湖 南 科 技 大 学

# 毕 业 设 计（ 论 文 ）

| | |
|---|---|
| 题　　目 | 基于 ProgrammableWeb 的 Web 服务爬虫系统 |
| 作　　者 | 刘羽鑫 |
| 学　　院 | 计算机科学与工程学院 |
| 专　　业 | 计算机科学与技术 |
| 学　　号 | **1805010305** |
| 指导教师 | 刘建勋 |

二〇二二 年 五 月 二十八 日

# 湖 南 科 技 大 学

# 毕业设计（论文）任务书

计算机科学与工程 院 计算机科学与技术 系（教研室）

系（教研室）主任:_____（签名）____年___月___日

**学生姓名: 刘羽鑫 学号: 1805010305 专业: 计算机科学与技术**

1 设计（论文）题目及专题： 基于 ProgrammableWeb 的 Web 服务爬虫系统

2 学生设计（论文）时间：自 2022 年 1 月 17 日开始至 2022 年 5 月 28 日止

3 设计（论文）所用资源和参考资料：

[1]詹恒飞，杨岳湘，方宏. Nutch 分布式网络爬虫研究与优化[J]. 计算机科学与探索，2011(1).

[2]李纪欣，王康，周立发等. Google Protobuf 在 Linux Socket 通讯中的应用[J]. 电脑开发与应用，2013(4).

[3]彭轲，廖闻剑. 基于浏览器服务的网络爬虫[J]. 2009 年 04 期.

4 设计（论文）应完成的主要内容：

1）学习题目项目论文，了解其相关内容

2）学习前端 Vue 框架的使用并完成前端页面

3）了解学习 SpringBoot 架构

4）学习 Java 的基础知识和 Spring 框架的使用，实现后端数据接口

5）调试接口，将数据在前端展示，完善页面功能

5 提交设计（论文）形式（设计说明与图纸或论文等）及要求：

1）毕业论文一份

2）程序源代码一份

3）毕业答辩 PPT 一份

4）毕业设计开题报告一份

5）查重论文一份

6）查重报告一份

6 发题时间： 2022 年 1 月 11 日

指导教师： （签名）

学 生： （签名）

# 湖 南 科 技 大 学

# 毕业设计（论文）指导人评语

[主要对学生毕业设计（论文）的工作态度，研究内容与方法，工作量，文献应用，创新性，实用性，科学性，文本（图纸）规范程度，存在的不足等进行综合评价]

**指导人：** （签名）

年 月 日

---

**指导人评定成绩：** _____

# 湖 南 科 技 大 学

# 毕业设计（论文）评阅人评语

[主要对学生毕业设计（论文）的文本格式、图纸规范程度，工作量，研究内容与方法，实用性与科学性，结论和存在的不足等进行综合评价]

**评阅人：** （签名）

年 月 日

---

**评阅人评定成绩：** _____

# 湖 南 科 技 大 学

# 毕业设计（论文）答辩记录

**日期：**_____

**学生：**_____**学号：**_____**班级：**_____

**题目：**_____

**提交毕业设计（论文）答辩委员会下列材料：**

**1 设计说明书/ 论 文 共 页**

**2 设计（论文）图 纸 共 页**

**3 指导人、评阅人评语 共 页**

**毕业设计（论文）答辩委员会评语：**

[主要对学生毕业设计（论文）的研究思路，设计（论文）质量，文本图纸规范程度和对设计（论文）的介绍，回答问题情况等进行综合评价]

**答辩委员会主任：** （签名）

**委员：** （签名）

（签名）

（签名）

（签名）

---

**答辩成绩：**_____

**总评成绩：**_____

# 摘　　要

在当今互联网快速发展的时代，人们对网络数据越来越依赖，数据科学迅速发展的今天，人们的生活得到了巨大的便利，能不能拥有一种能自动抓取数据的方法，能够摒弃传统的数据抓取方式，在这种需求的催促下，爬虫应运而生，爬虫的出现就是为了能够按照一定的规则快速获取人们所需要的数据。

本系统的主要功能模块是数据展示模块，首先需要使用 Python 爬虫，把 ProammableWeb 网站的所需要的四个模块数据通过 css 选择器和正则表达式等方式筛选出来，然后保存在 Excel 表格，CSV 文件和 MySQL 数据库之中，使用 Vue+SpringBoot 框架实现前后端分离项目，后端接口负责传递数据，前端获取数据后使用 Echarts 技术进行展示，同时项目还提供登录，数据查询，数据导出，根据数据的字段进行排序等功能，用户登录界面后，不仅可以查看各种数据图表，还可以对一些数据进行查询，同时支持把数据导出为 Excel 表格，还可以根据数据不同的字段进行排序。

**关键词**：ProammableWeb；Python 爬虫；MySQL；Vue；SpringBoot

# ABSTRACT

In today's era of rapid development of the Internet， people are increasingly dependent on network data， the rapid development of data science today， people's lives have been greatly facilitated， can there be a way to automatically capture data， can abandon the traditional way of data capture， in the urging of this demand， the crawler was born， the crawler appeared in order to be able to follow certain rules to quickly access the data that people need data.

The main functional module of the system is the data display module， which first requires the use of a Python crawler to filter out the four modules of data needed for the ProammableWeb website by means of css selectors and regular expressions， and then save them in Excel tables， CSV files and MySQL databases， using the Vue+SpringBoot framework. The back-end interface is responsible for passing data， and the front-end uses Echarts technology to display the data after acquiring it. You can also sort the data according to different fields.

KeyWords: ProammableWeb; Python crawler; MySQL; Vue; SpringBoot

# 目　　录

# 第一章　前　言

平时所用到的搜索引擎，如百度，帮助搜索一些大家想要了解的信息，但是通常情况下，这种通用的搜索引擎是有一定的局限性的，搜索的内容太过于广泛，有时候很难从如此多的内容中获取所需要的内容，所以有的时候，需要根据不同的用户的不同需求，设计一款能够辅助人们检索一些特定数据的工具，基于以上的要求，一个能够符合用户要求的，灵活的爬虫程序就有着其无可替代的重要意义。

爬虫的优势在于可以根据不同人的需求定制化抓取数据，自动地分析构造 URL，然后针对这些 URL 进行一些去重的工作，在使用爬虫爬取数据的时候，为了能够加快爬取的效率，一般情况下可能会加入一些多线程的技术，这样可以让爬虫得到更高的抓取能力，同时，在使用爬虫的过程中，可能会受到网络波动，网站本身响应速度的问题，所以需要进行一定的异常处理，避免无限制的一直等待下去。为了得到一个更加直观的视觉效果，需要把爬虫的数据清洗之后存入数据库之中，然后使用对数据进行可视化。

这个爬虫系统的设计的主要目的是用来爬取某个特定的网站的数据，当然，也会考虑一些性能方面的需求，在系统的设计过程中，还会对系统的细节部分进行深入的讨论和研究，争取做到不错过每一个能够想到的细节，尽量满足特定的需求。

## 1.1 开发背景

随着互联网的不断快速地发展，这些年以来，数据直接跟利益挂钩，谁能够在互联网中获取更多有价值的数据，谁获取到的利益就越多，在这种利益的驱使下，许多互联网公司绞尽脑汁只为能够比竞争对手获取更多更有价值的数据。

在对互联网信息进行一些抓取的时候，一般情况是把互联网上很多 Web 页面持久化到指定的存储位置，然后再进一步进行一些处理，网络的迅速发展，让万维网也成为了的千千万万数据获取的地方，但是这么多的数据，如何只获取对人们有用的数据呢，这就成为了人们的一个新的研究方向，网络爬虫就是这个新的研究方向的产物。

网络爬虫其实并没有大家想象中那么复杂，可以把爬虫比喻成爬取网页的蜘蛛或者比喻为机器人也可以，类似于一种替代人干活的机器人，就是根据自己定义的一些规则，不用手动请求，自动请求万维网网站，然后获取相应数据。这个可以说是搜索引擎的必不可少的部分，目前所说的传统爬虫，就是从某个 URL 开始向万维网发送请求，然后根据获取的网页数据，获取新的 URL，把获取到新的 URL 放入队列中等待使用，然后需要设置一些终止条件，等达到了终止条件，程序就会停止工作。但是聚焦网络爬虫要比传统的网络爬虫的工作流程更为复杂一些，程序一开始需要根据一些分析算法过滤到掉与客户需求没有关系的连接，只保存对人们有用的 URL，然后会将其加入队列中，接下来，程序根据一些搜索的策略想办法选择下一次需要被抓取网页的 URL，一直重复上

述的操作，直至满足终止条件。当然，爬虫的作用远不止于此，聚焦网络爬虫会把抓取过的网页存储起来，对数据进行一定的分析，然后进行一些过滤的操作，这是为了方便之后的查询和检索，可以建立相应的索引。对于聚焦爬虫来说，上述过程得到的结果对以后的数据获取工作有很大的指导作用。爬虫要实现三个基本功能：获取网页（收集数据）、解析网页（提取数据）、存储数据。

## 1.2 爬虫的类型

爬虫按照系统结构和实现技术，可以分为以下几类：

通用网络爬虫，通用网络爬虫在搜索引擎中起到了不可或缺的作用，主要作用是给门户网站和大型的 Web 服务提供采集到的数据。而且由于商务原因，他们的技术细节是保密的，一般都只供自己内部参考，这种爬虫无论是其所能爬取的数据数量还是所能爬取的范围都特别特别大，所以，这种类型的爬虫对爬取的速度和存储空间就有着较高的要求，由于系统比较复杂和带刷新的页面太多，所以为了能够提高效率，通常采用并行的方式工作，但是缺点就是因为其复杂性，所以长时间才能刷新一次页面，虽然有一些缺点，但是有很高的应用价值，非常合适使用在搜索引擎。

批量型网络爬虫，这类爬虫的抓取范围和抓取的目标是比较明确的，当爬虫达到一定的终止条件时，程序就会自动停止抓取的过程，当然，抓取哪些具体目标，就可能需要根据实际情况给定，而且有的时候抓取的时间可以设定为不一样。

增量型网络爬虫，相比批量性网络爬虫，这个的不同之处在于，这种类型的爬虫会一直保持对网页的抓取，然后把抓取到的数据，需要进行一些更新的操作。之所以这样做的目的，是因为互联网网页是有时效性的，过了一段时间之后，网络的数据可能就会发生变化，如果不进行更新的话，得到的数据可能就是旧的，利用价值不高的数据，所以为了能够随时掌握数据的最新动态，得到最新的数据，需要持续的对网页进行抓取，这里的增量指的的只会在某些需要的情况下或者发现网页更新的情况下，只抓取那些不同于以往的网页数据，对于没有变化的数据，也不用浪费时间去重新进行抓取，这样的话可以提高抓取的效率，大大减少了数据的下载量，但缺点就是增加了爬虫算法的复杂度还有爬虫的实现难度。

## 1.3 发展趋势

总体来说，从当前的形势来看，随着大数据和人工智能的不断发展，如何获取数据源越来越成为一种潮流，爬虫作为一种重要的，方便的获取数据源的方式，肯定会越来越受到人们的青睐。目前大家所使用的互联网，人们浏览网页时，随便点击几下数据就可以获取如此巨大的数据，随着互联网的不断普及，越来越多的人能够通过互联网浏览网页，在一定时间内获取的数据会变得越来越多，可以毫不夸张地说，互联网信息的抓

取、挖掘和再处理将会成为越来越多人的需要，而要达到这一步，就需要许许多多的爬虫和与其配套的数据清洗工具。所以，至少从目前来看，爬虫技术是一门非常有前景的技术。

## 1.4 小结

本章主要介绍基于 ProgrammableWeb 的 Web 服务爬虫系统设计的开发背景，爬虫的分类，以及爬虫的发展趋势。讨论在互联网和大数据迅速发展的环境下，爬虫系统所占据的重要地位，以及本文设计系统所主要的应用领域，下一章主要介绍系统设计过程中所采用的开发环境以及主要技术简介。

# 第二章　系统开发软件与技术

本系统分为多个部分，包括爬虫程序，前端项目，后端项目，先使用 Python 爬虫技术按照一定的规则抓取表单数据，把抓取的数据存储在 MySQL 数据库中，通过 HTML，CSS，JavaScrIPt，Java 的联合使用，实现一个前后端分离的项目，把数据通过 Echarts 渲染在前端页面，后端负责传递数据，本项目采用的主要框架是 SpringBoot+Vue。

## 2.1 开发环境

**(1) IDEA 简介**

在之前学过使用 EclIPse 编译器，EclIPse 虽然免费，但是很多插件使用起来需要各种配置，如果插件版本不一致，还可能存在版本的冲突问题，使用 EclIPse 启动项目时会偏慢，在我们使用的开发过程中，使用起来不是那么的便利，不适合用于 SrpingBoot 框架的编写，所以后端开发使用了 IDEA，IDEA 是收费的，大学生可以通过学校的企业邮箱免费申请，IDEA 自身集成了许多的插件，同时 IDEA 在 java 开发的领域，可以说是一款很好的开发工具，相比于使用 EclIPse 进行项目开发，IDEA 是业界公认的最好的 java 开发集成环境，IntelliJ IDEA 相对来说更加适合 Spring Boot 框架，编译器中可直接快速生成 Spring Boot 项目，界面不但简洁，而且包括代码的自动生成，代码的编译效率也得到了大大的提升。

**(2) IDEA 优势**

- 方便的 GIT 项目管理；
- 更加高效的开发效率：代码自动填补、对 JavaScrIPt（JS）、Cascading Style Sheets（CSS，层叠样式表单）以及插件更好的支持；
- 更好的调试；

**(3) WebStorm 简介**

WebStorm 适合用于前端开发，有着 Web 前端开发神器的美誉，不需要安装，内置了许多的插件，强大的编辑器可以让开发的效率得到大大的提升。

**(4) WebStorm 优势**

跟 IDEA 类似，代码补全，多个插入符号和选择，代码质量分析，调试、跟踪和测试等，同时，WebStorm 也支持目前许多主流的前端框架，像 Vue，React 等，里面包含了许多种错误的检查，覆盖所有支持的语言。

**(5) Pycharm 简介**

Pycharm 和 IDEA，WebStorm 的功能类似，不同的是适用的语言会有所不同，Pycharm 是适用于进行 Python 语言项目的开发，支持 Django 框架下的专业 Web 开发，同时支持 Google APP Engine。

## 2.2 开发技术

**(1) Python 爬虫简介**

Python 爬虫简而言之，就是一个用 Python 语言编写的自动抓取网页数据的程序，通过 Python 代码发送请求，获取整段 HTML 文本，然后使用 CSS 选择器，正则表达式或者 XPath 的方式精确的筛选出所需要的文本数据。

