Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

**Бакалаврська дипломна робота на тему:**

«Засіб моніторингу телеграм-каналів під час інформаційної війни»

Виконав студент 4 курсу групи 1БС-19б
спеціальності 125 «Кібербезпека»
_____Володимир ШОСТАК

Керівник: к.т.н., доцент кафедри ЗІ
_____Олеся ВОЙТОВИЧ
« 19 » _____ 2023 р.

Рецензент: к.т.н., доцент каф. ПЗ
_____Наталя БАБЮК
« 19 » _____ 2023 р.

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н, проф.
_____ Володимир ЛУЖЕЦЬКИЙ
« 19 » _____ 2023 р.

Вінниця ВНТУ – 2023 року

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти І (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 125 Кібербезпека
Освітньо-професійна програма – Безпека інформаційних і комунікаційних систем

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри ЗІ,**
**д. т. н., проф.**
**_____ Володимир ЛУЖЕЦЬКИЙ**
**«__» _____ 2023 року**

# З А В Д А Н Н Я
# НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Шостаку Володимиру Володимировичу

1. Тема роботи: «Засіб моніторингу телеграм-каналів під час інформаційної війни»,
керівник роботи: Войтович Олеся Петрівна, к. т. н., доцент кафедри ЗІ, затверджені наказом ректора ВНТУ №67 від 20.03.2023

2. Строк подання студентом роботи 19 червня 2023 р.

3. Вихідні дані до роботи:
- парсинг телеграм-каналів;
- збереження отриманої інформації;
- відслідковування першоджерела;
- виявлення користувачів, які розповсюджують інформацію;

4. Зміст текстової частини: Вступ. 1. Аналіз інформаційних джерел. 2. Розробка структури та алгоритмів роботи засобу. 3. Експериментальні дослідження. Висновки.Список використаних джерел. Додатки.

5. Перелік ілюстративного матеріалу: Алгоритми роботи засобу. Схема засобу. Результати тестування засобу.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| 1 | Войтович О. П., к.т.н., доц. каф.ЗІ | 20.03.23 | 16.04.23 |
| 2 | Войтович О. П., к.т.н., доц. каф.ЗІ | 20.05.23 | 16.06.23 |
| 3 | Войтович О. П., к.т.н., доц. каф.ЗІ | 16.03.23 | 16.06.23 |

7. Дата видачі 20 березня 2023 року.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів бакалаврської дипломної роботи | Строк виконання етапів роботи | Прим. |
|---|---|---|---|
| 1 | Аналіз завдання. Вступ | 20.03 – 26.03.2023 | |
| 2 | Аналіз інформаційних джерел за напрямком бакалаврської дипломної роботи | 27.03 – 09.04.2023 | |
| 3 | Розробка рішень, моделей, алгоритмів | 10.04 – 23.04.2023 | |
| 4 | Практична реалізація, моделювання, експериментування, результати | 24.04 – 21.05.2023 | |
| 5 | Оформлення пояснювальної записки | 22.05 – 24.05.2023 | |
| 6 | Попередній захист БДР | 25.05 – 31.05.2023 | |
| 7 | Виправлення зауважень, підготовка ілюстративного матеріалу | 01.06 – 15.06.2023 | |
| 8 | Представлення БДР до захисту, рецензування | 16.06 – 19.06.2023 | |
| 9 | Захист БДР | 20.06 – 23.06.2023 | |

Студент _____ Володимир ШОСТАК

Керівник роботи _____ Олеся ВОЙТОВИЧ

# АНОТАЦІЯ

Дипломна робота складається зі вступу, трьох розділів та висновків до них, загальних висновків, списку використаних джерел, додатків, загальним обсягом робота складає 53 сторінок, має 20 рисунків, 16 сторінок додатків. Список використаних джерел містить 30 найменувань і займає 3 сторінки.

Робота присвячена розробці засобу моніторингу телеграм-каналів під час інформаційної війни.

Обґрунтовується актуальність теми, визначається мета дослідження та завдання. Досліджується вплив фейкової інформації та важливість телеграма під час інформаційної війни. Визначаються методи аналізу телеграм-каналів. Визначені методи збирання та обробки інформації. Розроблено архітектура та блоки програми та визначено структуру бази даних. Також визначено засоби розробки, після чого було розроблено програмний засіб. В кінці роботи, проведено тестування програми, в результаті чого можна сказати що програма працює коректно.

Ключові слова: Телеграм, парсинг, моніторинг, пошук, аналіз, пошуковий двигун .

# ABSTRACT

The thesis consists of an introduction, three sections and conclusions to them, general conclusions, a list of used sources, appendices, the total volume of the work is 53 pages, has 20 figures, 16 pages of appendices. The list of used sources contains 30 names and occupies 3 pages.

The work is devoted to the development of a means of monitoring telegram channels during the information war.

The relevance of the topic is substantiated, the purpose of the research and the task are determined. The influence of fake information and the importance of the telegram during information warfare are explored. Methods of analysis of telegram channels are defined. The methods of collecting and processing information were determined. The architecture and application blocks were developed and the database structure was defined. Development tools were also determined, after which a software tool was developed. At the end of the work, the program was tested, as a result of which we can say that the program works correctly.



Keywords: Telegram, parsing, monitoring, search, analysis, search engine

# ЗМІСТ

# ВСТУП

**Актуальність.** У контексті інформаційної війни, де швидкість та точність інформаційних джерел мають велике значення, збір і аналіз інформації з публічних телеграм-каналів стає важливим завданням. У сучасному світі, де широке використання соціальних мереж та месенджерів стало невід'ємною частиною нашого повсякденного життя, телеграм-канали займають особливе місце в передачі новин, подій та поглядів [2].

Телеграм-канали є одним із ключових джерел інформації, які можуть бути використані для поширення пропаганди, маніпулювання громадською думкою та впливу на суспільні процеси. Розробка парсера телеграм-каналів стає необхідною для забезпечення доступу до цієї великої кількості інформації та її аналізу [3].

За допомогою парсера, можливо автоматично збирати потрібну інформацію та аналізувати її, а саме активність користувачів та зв’язки між каналами. Парсер виконує функцію збору інформації про повідомлення, авторів, дату публікації та інші атрибути, що дозволяє встановлювати зв'язки між повідомленнями, відстежувати активність авторів та аналізувати динаміку публікацій на каналах.

**Предметом дослідження** є парсинг телеграм-каналів, який дозволить автоматично збирати та аналізувати дані з публічних каналів.

**Об’єктом дослідження** є процес автоматизованого збору та аналізу даних у телеграм-каналах.

**Метою дослідження** покращення безпеки під час інформаційної війни шляхом використання парсера телеграм-каналів, який дозволить автоматизувати процес збору даних, забезпечити зручний доступ до необхідної інформації та відкрити можливості для подальшого аналізу зібраних даних.

Для цього виконано такі задачі:

1) проаналізувати вплив фейків та важливості телеграм-каналів під час інформаційної війни;

2) розробити алгоритмів програми;

3) розробити програмного засобу;

4) провести експериментальні дослідження.

**Практична цінність** даної бакалаврської роботи полягає в тому, що дозволяє автоматизувати процес збору даних, забезпечує зручний та швидкий доступ до необхідної інформації, а також надає можливість аналізу та використання зібраних даних. Застосування парсера дозволить забезпечити оперативну реакцію на нові події, виявляти тенденції та патерни розповсюдження інформації, а також ефективно вести контроль і вплив на інформаційне поле під час інформаційної війни.

Проміжні результати дослідження були представлені на LII Науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії [1].

# 1 АНАЛІЗ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

## 1.1 Вплив фейкової інформації під час інформаційної війни

Інформаційна війна – це сучасний феномен, що полягає у здійсненні систематичної маніпуляції і поширенні недостовірної, маніпулятивної або неповної інформації з метою впливу на громадську думку, дестабілізації суспільства, або досягнення політичних, економічних або військових цілей. Одним з поширених каналів інформаційної війни є телеграм-канали, що дозволяють анонімно поширювати і контролювати інформацію, надаючи можливість широкого впливу на аудиторію [2-3].

Інформаційні війни є ефективним знаряддям впливу на суспільство та масову свідомість. Вони можуть включати такі тактики, як [4]:

1) Вкиди (тролінг): Це стратегія, при якій надається недостовірна або провокативна інформація з метою створення хвилювання, суперечок та розбрату серед користувачів. Тролінг може бути організованим групою або окремими особами з певною метою.
2) Фейки: Це неправдива або спотворена інформація, яка поширюється з метою дезінформування та зміни громадської думки. Фейки можуть включати фальшиві новини, маніпулятивні фотографії або відео, а також спекулятивні теорії змови.
3) Маніпуляція масовою свідомістю: Інформаційні війни часто використовуються для маніпуляції масовою свідомістю шляхом поширення певних поглядів, ідеологій або політичних агенд. Це може включати використання психологічних методів, емоційні маніпуляції та зловживання соціальними мережами для залучення підписників та поширення впливу.

Шкідливість інформаційних війн через телеграм-канали [5]:

1) Розпалювання конфліктів: Використання телеграм-каналів для поширення недостовірної інформації може призвести до поглиблення

суспільних конфліктів, створення ворожнечі та зміцнення поділів у суспільстві.

2) Дезінформація та спотворення фактів: Фейкова інформація, що поширюється через телеграм-канали, може спричинити значні наслідки, такі як неправильні рішення, громадську паніку, або навіть загрозу безпеці громадян.
3) Зниження довіри до ЗМІ: Інформаційні війни, особливо через телеграм-канали, можуть призвести до загального зниження довіри громадськості до традиційних ЗМІ, оскільки вони можуть бути сприйняті як більш об'єктивні джерела інформації.
4) Загроза національній безпеці: Інформаційні війни, якщо вони спрямовані проти держави, можуть становити серйозну загрозу національній безпеці, особливо якщо вони включають дезінформацію щодо важливих подій, виборів або військових дій.

Відомий британський дослідник Ніл Фергюсон обґрунтовує ефективну функціональність мережі її потенційною вірусністю, медіативністю, інтерпретативністю, гемофільністю, щільністю (теорія «шести рукостискань» скорочується в мережевому середовищі до 3,5). Саме в мережах, позбавлених ієрархічних обмежень, шляхом «зараження» інших просуваються ідеї, настрої, оцінки. Такий спосіб впливу в суспільствах, де домінує Homo Irreti tus (людина мережева), є більш продуктивним, аніж вертикально побудована тиранія пропаганди. Соціальний інжиніринг, що сповідує мережеві принципи, вибудовує поля взаємодій, де діють різні соціокультурні стилі та практики. Урізноманітнення інформаційного простору нівелює роль вертикальних дискурсів, створює прерогативу вибору каналів та майданчиків для отримання інформації [6].

Як зазначає литовський дослідник М. П. Саулаускас, у мережевому суспільстві «претензії сучасної держави на монополію істини послідовно відкидають або толерантно ігнорують». Для інформаційного простору країни –

агресорки це означає, що вертикальна пропаганда втрачає дієвість під впливом полілогової архітектоніки мережі. Зараз спостерігаємо, як пропагандисти адаптують свій інструментарій до вимог нового комунікаційного часу та розгортають мережеві дискурсивні потоки. Горизонтальна мережева пропаганда фактично утворює тенета, у які потрапляє Homo Irretitus, що здійснює медіа споживання на нових засадах.

Отже, поширення пропагандистських наративів та дезінформації перебігає з урахуванням нових медіаумов. Архітектоніка соцмереж, юзероцентризм Веба сприяють виникненню закономірностей, які активізують рух маніпулятивних наративів, створених не інституційно, а окремими інфлюенсерами. «Камери відлуння», «бульбашкові фільтри», клікбейт – усе сприяє поширенню облудного контенту. Виникають контрфактуальні спільноти, які через постійні контентні впливи створюють особливий спосіб сприйняття дійсності учасниками таких ком'юніті. Він обумовлений намірами маніпуляторів і немає жодного відношення до реального перебігу подій, про які йдеться.

Для прикладу було розглянуто фейк, який розповсюджували наприкінці березня 2023 року. Багато телеграм-каналів розповсюджували інфографіку, на якій було зображено статистику мобілізації по областям (рис 1.1).


Рисунок 1.1 – Інфографіка, яку поширювали в телеграм-каналах

Українські ЗМІ не розповсюджували подібної інфографіки, також не вказано звідки було взято статистику, так як у відкритих джерелах не публікують загальну статистику з точною кількістю мобілізованих у кожному регіоні. Також, в інфографіці можна побачити помилку, замість «Миколаївська» написано «МІколаївська». Беручи в увагу всі ці факти, можна сказати що це один з багатьох фейків, який створений для розпалювання конфлікту між українцями, розділенням їх на «Західну» та «Східну» Україну.

Також було розглянуто результати соціологічного опитування КМІС. Після 24 лютого минулого року кількість українців, що читають телеграм-канали для отримання новин, збільшилася вдвічі й сягає 63,3%. Надмірна довірливість телеграм-каналам означає низький рівень медіаграмотності деяких користувачів, зазначають у КМІС. Про що свідчить і найпопулярніших новинний канал «Труха» з аудиторією майже три мільйони, який неодноразово публікував маніпулятивний контент [7].


Рисунок 1.2 – Інфографіка опитування

Як можна побачити в інфографіці, 8% людей повністю погоджуються, а 37.2% скоріше погоджуються. Звідси можна зробити висновок, що майже половина опитуваних довіряє телеграм-каналам.

Також можна побачити теми, які цікавлять людей на даний момент (рис. 1.3).

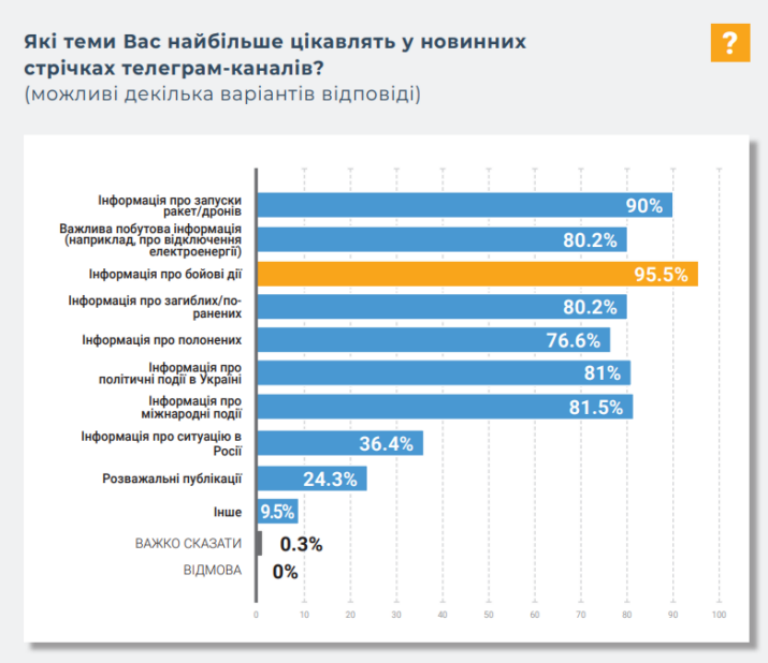



Рисунок 1.3 – Інфографіка опитування

Як можна побачити, більшість людей переглядає інформацію про бойові дії. Беручи в увагу минулу інфографіку, за допомогою телеграм-каналів, можна вводити в оману людей, які цікавляться бойовими діями. Для цього достатньо на 1 фейкову новину, робити 10 правдивих, в результаті чого в людей може виникнути почуття, що влада їх обманює, і все погано.

Отже інформаційні війни, включаючи вкиди, фейки та маніпуляцію масовою свідомістю, мають значний потенціал для шкоди під час інформаційних конфліктів. Через телеграм-канали, ці шкідливі впливи можуть бути поширені швидко і ефективно, ставлячи під загрозу стабільність суспільства, національну безпеку та довіру громадськості. Розуміння цих методів та розвиток навичок критичного мислення є важливими для захисту від негативних наслідків інформаційних війн.

## 1.2 Важливість телеграма під час інформаційної війни

У сучасному світі інформаційної війни, соціальні мережі та месенджери займають особливе місце у поширенні і впливі на інформацію. Серед них Телеграм відіграє важливу роль та має значний вплив на громадську думку та

суспільні процеси [8]. Ось декілька аспектів, що підкреслюють важливість Телеграма в інформаційній війні:

1) Анонімність та шифрування: Телеграм надає користувачам можливість залишатися анонімними та забезпечує ефективне шифрування повідомлень. Це робить його популярним серед активістів, опозиційних груп та тих, хто прагне поширювати альтернативні або контроверсійні погляди. Через анонімність і шифрування, Телеграм стає ідеальним майданчиком для поширення пропаганди, маніпулювання громадською думкою та ведення інформаційної війни.
2) Розповсюдження швидкості: Телеграм має мільйони активних користувачів по всьому світу та забезпечує миттєву передачу повідомлень. Це дозволяє інформації поширюватися зі швидкістю світла, створюючи потужні потоки новин та інформаційних повідомлень. Уміння швидко реагувати на актуальні події та поширювати свої погляди дозволяє впливати на громадську думку і формувати свої набори істин.
3) Створення спільнот та каналів: Телеграм дозволяє створювати публічні та приватні спільноти, а також канали, які можуть об'єднувати велику кількість користувачів. Це створює унікальну можливість для організації груп, які спільно працюють над поширенням певних ідей, поглядів або навіть дезінформації. Канали можуть набувати значної популярності та стати впливовими джерелами інформації, що розповсюджуються зі значною швидкістю.
4) Можливості впливу: Телеграм надає можливість вести активну комунікацію з аудиторією, розповсюджувати інформацію, формувати думки та впливати на погляди людей. Це стає важливим інструментом в інформаційній війні, де головною метою є контроль над інформаційним полем і вплив на громадську думку. Телеграм дозволяє

створювати та поширювати пропагандистські матеріали, маніпулювати фактами та ставленням людей до певних подій чи питань.

Також у телеграмі є боти – програми, які автоматично надають інформацію на запит. Наприклад, за допомогою ботів можна стежити за квитками на поїзди, інформувати Службу безпеки України про переміщення російських військ абощо. З ботами теж варто бути обережними, бо часто росіяни створюють аналог бота, щоб збирати персональні дані людей чи поширювати дезінформацію [9].

Якщо розглядувати більше чати, то є два умовні типи – чат, де всі люди особисто знайомі, або чат за інтересами, – наприклад, Чат ВНТУ, – де не всі один одного знають. У першому випадку небезпека в тому, що людина більше схильна довіряти повідомленням, які походять від знайомих.

У чатах, де зібрались особисто не знайомі люди, ніколи не відомо, хто з них – студент, а хто – майор КГБшнік. Росіяни часто долучаються до таких чатів й іноді організовують інформаційні диверсії, намагаються маніпулювати людьми, поширюють фейки.

Яскравий приклад диверсії в недалекому минулому – випадок із Новими Санжарами. Це була успішна російська операція. Коли в Китаї почалась епідемія коронавірусної хвороби, українських громадян евакуювали з Уханя, де був осередок зарази, і планували розмістити на самоізоляцію в Нових Санжарах на Полтавщині. Місцеві намагались їх не пустити, кидали каміння, протестували. Є свідчення, що їх підбурювали саме в чатах у месенджерах, і робили це користувачі, які вдавали місцевих.

За даними опитування Internews, 60% респондентів віддають перевагу телеграму як джерелу новин. За даними компанії Telemetrio, яка надає аналітику про телеграм-канали, на українську аудиторію орієнтовані майже 6 тисяч каналів. Інша компанія, яка працює в цій же ніші, – TGStat, підрахувала, що на українську аудиторію націлені 51 тисяча каналів у телеграмі. Згідно з рейтингами цих компаній, кількість читачів сотні найбільших каналів в українському сегменті телеграма на сьогодні коливається від 2,7 мільйона читачів до 200 тисяч.

«Детектор медіа» уже писав, що Telemetrio і особливо TGStat не публікують методологію, як відносять телеграм-канали до різних сегментів. Попри це, навіть ці дані ілюструють популярність телеграма в Україні як джерела новин, яке після повномасштабного російського вторгнення стало головним джерелом інформації для більшості українців [10].

Небажання засновників телеграма реагувати на дезінформацію, яка поширюється у месенджері, який вони створили, призвело до того, що туди мігрували проросійські та російські пропагандисти, чию свободу поширювати неправдиву інформацію обмежили адміністрації Twitter, Facebook чи YouTube.

