НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
«__»_____2021 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»
спеціальності 121 «Інженерія програмного забезпечення»

на тему: Медіа агрегатор з динамічним налаштуванням критерію відбору контенту

Виконав: студент 4 курсу, групи ІП-73

Іващук Владислав Андрійович_____ _____
(прізвище, ім'я та по батькові) (підпис)

Керівник: д.т.н., професор Новотарський М. А._____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль): _____ _____
(назва розділу) (посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент: _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2021 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

«Інженерія програмного забезпечення комп'ютерних систем»

спеціальність 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
"__" _____2021 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Іващука Владислава Андрійовича

1. Тема проєкту Медіа агрегатор з динамічним налаштуванням критерію відбору контенту
керівник проєкту д.т.н., професор Новотарський М. А.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 11.05 2021 року № 1139-с
2. Термін здачі студентом закінченого проєкту 3 червня 2021 р.
3. Вихідні дані до проекту *технічна документація, прикладний програмний алгоритм.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)
Опис предметної області, аналіз існуючих програмних рішень, підбір інструментів для реалізації, система агрегації медіа контенту із соціальних мереж, тестування системи

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) діаграма класів, структурна схема програмного забезпечення, блок-схема алгоритмів роботи програм, діаграми послідовностей сценаріїв роботи з системою

6. Консультанта проекту, з вказівкою розділів проекту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормконтроль	Сімоненко В. П.		

7. Дата видачі завдання 17 грудня 2020 р.

Календарний план

№ п/п	Найменування етапів дипломного проекту	Терміни виконання етапів проекту	Примітки
1.	*Затвердження теми проекту*	*10.12.2020-17.12.2020*	
2.	*Вивчення та аналіз завдання*	*17.12.2020-07.02.2021*	
3.	*Розробка архітектури та загальної структури системи*	*07.02.2021-17.03.2021*	
4.	*Розробка структур окремих підсистем*	*17.03.2021-30.03.2021*	
5.	*Програмна реалізація системи*	*30.03.2020-04.05.2021*	
6.	*Оформлення пояснювальної записки*	*04.05.2021-22.05.2021*	
7.	*Виправлення помилок*	*22.05.2021-03.06.2021*	
8.	*Передзахист та проходження нормативного контролю*	*03.06.2021*	
9.	*Захист*	*19.06.2021*	

Студент-дипломник _____
(підпис)

Керівник проєкту _____
(підпис)

Анотація

В бакалаврському дипломному проєкті спроектовано і реалізовано систему агрегації медіа контенту із соціальних мереж з можливістю його відбору за встановленими критеріями.

Програма дозволяє вводити користувацькі налаштування отримуваного потоку публікацій та переглядати сформований потік у вікні чату месенджера Telegram.

Програмний продукт створено у вигляді Telegram бота на мові Python з використанням бібліотеки PyTelegramBotAPI та реалізовано його роботу методом поллингу. В якості інтерфейсу користувача використовується офіційний клієнт месенджера Telegram.

Annotation

In this project for a Bachelor’s Degree, a social media content aggregation system, which has and ability to select content by customizable criterias, is designed and implemented.

The program makes it possible to input user settings for a received post feed and to view a formed feed in the Telegram messenger chat window.

The software is created as a Telegram bot in Python language using PyTelegramBotAPI library, and its work is realized using polling method. The Telegram messenger official client is used for a user interface.

ОПИС АЛЬБОМУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Додаток
	А4		Завдання на дипломний проект	2	
	А4	ІАЛЦ.467200.001 ОА	Опис альбому	1	
	А4	ІАЛЦ.467200.002 ТЗ	Технічне завдання	4	
	А4	ІАЛЦ.467200.003 ПЗ	Пояснювальна записка	72	
	А4	ІАЛЦ.467200.004 Д1	Принципова схема додатку для клієнта	1	
	А4	ІАЛЦ.467200.005 Д2	Структурна схема	1	
	А4	ІАЛЦ.467200.006 Д3	Функціональна схема додатку для клієнта	1	
	А4	ІАЛЦ.467200.007 Д4	Лістинг програми	5	

Зм.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467200.001 ОА		
Розроб.		Іващук В. А.			*Медіа агрегатор з динамічним налаштуванням критерію відбору контенту* Опис альбому	Лит.	Арк 1 / Аркушів 1
Перев.		Новотарський М. А.					
Реценз.						НТУУ «КПІ». ФІОТ. ІП-73	
Н. Контр.		Сімоненко В. П.					
Затверд.		Стіренко С. Г.					

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломної роботи
освітньо-кваліфікаційного рівня бакалавр

на тему: «Медіа агрегатор з динамічним налаштуванням критерію відбору контенту»

Київ – 2021

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання розповсюджується на розробку програмного забезпечення потреб бізнесу та персональних потреб користувачів соцмереж. Область застосування: використання для оптимізації інформаційного потоку із соціальних мереж на підприємстві чи особисто.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи кваліфікаційно-освітнього рівня «бакалавр інженерії програмного забезпечення», затверджене кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проєкту є часткове рішення проблеми надлишкової та розосередженої інформації в соціальних мережах та оптимізація часу і процесу перегляду публікацій.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково технічна-література з теорії і практики проєктування програмного забезпечення, розробки програм мовою Python, наукові статті, публікації в інтернеті з даних питань.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до продукту, що розробляється

1. Можливість доступу через офіційний клієнт месенджера Telegram.
2. Зручний інтерфейс користувача.
3. Можливість подальшого розширення фунціоналу.

5.2. Вимоги до програмного забезпечення

1. Встановлене середовище Python версії 3.7 і більше.
2. Unix-подібна операційна система або Windows вище 7-ї версії.

5.3. Вимоги до апаратної частини

1. Не менше 3 Гб оперативної пам’яті.
2. Обсяг вільного простору на жорсткому диску не менше 1Гб.

6. ЕТАПИ РОЗРОБКИ

	Дата
Затвердження теми роботи	10.12.2020-17.12.2020
Вивчення та аналіз завдання	17.12.2020-07.02.2021
Розробка архітектури та загальної структури системи	07.02.2021-17.03.2021
Розробка структур окремих підсистем	17.03.2021-30.03.2021
Програмна реалізація системи	30.03.2021-04.05.2021
Оформлення пояснювальної записки	04.05.2021-22.05.2021
Виправлення помилок	22.05.2021-03.06.2021
Передзахист та проходження нормативного контролю	03.06.2021

Пояснювальна записка

до дипломного проєкту
на тему: «Медіа агрегатор з динамічним налаштуванням критерію відбору контенту»

Київ – 2021

ЗМІСТ

Перелік скорочень

IDE	Інтегроване середовище розробки (англ. Integrated development environmen)
API	прикладний програмний інтерфейс (англ. Application Programming Interface)
PRM	Управління партнерськими відносинами (англ. Partner Relationship Management)
RSS	Насправді проста синдикація (англ. Really Simple Syndication)
SQL	Мова структурованих запитів (англ. Structured Query Language)

ВСТУП

На даний момент соціальні мережі знаходяться на дуже активному етапі розвитку: стрімко росте як сумарна кількість користувачів, так і кількість широковідомих соцмереж. Це призводить до проблеми споживання користувачами непотрібної або неякісної інформації, адже люди мають обмежений час на користування соцмережами, та не можуть повною мірою контролювати потік контенту, який вони переглядають.

Окрім цього розкиданість інформації по різних платформах призводить до дискомфорту та трати часу при переході між платформами. За останні роки відомі випадки коли великі соціальні мережі в рамках співпраці інтегрували елементи інших соцмереж в свій продукт, таким чином реалізуючи агрегацію контенту. Отже, проблема розкиданості контенту по різних соціальних мережах також є дуже актуальною на сьогоднішній день.

Дана робота має на меті спробу часткового вирішення проблем надлишкової та розосередженої інформації в соціальних мережах шляхом розробки системи агрегації медіа контенту з можливістю його фільтрації за умовним критерієм якості. Варто згадати, що соцмережі використовуються в дуже різних сферах діяльності людини та з різною метою, як от маркетинг чи акціонерна діяльність. Тому оптимізація часу та способу отримання інформації з різних соцмереж може бути корисною як при особистому, так і при комерційному використанні.

Така розробка несе схожу ідеологію як і розглянуті в роботі аналоги, проте функції відбору контенту за критеріями не зустрічається в існуючих рішеннях. Тому в роботі робиться акцент на такому підході до оптимізації інформаційного потоку в соцмережах та проводиться його аналіз.

Для досягнення мети обирається шлях реалізації бота в месенджері Telegram, що дозволяє зосередитись на реалізації концепції агрегації та відбору контенту.

РОЗДІЛ 1
АКТУАЛЬНІСТЬ АГРЕГАЦІЇ МЕДІА КОНТЕНТУ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Значення оптимізації інформаційних потоків у сучасному світі

Сьогодні інформація відіграє надважливу роль практично в усіх сферах діяльності людини. Правильно відібрана і інтерпретована інформація може бути перетворена в знання, а знання, як відомо, це сила [2]. Проте, часто, отримувана інформація не приносить жодної користі, оскільки, за словами Даніеля Бурруса, «Information is power only if you can take action with it. Then, and only then, does it represent knowledge and, consequently, power» [3, с. 43].

Отримана інформація дозволяє набувати нових, або змінювати раніше отримані знання, які в подальшому можуть бути використані. Таким чином, вона допомагає нам приймати рішення [4]. Прийняття рішень неможливе без інформації, а відповідно кожна людина постійно знаходиться в залежності від неї. Це змушує нас шукати нові, більш корисні джерела інформації та прагнути покращити доступ до неї з вже доступних джерел. При цьому чим якіснішою буде вибірка отриманих даних, тим більш правильними будуть прийняті рішення з того чи іншого питання.

Особливо потужний вплив інформація має в таких сферах діяльності людини як економічна та соціальна [5][6]. З інформацією тісно пов’язані будь-які події, що є наявними у зазначених сферах. Зокрема, світова торгівля здійснюється, базуючись на відборі інформації та правильній її інтерпретації для прийняття вигідного рішення. Крім цього, в умовах конкуренції, надзвичайно важливу роль грає швидкість отримання інформації [1].

Без сумніву, одним з інструментів розповсюдження інформації, популярність яких на даний момент росте найбільш стрімко, є соціальні мережі. За об’єднаною статистикою цілої низки досліджень, станом на 2021 рік, кількість користувачів соцмереж в світі становить близько 3,96 млрд осіб. Це майже вдвічі більше ніж 2,07 млрд користувачів станом на 2015 рік і

становить близько половини всього населення світу (рис. 1.1) [7]. В Україні на початку 2021 року соцмережами користувалося близько 25,7 мільйонів людей, тобто приблизно 58,9% всього населення країни [8]. Беручи до уваги динаміку росту популярності соціальних мереж, можна очікувати, що їх роль в розповсюдженні інформації буде тільки збільшуватись.

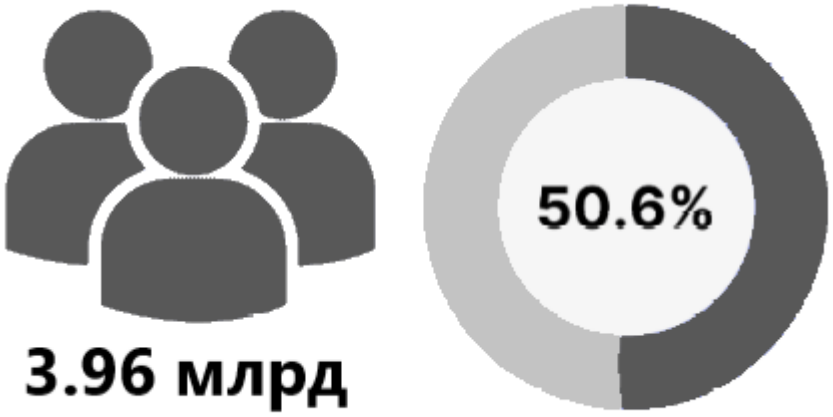


Рис. 1.1. Кількість користувачів соцмереж станом на січень 2021 року

Ключовою в даному контексті є статистика середнього часу користування соцмережами в день – він становить 2 години і 25 хвилини в світі (рис. 1.2) [10], а в Україні, за даними опитування 2020 року, 2 години і 19 хвилин [9]. Окрім цього, за статистикою глобальних опитувань, середня кількість створених людиною акаунтів в соцмережах становить 8 (рис. 1.3) [10], тобто більшість людей витрачають додатковий час на перехід між платформами. Така аналітика підтверджує існування проблеми, що набуває поширення з розвитком та популяризацією соціальних мереж – проблеми надлишкової інформації. Люди витрачають різну кількість часу на соцмережі кожен день, проте очевидно, що в кожної людини цей час обмежений, а відповідно обмежена кількість контенту, що його людина може спожити. Часто в зв’язку з цим, користувачі обирають для використання лише декілька найбільш необхідних соціальних мереж і відкидають решту.

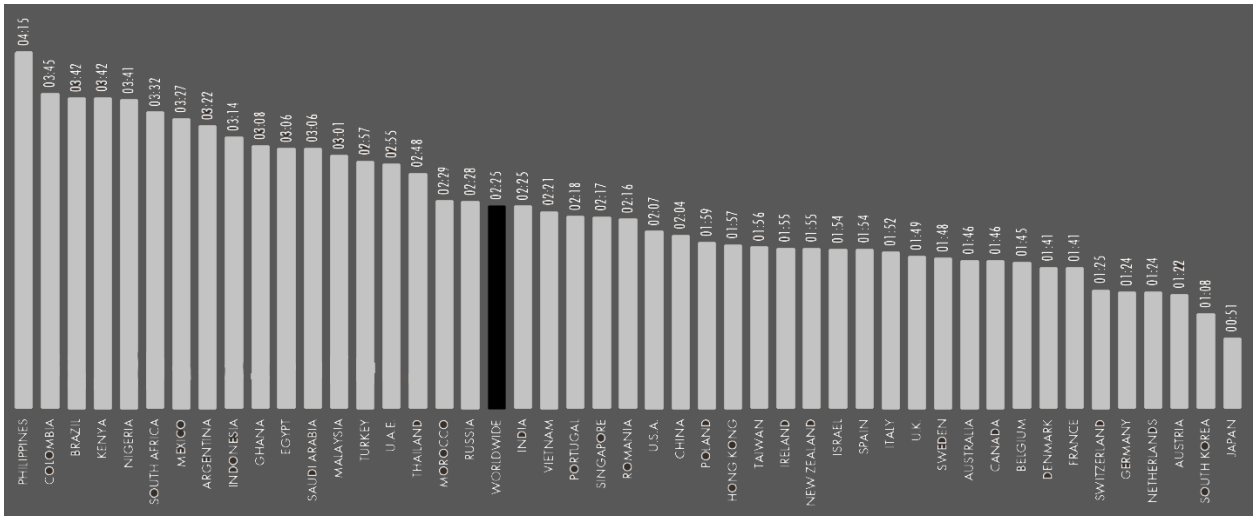


Рис. 1.2. Середній час використання соцмереж людиною щоденно станом на січень 2021 року

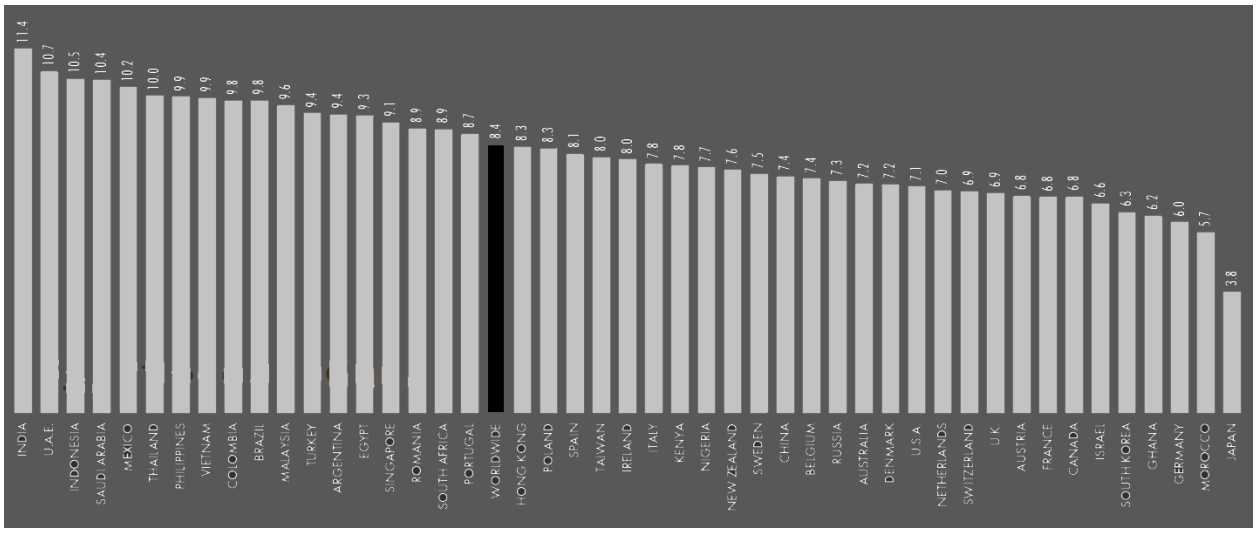


Рис. 1.3. Середня кількість акаунтів в соцмережах на одну людину станом на січень 2021 року

На даний момент найбільш популярною соціальною платформою є Facebook, в якому зареєстровано 2,7 млрд користувачів, що становить більше половини всіх користувачів Інтернету [10]. Враховуючи те, що перший мільйон активних користувачів було набрано мережею MySpace у 2004, варто сказати, що швидкість розвитку найпопулярніших соцмереж тримається на високому рівні вже близько 15 років і можна передбачити, що вона не буде спадати найближчим часом (рис. 1.4) [11]. Вже на даний момент соцмережі

набули надзвичайного поширення, а відповідно впливу, і, судячи з тенденцій, це буде тільки посилюватись.

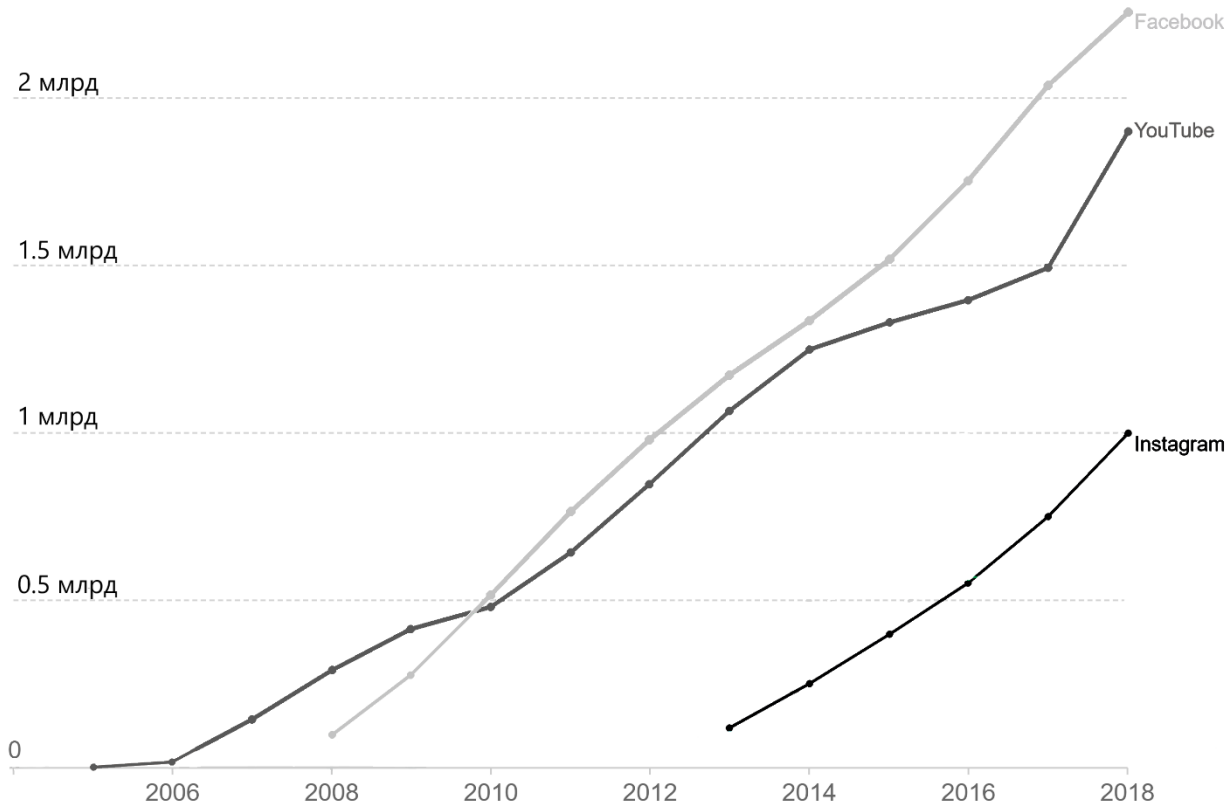


Рис. 1.4. Графік зростання популярності деяких популярних соціальних мереж з 2006 по 2018 рік

В таких умовах багато платформ зазнали змін навіть в своїх базових особливостях з метою забезпечення необхідних можливостей для кожного користувача. Наприклад, Twitter не дозволяв користувачам публікувати фото та відео контент – це було однією з його особливостей, але зрештою така функція була додана і, здебільшого, позитивно зустріта спільнотою. В результаті, на даний момент більше половини створюваних публікацій в Twitter включають фото чи відео. Таким чином, формат записів в більшості соцмереж стає майже однаковим, що тільки додає сенсу ідеї агрегації всього контенту в один потік.

Окрім Facebook, на даний момент гігантами даної індустрії є також YouTube, Instagram (більше 1 млрд користувачів кожна) та інші (рис. 1.5) [11].

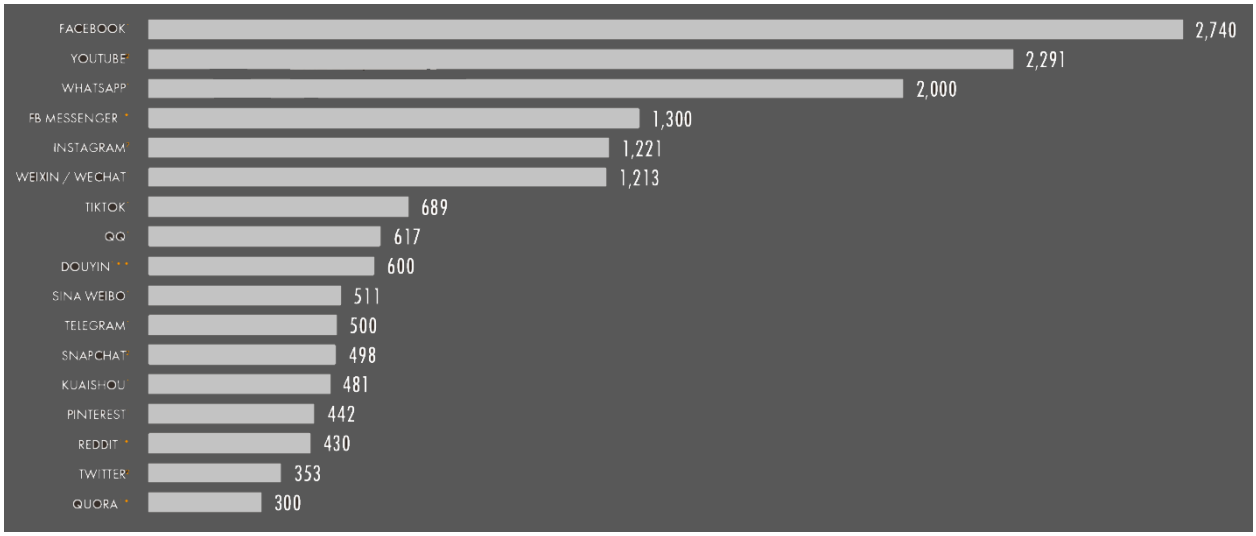


Рис. 1.5. Рейтинг популярності соціальних мереж станом на січень 2021 року

Завдяки такому великому впливу можуть також зростати збитки компаній, які залежать від інформації, отримуваної, у тому числі, з соціальних мереж, якщо якість отриманих даних є недостатньо високою.

Для інформації виділяють наступні критерії якості [12]:

- Достовірність
- Повнота
- Своєчасність
- Актуальність
- Конфіденційність

Ці показники дозволяють характеризувати дані, а з появою контексту використання вони можуть визначати також і її потенціальну користь. В випадку з соцмережами достовірність і повнота може визначатись лише вибором правильних джерел контенту. При цьому ніхто не залишається застрахованим від дезінформації чи недостачі інформації в тому чи іншому випадку. Конфіденційність – це відомий мінус поширення інформації соцмережами [13], здебільшого по причині невідповідності основному принципу відкритості соцмереж, оскільки особисті дані користувачів часто стають ціллю, або і здобиччю зловмисників.

Проте своєчасність і актуальність – це ті критерії, які можна покращувати не тільки способом вибору відповідних джерел інформації, а і за

допомогою технічних засобів. Проблемою тут є лише те, що важко досягнути балансу між цими двома показниками, оскільки кожна людина має своєрідний прохідний поріг, який визначає скільки інформації в одиницю часу вона може споживати. Він буває різним, але часто його не вистачає для отримання повного обсягу інформації, оскільки відбір актуального контенту займає свій час, а перегляд контенту без відбору займає занадто багато часу [17]. Тому існує необхідність в системі, яка зможе відфільтрувати лише актуальний контент, та надати його вчасно. Збереження часу в ній буде забезпечуватись за рахунок відсутності потреби в використанні переліку різних соцмереж та в можливості переглядати весь контент в одному місці.

