北京理工大學

本科生毕业设计（论文）

基于 Android 的二手商品实时比价系统的设计与实现

Design and Implementation of Second-Hand goods price comparison Based on Android

学　　院：计算机学院

专　　业：计算机科学与技术

班　　级：07112006

学生姓名：陈誉航

学　　号：1820201062

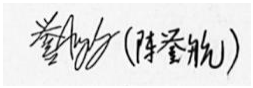
指导教师：汤世平

2024 年　5 月 30 日

原创性声明

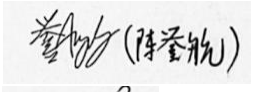
本人郑重声明：所呈交的毕业设计（论文），是本人在指导老师的指导下独立进行研究所取得的成果。除文中已经注明引用的内容外，本文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。

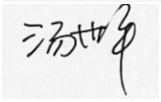
特此申明。

本人签名：  日 期：2024 年 5 月 30 日

关于使用授权的声明

本人完全了解北京理工大学有关保管、使用毕业设计（论文）的规定，其中包括：①学校有权保管、并向有关部门送交本毕业设计（论文）的原件与复印件；②学校可以采用影印、缩印或其它复制手段复制并保存本毕业设计（论文）；③学校可允许本毕业设计（论文）被查阅或借阅；④学校可以学术交流为目的,复制赠送和交换本毕业设计（论文）；⑤学校可以公布本毕业设计（论文）的全部或部分内容。

本人签名：  日 期：2024 年 5 月 30 日

指导老师签名：  日 期：2024 年 5 月 30 日

基于 Android 的二手商品实时比价系统的设计与实现

摘　要

随着互联网和人工智能的迅猛发展，绿色消费理念日益深入人心，人们对二手商品，尤其是电子产品，更加愿意进行购买。这是因为电子产品的迭代速度非常快，导致旧款电子产品的价格下跌。同时，现在市面上有许多售卖二手商品的平台，要找到适合价格的商品，需要在多个二手商品平台进行反复查询，这会浪费用户大量时间。本文设计了一个基于安卓的二手商品实时比价系统，该系统基于爬虫技术，采用服务端与客户端分离的架构，实现了多数据源的二手商品实时价格比较功能。

论文的主要工作和成果如下：

1) 基于Scrapy框架设计了一个能够高效爬取二手商品数据爬取子系统，并将数据保存到ElasticSearch内，同时也应能够通过添加解析文件将新数据源接入本系统，提高了系统的可扩展性。

2) 采用TF-Cosine+LD混合算法对爬取的数据进行商品型号的分类，确保商品分类到正确的型号下。使用真实数据对算法进行了有效性测试。

3）使用ReactNative框架设计实现基于安卓的二手商品实时比价系统。首先对系统需求进行了分析，明确所需的功能，包括商品比价、商品搜索、商品订阅。在此基础上进行前后端的架构设计，实现和测试该系统。

关键字：增量式爬虫，商品分类，价格比较

Design and Implementation of Second-Hand goods price comparison Based on Android

Abstract

With the rapid development of the internet and artificial intelligence, the concept of green consumption has become increasingly ingrained in people's minds. Consumers are more willing to purchase second-hand goods, especially electronic products, due to the fast iteration cycle of these items, which leads to significant price drops for older models. However, finding the best prices for such products often requires checking multiple second-hand platforms, which can be time-consuming. This paper designs an Android-based real-time price comparison system for second-hand goods, utilizing web scraping technology and a client-server architecture to achieve real-time price comparison across multiple data sources.

The main tasks and results of this paper are as follows:

1) Designed a subsystem based on the Scrapy framework for efficiently scraping second-hand product data and save the data to ElasticSearch. The system is designed to be extensible by allowing new data sources to be integrated into the system through the addition of parsing files, thus improving system scalability.

2) Applied a hybrid TF-Cosine+LD algorithm to classify the scraped data into correct product models, ensuring accurate product categorization. The algorithm's effectiveness was tested using real data.

3) Developed an Android-based real-time price comparison system for second-hand goods using the React Native framework. The paper first analyzed the system requirements and identified the necessary functionalities, including price comparison, product search, and product subscription. Based on this analysis, the client-server architecture was designed, implemented, and tested.

Key Words: incremental scraping, product classification, price comparison

目　录

第 1 章 绪论

1.1 研究背景

移动互联网技术的迅猛发展推动了电商行业进入了一个低价与服务优势、价格至上的时代。根据一份研究报告显示，截至2022年，中国网络购物用户规模已达8.5亿，占网民总数的近80%[1]。研究还指出消费者购物行为与商品价格密切相关，使得价格成为影响用户选择的最重要因素，从曾经的第三位跃升至第一位。随着互联网上出现越来越多的知名二手电商平台，如闲鱼、爱回收等，用户为了获取最佳利益不得不在各平台之间比价。

央视财经报道显示，随着绿色消费理念的深入，二手物品的消费量逐年增加。调查结果显示，国内二手手机已经形成了比较规范、成熟的市场。2020年，国内二手手机交易量达1.52亿台。同时，闲置服装、服饰也受到人们的青睐[2]。然而，许多二手平台，仅限于搜索自己平台的商品，无法搜索其他平台的商品。在这个商品销售量大且更新迭代频繁的时代，消费者往往需要在各大平台对商品进行价格比较，耗费大量时间和精力，给用户带来不佳的体验。同时这些平台也不全部提供对于特定商品订阅功能。除此之外，文献[3]解释了同一个商品会因为其所在国家而导致价格上差异。所以本文研究的系统提供了多平台商品的汇总、部分商品的不同国家历史最低价格以及商品订阅功能。

对于正处于工作或学习的人群，他们应该如何在耗费最少的时间在海量商品内以寻找更符合自己要求的商品及其重要。由于不同二手平台的价格有差异，同一商品的销售价格也会不一致。图 1-1展示了中国二手平台爱回收（左）以及马来西亚二手平台Mudah（右）分别查询佳能M50 Mark II的结果，搜索发现爱回收佳能相机的平均价格为4379（人民币），而Mudah平台的佳能相机平均价格为2400（马币），经过汇率转换约为3660（人民币），可以发现不同国家所出售的商品价格差距可以是非常的巨大的。为了使消费者在消耗最少时间内成功找到符合要求的商品，本文开发了基于安卓的二手商品实时比价系统。

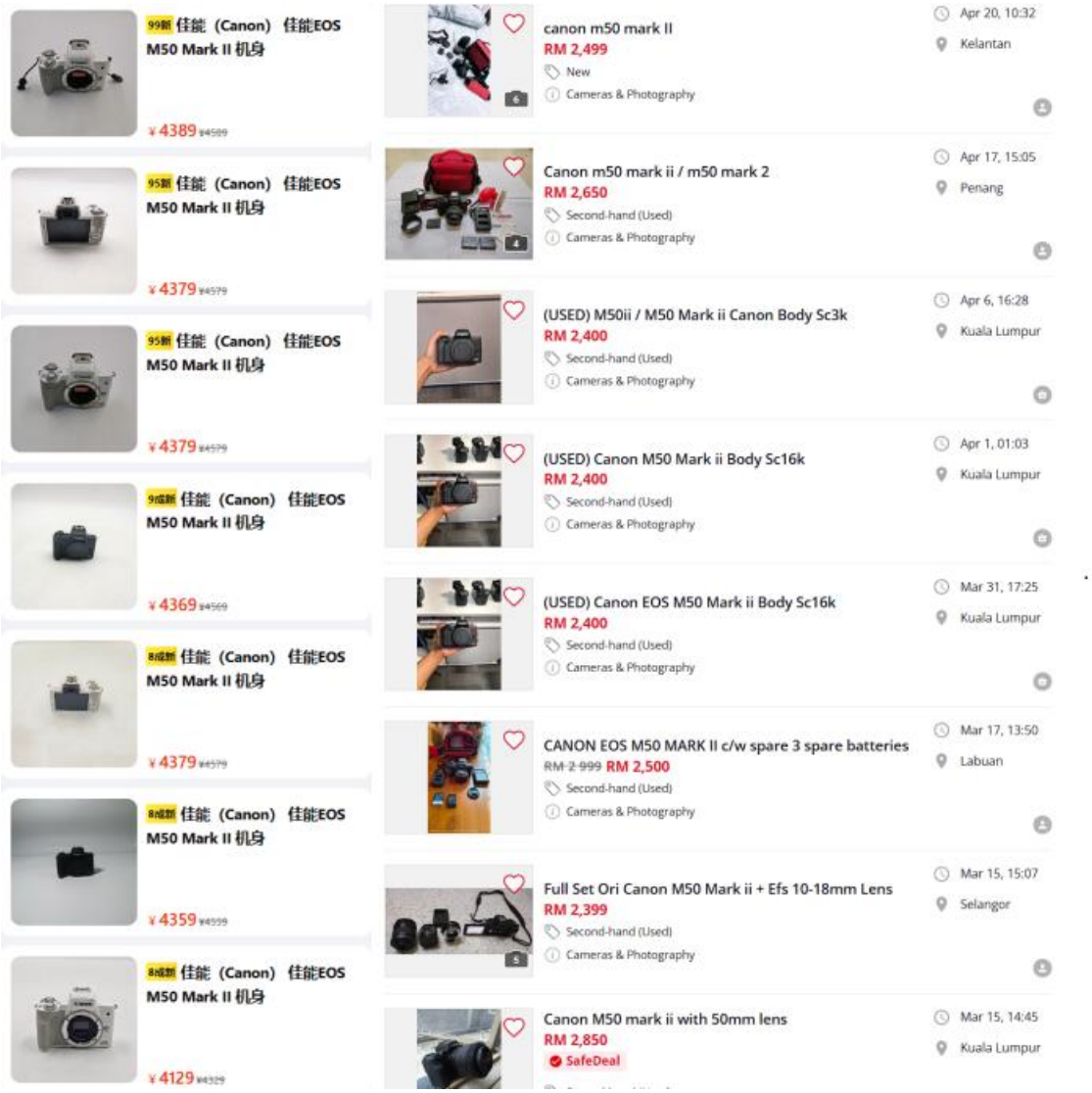


图 1-1 Mudah 以及爱回收查询佳能 M50 Mark II 结果

1.2 国内外研究现状

1.2.1 国外研究现状

国外早期已经出现了商品比价系统，并取得了相较国内更多的研究成果。当前，国外的商品比价系统已经相当完善，实现了多个不同平台商品的集成。浏览这些国外系统时，可以发现不同平台各有优点，这些优点吸引消费者在购买商品时先使用比价系统进行搜索，然后再前往对应官网查询，以获取心仪的商品。

文献[4]是数名孟加拉UIU大学的学生进行动态商品比价系统的研究，他们所实现的系统为使用者提供了一个综合平台，使得使用者能够做出更加明智的选择。其中

他们研究中的重大贡献在于用户在进行搜索商品可以以多语言进行搜索并返回同样的结果，目前他们所提供的语言包括孟加拉语言、拉丁化孟加拉语以及英语。

文献[5]是数名印度大学生进行基于机器学习的智能商品比价系统的研究，他们利用了支持向量机SVM算法基于商品的特征以及排名值来进行分类。通过支持向量机SVM算法进行商品的分类他们获得的商品分类准确率达到了94.71%。

除此之外，通过网上游览也发现几个国外以上线的此类系统，其中包含有：

1. PriceRunner：为英国、瑞士、挪威以及丹麦提供商品比价。他们每日更新126,000,000件商品价格，并且还对运费进行了计算，仅限能发货到英国的销售商。并且该平台将产品分类的非常清楚明了，也提供了许多供用户选择商品的参数，使其成为一个成功的比价系统。

2. ShopZilla：是Connexity公司旗下的一个产品，该公司有在美国洛杉矶、加利福利亚创办办公区。同样的，ShopZilla也提供了一个简单的界面，并且提供了所需的查找参数供用户能够更详细的查找想要的商品。

1.2.2 国内研究现状

何毅平，黄媛，湛茂溪，陈庚等人[6]设计并实现了一个基于网络爬虫的招聘信息可视化系统。他们主要解决的问题是当前社会人员求职的困难。通过爬取招聘网站的信息，并利用大数据分析技术发掘数据中隐含的未知趋势，他们帮助求职者了解当前局面并做好求职准备。此项研究充分说明了爬虫技术结合大数据往往可以得到隐含的信息。

平奥琦[6]设计实现了一个安卓端跨平台商品多维度比价系统，其主要创新点是多维度商品比价。第一步是使用最长公共子序列+最短编译距离对商品标题关键字和数据库中的商品数据集进行文本相似度计算，最后在通过感知哈希算法计算图片相似度，两轮的验证来保证商品的准确性。

除此之外，通过游览网站发现几个国内的此类系统，其中包含有：

1. 慢慢买：成立于2010年，从单纯的比价网站，发展称为“一个中立的商品搜索推荐引擎”，包含折扣爆料、历史价格走势、评论曝光、降价提醒等功能，给用户良好的购物体验。

2. 淘客联盟：该平台汇聚了网上所有声望高的电商平台，用户可以第一时间得到各个网站打折促销的信息。除此之外，该平台也会向用户提供淘宝、京东、拼多多商品平台优惠卷。

3. 什么值得买：是一个综合性的购物导购平台，提供了商品比价、折扣信息、用户评价等多方面的服务。

通过了解国内外的商品比价系统，发现这些系统主要关注新品，并提供了优质的搜索参数和历史最低价格功能。然而，本次研究将重点放在二手商品比价上，并致力于覆盖其他国家的价格，以便让用户了解该商品在其他国家的售价情况。这一选择既符合绿色消费理念，也为有购买二手商品需求的用户提供了一个类似的平台来搜索心仪的商品。

1.3 本文的主要工作和论文组织结构

1.3.1 主要工作

本文的研究内容如下：

1. 通过查询并阅读相关文献了解现有比价系统的实现方法，并了解目前比价系统的特点以及不足，以便明确本系统的突出点以及如何给用户带来更好的购物体验。

2. 了解Scrapy框架并利用该框架进行数据的爬取、清理以及持久化的操作。针对不同的二手电商平台的页面结构和反爬机制进行分析，设计合适的爬取策略，以最小的请求数量获取最多的数据。

3. 了解Elasticsearch框架，并利用该框架实现商品爬取的持久化。通过自动创建倒排索引表，辅助系统在用户查询商品时能够快速获取商品列表。

4. 研究和分析布隆过滤器、词频、余弦相似度以及最短距离算法，并利用这些算法来确保爬取的数据经过有效的清理。进行实验测试后发现，结合词干提取的TF-Cosine+LD混合算法在商品分类准确率上都有所改善。同时，布隆过滤器能够有效地实现增量式爬虫。

5. 最后，设计并实现基于Android的二手商品实时比价系统。从系统的需求分析开始，逐步完成了系统的设计目标、功能模块、架构设计、服务器接口设计、功能模块设计以及数据库设计。在这些基础上，对系统进行了详细设计，并对各个功能模块进行了类设计和流程设计，最终进行了系统的功能测试。

1.3.2 论文组织结构

本文共分为6章，各章的主要内容如下：

第一章：介绍了本文所研究内容的背景以及研究意义，随后提出了本文的主要工作以及后续的章节安排

第二章：介绍了本文研究以及实现所使用的技术框架以及算法。

第三章：提出了系统的设计目标，首先分析系统的需求，随后提出了系统的功能性以及非功能性需求。

第四章：提出了系统的架构。架构设计分为总体架构、客户端架构和服务端架构。首先概述总体架构，随后详细介绍客户端和服务端架构。此外，还会详细描述Scrapy的架构设计，因为它在本系统中扮演着重要的角色。

第五章：对系统的功能模块进行了详细论述，包括搜索商品、订阅商品、爬取商品、比价、爬虫以及商品型号匹配等。接下来进行系统的测试，测试各功能模块，并给出测试结果。

最后对本文系统的设计与实现进行一个总结，总结方式是结合系统现有存在的问题以及对未来的工作进行展望。

第 2 章 相关框架及算法介绍

2.1 网络爬虫以及 Scrapy 框架

2.1.1 网络爬虫概念

网络爬虫是一个能够快速收集数据收集的方法，他是按照特定规则自动爬取互联网信息的脚本或者程序，是搜索引擎的重要组成部分[7]。文献[8]里详细介绍了增量式爬虫的底层实现原理。本小节将介绍构成爬虫程序的主要部分然后对不同类型的爬虫进行分类，以下为爬虫程序的主要部分，分别有：

1. 配置文档：通常包含了要爬取的网站URL，通用爬虫参数（包括：爬取深度、对应解析模块等等）。

2. 中央处理模块：包含了定时器、爬虫实例分配、负载均衡。分别提供了何时进行爬取，例如：每天8点进行爬取；根据robots.txt协议分配特定数量实例进行爬取，避免被网站封锁；分配网站给各个爬虫实例进行爬取，减少单个爬虫实例的压力。

3. URL列表模块：该模块会从配置文档传来的URL，并且在后续的URL解析模块获取新URL时加入进来的URL。爬取直到列表为空。

4. 读取URL模块：该模块读取URL的信息并通过配置文件确定使用哪个解析模块对网页信息进行解析。

5. 页面下载模块：从互联网上下载页面。

6. 页面解析模块：对页面进行解析，常用的方法有BeautifulSoup、LXML方法[9]、XPath[10]、Re [11]、CSS Selector等方法。在文献[11]中，作者通过实验比较BeautifulSoup、Re和Xpath得出Re是在里面解析速度最快的，其次是XPath。综合考虑XPath最符合需求，既兼顾了速度以及易编写。

7. 信息解析模块：解析想要从网页获得的特定内容（比如：商品名称、价格、简介等等）

8. URL解析模块：解析网页中包含的URL链接，以获取更多相关的网页。

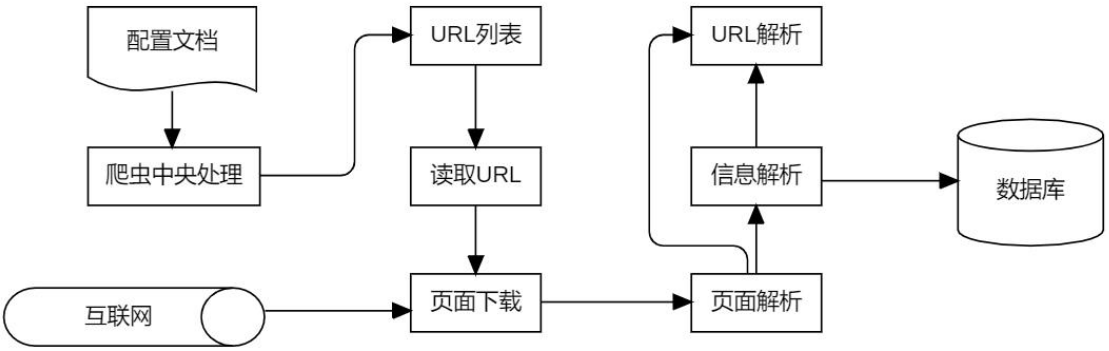


图 2-1 爬虫基本流程

2.1.2 增量式网络爬虫

网络爬虫又分为多个种类[12]：通用网络爬虫、聚焦网络爬虫、增量式网络爬虫。由于本次研究的更接近于增量式网络爬虫，所以只讨论增量式爬虫。

增量式网络爬虫是针对经常动态更新站点的网站，并自动定期抓取新产生或发生变化网页的爬虫 [13]。他的主要好处是减少了时间和空间的耗费，但是在爬取算法复杂度提升了。最简单的增量式网络爬虫的处理流程即在图 2-1爬虫基本流程的基础上添加判断URL是否存在于一个保存已爬去URL的数据库如图 2-2所示，避免再次进行爬取。

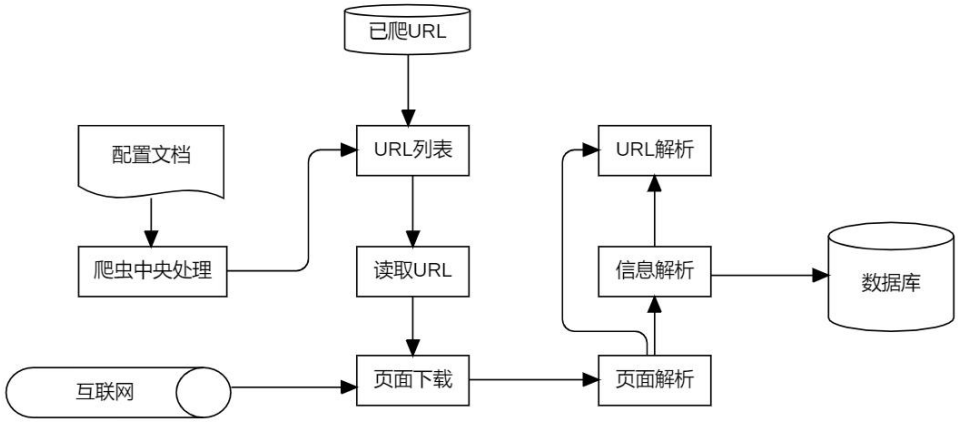


图 2-2 增量式爬虫基本流程

对于本文的增量式爬虫，基本的增量式爬虫足以满足需求，这是因为本文的爬取策略是在一个页面内获取多个商品的链接并进行分析是否已经记录过该网址，所

以实际网页请求也相对的少。对于本文的更新页面方式会使用固定频率以及批量式运行，如每星期进行已爬页面的更新[14]。

2.1.3 Scrapy 框架及其主要模块

Scrapy框架是一个基于异步I/O的Twisted框架，提供了高效、灵活、稳定的爬虫功能，在数据分析与处理领域得到广泛应用。它的高效性主要体现在采用了Twisted异步网络库，这使得它能够同时处理多个请求，从而大幅提高了爬取效率[15]。