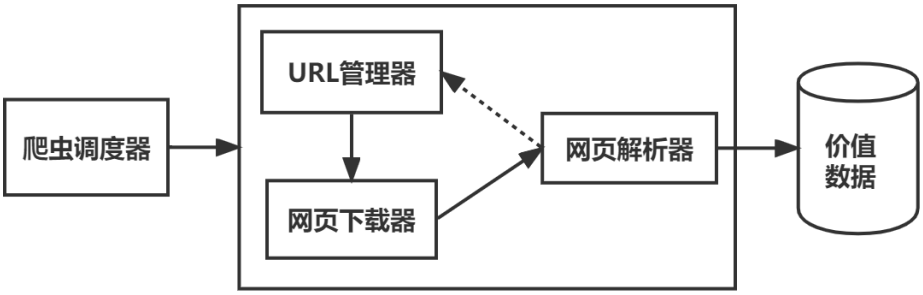


图 2.1 SpringBoot 响应交互图

**(2) SpringBoot 简介**

SpringBoot 简化了 Spring+Mybatis+SpringMVC 开发各种繁琐的配置和复杂的依赖管理，开发人员只需要花时间专注于业务代码，开发人员只需要使用少量的代码就可以很快地创建一个独立的、产品级别的 Spring 应用，而不用浪费大量的时间去整合配置文件，同时也解决了不同框架集成的时候版本不一致的问题。

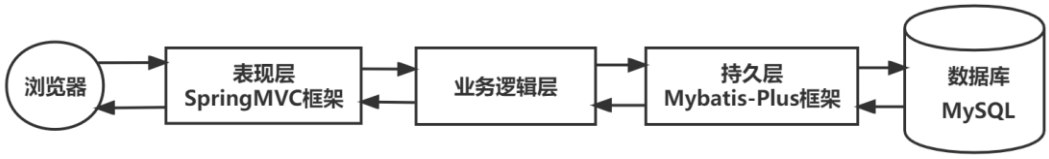


图 2.2 SpringBoot 流程图

**(3) Vue 简介**

Vue 是一个创建单页应用的 JavaScrIPt 前端 Web 框架，关注的重点是 MVC 中的视图层，能够很容易获取数据的更新，然后通过特定的方法实现 View 层和 Model 层的交互，Vue 就是一个已经搭建好的空屋，与单纯使用 JQuery 这种库相比，可以更好地实现代码复用，减少工作量，Vue 做了必须的事，又不会做职责之外的事，同时 Vue 使数据更改更为简单，不需要进行逻辑代码的修改，只需要操作数据就能完成相关操作，Vue 项目更新代码时不用重新启动项目，可以做到自动刷新，能够做到及时相应，是一款非常好用的前端框架。

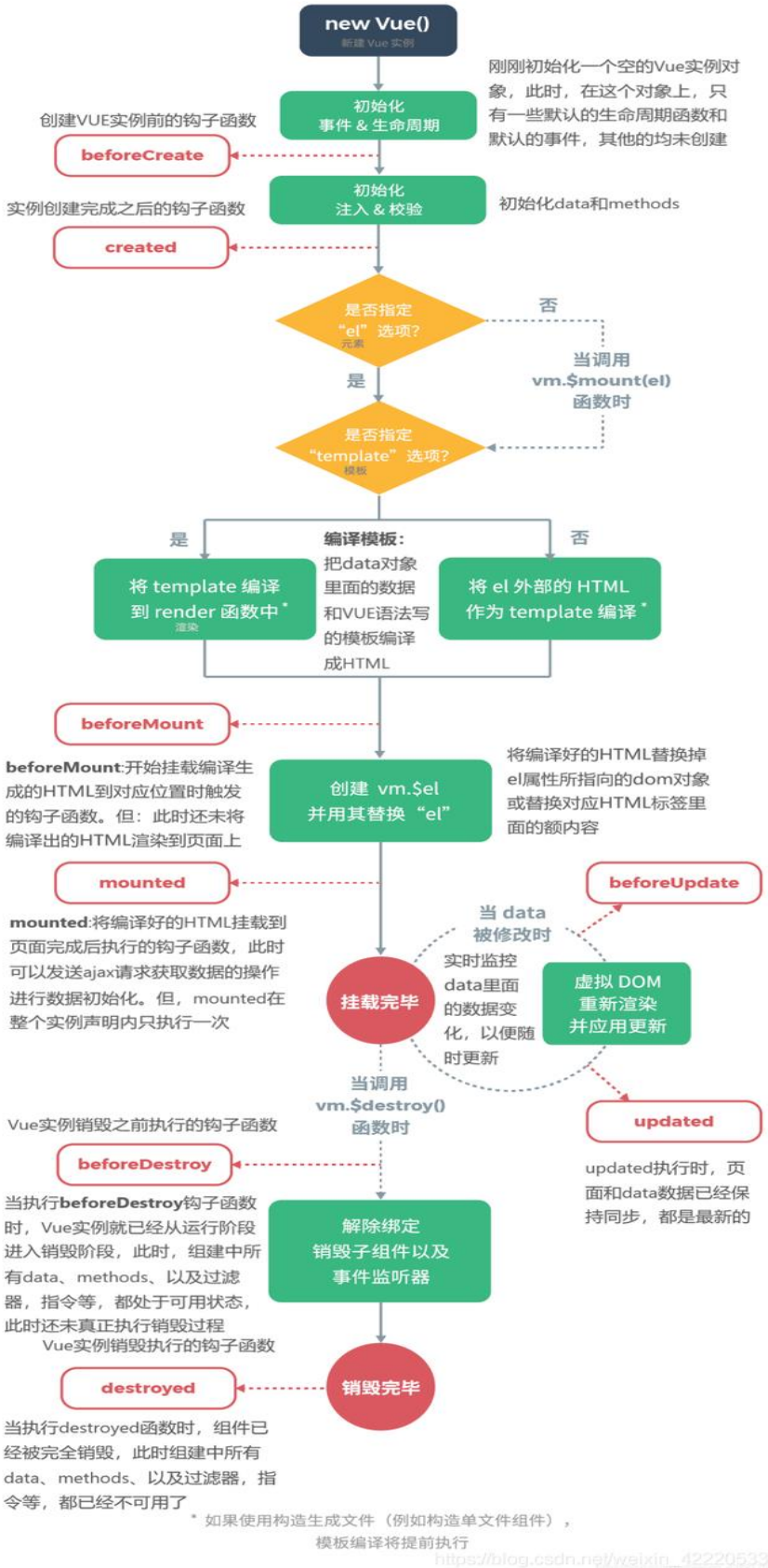


图 2.3 Vue 生命周期图

## 2.3 数据库

使用 MySQL 数据库保存爬虫程序抓取的不同的表单数据，通过 SQLyog 工具更加方便查看和管理 MySQL 的数据，同时也可以省略手写 SQL 语句带来的各种麻烦，SpringBoot 项目的 Dao 层通过 Mybatis-plus 与 MySQL 数据库建立连接，可以根据一定的条件查询数据库表中的数据，把数据封装到对应的实体类中，封装成集合返回给前端。

**(1) MySQL 简介**

MySQL 是用来存储数据的关系型数据库，本质就是一个文件系统，只不过 MySQL 数据库的数据不是存储在一个仓库，而是把不同的数据存储在不同的表格中，通过某些方法可以完成对数据库的 CRUD 操作，相比较于 Oracle 数据库需要花钱购买，MySQL 数据库是免费的，所以使用 MySQL 数据的成本要更低，是许多的中小型网站的理想型选择，MySQL 数据库的技术发展至今，已经相对比较成熟，能够和多种语言实现兼容，不但存储容量大，运行速度快，还支持跨平台。

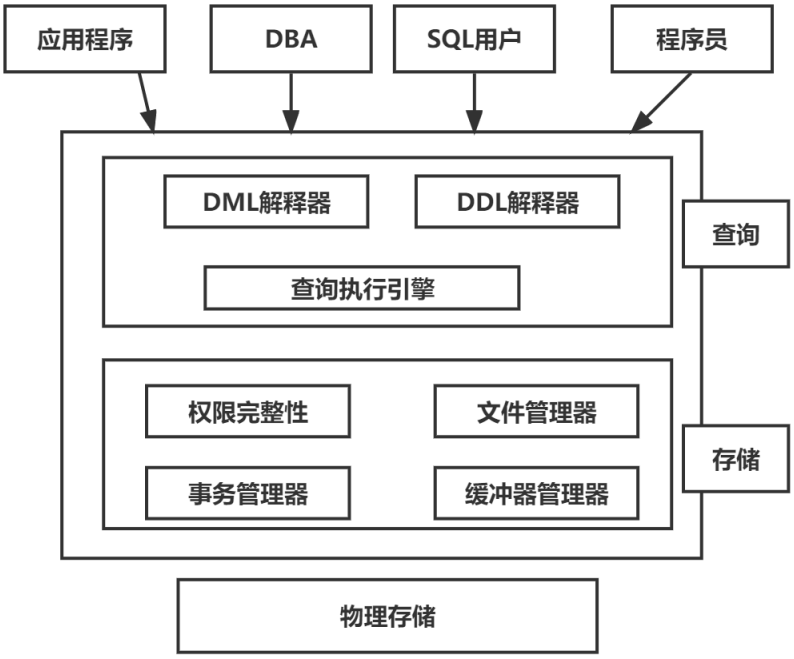


图 2.4 MySQL 原理图

**(2) Mybatis-Plus 简介**

Mybatis 是半自动的 Dao 层框架，因为 Mybatis 需要写大量的注解和 xml 配置文件，用起来不是那么方便，Mybatis-Plus 是对 Mybatis 的加强，可以简化配置，里面已经存在一些封装好的 CRUD 方法，可以直接进行调用，不需要自己重复手写 SQL，减少手写 SQL 语句的麻烦。

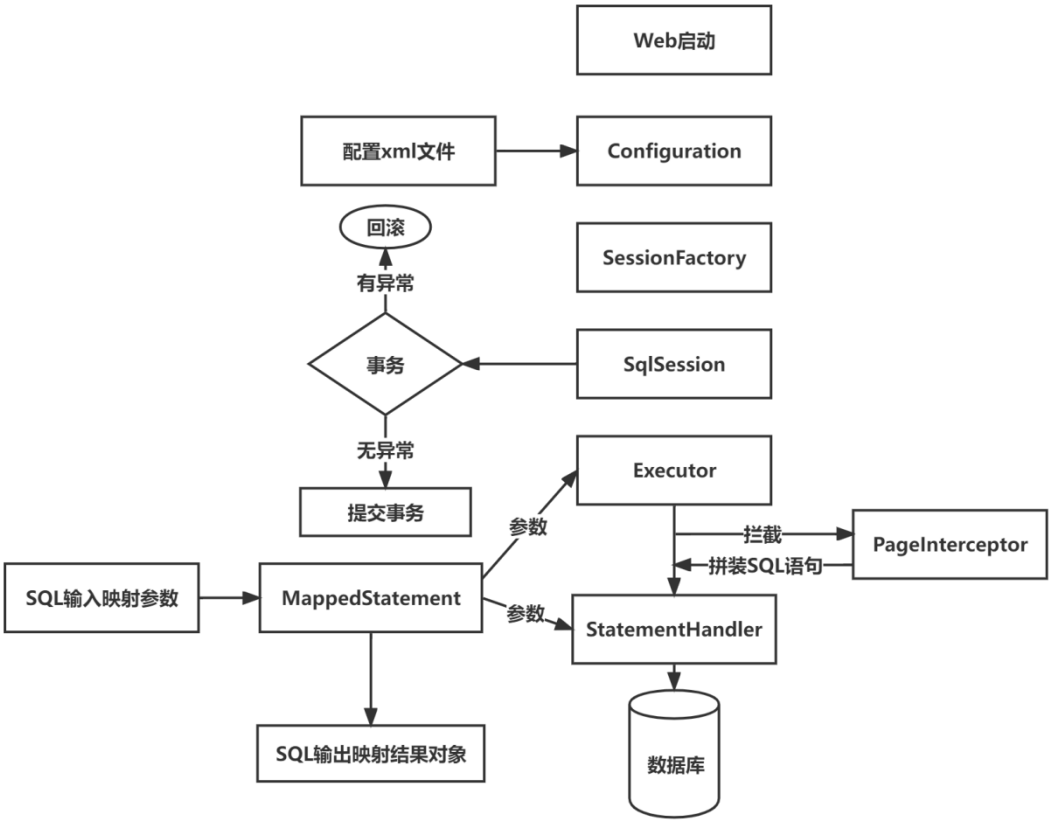


图 2.5 Mybatis-Plus 原理图

## 2.4 其它相关技术

**(1) Redis 缓存技术**

如图 2.6 所示，Redis 缓存主要出于性能和并发角度的考虑，在某些情况，由于数据量多且 SQL 语句执行比较久，且结果基本上没有变动，这些情况下，为了更快地得到数据同时也为了避免频繁访问数据库，就需要使用到 Redis 缓存，把 SQL 语句执行的结果放在 Redis 缓存中，后面再次请求就去缓存中读取数据，使得请求能够得到快速的相应。

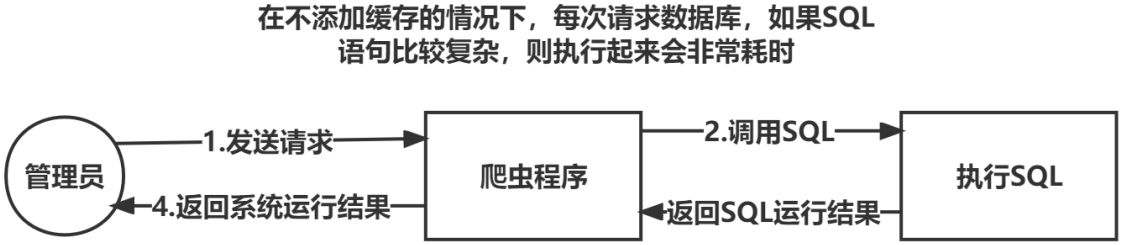


图 2.6 不加缓存的数据获取流程图

**(2) IP 代理技术**

如图 2.7 所示，IP 代理技术就是通过代理的方式帮助用户获取信息，用户访问网站时用的就不是自己主机的 IP 地址，而是代理服务器的 IP 地址，代理服务器把访问的信息返回给主机，简而言之，这就像中间人一样，负责信息的中转。


图 2.7 IP 代理的原理图

**(3) Echarts**

Echarts 技术主要是把数据通过各种图表的形式进行展示，如柱形图，饼形图，关系图等，这样是为了更好地向用户展示各种数据信息，给用户一个更加简洁，直观的感受，使用户一目了然。

**(4) ElementUI**

ElementUI 技术一套自带样式的组件，平时使用 HTML 这些标签，都是没有自带样式的，平时程序员进行前端开发时需要花费大量的时间去调整标签样式，有了 ElementUI 技术，里面有大量的已经调整好的样式组件，可以直接使用，可以省略大量地调整样式的时间，提高开发的效率。

**(5) 分页技术**

项目中使用的是 Mybatis-Plus 分页技术，这种技术主要的原理是通过拦截器拦截 SQL 语句，然后根据分页的参数，对 SQL 进行修改，能够得到一个分页的 SQL 语句，然后再执行。

## 2.5 小结

本章主要介绍系统通过 Python 爬虫获取网页数据，同时通过 MySQL 技术存储 Python 爬虫获取有价值的网页数据，然后介绍了系统设计的开发环境以及一些相关技术简介。主要技术包括前端所使用的 Vue 框架以及后端所使用的 Spring boot 框架。下章将要对系统的具体问题分析进行讨论。

# 第三章　需求分析

如图 3.1 所示，本系统主要分为两个部分，一个部分是通过 Python 语言写的爬虫程序，用于爬取 ProgrammableWeb 网站四个模块的数据，获取数据后将其存储在数据库，第二个部分是通过 java 语言写的前后端分离项目，把数据库的数据渲染在前端的页面，后端的接口负责获取数据库的数据，然后把得到的数据传递给前端。

