Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Душейко Михайло Юрійович,
студент групи РЗ-181

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Алгоритм збирання і систематизації інформації про користувачів ІМ-додатків

Спеціальність:
125 Кібербезпека

Спеціалізація, освітня програма:
Кібербезпека

Керівник:
Соколов Артем Вікторович,
к.т.н., доцент

Одеса – 2021

Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Спеціальність 125 – Кібербезпека
Освітня програма – Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри КБПЗ
_____
д.т.н., проф. А.А.Кобозєва
____ _____2022р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Душейко Михайло Юрійович

1. Тема роботи: *Алгоритм збирання і систематизації інформації про користувачів ІМ-додатків.*

Керівник роботи: *Соколов А.В. к.т.н., доцент*.

затверджені наказом ректора від „17” 05.2022р. № 168-в.

2. Зміст роботи: *розробка алгоритму та програмного комплексу для збирання і систематизації інформації про користувачів ІМ-додатків*

3. Перелік графічного матеріалу*: блок-схема алгоритму, рисунки інтерфейсу програмного продукту.*

4. Консультанти розділів роботи

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Охорона праці	*доц. Ярова І.А.*		

Дата видачі завдання “_____”_____2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	*Аналіз літератури з теми кваліфікаційної роботи*	*01-03-2022*	*виконано*
2	*Аналіз ІМ-додатків*	*20-03-2022*	*виконано*
3	*Розробка алгоритму*	*05-04-2022*	*виконано*
3	*Розробка програми*	*20-04-2022*	*виконано*
4	*Збирання і систематизація інформації про користувачів ІМ-додатків*	*29-04-2022*	*виконано*
5	*Підготовка пояснювальної записки.*	*16-05-2022*	*виконано*
6	*Підготовка презентації та доповіді*	*23-05-2022*	*виконано*
7	*Попередній захист*	*01-06-2022*	*виконано*
8	*Нормоконтроль*	*18.06.2022*	*виконано*

Здобувач вищої освіти _____ *Душейко М.Ю.*

Керівник роботи _____ *Соколов А.В.*

ЗАВДАННЯ

на розробку розділу “Охорона праці”

Душейко Михайлу Юрійовичу, група РЗ-181

Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій

Кафедра кібербезпеки та програмного забезпечення

Тема роботи *Алгоритм збирання і систематизації інформації про користувачів ІМ-додатків*

Зміст розділу:

1 Аналіз умов праці і вибір основних заходів виробничої безпеки.

2 Аналіз пожежної безпеки. Вибір заходів та засобів пожежної безпеки.

Керівник роботи	Консультант з охорони праці
_____ (А.В. Соколов)	_____ (Ярова І.А.)
«____» _____ 2021 р.	«____» _____2021р.

АНОТАЦІЯ

Дипломна робота на тему «Алгоритм збирання і систематизації інформації про користувачів ІМ-додатків» на здобуття першого (бакалаврського) рівня вищої освіти за спеціальністю 125 – Кібербезпека, спеціалізація, освітня програма: Кібербезпека, містить 14 рисунків, 1 додаток, 11 літературних джерела за переліком посилань. Робота виконана на 52 сторінках загального тексту і 44 сторінках основного тексту.

Основною метою даної дипломної роботи є розробка алгоритму збирання і систематизації інформації про користувачів ІМ-додатків.

Розробка системи базується на основі сучасного месенджеру “телеграм”.

Розроблено алгоритм збирання і систематизації інформації про користувачів ІМ-додатків за допомогою мови Python та новітніх фреймворків, для накопичення інформації використовується база даних MySQL.

Дана програма допомагає накопичити інформацію про користувачів ІМ-додатків та використати її у розслідуваннях з відкритих джерел, чи вбудувати у будь який сервіс котрий може працювати з базами даних.

Результати роботи може використовуватися при розслідуванні з відкритих джерел.

ТЕЛЕГРАМ, OSINT, бази даних, месенджер..

ANNOTATION

Thesis on the topic "algorithm for collecting and systematizing information about users of IM applications" for the first (Bachelor's) level of higher education in specialty 125 in cybersecurity, specialization, educational program: cybersecurity, contains 14 Figures, 1 Appendix, 11 literature sources on the list of references. The work was completed on 52 pages of general text and 44 pages of main text.

The main goal of this thesis is to develop an algorithm for collecting and organizing information about users of IM applications.

The development of the system is based on the modern Telegram messenger.

An algorithm for collecting and organizing information about users of IM applications using Python and the latest frameworks has been developed a MySQL database is used to accumulate information.

This program helps you accumulate information about users of IM applications and use it in investigations from open sources, or embed it in any service that can work with databases.

The results of the work can be used in investigations from open sources.

TELEGRAM, OSINT, databases, messenger

ЗМІСТ

ВСТУП

З кожним роком кількість користувачів Інтернету зростає. До початку жовтня 2020 року 4.9 млрд людей, а це 63.2% жителів Землі, користується Інтернетом. А розмір даних в Інтернеті досягав 2.7 Зеттабайт (1 СБ ~ 1012 ГБ). І щороку-кількість користувачів і пристроїв, підключених до мережі, збільшується на 6% і 10% відповідно. Більша частина цієї інформації є загальнодоступною. Джерела, що посилаються на ці дані або на дані з газет, журналів, радіо і телепередач, публічних звітах уряду, називаються відкритими.

Розвідка на основі відкритих джерел застосовується не тільки державами для захисту своїх інтересів. Розглянемо кілька сценаріїв використання, не пов'язаних із взаємодією країн. OSINT активно використовує дослідниками загроз і вразливостей різного програмного забезпечення. Вони досліджують можливі шляхи поширення шкідливих програм і перешкоджають цьому. Одним з відкритих джерел, для перевірки файлів або програм на наявність шкідливого функціоналу є virustotal.com. у користувача є можливість перевірити файл за контрольною сумою або ж завантажити його, і упевниться в його безпеці. Зловмисники теж використовують OSINT. Попередній аналіз дозволяє дізнатися інформацію про цільову інфраструктуру, про працівників, про клієнта, що дозволяє краще провести атаку. Це одна з причин для того, щоб обмежувати публічну інформацію про компанію. Ця методика також використовується в бізнесі для отримання аналітики. Наприклад, розвитку нових ринків, оцінка ризиків, моніторинг конкурентів, а також дослідження цільової аудиторії і клієнтів для рекламних кампаній, що сприятливо впливає на успішність бізнесу. Не можна забути про журналістів, які використовують відкриті джерела, як матеріали для своїх розслідувань. Наприклад, фонд боротьби з корупцією

використовує виписки з кадастру та інформацію про польоти літаків для своїх розслідувань.

1 АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ ВИКОРИСТАННЯ ІМ МЕРЕЖ У ЗАДАЧАХ ЗБИРАННЯ ТА СИСТЕМАТИЗАЦІЇ ІНФОРМАЦІЇ

1.1. Телеграм загальна інформація

Telegram - Кросплатформенна система миттєвого обміну повідомленнями (месенджер) з функціями VoIP, що дозволяє обмінюватися текстовими, голосовими і відеоповідомленнями, стікерами і фотографіями, файлами багатьох форматів. Також можна здійснювати відео - та аудіозвонки і трансляції в каналах і групах, організовувати конференції, багатокористувацькі групи і канали. За допомогою ботів функціонал програми практично не обмежений. Клієнтські додатки Telegram доступні для Android, iOS, Windows, macOS і GNU/Linux. Кількість щомісячних активних користувачів сервісу станом на січень 2021 року становить близько 500 млн осіб. У серпні 2017 року в своєму Telegram-каналі Павло Дуров заявив, що кількість користувачів месенджера щодня збільшується більш ніж на 600 тисяч.

Крім обміну повідомленнями в діалогах і групах, в месенджері можна зберігати необмежену кількість файлів, вести канали (мікроблоги), створювати і використовувати ботів.

Для месенджера був створений протокол MTProto, що передбачає використання декількох протоколів шифрування. При авторизації і аутентифікації використовуються алгоритми RSA-2048, DH-2048 для шифрування, при передачі повідомлень протоколу в мережу вони шифруються AES з ключем, відомим клієнту і серверу. З переходом на протокол MTProto 2.0 застосовується криптографічний хеш-алгоритм SHA-256.

С 8 жовтня 2013 року в месенджері з'явився режим «секретних» чатів (Secret Chats). Цей режим реалізує шифрування, при якому лише відправник і одержувач володіють загальним ключем (end-to-end шифрування), із

застосуванням алгоритму AES-256 в режимі IgE (англ. Infinite Garble Extension) для пересилаються повідомлень. На відміну від звичайного режиму, повідомлення в секретних чатах не розшифровуються сервером, історія листування зберігається лише на тих двох пристроях, на яких був створений чат.