Scrapy框架由5大组件构成。分别是调度器模块（Scheduler）、解析器模块（Spider）、实体管道模块（Item Pipeline）、Scrapy引擎模块（Scrapy Engine）以及下载器模块（Downloader）。其中Scrapy引擎是核心，控制了各个组件的工作流程及数据流，并通过爬虫中间件，完成了Scrapy引擎和对应组件的请求及响应[7]。

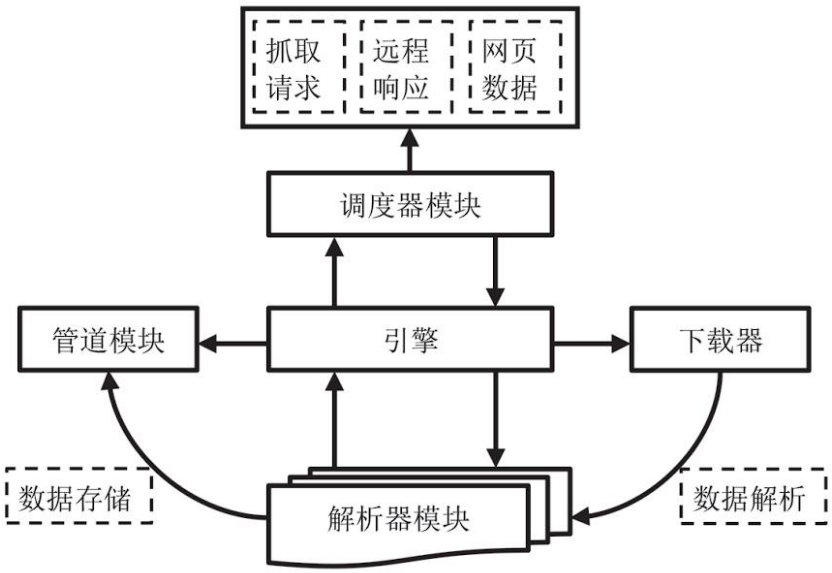


图 2-3 Scrapy 框架工作流程图[16]

1. Scrapy引擎模块：主要负责管理数据流的流动方向，并确保下一步的运行。它协调调度器、下载器、爬虫和实体管道之间的工作流程，使得整个爬取过程有序进行。

2. 调度器模块：负责接收并管理将要爬取的请求链接，并将这些链接保存在队列中，以便引擎在需要时获取并分发给下载器。调度器能够控制爬取的顺序和频率，确保爬虫不会对目标网站造成过大的负担。

3. 下载器模块：下载器负责从云端下载网页，并将抓取到的网页传送给引擎，作为爬虫的数据。它实现了对网络资源的请求和响应处理，包括处理HTTP请求、处理响应等功能。

4. 解析模块：解析模块是用户编写的解析返回信息的一套过程，用于从网页中提取所需的数据。它根据预先定义的规则，遍历目标网站的页面，并从中提取结构化数据，提取的方式有但不限于：Xpath，CssSelector，Regex或者是Json提取。爬虫可以根据需要进行定制和扩展，以满足特定的爬取需求。

5. 实体管道模块：实体管道负责对爬虫获取到的数据进行清理、验证和持久化到数据库等操作。在爬虫完成对信息的解析和提取后，实体管道可以对数据进行进一步处理，确保数据的完整性和可用性。

2.2 ElasticSearch

2.2.1 倒排索引概念

倒排索引（Inverted Index）作为搜索引擎的重要功能，通过将文档集合中的每个单词映射到包含该单词的文档列表，实现了快速的文本检索。倒排索引的创建到查询过程包括以下步骤：

1. 文档预处理： 首先对文档进行预处理，包括分词、去除停用词等操作，以便提取出文档中的有效信息。

2. 构建倒排索引表：对于每个单词,将其映射到包含该单词的文档列表,图 2-4显示了一个例子。对于单词 "iphone",将其映射到包含该单词的文档列表 [文档1, 文档2]；对于单词 "13"，将其映射到包含该单词的文档列表 [文档1]。

3. 查询处理： 当用户输入查询词时，系统对用户输入进行分词，得到查询词条。例如，对于查询 "iphone 13"，将其分词为词条 "iphone" 和 "13"，如图 2-5所示。

4. 匹配文档： 系统通过对比倒排索引表，找到包含查询词条的文档列表。对于 "iphone"，文档列表为 [文档1, 文档2]；对于 "13"，文档列表为 [文档1]。

5. 计算相关性： 最后，系统根据一些算法计算用户查询更可能属于哪个文档。

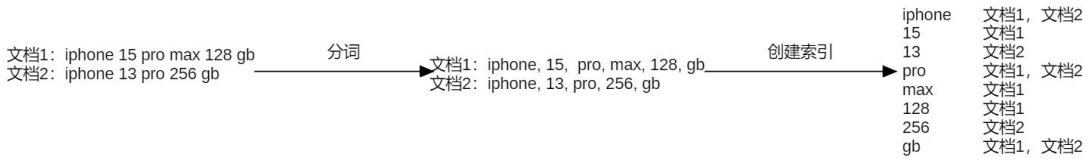


图 2-4 倒排索引表创建过程

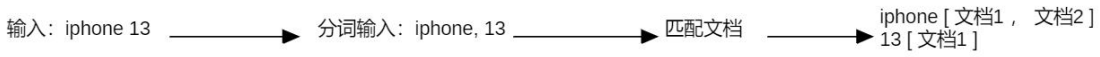


图 2-5 用户搜索

2.2.3 TF-IDF 算法

本小节首先通过介绍文本特征提取，再引入TF-IDF算法，并说明为什么TF-IDF是搜索引擎里非常重要的一个部分。

文本特征提取（Text Feature Extraction）可以视为减少词的空间上的维度（缩短词）和提取特征词，通俗来说就是在一段句子里提取能够体现出该句子的重要的词句。常见的文本特征提取方法有TF-IDF算法、信息增益、K最近邻算法等等[17]，这就是文本特征提取的主要功能。

TF-IDF全称（term frequency－inverse document frequency）是一种用于衡量一个词语在文档中的重要程度的统计方法，它结合了词频（TF）和逆文档频率（IDF）两个因素。

词频（TF）代表一个词语在文档中出现的频率，值得注意一点的是高频词并不一定代表重要性，例如：一段句子里可能包含多个“的”，但并不能体现出该句子的意思。因此需要用逆文档频率（IDF）来调整权重，逆文档频率代表了在整个文本集合中出现的文档频率，然后取其倒数并取对数，以降低高频词的权重，提高低频词的权重[18]。

TF-IDF的计算过程是将TF值与IDF值相乘，得到最终的TF-IDF值，来衡量该词语在文档中的重要性。

$$TF = \frac{term\_count}{document\_total\_word\_count} \tag{2.1}$$

$$IDF = Log(\frac{document\_count}{contain\_term\_document\_count}) \tag{2.2}$$

$$TF - IDF = TF \cdot IDF \tag{2.3}$$

表 2-1展示了利用文本1“iphone 15 pro max 128 gb”以及文本2“iphone 13 pro 256 gb”计算了它们的TF和IDF值，并计算了它们的TF-IDF值。根据计算结果，可以看出对于共同出现在两个文本中的词条，如"Iphone"和"pro"，它们的TF-IDF值为0，因为它们在所有文档中的出现频率都很高，不足以表示特定文档的重要性。而对于单独出现在某个文本中的词条，如"max"、"15"、"13"等，则计算出了非零的TF-IDF值，表示它们在对应文本中的重要性。

表 2-1 文本 1 及 2 的 TFIDF 值计算

词条	TF		IDF	TF • IDF	
	1	2		1	2
Iphone	1/6	1/5	0	0	0
15	1/6	0	0.3	0.05	0
pro	1/6	1/5	0	0	0
max	1/6	0	0.3	0.05	0
128	1/6	0	0.3	0.05	0
gb	1/6	1/5	0	0	0
256	0	1/5	0.3	0	0.06
13	0	1/5	0.3	0	0.06

通过TF-IDF算法，可以更好地衡量词语在文档中的重要性，从而实现更加准确的文本检索和信息检索。

2.2.4 ElasticSearch 功能介绍

1. 实时索引和更新[19]：Elasticsearch支持实时索引和更新操作，文档中指定为“text”类型的字段在变更时立即更新索引创建倒排索引，以支持快速的文本检索。除此之外ElasticSearch对构建倒排索引时进行了优化，包括对倒排索引进行压缩、分片存储等操作，以提高检索效率和减少存储空间占用。

2. 分布式存储与横向扩展： Elasticsearch 是一个分布式系统，可以水平扩展以处理大规模数据和高并发请求。它将数据分布到多个节点上，并自动处理数据的分片和复制，确保数据的高可用性和容错性。

3. 实时数据分析与聚合： Elasticsearch 支持实时数据分析和聚合操作，可以对大规模数据集进行复杂的数据分析和统计计算，如计数、求和、平均值、最大值、最小值等。

4. RESTful API： Elasticsearch 提供了基于 RESTful 风格的 HTTP API，使用户可以通过简单的 HTTP 请求进行索引、搜索、聚合等操作。这使得 Elasticsearch 可以轻松集成到各种应用程序和平台中。

5. 全文搜索：Elasticsearch 提供了强大的全文搜索功能，支持复杂的查询操作，包括布尔逻辑、模糊搜索、通配符搜索、范围查询等。借助倒排索引和词项权重等技术，可以实现高效的文本检索和相关性排序。其中Elasticsearch所使用的默认搜索算法为BM25算法，并且也提供了其他的算法如：DFR、DFI、IB等等的相似度算法。

2.4 布隆过滤器算法

增量式爬虫在爬取过程中可能会重复爬取同一网页，因此需要一个能够保存历史爬取记录的持久层。一种方法是利用哈希表，但这可能会占用大量空间。另一种方法是使用布隆过滤器，这是一种以空间换取准确率的数据结构[20]。

2.4.1 布隆过滤器算法介绍

布隆过滤器简单地说就是通过对元素应用多个哈希函数，将它们映射到一个位数组中，并将位数组中相应位置置为1来表示元素存在。在判断元素是否存在时，布隆过滤器会通过查询位数组来判断，如果所有对应的位都为1，则认为元素可能存在；如果存在任何一个位为0，则元素一定不存在。布隆过滤器可以通过调整位数组的大小和哈希函数的数量来控制空间占用和准确率，从而在一定程度上解决了哈希表占用空间过大的问题。图 2-6展示了输入“red”以及“blue”分别通过三次的哈希计算分别保存到位数组的相应位。

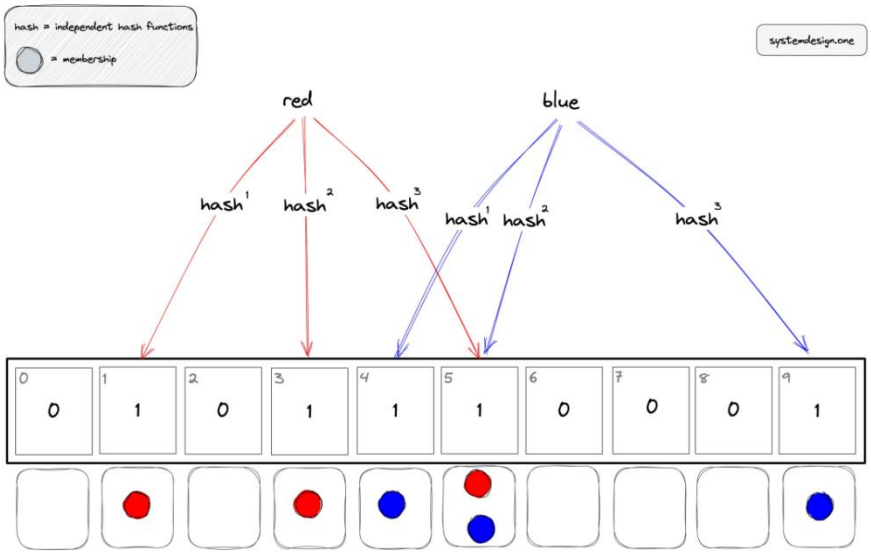


图 2-6 布隆过滤器的哈希映射到比特数组 S

布隆过滤器中的比特数组大小根据两个主要参数计算：预期容量（capacity）和期望的错误率（error_rate）。比特数组的大小（m）使用以下公式确定：

$$m = -\frac{n \times \ln(p)}{(\ln(2))^2} \tag{2.4}$$

其中：m是比特数组的大小（位数），n 是布隆过滤器的预期容量(要存储的元素数量），p 是期望的错误率（误判为正的概率）。

比特数组的大小决定了用于表示布隆过滤器中元素的位数。较大的比特数组可以容纳更多的元素并减少误判为正的概率，但需要更多的内存。相反，较小的比特数组节省内存但可能增加误判为正的可能性。

2.4.2 布隆过滤器的提升方案

但是显而易见的是普通的布隆过滤器只能添加数据并没法删除数据，若在本文的爬取过程中某个网址下架了，应该将他从位数组删除，此时可以使用计数布隆过滤器（Counting Bloom Filter），他的存储数组不再是位数组而是一个计数器，当元素被加入时，相关的位被增加；当元素需要被删除时，相关的位被减少。因此，计数布隆过滤器允许删除操作，但需要额外的空间来存储计数器。

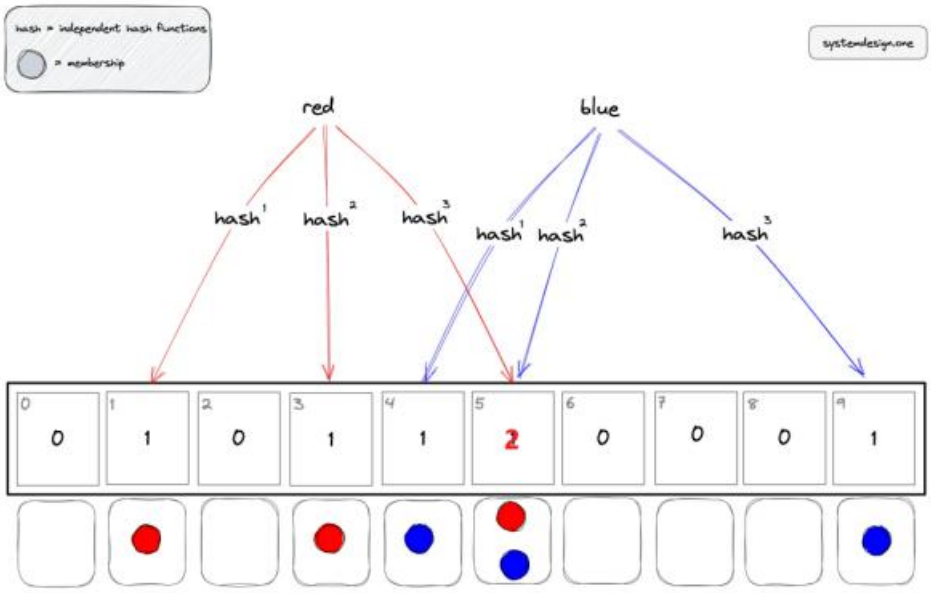


图 2-7 Counting Bloom Filter 映射结果

2.5 字符串匹配算法

2.5.1 字符串相似度匹配算法介绍

字符串相似度匹配算法是用于比较两个字符串之间的相似度的经典算法。文献[21]以及文献[6]进行关于比价系统的研究，他们都是在进行比价的时候进行字符串匹配，以便再匹配价格的时候是与相同商品进行匹配。

较为著名的字符串相似度算法有余弦相似度以及最短编译距离。本文将在这个小节内介绍这两种种算法并且分析为何使用他们作为本文的字符串相似度匹配算法来过滤商品的型号。

2.5.2 余弦相似度

余弦相似度利用两个向量点在一个向量空间里的夹角来衡量这两个向量的差异度。

$$cos\theta = \frac{a^2 + b^2 - c^2}{2ab} \tag{2.5}$$

公式2.5展示了余弦定理，余弦定理是在明确知道3个边的情况下得到其中一角的角度。而对于字符串的相似度计算，可以将字符串分解成字母向量进行计算，并让

这两个向量的起点从向量空间中的原点开始从而得到三个边的长度，即可计算出两个向量间的夹角，如图 2-8所示。

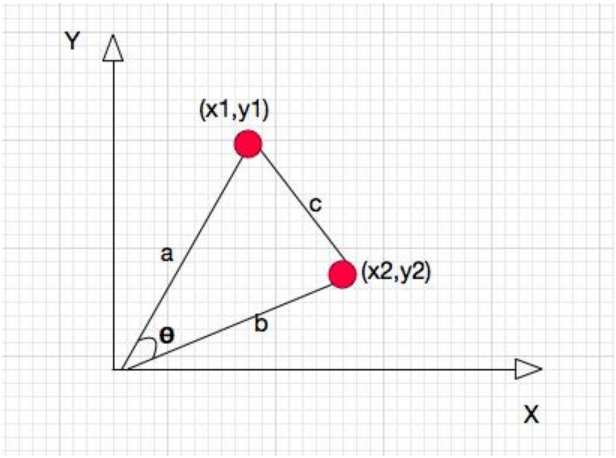


图 2-8 一个向量空间的三个点

但由于字符串在向量空间中会以多维形式展现如：

$$V_a = (v_{1,1}v_{1,2}...v_{1,n}) \tag{2.6}$$

所以将公式变形为公式 2.7，

$$cos\theta = \frac{< a, b >}{||a|| \star ||b||} \tag{2.7}$$

其中，

$$< a, b > \tag{2.8}$$

公式2.8为为向量的点积。

$$||a|| = \sqrt{a_1^2 + a_2^2 + ... + a_n^2} \tag{2.9}$$

公式2.9为向量范数。

表 2-2 TF+Cosine+词干提取错误情况

输入	预测 1	相似度 1	预测 2	相似度 2	预测结果	真实结果
11 iphone 12	Iphone 11	0.66	Iphone 12	0.66	Iphone 11	Iphone 12
Ipad 4 pencil 2	Ipad 4	0.5	Pencil 2	0.5	Pencil 2	Ipad 4

通过TF-IDF算法如公式2.3所示，计算出句子的向量值并使用公式2.7计算两个句子之间的相似度。通过余弦相似度算法可以了解到余弦相似度不注重文本的结构和语法例如表 2-2所示。余弦相似度更注重文本中词语的语义含义和分布。

2.5.3 最短编辑距离 LD 算法

最短编译距离最早由俄罗斯科学家Vladimir Levenshtein在1965年提出。其想法是在两个字符串$< w_1, w_2 >$之间，由其中一个字符串w_1变为另一个字符串w_2所需要最小的操作步骤[6]。

而具体的操作步骤分成3类，分别是：插入、删除以及替换。那么具体是如何在计算最短编译距离里的每一步中选择出最优解，本小节将进行讲解并说明该算法在进行型号匹配的好处。

首先举例两个字符串，分别为 $w_1 = abc, w_2 = aad$。此时，想要将w_1变更为w_2有如下的步骤。

1. 初始化变换矩阵表

表 2-3 最短编辑距离表 1

w1(i)/w2(j)	空	a	b	c
空	0	1	2	3
a	1	（1）	-	-
a	2	-	-	-
d	3	-	-	-

表 2-3显示了$w1$在某个状态转换成$w2$的某个状态所需要的步骤。例如：$w1$在空（empty）的时候，转变成$w2$的c状态需要3步骤，因为 $empty \rightarrow abc$ 需要进行3次的插入操作。由此获得上表。

2. 计算变更步骤

对于变换矩阵表的其他部分，则需要通过规律取得结果。从最简单的进行分析（1），

插入：在（1）的左方为$w_1a \rightarrow w_2empty$需要1步骤，而从$w_1a \rightarrow w_2empty$到$w_2a$需要插入一个$a$得到结果为2步骤。

删除：在（1）的上方为$w_2a \rightarrow w_1empty$需要1步骤，再将$w_2a \rightarrow w_1empty$便问$w_1a$需要1步骤，所以结果为2步骤。

更换：在（1）的左上方为$0 \to 0$需要0步骤，进行（1）的对应坐标有$a \to a$，为0个步骤。

在这三种操作内取最小的值表示字符串$w1$在某个状态转换到字符串$w2$的某个状态的最小步骤，获得表 2-4。

表 2-4 最短编辑距离表 2

w1(i)/w2(j)	空	a	b	c
空	0	1	2	3
a	1	0	-	-
a	2	-	-	-
d	3	-	-	-

3. 完善表格

最终得到从字符串$w1$转换成$w2$的最小步骤为2步，如表 2-5所示。

表 2-5 最短编辑距离表 3

w1(i)/w2(j)	空	a	b	c
空	0	1	2	3
a	1	0	1	2
a	2	1	1	2
d	3	2	2	2

最终获得以下表达式，

$$lev_{a,b}(i,j)=\begin{cases} max\{i,j\} & if, min\{i,j\}=0 \\ min\begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_i)} \end{cases} \end{cases} \tag{2.10}$$

经过了解最短编译距离（LD）后，可以发现它与字母的排序有极大的关系，所以比起余弦相似度在进行有要求先后次序的型号匹配更加的合适。但最短编辑距离（LD）也有其不足之地，表 2-6展示了使用LD可能会出错的情况，而这些错误都是能够由余弦相似度进行解决的。本文将提出通过TF-Cosine+LD混合算法来提高型号分类的准确率。

表 2-6 LD+词干提取错误情况

输入	预测 1	距离 1	预测 2	距离 2	预测结果	真实结果
14 pro	Ipad 4	6	iphone 14 pro	7	ipad 4	Iphone 14 pro
Air 3	Watch 3	4	Ipad air 3	5	Watch 3	Ipad air 3
iphone 15 8 13	Iphone 15 pro	4	Iphone 15	5	Iphone 15 pro	Iphone 15

