储的方式是使用一种类似于 json 的 bson 格式,支持多种复杂的数据类型如数据、对象、大文件等。还支持对数据建立索引。MongoDB 的查询语法也支持多种查询方式,有点类似于面对对象的查询语言,也支持正则表达式等多种查询方法。支持 RUBY、PYTHON、JAVA、C++、PHP、C#等多种语言。

2.3 Bootstrap 开发框架

Bootstrap 是 Twitter 推出的一个开源的用于前端开发的工具包。其基于 jQuery 框架开发,可以让前端的开发变得更加快速、简单,开发者能够快速上手,其具有响应式的 web 设计,可以适配各种设备和浏览器。即插即用非常适合前端基础较弱的开发者使用,也具有活跃的开发者的交流社区和中文文档,方便学习和交流使用。

2.4 Nginx 服务器

Nginx 是一个非常高效、轻量，功能强大的 Web 服务器。Nginx 可以用作服务器也可以作为反向代理使用，其强大的功能通过简单的配置即可使用。在性能上还可以进行横向扩展，部署为分布式模式，以满足高性能要求。也可以对 URL 进行高级匹配，或根据请求内容进行转发到相应的上流服务器，可以对分布式的服务器进行负载均衡。Apache 是损失优化和处理速度而致力于功能，在实践中，这种结果导致过多的内存和 CPU 开销，与其相对，Nginx 被证实是轻量级的、稳定的，能够提供大量的请求（与 Apache 相比使用最少的内存和 CPU）^[6]。相对于早期的 Apache 服务器来说，Nginx 的出现将许多使用 Apache 的用户转为更加简便和高效的 Nginx，而 Nginx 服务器作为 Python 的 Web 应用的服务器也符合 Python 的设计理念，简洁、高效。

2.5 uWSGI 服务器

uWSGI 是一款由 C 语言开发的高效服务器，能和 Python 的 Werkzeug 无缝衔接。Werkzeug 实现 Web 的简单请求、相应功能，其适合在 Web 开发阶段的简单自测中使用。而 uWSGI 则很好的弥补了 Werkzeug 在生产环境中运行的不足之处，不止在 HTTP 协议上使用，还可以在 socket 套接字协议上使用。还可以设置服务运行的进程、线程和异步数，数据缓存的大小、内存大小，request 请求数据的最大值，以及控制访问量大小。合理的配置可以有效的防止因请求导致内存溢出导致系统崩溃的风险。

2.6 [Docker](#) 容器技术

Docker 是一种轻量级的虚拟化容器技术。Docker 在容器的基础上，进行了进一步的封装，从文件系统、网络互联到进程隔离等等，极大的简化了容器的创建和维护，使得 Docker 技术比虚拟机技术更为轻便、快捷^[7]。每一个容器都可以看做是一个最小化的 Linux 系统，只需要运行所需的最小资源。每个容器之间相互独立，互不影响，用户可以在容器中安装部署任何的应用程序。而作为 Docker 项目工具三剑客之一的 [docker-compose](#) 为管理和集成项目容器起到了很大的作用。正如 docker-compose 的 logo（见-图 1）看上去一样，docker-compose 把项目中用到的一个个独立容器进行编排和管理，使其能做到一键将项目快速部署上线，项目后期运维也方便，也可以根据系统用户的访问量情况动态的扩展或收缩服务容器的数量来满足高性能的需求。

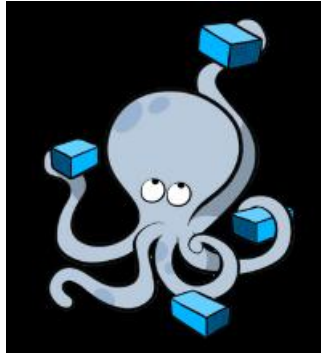


图 1 docker-compose logo

2.7 本章小节

本章主要介绍实现本系统所需要用到的主要技术，主要的 Python web 开发框架 Flask，用于存储数据的非结构化数据库 MongoDB，主流的反向代理服务器 Nginx，以及用于部署上线的 Docker 容器技术。

3 系统需求分析

需求分析是软件生命周期中极其重要的一步,没有做好需求分析,将会给整个软件项目的开发造成极大的难度^[8]。没有好的需求分析阶段,将会导致项目开发到后期时,要改动或添加很多功能需求,或是在没有了解需求的具体实现时,就进行盲目开发,而导致到后期才发现所需功能与用户需求不一致的问题。另一方面,做好需求分析也能提前发现实现过程所需要的技术,或技术难点,从而做好合理的开发计划周期。

3.1 系统功能性需求分析

3.1.1 前台功能需求

前台功能主要是提供用户访问浏览本系统的功能页面。主要分为用户层面的交易流程:主页商品信息、商品搜索、商品详情、用户购物车、结算和去支付,以及用户自己的信息管理功能:用户登录和注册、用户信息也、用户订单管理和收货人信息等。

1. 前台的主页功能

前台的主页功能是本系统的门面,基本上所有的用户访问本系统都是从主页面开始。所以在页面样式上需要设计得简单、友好、符合用户的操作习惯。

从上到下依次为:用户信息的导航栏,接着是展示商品部分类型的导航栏、系统 logo,以及商品搜索栏。用户可以点击感兴趣的图书类型进行浏览或者是通过搜索栏直接进行搜索。搜索栏处可以接受模糊查找图书标题、简介内容、图书作者,以及出版社。

广告轮播:需要在主页上设计有广告推荐的区域,在举行大型的促销活动或者热门商品时能够被访问本系统的用户发现,以提高活动产生的效益。

读者推荐:主页中的商品被用户访问的概率非常大,所以在滚动条下可以放置适量的、少数的商品信息,以供用户查阅。

新书推荐:部分用户可能会对新出售的图书商品感兴趣,所以在主页中也应该有一部分展示新书的区域。

畅销榜：有许多读者在浏览的时候，可能会因为从众心里而选择购买大多数用户都购买的图书，畅销榜专门针对此类用户来展示本系统出售最多的商品。

2. 商品详情页面

此页面主要展示商品的详细信息，如商品价格、名称、简介、作者、出版社、出版时间等，以及提供用户加入购物车或者直接购买的按钮。还有商品的详情，购买过的用户对此商品的评价等。到这里的内容为所用用户都能够访问，即不需要登录。此后的功能都将需要用户登录后才能更进一步地进行访问本系统。

3. 登录与注册

未登录的用户可以点击顶部的导航栏中的登录链接，在弹出的登录窗口进行登录，如果是未在本系统进行过注册的用户可以点击注册链接，前去进行注册后再进行登录。还有一种登录方式是未登录用户直接访问需要登录的功能，然后系统会自动跳转到登录页面。

4. 购物车功能：

用户可以通过点击‘+’、‘-’号或者输入需要加入到购物车里的商品的数量，加入成功后会显示成功的页面，然后用户可以选择继续浏览本系统的其他商品信息，或者是直接去购物页面结算。我的购物车功能，可以对购物车内的物品数量进行‘+’、‘-’，然后可以选择需要购买的商品去结算。

5. 结算页面：

需要提供物流的收货人信息，用户可以进行简单的收货人信息的新增、编辑或删除，以及确认商品数量等信息。最后是进过优惠或打折后，再加上物流运费的应付款的实际金额。

6. 支付页面

用户可以选择喜欢的支付方式进行支付，生成订单后，也可以在用户的个人信息中的订单管理中在有效时间内进行支付，也可以取消本次的订单，或在订单的详情页中取消订单。

7. 订单详情

展示订单的详细信息，收货信息、订单状态、物流信息、以及下单时间等。

8. 我的订单

此功能用来管理用户的订单，查看订单的详情，订单的状态分类等信息。

9. 我的信息

用户可以完善或修改其信息，也可以上传头像，或者是修改登录密码等。

10. 收货人信息

包含了用户所有的收货人信息，可以进行添加新的收货人，也可以对已有的信息进行修改、删除，或者是设置新的默认收货地址。

11. 商品评价

用户确认收货后，可以对购买的商品进行打分、评论、晒图。

3.1.2 后台管理功能需求

后台管理功能主要是对用户的订单状态的管理、图书信息的管理，以及用户管理等。

1. 系统数据可视化

后台管理的默认页面，将一些数据进行处理展示。有系统访问量的环形图和热力图。用户搜索内容的词云图。用户点击量和销售前十的柱形图

2. 订单管理

可以对订单进行查询、处理。待发货订单对已发货的订单进行处理，使其进入待后货状态。待付款订单，展示还未付款的订单。待退款订单，处理用户的退单。待收货订单，展示用户还未收货的订单。待评价订单，展示用户确认收货后，还未进行评价的订单。成功订单，即交易已完成的订单。失效订单，用户取消，或到时间用户未支付的订单。