1.2 Огляд підходів до агрегації контенту соціальних мереж

Варто виділити 4 основних види агрегаторів контенту:

- Веб-сайти агрегатори, що дозволяють сформувати персоналізований потік контенту з різних соцмереж. Вони орієнтовані на використання окремими користувачами і надають широкі можливості в плані детальних налаштувань системи. Основним недоліком даного виду агрегаторів є необхідність використання браузера для перегляду контенту на веб-сторінці. Це може бути незручним для частини потенціальних користувачів, особливо в випадку доступу з мобільного пристрою.
- Платформи веб агрегації, що використовуються для ведення блогів та сайтів і дозволяють керувати контентом, який потрапляє на веб сторінку. Такий інструмент не є персональним, він автоматизує відбір та відображення публікацій на сторінці, проте не забезпечує засобами особистих налаштувань кожного споживача даного контенту. Тим не менше, можливість створення особистого потоку контенту для показу його іншим користувачам користується попитом і загалом виконує всі функції агрегатора контенту. Крім цього, такі сервіси мають високий рівень входження і не підійдуть для повсякденного користування середньостатистичним користувачем.
- Програми агрегатори медіа контенту, зокрема мобільні додатки,

що дозволяють сформувати потік контенту взятого з соціальних мереж для окремого користувача. Платформи такого типу користуються популярністю в користувачів за рахунок роботи на мобільних пристроях, просте дуже часто вони мають дуже обмежені можливості в налаштуванні інтерфейсу та фільтрації публікацій з соцмереж. Також поширеною проблемою в таких системах є велика кількість рекламного вмісту, що відштовхує певну частину аудиторії.

- Інтегровані в соціальні мережі чи месенджери системи, що дозволяють переадресовувати контент з однієї соцмережі в іншу, або збирати контент з кількох джерел та відображати їх в одному місці. Зокрема, в мобільному додатку Instagram є можливість підключити переадресацію повідомлень з соцмережі Facebook. Проте основну частину такого роду систем становлять боти – алгоритми, що в соцмережі зареєстровані як звичайні користувачі, мають подібний образ і, зазвичай, такі самі права на доступ до інформації. В деяких випадках такі підсистеми можуть мати доступ і до публікації інформації, проте така можливість зазвичай дуже обмежена для запобігання різного роду зловмисних дій зі сторони ботів.

1.3 Огляд існуючих систем агрегації на веб-сайтах

Інструменти агрегації на веб-сайтах дозволяють налаштовувати потік контенту з різних джерел, таких як соцмережі, RSS потоки [14] та інше, в залежності від конкретної системи. Зазвичай такі платформи мають широкий функціонал для кастомізації зовнішнього вигляду публікацій на сайті а також для її відбору по різних критеріях. Також, вони часто бувають оснащені інструментами для створення власного контенту. Крім цього, такі сервіси, зазвичай, намагаються виділитись різноманітними додатковими функціями, як наприклад можливість відображення рейтингу публікацій на сайті, або збір аналітики і зворотній зв’язок з користувачами сайту [15].

Таким чином, ці системи є потужними засобами для автоматичного наповнення веб сайтів контентом з дуже широкими можливостями по його

відбору. Вони часто використовуються інтернет магазинами та іншими комерційними організаціями для агрегації відгуків про свої товари, послуги тощо.

Платформ для агрегації контенту з соціальних мереж на веб сайтах існує велика кількість. Частина з них більше підходить для використання в великому масштабі, наприклад, для ведення маркетингових кампаній, а частина більше пристосована для персонального використання.

1.3.1 Miappi

Miappi – це потужний засіб агрегації контенту з таких соцмереж як Instagram, TikTok, YouTube, Twitter та інші. Дана платформа дозволяє використовувати різноманітні критерії пошуку медіа, такі як хештеги, відмічання особи, теги зображень для подальшого відображення на сайті агрегаторі. Окрім цього, платформа пропонує великі кураторські можливості, тобто дозволяє відбирати контент не тільки автоматично, а і ручним способом, додавати власні коментарі, а також створювати записи самостійно. Додатково система пропонує засоби для зручної комунікації з авторами контенту, для швидкого узгодження будь-яких питань в разі такої потреби [15].

Miappi є дуже гнучким в питанні зовнішнього вигляду розміщуваних публікацій та дозволяє налаштовувати відображення для відповідності дизайну сайту (рис. 1.6). Платформа надає можливість створення багатьох потоків контенту для відображення на різних сторінках. Окрім цього, отримуваний потік може бути використаний і поза межами веб сайту, наприклад вивантажений в PRM систему тощо. Також сильною стороною даної платформи є система аналітики, яка дозволяє слідкувати за статистикою часу перебування користувачів на сторінці, популярності кожної з публікацій, виявляти тенденції і попит [16].

Miappi використовується здебільшого великими брендами та компаніями для просування свого продукту та ведення маркетингової кампанії в соціальних мережах. Вона не використовується для некомерційних потреб в силу своєї надлишкової громіздкості а також високої вартості.

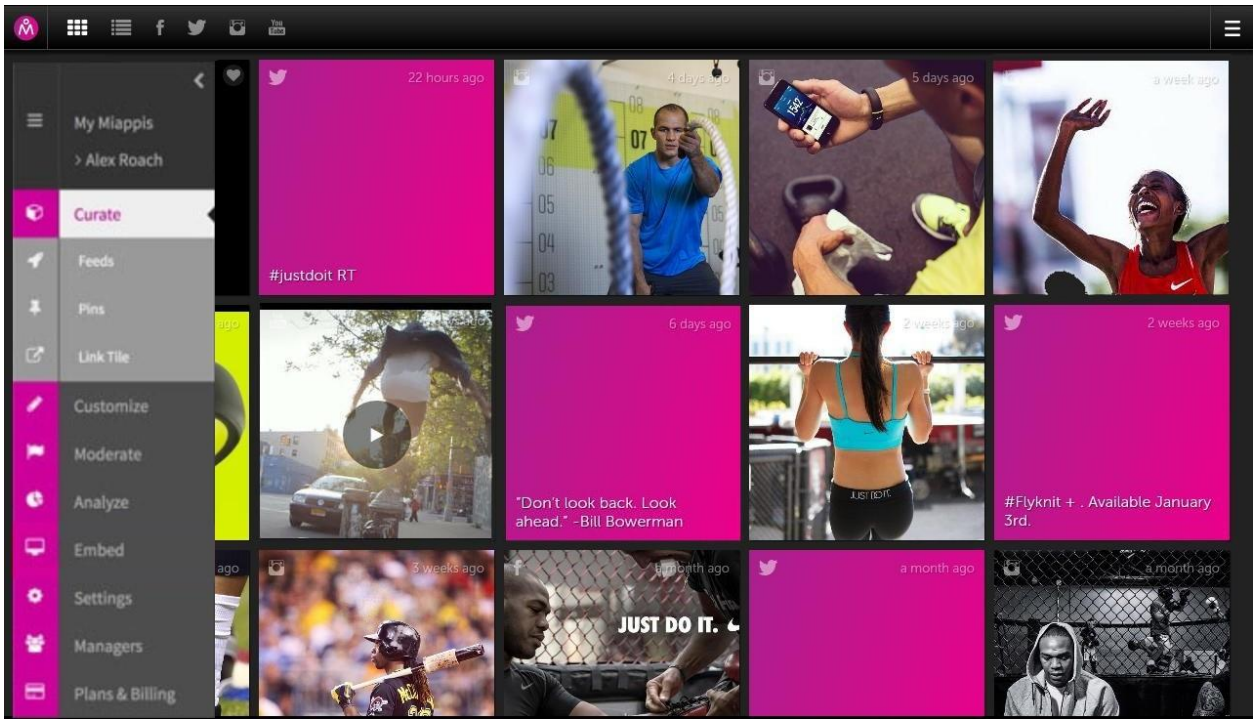


Рис. 1.6. Інтерфейс сайту медіа агрегатора Miappi

Перевагою даної платформи серед інших є дуже потужні засоби налаштування вмісту медіа потоку, адміністрування даного потоку та розміщення його на веб-сторінках. Окрім цього, дана система виділяється потужними засобами аналітики та інтеграції з іншими платформами. Проте серйозним недоліком є висока ціна підписки та високий поріг входження, що буде непрохідним бар'єром для середньостатистичного користувача.

1.3.2 Curator

Curator – це популярний в широкому колі користувачів агрегатор контенту з таких соцмереж як Facebook, Instagram, Twitter, YouTube ті інші. Окрім цього, він може «підтягувати» публікації і з RSS потоків, якими часто оснащують вебсайти блогів тощо. Дана платформа характеризується простим інтерфейсом для керування інструментами агрегації, що забезпечує низький поріг входження. Тим не менше, її функціональність включає основні функції системи агрегації, такі як можливість гнучкого налаштування відбору контенту, можливість його попереднього перегляду та редагування. Дана

система характеризується гнучкістю в відображенні і забезпечує можливість вбудовувати генерований потік контенту як HTML, JavaScript, і CSS у вебсайт, що дозволяє налаштовувати його зовнішній вигляд для відповідності сайту (рис. 1.7, 1.8) [15].

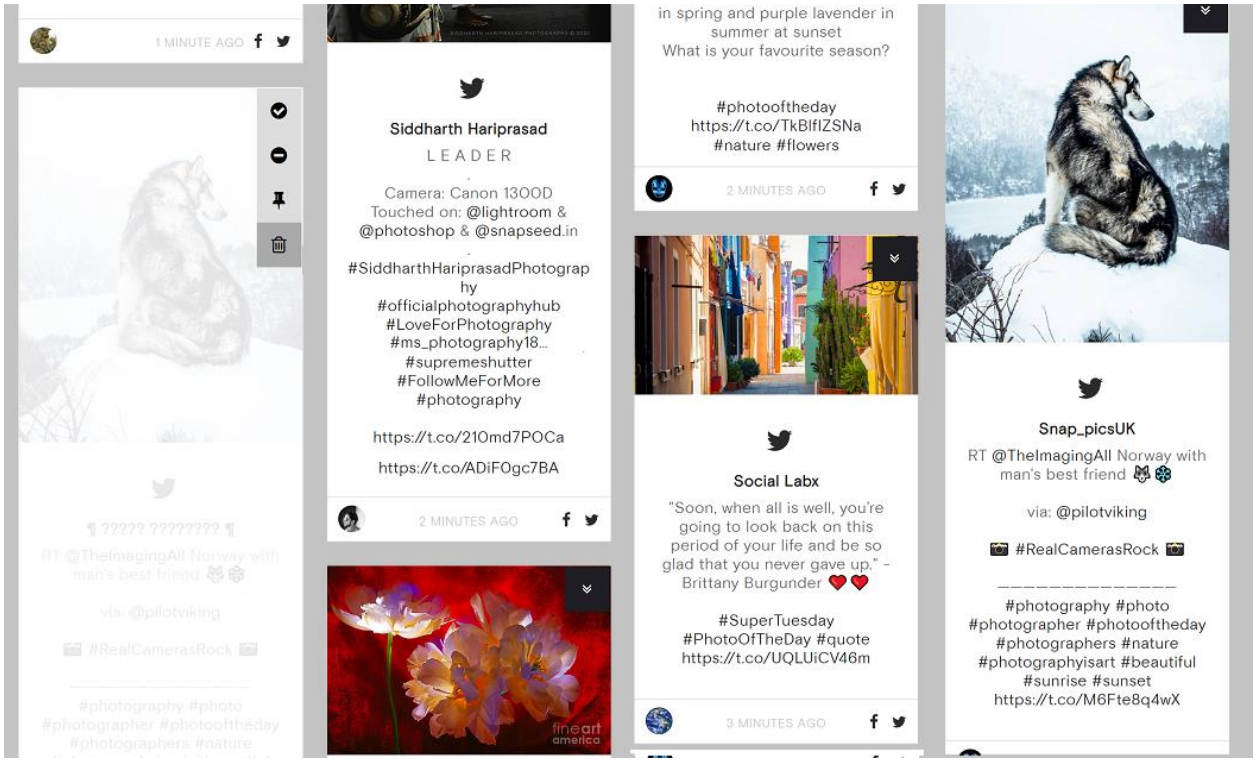


Рис. 1.7. Приклад роботи медіа агрегатора Curator

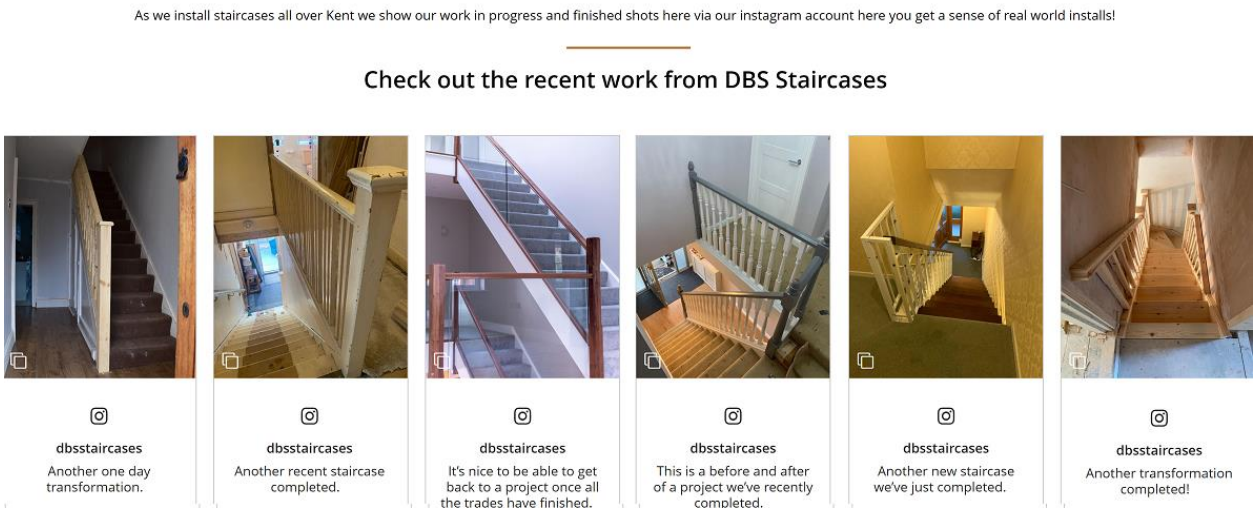


Рис. 1.8. Приклад модифікації зовнішнього вигляду медіа агрегатора Curator

Очевидною перевагою даної платформи є відносно невисокий поріг

входження, що для такого потужного інструменту є досить унікальною рисою.

Окрім цього, розробники даного продукту пропонують одразу декілька тарифних планів для потенційних користувачів, серед яких є і можливість безкоштовного використання системи. Таким чином, Curator може забезпечити користувача необхідними інструментами для агрегації контенту, є гнучким для відображення на будь-якій веб сторінці але, при цьому, він придатний для використання не тільки в комерційних цілях, а і в особистих.

Рис. 1.9. Таблиця тарифних планів платформи Curator

1.3.3 Hootsuite

Hootsuite – платформа персональної агрегації контенту, що, на відміну від багатьох аналогів, спрямована на створення потоку контенту тільки для одного користувача, а не для публікації його не веб сторінці. Тим не менше, сама система працює як веб сайт у браузері, що робить етап її первинного налаштування швидшим, оскільки система не потребує попереднього встановлення [15].

Платформа дозволяє підключити до неї аккаунти користувача в таких соцмережах як Facebook, Twitter, LinkedIn тощо і здійснювати як перегляд так і управління потоками контенту з них в одному місці. Особливістю даної системи є можливість не тільки отримання інформації з соціальних мереж, а і її розміщення там же. Платформа дозволяє створювати публікації, вивантажувати медіа контент, а також обмінюватись повідомленнями одночасно у декількох соцмережах (рис. 1.10).

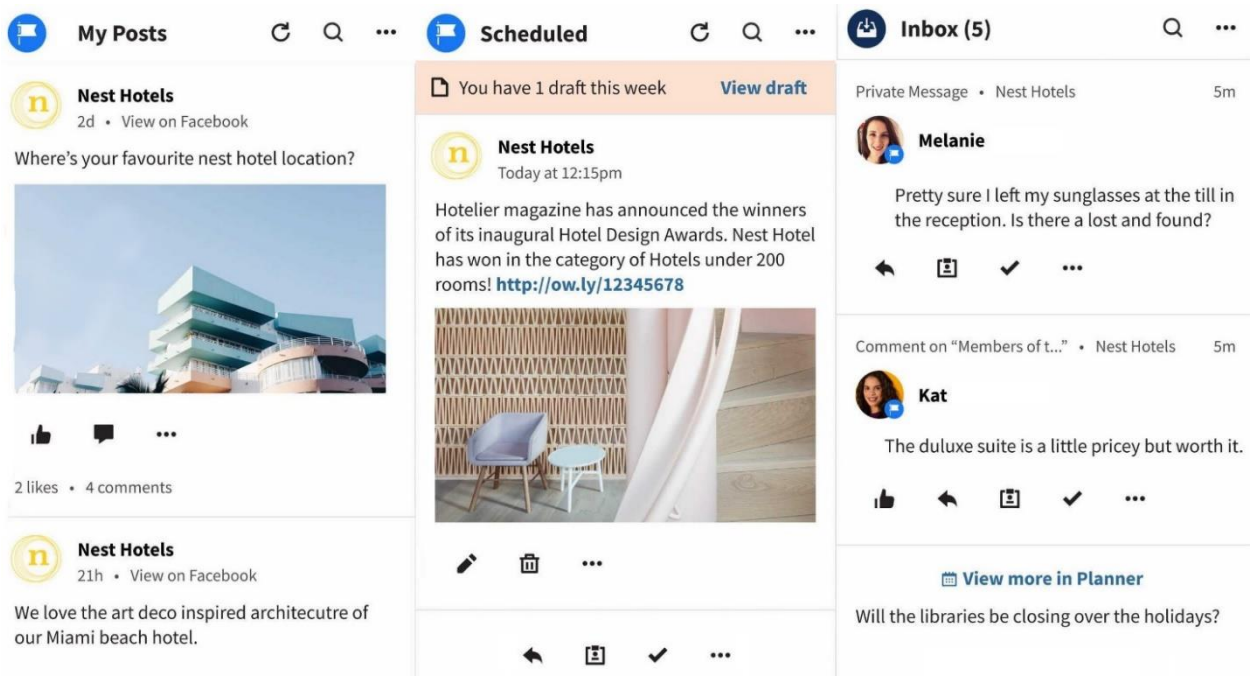


Рис. 1.10. Приклад роботи платформи Hootsuite

Дана платформа користується попитом як у комерційних компаній, так і в найбільш активних користувачів соцмереж, що створюють та поширюють контент. Вона не часто обирається для персонального використання по причині високої ціни та відсутності безкоштовного тарифу користування.

Hootsuite характеризується дуже широкою функціональністю для управління аккаунтами в соцмережах та налаштуванням отримуваного потоку медіа контенту. Оскільки дана система не потребує встановлення та не включає функціональності по курації сайтів, можна стверджувати, що поріг входження в неї відносно низький і вона може використовуватись в персноальних цілях. Проте недоліком цієї системи є цінова політика компанії-

розробника, а також відсутність можливості публікаці контенту для інших користувачів та на веб сторінках, що обмежує сферу її використання.

1.3.4 Social Meadia Wall

Дана платформа є однією з найбільш популярних для створення сторінок, наповнених медіа контентом з соцмереж. Вона має всі необхідні функції медіа агрегатора, оскільки дозволяє переглядати контент перед публікацією, фільтрувати його, а також надає можливість зміни зовнішнього вигляду публікацій. Її особливістю є простота в налаштуванні та використанні та можливість безкоштовного користування. Проте, коло під'єднаних соцмереж тут значно менше ніж у аналогів – платформа дозволяє підключити аккаунти Facebook, Instagram, Twitter, Google+ (рис. 1.11). Крім цього, система не має функції автоматично відбору контенту за встановленими критеріями, що робить її актуальною лише для використання з невеликим потоком контенту [15].

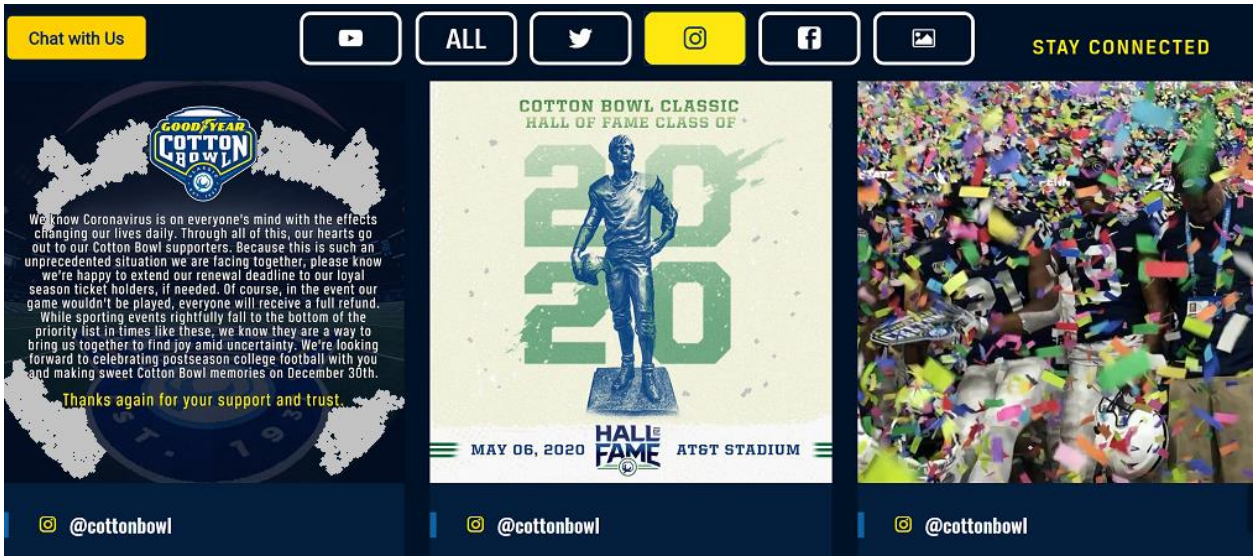


Рис. 1.11. Приклад роботи платформи Social Media Wall

Особливістю даної платформи є простота її використання, адже система побудована так, щоб користувачу не потрібно було робити великої кількості зайвих налаштувань. Проте, за рахунок цього страждає функціональність даного медіа агрегатора. Платформа не має можливостей

фільтрації контенту, публікації його для інших користувачів, а також кількість підключених джерел інформації дуже обмежена. Зокрема, дана система не надає можливості отримання контенту з таких соцмереж гігантів як YouTube чи Instagram. Тим не менше, дана система має свою базу користувачів, яким додаткові джерела та складні налаштування потоку контенту не потрібні, проте дуже важлива цінова політика та простота користування.

1.4 Огляд мобільних додатків агрегаторів

Мобільні додатки в цій сфері зараз знаходяться не на самому активному етапі розвитку. На даний момент існує незмірно велика кількість аналогів мобільних агрегаторів медіа контенту, і, хоча більшість таких програм можуть досить сильно відрізнятись за своєю функціональністю, серед них складно виділити представників, що суттєво виділяються серед інших. Велика частина з них тісно зв’язані з веб-сайтами агрегаторами та платформами для агрегації контенту на веб-сайтах і є лише доповненням для них. Часто такі додатки найкраще пристосовані для виконання якоїсь однієї конкретної функції, наприклад ефективно сповіщати про важливі події, що відбуваються в обраному полі соціальних мереж, або об’єднувати лише месенджери для зручного обміну повідомленнями.

1.5 Огляд ботів медіа агрегаторів

На даний момент боти набувають великої популярності серед розробників в силу відсутності потреби в створенні інтерфейсу та, відповідно можливості зосередитись на функціональній частині, а також серед користувачів. завдяки зручності, зумовленої інтегрованістю бота в систему, якою користується користувач. Завдяки зосередженню багатьох функцій в одному месенджері користувач економить час як при встановленні тієї чи іншої підсистеми, так і при її постійному використанні. Боти існують в різних

соцмережах та месенджерах, проте не завжди їх розробка є доступною для кожного і не завжди для цього існують необхідні інструменти.

Зокрема, серед ботів медіа агрегаторів, користуючись спеціальним довідником Telegram Channels (рис. 1.12) [18], можна знайти декілька функціонуючих систем, які користуються популярністю. Решта з них працюють виключно на RSS потоках, що сьогодні вже не так актуально як раніше, а інша частина взагалі перестала функціонувати по причині низької популярності серед користувачів (наприклад @EventsAggregatorBot @Feedler та @SingleFeed).