Державні та недержавні, українські і закордонні ініціативи неодноразово звертали увагу на телеграм-канали, які поширюють російську пропаганду. Наприклад, Центр стратегічних комунікацій та інформаційної безпеки при Міністерстві культури та інформаційної політики оприлюднив перелік зі 100 каналів, які мімікрують під українські. Нерідко такі телеграм-канали називають частиною російської мережі дезінформації.

| № п/п | Назва псевдоукраїнського Телеграм-каналу, який працює в інтересах рф |
|---|---|
| 1 | Крокодил |
| 2 | ШептунUкраина Война |
| 3 | Легитимный |
| 4 | Анатолий Шарий |
| 5 | Резидент |
| 7 | #МОНТЯН! |
| 8 | ЗеРада |
| 9 | Open Ukraine \| Открытая Украина |
| 10 | UKR LEAKS |
| 12 | Сплетница |
| 14 | Картель |
| 15 | ОЛЬГА ШАРИЙ |
| 17 | Украина.ру |
| 18 | MediaKiller |
| 20 | ТелеДНО |
| 21 | Женщина с косой |
| 22 | Типичная Одесса |
| 23 | Война с фейками |
| 24 | Главное в Херсоне |
| 27 | Чёрный Квартал |
| 28 | НачШтабу |
| 29 | Бунтарь |
| 30 | наглый |
| 31 | Украина. Спецоперация. Мониторинг |
| 32 | Я ♡ Краматорск |
| 33 | Одесский фраер |
| 34 | Запрещённая Украина |
| 35 | Отряд Ковпака |
| 36 | Шкварка News |
| 38 | Кріт СБУ |
| 39 | шпбуля UA |
| 40 | Администрация города Мелитополя |
| 41 | Лисичанск. Северодонецк. Рубежное. |
| 42 | Dirty Hatty \| Игорь Гомольский |
| 43 | Партия Шария |
| 44 | Черговий ООС |
| 45 | Нетипичное запорожье |
| 46 | Новости Херсонщины |
| 47 | Главное в Геническе |
| 48 | Херсон Life Новости |
| 49 | Украинский формат |
| 50 | Главное в Мелитополе |
| 51 | Тень на плетень |
| 52 | Нетипичное Запорожье |
| 53 | Херсон live |
| 54 | Кривой Рог онлайн |

| № п/п | Назва псевдоукраїнського Телеграм-каналу, який працює в інтересах рф |
|---|---|
| 55 | Николаев live |
| 56 | Тремпель Харьков |
| 57 | ХтоШо |
| 58 | 93 бригада "Холодний Яр" \| ОБРАТНАЯ СТОРОНА |
| 59 | Главное в Чернобаевке |
| 60 | Новый Мелитополь |
| 61 | Military photographer |
| 62 | ЗЕландия |
| 63 | ХЕРСОН сегодня |
| 64 | Херсонский Вестник |
| 66 | Главное в Каховке и Новой Каховке |
| 67 | Главное в Скадовске |
| 68 | Чернигов |
| 69 | Сумы |
| 70 | Глухов |
| 71 | Бровары |
| 72 | Шостка |
| 73 | Васильков |
| 74 | Новгород-Северский |
| 75 | Нежин |
| 76 | Обухов |
| 77 | Белая Церковь |
| 78 | Овруч |
| 79 | Ромны |
| 80 | Носовка |
| 81 | Славутич |
| 82 | Бахмач |
| 83 | Прилуки |
| 84 | Ирпень |
| 85 | Березань |
| 86 | Ахтырка |
| 87 | Конотоп |
| 88 | Бердянск ZaVtra |
| 89 | Новый Мелитополь |
| 90 | Южный спецдарм |
| 91 | Энергодарский связной |
| 92 | Энергодар Тудей |
| 93 | Токмак Сегодня |
| 94 | Главное в Пологах |
| 95 | Главное в Энергодаре |
| 96 | Главное в Бердянске |
| 97 | Харьковские антифашисты |
| 98 | Друзья Алексея Селиванова |
| 99 | Администрация города Васильевка |
| 100 | Нетленка |



Рисунок 1.4 – Список телеграм-каналів, які працюють в інтереси РФ

Проросійські та окупаційні телеграм-канали часто використовують у своїх дописах токсичні слова, спрямовані на приниження України, звинувачення

української влади в злочинних діях, а також на легітимізацію дій Росії та окупаційних адміністрацій. Наприклад, низка телеграм-каналів майже не вживає слово «влада» щодо українських політиків та чиновників, натомість використовують слово «режим» – «київський режим», «український режим», «режим Зеленського» тощо. Схожа історія зі словом «війна», замість якого найчастіше вживають російський термін «спецоперація»; також проросійські телеграм-канали послуговуються словами «українська криза» чи «український конфлікт».

Під час перевірки контенту аналітики й аналітикині також фіксували фейки та меседжі російської пропаганди, які поширювали телеграм-канали. Ознайомлення із порядком денним, який просувають телеграм-канали, є одним із головних критеріїв для того, щоб віднести будь-яке медіа і канал у телеграмі до проросійських [12].

Усі ці фактори підкреслюють важливість Телеграма в інформаційній війні. Вміння розуміти та аналізувати цю платформу, її вплив та механізми поширення інформації стають критичними для ефективного ведення інформаційної війни та захисту від маніпуляцій.

Відслідковування різних телеграм-каналів є важливим кроком у розумінні інформаційного пейзажу. Аналіз тематики, змісту та взаємодії з аудиторією допомагає визначити мету та спрямованість каналу. Це дає змогу виявити джерела недостовірної або маніпулятивної інформації, а також встановити популярні теми та проблематику, яку активно обговорюють [13].

Під час відслідковування телеграм-каналів, важливо враховувати такі аспекти:

1) Виявлення ключових тематик: Аналіз телеграм-каналів допомагає визначити, які теми є актуальними для каналів, та які погляди та ідеї вони пропагують. Це може включати політичні дискусії, суспільні проблеми, наукові дослідження, релігійні аспекти або інші тематики.

2) Аналіз повідомлень та змісту: Вивчення повідомлень, публікацій та змісту телеграм-каналів дає змогу зрозуміти, яку інформацію вони поширюють та якими способами це робиться. Детальний аналіз може допомогти виявити недостовірні, маніпулятивні або фейкові повідомлення, що розповсюджуються через ці канали.
3) Взаємодія з аудиторією: Вивчення взаємодії між телеграм-каналами та їх аудиторією є важливим елементом аналізу. Аналіз коментарів, реакцій, спільних згадок та обміну інформацією може допомогти встановити зв'язки між каналами, а також визначити, які канали мають більший вплив на громадську думку.
4) Розпізнавання активних акторів: Виявлення основних авторів або адміністраторів каналів, які мають значний вплив, може допомогти встановити зв'язки між різними каналами. Це може розкрити можливі мережі впливу та показати, які особи стоють за поширенням певних повідомлень та інформації.

Пошук перешоджерела та зв'язків:

Пошук перешоджерела є важливим етапом в аналізі телеграм-каналів. Виявлення джерела інформації, яке використовується каналом, допомагає з'ясувати походження інформації та можливі зв'язки з іншими джерелами. Аналіз спільних авторів або адміністраторів, реакцій та ретрансляції інформації між різними каналами розкриває можливі мережі впливу та способи поширення інформації.

Детальніше, важливими елементами пошуку перешоджерела та зв'язків є:

1) Аналіз посилань та джерел: Вивчення посилань, які використовуються телеграм-каналами, та джерел, з яких вони отримують інформацію, дозволяє з'ясувати, звідки походить поширювана інформація. Це може включати посилання на новинні портали, блоги, інформаційні агентства або інші джерела.

2) Виявлення спільних авторів або адміністраторів: Аналіз телеграм-каналів може допомогти виявити спільних авторів або адміністраторів, які мають зв'язки з іншими каналами. Це дає змогу розкрити можливі мережі впливу та встановити способи спільного поширення інформації.
3) Моніторинг обміну інформацією: Слідкування за ретрансляцією повідомлень та інформації між різними телеграм-каналами може допомогти встановити зв'язки та виявити способи поширення фейкової, маніпулятивної або недостовірної інформації.

Правильне відслідковування та пошук перешоджерела та зв'язків між телеграм-каналами дозволяє отримати більш повне розуміння інформаційного середовища та виявити потенційні загрози та маніпуляції.

### 1.3 Методи аналізу телеграм-каналів

Аналіз телеграм-каналів став важливим завданням в сучасному інформаційному середовищі. Зростаюча популярність та вплив телеграм-каналів призводить до необхідності вивчення їхнього змісту, взаємодії та впливу на громадську думку. Відслідковування та аналіз телеграм-каналів допомагають розкрити фейкові новини, маніпулятивну інформацію та шкідливі впливи, що поширюються через цей канал комунікації [16-17].

Нижче наведено кілька відомих методів аналізу телеграм-каналів, які дозволяють розкрити зв'язки, патерни та вплив, які можуть мати суттєвий вплив на сприйняття та розповсюдження інформації:

1) Словниковий аналіз: Цей метод включає аналіз текстового змісту повідомлень, що публікуються в телеграм-каналах. Словниковий аналіз вимагає побудови словників або списків ключових слів та фраз, які пов'язані з певними темами, настроями чи метою каналу. Це допомагає відстежувати та категоризувати інформацію, що поширюється, і виявляти специфічні теми або зміни в настроях.

2) Аналіз мережі: Цей метод полягає в вивченні зв'язків між різними телеграм-каналами. За допомогою графових алгоритмів можна визначити ключові актори, спільні авторства, популярність та взаємодію між каналами. Аналіз мережі допомагає встановити потенційні мережі впливу, групи зі спільними цілями або способи поширення інформації.
3) Соціальний аналіз: Цей метод передбачає вивчення взаємодії між телеграм-каналами та їх аудиторією. Аналіз коментарів, реакцій та спільних згадок може розкрити, які канали мають більший вплив на громадську думку, які теми стають віральними та які групи людей активно взаємодіють.
4) Географічний аналіз: Цей метод включає вивчення географічного поширення телеграм-каналів та їх аудиторії. Аналіз місцезнаходження та мови публікацій може дати уявлення про регіональну спрямованість каналу, розповсюдження пропаганди або міжнародні зв'язки.
5) Виявлення патернів: Цей метод полягає в ідентифікації патернів або шаблонів поведінки та активності каналів. Це можуть бути певні способи ретрансляції повідомлень, час публікацій, використання хештегів або інші характеристики, які можуть бути спільними для декількох каналів. Виявлення патернів допомагає зрозуміти тактику та стратегію каналів, а також виявляти можливі маніпуляції.

Ці методи аналізу телеграм-каналів можуть бути використані окремо або в поєднанні для отримання більш повного та розгорнутого розуміння інформаційного середовища та виявлення можливих загроз або маніпуляцій. Важливо застосовувати ці методи з дотриманням етичних принципів та з врахуванням конфіденційності даних.

## 1.4 Аналіз Telegram API, бібліотеки Telethon, системи зберігання та управління даними

Було проведено детальний аналіз Telegram API – набору функцій та протоколів, які надають можливість розробникам взаємодіяти з Телеграмом через програмні додатки. Також розглянемо бібліотеку Telethon, яка є зручним інструментом для роботи з Telegram API в мові програмування Python.

**Telethon** – це Python-бібліотека, яка надає зручний інтерфейс для роботи з Telegram API. Вона спрощує використання Telegram API в мові програмування Python та надає розробникам широкі можливості для створення програм, що взаємодіють з Телеграмом. Розглянемо переваги використання Telethon та його функціональні можливості.

1) Аутентифікація та безпека: Telegram API використовує протокол MTProto (Mobile Protocol), який забезпечує безпеку підключення та обміну даними між клієнтом та сервером. Крім того, Telegram пропонує двофакторну аутентифікацію, яка забезпечує додатковий рівень безпеки для користувачів.
2) Функціонал повідомлень: Telegram API дозволяє відправляти та отримувати повідомлення в різних форматах, таких як текстові повідомлення, фотографії, відео, аудіо, стікери та документи. Ви можете надсилати повідомлення одержувачам або використовувати спеціальні методи для розсилки повідомлень групам або каналам.
3) Керування групами та каналами: За допомогою Telegram API можна створювати та керувати групами та каналами. Можна створювати нові групи, додавати та видаляти учасників, назначати адміністраторів, налаштовувати права доступу, публікувати повідомлення та виконувати інші дії пов'язані з управлінням групами та каналами.
4) Робота з медіафайлами: Telegram API підтримує роботу з різними типами медіафайлів. Ви можете відправляти фотографії, відео, аудіо та документи, а також отримувати їх з повідомлень. Крім того, Telegram

надає можливість зберігати файли на їх серверах та отримувати посилання на них для подальшого використання.

5) Інформація про користувачів: Telegram API дозволяє отримувати інформацію про користувачів, таку як ім'я, ідентифікатор, фотографію профілю тощо. Дозволяє використовувати цю інформацію для взаємодії з користувачами та налаштування програм на основі цих даних.

Система зберігання та управління даними є важливим компонентом проекту, оскільки вона забезпечує ефективну обробку, доступ і збереження зібраної інформації. Ми розглянемо процес структурування даних, вибір системи керування базами даних (СКБД), збереження даних, управління даними, індексування та пошук даних, а також масштабуваність та оптимізацію.

1) Структурування даних: Перш за все, важливо розробити ефективну структуру для збереження даних з телеграм-каналів. Це означає визначення таблиць, полів та зв'язків між ними. Наприклад, ви можете мати таблиці для збереження повідомлень, авторів, дат, тегів тощо. Правильно спланована структура даних спрощує подальшу обробку і доступ до інформації [23].
2) Вибір системи керування базами даних (СКБД): Один з ключових кроків – це вибір підходящої СКБД для вашого парсера. На ринку існує кілька популярних СКБД, які варто розглянути:
   – MySQL: MySQL є однією з найбільш поширених відкритих СКБД. Вона володіє широким спектром можливостей, надійності та продуктивності.
   – PostgreSQL: PostgreSQL є потужною реляційною СКБД з підтримкою розширень, транзакційною безпекою та підтримкою географічних даних.
   – MongoDB: MongoDB – це документ-орієнтована СКБД, яка забезпечує гнучкість і швидкодію. Вона особливо корисна, якщо ваші дані мають неструктурований формат.

3) Збереження даних: Після вибору СКБД ви повинні створити механізми для збереження даних. Це включає створення таблиць, визначення полів, обмежень, індексів тощо. Правильно налаштована база даних допомагає забезпечити швидкий доступ до інформації та запобігти втраті даних.
4) Управління даними: Система повинна мати механізми для додавання нових записів, оновлення існуючих даних та видалення застарілих записів. Важливо також забезпечити безпеку даних, захищаючи їх від несанкціонованого доступу та резервне копіювання для запобігання втраті даних у разі аварійної ситуації.
5) Індексування та пошук даних: Для оптимізації доступу до даних важливо створити відповідні індекси. Індекси допомагають прискорити пошук та фільтрацію даних за різними критеріями. Розроблення ефективних запитів до бази даних дозволяє отримати необхідну інформацію зі збережених записів.
6) Масштабуваність та оптимізація: У процесі розвитку парсера можуть з'являтися нові вимоги до обробки даних або зростати їх обсяг. Важливо розглянути можливості масштабування системи зберігання та управління даними. Це може включати горизонтальне масштабування бази даних, розподілені системи або використання кешування для підвищення продуктивності.

Правильне виконання всіх пунктів, розглянутих у цьому підрозділі, допоможе забезпечити ефективну та надійну систему зберігання та управління даними для парсера телеграм-каналів. Це включає структурування даних, вибір підходящої СКБД, налаштування збереження даних, розробку механізмів управління даними, оптимізацію пошуку та можливість масштабування системи. Це забезпечує ефективну обробку та доступ до зібраної інформації, підвищує продуктивність та забезпечує надійність даних.

## 1.5 Фільтрація, відбір інформації та аналіз користувачів телеграм-чатів

Під час аналізу телеграм-каналів і отримання інформації з них, фільтрація та відбір необхідної інформації є важливим етапом. Цей підрозділ присвячений розгляду різних методів і стратегій фільтрації, які допоможуть зосередитися на суттєвій інформації та оптимізувати процес аналізу.

1) Використання ключових слів та фраз: Один з найефективніших способів фільтрації інформації – використання ключових слів або фраз, які є важливими для дослідження або інтересів. Перегляд заголовків, описи та текст повідомлень у пошуку цих ключових слів. Це дозволить відфільтрувати повідомлення, що не містять ці ключові слова, та зосередитися на тих, які стосуються потрібної тематики.
2) Використання категорій та тегів: Інший ефективний підхід полягає використанні категорій або тегів. Якщо канали, які аналізуються, мають систему категоризації або тегування, можна використовувати ці мітки для відбору конкретних типів повідомлень.
3) Фільтрація за авторами та джерелами: Фільтрація за авторами або джерелами може бути також ефективним підходом. Якщо відомо, що певні автори або джерела надають цікаву та достовірну інформацію, можна сконцентруватися на їх повідомленнях. Щоб відсіяти повідомлення від інших авторів або джерел, які не є пріоритетними, можна використовувати фільтри.
4) Географічна фільтрація: Якщо потрібна інформація про певний регіон чи місцеві події, використання географічної фільтрації може бути корисним. Деякі повідомлення можуть містити інформацію про місцезнаходження, що дозволяє відфільтрувати повідомлення, що стосуються конкретного регіону чи місцевості.
5) Використання комбінованих фільтрів: Оптимальним підходом є використання комбінації різних фільтрів для досягнення найкращих

результатів. Комбінуйте ключові слова, категорії, авторів та географічні фільтри, щоб точно визначити необхідну інформацію.

Отже фільтрація та відбір інформації є необхідним етапом аналізу телеграм-каналів. Використання ключових слів, категорій, авторів та географічної фільтрації, допомагає відсіяти зайву інформацію та зосередитися на потрібних повідомленнях. Важливо експериментувати з різними фільтрами та їх комбінаціями, щоб знайти оптимальний підхід для конкретного дослідження.

Аналіз користувачів телеграм-чатів є важливим аспектом вивчення динаміки поширення інформації та виявлення джерел розповсюдження пропаганди та фейкової інформації. У цьому підрозділі ми розглянемо, з яких каналів користувачі пересилають інформацію та який вплив вони мають на інших користувачів.

1) Вивчення пересилання інформації: Аналізуйте, з яких каналів користувачі пересилають інформацію в телеграм-чатах. Слідкуйте за посиланнями та джерелами, які вони активно використовують. Це допоможе вам ідентифікувати ключові джерела інформації та розуміти, як вони впливають на користувачів.
2) Аналіз впливу на інших користувачів: Слідкуйте за реакціями та взаємодією інших користувачів на пересилану інформацію. Аналізуйте коментарі, лайки, репости та інші прояви інтересу або взаємодії. Це допоможе вам оцінити вплив цих користувачів на інших учасників чату та виявити популярні джерела або повідомлення, які отримують найбільшу увагу та поширення.
3) Виявлення джерел пропаганди та фейкової інформації: Уважно стежте за поширенням пропагандистських повідомлень та фейкової інформації в телеграм-чатах. Ретельно аналізуйте посилання та джерела, які цю інформацію поширюють. Виявлення таких джерел допоможе вам усунути недостовірну інформацію та зберегти достовірність інформаційного середовища.

4) Виявлення впливових користувачів: Аналізуйте активність та впливовість користувачів телеграм-чатів. Визначте, які користувачі мають значний вплив та як їхні дії впливають на поведінку і реакції інших учасників чату. Це може надати вам уявлення про ключових учасників та лідерів думок у вашій аудиторії та допомогти вам зрозуміти, які джерела впливають на формування думок і настроїв учасників.

Аналіз користувачів телеграм-чатів дозволяє виявити джерела розповсюдження пропаганди та фейкової інформації, що забезпечує здатність втручатися у процеси інформаційного обміну та збереження достовірності інформації в спільноті.

### 1.6 Постановка завдання

Мета бакалаврської роботи  полягає в розробці та реалізації парсера телеграм-каналів з метою виявлення, аналізу та моніторингу інформації, що поширюється через ці канали під час інформаційної війни. Головними завданнями бакалаврської роботи є наступне:

1) Розробка алгоритму парсингу телеграм-каналів: Потрібно розробити ефективний алгоритм, який здатний отримувати та аналізувати дані з різних телеграм-каналів. Алгоритм повинен забезпечувати збір повідомлень, медіа-контенту, метаданих та інших відомостей, що дозволить провести детальний аналіз зібраної інформації.
2) Аналіз взаємодії та зв'язків між каналами: Потрібно розробити методи та алгоритми для виявлення та аналізу взаємодії між різними телеграм-каналами. Це допоможе встановити мережу впливу, виявити спільні авторства, визначити ключових акторів та впливові групи. Аналіз зв'язків між каналами дозволить отримати глибше розуміння динаміки поширення інформації та виявити можливі маніпуляції.

3) Розробка системи моніторингу та аналізу: Завданням є розробка системи моніторингу, яка буде автоматично відстежувати нові повідомлення та оновлення на телеграм-каналах, використовуючи розроблений парсер. Система повинна забезпечувати постійний аналіз інформації, виявлення негативного впливу, а також можливість генерації звітів та візуалізації отриманих даних.
4) Експериментальне дослідження та оцінка ефективності: Завершальним етапом бакалаврської роботи є проведення експериментального дослідження та оцінка ефективності розробленої системи парсингу телеграм-каналів. Це включає збір тестових даних, проведення аналізу та порівняння результатів, а також оцінку точності та швидкодії системи.

В результаті розробки, буде отримано інструмент, який може бути використаний для ефективного виявлення, аналізу та моніторингу інформації, що поширюється через телеграм-канали під час інформаційної війни. Виконання поставлених завдань дозволить покращити здатність розрізняти правдиву інформацію від фейкової, виявляти маніпулятивну інформацію та реагувати на неї швидко та ефективно.

# 2 РОЗРОБКА МЕТОДІВ

## 2.1 Методика збирання, обробки та перевірки інформації

У контексті інформаційної війни, де швидкість та точність інформаційних джерел мають велике значення, збір і аналіз інформації з публічних телеграм-каналів стає важливим завданням. Для ефективного ведення інформаційної війни необхідно мати інструмент, який дозволить автоматично збирати та аналізувати дані з таких каналів. Для цього буде розроблено парсер телеграм-каналів, який дозволить здійснювати автоматичний збір та узагальнення інформації з публічних каналів під час інформаційної війни.