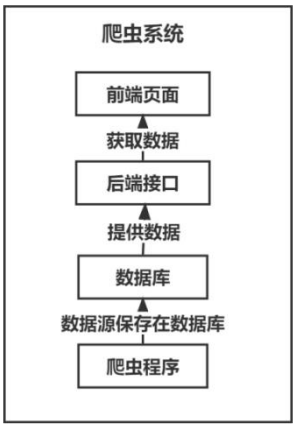


图 3.1 爬虫系统流程图

## 3.1 问题分析

基于 ProgrammableWeb 的 Web 服务爬虫系统主要是得先抓取所需要的数据，如果数据都不能得到，那么后面把数据渲染在前端页面也无从下手，所以，如何按照要求爬取数据是非常重要的。

经过一段时间的 Python 爬虫学习，会发现爬虫无非就是模拟浏览器发送请求，获取服务器响应的 HTML 文本，使用 CSS 选择器，正则表达式或者是 XPath 等方式对文本进行筛选，得到想要的那一部分数据即可，然后可以把数据存储在 Excel 文件，数据库或者 CSV 文件等，这样说起来还挺简单的，但是在 ProgrammableWeb 这个网站，由于这个网站本身响应比较慢加上抓取的数据量比较多，抓取的时间可能长达一周左右，所以需要考虑断网，超时等情况出现之后如何把请求进行重新发送，如果电脑关机，如何从上次关机之前爬取的位置开始爬取，而不是重新从零开始，这样有利于节省效率，还有就是网站本身的一些超链接存在失效的问题，需要把这些失效的超链接排除掉，还要考虑抓取数据的终止条件的问题，还有就是如何通过某个页面的超链接，发送请求得到另外一个页面的 HTML，这个就需要仔细观察网址链接的一些共性，以上所述的问题只是目前考虑的一部分，还有很多很多具体的问题，这些问题都需要使用 Python 代码进行具体的实现，所以，抓取 ProgrammableWeb 这个网站的数据要相比较于一般的小型网站，这个难度要大的多。

获取数据之后，就需要考虑如何把数据在前端的页面进行渲染，首先得仔细观察模块和模块之间，模块里面的数据和数据之间存在的共性和不同之处，需要仔细考虑好到底展示哪些数据，通过什么样的方式展示，这都是需要考虑的问题，数据展示首选当然是 Echarts 方式，通过不同的图表(如词云，条形图，饼状图等)进行展示，可以得到一个相对较好的视觉效果，所以首先需要搭建一个前后端开发的环境，搭建环境最麻烦的就是各种技术之间版本冲突的问题，遇到这种问题，只能考虑技术修改更高的版本或者某些技术使用更低的版本，同时可以使用 ElementUI 等 UI 框架，UI 框架可以省略大量的修改样式的时间，同时也会使界面更加美观，赏心悦目，同时需要考虑的是，如果请求数据时间过长，导致页面加载缓存，这样非常影响客户的整体体验，所以需要在项目中加入缓存技术(如 Redis 缓存)，这样就不用每次都频繁地请求 MySQL 数据库，基于以上问题的分析，将对系统的需求分析在进行进一步的细化，具体的需求分析如下。

## 3.2 数据源获取

需要获取四个模块的表单数据，下面展示获取一个 API 模块数据获取的示例，其余三个模块数据获取的方式跟这个类似，同时数据的类型也是类似的。

### 3.2.1 获取的 API 数据爬取示例

首先，一、进入 ProgrammableWeb 网站的首页，点击 API 模块，二、点击之后进入 API 所有记录的数据的页面，这里就是需要抓取的所有数据，但是这里的数据只显示了部分，所以每次需要详细的数据都需要通过代码先获取进入这条详细数据的超链接，三、得到这条详细数据的超链接，发送获取 HTML 文本，对 HTML 按照一定的规则进行解析，四、进入详细数据页面之后，按照一定的规则使用 Python 代码抓取相应的数据，有些数据还需要再次获取超链接然后进行爬取，再次点击按钮，最终还需得到下图 3.4 和 3.5 的具体数据，至此，只抓取完一条完整数据，五、回到图 3.3 完成第二条数据的抓取，第二条数据的抓取也和第一条类似，总而言之，抓取的顺序是先深度搜索然后再广度搜索。

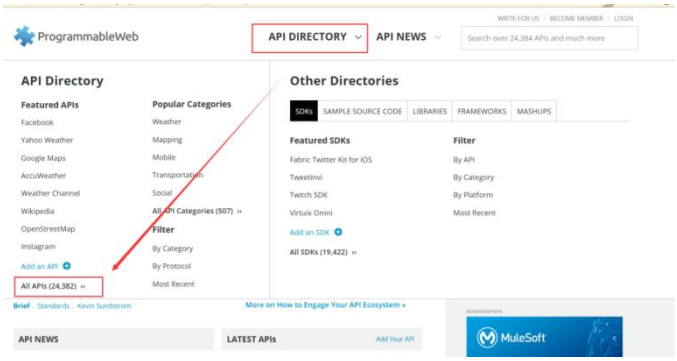


图 3.2 首页界面

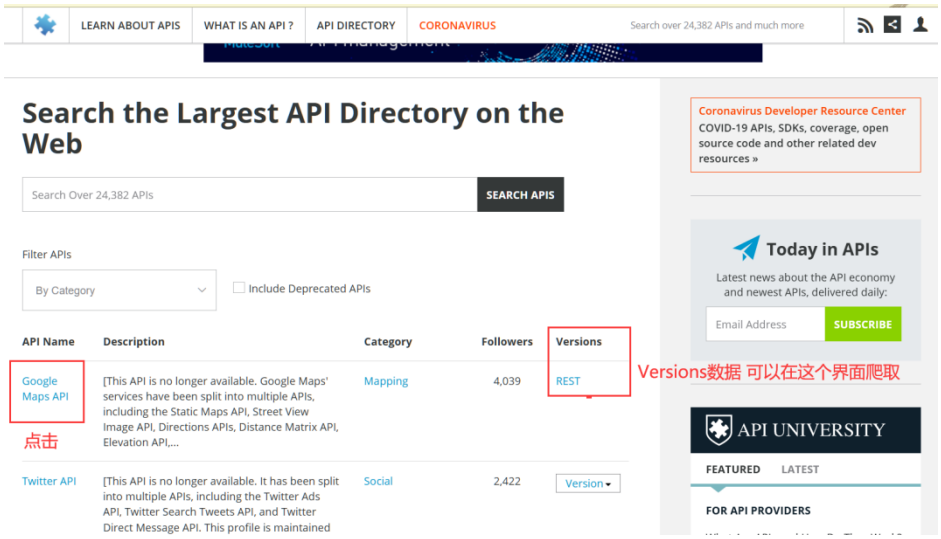


图 3.3 数据界面

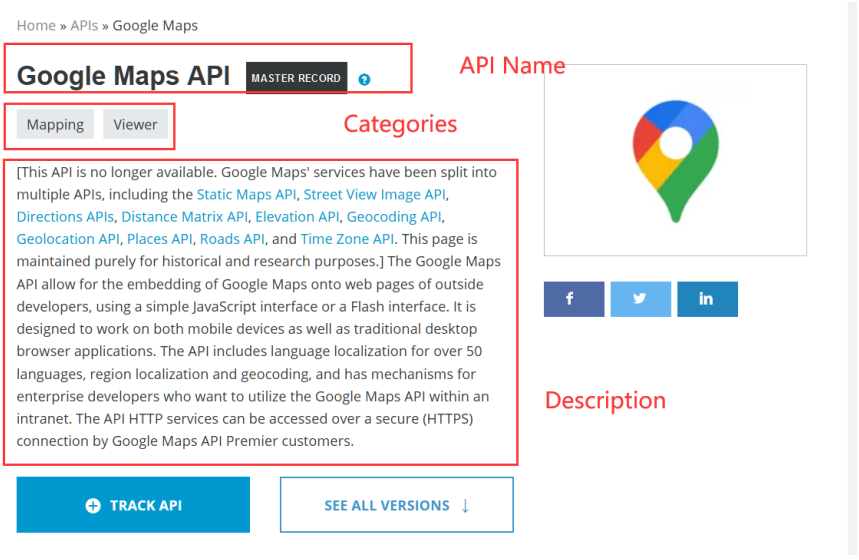


图 3.4 详细数据界面（1）

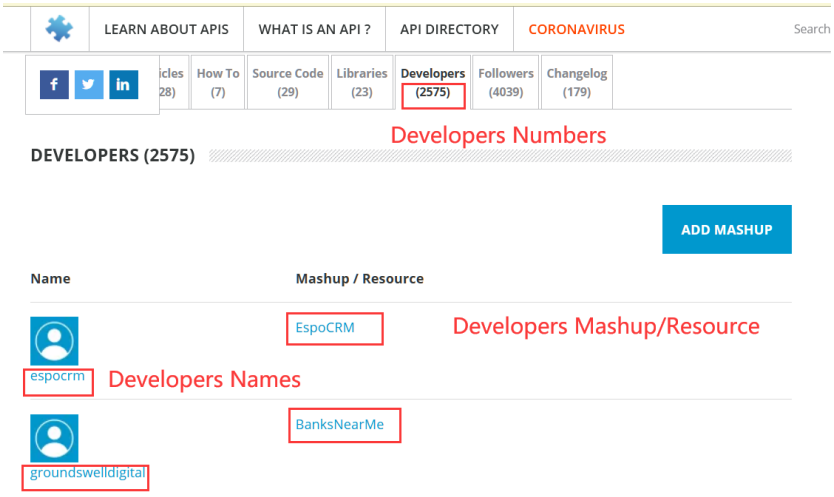


图 3.5 详细数据界面（2）

### 3.2.2 考虑出现的问题以及解决方案

在进行爬虫程序设计的过程中，我们可能会遇到许许多多的问题，以下就是针对可能出现的一些问题提出相应的解决方案：

- 点击更多的时候如何爬取数据，在 followers 或者 developers 中数据太多，可能需要点击更多才能加载更多数据。

解决：在开发过程中发现，点击更多时，其实时把另外一页的数据通过某种方式加载进来，所有只需要获取点击更多按钮 a 标签中的 url，然后获取数据即可。

- 爬虫数据过长，如何存储的问题，一开始使用的是 xlwt 来把数据存储到 excel 中，但是由于一些 API 模块中 followers 和 developers 数据过长，导致存储不了报错。

解决：使用 openpyxl 来存储就不会存在这个问题。

- 如何把数据封装的问题，一开始的想法是用一个字典存储所有的记录条数，然后再一起存到 excel 表格中，但是数据几万条，每条数据又存储很多字段，这样一次性存储虽然好像能减少 io 的次数，但是这样会导致字典的变量数据量过大。

解决：找到一条完整的记录就存储。

- 爬取过程中网络总是中断，爬取过程中，网络中断，然后获取不到 url 对应的 html，导致报错。

解决：增加了重发的机制，可以进行多次的重发，这样即使网络不稳定，也不会让程序报错停下来。

- 爬虫程序停止又得重新开始爬取，每次运行程序都需要从第一个数据开始爬取，这样要是程序中间出现一个问题导致中断，又得重新爬取。

解决：添加一个 field.properties 文件，记录文件爬取的位置，下一次重新运行程序读取配置文件会从记录的位置开始爬取，不用从头开始。

- 爬取到某些超链接直接不让访问。

解决：直接舍弃这种类型的 url，进行下一次循环。

## 3.3 数据渲染

### 3.3.1 数据展示

数据展示页面的主要功能，就是对数据库中的数据进行统计，把统计的接口通过 Echarts 进行展示，主要包括：一、统计每个模块的总记录数，然后使用饼状图进行展示，得到每个模块数据量的占比。二、每个模块的数据都有一个 Categories 字段，这个字段包含了一个或者多个类别，类别之间只是用###字符隔开，然后取第一个类别作为主类别，得到这些数据所属哪些主类别，然后得到主类别数据量的多少，然后根据多少来进行排名，把前几名使用词云图展示出来，每个模块使用的图形样式有所不同，这样可以得到更好的感官效果。三、其余三个模块都有一个 RelatedAPI 字段，这是跟 API 模块进

行关联的，统计其余三个模块的 RelatedAPI 字段，看看这些模块的数据关联哪些 API 更多，然后根据数量多少进行排名，把得到的前几名使用词云展示出来，三个模块使用的词云的样式可以有所不同，这样也是为了更好的页面效果，使页面显得更加丰富，避免过于单调。

### 3.3.2 数据查询和数据

数据查询页面，主要是把每个模块数据分别通过不同表格展示在前端页面，如果一次拿出所有数据，那么数据量是以万为单位的，页面相应可能会相对较慢，而且，页面也展示不了那么多数据，所以，这时候就需要使用分页的功能进行展示，点击第几页就值查询第几页的数据，即显示多少查多少，这样就可以提高数据获取的响应速度，同时，可以提供根据一些字段查询的功能，根据字段，实现模糊查询和数据排序的功能，这样，更加利于用户找到自己想要的数据，最后的最后，可以实现把数据导出为 Excel 的功能，这样便于用户把数据持久化到自己的本地磁盘中。

## 3.4 小结

本章主要对研究问题进行分析并根据问题分析的结果进行了相应的需求分析，在对如何获取数据源以及获取数据源过程中可能出现的一些问题进行了阐述，在下一章中将根据本章的需求分析对具体的系统进行设计。

# 第四章　系统设计

上一章对系统进行了分析，对系统的整体结构建有了一定的概念。本章主要针对系统需求分析进行系统设计，主要包括爬虫程序设计，数据库的设计、类图的设计、系统重点功能流程图设计以及相关接口的设计。下面将对具体的设计进行描述。

## 4.1 爬虫程序设计

### 4.1.1　主要功能

爬虫程序需要爬取四个模块的数据，主要功能包括：

- 发送 URL 请求，ProgrammableWeb 网站服务器响应返回 HTML 文本。
- 对 HTML 文本进行解析，然后按照一定的规则得到想要的字段数据，然后把数据封装在字典里面，最后保存在 Excel 文件，同时为了更加方便在 Pycharm 里面查看数据，把 Excel 文件转换为 CSV 文件。
- 无线网络有时候受一些因素的影响，有时候网络会断开连接，所以在发送 URL 请求后，如果得不到 ProgrammableWeb 网站服务器响应的 HTML 文本，就会尝试进行重发，这就解决了因为网络因素而导致的数据丢失问题。
- 经过不断地发送请求，发现有时候根据 HTML 文本得到的一些 URL 超链接是有问题的，这是网站本身的问题，并不是代码的问题，使用浏览器打开网站，然后访问超链接出问题的地方，同样是访问不了，所以对于这一部分超链接对应的数据，只能进行异常处理，不然代码运行的时候就会报错，异常处理的话就是直接丢弃这个超链接，然后继续访问下一个超链接，这样处理的话可以让程序正常运行下去而不至于导致程序终止的情况发生。
- 由于爬取数据量较多，爬取周期较长，特意增加保存位置信息的功能，通过读取 Properties 文件里面的配置，得到上一次运行爬取的位置信息，根据位置信息爬取接下来的数据。
- IP 代理功能，为了应付反爬机制，所以特意添加 IP 代理功能，这样发送请求时请求 ProgrammableWeb 网站的 IP 地址就不是自己本机的 IP 地址，而是代理的 IP 地址，可以较好的应对反爬的网站。