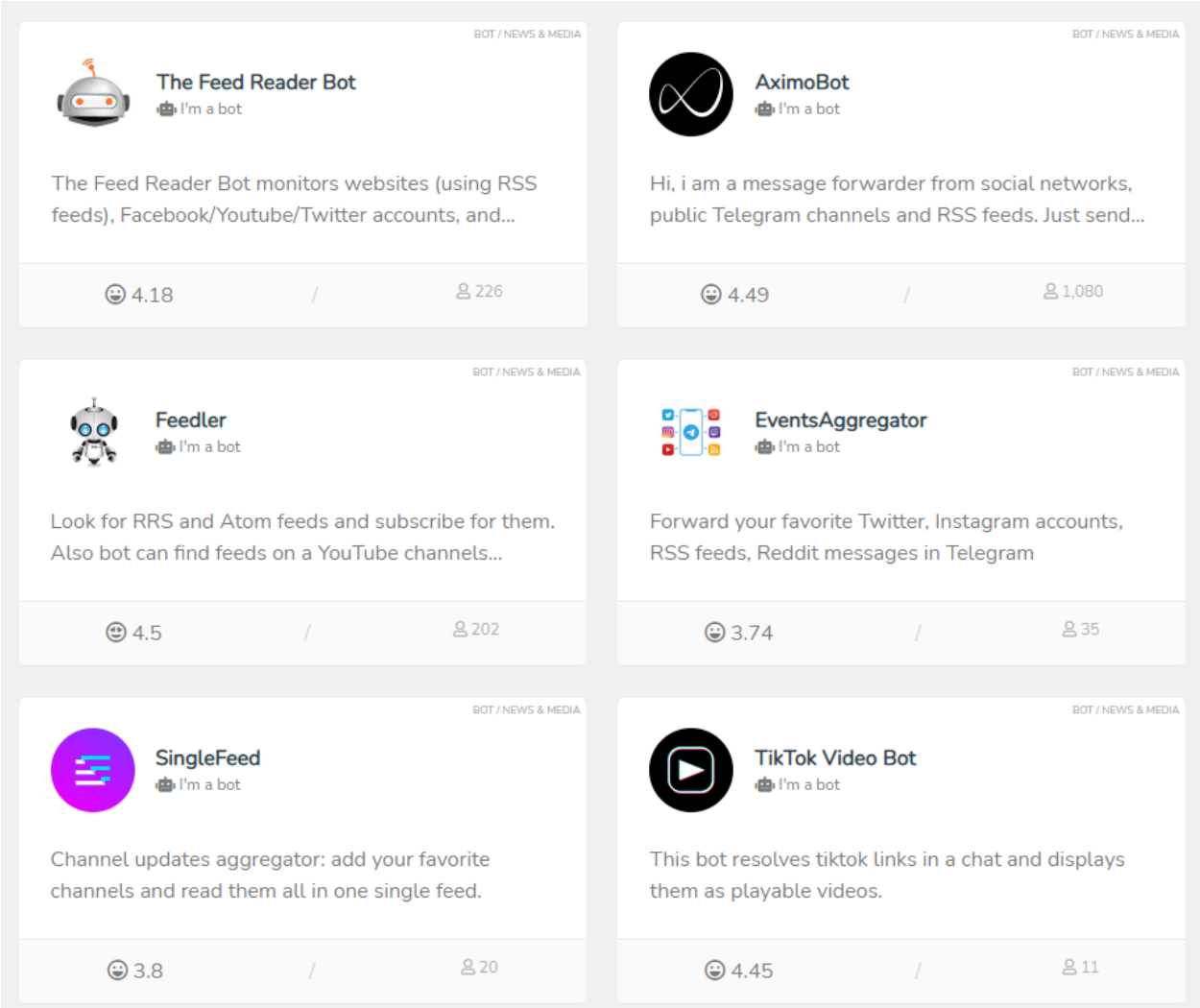


Рис. 1.12. Список опублікованих Telegram ботів, що працюють з медіа

1.5.1 AximoBot

AximoBot (@AximoBot) – телеграм бот з найбільшою користувацькою базою в цій тематиці [18]. Він дозволяє відслідковувати контент профілів в таких соцмережах як Instagram, YouTube, Facebook, Twitter, а також потоки RSS та публічні канали в Telegram. Для підписки на будь-яке джерело даних, в більшості випадків, необхідно надати боту посилання на нього, а для її скасування – потрібно вибрати джерело зі списку (рис. 1.13). Загалом бот дозволяє додати 25 джерел за умови його безкоштовного використання.

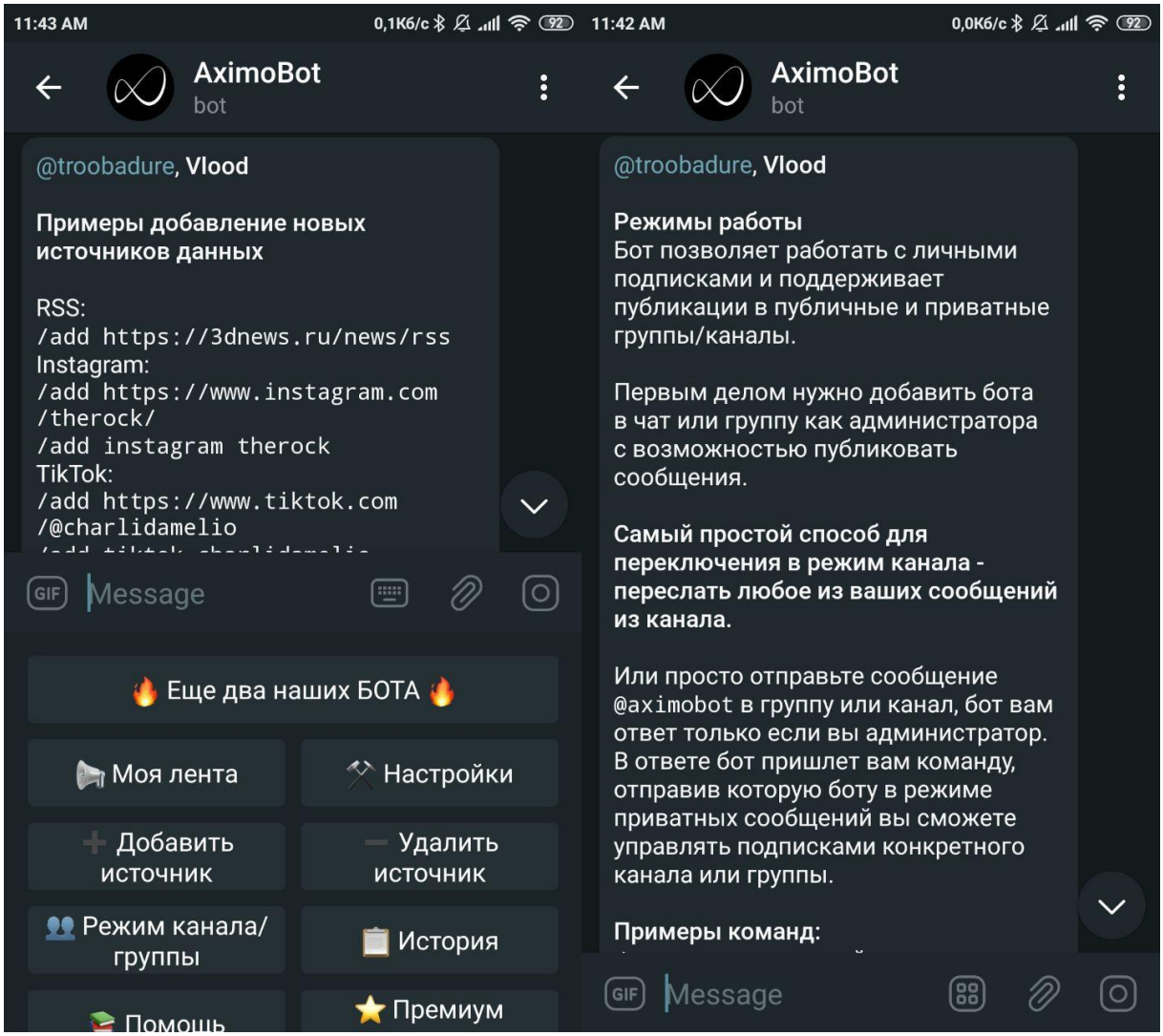


Рис. 1.13. Приклад роботи Telegram бота AximoBot

Ще однією з функцій бота є автоматичне наповнення контентом каналу, або чату в Telegram. Бот, переведений в даний режим отримує можливість пересилання потоку публікацій в назначені канали або чати і при цьому дає

можливість попереднього перегляду контенту. Таким чином можна здійснювати фільтрацію публікованої інформації.

Додатково бот оснащений спеціальною клавіатурою, що дозволяє здійснювати управління ботом не за допомогою команд, а способом натискання відповідних клавіш.

Найбільш суттєвою перевагою даного бота є можливість публікації сформованого потоку контенту в телеграм канал. Це дозволяє проводити агрегацію контенту не лише для однієї людини, а для цілої групи користувачів. Проте дана розробка має серйозне обмеження в кількості можливих підписок на джерела інформації – користувачу дозволяється обрати щонайбільше 25 таких джерел. Досить важливу роль відіграє також наявність клавіатури замість звичайних команд, оскільки зручний інтерфейс бота в месенджері – це досить рідкісне благо, яке серйозно впливає на досвід користування системою.

1.5.2 The Feed Reader Bot

The Feed Reader Bot (@TheFeedReaderBot) – телеграм бот, що слідкує за появою публікацій в таких соцмережах як Facebook, YouTube, Twitter. Даний список досить короткий оскільки робота програми базується на моніторингу RSS потоків, що не підтримується великою частиною сучасних соцмереж. The Feed Reader Bot дозволяє як формувати потік для персонального користування, так і перенаправляти його в канали та чати (рис. 1.14).

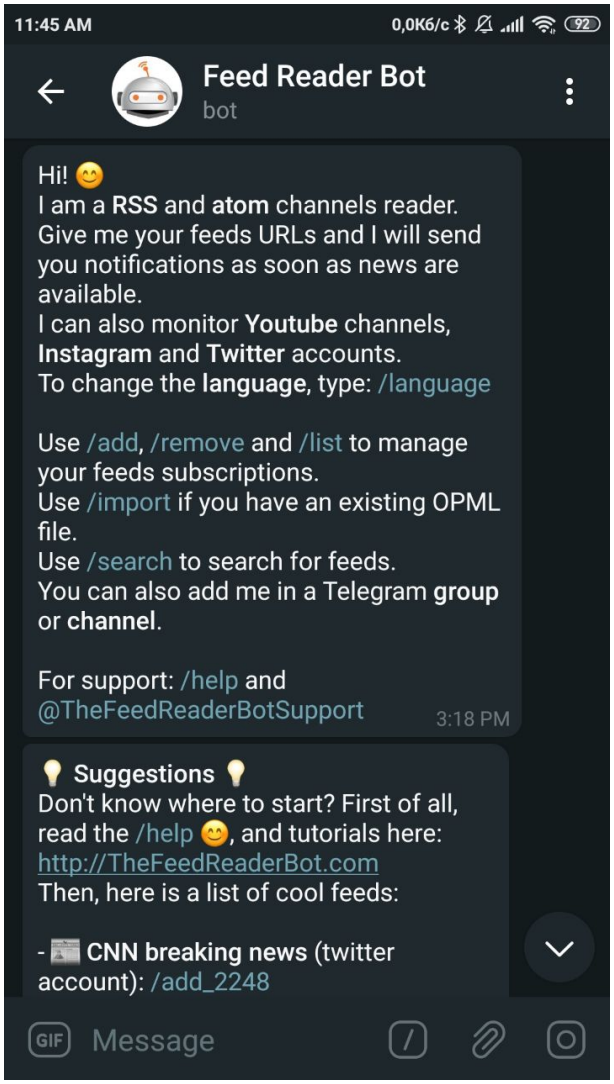


Рис. 1.14. Приклад роботи Telegram бота The Feed Reader Bot

Додатковою функцією даного бота є пошук доступних RSS потоків шляхом введення команди та пошукового запиту.

The Feed Reader Bot має функцію публікації контенту в канали та чати Telegram, що є великою його перевагою. При цьому вміст, що буде публікуватись для групи людей, може бути попередньо переглянутий користувачем та відібраний вручну. Проте бот не має можливості автоматичної фільтрації вмісту навіть для формування персонального потоку інформації, що є типовим недоліком таких систем. Окрім цього даний бот має обмеження в використанні джерел інформації, оскільки працює суто з RSS потоками, що робить його непрактичним в сьогоднішніх реаліях, коли багато соцмереж дану технологію не підтримують.

1.5.3 Feed 2 Telegram

Feed 2 Telegram (@Feed2Telegram) – бот, що дозволяє переглядати публікації з соцмереж Instagram, Twitter, Facebook, YouTube, а також з RSS потоків (рис. 1.15). Бот має функції додавання до 4 джерел контенту за безкоштовного його використання. Додатково бот дозволяє переглянути список популярних джерел на даний момент та надає можливість підписатись на них.

Функціонал даної системи описується коротко, проте, імовірно, це і робить його відносно популярним серед користувачів. Адже бот Feed 2 Telegram не має великого набору команд, є дуже простим в користуванні і не потребує багато часу для налаштування, проте здатен виконувати ключові функції медіа агрегатора.

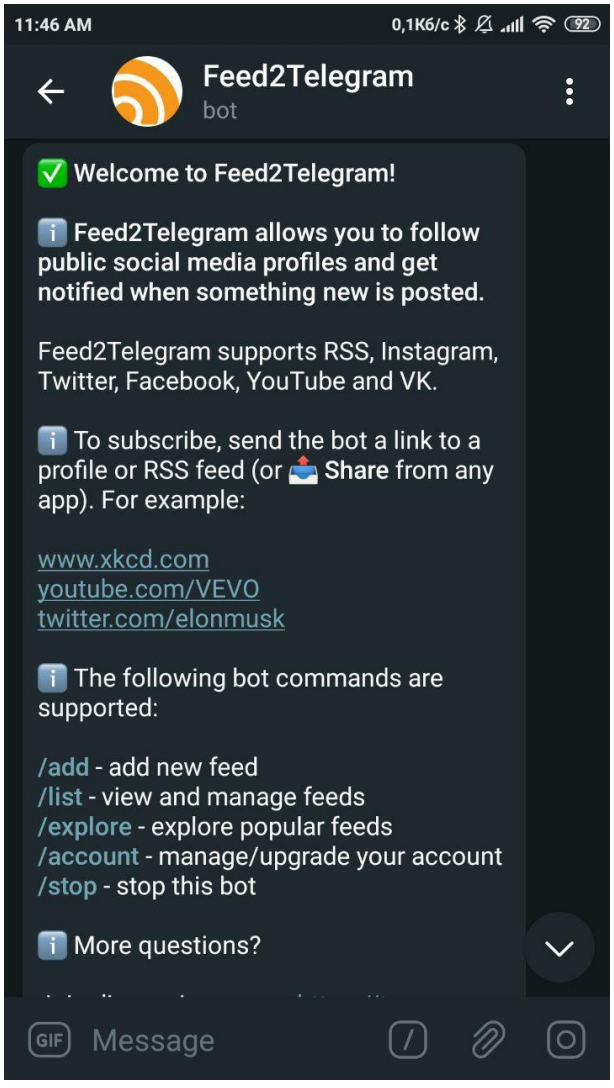


Рис. 1.15. Приклад роботи Telegram бота Feed 2 Telegram

Варто відмітити, що система підтримує використання в якості джерел інформації всього декілька найбільш популярних на даний момент соцмереж, а також потоки RSS, що охоплюють велику частину всього публікованого в мережі Інтернет вмісту. Проте, очевидним недоліком є серйозне обмеження максимальної кількості доданих джерел – їх користувач може додати всього 4. Окрім цього, бот не має ні функції фільтрації та відбору контенту, ні можливості його переадресації в інші Telegram канали чи чати. Це робить його дуже вузькоспеціалізованим, що характерно для даного типу медіа агрегаторів.

1.6 Порівняння розглянутих систем агрегації контенту

На основі проведеного вище огляду та аналізу кожного з підходів до агрегації медіа контенту варто виділити деякі тенденції розподілу користувацької бази між різними типами систем агрегації.

В середньому високим порогом входження та масштабами функціональності можна охарактеризувати платформи агрегації контенту на веб-сайтах. Вони, здебільшого, користуються популярністю в комерційних організацій в силу можливості використання їх в маркетингових компаніях та просто за рахунок потужних систем відбору вмісту та інструментів керування потоками контенту. Для персонального використання дані системи, здебільшого, не пристосовані, оскільки характеризуються складними процедурами налаштування та адміністрації, а також часто вони не мають можливості безкоштовного використання.

Веб-сайти та мобільні додатки, що слугують для агрегації та відображення вмісту персонально для одного користувача, користуються популярністю в великої кількості рядових користувачів. Такі системи характеризуються низьким порогом входження, а також, здебільшого, вони є вузькоспеціалізовані і спрямовані на детальну і глибоку реалізацію однієї обраної функції, замість того щоб одночасно реалізовувати широкий спектр

функціоналу. Популярними такі системи залишаються за рахунок великої їх кількості на ринку, що дозволяє покрити весь вищезгаданий спектр функціональності різними системами.

Інтегровані системи агрегації, зокрема боти, на даний момент лише набувають популярності, оскільки дана область є відносно молодою і в ній ще немає ні великої кількості рішень, ні великої кількості користувачів. Проте, можна стверджувати, що такого роду системи підходять найкраще для персонального користування і не розраховані на те, щоб вміщати в собі великий об’єм функціональності. Тим не менше, по зручності використання для простих сценаріїв, вони, імовірно, лідирують, хоча і уступають іншим типам систем агрегації в можливостях інтерфейсу та при необхідності реалізації складних сценаріїв використання.

В результаті аналізу кожної з оглянутих платформ медіа агрегації для веб-сайтів можна сформувати ряд критеріїв для якісного їх порівняння. Поріг входження в кожної з систем був різним і відгравав ключову роль при характеристиці користувацької бази системи. І хоча це поняття є дуже відносним і, наприклад, в порівнянні з Telegram ботами, будь-яка платформа агрегації контенту для веб-сайтів буде мати високий поріг входження, але розглянуті системи суттєво відрізняються за цим критерієм. Можливість безкоштовного використання, є, без сумніву, дуже важливою для тієї частини користувачів, що прагне використовувати платформу для некомерційних цілей. Функції налаштування відбору контенту та публікації потоку на веб-сайті є дуже об’ємними частинами функціональності, при цьому надаючи можливості, які в більшості розглянутих платформ представляються як базові. За даними критеріями можна сформувати таблицю порівняння систем такого роду (табл. 1.1).

Таблиця 1.1. Порівняння платформ медіа агрегації на веб сайтах.

№	Назва платформи медіа агрегації	Функція налаштування відбору медіа контенту	Функція публікації сформованого потоку контенту	Поріг входження в користування платформою	Можливість безкоштовного використання
1	Miappi	Присутня	Присутня	Високий	Відсутня
2	Curator	Присутня	Присутня	Середній	П\рисутня
3	Hootsuite	Присутня	Відсутня	Низький	Відсутня
4	Social Media Wall	Відсутня	Відсутня	Низький	Присутня

Говорячи про Telegram ботів, актуальними залишаються такі критерії порівняння як функція налаштування відбору контенту, та функція публікації сформованого потоку. Дані показники дозволяють визначити ключову різницю між системами. Проте, такі критерії як поріг входження та ціна не є доцільним використовувати для порівняння ботів, оскільки, очевидно, всі вони мають гранично низький поріг входження за рахунок простоти функціоналу та інтерфейсу, а також всі розглянуті системи можуть бути використані безкоштовно. Тим не менше, сильно обмежуючим користувача в можливостях фактором є максимальна кількість підписок на джерела інформації. Будь-яка розробка в цій області не повинна мати таких серйозних границь, оскільки вони перешкоджають основному завданню медіа агрегатора – збирати контент з багатьох різних місць в один потік. Також, в умовах переважання простої функціональності у Telegram ботів, набуває актуальності такий критерій як додаткові можливості. Адже в таких умовах дуже тривіальні речі, як от пошук чи покращений інтерфейс, мають посилений вплив на досвід використання системи. За даними показниками можна створити таблицю порівняння

розглянутих Telegram ботів (табл. 1.2). Крім цього, варто додати у список свій медіа агрегатор, який повинен покривати функціональність, якої не вистачає в даному полі, у вигляді системи з можливістю фільтрації контенту, публікації його для ширшої аудиторії, зі зручним інтерфейсом та без критичних обмежень.

Таблиця 1.2. Порівняння Telegram ботів медіа агрегаторів

№	Назва бота медіа агрегатора	Функція налаштування відбору медіа контенту	Функція публікації сформованого потоку контенту	Обмеження в кількості доданих джерел	Додаткові можливості
1	AximoBot	Відсутня	Присутня	25	Інтерфейс
2	The Feed Reader	Відсутня	Присутня	Немає	Пошук
3	Feed 2 Telegram	Відсутня	Відсутня	4	Пошук
4	Свій медіа агрегатор	Присутня	Присутня	Немає	Інтерфейс

Висновки до розділу 1

В результаті роботи над даним розділом для обгрунтування актуальності даної розробки проведено аналіз проблеми оптимізації інформаційних потоків в соцмережах. Також, з метою відслідкувати тенденції потреб користувачів та для того, щоб знайти функціональну область, не покриту існуючими рішеннями, виділено основні підходи до її рішення, оглянуто конкретних представників, що реалізують ці підходи, та вибрано критерії для їх якісного порівняння. На основі цього сформульовано такі висновки:

- Інформація грає дуже важливу роль в нашому соціальному та економічному житті. Наша продуктивність часто залежить від швидкості отримання інформації та її якості.
- Соцмережі сьогодні стрімко розвиваються я засіб поширення інформації і виникає проблема великої кількості джерел медіа вмісту, а також великої кількості низькоякісного контенту, що потрапляє до користувача випадково.
- Медіа вміст в різних соціальних мережах зводиться до одного вигляду і може бути відображений в одному потоці. Це дозволить зберегти час, витрачений на перехід між соцмережами.
- В багатьох соцмережах існують свої методи оцінки публікацій, що може бути використаним для його фільтрації. Це дозволить зменшити кількість низькоякісного контенту, що відображається для користувача.
- Існує декілька підходів до агрегації вмісту соціальних мереж, але більшість з них мають велику кількість представників, що реалізовують різні функціональні можливості. Проте в області інтегрованих систем, зокрема ботів Telegram, не всі потреби користувачів можуть бути задоволені існуючими рішеннями.
- Існуючих телеграм ботів об’єднує відсутність можливості фільтрації медіа вмісту соціальних мереж по якості, а також критичне обмеження кількості можливих джерел інформації.

Виходячи з цього, в даній роботі вирішено зосередитися на покритті

функціональності, якої не вистачає в сукупності існуючих ботів Telegram для повноцінного вирішення проблеми агрегації медіа контенту соціальних мереж та його відбору за якістю.

РОЗДІЛ 2
ВИБІР І ОБГРУНТУВАННЯ ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОГРАМИ МЕДІА АГРЕГАТОРА

2.1 Telegram Bot API

Для реалізації задачі агрегації контенту з соцмереж було вирішено розробити бота для популярного в Україні месенджера Telegram. Бот – це автоматизована програма, що може бути інтегрована в ту чи іншу систему в вигляді звичайного користувача. Існують різні типи ботів, як от чат-боти, боти-інформатори, боти-асистенти та інші, проте більшість з них орієнтовані на спрощення життя користувачів. Найчастіше для керування ботами використовується встановлений для кожного з них набір команд, на які бот реагує запрограмованими діями.

В різних соцмережах та інших платформах боти мають різні можливості та права, але в месенджері Telegram боти мають доступ майже до того самого набору функцій що і звичайний користувач. Ключова різниця між справжнім користувачем та ботом в Telegram з точки зору системи є прив’язка користувача до номеру його мобільного телефону, а бота – до, так званого, токену бота, що надається йому при створенні (рис. 2.1) [19].

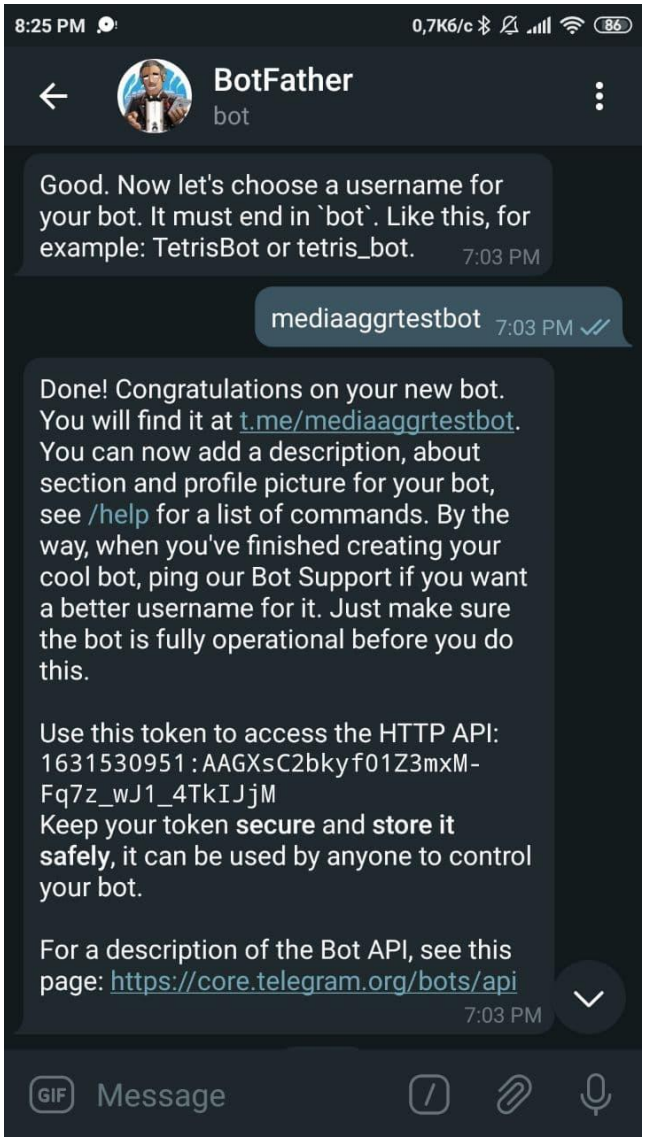


Рис. 2.1. Приклад процесу присвоєння токену бота Telegram

Можливість створення та роботи ботів Telegram забезпечується базовими архітектурними рішеннями даної платформи. Даний месенджер використовує власний протокол шифрування під назвою MTProto. MTProto – це криптографічний протокол, задачею якого є шифрування повідомлень користувачів у месенджері Telegram. В його основі лежить комбінація протоколу Диффі-Хеллмана, симетричного алгоритму шифрування AES та ряд хеш-функцій (рис. 2.2). Він допускає використання end-to-end шифрування з необов'язковим звірянням ключів [23].