Які дані необхідно збирати:

1) Текст повідомлення: Усередині кожного повідомлення міститься текстовий контент, що містить докладнішу інформацію про події, новини або коментарі, які публікуються на каналі.
2) Автор повідомлення: Інформація про автора повідомлення, включаючи нікнейм або ім'я, дозволить розпізнавати джерела інформації та відстежувати активність конкретних авторів на каналі.
3) Дата та час публікації повідомлення: Кожне повідомлення має мітку з датою та часом його публікації, що дозволить відстежувати хронологію подій та аналізувати динаміку публікацій на каналі.
4) Звідки переслано або кому відповідь: Для отримання більш повної карти поширення інформації, важливо збирати дані про те, з якого джерела було переслано повідомлення або кому була надана відповідь. Ця інформація допоможе виявити ланцюжки розповсюдження та зв'язки між різними акторами.

Яким чином збирати дані:

1) Налаштування програми для взаємодії з API телеграм: Розробка програмного інтерфейсу (API), який забезпечить комунікацію з телеграм платформою та доступ до публічних каналів.

2) Отримання доступу до публічних каналів за допомогою API ключа: Отримання API ключа від телеграм, який дасть можливість отримувати доступ до публічних каналів із забезпеченням автентифікації.
3) Проходження по повідомленням в каналах: Розробка парсера, який буде проходити по повідомленнях каналів і здійснювати збір необхідних даних, таких як заголовок, текст, автор, дату та час публікації, а також кількість переглядів та коментарів.
4) Збереження зібраних даних: Збереження зібраних даних у відповідному форматі, наприклад, у базі даних або у файловій системі, для подальшого використання та аналізу.

Розроблений парсер телеграм-каналів є потужним інструментом для збору та аналізу інформації з публічних каналів під час інформаційної війни. Він дозволяє автоматизувати процес збору даних, забезпечує зручний доступ до необхідної інформації та відкриває можливості для подальшого аналізу та використання зібраних даних. Розроблений парсер є значним внеском у сферу інформаційної безпеки та допомагає вести ефективну інформаційну війну, забезпечуючи швидкий та достовірний доступ до актуальної інформації з телеграм-каналів.

## 2.2 Розробка архітектури програми

У цьому підрозділі описується архітектура програмного забезпечення парсера телеграм-каналів. Нижче наведено опис загальної архітектури та ключові аспекти розробки.

1) Огляд загальної архітектури програми:

Архітектура програми складається з кількох основних компонентів, які взаємодіють між собою. Ці компоненти включають:

– Клієнтський інтерфейс: Забезпечує взаємодію з користувачем та дозволяє встановлювати параметри парсингу, відслідковувати канали тощо.

- Модуль отримання даних з Telegram API: Відповідає за взаємодію з Telegram API, включаючи аутентифікацію, отримання списку каналів та повідомлень з них.
- Модуль парсингу даних: Здійснює аналіз отриманих повідомлень, виявлення ключових слів, класифікацію та фільтрацію інформації.
- Модуль збереження даних: Забезпечує збереження оброблених даних у відповідному форматі для подальшого використання або аналізу.
- Модуль візуалізації результатів: Відповідає за візуалізацію оброблених даних у зручному для користувача форматі, наприклад, у вигляді графіків або діаграм.

2) Архітектурні шари:

Парсер телеграм-каналів можна розділити на кілька архітектурних шарів:

- Інтерфейсний шар: Включає всі елементи, необхідні для взаємодії з користувачем, забезпечує введення параметрів та відображення результатів.
- Логічний шар додатку: Містить бізнес-логіку програми, включаючи обробку даних з Telegram API, аналіз та фільтрацію повідомлень.
- Шар доступу до даних: Відповідає за збереження та отримання даних з бази даних або інших систем збереження.
- Цей розподіл на шари дозволяє забезпечити модульність та розширюваність програми.

3) Взаємодія з Telegram API:

Для отримання даних з телеграм-каналів програма буде взаємодіяти з Telegram API. Вона буде використовувати відповідні методи та запити, щоб отримувати необхідну інформацію. Для взаємодії з Telegram API буде використовуватись telethon.

4) Модуль парсингу даних:

Цей модуль відповідає за аналіз отриманих повідомлень з каналів. Він використовує різні алгоритми та методи, такі як обробка тексту, виявлення ключових слів, класифікація повідомлень тощо.

5) Зберігання даних:

Отримані та оброблені дані можна зберігати у базі даних, файловій системі або інших системах збереження. Вибір методу збереження залежить від потреб вашої програми та обсягу даних. Наприклад, ви можете використовувати реляційну базу даних, таку як MySQL або PostgreSQL, для збереження структурованих даних, або NoSQL базу даних, таку як MongoDB, для збереження неструктурованих даних.

## 2.3 Розробка блоків програми

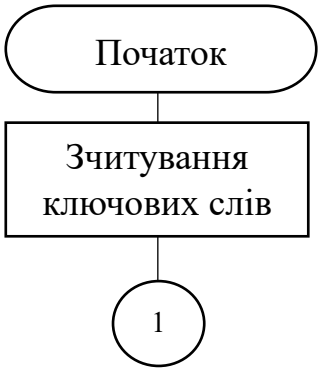
У цьому підрозділі ми розглянемо розбиття програми на окремі блоки та опис функціональності кожного блоку. Це дозволить нам зрозуміти роботу програми більш детально.

### 2.3.1 Блок зчитування нової інформації

Блок зчитування нової інформації є важливою складовою частиною програми і відповідає за отримання свіжих даних з телеграм-каналів. Цей блок можна розглядати як постійний процес, який виконується у фоновому режимі.

Після отримання нових повідомлень з каналів, цей блок виконує обробку і аналіз цих повідомлень, а саме виявлення ключових слів. Якщо було виявлено ключове слово в повідомленні, це повідомлення буде надіслано користувачу. Оброблені дані будуть збережені у відповідній базі даних за допомогою відповідних методів.

Як саме буде працювати цей блок, можна побачити на рисунку 2.1.



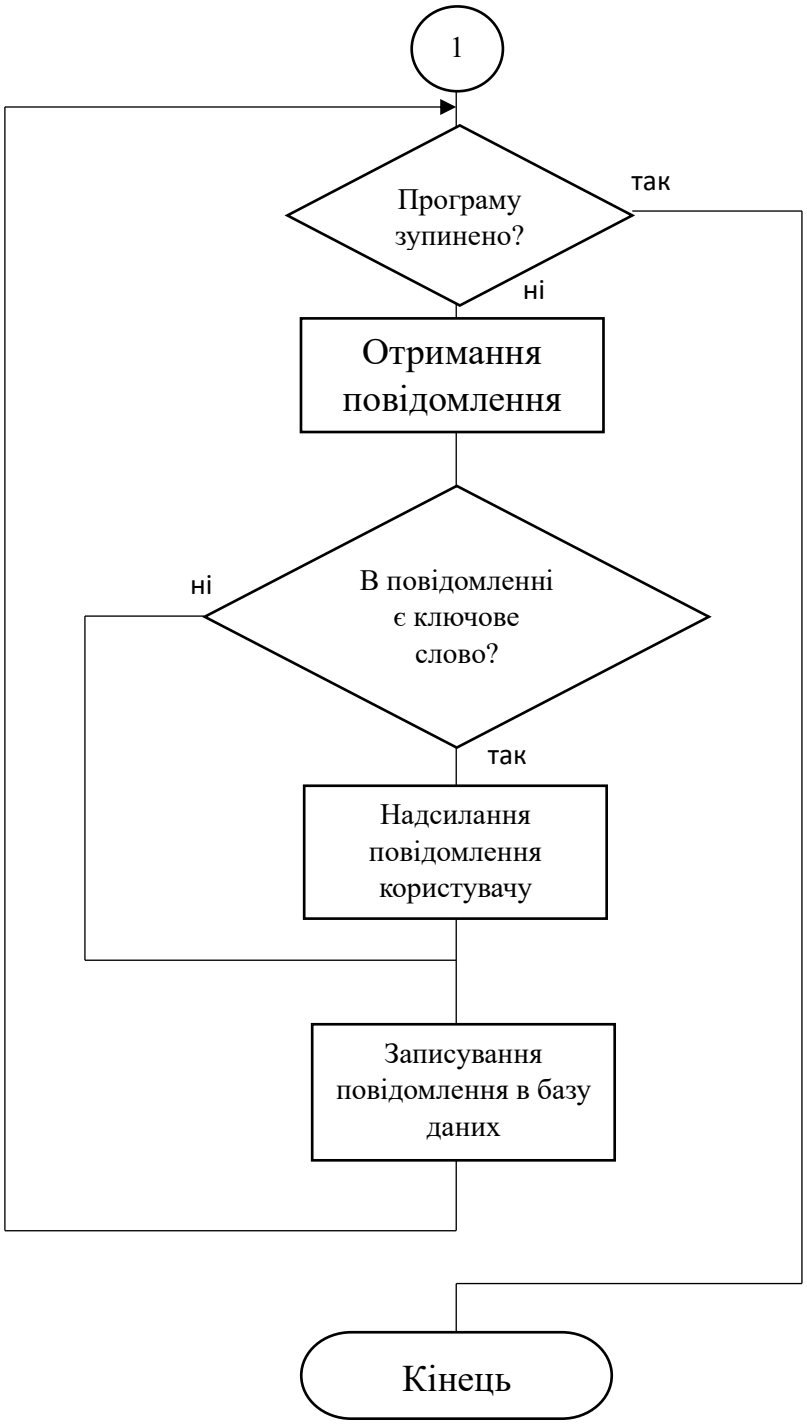



Рисунок 2.1 – Схема роботи блоку зчитування нової інформації

### 2.3.2 Блок входу в канал або чат

Блок входу в калал або чат є важливою частиною програми, яка відповідає за отримання інформації при першому вході в канал або чат у Telegram. Цей блок дозволить отримати відповідні дані з них, а саме: список користувачів та доступна інформація про них і останні повідомлення в каналі або чаті. Вся інформація буде записана в базу даних, де її можна буде переглянути.

Як саме буде працювати цей блок, можна побачити на рисунку 2.2.


Рисунок 2.2 – Схема роботи блоку входу в канал або чат

Після вибору канала або чата, програма починає зчитувати останні повідомлення. Якщо було обрано чат, а не канал, програма перед цим зчитує користувачів та інформацію про них, після чого перевіряє, чи існують вони в базі даних, якщо їх там немає, то додає в базу даних.

### 2.3.3 Блок пошуку повідомлень з ключовим словом в базі даних

Блок пошуку повідомлень з ключовим словом в базі даних є важливою частиною програми, яка дозволяє користувачеві здійснювати пошук специфічної

інформації за допомогою ключових слів або фраз. Цей блок дозволяє швидко знайти повідомлення, що містять вказані ключові слова, і виконується у контексті раніше збереженої бази даних.

Як саме буде працювати цей блок, можна побачити на рисунку 2.3.

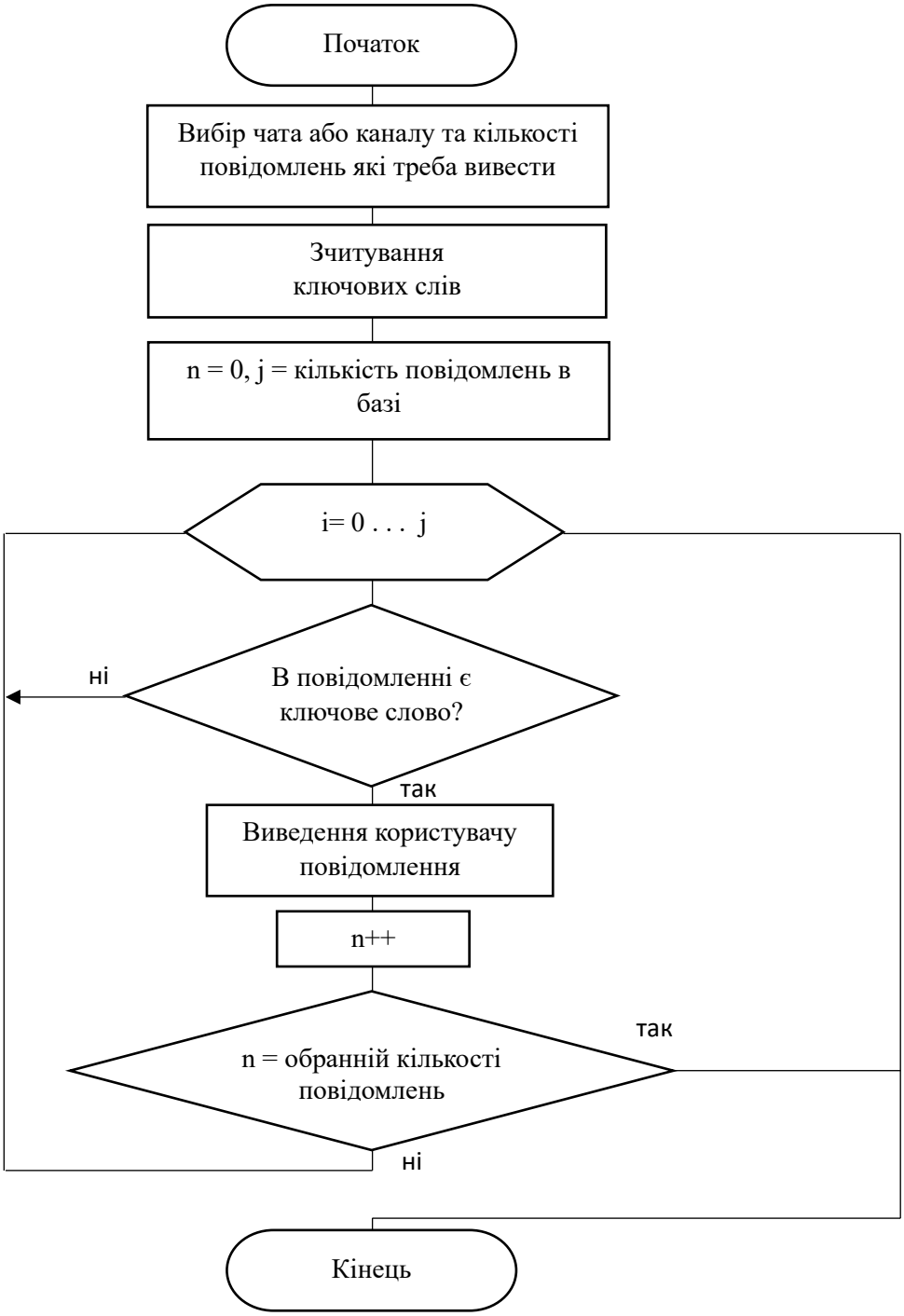



Рисунок 2.3 – Схема роботи блоку пошуку повідомлень з ключовим словом в базі даних

Після вибору канала або чата та кількості повідомлень, яке треба вивести, програма зчитує ключові слова. Далі програма загружає повідомлення, та виводить ті, в яких є ключові слова. Програма завершує роботу, після того як було виведено обрану кількість слів або було пройдено всі повідомлення.

### 2.3.4 Блок виведення статистики телеграм аккаунта

Блок виведення статистики телеграм аккаунта є важливою складовою частиною програми, яка надає можливість користувачеві переглядати статистичні дані про активність телеграм аккаунтів та їх взаємодію у Telegram. У цьому блоку увагу зосереджено на виведенні статистики щодо чатів, в яких телеграм аккаунт пише, а також каналів, з яких він пересилає повідомлення.


Рисунок 2.4 – Схема роботи блоку виведення статистики телеграм аккаунта

Після введення аккаунта, програма перевіряє чи існує він в базі даних. Якщо аккаунт існує, то програма починає зчитувати повідомлення аккаунта які є

в базі, також перевіряється, скільки повідомлень переслано. На цих повідомленнях формується статистика, а саме: в яких чатах аккаунт найбільш активний та з яких каналів він пересилає повідомлення.

### 2.3.5 Блок виявлення зв’зків телеграм-каналів

Блок виявлення зв'язків телеграм-каналів є важливою частиною розробленого парсера телеграм-каналів. Його основна мета полягає в аналізі та виявленні зв'язків між різними каналами на основі зібраних даних.


Рисунок 2.5 – Схема роботи блоку виведення пов’язаних каналів

Після вибору телеграм-каналу, програма переглядає повідомлення в каналі, та шукає переслані повідомлення. Далі програма починає шукати повідомлення які переслались з цього каналу, після чого виводить користувачу статистику.

## 2.4 Структура бази даних

При розробці парсера телеграм-каналів та збереженні зібраних даних, використовується база даних, яка має структуру, що складається з трьох таблиць: таблиці з користувачами, таблиці з каналами та чатами, і таблиці з повідомленнями. Ця структура дозволяє зберігати та організовувати дані ефективно, забезпечуючи швидкий доступ до інформації та зручну обробку.

1) Таблиця інформації про користувачів:

В цій таблиці зберігається інформація про користувачів, які взаємодіють з телеграм-каналами. Кожен запис у таблиці містить наступні поля:

- Унікальний ідентифікатор користувача: це унікальне значення, яке ідентифікує кожного користувача в системі.
- Ім'я: ім'я користувача.
- Нікнейм: нікнейм або псевдонім користувача.
- Контактна інформація: інформація, яка дозволяє зв'язатися з користувачем, наприклад, адреса електронної пошти або номер телефону.

Ця таблиця допомагає відстежувати активність користувачів, їх зв'язки з каналами та чатами, а також надає зручну основу для аналізу поведінки користувачів.

2) Таблиця з каналами та чатами.

У цій таблиці зберігається інформація про телеграм-канали та чати. Кожен запис у таблиці містить наступні поля:

- Унікальний ідентифікатор каналу або чату: унікальне значення, яке ідентифікує кожен канал або чат в системі.
- Назва: назва каналу або чату.
- Опис: опис, який пояснює тематику або мету каналу або чату.

Ця таблиця дозволяє відстежувати і категоризувати канали та чати, зберігати інформацію про них, а також встановлювати зв'язки між користувачами та каналами чи чатами.

3) Таблиця з повідомленнями.

У цій таблиці зберігається інформація про кожне повідомлення, яке було зібрано з телеграм-каналів. Кожен запис у таблиці містить наступні поля:

- Унікальний ідентифікатор повідомлення: унікальне значення, яке ідентифікує кожне повідомлення в системі.
- Канал або чат: посилання на канал або чат, з якого було зібрано повідомлення.
- Автор: інформація про автора повідомлення.
- Текст повідомлення: текстовий зміст повідомлення.
- Дата та час публікації: дата та час публікації повідомлення.

Ця таблиця дозволяє зберігати деталі повідомлень, зібраних з каналів, та забезпечує зручну основу для подальшого аналізу та обробки цих даних.

4) Таблиця зі списками ключових слів.

У цій таблиці зберігаються списки користувачів. Кожен запис у таблиці містить наступні поля:

- Унікальний ідентифікатор запису: унікальне значення, яке ідентифікує кожне повідомлення в системі.
- Унікальний ідентифікатор користувача: унікальне значення, яке ідентифікує користувачів в таблиці.
- Список унікальних слів: масив, в якому будуть зберігатись ключові слова та фрази користувача.

Ці чотири таблиці разом утворюють структуру бази даних для зберігання та організації даних, зібраних з телеграм-каналів. Користувачі, канали та повідомлення мають відповідні поля, які дозволяють зберігати необхідну інформацію. Правильна структура бази даних забезпечує швидкий доступ до

інформації, легку навігацію та зручну обробку даних для подальшого використання та аналізу.

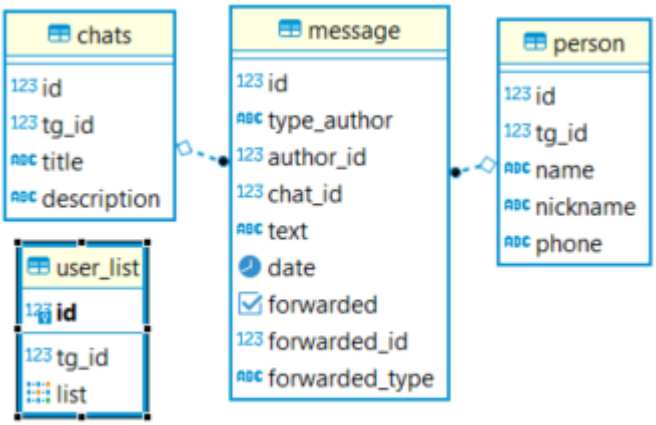



Рисунок 2.6 – Структура бази даних

Доступи до бази даних можуть бути налаштовані залежно від вимог проекту. Зазвичай, для забезпечення безпеки даних, рекомендується використовувати механізми автентифікації та авторизації. Це означає, що лише авторизовані користувачі мають доступ до бази даних і відповідні права для читання, запису та модифікації даних.

Щодо розгортання бази даних, можна використовувати різні підходи, такі як локальне розгортання на власному сервері або використання хмарних сервісів. Вибір методу розгортання залежить від потреб проекту, доступності ресурсів та безпекових вимог.