2.6 本章小结

本章主要介绍了系统所使用的技术，帮助理解其实现原理。介绍的技术包括数据获取模块所使用的Scrapy框架和网络爬虫，以便熟悉数据获取方式。随后介绍了搜索引擎和Elasticsearch框架，并解释了索引原理和Elasticsearch的倒排索引。最后，介绍了两种算法——布隆过滤器和字符串匹配算法，以用于商品过滤以及分类。

第 3 章 基于 Android 的二手商品实时比价系统需求分析

本章对安卓二手商品实时比价系统进行需求分析，在软件工程的生命周期中，需求分析至关重要，它需要确定并分析用户的需求，并决定该软件系统需要完成什么功能。所以首先将阐明系统的设计目标，随后分析系统的功能以及非功能需求。

3.1 系统目标

本系统的主要目标是实现一个能够对比多个二手网站平台上的商品的比价系统，系统会提供用户搜索产品的功能、搜索成功后也会显示该商品在不同平台上的历史价格、若没有用户想要查询的商品选项，用户可以提交申请商品供管理员进行筛选并纳入系统、除此之外，用户也可以通过比搜索更多选项进行商品的订阅，订阅系统会在用户订阅参数范围内爬取到更好的商品时提醒用户，从而使用户花费更少的精力就可以找到符合用户价格的商品，并第一时间获取最新数据。

对于数据获取来源，系统将会定期以两种方式进行爬取数据以保证数据的实时性，分别是基于种类品牌和基于种类品牌型号。

1. 基于种类品牌：例如（Mobile - Apple）。这种方式获取商品的广度较大，适合用于增加系统商品的覆盖范围。

2. 基于种类品牌型号：例如（Mobile - Apple - iPhone 15）。这种方式获取商品的深度较大，适合用于深度搜索某样商品。

这两种方式都有其用武之地，基于种类品牌是让系统有一个足够广泛的商品列表，而基于类型品牌型号是为了能够返回更具细节的商品给用户订阅的物品。

3.2 功能性需求分析

本小节根据上一节提到的系统涉及目标，主要将使用者分成了三种角色，如图3-1所示：

1. 用户：登录、注册、忘记密码、登出、更改账户信息、查询物品、订阅/取消订阅商品、收藏/取消收藏商品、查询历史记录

2. 管理员：管理Timmy商品、角色管理、索引管理、管理后台进程、爬取商品

3. 信号：爬取商品、更新数据、爬取订阅物品、爬取种类物品、索引最低价格商品

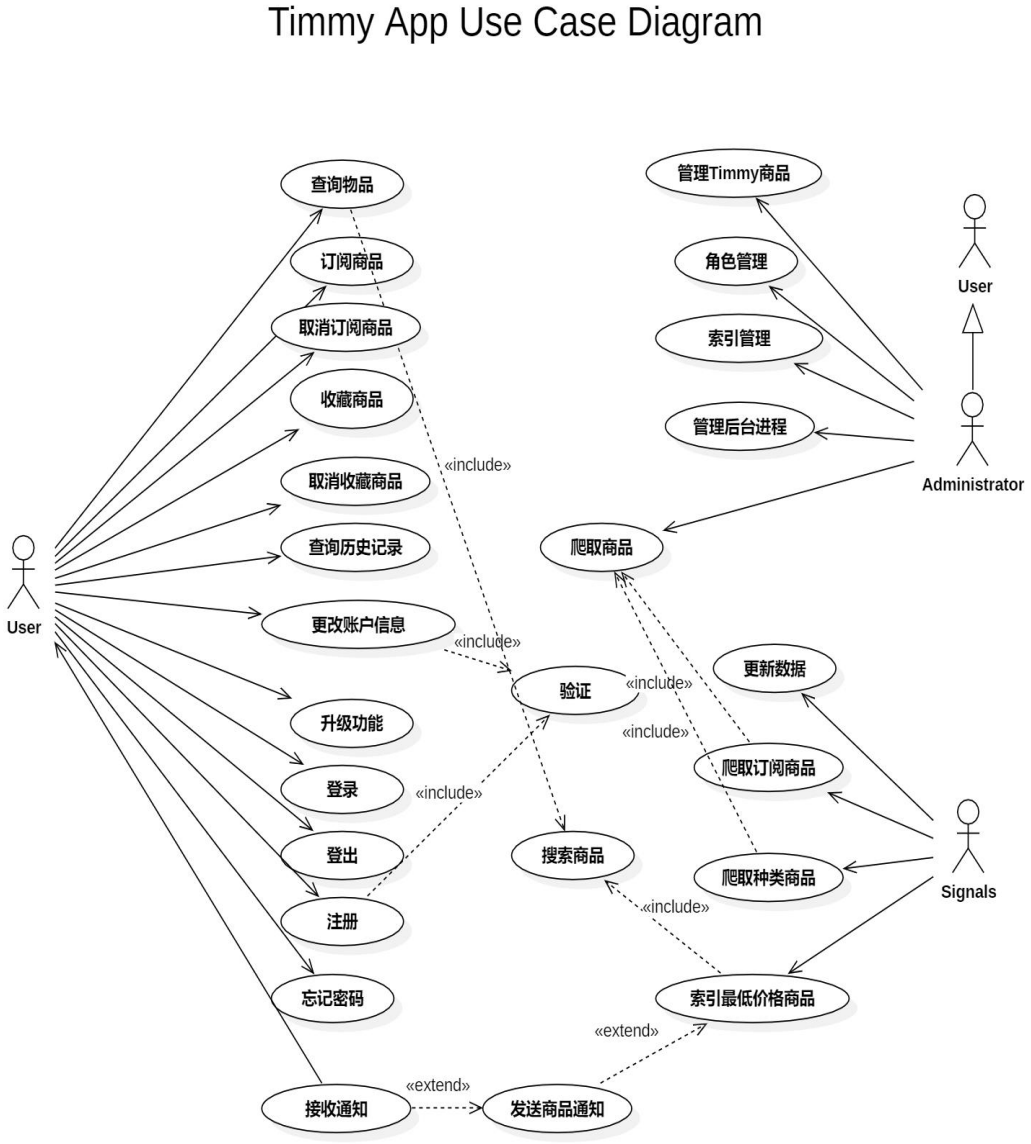


图 3-1 系统用例图

另外，管理员功能包括管理Timmy商品、角色管理、索引管理以及后台进程管理。这些功能可以进一步细分，如图 3-2所示。需要说明的是，Timmy商品实际上是该系统中用于保存商品型号集合的类，例如 Mobile Apple iPhone 15 Pro Max、Tablet Apple iPad Air 5等产品。使用“Timmy”作为命名是因为该系统的名称为提米商城（Timmy App），以便让用户对该应用程序有更深刻的印象。本文后续将频繁使用该术语。

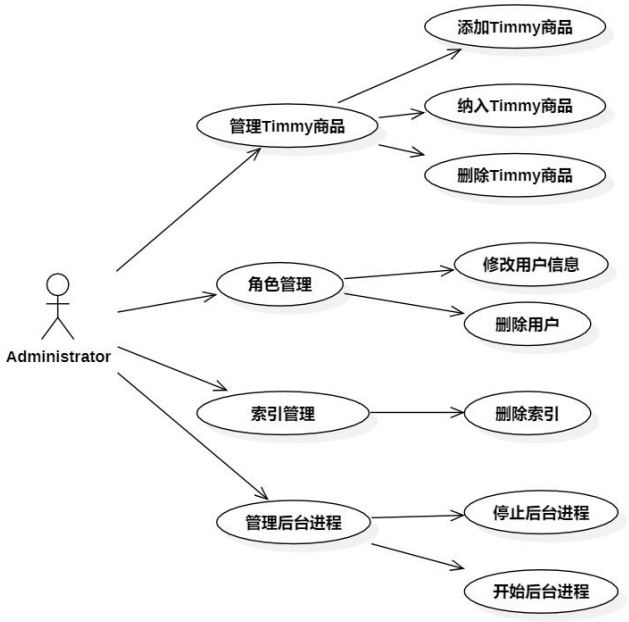


图 3-2 管理员用例图

3.3 非功能性需求分析

在设计和开发这个系统的过程中，除了实现上节提出的功能需求外，还应该考虑到系统的非功能需求。接下的小节中将分析系统的非功能性需求，其中包括：性能、易用性、安全性等。

3.3.1 性能需求

性能包括了系统的平均响应时间、吞吐量以及大量用户同时访问时情况。为了保证该系统能够顺利的运行。在设计该系统的时候应该多使用异步操作对数据库、网络请求等行为进行运行，以避免系统阻塞。除此之外，为了让用户快速获得信息，系统将会事先保证有一定的数据量，并在第一时间返回给用户，后续在进行数据的更新。

3.3.2 易用性需求

对于该系统，在系统界面设计中，为了使用户能够再操作过程中顺畅无阻碍，应该使用清楚的字体大小以及符合人眼视觉的配色。除此之外，应该直观的让使用

者清楚明了可进行点击的部分。尽量简化功能操作的复杂度、减少不必要的操作步骤。

3.3.3 安全性需求

对于系统的安全性，用户的密码将会进行加密传输以及保存，以确保用户的安全性。另外服务器采用安全的HTTPS协议进行通信。

3.4 系统静态模型

依据用例模型导出静态模型，静态模型主要将问题域里的实体转变为信息与的类与对象以及他们之间的关系，分别从问题域里分成实体、概念以及事件如图 3-3 所示。

实体包括：Timmy商品、Elastic商品

概念：用户商品收藏、用户商品订阅、用户订阅商品实例、用户查找记录、商品订阅、日常搜索、历史价格、爬虫、验证码、通知、Elastic实例

参与者：用户、管理员、信号发射器

其中Elastic商品为爬虫子系统通过爬取二手平台商品进行一系列清理后所保存到ElasticSearch数据库内的数据类名。本文后续将频繁使用该术语。

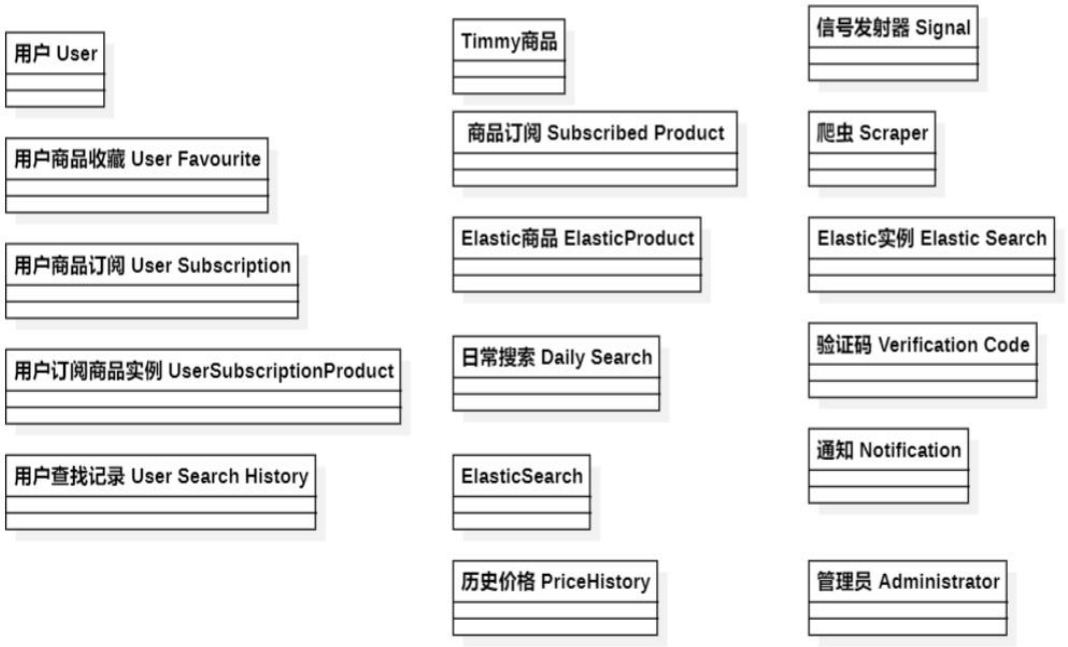


图 3-3 静态模型图

表 3-1至表 3-6列出了部分静态模型的属性以及服务。

表 3-1 用户商品订阅静态模型

类	属性	服务
用户商品订阅	用户商品订阅标识符、商品全名、商品种类、商品品牌、商品型号、商品简介、商品最高价格、商品最低价格、商品国家、商品省份、商品品质、订阅提醒方式、订阅提醒时间、订阅时间、订阅价格、订阅状态、订阅爬虫	添加用户商品订阅、删除用户商品订阅、获取用户商品订阅、通过订阅标识符获取商品订阅、通过等级获取商品订阅、通过提醒时间获取商品订阅

表 3-2 用户订阅商品实例静态模型

类	属性	服务
用户订阅商品实例	用户订阅商品实例标识符、用户商品订阅标识符、商品汇率、商品添加时间、商品链接、商品图片链接、商品标识符、商品标题、商品简介、商品品质、商品爬虫、商品价格、商品人民币价格	添加用户订阅商品、删除用户订阅商品

表 3-3 商品订阅静态模型

类	属性	服务
商品订阅	订阅商品全名、订阅商品种类、订阅商品品牌、订阅商品型号、订阅商品数量、订阅商品最高等级	添加订阅商品、删除订阅商品、获取所有订阅商品

表 3-4 爬虫静态模型

类	属性	服务
爬虫	爬虫标识符、爬取日期、爬取数量、爬取种类、爬取品牌、爬取型号、测试、爬取次数	爬取商品、爬取订阅商品、爬取种类品牌商品

表 3-5 Timmy 商品静态模型

类	属性	服务
Timmy 商品	商品全名、商品种类、商品品牌、商品型号、商品录入	添加商品、删除商品、获取商品种类、获取商品品牌、获取商品型号、获取商品字典

表 3-6 Elastic 商品静态模型

类	属性	服务
Elastic 商品	商品标题、商品价格、商品价格（人民币）、商品品质、商品详情、商品链接、商品图片链接、商品创建日期、商品爬取日期、商品国家、商品省份、商品汇率、商品唯一标识符、商品种类、商品品牌、商品型号、商品来源、测试商品、	添加商品、删除商品、获取商品、更新商品

3.5 本章小结

本章主要从用户的角度进行系统需求分析。在此阶段，确定了系统必须提供的功能。通过分析用户需求，提出了以下主要功能：用户搜索商品、订阅商品、添加商品、提醒用户、系统爬取订阅商品、以及爬取各种品牌商品。随后，提出了非功能需求，包括系统的性能、易用性和安全性要求。最后，建立了初步的静态模型，确定了系统所需类的属性和服务，确保后续系统设计和开发过程的顺利进行。

第 4 章 基于 Android 的二手商品实时比价系统架构设计

4.1 系统总体架构设计

二手商品实时比价系统采用前后端分离的开发模式。前端基于安卓平台以及网页版，使用React Native以及Vue进行开发。后端则采用EF CORE、WebAPI、Elastic Search和Scrapy进行设计，并使用Microsoft SQL Server和Elastic Search作为数据库来进行数据持久化操作。客户端采用视图层以及请求层，后端架构采用三层架构模式，包括控制层、业务层和数据层，这样的设计能够有效地提高开发效率。

客户端端通过HTTPS协议与服务器进行交互，采用HTTPS能够保证数据传输的安全性。服务器的控制层接收到请求后，通过业务层、模型层以及数据库最后再将数据以JSON数据格式发送给客户端。客户端最终将得到的响应数据进行处理后展示出来，系统的总体架构如图 4-1所示：

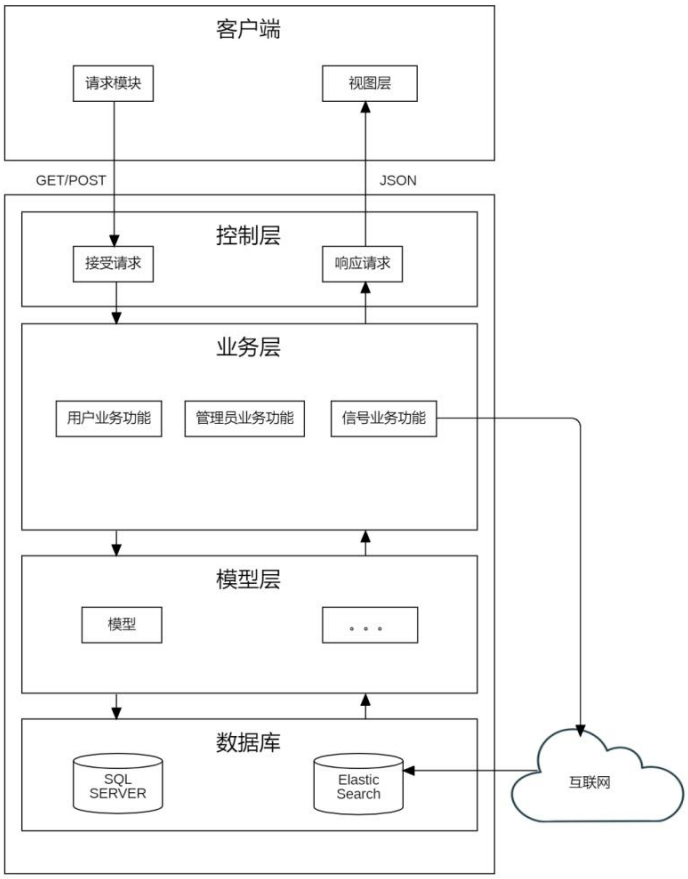


图 4-1 系统总体框架

其中位于业务层的信号业务功能是本系统的核心之一，该业务会通过信号在设定好的时间进行数据的爬取所以会与互联网连接，最后再将爬取到的数据存储到ElasticSearch数据库里进行持久化。

4.2 客户端架构设计以及功能

4.2.1 移动端架构设计

本系统的移动端开发架构使用了能够跨平台的框架React Native进行开发，使用React Native作为本文的移动端首选是因为它提供了经济高效的方式构建和维护跨平台app，并且可以减少开发时间，它可以同时开发安卓以及苹果应用程序。除此之外React Native也提供了自家开发的UI以同时兼容安卓以及苹果原生UI。

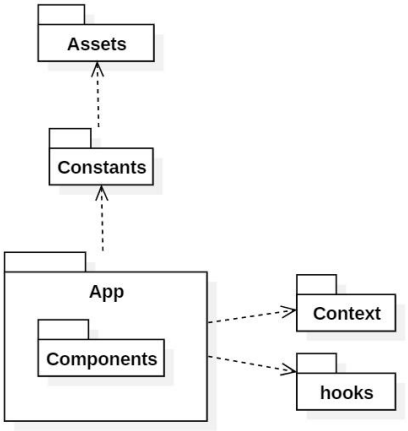


图 4-2 移动端设计架构图

本文移动端设计架构如图 4-2所示，可以将其归类为视图层、请求模块以及其他模块：

视图层：

1. App：该文件包含了实现页面展示的主要模块，包含home、about页面等等。
2. Assets：该文件包含了应用程序所使用的字体、图片以及图标。

请求模块：

1. Hooks：该文件包含了用户自定义的Hook函数，包含了请求模块在内。

其他模块：

1. Context：该文件包含了全局状态文件

2. Components：该文件包含了App以及Components本身能够使用的小组件，以加快开发速度，包含button、header等等。

3. Constants：该文件为导出静态文件代码，可在App以及Components内调用该文件来加载图片或图标。

4.2.2 移动端设备功能

移动端设备包含用户的功能，图 4-3展现了用户在移动端可以使用的功能。

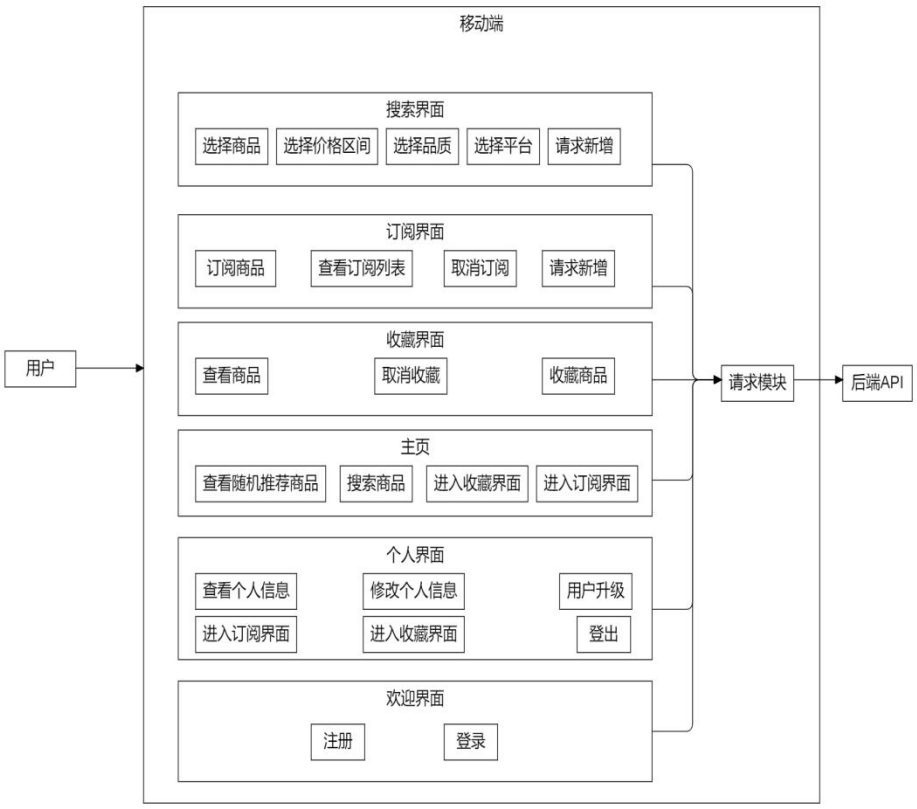


图 4-3 移动端功能图

1. 搜索界面：用户可以在搜索界面进行特定的商品选择，并且选择相应的参数例如：价格、品质以及平台，若特定商品没有包含用户想要的商品，用户可以利用请求新增功能对特定产品进行请求以便管理员将其纳入系统。