При обміні файлами можна як відправити файли з пристрою, так і шукати медіаконтент в Інтернеті, в тому випадку, якщо використовується мобільна версія для iOS або Android. Розмір переданих файлів обмежений 2 Гб. Програма використовує систему докачки файлів після обриву зв'язку.

1.2 Процес становлення месенджеру телеграм у суспільстві

У процесі активного розвитку мобільних соцмереж після 2015 року однією з найбільш швидкозростаючих майданчиків для комунікації, як з точки зору аудиторії, так і з точки зору контенту, став Telegram. У статті розглядаються основні технологічні особливості даної платформи і її перспективи як соцмережі, що забирає базовий функціонал у новинних мережевих видань і спільнот «ВКонтакте», що опинилися в правовій кризі.

Виникнення і швидкий розвиток мобільної соцмережі Telegram, побудованої за принципом месенджера, прив'язаного до телефонного номера, почалося в серпні 2013 року, коли засновник "Вконтакте" Павло Дуров представив свій новий проект після конфлікту з керівництвом Mail.ru Group, що змусило його покинути Росію.

Спочатку Telegram функціонував як класичний месенджер зі стандартним для подібних платформ функціоналом. Головною відмінною особливістю Telegram стало шифрування трафіку. Саме безпека передачі даних і можливість створювати всередині аккаунта групи привернуло до месенджера увагу так званого «невидимого Інтернету», для якого технології прихованої комунікації значно спростилися (наприклад, в порівнянні з

проектом I2P, що вимагає установки і настройки спеціального програмного забезпечення).

Це дозволило використовувати Telegram як кіберзлочинцям, так і, наприклад, інтернет-магазинам наркотиків, бірж порнографії і навіть терористам. Звичайні користувачі в свою чергу отримали одну з кращих систем захисту даних і простий інтерфейс, який став вирішальною перевагою для тих, хто хотів спілкуватися в Telegram в режимі простого чату, без заповнюється профілю і збирається в альбоми контенту.

Для генерації аудиторії в Telegram були створені канали-групи, аналогічні спільнотам (паблік)"Вконтакте". Перші канали створювали користувачі для спілкування в груповому чаті, з часом канали відкрили і мережеві ЗМІ, намагаючись таким чином залучити платноспроможну мобільну аудиторію, яку стало все складніше шукати через «Вконтакте». У Рунеті тренд задали "РБК" і "Ехо Москви", і вже до початку 2016 року канал в Telegram став для ЗМІ настільки ж звичним форматом як профіль в Facebook або Twitter.

Проблема полягала тільки в агрегуванні контенту - якщо індивідуальні користувачі публікували свої новини та повідомлення без певного принципу і графіка, ТО ЗМІ, щоб акцентувати увагу саме на свій контент (головним чином - анонси своїх публікацій на сайті), повинні були шукати інструменти автоматизації.

Таким інструментом стали боти-програмно написані скрипти, які самі знали, що де брати і в якому вигляді публікувати в Telegram. Боту було достатньо одноразово дати інструкцію, які матеріали треба транслювати в месенджері, за якими словами виводити користувачам результати пошуку або мультимедіа, і журналістам залишалося тільки займатися своїми прямими обов'язками - інше робив сам бот. Ботами обзавелися багато мережеві ЗМІ, як ті, що раніше працювали з телеграм-каналами (наприклад,

«РБК»), так і ті, яким виявилося достатньо одного лише «бота-розсильника» (наприклад, «Медуза»).

Інтерес до ботів став настільки великий, що Telegram створив окрему керуючу ботами середу - botfather. З моменту появи "батька ботів" писати програму-розсильника і створювати з її допомогою своє власне ЗМІ в соцмережі міг будь-хто, що володіє базовими навичками роботи з програмними інтерфейсами додатків. Для організації автопостін-га сервісу потрібно джерело даних-сайт ЗМІ, новинна стрічка або будь-який періодично оновлюваний RSS-канал. Після цього користувач повинен запросити налаштування у botfather і отримати у платформи авторизаційний код підтвердження - так званий токен. Використовуючи токен, можна як організувати своє ЗМІ в Telegram за кілька хвилин, так і запрограмувати бота на певні відповіді на запити користувачів.

"Роботизація" соцмережі пішла настільки активно, що багато хто з таких «прото-ЗМІ» стали прямими конкурентами традиційних видань, випереджаючи їх як мінімум у швидкості публікації. Так, наприклад, перші фото вбитого в Києві 23 березня екс-депутата Держдуми Дениса Воронен-кова було опубліковано в українському Telegram «ДТП і НП. Києва». Саме з цього джерела фотографії розійшлися по інтернету.

Окремий інтерес і помітний вплив на новинну повістку в Рунеті придбали канали в телеграмі, які принципово відрізняються від традиційних ЗМІ і публікують тільки так звані «зливи» або чутки та історії, перевірка достовірності яких неможлива. Наприклад, в таких каналах як «Методичка» або «Незигарь» кількість передплатників більш ніж в два рази перевищує середнє число передплатників в офіційних каналах традиційних ЗМІ. Середньомісячна аудиторія Telegram (за даними SimilarWeb) в кінці 2016 - початку 2017 року становила понад 110 млн користувачів.

При цьому окремі побоювання, що стосуються Telegrama, зв'язуються з частковим переміщенням підпадає під ті чи інші санкції аудиторії

«Вконтакте» в канали месенджера, де контролювати контент і учасників листування практично неможливо. З точки зору міді-абезпеки це майже така ж "сіра зона» як" невидимий інтернет " в рамках I2P.

Так адміністрація соцмережі тільки в 2016 році заблокувала 78 каналів «Ісламська держава», що діяли в Telegram. Чи будуть вони з'являтися надалі - питання риторичне.

Це означає, що соцмережа стає не тільки комунікативним каналом, вільним від будь-якої потенційної цензури, а й місцем, де досвідчений користувач може знайти будь-які дані, раніше йому не доступні.

1.3 Що таке база даних?

База даних-сукупність даних, що зберігаються відповідно до схеми даних, маніпулювання якими виконують відповідно до правил засобів моделювання даних

У визначеннях найбільш часто (явно або неявно) присутні наступні відмітні ознаки:

БД зберігається і обробляється в обчислювальній системі.

– Таким чином, будь-які позакомп'ютерні сховища інформації (Архіви, бібліотеки, картотеки тощо) базами даних не є.

– Дані в БД логічно структуровані (систематизовані) з метою забезпечення можливості їх ефективного пошуку і обробки в обчислювальній системі.

– Структурованість передбачає явне виділення складових частин(елементів), зв'язків між ними, а також типізацію елементів і зв'язків, при якій з типом елемента (зв'язку) співвідноситься певна семантика і допустимі операції.

– БД включає схему, або метадані, що описують логічну структуру БД у формальному вигляді (відповідно до деякої метамоделі).

– Відповідно до ГОСТ Р ІСО МЕК то 10032-2007, " постійні дані в середовищі бази даних включають в себе схему і базу даних. Схема включає в себе описи змісту, структури і обмежень цілісності, використовувані для створення і підтримки бази даних. База даних включає в себе набір постійних даних, визначених за допомогою схеми. Система управління даними використовує визначення даних у схемі для забезпечення доступу та управління доступом до даних у базі даних ".

– З перерахованих ознак тільки перший є суворим, а інші допускають різні трактування і різні ступені оцінки. Можна лише встановити деяку ступінь відповідності вимогам до БД.

У такій ситуації не останню роль відіграє загальноприйнята практика. Відповідно до неї, наприклад, не називають базами даних файлові архіви, Інтернет-портали або електронні таблиці, незважаючи на те, що вони в деякій мірі володіють ознаками БД. Прийнято вважати, що цей ступінь в більшості випадків недостатня (хоча можуть бути винятки).

1.4 Які бувають бази даних?

Існує величезна кількість різновидів баз даних, що розрізняються за різними критеріями. Наприклад, в»енциклопедії технологій баз даних", за матеріалами якої написаний даний розділ, визначаються понад 50 видів БД.

У класифікацію за моделлю даних зазвичай включають:

- ієрархічний;
- об'єктні або об'єктно-орієнтовані;
- об'єктно-реляційні;
- реляційні;
- мережний;
- функціональний.

Класифікація за середовищем зберігання розрізняє бази даних, що зберігають дані у вторинній пам'яті (»традиційні", англ. conventional database), резидентні (всі дані на стадії виконання знаходяться в оперативній пам'яті) і третинні (англ. tertiary database), що зберігають дані на від'єднуваних пристроях масового зберігання — на основі магнітних стрічок або оптичних дисків. При цьому у всіх класах так чи інакше використовуються всі середовища зберігання, наприклад, для резидентних баз даних СУБД записує в постійну пам'ять журнали предзаписи, а для традиційних баз використовується кеш в оперативній пам'яті.