Одним із способів масштабування бази даних може бути горизонтальне масштабування, коли дані розподіляються між кількома серверами або вузлами. Це дозволяє підвищити продуктивність та надійність системи шляхом розподілу навантаження.

Загалом, структура бази даних для парсера телеграм-каналів може бути налаштована та масштабована залежно від потреб проекту та вимог щодо безпеки та продуктивності.

## 2.5 Висновки до розділу

В результаті розділу описано методології, що використовується для розробки інформаційної системи. Це може включати методи збирання, обробки та перевірки інформації. Важливо мати ефективну методику для забезпечення якості та достовірності даних.

Розроблено загальну структуру програми. Це включає в себе опис компонентів та їх взаємодії, що допомагає забезпечити ефективну та стабільну роботу програми.

Описано окремі блоки програми, такі як блок зчитування нової інформації, блок входу в канал або чат, блок пошуку повідомлень з ключовим словом в базі даних, блок виведення статистики телеграм аккаунта та блок виявлення зв'язків телеграм-каналів. Кожен з цих блоків виконує конкретну функцію і взаємодіє з іншими блоками для досягнення поставленої мети.

Описано структуру бази даних, що використовується у програмі. Це може включати таблиці, поля, зв'язки між ними та інші елементи, необхідні для зберігання та організації даних.

В наступному розділі будуть розглянуті засоби розробки і програмна реалізація, також проведено тестування.

# 3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

## 3.1 Засоби розробки

У процесі розробки проекту було обрано наступні засоби розробки:

1) Python: Мова програмування Python була обрана з декількох причин, що надають переваги над іншими мовами програмування. По-перше, Python має простий та зрозумілий синтаксис, що полегшує розробку та зрозуміння коду. Це робить його ідеальним вибором для швидкого прототипування та розвитку проекту.

Другою перевагою Python є його велика та активна спільнота розробників. Це означає, що є безліч сторонніх бібліотек та фреймворків, які спрощують роботу з різними завданнями. У контексті розробки Telegram-бота, Python має популярні бібліотеки, такі як Telethon та Telebot, які спрощують взаємодію з Telegram API.

2) PostgreSQL: Система управління базами даних (СУБД) PostgreSQL була обрана замість інших аналогічних систем з-за своїх переваг. PostgreSQL є потужною та надійною СУБД з відкритим вихідним кодом, що надає широкі можливості для роботи з даними.

Одна з головних переваг PostgreSQL – це його розширені можливості в роботі зі складними типами даних, наприклад JSON, масиви, географічні дані тощо. Це дозволяє зручно зберігати та опрацьовувати дані в контексті Telegram-бота, де можуть бути потреби в роботі зі структурованими та розширеними даними.

3) Telethon та Telebot: Бібліотеки Telethon та Telebot були вибрані для взаємодії з Telegram API та реалізації функціональності бота. Обидві бібліотеки надають зручний та простий інтерфейс для створення та керування Telegram-ботами.

Telethon є повнофункціональною бібліотекою для взаємодії з Telegram API. Вона дозволяє відправляти повідомлення, отримувати оновлення, керувати контактами та багато іншого. Використання Telethon дозволяє швидко і зручно реалізувати потрібні функції та можливості бота.

Telebot, з іншого боку, є бібліотекою, спеціалізованою на розробку Telegram-ботів з використанням простого та зрозумілого API. Вона надає інструменти для обробки вхідних повідомлень, відправки відповідей та керування ботом. Використання Telebot спрощує розробку та розширення функціональності бота.

Обрані засоби розробки мають свої переваги над аналогами, такі як простота, ефективність, надійність та розширені можливості. Використання цих засобів дозволить розробити проект швидко, ефективно та з високою якістю. Вони допоможуть у реалізації багатьох функцій, забезпечать надійне зберігання даних та зручну взаємодію з користувачами через Telegram. Загалом, обрані засоби розробки відповідають вимогам та сприятимуть успішному втіленню проекту.

## 3.2 Програмна реалізація

### 3.2.1 Блок зчитування нової інформації

Цей код має на меті служити як обробник для нових повідомлень, які надходять до Telegram-клієнта.

1) Функція function1(chat_id) отримує chat_id і виконує запит до бази даних, витягуючи дані користувача з таблиці user_list за заданим tg_id.

```
def function1(chat_id):
    array = PG.select(sql="select * from user_list where tg_id
= " + str(chat_id))
    list = array[0]['list']
```

2) Після отримання результату запиту, вона викликає функцію monit(chat_id, list) в асинхронному режимі.
3) Функція monit(chat_id, list) створює Telegram-клієнта, підключається до Telegram за допомогою зазначених параметрів (таких як phone, api_id і api_hash) і розпочинає сесію клієнта.

```
client = TelegramClient(phone, api_id, api_hash)
    await client.start()
```

4) За допомогою декоратора @client.on(events.NewMessage) встановлюється обробник для нових повідомлень, який буде виконуватись при кожному новому отриманому повідомленні.

5) У функції handle_new_message(event) виконується обробка отриманого повідомлення.

```
@client.on(events.NewMessage)
    async def handle_new_message(event):
```

6) Повідомлення розглядаються з точки зору типу відправника (користувач або канал) та типу отримувача (чат).

7) За допомогою функції PG.check перевіряється наявність відправника та отримувача в базі даних і отримується їх ідентифікатор (author_id і chat_id).

```
if hasattr(sender, 'first_name') and hasattr(sender,
'last_name'):
            print('per')
            #print("Це користувач")
            result_person = {}
            result_person['tg_id'] = sender.id
            result_person['name'] = sender.first_name
            result_person['nickname'] = sender.username
            result_person['phone'] = sender.phone
            print(result_person['tg_id'])
            result_message['author_id'] = PG.check('person',
result_person, key="tg_id")
            print(result_message['author_id'])
            result_message['type_author'] = 'User'
        elif hasattr(sender, 'title'):
            print('cha')
            #print("Це канал")
            result_chats = {}
            result_chats['tg_id'] = sender.id
            result_chats['title'] = sender.title
            result_message['author_id'] = PG.check('chats',
result_chats, key="tg_id")
            result_message['type_author'] = 'Channel'
```

8) Інформація про повідомлення (включаючи відправника, отримувача, текст повідомлення, дату тощо) зберігається в таблиці message за допомогою функції PG.insert.

```
PG.insert('message', result_message)
```

9) Далі відбувається перевірка наявності ключових слів (зі списку list) у тексті повідомлення. Якщо знайдено співпадіння, бот надсилає повідомлення з інформацією про знайдене співпадіння.

```
if list is not None:
            for word in list:
```

```
                if word.lower() in message.text.lower():
                    bot.send_message(chat_id, f'Користучач:
{sender.username}\nЧат: {chat.title} | ID: {chat.id}\nЧас:
{date}\nКлючове слово:
{word}\n\nПовідомлення:\n{message.text}')
                    break
```

10) Запускається виконання клієнта за допомогою client.run_until_disconnected(), що дозволяє програмі працювати, поки не відбудеться відключення від Telegram.

### 3.2.2 Блок введення списку

Цей блок має на меті служити для збереження списку ключових слів або фраз в базі даних.

1) У функції process_function2_input отримується повідомлення (message) з Telegram. З нього витягуються chat_id (ідентифікатор чату) і user_id (ідентифікатор користувача), що будуть використовуватись пізніше.

```
chat_id = message.chat.id
user_id = message.from_user.id
```

2) Вхідний текст повідомлення (message.text) розбивається за допомогою команди split('|') на спиок. Отриманий список зберігається в змінну input_list.
3) Функція function2 викликається з аргументами chat_id, user_id і input_list.
4) У функції function2 створюється словник result, в який записується tg_id (chat_id) та list (список, що був отриманий з повідомлення).
5) Перевіряється, чи існує запис зі значенням tg_id в таблиці "user_list" у базі даних PostgreSQL. Якщо ні, то словник result вставляється в таблицю за допомогою PG.insert. Якщо такий запис вже існує, то він оновлюється за допомогою PG.update.

```
  if PG.get('user_list', {'tg_id': result['tg_id']}) is None:
        PG.insert('user_list', result)
    else:
        PG.update('user_list', {'tg_id': result['tg_id']},
result)
```

6) Завершення функції function2 відправляє користувачеві повідомлення через bot.send_message, яке підтверджує, що список був доданий успішно.

3.2.3 Блок додавання каналів

1) В функції function3() отримується chat_id (ідентифікатор чату) та channel_name (назва каналу) з об'єкту message. Після цього викликається асинхронна функція add_channel з передачею chat_id та channel_name як аргументів.

```
chat_id = message.chat.id
    channel_name = message.text
    asyncio.run(add_channel(chat_id, channel_name))
```

2) Визначається асинхронна функція add_channel, яка отримує chat_id та channel_name як параметри. У контексті з'єднання TelegramClient виконується запит на приєднання до каналу з використанням JoinChannelRequest. Потім отримується інформація про канал (entity) за допомогою get_entity, і зберігається його ідентифікатор у змінну channel_id.

```
async def add_channel(chat_id, channel_name):
    async with TelegramClient(phone, api_id, api_hash) as
client:
        await client(JoinChannelRequest(channel_name))
        entity = await client.get_entity(channel_name)
        channel_id = entity.id
```

3) Перевіряється, чи є запис про канал з channel_id у таблиці "chats" бази даних. Якщо запис відсутній, створюється словник result, до якого додаються ідентифікатор каналу (tg_id) та назва каналу (title). Потім цей словник вставляється у таблицю "chats" за допомогою PG.insert. Далі відправляється повідомлення користувачеві про додавання каналу та викликається асинхронна функція parse, якій передаються об'єкт client, назва каналу (channel_name) та chat_id. Якщо запис про канал вже існує, то відправляється повідомлення про це.

```
row = PG.get('chats', {'tg_id': channel_id})
if row is None:
    result = {}
```

```
    result['tg_id'] = channel_id
    result['title'] = entity.title
    PG.insert('chats', result)
    bot.send_message(chat_id, 'Канал додано канал, йде
зчитування останніх повідомлень')
    await parse(client, channel_name, chat_id)
else:
    bot.send_message(chat_id, f'Канал вже існує в базі даних
під id: {row["id"]}')
```

4) Відбувається отримання списку учасників заданої групи (target_group) за допомогою функції client.get_participants(). Для кожного користувача отриманого списку створюється словник result_user, де зберігаються інформація про його ID, ім'я, нікнейм та телефон. Потім цей словник додається до бази даних через функцію PG.insert('person', result_user). У випадку виникнення помилки, використовується except Exception: для перехоплення та обробки виняткових ситуацій.

```
async def parse(client, target_group, chat_s_id):
    try:
        all_participants = await
client.get_participants(target_group)
        for user in all_participants:
            result_user = {}
            result_user['tg_id'] = str(user.id)
            result_user['name'] = str(user.first_name)
            result_user['nickname'] = str(user.username)
            result_user['phone'] = str(user.phone)
            PG.insert('person', result_user)
    except Exception:
```

4) Відбувається отримання історії повідомлень для заданої групи (target_group) за допомогою функції client(GetHistoryRequest(...)). За допомогою циклу while перебираються всі повідомлення, отримані з історії. Кожне повідомлення перетворюється на словник та додається до списку all_messages. За допомогою змінної total_messages відстежується загальна кількість отриманих повідомлень. Змінні offset_id, limit, total_count_limit використовуються для керування отриманням історії повідомлень. Якщо total_messages досягне або перевищить total_count_limit, або ж в історії повідомлень більше немає повідомлень, цикл завершується, і виконується решта коду після цього фрагменту.

```
all_messages = []
offset_id = 0
limit = 100
total_messages = 0
total_count_limit = 1000
while total_messages < total_count_limit:
    history = await client(GetHistoryRequest(
        peer=target_group,
        offset_id=offset_id,
        offset_date=None,
        add_offset=0,
        limit=limit,
        max_id=0,
        min_id=0,
        hash=0
    ))
    if history.messages:
        messages = history.messages
        for message in messages:
            all_messages.append(message.to_dict())
            total_messages += 1
        offset_id = messages[len(messages) - 1].id
    if total_messages >= total_count_limit or not
history.messages:
        for message in all_messages:
```

5) Кожне повідомлення зі списку all_messages перебирається. Перевіряється його тип ('chat_id' або 'channel_id'). Залежно від типу повідомлення, створюється словник result, в який зберігається інформація про тип автора, ID автора, ID чату, текст повідомлення, дату та, у разі пересланого повідомлення, відповідна інформація про автора пересланого повідомлення.
6) Словник додається до бази даних за допомогою PG.insert('message', result).

3.2.4 Блок виведення статистики телеграм-каналу

1) Витягується chat_id з об'єкта message та channel_id з тексту повідомлення. Потім надсилається повідомлення до chat_id з текстом "Введіть мінімальний відсоток:". Далі, за допомогою bot.register_next_step_handler, реєструється обробник наступного кроку, який викличе функцію process_cha_stats з параметрами message, chat_id та channel_id.

```
def function4(message):
    chat_id = message.chat.id
    channel_id = message.text
    bot.send_message(chat_id, 'Введіть мінімальний відсоток:')
    bot.register_next_step_handler(message, process_cha_stats,
chat_id, channel_id)
```

2) Функція process_cha_stats приймає параметри message, chat_id та channel_id. Витягується текст повідомлення message.text та присвоюється змінній percentage. Потім використовується конструкція try-except для перевірки, чи можна перетворити percentage у тип float. Якщо це неможливо (виникає помилка ValueError), бот надсилає повідомлення до chat_id з текстом "Неправильний формат проценту. Будь ласка, введіть число." і функція завершується (return).

```
def process_cha_stats(message, chat_id, channel_id):
    percentage = message.text
    try:
        min_percentage = float(percentage)
    except ValueError:
        bot.send_message(chat_id, 'Неправильний формат
проценту. Будь ласка, введіть число.')
        return
```

4) Виконується запит до бази даних за допомогою PG.select для отримання записів з таблиці message, де chat_id дорівнює channel_id. Якщо отримано хоча б один запис, створюється порожній словник result. Потім проходиться по кожному запису у array. Якщо значення поля forwarded рівне True, витягуються значення полів forwarded_type та forwarded_id. Якщо forwarded_type ще не присутній у result, створюється порожній словник. Якщо forwarded_id ще не присутній у result[forwarded_type], створюється ключ зі значенням 0. Потім значення цього ключа збільшується на 1.

```
array = PG.select(sql="select * from message where chat_id = "
+ channel_id)
if len(array) > 0:
    result = {}
    for row in array:
        if row['forwarded'] is True:
            forwarded_type = row['forwarded_type']
            forwarded_id = row['forwarded_id']
            if forwarded_type not in result:
                result[forwarded_type] = {}
```

```
            if forwarded_id not in result[forwarded_type]:
                result[forwarded_type][forwarded_id] = 0
            result[forwarded_type][forwarded_id] += 1
```

5) Створюється порожній словник percentage_result. Проходиться по кожному запису у result. Обчислюється відсоток для кожного forwarded_type та forwarded_id шляхом ділення кількості на загальну кількість записів у array та множення на 100. Ці значення зберігаються в percentage_result[forwarded_type][forwarded_id].

```
percentage_result = {}
for forwarded_type, forwarded_ids in result.items():
    percentage_result[forwarded_type] = {}
    for forwarded_id, count in forwarded_ids.items():
        percentage = (count / len(array)) * 100
        percentage_result[forwarded_type][forwarded_id] =
percentage
```

5) створюється порожній список filtered_results. Проходиться по кожному запису у percentage_result. Якщо значення відсотку (percentage) більше або дорівнює min_percentage, то дані запису (forwarded_type, forwarded_id, percentage) додаються до списку filtered_results. Після цього список filtered_results сортується за третім елементом (відсотком) у зворотному порядку, щоб отримати найбільші значення відсотку на початку списку.

```
filtered_results = []
for forwarded_type, forwarded_ids in
percentage_result.items():
    for forwarded_id, percentage in forwarded_ids.items():
        if percentage >= min_percentage:
            filtered_results.append((forwarded_type,
forwarded_id, percentage))
filtered_results = sorted(filtered_results, key=lambda x:
x[2], reverse=True)
```

3.2.5 Блок пошуку повідомлень в базі даних по ключовим словам

1) Отримуються значення chat_id та channel_id з отриманого повідомлення. Відправляється повідомлення до chat_id з запитом ввести кількість необхідних повідомлень. Також реєструється наступний обробник кроку process_seacrch з аргументами message, chat_id та channel_id.

```
chat_id = message.chat.id
channel_id = message.text
```

```
bot.send_message(chat_id, 'Введіть потрібну кількість
повідомлень:')
bot.register_next_step_handler(message, process_seacrch,
chat_id, channel_id)
```

2) Отримується введена кількість повідомлень з message.text. Виконується SQL-запит до бази даних для отримання ключових слів користувача за його tg_id. Запит шукає запис у таблиці user_list, де значення tg_id співпадає з chat_id отриманим з message. Результат запиту зберігається у змінній key_word.

```
total_count = message.text
key_word = PG.select(sql="select * from user_list where tg_id
= " + str(chat_id))
```

3) Перевіряється, чи отримано які-небудь ключові слова з бази даних. Якщо key_word не є пустим, то виконується ще один SQL-запит до бази даних. Запит шукає всі повідомлення з таблиці message, де значення chat_id співпадає з channel_id отриманим з message. Результат запиту зберігається у змінній array. Якщо key_word є пустим (тобто не знайдено жодного ключового слова), то відправляється повідомлення "Ключових слів не знайдено".

```
if key_word is not None:
    array = PG.select(sql="select * from message where chat_id
= " + str(channel_id))
……
else:
    bot.send_message(chat_id, 'Ключових слів не знайдено')
```

4) Перевіряється, чи є повідомлення в результаті попереднього SQL-запиту. Якщо array не є пустим, то виконується подальша обробка. У циклі for перебираються всі записи row з array. Для кожного запису перебираються ключові слова word зі списку key_word[0]['list']. Якщо знайдене ключове слово (переведене у нижній регістр) знаходиться у тексті повідомлення row['text'] (також переведеному у нижній регістр), то виконується відправка повідомлення з відповідною інформацією до chat_id. Змінна n використовується для підрахунку кількості відправлених повідомлень з ключовими словами. Якщо кількість відправлених повідомлень досягає значення total_count, цикл

припиняється. Після циклу відправляється повідомлення про завершення зчитування. Якщо array є пустим (тобто не знайдено жодного повідомлення), то відправляється повідомлення "Повідомлень не знайдено".

```
n = 0
if array is not None:
    for row in array:
        for word in key_word[0]['list']:
            if word.lower() in row['text'].lower():
                bot.send_message(chat_id,
f'{row["type_author"]} id: {row["author_id"]}\nЧат id:
{row["chat_id"]}\nЧас: {row["date"]}\nКлючове слово:
{word}\n\nПовідомлення:\n{row["text"]}')
                n += 1
                break
        if n >= int(total_count):
            break
    bot.send_message(chat_id, 'Зчитування завершено')
else:
    bot.send_message(chat_id, 'Повідомлень не знайдено')
```

3.2.6 Блок виведення статистики телеграм аккаунта

1) Отримуються значення chat_id та user_id з отриманого повідомлення. Виконується SQL-запит до бази даних для отримання повідомлень, де тип автора є "User" і ідентифікатор автора співпадає з user_id. Результат запиту зберігається у змінній array.

```
chat_id = message.chat.id
user_id = message.text
array = PG.select(sql="select * from message where type_author
= 'User' and author_id = " + str(user_id))
```

2) Перевіряється, чи отримано результати від SQL-запиту. Якщо array не є пустим, то виконується обробка результатів. Створюється словник result, де ключами є значення chat_id, а значеннями – кількість повідомлень автора з user_id в цьому чаті. Якщо chat_id не міститься у словнику result, воно додається зі значенням 0. Кількість повідомлень для кожного chat_id збільшується на 1.

```
if array is not None:
    result = {}
    for user in array:
        chat = user['chat_id']
        if chat not in result:
```

```
            result[chat] = 0
        result[chat] += 1
```

3) Обчислюється відсоткове співвідношення кількості повідомлень автора в кожному чаті до загальної кількості повідомлень. Створюється порожній словник percentage. Для кожного chat_id та кількості повідомлень count в словнику result обчислюється відсоткове співвідношення (count / len(array)) * 100 та зберігається у словнику percentage. Відправляється повідомлення до chat_id зі списком chat_id та відсотковим співвідношенням. Для кожного chat_id і його відсотку виконується відправлення повідомлення зі значеннями Chat ID: <chat_id>, Відсоток: <percent>%.

```
percentage = {}
for chat, count in result.items():
    percentage[chat] = (count / len(array)) * 100
bot.send_message(chat_id, 'Список chat_id та відсоткове
співвідношення:')
for chat, percent in percentage.items():
    bot.send_message(chat_id, f'Chat ID: {chat}, Відсоток:
{percent}%')
```

### 3.3 Результати тестування

В даному підрозділі було проведено тестування програмного застосунку.

При введені команди /start, з’являється повідомленнями з 7 кнопками (рис. 3.1).

