



**本科毕业论文**

|  |
| --- |
| **生活类爬虫系统** |
|  |

**刘炜铭**

**201527010414**

|  |  |
| --- | --- |
| 指导教师 | **綦羽 讲师** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 学院名称 |  | **数学与信息学院** | 专业名称 |  | 软件工程 |
| 论文提交日期 |  | 2019年4月10日 | 论文答辩日期 |  | 2019年4月21日 |

摘 要

网络爬虫的本质是一个24小时自动探测机器，它的行为就是模拟人去各个网站看看，然后带着我们所需的信息回来。我们使用的百度搜索引擎，它其实也是一个爬虫，每天放出无数个爬虫到各个网站，把信息带回来后，再整合然后提供给人们去检索。为了让某个网站更加容易被检索出来，我们可以通过爬虫提高它的点击率，这个过程就叫做 SEO（搜索引擎优化）。现在出现很多聚合电商，他们自己不生产任何产品，他们是搬运工，把天猫、淘宝、京东等商品聚合起来让用户筛选，成为电商中的百度，从而获益。可见网络爬虫与我们息息相关，它能极大的”优化”我们的生活。

本文将研究并开发一个生活类爬虫软件，它可以第一时间提示并且抢购附近低价美食、推送有关编程技术类型的文章、正在热播和准备上映的电影和机票价格等。为了方便使用，本文将用Chrome插件作为客户端，让用户只要打开流浪器，就可以实时收到更新消息。从技术架构上，前端将使用React和Material-UI框架开发。web服务后台使用Python3的Flask框架，为了提高系统稳定性，本文将介绍Nginx加uWSGI等反向代理工具，开发一个多进程、单线程的web服务后台。爬虫系统使用Crontab定时Python3任务和XPath语法分析网页HTML结构，再使用正则表达式筛选正真所需的内容。使用MySQL:5.70作为持久层，Redis作为缓存，Docker作为虚拟化工具，SQLAlchemy作为ORM框架，Alembic作为数据库表版本控制。

关键词：网络爬虫 Chrome插件 XPath Crontab Docker

Application of flask Framework Combination Technology in web crawler System

Liu Weiming

(College of Software Technology, South China Agricultural University, GuangZhou, 510000, China)

Abstract: In fact, web crawler is a detection 24-hours machine. It`s base action is searching on the internet like common human do and back with some message, which is what we want. Baidu Search Engines, a web crawler as well, Baidu product a lot of web crawler every day, back and transfer to useful message, open for us searching. We can use web crawler to increase a website click through rate so that will be more easily retrieve, this skill is called SEO(Search Engine Optimization). Now is coming up so many polymeric e-commerce, they don`t need to produce own products, just showing commodity from TMALL, Taobao, Jindong and so on, become the Baidu of e-commerce. It easy to see, web crawler connected our life, it can make a hug convenient to our life.

This article is going to build a life-use web crawler application, the crawler will push and buy cheapest food around your city, showing the blog of programming, movie on show and upcoming, air ticket price and so on. For easier use, this article will build a Chrome extensions for client, so user can receive message only by opening Chrome browser. From the technical framework, using react and material-ui framework for client. Web server will use Python3-flask framework, to more stable, this article will introduce Nginx and uWSGI reverse proxy tools to build a multi process, single thread web server application. Web crawler system will use linux-crontab timing task Python3 job and XPath syntax analysis website html structure, RegEx for select the infORMation we want. MySQL:5.70 for database, Redis for cache, Docker for virtualization, SQLAlchemy for ORM framework, and Alembic for database version control.

Keywords: Web-Crawler Chrome-Extensions XPath Crontab Docker

目 录

[1. 前言](#_Toc32746_WPSOffice_Level1) [1](#_Toc32746_WPSOffice_Level1)

[1.1 课题的研究背景](#_Toc28551_WPSOffice_Level2) [1](#_Toc28551_WPSOffice_Level2)

[1.2 国内外相关课题的研究现状](#_Toc27709_WPSOffice_Level2) [1](#_Toc27709_WPSOffice_Level2)

[1.3 论文结构](#_Toc19676_WPSOffice_Level2) [2](#_Toc19676_WPSOffice_Level2)

[2. 相关技术介绍](#_Toc28551_WPSOffice_Level1) [3](#_Toc28551_WPSOffice_Level1)

[2.1 React框架介绍](#_Toc1258_WPSOffice_Level2) [3](#_Toc1258_WPSOffice_Level2)

[2.2 Material-UI框架介绍](#_Toc29808_WPSOffice_Level2) [3](#_Toc29808_WPSOffice_Level2)

[2.3 Flask框架介绍](#_Toc11901_WPSOffice_Level2) [4](#_Toc11901_WPSOffice_Level2)

[2.4 SQLAlchemy框架介绍](#_Toc14185_WPSOffice_Level2) [4](#_Toc14185_WPSOffice_Level2)

[2.5 Alembic框架介绍](#_Toc17569_WPSOffice_Level2) [4](#_Toc17569_WPSOffice_Level2)

[2.6 MySQL:5.70数据库介绍](#_Toc32118_WPSOffice_Level2) [4](#_Toc32118_WPSOffice_Level2)

[2.7 Redis介绍](#_Toc26669_WPSOffice_Level2) [4](#_Toc26669_WPSOffice_Level2)

[2.8 Docker虚拟化技术介绍](#_Toc1109_WPSOffice_Level2) [4](#_Toc1109_WPSOffice_Level2)

[2.9 Crontab介绍](#_Toc1830_WPSOffice_Level2) [5](#_Toc1830_WPSOffice_Level2)

[2.10 XPath路径表达式语法介绍](#_Toc29770_WPSOffice_Level2) [5](#_Toc29770_WPSOffice_Level2)

[2.11 uWSGI介绍](#_Toc6599_WPSOffice_Level2) [6](#_Toc6599_WPSOffice_Level2)

[2.12 Nginx介绍](#_Toc14724_WPSOffice_Level2) [6](#_Toc14724_WPSOffice_Level2)

[3. 需求分析](#_Toc27709_WPSOffice_Level1) [7](#_Toc27709_WPSOffice_Level1)

[3.1 系统概述](#_Toc25824_WPSOffice_Level2) [7](#_Toc25824_WPSOffice_Level2)

[3.2 功能需求](#_Toc20873_WPSOffice_Level2) [7](#_Toc20873_WPSOffice_Level2)

[3.3 系统可行性分析](#_Toc21507_WPSOffice_Level2) [7](#_Toc21507_WPSOffice_Level2)

[3.4 运行环境](#_Toc29982_WPSOffice_Level2) [7](#_Toc29982_WPSOffice_Level2)

4. 系统分析与设计 8

[4.1 系统解决思路](#_Toc21092_WPSOffice_Level2) [8](#_Toc21092_WPSOffice_Level2)

[4.2 系统体系结构设计](#_Toc2625_WPSOffice_Level2) [8](#_Toc2625_WPSOffice_Level2)

[4.3 系统包设计](#_Toc32588_WPSOffice_Level2) [8](#_Toc32588_WPSOffice_Level2)

[4.3.1 爬虫系统包设计](#_Toc6599_WPSOffice_Level3) [8](#_Toc6599_WPSOffice_Level3)

[4.3.2 后台系统包设计](#_Toc14724_WPSOffice_Level3) [9](#_Toc14724_WPSOffice_Level3)

[4.3.3 前端包设计](#_Toc25824_WPSOffice_Level3) [9](#_Toc25824_WPSOffice_Level3)

[4.4 系统实体类设计](#_Toc26422_WPSOffice_Level2) [9](#_Toc26422_WPSOffice_Level2)