2. 订阅界面：用户可在订阅界面进行商品的订阅。除此之外，若用户已有订阅的商品，将会以列表形式展现给用户，用户可以进行取消订阅操作。

3. 收藏界面：用户的收藏商品将会以列表形式展示在收藏界面，用户可对收藏商品进行取消收藏或者收藏操作。

4. 主页：用户可在主页观看选定的种类商品，并且主页面提供了搜索、收藏以及订阅的链接。

5. 个人界面：用户可在个人界面进行个人信息的查询、修改以及密码修改。

6. 欢迎界面：用户可以在欢迎界面进行注册或者登录。

4.3 管理端架构设计以及功能

4.3.1 管理端架构设计

本系统的系统管理端使用了Vue框架进行开发，使用Vue作为本文的系统管理员端首选是因为它提供了许多功能，包括：双向数据绑定、组件化开发等等，所以可以有效的减少开发时间。

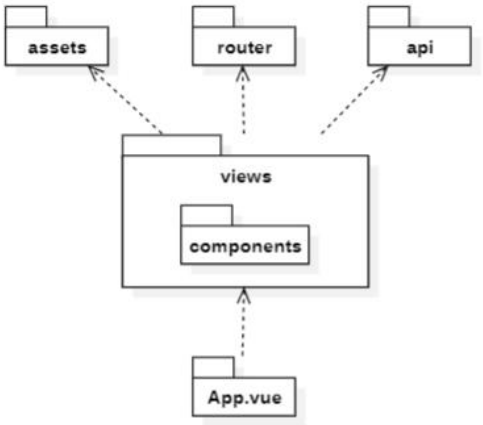


图 4-4 管理员端设计架构图

本文管理端设计架构如图 4-4所示，也可以将其归类为视图层、请求模块以及其他模块：

视图层：

1. App.vue：该文件包含了程序入口点。

2. Views：包含了管理端各个界面。例如：用户管理、商品管理界面等等。

3. Components：包含了组件，提高了开发的速度。例如：特殊按钮、主页面头部等组件。

4. Assets：包含了应用程序所使用的字体、图片以及图标。

请求模块：

1. Api：包含了请求函数，用以在Views以及Components进行请求操作。

其他模块：

1. Router：包含了各个Views之间的路由切换函数。

4.3.2 管理端功能

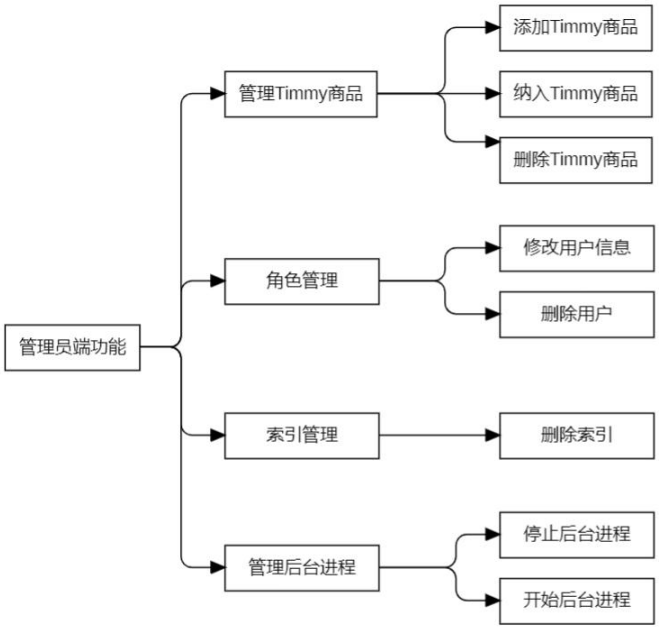


图 4-5 管理员端功能

管理端的功能主要处理一下关键业务如图 4-5所示：

1. Timmy商品管理 - 管理员可对系统Timmy商品进行添加、纳入以及删除操作，以方便管理员在后续新增商品进行爬取。

2. 角色管理 - 管理员可对存在于系统的用户进行删除、修改操作，确保系统的准确性及安全性。

3. 索引管理 - 管理员可以查看ElasticSearch内商品的统计信息，以便进行分析和决策。

4. 管理后台进程 - 管理员可以手动启动或停止后台进程，确保系统后台任务的正常运行和维护。

4.3 服务端架构设计以及功能

本系统的服务器使用了C# 12.0作为开发语言，并且使用.NET Framework所提供的ASP.NET core、EF core以及webapi框架进行开发，本文所使用的编译器为Visual Studio Code 2022，并且在C#中利用Pythonnet库调用Scrapy脚本执行爬虫子系统进行商品的爬取，爬虫子系统使用的语言为Python并且编译器为VS Code，最后使用了SQL SERVER以及Elastic Search作为数据库进行数据持久化。

4.3.1 服务端架构实现与展示

为了使得开发过程更加的顺畅以避免“重复造轮子”，本系使用了大量包，所以需要安装特定软件包，如表 4-1所示。

表 4-1 服务器安装包

包	版本	用途
Pythonnet	3.0.3	在C#内运行python脚本
Elasticsearch.Net	7.17.5	ElasticSearch API
NEST	7.17.5	ElasticSearch API
Nito.Guids	1.1.1	生成随机号码
Microsoft.EntityFrameworkCore.Design	8.0.3	
Microsoft.EntityFrameworkCore.SqlServer	8.0.3	链接SQL SERVER
Microsoft.AspNetCore.OpenApi	8.0.3	
System.IdentityModel.Tokens.Jwt	7.4.1	JWT令牌包
Hangfire.Core	1.8.11	后台进程
Hangfire	1.8.11	后台进程
Hangfire.SqlServer	1.8.11	让Hangfire链接SQL SERVER
Hangfire.Dashboard.Basic.Authentication	7.0.1	
Microsoft.Data.SqlClient	5.2.0	

完成上述重要包的下载后，接下来对服务端以WebAPI的形式进行开发。WebAPI与传统的MVC架构有一些区别，其中主要在于WebAPI旨在满足REST数据接口的需

求，负责通过HTTP协议向各个渠道提供数据，更注重接口API的开发。由于本系统计划开发移动端和网页端，因此WebAPI的开发形式更加适合。

在WebAPI的开发形式中，与MVC相似，但少了视图层（View），包含的主要模块有控制器（Controller）、DAO（数据访问层）、模型（Models）、业务层（Service）。此外，还包括Product（产品）等其他功能。图 4-6展现了本系统利用WebAPI所构建出架构。

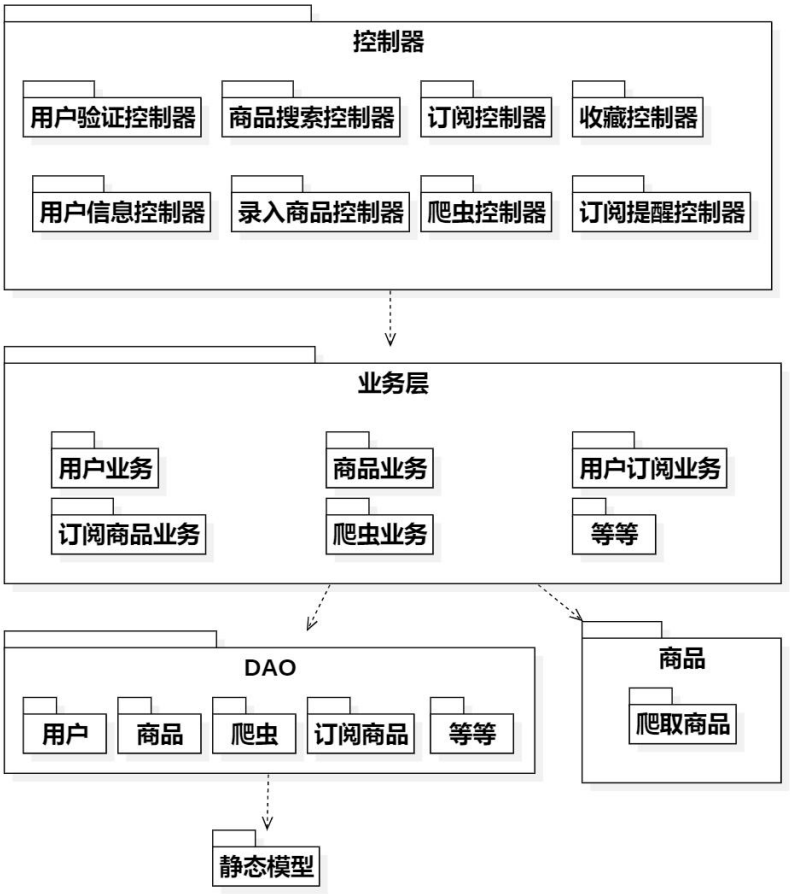


图 4-6 服务端设计架构图

4.3.2 服务端功能

服务端功能包括了如图 4-7所示的8个模块：

1. 用户验证模块：服务端提供了对用户的登录、登出、注册、更改信息等需要进行验证的功能，该模块会利用JWT令牌来对用户进行验证。

2. 商品搜索模块：服务端接收商品搜索参数如：商品种类、品牌、型号等等，并通过接口查找ElasticSearch数据库上的商品并返回给用户，同时保存搜索记录。

3. 订阅模块：服务端接收商品订阅参数如：商品种类、品牌、订阅提醒时间等等，并保存到数据库以便系统下次进行爬取时，系统会进行基于订阅商品爬取（深度爬取），便在用户设定的订阅提醒时间提醒用户。

4. 收藏模块：服务端提供了商品的收藏以及取消收藏功能，并将内容保存到数据库。

5. 用户信息模块：服务端提供用户个人信息、查找历史信息功能。

6. 录入商品模块：服务端提供申请录入商品、录入商品、删除商品功能，供用户以及管理员使用。

7. 爬虫模块：服务端提供了爬虫、种类品牌爬虫以及订阅商品爬虫功能，供管理员使用。

8. 订阅提醒模块：服务器将会定期调用该模块，进行商品的搜索更新用户订阅商品并发送提醒。

9. 商品比价模块：用户订阅商品时所调用的比价算法即在该模块内，系统会通过一系列操作获取最佳的商品选项。

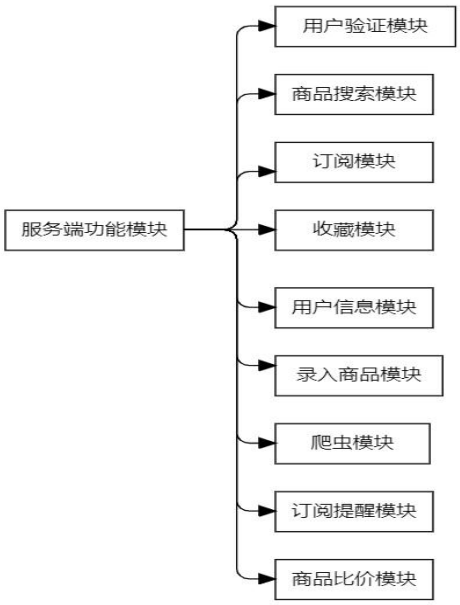


图 4-7 服务端功能图

4.3.3 Scrapy 架构实现

对于本文的数据爬取模块，利用了Scrapy框架，它是一个基于异步I/O的Twisted框架实现的。第二章内提到了Scrapy的几个主要模块，在这里实现的时候主要关注的是爬虫模块以及实体管道，他们所对应的文件分别是如图 4-8以及图 4-9所示的pipelines.py以及spiders内的文件。随后，为了方便调用该爬虫程序，建立了一个入口文件“crawlerProcess.py”进行数据的爬取，并返回状态给用户。

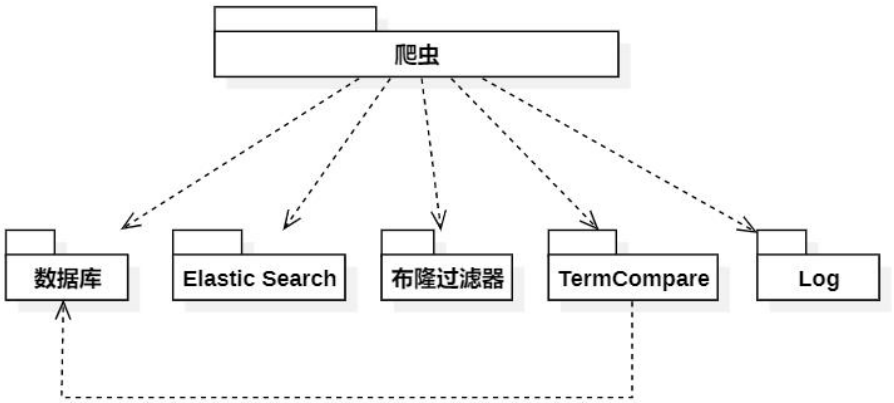


图 4-8 Scrapy 设计架构图

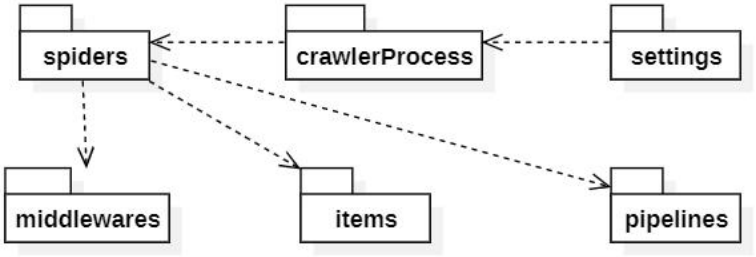


图 4-9 Scrapy/Product 设计架构图

除了Scrapy所提供的主要模块以外，在本文内还对Scrapy进行了一些新增功能，其文件如图 4-8所示，分别有：

1. 布隆过滤器：负责提供布隆过滤器的功能，提供给pipelines.py文件进行数据的清理。

2. 数据库：负责提供数据库的链接、读写操作，获取系统的商品列表。

3. ElasticSearch：负责提供ElasticSearch的链接、读写操作，主要对爬取后的商品进行持久化操作。

4. Log：保存每次进行爬取后的日志，格式以爬取时间日期为标题，例如：Crawler_Process_2024-04-26T141113.329615_logs。

5. TermCompare：该模块提供了商品型号的匹配算法，由pipelines.py文件在对商品标题进行分配时使用，4.3.7节TermCompare将介绍其接口。

本文的Scrapy框架整体流程如图 4-10所示，用户只需要调整Config.ini文件内的Database以及ElasticSearch的配置即可成功链接本地数据库以及ElasticSearch。

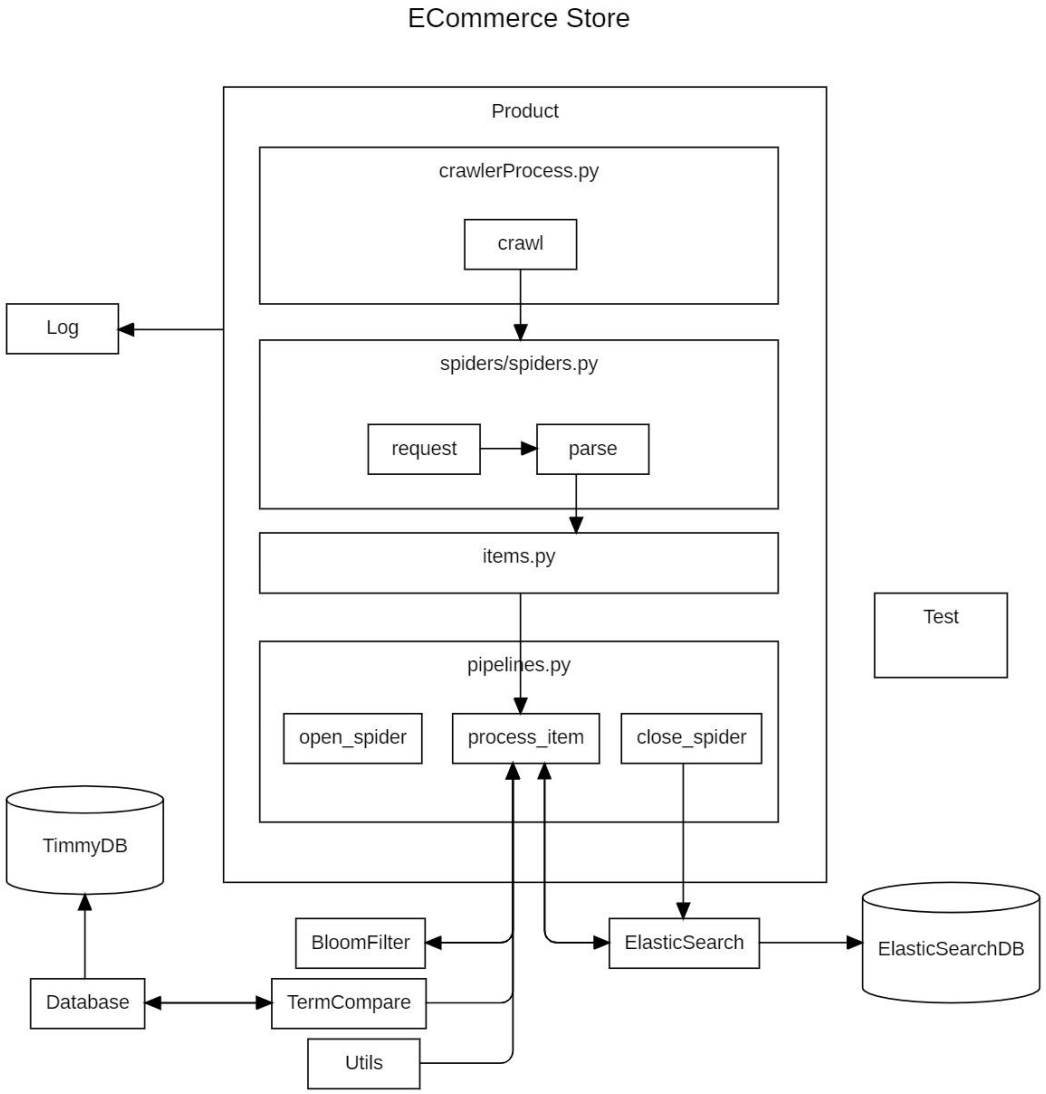


图 4-10 Scrapy 框架流程图

首先，用户可以使用由crawlerProcess.py文件所提供的api进行数据的爬取，例如调用：python .\Product\crawlerProcess.py -c mobile -b apple -m iphone 15 pro max -s mudah aihuishou -t 1 -i 10，即可爬取商品种类为mobile、商品品牌为apple、商品型号为iphone 15 pro max、平台为mudah以及aihuishou、遍历次数为10的爬虫任务。数据爬取过程包含了请求url链接（request）、分析（parse）、处理商品（process_item）最后关闭爬虫（close_spider）。其中，最为关键的一步在处理商品（process_item），因为在这一阶段会通过调用TermCompare模块所提供的功能对商品进行清理、归一化以及分类型号操作。当所有爬虫任务都结束后，将会利用ElasticSearch模块上传数据到ElasticSearch数据库进行持久化。

4.3.4 ElasticSearch 架构实现

本文利用Docker安装ElasticSearch以快速调用其框架。为了日后可以对中文进行准确分词，本文优先安装了ik分词器，ik分词器是一个专门对中文词语进行分词的库。下载后安装路劲如下：/usr/share/elasticsearch/plugins/。

图 4-11 安装 ik 分词器路径

接下来便可启动 docker 容器使用 ElasticSearch。

图 4-12 启动 docker 容器

在ElasticSearch里，索引（index）是关系型数据库里面的表，所以通过在本地ElasticSearch里创建名为product的索引，该索引的数据类型映射如表 4-2所示。在表4-2内可以看到一些字段设置为Text类型，在ElasticSearch里被设置为Text的字段会在搜索的时候进行分词并评分，所以在这里将有可能在未来用以作为搜索关键字的字段设置位Text类型如：Title、Description，而那些与搜索字段没太大关键的字段则设置成Keyword，对于日期类型使用了date类型。

表 4-2 ElasticSearch product 索引

字段	类型
Brand	Keyword
Category	Keyword
Condition	Keyword
Country	Keyword
Created_date	Date
Currency	Keyword
Description	Keyword
Is_test	Keyword
Model	Keyword
Price	Float
Price_CNY	Float
Product_detail_url	Text
Product_image	Text
Product_url	Text
Scraped_date	Date
Server	Text
Spider	Text
State	Keyword
Title	Text
Unique_id	Keyword

4.4 接口设计

接口（API）是指不同软件系统之间进行交互的方式和约定。在完整的软件开发过程中，重要功能的接口设计不可或缺。优良的接口设计能够让开发者和使用者清晰了解某个功能的使用过程并获得理想结果，同时也便于开发者进行测试。本小节将介绍服务器、爬虫子系统以及商品分类算法的接口。

4.4.1 服务器接口设计

服务器作为该系统的核心部分，必须提供良好的接口供客户端使用，并为后续开发增加便利性。本节将详细介绍该系统的重要接口如搜索商品、订阅商品、上爬取等等功能。其中详细讲解每个接口进行输入、输出以及处理过程。

1. 搜索商品

当用户正常输入搜索参数并点击搜索商品按钮的时候将会调用以下接口，并按照下述的请求参数列表发送用户Jwt令牌、页面信息以及商品信息等到服务端，此时服务端将调用其内部的ElasticSearch业务进行商品的搜索，并将成功搜索到的信息携带状态码200返回给客户。若此时业务任何一个部分失败，则会返回状态码为400代表业务处理错误，并在客户端提醒用户错误的原因。搜索商品接口设计表 4-3所示：