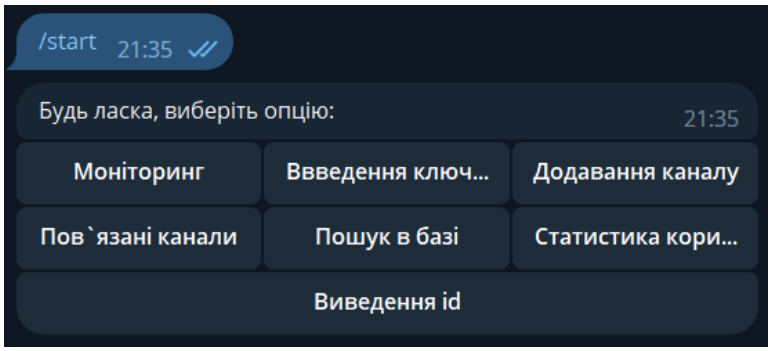
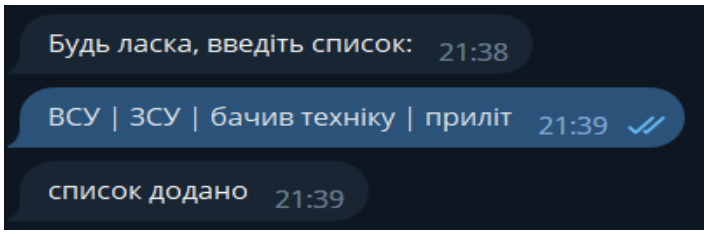


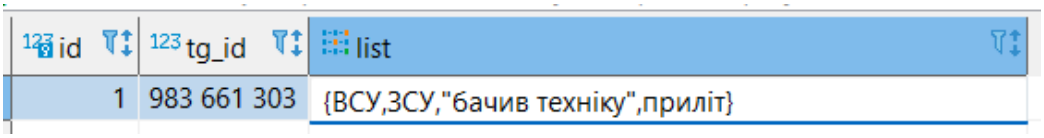

Рисунок 3.1 – Результат виконання команди /start

В кожної кнопки є свій функціонал, розглянемо їх більше детально

Спробуємо ввести ключові слова та фрази (рис. 3.2).




а)



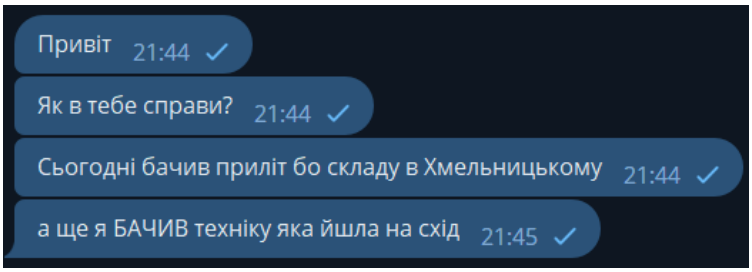
| id | tg_id | list |
|---|---|---|
| 1 | 983 661 303 | {ВСУ,ЗСУ,"бачив техніку",приліт} |

б)

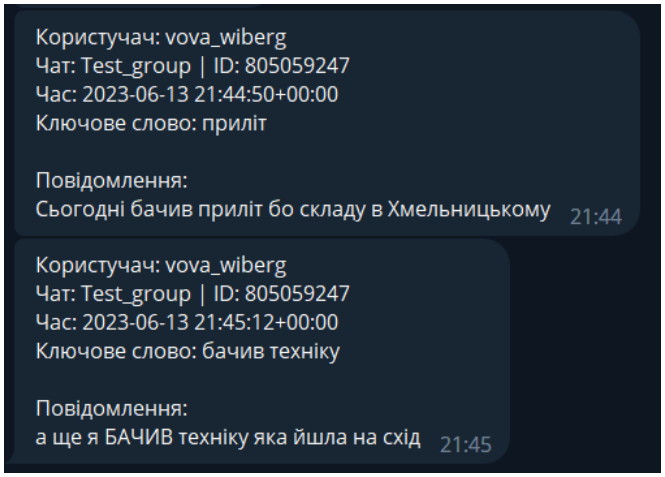
Рисунок 3.2 – Результат виконання команди для додавання списку: а – у вікні телеграма , б – в базі даних

Як можна побачити, програма зчитала наше повідомлення, розділила та записало в базу даних списком, по нашому tg_id.

Далі розглянемо частину моніторингу, тестування будуть проводитись в групі Test_group (рис. 3.3).




а)




б)

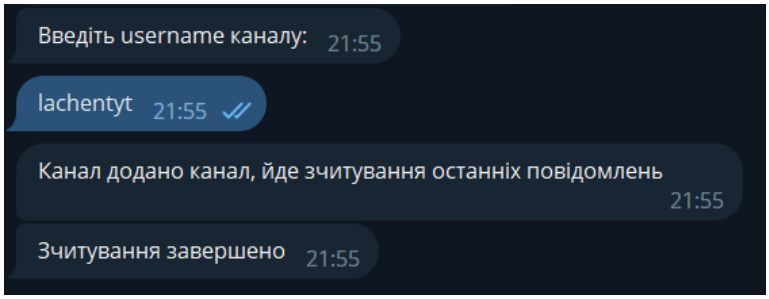
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 393 | 397 | User | 1 | 2 | Привіт | 2023-06-13 21:44:31.000 | [ ] | [NULL] [NULL] |
| 394 | 398 | User | 1 | 2 | Як в тебе справи? | 2023-06-13 21:44:37.000 | [ ] | [NULL] [NULL] |
| 395 | 399 | User | 1 | 2 | Сьогодні бачив приліт бо складу в Хмельницькому | 2023-06-13 21:44:50.000 | [ ] | [NULL] [NULL] |
| 396 | 400 | User | 1 | 2 | а ще я БАЧИВ техніку яка йшла на схід | 2023-06-13 21:45:12.000 | [ ] | [NULL] [NULL] |

в)

Рисунок 3.3 – Результат виконання команди моніторингу: а – повідомлення в чаті, б – виведення ботом співпадінь, в – в базі даних

Як можна побачити, було надіслано 4 повідомлення, всі вони були записані в базу даних коректно. В 3 та 4 повідомленні були ключове слово та фраза, в результаті чого телеграм бот вивів нам їх повідомленням з додатковою інформацією. Також можна помітити, що в фразі слово «бачив» був записаний верхнім регістром, але ключову фразу все одно було знайдено, це показує, що пошук стійкий до зміни регістра.

Наступним ділом перевіримо додавання нового каналу в базу даних та зчитування останніх повідомлень (рис. 3.4)




а)

| id | tg_id | title |
|---|---|---|
| 1 | 1 621 065 700 | test_cha |
| 2 | 805 059 247 | Test_group |
| 3 | 1 127 874 961 | [NULL] |
| 7 | 907 832 413 | Test_group2 |
| 8 | 1 246 634 572 | Чат \|\| ВНТУ |
| 9 | 1 536 630 827 | Лачен пише |

б)

| | id | type_author | author_id | chat_id | text | date | forwarded | forwarded_id | forwarded_type |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 401 | channel | 9 | 9 | Судді Верховного Суду висловили недовіру голові с | 2023-05-16 12:26:20.000 | [ ] | [NULL] | [NULL] |
| 2 | 1 400 | channel | 9 | 9 | Протягом декількох днів ЗСУ звільнили близько 20 і | 2023-05-16 13:57:49.000 | [ ] | [NULL] | [NULL] |
| 3 | 1 399 | channel | 9 | 9 | Частина підривів медійних осіб у РФ - справа рук Г | 2023-05-16 14:23:33.000 | [ ] | [NULL] | [NULL] |
| 4 | 1 398 | channel | 9 | 9 | Компанія Hensoldt уклала контракт на постачання У | 2023-05-16 14:41:01.000 | [ ] | [NULL] | [NULL] |
| 5 | 1 397 | channel | 9 | 9 | Захід підтверджує використання Україною ракет Stc | 2023-05-16 16:13:21.000 | [ ] | [NULL] | [NULL] |
| 6 | 1 396 | channel | 9 | 9 | Прем'єр-міністр Британії Сунак планує обговорити | 2023-05-16 17:12:18.000 | [ ] | [NULL] | [NULL] |
| 7 | 1 395 | channel | 9 | 9 | Відповідальність РФ за війну стане однією з тем сам | 2023-05-16 18:30:07.000 | [ ] | [NULL] | [NULL] |
| 8 | 1 394 | channel | 9 | 9 | ❗ В Білому домі не можуть підтвердити інформаці | 2023-05-16 18:46:49.000 | [ ] | [NULL] | [NULL] |
| 9 | 1 393 | channel | 9 | 9 | ❗ Велика Британія і Нідерланди домовилися про р | 2023-05-16 18:59:33.000 | [ ] | [NULL] | [NULL] |
| 10 | 1 392 | channel | 9 | 9 | Бельгія готова навчати українських пілотів-винищу | 2023-05-16 19:19:29.000 | [ ] | [NULL] | [NULL] |
| 11 | 1 391 | channel | 9 | 9 | Тривогу не ігноруйте, — ОП. | 2023-05-16 20:12:49.000 | [ ] | [NULL] | [NULL] |
| 12 | 1 390 | channel | 9 | 9 | Вибух у Миколаєві. | 2023-05-16 20:14:24.000 | [ ] | [NULL] | [NULL] |
| 13 | 1 389 | channel | 9 | 9 | Миколаїв. Маємо пожежі після вибухів, — мер міста | 2023-05-16 20:37:33.000 | [ ] | [NULL] | [NULL] |
| 14 | 1 388 | channel | 9 | 9 | ❗ Державний секретар США уперше допустив, що | 2023-05-16 21:29:33.000 | [ ] | [NULL] | [NULL] |
| 15 | 1 387 | channel | 9 | 9 | 200 тис знищених росіян | 2023-05-17 07:54:31.000 | [ ] | [NULL] | [NULL] |
| 16 | 1 386 | channel | 9 | 9 | СБУ викрила мразот та заблокувала онлайн-камери | 2023-05-17 08:01:45.000 | [ ] | [NULL] | [NULL] |
| 17 | 1 385 | channel | 9 | 9 | У Палаті представників США підтримали резолюцік | 2023-05-17 08:19:24.000 | [ ] | [NULL] | [NULL] |
| 18 | 1 384 | channel | 9 | 9 | Швейцарія арештувала активи Жеваго на понад $11 | 2023-05-17 08:41:06.000 | [ ] | [NULL] | [NULL] |
| 19 | 1 383 | channel | 9 | 9 | ❗ 43 держави схвалили угоду про реєстр збитків в | 2023-05-17 09:31:17.000 | [ ] | [NULL] | [NULL] |
| 20 | 1 382 | channel | 9 | 9 | Польща не може надати Україні винищувачі F-16 ос | 2023-05-17 09:59:24.000 | [ ] | [NULL] | [NULL] |
| 21 | 1 381 | channel | 9 | 9 | Південна Корея планує виділити Україні 8 млрд дол | 2023-05-17 10:13:39.000 | [ ] | [NULL] | [NULL] |
| 22 | 1 380 | channel | 9 | 9 | Прем'єр Грузії, яка відновлює сполучення з Росією: | 2023-05-17 11:15:57.000 | [ ] | [NULL] | [NULL] |
| 23 | 1 379 | channel | 9 | 9 | ❗ Франція вивчає можливість постачання Україні | 2023-05-17 11:50:31.000 | [ ] | [NULL] | [NULL] |
| 24 | 1 378 | channel | 9 | 9 | ЗСУ просунулися на 500 м на Бахмутському напрям | 2023-05-17 12:11:22.000 | [ ] | [NULL] | [NULL] |
| 25 | 1 377 | channel | 9 | 9 | Німеччина не планує брати участь в коаліції з поста | 2023-05-17 12:57:49.000 | [ ] | [NULL] | [NULL] |

в)

Рисунок 3.4 – Результат виконання команди додавання каналу: а – у вікні телеграма , б – в таблиці каналів, в – в таблиці повідомлень

В результаті бот видав бот повідомлення, що канал додано та повідомлення зчитано. Переглядаючи базу даних, можна в цьому впевнитись, канал записався під id 9 та з'явились останні повідомлення з цього каналу.

Тепер спробуємо додати цей канал ще раз (рис. 3.5).

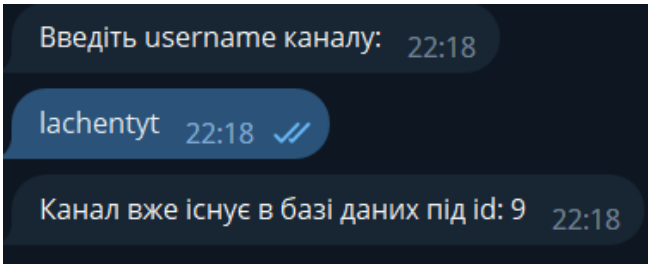



Рисунок 3.5 – Результат виконання команди додавання каналу

В результаті бот написав що канал вже існує та видав його id.

Тепер скористаємось кнопкою виведення id  та спробуємо знайти id цього каналу по його назві (рис. 3.6).

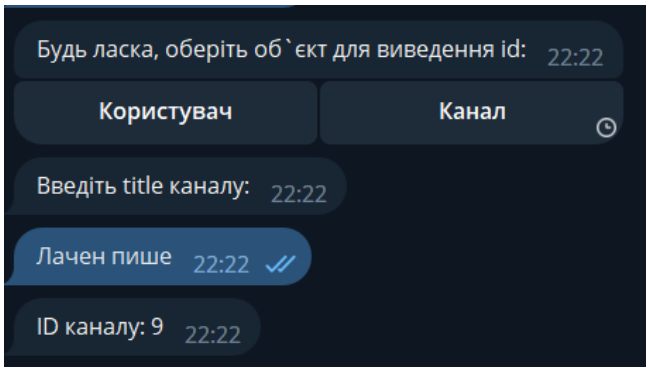



Рисунок 3.6 – Результат виконання команди пошуку каналу

Як можна побачити, програма знайшла запис в базі даних та вивела нам його id.

Наступною буде розглянуто кнопку «Пов`язані канали» (рис. 3.7)

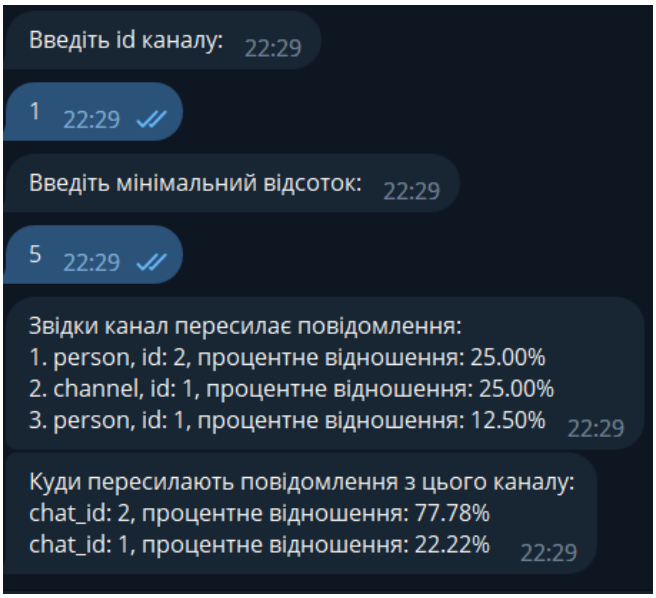



Рисунок 3.7 – Результат виконання команди виведення пов’язаних каналів

В результаті команда показала, який процент користувачів та каналів персилає собі обраний канал, відносно всіх повідомлень в каналі. Також показано канали та чати, в які пересилаються повідомлення з обраного каналу.

Спробуємо збільшити процент до 20 (рис. 3.8).

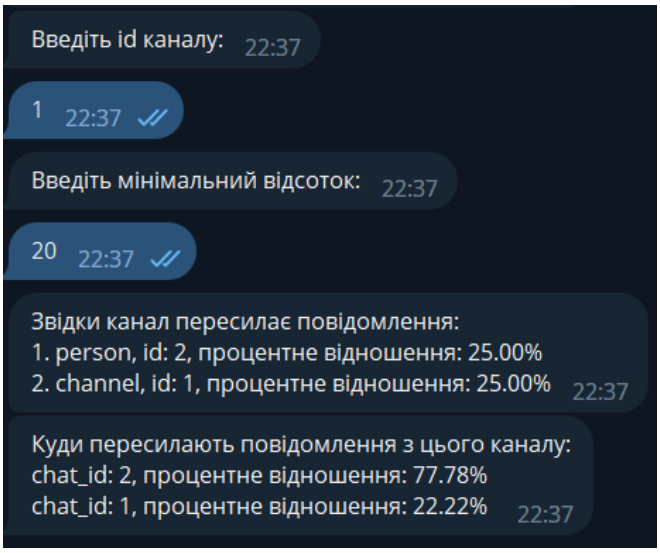



Рисунок 3.8 – Результат виконання команди виведення пов’язаних каналів

Як можна побачити, на цей раз користувача з id 1 не було виведено, так як його процент становив меньше 20.

Тепер спробуємо отримати статистику користувача (рис. 3.9)

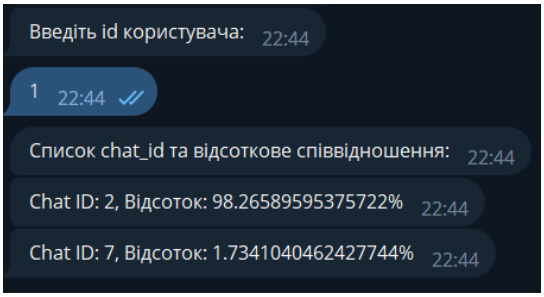



Рисунок 3.9 – Результат виконання команди виведення статистики користувачів

В результаті було отримано два chat id та їхнє процентне відношення, перевіривши базу даних, можна сказати що процент розраховано правильно. Переглядаючи цю інформацію, можна дійти до висновку, що користувач найбільше сидить в чаті, з існуючих в базі даних, з id 1.

Тепер розглянемо кнопку пошуку повідомлень по ключовим словам в базі даних (рис. 3.10).

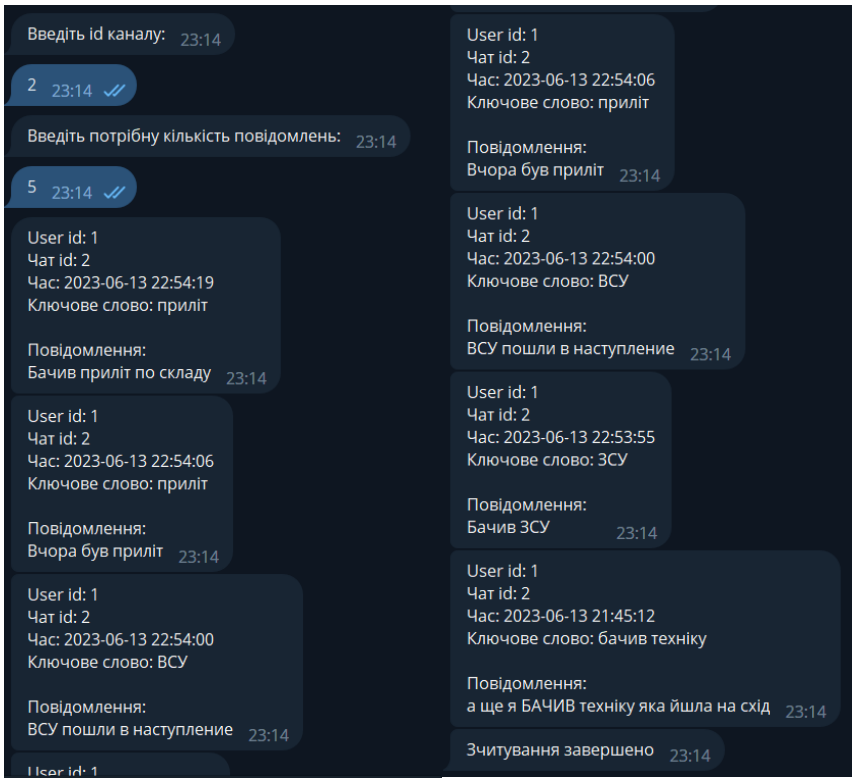



Рисунок 3.10 – Результат виконання команди пошуку в базі даних

Як можна побачити, кнопка працює коректно виводить потрібну кількість повідомлень з ключовими словами та додаткову інформацію.

В результаті даного підрозділу, було протестовано програмний застосунок. Можна зробити висновок, що програмний застосунок працює коректно та швидко. З урахуванням проведених тестів, можна стверджувати, що програма виконує свої функції без помилок і з надійною швидкістю.

### 3.4 Висновки до розділу

В даному розділі описано інструменти та технології, які використовувалися для розробки програмного забезпечення в рамках дослідження. Це включає розгляд використаних програмних середовищ, мов програмування та інших релевантних інструментів.

Розглянуто розроблену програму, що включає ряд блоків з різними функціями. Наприклад, блоки зчитування нової інформації, введення списку, додавання каналів, виведення статистики телеграм-каналу, пошуку повідомлень в базі даних за ключовими словами та виведення статистики телеграм аккаунта.

Ці блоки виконують конкретні завдання, пов'язані з обробкою та аналізом інформації.

Представлено результати проведеного тестування розробленої програми. Це включає оцінку продуктивності, надійності та інших аспектів функціонування програми. Тестування допомагає підтвердити працездатність програми та виявити потенційні проблеми чи помилки.

В результаті було отримано програмний застосунок, який дає можливість моніторингу телеграм-каналів та побудови зв’язків між ними. Всі алгоритми з другого розділу реалізовано та працюють коректно.