Також бази даних можуть класифікуватися за вмістом, наприклад, можуть бути географічними, історичними, науковими, мультимедійними. Для деяких форм змісту будуються спеціалізовані СУБД, або додаються спеціалізовані можливості в СУБД загального призначення, серед таких баз даних:

– просторові (англ. spatial database): бази з просторовими властивостями сутностей предметної області, використовуються в геоінформаційних системах;

– тимчасові (temporal database): підтримують будь-який аспект часу, не рахуючи часу, що визначається користувачем.

За ступенем розподіленості бази даних поділяються на централізовані (centralized database) - повністю підтримувані на одному обладнання, і розподілені (distributed database). Серед різноманіття варіантів розподілених баз даних виділяються:

– сегментовані: розділені на частини під управлінням різних екземплярів СУБД за яким-небудь критерієм;

– тиражовані (репліковані; англ. replicated database): одні й ті ж дані рознесені під управління різних екземплярів СУБД;

– неоднорідні (heterogeneous distributed database): фрагменти розподіленої бази в різних вузлах мережі підтримуються засобами більше однієї СУБД.

Можливі змішані варіанти, наприклад, для однієї і тієї ж розподіленої бази для великих об'єктів використовується сегментування, а для невеликих — реплікація.

За способами організації зберігання можуть виділятися циклічні бази даних (записують нові дані замість застарілих), потокові бази даних.

2 РОЗРОБКА АЛГОРИТМУ ЗБИРАННЯ ТА СИСТЕМАТИЗАЦІЇ ІНФОРМАЦІЇ ПРО КОРИСТУВАЧІВ TELEGRAM

2.1 Парсинг

Для пошуку і збирання даних про користувачів месенджеру телеграм будемо використовувати парсинг

Парсинг (Parsing) – це прийняте в інформатиці визначення синтаксичного аналізу. Для цього створюється математична модель порівняння лексем з формальною граматикою, описана однією з мов програмування

Для ефективного парсингу нам треба визначити де у телеграммі збираються найбільша кількість користувачів.

Діалог - оформлення і функціональність діалогів не дуже відрізняється від інших месенджерів. Присутні стандартні можливості: голосові повідомлення, відео повідомлення, прикріплення файлів, стікери, GIF-анімації та емодзі, можливість побачити, що співрозмовник прочитав повідомлення, попередній перегляд посилань і т. д. Цей варіант нам не підходить так як треба буде чекати доки користувач сам нам напише щоб його спарсити.

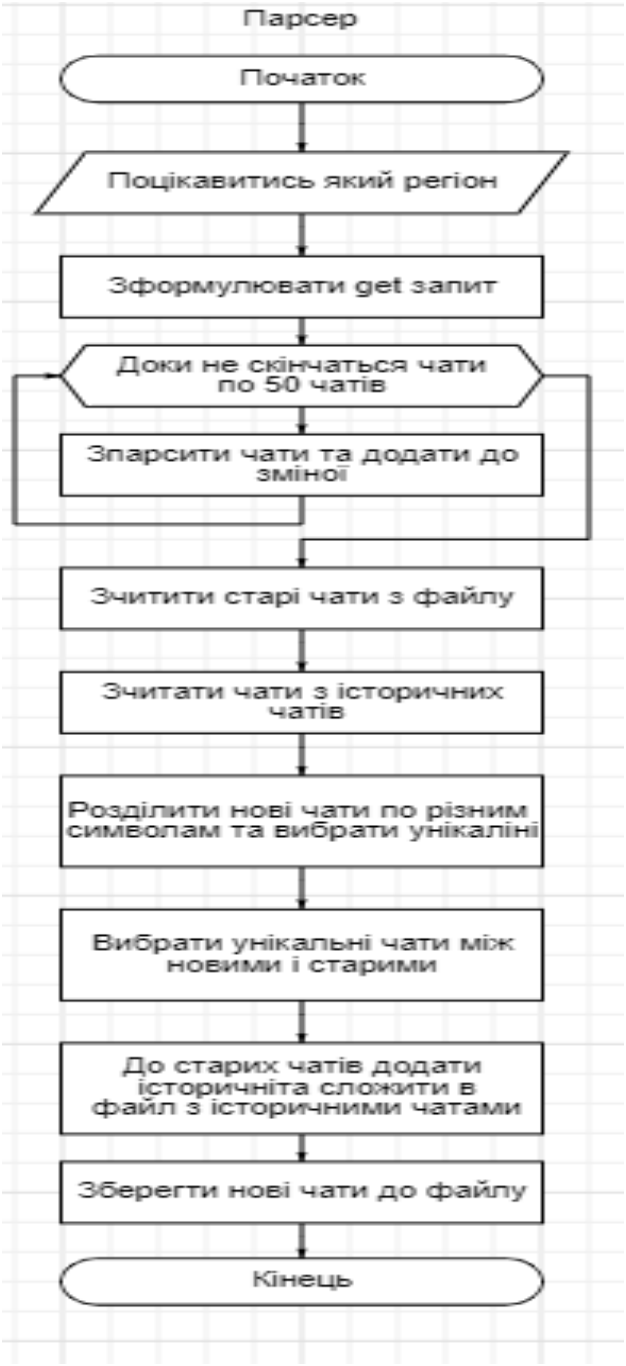
Група - є можливість організовувати групи до 200 учасників, починаючи з листопада 2015 року, супергрупи до 1000 учасників, з 14 березня 2016-супергрупи до 5000 учасників. З 30 Червня 2017 року розмір супергруп збільшився до 10000 учасників, з 30 Січня 2018 року — супергрупи до 100000 учасників. Також є закриті групи в котрі можно додатися лише за посиланням-запршенням.

Канал - Найважливішою особливістю, що відрізняє Telegram від своїх конкурентів, можна вважати інструмент комунікації в форматі публічних каналів. Такий спосіб дозволяє автору або групі авторів ділитися інформацією з необмеженим колом осіб з мінімальною дистанцією між

читачем і контентом, але і зберігати при цьому анонімність. Дуже часто до канала прикріплюють лінк на коментарі що представляє собою звичайну групу котру ми розляли вище.

З цього виходить що ми маємо парсити групи та канали з чатами у телеграмі щоб зібрати я умога більше користувачів.

Для пошуку великого пулу посилань на групи та канали я звернувся до гугла та знайшов там ститистику від бота-адміністратора груп. Тому треба було розібратися як складати дані з сайту до мого файлу групами для



подальшого просмотру користувачів.

Рисунок 2.1 – Блок-схема

Розробимо блок-схему алгоритму парсингу чатів (Рисунок 2.1), та розробимо програму мовою Python для цього алгоритму

Перша частина в нас цікавиться з якого регіону ми бажаємо збирати чати, тому виділимо це в окрему функцію :

```
def settings_parser():
    print('''____Choose region:____
|en - English       |
|ru - Русский       |
|uk - Українська    |
|uz - O‘zbek        |
|tr - Türkçe        |
|id - Indonesia     |
|es - Español       |
|hi - □□□□□         |
|az - Azərbaycan    |
|all - all          |
----------------------''')
    return input('>>> ')
```

Наступним кроком буде парс обраних чатів з сайту та пошук унікальних з них. Запишемо цю функцію наступним чином:

```
def parse_from_combot(patch):
    limit = 50
    counter = 0
    chats = []
    region = settings_parser()
    print('Parse chats start')
    # requests to site
```

```
        while True:
                                                    temp_chats =
get(f'https://combot.org/api/chart/{region}?limit={limit}&offset={limit *
counter}').json()
            if temp_chats == []:
                break
            for chat in temp_chats:
                chats.append(chat['u'])
            counter += 1
        # split chats name '_'
        temp_chats = copy(chats)
        for chat in temp_chats:
            for word in chat.split('_'):
                chats.append(word)
        history(patch, chats)
```

Після чого оновимо чати у файлі та перезапишемо старі у інший файл у програмі це буде виглядати настуаним чином:

```
    def history(patch, chats):
        with open(patch, 'wb') as dump_file, open(all_chats_patch, 'rb') as
chat_storage:
            all_chats = load(chat_storage)
            chats = set(chats) - set(all_chats)
            all_chats = list(all_chats) + list(chats)
            dump(chats, dump_file)
        with open(all_chats_patch, 'wb') as chat_storage:
            dump(all_chats, chat_storage)
        print(f'Parse succesfull! {len(chats)}')
```

2.2 Формування бази даних

Так як планується зберігання великої кількості даних та для збільшення швидкодії програми я обрав базу даних MySQL, так як вона підтримує одразу декілька запитів до себе.

Так як ми будемо зберігати користувачів групи та повідомлення користувачів необхідно створити таблиці для них.

