GeekBrains

Дипломный проект

По специальности: «Программист Python. Цифровые профессии.»

Разработка программы-парсера новостных каналов

Работу выполнил:
Храпцов Роман Александрович

Сочи 2023

Тема проекта: Разработка универсальной программы парсера новостных каналов.

Цель: Целью работы является разработка модуля парсинга данных новостных каналов с официальных сайтов СМИ. А также с размещением этой информации на портале телеграм-канала.

Задачи:

1. Изучить литературу, касающуюся темы исследования.
2. Рассмотреть основные виды и методы работы синтаксических анализаторов.
3. Ознакомиться с основными принципами составления программы.
4. Изучить необходимые библиотеки.
5. Разработать и написать код программы.

Инструменты: Python 3.11, Telegram, HTTPX, Asyncio, Logging, Collection, Telethon, Utils, Config

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время развитие информационных технологий оказывает большое влияние на все области человеческой деятельности, так или иначе связанных с накоплением и обработкой информации. Информационные технологии (ИТ) — это процессы, которые используют совокупность средств и методов сбора, обработки и передачи данных (первичной информации) для получения информации нового качества о состоянии объекта, процесса или явления (информационного продукта). Информационная технология является процессом, состоящим из четко регламентированных правил выполнения операций, действий, этапов разной степени сложности над данными, хранящимися в компьютерах. [1]

Наиболее распространённым источником информации в современном мире является интернет – всемирная система объединённых компьютерных сетей. Она построена на базе протоколов TCP/IP. На основе интернета работает всемирная паутина (World Wide Web) и множество других систем передачи данных. [2]

Именно особенности Интернета определяют его огромный потенциал и все возрастающую роль в современном мире. Интернет является таким техническим средством и каналом коммуникации, который характеризуется отсутствием централизованной организационной структуры. Также этот канал характеризуется большой скоростью распространения информации. Главная отличительная особенность интернета — своевременное обновление контента. А главное требование, которое предъявляет общество к сети — это наличие актуальной информации. [2]

Постоянное обновление контента приводит к увеличению темпа роста объёма информации. По прогнозам количество данных на планете будет как минимум удваиваться каждые два года. [3]

Сегодня уже практически невозможно представить Интернет без информационно-поисковых систем: поисковые роботы, парсеры, скраперы. Названий много, но цель одна – сбор информации.

В связи с этим появляется необходимость внедрения специальных информационно-поисковых систем для поиска нужной информации. Информационно-поисковая система – это программный комплекс, обеспечивающий поиск и отбор необходимых данных в специальной базе с описаниями источников информации на основе информационно-поискового языка и соответствующих правил поиска.

Главной задачей любой информационно-поисковой системы является поиск информации, релевантной информационным потребностям пользователя.

Релевантность – это соответствие результатов поиска сформулированному запросу. Такими информационно-поисковыми программами являются поисковики и парсеры.

Парсинг (от англ. «parsing» - «анализ, разбор») – это линейное сопоставление последовательности слов с правилами языка. Понятие «язык» рассматривается в самом широком контексте. Это может быть человеческий язык, используемый для коммуникации людей. А может и формализированный язык, в частности, любой язык программирования.

Парсинг сайтов – последовательный синтаксический анализ информации, размещённой на интернет-страницах. Парсинг сайтов является наиболее эффективным решением для автоматизации сбора и изменения информации.

Для написания парсеров подходят любые языки программирования, на которых создаются программы для работы со Всемирной Паутиной. Веб-приложения для парсинга обычно пишут на C++, Ruby, Python, PHP. [4]

По сравнению с человеком, компьютерная программа-парсер:

– Способна быстро проанализировать более тысячи веб-страниц (за несколько секунд);

– Безошибочно отобрать нужную информацию;

– Представит конечные данные в необходимом виде (база данных или электронная таблица).

По сравнению с поисковиком, компьютерная программа-парсер:

– Обладает более высокой скоростью подбора требуемой информации;

– Отличается повышенной точностью: содержит конкретные данные по заданным критериям поиска;

– Может быть универсально настроена под требования и запросы любого пользователя.

В этой связи в данной работе была поставлена задача создания парсера на языке Python.

Главные требования к разрабатываемой программе:

1. Получение актуальных данных;
2. Запись полученной информации в электронную таблицу;
3. Формирование баз данных на основании полученной информации;
4. Размещение информации на портале.

1 ПРОГРАММЫ И МЕТОДЫ ПОИСКА И СБОРА ДАННЫХ

1.1 Информационные технологии

Как ранее упоминалось информационные технологии (ИТ, от англ. information technology, IT) – совокупность средств вычислительной техники, телекоммуникационного оборудования, каналов передачи данных и информационных систем, средств коммутации и управления информационными потоками, а также организационных структур, правовых и нормативных механизмов, обеспечивающих их эффективное функционирование. [5]

Анализируя общее понимание информационных технологий, можно выделить, что ИТ охватывает все области передачи, хранения, восприятия информации. Но в большинстве случаев ИТ ассоциируются с компьютерными технологиями, так как возникновения компьютеров вывело информационные технологии на новый уровень развития.

То есть с появлением персонального компьютера начался новый этап развития информационных технологий. Главной целью ИТ – удовлетворение персональных информационных потребностей человека, как для профессиональной сферы, так и для повседневного пользования.

Существуют определенные этапы развития информационных технологий:

1 этап (с начала 60 – 70- е гг.) – основным направлением являлась автоматизация рутинных действий человека, обработка данных в вычислительных центрах в режиме коллективного использования, в условиях ограниченных возможностей аппаратных средств, характеризуется проблемой обработки больших объемов информации.

2 этап (70-е годы) – появления персональных компьютеров, распространение ЭВМ серии IBM/360, ориентация на индивидуального

пользователя, использование централизованных обработок данных, так и децентрализованного, который базируется на решении локальных задач и работе с локальными базами данных на рабочем месте пользователя.

3 этап (80-е годы) – компьютером начинаю пользоваться непрофессионалы, создание информационных технологий, направленных на решение задач, одна из которых максимально удовлетворить потребность пользователя и создания соответствующего интерфейса работы в компьютерной среде.

4 этап (90-е годы) - создание современной технологии междуорганизационных связей и информационных систем, организация защиты и безопасности информации, организация доступа к стратегической информации, выработка соглашений и установление стандартов, протоколов для компьютерной связи.

Цель информационной технологии — производство информации для её анализа человеком и принятия на его основе решения по выполнению какого-либо действия. [6]

1.2 Интернет

Интернет - глобальная сеть передачи данных, связывающая информационные системы и сети электросвязи различных стран посредством глобального адресного пространства, основанном на использовании протоколов TCP/IP. Обеспечивает доступ к вычислительным ресурсам подключенных к сетям компьютеров, к информационному наполнению всемирной паутины, к электронной почте, базам данных, телеконференциям и к ряду других сервисов. [5]

Интернет сейчас самый большой источник информации, и уникальную информацию найти достаточно сложно. В виду того, что количество

информации постоянно растёт. Для поиска уникального контента разрабатывают аппаратно-программное обеспечение.

1.3 Анализ источников по преобразованию данных

Перед разработкой универсального парсера требуется обосновать технологию парсинга, как эффективную технологию для преобразования данных.

В книге Б. Бенгфорт, Р. Билбро, Т. Охедо «Прикладной анализ текстовых данных на Python. Машинное обучение и создание приложений обработки естественного языка» даётся чёткое понимание технологии парсинга для чего нужны парсеры какие входные параметры.

Для лучшего понимания технологии парсинга была проанализирована статья «Методы парсинга сайтов» в которой описаны основные методы парсинга и методология парсинга в целом. Рассмотрены основные этапы работы парсера.

На основе анализа информации литературных и интернет источников были выделены особенности, которые будут учтены в работе.

1.4 Определение парсера. Используемые языки программирования

Парсер – скрипт или программа, которые используются для сбора информации с сайтов для последующей обработки и представления. Он может быть написан на любом языке работающим с веб контентом.

Проще всего писать парсер на языках высокого уровня, так как он уже содержит модули или библиотеки работы с веб контентом. Высокоуровневый язык программирования — язык программирования, разработанный для быстроты и удобства использования программистом. Основная черта

высокоуровневых языков — это абстракция, то есть введение смысловых конструкций, кратко описывающих такие структуры данных и операции над ними, описания которых на машинном коде (или другом низкоуровневом языке программирования) очень длинны и сложны для понимания. [7]

1.5 Области применения парсеров

Об использовании парсинга говорят мало. Это подтверждает статистика запросов в Яндексе. Парсинг может быть эффективен во многих сферах. Можно обрабатывать данные веб-страниц интернет-магазинов, форумов, блогов и других интернет ресурсов, а также файлов различных форматов.

Данные в сети интернет расположены на веб-сайтах и представлены для человека в виде некоторого набора графических элементов, текста, изображений. Человек осуществляет парсинг каждый день: ищет номер телефона на веб-страничке, нужное изображение, просматривает товары в интернет-магазине.

Однако способности человека ограничены. Поиск больше нескольких десятков номеров на сайте может стать современной пыткой.

А если необходимо найти сотни и тысячи номеров, адресов страниц в соцсетях на сотнях веб-страниц по определенным условиям и запросам? Вручную нереально освоить такой объем информации. Тогда знающие люди используют специальные программы–парсеры. Также какая-то информация может быть скрыта от глаз пользователя, но она есть в коде веб-страницы.

Специальные программы анализируют код страницы с помощью различных алгоритмов от совсем простых (которые может написать начинающий программист) до сложнейших статистических моделей с использованием теории хаоса и нейронных сетей.

Парсеры вытаскивают нужную информацию, даже если владелец информации не хотел ею делиться. На многих сайтах номера телефонов отображаются не цифрами, а картинкой. Но хороший парсер справиться и с таким препятствием.

Парсинг имеет сомнительную репутацию, так как часто его используют для составления спам-баз. Вспомните, как после размещения резюме на портале интернет-рекрутмента, всю следующую неделю вам постоянно звонили сомнительные организации и предлагали работу. Фирмы получили ваш номер и другие данные с помощью парсера.

Зато парсинг любят маркетологи и предприниматели. Они ищут клиентов в соцсетях, на тематических форумах, торговых площадках, анализируя страницы, хэштеги и прочие данные. Создают себе свою базу клиентов, которую могут собирать годами.