# ВИСНОВКИ

Дана робота присвячена розробці засобу моніторингу телеграм-каналів під час інформаційної війни. Проведено аналіз впливу фейкової інформації під час інформаційної війни та визначено важливість телеграма як засобу комунікації у цьому контексті.

У першому розділі детально розглянуто методи аналізу телеграм-каналів та висвітлено важливість використання Telegram API, бібліотеки Telethon, системи зберігання та управління даними. Також досліджено процес фільтрації, відбору інформації та аналізу користувачів телеграм-чатів. На завершення розділу поставлено завдання для подальшої роботи.

У другому розділі розроблені методики збирання, обробки та перевірки інформації, а також архітектура програми та окремі блоки програми. Розглянуто блоки зчитування нової інформації, входу в канал або чат, пошуку повідомлень з ключовим словом в базі даних, виведення статистики телеграм аккаунта та виявлення зв'язків телеграм-каналів. Також визначена структура бази даних.

Третій розділ містить експериментальні дослідження, в яких використані певні засоби розробки. Реалізована програма з відповідними блоками для зчитування нової інформації, введення списку, додавання каналів, виведення статистики телеграм-каналу, пошуку повідомлень в базі даних за ключовими словами та виведення статистики телеграм аккаунта. Результати тестування були також описані.

В цілому, розробка ефективного засобу моніторингу телеграм-каналів є важливим завданням у сучасному інформаційному просторі, що допоможе виявляти та реагувати на поширення фейкової інформації та зміну інформаційного ландшафту під час інформаційних війн.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шостак В. В. Засіб моніторингу телеграм-каналів під час інформаційної війни [Електронний ресурс] / Шостак В. В., О. П. Войтович // Матеріали Всеукраїнської науково-практичної інтернет-конференції "Молодь в науці: дослідження, проблеми, перспективи (МН-2023)", Вінниця, 22-23 червня 2023 р. – Електрон. текст. дані. – 2023. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2023/paper/view/18159

2. Почепцов Г.Г. Сучасні інформаційні війни. Києво-Могилянська академія, 2015. С. 497

3. Курбан О.В. Інформаційні війни у соціальних он-лайн-мережах. Київський університет ім. Б. Грінченка, 2017. 392 с.

4. Фармагей О.І., Остроухов В.В., Присяжнюк М.М., Чеховська М.М., Мельник Д., Петрик В., Карпович О., Інформаційна безпека. Ліра-К, 2021. 412 с.

5. Почепцов Г.Г., Чукут С.А Інформаційна політика. Знання, 2008. С. 663

6. Бутиріна М., Темченко Л. Телеграм як середовище просування російських дезінформаційних наративів: канали, методи, фрейми. Communications and Communicative Technologies. 2023. С. 9. URL: https://doi.org/10.15421/292311.

7. Понад 45% українців вірить у достовірність новин з телеграм-каналів. URL: https://gwaramedia.com/ponad-45-ukrainczіv-virit-u-dostovirnist-novin-z-telegram-kanaliv-opituvannya/

8. Яковенко Є.І., Журавель І.М., Горбатий І.В., Бондарєв А.П Інформаційна безпека. Львівська політехніка, 2019. 580 с.

9. Дослідження: безпека військових у соціальних мережах URL: https://defence-ua.com/weapon_and_tech/doslidzhennja_armijainform_bezpeka_vijskovih_u_sotsialnih_merezhah-194.html

10.Небезпечний телеграм: підводні камені найпопулярнішого в Україні месенджера. URL: https://ms.detector.media/kiberbezpeka/post/30782/2022-12-04-

nebezpechnyy-telegram-pidvodni-kameni-naypopulyarnishogo-v-ukraini-mesendzhera/

11.«Кремлівська гідра»: 300 телеграм-каналів, які отруюють український інфопростір. URL: https://detector.media/monitorynh-internetu/article/205954/2022-12-14-kremlivska-gidra-300-telegram-kanaliv-yaki-otruyuyut-ukrainskyy-infoprostir/

12.Роль соціальних мереж у кризовій комунікації в умовах війни URL: https://eba.com.ua/rol-sotsialnyh-merezh-u-kryzovij-komunikatsiyi-v-umovah-vijny/

13.Соціальні мережі, кібератаки та гібридні війни URL: https://www.radiosvoboda.org/a/28598299.html

14.Курбан О.В. Сучасні інформаційні війни у мережевому он-лайн просторі. Київ, 2016, С. 292

15. Анонімні Telegram-канали: що робити з фейками, маніпуляціями та «російським слідом» URL: https://www.radiosvoboda.org/a/anonimni-telegram-kanaly/31092089.html

16. Методологія аналізу проросійських і окупаційних телеграм-каналів в українському сегменті телеграма URL: https://detector.media/monitorynh-internetu/article/205956/2022-12-14-metodologiya-analizu-prorosiyskykh-i-okupatsiynykh-telegram-kanaliv-v-ukrainskomu-segmenti-telegrama/

17. Методологія аналізу українського сегменту соціальних мереж та месенджерів URL: https://detector.media/infospace/article/194698/2021-12-11-metodologiya-analizu-ukrainskogo-segmentu-sotsialnykh-merezh-ta-mesendzheriv

18. Навіщо потрібен моніторинг соцмереж: пояснення, поради, інструменти URL: https://www.imena.ua/blog/why-is-social-media-monitoring-necessary/

19.Wasserman S. Social Network Analysis: Methods and Applications. Cambridge University Press, 1994. 857 p.

20. Демків О.Б. Аналітичні принципи та категорії мережевого аналізу. Вісник Харківського національного університету імені. В.Н.Каразіна. Серія: Соціологічні дослідження сучасного суспільства: методологія, теорія, методи. 2004. No 621. С.45–55

21. Bunzel N., Chen T., Steinebach M. Using Telegram as a carrier for image steganography: Analysing Telegrams API limits. ARES 2022: The 17th International Conference on Availability, Reliability and Security, Vienna Austria. New York, NY, USA, 2022. URL: https://doi.org/10.1145/3538969.3544440

22. Telegram Monitor: Monitoring Brazilian Political Groups and Channels on Telegram / M. Júnior et al. HT '22: 33rd ACM Conference on Hypertext and Social Media, Barcelona Spain. New York, NY, USA, 2022. URL: https://doi.org/10.1145/3511095.3536375

23. Ramakrishnan S., Nwosu E. DBMS course. the 34th SIGCSE technical symposium, Reno, Navada, USA, 19–23 February 2003. New York, New York, USA, 2003. URL: https://doi.org/10.1145/611892.611922

24. Стеблина Н. АЛГОРИТМ ВИЯВЛЕННЯ НЕДОСТОВІРНИХ ПОСИЛАНЬ ТА ЧУТОК У ТЕЛЕГРАМ-КАНАЛАХ. Інформатизація та інтелектуалізація соціуму – проблеми взаємин. 2022. С. 94–100. URL: https://doi.org/10.21272/Obraz.2022.3(40)-94-100.

25.ГУРЖІЙ С. Сучасні загрозливі тенденції використання Telegram-каналів на шкоду державним інтересам. ІНФОРМАЦІЯ І ПРАВО. 2021. С. 8. URL: https://doi.org/10.37750/2616-6798.2021.4(39).249297.

26.Стеблина Н. ВИЯВЛЕННЯ ПРОРОСІЙСЬКОЇ ПРОПАГАНДИ У НАЙПОПУЛЯРНІШИХ ТЕЛЕГРАМ-КАНАЛАХ ОДЕЩИНИ (АНАЛІЗ ФРЕЙМІВ). 2021. С. 9. URL: https://doi.org/10.23939/sjs2022.01.080.

27.M. Alex, A. David Python cookbook. Sebastopol, CA : O'Reilly, 2002. 574 p.

28.Rehim R. Effective Python Penetration Testing. Packt Publishing - ebooks Account, 2016. 164 p.

29.Obe R. PostgreSQL: Up and running. Beijing : O'Reilly, 2012. 147 p.

30.Kumar D., Chauhan C. PostgreSQL High Performance Cookbook. Packt Publishing - ebooks Account, 2017. 360 p.

**Додаток А**

**ПРОТОКОЛ ПЕРЕВІРКИ**
**БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ**
**НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ**

Назва роботи: Засіб моніторингу телеграм-каналів під час інформаційної війни

Автор роботи: Шостак Володимир Володимирович

Тип роботи: бакалаврська дипломна робота
(КДР, МКР)

Підрозділ кафедра захисту інформації ФІТКІ
(кафедра, факультет)

**Показники звіту подібності Unicheck**

Оригінальність – 89,8%. Схожість – 10,2%.

Аналіз звіту подібності (відмітити потрібне):

√ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

☐ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

☐ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Каплун В. А.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Шостак В. В.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Войтович О. П.
(підпис) (прізвище, ініціали)

## Додаток Б

## Код програми

main.py

```
import re
import time
from datetime import datetime, timedelta
import telebot
from telebot import types
from telethon.sync import TelegramClient, events
from telethon.tl.functions.channels import JoinChannelRequest
import asyncio
from lib.pgsql import PGDB
import requests

import csv

from telethon.tl.functions.messages import GetDialogsRequest
from telethon.tl.types import InputPeerEmpty
from telethon.tl.functions.messages import GetHistoryRequest
from telethon.tl.types import PeerChannel
from telethon.tl import types

PG = PGDB({
    'host': "localhost",
    'port': "5432",
    'dbname': "telegram",
    'user': "postgres",
    'password': "vovan2001"
  })

api_id = 29690090
api_hash = '94b742a59ad6e60766a399eee6d1b407'
phone = '+380507034877'

bot =
telebot.TeleBot('5571173018:AAF06YWODDMSZ5kFnOK9LI6Plvd_jwW9GEo')

async def parse(client, target_group, chat_s_id):
    #print(1)
    #target_group = 'Test_group'
    try:
        all_participants = await
client.get_participants(target_group)
        for user in all_participants:
            result_user = {}
            result_user['tg_id'] = str(user.id)
            result_user['name'] = str(user.first_name)
            result_user['nickname'] = str(user.username)
            result_user['phone'] = str(user.phone)
```

```
            PG.insert('person', result_user)

    except Exception:
        v=1

    all_messages = []
    offset_id = 0
    limit = 100
    total_messages = 0
    total_count_limit = 1000
    #print(2)
    while total_messages < total_count_limit:
        history = await client(GetHistoryRequest(
            peer=target_group,
            offset_id=offset_id,
            offset_date=None,
            add_offset=0,
            limit=limit,
            max_id=0,
            min_id=0,
            hash=0
        ))

        if history.messages:
            messages = history.messages
            for message in messages:
                all_messages.append(message.to_dict())
                total_messages += 1
            offset_id = messages[len(messages) - 1].id
        if total_messages >= total_count_limit or not
history.messages:
            #print(len(all_messages))
            for message in all_messages:
                #print(message)
                #print(message['peer_id']['_'])
                #time.sleep(10)
                if re.match(message['_'], 'Message'):
                    #print(1)
                    if 'chat_id' in message['peer_id']:
                        # print(message['message'])
                        #print(1)
                        result = {}
                        result['type_author'] = 'User'
                        result_person = {}
                        result_person['tg_id'] =
message['from_id']['user_id']
                        result['author_id'] = PG.check('person',
result_person, key="tg_id")
                        result_channel = {}
                        result_channel['tg_id'] =
message['peer_id']['chat_id']
```

```
                        result['chat_id'] = PG.check('chats',
result_channel, key="tg_id")
                        result['text'] = message['message']
                        result['date'] = message['date']
                        if message['fwd_from'] is not None:
                            print(message['fwd_from'])
                            result['forwarded'] = True
                            result_for_aut = {}
                            if 'channel_id' in
message['fwd_from']['from_id']:

                                print('ch')
                                result_for_aut['tg_id'] =
message['fwd_from']['from_id']['channel_id']

                                result['forwarded_id'] =
PG.check('chats', result_for_aut, key="tg_id")
                                result['forwarded_type'] =
'channel'


                            elif 'user_id' in
message['fwd_from']['from_id']:
                                print('pe')
                                result_for_aut['tg_id'] =
message['fwd_from']['from_id']['user_id']

                                result['forwarded_id'] =
PG.check('person', result_for_aut, key="tg_id")
                                result['forwarded_type'] =
'person'

                        PG.insert('message', result)
                    elif 'channel_id' in message['peer_id']:
                        # print(message['message'])
                        #print(1)
                        result = {}
                        result['type_author'] = 'channel'
                        result_channel = {}
                        result_channel['tg_id'] =
message['peer_id']['channel_id']
                        result['chat_id'] = PG.check('chats',
result_channel, key="tg_id")
                        result['author_id'] = result['chat_id']
                        result['text'] = message['message']
                        result['date'] = message['date']
                        if message['fwd_from'] is not None:
                            result['forwarded'] = True
                            result_for_aut = {}
```

```
                            if
hasattr(message['fwd_from']['from_id'], 'channel_id'):
                                result_for_aut['tg_id'] =
message['fwd_from']['from_id']['channel_id']

                                result['forwarded_id'] =
PG.check('chats', result_for_aut, key="tg_id")
                                result['forwarded_type'] =
'channel'


                            elif
hasattr(message['fwd_from']['from_id'], 'user_id'):
                                result_for_aut['tg_id'] =
message['fwd_from']['from_id']['user_id']

                                result['forwarded_id'] =
PG.check('person', result_for_aut, key="tg_id")
                                result['forwarded_type'] =
'person'

                        PG.insert('message', result)
                        # print(str(message['from_id']['user_id'])
+ ' Написав: ' + message['message'])

            bot.send_message(chat_s_id,'Зчитування завершено')
            break

async def monit(chat_id, list):
    client = TelegramClient(phone, api_id, api_hash)
    await client.start()


    @client.on(events.NewMessage)
    async def handle_new_message(event):
        sender = await event.get_sender()
        message = event.message
        chat = event.message.chat

        #print(chat)
        #print(sender)
        #print(message)
        result_message = {}
        date = message.date + timedelta(hours=3)
        if hasattr(sender, 'first_name') and hasattr(sender,
'last_name'):
            print('per')
            #print("Це користувач")
            result_person = {}
            result_person['tg_id'] = sender.id
            result_person['name'] = sender.first_name
```

```
            result_person['nickname'] = sender.username
            result_person['phone'] = sender.phone
            print(result_person['tg_id'])
            result_message['author_id'] = PG.check('person',
result_person, key="tg_id")
            print(result_message['author_id'])
            result_message['type_author'] = 'User'
        elif hasattr(sender, 'title'):
            print('cha')
            #print("Це канал")
            result_chats = {}
            result_chats['tg_id'] = sender.id
            result_chats['title'] = sender.title
            result_message['author_id'] = PG.check('chats',
result_chats, key="tg_id")
            result_message['type_author'] = 'Channel'
        result_chat = {}
        print(chat)
        result_chat['tg_id'] = chat.id
        result_chat['title'] = chat.title
        result_message['chat_id'] = PG.check('chats', result_chat,
key="tg_id")
        result_message['text'] = message.text
        result_message['date'] = date

        if message.fwd_from is not None:
            result_message['forwarded'] = True
            result_for_aut = {}



            if hasattr(message.fwd_from.from_id, 'channel_id'):
                result_for_aut['tg_id'] =
message.fwd_from.from_id.channel_id

                result_message['forwarded_id'] = PG.check('chats',
result_for_aut, key="tg_id")
                result_message['forwarded_type'] = 'channel'


            elif hasattr(message.fwd_from.from_id, 'user_id'):
                result_for_aut['tg_id'] =
message.fwd_from.from_id.user_id

                result_message['forwarded_id'] =
PG.check('person', result_for_aut, key="tg_id")
                result_message['forwarded_type'] = 'person'



        PG.insert('message', result_message)
```

```
        if list is not None:
            for word in list:
                if word.lower() in message.text.lower():
                    bot.send_message(chat_id, f'Користучач:
{sender.username}\nЧат: {chat.title} | ID: {chat.id}\nЧас:
{date}\nКлючове слово: {word}\n\nПовідомлення:\n{message.text}')
                    break

    await client.run_until_disconnected()

async def add_channel(chat_id, channel_name):
    async with TelegramClient(phone, api_id, api_hash) as client:

        await client(JoinChannelRequest(channel_name))


        #bot.send_message(chat_id, f'Додано канал {channel_name}')
        entity = await client.get_entity(channel_name)
        channel_id = entity.id


        row = PG.get('chats', {'tg_id': channel_id})
        if row is None:
            result = {}
            result['tg_id'] = channel_id
            result['title'] = entity.title
            PG.insert('chats', result)
            bot.send_message(chat_id,'Канал додано, йде зчитування
останніх повідомлень')
            await parse(client, channel_name, chat_id)
        else:
            bot.send_message(chat_id,f'Канал вже існує в базі
даних під id: {row["id"]}')



# Визначаємо обробник команди /start
@bot.message_handler(commands=['start'])
def start(message):
    keyboard = telebot.types.InlineKeyboardMarkup()
    button1 =
telebot.types.InlineKeyboardButton(text='Моніторинг',
callback_data='button1')
    button2 = telebot.types.InlineKeyboardButton(text='Введення
ключових слів', callback_data='button2')
    button3 = telebot.types.InlineKeyboardButton(text='Додавання
каналу', callback_data='button3')
    button4 = telebot.types.InlineKeyboardButton(text='Пов`язані
канали', callback_data='button4')
    button5 = telebot.types.InlineKeyboardButton(text='Пошук в
базі', callback_data='button5')
```

```
    button6 = telebot.types.InlineKeyboardButton(text='Статистика
користувача', callback_data='button6')
    button7 = telebot.types.InlineKeyboardButton(text='Виведення
id', callback_data='button7')
    keyboard.add(button1, button2, button3, button4, button5,
button6, button7)
    bot.send_message(message.chat.id, 'Будь ласка, виберіть
опцію:', reply_markup=keyboard)

# Визначаємо обробник відповідей на кнопки
@bot.callback_query_handler(func=lambda call: True)
def handle_button_click(call):
    if call.data == 'button1':
        function1(call.message.chat.id)
    elif call.data == 'button2':
        bot.send_message(call.message.chat.id, 'Будь ласка,
введіть список:')
        bot.register_next_step_handler(call.message,
process_function2_input)
    elif call.data == 'button3':
        bot.send_message(call.message.chat.id, 'Введіть username
каналу:')
        bot.register_next_step_handler(call.message, function3)
    elif call.data == 'button4':
        bot.send_message(call.message.chat.id, 'Введіть id
каналу:')
        bot.register_next_step_handler(call.message, function4)
    elif call.data == 'button5':
        bot.send_message(call.message.chat.id, 'Введіть id
каналу:')
        bot.register_next_step_handler(call.message, function5)
    elif call.data == 'button6':
        bot.send_message(call.message.chat.id, 'Введіть id
користувача:')
        bot.register_next_step_handler(call.message, function6)
    elif call.data == 'button7':
       function7(call.message)
    elif call.data == 'button7_1':
        bot.send_message(call.message.chat.id, 'Введіть username
користувача:')
        bot.register_next_step_handler(call.message,
function_id_user)
    elif call.data == 'button7_2':
        bot.send_message(call.message.chat.id, 'Введіть title
каналу:')
        bot.register_next_step_handler(call.message,
function_id_channel)

# Функція, яка обробляє введений список у функції function2
def process_function2_input(message):
    chat_id = message.chat.id
```

```
    user_id = message.from_user.id
    input_list = message.text.replace(' |', '|')
    input_list = input_list.replace('| ', '|')
    input_list = input_list.split('|')


    function2(chat_id, user_id, input_list)


@bot.message_handler(commands=['function1'])
def command_function1(message):
    function1(message.chat.id)


def function1(chat_id):

    array = PG.select(sql="select * from user_list where tg_id = "
+ str(chat_id))
    list = array[0]['list']
    asyncio.run(monit(chat_id, list))


def function2(chat_id, user_id, input_list):

    result = {}
    result['tg_id'] = chat_id
    #print(input_list)
    result['list'] = input_list
    if PG.get('user_list', {'tg_id': result['tg_id']}) is None:
        PG.insert('user_list', result)
    else:
        PG.update('user_list', {'tg_id': result['tg_id']}, result)
    bot.send_message(chat_id, 'список додано')

def function3(message):

    chat_id = message.chat.id
    channel_name = message.text
    asyncio.run(add_channel(chat_id, channel_name))



def function4(message):
    chat_id = message.chat.id
    channel_id = message.text


    bot.send_message(chat_id, 'Введіть мінімальний відсоток:')
    bot.register_next_step_handler(message, process_cha_stats,
chat_id, channel_id)
```

```
def process_cha_stats(message, chat_id, channel_id):
    percentage = message.text

    try:
        min_percentage = float(percentage)
    except ValueError:
        bot.send_message(chat_id, 'Неправильний формат проценту.
Будь ласка, введіть число.')
        return

    array = PG.select(sql="select * from message where chat_id = "
+ channel_id)
    if len(array) > 0:
        result = {}
        for row in array:
            if row['forwarded'] is True:
                forwarded_type = row['forwarded_type']
                forwarded_id = row['forwarded_id']
                if forwarded_type not in result:
                    result[forwarded_type] = {}
                if forwarded_id not in result[forwarded_type]:
                    result[forwarded_type][forwarded_id] = 0
                result[forwarded_type][forwarded_id] += 1

        percentage_result = {}
        for forwarded_type, forwarded_ids in result.items():
            percentage_result[forwarded_type] = {}
            for forwarded_id, count in forwarded_ids.items():
                percentage = (count / len(array)) * 100
                percentage_result[forwarded_type][forwarded_id] =
percentage

        filtered_results = []
        for forwarded_type, forwarded_ids in
percentage_result.items():
            for forwarded_id, percentage in forwarded_ids.items():
                if percentage >= min_percentage:
                    filtered_results.append((forwarded_type,
forwarded_id, percentage))

        filtered_results = sorted(filtered_results, key=lambda x:
x[2], reverse=True)

        if len(filtered_results) > 0:
            message = 'Звідки канал пересилає повідомлення:\n'
            for i, (forwarded_type, forwarded_id, percentage) in
enumerate(filtered_results, 1):
                line = f'{i}. {forwarded_type}, id:
{forwarded_id}, процентне відношення: {percentage:.2f}%\n'
                message += line
            bot.send_message(chat_id, message)
```

```
        else:
            bot.send_message(chat_id, 'Немає записів з процентом '
+ str(min_percentage) + ' або більше.')
    else:
        bot.send_message(chat_id, 'Немає записів для обробки.')

    array = PG.select(sql="select * from message where
forwarded_type = 'channel' and forwarded_id = " + channel_id)
    if len(array) > 0:
        result = {}
        for row in array:
            channel = row['chat_id']
            if channel not in result:
                result[channel] = 0
            result[channel] += 1

        percentage_result = {}
        for channel, count in result.items():
            percentage = (count / len(array)) * 100
            percentage_result[channel] = percentage


        message = 'Куди пересилають повідомлення з цього
каналу:\n'
        for channel, percentage in percentage_result.items():
            if percentage >= min_percentage:
                line = f'chat_id: {channel}, процентне відношення:
{percentage:.2f}%\n'
                message += line

        bot.send_message(chat_id, message)
    else:
        bot.send_message(chat_id, 'Немає записів для даного
chat_id')


def function5(message):
    chat_id = message.chat.id
    channel_id = message.text
    bot.send_message(chat_id, 'Введіть потрібну кількість
повідомлень:')
    bot.register_next_step_handler(message, process_seacrch,
chat_id, channel_id)

def process_seacrch(message, chat_id, channel_id):
    total_count = message.text
    key_word = PG.select(sql="select * from user_list where tg_id
= " + str(chat_id))
    if key_word is not None:
        array = PG.select(sql="select * from message where chat_id
= " + str(channel_id) + " ORDER BY id DESC")
```

```
        n = 0
        if array is not None:
            for row in array:
                for word in key_word[0]['list']:
                    if word.lower() in row['text'].lower():
                        bot.send_message(chat_id,
f'{row["type_author"]} id: {row["author_id"]}\nЧат id:
{row["chat_id"]}\nЧас: {row["date"]}\nКлючове слово:
{word}\n\nПовідомлення:\n{row["text"]}')
                        n += 1
                        break
                if n >= int(total_count):
                    break
            bot.send_message(chat_id, 'Зчитування завершено')
        else:
            bot.send_message(chat_id, 'Повідомлень не знайдено')
    else:
        bot.send_message(chat_id, 'Ключових слів не знайдено')


def function6(message):
    chat_id = message.chat.id
    user_id = message.text
    array = PG.select(sql="select * from message where type_author
= 'User' and author_id = " + str(user_id))
    if array is not None:
        result = {}
        for user in array:
            #print(user)
            chat = user['chat_id']
            if chat not in result:
                result[chat] = 0
            result[chat] += 1
            #print(result[chat])

        percentage = {}
        for chat, count in result.items():
            percentage[chat] = (count / len(array)) * 100

        bot.send_message(chat_id, 'Список chat_id та відсоткове
співвідношення:')
        for chat, percent in percentage.items():
            bot.send_message(chat_id, f'Chat ID: {chat}, Відсоток:
{percent}%')


    else:
        bot.send_message(chat_id, 'Користувача не знайдено')

def function7(message):
    keyboard = telebot.types.InlineKeyboardMarkup()
```

```
    button7_1 =
telebot.types.InlineKeyboardButton(text='Користувач',
callback_data='button7_1')
    button7_2 = telebot.types.InlineKeyboardButton(text='Канал',
callback_data='button7_2')
    keyboard.add(button7_1, button7_2, )
    bot.send_message(message.chat.id, 'Будь ласка, оберіть об`єкт
для виведення id:', reply_markup=keyboard)

def function_id_user(message):
    username = message.text
    row = PG.get('person', {'nickname': username})
    if row is not None:
        bot.send_message(message.chat.id, f"ID користувача:
{row['id']}")
    else:
        bot.send_message(message.chat.id, "Користувача не
знайдено")

def function_id_channel(message):
    title = message.text
    row = PG.get('chats', {'title': title})
    if row is not None:
        bot.send_message(message.chat.id, f"ID каналу:
{row['id']}")
    else:
        bot.send_message(message.chat.id, "Канал не знайдено")


# Запускаємо бота
bot.polling()
```