[4.5 系统数据库设计](#_Toc17294_WPSOffice_Level2) [12](#_Toc17294_WPSOffice_Level2)

[4.5.1 数据库表设计格式说明](#_Toc26422_WPSOffice_Level3) [12](#_Toc26422_WPSOffice_Level3)

[4.5.2 数据库模型](#_Toc15689_WPSOffice_Level3) [16](#_Toc15689_WPSOffice_Level3)

[4.6 系统功能设计](#_Toc30386_WPSOffice_Level2) [16](#_Toc30386_WPSOffice_Level2)

[4.6.1 用户注册](#_Toc10043_WPSOffice_Level3) [17](#_Toc10043_WPSOffice_Level3)

[4.6.2 编程类博客主页](#_Toc22556_WPSOffice_Level3) [17](#_Toc22556_WPSOffice_Level3)

[4.6.3 电影主页](#_Toc13799_WPSOffice_Level3) [17](#_Toc13799_WPSOffice_Level3)

[4.6.4 美食主页](#_Toc4844_WPSOffice_Level3) [17](#_Toc4844_WPSOffice_Level3)

[4.6.5 机票主页](#_Toc414_WPSOffice_Level3) [17](#_Toc414_WPSOffice_Level3)

[4.6.6 评论](#_Toc10617_WPSOffice_Level3) [18](#_Toc10617_WPSOffice_Level3)

[4.6.7 推送](#_Toc1668_WPSOffice_Level3) [18](#_Toc1668_WPSOffice_Level3)

5. 系统的实现 19

[5.1 前端发起api请求实现](#_Toc14192_WPSOffice_Level2) [19](#_Toc14192_WPSOffice_Level2)

[5.2 后端接受请求实现 2](#_Toc3753_WPSOffice_Level2)1

6. 系统效果展示 26

[6.1 用户注册登录页面](#_Toc1233_WPSOffice_Level2) [26](#_Toc1233_WPSOffice_Level2)

[6.2 主页概览](#_Toc5713_WPSOffice_Level2) [26](#_Toc5713_WPSOffice_Level2)

7. 总结与期望 29

[7.1 总结](#_Toc3995_WPSOffice_Level2) [29](#_Toc3995_WPSOffice_Level2)

[7.2 展望](#_Toc15689_WPSOffice_Level2) [29](#_Toc15689_WPSOffice_Level2)

[参考文献 3](#_Toc19676_WPSOffice_Level1)1

[致谢 3](#_Toc1258_WPSOffice_Level1)2

2. 前言
   1. 课题的研究背景

爬虫技术是一种最贴近人类生活的自动化技术。既可为AI自动提供有效的大数据，也可以方便人类生活。比如你的短期计划：租房、旅游、聚会。往往希望有个工具为了筛选心仪的对象。可是网上太多信息了，对于一个有其他重任的人来说，要精打细算浪费的时间（也就是金钱）是不划算的。那么很希望有这样一个软件，来帮你自动筛选，把你想要的消息推送给你。

计算机与人类息息相关，计算机的自动化特性更是取代了大多数疲惫又杂乱的工作，工人易犯错且累，计算机精准而不累。爬虫作为自动化技术的一员，对人类生活起着重要作用。

(1).搜索引擎

搜索引擎的建立需要大量的数据支撑，假如数据样本充足，人们采集信息将会极其对称且便利，人工采集显得有心无力，爬虫技术将可以代替人们源源不断采集数据。

(2).电商领域

如今电商领域，天猫、淘宝、京东、拼多多......信息五彩缤纷，价格也有高有底。如何快速搜索，快速对比成为一大难题。爬虫具有自动化的特征，可以夸张地说，爬虫可以模拟人类一切点击行为，于是聚合电商帮人类解决了这类问题。

(3).出行领域

堵车预报，天气预报，机票、动车价格，买票抢票等等，成为琐碎又不得不去关注的事情，人们每天都浪费很多时间在此。爬虫具有自动提醒功能，可以帮人们省去查看各种事儿的时间。

* 1. 国内外相关课题的研究现状

自1993年初Matthew Grays Wandered在麻省理工学院开发出有史记载的第一个网络爬虫以来，爬虫技术已经发展了20多年，技术已日趋多样。为满足不同用户的不同需求，爬虫分为以下两大类:

(1).通用式爬虫

通用式爬虫指的是通过一系列预设的初始链接开始，根据规则提取当前网页的所需内容，然后提取当前网页里面所有的其他链接，再把这些新链接使用open-close表判重后，放去等待队列，直到到达系统预设指定条件为止的全网式爬虫技术。此类爬虫通常有两类不同的实现方式，可分为:

1).深度优先搜索策略(Depth First Search)，指的是从初始链接开始，选出第一个链接进行访问，处理出有用信息和新的链接集合，再从新的链接集合里面选出一个重复上诉过程，直到没有新的链接，就返回到上一层中选出一个链接进行访问。此策略具有极强的专攻性，可以爬取到最深入灵魂的隐蔽信息。

2).广度优先搜索策略(Breadth First Search)，又或者叫宽度优先搜索策略，指的是从初始链接开始，先把所有初始链接处理完毕，再处理从初试链接生成出来的其他链接，也就是说是一层一层的处理，这样就避免了深度优先搜索策略会使得有些链接没机会处理的情况。

(2).倍增式爬虫(Incremental Web Crawler)

倍增式爬虫解决了僵尸链接问题，主要体现在：网站长久不更新，没有新的信息获取。这个问题对于通用式爬虫来说是一种诟病。倍增式爬虫根据网页更新的频率来自动调整网页权重，具有较高的权重的网页可以占用更高的硬件和网络资源，权重最低的只需分配剩下的计算资源即可。

* 1. 论文结构

本文将研究并开发一个生活类爬虫软件，包括从生产到正式环境部署的过程，并且投入到实际生活使用。本文主要的工作和内容分为以下几个章节：

第一章主要介绍了网络爬虫这一领域的研究背景与意义，分析了这一领域的研究现状和应用方向，说明网络爬虫这种自动化技术已经长达20年的研究历史及人们生活对此的依赖，同时也说明爬虫与我们生活息息相关。

第二章主要介绍了整个爬虫软件所使用的相关技术知识。

第三章主要介绍了需求分析，确定整个系统所需的功能，同时也阐述了这个系统的实现可行性以及发展前景。

第四章主要介绍了系统的分析与设计，同时也详细介绍了数据库部分的设计。

第五章主要介绍的系统的实现，同时截取了部分设计比较优秀的代码。

第六章是整个系统的效果展示图。

第七章对整个毕业设计作出总结，并分析可拓展之处。

1. 相关技术介绍
   1. React框架介绍

React 起源于 Facebook 的内部项目，是一个前端JavaScript MVC框架，具有强大的钩子函数管理功能。因为该公司对觉得现有的框架都有一个毛病，就是都要管理真实的DOM，而经过校验可知，操作DOM的成本是非常非常的高的，可以这么说，前端渲染的主要时间都花在了操作DOM上，于是他们就决定自己写一套，是否可以用虚拟DOM来代替真正的DOM呢？答案是可以的，但是虚拟DOM的更改和重新渲染问题，效率是否会太差呢？不会，虽然比较两个树不同的传统算法是需要O(n^3)的，但是因为web应用来说，还有很多启发式的性质可以利用，所以可以达到O(n)的复杂度。React在Facebook内部用来架设完instagram后，觉得非常好用，于是就开源了。

对于React的新目标React Naive更有一个雄伟的目标,就是用来代替传统的app，当然这个目标还有待实现，由于和本文无关，这里不详细讨论。

React生命周期可以分为七个钩子函数，分别是：

(1).componentWillMount

(2).componentDidMount

(3).componentWillReceiveProps

(4).sholudComponentUpdate(这是一个性能优化点)

