МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА ПРОГРАМУВАННЯ ТА МАТЕМАТИКИ

До захисту допускається
Завідувач кафедри
_____ Лифар В.О.
«____»_____ 20__ р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Інформаційна система аналізу текстових даних з сайтів територіальних громад на основі бот-парсерів

Освітній рівень“Магістр”
Спеціальність 126 “Інформаційні системи та технології”

Науковий керівник роботи:	_____ (підпис)	В.О. Лифар (ініціали, прізвище)
Студент:	_____ (підпис)	О.П. Арєф'єв (ініціали, прізвище)
Рецензент:	_____ (підпис)	С.О. Митрохін (ініціали, прізвище)
Група:		ІСТ-20дм

Сєвєродонецьк 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки

Кафедра Програмування та математики

Освітній рівень Магістр

Напрям підготовки
(шифр і назва)

Спеціальність 126 “Інформаційні системи та технології”
(шифр і назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри _____

_____ В.О.Лифар

«_____» _____20____р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Арєф’єву Олександру Павловичу
(прізвище, ім’я, по батькові)

1. Тема роботи Інформаційна система аналізу текстових даних з сайтів територіальних громад на основі бот-парсерів

керівник проекту (роботи) Лифар Володимир Олексійович, д.т.н., доц.
(прізвище, м.’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від 30» 11 2021 р. №182/15.16

2. Строк подання студентом роботи 20.12.2021

3. Вихідні дані до роботи Матеріали науково-дослідної практики, науково-методична література; дані інтернет-мережі;

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Пошук актуальної інформації, процес пошуку, висновки;

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Розробка технічного завдання	07.11.2021-14.11.2021	
2	Пошук і аналіз літератури з досліджуваної проблеми	14.11.2021-17.11.2021	
3	Аналіз технічних засобів	18.11.2021-22.11.2021	
4	Аналіз альтернативних рішень	23.11.2021-30.11.2021	
5	Оформлення пояснювальної записки і презентації	01.12.2021-10.12.2021	

Студент _____ Арєф'єв О.П.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Лифар В.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Арєф’єв О.П. Інформаційна система аналізу текстових даних з сайтів територіальних громад на основі бот-парсерів.

Метою роботи є аналіз та проектування системи пошуку та сортивування інформації на сайті громадських новин через телеграм бота.

Об’єкт дослідження - методи та інструментальні засоби побудови інформаційних систем пошуку актуальнох новин на сайті територеальних новин.

Робота присвячена складанню концепту системи пошуку зображень, тексту та посилань на сайті громадських новин через телеграм меседжер та боту. А також пошуку альтернативних рішень задачі.

Було проведено дослідження методів і засобів пошуку актуальної інформації на сайтах громадських новин завдяки боту у телеграм месенджері. Це дозволяє побудувати більш зручну систему для пошуку актуальної інформації.

ABSTRACT

Ariefiev O.P. Information system for analyzing text data from local community sites based on bot parsers.

The aim of the work is to analyze and design a system for searching and sorting information on the public news site via a telegram bot.

The object of research - methods and tools for building information systems for searching for current news on the site of territorial news.

The work is devoted to drafting the concept of a system for searching images, text and links on the public news site through a telegram messenger and bot. As well as finding alternative solutions to the problem.

A study was conducted on methods and means of finding relevant information on public news sites thanks to the bot in the telegram messenger. This

allows you to build a more convenient system for finding relevant information.A study was conducted on methods and means of finding relevant information on public news sites thanks to the bot in the telegram messenger. This allows you to build a more convenient system for finding relevant information.

ЗМІСТ

ВСТУП

З кожним роком кількість подій у світі зростає швидкими темпами. У всьому світі, у кожній країні, щось відбувається кожен день. Це може бути актуальна або ні інформація, корисна або ні для нас. Якщо брати останні два роки, то можна побачити, що зріст новин зростає так стрімко, що неможливо побачити все. Через це губиться велика кількість актуальної інформації, яка потрібна.

Наприклад, статистика та новини COVID-19 у країні, політика, економіка, або події в якійсь області країни.

Саме в цьому є актуальність даної роботи. Для пошуку зображень, інформації та посилань на переповнених сайтах громадських новин. Особливо зараз, коли потрібно завжди знати, що відбувається навколо нас і як це впливає на наше життя. Володіння актуальною інформацією завжди важливо.

У цій роботі буде розглянуто процес створення програмного забезпечення парсер-боту і буде розляглянуто альтернативні варианти рішеня даної задачі.

Дане програмне забезпечення допоможе користувачу швидко дізнатися потрібну для нього інформацію за декілька кліків, що позбавить його від трати важливого часу на пошуки серед непотрібної інформації.

1. АНАЛІЗ МЕТОДІВ ТА ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ АНАЛІЗУ ТЕКСТОВИХ ДАНИХ З САЙТІВ ТЕРИТОРІАЛЬНИХ ГРОМАД НА ОСНОВІ БОТ-ПАРСЕРІВ

Для швидкої обробки інформації застосовується парсинг. Так називають послідовний синтаксичний аналіз інформації, розміщеної на веб-сторінках. Цей метод використовується для оперативного опрацювання та копіювання великої кількості даних, якщо ручна робота вимагає багато часу. Для цього використовуються парсери - спеціальні програми, здатні аналізувати контент в автоматичному режимі і знаходити потрібні фрагменти.

Парсинг - це збір і сортування даних з певними параметрами. У цього інструменту маса переваг: швидкість, відсутність помилок у вибірці, можливість проводити парсинг регулярно. Плюс, багато парсери не просто збирають дані, але і радять, як виправити критичні помилки на вашому сайті.

1.1. Навіщо потрібен парсинг і парсери

Дані в мережі інтернет розташовані на веб-сайтах і представлені для людини у вигляді деякого набору графічних елементів, тексту, зображень. Людина здійснює парсинг кожен день: шукає номер телефону на веб-сторінці, потрібне зображення, переглядає товари в інтернет-магазині.

З англійської мови «to parse» - розбирати, аналізувати. Однак здібності людини обмежені. Пошук більше декількох десятків номерів на сайті може стати сучасною тортурами.

А якщо необхідно знайти сотні і тисячі номерів, адрес сторінок в соцмережах на сотнях веб-сторінок за певними умовами і запитам? Тоді знаючі люди використовують спеціальні програми - парсери. Вручну нереально освоїти такий обсяг інформації.

Також якась інформація може бути прихована від очей користувача, але вона є в коді веб-сторінки.

Спеціальні програми аналізують код сторінки за допомогою різних алгоритмів від зовсім простих (які може написати програміст) до найскладніших статистичних моделей з використанням теорії хаосу і нейронних мереж.

Парсери витягують потрібну інформацію, навіть якщо власник інформації не хотів нею ділитися. На багатьох сайтах номери телефонів відображаються не цифрами, а картинкою. Але хороший парсер впоратися з таким перешкодою.[5]

1.2. Чи законно парсити сайти

Якщо коротко, то законно - якщо ви Парс інформацію, яка є у відкритому доступі. Це логічно, адже так будь-яка людина і без парсеру може зібрати дані, що цікавлять.

Що переслідується законом:

- парсинг з метою DDOS-атаки;
- збір особистих даних користувачів, які знаходяться не на виду - наприклад, в особистому кабінеті, вказувалися при реєстрації тощо;
- парсинг для крадіжки контенту - наприклад, перепост чужих статей під своїм ім'ям, використання авторських фото не з безкоштовних стоків;
- збір інформації, яка становить державну або комерційну таємницю.

1.3. Ідеї для парсинга

Телеграм-бот, який щоранку надсилає вам прогноз погоди. Для цього він йде на погодний сайт і парсит з нього потрібні дані.

Следільщік за ціною товару на сайті. Налаштовуєте його кожен день ходити на потрібний сайт і дивитися, як змінюється ціна. Як тільки ціна впаде до потрібного вам показника, вам приходить повідомлення.

Удалятель прикметників з художніх творів або подсвечівальщік ключових слів. Наприклад, налаштували парсер, щоб він знаходив в будь-якому художньому тексті імена героїв і дієслова. І отримуєте Кривеньке, але читається короткий зміст твору без описів природи.

Погодні інформери для боротьби з пригніченням: налаштовуєте парсер на ключові слова, які використовуються для пригнічення чогось, що близько вашому серцю. Нацьковуєте парсер на форуми і сайти, де може траплятися пригнічення. Чи отримуєте список сторінок, де відбувається пригнічення. Але не забувайте, що якщо ви нацькували на щось парсер, то це вважається цькуванням.

1.4. Месенджери

Додаток для обміну повідомленнями - це будь-яка програма, яка забезпечує можливість обміну особистими повідомленнями між двома чи більше людьми. З кожним днем з'являється все більше і більше програм для обміну повідомленнями, і цей тип технології швидко стає найпопулярнішим способом надсилання текстових повідомлень, замінюючи SMS та MMS як перевагу більшості споживачів.

Програми для обміну повідомленнями стали звичайним явищем, і багато брендів використовують цю технологію для зв'язку своїх клієнтів з агентами з обслуговування клієнтів.

Технологія, що лежить в основі програм для обміну повідомленнями, широко використовується з початку 90-х років, але дійсно стала популярнішою на початку 2000-х років з появою таких програм для обміну повідомленнями, як AOL Messenger. До 2013 року кількість повідомлень, що надсилаються через ці програми, перевищила кількість SMS-повідомлень.

Тепер програми для обміну повідомленнями приваблюють мільярди користувачів, і вони використовуються в усьому світі. У WhatsApp зараз 1,6 мільярда користувачів, у месенджера Facebook – 1,3 мільярда, а у популярної платформи обміну повідомленнями WeChat – 1,1 мільярда користувачів.

1.5. Telegram

Telegram – це програма для обміну повідомленнями, яка зосереджена на швидкості та безпеці, вона надзвичайно швидка, проста та безкоштовна. Ви можете використовувати Telegram на всіх своїх пристроях одночасно — ваші повідомлення легко синхронізуються на будь-якій кількості ваших телефонів, планшетів або комп’ютерів. Telegram має понад 500 мільйонів активних користувачів щомісяця і входить до 10 найбільш завантажуваних додатків у світі.

За допомогою Telegram ви можете надсилати повідомлення, фотографії, відео та файли будь-якого типу (doc, zip, mp3 тощо), а також

створювати групи до 200 000 осіб або канали для трансляції необмеженій аудиторії. Ви можете писати контактам свого телефону та знаходити людей за їхніми іменами користувачів. У результаті Telegram схожий на SMS і електронну пошту разом — і може подбати про всі ваші особисті або ділові потреби в обміні повідомленнями. Крім того, ми підтримуємо наскрізне зашифровані голосові та відеодзвінки, а також голосові чати в групах для тисяч учасників.

Для кого Telegram.

Telegram для всіх, хто хоче швидкого та надійного обміну повідомленнями та дзвінків. Бізнес-користувачам і невеликим командам можуть сподобатися великі групи, імена користувачів, настільні програми та потужні параметри обміну файлами.

Оскільки групи Telegram можуть налічувати до 200 000 учасників, ми підтримуємо відповіді, згадки та хештеги, які допомагають підтримувати порядок та підтримувати ефективне спілкування у великих спільнотах. Ви можете призначити адміністраторів із розширеними інструментами, щоб допомогти цим спільнотам процвітати в мирі. До публічних груп може приєднатися будь-хто, і вони є потужною платформою для обговорення та збору відгуків.

Якщо ви більше любите зображення, у Telegram є анімований пошук gif, сучасний фоторедактор і відкрита платформа для наклейок (знайдіть кілька крутих наклейок тут або тут). Більше того, не потрібно турбуватися про місце на диску на вашому пристрої. Завдяки хмарній підтримці Telegram і параметрам керування кешом, Telegram може займати майже нуль місця на вашому телефоні.

Тим, хто шукає додаткову конфіденційність, слід ознайомитися з нашими розширеними налаштуваннями та досить революційною політикою. А якщо вам потрібна таємниця, спробуйте наші секретні чати для окремих пристроїв із повідомленнями, фотографіями та відео, що самознищуються, і заблокуйте свою програму за допомогою додаткового пароля.

1.6. TelegramBot

Боти - це сторонні програми, які працюють усередині Telegram. Користувачі можуть взаємодіяти з ботами, надсилаючи їм повідомлення, команди та вбудовані запити. Ви керуєте своїми ботами за допомогою HTTPS-запитів до нашого Bot API.