А потом можно делать рекламу не по безликим настройкам таргета, а уже по готовой базе живых людей.

Таким образом, парсер — это программа, которая анализирует данные с интернет ресурсов и систематизирует их в файл.

Парсер нужен не только, чтобы «заглядывать» в окна конкурентам. Парсер поможет оптимизировать свой сайт: найти битые ссылки, пробелы в тексте, отсутствие изображений. Сервис соотнесёт информацию о наличии товара на складе и информацию на сайте. И информация будет постоянно обновляться.

Для парсинга собственного сайта или соцсетей можно обратиться к специалисту. Но не все начинающие предприниматели готовы платить за это. Тогда можно выбрать простую программу для парсинга, которая рассчитана на людей без навыков программирования.

Как правило, используют SEO-парсеры для анализа собственного сайта. С задачей хорошо справится сервис по продвижению сайтов. Такой парсер не только проверит внутренние, внешние и технические характеристики веб-страницы, но и даст рекомендации, как исправить.

Если вам не нужен такой полный анализ, установите специальное расширение для браузера. Это самый простой вид парсеров. Например, расширение Parsers или Scraper.

Парсеры очень часто используют в следующих областях:

- там, где информация быстро теряет свою актуальность и уже неприменима спустя несколько минут;

- там, где ручное копирование не представляется возможным или требует колоссальных человеческих затрат (например, для отображения курса валют, стоимости ценных бумаг или инсайдерской информации);

- там, где идёт полное или частичное копирование материалов сайта с последующим размещением этих материалов на своих ресурсах. [7]

Таким образом, можно представить примерную область применения парсеров:

- маркетинговые исследования;

- мониторинг СМИ в реальном времени;

- обновление новостных порталов (при этом текст может быть предварительно пропущен через синонимайзер или обработан рерайтером для повышения уникальности);

- анализ общественного мнения;

- автоматическое ценообразование на базе анализа цен конкурентов;

- построение списка потенциальных пользователей на базе информации о пользователях ресурсов конкурентов;

- разработка мобильных приложений;

- анализ социальных связей;

- сбор и обработка спортивной статистики;

- анализ документов в области судопроизводства;

- парсингу также подвергаются сайты с отзывами о кино и книгах, а также сайты с рецептами, текстами песен и стихов. [8]

1.6 Этапы парсинга

Парсинг html-страницы представляет из себя процесс, который можно разбить на три этапа:

1 этап: получение исходного кода веб-страницы.

В разных языках для этого предусмотрены различные способы, например, в языке программирования Python чаще всего используется библиотека «reqests» которая сохраняет дерево сайта в переменную.

2 этап: извлечение из html-кода необходимых данных.

Получив страницу, необходимо её обработать:

- отделить обычный текст от гипертекстовой разметки;

- выстроить иерархическое дерево элементов документа;

- корректно среагировать на не валидный код;

- сделать выборку нужной информации со страницы.

Для этого можно использовать регулярные выражения или специализированные библиотеки. [7]

3 этап: фиксация результата.

Благополучно обработав данные на странице, требуется их сохранить в необходимом виде для последующей обработки. Существует несколько способов манипулирования полученной информацией:

- внести в базу данных;

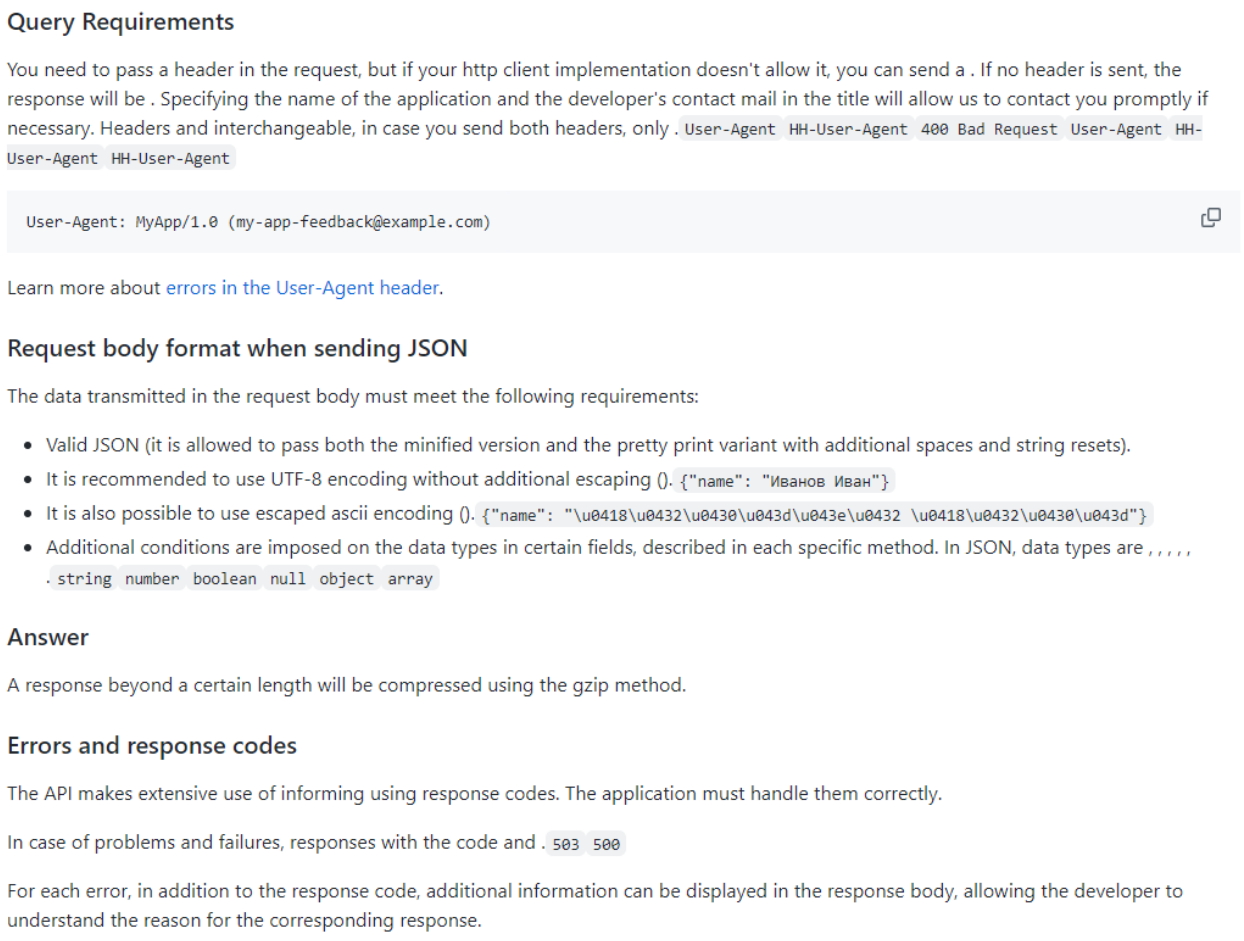
- записать в CSV-файл;

- построить иерархические JSON-структуры;
- сконвертировать в excel-таблицу;
- сгенерировать динамический rss-поток. [8]

Вот несколько примеров работы парсеров:

Поиск резюме

У известного российского интернет-рекрутмента есть API, но пока его функционал не решает все потребности клиентов, поэтому многие обращаются к парсерам. (Хоть и администрация портала против парсинга данных) (рисунок 1).



Query Requirements

You need to pass a header in the request, but if your http client implementation doesn't allow it, you can send a . If no header is sent, the response will be . Specifying the name of the application and the developer's contact mail in the title will allow us to contact you promptly if necessary. Headers and interchangeable, in case you send both headers, only . `User-Agent` `HH-User-Agent` `400 Bad Request` `User-Agent` `HH-User-Agent` `HH-User-Agent`

```
User-Agent: MyApp/1.0 (my-app-feedback@example.com)
```

Learn more about errors in the User-Agent header.

Request body format when sending JSON

The data transmitted in the request body must meet the following requirements:

- Valid JSON (it is allowed to pass both the minified version and the pretty print variant with additional spaces and string resets).
- It is recommended to use UTF-8 encoding without additional escaping (). `{"name": "Иванов Иван"}`
- It is also possible to use escaped ascii encoding (). `{"name": "\u0418\u0432\u0430\u043d\u043e\u0432 \u0418\u0432\u0430\u043d"}`
- Additional conditions are imposed on the data types in certain fields, described in each specific method. In JSON, data types are , , , , , . `string` `number` `boolean` `null` `object` `array`

Answer

A response beyond a certain length will be compressed using the gzip method.

Errors and response codes

The API makes extensive use of informing using response codes. The application must handle them correctly.

In case of problems and failures, responses with the code and . `503` `500`

For each error, in addition to the response code, additional information can be displayed in the response body, allowing the developer to understand the reason for the corresponding response.

Рисунок 1 – API запрос

Например, нужно найти на сайте с вакансиями всех программистов определенного возраста с образованием и определенным стажем работы. И потом вытащить их ФИО и номера телефонов и сохранить это в таблицу.

Работодатель и соискатель смогут находить подходящие варианты без ручного поиска. Главное, не нарушать нормальную активность, иначе ваш аккаунт могут заблокировать. Чтобы избежать блокировки, имитируйте скорость человеческой активности при работе с парсером.

Сбор контактной информации

Парсер поможет составить списки контактов с дополнительной информацией: номера телефонов, почта, адрес. Данные потенциальных клиентов бесценны для бизнеса. Можно рассылать выгодные предложения на почту, оповещать об акциях по смс, сегментировать аудиторию.

Обычно контактную информацию собирают с соцсетей. В связи с их популярностью и эффективностью таргетированной рекламы есть много специальных парсеров. Самые популярные для Instagram: Zengram, Tooligram, Pepper.Ninja. Для работы с Вконтакте используют: TargetHunter, Церебро Таргет, Segmento Target.

Работа агентства таргетированной рекламы неизбежно сопряжена с работой в сервисах парсинга. Если говорить о работе с таргетированной рекламой во ВКонтакте, то функционал их рекламного кабинета довольно скудный, поэтому чтобы более эффективно расходовать бюджет и показывать рекламу только нужным нам пользователям, необходимо использовать парсеры (рисунок 2).