3. 图书管理

可以搜索图书商品，查看物品信息，或者将商品下架处理。图书的详情中可以对其所有的信息进行编辑。下架图书不会在前台用户页面中展示。也可以在下架了的图书页面中对商品进行重新上架处理，或者把图书加入回收站。也可以进行添加商品操作。

4. 用户管理

展示用户的用户名、手机号、Email 等信息。如果用户忘记了密码，可以帮助用户进行密码重置，将会产生新的密码。

5. 已删除

最后是对已删除的图书和用户的处理，可以进行还原，也可以进行永久删除。

3.2 系统非功能性需求分析

非功能性需求，是指软件产品为满足用户业务需求而必须具有且除功能需求以外的特性，包括安全性、可靠性、互操作性、健壮性等^[9]。

安全性：保密性，系统在被入侵或被其他的方式攻击，能确保用户的私密信息不被泄露。

可靠性：web 系统开发是一个不断迭代，逐渐补充功能的一个过程，所以需要有简单方便的升级过度功能。在升级过程中不影响用户的使用或用户没有感觉到系统升级。

健壮性：系统能够 24 小时无停机的在线提供用户服务，即使服务器出现故障或被攻击的情况下，系统依然能继续提供服务。

3.3 本章小结

本章主要讲了关于本系统的功能性需求分析和非功能性需求分析。在功能性需求分析方面，具体讲了系统的前台用户的功能需求，以及后台管理方面的一些功能需求。

4 系统设计

4.1 系统架构设计

1. 系统流程图

用户通过浏览器访问本系统，而本系统的所有组件都将部署在 Docker 容器中。假如系统部署在云服务器上，用户访问本系统到达云服务器时，首先访问的是 Nginx 服务器容器 bookstore_nginx_1，Nginx 判断请求 URL 是否为 `http://[bookstore_web_1-ip]:[port]`。如果是将请求代理到 bookstore_web_1 容器，即到达了 uWSGI 服务器，uWSGI 服务器在启动时会加载 Flask 应用的入口，启动 Web 程序，监听 8000 端口，根据请求的内容由 Flask 应用跟底层的 DB 进行数据的交互，并将获取的结果进行封装，然后响应请求，原路返回。系统的流程图如图 2 所示。

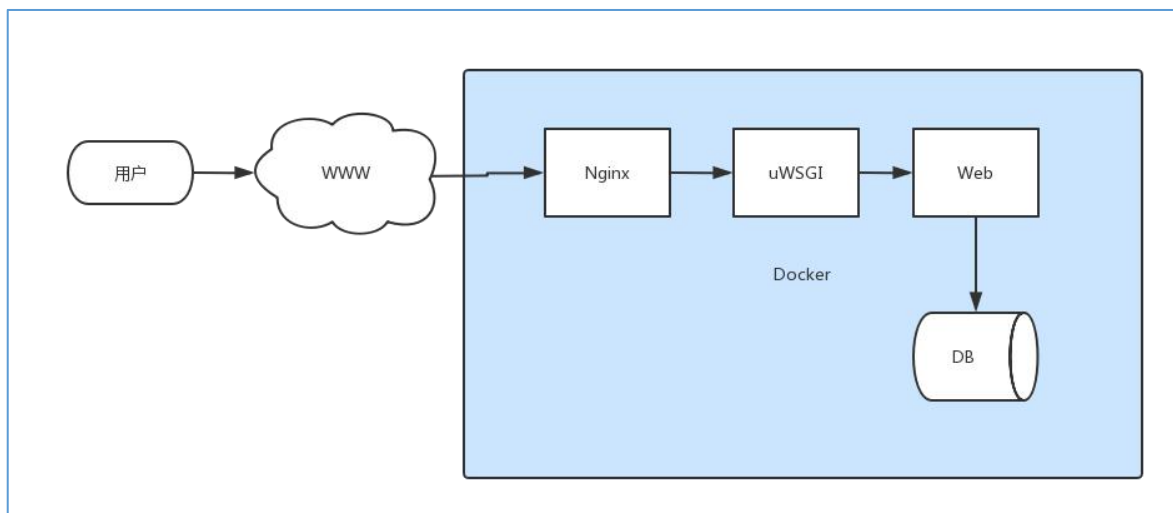


图 2 系统流程图

2. 系统架构图

本系统采用的架构是传统的 MTV，即 Model Template View。Model 模型是操作数据库的部分，Template 模板渲染 HTML 页面展示给用户，View 视图是负责业务逻辑的部分，在很多情况下为调用 Model 和 Template。系统 MTV 架构图如图 3 所示。

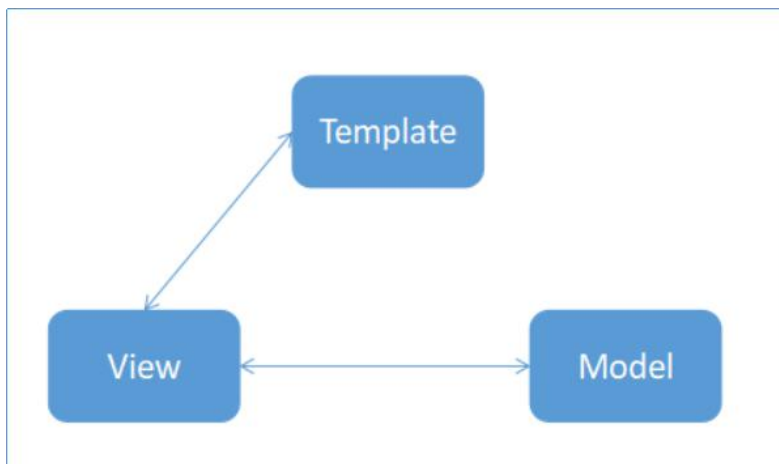


图 3 MTV 架构图

4.2 功能设计

根据系统需求分析实现详细的功能设计模块。基本功能模块设计如图 4 所示。

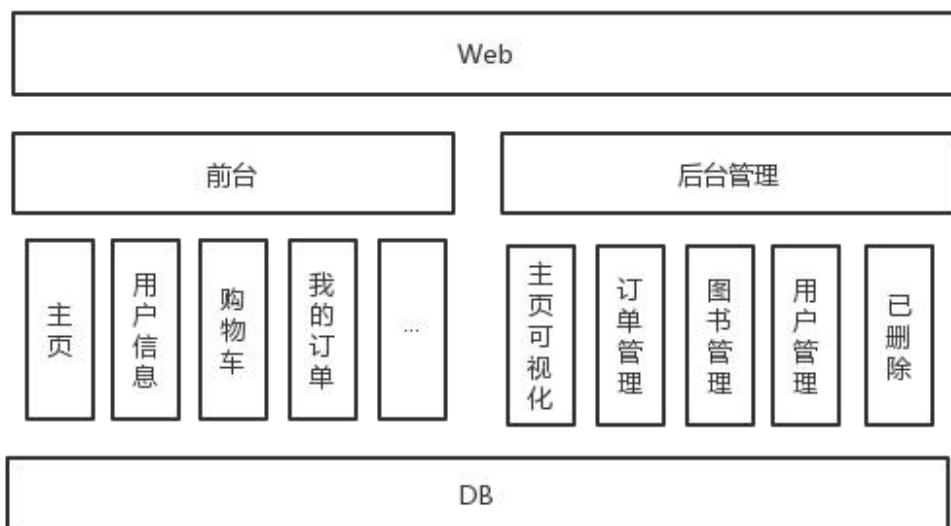


图 4 系统功能

4.2.1 商品订单流程设计

用户通过商品详情页或搜索页，可以找到订单交易流程的开始入口，即“加入购物车”、“立即购买”。接着系统会判断用户是否登录，未登录的会自动跳转登录页面，要求用户进行登录，如果用户还未注册则可以先注册再进行登录操作。如果用户已经登录了，则继续交易流程。

如果是“加入购物车”流程，则可以选择需要添加的商品数量，点击后将跳

转到添加成功页面。此说明其添加的商品图片、名称和数量，以及提供“去购物车结算”和“继续购物”选择。

购物车页面展示用户购物车的所有物品的图片、名称、单价、商品数量和总价。用户可以在购物车中操作商品的数量，当减为 0 的时候代表用户要将此物品移除购物车，购物车的操作为使用 JQuery 的异步操作。用户选择的商品将动态生成有多少件商品和总的金额。