1.7. Що я можу робити із ботами

Назвемо лише кілька речей: ви можете використовувати ботів для:

Отримуйте індивідуальні повідомлення та новини . Бот може діяти як розумна газета, надсилаючи вам релевантний контент, як тільки він буде опублікований.

Інтегруйте з іншими сервісами. Робот може доповнювати чати Telegram контентом із зовнішніх сервісів.
Gmail Бот , GIF-бот , IMDB-бот , Wiki-бот , Музичний бот , Youtube-бот , GitHubBot

Приймайте платежі від користувачів Telegram. Робот може пропонувати платні послуги або працювати як віртуальна вітрина. Детальніше "
Demo Shop Bot, Demo Store

Створюйте власні інструменти. Бот може надавати вам оповіщення, прогнози погоди, переклади, форматування та інші послуги.
Markdown боти , наклейки боти , Голосуйте боти , як і личинка

Створюйте одиночні та розраховані на багато користувачів ігри . Бот може запропонувати багаті можливості HTML5 від простих аркад і головоломок до 3D-шутерів і стратегічних ігор в реальному часі.
GameBot, Gamee

Створюйте соціальні служби. Бот може пов'язувати людей, які шукають партнерів для розмови на основі спільних інтересів чи близькості.[2]

1.8. Як працюють боти

По суті, роботи Telegram – це спеціальні облікові записи, для яких не потрібний додатковий номер телефону. Користувачі можуть взаємодіяти з ботами двома способами:

Надсилайте повідомлення та команди ботам, відкриваючи з ними чат або додаючи їх до груп.
Надсилайте запити прямо з поля введення, вводячи @username бота та запит. Це дозволяє відправляти контент від вбудованих ботів прямо до будь-якого чату, групи або каналу.

Повідомлення, команди та запити, надіслані користувачами, надсилаються програмному забезпеченню, що працює на ваших серверах. Наш проміжний сервер обробляє все шифрування та зв'язок з Telegram API за вас. Ви спілкуєтеся з цим сервером через простий інтерфейс HTTPS, який пропонує спрощену версію Telegram API. Ми називаємо цей інтерфейс нашим Bot API.[3]

1.9. Як створити бота

Батько-бот.

Натисніть, щоб побачити картинку з високою роздільною здатністю Для цього є бот. Просто поговоріть з BotFather і виконайте кілька простих кроків. Після того, як ви створили бота і отримали токен аутентифікації, перейдіть до посібника з API бота, щоб дізнатися, чого ви можете навчити свого бота.

1.10. Чим боти відрізняються від людей

Боти не мають онлайн-статусу і позначок часу останнього відвідування, натомість в інтерфейсі відображається мітка «бот» .

Боти мають обмежене хмарне сховище - старі повідомлення можуть бути видалені сервером невдовзі після обробки.

Боти не можуть ініціювати розмови з користувачами. Користувач повинен або додати їх до групи, або спочатку надіслати їм повідомлення. Люди можуть використовувати t.me/<bot_username>посилання або пошук на ім'я користувача, щоб знайти ваш бот.

Імена користувачів ботів завжди закінчуються на "бот" (наприклад, @TriviaBot, @GitHub_bot).

При додаванні в групу роботи за замовчуванням не отримують усі повідомлення (див. Режим конфіденційності).
Боти ніколи не їдять, не сплять і не скаржаться (якщо спеціально не запрограмовано інше).

1.11. Бонуси для роботів.

Телеграма ботами є унікальними багато в чому - ми пропонуємо два види клавіатур, додаткові інтерфейси для команд за замовчуванням і глибоке зв'язування, а також форматування тексту, інтегрованих платежів та багато іншого.

Вбудований режим
Користувачі можуть взаємодіяти з вашим ботом за допомогою вбудованих запитів з поля введення тексту в будь-якому чаті. Все, що потрібно зробити, це почати повідомлення з ім'ям користувача вашого бота, а потім ввести запит.

Отримавши запит, ваш бот може повернути деякі результати. Як тільки користувач натискає один з них, він відправляється в поточний відкритий чат користувача. Таким чином, люди можуть вимагати контент у вашого бота в будь-якому зі своїх чатів, груп або каналів.

Завітайте на цей блог, щоб побачити в дії приклад вбудованого бота. Ви також можете спробувати боти @sticker і @music, щоб переконатися в цьому самі.

1.12. Платіжна платформа

Ви можете використовувати боти для отримання платежів від користувачів Telegram по всьому світу.

Надсилайте рахунки в будь-який чат, у тому числі в групи та канали.

Створюйте рахунки-фактури, які можуть бути відправлені та використані кількома покупцями для замовлення речей.

Використовуйте вбудований режим, щоб допомогти користувачам показувати ваші товари та послуги своїм друзям та спільнотам.

Дозволити чайові від користувачів із заздалегідь встановленими сумами, що налаштовуються.

Приймайте платежі від користувачів у мобільних або настільних програмах.

Спробуйте @ShopBot, щоб створити тестовий рахунок - або почати повідомлення @ShopBot ...в будь-якому чаті для вбудованого рахунку.

Відвідайте Demo Shop, щоб побачити приклад каналу Telegram, який використовується як віртуальна вітрина.

1.13. Ігрова платформа

Боти можуть пропонувати своїм користувачам ігри HTML5 для гри поодинці або для змагань один з одним у групах та в чатах один на один. Платформа дозволяє вашому роботу відстежувати рекорди кожної гри, зіграної в кожному чаті. Щоразу, коли у грі з'являється новий лідер, інші учасники чату повідомляються, що їм потрібно активізувати його.

1.14. Клавіатури

Традиційних чат-ботів, звісно, можна навчити розуміти людську мову. Але іноді вам потрібно більш формальне введення від користувача - і саме тут клавіатури, що настроюються, можуть стати надзвичайно корисними.

Щоразу, коли ваш бот відправляє повідомлення, може передати спеціальну клавіатуру з визначеними параметрами відповіді (див. ReplyKeyboardMarkup). Програми Telegram, які отримують повідомлення, відображатимуть вашу клавіатуру для користувача. Натискання будь-якої кнопки негайно надішле відповідну команду. Таким чином, ви можете спростити взаємодію користувача з вашим ботом.

1.15. Вбудовані клавіатури та оновлення на льоту

Бувають випадки, коли ви волієте робити щось, не надсилаючи жодних повідомлень у чат. Наприклад, коли користувач змінює налаштування або переглядає результати пошуку. У таких випадках можна використовувати вбудовані клавіатури, які інтегровані безпосередньо в повідомлення, яким вони належать.

На відміну від клавіатур, що налаштовуються для відповіді, натискання кнопок на вбудованих клавіатурах не призводить до відправки повідомлень в чат. Натомість, вбудовані клавіатури підтримують кнопки , які працюють за лаштунки: кнопки зворотного виклику , кнопки URL і перейти на вбудовані кнопки .

Кнопки зворотного виклику @music Більше кнопок зворотного виклику @music Кнопка URL

Коли використовуються кнопки зворотного дзвінка, ваш бот може оновлювати свої існуючі повідомлення (або лише свої клавіатури), щоб чат залишався акуратним. Ознайомтеся з цими прикладами ботів, щоб побачити вбудовані клавіатури в дії: @music, @vote, @like.

1.16. Команди

Є більш гнучкий спосіб зв'язку з вашим ботом. Може використовуватися наступний синтаксис:

/command
Команда завжди повинна починатися з символу «/» і не може бути довшою за 32 символи. Команди можуть використовувати латинські літери, цифри та символи підкреслення. Ось кілька прикладів:

/get_messages_stats
/set_timer 10min Alarm!
/get_timezone London, UK
Повідомлення, що починаються з косої межі, завжди передаються боту (разом з відповідями на його повідомлення та повідомленнями, які @ згадують бота на ім'я користувача). Програми Telegram:

Запропонуйте список команд, що підтримуються, з описами, коли користувач вводить '/' (щоб це працювало, ви повинні надати список команд для BotFather). Натискання на команду у списку негайно надсилає команду. Показувати додаткову кнопку (/) у полі введення у всіх чатах із ботами. При натисканні на неї набирається '/' і з'являється список команд.
Виділіть / команди у повідомленнях. Коли користувач натискає на виділену команду, команда відправляється відразу.

1.17. Глобальні команди

Щоб спростити користувачам навігацію по мультивсесвіту ботів, ми просимо всіх розробників підтримувати кілька основних команд. Програми Telegram матимуть ярлики інтерфейсу для цих команд.

/start - починає взаємодію з користувачем, наприклад, надсилаючи вітальне повідомлення. Цю команду також можна використовувати для передачі додаткових параметрів (див. Глибинні посилання)
/help – повертає довідкове повідомлення. Це може бути короткий текст про те, що може робити ваш бот і список команд.
/settings - (якщо застосовно) повертає налаштування бота для цього користувача та пропонує команди для редагування цих налаштувань.
Користувачі побачать кнопку "Пуск", коли вперше почнуть розмову з вашим ботом. Посилання "Довідка" та "Налаштування" будуть доступні в меню на сторінці профілю бота.

1.18. BotFather

BotFather - єдиний бот, який керує ними всіма. Це допоможе вам створити нових ботів і змінити існуючі настройки.

Створення нового бота

Використовуйте команду /newbot, щоб створити новий бот. BotFather запросить у вас ім'я та ім'я користувача, а потім згенерує токен автентифікації для нового бота.

Ім'я вашого бота відображається в контактній інформації та інших місцях.

Ім'я користувача це коротке ім'я, яке буде використовуватися в згадує і t.me посилання. Імена користувачів складаються з 5–32 символів та нечутливі до регістру, але можуть включати лише латинські символи, числа та символи підкреслення. Ім'я користувача вашого бота має закінчуватися на "бот", наприклад, "tetris_bot" або "TetrisBot".

Створення токена аутентифікації

Якщо ваш токен скомпрометований або ви втратили його з якоїсь причини, використовуйте команду /token для створення нового.

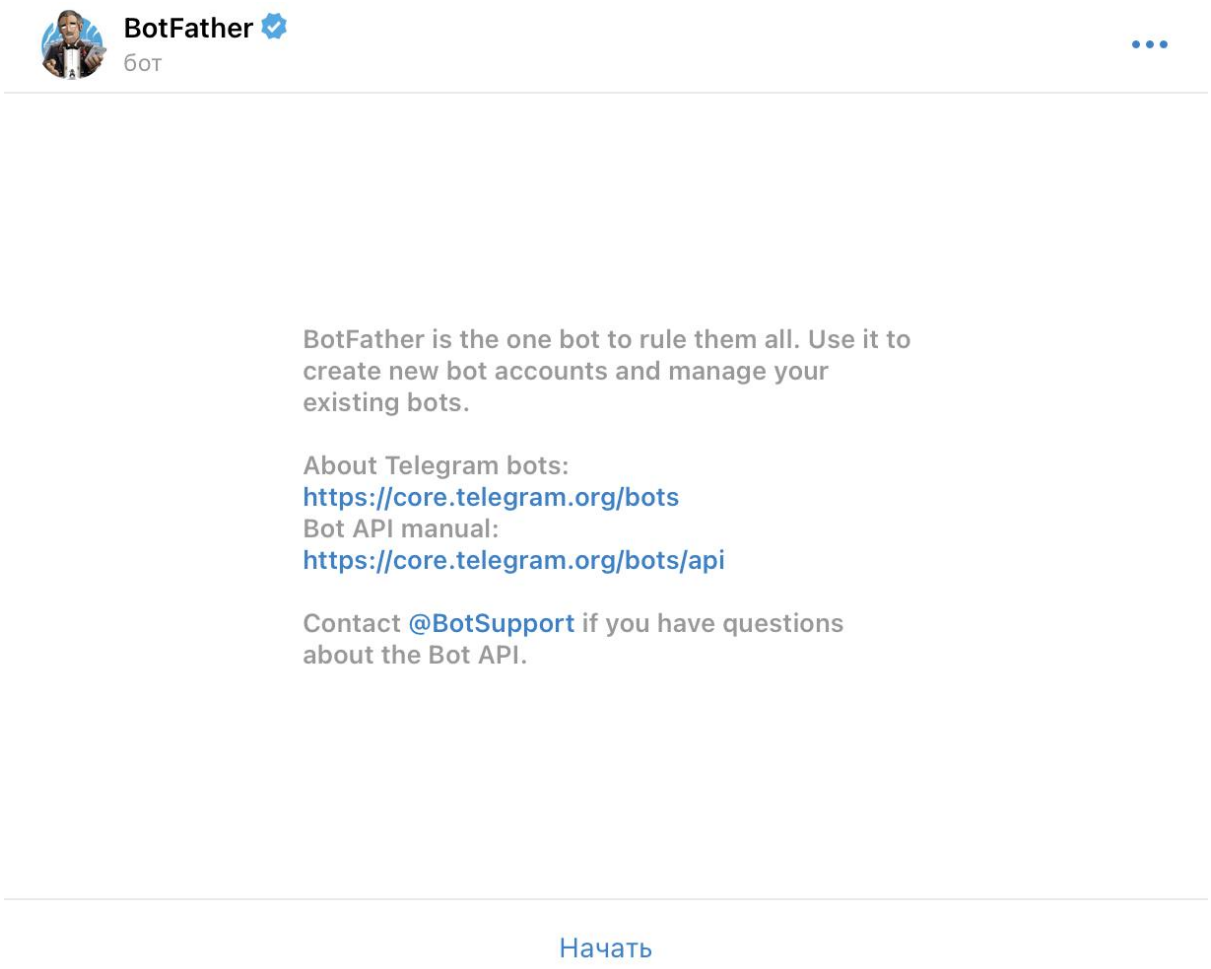


Рис. 1.1 – Telegram. Botfather

1.19. Команди Botfather

Інші команди говорять самі за себе:

/mybots - повертає список ваших ботів із зручними елементами керування для редагування їхніх налаштувань.
/mygames - робить те саме для ваших ігор
Редагувати ботів

/setname – змінити ім'я вашого бота.
/setdescription - змінити опис бота, короткий текст до 512 символів, що описує ваш бот. Користувачі побачать цей текст на початку розмови з ботом під назвою «Що вміє цей бот?».
/setabouttext - зміна бота про інформацію, ще коротший текст до 120 символів. Користувачі побачать цей текст на сторінці профілю бота. Коли вони діляться вашим ботом із кимось, цей текст відправляється разом із посиланням.
/setuserpic - змінити фото профілю бота. Завжди приємно змінити ім'я на обличчя.
/setcommands - змінити список команд, які підтримуються вашим ботом. Користувачі будуть бачити ці команди як пропозиції, коли вони набирають текст у чаті з вашим ботом. Кожна команда має ім'я (має починатися з косої межі '/', буквено-цифрове значення плюс підкреслення, не більше 32 символів, без урахування регістру), параметри та текстовий опис. Користувачі будуть бачити список команд щоразу, коли вони набирають '/' у розмові з вашим ботом.
/deletebot - Видалити свого бота і звільнити його логін.
Змінити налаштування

/setinline – переключити вбудований режим для вашого бота.
/setinlinegeo - запитати дані про місцезнаходження для надання вбудованих результатів на основі розташування.
/setjoingroups - переключити, чи можна додавати вашого бота до групи чи ні. Будь-який бот повинен мати можливість обробляти особисті повідомлення, але якщо ваш бот не призначений для роботи в групах, ви можете вимкнути це.
/setprivacy - встановити, які повідомлення отримуватиме ваш бот при додаванні до групи. Якщо режим конфіденційності вимкнено, бот отримуватиме всі повідомлення. Ми рекомендуємо залишити режим конфіденційності увімкненим. Вам потрібно буде повторно додати бота в існуючі групи, щоб ця зміна набула чинності.
Керувати іграми

/newgame - створити нову гру.
/listgames – отримати список ваших ігор.
/editgame – редагувати гру.
/deletegame - Видалити існуючу гру.

1.20. Висновок

Якщо ж ваші завдання ґрунтуються на зборі даних, на величезних базах даних і пошуку потрібного контенту або контактів - то ви або вже практикуєте парсинг, або за вас це робить хтось інший.

У будь-якому випадку без парсинга не обходиться практично жодна сфера, все так чи інакше вдавалися до нього, а хтось користується постійно.

Завдяки Telegram-боту, цю задачу можна зробити більш зручною та швидкою для користувачів.

2. МЕТОДИ ТА ІНСТРУМЕНТАЛЬНІ ЗАСОБИ ПОБУДОВИ ІНФОРМАЦІЙНОЇ СИСТЕМИ АНАЛІЗУ ТЕКСТОВИХ ДАНИХ

2.1. Можливості парсерів

- Оновлення інформації для підтримки актуальності. Відстежувати зміни курсу валют або прогнозу погоди в ручному режимі неможливо, тому вдаються до парсингу.
- Збір і швидке копіювання інформації з інших сайтів для розміщення на власному ресурсі. Дані, отримані за допомогою парсинга, піддають рерайтінгу. Таке рішення використовується для заповнення сайтів регіону, новинних проектів, ресурсів з кулінарними рецептами та інших майданчиків.
- З'єднання потоків даних. Проводиться збір великої кількості даних з декількох джерел, обробка та розміщення. Це зручно для заповнення новинних майданчиків.
- Парсинг істотно прискорює процес роботи з ключовими словами. Налаштувавши роботу, можливо оперативно підібрати необхідні для просування запити. Після кластеризації по сторінках готується SEO-контент, в якому буде враховано максимум ключів.

2.2. Особливості роботи парсера

Парсери пишуться на будь-якій мові програмування (PHP, C ++, Delphi і інших), де є підтримка регулярних виразів. Це набір метасимволов, використовуваних для пошуку необхідних даних.

Парсер за короткий термін обходить тисячі сторінок, фільтрує представлені дані, відбираючи серед них потрібні, після чого пакує отриманий результат для подальшої обробки.

2.3. Етапи парсинга

Якщо не занурюватися в технічні подробиці, то парсинг будується з таких етапів:

- користувач задає в парсером умови, яким повинна відповідати вибірка - наприклад, всі ціни на конкретному сайті;
- програма проходиться по сайту або декільком та збирає релевантну інформацію;
- дані упорядковано;
- користувач отримує звіт - якщо проводилася перевірка на помилки, то критичні виділяються контрастним кольором;
- звіт можна вивантажити в потрібному форматі - зазвичай парсери підтримують кілька.[6]

Рис. 2.1 – Етапи парсинга

2.4. Що таке RSS. Альтернативне рішення

RSS (Rich Site Summary, багате зведення сайту) – це автоматично генероване зведення у форматі rss або xml, в якому відображаються нещодавно опубліковані статті та новини. При цьому на повну версію вказаних матеріалів дається гіперпосилання. Дуже часто цей формат використовується інформаційними порталами та блогами. RSS-стрічку можна підключити до Яндекс.Новин, Google News, Яндекс.Дзен, Турбо-сторінок тощо.[8]

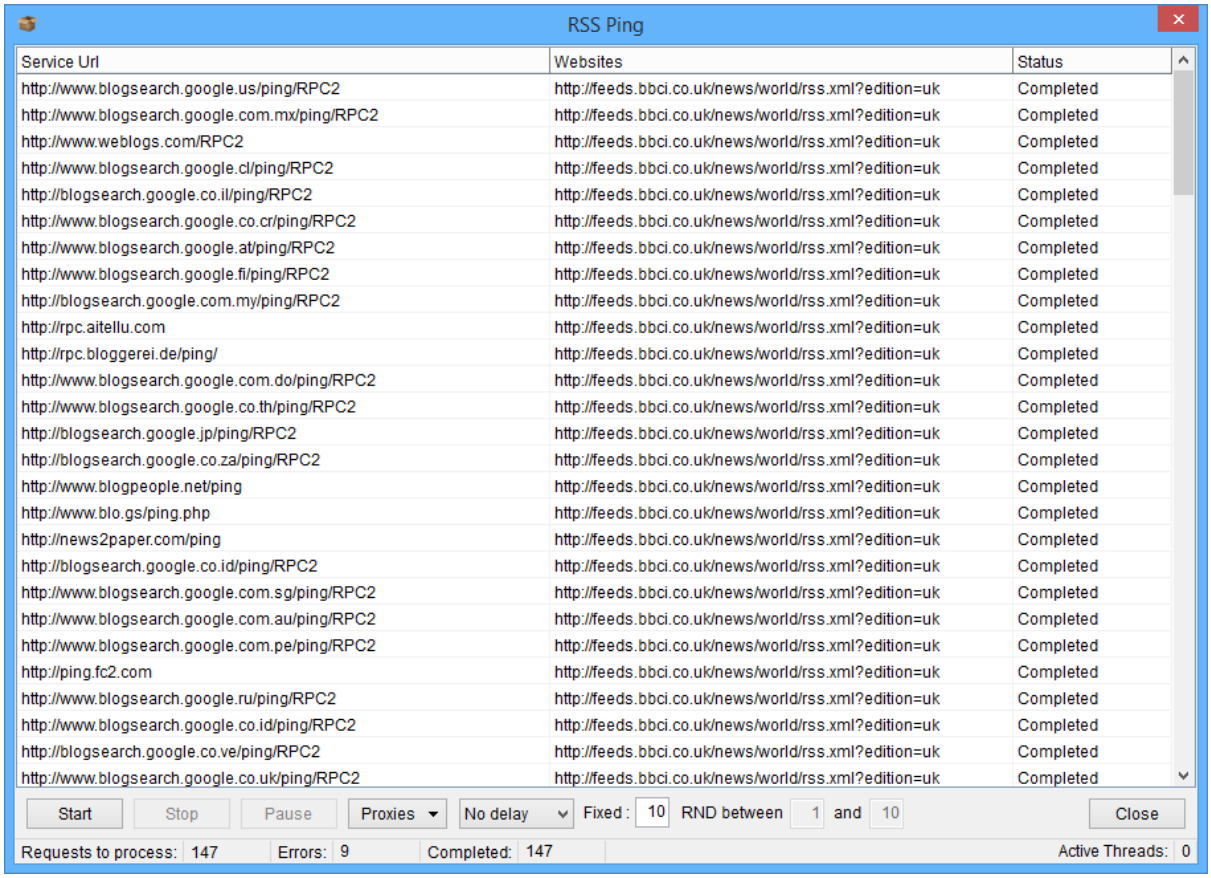
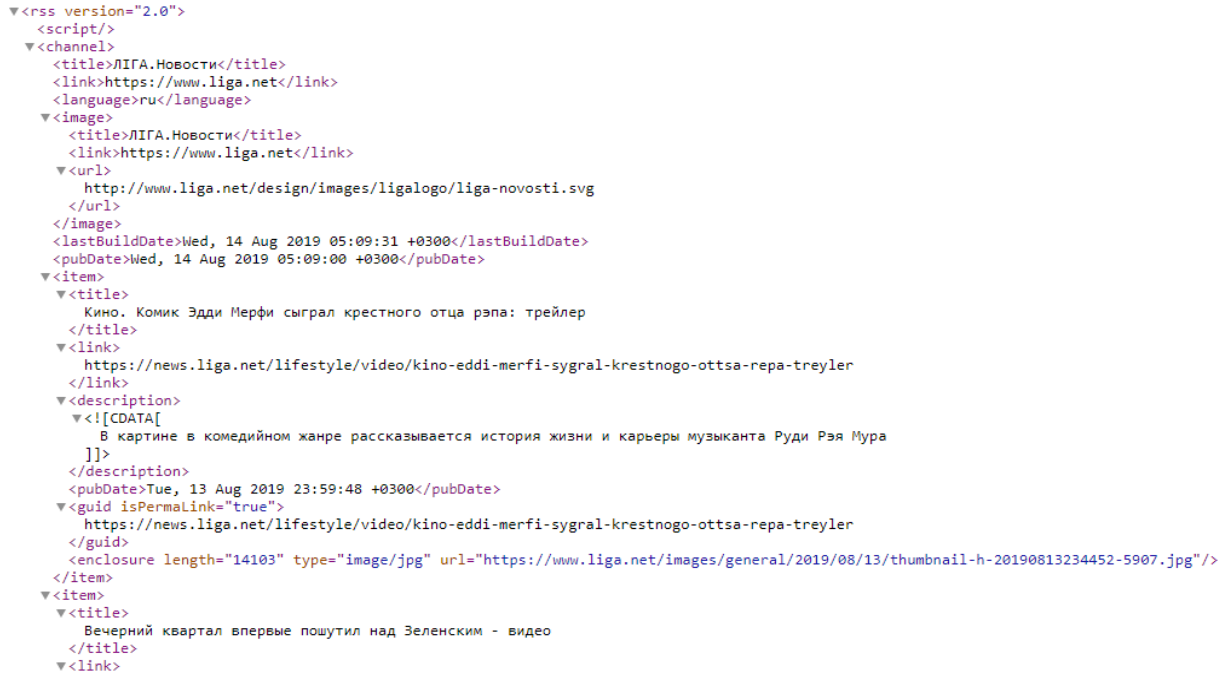