Для створення таблиці для користувачів котра буде зберігати поля:

1 Id – унікальний індефікатор кожного користувача
2 Is_bot – чи являється користувач ботом;
3 Is_deleted – чи заблокований акаунт;
4 First_name – ім’я користувача;
5 Username – нік користувача;

використовуючи мову SQL зробимо запит до бази даних який буде виглядати наступним чином:

```
CREATE TABLE `users` (
	`id` BIGINT(20) NULL DEFAULT NULL,
	`is_bot` TEXT NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
	`is_deleted` TEXT NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
	`first_name` TEXT NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
	`username` TEXT NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
)
COLLATE='utf8mb4_0900_ai_ci'
ENGINE=InnoDB
;
```

Для створення таблиці для груп яка буде зберігати поля:

1 Id – унікальний індефікатор кожної групи;

2 Title – назва групи;

3 Username – нік групи;

використовуючи мову SQL зробимо запит до бази даних який буде виглядати наступним чином:

```
CREATE TABLE `chats` (
        `id` BIGINT(20) NOT NULL DEFAULT '0',
        `title` TEXT NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
        `username` TEXT NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
        PRIMARY KEY (`id`) USING BTREE
)
COLLATE='utf8mb4_0900_ai_ci'
ENGINE=InnoDB
;
```

Для створення таблиці для повідомлень яка буде зберігати поля:

1 User_d – унікальний індефікатор кожного користувача

2 Chat_id – унікальний індефікатор кожної групи

3 Forward_from_chat_id – id чату з котрого переслано повідомлення, “---” якщо повідомлення не переслане

4 Forward_from_chat_message_id - id повідомленя з котрого переслано повідомлення, “---” якщо повідомлення не переслане

5 Forward_text - текст повідомленя з котрого переслано повідомлення, “---” якщо повідомлення не переслане

6 Message_text – текст повідомлення котре залишив користувач

7 Message_id – id повідомлення користувача

8 Message_date – число та час залишеного повідомлення

використовуючи мову SQL зробимо запит до бази даних який буде виглядати наступним чином:

```
CREATE TABLE `messages` (
```

```
    `user_id` BIGINT(20) NOT NULL DEFAULT '0',
    `chat_id` BIGINT(20) NOT NULL DEFAULT '0',
    `forward_from_chat_id` BIGINT(20) NULL DEFAULT NULL,
    `forward_from_chat_message_id` BIGINT(20) NULL DEFAULT NULL,
    `forward_text` TEXT NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
    `message_text` TEXT NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
    `message_id` BIGINT(20) NULL DEFAULT NULL,
    `message_date` TEXT NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci'
)
COLLATE='utf8mb4_0900_ai_ci'
ENGINE=InnoDB
;
```

Також створемо допоміжну таблицю в котрій будемо позначати відносність користувача до певної групи та його статус у групі, коли він приєднався. Таблиця буде мати такі поля:

1 User_id – унікальний індефікатор кожного користувача
2 Chat_id – унікальний індефікатор кожної групи
3 Status – статус користувача у групі (адмін, підписник, тощо)
4 Joined_date – час коли користувач приєтнався до групи

використовуючи мову SQL зробимо запит до бази даних який буде виглядати наступним чином:

```
CREATE TABLE `connect` (
    `user_id` BIGINT(20) NOT NULL DEFAULT '0',
    `chat_id` BIGINT(20) NOT NULL DEFAULT '0',
```

```
        `status` TEXT NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
        `joined_date` TEXT NOT NULL COLLATE 'utf8mb4_0900_ai_ci'
)
COLLATE='utf8mb4_0900_ai_ci'
ENGINE=InnoDB
;
```

Основною функцією котра буде взаємодіяти з базую даних у програмі:

```
def send_to_mysql(sql):
        connection = pymysql.connect(host=host_db, user=user_db, password=password_db, database=database_db,
                                     port=port_db, cursorclass=pymysql.cursors.DictCursor)

    with connection:
        with connection.cursor() as cursor:
            cursor.execute(sql)
            connection.commit()
            return cursor.fetchall()
```

вона відкриває підключення передає базі даних запит та при необхідності повертає відповідь.

Для заповнення таблиці чатів використовуємо :

```
def chat_info(app, chat):
    chat_info = app.get_chat(chat)

        sql = f"INSERT INTO chats (id, title, username) VALUES ({chat_info.id}, '{chat_info.title}', '{chat_info.username}');"

    send_to_mysql(sql)
```

```
    return chat_info.id
```

Вона формує запит до бази даних про чат та повертає в основну програму id чату

За допомогою цієї функції отримаємо інформацію про користувачів чату та як вони відносяться до певного чату та формує відповідний запит до бази даних:

```
def members_info(app, chat, chat_id):
                    cht = app.get_chat_members(chat, limit=app.get_chat_members_count(chat))
    for member in cht:
        try:
                sql = f"INSERT INTO users (id, is_bot, is_deleted, first_name, username, status) VALUES ({member.user.id}, '{member.user.is_bot}', '{member.user.is_deleted}', '{member.user.first_name}', '{member.user.username}', '{member.user.status}');"
            print(sql)
            send_to_mysql(sql)
        except Exception as e:
            print(member.user.id, str(e))
        sql = f"INSERT INTO connect (user_id, chat_id, status, joined_date ) VALUES ({member.user.id}, {chat_id}, '{member.status}', '{member.joined_date}')"
        print(sql)

        send_to_mysql(sql)
```

За допомогою цієї функції отримаємо історію групи та записуємо ввідповідні дані до бази даних:

```
def message_info(app, chat):
```

```
                msgs = app.get_chat_history(chat, limit=app.get_chat_history_count(chat))
        for message in msgs:
            print(message)
            try:
                try:
                    forward_from_chat_id = message.reply_to_message.forward_from_chat.id
                    forward_from_chat_message_id = message.reply_to_message.forward_from_message_id
                    forward_text = message.reply_to_message.text.replace('\n', '').replace("""""", '')
                except:
                    forward_from_chat_id = 0
                    forward_from_chat_message_id = 0
                    forward_text = '---'
                    text = message.text.replace('\n', '')
                    sql = f"INSERT INTO messages (user_id, chat_id, forward_from_chat_id, forward_from_chat_message_id, forward_text, message_text, message_id, message_date) VALUES ({message.from_user.id}, {message.chat.id}, {forward_from_chat_id}, {forward_from_chat_message_id}, '{forward_text}', '{text}', {message.id}, '{message.date}');"
                    print(sql)
                    send_to_mysql(sql)
            except Exception as e:
                print(str(e))
```

2.3 Взаємодія додатку з телеграмом

Для взаємодії додатку з телеграмом використовується бібліотека для Python – Pyrogram при першлому запуску вона генерує токен по даним акаунта та додає його до папки щоб в наступні запити не генерувати його.