购物车页面的结算操作和“立即购买”一样跳转到结算页面，不同的是立即购买的是一种物品，而从购物车跳转的则可以不只是一类商品。此页面必须要有收货人信息，否则无法生成订单。用户在个人信息管理中有设置默认的收货人信息的将会在此页面中展示，并且使用默认收货人信息来生成订单。如果没有填写收货人信息用户也在此页面新增收货地址，也可以对默认的收货信息进行编辑修改。此页面将会根据用户选择的商品，价格、数量、运费、优惠项目和包含的运费生成订单应付的金额。用户点击“去支付”将会生成订单，并且跳转到支付页面，让用户选择一项支付方式进行支付费用。生成的订单会有待支付时间，超时则交易失败。

支付页面将会展示应支付的金额和订单号，以及本系统支持的支付方式，用户可以在其中选择一种进行支付。也可以点击订单号查看订单的详细信息，未支付的订单也可以在详情页中去支付。已支付的订单可以在详情页面中查看物流信息、订单的物品信息、下单时间，也可以对已支付的订单提交退货申请。

后台管理中确认订单已付款后，就可以对此订单物品进行发货处理。已发货的订单，用户可以在“我的订单”中国进行查看，其中会展示待收货的订单数量。如果用户收到货后确认无误，则可以在待收货中“确认收货”。确认收货后的订单可以对商品进行评价。评论提交后将会在商品的详情页的用户评论中展示。

用户可以在支付成功后再“我的订单”中待发货、待收货中提交退款申请，提交的申请将会在后台管理系统中可见，等待人员确认申请。提交退款申请成功的订单，其流程将从交易失败中结束。订单流程如图 5 所示。

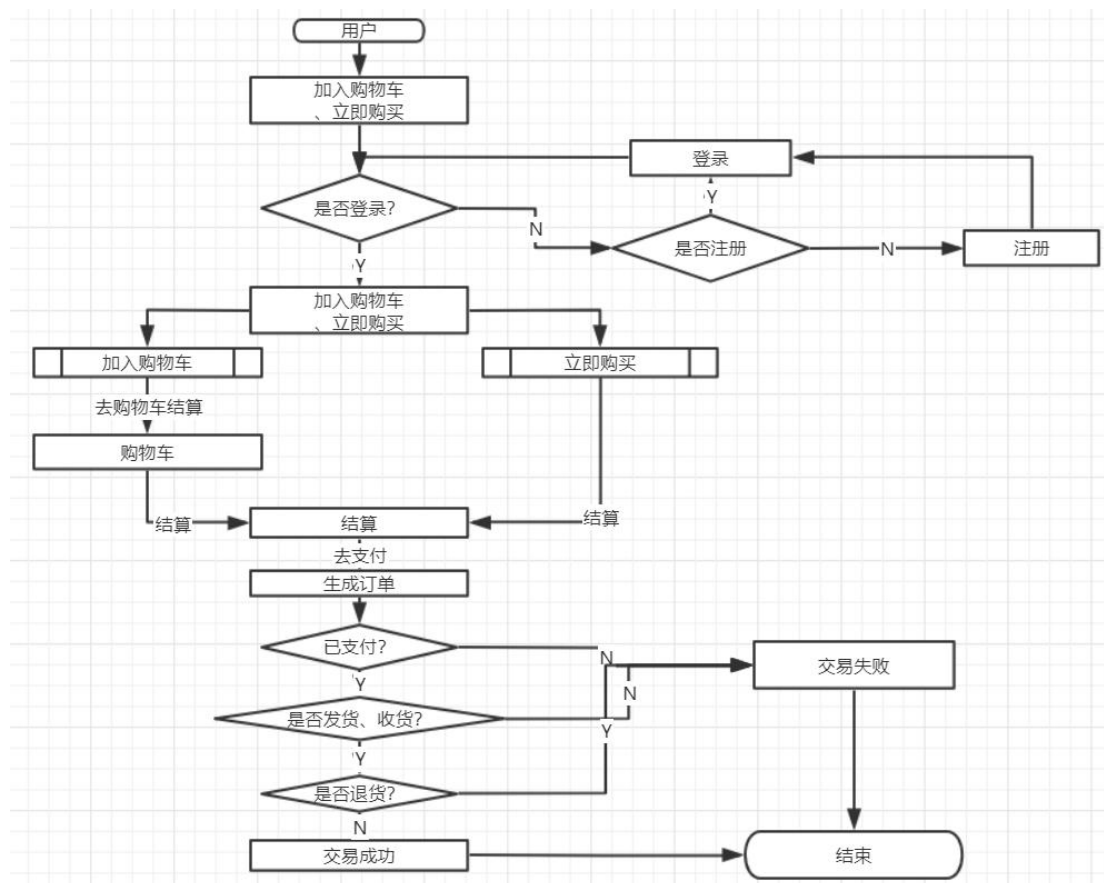


图 5 订单交易流程

4.2.2 后台订单管理功能设计

用户提交生成的订单将会在“待付款订单”中展示，超过支付时间还未支付的订单则会在“失效订单”中。用户已支付的订单其状态在“待发货订单”中等待人员处理。发货后，订单状态转为“待收货订单”中。用户确认收货后，订单交易成功。后台订单管理流程设计如图 6 所示。

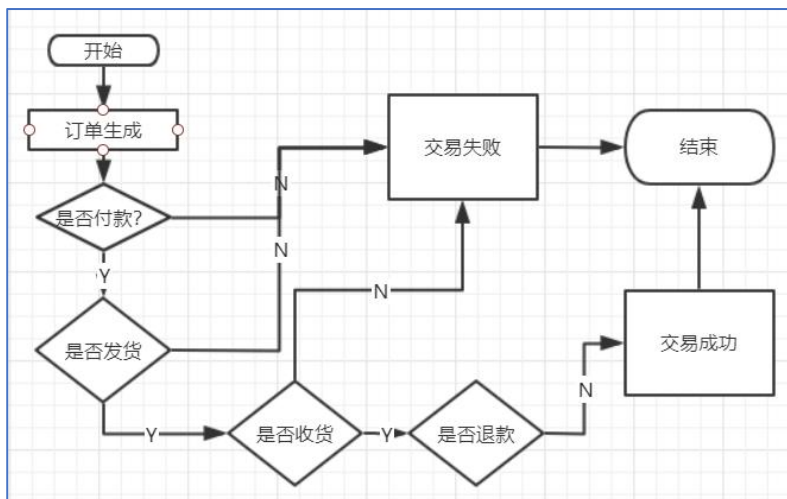


图 6 后台订单管理流程设计

4.2.3 图书管理和用户管理功能设计

图书管理可以对图书的书名、作者、简介、售价、定价、出版社、出版时间、封面进行编辑修改。也可以对图书进行下架处理，下架后的图书将不会在用户搜索中出现，也可以对下架的图书进行上架处理。只有下架的图书才可以删除，而只有“已删除图书”才可以被永久删除。

添加图书部分，通过输入图书信息，以及上传封面进行添加操作。

在用户管理功能中提供替忘记密码的用户进行重置随机密码的操作和冻结、激活账号，以及将账号进行删除操作，同样的只有“已删除账号”才能进行永久删除操作

4.2.4 系统其他功能设计

用户可以在前台任何页面进行登录或者退出登录操作。点击用户的名称将进入用户的菜单页面，包括我的订单、我的信息、收货人信息。

用户可以在我的订单中的订单进行操作，在我的信息中添加或修改用户的个人信息，也可以修改登录密码和上传更好头像。

可以在收货人信息中添加收货人信息，以及设置默认的收货地址。

4.3 数据库设计

MongoDB 属于惰性的键值对数据库，介于关系型与非关系型数据库之间，

因此其数据库设计与传统的关系型数据库有许多的差异之处。

4.3.1 图书集合设计

表 1 MongoDB 图书集合设计表

键	数据类型	说明
_id	ObjectId	默认 id
Price_m	Float	定价
Price	Float	售价
Title	String	书名
Subheading	String	简介
Author	String	作者
Press	String	出版社
pub_time	String	出版时间
img_url	String	封面
ISBN	String	国际标准书号
Category	String	类别
Type	String	类型
Hits	Int	点击量
Sales	Int	销量
is_off_shelf	Int	下架

Book 文档内容示例：

```
{
  "_id" : ObjectId("5ee96be2360d930a489db218"),
  "type" : "小说",
  "pub_time" : "出版时间:2016 年 05 月",
  "press" : "长江文艺出版社",
  "is_off_shelf" : 1,
  "title" : "左宗棠（全二册）（长篇历史小说经典书系）",
```