Рис. 2.1 – RSS Ping

2.5. Як працює RSS

Тут все просто – на сайті публікується стаття чи новина. Після цього спеціальний скрипт заносить анонс цієї публікації в RSS-файл, що оновлюється. Потім цей файл зчитують спеціальні програми або онлайн-ресурси, звані агрегаторами. І вже потім ці самі агрегатори сповіщають користувачів про нові матеріали на сайті. Повідомлення можуть надходити не щоразу після публікації нового матеріалу, а із зазначеною періодичністю.



```
<rss version="2.0">
  <script/>
  <channel>
    <title>ЛІГА.Новости</title>
    <link>https://www.liga.net</link>
    <language>ru</language>
    <image>
      <title>ЛІГА.Новости</title>
      <link>https://www.liga.net</link>
      <url>
        http://www.liga.net/design/images/ligalogo/liga-novosti.svg
      </url>
    </image>
    <lastBuildDate>Wed, 14 Aug 2019 05:09:31 +0300</lastBuildDate>
    <pubDate>Wed, 14 Aug 2019 05:09:00 +0300</pubDate>
    <item>
      <title>
        Кино. Комик Эдди Мерфи сыграл крестного отца рэпа: трейлер
      </title>
      <link>
        https://news.liga.net/lifestyle/video/kino-eddi-merfi-sygral-krestnogo-ottsa-repa-treyler
      </link>
      <description>
        <![CDATA[
          В картине в комедийном жанре рассказывается история жизни и карьеры музыканта Руди Рэя Мура
        ]]>
      </description>
      <pubDate>Tue, 13 Aug 2019 23:59:48 +0300</pubDate>
      <guid isPermaLink="true">
        https://news.liga.net/lifestyle/video/kino-eddi-merfi-sygral-krestnogo-ottsa-repa-treyler
      </guid>
      <enclosure length="14103" type="image/jpg" url="https://www.liga.net/images/general/2019/08/13/thumbnail-h-20190813234452-5907.jpg"/>
    </item>
    <item>
      <title>
        Вечерний квартал впервые пошутил над Зеленским - видео
      </title>
      <link>
```


Рис. 2.2 – RSS-стрічка

2.6. Плюси та мінуси RSS

Спочатку варто розглянути переваги RSS-каналу для користувачів.

- Збір усієї публікованої інформації в єдиний потік, який можна швидко переглянути.
- Безкоштовне використання.

- Швидке оповіщення – як тільки стаття опублікується на сайті, передплатник практично одразу дізнається про це.
- Відсутність реклами.
- Економія трафіку, особливо на мобільних пристроях.
- Економія часу. Легше, звичайно ж, подивитися зведення за новими опублікованими матеріалами і знайти щось цікаве, ніж гортати весь ресурс по кілька разів.

А тепер подивимося, які плюси отримає сайт, якщо в ньому використовуватиметься технологія RSS.

- Впровадження подібної стрічки дає зростання трафіку та підвищену залученість аудиторії. Це також допоможе просунутися у пошуковій видачі.
- Підвищення пізнаваності ресурсу серед передплатників.
- При прочитанні анонсу велика ймовірність того, що користувач відкриє посилання на сайт, щоб прочитати матеріал повністю.

Мінуси такого формату, звичайно ж, є, і в сучасних реаліях вони більш ніж істотні.

- Надлишок інформації. Якщо користувач підписується на безліч стрічок від різних сайтів, він буде перевантажений інформацією та не встигне обробити її.
- Крадіжка контенту. Більші ресурси можуть брати унікальний контент із вашого сайту та видавати за свій. Навіть вказівка посилання не врятує ситуацію, а якщо її взагалі немає, то довести статус першоджерела значно проблематичніше.

2.7. Методи отримання інформації

Можна виділити такі методи отримання інформації з сайтів:

1. **Ручний** - у разі роль парсера виконує людина. Він здійснює весь ланцюжок дій, необхідну для отримання необхідної інформації. Інформація збирається звичайним копіпастом.

2. **Гібридний** – користувач, як і раніше, виконує основні дії для отримання інформації, але може використовувати допоміжні програмні засоби для автоматизації збору, наприклад, браузерний плагін, який на основі конфігурації витягує та структурує інформацію з певних місць сторінки під час активації.

3. **Автоматичний** – отримання та структурування інформації виконується автоматично.

Для того щоб один раз отримати значення кількох полів з 10-15 сторінок, писати окремий парсер може бути недоцільно, проте у випадку великої кількості сторінок, полів або високої періодичності збору інформації, варто задуматися про автоматизацію даного процесу.

Процес отримання інформації з окремої веб-сторінки можна розбити на наступні етапи:

1. Побудова запиту на отримання информации.

2. Виконання запиту та отримання відповіді.

3. Обробка відповіді, вилучення та структурування необхідної інформації.

4. Передача отриманої інформації для подальшої обробки.

Процес отримання інформації може бути простим: завантажити URL, рахувати інформацію і віддати одержувачу; а може бути й складним: авторизуватися в системі, сконструювати запит щодо інформації із

заголовків та значень JavaScript-змінних на сторінці, ім'я яких може змінюватися з кожним запитом, а JS-код перебувати у мініфікованому чи обфусцованому вигляді.

Якщо в першому випадку все досить просто, то в другому, щоб за розумний час розробити парсер, варто вдатися до використання headless-браузерів (без графічного інтерфейсу) з підтримкою сценаріїв на кшталт

PhantomJS для отримання даних для оптимізації часу на вивчення того, як сайт взаємодіє з бекендом. Також для цих цілей можна вдатися до Selenium WebDriver з одним із реальних браузерів.

2.8. Інструменти

Для автоматизованого вилучення інформації з веб-сторінок існує 4 типи інструментів:

1. Бібліотеки.

Цей підхід вимагає розуміння процесу формування запитів та логіки роботи програми, що спричиняє додаткові витрати на вивчення низькорівневої роботи сайту. Можливо, це доречно і виправдано для одиничного сайту, але якщо потрібно написати кілька парсерів для декількох різнорідних сайтів за обмежений час — навряд.

До таких інструментів належать численні бібліотеки для різних мов програмування: jSoup для Java, SimpleHTMLDom для PHP, lxml.html для Python та інші.

2. Headless-браузери.

Даний підхід дозволяє обробляти сторінку в браузері з підтримкою JavaScript, що дозволяє писати свої сценарії для отримання необхідної інформації і навіть використовувати JavaScript бібліотеки на кшталт jQuery для отримання інформації зі сторінки, що прискорює розробку парсерів. Відсутність графічного інтерфейсу дозволяє запускати дані браузери навіть серверах, підтримують лише консольний режим.

До таких інструментів можна віднести PhantomJS та SlimerJS.

3. SaaS рішення.

Дані сервіси надають графічний інтерфейс, за допомогою якого можна вказати адресу сторінки, вказати блоки, з яких потрібно отримати інформацію, а також створити низку правил вилучення даних. Такі послуги не мають тієї гнучкості, яку надають низькорівневі рішення. Деякі з них коштують досить дорого, проте ними просто користуватися.

До таких сервісів можна віднести Mozenda, Octoparse (безкоштовний), Import.io (безкоштовний).

4. Настільні програми.

Надають графічний інтерфейс з можливістю завдання правил отримання інформації зі сторінки. Як і SaaS рішення, вони полегшують роботу користувача, позбавляючи його необхідності написання коду, мають певну гнучкість, але не можуть зрівнятися з рішеннями, що розробляються під конкретний сайт.

Приклад: IRobotSoft.

Основна проблема автоматичних парсерів вони перестають працювати при зміні логіки роботи сайту. Тому важливо проектувати

парсери таким чином, щоб вони могли сповіщати про неможливість отримання необхідної інформації, а в разі потреби їх можна було адаптувати під зміни, що відбулися.

2.9. Методи реалізації

Метод рекурсивного спуску - алгоритм низхідного синтаксичного аналізу, що реалізується шляхом взаємного виклику процедур парсингу, де шкірна процедура відповідає одному з правил контекстно-вільної граматики або БНФ. Застосування правил послідовно, зліва поглинають токени, отримані від лексичного аналізатора. Це один із найпростіших алгоритмів парсингу, що підходить для повністю ручної реалізації.

2.10. Python

Python, одна з найпопулярніших мов програмування у світі, створила все: від алгоритму рекомендацій Netflix до програмного забезпечення, яке керує самокерованими автомобілями. Python — це мова загального призначення, що означає, що вона призначена для використання в ряді додатків, включаючи науку про дані, програмне забезпечення та веб-розробку, автоматизацію та загалом виконання завдань.

Python – це мова комп’ютерного програмування, яка часто використовується для створення веб-сайтів і програмного забезпечення, автоматизації завдань і аналізу даних. Python — це мова загального призначення, що означає, що його можна використовувати для створення різноманітних програм і не спеціалізується на будь-яких конкретних проблемах. Ця універсальність разом із зручністю для початківців зробили її однією з найбільш використовуваних мов програмування сьогодні.

Опитування, проведене аналітичною фірмою RedMonk, показало, що це була найпопулярніша мова програмування серед розробників у 2020 році.

Python зазвичай використовується для розробки веб-сайтів і програмного забезпечення, автоматизації завдань, аналізу даних і візуалізації даних. Оскільки його відносно легко навчитися, Python був прийнятий багатьма непрограмістами, такими як бухгалтери та вчені, для виконання різноманітних повсякденних завдань, наприклад, організації фінансів.

Python став основним продуктом науки про дані, дозволяючи аналітикам даних та іншим фахівцям використовувати цю мову для проведення складних статистичних обчислень, створення візуалізації даних, створення алгоритмів машинного навчання, маніпулювання та аналізу даних, а також виконання інших завдань, пов’язаних з даними.

Python може створювати широкий спектр різноманітних візуалізацій даних, таких як лінійні та стовпчасті графіки, кругові діаграми, гістограми та тривимірні графіки. Python також має ряд бібліотек, які дозволяють програмістам писати програми для аналізу даних і машинного навчання швидше та ефективніше, як-от TensorFlow і Keras.

Python часто використовується для розробки задньої частини веб-сайту або програми — частин, які користувач не бачить. Роль Python у веб-розробці може включати надсилання даних на сервери та з них, обробку даних і зв’язок з базами даних, маршрутизацію URL-адрес і забезпечення безпеки. Python пропонує кілька фреймворків для веб-розробки. Зазвичай використовуються Django і Flask.

У розробці програмного забезпечення Python може допомогти в таких завданнях, як контроль збірки, відстеження помилок і тестування. За допомогою Python розробники програмного забезпечення можуть автоматизувати тестування нових продуктів або функцій. Деякі інструменти Python, які використовуються для тестування програмного забезпечення, включають Green і Requestium.

Python призначений не тільки для програмістів і науковців, що вивчають дані. Вивчення Python може відкрити нові можливості для тих, хто має менш важкі професії, як-от журналістів, власників малого бізнесу або маркетологів у соціальних мережах. Python також може дозволити непрограмістам спростити певні завдання в їхньому житті. Ось лише кілька завдань, які можна автоматизувати за допомогою Python:

- Слідкуйте за цінами на фондовому ринку або криптовалютах;
- Надішліть собі текстове нагадування носити парасольку під час дощу;
- Оновіть свій список покупок;
- Перейменування великих пакетів файлів;
- Перетворення текстових файлів у електронні таблиці;
- Довільно розподіляйте обов’язки між членами сім’ї;
- Автоматично заповнюйте онлайн-форми.

Python популярний з кількох причин:

- Він має простий синтаксис, який імітує природну мову, тому його легше читати та розуміти. Це дозволяє швидше створювати проекти та швидше їх покращувати.
- Він універсальний. Python можна використовувати для багатьох різних завдань, від веб-розробки до машинного навчання.