### 4.1.2　爬虫程序

爬虫程序的整体流程如图 4.1 所示，先读取配置文件，根据配置文件信息进行不同模块的爬取操作，为了避免网络波动，添加了一些重发的机制，同时为了应对网站本身的一些 URL 存在问题，这个程序还添加了一些异常的处理，当 URL 有问题时，就会考虑丢弃这个 URL，对下一个 URL 进行一些相应的操作。

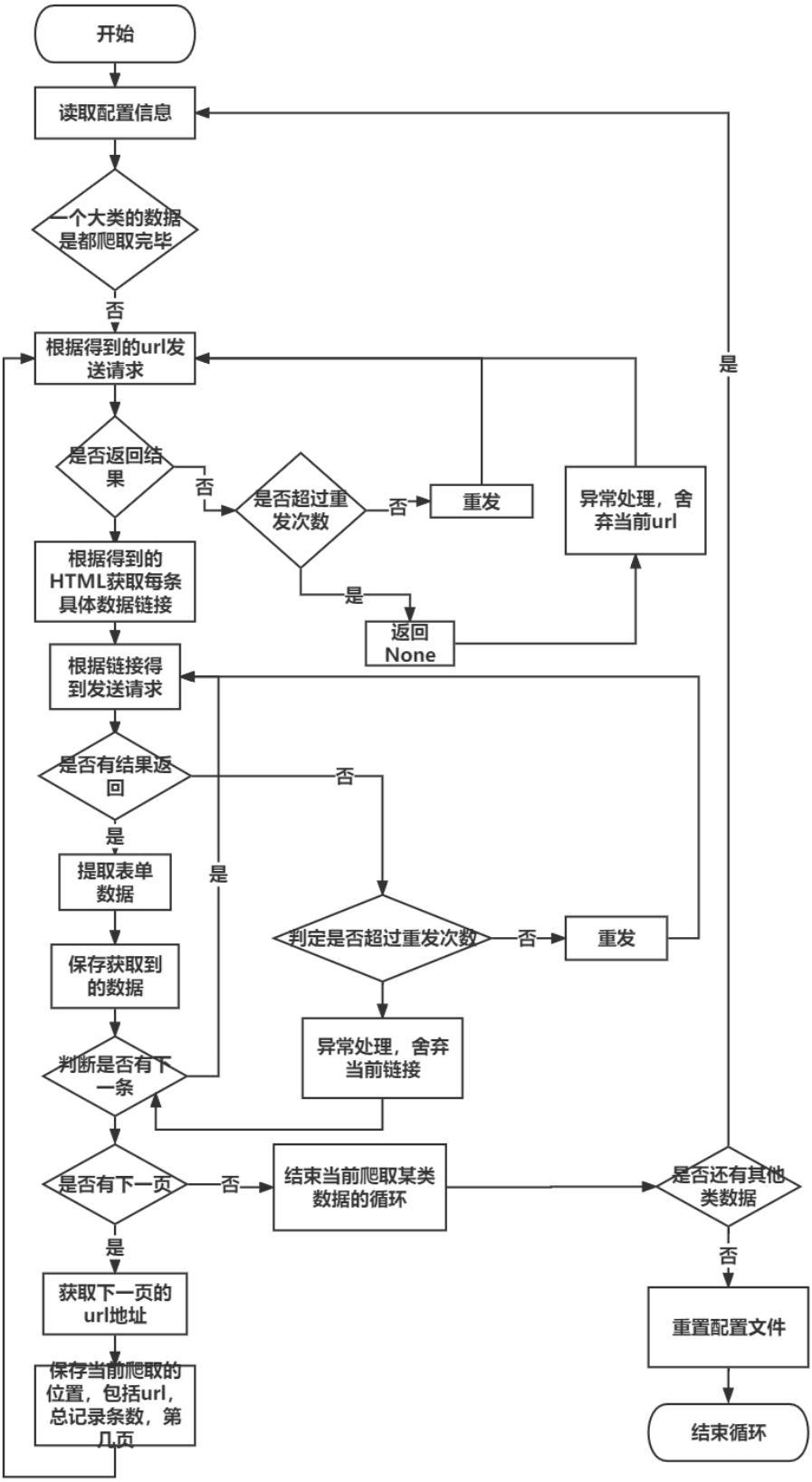


图 4.1 爬虫程序流程示意图

## 4.2 数据库设计

### 4.2.1 实体属性图

本系统主要只有一个管理员和爬虫数据的四个模块，管理员主要是进入系统后，可以查看各种 Echarts 的图表，同时管理员还可以进入数据库查看每个实体的属性。

**(1)管理人员**

管理人员的实体属性图如 4.2 图所示。管理人员的功能比较单一，只需登录以及对其他的实体进行操作即可，因此管理人员的属性主要包括管理人员 id、登录账号、登录密码。

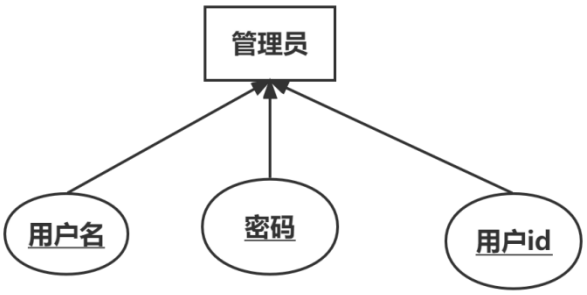


图 4.2 管理员实体属性图

**(2)API 模块**

如图 4.3 所示，API 模块主要包括 api_name，descrIPtion，categories 等 11 个属性，这些属性是根据爬虫程序得到的数据进行设置的，字段都是用字符串的形式来进行存储，有些字段包含着多个数据，这些数据由于都是属于同一个 api_name 字段，所以就保存在一个字段中，然后使用分隔符进行分隔。

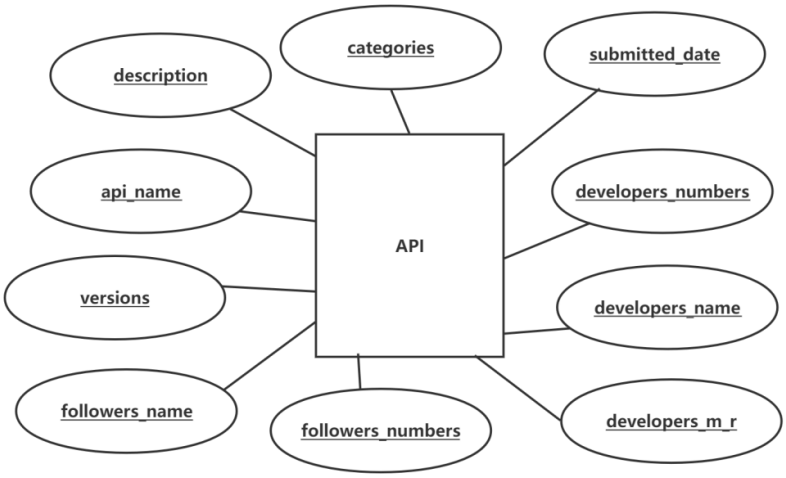


图 4.3 API 实体属性图

**(3)Mashups 模块**

如图 4.4 所示，Mashups 模块主要包括 mashups_name，categories，related_apis 等 9 个属性，这些属性是根据爬虫得到的数据进行设置的，字段都是用字符串的形式来进行存储，有些字段包含着多个数据，这些数据由于都是属于同一个 mashups_name 字段，所以就保存在一个字段中，然后使用分隔符进行分隔。

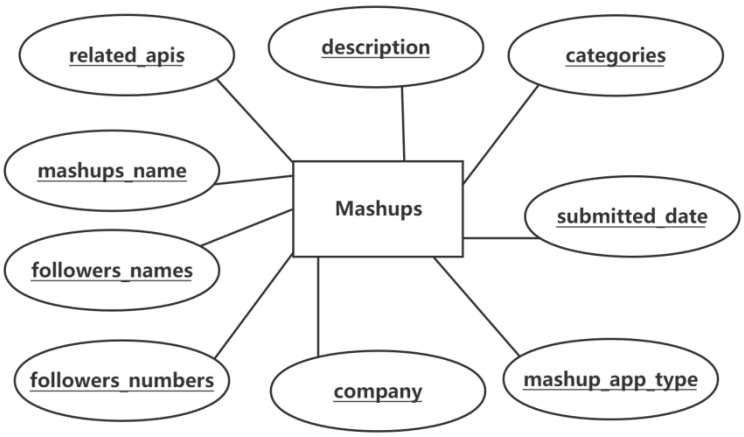


图 4.4 Mashups 实体属性图

**(4)SampleSourceCode 模块**

如图 4.5 所示，SampleSourceCode 模块主要包括 sample_source_code_name，categories，related_apis 等 10 个属性，这些属性是根据爬虫得到的数据进行设置的，字段都是用字符串的形式来进行存储，有些字段包含着多个数据，这些数据由于都是属于同一个 sample_source_code_name 字段，所有就保存在一个字段中，然后使用分隔符进行分隔。

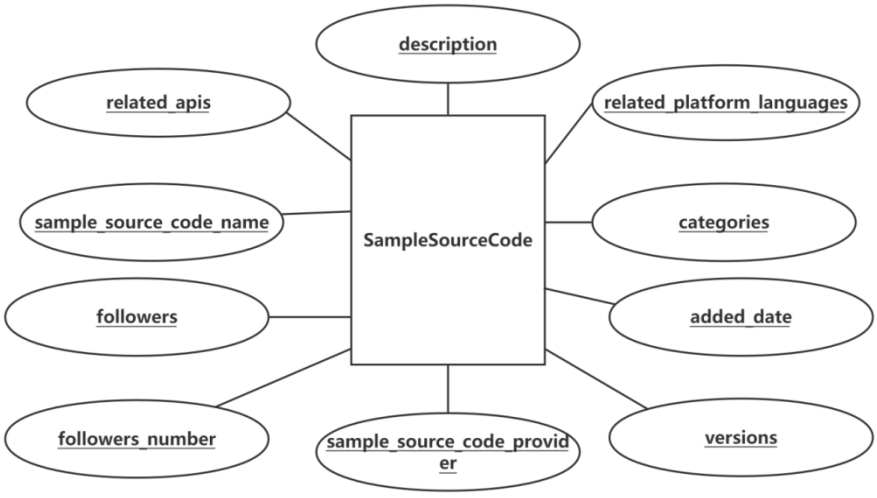


图 4.5 SampleSourceCode 实体属性图

**(5)SDK 模块**

如图 4.6 所示，SDK 模块主要包括 sdks_name，categories，related_apis 等 9 个属性，这些属性是根据爬虫得到的数据进行设置的，字段都是用字符串的形式来进行存储，有些字段包含着多个数据，这些数据由于都是属于同一个 sdks_name 字段，所有就保存在一个字段中，然后使用分隔符进行分隔。

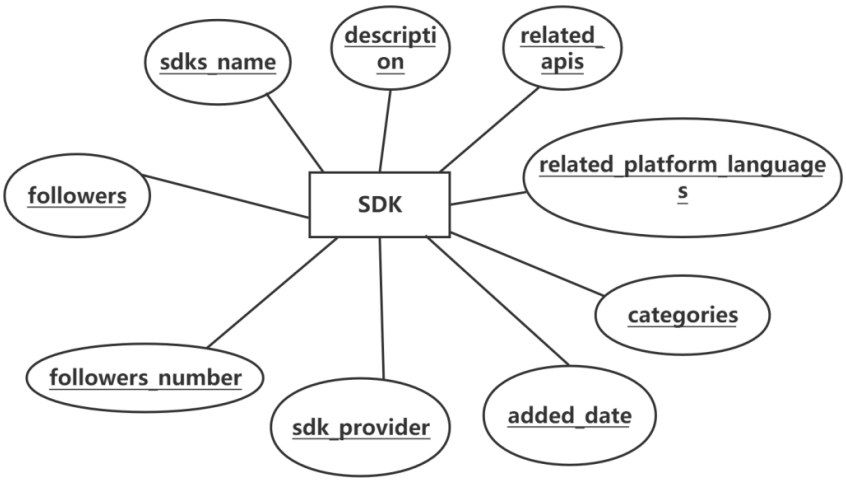


图 4.6 SDK 实体属性图

### 4.2.2 E-R 图

E-R 图数据库中是用来描述实体之间的关联，有一对一，一对多，多对多的关联关系，实体之间具体对应哪一种实体关系需要根据实际情况来定，管理员和 API，Mashups，SampleSourceCode，SDK 之间的联系如图 4.7 所示。

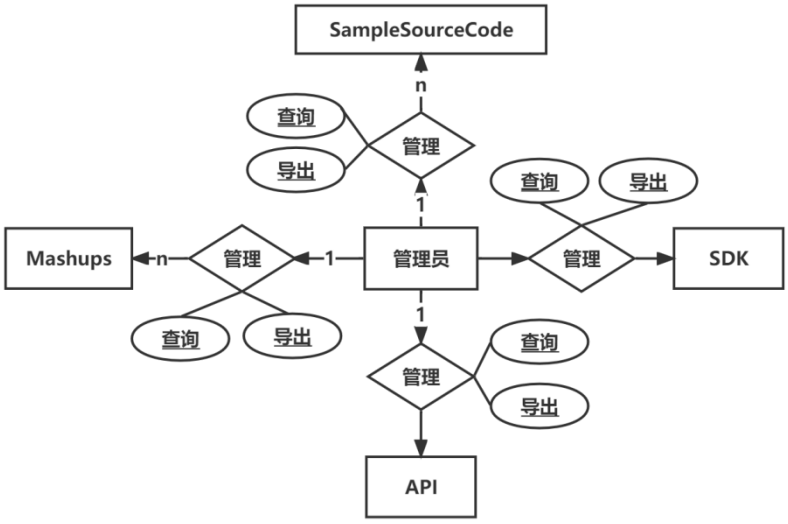


图 4.7 E-R 图

### 4.2.3 数据库表

**(1) 登录管理表**

如表 4.1 所示，这个表主要是为了管理登录的功能，里面包含用户名，用户 id，用户密码，登录时会根据这个表里面存储的数据进行校验，判断是否校验成功。

**表 4.1 Login 表**

| 名称 | 类型 | 长度 | 是否必须有值 | 备注 |
|---|---|---|---|---|
| id | Int | 10 | 不能为空 | 主键，自增 |
| user_name | varchar | 20 | 不能为空 | 用户名 |
| password | varchar | 20 | 不能为空 | 密码 |

**(2) API 数据表**

如表 4.2 所示，这个 API 数据表主要存储的是爬虫程序 API 模块爬取的数据，系统根据 API 数据表里面的数据的特性，然后在前端页面进行不同的展示。

**表 4.2 API 表**

| 名称 | 类型 | 长度 | 是否必须有值 | 备注 |
|---|---|---|---|---|
| API_name | varchar | 255 | 不能为空 | API 名称 |
| descrIPtion | varchar | 255 | 不能为空 | 描述信息 |
| categories | varchar | 255 | 不能为空 | 类别 |
| submitted_date | varchar | 255 | 不能为空 | 提交日期 |
| developers_numbers | varchar | 255 | 不能为空 | Developers 数量 |
| developers_name | varchar | 255 | 不能为空 | Developers 名称 |
| developers_m_r | varchar | 255 | 不能为空 | Follower 的 Mashups 或 Resources |
| followers_numbers | varchar | 255 | 不能为空 | Followers 数量 |
| followers_name | varchar | 255 | 不能为空 | Follower 名称 |
| versions | varchar | 255 | 不能为空 | 版本 |