pgsql.py

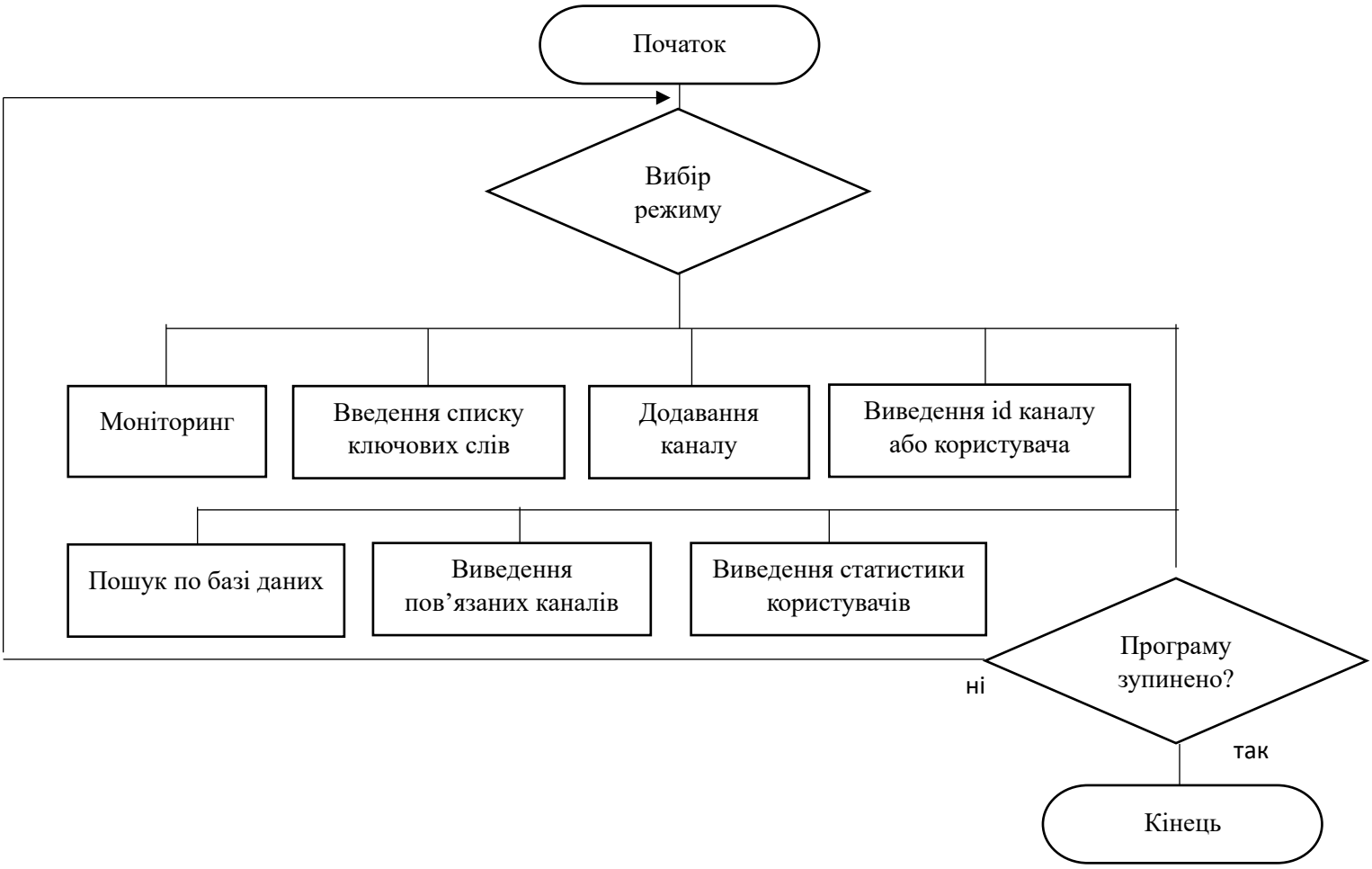
```
import psycopg2
import psycopg2.extras

class PGDB:
     con = None

     def insert(self, table, data, returning='id', conflict='DO
NOTHING'):
          try:
               cursor = self.con.cursor()
               sql = 'INSERT INTO %s (%s) OVERRIDING SYSTEM VALUE
VALUES (%%(%s)s ) ON CONFLICT %s%s;' % (table, ', '.join(data),
')s, %('.join(data), conflict, f' RETURNING {returning}' if
returning else "")
               cursor.execute(sql, data)
               self.con.commit()
               if returning:
```

```
                    try:
                         result = cursor.fetchone()[0]
                    except Exception as e:
                         result = None
               else:
                    result = None
               cursor.close()
          except Exception as e:
               print(f"{e}")
               return None
          return result

     def insertBulk(self, table, data, conflict='DO NOTHING'):
          cursor = self.con.cursor()
          _data = data[0]
          sql = 'INSERT INTO %s (%s) VALUES (%s) ON CONFLICT %s;'
% (table, ', '.join(_data), ', '.join('%%s' for i in
range(len(_data))), conflict)
          cursor.executemany(sql, data)
          self.con.commit()
          cursor.close()
          return True

     def update(self, table, condition=None, data=None):
          cursor = self.con.cursor()
          sql = 'UPDATE %s SET %s WHERE %s;' % (table, ',
'.join(map(lambda i: f"{i}=%s", data)), ' and '.join(map(lambda i:
f"{i}=%s", condition)))
          param = list(data.values()) + list(condition.values())
          cursor.execute(sql, param)
          self.con.commit()
          cursor.close()

     def select(self, table=None, condition=None, fields='*',
sql=None, order=None, limit=None):
          cursor =
self.con.cursor(cursor_factory=psycopg2.extras.DictCursor)
          if sql is None:
               sql = 'SELECT %s FROM %s' % (fields, table)
               if condition:
                    sql += ' WHERE %s' % (' and '.join(map(lambda
i: f"{i}=%({i})s", condition)))
               if order:
                    sql += ' ORDER BY %s' % order
               if limit:
                    sql += ' LIMIT %s' % limit
               cursor.execute(f'{sql};', condition)
          else:
               cursor.execute(f'{sql};')
          data = cursor.fetchall()
          cursor.close()
```

```
          if data:
               result = []
               for row in data:
                    result.append(dict(row))
               return result
          else:
               return None

     def get(self, table=None, condition=None, fields='*',
sql=None, order=None, limit=None):
          result = self.select(table, condition, fields, sql,
order, limit)
          return result[0] if result else None


     def check(self, table, data, key='uuid', returning='id'):
          result = None
          condition = None
          if(type(key) is tuple):
               condition = {}
               for k in key:
                    condition[k] = data[k]
          else:
               condition = {key: data[key]}
          row = self.select(table, condition)
          if row:
               result = row[0]['id'] if 'id' in row[0] else row
          else:
               result = self.insert(table, data, returning)
          return result # if result is not None else
self.check(table, data, key)

     def delete(self, table, condition=None):
          cursor = self.con.cursor()
          sql = 'DELETE FROM %s WHERE %s;' % (table, ' and
'.join(map(lambda i: f"{i}=%s", condition)))
          param = list(condition.values())
          cursor.execute(sql, param)
          self.con.commit()
          cursor.close()

     def Json(self, data):
          return psycopg2.extras.Json(data)
     def Bin(self, data):
          return psycopg2.Binary(data)

     def __init__(self, config):
          self.con = psycopg2.connect(dbname=config['dbname'],
user=config['user'], password=config['password'],
host=config['host'], port=config['port'])
          self.con.autocommit = True
```

# Додаток В

## Схема роботи прогр



# ІЛЮСТРАТИВНА ЧАСТИНА

Засіб моніторингу телеграм-каналів під час інформаційної війни

(Назва бакалаврської кваліфікаційної роботи)

Виконав: студент 4 курсу групи 1БС-19б
спеціальності 125 Кібербезпека
_____ Володимир ШОСТАК
_____ 2023 р.

Керівник: к. т. н., доцент каф. ЗІ
_____ Олеся ВОЙТОВИЧ
_____ 2023 р.

**Інфографіка, яку поширювали в телеграм-каналах**

# Інфографіка опитування


# Список телеграм-каналів, які працюють в інтереси РФ

| № п/п | Назва псевдоукраїнського Телеграм-каналу, який працює в інтересах рф |
|---|---|
| 1 | Крокодил |
| 2 | ШептуньUкраина Война |
| 3 | Легитимный |
| 4 | Анатолий Шарий |
| 5 | Резидент |
| 7 | #МОНТЯН! |
| 8 | ЗеРада |
| 9 | Open Ukraine \| Открытая Украина |
| 10 | UKR LEAKS |
| 12 | Сплетница |
| 14 | Картель |
| 15 | ОЛЬГА ШАРИЙ |
| 17 | Украина.ру |
| 18 | MediaKiller |
| 20 | ТелеДНО |
| 21 | Женщина с косой |
| 22 | Типичная Одесса |
| 23 | Война с фейками |
| 24 | Главное в Херсоне |
| 27 | Чёрный Квартал |
| 28 | НачШтабу |
| 29 | Бунтарь |
| 30 | наглый |
| 31 | Украина. Спецоперация. Мониторинг |
| 32 | Я ♡ Краматорск |
| 33 | Одесский фраер |
| 34 | Запрещённая Украина |
| 35 | Отряд Ковпака |
| 36 | Шкварка News |
| 38 | Кріт СБУ |
| 39 | шбуля UA |
| 40 | Администрация города Мелитополя |
| 41 | Лисичанск. Северодонецк. Рубежное. |
| 42 | Dirty Harry \| Игорь Гомольский |
| 43 | Партия Шария |
| 44 | Черговий ООС |
| 45 | Нетипичное запорожье |
| 46 | Новости Херсонщины |
| 47 | Главное в Геническе |
| 48 | Херсон Life Новости |
| 49 | Украинский формат |
| 50 | Главное в Мелитополе |
| 51 | Тень на плетень |
| 52 | Нетипичное Запорожье |
| 53 | Херсон live |
| 54 | Кривой Рог онлайн |

| № п/п | Назва псевдоукраїнського Телеграм-каналу, який працює в інтересах рф |
|---|---|
| 55 | Николаев live |
| 56 | Трембель Харьков |
| 57 | ХтоШо |
| 58 | 93 бригада "Холодний Яр" \| ОБРАТНАЯ СТОРОНА |
| 59 | Главное в Чернобаевке |
| 60 | Новый Мелитополь |
| 61 | Military photographer |
| 62 | ЗЕландия |
| 63 | ХЕРСОН сегодня |
| 64 | Херсонский Вестник |
| 66 | Главное в Каховке и Новой Каховке |
| 67 | Главное в Скадовске |
| 68 | Чернигов |
| 69 | Сумы |
| 70 | Глухов |
| 71 | Бровары |
| 72 | Шостка |
| 73 | Васильков |
| 74 | Новгород-Северский |
| 75 | Нежин |
| 76 | Обухов |
| 77 | Белая Церковь |
| 78 | Овруч |
| 79 | Ромны |
| 80 | Носовка |
| 81 | Славутич |
| 82 | Бахмач |
| 83 | Прилуки |
| 84 | Ирпень |
| 85 | Березань |
| 86 | Ахтырка |
| 87 | Конотоп |
| 88 | Бердянск ZaVtra |
| 89 | Новый Мелитополь |
| 90 | Южный плацдарм |
| 91 | Энергодарский связной |
| 92 | Энергодар Тудей |
| 93 | Токмак Сегодня |
| 94 | Главное в Пологах |
| 95 | Главное в Энергодаре |
| 96 | Главное в Бердянске |
| 97 | Харьковские антифашисты |
| 98 | Друзья Алексея Селиванова |
| 99 | Администрация города Васильевка |
| 100 | Нетленка |