"subheading": "全新修订珍藏版。帝国不屈的鹰派、晚清不垮的脊梁，中国不可一日无湖南，湖南不可一日无左宗棠。",

"ISBN": "国际标准书号 ISBN: 9787535486905",

"price_m": 72,

"price": 65.9,

"category": "历史",

"img_url": "http://img3m6.ddimg.cn/3/12/23958696-1_w_6.jpg",

"author": "张鸿福",

"hits": 16,

"sales": 1

}

4.3.2 订单集合设计

表 2 MongoDB 订单集合设计表

键	数据类型	说明
_id	ObjectId	默认 id
Is_processed	Int	处理
Create_time	Int	创建时间
Amount	Float	总额
Books	Array	图书列表
book_id	ObjectId	默认 id
book_num	Int	图书数量
user_id	Id	用户 id
Logistics	Array	物理信息
Create_time	Int	创建时间
Info	String	物理详情
order_no	String	订单号

pay_status	Int	支付状态
is_effective	Int	有效
exp_status	Int	物理状态
orders_status	Int	订单状态
Address	Object	收货地址对象
Details	String	详细地址
District	String	区
City	String	市
Province	String	省
Tel	String	手机号
Name	String	姓名

Order 文档内容示例:

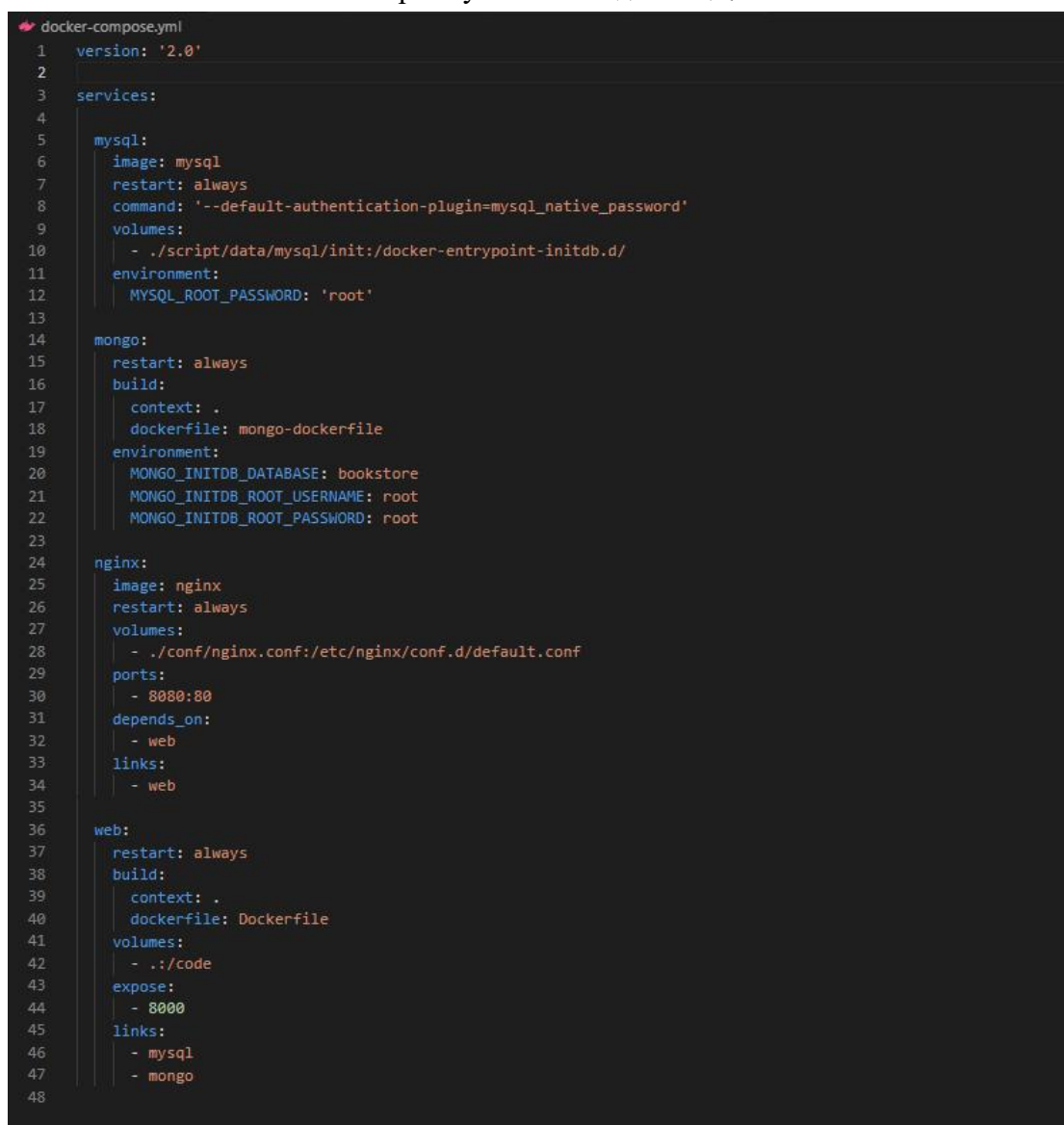
```
{
  "_id" : ObjectId("604daa4eaf5c3d633d3f0b5c"),
  "pay_status" : 0,
  "logistics" : [
    {
      "create_time" : 1615702964,
      "info" : "商品已经下单"
    },
    {
      "create_time" : 1615703942,
      "info" : "包裹正在等待揽收"
    }
  ],
  "orders_status" : 4,
  "user_id" : 5,
```

```
"amount" : 197.55,
"is_effective" : 1,
"exp_status" : 0,
"is_processed" : 0,
"create_time" : 1615702606,
"order_no" : "202103140616720346",
"address" : {
  "city" : "来宾市",
  "details" : "广西科技师范学院",
  "_id" : ObjectId("604da6cbaf5c3d633d3f0b49"),
  "district" : "兴宾区",
  "user_id" : 5,
  "name" : "阿斯蒂芬",
  "province" : "广西",
  "tel" : "12341234123"
},
"books" : [
  {
    "book_num" : 2,
    "book_id" : "5ee97b2d360d930a489dc295"
  },
  {
    "book_num" : 1,
    "book_id" : "5ee96eaf360d930a489db51c"
  },
  {
    "book_num" : 1,
```

```
        "book_id" : "5ee979f1360d930a489dc13b"
    }
]
}
```

4.4 基于 Docker 的一键部署设计

使用 Docker 可以简化系统部署上线的流程，也可以简单的实现跨平台部署应用。并且 docker-compose 工具可以简单的使用 scale 配置参数来达到容器负载均衡的效果。项目 docke-compose.yml 配置文件关键代码如图 7 所示。



```
1  version: '2.0'
2
3  services:
4
5      mysql:
6          image: mysql
7          restart: always
8          command: '--default-authentication-plugin=mysql_native_password'
9          volumes:
10             - ./script/data/mysql/init:/docker-entrypoint-initdb.d/
11          environment:
12             MYSQL_ROOT_PASSWORD: 'root'
13
14      mongo:
15          restart: always
16          build:
17             context: .
18             dockerfile: mongo-dockerfile
19          environment:
20             MONGO_INITDB_DATABASE: bookstore
21             MONGO_INITDB_ROOT_USERNAME: root
22             MONGO_INITDB_ROOT_PASSWORD: root
23
24      nginx:
25          image: nginx
26          restart: always
27          volumes:
28             - ./conf/nginx.conf:/etc/nginx/conf.d/default.conf
29          ports:
30             - 8080:80
31          depends_on:
32             - web
33          links:
34             - web
35
36      web:
37          restart: always
38          build:
39             context: .
40             dockerfile: Dockerfile
41          volumes:
42             - ./code
43          expose:
44             - 8080
45          links:
46             - mysql
47             - mongo
48
```

图 7 docker-compose 配置文件

Docker 启动设计:

首先 docker 获取本地是否有相关的镜像，如：Python3.5、Nginx、mongo。如果没有将尝试拉取远程的镜像文件，即 dockerhub 仓库的镜像。如果使用 dockerfile 文件进行构建容器，则在 build 阶段会执行 dockerfile 文件的命令，否则跳过直接等待启动容器的时候初始化容器数据。Docker 容器启动过程如图 8 所示。