(5).componentWillUpdate

(6).componentDidUpdate

(7).componentWillUnmount

每次钩子函数监听到变化，会重新调用render函数执行UI渲染，根据React-diff算法，不必刷新全部DOM，使得渲染效率大大提高。

React-diff算法的优势：

传统 diff 算法：传统 diff 算法通过循环递归对节点进行依次对比，效率低下。

React-diff 策略，使得渲染达到O(n)复杂度。算法主题是：

(1).Web UI中DOM节点跨层级的移动操作特别少，可以忽略不计。

(2).拥有相同类的两个组件将会生成相似的树形结构，拥有不同类的两个组件将会生成不同的树形结构。

(3).对于同一层级的一组子节点，它们可以通过唯一id进行区分。

* 1. Material-UI框架介绍

Material-UI是在React最流行的UI库，遵循Material Design，具有拟物化、质感风格、强烈的阴影和高光。

* 1. Flask框架介绍

Flask是一个使用Python编写的轻量级web应用框架，是基于werkzeug wsgi工具箱和jinjia2模板引擎的开源框架。由于它足够精简而且易上手，易配置扩展等特性，在GitHub收到42731个start，相比与dingo，Flask的可配置性显得十分灵活。

* 1. SQLAlchemy框架介绍

SQLAlchemy是一个由Python编写的ORM库。提供了SQL工具包和对象关系映射工具，SQLAlchemy把数据库表表达为Python的list，每个list是一个truple，表示一行记录。SQLAlchemy采用的是数据映射模型，并非Active Record模型。加上Python这类动态解析语言等特性，想到达到操作数据库的目的，往往只需要写一个公共使用的ORM文件即可，极其方便。SQLAlchemy首次发行于2006年2月，并迅速地在Python社区中最广泛使用的ORM工具之一。SQLAlchemy采用两层架构，最底层的操作数据库采用第三方库的形式完成。

* 1. Alembic框架介绍

Alembic是一个基于SQLAlchemy的数据库版本迁移工具，在往后开发中，往往需要增表改表，Alembic可以做到类似Git一样的，保存数据库的各种历史版本，随着代码更新而部署不同的数据库结构。

* 1. MySQL:5.70数据库介绍

MySQL是一个关系形数据库管理系统，是最好用web-base应用数据库之一，支持事务、表级、行级锁等等。目前属于Oracle下的产品，是一个开源软件。其中，5.70是MySQL一个比较稳定的版本，也是网上技术文章最多的版本，本文使用此版本作为主要存储工具。

* 1. Redis介绍

Redis是一个内存型的key-value数据库，支持hash，string, list, set，order\_set等多种数据结构。内存型数据库能火爆体现在：1、内存越来越廉价，网络速度越来越快。2、效率比落磁盘要高得多。3、Redis也能极大可能保存用户存取的数据。Redis是一个开源项目，于2010年3月15日进行开发。

* 1. Docker虚拟化技术介绍

在Docker问世之前，部署项目代码视乎不是一件容易的事情。还有专门运维的团队，由于直接在服务器上安装依赖往往会弄坏系统变量，导致部署流程很繁琐。Docker提供一种虚拟化技术，支持把服务运行在docker-deam下的container中。一句话概括容器就是将软件打包成标准化单元，以用于开发、交付和部署。如下图所示，每一个container是独立的，因为每个container都有独立的PID，NET，IPC, USER, UTS, MNT，所以每一个container都是互不影响的，只有这样，我们才可以为每一个container安装不同的依赖而不影响宿主机。

相比于虚拟机，Docker的优势是很明显的。传统的虚拟机技术首先需要虚拟出一套硬件，再在其上面运行一个完整的操作系统功能，在该系统上再运行所需的进程。而Docker的容器内的应用进程则直接运行于宿主机内核中，容器没有自己的内核，而且没有进行硬件虚拟。因此容器要比传统虚拟机更为轻便。对比如表1所示。

表1 容器与虚拟机比较

|  |  |  |
| --- | --- | --- |
| 特性 | 容器 | 虚拟机 |
| 启动 | 秒级 | 分钟级 |
| 硬盘使用 | MB级 | GB级 |
| 性能 | 接近原生 | 弱于原生 |
| 系统支持量 | 上千个 | 几十个 |

* 1. Crontab介绍

Linux下的定时任务，Crontab成为很多服务器定时任务的首选。在Linux中，周期执行的任务一般由cron这个守护进程来处理。cron读取一个或多个配置文件，这些配置文件中包含了命令行及其调用时间。cron的配置文件称为“Crontab”，是“cron table”的简写。

Crontab是在后台维护一个守护进程，每一分钟调用一次，每一次调用，则去三个地方查找Crontab文件，进行解析，运行对应语句，然后主程序休眠，如此反复。

1./var/spool/cron/ 这个目录下存放的是每个用户包括root的Crontab任务，每个任务以创建者的名字命名。

2./etc/crontab/ 这个文件负责安排由系统管理员制定的Crontab。

3./etc/cron.d/ 这个目录用来存放任何要执行的Crontab文件或脚本。

* 1. XPath路径表达式语法介绍

XPath是一种选择XML文件中节点的方法，HTML文档也是XML文档的一种，利用XPath语法可以很方便地找出对应元素的对应DOM。

* 1. uWSGI介绍

uWSGI基于c语言开发的一个web服务器，它实现了WSGI协议，它会启动若干个Fask程序进程，每当有请求进来时，它会随机分配个一个空闲的进程处理，每个Flask程序配置成单线程，就构造出一个多进程，单线程的web-server程序。因为web服务瓶颈在IO，不在CPU，所以设计成单线程可以减免线程和锁之间的切换花销。

* 1. Nginx介绍

Nginx是一个网页服务器，它能方向代理http，https，smtp等等链接协议，以及是一个负载均衡器。

为什么需要使用Nginx？作为web-server，很容易受到别人攻击，如果只单单使用uWSGI作为服务器的话，很容易被别人使用慢链接攻击，导致服务器所有Flask程序进程处于busy状态，从而导致服务不可用。Nginx是一个使用c语言开发的高效http反向代理工具，处理慢链接是它的强项。Nginx会把整个http请求接受完整后，再反向代理转发到uWSGI中，uWSGI再根据情况选择一个进程处理，然后再交给Nginx返回给客户端。整个过程Nginx处于第三者工作，抵挡外面先来的请求。保护后台服务进程。同时，Nginx具有高效的转发静态资源能力，可以快速传输静态资源给客户端。

slowloris慢速攻击是rsnake发明的一种http攻击技术，首先http协议规定，http request以\r\n\r\n(0d0a0d0a)结尾表示客户端发送结束，服务端开始可以处理。那么如果永远都不发送\r\n\r\n，客户端以一定的时间间隔发送一个key-value格式的数据到服务端，就可以形成类似的攻击。

1. 需求分析
   1. 系统概述

生活类爬虫系统旨在“优化”人们生活，减少人类机械又低效的网页查询工作。系统采用Chrome插件作为前端展示，结合Nginx、uWSGI、Flask、SQLAlchemy框架作为web服务后台，XPath，RegEx，linux-crontab作为爬虫后台系统。实现了包括用户管理，信息模糊查询，热播、准备上映电影展示，附近美食展示，编程类技术博客展示等功能模块。