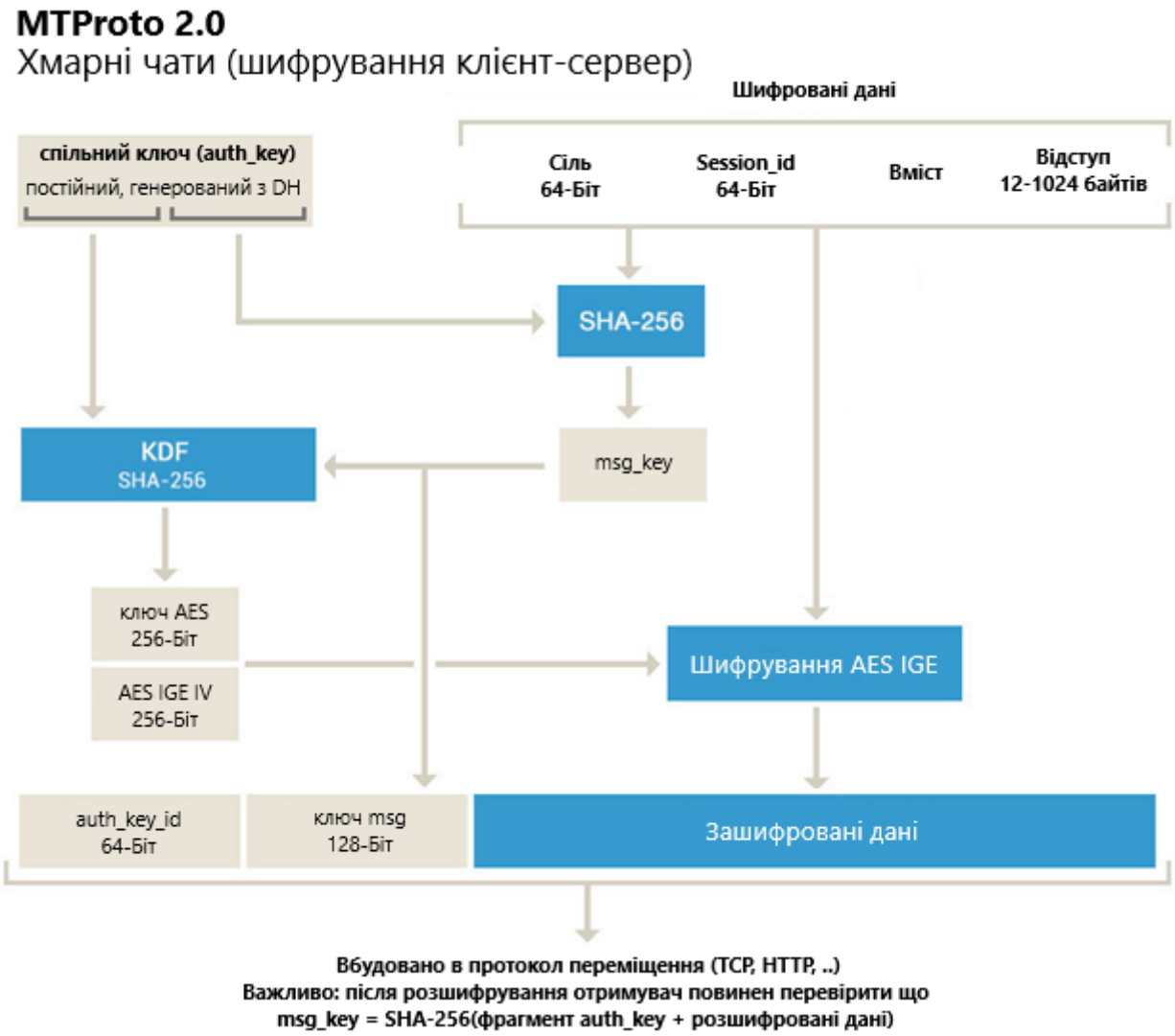


Рис. 2.2. Схема роботи протоколу шифрування MTProto

На основі цього протоколу побудовано MTProto API, або Telegram API [21], що дозволяє додатку Telegram зв'язуватись з сервером. Оскільки Telegram API є відкритим, клієнт месенджера може бути частково або повністю написаним самостійно для індивідуального використання. Здебільшого, така політика і робить ботів Telegram гранично функціональними та гнучкими.

Розробниками платформи Telegram також було створено спеціальну надбудову над API месенджера під назвою Telegram Bot API. Вона представляє собою посередницький сервер між клієнтом та Telegram API, що забезпечує обробку шифрування та спрощує взаємодію з сервером (рис. 2.3). Це робить

можливою розробку ботів при відсутності знань про особливості роботи MTProto API та однойменний протокол шифрування [20]. Окрім цього, таким чином забезпечується можливість реалізації деяких методів функціонування ботів, як от робота через вебхуки.

Рис. 2.3. Схематичне зображення функції Telegram Bot API

Базова функціональна вимога до будь-якого бота Telegram є можливість відправляти запити на сервер Telegram та отримувати від нього відповіді, або оновлення. Для цього Telegram Bot API підтримує два способи: поллинг і вебхук. Поллинг – регулярне відправлення запитів на сервер для отримання оновлень. Вебхук – реалізація оберненої схеми, коли сервер сам відправляє оновлення на URL клієнта коли це потрібно [35].

2.2 Мова програмування Python

Для написання програмного коду агрегатора медіа контенту бота Telegram було обрано мову Python. Python є чи не найпопулярнішою мовою для розробки ботів та інших інтегрованих систем на даний момент і досить поширеною мовою програмування в цілому. Про це говорить статистика сервісу PYPL, що базується на аналізі кількості запитів Google про уроки та навчальні курси тої чи іншої мови (рис. 2.4) [24]. В мережі існує багато матеріалів, які стосуються розробки такого роду програм на Python, що дозволить зосередитись на вирішенні проблеми агрегації контенту, замість вирішення технічних проблем. Цьому сприятиме також велика кількість бібліотек, серед яких можна обрати потрібні для конкретної задачі

інструменти, що можуть спростити роботу з Telegram Bot API, зокрема допомогти автоматизувати процес обміну запитами. Окрім цього, завдяки адаптивності даної мови програмування, написаний код можна буде перевести в будь-яку іншу мову програмування без особливих проблем.

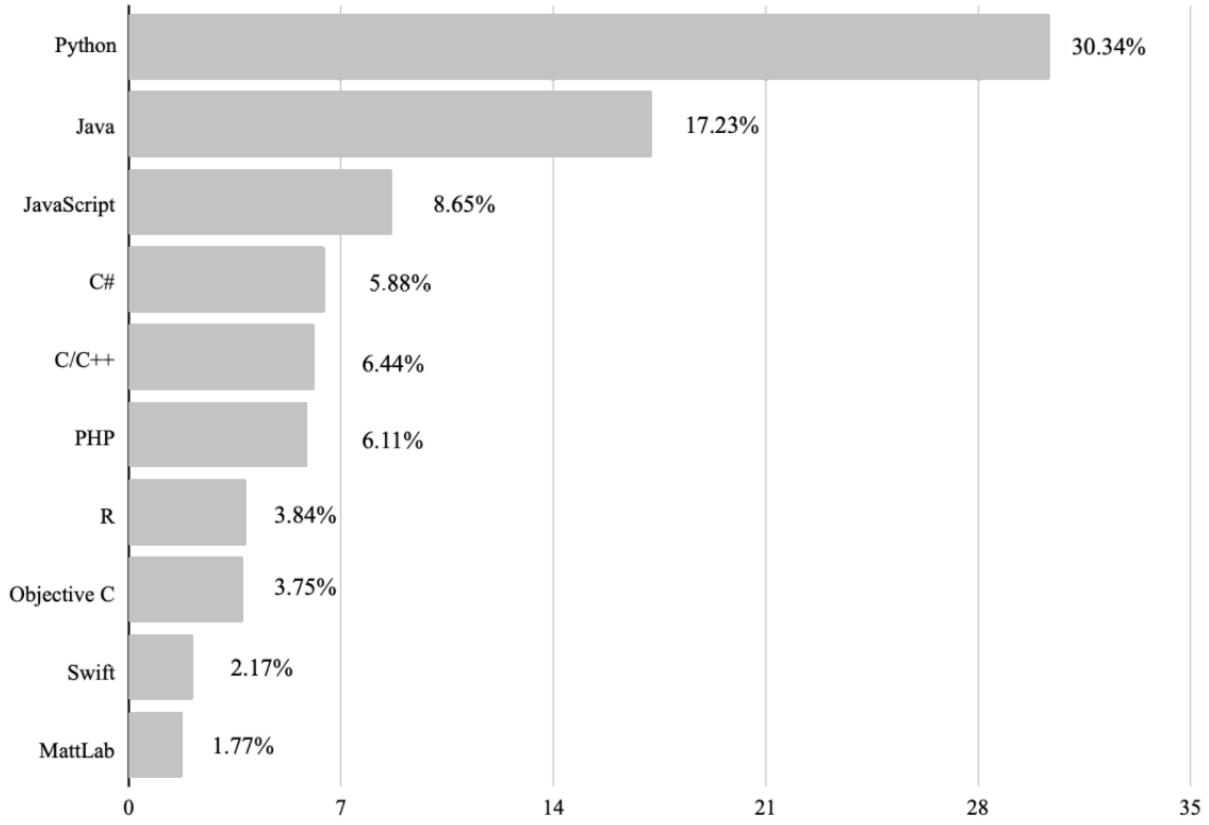


Рис. 2.4. Світовий рейтинг популярності мов програмування за даними PYPL станом на травень 2021 року

Область застосування мови програмування Python дуже обширна. Його часто використовують для створення скриптів по автоматизації, різного роду інтегрованих систем та іншого. За весь час свого існування дана технологія постійно розвивалась та знаходила нових прихильників серед розробників. В цьому процесі створювалися стандартні бібліотеки для підтримки сучасних технологій, для прикладу, розрахованих на роботу базами даних, мережевими протоколами та іншого, а також розроблялися користувацькі бібліотеки, яких на даний момент існує вже дуже велика кількість.

Мова програмування Python має свої особливості, що виділяють її в

порівнянні з іншими мовами. Серед них не на останньому місці стоїть кросплатформенність. Python - це інтерпретована мова, а його інтерпретатори були створені для багатьох платформ. Тому його запуск без особливих проблем можливий на, практично, будь-якій платформі.

Окрім цього перевагою перед частиною інших мов є величезна кількість, середовищ розробки, фреймворків та сервісів, орієнтованих на Python. Це дозволяє знайти інструменти та створити умови, комфортні для виконання завдань різного роду. Те саме стосується джерел інформації та матеріалів, що допомагають працювати з даною технологією, адже на будь-яке технічне запитання стосовно розробки засобами Python в мережі, імовірно, знайдеться відповідь.

Однією з найбільш очевидних особливостей Python є інтуїтивність його синтаксису та загальна філософія, що спрямована на спрощення процесу написання коду. Адже однією з найбільш важливих його переваг є динамічна типізація. Це ще один спосіб зробити написання коду простішим за рахунок уникнення складнощів з типами даних, що стоїть на ряду з відсутністю операторних дужок як в С-подібних мовах програмування та заміною їх на табуляцію [22].

Проте за такі серйозні переваги доводиться платити швидкодією. В випадку великих розробок та проектів Python поступається в швидкості таким мовам як C++, C, Java, проте все ще перемагає Ruby чи PHP. Роль тут відіграють як динамічна типізація, так і автоматичне керування пам'яттю. Все ж, в даної мови програмування є також особливості, за рахунок яких швидкодія покращується. Це і можливість підключення бібліотек на мові С, і попередня компіляція коду в байт-код [36].

Окрім вище перерахованих особливостей, на користь даної мови програмування говорять рейтинги. За даними дослідження порталу DOU.UA 2021 року [8] Python займає 4 місце серед всіх мов програмування по кількості розробників, що використовують його в якості основної мови (рис. 2.5). Це означає що будь-яка розробка засобами даної мови є доцільною на даний момент і, імовірно, буде залишатись такою ще довгий час.

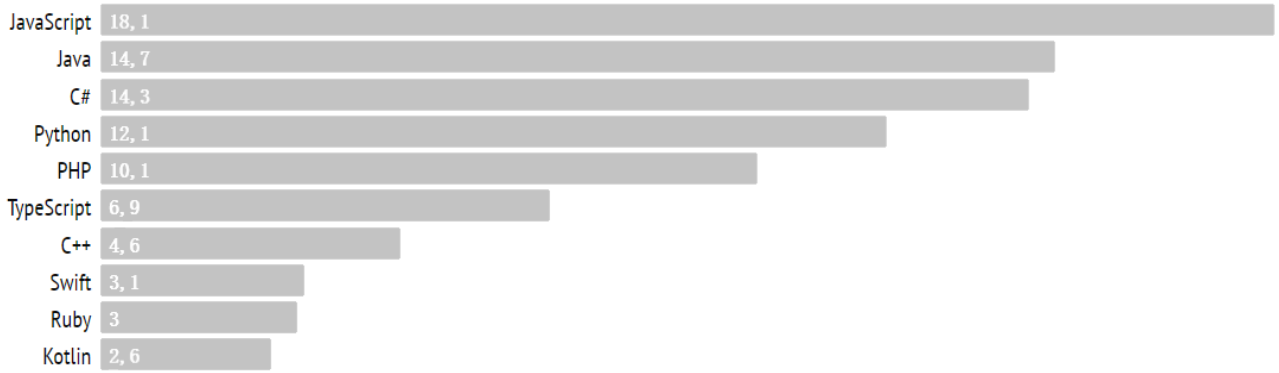


Рис. 2.5. Рейтинг популярності мов програмування серед розробників України у 2021 році

Більше того, Python лідирує з дуже великим розривом по популярності у сфері обробки даних, про що говорить інший рейтинг (рис. 2.5), сформований на основі того ж дослідження. Це особливо актуально в контексті тематики даної роботи, адже будь-які телеграм боти можна в тій чи іншій мірі підвести під дану категорію.

Рис. 2.5. Рейтинг популярності мов програмування в сфері обробки даних серед розробників України у 2021 році

2.3 Бібліотеки для розробки Telegram ботів

Існує ціла низка бібліотек для мови програмування Python, що

реалізовують функціонал керування ботом Telegram та спрощують програмування двостороннього зв’язку бота та серверу месенджера. Найбільш популярними та помітними в мережі модулями такого призначення є Python-telegram-bot, PyTelegramBotAPI і Aiogram.

2.3.1 Python-telegram-bot

Бібліотека python-telegram-bot реалізовує інтерфейс для Telegram Bot API версії 5.2, та є сумісною із мовою Python версії 3.6 та новіших.

Окрім чистої реалізації API, Python-telegram-bot містить збірку класів високого рівня, за допомогою яких розробка телеграм ботів стає більш інтуїтивно простою та зрозумілою (рис. 2.6). В структурі бібліотеки для даних класів відведено окремий підмодуль telegram.ext. Додатково дана бібліотека доступна для завантаження в вигляді окремого пакету Python-telegram-bot-raw і в цьому випадку вона не включає в себе розширення telegram.ext [26].

Перевагою використання модуля Python-telegram-bot є велика кількість матеріалів і документації та активна спільнота розробників, що може спростити вирішення технічних питань. Проте, оскільки робота над даною технологією була почата ще у 2015 році, а з того часу зазнавав змін навіть Telegram Bot API, методи реалізації керування ботом в ній вважаються застарілими та придатними лише для вирішення дуже простих задач. І хоча бібліотека продовжує підтримуватись розробниками, її актуальність знижується через наявність новіших аналогів, що пропонують більш зручні інструменти реалізації тієї ж задачі.

```python
import logging
from telegram import Update, ForceReply
from telegram.ext import Updater, CommandHandler, MessageHandler, Filters, CallbackContext

logging.basicConfig(
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s', level=logging.INFO
)

logger = logging.getLogger(__name__)

def start(update: Update, _: CallbackContext) -> None:
    """Send a message when the command /start is issued."""
    user = update.effective_user
    update.message.reply_markdown_v2(
        fr'Hi {user.mention_markdown_v2()}\!',
        reply_markup=ForceReply(selective=True),
    )

def help_command(update: Update, _: CallbackContext) -> None:
    """Send a message when the command /help is issued."""
    update.message.reply_text('Help!')

def echo(update: Update, _: CallbackContext) -> None:
    """Echo the user message."""
    update.message.reply_text(update.message.text)

def main() -> None:
    """Start the bot."""
    updater = Updater("TOKEN")
    dispatcher = updater.dispatcher
    dispatcher.add_handler(CommandHandler("start", start))
    dispatcher.add_handler(CommandHandler("help", help_command))
    dispatcher.add_handler(MessageHandler(Filters.text & ~Filters.command, echo))
    updater.start_polling()
    updater.idle()

if __name__ == '__main__':
    main()
```

Рис. 2.7. Приклад реалізації елементарного бота Telegram засобами бібліотеки Python-telegram-bot

2.3.2 Aiogram

Aiogram - це асинхронний фреймворк, що великою мірою використовує декоратори і реалізовує зручні інструменти для розробки ботів Telegram [30]. Оскільки дана технологія є відносно новою, розробники змогли врахувати досвід вже існуючих аналогів при її розробці. Aiogram може використовуватись для написання простих програм (рис. 2.8), проте, оскільки він надає достатньо потужні інструменти для реалізації повномасштабних проектів, поріг входження в дану технологію є значно вищим ніж в інших розглянутих бібліотек [29]. Даний модуль включає в себе такі функціональні

частини як машина кінцевих станів [27], middleware [28] та обробник помилок. Окрім цього, даний фреймворк досить активно оновлюється та характеризується широкою спільнотою користувачів.

```python
import logging

from aiogram import Bot, Dispatcher, executor, types

API_TOKEN = 'BOT TOKEN HERE'

logging.basicConfig(level=logging.INFO)

bot = Bot(token=API_TOKEN)
dp = Dispatcher(bot)


@dp.message_handler(commands=['start', 'help'])
async def send_welcome(message: types.Message):
    await message.reply("Hi!\nI'm EchoBot!\nPowered by aiogram.")


@dp.message_handler(regexp='(^cat[s]?$|puss)')
async def cats(message: types.Message):
    with open('data/cats.jpg', 'rb') as photo:
        await message.reply_photo(photo, caption='Cats are here 😺')


@dp.message_handler()
async def echo(message: types.Message):
    await message.answer(message.text)


if __name__ == '__main__':
    executor.start_polling(dp, skip_updates=True)
```

Рис. 2.9. Приклад реалізації елементарного бота Telegram засобами бібліотеки Aiogram

2.3.3 PyTelegramBotAPI

Дана бібліотека реалізовує інтерфейс Telegram Bot API аналогічно до Python-telegram-bot та містить співставний функціонал [29]. PyTelegramBotAPI, за рахунок більш продуманої архітектури, дозволяє писати нескладних ботів, при цьому вкладаючись в помітно меншу кількість рядків

коду ніж Python-telegram-bot (рис. 2.8). Частково це зв'язано з тим, що модуль PyTelegramBotAPI є новішим аналогом модуля Python-telegram-bot, і на даний момент дана бібліотека досить активно оновлюється розробниками та є найбільш помітною серед користувачів на просторах мережі Інтернет. І хоча даний модуль може уступати деяким новітнім та більш функціональним бібліотеками у гнучкості та наборі інструментів, активна спільнота, якісна документація та відносно невисокий поріг входження робить PyTelegramBotAPI оптимальним інструментом для розробки Telegram бота агрегатора медіа контенту.

```
import telebot

API_TOKEN = '<api_token>'

bot = telebot.TeleBot(API_TOKEN)


@bot.message_handler(commands=['help', 'start'])
def send_welcome(message):
    bot.reply_to(message, """\
Hi there, I am EchoBot.
I am here to echo your kind words back to you. Just say anything nice and I'll say the exact same thing to you!\
""")


@bot.message_handler(func=lambda message: True)
def echo_message(message):
    bot.reply_to(message, message.text)


bot.polling()
```

Рис. 2.8. Приклад реалізації елементарного бота Telegram засобами бібліотеки PyTelegramBotAPI

2.4 Середовище розробки Visual Studio Code

Маючи на меті реалізувати агрегатор медіа контенту, для написання програмного коду мовою програмування Python було обрано редактор коду Visual Studio Code. Причинами стали високий рівень деталізації налаштувань даного IDE, потужні засоби налагодження коду, вбудована система контролю версій та можливість встановлення додаткових розширень, зокрема Jupyter Notebook.

Visual Studio Code - це безкоштовний редактор програмного коду,

створений компанією Microsoft для таких операційних систем як Windows, Linux та MacOS. В його базові функції входить налагодження коду, підсвічування синтаксису, інтелектуальне заповнення коду, рефакторинг коду та вбудована система контролю версій Git. Даний редактор програмного коду можна використовувати з різними мовами програмування, включаючи Python (рис. 2.10). Окрім цього, дане середовище розробки є дуже гнучким і дозволяє як налаштовувати його зовнішній вигляд та змінювати параметри вбудованих інструментів, так і встановлювати розширення, що впроваджують додаткову функціональність [31].

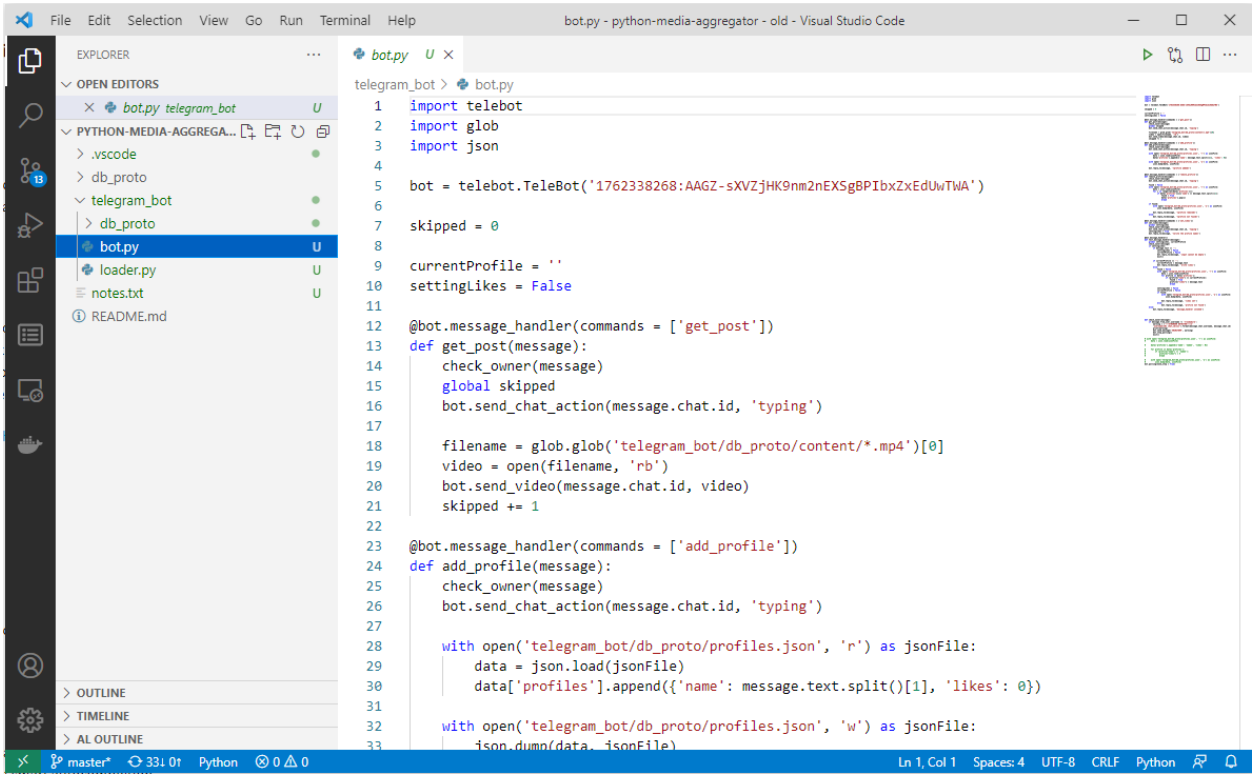


Рис. 2.10. Інтерфейс редактора коду Visual Studio Code при використанні з файлами Python

Контроль версій - це вбудована функція Visual Studio Code. Під неї відведено спеціальну вкладку на панелі бокового меню, де можна отримати доступ до основних налаштувань системи контролю версій та переглянути зміни, внесені до поточного проекту, але не записані в репозиторій (рис. 2.11). Щоб розпочати використовувати цю функцію необхідно зв'язати Visual Studio

Code з будь-якою підтримуваною системою контролю версій (Git, Subversion, Perforce тощо). Це дозволить створювати репозиторії, а також робити запити push і pull безпосередньо з програми Visual Studio Code.

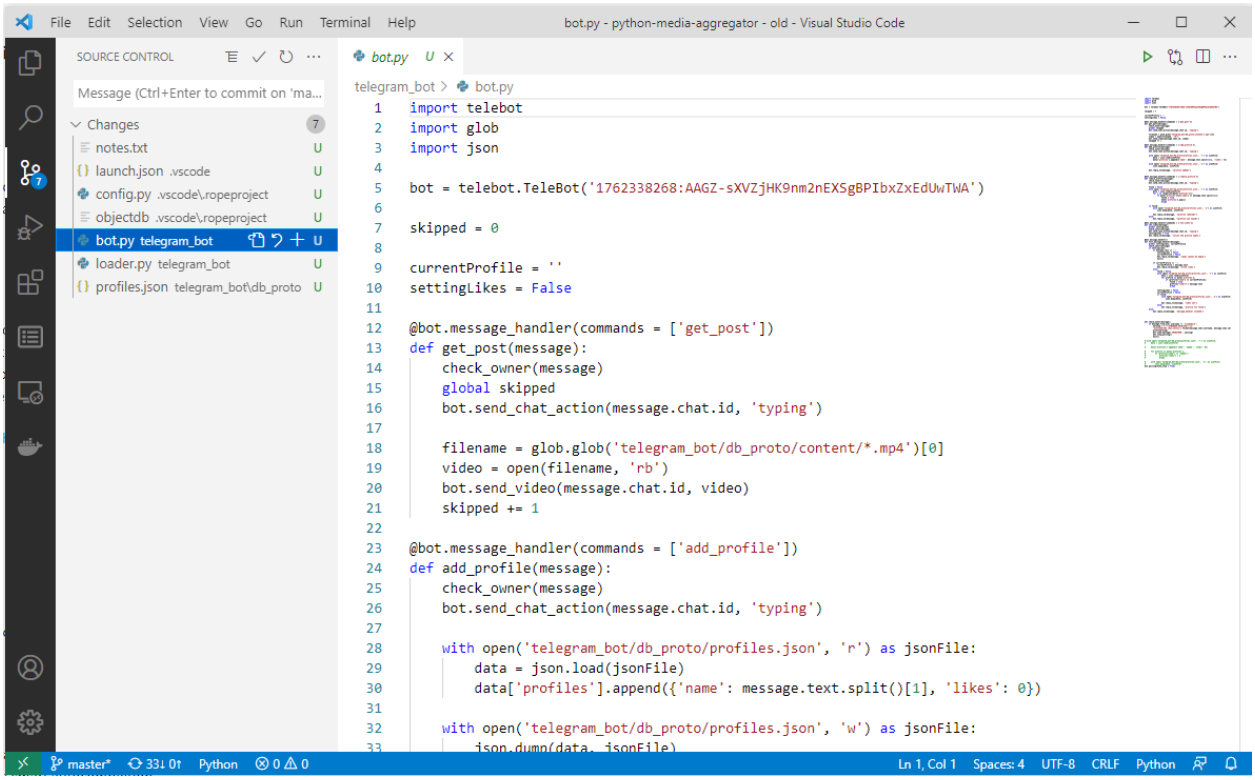


Рис. 2.11. Інтерфейс вбудованої системи контролю версій в редакторі коду Visual Studio Code

Замість системи проектів дане IDE дозволяє користувачам відкривати один або кілька каталогів, які потім можна зберегти для подальшого повторного використання. Це забезпечує його роботу з будь-якою мовою програмування, незалежно від особливостей структури проекту. Багато функцій Visual Studio Code не відображаються через меню або користувацький інтерфейс, але доступ до них здійснюється за допомогою палітри команд.