- Він зручний для початківців, що робить його популярним для програмістів початкового рівня.
- Це відкритий вихідний код, що означає, що його можна безкоштовно використовувати та поширювати навіть у комерційних цілях.
- Архів модулів і бібліотек Python — пакети коду, які сторонні користувачі створили для розширення можливостей Python — величезний і зростає.
- Python має велику та активну спільноту, яка вносить свій внесок у пул модулів і бібліотек Python, а також є корисним ресурсом для інших програмістів. Велика спільнота підтримки означає, що якщо кодери стикаються з каменем спотикання, знайти рішення відносно легко; хтось обов'язково стикався з такою ж проблемою раніше.[1]

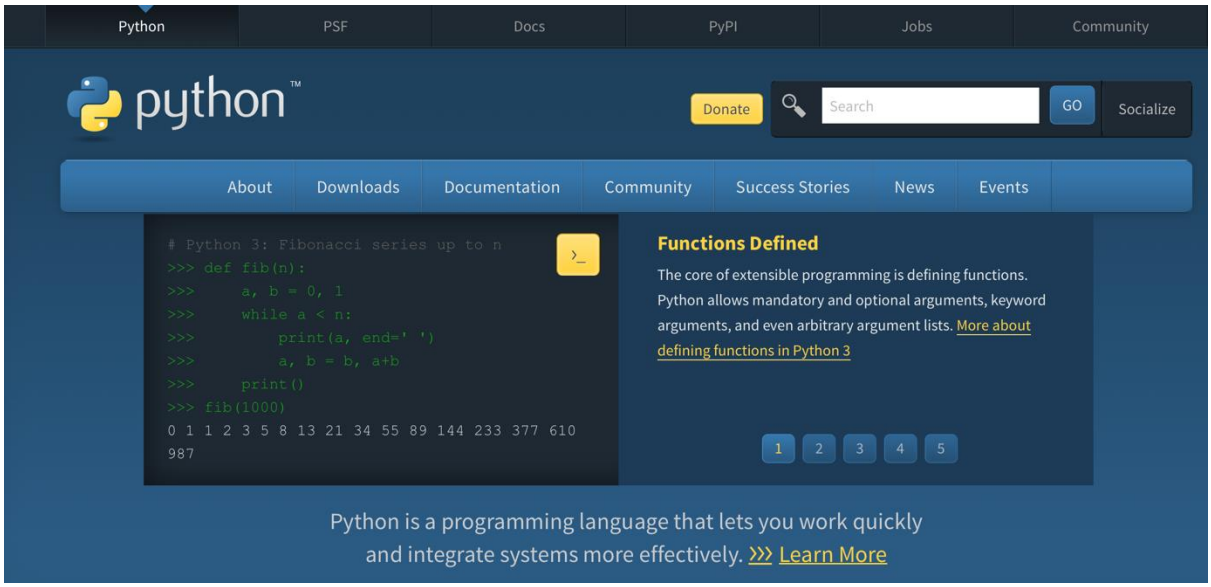


Рис. 2.3 - Python

2.11. Середа розробки

PyCharm — це спеціальне інтегроване середовище розробки Python (IDE), що надає широкий спектр важливих інструментів для розробників

Python, тісно інтегрованих для створення зручного середовища для продуктивної розробки Python, Інтернету та науки про дані.

2.12. Модулі

- TelegramBotApi

Ця бібліотека надає чистий інтерфейс Python для API бота Telegram. Він працює з версіями Python від 2.6+. Він також працює з Google App Engine.

- BeautifulSoup

Beautiful Soup — це бібліотека, яка дозволяє легко витягувати інформацію з веб-сторінок. Він розташовано на вершині аналізатора HTML або XML, надаючи ідіоми Pythonic для ітерації, пошуку та модифікації дерева аналізу.

- Requests

Оскільки під час використання API надсилаються запити HTTP та відповіді, бібліотека Requests відкриває можливість використання API у Python.

Запити HTTP є основою всесвітньої мережі. Щоразу, коли ви відкриваєте веб-сторінку, ваш браузер надсилає безліч запитів на сервер цієї веб-сторінки. Сервер відповідає на них, надсилаючи всі необхідні дані для виведення сторінки, і ваш браузер відображає сторінку, щоб ви могли побачити її.

В цілому цей процес виглядає так: клієнт (наприклад, браузер або скрипт Python, що використовує бібліотеку Requests) відправляє дані на URL, а сервер з цим URL зчитує дані, вирішує, що з ними робити, і надсилає клієнту відповідь. Після цього клієнт може вирішити, що робити з даними, отриманими у відповіді.

У складі запиту клієнт надсилає дані за методом запиту. Найбільш поширеними методами запиту є GET, POST та PUT. Запити GET зазвичай призначені лише читання даних без їх зміни, а запити POST і PUT зазвичай призначаються зміни даних на сервері.

- Time

  Time - модуль для роботи з часом в Python.

- SQLite

  SQLite — це автономна файлова база даних SQL. SQLite постачається в комплекті з Python і може використовуватися в будь-якій вашій програмі Python без необхідності встановлення будь-якого додаткового програмного забезпечення.

- Aiogram

  Aiogram — це досить простий і повністю асинхронний фреймворк для Telegram Bot API, написаний на Python 3.7 з asyncio і aiohttp. Це допоможе вам зробити ваших ботів швидшими та простішими.

3. РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ АНАЛІЗУ ТЕКСТОВИХ ДАНИХ З САЙТІВ ТЕРИТОРІАЛЬНИХ ГРОМАД НА ОСНОВІ БОТ-ПАРСЕРІВ

3.1. Створення парсера

Спочатку потрібно вирішити, який громадський сайт потрібно парсити. Це буде «https://www.minregion.gov.ua/». Сайт містить багато новин зі всієї країни та оновлює стрічку новин багато раз у день, що добре, тому що бот буде перевіряти інформацію з даного сайту багато раз у день.

Зображення сайту нижче на рисунку 3.1.

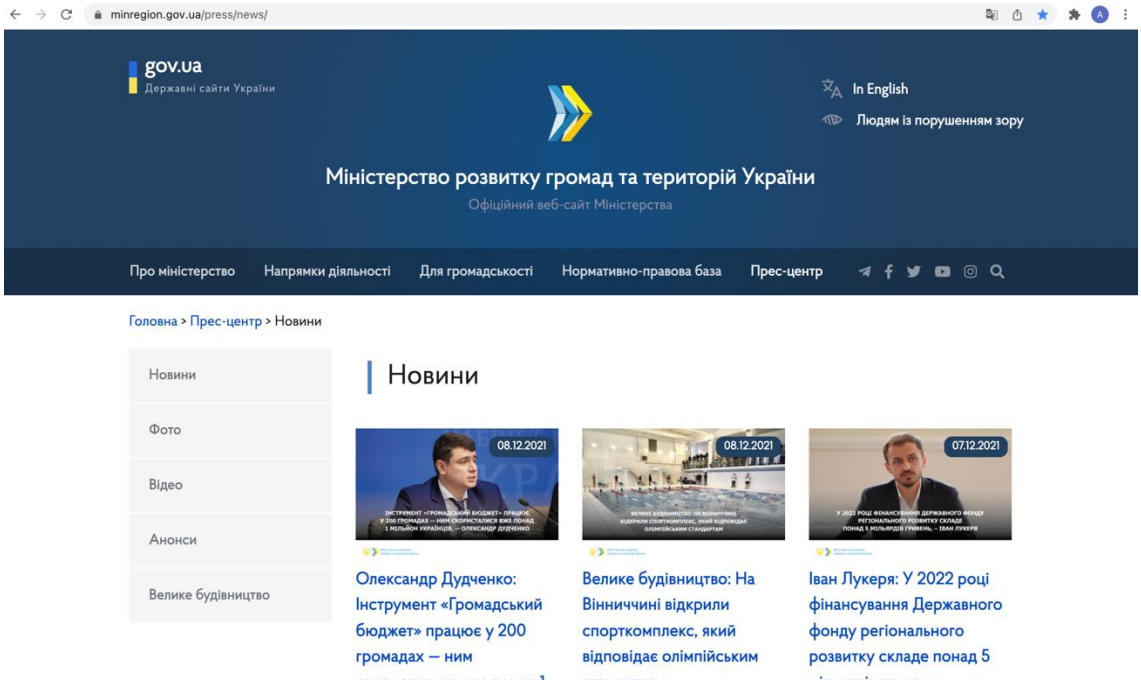


Рис. 3.1 – Громадський сайт

На сайті потрібно перейти до вкладки, де публікуються новини, а саме «https://www.minregion.gov.ua/press/news/». Наступним кроком буде знайти

елементи потрібних об’єктів. Для цього потрібно скористатися кодом потрібної сторінки.

Код даної сторінки можна побачити нижче на рисунку 3.2.

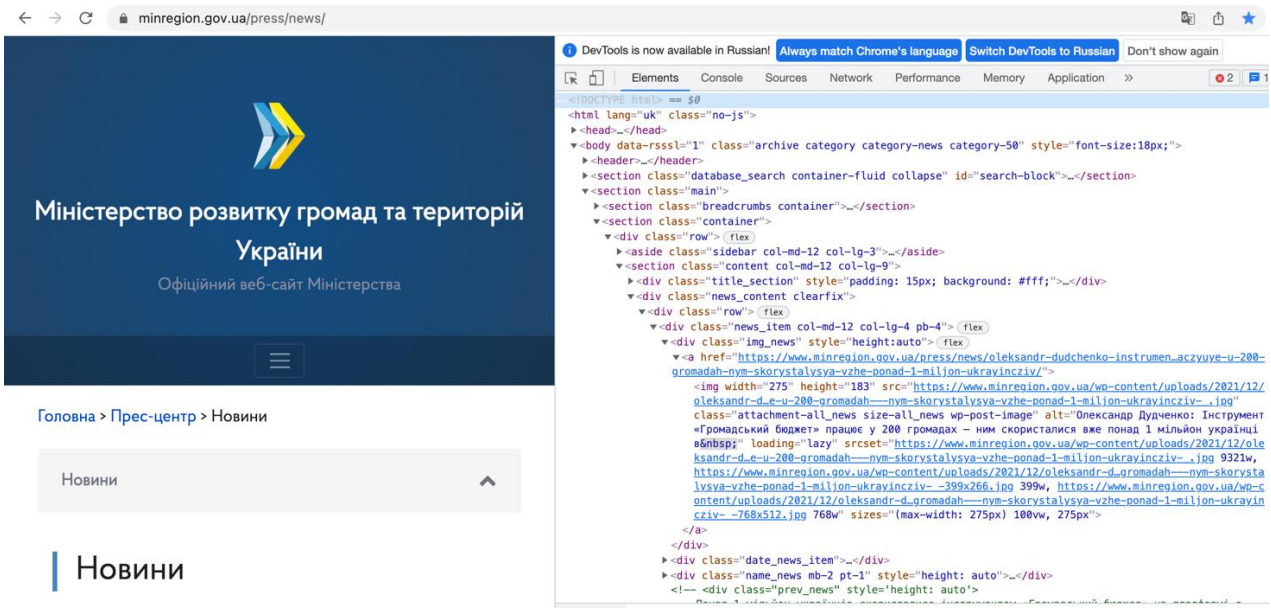


Рис. 3.2 – Код сторінки

Щоб знайки потрібний нам елемент, тобто актуальну статтю, нам потрібно знайти клас, який містить назву, описання, зображення, посилання та дату на цю новину. Достатньо щось одне.

Приклад на рисунку 3.3.

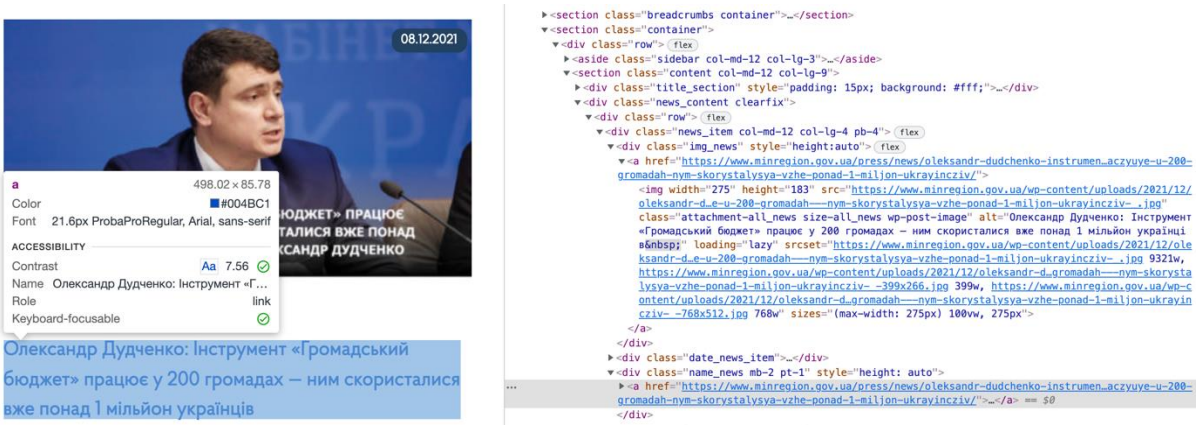


Рис. 3.3 – Класи елементів

Можна одразу побачити, що кожний елемент на сторінці має свій блок «div» та клас «class». Це головні елементи, які потрібні для парсингу. Назва даної статті знаходиться в блоці «div» та має «class = «name_news mb-2 pt-1»».