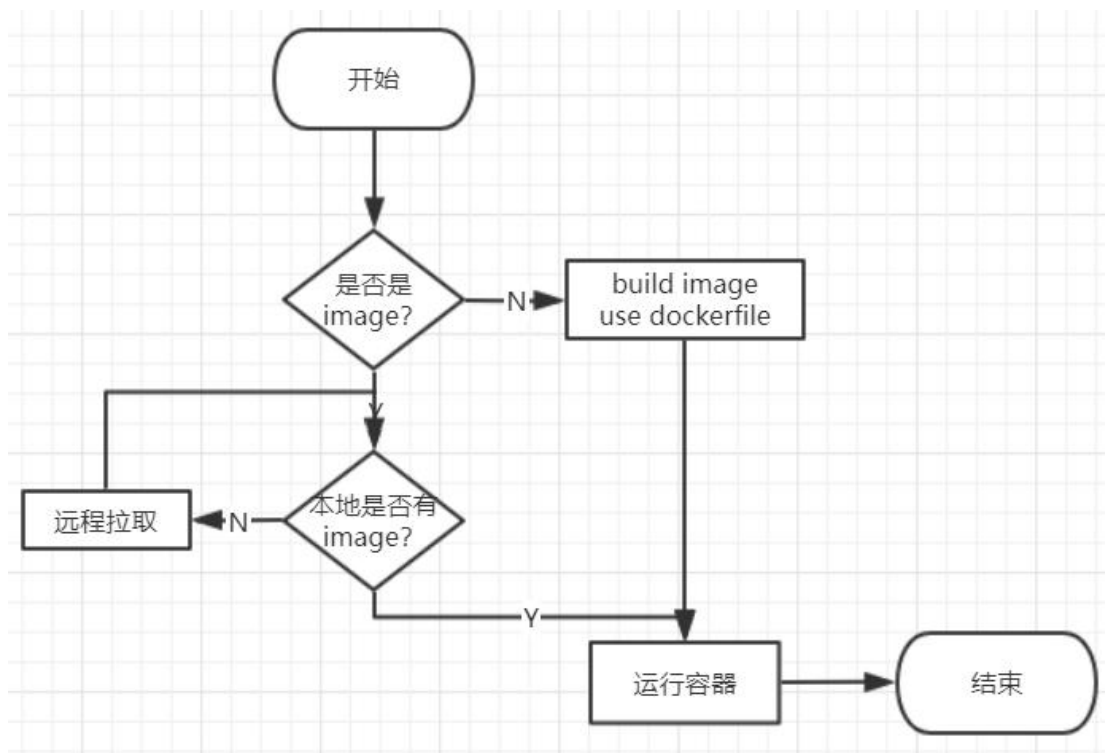


图 8 docker 启动流程图

4.5 本章小结

本章主要将本系统的架构设计，采用 MTV 架构实现。以及本系统的核心功能设计，商品的订单流程设计、后台的订单处理设计、图书和用户的管理设计和系统的其他功能模块设计。举例非关系型数据库设计，说明了图书和订单集合的设计。最后简单的介绍了 Docker-compose 的部署。

5 系统的实现

本章结合前两章中的需求分析和系统设计部分的内容,在结合第二章中提到的开发工具,进行详细的介绍本系统核心功能的实现过程。

5.1 环境搭建

5.1.1 开发环境搭建

本系统是基于 Ubuntu 16.04 Desktop 操作系统上进行开发的,编程语言采用操作系统自带的 Python 3.5 版本。因此省了编程语言的安装过程,IDE 工具则是使用 Pycharm。

python 相应的库的安装则是采用 requirements.txt 文件,不管是在 Pycharm 中,还是在部署的时候都能很方便的结合 pip 来管理第三方库。前端页面测试工具则是使用 Chrome/Firefox,方便页面调试和修改。

5.1.2 部署环境搭建

虽然本系统的各个服务工具都是部署在 Docker 容器中,而 Docker 在 Windows 和 Linux 系统都可以安装。由于 Linux 在服务器方面有较强的稳定性,因此将部署的系统环境也是 Ubuntu16.04 系统。

Docker 的安装,首先需要添加 Docker 官方的 GPG 秘钥到系统中:“curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg”。然后可以使用其官方提供的 shell 脚本进行相应的安装包下载以及安装,“curl -fsSL https://get.docker.com -o get-docker.sh”。

大多数时候安装 Docker 的时候都自带安装了 docker-compose 工具,如果没有默认安装到,也可以使用“sudo apt-get docker-compose”进行安装。

5.2 Flask 项目搭建和 Blueprint

因为本系统分为前台和后台管理,所以 Flask 项目的入口将采用工厂函数的形式实现。而各个功能之间则使用 Blueprint 将视图区分来,最终将各个功能视图统一在工厂函数中进行注册。其核心代码如下 app.py:

```
def create_app(test_config=None):
    app = Flask(__name__, instance_relative_config=True)
    app.config.from_mapping(
        SECRET_KEY='secret',
    )
    # 自定义 404 错误页面
    app.register_error_handler(404, page_not_found)
    # 注册 user 蓝图，用于登录、注册、注销、账户管理等
    app.register_blueprint(user.bp)
    app.register_blueprint(products.bp)
    . . .
    app.register_blueprint(userinfo.bp)
    app.register_blueprint(admin.bp)
    return app
```

一些功能在实现上相类似的视图将作为同一个模块存放在相同的 **Blueprint** 中。例如，用户的登录、注册，以及退出登录等功能。

使用 **Blueprint** 可以实现视图和入口函数的配置分离，每个视图只负责各自的功能实现就可以了。前台主页视图的核心代码如下：

```
bp = Blueprint('products', __name__)

bp.route('/')
def index():
    """主页"""
    books, new_books, book_top, book_top2 = index_model()
    book_type_list = choice_book_type()
    return render_template('front/index_products/index.html',
```

```

books=books,

new_books=new_books,

book_top=book_top,

book_top2=book_top2,

book_type_list=book_type_list,

)

```

在这里的视图函数做的事情很简单，定义本视图的 URL 以及对应的路由函数 `index()`，然后调用相应的 `model` 和 `Template`，并将 Jinja2 渲染过的 HTML 页面返回即可，没有复制的业务逻辑和请求参数。Flask 项目目录如图 9 所示。

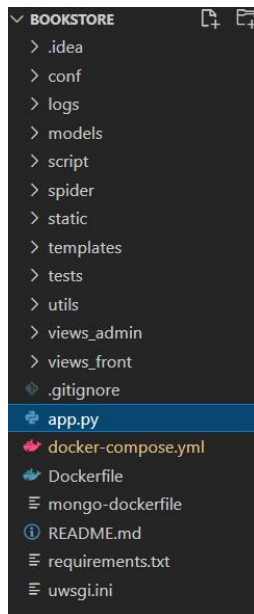


图 9 项目目录

5.3 系统前台实现

5.3.1 index 主页实现

主页顶部导航栏实现，分为用户导航栏和图书类型导航栏。用户导航栏提供用户快速查询个人信息的链接。图书导航栏提供给用户查找感兴趣的图书，可以模糊匹配图名、简介、作者和出版社。右侧的导航栏使用 AngularJS 异步获取随机的几种图书类型。主页菜单设计实现如图 10 所示。



图 10 主页菜单实现

主页内容的实现，因为主页是本系统前台的首页，大部分的用户看到的都是这个页面，因此将五个部分放置在了主页，分别是广告轮播、读者推荐、本月新书推荐、新书榜和畅销榜。由于篇幅较长所以会有较长的滚动条，所以设置了图书导航栏为吸顶效果，会跟随页面向下滚动而固定在顶部位置，方便用户搜索和查看其他类型的图书。并且右下角也设置了“回顶部”效果，方便用户快速回到顶部位置。系统主页设计实现如图 11 所示。



图 11 主页设计实现

5.3.2 商品详情页实现

普通的游客用户也可以访问商品的详情页面，此页面有图书的封面图片信息、书名、简介、作者、出版社、出版时间和价格等信息。已登录的用户可以在此页面选择适当的数量，并将物品加入到购物车中，或者是立即购买，即直接到结算页面。图书详情设计如图 12 所示。



图 12 图书详情设计实现图

商品详情页的底部有已购买了的用户对此物品的打分、评论、晒照和评论时间等。用于评论实现如图 13 所示。



图 13 用户评论实现

5.3.3 购物车实现

购物车功能需要用户进行登录后才能进行访问。没有登录的用户尝试访问购物车则会跳转登录页面。用户可以对购物车里面的物品数量进行简单的加减操作，这里使用 JQuery 的 Ajax 进行局部的请求，并将请求响应的结果刷新的对应的数量中。当减到小于 1 时为请求将物品移除购物车。底部的商品总件数与总金额，则使用 JavaScript 实现求和。购物车功能实现如图 14 所示。