表 4-3 搜索商品接口表

API 地址	/api/ElasticSearch/SearchProduct			
请求方式	POST			
应用场景	用户进行搜索商品，返回商品结果在用户界面端显示			
请求参数	变量名	类型	示例值	描述
	JwtToken	String	jwt 令牌	用户 jwt 令牌
	PageSize	int	10	返回数据一页的大小
	CurrentPage	int	1	当前页面
	Category	string	Mobile	商品类型
	Brand	string	Apple	商品品牌
	Model	string	Iphone 15 pro	商品型号
	Description	string	128	商品详情
	Highest_price	float	10000	设置最高价格
	Lowest_price	float	8000	设置最低价格
	Country	string	China	设置国家
	State	string	China	设置城市
	Condition	string	New/Mint/Used	设置品质
	Spider	String[]	[“aihuishou”]	设置来源
	Sort	string	priceasc	设置排序方式
	isTest	int	0	是否为测试
返回结果	statusCode	Int	200	返回状态值
	Message	String	Success	返回文本信息
	Data	PageEntity<ElasticProductDTO>>	无	返回获取的数据集合

2. 订阅商品

当用户正常输入订阅商品的参数后点击订阅按钮的时候将会调用以下接口，并按照下述的请求参数列表发送用户Jwt令牌、订阅信息到服务端，此时服务端将调用其内部的UserSubscription业务进行商品的订阅，并将订阅结果的信息携带状态码200返回给客户。若此时业务任何一个部分失败，则会返回状态码为400代表业务处理错误，并在客户端提醒用户错误的原因。用户订阅商品接口设计表 4-4所示：

表 4-4 用户订阅商品接口表

API 地址	/api/UserSubscription/AddUserSubscription			
请求方式	POST			
应用场景	用户进行商品订阅，返回订阅商品结果在用户界面端			
请求参数	变量名	类型	示例值	描述
	JwtToken	String	jwt 令牌	用户 jwt 令牌
	Subscription_notification_method	String	Email	订阅提醒方式
	Subscription_notification_time	int	1	订阅发送通知时间
	Category	string	Mobile	商品类型
	Brand	string	Apple	商品品牌
	Model	string	Iphone 15 pro	商品型号
	Description	string	128	商品详情
	Highest_price	float	10000	设置最高价格
	Lowest_price	float	8000	设置最低价格
	Country	string	China	设置国家
	State	string	China	设置城市
	Condition	string	New/Mint/Used	设置品质
	Spider	String[]	[“aihuishou”]	设置来源
	Description	string	128 gb	订阅的详情
返回结果	statusCode	Int	200	返回状态值
	Message	String	Success	返回文本信息
	Data	<UserSubscription>	无	返回获取的数据集合

3. 商品爬取

当管理员输入爬取参数并点击爬取按钮的时候将会调用以下接口，并按照下述的请求参数列表发送爬取信息到服务端，此时服务端将调用其内部的ScrapyService业务进行商品的爬取，并将爬取调用信息携带状态码200返回给客户。若此时业务任何一个部分失败，则会返回状态码为400代表业务处理错误，并在客户端提醒管理员错误的原因。爬取商品接口设计表 4-5所示：

表 4-5 爬取商品接口表

API 地址	/api/Scraper/ScrapeProduct			
请求方式	POST			
应用场景	管理员手动爬取商品			
请求参数	变量名	类型	示例值	描述
	Category	string	Mobile	商品类型
	Brand	string	Apple	商品品牌
	Model	string	Iphone 15 pro	商品型号
	Spider	String[]	[“aihuishou”]	设置来源
	isTest	Int	1	是否为测试爬取
	Iteration	Int	10	爬取遍历次数
返回结果	statusCode	Int	200	返回状态值
	Message	String	Success	返回文本信息
	Data	Bool	True	成功调用爬虫进程

4. 收藏商品/取消收藏

当用户在界面点已/未收藏的商品的收藏按钮时候将会调用以下接口，并按照下述的请求参数列表发送用户Jwt令牌、商品唯一标识符到服务端，此时服务端将调用其内部的UserFavourite业务进行商品的收藏，并将收藏的信息携带状态码200返回给客户。若此时业务任何一个部分失败，则会返回状态码为400代表业务处理错误，并在客户端提醒用户错误的原因。用户收藏商品接口设计表 4-6所示：

表 4-6 用户收藏商品接口表

API 地址	/api/UserFavourite/[FavouriteProduct,UnFavouriteProduct]			
请求方式	POST			
应用场景	用户进行收藏商品，返回收藏商品结果在用户界面端			
请求参数	变量名	类型	示例值	描述
	JwtToken	String	jwt 令牌	用户 jwt 令牌
	productUniqueId	String	ahs2024042115 3801121063	商品唯一标识符
返回结果	statusCode	Int	200	返回状态值
	Message	String	Success	返回文本信息
	Data	Bool	True	收藏商品状态

5. 录入商品

当用户或管理员输入录入商品参数并点击录入按钮的时候将会调用以下接口，并按照下述的请求参数列表发送商品种类、品牌、型号以及录入到服务端，其中Adopt对于用户使用只能是0，表示必须等待管理员确认。此时服务端将调用其内部的商品业务进行商品的录入，并将结果携带状态码200返回给客户。若此时业务任何一个部分失败，则会返回状态码为400代表业务处理错误，并在客户端提醒用户错误的原因。录入商品接口设计表 4-7所示：

表 4-7 录入商品接口表

API 地址	/api/TimmyProduct/AddTimmyProduct			
请求方式	POST			
应用场景	录入商品分为用户及管理员，用户的录入必须等待管理员接收			
请求参数	变量名	类型	示例值	描述
	Category	String	Mobile	录入商品种类
	Brand	String	Huawei	录入商品品牌
	Model	String	Pura 70	录入商品型号
	Adopt	String	0	用户:0（未取人） 管理员：1（确认）
返回结果	statusCode	Int	200	返回状态值
	Message	String	Success	返回文本信息
	Data	TimmyProduct	无	录入商品信息

6. 用户登录

当用户在登陆界面输入参数并点击登录按钮的时候将会调用以下接口，并按照上述的请求参数列表发送用户标识以及用户密码到服务端，此时服务端将调用其内部的User业务进行用户的验证，并将结果携带状态码200返回给客户。若此时业务任何一个部分失败，则会返回状态码为400代表业务处理错误，并在客户端提醒用户错误的原因。用户登录接口设计表 4-8所示：

表 4-8 用户登录接口

API 地址	/api/User/Login			
请求方式	POST			
应用场景	用户登录系统			
请求参数	变量名	类型	示例值	描述
	UserToken	String	User1	用户邮箱/名字
	UserPassword	String	1234	用户密码
返回结果	statusCode	Int	200	返回状态值
	Message	String	Success	返回文本信息
	Data	String	jwt 令牌	登陆后的有效期 jwt 令牌

4.4.2 Scrapy 子系统接口设计

在4.3.3小节介绍了Scrapy的架构实现，由于爬虫任务可以由管理员以及系统定时任务进行。所以务必实现一个接口来分别让它们进行调用。由于本文是使用Python语言进行Scrapy爬虫框架的实现，所以提供的API为程序API，不同于4.4.1所示的服务器API，不是使用RESTful API。但是仍然可以调用商品爬取接口如表 4-5所示的通过RESTful API进行数据的爬取。

当用管理员调用该API时，并按照下述的请求参数列表发送商品信息、平台信息、遍历次数以及是否为测试时将会发送请求到服务端并进行相应的数据爬取。

由于进行爬虫会消耗较大的时间，所以本系统利用了Hangfire进行该服务的调用，在该服务完成的时候将会返回爬取的状态，信息将保存到Hangfire生成的数据库内。表 4-9展示了调用Scrapy子系统的接口：

表 4-9 Scrapy 接口表

API 地址	\Product\crawlerProcess.py -c mobile -b apple -m iphone 15 pro max -s mudah aihuishou -t 1 -i 10			
请求方式	Python 运行			
应用场景	管理员/系统进行数据爬取			
请求参数	变量名	类型	示例值	描述
	-c	String	Mobile	商品种类
	-b	String	Apple	商品品牌
	-m	String	Iphone 15 pro	商品型号
	-s	String	Mudah aihuishou	爬取商品平台数组
	-t	Int	1	是否为测试爬取
	-i	Int	10	爬取次数
返回结果	ExitCode	Int	0	0表示成果1表示失败

4.4.3 TermCompare 接口设计

TermCompare作为Scrapy子系统里对商品型号分类的匹配算法，在该系统扮演者了非常重要的角色，它可以直接影响用户的体验，所以务必实现一个接口来让用户自行替换内部实现以达到更高的匹配准确率，表 4-10展示了TermCompare接口表：

表 4-10 TermCompare 接口表

方法	ScrapyTermCompare.GetMostSimilarProduct			
请求方式	Python 包			
应用场景	系统进行商品型号分类数			
请求参数	变量名	类型	示例值	描述
	Title	String	Iphone 14 pro max 128gb	商品标题
返回结果	Most_similar_product	String	Iphone 14 pro max	判断最相近的产品
	Most_similar_category	String	Mobile	判断最相近的种类
	Cleaned_title	String	Iphone 14 pro max 128 gb	经过清理后的标题
	Distance	Float	1.00	相似度

4.5 数据库设计

对于本系统将会利用两个数据库分别存储不同类型的数据，对于不需要进行字段检索的数据如：用户、搜索历史、用户验证码等等将会存储在Sql Server里的TimmyDBV2数据库内；而对于需要进行字段检索的数据如实时商品将会存储到ElasticSearch内，方便进行检索操作。

4.5.1 关系型数据库设计

本小节主要介绍系统的主要数据表结构进行介绍，包括管理员信息表、用户信息表、用户验证码表、用户搜索历史表、用户收藏表、通知表、Timmy商品表、历史价格表、订阅商品表、订阅商品物品表等如图 4-13所示：

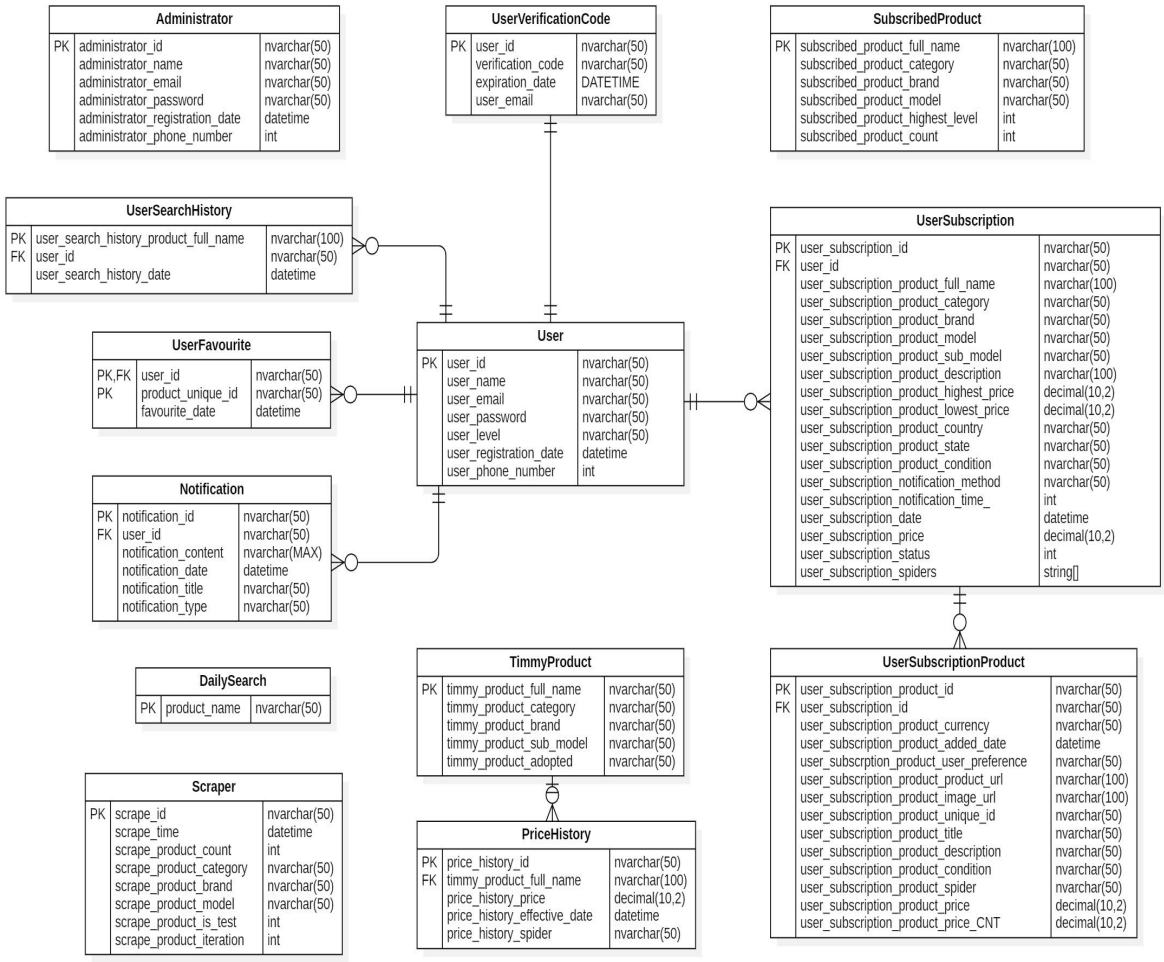


图 4-13 关系型数据库设计图

4.5.2 非关系型数据库设计

非关系型数据库的设计只设计了一个索引（index）具体见4.3.4节。

4.6 本章小结

本章的主要介绍内容是系统的架构设计及实现。首先通过系统总体架构设计来给出本文系统的整体概念包括系统的前端以及后端如何链接。其次分别介绍了客户端以及服务端架构，并详细介绍了客户端以及服务端所提供的功能。在服务端同时详细介绍了Scrapy的整体爬取流程，并进行了接口设计的介绍。最后进行了数据库的设计。为下一阶段的详细设计和编码实现打下了初步想法。

第5章 基于Android的二手商品实时比价系统系统实现与测试

本章在需求分析和架构设计完成的基础上，首先讲述了系统功能模块的详细设计和实现方案。每个模块通过时序图和流程图描述功能业务流程。接下来，采用黑盒测试方法设计测试用例，对系统进行功能和非功能性测试

5.1 系统模块设计

5.1.1 欢迎界面模块

用户启动APP进入程序主欢迎界面，欢迎界面包含立即登录按钮以及APP简单介绍。若用户曾经登陆过该程序，将会直接跳转至主页面。除此之外，在登录界面有一个选项可以让用户注册账号，方便新用户进行账号注册，欢迎界面的流程图如图5-1所示。

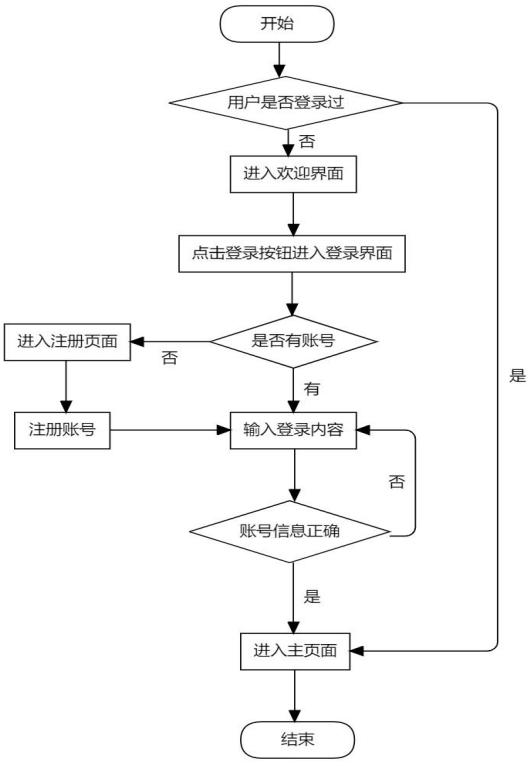


图 5-1 欢迎界面流程图

用户在进入登陆页面前，系统将会判断用户是否曾经登录过，若登陆过且经过JwtService确定令牌有效则返回主页面；否则跳转至登录界面，在登陆界面，用户被

要求填写的信息包括用户邮箱/用户名以及密码，并发送给UserService进行验证。同时生成Jwt令牌以便下次登录不需要再进行登录而是直接跳转至主页面。用户登录顺序图如图 5-2所示：

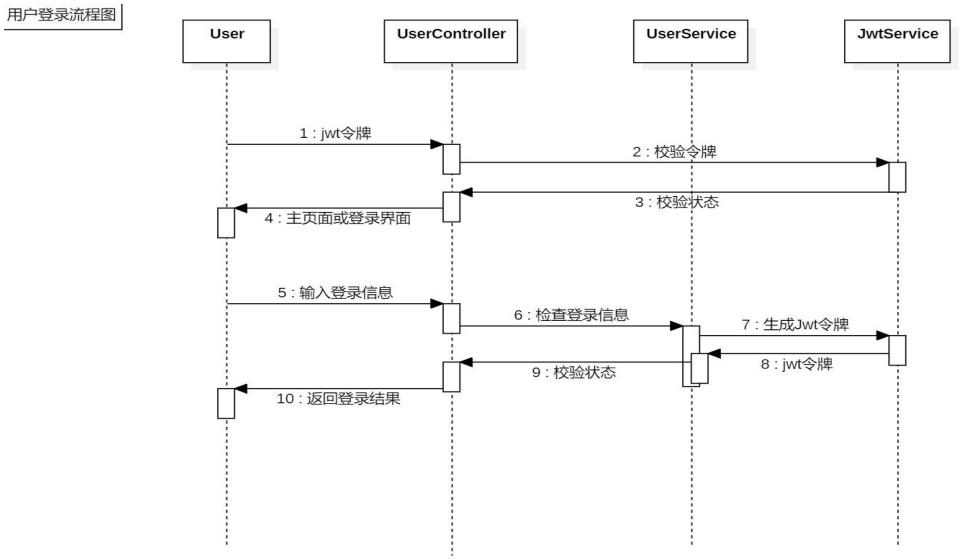


图 5-2 用户登录顺序图

用户再注册界面填写完注册信息后，点击发送验证码调用UserVerficationCodeController发送验证码，系统会保存限时验证码并且发送到特定邮箱，随后用户输入验证码后点击注册按钮，进行账号注册。用户注册顺序图如图 5-3所示：

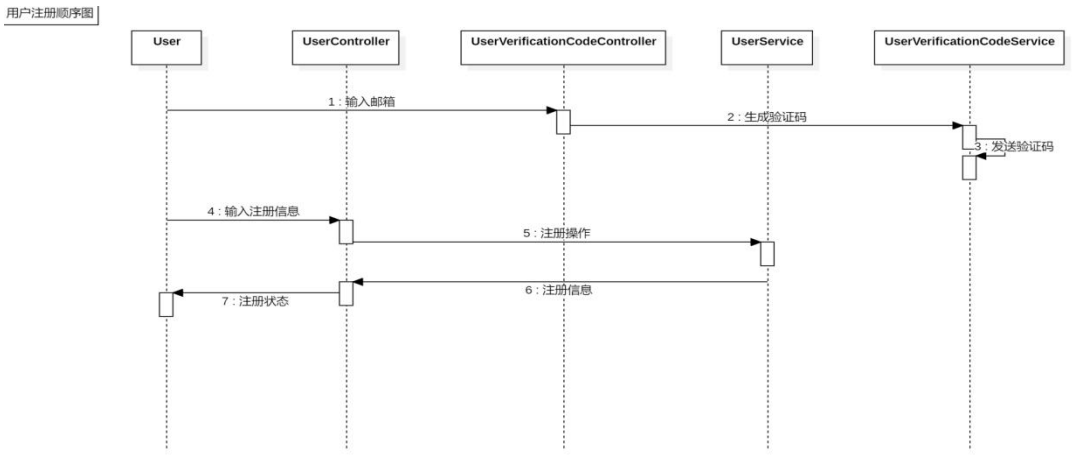


图 5-3 用户注册顺序图

5.1.2 商品搜索模块

用户进入搜索界面后，可以选择系统提供的商品类型、品牌、型号进行商品的筛选，并且会将用户搜索记录保存起来，随后从ElasticSearch获取数据返回给用户；若用户想要搜索的商品不在系统提供范围内，用户可以请求录入商品，并等待管理员的审批状态。搜索模块的流程图如图 5-4所示：

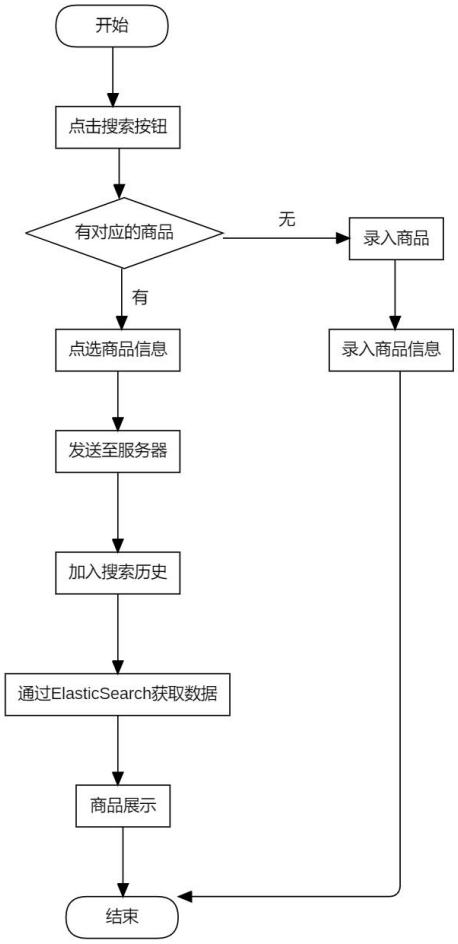


图 5-4 商品搜索流程图

用户再搜索商品选择对应的商品类别、品牌、型号、平台等等搜索参数后，将调用ElasticSearchController接口，随后服务器会通过ElasticSearch业务的SearchProduct获取商品列表，同时调用UserSearchHistory业务进行搜索的保存，并返回商品列表给用户。用户搜索顺序图如图 5-5所示：