## Схема роботи блоку зчитування нової інформації

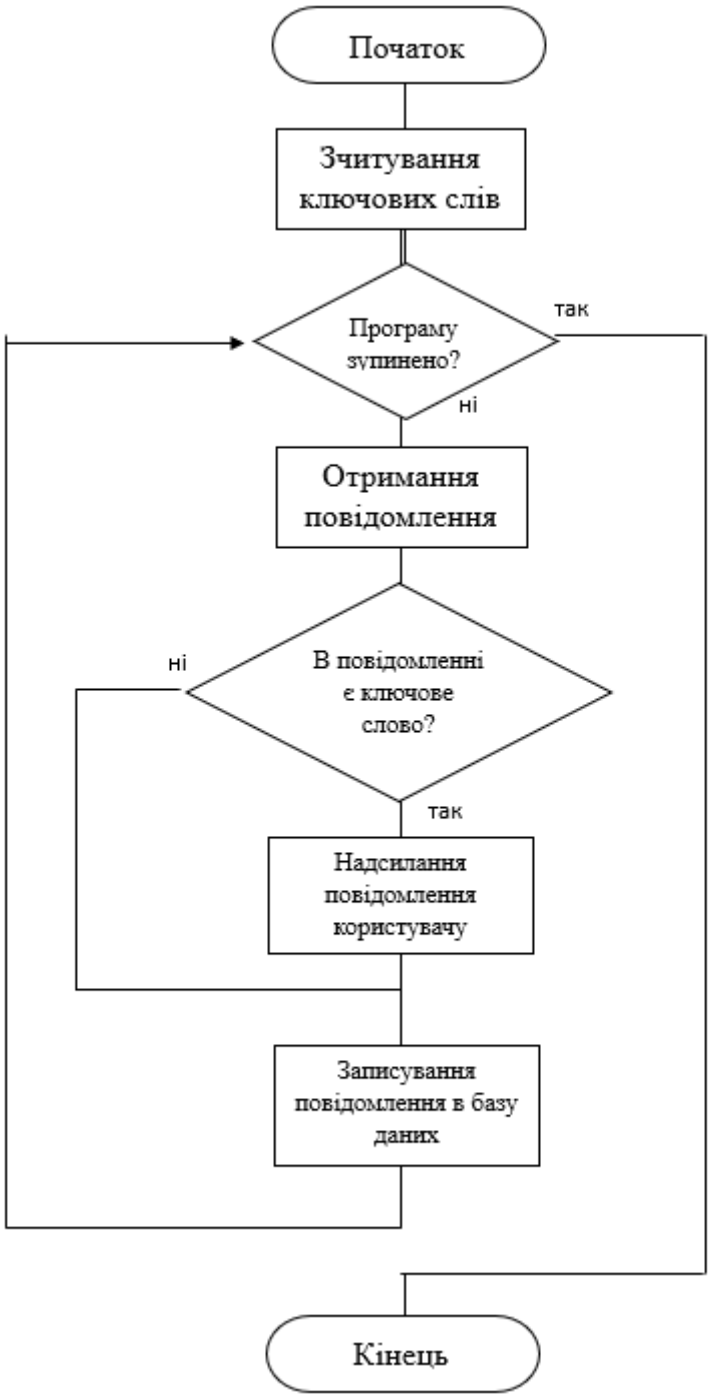


**Схема роботи блоку входу в канал або чат**

**Схема роботи блоку пошуку повідомлень з ключовим словом в базі даних**

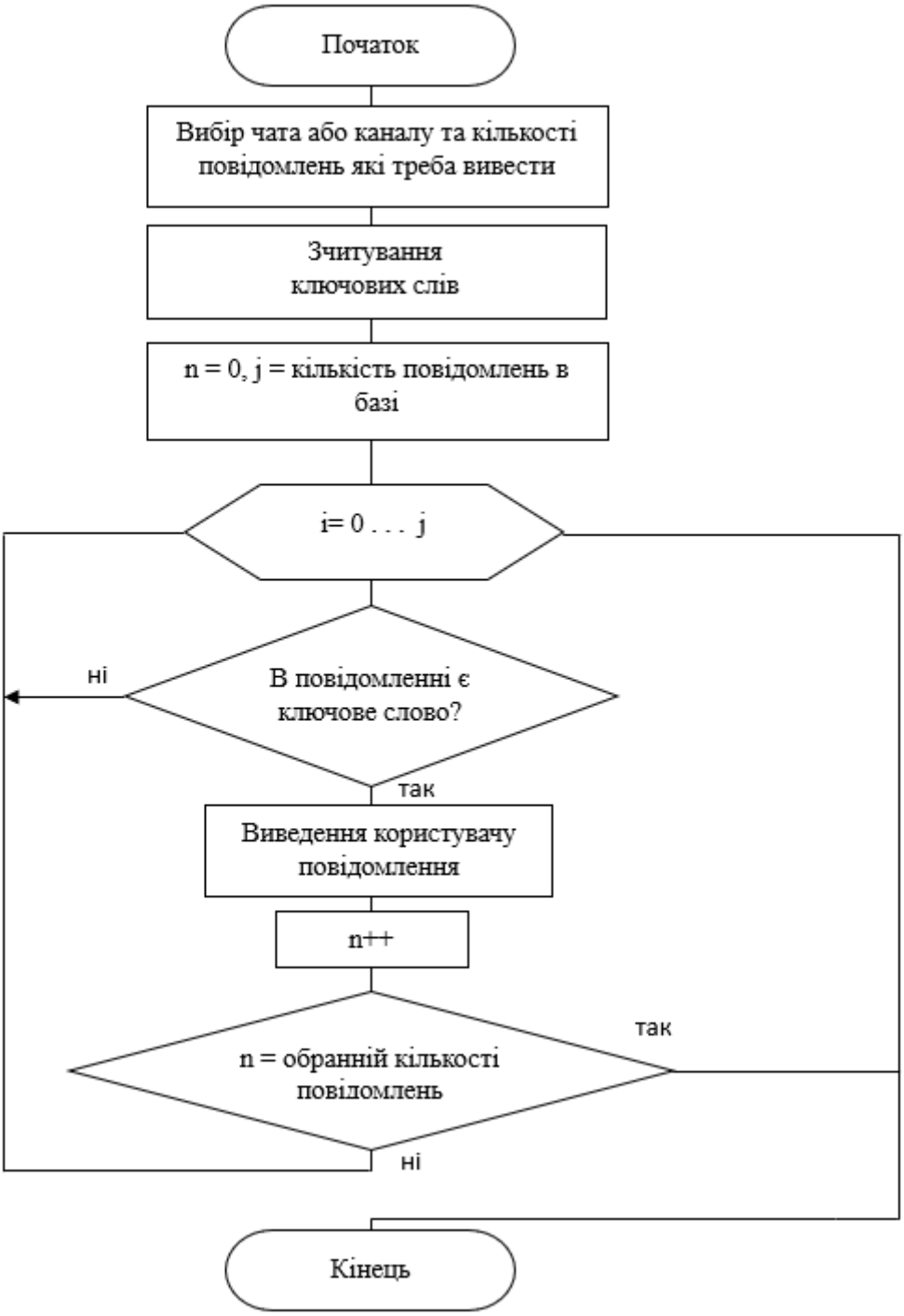
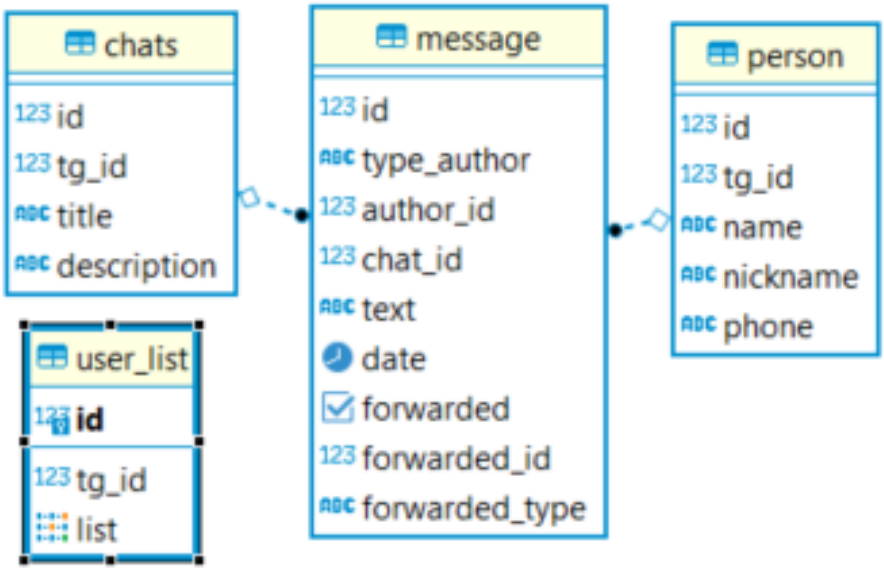


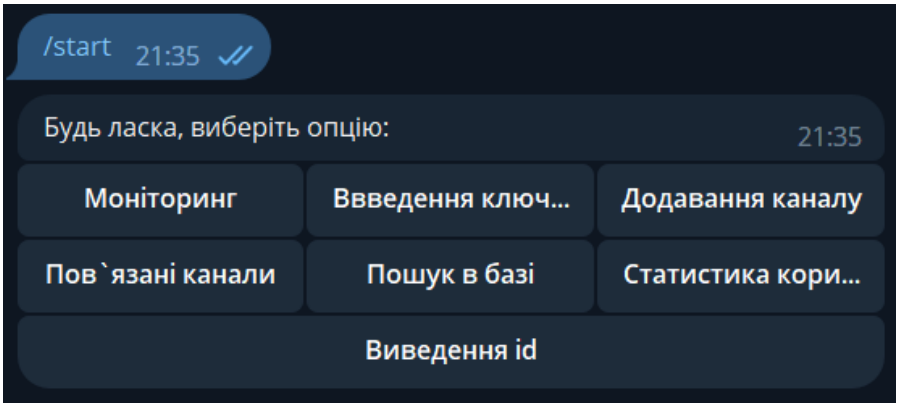
**Схема роботи блоку виведення статистики телеграм аккаунта**

## Схема роботи блоку виведення пов’язаних каналів

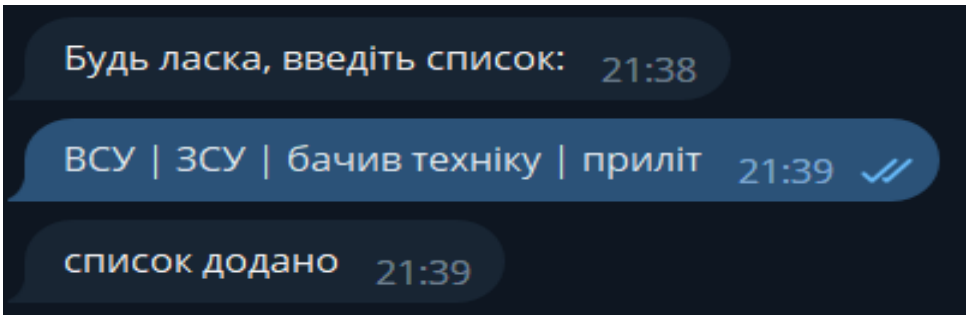
**Структура бази даних**



**Результат виконання команди /start**

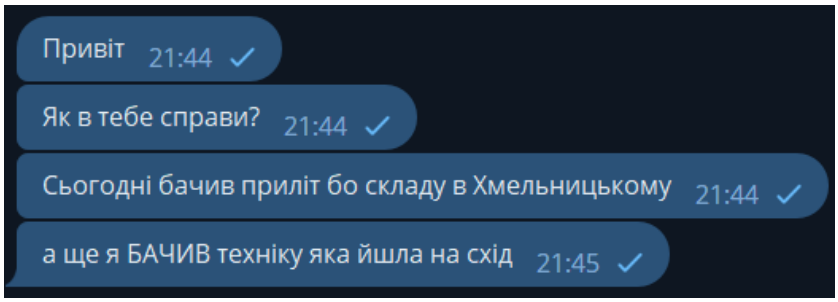


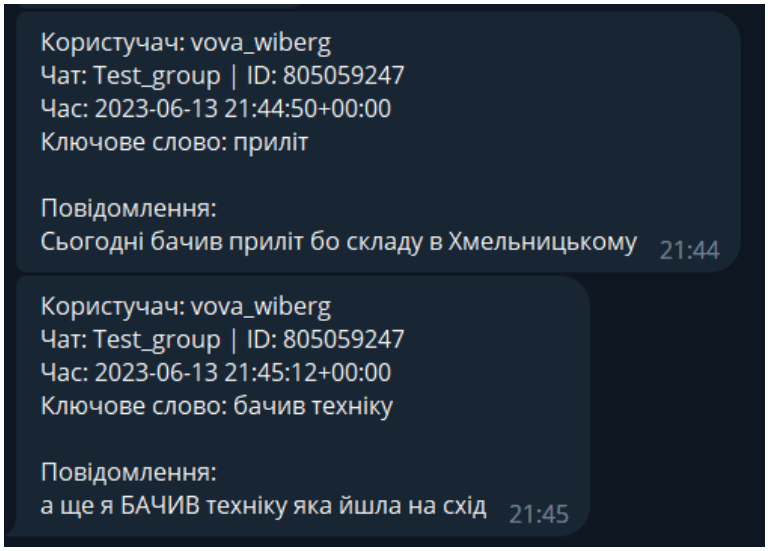
**Результат виконання команди для додавання списку**




| id | tg_id | list |
|---|---|---|
| 1 | 983 661 303 | {ВСУ,ЗСУ,"бачив техніку",приліт} |

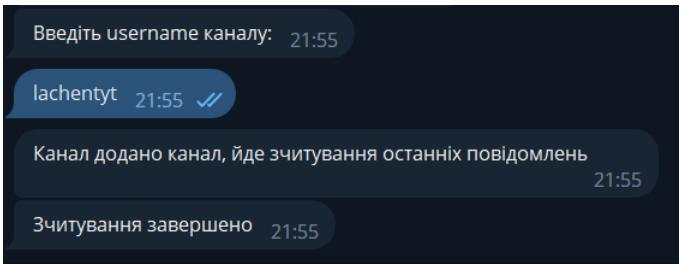
# Результат виконання команди моніторингу



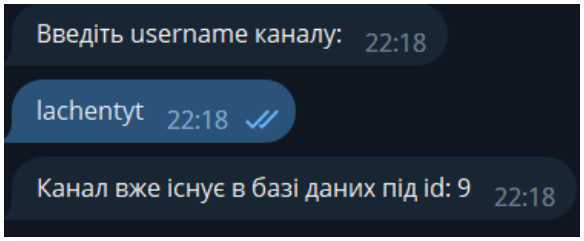

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 393 | 397 | User | 1 | 2 | Привіт | 2023-06-13 21:44:31.000 | [ ] | [NULL] | [NULL] |
| 394 | 398 | User | 1 | 2 | Як в тебе справи? | 2023-06-13 21:44:37.000 | [ ] | [NULL] | [NULL] |
| 395 | 399 | User | 1 | 2 | Сьогодні бачив приліт бо складу в Хмельницькому | 2023-06-13 21:44:50.000 | [ ] | [NULL] | [NULL] |
| 396 | 400 | User | 1 | 2 | а ще я БАЧИВ техніку яка йшла на схід | 2023-06-13 21:45:12.000 | [ ] | [NULL] | [NULL] |

# Результат виконання команди додавання каналу

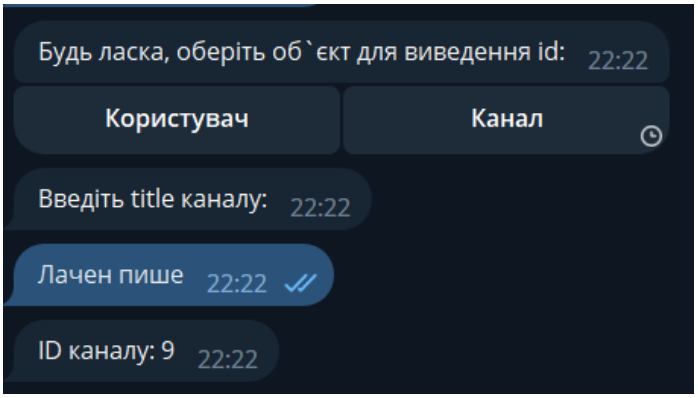



| id | tg_id | title |
|---|---|---|
| 1 | 1 621 065 700 | test_cha |
| 2 | 805 059 247 | Test_group |
| 3 | 1 127 874 961 | [NULL] |
| 7 | 907 832 413 | Test_group2 |
| 8 | 1 246 634 572 | Чат \|\| ВНТУ |
| 9 | 1 536 630 827 | Лачен пише |

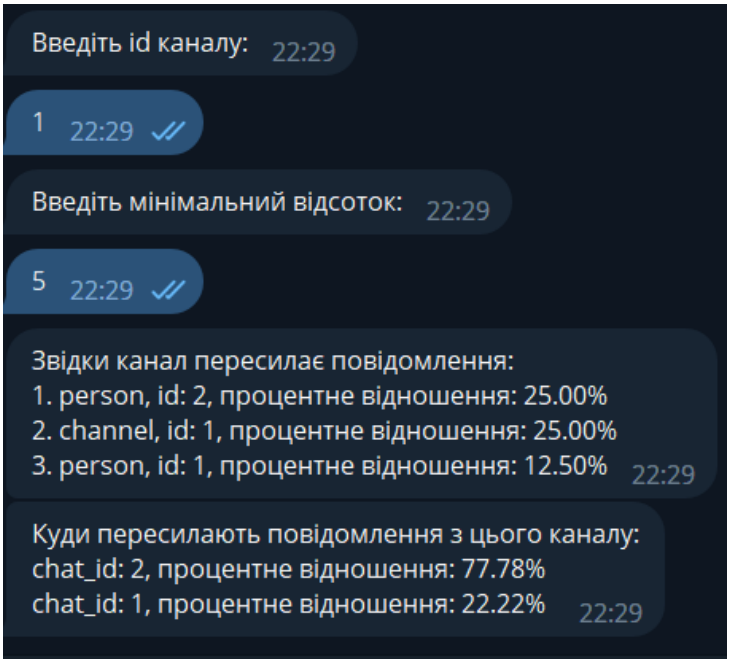
| | id | type_author | author_id | chat_id | text | date | forwarded | forwarded_id | forwarded_type |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 401 | channel | 9 | 9 | Судді Верховного Суду висловили недовіру голові с | 2023-05-16 12:26:20.000 | [ ] | [NULL] | [NULL] |
| 2 | 1 400 | channel | 9 | 9 | Протягом декількох днів ЗСУ звільнили близько 20 і | 2023-05-16 13:57:49.000 | [ ] | [NULL] | [NULL] |
| 3 | 1 399 | channel | 9 | 9 | Частина підривів медійних осіб у РФ - справа рук Г | 2023-05-16 14:23:33.000 | [ ] | [NULL] | [NULL] |
| 4 | 1 398 | channel | 9 | 9 | Компанія Hensoldt уклала контракт на постачання У | 2023-05-16 14:41:01.000 | [ ] | [NULL] | [NULL] |
| 5 | 1 397 | channel | 9 | 9 | Захід підтверджує використання Україною ракет Stc | 2023-05-16 16:13:21.000 | [ ] | [NULL] | [NULL] |
| 6 | 1 396 | channel | 9 | 9 | Прем'єр-міністр Британії Сунак планує обговорити | 2023-05-16 17:12:18.000 | [ ] | [NULL] | [NULL] |
| 7 | 1 395 | channel | 9 | 9 | Відповідальність РФ за війну стане однією з тем сам | 2023-05-16 18:30:07.000 | [ ] | [NULL] | [NULL] |
| 8 | 1 394 | channel | 9 | 9 | ❗ В Білому домі не можуть підтвердити інформаці | 2023-05-16 18:46:49.000 | [ ] | [NULL] | [NULL] |
| 9 | 1 393 | channel | 9 | 9 | ❗ Велика Британія і Нідерланди домовилися про р | 2023-05-16 18:59:33.000 | [ ] | [NULL] | [NULL] |
| 10 | 1 392 | channel | 9 | 9 | Бельгія готова навчати українських пілотів-винищує | 2023-05-16 19:19:29.000 | [ ] | [NULL] | [NULL] |
| 11 | 1 391 | channel | 9 | 9 | Тривогу не ігноруйте, — ОП. | 2023-05-16 20:12:49.000 | [ ] | [NULL] | [NULL] |
| 12 | 1 390 | channel | 9 | 9 | Вибух у Миколаєві. | 2023-05-16 20:14:24.000 | [ ] | [NULL] | [NULL] |
| 13 | 1 389 | channel | 9 | 9 | Миколаїв. Маємо пожежі після вибухів, — мер міста | 2023-05-16 20:37:33.000 | [ ] | [NULL] | [NULL] |
| 14 | 1 388 | channel | 9 | 9 | ❗ Державний секретар США уперше допустив, що І | 2023-05-16 21:29:33.000 | [ ] | [NULL] | [NULL] |
| 15 | 1 387 | channel | 9 | 9 | 200 тис знищених росіян | 2023-05-17 07:54:31.000 | [ ] | [NULL] | [NULL] |
| 16 | 1 386 | channel | 9 | 9 | СБУ викрила мразот та заблокувала онлайн-камери | 2023-05-17 08:01:45.000 | [ ] | [NULL] | [NULL] |
| 17 | 1 385 | channel | 9 | 9 | У Палаті представників США підтримали резолюцію | 2023-05-17 08:19:24.000 | [ ] | [NULL] | [NULL] |
| 18 | 1 384 | channel | 9 | 9 | Швейцарія арештувала активи Жеваго на понад $11 | 2023-05-17 08:41:06.000 | [ ] | [NULL] | [NULL] |
| 19 | 1 383 | channel | 9 | 9 | ❗ 43 держави схвалили угоду про реєстр збитків в | 2023-05-17 09:31:17.000 | [ ] | [NULL] | [NULL] |
| 20 | 1 382 | channel | 9 | 9 | Польща не може надати Україні винищувачі F-16 ос | 2023-05-17 09:59:24.000 | [ ] | [NULL] | [NULL] |
| 21 | 1 381 | channel | 9 | 9 | Південна Корея планує виділити Україні 8 млрд дол | 2023-05-17 10:13:39.000 | [ ] | [NULL] | [NULL] |
| 22 | 1 380 | channel | 9 | 9 | Прем'єр Грузії, яка відновлює сполучення з Росією: | 2023-05-17 11:15:57.000 | [ ] | [NULL] | [NULL] |
| 23 | 1 379 | channel | 9 | 9 | ❗ Франція вивчає можливість постачання Україні | 2023-05-17 11:50:31.000 | [ ] | [NULL] | [NULL] |
| 24 | 1 378 | channel | 9 | 9 | ЗСУ просунулися на 500 м на Бахмутському напрям | 2023-05-17 12:11:22.000 | [ ] | [NULL] | [NULL] |
| 25 | 1 377 | channel | 9 | 9 | Німеччина не планує брати участь в коаліції з поста | 2023-05-17 12:57:49.000 | [ ] | [NULL] | [NULL] |

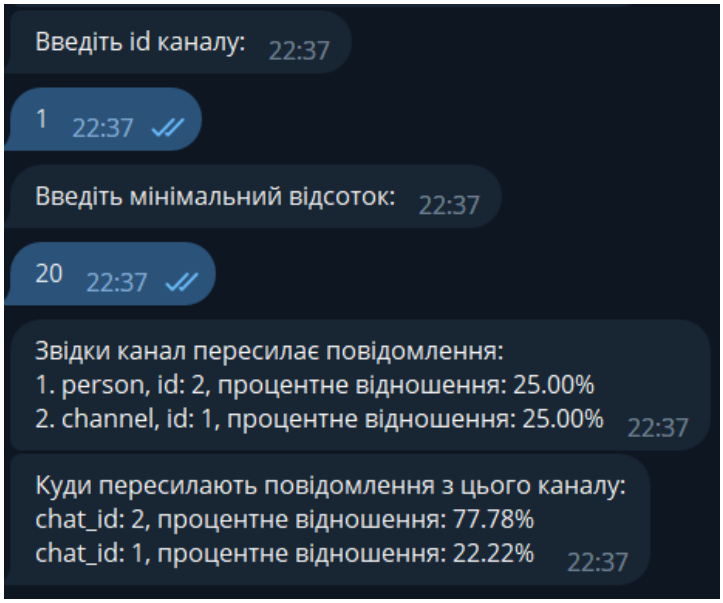


**Результат виконання команди пошуку каналу**

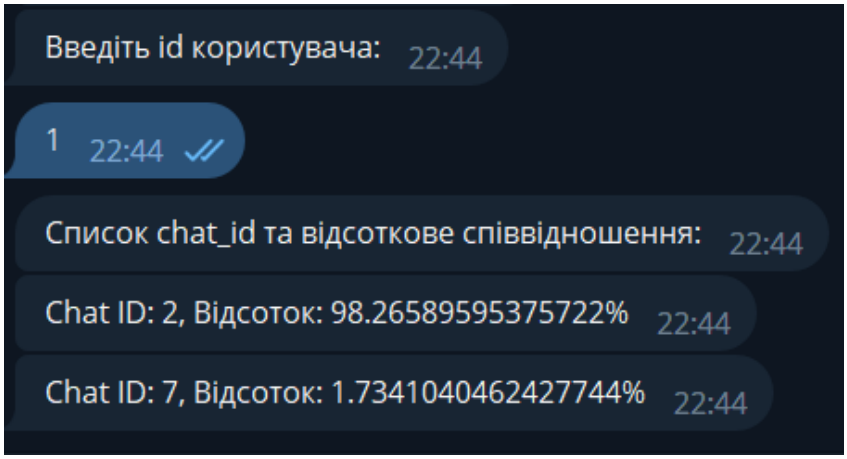


**Результат виконання команди виведення пов’язаних каналів**



**Результат виконання команди виведення статистики користувачів**



# Результат виконання команди пошуку в базі даних