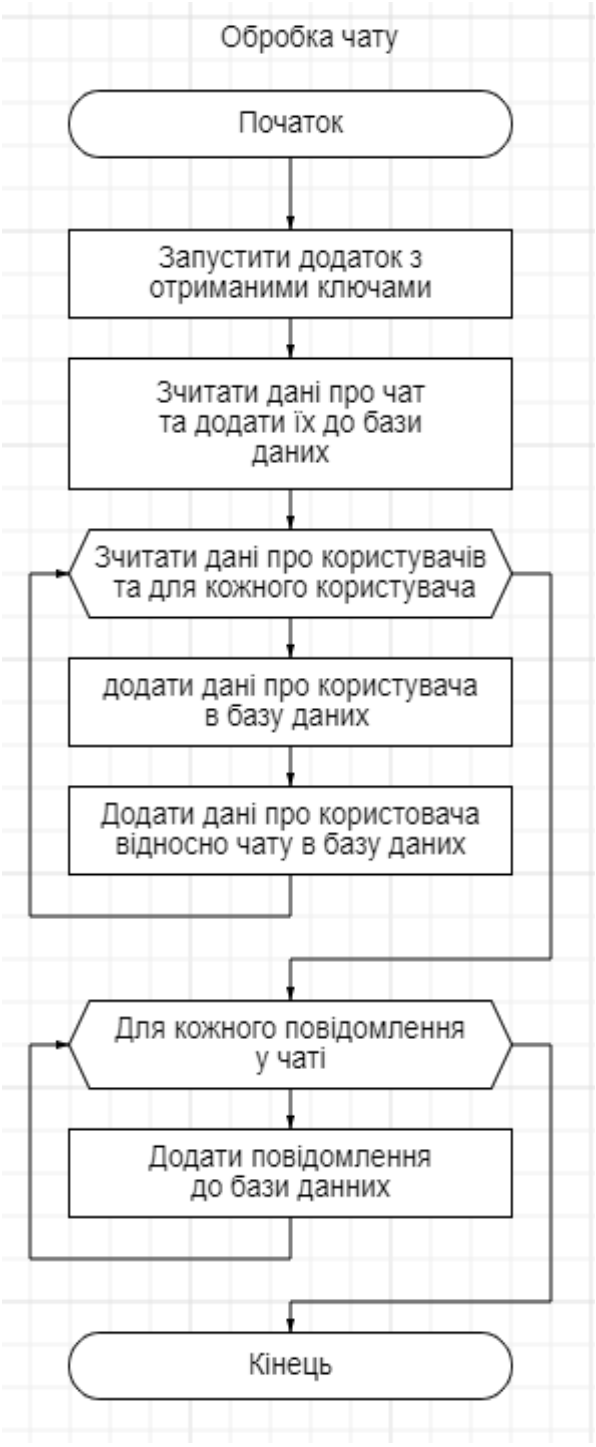


Рисунок 2.2 - Блок-схема

За допомогою наступної підпрограми ми можемо передати номер аккаунту та чат з якого ми запускаємо додаток для взаємодії з телеграму та чат котрий ми хочемо додати до нашої бази він визиває функції котрі ми розгляли вище та починає додовати чат, користувачів, повідомлення. Докладніше на рис. 2.2.

```
import sys
from configparser import ConfigParser
from pyrogram import Client
from work_database import chat_info, members_info, message_info
chat = sys.argv[1]
acc_id = sys.argv[2]
config = ConfigParser()
config.sections()
config.read(f'./account/{acc_id}/api.ini')
with Client(str(acc_id), config['API']['api_id'], config['API']['api_hash'], workdir=f'./account/{acc_id}') as app:
    try:
        chat_id = chat_info(app, chat)
        members_info(app, chat, chat_id)
        message_info(app, chat)
        print(f'{chat} add succesfull!')
    except Exception as e:
      print(str(e)
```

2.4 Основна програма

Основна програма слугує “прослойкою” між всіми основними функціями. Воно опитує користувача на відтворення нової бази чатів та бази

даних та запускає у відокремлених процесах взаємодію з телеграмом (алгоритм наведено у блок схемі, рис. 2.3).

За допомогою мови пайтон ми також можемо бачити котрі акаунти працюють прямо зараз, у випадку якщо акаунт було заболаковано програма прибирає його з черги

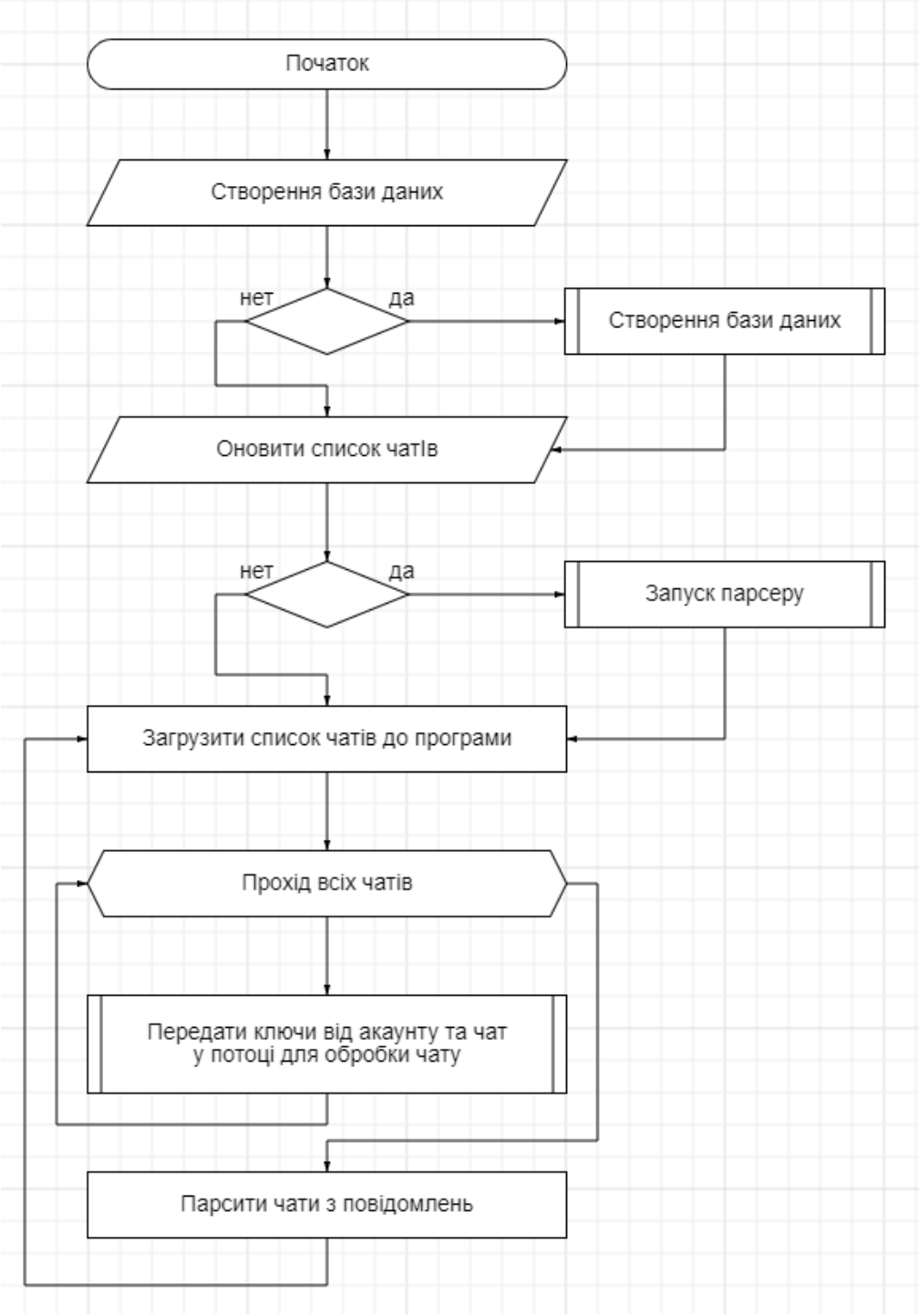


Рисунок 2.3 - Блок-схема

Код підпрограми:

```
def start_stream(chat, acc_id):
    global acc_numbers, stream_acc
    try:
```

```
                answer = os.popen(f'python ./parse_chats_to_db_stream.py {chat} {acc_id}').read()
                if '420 FLOOD_WAIT_X' in answer:
                    acc_numbers -= 1
                    stream_acc = stream_acc[:-1]
                    print(
                        f'{chat}, account: {acc_id}, BAN : account will be removed from the list')
            except Exception as e:
                print(str(e))
            finally:
                stream_acc[acc_id] = False
        def start():
            while True:
                temp = 0
                for chat in list(dump_file(chats_patch)):
                    while stream_acc[temp % acc_numbers]:
                        sleep(1)
                        temp += 1
                    stream_acc[temp % acc_numbers] = True
                    print(stream_acc)
                    # start_stream(chat, temp % acc_numbers)
                        Thread(target=start_stream, args=(chat, temp % acc_numbers), daemon=True).start()
                    sleep(timeouts - acc_numbers * 20)
                    temp += 1
                parse_from_messages_database(chats_patch)
```

Також ми бачимо що існує певний таймаут між запусками процесів це зроблено задля того щоб телеграм не заблокував акаунти за підозрілу активність

3 ІМПЛЕМЕНТАЦІЯ ТА ТЕСТУВАННЯ УТИЛІТИ ДЛЯ ЗБИРАННЯ ТА СИСТЕМАТИЗАЦІЇ ІНФОРМАЦІЇ ПРО КОРИСТУВАЧІВ TELEGRAM

3.1 Запуск програми

При першому запустці програма пропонує створення бази даних (рис. 3.1

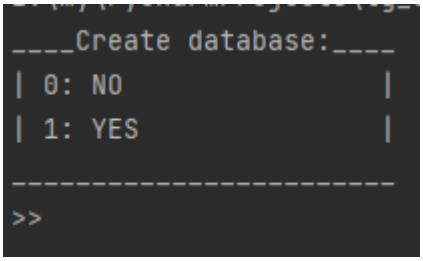


Рисунок 3.1 - Запит на створення бази даних

Пропонується зібрати перші чати та регіон цих чатів (рис. 3.2)