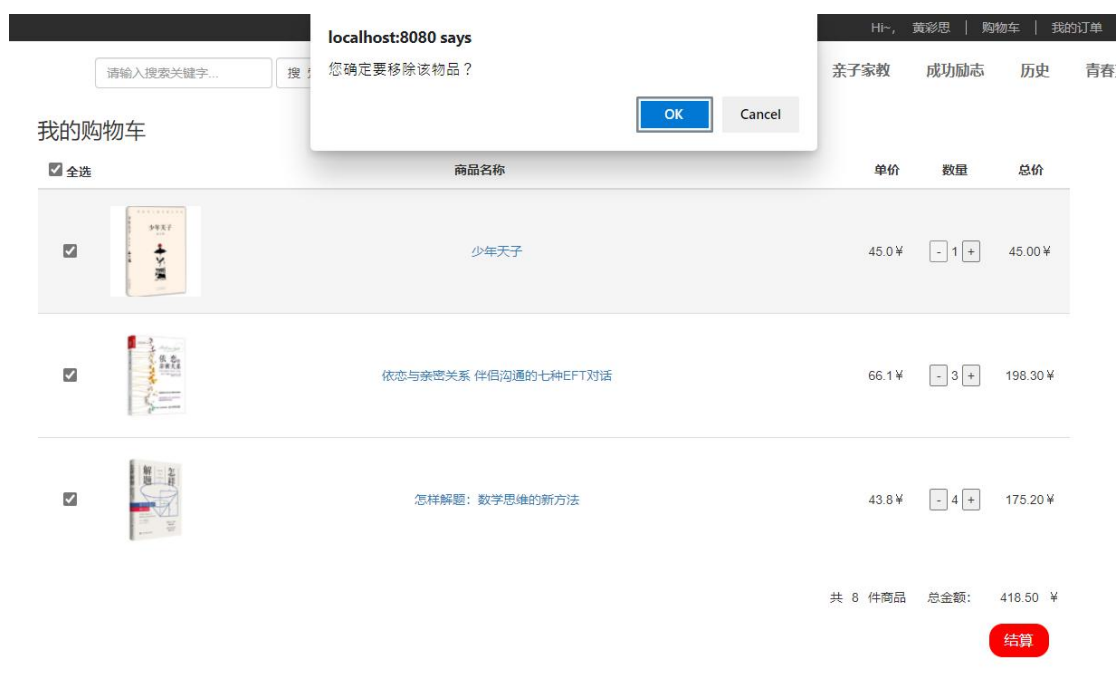


图 14 购物车实现图

5.3.4 结算页实现

结算页面计算用户需要购物的商品的数量和金额，以及系统的一些优惠打折活动。用户需要在此页面确定商品的收货人信息，可以删除现有的默认收货人信息，或者编辑现有的信息，也可以新增信息。新增的将作为默认的处理，也可以到个人信息中的收货人信息菜单中选择需要的收货地址。结算页面功能实现如图 15 所示。

收货人信息

黄彩思 13481470145

北京 密云区 城区以外 大道中2

编辑 删除

新增收货地址

收货清单

商品名称	单价	数量	总价
 怎样解题：数学思维的新方法	43.8	2	87.6

1个包裹 商品金额: 87.6 ¥ 运费: 0.0 ¥ 促销优惠: -1.01 ¥

合计: 86.59 ¥

赠送至: 北京 密云区 大道中2 收货人: 黄彩思 13481470145
 预计2021年03月20日送达
 共2件商品 应付金额: 86.59 (含运费0.0元)

去支付

图 15 结算页面实现图

5.3.5 用户订单页实现

用户生成的所有的订单，都可以在我的订单中进行相应的操作。在全部列表包含了所有的订单内容用户可以对未支付的订单，并且还在有效时间内，可通过支付按钮去继续支付，否则超过了时间则变为失效的订单。也可以取消未支付的订单。

用户支付了订单将会在后台管理中，并且其订单状态在用户中变为待发货。发货人员可以对订单的物品打包发货。而已发货的订单状态则变为待收货状态，用户确认收货后就可以对购买的商品进行评价。

从用户支付了订单的费用后，都可以提交退款申请，并等待系统后台管理人员进行处理。对于已完成和已失效的订单用户可以选择将其删除。正在进行交易流程的订单会在对应的标头中显示其数量的徽章，如待付款①、待发货②、待收货②、待评价①，提示用户查看对应的订单状态。用户订单页实现如图 16 所示。

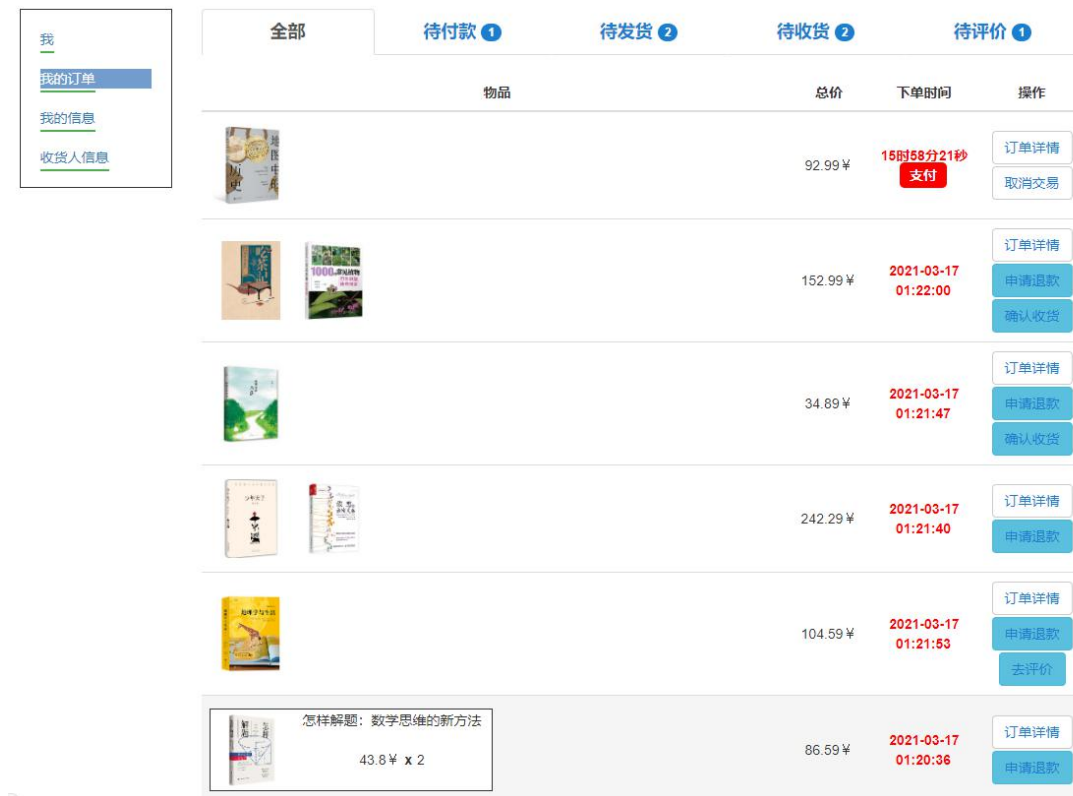


图 16 “我的订单”实现图

5.3.6 订单详情页实现

订单的详情页面展示了此订单的配送地址、收货人、订单号、订单金额、订单状态、物流信息、商品的信息和下单的具体时间等。对于未支付的订单，用户也可以在此页面继续进行支付，而已支付的订单，则可以在此页面中进行退款申请。订单详情实现如图 17 所示。



图 17 订单详情实现图

5.4 系统后台管理实现

5.4.1 后台管理主页

后台管理的主要是系统的数据可视化界面，由 pyecharts 绘制。在用户登录成功后就当天的访问量加 1，用于绘制近期本系统访问量情况的散列图和玫瑰图。用户的每次搜索关键字也进行了保存，作为词云图进行绘制。还有用户每次点击图书进行查看详情信息，将作为其商品的点击量保存，作为其点击量 TOP 进行柱形图绘制。用户购买物品的数量，也作为销售 TOP 绘制柱形图。后台管理主页数据可视化实现如图 18 所示。