**(3) Mashups 数据表**

如表 4.3 所示，这个 Mashups 数据表主要存储的是爬虫程序 Mashups 模块爬取的数据，系统根据 Mashups 数据表里面的数据的特性，然后在前端页面进行不同的展示。

**表 4.3 Mashups 表**

| 名称 | 类型 | 长度 | 是否必须有值 | 备注 |
|---|---|---|---|---|
| Mashups_name | varchar | 255 | 不能为空 | 名称 |
| related_API | varchar | 255 | 不能为空 | 关联 API |
| descrIPtion | text | 自定义 | 不能为空 | 描述 |

| | | | | |
|---|---|---|---|---|
| categories | varchar | 255 | 不能为空 | 类别 |
| submitted_date | varchar | 255 | 可以为空 | 提交日期 |
| Mashup_app_type | varchar | 255 | 可以为空 | 类型 |
| company | varchar | 255 | 可以为空 | 公司 |
| followers_numbers | varchar | 255 | 可以为空 | Followers 数量 |
| followers_names | varchar | 255 | 可以为空 | Follower 名称 |

**(4) SampleSourceCode 数据表**

如表 4.4 所示，这个 SampleSourceCode 数据表主要存储的是爬虫程序 SampleSourceCode 模块爬取的数据，系统根据 SampleSourceCode 数据表里面的数据的特性，然后在前端页面进行不同的展示。

**表 4.4 SampleSourceCode 表**

| 名称 | 类型 | 长度 | 是否必须有值 | 备注 |
|---|---|---|---|---|
| sample_source_code_name | varchar | 255 | 不能为空 | 名称 |
| descrIPtion | text | 255 | 不能为空 | 描述信息 |
| related_API | varchar | 自定义 | 不能为空 | 关联的 API |
| related_platform_languages | varchar | 255 | 不能为空 | 相关平台语言 |
| categories | varchar | 255 | 不能为空 | 类别 |
| added_date | varchar | 255 | 可以为空 | 添加日期 |
| versions | varchar | 255 | 可以为空 | 版本 |
| sample_source_code_provider | varchar | 255 | 可以为空 | 提供者 |
| followers_number | varchar | 255 | 不能为空 | Follower 数量 |
| followers | varchar | 255 | 不能 | Follower 名称 |

**(5) SDK 数据表**

如表 4.5 所示，这个 SDK 数据表主要存储的是爬虫程序 SDK 模块爬取的数据，系统根据 SDK 数据表里面的数据的特性，然后在前端页面进行不同的展示。

**表 4.5 SDK 表**

| 名称 | 类型 | 长度 | 是否必须有值 | 备注 |
|---|---|---|---|---|
| SDK_name | varchar | 255 | 不能为空 | 名称 |
| descrIPtion | text | 255 | 不能为空 | 描述信息 |
| related_API | varchar | 自定义 | 不能为空 | 关联的 API |
| related_platform_languages | varchar | 255 | 不能为空 | 相关平台语言 |
| categories | varchar | 255 | 不能为空 | 类别 |

| | | | | |
|---|---|---|---|---|
| added_date | varchar | 255 | 可以为空 | 添加日期 |
| SDK_provider | varchar | 255 | 可以为空 | 提供者 |
| followers_number | varchar | 255 | 不能为空 | Follower 数量 |
| followers | varchar | 255 | 不能 | Follower 名称 |

## 4.3 系统类图设计

这个系统所包含的类图有很多，主要分为以下这 6 个模块，一、controller 层里面的类主要是用来定义数据的接口，设置接口的传入参数和返回参数，同时设置接口的访问地址。二、service 里面的类主要是为了应对多张表的操作，进行一些业务逻辑的处理，service 使用的目的主要是为了解耦。三、mapper 里面的类主要是为了跟数据库连接，然后执行相应的 SQL 语句得到响应的数据。四、util 里面的类主要是封装的一些工具类，这些类可以是一些公共的代码逻辑，当在很多个类里面都需要用到这个业务逻辑时，就需要把这段业务逻辑的代码抽取出来成功公共的部分，需要用的时候调用这个类封装的方法即可。五、config 里面的类主要用来存在一些自定义的配置，有时候根据需求定义一些配置，当项目启动时，程序能够识别定义的配置，然后进行一些相应的操作，entity 里面的类主要是用来存放一些实体，当连接数据库执行完 SQL 语句后会得到一些数据，会把这些数据封装在这些实体类中，方便能够使用。

由于 API，Mashups，SampleSourceCode，SDK 中的类图都十分地类似，所以就展示一个跟 Mashups 关联的类图，如图 4.8 所示，当前端发送请求给后端时，后端一般会使用 dispatchServlet 来转发给后端 controller 相应的类方法进行处理，controller 会把解析得到的参数传给 service 进行处理，service 会调用 mapper 中的方法来连接 MySQL 数据库，执行相应的 SQL 语句返回执行后的结果，把结果封装成集合返回给 service，service 得到结果的集合后就进行一些业务逻辑的处理，把处理后的数据返回给 controller，controller 再把数据封装在返回结果集中，结果集中除了数据，还有相应的状态码，前端会根据返回的状态码进行不同的操作。

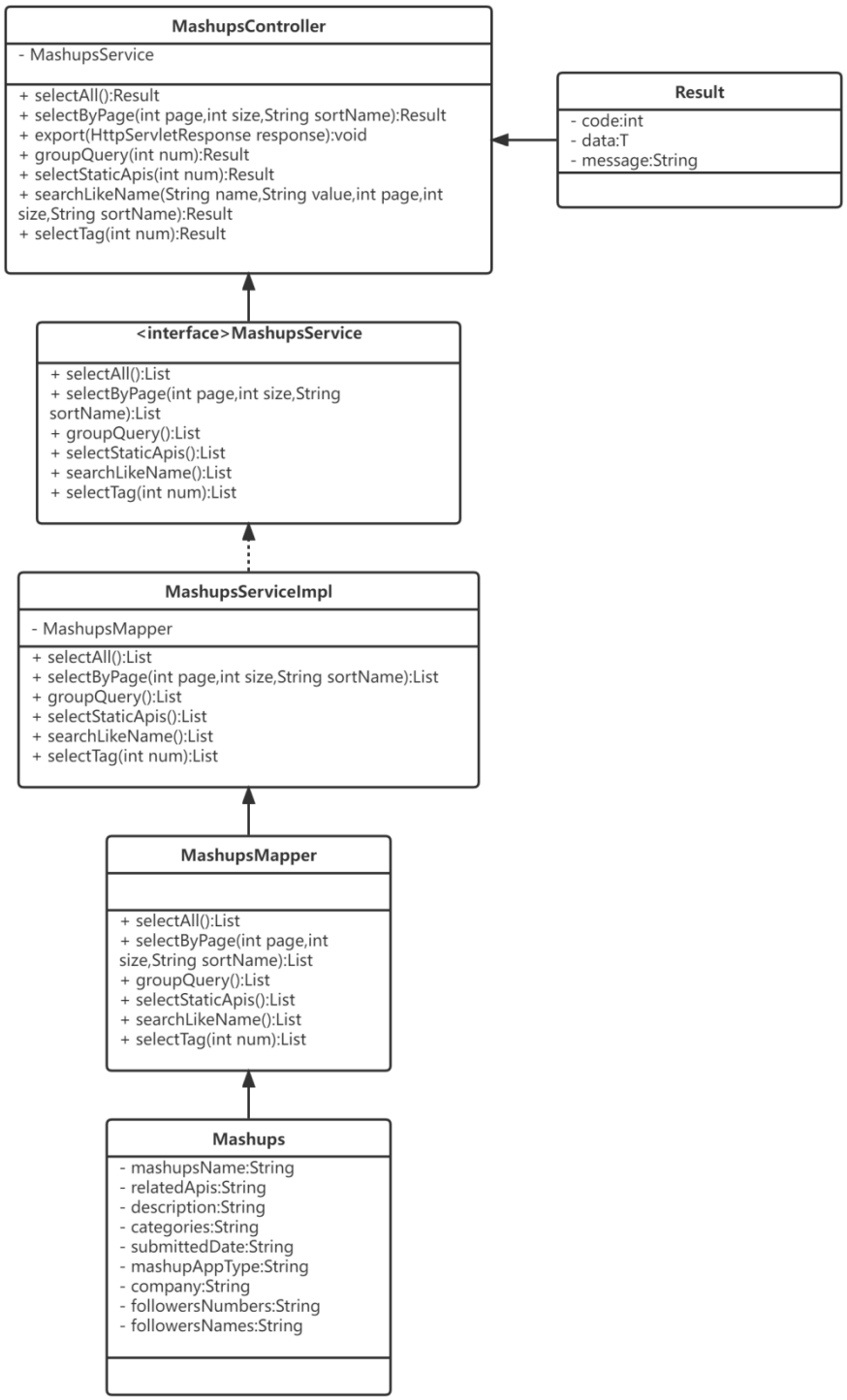


图 4.8 Mashups 类图

## 4.4 系统重点功能流程设计

这个系统指的的是使用 java 语言开发的基于 Vue+SpringBoot 框架的前后端分离的数据展示项目，这个项目里面包含登录功能，数据展示功能，数据查询功能，根据字段进行排序等功能。

### 4.4.1 登录

如图 4.9 所示，登录的时候有一个自动登录的功能，如果之前登录过，然后有相应的缓存数据的话，就不用重新输入账号和密码，会自动跳转到主界面，这是通过 Vue 中的 Vuex 实现的，每次输入账号密码如果验证通过，就会根据账号和密码生成一个随机的 token，如果下次再次登录时，程序就会检查 Vuex 中是否存在这个 token，如果存在，就会自动登录，然后自动跳转到主界面，如果不存在，就跳转到输入账号和密码的界面，接着会把输入的账号和密码与正确的账号的密码进行比对，如果比对不成功，则会返回账号或者密码错误的提示，这个时候就需要重新输入账号和密码，如果输入正确，就生成 token 然后保存在 Vuex 中，然后就自动跳转到主界面，到了主界面就可以进行一些相应的操作了。

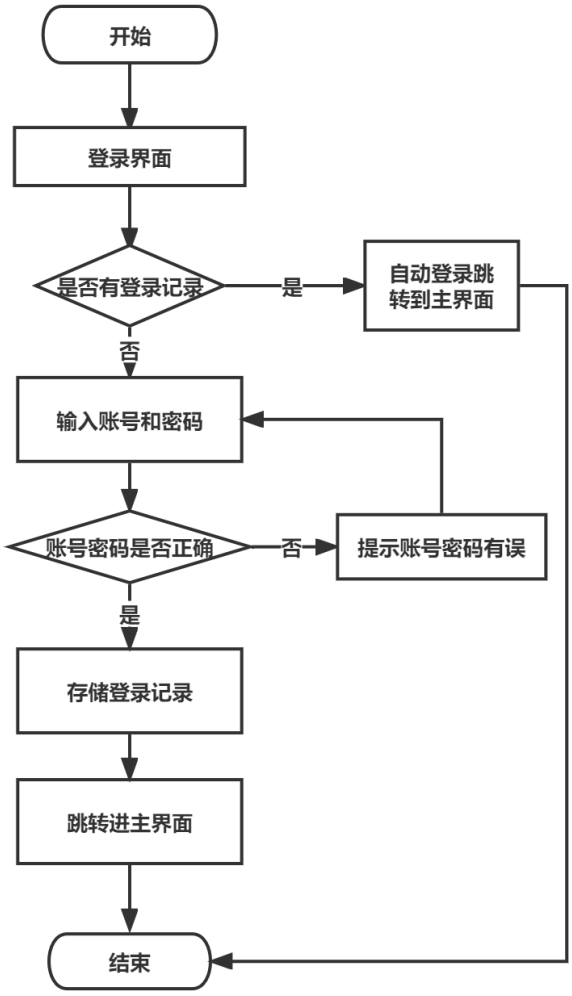


图 4.9 登录流程图

### 4.4.2 数据展示模块

数据展示模块又分为很多分模块，包括总记录数模块，API 数据图表展示模块，Mashups 数据图表展示模块，SampleSourceCode 数据图表展示模和 SDK 数据图表展示模块，这几个分模块数据图表的展示流程是是类似，如图 4.10 所示，都是前端先通过 Axios 发送请求给后端，后端根据对应的 url 选择相应的接口来进行处理，然后接口代码去查询 Redis 缓存中是否有所需要的数据，如果有的话，则把 Redis 缓存数据封装返回给前端，如果 Redis 缓存没有，那么就需要连接数据库，根据对应的 SQL 语句查询数据库中的数据，把数据返回给后端，后端再把数据封装返回给前端，前端把数据进行解析，然后传递给 Echarts 图表，Echarts 根据不同的数据显示不同的图表，可以根据数据的不同，使用不同的 Echarts 图表，值得一提的是，该模块的排行榜数据个数可以动态根据用户的需求进行动态改变，样式也会随之改变，有的排行榜默认展示前五名，但是可以根据用户需求改变排行榜的数量。

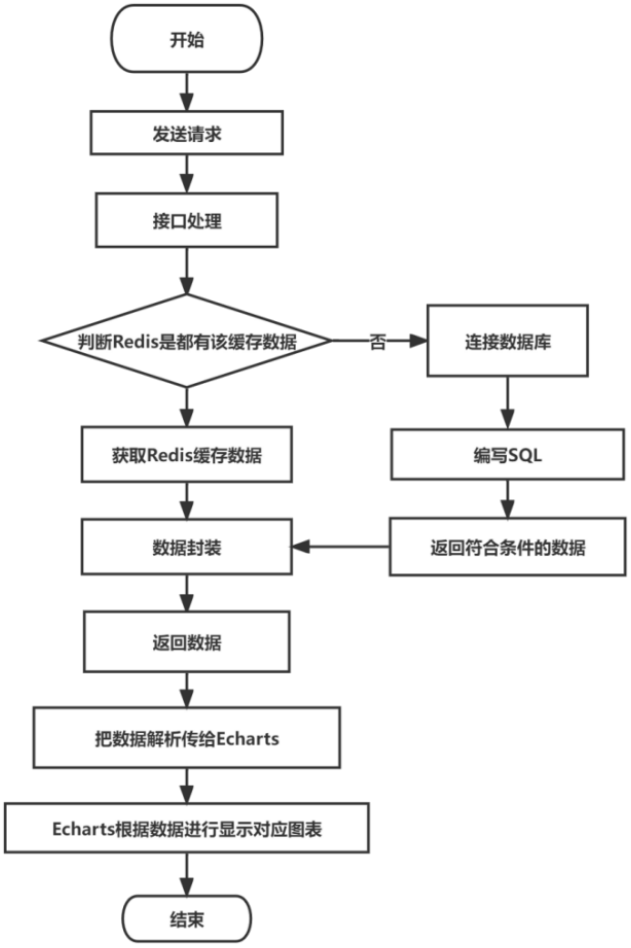


图 4.10 数据展示流程图

### 4.4.3 数据查询模块

如图 4.11 所示，数据查询模块也为四个分模块，分为 API 数据查询模块，Mashups 数据查询模块，SampleSourceCode 数据查询模块，SDK 数据查询模块，这几个模块功能也是类似的，只是查询的数据有所不同，模块中主要有三个功能：分页查询功能，数据导出为 Excel 的功能，还有选择字段，然后根据字段的值来进行模糊分页查询的功能，分页功能，顾名思义，是根据点击的页数和每一页查询的数据条数向后端发送请求，然后后端选择相应的接口进行处理，处理的时候先查看 Redis 缓存中是否有相应的数据，如果没有，则再根据 SQL 语句连接 MySQL 数据库请求相应的数据，后端收到数据库的数据，然后把数据进行封装，返回给前端，如果 Redis 缓存存在相应的数据，则从缓存中把数据封装返回给前端，前端把获取的数据进行解析然后显示在页面上，数据导出功能，根据每个模块的不同，向后端发送不同的请求，然后把得到的数据导出为 Excel，支持自定义文件名称，最后一个根据输入的名称进行模糊查询，模糊查询就是利用数据的一部分信息来查询数据的一种方式，在查询某些数据时，不需要输入某个字段的全部，只需要输入部分，就能查询相应的数据。