Окрім цього, дане середовище розробки надає можливість встановлення додаткових модулів для його розширення. За рахунок цього можна суттєво підсилити існуючий функціонал в тих областях, де це необхідно. Одним з розширень Visual Studio Code, що були обрані для використання в розробці Telegram бота агрегатора медіа контенту, є Jupyter (рис. 2.12) – розширення,

що вносить в даний редактор коду основні функції середовища Jupyter Notebook (рис. 2.13).

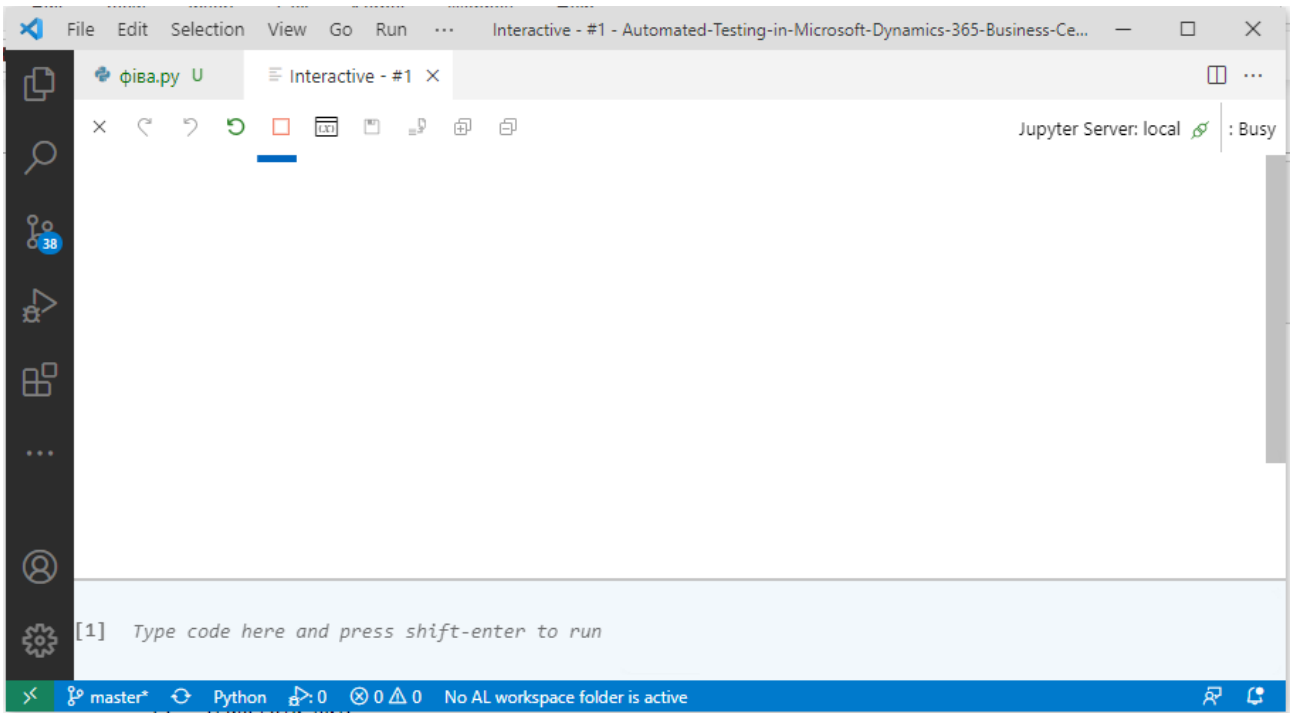


Рис. 2.12. Інтерфейс розширення Jupyter в редакторі коду Visual Studio Code

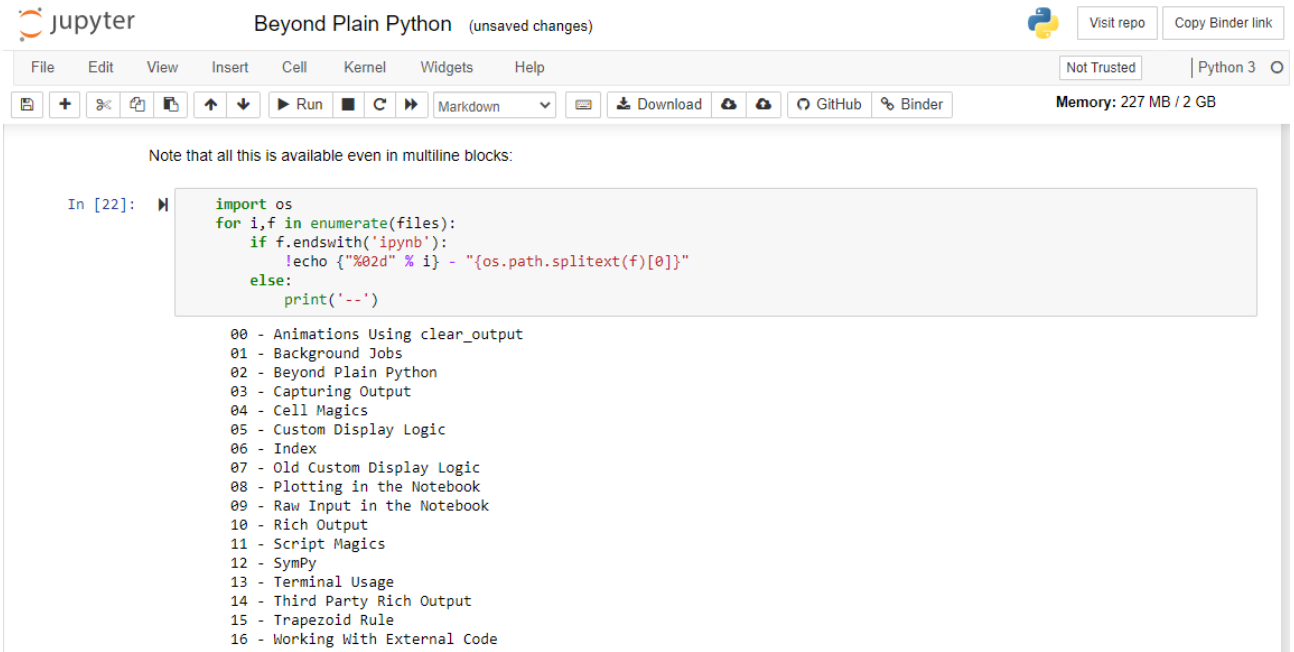


Рис. 2.13. Інтерфейс середовища Jupyter Notebook

Jupyter Notebook - це веб-програма з відкритим вихідним кодом, яка

дозволяє створювати та обмінюватися документами, що містять водночас код, звичайний текст, зображення та інше [32]. Це забезпечує інтуїтивно зрозумілий робочий процес сприяє ітеративній і швидкій розробці. В даному випадку, розширення може бути використане для миттєвого запуску вибраних рядків коду в ізоляції. Це дозволить пришвидшити процес написання коду та його налагодження.

2.5 База даних PostgreSQL

Для зберігання усіх особистих даних та налаштувань користувачів, а також файлів медіа вмісту було вирішено скористатися платформою PostgreSQL. Такий вибір зумовлений сучасністю даної технології та існуванням великої кількості документації по ній.

Ідеологія PostgreSQL базується на використанні об’єктно реляційної моделі, що є його основною особливістю в порівнянні з іншими популярними платформами. Дана технологія підтримує велику кількість типів даних, включаючи деякі умовно рідкісні, дозволяє описувати свої типи даних. В даній платформі велике значення надається цілісності даних, а її код є відкритим [34].

Хоча більшість просунутих функцій PostgreSQL не знайдуть застосування в даній роботі, мати їх в своєму арсеналі інструментів не буде зайвим. Окрім цього, завдяки передовій популярності даної технології, для неї існує велика кількість якісної документації та навчальних матеріалів, а це дозволить скоротити середній час вирішення технічних питань.

Для керування базою даних з коду, написаного на мові програмування Python, буде використовуватись бібліотека psycopg2, найпопулярніший на даний момент драйвер баз даних PostgreSQL для мови Python (рис. 2.14). Даний модуль реалізовано на мові C як обгортка над lobpq, що забезпечує швидкою та безпеку даних. Крім цього psycopg2 дозволяє приводити деякі типи даних Python до типів даних PostgreSQL.

Рис. 2.14. Схема взаємодії програми Python з базою даних PostgreSQL через посередництво модуля psycopg2

Висновки до розділу 2

В результаті огляду доступних інструментів та технологій в даному розділі обрано ті з них, що найкраще підійдуть для створення агрегатора медіа контенту соцмереж у вигляді бота Telegram.

Зокрема, для визначення основного напрямку руху проаналізовано можливості, що надаються розробникам платформою Telegram для створення ботів. На основі даного аналізу зроблено висновок, що найефективнішим засобом для рішення нашої задачі тут стане Telegram Bot API.

Окрім цього, для уникнення серйозних технічних перешкод при розробці системи оглянуто можливості та переваги мови програмування Python в сфері розробки ботів Telegram. З'ясовано, що дана мова має весь необхідний для даних цілей набір інструментів та часто використовується саме в таких цілях.

Також оглянуто найбільш потужні бібліотеки Telegram, що спрощують роботу з Telegram Bot API та дозволяють приділяти менше уваги технічним особливостям взаємодії з сервером Telegram. Використання такого інструменту дозволить сфокусуватись на реалізації логіки системи замість технічних нюансів. Виходячи з проведеного аналізу, вирішено використовувати бібліотеку PyTelegramBotAPI в силу її простоти, ефективності та якісної документації.

Для забезпечення безперешкодного та швидкого процесу створення системи проаналізовано всі переваги редактора коду Visual Studio Code при розробці на мові програмування Python. З'ясовано, що ними являються зручні та потужні інструменти налагодження, рефакторингу, контролю версій, а також можливість встановлення функціональних розширень середовища розробки.

Для забезпечення розроблюваної системи структурованим сховищем даних обрано систему управління реляційними базами даних PostgreSQL завдяки своїй популярності, швидкодії та присутності якісної документації.

РОЗДІЛ 3
РОЗРОБКА СИСТЕМИ АГРЕГАТОРА КОНТЕНТУ

3.1 Формулювання сценаріїв взаємодії користувача із системою

Для правильної побудови структури програми в першу чергу потрібно передбачити основні сценарії роботи користувача в системі. Базова функціональність агрегатора медіа контенту включає в себе встановлення низки користувацьких налаштувань для визначення вибірки отримуваного контенту, завантаження відфільтрованих публікацій відповідно даних налаштувань та показ вмісту користувачеві у встановленому порядку. На основі цього можна сформувати ключові сценарії роботи користувача:

- Реєстрація в системі
- Додавання профілю для відслідковування
- Додавання критерію фільтрації контенту
- Завантаження публікацій із соцмереж
- Відображення чергової публікації

3.1.1 Реєстрація в системі

В звичному понятті цього слова користувач не потребує реєстрації в системі розроблюваного бота, оскільки основні задачі даної функції, що стосуються безпеки та синхронізації даних забезпечуються системою реєстрації, реалізованою в месенджері Telegram. Проте, все ж існує необхідність в наданні персонального доступу до даних кожному користувачеві бота медіа агрегатора, забезпечуючи їх безпеку та конфіденційність у відношенні інших користувачів. Зокрема, усі особисті налаштування, обрані джерела контенту та критерії фільтрації, переглянуті публікації повинні бути видимими лише для користувача, який їх встановлював та переглядав. Це допоможе реалізувати в системі можливість повноцінної персоналізації потоку інформації та забезпечить її конфіденційність.

Програма телеграм бота працює з користувачем через посередництво інтерфейсу та серверу месенджера, таким чином маючи доступ до відкритих через Telegram API даних користувача. Зокрема, в такі дані входить унікальний ідентифікатор користувача в Telegram, який доцільно використати в якості ідентифікатора і в розроблюваній системі медіа агрегації. Доступ є також і до інформації про користувача Telegram, наприклад до його імені та нікнейму в системі, проте ці позиції можуть змінюватись, тому їх використання доцільне лише в косметичних цілях та в такому разі буде існувати потреба в їх періодичному оновленні.

Загальний механізм роботи ботів Telegram передбачає, що перша взаємодія користувача з будь-яким ботом супроводжується викликом вбудованої команди «start» [37]. Це дає змогу проводити реєстрацію нових користувачів або інші сценарії в момент коли користувач використовує бота вперше. У випадку розроблюваної системи медіа агрегатора команда “start” ініціює процес створення запису про нового користувача в базі даних і тим самим робить можливим подальшу його роботу з системою. Спершу дана команда відправляється з клієнта на сервер Telegram, який реєструє її появу. Після цього сервіс бота, який працює шляхом поллингу, тобто періодичного звіряння з сервером Telegram на наявність оновлень, надсилає черговий запит та отримує у відповідь команду «start» в супроводі інших даних про користувача, та відправлене повідомлення в вигляді команди. Обробляючи дану команду, система медіа агрегації перевіряє існування даного користувача в базі даних і, в випадку його відсутності, створює новий запис. Після цього сервіс надсилає на сервер Telegram відповідь у вигляді повідомлення користувачу, що містить коротку інструкцію користування ботом. Таке повідомлення означає, що процес спрощеної реєстрації в системі пройшов успішно і користувач може використовувати функціональність, що надається ботом медіа агрегатора (рис. 3.1).

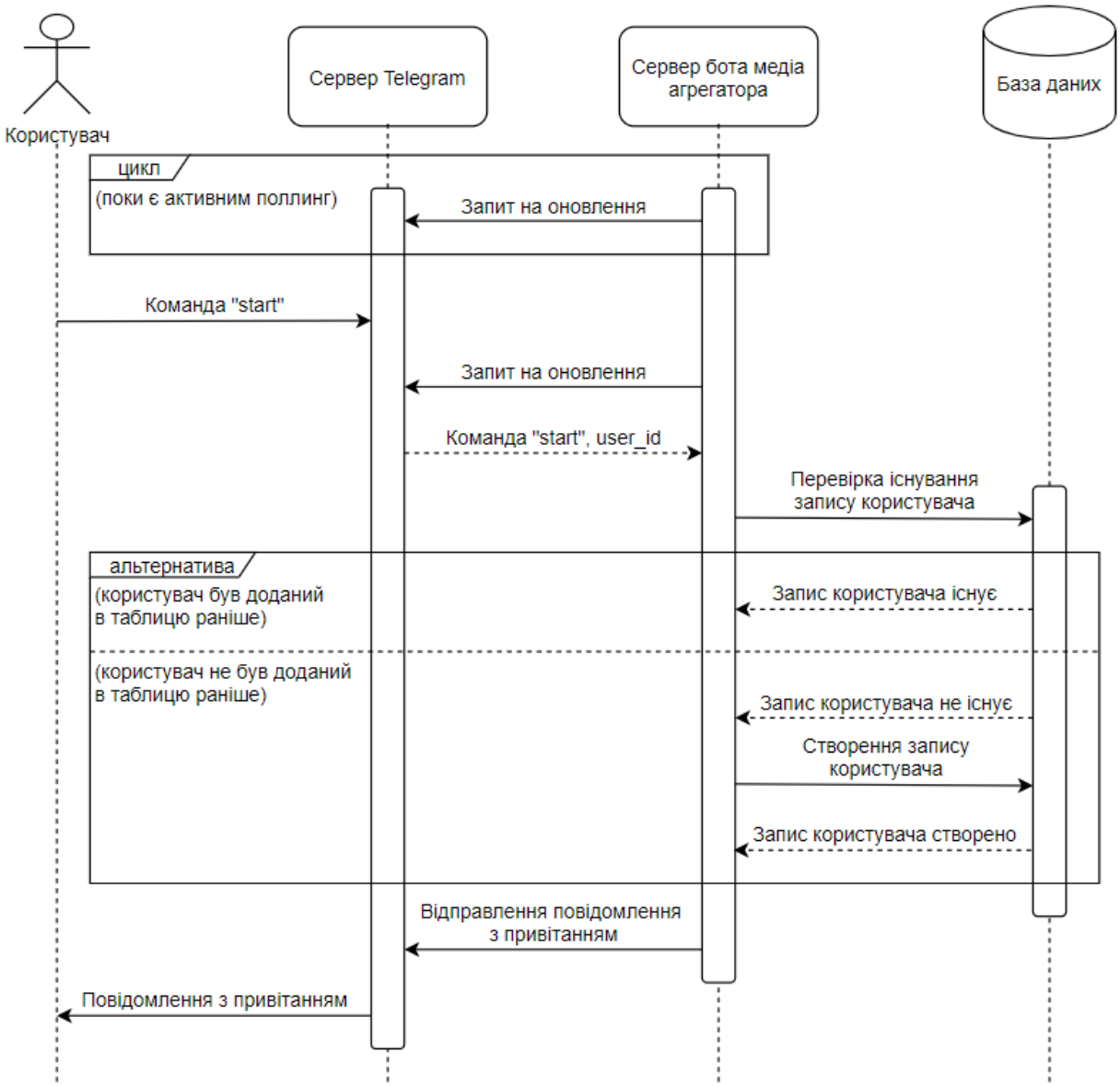


Рис. 3.1. Діаграма послідовності реєстрації користувача в системі

3.1.2 Додавання профілю для відслідковування

Оскільки основною функцією бота агрегатора медіа контенту є відображення публікацій з різних соціальних мереж в одному потоці чату месенджера Telegram, першим що повинен зробити користувач після початку роботи з системою є створення списку профілів у соцмережах, які потрібно відслідковувати. Для цього необхідно виділити окрему команду «add profile», у відповідь на яку розроблювана програма запитує у користувача до якої соціальної мережі належить профіль, а також який ідентифікатор належить даному профілю. Коли всі необхідні дані отримано від користувача,

відбувається перевірка чи запис про даний профіль для даного користувача не був створений раніше. Якщо ні, тоді запис повинен створюватись в базі даних та повинно відбуватись сповіщення в клієнті про успішне додавання профілю. Інакше користувачу надсилається повідомлення про те, що профіль вже існує в базі даних (рис. 3.2).

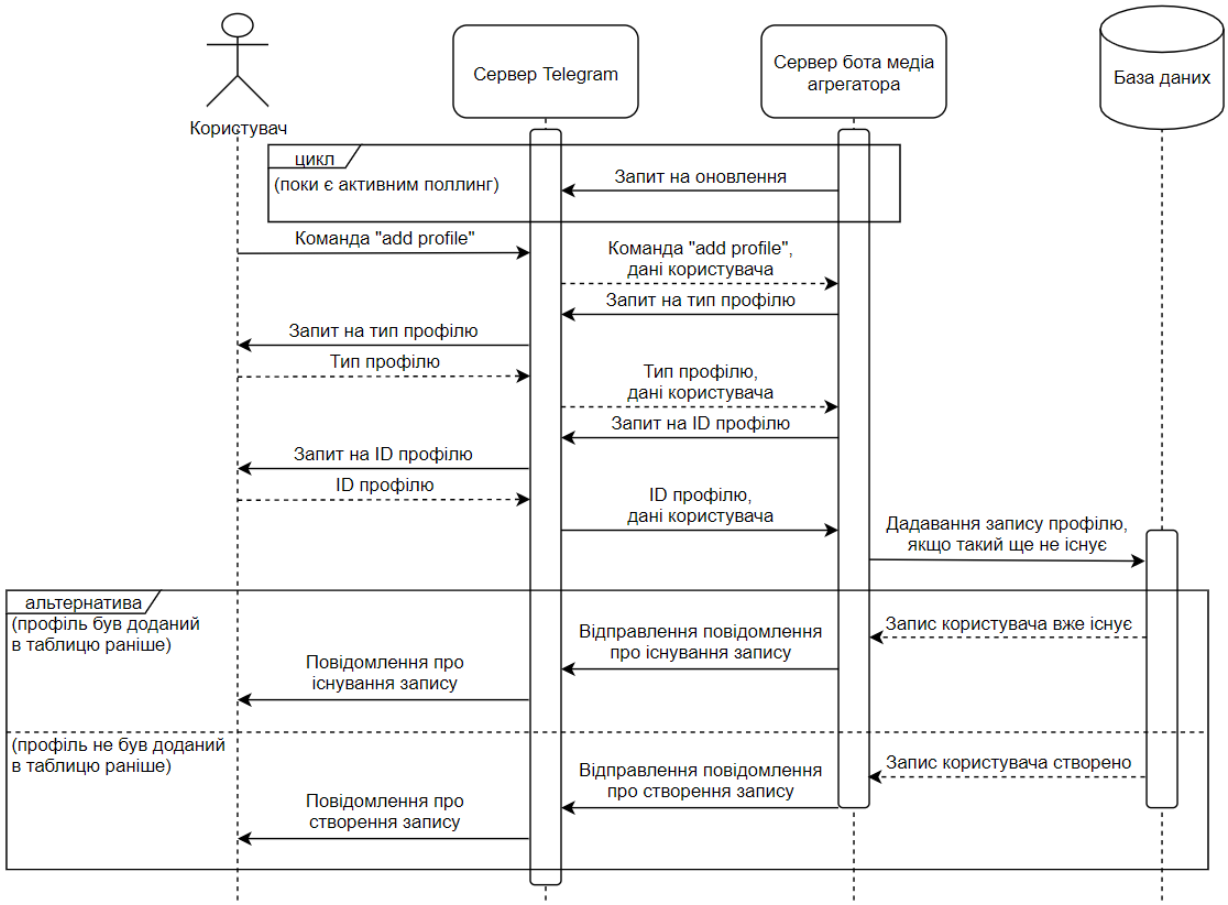


Рис. 3.2. Діаграма послідовності додавання профілю для відслідковування

3.1.3 Додавання критерію фільтрації контенту

Особливістю розроблюваної системи медіа агрегації є можливість фільтрації медіа вмісту за його умовною якістю. Для реалізації даної функціональності необхідно надати користувачу можливість створювати критерії відбору контенту з прив’язкою до одного з раніше створених профілів. Окрім цього, потрібно врахувати, що типів критеріїв може бути декілька в силу різноманітності отриманих з API даних про завантажені медіа дані. Для прикладу, з API мережі Instagram можна виділити такі критерії як

відношення кількості вподобань до кількості переглядів публікації, а також наявність або відсутність в публікації відео файлів.

Додавання критерію відбору контенту відбувається шляхом надсилання команди «add criteria», на що сервер реагує запитом про профіль до якого необхідно застосувати фільтр, тип критерію та його значення. Після того як користувач ввів всі необхідні дані, створюється запис критерію в базі даних для вибраного профілю та користувача. У випадку якщо такий критерій певного профілю та типу вже був створений раніше, то його значення перезаписується новим, вказаним користувачем, та відображається відповідне повідомлення. Якщо ж запис до цього не існував, він створюється в системі і про це також оповіщається користувач (рис. 3.3).

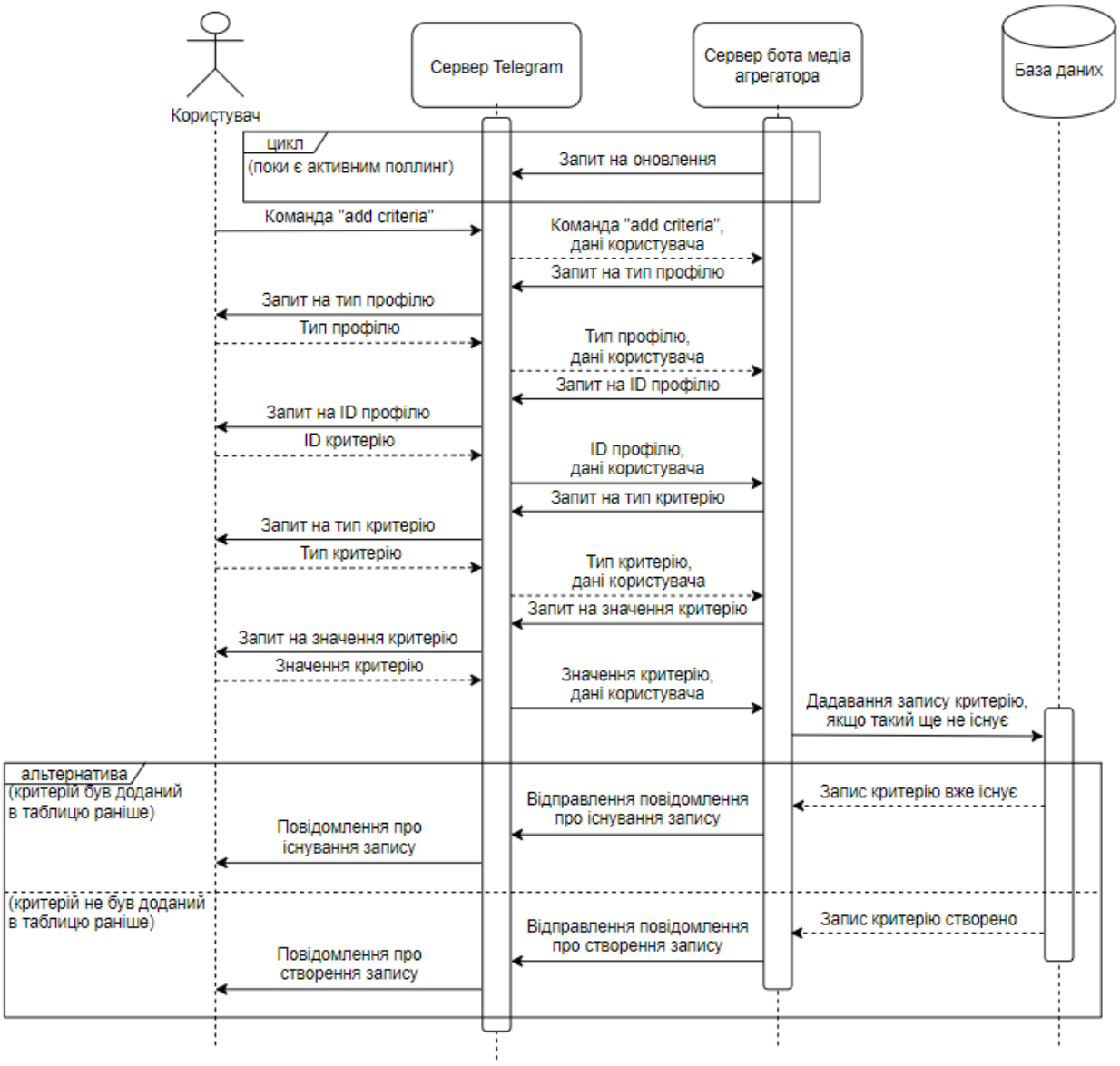


Рис. 3.3. Діаграма послідовності додавання критерію відбору контенту

3.1.4 Завантаження публікацій із соцмереж

Оскільки основною задачею розроблюваного бота Telegram є агрегація різноманітного контенту з різних соцмереж у чаті Telegram, найбільш об’ємною та важливою зі структурних частин розроблюваної функціональності є функція завантаження публікацій. В ній обробляються особливості роботи з API різних соціальних мереж, реалізовується фільтрація контенту, ведеться облік завантажених файлів.