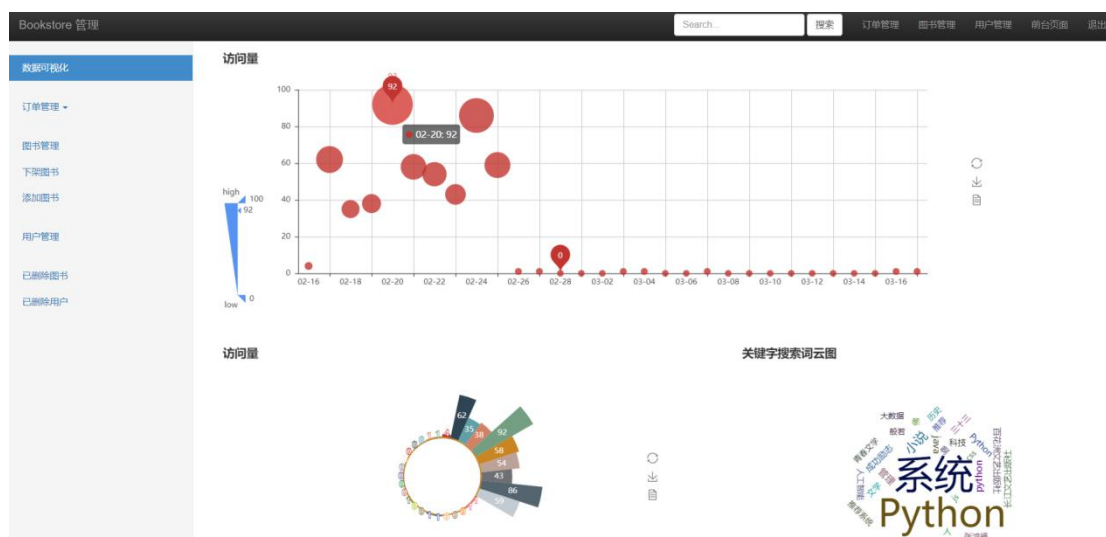


图 18 后台管理主页实现图

5.4.2 订单管理

订单管理有多个子菜单，待发货、待付款、待退款、待收货、待评价、成功和失效订单，分别对于其订单状态列表。列表展示其订单号、用户号、下单时间、图书缩略信息、收货人信息和订单总价。图书信息中有此图书的扩展详情按钮，点击可以查看隐藏的商品的详情，使用 bootstrap 的 JavaScript 差距 collapse，可以将 div 中的内容隐藏，通过点击按钮事件时显示，可以节省占用空间，并且满足展示需求。最后是管理人员可以对此订单的对应操作，如已发货的订单将其确认处理，处理用户的退款申请等。订单管理实现如图 19 所示。

失效订单中需要查看所有的，因此用户删除的订单并不做真正的硬删除，只打上删除的标记。否则在后台中也没有全部的失效订单存在。



图 19 订单管理实现图

5.4.3 图书管理

图书管理实现，用户可以对图书进行上架和下架操作。只有处在下架状态的图书才能对其进行删除操作。也可以查看图书的详情，在图书的详情页中，可以进一步的编辑修改图书的信息。添加图书的操作跟修改图书的类似，只不过多了上传图片的步骤。后台管理实现如图 20 所示。



图 20 下架图书实现图

5.4.4 用户管理

在用户管理页面，管理员可以对用户的账号进行重置密码、冻结和计划账号、删除等操作。重置密码功能系统会随机生成 8 位随机的密码。冻结的用户账号将不能正常的登录本系统，需要联系管理员进行处理。后台用户管理实现如图 21 所示。

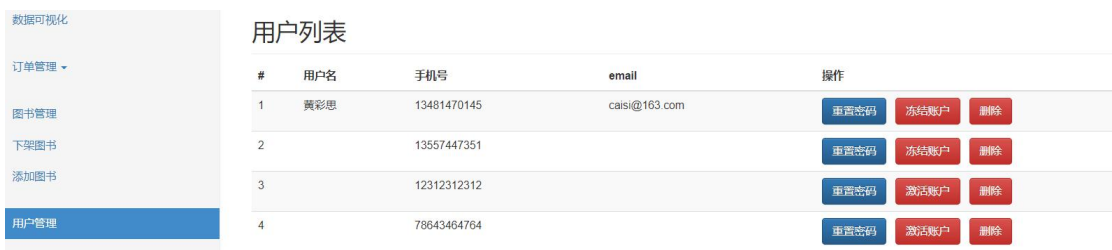


图 21 用户管理实现

5.5 本章小结

本章对系统的核心功能实现做了详细的说明，包括系统开发环境和部署环境的介绍，Flask 项目的工厂模式搭建结构，系统的前台、后台管理中主要功能的实现和运行截图。

6 系统测试

网络环境是不安全的，也没有一个系统不存在安全漏洞。而系统开发中占据重要步骤的测试可以在上线前发现已知的系统漏洞，一定程度上减少系统漏洞，提高系统的安全性。据统计，在软件开发总成本中，用在测试上的开销要占 20% 到 50%^[10]。测试一般有功能测试、性能测试、API 测试和单元测试等。下面将对系统进行功能测试和性能测试。

6.1 功能测试

软件测试是为了发现错误而执行程序的过程^[11]。功能测试按照系统的需求分析进行编写测试用例，提前发现 bug，保证上线时能给用户友好的系统体验。

登录与注册模块测试用例见表 3。

表 3 登录与注册测试用例

功能模块	ID	测试步骤	预期结果	测试结果
登录与注册	01	点击主页的“登录”	弹出登录界面	符合预期
	02	在详情页点击“加入购物车”	跳转登录界面	符合预期
	03	使用任意账号密码登录	提示用户名或密码不正确	符合预期
	04	在登录页面点击“去注册”	跳转注册页面	符合预期
	05	输入 11 位手机号和 8 位密码，点击注册	注册成功跳转登录页面	符合预期

购物车模块测试用例见表 4。

表 4 购物车测试用例

功能模块	ID	测试步骤	预期结果	测试结果
购物车	01	在商品详情页点击“加入购物车”	跳转添加成功页面	符合预期
	02	在搜索结果页点击“加入购物车”	跳转添加成功页面	符合预期

03	点击“购物车”	跳转到购物车页面	符合预期
04	点击勾选全选按钮	所有商品被全选中,并计算出数量和总金额	符合预期
05	点击“+”、“-”按钮	数量和总金额跟随变化,刷新页面数量也不变	符合预期
06	当数量为1时,点击“-”	弹出移除确认框	符合预期
07	点击移除确认框	物品被移除	符合预期
08	取消全选,点击“结算”	弹出没有选中任何物品提示框	符合预期

结算模块测试用例见表 5。

表 5 结算功能测试用例

功能模块	ID	测试步骤	预期结果	测试结果
结算	01	购物车点击“结算”或商品详情页点击“立即购买”	跳转到结算页面	符合预期
	02	无收货人信息时,点击“去支付”	弹出需要添加收货人提示框	符合预期
	03	点击“新增收货地址”按钮	弹出填写收货人信息输入框	符合预期
	04	输入收货人信息并提交	页面刷新并存在添加的信息	符合预期

支付模块测试用例见表 6。

表 6 支付功能测试用例

功能模块	ID	测试步骤	预期结果	测试结果
支付	01	在结算页面点击“去支付”	跳转到支付页面	符合预期

02	在“我的订单”中点击有效的未支付的订单的“支付”按钮	跳转到支付页面	符合预期
03	在未支付的订单的详情页面点击“支付”按钮	跳转到支付页面	符合预期
04	点击“立即支付”	弹出模拟支付窗口	符合预期

我的订单模块测试用例见表 7。

表 7 “我的订单”功能测试用例

功能模块	ID	测试步骤	预期结果	测试结果
我的订单	01	点击“我的订单”按钮	跳转到订单页面	符合预期
	02	点击含有①徽章的待支付标头	跳转到待支付的订单页面	符合预期
	03	点击订单的“订单详情”	跳转到订单详情页面	符合预期
	04	点击“取消交易”	待支付中无订单	符合预期
	05	点击“删除”，并确定	全部列表中订单被删除	符合预期

后台订单管理模块测试用例见表 8。

表 8 后台订单管理功能测试用例

功能模块	ID	测试步骤	预期结果	测试结果
后台订单管理	01	点击“待发货”按钮	跳转到待发货订单列表	符合预期
	02	点击图书信息的 	显示详细的图书信息	符合预期
	03	点击订单的“确认已发货”按钮	此订单转移到“待收货”列表	符合预期
	04	点击待退款的“确认退款”	此订单转移到“失	符合预期

败”列表

后台图书管理模块测试用例见表 9。

表 9 后台图书管理功能测试用例

功能模块	ID	测试步骤	预期结果	测试结果
后台图书管理	01	点击图书的“下架”按钮	此图书将移到“下架图书”列表	符合预期
	02	点击图书的“详情”->“编辑”，详情中图书信息修改图书信息	被修改	符合预期
	03	点击“添加图书”，并输入信息和上传图片，提交	搜索添加的图书，存在结果	符合预期
	04	点击“删除”按钮，并确认	此图书转移到“已删除”列表	符合预期

后台用户管理模块测试用例见表 10。

表 10 后台用户管理功能测试用例

功能模块	ID	测试步骤	预期结果	测试结果
后台用户管理	01	点击“重置密码”并确认	弹出重置后的 8 位随机密码	符合预期
	02	点击“冻结账号”并确认	此账号登录时提示“已被冻结”	符合预期
	03	点击“删除”按钮，并确认	此账号转移到“已删除”列表	符合预期