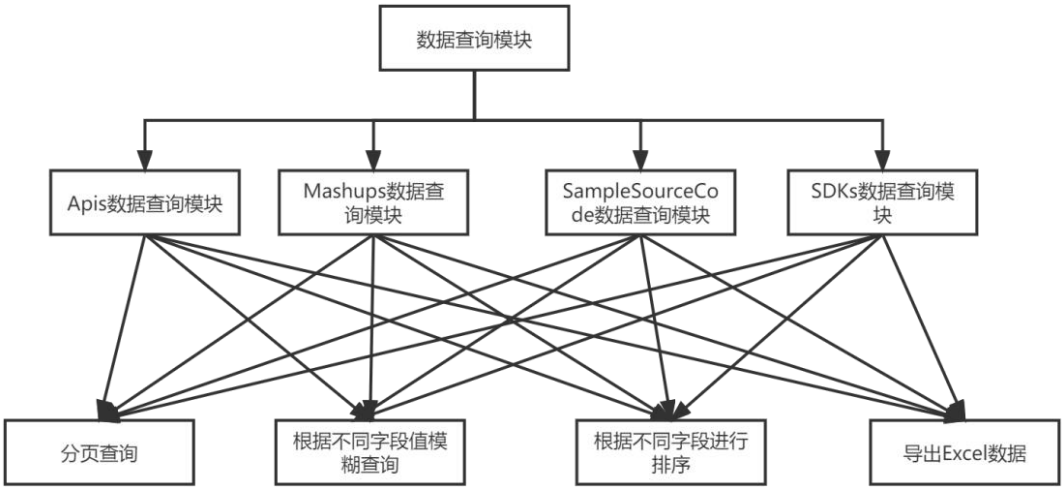


图 4.11 数据查询功能结构图

## 4.5 系统接口设计

### 4.5.1 登录模块

**(1) 登录接口**

如表 4.6 所示，下表为登录接口，输入账号名和密码，然后跟 MySQL 数据库的账号和密码进行比对，如果一致，则返回成功状态码 200 和登录成功的信息，如果不一致，则返回失败的状态码 400，并返回登录失败的信息。

**表 4.6 登录接口**

| url | /passowrdLogin |
|---|---|
| 参数 | username，password |
| 返回 | 状态码和校验新信息 |
| 使用结果 | 管理人员登录成功 |

**(2) 更新密码接口**

如表 4.7 所示，下表为密码更新的接口，当需要更改密码的时候，可以使用这个接口进行密码的更改，传入用户名和密码，会根据用户名更改新的密码，如果找不到这个对应的用户名，则更改密码的操作不成功，如果能够找到相对应的用户名，然后使用更新语句对密码进行更新。

**表 4.7 密码更新接口**

| url | /updatePassowrd |
|---|---|
| 参数 | username，password |
| 返回 | 状态码和校验新信息 |
| 使用结果 | 管理人员登录成功 |

### 4.5.2 通用模块

**(1) 各个模块总记录数获取接口**

如表 4.8 所示，这个接口的主要作用是用来统计每个模块的总记录数，然后把几个模块的总记录数封装成一个在结果集之中，前端根据状态码判断数据是否成功。

**表 4.8 各个模块总记录数获取接口**

| url | /common/getCount |
|---|---|
| 参数 | null |
| 返回 | 状态码和结果集 |
| 使用结果 | 得到各个模块的总记录条数 |

**(2) 爬虫程序目录接口**

如表 4.9 所示，这个接口用来打开爬虫程序所在的目录，目录里面是已经打包好的用 Python 代码书写的程序，用户直接点击 main.exe，就可以直接运行爬虫程序，从而达到更新爬虫数据的目的。

**表 4.9 爬虫程序目录接口**

| url | /common/openDir |
|---|---|
| 参数 | null |
| 返回 | 状态码和结果集 |
| 使用结果 | 打开爬虫程序所在的目录 |

### 4.5.3 爬虫数据模块

**(1) 查询 API 模块所有数据接口**

如表 4.10 所示，这个接口主要是用来获取 API 模块中所有的数据，这个接口可以跟导出数据的接口配合使用，实现把数据导出为 Excel 的目的，当需要 API 模块的所有数据时，可以使用这个接口，先从数据库查询到所有的数据，再把数据封装为 List 集合的形式,然后再把这个 List 集合中的数据全部传递给到位 Excel 表格的接口,简而言之，这个接口的目的是为了查询所有的数据，但是配合可以配合其他的一些接口进行使用。

**表 4.10 查询 API 模块所有数据接口**

| url | /API/selectAll |
|---|---|
| 参数 | null |
| 返回 | 状态码和结果集 |
| 使用结果 | 查询 API 模块所有数据 |

**(2) 根据页数，页面显示的记录数以及排序的字段进行分页查询**

如表 4.11 所示，这个接口是用来实现分页查询，page 是查询的页数，size 是查询的记录条数，sortName 是根据什么字段进行排序，如果 sortName 为空，则不排序，因为在前端的页面需要使用表格来渲染数据，但是如果一次性把所有数据都拿出，会影响数据渲染的功能，而且一次性也显示不了那么多条数据，所以使用分页查询的功能，让前端的数据得到快速渲染的同时，也不影响用户查看数据。

**表 4.11 分页查询接口**

| url | /API/selectByPage |
|---|---|
| 参数 | page，size，sortName |
| 返回 | 状态码和结果集 |
| 使用结果 | 得到分页数据 |

**(3) 导出数据**

如表 4.12 所示，导出数据，顾名思义就是把某个模块的数据全部查询出来，然后把数据导出为 Excel 表格，为用户提供数据。

**表 4.12 导出数据接口**

| url | /API/export |
|---|---|
| 参数 | null |
| 返回 | 状态码和结果集 |
| 使用结果 | 导出 Excel 数据 |

**(4) 主类别排行榜查询**

如表 4.13 所示，num 代表数量，表示获取前多少名的数据，可以根据不同的需求获取排行榜数据的数量，每条数据都有一个 Categories 字段，Categories 字段存在一个或

者多个类别，使用###分隔，该接口会调用 SQL 语句，取出每个 Categories 字段的第一个类别作为主类别，然后统计每个主类别都有多少条数据，根据数量的不同，进行排序，得到数量最多的 num 个主类别。

**表 4.13　主类别排行榜接口**

| url | /API/groupQuery |
|---|---|
| 参数 | num |
| 返回 | 状态码和结果集 |
| 使用结果 | 得到排行榜结果数据 |

**(5) 分页模糊查询**

如表 4.14 所示，name 是表示根据哪个字段来进行模糊查询，value 表示根据哪个值进行模糊查询，page 表示得到查询结果的第几页，siez 表示得到查询结果的几条记录数，sortName 表示按照哪个字段排序，这个接口可以实现多个页面效果，不仅实现了根据字段进行模糊查询，而且还有分页的功能，同时也可以实现的数据排序的功能。

**表 4.14 分页模糊查询接口**

| url | /API/searchLikeName |
|---|---|
| 参数 | name，value，page，size，sortName |
| 返回 | 状态码和结果集 |
| 使用结果 | 得到分页模糊查询的结果 |

**(6) 统计关联 API 的排行**

如表 4.15 所示，这个爬虫数据模块以上的接口只展示了 API 的接口，但是 Mashups，SampleSource，SDK 都是类似的接口，而且实现的功能也是类似的，所以下面就不再一一展示，下面要展示的这个接口是 API 没有的，但是其他三个模块都有的一个接口，以 Mashups 为例，num 也是代表数量，表示需要得到多少个排行数据，这个也是得到的一个排行榜的数据，但是统计的字段不同，这个统计的是 Related_API，三个模块(Mashups，SampleSourceCode，SDK)跟 API 模块都有关联，关联的字段就是 Related_API，这个字段可能存在一个或者多个 API，用###分隔，这个接口会调用 SQL 语句把 Related_API 字段的每条数据进行分隔，然后再进行相应的统计，统计出现次数最多的 num 个 API。

**表 4.15 统计关联 API 接口**

| url | /Mashups/selectStaticAPI |
|---|---|
| 参数 | num |
| 返回 | 状态码和结果集 |
| 使用结果 | 得到相关联 API 排行榜数据 |

**(7) 调用爬虫程序更新数据**

如表 4.16 所示，这个接口可以根据用户选择不同的模块，然后定制化地只爬取某一个模块的数据，也可以爬取所有的模块数据，这个是根据用户可以根据自己的选择，程序会根据不同的值对爬虫程序的配置文件进行修改，进而达到分模块爬取数据的要求，如下表 4.16 所示，preName 就是选择模块的名称，fileName 就是保存数据的文件名称，flag 的值为布尔值，根据这个值来判定爬虫所有模块数据还是只是爬取某一个特定的模块数据，设置这个接口的主要目的是爬取所有的模块数据太过于浪费时间，所以为了节约时间，可以爬取所需要的模块数据即可。

**表 4.16 分模块爬取数据的接口**

| **url** | **/common/executePython** |
|---|---|
| 参数 | preName，fileName，flag |
| 返回 | 状态码和结果集 |
| 使用结果 | 得到相关联 API 排行榜数据 |

## 4.6 小结

本章介绍爬虫程序的主要功能，爬虫程序的流程图，系统的数据库设计，包括数据库的实体属性图，数据库的表格展示，数据库中的 E-R 图，系统的类图设计，前后端项目的重点功能介绍和流程图设计还有后端的具体接口，包括 URL，传入参数和返回的结果。展示系统分析的设计内容，确定系统的功能、设计方法，讨论系统应该如何进行设计，以满足系统设计目的。

# 第五章　系统实现

系统整体架构如图 5.1 所示：主要包含两个部分，一部分是由 Python 语言写的爬虫程序，爬取的网站为 ProgrammableWeb 网站，爬虫程序想 ProgrammableWeb 网站发送 Https 请求获取 HTML 文本，爬虫程序主要包括 API 模块数据抓取功能，Mashups 模块数据抓取功能，SampleSourceCode 模块数据抓取功能，SDK 模块数据抓取功能，爬虫程序把抓取的数据放入 MySQL 数据库中，然后浏览器通过 http 请求访问本地 Web 服务器，Web 服务器根据代码的具体实现进行相应的显示。

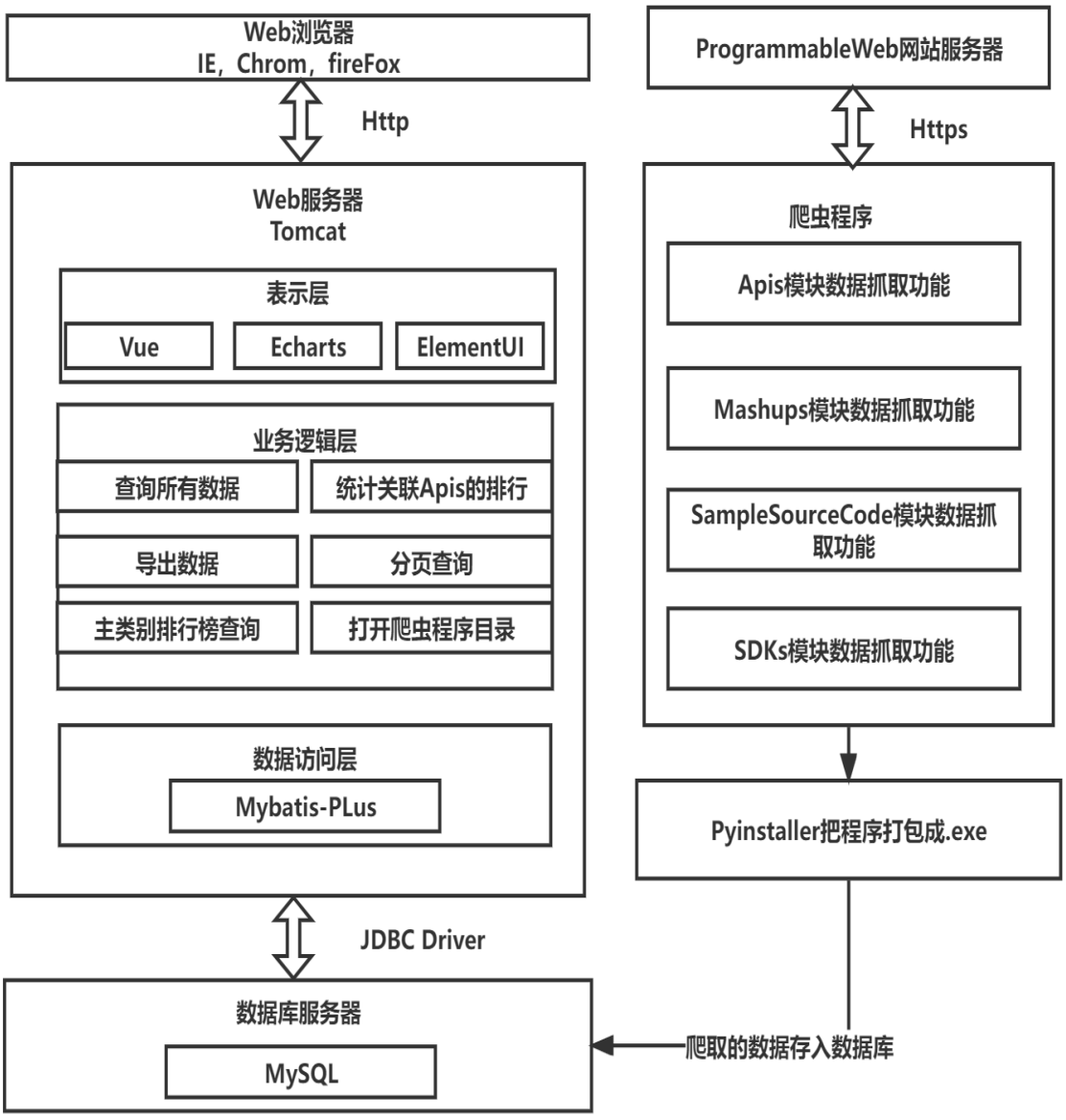


图 5.1 爬虫系统总体架构图

## 5.1 项目功能

### 5.1.1 爬虫程序项目

**(1) 项目结构**

如图 5.2 所示，爬虫程序项目结构主要包括数据部分和代码部分，数据部分有存放爬虫数据的 Excel 表格，CSV 文件，也有记录爬虫位置信息的.properties 文件。