Саме завдяки цьому можна витягнути будь-яку потрібну інформацію з будь-якого сайту. Хоч це текст, зображення або посилання.

Залишається додати словник з декількома громадськими сайтами. Додаємо словник, а не кортеж, щоб у майбутньому користувач міг додати потрібні для нього сайти.

3.2. Тестування звичайного парсеру

Спробуємо витягнути якийсь елемент з сайту для тестування звичайного парсеру. Тестування зробимо через термінал. Приклад звичайного парсеру можна побачити у ДОДАТКУ В.

Результат актуального пошуку зображено нижче на рисунку 3.4.

```
Last login: Wed Dec  8 11:41:14 on console
aleksandrarefev@MacBook-Pro-Aleksandr ~ % cd Downloads
aleksandrarefev@MacBook-Pro-Aleksandr Downloads % python3 parse.py
Наталія Козловська під час зустрічі з архітекторами: Інклюзивність будівель має
стати єдиним стандартом для всіх громад України
Дата: 08.12.2021
Деталььніше: https://www.minregion.gov.ua/press/news/nataliya-kozlovska-pid-chas
-zustrichi-z-arhitektoramy-inklyuzyvnist-budivel-maye-staty-yedynym-standartom-d
lya-vsih-gromad-ukrayiny/
aleksandrarefev@MacBook-Pro-Aleksandr Downloads %
```

Рис. 3.4 – Пошук актуальної новини

Як можна побачити, вдалося витягнути не лише назву головної новини, але і дату с посиланням. Завдяки такій інформації як дата, яка теж

зберігається у своєму блоці на сторінці коду, можна реалізувати сортування новин, що дасть можливість дізнатися потрібні статті.

Доповнимо код додатковою інформацією для зручності. Та доповнимо додатковими функціями для сортування інформації.

Окрім сортування по даті, можна додати сортування за релевантністю, тобто пошук інформації, який означає рівень відповідності знайденого документа або набору документів інформаційним потребам користувача.

3.3. Створення боту

Як можна було почати раніше, для створення боту потрібно мати месенджер Telegram та знайти у пошуку Botfether.

Звертаємося до головного бота за допомогою команди «Start». Наступним кроком обираємо команду «Newbot», що допоможе створити власного бота, якому можна дати назву, власний унікальний токен, опис, власну картинку, підключити до публічної групи, щоб була можливість парсити новини для великої кількості людей. Завдяки Botfether можна змінювати велку кількість фунцкції, які відоносяться до бота. Це є дуже важливливим інструментом для розробників.

Що дасть можливість монетизувати даний проект у майбутньому, завдяки тому, що в публічних групах підключають рекламні інтеграції.

Це зображено нижче на рисунку 3.5.

Рисунок 3.5 – Створення боту

3.4. Огортання парсеру. Бот-парсер

Завдяки бібліотеці Aiogram і TelegramBotApi огорнемо парсер у бота Telegram. Використовуємо унікальний ТОКЕН, який видав Botfather. Змінюємо звертання до боту та отримання зворотної інформації.

Використовуємо функції «bot.pooling()» для того, щоб бот отримував звертання до нього на сервер.

Реалізації даної частини можна побачити у ДОДАТКУ Б.

3.5. База даних

Для повної зручності користуванням ботом зробимо базу даних користувачів та теги, які будуть використовувати користувачі.

База даних не буду складною. Реалізації бази даних допоможе зрозуміти боту чи зареєстрований користувач у даній системі. Статус 1 – є у базі, статус 0 – ні. Якщо користувач має статус 1, тоді користування дозволено. База даних створена за допомогою бібліотеки SQLite.

3.6. Словник тегів

Далі за допомогою словника створюємо базу даних тегів. Наприклад, COVID, економіка, статистика, термінові новини тощо.

Даний словник користувач може поповнювати потрібними для нього тегами, які у майбутньому будуть потрібні для нього.

Завдяки тегам можна швидко знайти новини, до яких відноситься даний тег. Зрозуміло, що всі новини, які пов’язані з COVID-19, будуть мати тег «#ковід» тощо.

3.7. Процес звертання до сайту громадських новин

Модуль парсинга витягує потрібну інформацію з потрібних джерел, тобто сайтів, але це можуть бути не лише сайти. Наприклад інші Telegram канали.

Вхідні дані поступають від користувача до веб інтерфейсу, де веб інтерфейс взаємодіє з модулем парсингу.

Далі модуль парсингу відправляє результати до модуля експорту. Де останній заповнює та відсилає потрібну інформацію до бази даних або до файлів потрібних форматів. Наприклад CSV, XML, JSON тощо.

Як відбуваються парсинг сайту можна побачити нижче на рисунку 3.6.

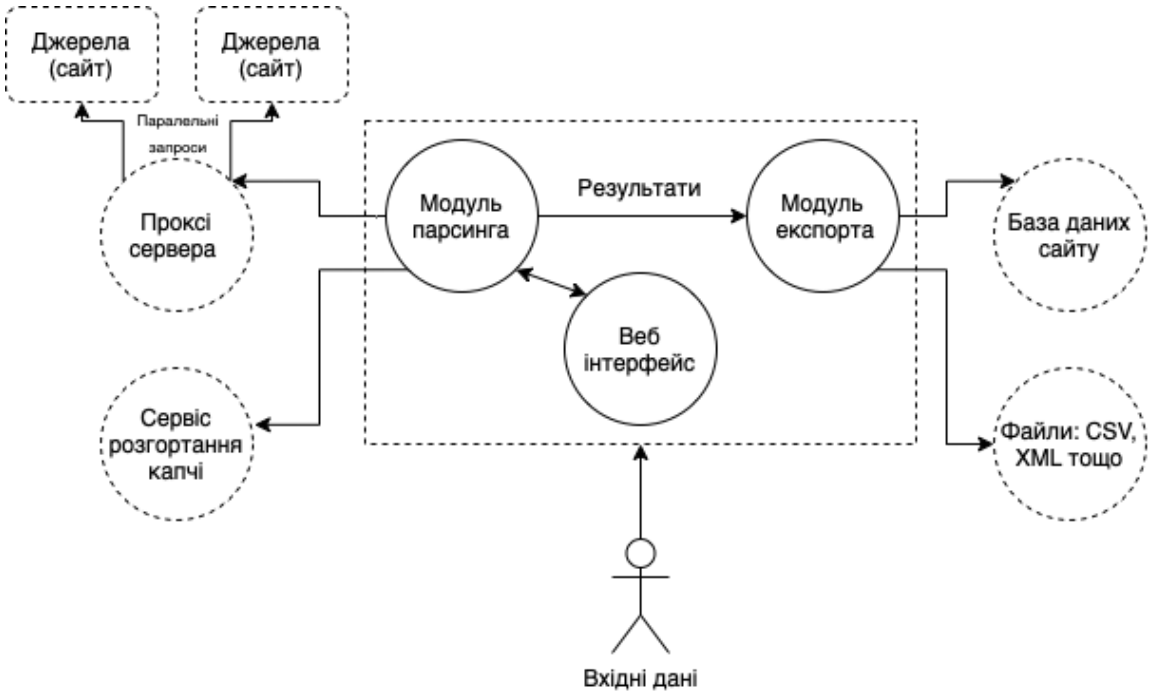


Рис. 3.6 – Парсинг

3.8. Час оновлення стрічки

Щоб відстежувати останні новини, скористаємося бібліотекою time. Даний модуль дозволить налаштувати час оновлення стрічки, завдяки чому можна буде налаштувати бота на те, щоб надсилання новин відбувалося кожні 30 хвилин. Це допоможе дізнаватися нові та актуальні статті і навіть не звертатися до бота самостійно. Бот все зробіть сам.

3.9. Запит до бота

На рисунку 3.8 можна побачити як відбується запит до бота від користувача. Спочатку відбувається звернення до бота, наступним кроком відбувається додання клієнта до бази даних та запит потрібних йому тегів. Через декілька секунд користувач отримує актуальну або потрібну йому інформацію.

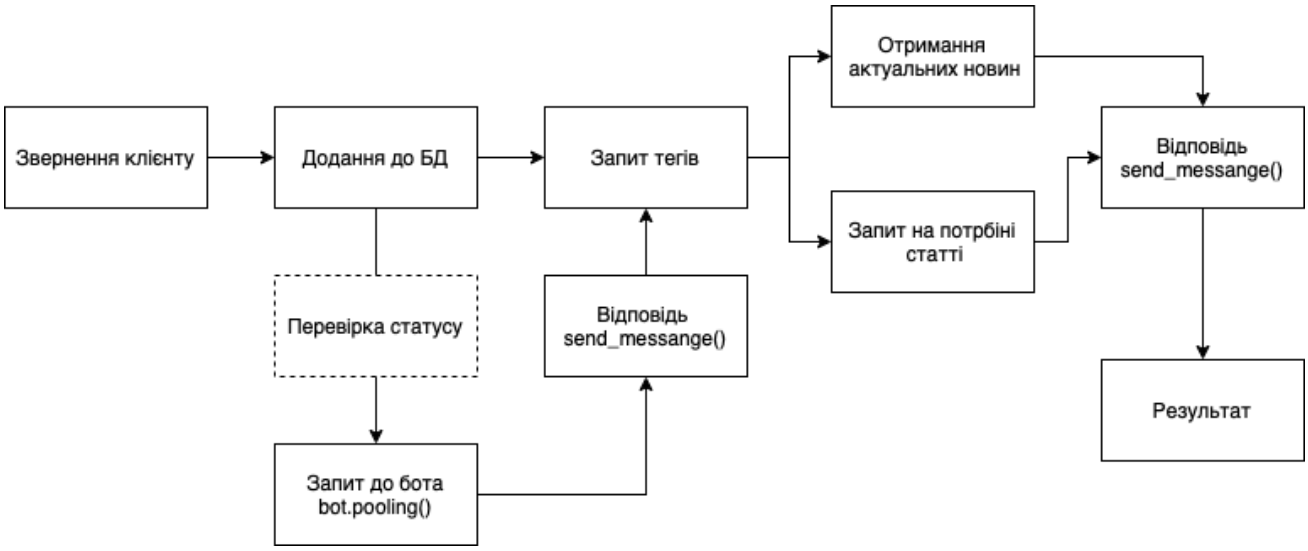


Рис. 3.7 – Звернення до бота

3.10. Запит і результат

Почнемо пошук по тегу, який нас цікавить. Наприклад, тег «ковід», шукаємо актуальну інформацію. По запиту «ковід», бот знайде останню статтю, яка зв’язана с цим тегом і зворотнім зв’язком відправить до чату з користувачем.

На рисунку 3.8, який зображено нижче, можна побачити результат пошуку за відповідним тегом «ковід».

На рисунку 3.9 і 3.10 зображено більш детальніше.