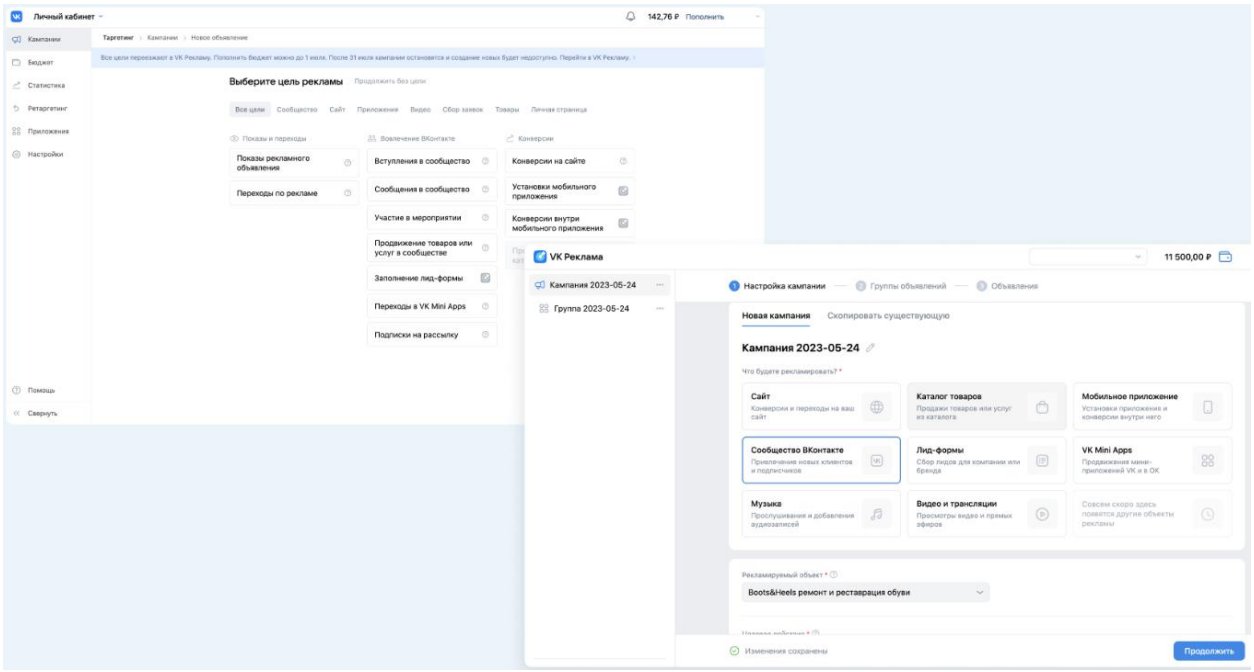


Рисунок 2 – Рекламный кабинет

Парсинг контента в соцсетях

Можно быть самым крутым писателем, но какой в этом смысл, если твои посты не набирают лайки и репосты. Парсер поможет понять, какой контент в сети вызывает отклик у аудитории.

Особенно парсинг помогает начинающим блогерам: можно сразу составить план из самых интересных тем для потенциальной целевой аудитории блога.

Однако сложно завоевать популярность, генерируя вторичный контент. Эксперименты с темами могут оказаться как провальными, так и вывести ваш блог в топ. Просто соблюдайте баланс между «безопасными» темами и новыми форматами. [16]

2 ПАРСИНГ. СТРУКТУРА, КОМПОНЕНТЫ (ФРЕЙМВОРКИ)

2.1 Аппаратно-программное обеспечение для поиска информации

Программирование можно сравнить с творчеством. Код можно писать по-разному: быстро или медленно, понятно и читабельно или плохо и не поддерживаемо. Это зависит от предпочтений, навыков и бэкграунда. Использование фреймворков (англ. framework — «каркас, структура») помогает уравнять эти различия. Программист использует готовые шаблоны, дополняет их своим кодом и с их помощью вносит определённую логику решения задач и проблем бизнеса. Все эти библиотеке хранятся на специальных репозиториях. Одним из таких хранилищь является PyPi.org.

Python Package Index, сокращенно PyPI является официальным сторонний репозиторий программного обеспечения для Python, которое могут использовать все. Все, кто программируют на Python, хотя бы раз устанавливали и использовали модули. Он аналогичен репозиторию CPAN для Perl. PyPI управляется благотворительной организацией Python Software Foundation. Некоторые менеджеры пакетов, включая pip, используют PyPI в качестве источника по умолчанию для пакетов и их зависимостей.

По состоянию на 17 января 2022 года через PyPI можно получить доступ к более чем 350 000 пакетов Python.

В мае 2023 года доступно более 450 000 пакетов Python.

PyPI в основном размещает пакеты Python в виде архивов, называемых sdists (дистрибутивы исходного кода) или предварительно скомпилированными «колесами».

PyPI в качестве индекса позволяет пользователям искать пакеты по ключевым словам или по фильтрам по их метаданным, таким как лицензия свободного программного обеспечения или совместимость с POSIX. Одна запись в PyPI может хранить, помимо пакета и его метаданных, предыдущие

выпуски пакета, предварительно скомпилированные колеса (например, содержащие библиотеки DLL в Windows), а также различные формы для разных операционных систем и версий Python (рисунок 3).

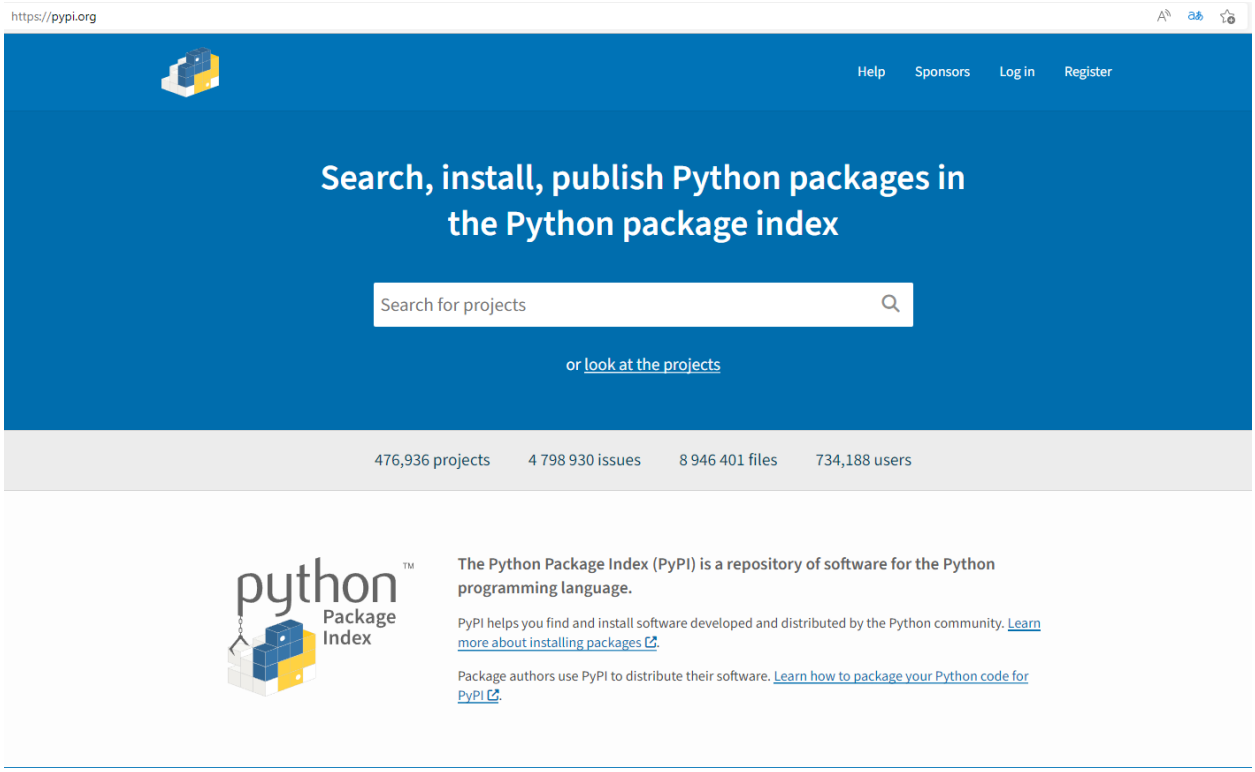


Рисунок 3 – Страница сайта PyPi.org

Фреймворки в программировании подходят для решения многих задач: создания интернет-магазинов, блогов, приложений с множеством тысяч активных пользователей. Любая CMS-система — это framework, который достаточно наполнить контентом, и простой интернет-магазин готов.

При разработке фреймворка программист учитывает его структуру и ограничения. Во фреймворке есть реализованные классы, предопределённые переменные, константы и готовые решения отдельных функциональностей: валидация запросов, работа с БД, авторизация, работа с формами. Как в тетрисе, во фреймворке нужно взять нужные блоки, связать их и использовать.

Фреймворки подходят для решения многих задач: автоматизации, создания MVP (англ. minimum viable product — «минимально жизнеспособный продукт») или проверки идей. Обычно перед программистом стоят следующие задачи: уменьшить время разработки и последующей поддержки, обеспечить стабильность и защищённость приложения. Во всех этих случаях framework приходит на помощь: он протестирован, позволяет решать любую бизнес задачу и выдерживать большие нагрузки. В отличие от CMS, фреймворк — это низкоуровневое решение, то есть он содержит некоторый базовый функционал, и многие компоненты придётся дорабатывать вручную (рисунок 4).

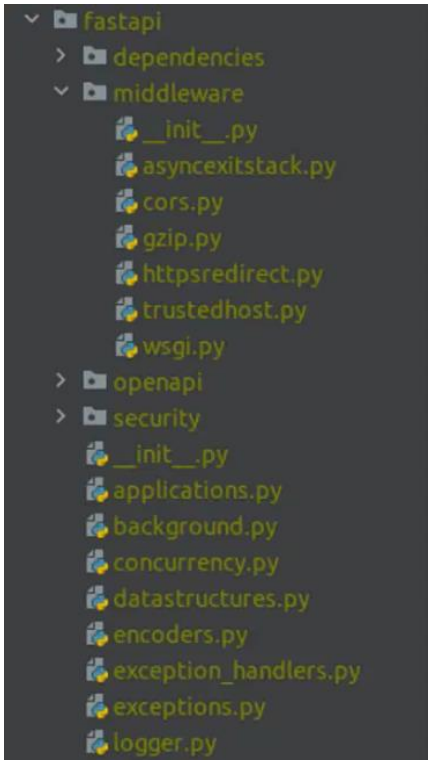


Рисунок 4 - Структура фреймворка на примере FastAPI

Для чего нужен фреймворк:

– Увеличить скорость разработки.