* 1. 功能需求

用户管理：用户注册，用户登录或者退出登录。

信息查询系统：把用户输入的信息在所有爬回来的消息中进行模糊查询。

电影展示模块：展示所有爬回来的电影信息，带有购票功能，查看电影详细信息，以及查看电影预告片等。

编程类技术博客展示模块：显示博客关键词信息，标题。跳转到博客页面等功能。

美食展示模块：爬取美团网信息，展示地址和图片信息。带有查看团购价和一键购买。 机票价格展示：查询来自北方航空，南方航空，春秋，九元航空等机票价格信息。

* 1. 系统可行性分析

1、从用户角度上，据调查大多数只坐在电脑前面的用户，都希望有一个自动提醒各种自己心仪消息的软件。而自己只需要正常工作，就能看到最新的消息。其实和能自动提示邮件信息类似，每个人都想有一个软件自动提示收件箱的最新邮件，而不是自己担惊受怕的点开邮箱，生怕错过重要邮件。

2、从技术角度上，爬虫已经经过20余年的发展，网上成功例子或者开源的爬虫库也很多，可以参考。其实整个互联网世界充斥着爬虫，是有先人铺垫的。前端Chrome插件，Chrome作为最多人使用的浏览器，用户群大，背后的谷歌公司是世界上最大的IT公司。web-server端采用Nginx + uWSGI + Python3-Flask实现的这种多进程单线程服务是web开发主流，所以从技术角度上是完全可行的，都有前辈的经验指导。

3、从经济角度上，只需要一台电脑即可开发，一台服务器也可部署。

从以上三方面考虑，整个爬虫软件是很有开发前景的。

* 1. 运行环境

硬件：RAM: 512mb, Disk-size: 10GB, system: Ubuntu 16.04 x86\_64, CPU:QEMU Virtual CPU version (cpu64-rhel6)

软件：Docker、MySQL:5.70、Redis、linux-crontab

1. 系统分析与设计
   1. 系统解决思路

（1）爬虫系统的实现：使用Crontab定时任务处理，不使用语言本身自带的定时器处理是因为不希望程序一直占用内存和数据库，redis的链接。每次唤醒爬虫服务，都会读取待爬的链接，然后一个一个去带着cookies请求，返回HTML后使用XPath语法获取所需元素的节点，RegEx判断内容是否合适，然后存去Redis和MySQL准备推送。

（2）后台系统的实现：使用Nginx + uWSGI + Flask构造一个多进程，单线程的web服务后台。对于每一个请求，先经过Nginx反向代理到uWSGI守护进程，再由uWSGI拉起一个空闲的Flask进程处理该请求。处理请求时，先判断cookies和所需参数是否合法。

（3）前端系统的实现：使用React作为主要开发框架，Material-UI作为主要UI框架，react-router作为前端路由控制，Chrome插件notification作为主要提醒机制。前端首次加载会引起background.js的调用，设置每5分钟去轮询后台接口是否有更新数据。有则提示推送消息，没有则挂起。

* 1. 系统体系结构设计

本系统采用Nginx反向代理web应用服务器以及前端使用React配合fetch进行前后端通信。后台则使用Python的Flask框架进行处理http请求，SQLAlchemy作为数据库ORM工具。后台设计成多进程，单线程模型，由uWSGI管理，每一个请求仅会被一个进程处理，同一时间内，一个进程只会处理一个请求。前端使用async和await模型来实现异步操作。

系统集成环境与集成测试，本系统所有服务均启动在docker的container里面，无需依赖外部环境。每一次启动的时候，需要先经过测试文件，才能启动正式的docker服务。

单例模式与装饰器模式，系统强依赖这两种模式的实现，单例模式需要固定一个链接比如redis的连接，不允许重复和多次连接到redis的server。装饰器模式运行环境比较复杂多样，有辅助型，也有应用型。比如抓住各种异常错误，这种属于辅助型。而解析用户请求信息和获取用户信息等，属于应用型装饰器。

* 1. 系统包设计

### 爬虫系统包设计

爬虫系统没有使用任何框架，其中主要分包如下：

Local\_dev包：主要负责部署模块，包含makefile和dockerfile等部署文件。

Config包：爬虫系统的主要配置，包括Redis主机IP和MySQL主机IP等。

Database包：SQLAlchemy库的DAO封装

Lib包：包括日志系统，还有错误处理逻辑。

Decorator包：一些装饰器包，包括Python的单例实现等。

Testing包：单元测试用例。

Models包：model层操作。

Request\_client包：爬虫系统的主要部分。

### 后台系统包设计

后台系统使用Flask框架，其中主要分包如下：

Local\_dev包：主要负责部署模块，包含makefile和dockerfile等部署文件。

Endpoint包：后台系统各种api。

Models包：model层操作。

Config包：后台系统的主要配置，包括Redis主机IP和MySQL主机IP等。

Database包：SQLAlchemy库的DAO封装

Lib包：包括日志系统，还有错误处理逻辑。

Decorator包：一些装饰器包，包括Python的单例实现等。

Testing包：单元测试用例。

### 前端包设计

前端展示使用React框架，其中主要分包如下：

Api包：主要用来发起ajax请求，使用async和await封装fetch。

Lib包：主要负责前端日志系统。

Account包：前端账号系统包。

Page包：前端主要页面包。

* 1. 系统实体类设计

表2 User（用户类）

|  |  |  |
| --- | --- | --- |
| 字段 | 数据类型 | 含义 |
| id | Int | 用户id |
| username | String | 用户名称 |
| password | String | 用户密码 |
| Create\_time | Timestamp | 用户创建时间 |
| Json | Json | 用户属性 |

用户类如表2所示，此表说明了存取用户信息，用户管理系统，是数据库user表的操作映射。

表3 Meishi（美食类）

|  |  |  |
| --- | --- | --- |
| 字段 | 数据类型 | 含义 |
| id | Int | 美食id |
| poiId | Int | 美食poiId |
| frontImg | Varchar | 美食题目 |
| avgScore | Float | 美食评分 |
| allCommentNum | Int | 美食评论数目 |
| Address | Varchar | 美食地址 |
| avgPrice | Float | 美食平均价格 |
| create\_time | Timestamp | 美食创建时间 |
| dealList | Json | 美食团购信息 |
| Json | Json | 美食其他属性 |

美食类如表3所示，此表说明了存取附近美食，地址和团购信息，是数据库美食表的操作映射。

表4 Dianying（电影类）

|  |  |  |
| --- | --- | --- |
| 字段 | 数据类型 | 含义 |
| id | Int | 电影id |
| file\_id | Int | 电影豆瓣id |
| password | String | 用户密码 |
| Create\_time | Timestamp | 用户创建时间 |
| Json | Json | 用户属性 |

电影类如表4所示，此表说明了存取最新或准备上映的电影和购票信息，是数据库电影表的操作映射。

表5 Programming（技术博客类）

|  |  |  |
| --- | --- | --- |
| 字段 | 数据类型 | 含义 |
| id | Int | 博客id |
| hash\_id | Int | 博客哈希id |
| Title | String | 博客标题 |
| Create\_time | Timestamp | 博客创建时间 |
| Json | Json | 博客其他属性 |
| Url | String | 博客链接 |

技术博客类如表5所示，此表说明了存取最新的博主更新的博客，是数据库博客表的操作映射。

表6 Plane（机票类）

|  |  |  |
| --- | --- | --- |
| 字段 | 数据类型 | 含义 |
| id | Int | 机票id |
| From | String | 机票起点 |
| To | String | 机票终点 |
| By | String | 机票中途转站 |
| Money | Float | 机票价格 |
| Url | String | 机票链接 |
| Json | Json | 机票其他属性 |

机票类如表6所示，此表说明了存取最新的机票价格，机票路线和购票信息，是数据库机票表的操作映射。

表7 Comment（评论类）

|  |  |  |
| --- | --- | --- |
| 字段 | 数据类型 | 含义 |
| Id | Int | 评论id |
| User\_id | Int | 评论者id |
| Type | Int | 评论类型 |
| Json | Json | 评论类其他属性 |
| Content | String | 评论内容 |