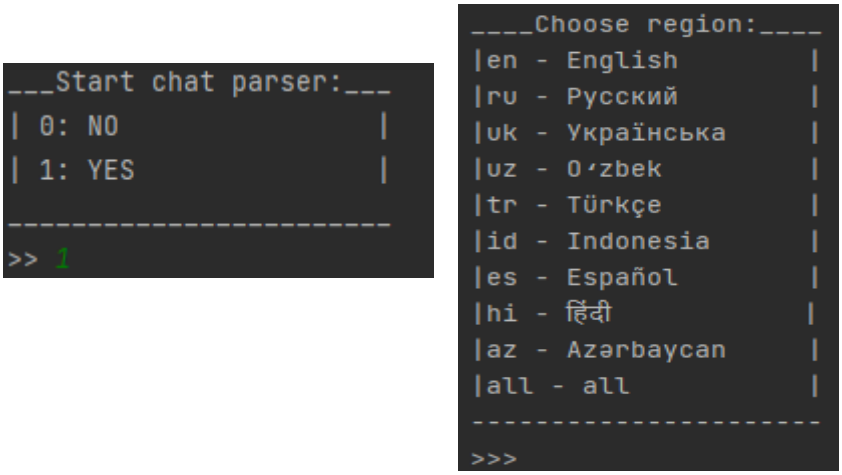
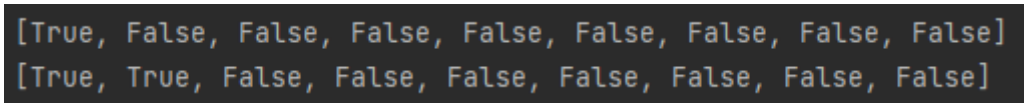


Рисунок 3.2 - Запит на початок парсингу

На рис. 3.3 бачимо, як починають працювати відокремлені процеси для



взаємодії з телеграмом та базою даних.

Рисунок 3.3 - Управління процесами

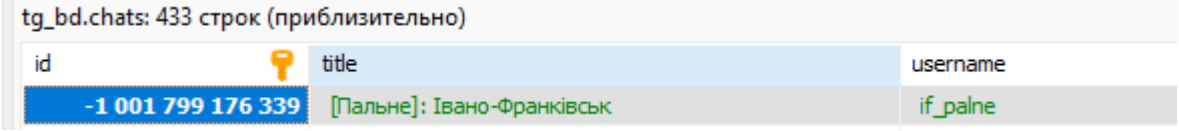


Рисунок3.4 –Кількість груп

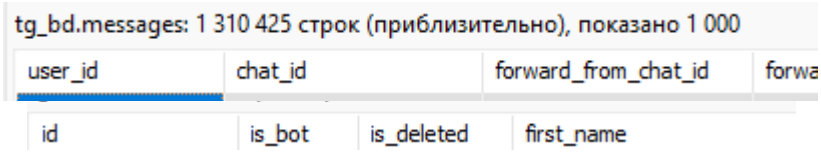


Рисунок 3.5 Кількість користувачів

За 2 години вдалося зібрати приблизно 433 групи (Рисунок 3.4), 1310450 повідомлень (Рисунок 3.6), та 30555 користувачів (Рисунок 3.5).

3.2 Разробка підпрограми для виводу інформації

Після того як ми зібрали та систематизували дані про користувачів ІМ-додатку телеграм, ми можемо запросити інформацію про будь якого користувача у нашій базі.

Щоб автоматизувати запити до нашої бази даних, а не прописувати їх кожен раз власноруч я створив телеграм бота котрий автоматизує цей процес, та видає відповідь у зрозумілому виді.

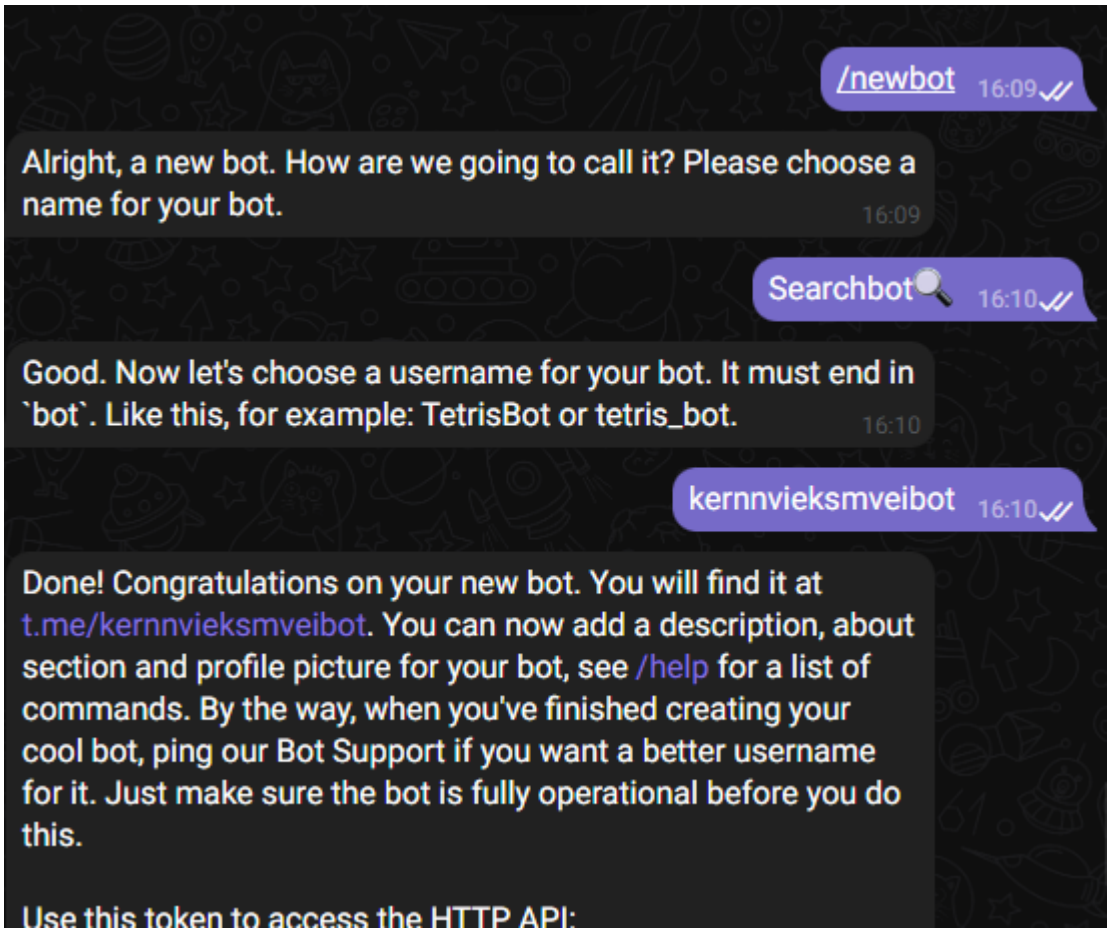


Рисунок 3.7 - Отримання токену

Для цього нам необхідно зайти до офіційного телеграм бота телеграму та попросити видати новий ключ, пишемо йому наступні команди та отримаємо ключ Рисунок 3.7.

Отриманий токен додаємо до бота та за допомогою такої простої функції чекаємо доки боту напишуть.

```
@bot.message_handler(content_types=['text'])
def start(message: telebot.types.Message) -> None:
    if message.text in ('/start', '/help'):
        bot.send_message(chat_id=message.chat.id, text=answer.start)
    else:
        try:
```

```
                user_id = database_interaction.input_func(message.text)
                print(user_id)
                user_id = int(user_id)
                                        bot.send_document(chat_id=message.chat.id,
document=open(f'./output/{user_id}.html' ,'rb'))
            except Exception as ERROR:
                bot.send_message(chat_id=message.chat.id, text=ERROR)
```

Якщо боту прислали id користувача то він одразу викликає функцію яка достає дані про цього користувача з бази даних.

```
    def _getdatafromid(id: int) -> str:
        """Return info about user from id"""
        try:
                sql_request_messages = f"SELECT chat_id, message_text,
message_date, message_id FROM tg_bd.messages WHERE
tg_bd.messages.user_id = {id};"
                sql_request_chats_member_info = f"SELECT chat_id, status,
joined_date  FROM tg_bd.connect WHERE tg_bd.connect.user_id = {id};"
            sql_request_user_info = f"SELECT * FROM tg_bd.users WHERE
tg_bd.users.id = {id};"

            messages = _send_to_mysql(sql_request_messages)
            chats = _send_to_mysql(sql_request_chats_member_info)
            user = _send_to_mysql(sql_request_user_info)[-1]
            print(user)
            for chat in chats:
                chat['title'] = _send_to_mysql(f'SELECT title FROM tg_bd.chats
WHERE tg_bd.chats.id = {chat["chat_id"]};')[-1][
                'title']
```

```
            for message in messages:
                message['title'] = \
                    _send_to_mysql(f'SELECT title FROM tg_bd.chats WHERE tg_bd.chats.id = {message["chat_id"]};')[-1][
                        'title']
            print(1)
            create_file_with_info(user, chats, messages)
            print(1)
            return user['id']
        except Exception as ERORR:
            return ERORR
```

Якщо прислали юзернейм користувача то за допомогою бази даних він перетворюється на айді та визивається функція котру ми згадали раніше.

```
    def _getdatafromusername(username: str) -> str:
        """Return info about user from username"""
            sql_request = f"SELECT id FROM tg_bd.users WHERE tg_bd.users.username = '{username}';"
        try:
            id = _send_to_mysql(sql_request)[0]['id']
            return _getdatafromid(id)
        except Exception as Error:
            return Error
```

У результаті бот формує html сторінку удобну для перегляду та повертає її.