За счёт переиспользования готовых модулей фреймворк уменьшает время релиза новой функции и разработки сервиса или приложения. Если бы пришлось разрабатывать с нуля, то затраченное время было бы колоссальным.

Это полезная функция для начинающих программистов. Если использовать готовый шаблон, лишь дополняя или связывая некоторые функциональные блоки, и наполнять его контентом, это уменьшает шансы ошибиться. На выходе получается «чистый» код, отвечающий стандартам разработки.

– Упростить работу.

Фреймворки похожи, поэтому программист сможет разобраться с любым проектом, созданным на базе фреймворка, если ранее работал с ним или с другим аналогичным.

Обычно фреймворк — это открытый-проект, то есть если чего-то не хватает, то всегда можно дополнить, а сообщество специалистов поможет советом. Над открытым проектом работают, например, в системах хранения GitHub или GitLab, и здесь нет каких-то строгих правил — дополнять и улучшать фреймворк может любой специалист.

– Обеспечить безопасность.

Фреймворки защищены и протестированы. Самые распространённые уязвимости или способы получения конфиденциальных данных — SQL-инъекции, XSS-атаки, SSRF, брутфорс — учтены и не представляют угрозы. По уровню безопасности фреймворки превосходят самописные решения и помогают обезопасить приложения от взлома.

2.2 Библиотеки используемые при написании парсеров

Selenium

Одним из средств эмуляции поведения пользователя в браузере является Selenium. Selenium – это проект, в рамках которого разрабатывается серия программных продуктов с открытым исходным кодом: [9]

- Selenium WebDriver,
- Selenium RC,
- Selenium Grid,
- Selenium IDE.

Selenium WebDriver – это программная библиотека для управления браузерами.

Часто употребляется также более короткое название WebDriver.

Это целое семейство драйверов для различных браузеров, а также набор клиентских библиотек на разных языках, позволяющих работать с этими драйверами (рисунок 5).

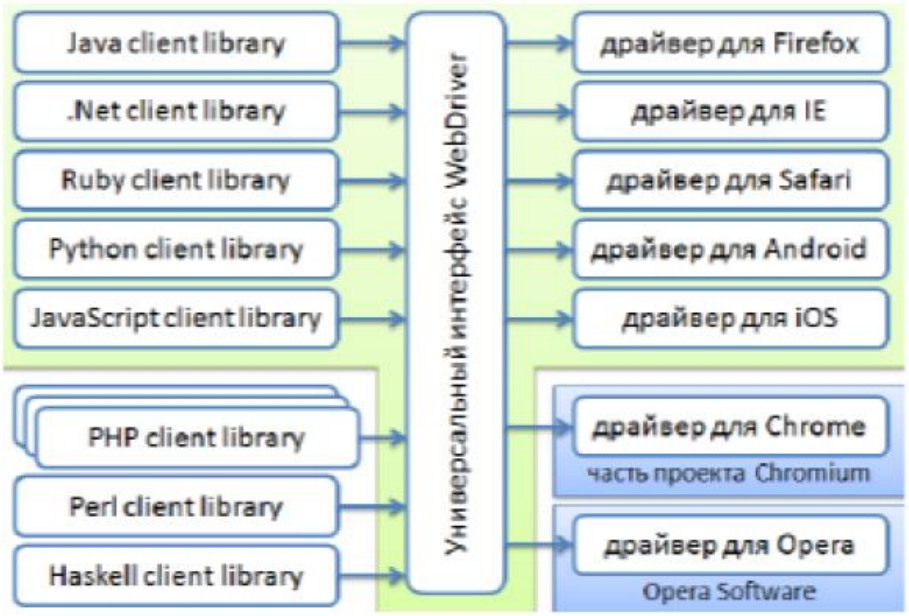


Рисунок 5 – Библиотека Selenium WebDriver

Selenium RC – это предыдущая версия библиотеки для управления браузерами. Аббревиатура RC в названии этого продукта расшифровывается как Remote Control, то есть это средство для «удалённого» управления браузером.

Selenium Grid – это кластер, состоящий из нескольких Selenium-серверов. Он предназначен для организации распределённой сети, позволяющей параллельно запускать много браузеров на большом количестве машин. Selenium Grid имеет топологию «звезда», то есть в его составе имеется выделенный сервер, который носит название «хаб» или «коммутатор», а остальные сервера называются «ноды» или «узлы» (рисунок 6).

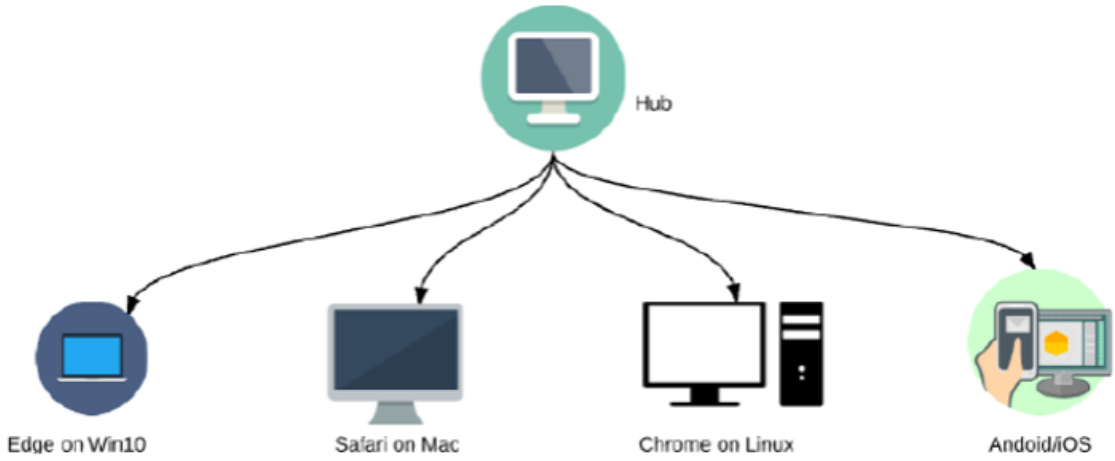


Рисунок 6 – Библиотека Selenium Grid

Сеть может быть гетерогенной, то есть коммутатор и узлы могут работать под управлением разных операционных систем, на них могут быть установлены разные браузеры. Одна из задач Selenium Grid заключается в том, чтобы «подбирать» подходящий узел, когда во время старта браузера указываются требования к нему – тип браузера, версия, операционная система, архитектура процессора и ряд других атрибутов.

Selenium IDE – плагин к браузеру Firefox, который может записывать действия

пользователя, воспроизводить их, а также генерировать код для WebDriver или Selenium RC, в котором выполняются те же самые действия. [9]

HTTPX

HTTPX — это современный HTTP-клиент для Python, целью которого является предоставление более приятных и мощных возможностей для выполнения HTTP-запросов. Он был создан для добавления поддержки асинхронного программирования и быстро завоевал популярность в сообществе Python благодаря многофункциональному API и хорошей производительности. Для наглядности мы сравним HTTPX с другим популярным HTTP-клиентом для Python, Requests. [10]

HTTPX — это более новый и многофункциональный HTTP-клиент для Python, чем популярная библиотека Requests. Вот некоторые из ключевых различий между HTTPX и запросами:

Асинхронная совместимость: HTTPX имеет как асинхронную, так и синхронную совместимость, что означает, что вы можете использовать его как в асинхронных, так и в синхронных программах. Библиотека запросов, с другой стороны, является только синхронной.

Поддержка HTTP/2: HTTPX поддерживает HTTP/2, последнюю версию протокола HTTP. В библиотеке Requests эта функция отсутствует.

Автоматическое декодирование: HTTPX включает автоматическое декодирование JSON и других распространенных форматов, что может сэкономить время и усилия разработчиков при работе с ответами, содержащими структурированные данные.

Размер: HTTPX больше, чем запросы, что может быть рассмотрено для приложений, где размер является проблемой.

Производительность: в большинстве случаев HTTPX имеет лучшую производительность по сравнению с запросами.

Ниже приведены синтаксические различия между ними. Следующий код использует библиотеку Requests для отправки запроса GET (рисунок 7):

```
import requests
response = requests.get("https://example.com")
print(response.text)
```

Рисунок 7 – Пример кода

В эквивалентном коде используется библиотека HTTPX (рисунок 8):

```
import httpx
response = httpx.get("https://example.com")
print(response.text)
```

Рисунок 8 – Пример кода

Эти две библиотеки используются практически одинаково в простых синхронных ситуациях. Важным отличием является то, что HTTPX позволяет писать асинхронный код следующим образом (рисунок 9):

```python
import httpx
import asyncio

async def main():
    async with httpx.AsyncClient() as client:
        response = await client.get("https://example.com")
        print(response.text)

asyncio.run(main())
```

Рисунок 9 – Пример кода

Как видите, синтаксис для выполнения запроса с помощью HTTPX немного более подробен из-за синтаксиса Python async/await. Однако это небольшая цена за возможность делать асинхронные HTTP-запросы.

HTTPX также включает автоматическое декодирование JSON и других распространенных форматов, что может сэкономить время и усилия разработчиков при работе с ответами, содержащими структурированные данные.

Несколько функций выделяют HTTPX как мощный и гибкий HTTP-клиент для Python. Эти функции включают поддержку синтаксиса async/await, совместимость с синхронизацией, протокол HTTP/2 и поддержку потокового ответа.

Если вам нужно загрузить большие объемы данных, вы можете использовать потоковые ответы, которые не загружают в память весь текст ответа сразу. К счастью, HTTPX имеет встроенную поддержку потоковых ответов.