评论类如表7所示，此表说明了存取最新的用户评论和交流等信息，是数据库评论表的操作映射。

* 1. 系统数据库设计

### 数据库表设计格式说明

表8 用户（user）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 数据类型 | 主要的 | 外来键 | 是键 |
| 用户id | uid | Int | TRUE | FALSE | TRUE |
| 用户名 | username | varchar | FALSE | FALSE | TRUE |
| 密码 | password | varchar | FALSE | FALSE | FALSE |
| 创建时间 | Create\_time | TImestamp | FALSE | FALSE | TRUE |
| 用户其他属性 | Json | Json | FALSE | FALSE | FALSE |

数据库用户表如图8所示，此表说明了用户表的结构，包括用户id、账号密码等，用户其他属性是一个保留的Json字段，主要用来方便以后迁移，有些额外的小字段可以直接放进去，然后读出来再解析。

表9 美食（meituan\_meishi）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 数据类型 | 主要的 | 外来键 | 是键 |
| 美食id | id | Int | TRUE | FALSE | TRUE |
| 美食美团id | poiId | Int | FALSE | FALSE | TRUE |
| 美食图片url | frontImg | varchar | FALSE | FALSE | FALSE |
| 创建时间 | Create\_time | TImestamp | FALSE | FALSE | TRUE |
| 美食其他属性 | Json | Json | FALSE | FALSE | FALSE |
| 美食名字 | Title | Varchar | FALSE | FALSE | FALSE |
| 美食平均评分 | avgScore | Float | FALSE | FALSE | FALSE |
| 美食地址 | Address | String | FALSE | FALSE | FALSE |
| 美食平均价格 | avgPrice | Float | FALSE | FALSE | FALSE |
| 美团团购消息 | dealList | Json | FLASE | FLASE | FLASE |

数据库美表如图9所示，此表说明了美食表的结构，包括美食id、地址团购等，美食其他属性是一个保留的Json字段，主要用来方便以后迁移，有些额外的小字段可以直接放进去，然后读出来再解析。

表10 编程博客（medium\_programming）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 数据类型 | 主要的 | 外来键 | 是键 |
| 博客id | id | Int | TRUE | FALSE | TRUE |
| 博客哈希id | Hash\_id | Int | FALSE | FALSE | TRUE |
| 博客url | Url | varchar | FALSE | FALSE | FALSE |
| 创建时间 | Create\_time | TImestamp | FALSE | FALSE | TRUE |
| 博客其他属性 | Json | Json | FALSE | FALSE | FALSE |
| 博客题目 | Title | Varchar | FALSE | FALSE | FALSE |

数据库博客表如图10所示，此表说明了博客表的结构，包括博客id、博客url等，博客表其他属性是一个保留的Json字段，主要用来方便以后迁移，有些额外的小字段可以直接放进去，然后读出来再解析。

表11 电影（douban\_dianying）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 数据类型 | 主要的 | 外来键 | 是键 |
| 电影id | id | Int | TRUE | FALSE | TRUE |
| 电影豆瓣id | file\_id | Int | FALSE | FALSE | TRUE |
| 电影url | Url | varchar | FALSE | FALSE | FALSE |
| 创建时间 | Create\_time | TImestamp | FALSE | FALSE | TRUE |
| 电影其他属性 | Json | Json | FALSE | FALSE | FALSE |
| 电影名字 | Title | Varchar | FALSE | FALSE | FALSE |
| 电影国家 | Region | Varchar | FALSE | FALSE | FALSE |
| 电影分数 | Score | Float | FALSE | FALSE | FALSE |
| 电影图片 | Img | Varchar | FALSE | FALSE | FALSE |
| 电影购票链接 | buy\_ticket | Varchar | FALSE | FALSE | FALSE |
| 电影上映日期 | onshow\_time | Varchar | FALSE | FALSE | FALSE |

数据库电影表如图11所示，此表说明了电影表的结构，包括电影id、电影url、电影评分等，电影表其他属性是一个保留的Json字段，主要用来方便以后迁移，有些额外的小字段可以直接放进去，然后读出来再解析。

表12 机票（plane）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 数据类型 | 主要的 | 外来键 | 是键 |
| 机票id | id | Int | TRUE | FALSE | TRUE |
| 机票起点 | From | Varchar | FALSE | FALSE | TRUE |
| 机票终点 | To | Varchar | FALSE | FALSE | FALSE |
| 机票中转 | By | Varchar | FALSE | FALSE | TRUE |
| 机票其他属性 | Json | Json | FALSE | FALSE | FALSE |
| 机票价格 | Money | Float | FALSE | FALSE | FALSE |

数据库机票表如图12所示，此表说明了机票表的结构，包括机票id、机票url、机票路线等，机票表其他属性是一个保留的Json字段，主要用来方便以后迁移，有些额外的小字段可以直接放进去，然后读出来再解析。

表13 评论（comment）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 数据类型 | 主要的 | 外来键 | 是键 |
| 评论id | id | Int | TRUE | FALSE | TRUE |
| 评论者id | User\_id | Int | FALSE | FALSE | TRUE |
| 评论类型 | Type | Int | FALSE | FALSE | TRUE |
| 评论其他属性 | Json | Json | FALSE | FALSE | FALSE |
| 评论内容 | Content | Varchar | FALSE | FALSE | FALSE |

数据库评论表如图13所示，此表说明了评论表的结构，包括评论id、评论者id、评论类型等，评论类型是一个枚举类型，说明的是评论哪一种东西，评论表其他属性是一个保留的Json字段，主要用来方便以后迁移，有些额外的小字段可以直接放进去，然后读出来再解析。

表14 联系集（comment-user）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 数据类型 | 主要的 | 外来键 | 是键 |
| 评论id | id | Int | TRUE | FALSE | TRUE |
| 评论者id | User\_id | Int | FALSE | FALSE | TRUE |