Дана функція може бути викликана користувачем за допомогою команди «fetch posts», на яку серверна частина системи бота відповідає запитом про кількість днів, що будуть використані для визначення найбільш давньої дати завантажених публікацій. Після отримання необхідних даних, сервіс сповіщає користувача про початок завантаження, а потім запускає основний алгоритм даного модуля. Спершу система проведе пошук по таблицях профілів та критеріїв даного користувача, встановлюючи зв’язки між отриманими даними та формуючи фільтр публікацій для кожного профілю в соціальних мережах. Після цього, за встановленими критеріями проводиться завантаження контенту для кожного профілю та його збереження у відповідній директорії файлової системи. При цьому для кожного завантаженого файлу створюється запис в базі даних для можливості контролювати завантаженість постійної пам’яті системи та відслідковувати усі завантажені дані. Необхідно також зазначити, що публікації, записи про які існують в таблиці переглянутих публікацій для всіх користувачів що обрали для відслідковування певний профіль не повинні бути завантаженими. Проте, оскільки не всі API соціальних мереж надають достатньо гнучкі інструменти фільтрації контенту для безпосередньої реалізації даної вимоги, файли зайвих публікацій видаляються одразу після основного завантаження контенту. Коли всі вищеописані процедури завершено, сервер оповіщає користувача про закінчення роботи функції та готовність до показу найбільш свіжих публікацій (рис. 3.4).

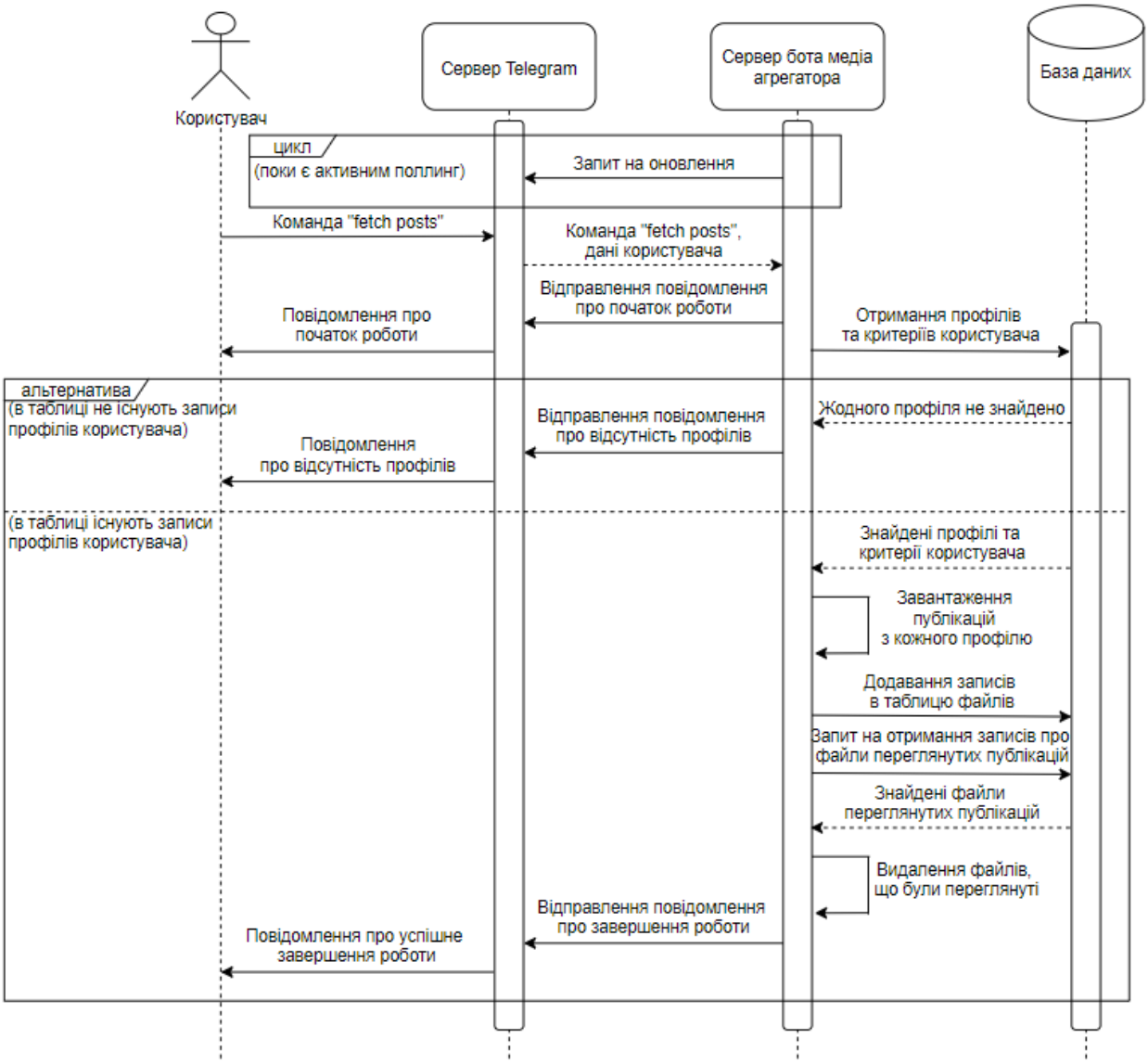


Рис. 3.4. Діаграма послідовності завантаження публікацій із соцмереж

3.1.5 Відображення чергової публікації

Останній із ключових сценаріїв роботи системи це показ публікації, тобто те, для чого проводяться всі налаштування реалізовані в інших сценаріях. В випадку коли було проведено завантаження публікацій для користувача, він може використати команду «get post» для того щоб переглянути останню за часом публікацію. Дана команда не потребує додаткових параметрів і вводу від користувача. Після її запуску сервер знаходить файли відповідної публікації, компонує їх в одне повідомлення і відправляє користувачеві. Після цього публікація записується в таблицю

переглянутих публікацій і, якщо вона не може бути потенційно потрібна іншим користувачам, файли публікації та відповідні їм записи в таблиці видаляються. В результаті роботи сценарію користувач отримує повідомлення з черговою публікацією, або сповіщення про відсутність завантажених публікацій (рис. 3.5).

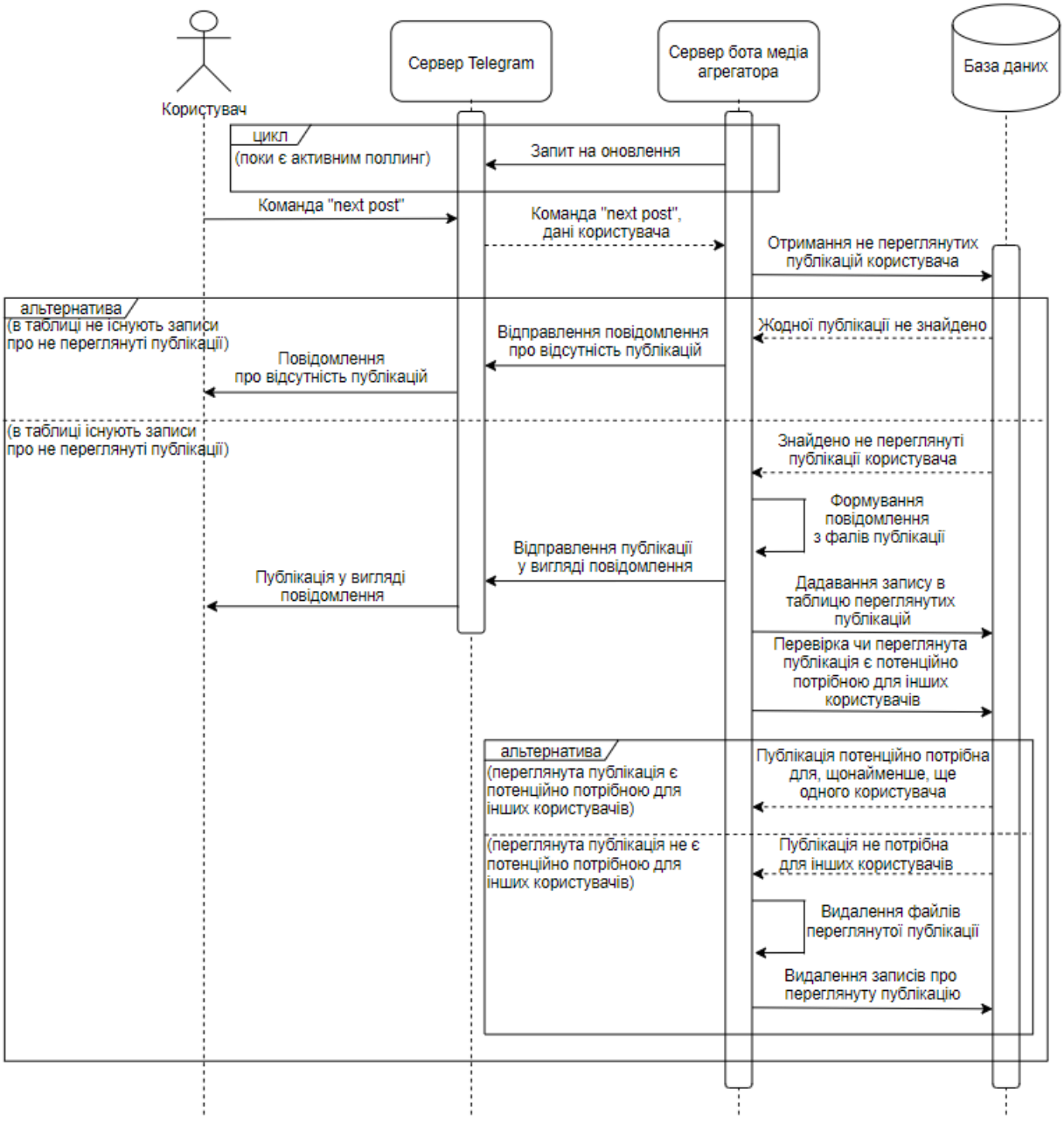


Рис. 3.5. Діаграма послідовності показу наступної публікації

3.2 Формування структури бази даних

Для зберігання всіх користувацьких налаштувань, особистих даних та інших службових даних спроектовано базу даних PostgreSQL, яка складається з 6 таблиць (рис. 3.6):

- Users – таблиця для зберігання даних про користувачів системи, зокрема унікального ідентифікатора Telegram, дати реєстрації та останньої активності, тег користувача та службове поле для відслідковування стадії діалогу. Кожен запис відповідає одному реальному користувачеві.
- Profiles – таблиця для зберігання набору профілів, що відслідковуються для кожного користувача. Кожен запис має тип, тобто одну з підтримуваних соціальних мереж, унікальний ідентифікатор профілю в соцмережі та прив’язку до одного з існуючих користувачів.
- Criterias – таблиця для зберігання набору критеріїв фільтрації контенту з кожного профілю для кожного користувача. Кожен запис має тип та значення критерію і прив’язку до запису профілю, який в свою чергу прив’язаний до певного користувача.
- Posts – таблиця для ведення обліку всіх завантажених публікацій. Кожен запис прив’язаний до профілю, з якого був завантажений, та до користувача, для якого відбувалось завантаження.
- Seen_posts – таблиця для обліку переглянутих публікацій. Кожен запис зв’язаний з одною з завантажених публікацій та одним користувачем.
- Files – таблиця для обліку всіх завантажених файлів публікацій, необхідна тому що одна публікація може містити декілька файлів різних форматів. Тому кожен запис прив’язаний до однієї з завантажених публікацій та містить шлях до файлу.

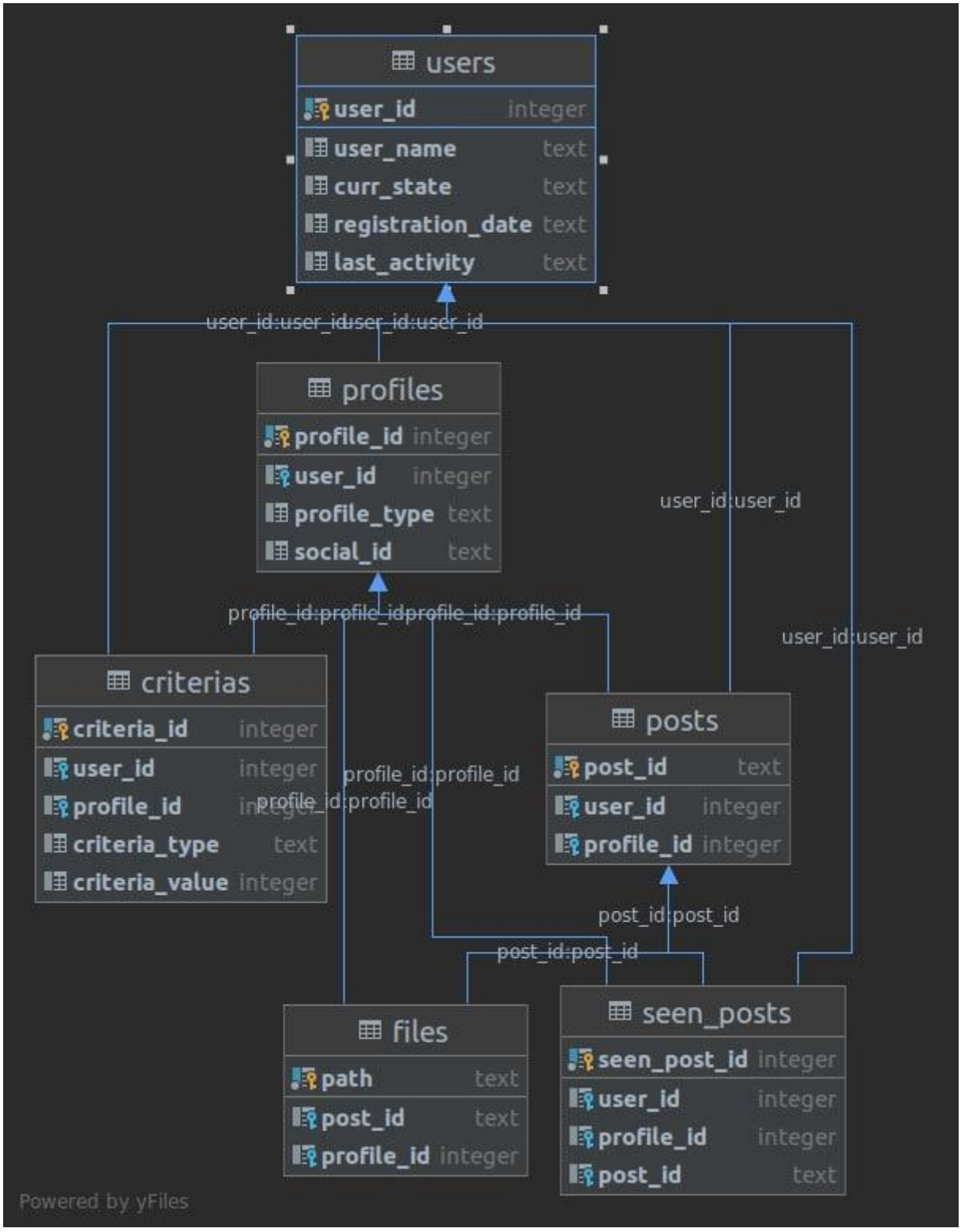


Рис. 3.6. Модель даних розробленої бази даних

3.3 Розробка компонентів системи телеграм бота

Даний бот Telegram побудовано на мові Python з використанням бібліотеки PyTelegrambotAPI, яка реалізовує функціонал керування Telegram ботом та спрощує програмування двостороннього зв’язку бота та серверу

месенджера. Більшість ботів, що базуються на даній технології працюють способом поллингу, оскільки це загальноприйнята практика для даної бібліотеки. Такий підхід в цьому модулі є найбільш функціонально забезпечений, існує повний набір необхідних для його реалізації інструментів.

Програма бота агрегатора контенту з соціальних мереж реалізовує всі ключові сценарії взаємодії користувача з системою, як от реєстрація користувача, додавання профілю для відслідковування, додавання критерію відбору контенту, завантаження публікацій з соцмереж в залежності від встановлених раніше профілів та критеріїв, показ чергової публікації користувачеві. В рамках вищеописаного, відбувається також взаємодія з базою даних, та API кожної з підтримуваних соціальних мереж.

Оскільки всі реалізовані сценарії працюють окремо один від одного, а робота з базою даних та API є дуже об’ємною по коду, програму розбито на декілька структурних елементів:

- Main – точка входу в програму, підмодуль, який проводить автентифікацію бота в месенджері Telegram, організовує запуск поллингу (рис. 3.7).

```
8
9   import telebot
10
11  BOT_TOKEN = os.getenv("BOT_TOKEN")
12
13  bot = telebot.TeleBot(BOT_TOKEN)
14
15
16  bot.polling()
17
18
```

Рис. 3.7. Фрагмент коду підмодуля Main

- Config – підмодуль, в якому зберігаються або обробляються всі програмовані налаштування системи, як от змінні середовища, що містять персональний токен Telegram бота, або параметри підключення до бази даних PostgreSQL (рис 3.8).

```
import os

DATABASE_URL = os.getenv('DATABASE_URL')
if not DATABASE_URL:
    print('You have not set DATABASE_URL')
    quit()

BOT_TOKEN = os.getenv('BOT_TOKEN')
if not BOT_TOKEN:
    print('You have not set BOT_TOKEN')
    quit()
```

Рис. 3.8. Фрагмент коду підмодуля Config

- RegisterHandler – підмодуль, що обробляє запит на реєстрацію користувача, створює відповідний запис в базі даних та записує туди індивідуальний ідентифікатор користувача в Telegram, його ім’я, час реєстрації, та інші дані (рис. 3.9).

```
@bot.message_handler(commands=['start', 'START'])
def start_message(message):
    user_name = message.from_user.first_name

    if message.from_user.last_name:
        user_name = f"{user_name} {message.from_user.last_name}"

    bot.send_message(message.chat.id,
                     f"Hellol, {user_name}",
                     reply_markup=keyboards.keyboard1)

    dbmanager.register_user(message.chat.id,
                            user_name,
                            'main_menu',
                            time.strftime('%d/%m/%y, %X'),
                            time.strftime('%d/%m/%y, %X'))
```

Рис. 3.9. Фрагмент коду підмодуля RegisterHandler

- AddProfileHandler – підмодуль, що обробляє запит на додавання профілю в соціальній мережі для певного користувача. При цьому від користувача отримуються такі параметри як тип профілю, тобто соціальна мережа в якій він існує, та ідентифікатор профілю в соціальній мережі. Отримані дані компонуються в запис та потрапляють у відповідну таблицю бази даних (рис. 3.10).

```python
@bot.message_handler(func=lambda message: message.text == 'add profile')
def add_profile(message):
    user_name = message.from_user.first_name

    if message.from_user.last_name:
        user_name = f"{user_name} {message.from_user.last_name}"

    dbmanager.update_state(user_name,
                           'main_menu',
                           time.strftime('%d/%m/%y, %X'),
                           message.chat.id)
    bot.send_message(message.chat.id,
                     'Оберіть соцмережу',
                     reply_markup=keyboards.add_users_inline_keyboard)
```

Рис. 3.10. Фрагмент коду підмодуля AddProfileHandler

- AddCriteriaHandler – підмодуль, що обробляє запит на додавання критерію відбору контенту для одного з профілів певного користувача. При його роботі від користувача отримується інформація про тип профілю, тобто соцмережу в якій він зареєстрований, ідентифікатор профілю, тип критерію та значення фільтру. Отримані дані компонуються в запис відповідної таблиці та вставляються в базу даних (рис. 3.11).

```python
@bot.message_handler(func=lambda message: message.text == 'add criteria')
def add_criteria(message):
    user_name = message.from_user.first_name

    if message.from_user.last_name:
        user_name = f"{user_name} {message.from_user.last_name}"

    dbmanager.update_state(user_name,
                           'main_menu',
                           time.strftime('%d/%m/%y, %X'),
                           message.chat.id)
    bot.send_message(message.chat.id,
                     'Оберіть соцмережу',
                     reply_markup=keyboards.add_users_inline_keyboard)
```

Рис. 3.11. Фрагмент коду підмодуля AddCriteriaHandler

- FetchPostsHandler – підмодуль, що обробляє запит на завантаження публікацій соцмереж відповідно налаштувань профілів та критеріїв певного користувача. Для цього інформація береться з відповідних таблиць в базі даних, запускаються підмодулі для завантаження публікацій з

- різних соцмереж за вказаними профілями та критеріями (рис. 3.12).

```
@bot.message_handler(func=lambda message: message.text == 'fetch posts')
def fetch_posts(message):
    user_name = message.from_user.first_name

    if message.from_user.last_name:
        user_name = f"{user_name} {message.from_user.last_name}"

    dbmanager.update_state(user_name,
                           'main_menu',
                           time.strftime('%d/%m/%y, %X'),
                           message.chat.id)
    bot.send_message(message.chat.id,
                     'Оберіть соцмережу',
                     reply_markup=keyboards.add_users_inline_keyboard)
```

Рис. 3.12. Фрагмент коду підмодуля FetchPostsHandler

- GetPostHandler – підмодуль, що обробляє запит на показ чергової публікації з завантаженої вибірки. Для цього перевіряється наявність завантаженого контенту, його сортування по даті та часу публікації та пошук файлів останньої з них. Скомпоновані в повідомлення елементи публікації у файлах відправляються користувачеві. Після цього відбувається видалення відправленої публікації у випадку коли вона не може бути потенційно потрібна іншому користувачеві (рис. 3.13).

```
89  @bot.message_handler(func=lambda message: message.text == 'get post')
90  def get_post(message):
91      video = open(
92          videopath,
93          'rb')
94      bot.send_message(message.chat.id,
95                       'ВИдео оброботіваеца',
96                       reply_markup=keyboards.keyboard1)
97
98      bot.send_video(message.chat.id, video, timeout=60)
99      bot.delete_message(message.chat.id,
100                         message.message_id + 1)
101     video.close()
102
```

Рис. 3.13. Фрагмент коду підмодуля GetPostHandler

- Keyboards – підмодуль, що зберігає в собі всі клавіатури Telegram бота, котрі використовуються в різноманітних сценаріях. Клавіатури

дозволяють замінити ручний ввід команди натисканням звичайної кнопки, що суттєво покращує враження та ефективність користування програмою (рис. 3.14).

```python
import telebot

keyboard1 = telebot.types.ReplyKeyboardMarkup(True, False)

button1keyboard1 = telebot.types.KeyboardButton("fetch posts")
button2keyboard1 = telebot.types.KeyboardButton("get post")

button3keyboard1 = telebot.types.KeyboardButton("add profile")
button4keyboard1 = telebot.types.KeyboardButton("add criteria")
```

Рис. 3.14. Фрагмент коду підмодуля Keyboards

- DBManager – підмодуль, в якому розміщено всі сценарії взаємодії з базою даних, включаючи всі SQL запити. Його виокремлення дозволяє суттєво покращити чистоту програмного коду розроблюваної системи (рис. 3.15).

```python

def register_user(user_id, user_name: str, curr_state: str, registration_date: str, last_activity: str):
    conn = get_connection()
    c = conn.cursor()
    c.execute(
        'INSERT INTO users (user_id, user_name, curr_state, last_activity, registration_date) VALUES (%s, %s, %s, %s, %s)',
        (user_id, user_name, curr_state, last_activity, registration_date,)
    )
    conn.commit()


def update_state(user_id, user_name, curr_state, last_activity):
    conn = get_connection()
    c = conn.cursor()
    c.execute(
        'UPDATE users SET user_name = %s, curr_state = %s, last_activity = %s WHERE user_id = %s',
        (user_id, user_name, curr_state, last_activity)
    )
    conn.commit()
```

Рис. 3.15. Фрагмент коду підмодуля DBManager

- InstagramLoader – підмодуль, що реалізовує функціональність завантаження публікацій для обраних профілів та з обраними критеріями з соціальної мережі Instagram. При цьому фільтрація може відбуватись за двома різними критеріями: відношення вподобань до переглядів публікації та наявність в публікації відео файлів (рис. 3.16).

```
def main():
    inputs = open('telegram_bot/db_proto/profiles.txt', 'r')
    names = inputs.readlines()
    names = list(map(str.strip, names))

    loader = Instaloader_parameters('_{score}_{target}_{date_utc}')
    loader.login_paremeters()
    profiles = set(map(lambda name: Profile.from_username(loader.context, name), names))
    loader.download_profiles_custom_parameters(profiles, date_filter_factory(datetime.now()-timedelta(days=1500)), post_filter_factory(15))
```

Рис. 3.16. Фрагмент коду підмодуля InstagramLoader

Висновки до розділу 3

В результаті процесу проектування та розробки Telegram бота системи агрегації медіа контенту соцмереж визначено ключові сценарії взаємодії користувача з системою, побудовано базу даних та розроблено програмні компоненти системи.