Requests

Requests — это широко используемая библиотека Python, особенно для выполнения HTTP-запросов. Его простой API и долгое существование делают

его популярным выбором для многих разработчиков. Чтобы узнать больше о запросах, прочтите подробное руководство по библиотеке запросов Python (рисунок 10). [11]

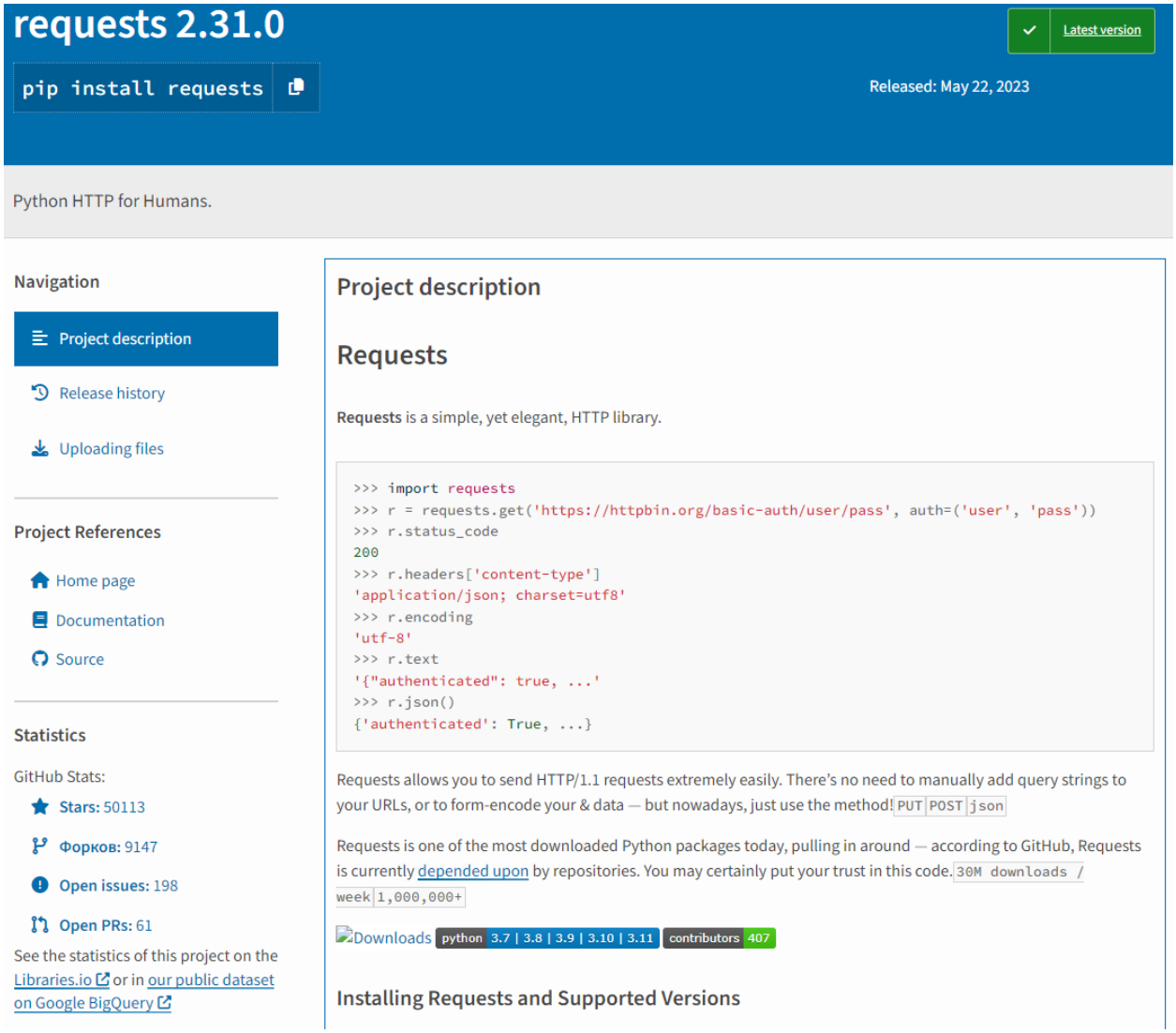


Рисунок 10 – Библиотека Requests

Это сторонняя альтернатива стандартным «urllib», «urllib2» и «urllib3», поскольку они могут сбивать с толку и часто должны использоваться вместе. Requests в Python значительно упрощает процесс отправки HTTP-запросов по назначению.

Обучение отправке запросов на Python является частью пути любого начинающего разработчика.

По сути, это широко используемая библиотека для выполнения HTTP-запросов в Python, предназначенная для предоставления вам упрощенного способа взаимодействия с API и веб-сервисами, а также для очистки веб-сайтов и выполнения других задач на основе HTTP.

Его интуитивно понятный API позволяет легко отправлять HTTP-запросы, в то же время поддерживая различные методы HTTP, включая GET, PUT, DELETE, HEAD, OPTIONS и PATCH.

Модуль Python Requests — это библиотека, которая стремится быть максимально простой в использовании и анализе. Стандартные HTTP-библиотеки Python сложны в использовании, анализе и часто требуют значительно большего количества операторов для выполнения того же самого. Давайте посмотрим на пример Urllib3 и Requests (рисунок 11):

```
#!/usr/bin/env python# -*- coding: utf-8 -*-
import requests
r = requests.get('https://api.github.com', auth=('user', 'pass'))
print r.status_codeprint r.headers['content-type']
# ------# 200# 'application/json'
```

Рисунок 11 – Пример кода

Requests не только уменьшает количество необходимых операторов, но и значительно упрощают понимание и отладку кода даже для неподготовленного глаза.

Как видно, Requests заметно эффективнее любой стандартной библиотеки Python, и это не случайно. Requests разрабатывалась с учетом нескольких идиом PEP 20.

Сбор данных с помощью API

API (от англ. application programming interface) – это интерфейс взаимодействия между сайтом и сторонними программами и серверами, набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах.

Вход и регистрация на разных онлайн-сервисах или платформах, осуществляемые через аккаунты в социальных сетях, является использованием API. В данном случае сервисы или приложения используют базы данных социальных сетей. При этом сервис может получать информацию о пользователе и манипулировать ею в своих целях.

Open API — система для разработчиков сторонних сайтов, которая предоставляет возможность легко авторизовать пользователей социальных сетей на сайте. Кроме этого, с согласия пользователей, можно получить доступ к информации об их друзьях, фотографиях, аудиозаписях, видеороликах и прочих данных для более глубокой интеграции с проектом.

Несмотря на все удобство использования API, существует одно ограничение - социальная сеть не может отдавать все данные, которые видны пользователям в интерфейсе. Эти ограничения имеют две причины:

- социальные сети стараются сохранять приватность своих пользователей;

- некоторые функции слишком сильно нагружают серверную часть приложения.

Для того чтобы преодолеть это ограничение следует использовать такой механизм как парсинг веб-сайта.

Сбор данных с помощью семантического разбора веб-страниц

Для парсинга html наиболее распространены следующие варианты:

- Регулярные выражения. Они являются наиболее универсальным и настраиваемым средством семантического разбора. Однако использовать исключительно их – довольно сложная задача для разработчика системы, из-за того, что потребуется очень сильно специализировать каждое регулярное выражение, а они создают дополнительную нагрузку на ОС.

- BeatifulSoup, lxml – наиболее популярные библиотеки для парсинга html-страниц и выбор одной из них обусловлен личными предпочтениями разработчика. Кроме того, эти библиотеки тесно связаны: BeautifulSoup стал использовать lxml в качестве внутреннего парсера для ускорения, а в lxml был добавлен модуль soupparser.

2.3 Аналоги парсеров

В настоящее время в сети интернет доступны два типа парсеров:

1) Услуга парсинга интересующих ресурсов.

2) Использование универсальных программ-парсеров.

В первом случае заказчик обращается с запросом на сбор необходимых данных. В результате он получает искомую информацию в требуемом виде и формате.

Достоинства:

- возможность получить большое колличество записей; [7]

- парсинг любого веб-ресурса;

- нет необходимости проходить регистрацию на сайте и использовать VPN;

- предоставление информации в удобном формате.

Недостатки:

- высокая стоимость оказываемой услуги парсинга;

- длительное время на сбор данных;

- дополнительные расходы за необходимость дополнения или обновление данных.

В случае приобретения лицензии на использование готовой универсальной парсер-программы заказчик получает возможность парсить ресурсы различного рода.

Примером таких готовых парсеров являются X-Parser Light и uParser.

Достоинства:

- автоматическое распознавание тегов на страницах любого сайта;

- наличие редактора для ручной проверки отобранного контента при помощи менеджера обработки контента.

- возможность парсить контент даже при отсутствии тегов и с формированием базовой разметки на базе исходной;

- возможность фильтровать статьи и отдельные абзацы по собственным фильтрам на стадии сбора контента;

- возможность проверки статей на наличие ключевой информации в теле статьи;

- использование доступных поисковых систем;

- возможность парсить контент на любых языках.

Недостатки:

- отсутствует возможность интеграции в программу или на сервер;

- достаточно высокая стоимость.

Однако помимо указанных двух способов сбора данных при помощи семантического разбора веб-страниц существует альтернативный вариант, заключающийся в разработке собственной программы-парсера. При этом

заказчик формирует требования на источники информации и её форму записи после процедуры парсинга.

В данном случае сочетаются сохранение достоинств первых двух типов и избавление от их недостатков. Тем не менее, присутствуют свои минусы:

- наличие более точной настройки программы;
- чёткое определение источников информации.

Исходя из вышесказанного можно сделать вывод, что при необходимости сбора информации нужно опираться на цели и требования заказчика. Не менее важным фактором выбора служит анализ достоинств и недостатков представленных способов парсинга сайтов.

В результате чего можно сделать следующие выводы: в ходе работы были проанализированы и изучены различные методы поиска и сбора информации. Каждый из представленных способов имеет свои достоинства, недостатки и особенности работы. Парсинг сайтов оказался самым оптимальным методом сбора и обработки информации, так как он совмещает достоинства представленных методов.

При этом создание собственной программы-парсера помогает заказчику сэкономить значительное количество средств. Что касается временных затрат, на разработку парсера может уйти до 10 дней, однако обновление информации будет осуществляться мгновенно.

3 РАЗРАБОТКА ПАРСЕР-ПРОГРАММЫ

3.1 Техническое задание

1 Назначение и цели.

Необходимо разработать код для программы-парсера. Написанная программа должна анализировать требуемые источники и осуществлять сбор актуальных данных.