### 

数据库联系集用户-评论表如图14所示，此表说明了用户表和评论表之间的联系。

### 数据库模型

数据库逻辑图如图1 数据库模型所示。

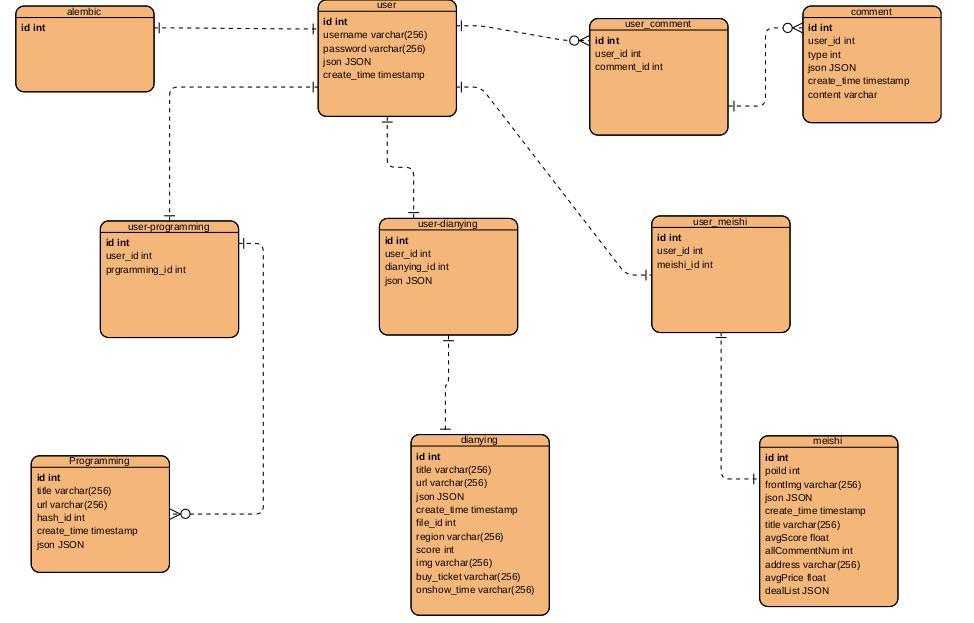


图1 数据库模型

* 1. 系统功能设计

系统功能设计如图2 系统功能设计所示。

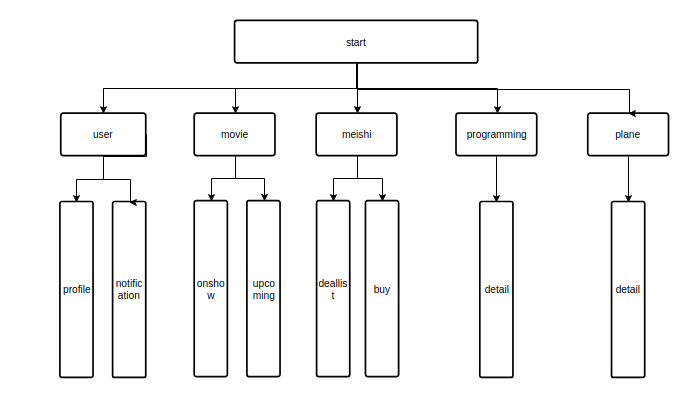


图2 系统功能设计

### 用户注册

用户在注册页面中填写登录账号和密码，如果账号存在，则会判断账号密码与用户所输入的密码是否一致，如果一致则登录成功。如果不一致，则会自动创建一个全新的用户。

### 编程类博客主页

用户登录成功后，会向后台拉取最新的博客，拉取成功后一共有5篇，前端只显示前三篇，后两篇需要点击才能显示。

### 电影主页

用户登录成功后，会向后台拉取最新的正在上映的电影，还有最新的准备上映的电影返回前端渲染。前端采用轮播图展示。包括电影的图片还有购票链接、电影预告片等。

### 美食主页

用户登录成功后，会向后台拉取附近周围的美食链接，返回前端渲染。包括美食图片，地址，标题还有购买美食链接等。

### 机票主页

用户登录成功后，会向后台拉取机票价格主页，支持可配置拉取，默认显示起点、终点、转站点和价格。

### 评论

登陆用户后可以对爬虫信息进行评论，评论后别人可以看到评论，但本系统暂未支持对外开放功能。

### 推送

登陆用户后每当有新的消息出现，比如新的技术文章，新的电影，新的美食等等，都会有一个实时推送功能，以方便用户查看。

1. 系统的实现
   1. 前端发起api请求实现

通过使用async和await封装fetch，使用fetch而不是用ajax的原因是fetch使用了Promise的异步机制，ajax只是简单的发送XMLHttpRequest。但是对fetch使用的异步机制还不是很满意，于是使用ES6特有的async和await进行封装，使得写代码可以像单线程的逻辑写下去，这符合正常人的思维。

主要代码如下：

const API\_HOST = 'http://119.23.231.141:8888';

const OPT = Object.freeze({

method: 'GET',

mode: 'cors',

redirect: 'follow',

credentials: 'include',

headers: new Headers({

'content-type': 'application/json',

})

});

const processResponse = async (response) => {

const contentType = response.headers.get('Content-Type');

if (!contentType) {

throw "no content type";

} else {

if(contentType.indexOf('text') !== -1) {

return await response.text()

} else if(contentType.indexOf('form') !== -1) {

return await response.formData();

} else if(contentType.indexOf('video') !== -1) {

return await response.blob();

} else if(contentType.indexOf('json') !== -1) {

return await response.json();

}

}

}

const processOptions = (options) => {

return Object.assign(options, OPT);

}

export default class RequestBase {

constructor() {

}

doRequest = async (url, options) => {

try {

const fetchUrl = `${API\_HOST}${url}`;

const response = await fetch(fetchUrl, options);

if (response.status !== 200) {

throw await processResponse(response);

} else {

return await processResponse(response);

}

} catch(fetchError) {

const error = new Error(fetchError.message);

error.data = fetchError;

messageBox(`request '${url}' error`, JSON.stringify(error));

throw 'error';

}

}

get = async (url, options = {}) => {

const fetchOptions = processOptions(options);

fetchOptions.method = 'GET';

return await this.doRequest(url, fetchOptions);

}

post = async (url, options = {}, body = {}) => {

const fetchOptions = processOptions(options);

fetchOptions.method = 'POST';

fetchOptions.body = JSON.stringify(body);

return await this.doRequest(url, fetchOptions);

}

put = () => {}

del = () => {}

options = () => {}

};

* 1. 后端接受请求实现

后端通过Flask的蓝图实现http后台服务路由匹配，其中需要从cookies中解析出用户和参数检查。

解析用户和参数检查使用Python的适配器实现，适配器模式也就是装饰器模式，可以把相同的代码抽出来，然后需要用到的地方只需要简单的在函数上方加上一个标签即可用上。这里，做了一个抽取用户信息的适配器。经过对称加密，我们是可以从cookies中解析出session\_id的，但是解析出来session\_id还是不能完全相信，我们还需要从数据库中判断这个用户是否存在。

代码如下：

"""

load user from cookies

"""

def parse\_user():

def decorator(func):

@functools.wraps(func)

def wrapper\_func(\*args, \*\*kwargs):

user\_id = session.get('user\_id')

g.current\_user = User.by\_id(user\_id)

if not g.current\_user:

raise ArgsError('Require login')

return func(\*args, \*\*kwargs)

return wrapper\_func

return decorator

解析用户参数列表适配器，这个函数的功能是帮我们解析用户发送请求的参数。无论是GET请求还是POST请求，都可以解析出来。同一放在Flask的g函数上面，这样就可以全局通用了。解析参数的时候需要判断类型，如果类型不正确会抛出异常，返回对应状态码给前端展示。特别的，由于前端发过来的请求都是字符串，即使对于int类型，前端发过来的也是字符串，所以我们需要为int类型进行尝试转换，使用try...except来抓取异常，以免有恶意攻击着发送类似“123abc”等非整形字符串。

代码如下：

"""

parse args from request

"""

def parse\_args(arg\_name: str, arg\_type: type):

def decorator(func):

@functools.wraps(func)

def wrapper\_func(\*args, \*\*kwargs):

value= request.form.getlist(arg\_name) if issubclass(arg\_type, list) else request.form.get(arg\_name)

value=request.args.get(arg\_name) if not value else value

value=request.json.get(arg\_name) if not value and request.json else value

if not value:

raise ArgsError('Require args {}'.format(arg\_name))

if issubclass(arg\_type, int) and issubclass(type(value), str):

try:

value = int(value)

except ValueError:

raise ArgsError('Require args {}, require type: {}, but found type: {}'.format(arg\_name, str(arg\_type), str(type(value))))

else:

if not issubclass(type(value), arg\_type):

raise ArgsError('Require args {}, require type: {}, but found type: {}'.format(arg\_name, str(arg\_type), str(type(value))))

return func(\*args, \*kwargs.values(), value)

return wrapper\_func

return decorator

其中，后端使用Flask的蓝图实现路由匹配。这个例子整合了前面所说的解析参数的适配器，可以看到，假如我们需要解析用户参数，只需要在对应api上面加上适配器即可。解析出来的参数将有序的放去对应的api函数。使用Flask的蓝图是因为这符合微服务架构，有时候我们需要把某一类的api放去不同的服务器，那么就只需要注册那一类的api即可，不需要注册全部。可以这样说，一个蓝图就是一类的api。所以，这样是非常方便我们迁移和升级的。