Зокрема, для постановки чітких функціональних вимог до розробки детально описано п’ять сценаріїв роботи системи. Реєстрація користувача слугує точкою відліку усієї взаємодії. Додавання профілів та критеріїв дозволяє динамічно налаштовувати джерела публікацій у соцмережах, та накладені на них фільтри. Завантаження публікацій дозволяє застосувати раніше зроблені налаштування для відбору персонального потоку контенту. Відображення чергової публікації забезпечує користувача можливістю почергово переглядати завантажений медіа контент.

Окрім цього, для зберігання всіх користувацьких налаштувань спроектовано базу даних PostgreSQL, що складається з 5 таблиць. Таблиця users містить дані про кожного зареєстрованого користувача. Таблиці Profiles і Criterias складають основу сховища користувацьких налаштувань і зберігають дані в розрізі користувачів. Таблиці Posts та Files дозволяють вести облік всіх завантажених публікацій та їх розбиття на окремі файли. Таблиця Seen_posts дозволяє уникати повторного відображення певної публікації для одного і того ж користувача.

Для реалізації описаних сценаріїв роботи користувача в системі розроблено низку компонентів програми. Підмодулі роду Handler містять основну обробку запитів користувача. Підмодулі роду Loader забезпечують завантаження контенту з соцмереж з врахуванням вказаних критеріїв. Підмодуль DBManager включає в себе всю взаємодію з базою даних. Підмодулі Keyboard та Config мають конфігураційний характер, а підмодуль Main є точкою входу в програму та реалізує поллинг серверів Telegram.

РОЗДІЛ 4
ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ АГРЕГАТОРА КОНТЕНТУ

Розроблена програма Telegram бота агрегатора медіа контенту використовує в якості інтерфейсу клієнтську частину месенджера Telegram та реалізовує 5 основних сценаріїв взаємодії користувача з системою. Тому основою тестування системи повинна бути перевірка коректної роботи програми при виконанні користувачем даних сценаріїв.

4.1 Тестування реєстрації в системі

Для того щоб почати користування Telegram ботом необхідно, в першу чергу, знайти його через пошук у месенджері Telegram за тегом бота. В даному випадку тегом бота є @mediaaggrbot, тому потрібно відкрити пошуковий рядок, ввести тег та обрати бота з іменем Media Aggregator (рис. 4.1).

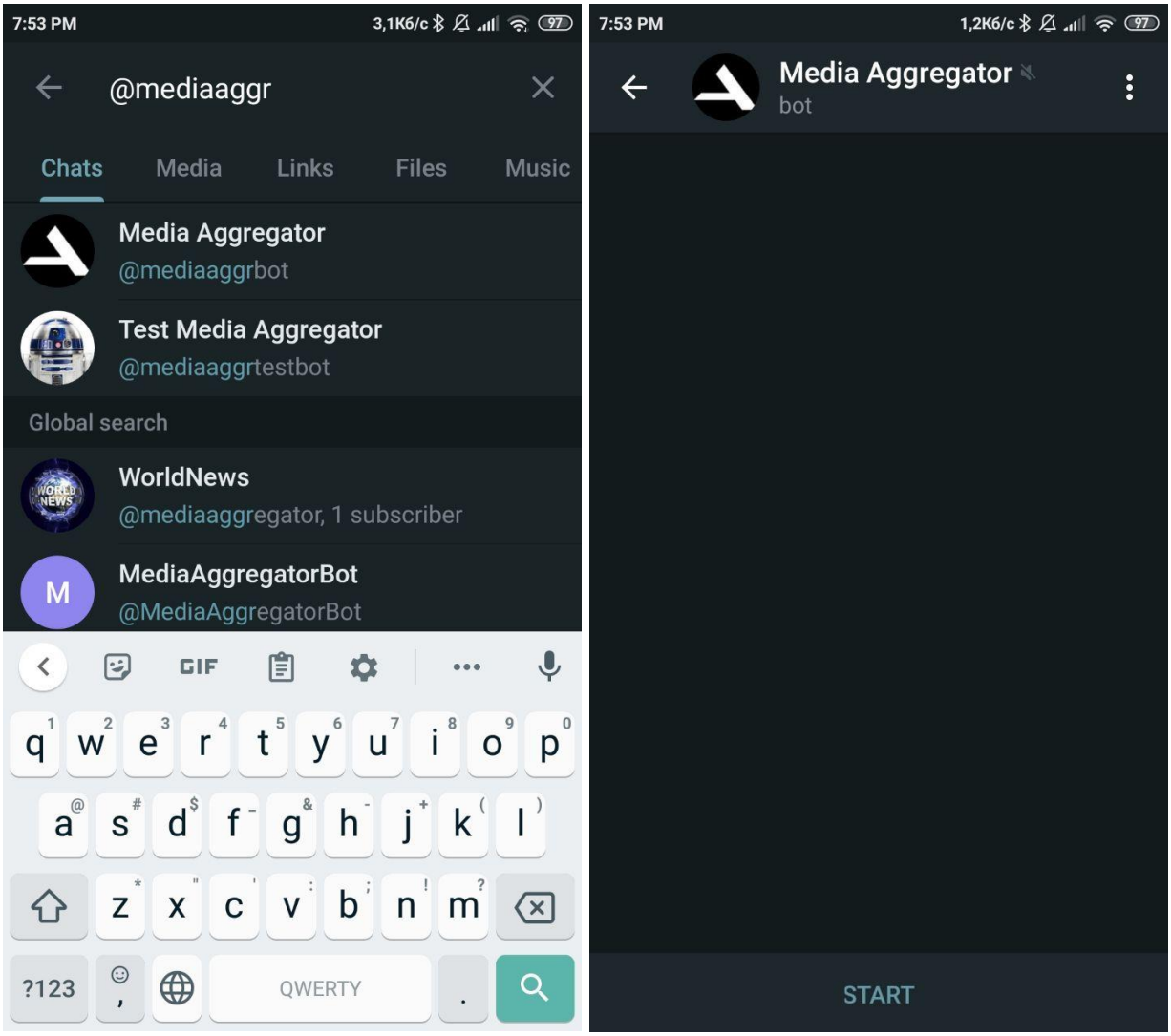


Рис. 4.1 Пошук бота Media Aggregator в месенджері Telegram

Для відправлення команди «start» можна натиснути кнопку «START» (див. рис. 4.1), або ввести «/start» в полі вводу повідомлення. Відповіддю сервера на дану команду є повідомлення про успішну реєстрацію та інструкція по користуванню ботом, а також поява клавіатури головного меню. Отже сценарій реєстрації користувача в системі відпрацював правильно (рис. 4.2).

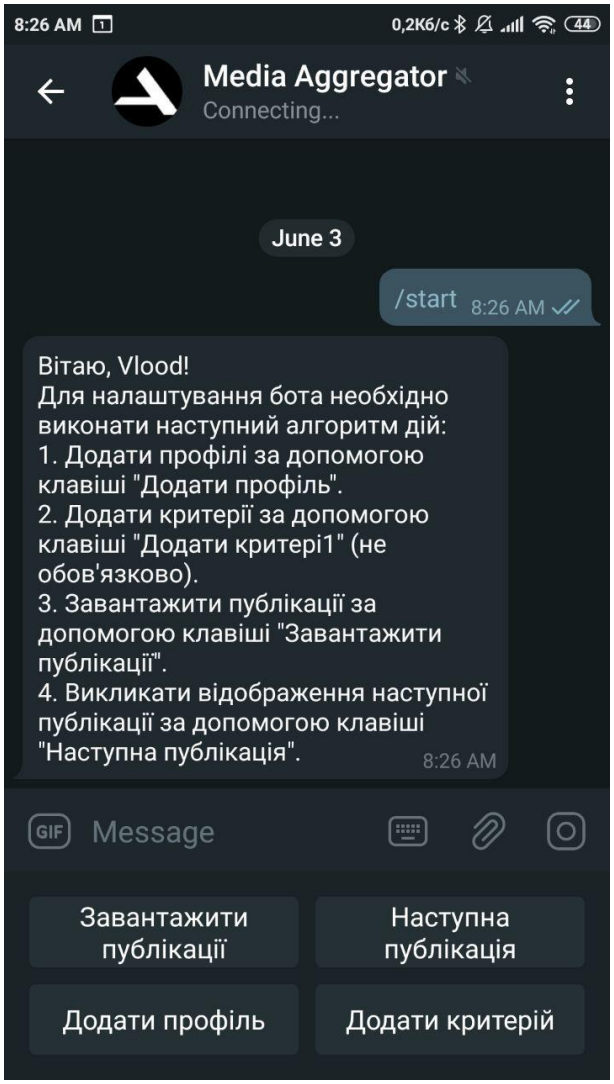


Рис. 4.2 Результат роботи команди «start»

4.2 Тестування додавання профілю

Для того щоб програма могла завантажувати публікації з соціальних мереж, необхідною передумовою є існування набору профілів, які користувач обрав для відслідковування. Для того щоб додати профіль необхідно натиснути клавішу «Додати профіль» на клавіатурі головного меню. У відповідь сервер надсилає вбудований в повідомлення список підтримуваних системою соціальних мереж. Користувач вибирає варіант Instagram, після чого сервер пропонує ввести ще й ідентифікатор профілю, тобто його унікальне ім’я в соціальній мережі. Після введення імені профілю сервер надсилає повідомлення про успішне додавання профілю та відкриває клавіатуру головного мені. Це означає що сценарій додавання профілю відпрацював правильно (рис. 4.3).

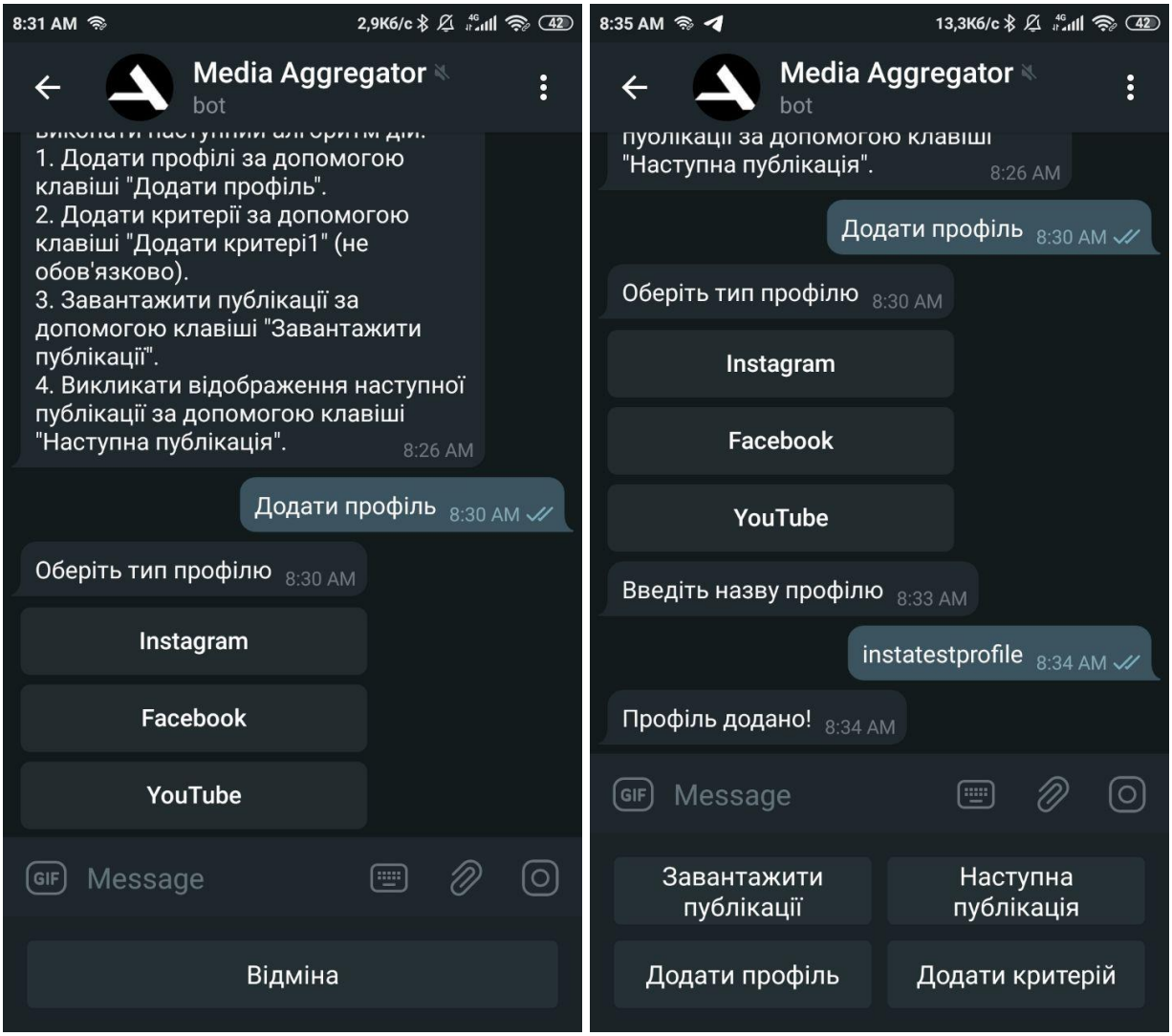


Рис. 4.3 Процес додавання профілю

4.3 Тестування додавання критерію

Для використання найбільш унікальної функції даного Telegram бота, тобто фільтрації отриманого з соціальних мереж контенту, необхідно встановити критерії на профілі в яких потрібно застосовувати фільтрацію. Для створення критерію користувач повинен натиснути клавішу «Додати критерій» на клавіатурі головного меню та обрати раніше створений профіль у відповідь на запити сервера. Після цього сервер відправляє користувачеві вбудований в повідомлення список доступних типів критеріїв, з якого необхідно обрати лише один, та надсилає запит на введення значення критерію. Відправивши потрібне значення, користувач отримує повідомлення

про успішне створення критерію від сервера та бачить появу клавіатури головного меню. Це і є ознакою правильного закінчення роботи сценарію додавання критерію (рис. 4.4).

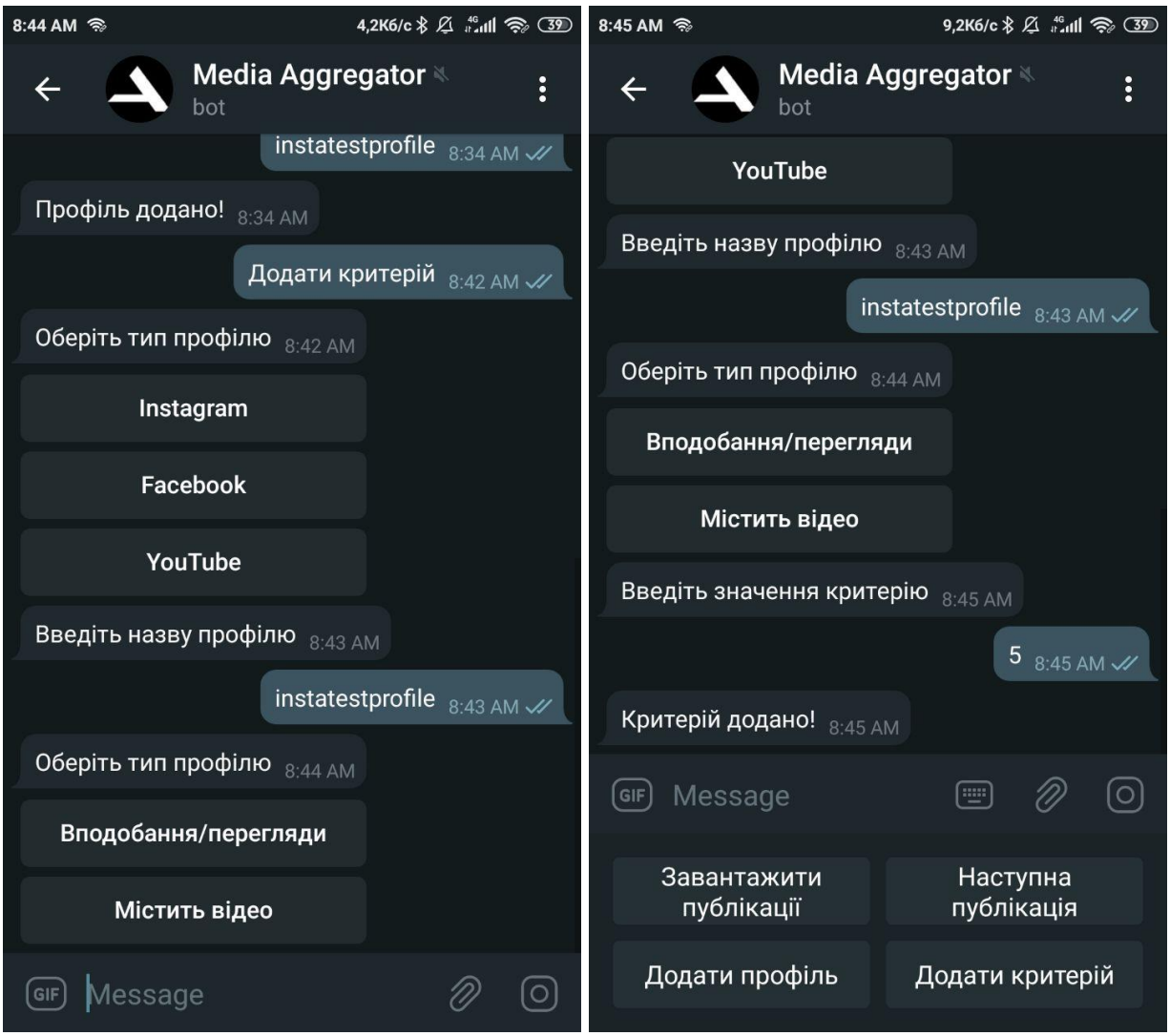


Рис. 4.4 Процес додавання критерію

4.4 Тестування завантаження контенту

Найбільша частина функціональності даного Telegram бота зосереджена саме в алгоритмі завантаження контенту з соціальних мереж. Для запуску завантаження користувач повинен натиснути клавішу «Завантажити публікації» на клавіатурі головного меню. В даному сценарії від користувача не очікується більше ніяких дій, адже всі налаштування були зроблені в попередніх сценаріях. В момент запуску функції завантаження публікацій сервер сповіщає користувача про початок роботи. Час роботи даної команди

напряму залежить від вибраних користувачем профілів та критеріїв. Після завантаження всіх публікацій сервер надсилає користувачу повідомлення про успішне завершення процедури та відкриває клавіатуру головного меню. Отже, сценарій працює правильно (рис. 4.5).

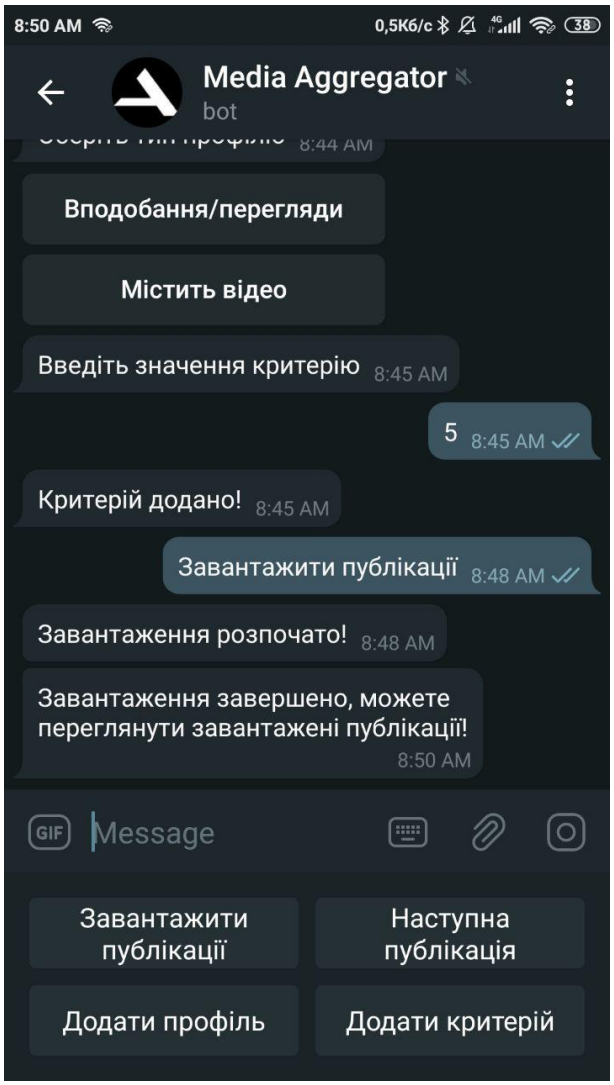


Рис. 4.5 Виклик функції завантаження публікацій

4.5 Тестування відображення публікації

Коли користувачем вже встановлено всі необхідні налаштування профілів та критеріїв і проведено процес завантаження публікацій можна приступати до використання даної розробки за її безпосереднім призначенням. Для перегляду наступної по оберненому хронологічному порядку публікації користувач повинен натиснути клавішу «Наступна публікація». У відповідь на це сервер надсилає повідомлення, що містить в собі текст та медіа контент

певної публікації. Це означає, що сценарій відображення публікації відпрацював правильно.

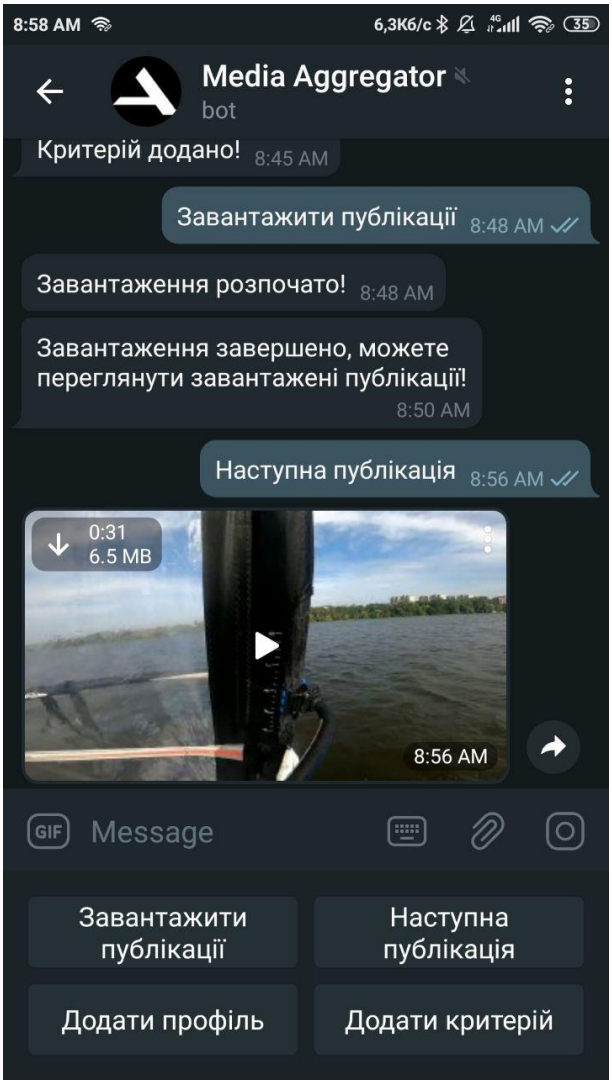


Рис. 4.6 Виклик функції відображення чергової публікації

Висновки до розділу 4

Для підтвердження раніше поставлених сценаріїв вимог до системи проведено тестування розроблених функцій з точки зору користувача. Було відтворено на практиці та деталізовано з використанням існуючих елементів інтерфейсу такі сценарії взаємодії користувача з системою як реєстрація, додавання профілю та критерію, завантаження публікацій та відображення чергової публікації. Усі тести підтвердили успішну роботу ключових функцій системи та продемонстрували базові способи використання даної розробки на практиці.

ВИСНОВКИ

В роботі розглянуто проблему надлишкової та розосередженої інформації в соціальних мережах, запропоновано методи її вирішення та перевірено їх на практиці шляхом розробки системи агрегації контенту з можливістю його відбору за критеріями якості.

Огляд та аналіз відомих існуючих аналогів показав, що існують потенційні фрагменти функціональності, спрямовані на вирішення поставленої задачі, які не були реалізовані в жодному з них. Таким є відбір контенту за критеріями, що можуть змінюватися в залежності від соціальної мережі, та які дозволяють умовно визначити рівень якості вмісту.

Система агрегації контенту розроблена у вигляді бота в месенджері Telegram, що дозволяє користувачеві застосувати особисті налаштування джерел та критеріїв відбору публікацій із соціальних мереж. Такий підхід не передбачає серйозної розробки клієнтської частини програми, але надає більші шанси застосунку знайти свою цільову аудиторію. Функціональність відбору контенту за критеріями в роботі реалізовано в простому вигляді, проте це не заважає розробці бути достатньо практичною для застосування та аналізу концепції.

Розглянута проблема може торкатися багатьох сфер діяльності людини, що залежать від потоку отримуваної інформації. Розроблена програма дозволяє пришвидшити та покращити отримуваний потік інформації із соцмереж, тому дана розробка є актуальною як концепція, а також як інструмент для особистого або комерційного використання.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. George J. Stigler. The Economics of Information. Journal of Political Economy. 1961. Vol. 69. No 3. P. 113–225. DOI: https://doi.org/10.1086/258464.

2. Park S. Information is Power. *Digital Capital.* London : Palgrave Macmillan, 2017. P. 161–183. DOI: https://doi.org/10.1057/978-1-137-59332-0_8.

3. Burrus D. How to use technology to go beyond your competition. York, NY : HarperBusiness, 2017. 376 p.

4. Importance And Importance Of Information. URL: https://www.ipl.org/essay/Importance-And-Importance-Of-Information-PKVM4XH4ACP6 (дата звернення: 18.04.2021).

5. Шаргородська В. А., Поліщук Ю. А. Роль інформації в сучасній економіці. URL: http://www.rusnauka.com/11_EISN_2016/Informatica/4_84863.doc.htm (дата звернення: 18.04.2021).