6.2 性能测试

性能测试可以对系统的运行状态进行综合的测试。这里使用的测试工具是 Apache Jmeter Web，其测试脚本使用 badboy 工具进行获取，保存为 Jmeter 脚本。调试好测试脚本后使用“jmeter -n -t result-view\bookstore.jmx -l bookstore_result -e -o C:\Users\Caisi\Desktop\result”命令执行测试脚本。

使用 100 个线程执行 5 个用户从登录到生成支付订单过程的地步测试结果如

图 22 所示。

Statistics													
Requests		Executions			Response Times (ms)							Throughput	Network (KB/sec)
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	1000	0	0.00%	2685.48	165	4764	2736.50	4251.90	4489.65	4696.96	35.37	490.94	0.00
http://localhost/product%3F5ee96be5360d930a489db21b	100	0	0.00%	4143.73	3516	4732	4159.00	4624.60	4668.85	4731.75	8.46	102.23	0.00
http://localhost/user_login_register/index_login	100	0	0.00%	4036.59	1078	4764	4278.50	4676.60	4717.30	4763.81	10.62	3.18	0.00
http://localhost/product/add_to_cart	100	0	0.00%	3556.99	3369	4595	3457.50	4098.40	4358.25	4594.72	8.93	85.49	0.00
http://localhost/cart	100	0	0.00%	2765.41	1893	3513	2794.00	3389.80	3454.40	3512.76	11.78	148.29	0.00
http://localhost/buy_llst	200	0	0.00%	2588.53	1898	3270	2657.00	2965.90	3026.75	3227.54	14.96	192.06	0.00
http://localhost/cart#	100	0	0.00%	2284.05	1308	2969	2303.50	2944.80	2948.95	2968.98	16.70	175.45	0.00
http://localhost/	100	0	0.00%	2069.46	165	3912	2038.50	3569.60	3796.65	3911.92	20.53	990.98	0.00
http://localhost/to_pay	100	0	0.00%	1563.48	1235	2961	1425.50	2221.70	2867.45	2960.85	17.38	3.39	0.00
http://localhost/pay	100	0	0.00%	1258.06	1018	1478	1273.00	1426.90	1446.70	1477.84	27.04	530.74	0.00

图 22 性能测试结果图

其中共九个请求 URL，在模拟一百个用户的情况下，错误率为 0，平均响应时间为 2.6 秒，网络传输速率平均为 490KB/sec，基本符合系统预期结果。

6.3 结果分析

通过本章的功能测试和性能测试，系统的实现符合第三章对系统所做的功能性需求，在系统的压力测试中，系统依然能够良好的运行，并未出现服务器宕机、响应错误的情况，满足系统的需求。

6.4 本章小结

本章主要对系统的核心功能进行了测试用例的编写，对系统的功能测试，对其测试出现的问题进行修复，以满足系统的需求设计。最后对部署的系统 Web 服务使用 Jmeter 进行了相关的测试，结果满足系统需求。

7 总结与展望

7.1 总结

本文介绍了基于 Flask 和 MongoDB 的 Web 应用的设计和开发过程。通过查阅相关的资料，了解目前 Web 应用的研究现状，在大数据时代的背景下，网上书店的发展设计方向。结合目前所学的知识，通过对比传统的关系型数据和非关系型数据库的 Web 应用，得出分布式的非关系型数据库是今后发展的方向。因此选择了 MongoDB 作为本系统设计的底层数据存储服务。

结合当下流行的技术，使用轻量级的 Flask 框架作为 Web 应用进行开发，使用 uWSGI 作为 Flask 应用的启动服务器，Nginx 作为系统的代理服务器进行部署上线。系统采用的是 MTV 架构，对系统做了详细的需求分析，系统的功能设计和数据库设计，由于 MongoDB 的惰性，因此在使用上跟以往有很大的不同。在系统的实现上，通过 Flask 的 Blueprint 将不同的功能进行了拆分。将开发完成的应用提交到 GitHub 远程仓库进行管理，并使用 Docker 容器进行一键轻松部署，方便今后的系统功能扩展、发布等。除了开发并行测试外，还对系统进行了功能测试和性能测试，并对结果进行可行性分析，结果符合预期。

7.2 展望

互联网技术的技术迭代非常的快，虽然本项目在部署上使用了 Docker 容器技术，在部署上云服务器方面较方便，但是现在许多的容器服务都向 k8s 发展，因为 k8s 在服务的治理和安全方面比 Docker 更好、更全。在云服务方面，也有更多的高并发、需要敏捷开发、快速交付的应用往微服务方面发展。将整个应用分解为多个微服务，各个微服务独立运行在自己的进程中，并分别有自己的数据库，微服务之间使用 REST 或者其他协议通信^[12]。而目前，容器技术是实现微服务最好的选择方案。

本系统在后续开发和运维方面还有许多的不足之处，后续可以往 CICD 方面完善，使软件的开发、测试、部署更简单，达到快速迭代版本，快速交付产品的目的。

本系统还有许多的功能需要进一步的完善。在技术方面，由于前端技术的欠

缺导致前台页面的风格有许多不一致的地方，后续可以往前后端分离的 API 方面进行接口开发，可以有更多的复用功能使用一个接口解决，在和用户交互上也将更加的友好。在系统的性能上，可以对性能测试结果中响应较高的 URL 采用 Redis 数据库缓存技术，以提高系统的性能，提供更友好的用户体验。

参考文献

- [1] 宁兆龙. 大数据导论[M]. 北京:科学出版社, 2017. 第 6 页
- [2] 杨一雄. 网上书店系统的设计与实现[D]. 厦门大学, 2014.
- [3] 王安瑾. 基于 Flask 的金融自动化运维平台的设计与实现[D]. 上海:东华大学, 2018.
- [4] Fluent Python by Luciano Ramalho (O'Reilly) (中文版)[M]. Copyright 2015 Luciano Ramalho, 978-1-491- 94600-8. 第 301 页.
- [5] 王爱国, 许桂秋. NoSQL 数据库原理与应用[M]. 人民邮电出版社, 2019. 87~90
- [6] Nedelcu, 陶利军译. 学习 Nginx HTTP Server(中文版)[M]. 清华大学出版社, 2012. 217~218
- [7] 杨保华. Docker 技术入门与实战[M]. 机械工业出版社, 2017. 第 14 页.
- [8] 张素芬, 张黎, 杨玲萍. 软件测试需求分析方法和过程研究[J]. 信息化研究, 2017(1):56-61.
- [9] 陈品华, 李鸿君. ASP.NET 项目开发实践教程[M]. 武汉大学出版社, 2016. 10: 第 210 页
- [10] Elfriede Dustin. 有效软件测试[M]. 北京:清华大学出版社, 2004. 52~55
- [11] Ron Patton. Software Testing[M]. 北京:机械工业出版社, 2006. 11~12
- [12] 周立. Spring Cloud 与 Docker 微服务架构实战[M]. 北京:电子工业出版社, 2017. 5. 第 4 页

致谢

四年的本科学习生涯转瞬即逝，在这不算长，也不算短的时间里，有付出，也有收获；回首四年前的无知和懵懂，成长了许多。

在这短旅途中感谢父母、老师和同学们的陪伴。感谢学校的导师和专业合作的老师的指导和建议，帮助我提前确定自己的方向。其次，感谢学校的领导老师们，对此自选题目的论文的支持和认可。还有，感谢绿盟科技武汉研发提供的实现经历，让我能扩展自己的眼界，认识到自己的不足，并提出了整改的建议。感谢那些在实习期间认识给我帮助和陪伴的人。

最后，感谢所以帮助、支持，以及陪伴，见证我成长的每个人。

声 明

本人郑重声明：本论文（设计）是本人在广西科技师范学院学习期间，在指导教师指导下独立完成的。其内容真实可靠，如存在抄袭、剽窃现象，本人愿承担全部责任。

同时，本人完全了解并愿意遵守广西科技师范学院有关保存、使用毕业论文（设计）的规定，其中包括：

1. 学校有权保管并向有关部门递交毕业论文（设计）的原件与复印件。
2. 学校可以采用影印、缩印或其他复制方式保存毕业论文（设计）。
3. 学校可以以学术交流为目的，赠送和交换毕业论文（设计）。
4. 学校可以允许毕业论文（设计）被查阅或借阅。
5. 学校可以按著作权法的规定公布毕业论文（设计）的全部内容或部分内容（保密毕业论文（设计）在解密后遵守此规定）。

除非另有科研合同或其他法律文书制约，本论文的科研成果属于广西科技师范学院。

特此声明！

声明人签名：

二〇二一年 月 日