例子如下：

user\_api = flask.Blueprint('user', \_\_name\_\_, url\_prefix='/user')

@user\_api.route('/load\_or\_create/', methods=['POST'])

@parse\_args('username', str)

@parse\_args('password', str)

def load\_or\_create(username, password):

user\_obj = User.load\_or\_create(username, password)

session['user\_id'] = user\_obj.id

return jsonify(user\_obj.to\_json())

Docker集成环境里面包括配置uWSGI和Nginx代码，同一封装成一个container。包括4个Nginx的worker和5个uWSGI进程，负责接受http请求和管理调度。

主要代码如下:

FROM ubuntu:16.04

# proxy for apt-get

COPY ./local\_dev/docker/sources.list /etc/apt/sources.list

RUN apt-get update && apt-get install -y --no-install-recommends \

python3 uwsgi libmysqlclient-dev python3-pip python3-mysqldb nginx uwsgi-plugin-python3

# proxy for pip3 install

COPY ./local\_dev/docker/pip.conf /etc/pip.conf

RUN pip3 install --no-cache-dir --upgrade pip

RUN ["/bin/bash", "-c", "pip3 install --upgrade setuptools"]

ARG WORKSPACE="/python3/server"

COPY ./app ${WORKSPACE}/app

COPY ./local\_dev/docker ${WORKSPACE}/local\_dev/docker

WORKDIR ${WORKSPACE}/local\_dev/docker

RUN pip3 install -r ./chrome\_server\_requirements.txt

WORKDIR ${WORKSPACE}/app

ENTRYPOINT nginx -g "daemon on;" && uwsgi myapp.ini

Nginx配置如下所示，log文件输出到对应的container目录，然后使用Docker就可以挂载出来。

server {

listen 8888;

server\_name localhost;

access\_log /log/nginx/access.log;

error\_log /log/nginx/error.log;

location / {

include uwsgi\_params;

uwsgi\_pass unix:///tmp/uwsgi.sock;

proxy\_redirect off;

proxy\_set\_header Host $http\_host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

}

}

uWSGI文件配置如下，使用sock文件进行通信。

[uwsgi]

module = run\_server:app

master = true

processes = 5

buffer-size = 65535

socket = /tmp/uwsgi.sock

# logto = /log/application/uwsgi.log

chmod-socket = 660

plugin = python3

1. 系统效果展示
   1. 用户注册登录页面

用户注册登录页面如图3 登录/注册页面所示

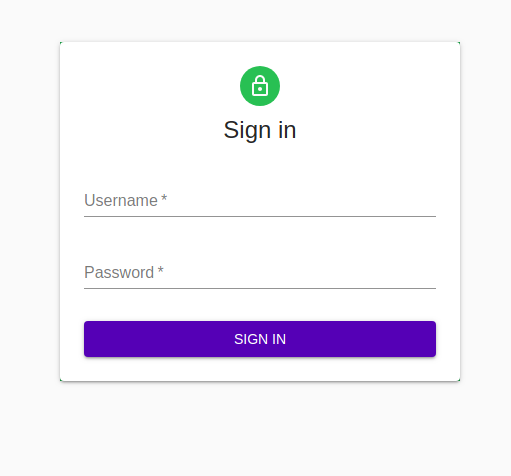


图3 登录/注册页面

* 1. 主页概览

主页概览如图4和图5主页概览所示

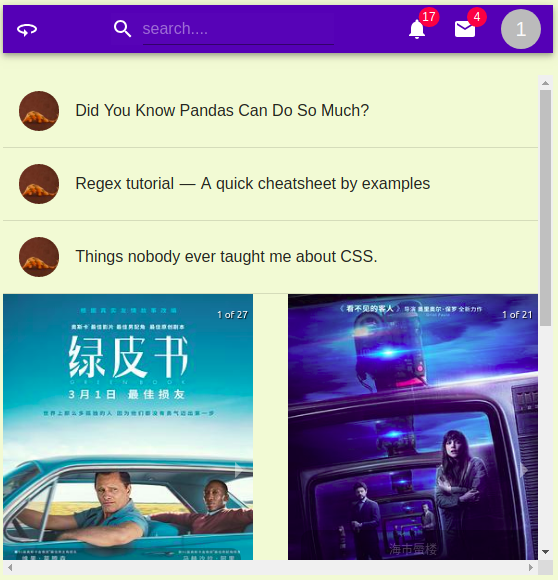


图4 主页概览

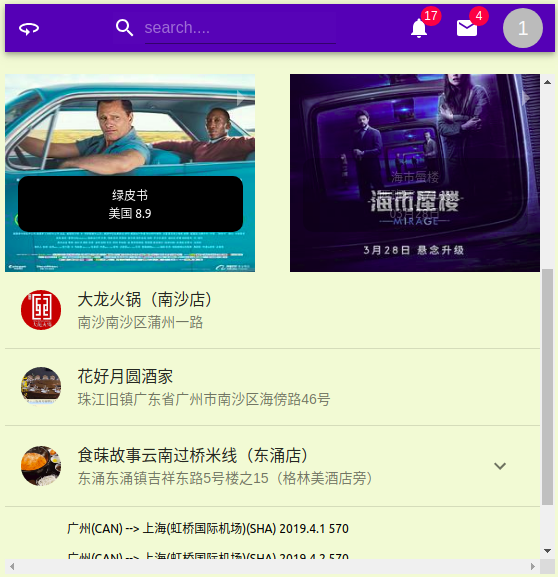


图5 主页概览

由于是Chrome插件作为前端展示，所以界面可能略为简陋。经研究，一个新的app不应该有太多的功能，当目录跳转达到三级的时候，用户就会有厌烦的感觉。新的app讲究的是简单、易上手，也很符合本文。第一栏是博客展示，它将会展示博客的标题，最多展示三条，点击对应用户头像可以跳转到对应的博客页面。第二栏是电影栏目，左边的是正在上映的电影，右边的是准备上映的电影。每个电影中点击对应电影的背景，将可以看到电影的简介，点击黑框框（黑框框展示了电影的评分、地区还有热度）将可以跳转去对应地区的购票链接。对于还没有上映的电影，点击黑框框将会跳转去电影的预告片，让用户了解电影的大概画面。第三栏是用户地区附近的美食，包括美食的题目还有地址等，右边还有一个更多的按钮，表示美团的团购信息，可以一键点击，进行购买。最后一栏是飞机票价的信息，显示的是机票路径，机票日期还有机票价格。可以一键点击，进行购买。顶头的app栏具有搜索和消息提示功能，右边还有用户信息配置功能，由于这些功能实现的比较简陋而且不是本文所研究，这里不一一阐述。

1. 总结与展望
   1. 总结

本文对爬虫这一计算机领域做了较浅的研究和分析，分别对美团、携程、meidum博客网页和豆瓣电影等网页结构进行了分析，利用XPath提取出所需要的XML节点以及使用正则筛选出自己想要的信息。为了更好的用户体验，前端使用Chrome插件作为客户端，使得用户只需要打开Chrome浏览器，就能实时看到最新的推送消息。最后，为了各个用户之间可以交流，本文增加了评论模块，使得用户之间可以互相反馈，但由于用户生态系统还没完善，本文并未对这模块进行深入实现。总的来说，本文还只是一个可运行的系统，没有经过长时间的累积，只在服务器上部署了一个月时间左右。经测试，后台系统可以撑住一秒1000左右的并发访问量，对于Python这种弱类型语言还有使用了ORM框架来说，个人感觉达到预期。当然，可圈可点的地方还是有很多的，以后本系统还会一直的运行着，并为用户提供优质的信息推送服务。