2 Запись результатов.

Запись и сохранение результатов парсинга данных должно осуществляться на портал.

3 Источники данных.

Источниками данных являются новостные порталы.

3.2 Компьютерная программа

Термину «компьютерная программа» соответствует два определения. Во-первых, это комбинация компьютерных инструкций и данных, позволяющая аппаратному обеспечению вычислительной системы выполнять вычисления или функции управления. Во-вторых, это синтаксическая единица, которая соответствует правилам определённого языка программирования. Она состоит из определений и операторов или инструкций, необходимых для определённой функции, задачи или решения проблемы. [17]

В состав программы может входить как машинный код, исполняемый процессором для достижения некоторой цели, так и необходимые для этого данные. Отличительной особенностью программы является её нахождение в памяти и исполнение процессором.

3.3 Создание программы

Образ программы хранится на компьютере в виде исполняемого модуля. Он представляет собой отдельный файл или группу файлов. Из этого образа исполняемая программа в оперативной памяти может быть построена программным загрузчиком.

Процесс разработки программного обеспечения состоит из нескольких этапов, из которых лишь непосредственное создание программного кода носит название «программирование».

Запись исходных текстов программ при помощи языков программирования облегчает понимание и редактирование кода. Программисту помогают комментарии, допустимые в синтаксисе большинства языков. Для выполнения на компьютере готовый текст программы преобразуется (компилируется) в машинный код.

Некоторые языки программирования позволяют совершать динамическую компиляцию. Это означает, что можно обойтись без предварительной компиляции программы, а сразу перевести её в инструкции машинного кода непосредственно во время исполнения. Этот процесс позволяет добиться большей переносимости программ между разными аппаратными и программными платформами при сохранении многих плюсов компиляции.

Интерпретируемые программы, для которых, не применяется процесс компиляции и которые, интерпретируются операционной системой или специальными программами-интерпретаторами, называются скриптами или «сценариями».

Исходные тексты компьютерных программ в большинстве языков программирования содержат заложенный алгоритм. Подобный подход программировании называется императивным.

Также применяются и другие методологии. Описание исходных и требуемых характеристик, обрабатываемых данных и предоставление выбора

подходящего алгоритма решения специализированной программе-интерпретатору называется декларативным программированием. К нему относятся функциональное и логическое виды программирования.

Программы могут создаваться в текстовом виде и визуально. В первом случае исходный код набирается вручную. Во втором функциональность программы задаётся с помощью элементов графического интерфейса пользователя, а текст программы генерируется автоматически. При таком подходе код может быть доступен для изменения вручную или полностью скрыт от программиста [12].

3.4 Выбор языка программирования

Повторюсь приложения для парсинга обычно пишут на высокоуровневых языках, таких как C++, Ruby, Python, PHP. В ходе работы был проведено тщательное исследование достоинств и недостатков каждого из представленных языков программирования.

В результате самым оптимальным языком программирования для написания программы-парсера был выбран язык Python.

Так же язык Python славится наличием следующих достоинств:

- простота синтаксиса;

- наличие готовых модулей и библиотек;

- наличие множества книг с практическими примерами, которые упрощают написание кода и облегчают понимание языка.

Как следствие, на этом языке легко создавать приложения. На Python создано немало зарекомендовавших себя библиотек, которые помогли при написании данной работы.

3.5 Язык программирования Python

Python (в русском языке распространено название «Питон») — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Основной особенностью является минималистичный синтаксис ядра Python с одновременным наличием большого объёма полезных функций в стандартной библиотеке.

Python поддерживает несколько парадигм программирования:

- структурное;
- объектно-ориентированное;
- функциональное;
- императивное;
- аспектно-ориентированное.

Основные архитектурные черты:

- динамическая типизация;
- автоматическое управление памятью;
- полная интроспекция
- наличие механизма обработки исключений;
- поддержка многопоточных вычислений;
- удобные высокоуровневые структуры данных.

Код в Python организовывается в функции и классы, которые могут объединяться в модули, которые в свою очередь могут быть собраны в пакеты.

Эталонной реализацией Python является интерпретатор CPython, поддерживающий большинство активно используемых платформ. Он распространяется под свободной лицензией Python Software Foundation License, позволяющей использовать его без ограничений в любых приложениях. Существуют реализации интерпретаторов для JVM (с возможностью компиляции), MSIL (с возможностью компиляции), LLVM и других. Проект PyPy предлагает реализацию Python с использованием JIT-

компиляции, которая значительно увеличивает скорость выполнения Python-программ. [13]

3.6 Используемые библиотеки

В данной работе были использованы следующие библиотеки:

- HTTPX – это современный HTTP-клиент для Python, целью которого является предоставление более приятных и мощных возможностей для выполнения HTTP-запросов. Это более новый и многофункциональный HTTP-клиент для Python, чем популярная библиотека Requests. Он был создан для добавления поддержки асинхронного программирования благодаря многофункциональному API и хорошей производительности. HTTPX основан на хорошо зарекомендовавшем себя удобстве использования и предоставляет requests.

- API, совместимый с широкими запросами.

- Стандартный синхронный интерфейс, но с асинхронной поддержкой, если вам это нужно.

- Поддержка HTTP/1.1 и HTTP/2.

- Возможность делать запросы непосредственно к приложениям WSGI или приложениям ASGI.

- Везде строгие тайм-ауты.

- Полный текст аннотированный.

- 100% тестовое покрытие.

Плюс все стандартные возможности requests

- Международные домены и URL-адреса

- Keep-Alive и объединение в пул подключений

- Сеансы с сохранением файлов cookie

- Проверка SSL в стиле браузера

- Базовая/дайджест-проверка подлинности

- Элегантные файлы cookie с ключом и значением
- Автоматическая декомпрессия
- Автоматическое декодирование контента
- Тексты ответов в Юникоде
- Загрузка составных файлов
- Поддержка прокси-сервера HTTP(S)
- Тайм-ауты подключения
- Потоковые загрузки
- Поддержка .netrc
- Фрагментированные запросы;

- Asyncio – это библиотека для написания асинхронного кода с использованием синтаксиса async/await. Asyncio используется в качестве основы для нескольких асинхронных фреймворков Python, которые предоставляют высокопроизводительные сетевые и веб-серверы, библиотеки подключений к базам данных, распределенные очереди задач и т. д. Asyncio часто идеально подходит для связанного с вводом-выводом и высокоуровневого структурированного сетевого кода. Asyncio предоставляет набор высокоуровневых API-интерфейсов;

- Logging – этот модуль определяет функции и классы, которые реализуют гибкую систему регистрации событий для приложений и библиотек. Основное преимущество наличия API ведения журнала, предоставляемого стандартным библиотечным модулем, заключается в том, что все модули Python могут участвовать в ведении журнала, поэтому журнал приложения может включать ваши собственные сообщения, интегрированные с сообщениями из сторонних модулей;

- Collection – В этом модуле реализованы специализированные контейнерные типы данных, предоставляющие альтернативы встроенным контейнерам общего назначения Python, dict, list, set и tuple;

- Telethon – это асинхронная библиотека Python MTProto для взаимодействия с API Telegram в качестве пользователя или через учетную запись бота (альтернатива API бота). Эта библиотека предназначена для того, чтобы упростить написание программ на Python, которые могут взаимодействовать с Telegram. Думайте об этом как об оболочке, которая уже сделала тяжелую работу за вас, чтобы вы могли сосредоточиться на разработке приложения;

- Utils – это набор небольших функций и классов Python, которые делают общие шаблоны короче и проще. Этот модуль упрощает выполнение общих задач в скриптах Python, таких как преобразование текста в числа и обеспечение того, чтобы строка была в формате юникода или байтов.;

- Config – этот модуль позволяет использовать иерархическую схему конфигурации с поддержкой сопоставлений и последовательностей, перекрестные ссылки между одной частью конфигурации и другой, возможность гибкого доступа к реальным объектам, возможность включения, простое вычисление выражений и возможность изменения, сохранения, каскадирования и слияния конфигураций. Простое взаимодействие с переменными среды и параметрами командной строки; [14]

- Telegram_parser – Синтаксический анализ Telegram Bot API в OpenAPI, RAML или использование в качестве класса данных;

- Rss_parser – это типизированный модуль синтаксического анализа RSS на Python, построенный с использованием pydantic и xmltodict;

- Bcs_parser – этот модуль парсит отчеты от брокера БКС чтобы достать оттуда только тикеты и сделки;

3.7 Основные функции используемые в коде

В созданной программе-парсере применяются функции запросов:

1) get_запросы.

Эта функция отправляет запрос на сервер, чтобы получить дерево сайта в html формате. На вход функция принимает адрес страниц в интернете.

У нее есть параметр User-Agent. Это заголовок, с помощью которого анализируемый ресурс принимает запрос данной программы как от браузера, а не как от сторонней программы (рисунок 12).

```
19          response = await httpx_client.get(bcs_link, headers=random_user_agent_headers())
20          response.raise_for_status()
```

Рисунок 12 - Пример кода функции запроса

2)check func

Данная функция выбирает посты или статьи по ключевым словам. Полученный результат проверяется на наличие ключевых значений, в случае положительного результата информация публикуется, в случае отрицательного результата информация игнорируется (рисунок 13).

```
35  def check_pattern_func(text):
36      '''Вибирай только посты или статьи про газпром или газ'''
37      words = text.lower().split()
38
39      key_words = [
40          'газп',     # газпром
41          'газо',     # газопровод, газофикация...
```

Рисунок 13 - Пример функции проверки

3) send func

Данная функция отправляет посты в канал через бота. Параметр, который принимает функция на вход — это словарь. Словарь в языке Python – это структура данных, которая используется для хранения произвольных ключей и заданных им значений (рисунок 14).

```
89  async def send_message_func(text):
90      '''Отправляет посты в канал через бот'''
91      await bot.send_message(entity=gazp_chat_id,
92                             parse_mode='html', link_preview=False, message=text)
93
94      logger.info(text)
```

Рисунок 14 - Пример функции отправки

Функции parser написаны для трех агрегаторов новостей:

– bcs parser

Получает данные от крупной инвестиционной компании. Сводки котировок с крупнейшего финансового рынка (рисунок 15).

```
10  async def bcs_parser(httpx_client, posted_q, n_test_chars=50,
11                       timeout=2, check_pattern_func=None,
12                       send_message_func=None, logger=None):
```

Рисунок 15 - функция bcs_parser

– rss parser

Получает данные с новостных rss каналов (рисунок 16).

```
10  async def rss_parser(httpx_client, source, rss_link, posted_q, n_test_chars=50,
11                       timeout=2, check_pattern_func=None,
12                       send_message_func=None, logger=None):
```

Рисунок 16 - функция rss_parser

– telegram parser

Получает данные с новостных порталов телеграмм каналов (рисунок 17)

```
7   def telegram_parser(session, api_id, api_hash, telegram_channels, posted_q,
8                       n_test_chars=50, check_pattern_func=None,
9                       send_message_func=None, logger=None, loop=None):
```

Рисунок 17 - функция telegram_parser

3.8 Описание работы парсера

Как мы видим код каждого из трех парсеров написан таким образом, чтобы каждый парсер мог быть запущен самостоятельно, отдельно от других. Это значительно упрощает процесс добавления новых источников, их лучше проверять отдельно, чтобы убедиться в работоспособности. Так же это позволяет оперировать информацией полученной из определенного источника. Что позволяет сократить время на обработку большого потока информации.