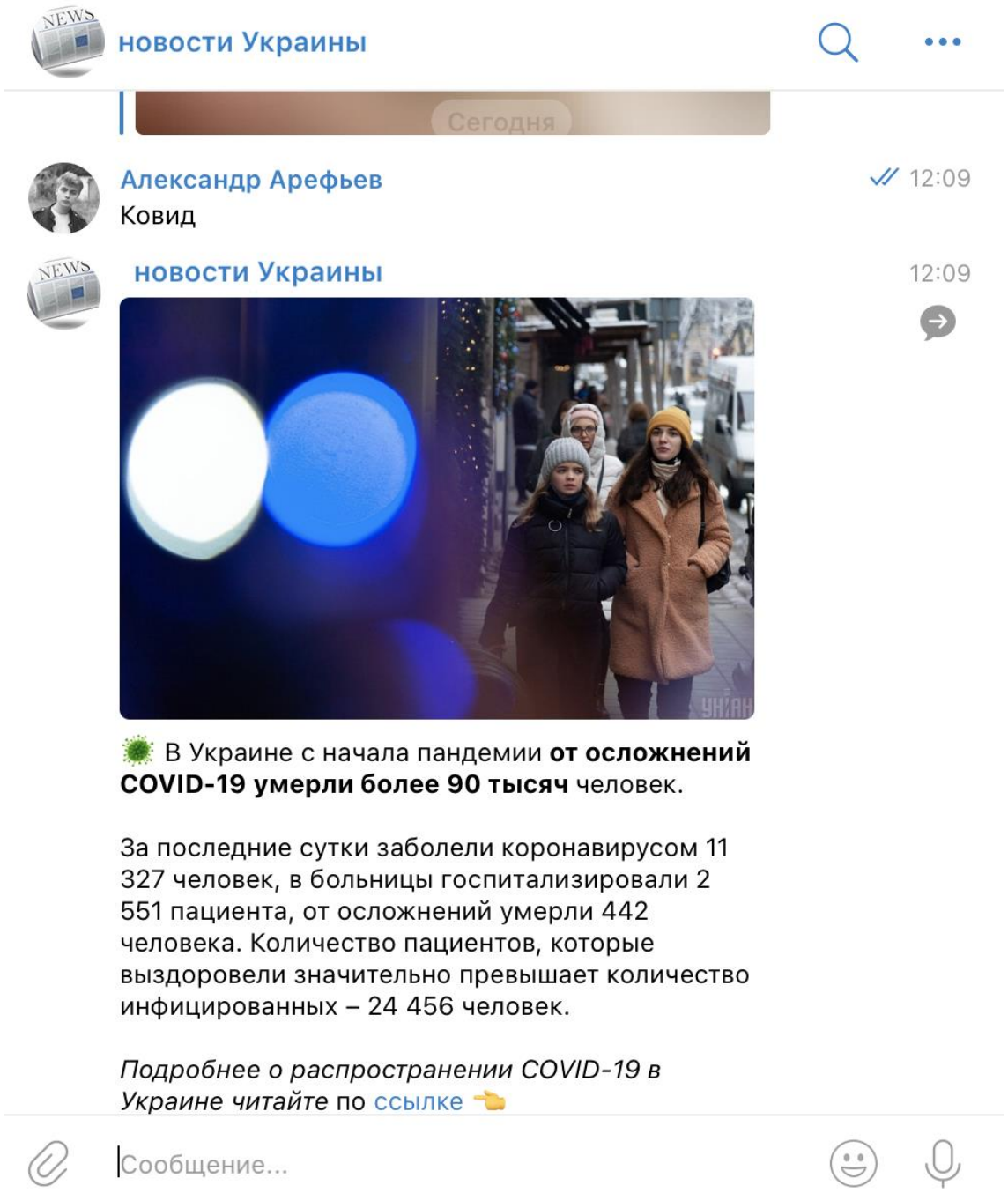


Рис. 3.8 – Запит і результат

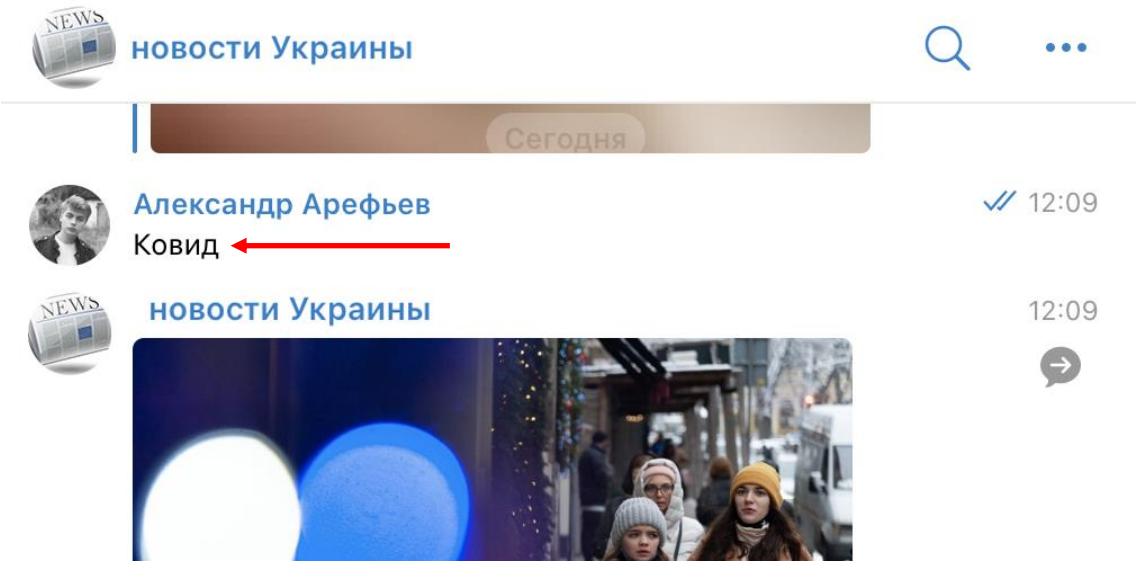


Рис. 3.9 – Тег

Рис. 3.10 – Результат пошуку

Як можна побачити, вдалося зробити парсинг не лише текстових новин. Вийшло додати зображення статі, посилання на джерело, назву та короткий опис статті, що робить бота більш функціональним та зручним для користування.

ВИСНОВКИ

Мета даної роботи полягала у розробці інформаційної системи аналізу текстових даних з сайтів територіальних громад на основі бот-парсерів. Для досягнення цієї мети було визначені такі завдання:

1. Вивчити технології парсинга для підвищення ефективності при розробці даної задачі та для зручності користуванням даним інструментом.
2. Вивчити технології розробки ботів для месенджера Telegram для підвищення ефективності при розробці.
3. Проаналізувати, яка мова буде зручна для розробки даної системи. Знайти та проаналізувати потрібні модулі для розробки бот-парсеру.
4. Розробити дану систему для пошуку новин на територіальних сайтах на основі месенджер боту.
5. Провести тестування розробленого бот-парсера.

Відповідно о поставлених завдань було зроблено такі висновки:

1. Вивчено технологію парсингу, у результаті можна сказати, що ця технологія підходить для синтаксичного аналізу текстових даних. Технологія парсига застосовується для зчитування великої кількості неструктурованої інформації для приведення до необхідного формату даних. Технологія парсингу дозволяє оптимізувати часові витрати на обробку даних.
2. Вивчено технології розробки ботів для месенджера Telegram. Саме використання даного бота оптимізує часові витрати користувача і дозволяє знайти потрібну інформацію.
3. Розробка даної задачі була виконана на мові програмування Python. Завдяки даній мові можна реалізувати поставлену задачу швидше і зручніше, а також простіше знайти потрібні модулі для виконання задачі. Бібліотеки для розробки бот-парсеру:

3.11. TelegramBotApi

3.12. Aiogram

3.13. Time

3.14. BeautifulSoup

3.15. Requests

4. Розроблений бот-парсер призначений для пошуку текстової інформації на сайтах територіальних новин.

5. Проведено тестування розробленого бот-парсеру, де можна побачити, що пошук новин на сайтах територіальних громад відбувається добре та заходить актуальні статті, які потрібні користувачу.

Таким чином, завдання роботи вирішено.

ДЖЕРЕЛА

1. Python. Мова високого рівня. Режим доступу: www. URL: https://uk.wikipedia.org/wiki/Python

2. Боти. Загальна інформація. Режим доступу: www. URL: https://ru.wikipedia.org/wiki

3. Що таке боти і як вони працюють. Режим доступу: www. URL: https://www.dw.com/ru/что-такое-боты-и-как-они-работают/a-41716550

4. Боти у сервісі Telegram. Для чого вони потрібні? Режим доступу: www. URL: https://techtoday.in.ua/news/shho-take-boti-v-servisi-telegram-ta-yak-nimi-koristuvatisya-64222.html

5. Парсер. Синтаксичний аналізатор. Режим доступу: www. URL: https://ru.wikipedia.org/wiki/Синтаксический_анализатор

6. Уява про парсери. Режим доступу: www. URL: https://www.activetraffic.ru/wiki/parser/

7. Парсинг. Що це і де використовують. Режим доступу: www. URL: https://ipipe.ru/info/parsing

8. Парсинг даних сайтів. Режим доступу: www. URL: https://blog.ringostat.com/ru/parsing-dannyh-s-saytov-chto-eto-i-zachem-on-nuzhen/

9. Сортування даних у діапазоні або таблиці. Режим доступу: www. URL: https://support.microsoft.com/uk-ua/office/сортування-даних-у-діапазоні-або-таблиці-62d0b95d-2a90-4610-a6ae-2e545c4a4654

10. Telegram. Режим доступу: www. URL: https://core.telegram.org/bots/api

11. Месенджер Telegram. Режим доступу: www. URL: https://telegram.org/faq

12. Месенджери. Режим доступу: www. URL: https://www.voipoffice.ru/tags/messendzhery/

13. Інструменти для парсинга. Режим доступу: www. URL: https://habr.com/ru/post/340038/

14. З чого почати парсити. Режим доступу: www. URL: https://qna.habr.com/q/254656

ДОДАТОК А

```
import bs4
import requests



class Basic:
    def parse(self, link):
        if link['type'] == 'stickers':
            return self.parse_stickers(link['link'])
        if link['type'] == 'user':
            return self.parse_user(link['link'])
        if link['type'] == 'bot':
            return self.parse_bot(link['link'])
        raise NotImplementedError(f'Type {link["type"]} not implemented')


    def parse_user(self, link):
        url = 'https://t.me/' + link  # getting data from link
        r = requests.get(url, stream=True)
        soup = bs4.BeautifulSoup(r.text, "lxml", )
        type_link = str(soup.find_all('a', class_="tgme_action_button_new"))
        members_str = str(soup.find_all('div', class_="tgme_page_extra"))
        if 'Preview channel' in type_link:
            return {
                'type': 'channel',
                'link': url
            }
        if 'Preview channel' not in type_link and 'members' in members_str:  # check for group
            return {
                'type': 'group',
                'link': url
            }
        if 'tgme_action_button_new' in type_link and 'member' not in members_str and 'Send Message'
in type_link:
            return {
                'type': 'user',
                'link': url
            }
        return None


    def parse_stickers(self, link):
```

```
        url_stickers = 'https://t.me/addstickers/' + link  # getting data from link
        r_stickers = requests.get(url_stickers, stream=True)
        soup_stickers = bs4.BeautifulSoup(r_stickers.text, "lxml", )
        type_link = str(soup_stickers.find_all('div', class_="tgme_page_description")).replace(u'\xa0', '
').replace(';',
                                                                                ':')
        if re.search('Sticker Set', type_link):  # check for channel
            return None
        return {
            'type': 'stickers',
            'link': url_stickers
        }


    def parse_bot(self, link):
        url_bot = 'https://t.me/' + link
        r_bot = requests.get(url_bot, stream=True)
        soup_bot = bs4.BeautifulSoup(r_bot.text, "lxml", )
        type_link = soup_bot.find_all('div', class_="tgme_page_extra")
        if type_link != []:
            return {
                'type': 'bot',
                'link': url_bot
            }
        return None




class FullInfo:
    def parse(self, link):
        if link['type'] == 'stickers':
            return self.parse_stickers(link['link'])
        if link['type'] == 'user':
            return self.parse_user(link['link'])
        if link['type'] == 'bot':
            return self.parse_bot(link['link'])
        raise NotImplementedError(f'Type {link["type"]} not implemented')


    def parse_user(self, link):
        title = None
        description = None
        members = None
        url = 'https://t.me/' + link  # getting data from link
        s = requests.Session()
```

```
r = s.get(url, stream=True)
soup = bs4.BeautifulSoup(r.text, "lxml", )
type_link = str(soup.find_all('a', class_="tgme_action_button_new"))
members_str = str(soup.find_all('div', class_="tgme_page_extra"))


try:
    title = str(soup.find('div', class_="tgme_page_title").text)[
        1:-1].replace(';', ':')
    try:
        description = str(
            soup.find('div', class_="tgme_page_description").text).replace(';', ':')
    except:
        pass
except AttributeError:
    return None


if re.search('Preview channel', type_link):  # check for channel
    members_int = re.findall(r'\d+', members_str)
    members = ''
    members = members.join(members_int)
    if members == '':
        members = '0'
    return {
        'type': 'channel',
        'link': url,
        'title': title,
        'description': description,
        'members': members
    }


if 'Preview channel' not in type_link and 'members' in members_str:  # check for group
    members_str = members_str.split(',')[0]
    members_int = re.findall(r'\d+', members_str)
    members = ''
    members = members.join(members_int)
    if members == '':
        members = '0'
    return {
        'type': 'group',
        'link': url,
        'title': title,
        'description': description,
        'members': members
```

```
            }


        if 'tgme_action_button_new' in type_link and 'member' not in members_str and 'Send Message'
in type_link:
            return {
                'type': 'user',
                'link': url,
                'title': title,
                'description': description
            }
        return None


    def parse_stickers(self, link):
        url_stickers = 'https://t.me/addstickers/' + link  # getting data from link
        r_stickers = requests.get(url_stickers, stream=True)
        soup_stickers = bs4.BeautifulSoup(r_stickers.text, "lxml", )
        type_link = str(soup_stickers.find_all('div', class_="tgme_page_description")).replace(u'\xa0', '
').replace(';',
                                                                                    ':')
        if re.search('Sticker Set', type_link):
            return None


        start_name = [(m.start(0), m.end(0))
                      for m in re.finditer("<strong>", type_link)][1][1]
        end_name = [(m.start(0), m.end(0))
                    for m in re.finditer("</strong>", type_link)][1][0]
        title_stickers = type_link[start_name:end_name]
        return {
            'type': 'stickers',
            'link': url_stickers,
            'title': title_stickers
        }


    def parse_bot(self, link):
        url_bot = 'https://t.me/' + link
        r_bot = requests.get(url_bot, stream=True)
        soup_bot = bs4.BeautifulSoup(r_bot.text, "lxml", )
        type_link = soup_bot.find_all('div', class_="tgme_page_extra")
        if type_link != []:
            title_bot = soup_bot.find('div', class_='tgme_page_title').text
            try:
                description_bot = soup_bot.find(
```

```
                'div', class_='tgme_page_description').text
            except:
                description_bot = None
            return {
                'type': 'bot',
                'link': url_bot,
                'title': title_bot,
                'description': description_bot
            }
        return None
```