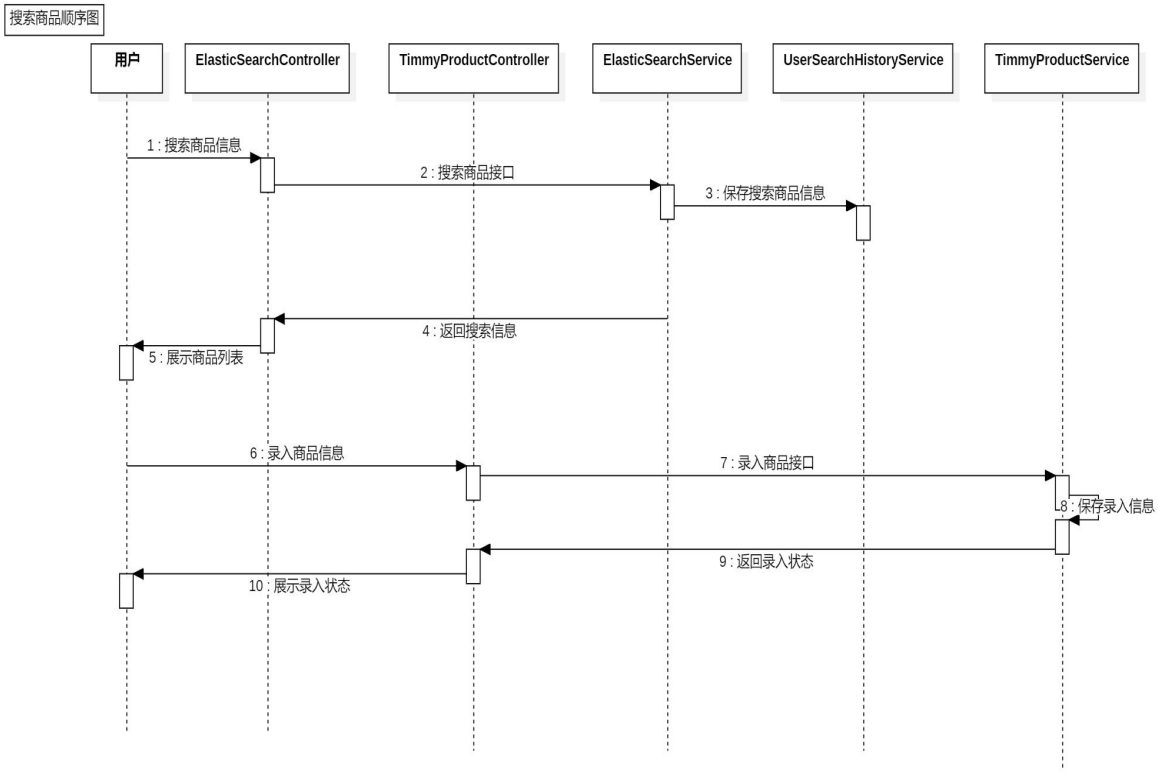


图 5-5 用户搜索顺序图

5.1.3 订阅模块

用户进入订阅界面后，可以选择系统提供的商品类型、品牌、型号、订阅通知时间等进行商品的筛选，并且会将其记录保存起来，随后系统会每日特定时间发送最新消息给用户；若用户想要搜索的商品不在系统提供范围内，用户可以请求录入商品，并等待管理员的审批状态。搜索模块的流程图如图 5-6所示。

用户再订阅商品选择对应的商品类别、品牌、型号、订阅通知时间等等搜索参数后，客户端将调用UserSubscriptionController接口，随后服务器会通过UserSubscription业务的AddUserSubscription进行判断输入是否正确并保存至UserSubscription表，同时调用SubscribedProduct业务的AddSubscribedProduct进行订阅商品的更新，保证同步。用户订阅商品顺序图如图 5-7所示：

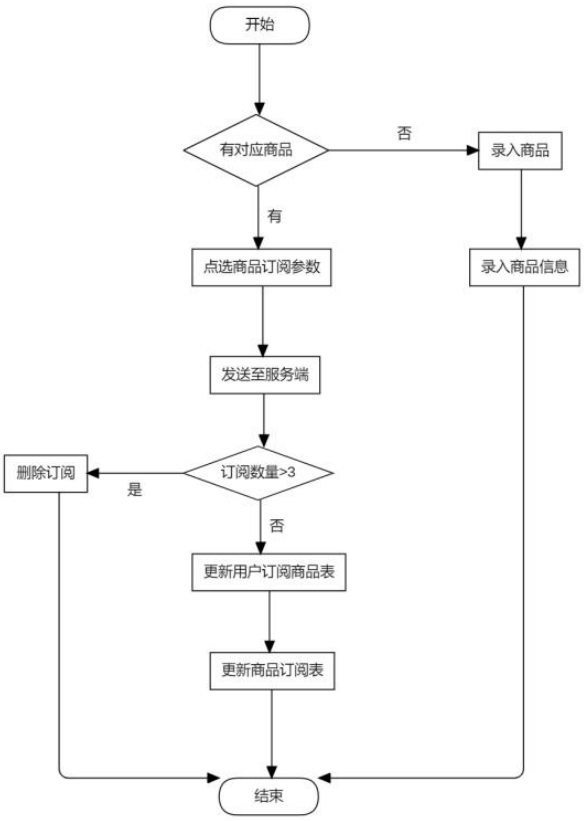


图 5-6 商品订阅流程图

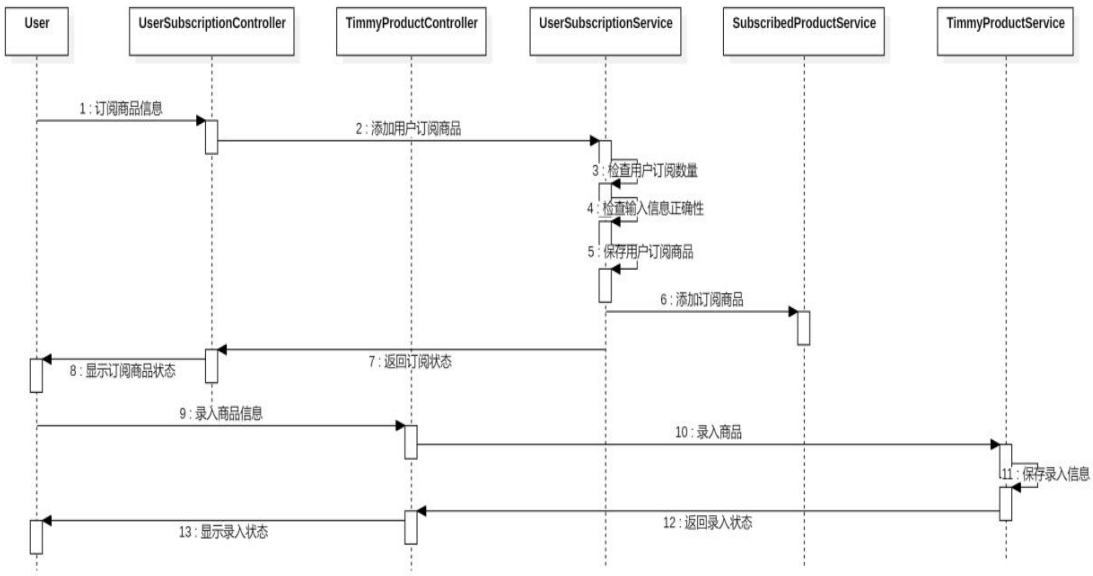


图 5-7 商品订阅顺序图

5.1.4 收藏模块

用户再任何有商品列表的界面可以点击收藏按钮，前端将会判断其是否进行收藏还是取消收藏功能，并发送相应的请求给服务器；收藏流程图如图 5-8所示：

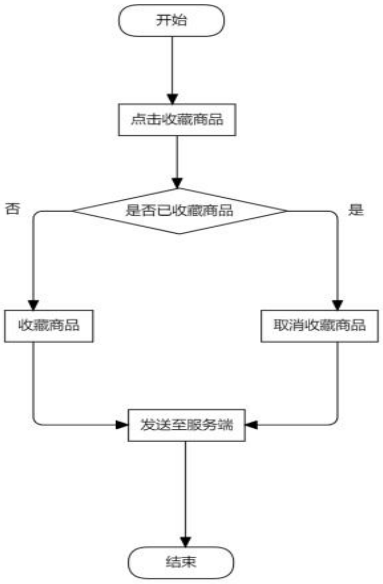


图 5-8 商品收藏流程图

用户在点击收藏商品图表的时候，会通过UserFavourite的GetUserFavourite事先获取的用户收藏列表进行判断用户此次操作应该调用UserFavouriteController接口的收藏还是取消收藏商品。用户收藏商品顺序图图 5-9所示：

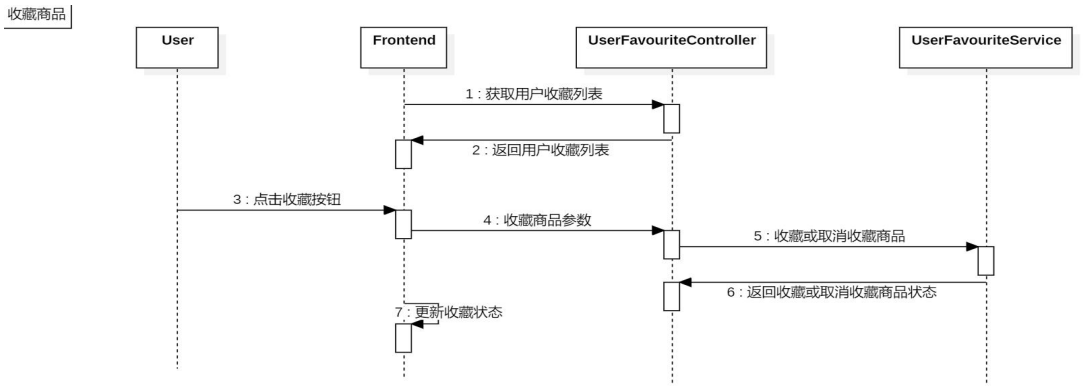


图 5-9 商品收藏顺序图

5.1.5 用户信息模块

1. 个人信息详细设计流程

用户在登录后进入个人信息界面将会发送Jwt令牌，后端验证Jwt令牌后会返回用户的公共信息，便用户查看个人信息，同时用户可以进行密码的更改，系统会调用UserController的ResetPassword接口进行更新。用户信息顺序图如图 5-10所示：

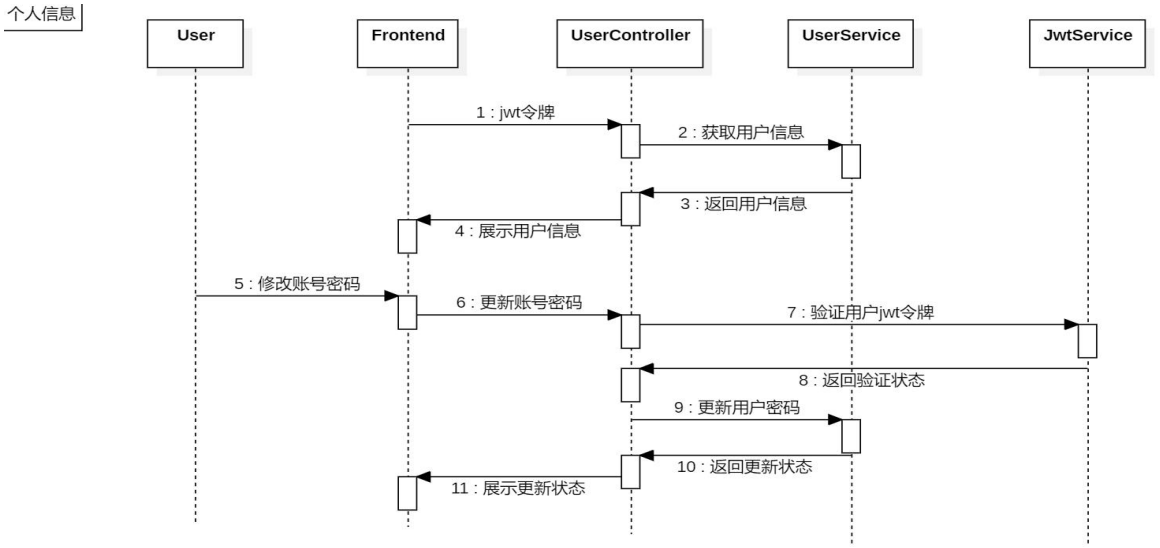


图 5-10 修改个人信息顺序图

2. 用户查找历史详细设计流程

用户在登录后进入个查找历史界面将会发送Jwt令牌，系统会调用UserSearchHistoryController的GetUserSearchHistory接口进行查询历史的获取。首先验证Jwt令牌并返回用户的个人id，然后会发送用户个人id获取用户查找历史记录。用户查找历史顺序图如图 5-11所示：

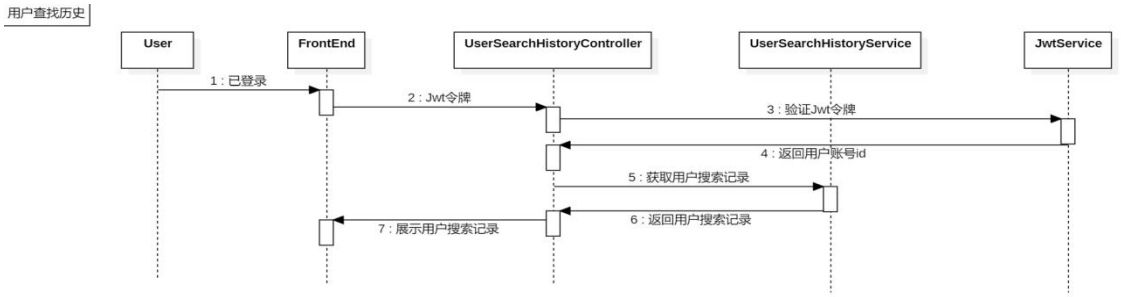


图 5-11 查找历史记录流程图

5.1.6 录入商品模块

录入商品模块分别分为用户录入商品以及管理员录入商品。用户进入APP时可在搜索界面或订阅界面点击录入商品以申请新的商品供用户选择。随后管理员可以在管理录入商品界面进行查看用户申请的商品以便进行录入进系统。录入商品的流程图如图 5-12所示。

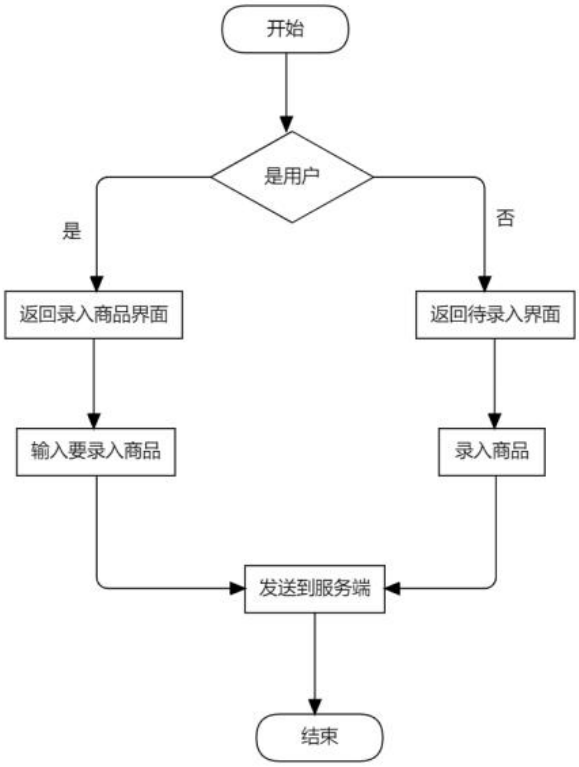


图 5-12 录入商品流程图

用户/管理员在登陆录入商品模块时会发送Jwt令牌并辨认操作人员是否为用户还是管理员，前端并返回相应的录入商品界面。对于用户，用户输入要申请的商品种类、品牌以及型号，并发送请求，后端将使用TimmyProductController的AddTimmyProduct并且adopt字段的数值为0，代表待录入。对于管理员，管理员会获得待录入的商品，其adopt字段数值为0，并通过人工验证将其录入系统。录入商品顺序图如图 5-13所示：

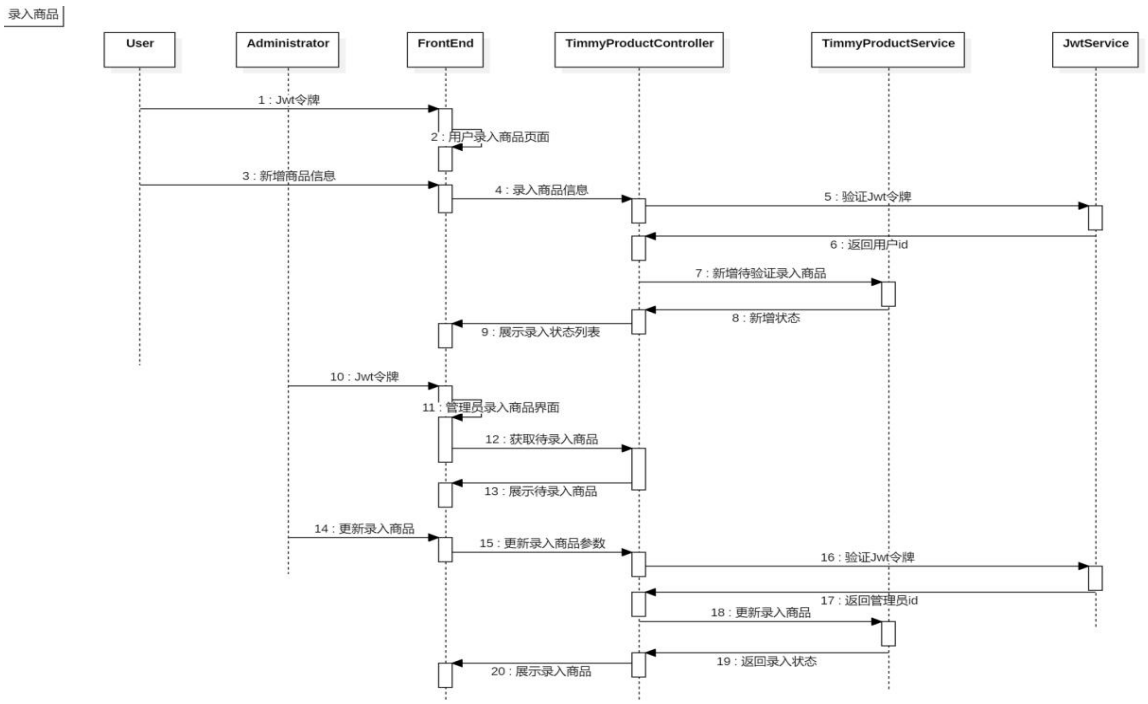


图 5-13 录入商品顺序图

5.1.7 爬虫模块

爬虫模块分别分为订阅商品爬虫、种类品牌爬虫以及普通爬虫。管理员可在后台程序进行爬虫类型的参数输入以及选定爬虫方式即可进行相应的爬取。爬虫的流程图如图 5-14所示。

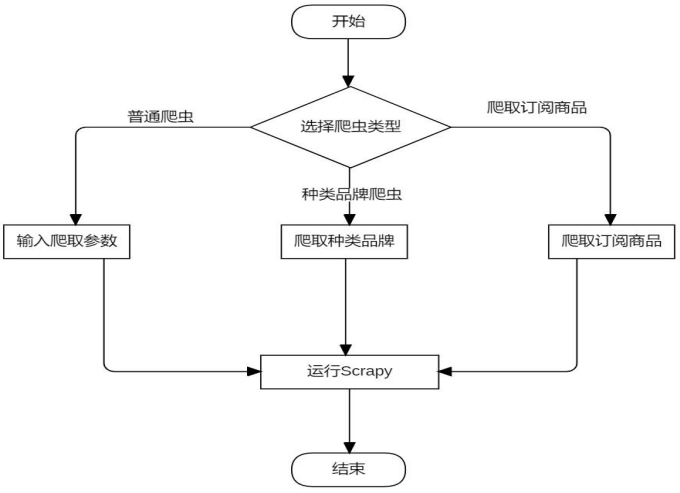


图 5-14 爬虫流程图

5.1.8 商品更新模块

本系统通过增量式爬虫获取数据，因此需要定期检查和更新数据，以确保信息的实时性，并避免已下架的商品仍存在于系统内。为解决这一问题，本文提出了分流定时商品更新策略。系统每小时对k个商品进行更新操作，从而避免过度请求对方服务器。