Перед началом работы необходимо заполнить установочные данные в фале config.py. Вам обязательно понадобятся данные вашего приложения телеграмм, api_id и api_hash. Регистрацию приложение и получение всех необходимых данным проходим по адресу: my.telegram.org.

Далее обязательно указываем bot_token вашего телеграм-бота полученный при регистрации через BotFather и id телеграмм канала куда будут направляться все полученные результаты парсинга (рисунок 18).

```
1  # Параметры из my.telegram.org
2  api_id = [redacted]
3  api_hash = '[redacted]'
4
5  # Бот из @BotFather
6  bot_token = '[redacted]'
7
8  # id канала куда будут сливаться все новости
9  gazp_chat_id = [redacted]
```

Рисунок 18 - файл конфигурации

Следующим шагом настраиваем каналы агрегаторов новостей. В файле main.py указываем интересующие нас ссылки на каналы новостей в телеграмме. Там же сохраняем ссылки на каналы rss новостей(рисунок19).

```
telegram_channels = {
    1099860397: 'https://t.me/rbc_news',
    1428717522: 'https://t.me/gazprom',
    1101170442: 'https://t.me/rian_ru',
    1133408457: 'https://t.me/prime1',
    1149896996: 'https://t.me/interfaxonline',
    1001029560: 'https://t.me/bcs_express',
    1203560567: 'https://t.me/markettwits',
}

rss_channels = {
    'www.rbc.ru': 'https://rssexport.rbc.ru/rbcnews/news/20/full.rss',
    'www.ria.ru': 'https://ria.ru/export/rss2/archive/index.xml',
    'www.1prime.ru': 'https://1prime.ru/export/rss2/index.xml',
    'www.interfax.ru': 'https://www.interfax.ru/rss.asp',
}
```

Рисунок 19 - ссылки на новостные каналы

Несколькими строками ниже в коде в файле main.py указываем ключевые слова (рисунок 20).

```
    key_words = [
        'газп',
        'газо',
        'поток',
        'спг',
        'gazp',
    ]
```

Рисунок 20 - ключевые слова

После того как произведены все настройки запускаем телеграм-парсер telegram_parser.py проходим аутентификацию в telethon и получаем свои файлы сессии.

3.9 Время работы программы

Время работы программы зависит от многих факторов.

В первую очередь это качество интернет соединения и скорости передачи данных.

Так же это сильно зависит от таких вводных как:

- хотим ли мы собрать информацию со всех информационных порталов или только с определенного ресурса;

- количество информационных каналов с которых мы хотим собрать информацию;

- время последнего запуска программы-парсера;

- количества ключевых слов по которым происходит анализ данных используемых в работе программы;

- техническое состояние ПК, на котором осуществляется запуск программы-парсера;

Как мы видим из выше написанного, точное определение времени выполнения программы не представляется возможным. Единственными показателями измерения времени работы кода, являются замеры при тестовых запусках программы.

Тестовые запуски осуществлялись со следующими параметрами:

- компьютер на процессоре Core I7, оперативная память 32 гб, скорость интернет соединения 1 гбит;

- количество используемых ссылок в каждом парсере ресурса было от 3 до 5;

- количество ключевых слов было от 5 до 7, все слова были однокоренные;

При запуске кода с такими показателями, время выполнение в среднем составляло от 30 до 60 секунд. Что в целом считаю не плохим результатом.

3.10 Результат работы программы-парсера

Результатом работы парсера являются ссылки на новостные каналы с краткой информацией по ним (рисунок 21).

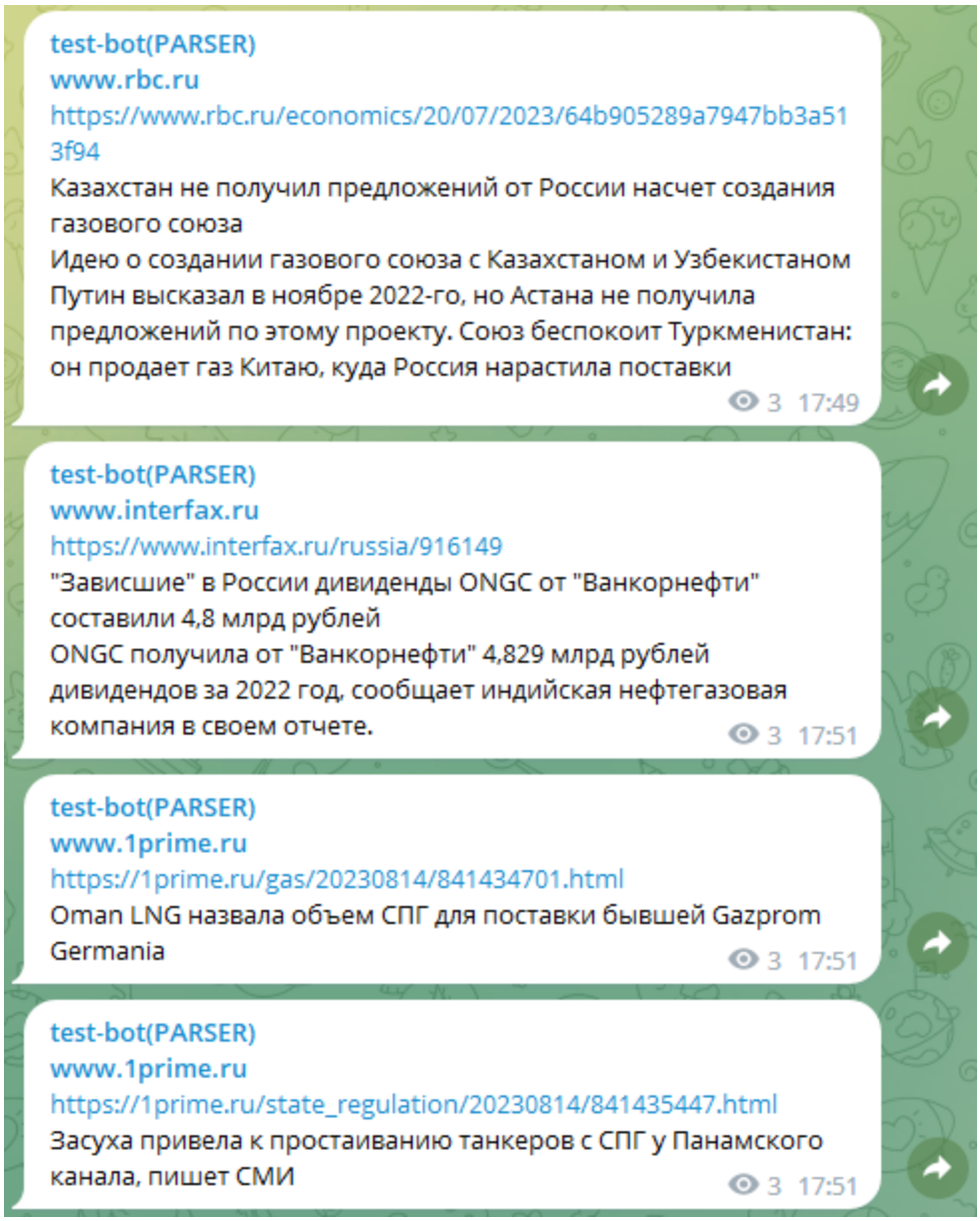


Рисунок 21 – Пример работы парсера

3.11 Область применения

Парсеры имеют широкое применение в различных отраслях. Один из наиболее распространенных случаев использования парсеров — это автоматический сбор и анализ информации с веб-страниц. Такое использование парсера облегчает задачу сбора и обработки данных при индексации веб-сайтов и составлении баз данных сайтов.

Организации, которые предоставляют услуги парсинга, могут использовать парсеры для сбора и анализа данных из открытых источников, таких как социальные сети, форумы и блоги. В коммерческих целях парсеры могут использоваться для сбора цен на товары, информации о конкурентах и других сведений, которые могут помочь в принятии бизнес-решений. Например на рисунки ниже представлена таблица с результатами торгов валют на одной из бирж (рисунок 22).