ДОДАТОК Б

```
import telebot
import requests
import time
from bs4 import BeautifulSoup

bot = telebot.TeleBot('2123273069:AAHg5m8kveKHlSzeEAKxhbBM740SWCTVmmw')

HEADERS = {
        'user-agent' : 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML,
        like Gecko) Chrome/96.0.4664.45 Safari/537.36'
}

@bot.message_handler(content_types = ['text'])
def news(message):
        if message.text == '/start':
        bot.send_message(message.chat.id, perse)



def perse():

URL = 'https://www.minregion.gov.ua/press/news/'

page = requests.get(URL)
soup = BeautifulSoup(page.content, 'html.parser')

post = soup.find('div', class_= 'news_content clearfix')


title = post.find('div', class_= 'name_news mb-2 pt-1').text.strip()
```

```
description = post.find('span', class_= '').text.strip()
url = post.find('a', class_= '', href = True)["href"].strip()



bot.polling()



#import requests
#from bs4 import BeautifulSoup

#URL = 'https://www.minregion.gov.ua/press/news/'

#page = requests.get(URL)
#soup = BeautifulSoup(page.content, 'html.parser')

#post = soup.find('div', class_= 'news_content clearfix')



#title = post.find('div', class_= 'name_news mb-2 pt-1').text.strip()
#description = post.find('span', class_= '').text.strip()
#url = post.find('a', class_= '', href = True)["href"].strip()

#print(title,'\n' 'Дата:', description,'\n' 'Деталььніше:', url)
```

ДОДАТОК В

```
#import requests
#from bs4 import BeautifulSoup

#URL = 'https://www.minregion.gov.ua/press/news/'

#page = requests.get(URL)
#soup = BeautifulSoup(page.content, 'html.parser')

#post = soup.find('div', class_= 'news_content clearfix')



#title = post.find('div', class_= 'name_news mb-2 pt-1').text.strip()
#description = post.find('span', class_= '').text.strip()
#url = post.find('a', class_= '', href = True)["href"].strip()
```

```
#print(title,'\n' 'Дата:', description,'\n' 'Деталььніше:', url)
```

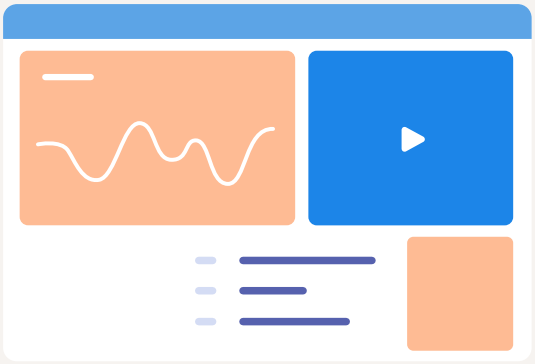
ДОДАТОК Г

Інформаційна система аналізу текстових даних з сайтів територіальних громад на основі ботів-парсерів

Арєф'єв Олександр Павлович
група ІСТ-20дм

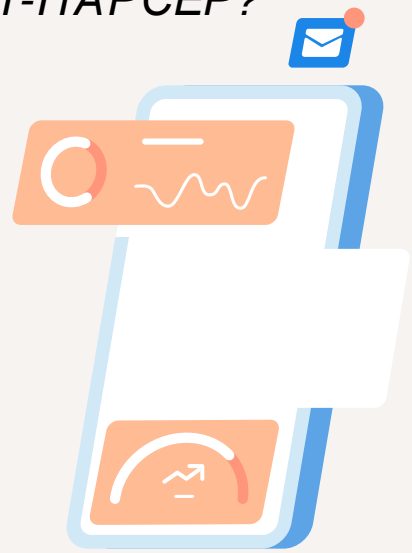
АКТУАЛЬНІСТЬ

- *БАГАТО ЗАЙВОЇ ІНФОРАМЦІЇ*
- *ВТРАТА ПОТРІБНИХ НОВИН*
- *ВРАТА ВЛАСНОГО ЧАСУ НА ПОШУК*
- *ВІДСТЕЖЕННЯ БАГАТЬОХ ДЖЕРЕЛ*



ЧОМУ САМЕ TELEGRAM БОТ-ПАРСЕР?

- НАДАННЯ АКТУАЛЬНАЯ ІНФОРМАЦІЇ
- ФІЛЬТРАЦІЯ ІНФОРМАЦІЇ
- ПОШУК ПОТРІБНОЇ НОВИНИ ПО ТЕГАМ
- ЗБЕРЕЖЕННЯ ЧАСУ
- ЗРУЧНІСТЬ

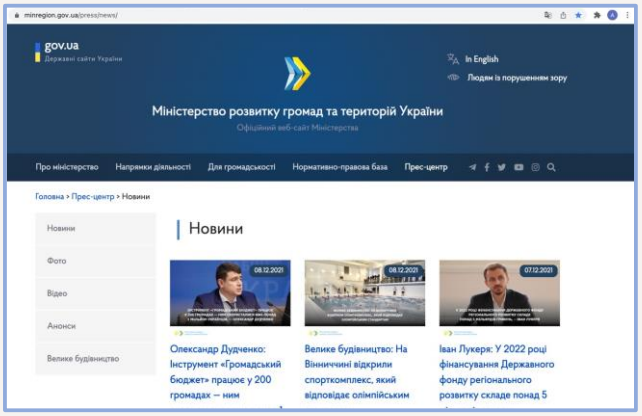


МЕТОДИ РОЗРОБКИ

- ЯЗИК ПРОГРАМУВАННЯ PYTHON
- БІБЛІОТЕКА BeautifulSoup для сортування інформації
- БІБЛІОТЕКА Requests для звернення до інформації на HTML сторінки
- БІБЛІОТЕКА Time для пошуку и оновлення по часу

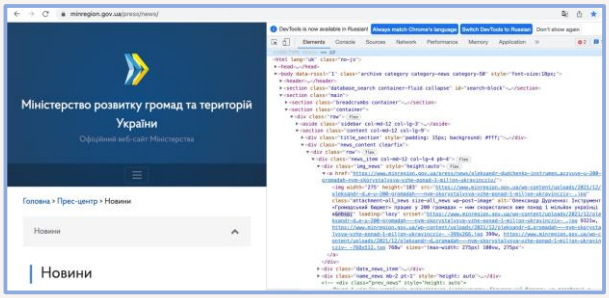
ПАРСИНГ САЙТУ ТЕРИТОРІАЛЬНИХ ГРОМАД

ГОЛОВНА СТОРІНКА НОВИН ТЕРИТОРІАЛЬНОГО САЙТУ ДЛЯ ПАРСИНГУ БОТОМ



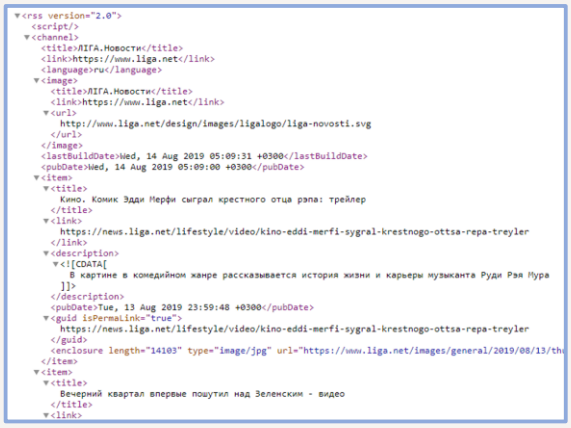
ПАРСИНГ ІНФОРМАЦІЇ

КОД ГОЛОВНОЇ СТОРІНКИ ТЕРИТОРІАЛЬНОГО САЙТУ, ЗАВДЯКИ ЧОМУ МОЖНА ВИТЯГНУТИ ВСЮ ПОТРІБНУ ІНФОРМАЦІЮ (ТЕКСТОВУ ІНФОРМАЦІЮ, ЗОБРАЖЕННЯ, ПОСИЛАННЯ ТОЩО)



АЛЬТЕРНАТИВНІ РІШЕННЯ

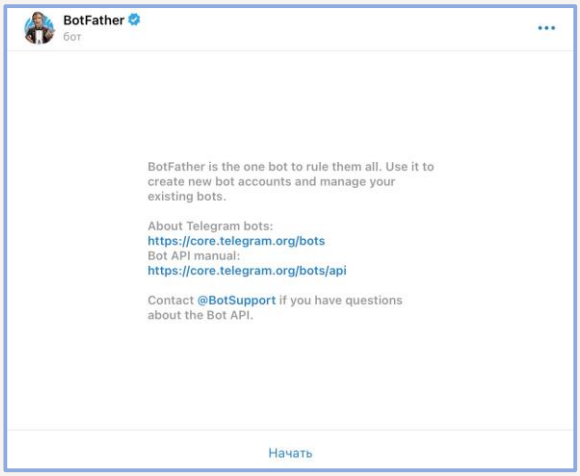
RSS РІШЕННЯ. Автоматично генероване зведення у форматі rss або xml, в якому відображаються нещодавно опубліковані статті та новини. При цьому на повну версію вказаних матеріалів дається гіперпосилання. Дуже часто цей формат використовується інформаційними порталами та блогами. RSS-стрічку можна підключити до Google News, Яндекс.Дзен тощо.



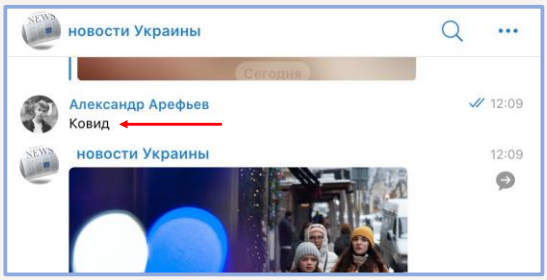
РОЗРОБКА БОТУ

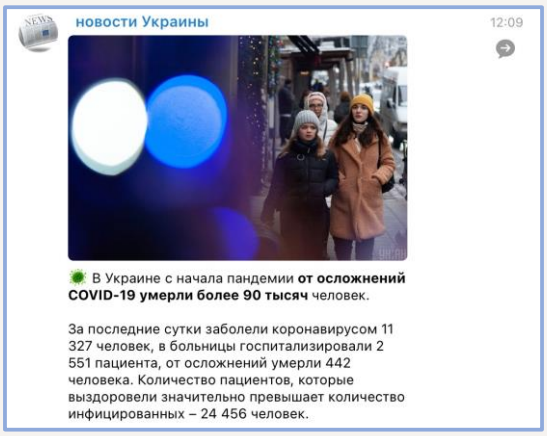
BotFather. Інструмент для створення боту і отримання ТОКЕНУ для подальшої розробки.

БІБЛІОТЕКИ: Requests, BeautifulSoup, Time, Aiogram, TelegramBotApi.



РЕЗУЛЬТАТИ ПОШУКУ





ДЯКУЮ ЗА УВАГУ!