图 5.2 爬虫程序项目结构图

**(2) 功能介绍**

- 发送 request 请求获取 HTML 文本并解析

这个功能主要 urllib 封装的方法，传入需要 url 参数，设置响应超时时间然后向 ProgrammableWeb 发送请求，得到响应的 HTML 文本，然后把 HTML 解析成文本的形式。

- 重发功能

这个功能主要是为了防止网络异常导致获取相应数据为空的情况，当第一次发送请求失败后，就会让程序休眠几秒中，然后继续发送，设置重发的次数，当重发达到一定次数时，就启动异常处理的机制，丢弃这个 url 然后使用下一个 url 继续发送。

- 记录爬虫爬取 ProgrammableWeb 网站的位置信息

这个功能主要是靠.properties 文件实现的，每次爬取完数据后，记录下一个需要爬取的 url 地址，页数，总记录数，如果出现某些意外情况(如电脑关机)，那么下一次重新运行程序时，就不需要从头开始爬取，就从上一次运行程序时爬取到的位置开始，这样有利于解决爬虫时间周期长而导致的重新爬取的问题。

- 获取需要的数据

每次把解析的 HTML 文本进行提取，通过 CSS 选择器，正则表达式，XPath 等方

式对文本内容进行提取，得到想要的关键数据，CSS 选择器和 Xpath 的方式都可以通过浏览器复制，得到相应的规则代码，但是正则表达式需要自己来定义，所以，能否正确定义正则表达式关系到数据能否获取成功。

- 保存爬取的数据

这个功能是为了数据的持久化，做到爬取一次数据，就可以使用很久，不用每次需要数据的时候重新爬取，程序会把爬取的数据保存在 Excel 表格和 CSV 文件中。

### 5.1.1 数据展示项目的功能

**(1) 项目结构**

- 前端项目结构

前端主要包括发送请求的 js 文件，用于路由跳转的 router.js 文件，程序运行的 main.js 文件，显示页面效果的 Vue 文件，用来改变组件样式的 css 文件，还有各种的配置文件。

- 后端项目结构

后端主要包括 controller，service，mapper，config，entity 这几个包文件，controller 选择相应的接口来处理 http 请求，service 主要用于实现业务逻辑代码，mapper 用于跟数据库进行交互，config 用于添加一些配置代码，entity 用来存放实体类。

**(2) 功能介绍**

- 登录功能

登录时会输入账号名和密码，然后会对账号名和密码进行校验，如果正确，会保存登录的状态，登录成功进入主界面，下次再次登录时就会检验登录状态，如果登录过，就自动登录进入主界面。

- 数据查询

数据查询分为一般分页查询，根据某个字段的某个值进行分页查询和模糊查询，然后根据某个字段进行进行查询和排序。

- 数据显示

数据显示的方式有各种样式的条形图，词云，饼状图，条形图分动态条形图和静态条形图，词云也分动态词云和静态词云。

- 调用爬虫程序

在前端页面点击按钮，就可以打开爬虫程序的目录，双击 main.exe 就可以实现运行爬虫程序达到更新数据的目的。

## 5.2 系统效果展示

### 5.2.1 爬虫程序

**(1) 运行结果**

如图 5.3 所示，程序边运行边把发送请求的 url 和抓取的数据在控制台打印，方便随时了解到数据抓取的进度，程序都是抓取完一条完整的表单数据之后，然后把数据封装在字典中，然后把数据持久化到本地的 Excel 表格中，数据封装的格式如下图 5.3 所示。

```
https://www.programmableweb.com/category/all/mashups?page=0
https://www.programmableweb.com/mashup/behold
https://www.programmableweb.com/mashup/behold/followers
{'mashups_name': 'Behold', 'related_apis': 'Instagram Graph', 'description': 'Add Instagram fee
  JSON API. No API keys or back end code required. Perfect for SPAs and JAMstack.', 'categories
 '03.23.2022', 'mashup_app_type': 'Web', 'company': '', 'followers_numbers': '1', 'followers_na
row: 2
https://www.programmableweb.com/mashup/motics
https://www.programmableweb.com/mashup/motics/followers
{'mashups_name': 'Motics', 'related_apis': 'Google AdMob', 'description': 'Motics helps you sta
 Mopub. You have all your advertising data in one place.\nMotics has real-time data, view your
 date range.\nfeature:\n- Monitor app monetization with real-time reports.\n- Set goals and com
 impressions, fill rates, revenue and more from Admob.\nSupported ad networks:\n- Admob.\n- Mop
 .com', 'categories': 'Advertising###Mobile', 'submitted_date': '03.22.2022', 'mashup_app_type'
 '0', 'followers_names': ''}
row: 3
https://www.programmableweb.com/mashup/daily-crypto
https://www.programmableweb.com/mashup/daily-crypto/followers
```

图 5.3　爬虫程序运行结果图

**(2) 位置信息结果**

如图 5.4 所示，这个.properties 文件中记录的程序的位置信息，name 表示模块的名称，url 代表对应模块的抓取的网址，page 保存的页数信息，count 记录的是数据的总数量，这个文件很重要，这是保证爬虫程序能够在程序中断的情况下，恢复上一次爬取位置关键，如果没有这个文件，那么程序每次意外中断，都是要重新运行，这就需要另外花费大量的时间，有了这个文件，也不需要时刻关注程序的运行。

```
apis.name=apis
apis.url=https://www.programmableweb.com/category/all/apis?page
apis.page=0
apis.count=0

mashups.name=mashups
mashups.url=https://www.programmableweb.com/category/all/mashups?page
mashups.page=0
mashups.count=0
```

图 5.4　爬虫程序运行结果图

**(3) 保存的数据(部分展示)**

如图 5.5 所示，这个里面就是通过爬虫程序爬取到的不同的数据，然后把数据保存在数据库之中，数据有 7 万多条，所以这里只展示一部分爬取到的数据。

| mashups_name | related_apis | descript |
|---|---|---|
| Triporia Hotel Search | Expedia | Triporia |
| Yoke | Zapier | Yoke is |
| Flex Mail | TEC Mailing | Flex Mai |
| Perfecto | Bike Index | Perfecto |
| Triporia | Expedia | Triporia |
| Local Weather | ipinfo.io IP geolocation###OpenWeatherMap | Local We |
| MIT Visual Traceroute | ipinfo.io IP geolocation | Visual T |
| Dinero.no | zanox###Finansportalen | Dinero.n |
| Your Automated China Investment Advisor | Highcharts###Highcharts Highstock | Your Aut |
| Open Data Companion (ODC) | CKAN###CKAN Ireland###Brazil CKAN###CKAN Czech Republic###CKAN Ital | The Oper |
| Kissmetrics on Shopify | Shopify Admin###KISSmetrics | Kissmetr |
| Segment | Pipl | Segment |
| Jabify | Telegram Bot | Jabify i |
| Posto | Resfly | Posto is |
| 99designs Tasks Slack Bot | 99designs Tasks###Slack Real Time Messaging | The 99de |
| Swyft | GTFS Data Exchange | Swyft is |
| Elsy Arres Flights | Pyton Flight Portal | Elsy Arr |
| Boom Cellar | Highcharts Highstock | Boom cel |
| Spotisoma | Spotify Metadata | Spotisom |
| Api Expert - MyMemory Language Translator | MyMemory | MyMemory |
| Api Expert - World Weather Forecast | World Weather Online | World We |
| Api Expert - Yelp Local Business Search | Yelp Fusion | Yelp Loc |
| Api Expert - Stack Overflow | Stack Exchange API | Stack Ov |
| Api Expert - Indeed Job Search | indeed | Indeed J |

图 5.5 爬虫程序运行结果图

### 5.2.2 数据展示

**(1) 登录界面**

如图 5.6 所示，登录界面输入密码时，有个小的动态效果，可以点击眼睛的图标，这样密码就可以就可以显示出来，这个界面就是为了输入用户名和密码然后进行登录。


图 5.6 登录界面

**(2) 总记录数界面**

如图 5.7 所示，总记录数界面显示了每个模块的总记录数，同时有一个饼状图，蓝色代表 API 模块，绿色代表 Mashups 模块，黄色代表 SampleSourceCode 模块，红色代表 SDK 模块，这个四种颜色在饼状图都有一定的占比，代表这四个模块的记录数在所有模块加起来的记录数中的相应占比，占比越大，代表模块记录数越多。

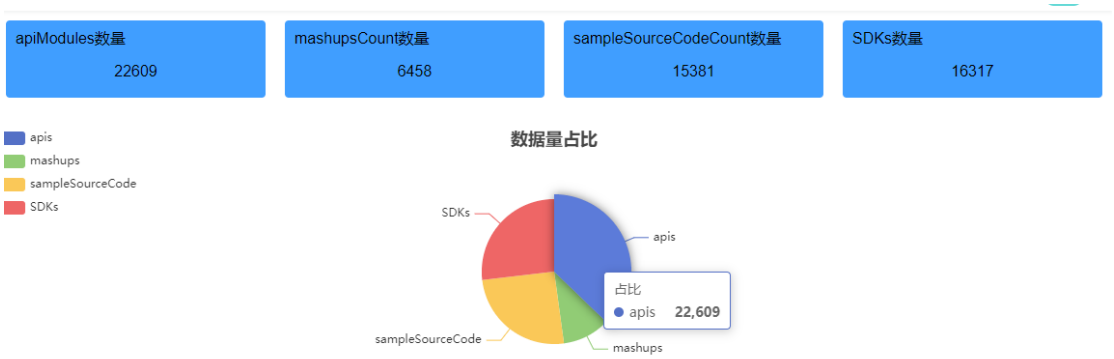


图 5.7 总记录数占比饼状图

**(3) API 模块界面**

如图 5.8 所示，图上方的数字可以根据用户的需求进行动态的改变，让其显示几个数据，就显示几个数据，下方条形图也会根据数字的改变而改变，这个数据的来源是把类别字段进行分隔得到主类别然后进行进行统计得出来的。

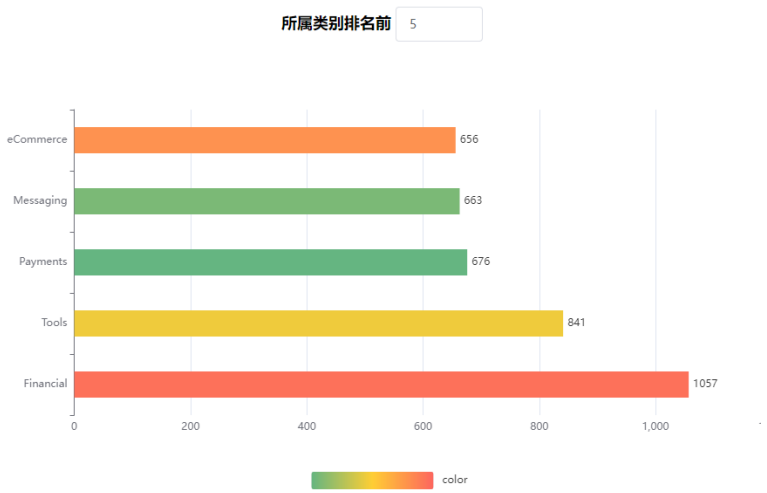


图 5.8 API 主类别排行榜统计柱形图

数据结果查询的界面如图 5.9 所示，有一个导出所有数据的按钮，点击即可把 API 模块的所有数据导出为 Excel 表格，同时有一个下拉选择框和一个输入框，连个配置一起使用，可以根据不同的字段的不同值进行分页的模糊查询，由于有些字段的数据太过于多，在页面之上不能够完全展示，所以这个表单里面设置了当数据超过一定长度时，后面的数据就省略，如果想要查询完整的字段数据，可以把鼠标放在数据上，这样数据就可以完整显示，最下面是一个分页条，展示了总记录数，总页数，同时可以自定义每一页显示的记录条数，在分页条都可以进行选择，有 5 条/页，10 条/页，20 条/页、30 条/页，可以根据自己的需求进行相应的分页显示，同时可以选择查看第几页的数据，选择相应的数据就可以进行页面的跳转，同时由于页数过多，不能完整显示，后面还提供了一个自己输入的功能，输入想要跳转的页数，然后就会搜索相应的结果展示在前端界面之上。

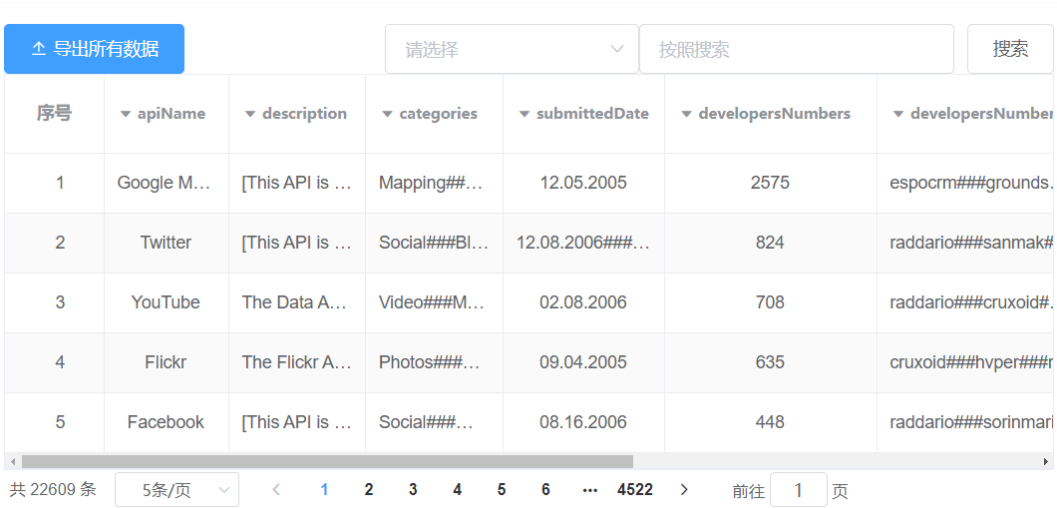


图 5.9 API 数据查询结果图

**(4) Mashups 模块界面**

如下图 5.10 所示，图上方的数字可以根据用户的需求进行动态的改变，让其显示几个数据，就显示几个数据，下方条形图也会根据数字的改变而改变，这个数据的来源是把类别字段进行分隔得到第一为主类别然后进行进行统计得出来的，值得一提的是，这个条形图的样式并不是静态的，而是动态的，这个可以实现一个动态递增的效果。