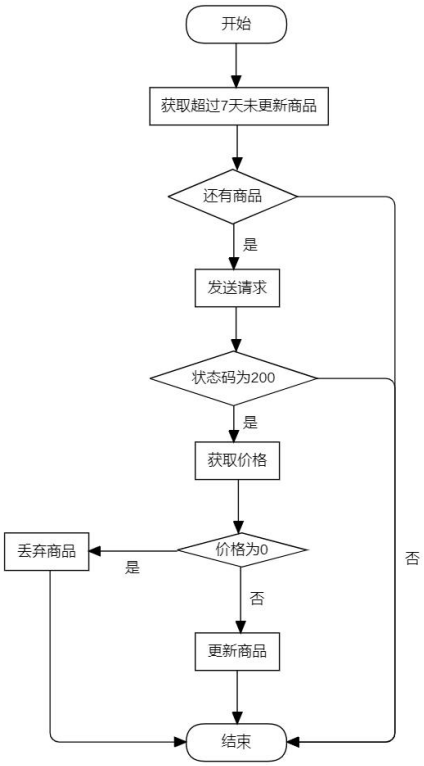


图 5-15 商品更新流程图

5.1.9 商品型号类型匹配（TermCompare）

对于一个提供二手商品信息的系统来说，一个非常至关重要的问题就是该系统所返回的商品是否有意义，换另外一种说法就是系统所返回的商品是否符合用户的需求[22]。针对此问题，本文研究重点方向即是如何将网页抓取模块所爬取的二手商品商品以更高效以及准确的方法分类。

通过提供数据库以及ElasticSearch，即可输入商品标题进行相应的商品型号类型匹配，首先标题会经过语言切换将各国语言统一转换成英语，然后通过词干提取模块将冗余的字、标点符号进行去除，随后进行TF+Cosine相似度匹配获取两种结果，

若两中结果长度相等，则直接获取相似度更高的那个，否则那两个结果进行LD计算获取结果。

图 5-16展示了商品型号分类的流程图。

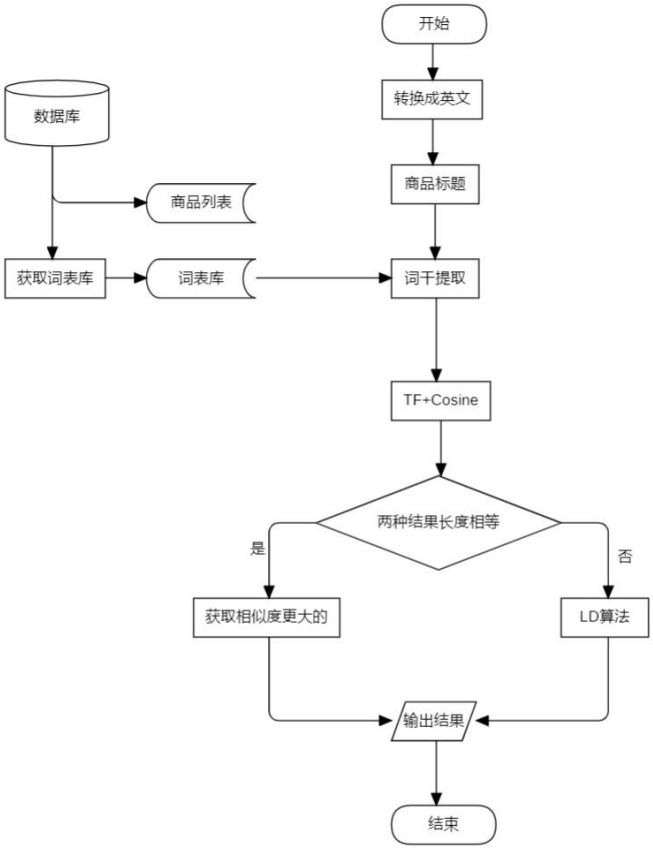


图 5-16 商品型号分类算法流程图

由于用户所提供的商品名称时常会根据用户个人习惯进行输入，常见的情况如表 5-1所示：

表 5-1 商品名称情况

类型	例子
字符链接	Promax, 11tpro
字符于数字连接	2GEN, 1st
字符于数字分开	12 c, 10 t
被符合包围	(1GEN)
冗余的字	全新 iphone 14 pro

对于以上的这些情况，会分别使用两种不同的正则表达式如表 5-2所示，最后得到词干提取的流程图如图 5-17所示：

表 5-2 正则表达式表

	类型	正则表达式
1	去除中文字	r’[\u4e00-\u9fff]
2-1	清理（数字分开）	r’[a-zA-Z]+|[\d.]+|[^,.()\-\s]+’
2-2	清理（数字不分开）	r’[a-zA-Z0-9]+|[\d.]+|[^,.()\-\s]+’

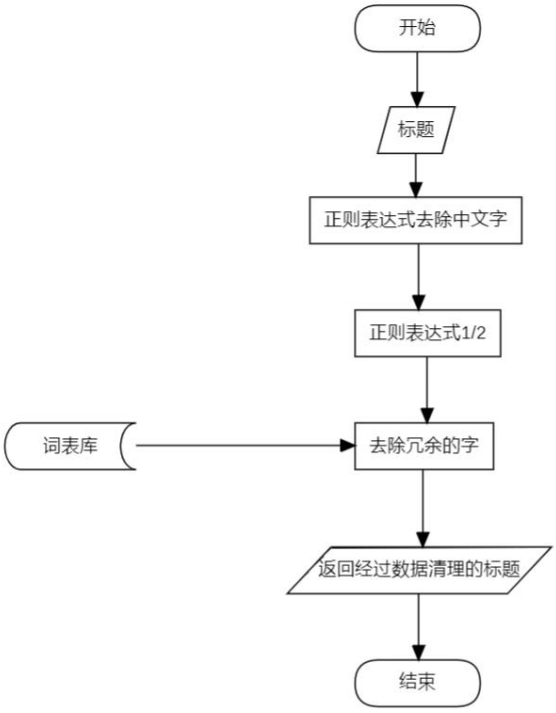


图 5-17 词干提取流程图

其中词表库是通过读取商品的品牌获取的，例如对于品牌苹果的词表库为['iphone', '11', 'pro', 'max', '12', '13', '14', '15', '6', '6s', 'plus', '7', '8', 'se', 'x', 'xr', 'xs']。

5.1.10 订阅提醒模块

订阅提醒模块为系统再每小时自动进行调用的功能，用以提醒用户有新的符合要求的商品已被探索到，系统会根据用户在订阅商品时设置的提醒时间以及提醒方式进行提醒用户。

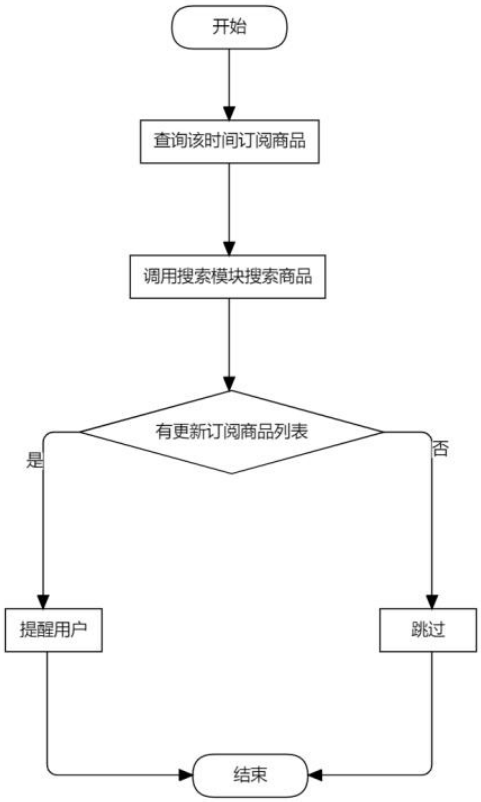


图 5-18 订阅提醒流程图

5.1.11 商品比价功能

本系统的主要功能之一是对比不同平台的商品价格，为用户提供最佳价格数据。然而，由于用户的个性化需求，不同用户对商品比价的要求可能有所不同。例如，有些用户可能需要“256GB”容量的手机，而另一些用户可能需要“128GB”容量的手机。因此，比价功能不能仅仅通过搜索某商品型号的最低价格直接返回结果。

为了实现更精准的商品比价功能，结合了ElasticSearch强大的文本搜索功能和本系统在爬虫阶段通过商品型号类型匹配算法所得出的商品类型。这种方式能够有效地返回有用的信息给用户。

系统可以基于用户提供的商品详情进行比价。例如，用户在订阅时提供了以下表 5-3中的数据。系统会根据表中的所有数据项进行过滤，并将过滤结果发送给ElasticSearch进行商品搜索，然后按价格排序，流程图如图 5-19所示。

表 5-3 用户订阅输入参数

变量名	用户输入
平台	Mudah, Aihuishou
型号	Iphone 13
品牌	Apple
种类	Mobile
商品详情	128
价格区间	1000 至 5000
商品品质	新

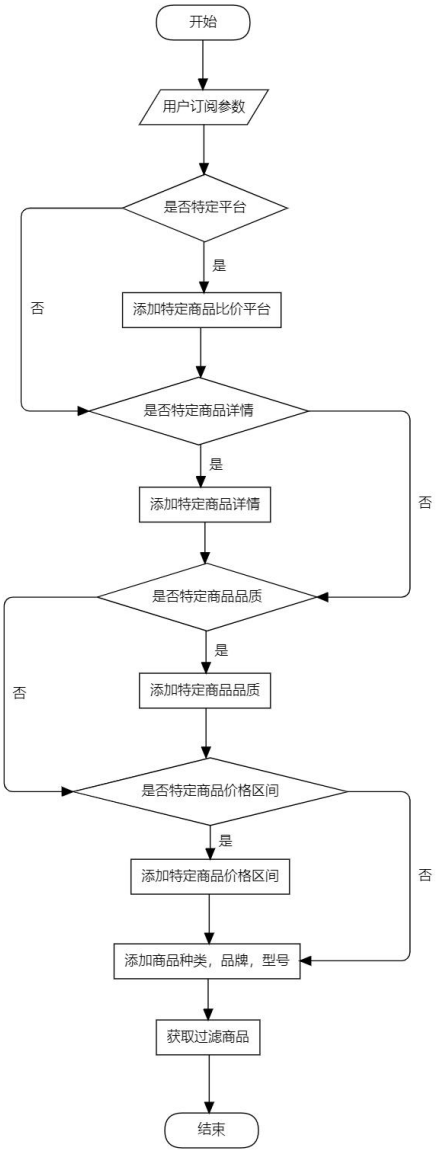


图 5-19 商品比价流程图

5.2 测试方法与环境

本小节将使用黑盒的测试方式对系统进行测试来确保该系统能够顺利的运行并返回理想的结果给用户，测试的功能主要包括搜索、订阅、收藏功能得等等。手机测试配置如表 5-4所示，部署环境如表 5-5所示。

表 5-4 手机测试环境

环境配置	小米 A1（2018）
系统内核	安卓 9
CPU 型号	高通 骁龙 625
RAM 内存容量	4G

表 5-5 部署环境配置

环境配置	华为 Matebook 14 2020 AMD
操作系统	Windows 10 Home Basic
CPU 型号	AMD Ryzen 7 4800H
RAM 内存容量	16G
GPU 显卡	集成显卡

5.3 系统测试

本小节将测试第四章内需求分析所提出的功能从用户角度、管理员角度进行测试，分别需要进行的功能测试包括：

1. 用户 - 登录、注册、忘记密码、登出、查询物品、订阅/取消订阅商品、收藏/取消收藏商品

2. 管理员 - 添加Timmy商品、删除Timmy商品、管理后台进程、爬取商品、更新数据、爬取订阅物品、爬取种类物品、索引最低价格商品

5.3.1 用户功能测试

表 5-6 用户登录测试

测试目的	测试操作	预期结果	测试结果
用户是否能正常输入	用户输入正确账号名以及密码	用户正常输入	正常
用户是否能够登录	用户点击登录按钮	进入主页面	正常

表 5-7 用户注册测试

测试目的	测试操作	预期结果	测试结果
用户是否能正常输入	用户输入合理账号名、邮箱以及密码	用户正常输入	正常
用户是否能够获取验证码	输入正确邮箱的情况点击发送验证码	将验证码发送到对应邮箱	正常
用户是否能够注册账号	用户点击注册按钮	系统返回注册成功	正常

表 5-8 用户忘记密码测试

测试目的	测试操作	预期结果	测试结果
用户是否能正常输入	用户输入合理邮箱以及新密码	用户正常输入	正常
用户是否能够获取验证码	输入正确邮箱的情况点击发送验证码	将验证码发送到对应邮箱	正常
用户是否能替换密码	用户点击更换密码按钮	系统返回更换成功	正常

表 5-9 用户登出测试

测试目的	测试操作	预期结果	测试结果
用户是否能正常登出	用户点击登出按钮	用户正常登出	正常

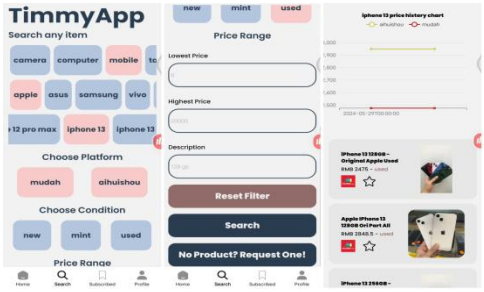


图 5-20 用户查询商品测试

图 5-20显示了用户查询的整体流程，其中左图显示的是用户选择一个型号，中图显示的是用户选择搜索平台、品质、价格以及详情。最后点击搜索后进入商品列表页面如右图所示。

表 5-10 用户查询商品测试

测试目的	测试操作	预期结果	测试结果
用户是否能进入搜索界面	用户点击搜索界面	用户进入搜索界面	正常
用户是否能够正常选择查询商品型号	用户点击一个搜索商品型号	用户成功选择一个商品型号	正常
用户是否能够正常选择多个商品平台	用户点击要搜索的平台	用户成功选择一个或多个平台	正常
用户是否能够正常选择商品品质	用户点击商品品质按钮	用户成功选择一个商品品质	正常
用户是否能够输入商品最低及最高价格	用户输入最低以及最高价格	用户成功填写价格	正常
用户是否能够输入商品详情	用户输入搜索详情	用户成功写入搜索详情	正常
用户是否能够查询商品	用户点击查询商品	用户跳转至商品列表页面	正常
用户是否获取商品列表	用户观看商品列表页面	显示搜索商品列表	正常

图 5-21显示了用户订阅的整体流程，其中左1图显示的是用户选择一个型号，左2图显示的是用户选择搜索平台、品质、消息提醒方式以及提醒时间。右2图显示了用户输入价格区间以及详情，最后点击订阅后进入用户订阅列表页面如右1图所示。

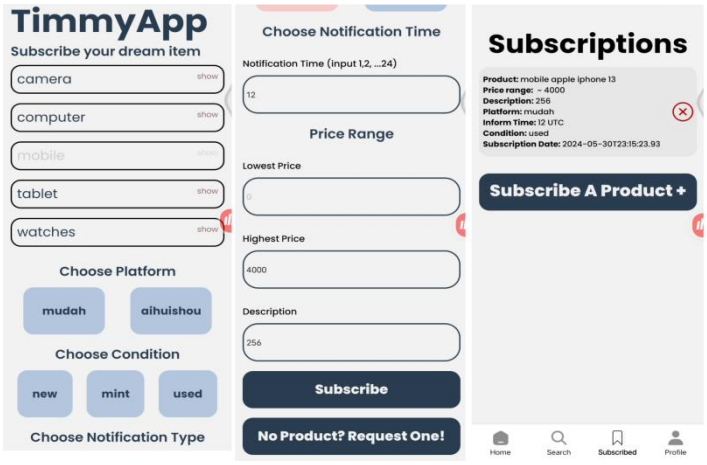


图 5-21 用户订阅商品测试

表 5-11 用户订阅商品测试

测试目的	测试操作	预期结果	测试结果
用户是否能进入订阅界面	用户点击订阅界面	用户进入订阅界面	正常
用户是否能够正常选择订阅商品型号	用户点击一个订阅商品型号	用户成功选择一个商品型号	正常
用户是否能够正常选择多个商品平台	用户点击要订阅的平台	用户成功选择一个或多个平台	正常
用户是否能够正常选择商品品质	用户点击商品品质按钮	用户成功选择一个商品品质	正常
用户是否能够选择提醒方式	用户点击订阅提醒方式	用户成功选择提醒方式	正常
用户是否能够输入提醒时间	用户输入订阅提醒时间	用户成功写入提醒时间	正常
用户是否能够输入商品最低及最高价格	用户输入最低以及最高价格	用户成功填写价格	正常
用户是否能够输入商品详情	用户输入订阅详情	用户成功写入订阅详情	正常
用户是否能够订阅商品	用户点击订阅商品	用户跳转至订阅界面	正常

表 5-12 用户收藏商品测试

测试目的	测试操作	预期结果	测试结果
用户是否能点击收藏按钮	用户点击收藏按钮	用户成功点击收藏按钮	正常

5.3.2 管理员功能测试

对于管理员，通过黑盒测试了主要的核心功能。其核心的功能包括添加Timmy商品、管理后台进程、爬取商品、更新数据、爬取订阅物品以及爬取种类物品。

表 5-13 添加 Timmy 商品测试

测试目的	测试操作	预期结果	测试结果
管理员能否正常输入 Timmy 商品参数	管理员输入 Timmy 商品参数	管理员能够成功输入参数	正常
管理员是否能够成功添加 Timmy 商品	管理员点击发送请求按钮	管理员能够成功发送添加请求	正常

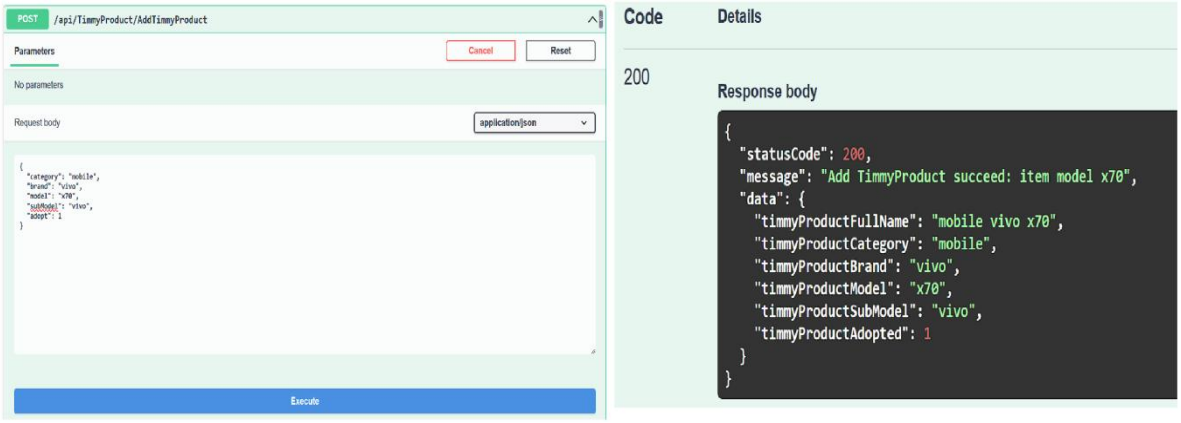


图 5-22 添加 Timmy 商品测试

表 5-14 管理后台进程测试

测试目的	测试操作	预期结果	测试结果
管理员能否正常查看后台进程	管理员进入后台进程页面	管理员可查看后台进程列表	正常
管理员是否能够停止后台进程任务	管理员点击要停止的进行	管理员能够停止后台进程	正常
管理员是否能够开始一项后台进程	管理员点击要运行的进程	管理员能够开始一项后台进程	正常

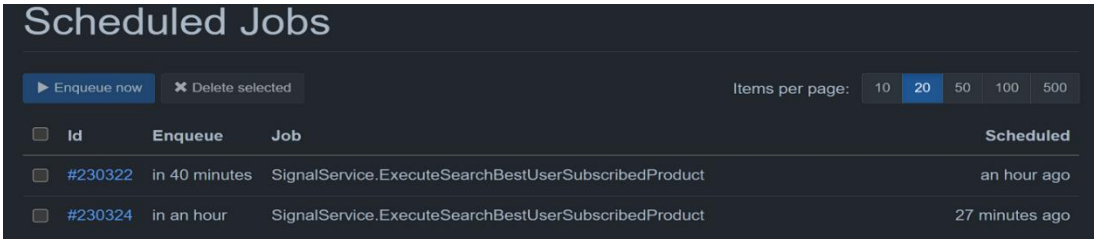


图 5-23 管理后台进程测试

表 5-15 爬取商品测试

测试目的	测试操作	预期结果	测试结果
管理员是否能够输入爬取商品参数	管理员输入爬取商品参数	管理员能够成功输入参数	正常
管理员是否能够爬取商品	管理员点击发送请求按钮	管理员能够成功发送爬取请求	正常