本文也有很多不足的地方，比如对于medium博客爬虫的实现，无法利用XPath获得博客发表者的头像，转而使用正则表达式获得。前端只是简单的显示逻辑，并没有涉及可配置部分。只部署在服务器上但运行情况还算良好。在UI显示方面，也没有过多使用炫酷的UI效果，使得用户使用起来可能会有点不适应。架构起来写代码也不算特别的方便。总的来说，无论是架构还是思路，都有很大的上升空间。

在完成毕业设计和毕业论文过程中，遇到问题能以积极的心态去面对，不逃避。认真思考，从问题中吸取经验教训，通过不断的努力，完成了毕业设计和毕业论文。

* 1. 展望

在毕业设计与撰写毕业论文中，从零到无学习了前端React框架和Material-UI库，前端对于我的挑战是非常大的。在后台使用SQLAlchmey数据库框架的时候，也由于没管理好数据库事务，出现了意想不到的的情况，最后经过网上搜索，从中学习了很多技术也解决了这个问题。通过利用网上资源，分析并解决问题是一名软件工程师应有的素质。

选择爬虫系统作为毕业设计的原因有两个，第一个是自己需要这样的一个软件，来帮助自己自动拉取有用的信息。第二个是因为工作需要，日后工作也比较偏向于web开发，因此借助毕业设计的机会多了解一下web开发。

软件技术日新月异，或许再过几年，本文所用的技术也就成了老一代技术了。希望自己能保持一颗主动学习的心。

最后，希望自己能通过本毕业设计，吸取教训，有则改之无则加勉，希望以后处理问题能更加严谨。也希望以后能够一直完善本系统。

参 考 文 献

柳海燕.基于Material-UI的企业微信应用的设计与实现[J].电脑知识与技术,2018,14(29):274-275.

马明阳,郭明亮,魏留强.网络爬虫的专利技术综述[J].科技视界,2018(22):12-13.

潘晓英,陈柳,余慧敏,赵逸喆,肖康泞.主题爬虫技术研究综述[J/OL].计算机应研究:1-6[2019-04-08].https://doi.org/10.19734/j.issn.1001-3695.2018.11.0790.

仝野. 基于NoSQL数据库的系统设计与开发[D].南京邮电大学,2018.

杨力.布隆算法在网络爬虫中的应用[J].电子世界,2019(03):156+158.

朱贇.Python语言的Web开发应用[J].电脑知识与技术,2017,13(32):95-96.

金诗谱.基于“7+1”模式的Nginx服务器安全加固应用研究[J].九江学院学报(自然科学版),2019(01):57-60.

Arijit Panda,Narendrakumar M. Chaudhari,Sucheta Tripathy. Genome Annotator Light (GAL): A Docker-based package for genome analysis and visualization[J]. Genomics,2019.

Kitti Klinbua. Translating TOSCA into Docker-Compose YAML File Using ANTLR[A]. The Institute of Electrical and Electronics Engineers、IEEE Beijing Section.Proceedings of 2017 IEEE 8th International Conference on Software Engineering and Service Science[C].The Institute of Electrical and Electronics Engineers、IEEE Beijing Section:IEEE BEIJING SECTION(跨国电气电子工程师学会北京分会),2017:4.

Mohamad Rotmianto,Eko Wahyudi. Developing Plugin e-DDC as an Additional Application for Senayan Library Management System with PHP Language Programming and MySQL Database[J]. Record and Library Journal,2016,2(1).

致 谢

在学士学位论文完成之前，衷心感谢綦羽老师的悉心指导以及一路上协助我完成论文的同学和朋友。

綦羽老师在生活类爬虫系统应用程序上，给了很多交互设计方面的思路，在爬虫系统上也指导了爬虫所应使用的公式和优化。在论文撰写方面，也悉心一一指出论文措辞和格式上的缺点和不足。在此，再次感谢綦羽老师的宝贵意见。

在即将毕业之际，衷心感谢大学四年以来教导过我的每一位老师，他们都让我领略到计算机的魅力以及我的知识黑洞，每一轮新课程都可以让我觉得原来计算机的逻辑原来如此。谢谢学校ACM队伍给了我训练的机会，让我谦虚起来，认真学习。感谢大学四年陪伴身边的朋友和同学，我们一起学习，一起生活，一起找工作，感谢他们的陪伴，感谢他们点缀了我的大学生活。

感谢家人对我的支持和鼓励，感谢父母的养育之恩，抚养我长大成人。

最后，衷心祝福每一位在我大学生活遇到的人，感谢你们点缀了我的生活，祝你们身体健康，工作顺利，学习愉快。

**华南农业大学**

**本科生毕业论文成绩评定表**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 学号 | 201527010414 | | | | | 姓名 | | 刘炜铭 | 专业 | 软件工程（软件工程） | | |
| 毕业论文题目 | | | 生活类爬虫系统 | | | | | | | | | |
| 指导教师评语  成绩（百分制）：　　　　　　　　　　　　　　　　　　　　指导教师签名：　　　　　　　　　　　　　　　年　　月　　日 | | | | | | | | | | | | |
| 评  阅  人  评  语  及  成  绩  评  定 | | 成绩  评定  标准 | | **评分项目** | | | | | | | **分值** | **得分** |
| 选题  质量  20% | 1 | | 专业培养目标 | | | | 5 |  |
| 2 | | 课题难易度与工作量 | | | | 10 |  |
| 3 | | 理论意义或生产实践意义 | | | | 5 |  |
| 能力  水平40% | 4 | | 查阅文献资料与综合运用知识能力 | | | | 10 |  |
| 5 | | 研究方案的设计能力 | | | | 10 |  |
| 6 | | 研究方法和手段的运用能力 | | | | 10 |  |
| 7 | | 外文应用能力 | | | | 10 |  |
| 成果  质量40% | 8 | | 写作水平与写作规范 | | | | 20 |  |
| 9 | | 研究结果的理论或实际应用价值 | | | | 20 |  |
| 评阅人评语：  成绩（百分制）：　　　　　　　　　　　　　评阅人签名：　　　　　　　　　　　　年　　月　　日 | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 答  辩委员会  意见与  成  绩  评  定 | 评价项目 | 具体要求（A级标准） | 最高分 | 评分 | | | | |
| A | B | C | D | E |
| 论文质量 | 论文（设计）结构严谨，逻辑性强；有一定的学术价值或实用价值；文字表达准确流畅；论文格式规范；图表（或图纸）规范、符合要求。 | 60 | 55-60 | 49-54 | 43-48 | 37-42 | ≤36 |
|  |  |  |  |  |
| 论文报告、讲解 | 思路清晰；概念清楚，重点（创新点）突出；语言表达准确；报告时间、节奏掌握好。 | 20 | 19-20 | 17-18 | 15-16 | 13-14 | ≤12 |
|  |  |  |  |  |
| 答辩情况 | 答辩态度认真，能准确回答问题 | 20 | 19-20 | 17-18 | 15-16 | 13-14 | ≤12 |
|  |  |  |  |  |
| 是否同意通过论文答辩（打√）   1. 同意 2. 不同意   成绩（百分制）：　　　　　　　　 答辩委员会主席（签名）：  　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　 年 月 日 | | | | | | | |

续上表：

|  |  |
| --- | --- |
| 成绩总评 | 论文总评分数：  论文成绩总评等级：　　　 学院盖章：  年 月 日 |

**注：1、**论文成绩评定等级：参考评阅人的评阅、指导教师评阅情况，结合答辩情况，建议按指导教师评分、评阅人评分、答辩评分为4∶3∶3的比例评定论文总成绩分数，然后按优（90－100分）、良（80－89分）、中（70－79分）、及格（60－69分）、不及格（<60分）给出成绩等级。**2**、论文成绩以分数的形式登记到教务管理系统。