6. Отреп'єва Ю. Роль інформації в сучасному суспільстві. Природничі та гуманітрані науки. Актуальні питання : матеріали X Всеукр. студ. наук.-тех. конф. : в 2 т., 25–26 квіт. 2017 р. Тернопіль : ТНТУ ім. І. Пулюя, 2017. Т. 2. С. 71–72. URL: http://elartu.tntu.edu.ua/handle/123456789/20648.

7. Dean B. Social Network Usage & Growth Statistics: How Many People Use Social Media in 2021? URL: https://backlinko.com/social-media-users (дата звернення: 23.04.2021).

8. Digital 2021: Ukraine. URL: https://datareportal.com/reports/digital-2021-ukraine (дата звернення: 23.04.2021).

9. Українці витрачають у 8 разів більше часу на соціальні мережі, ніж на біг — результати соціального експерименту. URL: https://novaposhta.ua/news/rubric/2/id/7666 (дата звернення: 26.04.2021).

10. Digital 2021: Global Overview Report. URL: https://datareportal.com/reports/digital-2020-global-digital-overview (дата звернення: 26.04.2021).

11. Esteban Ortiz-Ospina. The rise of social media. URL: https://ourworldindata.org/rise-of-social-media#:~:text=The%20first%20social%20media%20site,%2C%20by%20platform%2C%20since%202004 (дата звернення: 26.04.2021).

12. Показники якості економічної інформації. URL: https://stud.com.ua/97158/informatika/pokazniki_yakosti_ekonomichnoyi_informatsiyi (дата звернення: 05.05.2021).

13. Lüders M. Researching social meadia: Confidentiality, anonymity and reconstructing online practices. Oslo, Norway : Cappelen Damm Akademisk, 2015. P. 77–97. URL: http://scholar.google.com/scholar_lookup?hl=en&publication_year=2015&pages=77-97&author=M.+L%C3%BCders&title=Internet+research+ethics.

14. Jessica Greene. How to use RSS feeds to boost your productivity. URL: https://zapier.com/blog/how-to-use-rss-feeds/ (дата звернення: 05.05.2021).

15. The best social media aggregators for 2020. URL: https://curator.io/blog/the-best-social-media-aggregators (дата звернення: 05.05.2021).

16. What is a content discovery platform? https://miappi.com/what_is_a_content_discovery_platform/ (дата звернення: 05.05.2021).

17. Chiou L, Tucker C. Content aggregation by platforms: The case of the news media. Journal of Economics & Management Strategy. 2017. Vol. 26. No. 4. P. 782–805. DOI: https://doi.org/10.1111/jems.12207.

18. Telegram Bots in News & Media Category Sorted by Rating. URL: https://telegramchannels.me/bots?category=news&sort=rating (дата звернення: 05.05.2021).

19. Alexander Meshcheryakov. Боты в Telegram что это такое и как они работают. URL: https://sharkdevelop.com/boty-v-telegram/ (дата звернення: 08.05.2021).

20. Bots: And introduction for developers. URL: https://core.telegram.org/bots (дата звернення: 08.05.2021).

21. Telegram APIs. URL: https://core.telegram.org/#bot-api (дата звернення 08.05.2021).

22. Oliphant, Travis E. Python for scientific computing. Computing in Science & Engineering. 2007. Vol. 9, No. 3. P. 10-20. DOI: https://doi.org/10.1109/MCSE.2007.58.

23. Jakob Jakobsen, Claudio Orlandi. On the CCA (in)Security of MTProto. Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices. Association for Computing Machinery, New York, NY : 2016. P. 113–116. DOI: https://doi.org/10.1145/2994459.2994468.

24. PYPL Popularity of Programming Languages. URL: https://pypl.github.io/PYPL.html (дата звернення 16.05.2021).

25. Шевченко Р. Рейтинг мов програмування 2021: частка Python зменшується, а TypeScript обійшов C++. URL: https://dou.ua/lenta/articles/language-rating-jan-2021/ (дата звернення: 13.05.2021).

26. Python-telegram-bot. URL: https://github.com/python-telegram-bot/python-telegram-bot (дата звернення: 14.05.2021).

27. Finite state machinery example. URL: https://github.com/aiogram/aiogram/blob/dev-2.x/examples/finite_state_machine_example.py (дата звернення: 14.05.2021).

28. Использование Middlwares в aiogram. URL: https://telegra.ph/Ispolzovanie-Middlwares-v-aiogram-08-14.

29. Крутолевич С. А. Изучение языка программирования Python в популярном мессенджере Telegram при помощи бота. 71-я научно-техническая конференция учащихся, студентов и магистрантов. Секция "Первый шаг в науку": сборник научных работ. Минск : БГТУ, 2020. С. 24–28.

30. Aiogram 2.13. URL: https://pypi.org/project/aiogram/ (дата звернення: 15.05.2021).

31. Visual Studio Code. URL: https://code.visualstudio.com/docs (дата звернення: 15.05.2021).

32. Perkel, Jeffrey M. Why Jupyter is data scientists' computational notebook of choice. Nature. 2018. Vol. 563, No. 7732. P. 145-147.

33. PostgreSQL Python. URL: https://www.postgresqltutorial.com/postgresql-python/ (дата звернення: 15.05.2021).

34. Geschwinde, E., & Schonig, H. J. PostgreSQL developers's handbook. Sams Publishing, 2002.

35. Rosid, M. A., et al. Integration telegram bot on e-complaint applications in college. IOP Conference Series: Materials Science and Engineering. Vol. 288, No. 1. Bandung : IOP Publishing, 2018.

36. Lutz, M. Programming python. O'Reilly Media, Inc., 2001.

37. Commands. URL: https://core.telegram.org/api/bots/commands (дата звернення 28.05.2021).

ДОДАТОК 1

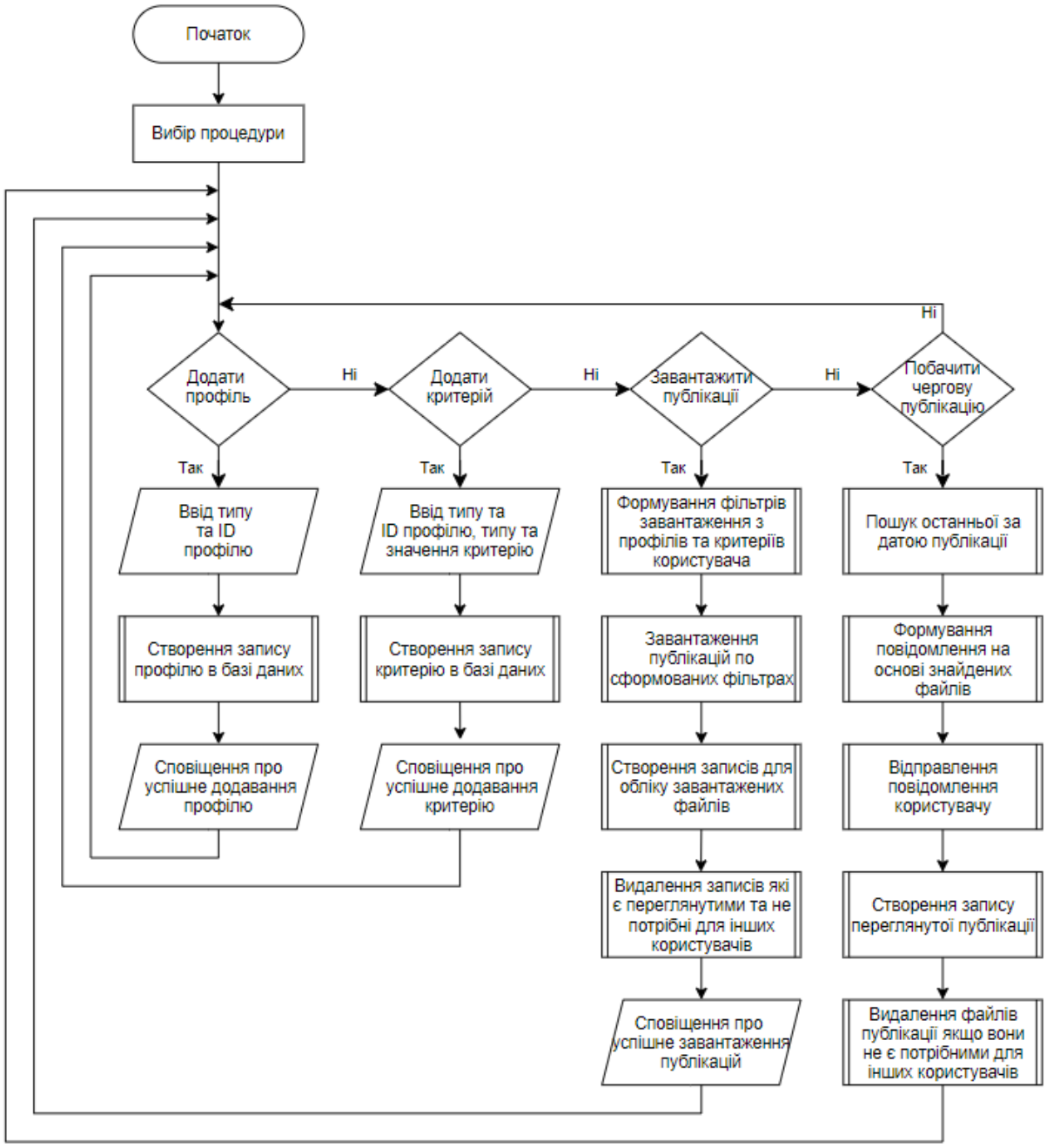
Медіа агрегатор з динамічним налаштуванням критерію відбору контенту

Принципова схема – алгоритм роботи програми

ІАЛЦ.467200.004 Д1

Аркушів 1

Київ – 2021



					ІАЛЦ.467200.004 Д1	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Іващук В. А.			*Принципова схема.*	Лит. / Арк 1 / Аркушів 1
Перев.		Новотарський М. А.				
Реценз.						
Н. Контр.		Сімоненко В. П.			Алгоритм роботи програми	НТУУ «КПІ». ФІОТ. ІП-73
Затверд.		Стіренко С. Г.				

ДОДАТОК 2

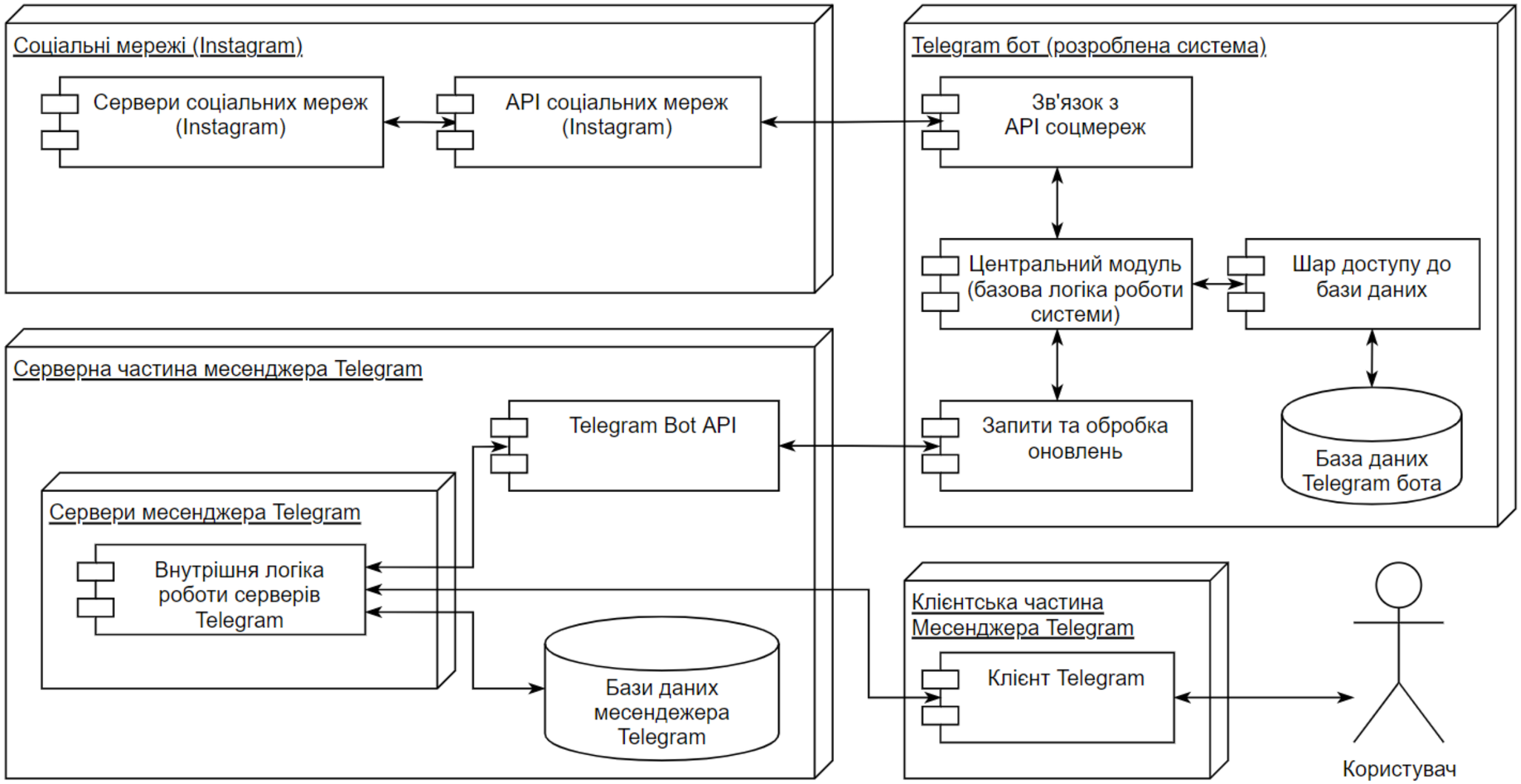
Медіа агрегатор з динамічним налаштуванням критерію відбору контенту

Структурна схема – діаграма структури системи

ІАЛЦ.467200.005 Д2

Аркушів 1

Київ – 2021



					ІАЛЦ.467200.005 Д2
Зм.	Арк.	№ докум.	Підпис	Дата	
Розроб.		Іващук В. А.			*Структурна схема.*
Перев.		Новотарський М. А.			Діаграма структури системи
Реценз.		Сімоненко В. П.			
Н. Контр.		Сімоненко В. П.			*Дипломна робота*
Затверд.		Стіренко С. Г.			

Літ.	*Маса*	*Масш.*
Аркуш	*Аркушів*	
НТУУ «КПІ». ФІОТ. ІП-73		

ДОДАТОК 3

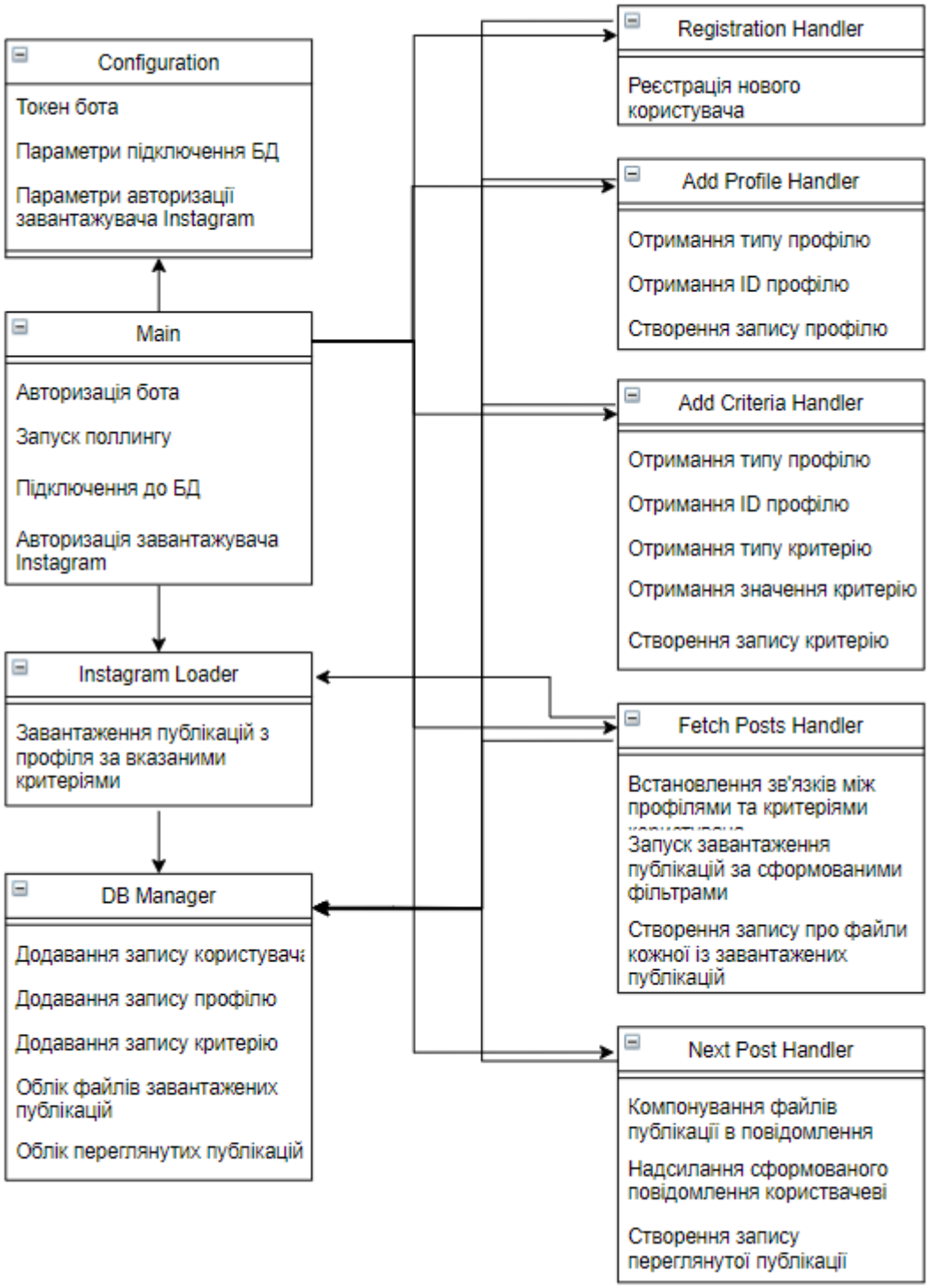
Медіа агрегатор з динамічним налаштуванням критерію відбору контенту

Функціональна схема – діаграма програмних компонентів

ІАЛЦ.467200.006 Д3

Аркушів 1

Київ – 2021



					ІАЛЦ.467200.006 ДЗ		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Іващук В. А.			*Функціональна схема.*	Лит.	Арк 1 / Аркушів 1
Перев.		Новотарський М. А.					
Реценз.							
Н. Контр.		Сімоненко В. П.			Діаграма програмних компонентів	НТУУ «КПІ». ФІОТ. ІП-73	
Затверд.		Стіренко С. Г.					

ДОДАТОК 4

Медіа агрегатор з динамічним налаштуванням критерію відбору контенту

Лістинг програми

ІАЛЦ.467200.007 Д4

Аркушів 1

Київ – 2021

Main.py

```
import telebot
import glob
import json
import os
import psycopg2
from keyboards import keyboards
from db import dbmanager
from loader import main
import time

BOT_TOKEN = os.getenv("BOT_TOKEN")

bot = telebot.TeleBot(BOT_TOKEN)
dbmanager.init_db()

@bot.message_handler(commands=['start', 'START'])
def start_message(message):
    user_name = message.from_user.first_name

    if message.from_user.last_name:
        user_name = f"{user_name} {message.from_user.last_name}"

    bot.send_message(message.chat.id,
                     f"Hellol, {user_name}",
                     reply_markup=keyboards.main_menu_keyboard)

    dbmanager.register_user(message.chat.id,
                            user_name,
                            'main_menu',
                            time.strftime('%d/%m/%y, %X'),
                            time.strftime('%d/%m/%y, %X'))


@bot.message_handler(func=lambda message: message.text == 'fetch posts')
def fetch_posts(message):
    user_name = message.from_user.first_name

    if message.from_user.last_name:
        user_name = f"{user_name} {message.from_user.last_name}"
```

```
    dbmanager.update_state(user_name,
                           'main_menu',
                           time.strftime('%d/%m/%y, %X'),
                           message.chat.id)

    bot.send_message(message.from_user.id,
                     '<b>(В РАЗРАБОТКЕ)</b>',
                     reply_markup=keyboards.main_menu_keyboard,
                     parse_mode='HTML')


@bot.message_handler(func=lambda message: message.text == 'get post')
def get_post(message):
    user_name = message.from_user.first_name

    if message.from_user.last_name:
        user_name = f"{user_name} {message.from_user.last_name}"

    dbmanager.update_state(user_name,
                           'main_menu',
                           time.strftime('%d/%m/%y, %X'),
                           message.chat.id)

    video = open(
        "/home/leap_sunrise/python-media-
aggregatorq/telegram_bot/db_proto/content/_score_shadowguy.__2020-10-04_14-
38-25_1.mp4",
        'rb')
    bot.send_message(message.chat.id,
                     'Видео обрабатывается...',
                     reply_markup=keyboards.main_menu_keyboard)

    bot.send_video(message.chat.id, video, timeout=60, reply_markup=keyboards.m
ain_menu_keyboard)
    bot.delete_message(message.chat.id,
                       message.message_id + 1)
    video.close()


@bot.message_handler(func=lambda message: message.text == 'add profile')
def add_profile(message):
    user_name = message.from_user.first_name

    if message.from_user.last_name:
        user_name = f"{user_name} {message.from_user.last_name}"
```

```
    dbmanager.update_state(user_name,
                           'main_menu',
                           time.strftime('%d/%m/%y, %X'),
                           message.chat.id)

    bot.send_message(message.chat.id,
                     'Выбери соц. сеть',
                     reply_markup=keyboards.add_profile_inline_keyboard)

    print(dbmanager.get_state(message.chat.id)[0])


@bot.message_handler(func=lambda message: message.text == 'add criteria')
def add_criteria(message):
    user_name = message.from_user.first_name

    if message.from_user.last_name:
        user_name = f"{user_name} {message.from_user.last_name}"

    dbmanager.update_state(user_name,
                           'main_menu',
                           time.strftime('%d/%m/%y, %X'),
                           message.chat.id)

    bot.send_message(message.chat.id,
                     'Добавь критерию',
                     reply_markup=keyboards.add_criteria_inline_keyboard)


@bot.callback_query_handler(func=lambda call: call.data.startswith('add_'))
def picking_inst_soc(call):

    if call.data.startswith('add_profile'):
        dbmanager.update_state('FI',
                               f"inserting{call.data[call.data.find('_')::]}",
                               time.strftime('%d/%m/%y, %X'),
                               call.message.chat.id)

        bot.send_message(call.message.chat.id,
                         f"Введи юзернейм профиля из {(call.data[call.data.find('_') + 9::]
).capitalize()}",
                         reply_markup=keyboards.cancel_keyboard)

        print(dbmanager.get_state(call.message.chat.id))
```

```
# curr_state == inserting inst uname
@bot.message_handler(func=lambda message: (dbmanager.get_state(message.chat
.id)).startswith('inserting_profile'))
def inserting_inst_uname(message):

    if message.text == 'Отмена':
        dbmanager.update_state('FI',
                               'main_menu',
                               time.strftime('%d/%m/%y, %X'),
                               message.chat.id)
        bot.send_message(message.chat.id,
                         'Главное меню',
                         reply_markup=keyboards.main_menu_keyboard)

    else:
        bot.send_message(message.chat.id,
                         'В РАЗРАБОТКЕ',
                         reply_markup=keyboards.cancel_keyboard)
        dbmanager.update_state('LOH',
                               'inserting_instagram_us',
                               time.strftime('%d/%m/%y, %X'),
                               message.chat.id)

bot.polling()
```

Keyboards.py

```
import telebot

main_menu_keyboard = telebot.types.ReplyKeyboardMarkup(True, False)
main_menu_keyboard_fetch_posts_button = telebot.types.KeyboardButton("fetch
posts")
main_menu_keyboard_get_post_button = telebot.types.KeyboardButton("get post"
)
main_menu_keyboard_add_profile_button = telebot.types.KeyboardButton("add p
rofile")
main_menu_keyboard_add_criteria_button = telebot.types.KeyboardButton("add c
riteria")
main_menu_keyboard.add(main_menu_keyboard_fetch_posts_button, main_menu
_keyboard_get_post_button)
main_menu_keyboard.add(main_menu_keyboard_add_profile_button, main_menu
_keyboard_add_criteria_button)
```

```
# add_users
add_profile_inline_keyboard = telebot.types.InlineKeyboardMarkup()
add_profile_inline_button_insta = telebot.types.InlineKeyboardButton(text='Instag
ram',
                                                    callback_data='add_profile_instagram')
add_profile_inline_button_facebook = telebot.types.InlineKeyboardButton(text='F
acebook',
                                                     callback_data='add_profile_facebook'
)
add_profile_inline_button_youtube = telebot.types.InlineKeyboardButton(text='Yo
uTube',
                                                     callback_data='add_profile_youtube')
add_profile_inline_keyboard.add(add_profile_inline_button_insta)
add_profile_inline_keyboard.add(add_profile_inline_button_facebook)
add_profile_inline_keyboard.add(add_profile_inline_button_youtube)

# add_criteria
add_criteria_inline_keyboard = telebot.types.InlineKeyboardMarkup()
add_criteria_inline_button_insta = telebot.types.InlineKeyboardButton(text='Insta
gram',
                                                    callback_data='add_criteria_instagram'
)
add_criteria_inline_button_facebook = telebot.types.InlineKeyboardButton(text='F
acebook',
                                                     callback_data='add_criteria_faceboo
k')
add_criteria_inline_button_youtube = telebot.types.InlineKeyboardButton(text='Y
ouTube',
                                                     callback_data='add_criteria_youtube')
add_criteria_inline_keyboard.add(add_criteria_inline_button_insta)
add_criteria_inline_keyboard.add(add_criteria_inline_button_facebook)
add_criteria_inline_keyboard.add(add_criteria_inline_button_youtube)

# cancel keyb
cancel_keyboard = telebot.types.ReplyKeyboardMarkup(True, False)
cancel_keyboard.add('Отмена')
```