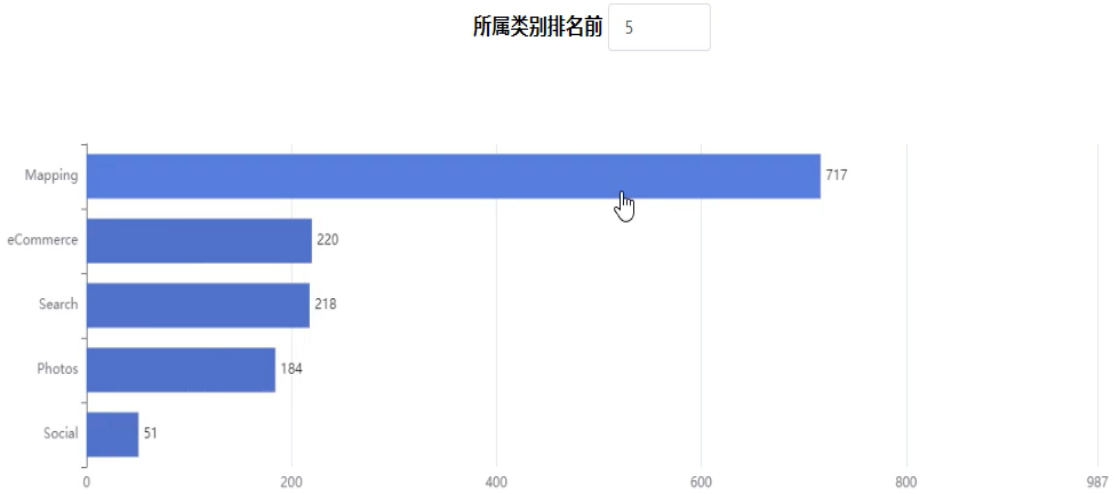


图 5.10 Mashups 主类别排行榜统计柱形图

如图 5.11 所示，图上方的数字可以根据用户的需求进行动态的改变，让其显示几个数据，就显示几个数据，下方词云图显示的文字个数也会根据数字的改变而改变，这个数据的来源是把类别字段中按照分隔符进行分隔，分隔之后，然后使用统计函数进行统计，得到排名最高的几个类别，把这几个使用频率最高的类别使用词云图进行展示。

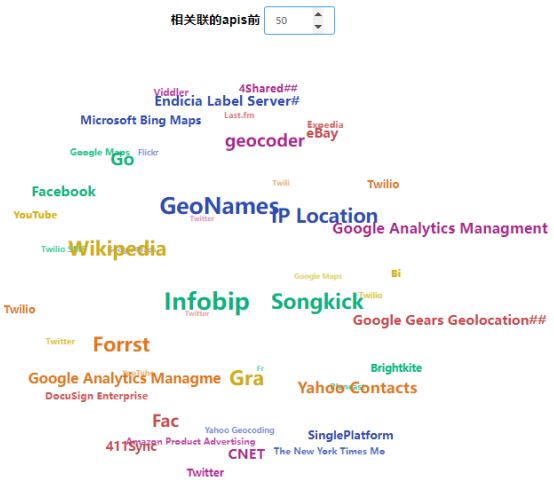


图 5.11 Mashups 相关联 API 词云图

如图 5.12 所示，这个查询界面跟 API 模块的查询界面功能和效果都比较类似，只不过数据源的不同而已，所以这里就不进行具体的阐述。

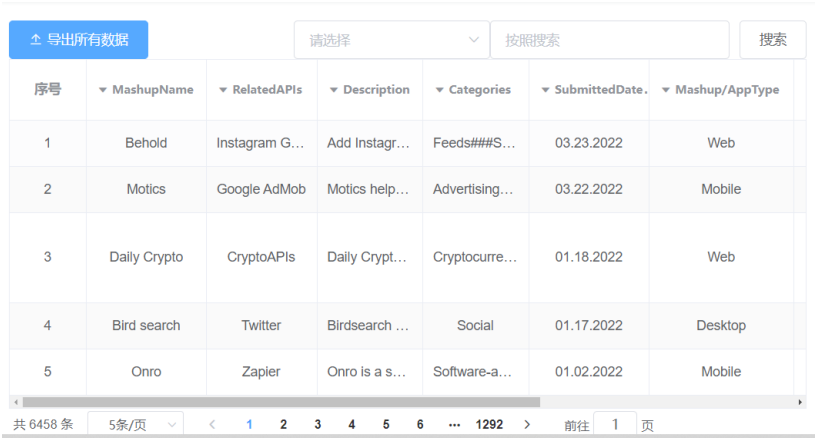


图 5.12 Mashups 数据查询结果图

**(5) SampleSourceCode 模块界面**

如图 5.13，5.14 和 5.15 所示，这个模块与 API 模块功能类似，只是数据源不同，图表的样式也有所不同，代码的实现方式是一致的，所以就不具体进行阐述。

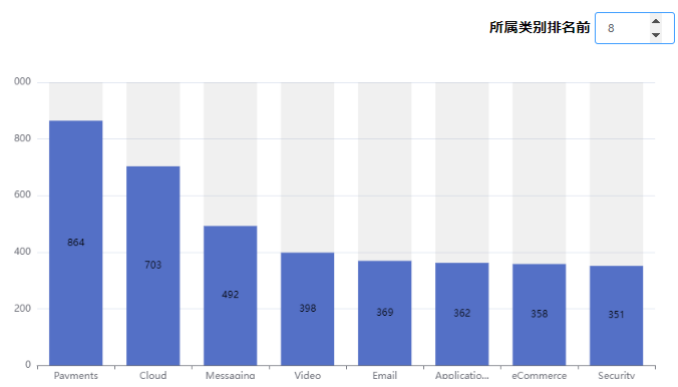


图 5.13 SampleSourceCode 主类别排行榜统计柱形图

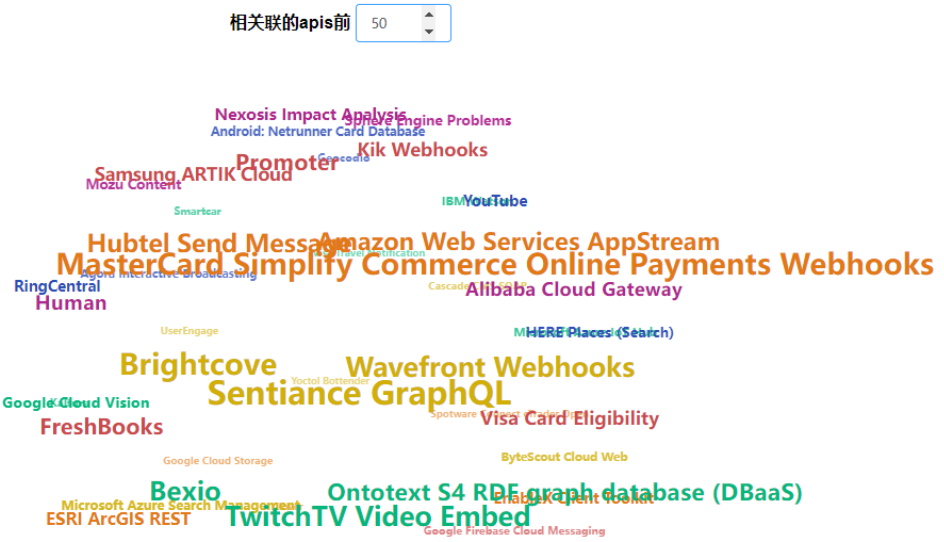


图 5.14 SampleSourceCode 相关联 API 词云图

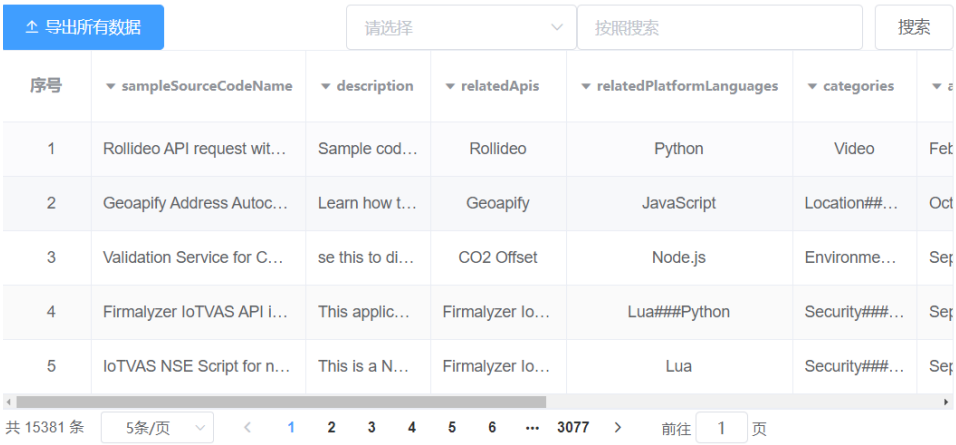


图 5.15 SampleSourceCode 数据查询结果图

**(6) SDK 模块界面**

如图 5.16 所示，图上方的数字可以根据用户的需求进行动态的改变，让其显示几个数据，就显示几个数据，下方条形图也会根据数字的改变而改变，这个数据的来源是把类别字段进行分隔得到第一为主类别然后进行进行统计得出来的。


图 5.16 SampleSourceCode 主类别排行榜统计柱形图

如下图 5.17 所示，这个 tag 的排行榜，tag 表示的意思是除了主类别之外的数据，都为 tag，如一个 A###B##C，第一个 A 为主类别，其余的 B，C 视为 tag，所以我们需要先把字符分隔，然后去掉第一个才能得到 tag，然后把得到的 tag 进行统计，得到一个排行榜，下图中字体越大表示某个 tag 统计的次数越多。


图 5.17 SampleSourceCode 中 Tag 排行榜词云图

## 5.3 小结

本章主要介绍系统的具体功能以及系统的页面实现。通过系统的具体功能页面实现，展示了根据系统设计进行开发的系统的每一部分的具体实现，体现系统的整体设计目标，同时对整个系统有个大概的认识和了解，系统中有许多具体实现的图，通过查看这些图，可以让我们对系统有一个更加直观的感受。

# 第六章　总结

此次我的做是基于 ProgrammableWeb 的 Web 服务爬虫系统，这对我来说具有一定的挑战性，因为爬虫这个技术之前是没有接触过的，不过当过接触爬虫之后，发现这个技术其实挺简单的。

本次设计主要包括三个内容，第一个是使用 Python 代码编写一个爬虫程序，这个爬虫程序向 ProgrammableWeb 发送请求，然后获取 HTML 文本，再对 HTML 进行解析，最后筛选出我们所需要的表单数据，接着把爬取的表单数据存储在 MySQL 数据库中。第二个是后端项目，这个主要是通过跟 MySQL 数据库进行交互，对数据进行一些处理之后，把数据封装成 JSON 格式返回。前端项目，主要是发送 Axios 请求给后端接口，然后获取 JSON 数据，把获取的 JSON 数据进行解析，再传递给 Echarts 图表，这样这就可以把数据显示在 Echarts 图表，同时前端项目还可以实现导出数据到 Excel 表格中，根据不同的字段进行模糊分页查询，根据不同的字段进行排序的功能，还可以实现调用 Python 的爬虫程序进行数据的更新操作。

本次设计的基本功能都已经实现了，自己也从中收获到了许多的东西，但是还存在一些问题，如存在许多的冗余代码，SQL 执行的时间过于长，在前端项目中，样式的布局存在一些问题，这些都是我后面需要改进的地方，通过本次毕业设计，我感觉把之前所学的框架技术进行了一次动手的实践，提高了我自己的代码水平，也使我对学习到的理论知识有了更加深刻的理解。

# 参 考 文 献

[1]詹恒飞，杨岳湘，方宏. Nutch 分布式网络爬虫研究与优化[J]. 计算机科学与探索，2011(1).

[2]李纪欣，王康，周立发等. Google Protobuf 在 Linux Socket 通讯中的应用[J]. 电脑开发与应用，2013(4).

[3]彭轲，廖闻剑. 基于浏览器服务的网络爬虫[J]. 2009 年 04 期.

[4]王江红，朱丽君，李彩虹. 一种新型网络爬虫的设计与实现[J]. 微计算机信息，2010 年 03 期.

[5]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 年 15 期.

[6]李培. 基于 Python 的网络爬虫与反爬虫技术研究[J]. 计算机与数字工程， 2019， 47(6):7.

[7]毛国君，段立娟. 数据挖掘原理与算法(第二版)教师用书[M]. 清华大学出版社，2009.

[8]winter. 中文搜索引擎技术解密：网络蜘蛛[M].北京：人民邮电出版社，2004.

[9]罗刚，王振东. 自己动手写网络爬虫[M]. 北京：清华大学出版社，2010，10.

[10]郭涛，黄铭均. 社区网络爬虫的设计与实现[J]. 智能计算机与应用，2012（4）.

[11]刘晶晶. 面向微博的网络爬虫研究与实现[D]. 上海：复旦大学，2012.

[12]崔庆才. python3 网络爬虫开发实战上[M].北京：中国工信出版集团，人民邮电出社，2017.

[13]李晓明，闫宏飞，王继民. 搜索引擎：原理、技术与系统——华夏英才基金学术库 [M]. 北京：科学出版社，2005 年 04 月.

# 致　　谢

本次论文写到这里就结束了，致谢的开始同时也意味着四年的大学生活的即将结束，意味着我们将要带着大学四年的成长和收获走入一条通往未来的未知道路了。大学四年的收获让我不知道选择什么样的语言才能清楚的描述出我这一时刻的心情，湖南科技大学有我带不走的回忆，更有带的走的收获，四年的一切仿佛还历历在目，让人非常的留恋，这些都将会成为我终生难忘的回忆，使我终身受益，让我倍感珍惜。

在本次毕业设计中，有很多同学和老师都给我很大的帮助，首先，要感谢我的指导老师——刘建勋老师，刘建勋老师从大一就开始带我学习项目，教会了我许多的框架技术和开发技巧，同时也让我的眼界得到了很大的的提升。其次，要感谢的老师是康国胜老师。在毕业设计以及毕业论文的撰写过程中，康国胜老师都帮助了我很多，从选题开始一直到最后顺利完成论文，都离不开康国胜老师对我的悉心教导，康老师深厚的专业知识和严谨的教学态度，使我对老师的敬意油然而生。最后，感谢学长学姐，每次我有很多的疑惑，总是能耐心地帮助我，在毕业设计的过程中，我曾经一度非常迷茫，不知道如何开始进行设计，学长学姐一直非常耐心的帮助我，帮我解决我的困惑，给我提供了非常多宝贵的建议，在大学四年的最后，能和学长学姐一起度过这一段虽然辛苦但是却非常珍贵的时光，将会是我受益匪浅的记忆。

当然，论文的完成也离不开我身边的同学和舍友。在这段时间里，我和同学们之间互相交流实验心得以及遇到的一些问题，寻找解决这些问题的方法，探讨论文思路的构想。还有我的舍友也给我提供了很大的帮助，我们一起制定毕业设计计划，互相督促对方实施计划。可以说他们是我毕业设计过程中最坚实的后盾和伙伴，给了我很大的帮助，非常感谢他们。

最后也为自己能够独立的完成一个系统的设计而感觉十分的开心，说明的大学四年的努力没有白费，自己所学到的理论知识能够跟实际相结合，非常感觉学校和学院给我们提供的各种便利，让我能够没有后顾之忧的沉浸在系统的设计中，在这个过程中，我不断地成长，不断地学习新的技术，感觉这对我的实际操作和动手能力是一次大的提升。

以梦为马，不负韶华，感谢曾经努力、不曾放弃的自己，纵使前路仍然迷茫，但我会一直努力下去，不断前行，不断成长，不负母校的教诲。

行文至此，感谢所有帮助支持我的人，也祝愿所有人身体健康、工作顺利、平安喜乐。