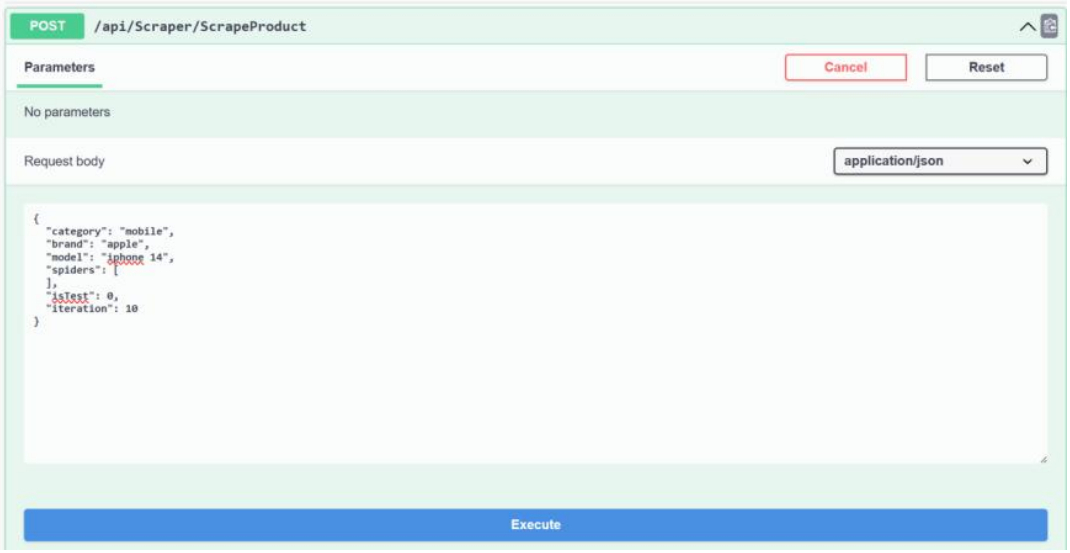


图 5-24 管理员输入爬取商品参数

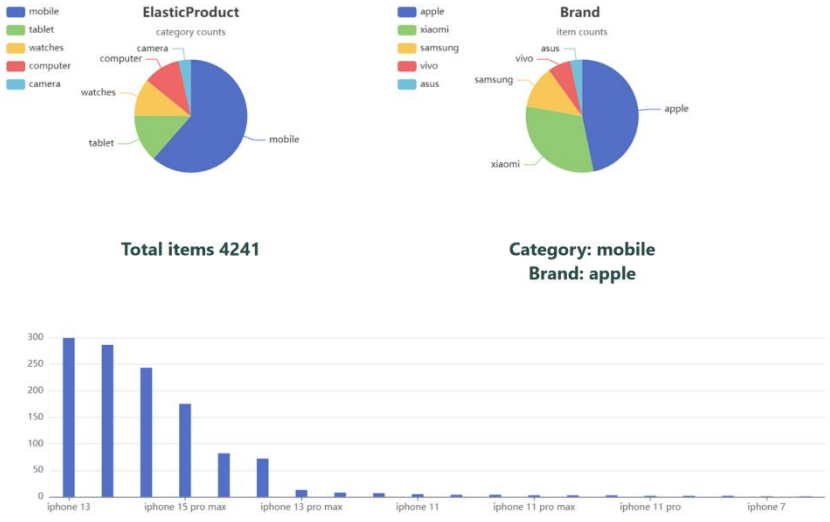


图 5-25 爬取商品前数据

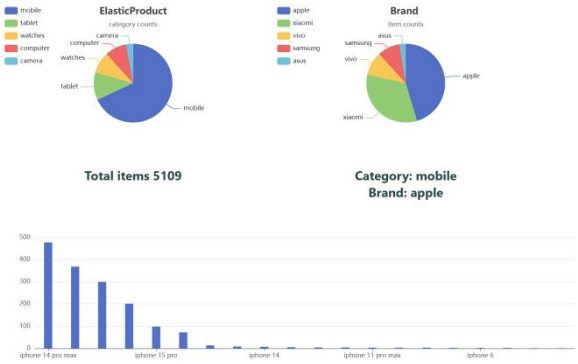


图 5-26 爬取商品后数据

表 5-16 更新商品测试

测试目的	测试操作	预期结果	测试结果
管理员能否正常更新商品	管理员点击更新商品按钮	管理员能够成功发送更新请求	正常

表 5-17 爬取订阅商品测试

测试目的	测试操作	预期结果	测试结果
管理员是否能够输入爬取特定等级订阅商品	管理员输入订阅等级	管理员能够成功输入参数	正常
管理员是否能够爬取商品	管理员点击发送请求按钮	管理员能够成功发送爬取请求	正常

表 5-18 爬取种类商品测试

测试目的	测试操作	预期结果	测试结果
管理员是否能够输入爬取种类品牌商品	管理员点击发送请求按钮	管理员能够成功发送爬取请求	正常

5.3.3 商品型号类型测试

为了模拟真实的爬取到分类的情况，本文通过数据抓取模块分别从马来西亚二手平台Mudah抓取了苹果Apple以及小米Xiaomi的商品，选择马来西亚二手平台Mudah是因为该平台上的商品标题都是由用户自行填写的，所以更加能够检查本文所

使用的算法是对分类商品有效的。图 5-27显示了使用Mudah平台搜索Iphone13所返回的结果，容易发现对于“pro max”该字也可能出现为“promax”。

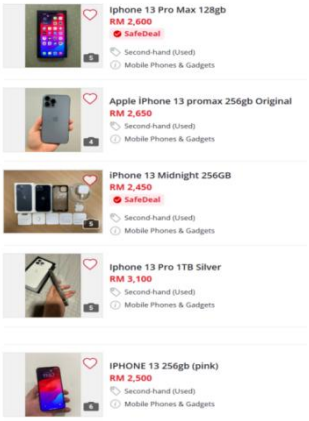


图 5-27 马来西亚二手平台 Mudah 查询 iphone 13

图 5-28 爬取 Mudah 的 Iphone 商品列表

图 5-28给出了通过数据抓取模块抓取的Iphone商品列表共有188项数据，另外也爬取了小米商品共有187项数据分别存储再各自的json文件内。随后为了方便测试，对这两个文件进行初始化如图 5-29所示，其中每一个对象内的属性如下所解释：

1. title：商品原标题
2. trimmedTitle：商品进行词干提取后的标题
3. predictedModel：预测型号
4. realModel：手动录入的正确型号
5. Category：将商品分类到种类

6. Distance：进行余弦相似度计算或最短编辑距离的指标

```
{
  "count": 188,
  "correct": 0,
  "executedTime": 0,
  "data": [
    {
      "title": "Apple IPhone 15 Pro Sealed",
      "trimmedTitle": "",
      "predictedModel": "",
      "realModel": "iphone 15 pro",
      "category": "",
      "distance": 0
    },
    {
      "title": "APPLE WATCH S7-41M- Untuk dilepaskan",
      "trimmedTitle": "",
      "predictedModel": "",
      "realModel": "watch 7",
      "category": "",
      "distance": 0
    },
    {
      "title": "Apple watch series 8 45mm cellular",
      "trimmedTitle": "",
      "predictedModel": "",
      "realModel": "watch 8",
      "category": "",
      "distance": 0
    },
    {
      "title": "Apple watch Ultra 2",
      "trimmedTitle": "",
      "predictedModel": "",
      "realModel": "watch ultra 2",
```


图 5-29 初始化测试文件

表 5-19 正则表达式表

	类型	正则表达式
1	去除中文字	r'[\u4e00-\u9fff]
2-1	清理（数字分开）	r'[a-zA-Z]+|[\d.]+|[^,.()\-\s]+'
2-2	清理（数字不分开）	r'[a-zA-Z0-9]+|[\d.]+|[^,.()\-\s]+'

表 5-20 商品标题词干提取

输入	（1）ipad mini (6th generation) & apple pencil	（2）xiaomi 13t pro 12/512gb
正则表达式 1	ipad mini (6th generation) & apple pencil	xiaomi 13t pro 12/512gb
正则表达式 2-1	['ipad', 'mini', '6', 'th', 'generation', '&', 'apple', 'pencil']	['xiaomi', '13', 't', 'pro', '12', '/512gb']
正则表达式 2-2	['ipad', 'mini', '6th', 'generation', '&', 'apple', 'pencil']	['xiaomi', '13t', 'pro', '12', '/512gb']
除去不存在与词表库内的词 3-1	['ipad', 'mini', '6', 'pencil']	['13', 'pro', '12']
除去不存在与词表库内的词 3-2	['ipad', 'mini', 'pencil']	['13t', 'pro', '12']

表 5-20是通过输入两个不同类型的商品分别经过标题词干提取所得到的结果，可以发现对于输入（1）通过正则表达式2-1所得到的结果更符合要求，因为包含关键字['ipad', 'mini', '6']，而对于输入（2）通过正则表达式2-2所得到的结果更加符合要求，因为包含关键字['13t', 'pro']，该实验也证实了需要使用两种不同的正则表达式以获取更准确的结果。

TF算法已于2.2.3小节进行介绍了，不结合逆文档频率（IDF）一起使用是因为对于标题类型并不会有太多的多余内容而且也进行了词干提取所以不需要使用IDF来调整权重，经过数据统计，也明确了这个说法。随后，通过计算完标题以及商品库里的TF值后，进行余弦相似度计算，表 5-21显示了同样的商品经过TF+Cosine相似度计算的结果，可以发现通过两种不同的正则表达式得出的结果不一样，如对于输入（1）正确的结果应该是通过正则表达式2-1的3选项，而对于输入（2）正确的结果应该是正则表达式2-2的3选项。

表 5-21 商品经过 TF+Cosine 相似度计算结果

输入	（1）ipad mini (6th generation) & apple pencil	（2）xiaomi 13t pro 12/512gb
经正则表达式 2-1 的输出	Ipad mini 6 pencil	13 pro 12
TF+Cosine 结果	1. ipad air 6 - 相似度: 0.5773502691896258 2. ipad mini 5 - 相似度: 0.5773502691896258 3. ipad mini 6 - 相似度: 0.8660254037844388	1. redmi 12 pro - 相似度: 0.6666666666666669 2. 12 pro - 相似度: 0.8164965809277261 3. 13 pro - 相似度: 0.8164965809277261
结果 1	Ipad mini 6	13 pro
经正则表达式 2-2 的输出	ipad mini pencil	13t pro 12
TF+Cosine 前三	1. ipad mini 4 - 相似度: 0.6666666666666669 2. ipad mini 2 - 相似度: 0.6666666666666669 3. ipad mini 5 - 相似度: 0.6666666666666669	1. redmi 12 pro - 相似度: 0.6666666666666669 2. 12 pro - 相似度: 0.8164965809277261 3. 13t pro - 相似度: 0.8164965809277261
结果 2	Ipad mini 4	13t pro

因为余弦相似度（Cosine Similarity）不考虑字符串出现的先后顺序，再（2）的2-2结果内发现结果12 pro、13t pro的相似度都为0.8164965809277261。所以本文再进行余弦相似度计算后对比哪个结果在原输入的出现位置更前。如：12和13t分别出现在xiaomi 13t pro 12/512gb的7和15位，因此判定结果13t pro为正确结果。

随后通过结果的长度进行比较，若长度相同，则通过比较他们的相似度，取大的。若长度不相同，则通过利用最短编辑距离（LD）进行计算，获取最短的编辑距离，最终得到结果。

分别对两个json文件进行不同方式的商品型号分类算法，分别有：

1. 词干提取 + 词频 + 逆文档频率 + 余弦相似度 + 最短编辑距离
2. 最短编辑距离
3. 词干提取 + 最短编辑距离
4. 词干提取 + 词频 + 余弦相似度
5. 词干提取 + 词频 + 余弦相似度 + 最短编辑距离

表 5-22 商品型号分类测试结果

测试编号	小米（187 件）		苹果（188 件）	
	时间（s）	准确率（%）	时间（s）	准确率（%）
1	0.46	88%	0.40	95%
2	0.007	42%	0.008	30%
3	0.010	61%	0.010	88%
4	0.16	56%	0.16	95%
5	0.32	93%	0.41	93%

表 5-22显示了利用5中不同算法的结合所获得的测试结果。测试编号1与5的情况很相似，但由于有多一步逆文档的计算，所以选择5作为最终的商品型号分类算法。对于测试编号3以及4，耗时相比于1、5少许多但是准确率不平稳所以舍弃；编号2则准确率太低直接舍弃。

5.3.4 商品订阅提醒测试

为了测试用户订阅商品是否会在有更好的商品出现时系统向用户发送信息，本文让一个用户订阅商品“iphone 14 pro max”，此时用户该订阅商品并没有实际上的任何产品，所以进行了第一轮测试，测试结果也表明系统该功能能够正常的运行图

5-30至图 5-32展示了用户初次订阅商品到管理员手动调用更新商品最后发送信息给用户表示用户订阅商品已更新。

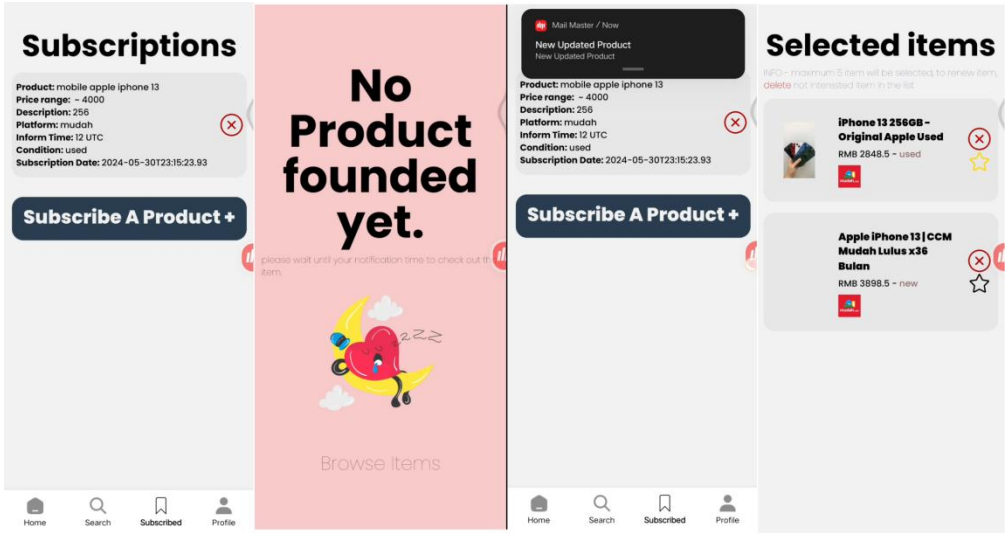


图 5-30 商品订阅提醒测试

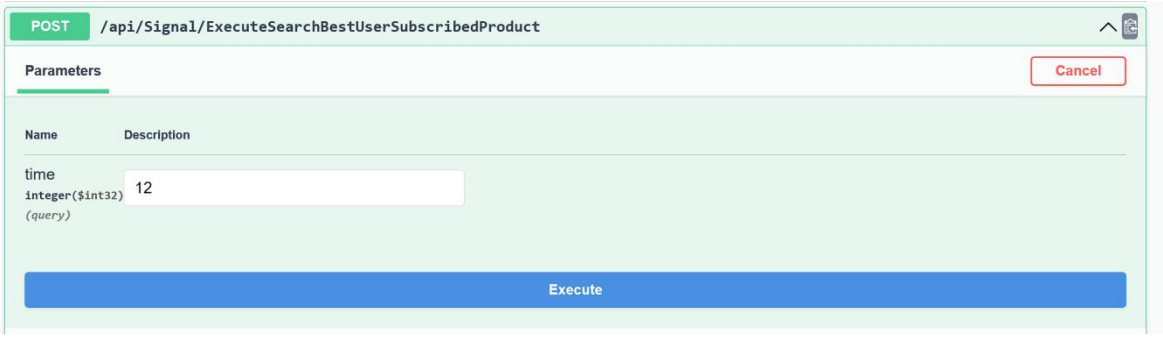


图 5-31 管理员手动进行更新用户订阅商品

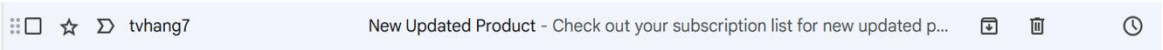


图 5-32 商品订阅提醒邮箱测试

5.4 本章小结

本章主要针对解释了系统的各个模块设计并且在主要功能方面进行了测试，并且测试结果满足系统需求。

结　论

本研究通过研究并实现了一个汇集多个二手平台的系统，并在研究过程中发现使用TF-Cosine+LD混合算法可以以更高的准确率将商品分类到正确的型号下。随后，解决了用户需要在不同二手平台查询价格的问题，也为用户提供了某些国家出售该商品的价格。

在进行该研究期间，发现了这项研究是值得更深入了解的。这是因为世界各地都有售卖二手商品的平台，而不同国家所出售的价格也不同，若能够将该研究范围深入探讨如进行多个国家二手平台汇集并使用更合适的商品型号分类算法将各国不同语言的商品统一，并且提供渠道让买家通过代理购买海外合法商品，则可以为买家省下不少钱以及时间。

由于开发一个完整的系统需要大量的时间，同时也存在对开发的不熟练，在进行软件的需求分析、概要设计以及详细设计的时候不断进行更改，导致对于系统的优化没有做得很好，并且在软件的设计过程中也可能存在不准确的情况。

但通过完成此项目我相信在日后会以更好的方式进行开发项目，并且运用这段时间掌握的知识为日后做铺垫。

参考文献

[1] 艾瑞网. 2023年中国电商市场研究报告 https://t.cj.sina.com.cn/articles/view/1796217437/6b101a5d0190174nk [R], 2024.
[2] 央视财经《经济半小时》. 我国二手手机年交易量达1.52亿台！消费市场吹来“绿色风”：3万亿元大风口已打开 [N]. 央视财经, 2023-2（25）：1.
[3] 胡越明. 不同国家商品价格体系差异形成原因分析 [J]. 合作经济与科技, 2019, (18): 88-91.
[4] ALAM A, ANJUM A A, TASIN F S, et al. Upoma: A Dynamic Online Price Comparison Tool for Bangladeshi E-commerce Websites; proceedings of the 2020 IEEE Region 10 Symposium (TENSYMP), F 5-7 June 2020, 2020 [C].
[5] HARIKIRSHNAN K, NAGAVIGNESHWAR R, VIGNESH R, et al. Intelligent Online Shopping using ML-based Product Comparison Engine; proceedings of the 2023 International Conference on Inventive Computation Technologies (ICICT), F 26-28 April 2023, 2023 [C].
[6] 平奥琦. 安卓端跨平台商品多维度比价系统 [D], 陕西：长安大学，2022.
[7] 何毅平, 黄媛, 湛茂溪, et al. 基于网络爬虫的招聘信息可视化系统设计与实现 [J]. 长江工程职业技术学院学报, 2023, 40(03): 24-8.
[8] ALEXANDRESCU A. A distributed framework for information retrieval, processing and presentation of data; proceedings of the 22nd International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, ROMANIA, F Oct 10-12, 2018 [C]. 2018.
[9] 王龙霄, 李健, 沈丽民. 基于增量式爬虫技术的新闻分析系统设计 [J]. 现代计算机, 2023, 29(09): 117-20.
[10] 赵文杰, 古荣龙. 基于Python的网络爬虫技术 [J]. 河北农机, 2020, (8): 65-6.
[11] ZHANG L, LI J, FENG D, et al. Design And Implementation of Web Crawler Based on 'Internet +'Data Automatic Extraction; proceedings of the 2023 3rd International Conference on Consumer Electronics and Computer Engineering (ICCECE), F 6-8 Jan. 2023, 2023 [C].
[12] 孙立伟, 何国辉, 吴礼发. 网络爬虫技术的研究 [J]. 电脑知识与技术, 2010, 6(15): 4112-5.
[13] BISHT V, CHOYAL R, NEGI A S, et al. Utilizing Python for Web Scraping and Incremental Data Extraction; proceedings of the 2023 2nd International Conference on Automation, Computing and Renewable Systems (ICACRS), F 11-13 Dec. 2023, 2023 [C].
[14] CHO J, GARCIA-MOLINA H. The Evolution of the Web and Implications for an Incremental Crawler [Z]. Proceedings of the 26th International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc. 2000: 200 - 9
[15] GAO L, MENG Q. Design of Crawler and Visual Interactive Interface Based on Scrapy Framework; proceedings of the 2023 IEEE 3rd International Conference on Electronic Technology, Communication and Information (ICETCI), F 26-28 May 2023, 2023 [C].
[16] 郭向民, 袁许龙, 朱洛凌. 基于Scrapy和Elasticsesarch的网站敏感词检测系统 [J]. 网络空间安全, 2024, 15(01): 70-5.

[17] WANG Y, ZHANG D, YUAN Y, et al. Improvement of TF-IDF Algorithm Based on Knowledge Graph; proceedings of the 2018 IEEE 16th International Conference on Software Engineering Research, Management and Applications (SERA), F 13-15 June 2018, 2018 [C].
[18] 王宇哲. 基于内容的电影推荐算法研究 [J]. 信息系统工程, 2023, (12): 117-20.
[19] 庄旭菲, 田雪. 基于Scrapy和Elasticsearch的校园网搜索引擎的研究与实现 [J]. 科技资讯, 2019, 17(29): 12-5.
[20] SHI Z J, SHI M Y, LIN W G. The Implementation of Crawling News Page Based On Incremental Web Crawler; proceedings of the 4th Int Conf on Applied Computing and Information Technology / 3rd Int Conf on Computational Science/Intelligence and Applied Informatics / 1st Int Conf on Big Data, Cloud Computing, Data Science and Engineering (ACIT-CSII-BCD), Univ Nevada, Las Vegas, NV, F Dec 12-14, 2016 [C]. 2016.
[21] 邱磊. 基于Web的比价系统的研究与实现 [D], 浙江：复旦大学，2012.
[22] 张恩伟, 胡凯, 卓俊杰, et al. 基于预训练的谷歌搜索结果判定 [J]. 中文信息学报, 2024, 38(03): 102-12.

致　谢

在完成本篇论文的过程中，我要衷心感谢给予我关怀和支持的家人、导师以及朋友们。

首先，我要深深感谢我的导师汤世平老师。在整个研究过程中，他给予了我无私的指导和支持。他丰富的知识、独到的见解以及耐心的引导让我受益匪浅。我由衷感激导师对我的信任和精心的培养，让我得以完成这篇论文的撰写。

接下来，我要特别感谢我的家人。感谢他们支持我前往中国上大学，并在过去四年中给予我无尽的鼓励与支持，使我能够度过充实而有意义的大学生活。这段时光不仅让我对计算机学科有了更深入的理解，也是我成长过程中至关重要的一部分，因为他们赋予了我独立生活的能力。

我还要感谢我的朋友们。在我整个学术生涯中，他们一直是我最亲密的伙伴，与我分享学习的喜悦，一同探索未知的世界。他们的理解、支持和鼓励给予了我无限的动力和勇气，让我能够坚持到最后。

我希望未来的学术道路上，能够成为那个有能力帮助他人的人。再次向所有支持我的人表示由衷的感谢！

谢谢你们的陪伴与支持！