3.3 Вивід інформації

Опишемо блоки про користувача рис.3.8, блок про чати рис. 3.9, блок

USER INFO

id	is_bot	is_deleted	first_name	username	status
765320071	False	False	Pavel krg	Pavel_azia	UserStatus.LAST_WEEK

про повідомлення рис. 3.10, блок про повідомлення котрі користувач надсилав у певні чати рис. 3.10, повідомлення з номером телефону рис.3.11, повідомлення з домашньою адресою рис. 3.12

Рисунок 3.8 - Блок про користувача

Рисунок 3.9 - Блок про чати

MESSAGE INFO

chat_id	message_text	message_date	message_id	title
-1001412764840	Еду в город сейчас	2022-05-23 05:31:10	7972	Попутчики Абай - Карагандинская область
				Попутчики

CHAT INFO

chat_id	status	joined_date	title
-1001412764840	ChatMemberStatus.MEMBER	2020-11-27 04:31:06	Попутчики Абай - Карагандинская область
-1001467207747	ChatMemberStatus.MEMBER	2020-10-02 15:07:43	Абай Диалог

Рисунок 3.10 – Блок про повідомлення

-1001412764840	В 7:05 выезжаю Абай-Караганда 19:00 Караганда-Абай4 места. +77053143145	2021-07-25 20:05:35	5664	Попутчики Абай - Карагандинская область

Рисунок 3.11 - Повідомлення з номером телефону

Також цей користувач залишив свою домашню адресу рис. 3.12.

А почему не по Абая-Энгельса 33??? 4 месяца тепла осталось, ничего не делалось, только в плане акимата уже 2 года.

Абая 33 тоже стало холоднее, батареи теплые, за что 6000...

Здравствуйте, с той же проблемой, что происходит, батарпи еле теплые, дома всего 19, у нас все года было 26! За что добавили +1000 тенге??? Абая 33. Почему как мороз - холодрыга, а когда на улице тепло - батареи огонь

Рисунок 3.12 – Повідомлення з домашньою адресою

4 ОХОРОНА ПРАЦІ

5

Об’єктом дослідження є робоче місце із персональним комп’ютером. Оскільки тема кваліфікаційної роботи – «Алгоритм збирання і систематизації інформації про користувачів ІМ-додатків», а на комп’ютері виконується основна робота.

Кожна організація має вживати заходів виробничої безпеки, але не для всіх вони будуть однаковими. Для того, щоб здійснити вибір заходів безпеки необхідно проаналізувати умови праці робітника. В якості об’єкта проектування для аналізу умов праці буде розглянуто робоче місце OSINT розслідувача.

Об'єктом проведення досліджень було обрано бюро інтернет розслідувань. Приміщення бюро знаходиться на другому поверсі. Загальна площа робочого приміщення 60 м^2, висота - 2,7 м, приміщення має 2 вікна та жалюзі, які регулюють напрямок світлового потоку, щоб він не засвітлював монітори. Кількість осіб, що працюють в приміщенні - 6. Згідно [20] на кожного працюючого в приміщенні виходить: 60/6 = 10 м^2 робочої площі. При висоті приміщення більше 2,5 м всі нормативи по розмірах і забезпеченню робочої площі в офісі дотримані. В бюро розташовано 6 робочих місць із комп'ютерами, два графічних планшети, проектор. Також розміщено 2 окремих письмових стола, одна шафа для зберігання паперів. На стелі розташовані 9 світильників із люмінесцентними лампами білого світла зі значеннями світлового потоку по 1200 лм кожна.

Наступним кроком в аналізі умов праці є визначення небезпечних і шкідливих виробничих факторів (НШВФ) за нормативним документом [21].

Температура повітря в холодний і теплий період року повинна бути в межах + 20 – 25 °С, відносна вологість 60 – 40 % при швидкості руху повітря не більш 0,2 м/с. У приміщеннях необхідно дотримуватися

необхідних параметрів мікроклімату за допомогою вентиляційних і опалювальних систем.

У приміщенні повітря робочої зони повинне відповідати встановленим вимогам до приміщень із незначними надлишками тепла від відеотерміналів і пристроїв відображення інформації. Повітря в приміщеннях повинне бути очищене від шкідливих речовин, пилу й мікроорганізмів. Тобто для того щоб підтримувати мікроклімат у приміщені в нормі треба регулярно робити провітрювання та вологе прибирання.

Шум – один із найбільш поширених несприятливих фізичних факторів навколишнього середовища, який може негативно впливати на організм людини. Коли мова йде про вплив шуму, то зазвичай основну увагу приділяють стану органу слуху, так як слуховий аналізатор у першу чергу сприймає звукові коливання і подразнення його є адекватним дії шуму на організм. Навіть шуми невеликої інтенсивності, які присутні протягом усього робочого дня на організм людини впливають негативно. Допустимий рівень шуму для роботи розслідувача має не перевищувати 50 дБ за [21]. Нормовані рівні шуму забезпечуються шляхом використання малошумного устаткування, застосуванням звуковбирних матеріалів для облицювання приміщень. Для захисту від шуму потрібно забезпечити кожного працівника навушниками, або зробити шумоізоляцію.

Робоча поза – це основне положення тіла працівника в просторі. Робота за комп’ютером передбачає пряме положення тулуба сидячи, однак присутні і нахили корпусу більше 30° до монітора з частотою в межах норми за [21]. Площа, відведена на одне робоче місце має становити не менше 6 кв.м., а об’єм – не менше 20 куб.м.. Робочий стіл повинен мати стабільну конструкцію. Його мінімальні розміри 160 × 90 см. Крісло й площа стола повинні регулюватися по висоті на 42 – 55 см і 65 – 85 см відповідно. Тип робочого крісла вибирають залежно від тривалості роботи: при тривалої –

масивне крісло, при короткочасної – крісло легкої конструкції, яке вільно відсувається.

При розміщенні робочих місць з персональними комп'ютерами відстань між робочими столами з відеомоніторами (у напрямі тилу поверхні одного відеомонітора і екрану іншого відеомонітора) повинна бути не менше 2,0 м, а відстань між бічними поверхнями відеомоніторів - не менше 1,2 м.

Екран відеомонітора повинен знаходитися від очей користувача на відстані 600 – 700 мм, але не ближче 500 мм з урахуванням розмірів алфавітно-цифрових знаків і символів. Модульними розмірами робочої поверхні столу для ПК, на підставі яких повинні розраховуватися конструктивні розміри, слід вважати: ширину - 800, 1000, 1200 і 1400 мм, глибину - 800 і 1000 мм при нерегульованої його висоті, рівній 725 мм.

Необхідно забезпечити достатньо рівномірне розподілення яскравості на робочій поверхні монітору, а також в межах навколишнього простору через підвищене навантаження на орган зору під час роботи із ПК.

Напруженість аналізаторних функцій. Особливо при роботі з інформацією порушується зір та слух.

Емоційна та інтелектуальна напруженість. Відбувається при вирішенні важких завдань в умовах дефіциту часу та інформації з підвищеною відповідальністю. Для вирішення цієї проблеми потрібно зробити перерву , або підвищити заробітну плату.

Дрібні стереотипні рухи кистей та пальців рук. За [22] нормована кількість однакових рухів у межах від 20000 – 40000 разів за зміну. Для запобігання виникнення больових відчуттів у ліктьових суглобах, передпліччях, зап’ястях, кистях та пальцях рук у тих працівників, які регулярно працюють із комп’ютерною «мишкою» можна використовувати опори для кистей, що повторюють їх переміщення, а також спеціальну

гімнастику для рук, заняття якимось видом спорту, пов’язаним із великим навантаженням на руки, наприклад, плаванням.