	A	B	C	D	E	F	G	H
1	EUR/USD	1,1796	1,1798	1,1808	1,1771	0,0027	+0,23%	05:57:26
2	USD/JPY	109,68	109,7	109,74	109,31	0,14	+0,13%	05:57:44
3	GBP/USD	1,3411	1,3414	1,3425	1,3405	0,0006	+0,04%	05:57:38
4	USD/CHF	0,9858	0,9859	0,9861	0,9839	0,0006	+0,06%	05:57:32
5	USD/CAD	1,2963	1,2965	1,2999	1,2955	0,0038	+0,29%	05:57:43
6	EUR/JPY	129,39	129,42	129,45	128,72	0,48	+0,37%	05:57:38
7	AUD/USD	0,7607	0,7609	0,7673	0,7561	0,0008	+0,11%	05:57:16
8	NZD/USD	0,704	0,7044	0,7047	0,7022	0,0008	+0,11%	05:56:59
9	EUR/GBP	0,8795	0,8797	0,8796	0,8778	0,0018	+0,21%	05:57:36
10	EUR/CHF	1,1628	1,1631	1,1631	1,1591	0,0038	+0,33%	05:57:43
11	AUD/JPY	83,44	83,47	83,48	82,99	0,2	+0,24%	05:57:35
12	GBP/JPY	147,11	147,15	147,18	146,6	0,2	+0,14%	05:57:43
13	CHF/JPY	111,26	111,29	111,3	111,02	0,09	+0,08%	05:57:43
14	EUR/CAD	1,5293	1,5294	1,5307	1,5266	0,0079	+0,52%	05:57:41
15	AUD/CAD	0,9862	0,9863	0,9869	0,9849	0,0039	+0,40%	05:56:59
16	CAD/JPY	84,6	84,64	84,66	84,11	-0,13	-0,15%	05:57:43
17	NZD/JPY	77,22	77,27	77,26	76,78	0,19	+0,25%	05:57:43
18	AUD/NZD	1,0802	1,0807	1,0817	1,0796	0	0,00%	05:56:26
19	GBP/AUD	1,7629	1,763	1,7667	1,7626	-0,0013	-0,07%	05:57:44
20	EUR/AUD	1,5504	1,5509	1,5514	1,5489	0,0019	+0,12%	05:57:31
21	GBP/CHF	1,322	1,3224	1,3224	1,3201	0,0008	+0,06%	05:57:44
22	EUR/NZD	1,675	1,6758	1,6768	1,6738	0,0023	+0,14%	05:57:43
23	AUD/CHF	0,7498	0,7502	0,7502	0,7474	0,0013	+0,17%	05:57:21
24	GBP/NZD	1,9045	1,9048	1,9099	1,9045	-0,0012	-0,06%	05:57:44
25	USD/INR	67,48	67,51	67,88	67,44	-0,02	-0,03%	05:57:41
26	USD/CNY	6,4077	6,4097	6,4088	6,4009	0,0021	+0,03%	05:57:45
27	USD/SGD	1,333	1,3353	1,336	1,3335	-0,0012	-0,09%	05:56:28
28	USD/HKD	7,8453	7,8478	7,8472	7,8452	0,0009	+0,01%	05:54:53
29	USD/DKK	6,3118	6,3168	6,328	6,3087	-0,0153	-0,24%	05:57:33
30	GBP/CAD	1,7385	1,739	1,7434	1,7383	0,0055	+0,32%	05:57:40
31	USD/SEK	8,6995	8,7025	8,7242	8,6918	-0,0089	-0,10%	05:57:32
32	USD/RUB	62,5143	62,5143	62,5143	62,5143	0	+0,00%	09/06
33	USD/TRY	4,4655	4,4717	4,4849	4,4665	-0,0028	-0,06%	05:57:09
34	USD/MXN	20,3175	20,3221	20,3484	20,3122	0,0293	+0,14%	05:57:32
35	USD/ZAR	13,0476	13,0634	13,1215	13,0328	-0,0049	-0,04%	05:57:44
36	CAD/CHF	0,7602	0,7607	0,7608	0,7575	-0,0016	-0,21%	05:57:35
37	NZD/CAD	0,9128	0,913	0,9133	0,9108	0,0036	+0,40%	05:57:10
38	NZD/CHF	0,6939	0,6945	0,6943	0,6915	0,0012	+0,17%	05:57:30
39	BTC/USD	6.772,0	6.772,1	7.323,1	6.618,9	-551	-7,52%	05:57:21
40	BTC/EUR	5.764,3	5.764,3	6.271,4	5.591,8	-478,3	-7,66%	05:55:58
41	ETH/USD	529,31	529,32	576,93	496,27	-47,61	-8,25%	05:57:36

Рисунок 22 – пример результата парсера, таблица csv

Парсеры могут быть использованы и для проверки корректности веб-страниц и контента, например, в процессе тестирования сайтов. Автоматический разбор HTML-структур может помочь определить наличие ошибок и опечаток в тексте, а также других проблем с кодом.

Кроме того, парсеры используются в компьютерном зрении и обработке естественного языка. Такие системы могут использоваться, например, для анализа текстов на естественном языке, которые затем используются для

принятия решений в сферах медицины, права и финансов. Парсеры могут также использоваться для автоматического извлечения фактов и именных сущностей из текста для составления баз данных и поиска информации.

Созданная программа-парсер собирает актуальные данные новостных каналов. Полученная, обработанная и приведённая в нужный формат информация в дальнейшем используется для развития различных новостных проектов.

Выводы по разделу: до начала работы были тщательно проанализированы все достоинства, недостатки и возможности высокоуровневых языков программирования. В результате этого анализа было принято решение о работе с использованием наиболее оптимального для этой цели языка программирования Python.

В ходе работы были подробно изучены синтаксис, модули и библиотеки выбранного языка.

В данном разделе подробно описаны основные функции программы и результат её работы. Также представлены показатели эффективности работы парсера. По результатам тестирования парсера можно сделать вывод о том, что все поставленные цели были достигнуты.

Поскольку данный парсер разрабатывался исключительно с целью изучения работы программы в ходе учебного процесса, у него есть потенциал развития. При этом можно сказать, что есть возможность доработки и обновления программы для выполнения дополнительных функций.

ЗАКЛЮЧЕНИЕ

Цель данного дипломного проекта заключалась в разработке программы-парсера новостей на основе новостных каналов официальных СМИ. В данной работе были изучены принципы, структура и функционал работы программы-парсера. Для достижения данной цели были определенны следующие задачи:

1 Изучить технологии парсинга.

2 Разработать проект модуля с моделью базы данных и структуры выходного файла.

3 Реализовать программный продукт в выбранных технологиях.

5 Провести тестирование модуля.

В соответствии поставленными задачами были сделаны следующие выводы:

- Изучена технология парсинга, в результате чего можно сказать, что данная технология отлично подходит для синтаксического анализа текстовых данных. Технология парсига применятся для считывания большого количества неструктурированной информации для приведения к необходимому формату данных. Технология парсинга позволяет оптимизировать временные затраты на обработку данных.

- При разработке проекта программы была определенна структура входных и выходных данных. Описана структура последовательностей данных для входного текстового файла. Описана структура выходной таблицы. Описана структура файла соединения в зависимости от типа подключения.

Таким образом, задачи дипломного проекта решены. Цель достигнута в полном объеме.

В первом разделе было проведено исследование проблемы семантического разбора html страниц в интернете. Были проанализированы и изучены все известные методы поиска и сбора информации с html страниц.

Кроме того, в первом разделе содержится описание всех функциональных возможностей парсеров. Рассматривались и сравнивались существующие методы парсинга. Были изучены их особенности, отличительные черты и достоинства. В ходе разработки были также учтены все недостатки самостоятельного создания программы парсера.

Во втором разделе рассмотрены различные варианты компонентов, фреймворков используемых при написании парсеров. Язык Python один из самых популярных языков программирования в наши дни. Одной из главных причин такой популярности, наряду с легким синтаксисом и универсальностью, стало наличие огромного числа дополнительных инструментов, значительно упрощающее и ускоряющее программирование на Python.

В третьем разделе представлено подробное описание разработанной программы. Для её написания был выбран отличающийся синтаксической простотой высокоуровневый язык программирования Pyton. В работе даны всеобъемлющие характеристики используемых библиотек и готовых модулей данного языка.

В данном разделе тщательно разобраны используемые функции программы, дано их подробное описание, а их применение проиллюстрировано рисунками.

В процессе разработки были выявлены проблемы к доступу сайтов. Которые решились с помощью модуля HTTPX. Добавлением заголовков в функции get_html. С помощью которых. Удалось эмулировать работу программы как браузера.

В конце работы был проанализирован показатель эффективности программы. Для этого было проведено тестирование парсера для проверки выполнения всех поставленных задач. По результатам тестирования стало ясно, что разработанный парсер работает корректно. Программа выполняет все предъявленные требования верно и за короткий промежуток времени. Ошибок в процессе работы не возникает.

Таким образом, разработанная программа-парсер отвечает всем требованиям. Исходя из этого можно сделать вывод об успешном исполнении технического задания.

СПИСОК ЛИТЕРАТУРЫ

1. Саркисян А.А. Влияние информационных технологий на жизнь человека в современных условиях // Молодежный научный форум: Технические и математические науки.
2. Wikipedia.org. Интернет ресурс – https://ru.wikipedia.org/wiki/Интернет
3. RG.ru. Электронный ресурс – https://rg.ru/2013/05/14/infa-site.html
4. Wikipedia.org. Интернет ресурс – https://ru.wikipedia.org/wiki/Синтаксический анализатор
5. Хохлова, Ю. Глоссарий по информационному обществу / Хохлова Ю.Е., Бунчук М.А. // Институт развития информационного общества.
6. Wikipedia.org. Интернет ресурс – https://ru.wikipedia.org/wiki/Портал:Компьютерные технологии
7. Парсинг сайтов. Интернет ресурс – http://parsing.valemak.com/ru/what-why-how/stages-of-parsing
8. Суханов, А.А., Маратканов, А.С. Анализ способов сбора социальных данных из сети интернет / Суханов, А.А., Маратканов, А.С. // International Scientific Review. – 2017
9. Wikipedia.org. Интернет ресурс – https://ru.wikipedia.org/wiki/Selenium/
10. Oxylabs.io. Интернет ресурс – https://oxylabs.io/blog/httpx-vs-requests-vs-aiohttp
11. Oxylabs.io. Интернет ресурс – https://oxylabs.io/blog/python-requests
12. Интернет: особенности и возможности. Интернет ресурс – https://studme.org/50396/menedzhment/internet_osobennosti_vozmozhnosti
13. Wikipedia.org. Интернет ресурс – https://ru.wikipedia.org/wiki/Python
14. Python (PyPI). Интернет ресурс – https://pypi.org/
15. Python 3.11.4 documentation. Интернет ресурс – https://docs.python.org

16. Цхошвили Д.З., Иванова Н.А. Примеры Использования Технологии Парсинга // Цхошвили Д.З., Иванова Н.А. // Статья в сборнике трудов конференции Язык: русский Год издания. – 2017
17. Wikipedia.org. Интернет ресурс – https://ru.wikipedia.org/wiki/Компьютерная программа
18. Обрабатываем csv файлы — Модуль csv. Интернет ресурс – https://python-scripts.com/import-csv-python