Тривала робота за комп’ютером може призвести до проблем із органами дихання. Протягом довгого періоду роботи комп'ютера корпус і плати виділяють в повітря ряд шкідливих речовин. Навколо комп’ютера утворюється статичне поле, яке притягує пил, який осідає в легенях. Найбільш тяжким проявом алергії є анафілактичний шок. Комп'ютер є досить серйозним джерелом низки алергенів. Наприклад, корпус монітора, нагріваючись до 50 - 55 ° С починає виділяти в повітря пари тріфенілфосфата. Крім монітора нагрівається і материнська плата, блок живлення, процесор, відеокарта, які так само можуть виділяти в навколишнє середовище шкідливі органічні та неорганічні речовини (фтор-, хлор-, фосфоровмісні). Крім того, в комп'ютері є дуже багато місць, де накопичується пил і бруд, розмножуються мікроби і грибки.

Оскільки професія OSINT розслідувача передбачає роботу за комп’ютером, то 90% тривалості зосередження – це увага на монітор. За [21] таке значення фактору перевищує допустиме нормоване, тож виникає необхідність захисту оболонки ока від пересихання та подальшого погіршення зору. Хвороба «сухого» ока є досить поширеним явищем. Ознаками захворювання є почервоніння очей, сльозотеча, світлобоязнь. Щоб захистити свій зір потрібно робити перерву та вправи , або також купити спеціальні комп’ютерні окуляри

Заходи захисту від ураження електричним струмом передбачають використання їх при нормальному режимі роботи електроулаштування і підтримують їх безпеку в аварійних умовах.

Захист від ураження електричним струмом повинен забезпечуватися: конструкцією електроулаштування, технічними засобами і засобами захисту, організаційними заходами.

Одним з важливих заходів попередження електротравматизму є проведення інструктажів з охорони праці з питань електробезпеки. Доцільно проводити інструктажі у формі співбесіди. З власного досвіду можу сказати, що нерідко працівники мають дуже віддалені поняття про небезпеку електроструму.

Тому під час проведення вступного інструктажу варто більш детально зупинитись на наступних питаннях:

– в чому саме полягає небезпека електроструму;

– як впливає дія струму на тіло людини та можливі наслідки для життя та здоров’я;

– фактори, за яких людина може потрапити під дію струму;

– перша допомога при ураженні електричним струмом;

– як вберегти себе та оточуючих від вражаючих факторів струму при стихійних лихах, аваріях.

Аналіз пожежної безпеки і вибір заходів і засобів пожежної безпеки.

У певних умовах електрична енергія легко переходить у теплову і це може викликати пожежі і вибухи. Пожежна небезпека електрообладнання, електронних приладів і персональних комп’ютерів, пов’язана з використанням спалимих матеріалів: гуми, пластмас, лаків, олій.

Правила пожежної безпеки в Україні затверджені наказом Міністерства внутрішніх справ України, зі змінами, які періодично вносяться відповідними наказами.

Вимоги до пожежної безпеки на підприємстві неухильно повинен дотримуватися кожен співробітник, а організаційна складова при цьому покладається на посадових осіб за відповідним рішенням керівництва і прописується в посадових інструкціях і положеннях по структурним підрозділам. Відповідальний за пожежну безпеку обов’язково повинен пройти навчання за програмою пожежно-технічного мінімуму в органах державного пожежного нагляду і отримати відповідне посвідчення.

Загальні відомості про причини появи небезпечних ситуацій допоможуть організувати комплекс протипожежних заходів.

Причини виникнення пожежі в офісі:

– несправне електричне обладнання викликає пожежу на робочому місці;

– електророзподільне та освітлювальне обладнання;

– опалювальне обладнання;

– використання несправного електрообладнання;

– перегрів внутрішніх комплектуючих комп’ютера;

– перевантаження електромережі;

– порушення правил користування електроприладами;

Звернемося до нормативного документу НАПБ Б.03.002-2007 «визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою». Він містить за класифікацією 5 категорій, для офісних приміщень визначається категорія «В» , яка є пожежонебезпечною.

Для протидії пожежі потрібно розробити план та шляхи евакуації, мати первинні засоби та пожежну сигналізацію.

До первинних засобів пожежогасіння відносяться: вогнегасники, пожежний інвентар (покривала з негорючого теплоізоляційного полотна, грубововняної тканини або повсті, ящики з піском, бочки з водою, пожежні відра, совкові лопати) та пожежний інструмент (гаки, ломи, сокири тощо).

Головною умовою для успішної ліквідації пожежі є швидке повідомлення пожежно-рятувальної служби про виникнення загоряння. Для виклику пожежної команди на кожному об’єкті має бути телефонний або радіозв’язок. Для швидкого повідомлення про пожежу облаштовують електричну пожежну сигналізацію, яка виявляє займання на початковій стадії, що забезпечує успішну боротьбу з вогнем. До автоматичних систем пожежної сигналізації належать: теплові (реагують на підвищення

температури), димові(реагують на появу диму), світлові й комбіновані сповіщувачі (здатні одночасно реагувати на підвищення температури у навколишньому середовищі і появу диму).

Вогнегасники є найнадійнішими первинними засобами пожежогасіння. Законом України «Про пожежну безпеку» визначаються вимоги до експлуатації вогнегасниками. У службових приміщеннях на видних та легкодоступних місцях встановлюють порошкові або вуглекислотні вогнегасники, які повинні мати сертифікат відповідності, паспорт, не порушені пломби.

Автоматичні спринклери призначені для гасіння всіх видів пожеж і навіть для гасіння пожеж, які були навмисно створені для зловмисних цілей.

Плани евакуації – один із обов'язкових елементів документації. Вони розробляються для визначення основних та запасних маршрутів евакуації, їхнього графічного відображення з метою візуального сприйняття та запам'ятовування персоналом і відвідувачами на об'єктах, вивішуються на видних місцях та відпрацьовуються у тих будівлях і спорудах, (окрім житлових будинків), де знаходяться одночасно на поверсі більш ніж 10 осіб персоналу.

Евакуаційні виходи залежать від числа офісних працівників і площі приміщення. Евакуаційний вихід повинен бути позначений таблом «Вихід», а по маршруту евакуації повинні розташовуватися напрямні стрілки, які виведуть назовні. Двері на шляхах евакуації повинні відкриватися у напрямку виходу з будівлі. Так вони не будуть перешкоджати виходу людей. Коридори не повинні бути захаращені меблями, сміттям. При цьому, якщо офіс розташований в житловому будинку, то його евакуаційний вихід повинен бути ізольований від житлової частини будинку. Евакуаційні шляхи та виходи слід завжди утримувати вільними, нічим не захараченими. Шляхи евакуації, що не мають природного освітлення, у разі наявності людей повинні постійно освітлюватись електричним світлом.

Висновки: в ході виконання розділу «Охорона праці» було розглянуто, які небезпечні та шкідливі виробничі фактори при експлуатації персонального комп'ютера можуть впливати на працівника та яких рекомендацій потрібно дотримуватися, запропоновано заходи захисту і профілактики.

Для запобігання виявлення погіршення самопочуття та здоров’я користувача, у процесі роботи з комп'ютером необхідно дотримуватися правильного режиму праці та відпочинку. Під час перерв в роботі рекомендується виконувати гімнастику для очей для підтримки загального м'язового тонусу і профілактики кістково-м'язових порушень.

Розглянули інструкції для працівників при виникненні аварійних ситуацій, таких як загоряння електропроводки чи задимлення.

ВИСНОВКИ

Незважаючи на тривалу історію, розвідка за відкритими джерелами тільки починає свій розвиток, у зв'язку з бурхливим зростанням обсягу інформації. Дана область є затребуваним видом дослідження матеріалів. Для тих, хто її використовує, відкривають додаткову інформацію: для працівників – можливість більше дізнатися про майбутнього роботодавця, для бізнесу – Аналіз ринку і клієнтів. Але з іншого боку, кожна людина повинна розуміти, яку інформацію варто повідомляти публічно і розуміти, що може стати загальнодоступною інформацією.

ПЕРЕЛІК ПОСИЛАНЬ

1. Telegram MTProto API Framework for Python. URL: docs.pyrogram.org.
2. PyMySQL’s documentation.URL: https://pymysql.readthedocs.io./
3. Конопльов Д. Е. Telegram як нове середовище комунікації в ЗМІ та соцмережах, проблемне поле медіаосвіти, 2017.
4. Суходолов О. П., Бичкова О. М. Цифрові технології та наркозлочинність: проблеми протидії використанню месенджера "Телеграм" у розповсюдженні наркотиків. Кримінологічний форум, 2019.
5. Glassman M., Kan M.J. Intelligence in the internet age: The emergence and evolution of Open Source Intelligence (OSINT), Computers in Human Behavior. Elsevier, 2012
6. Hulnick A.S. The dilemma of open sources intelligence: Is OSINT really intelligence? The Oxford handbook of national security. 2010.
7. Гольцман В. MySQL 5.0. URL: dspace2.regi.rovno.ua
8. Schwartz B., Zaitsev P., Tkachenko V. High performance MySQL: optimization, backups, and replication, 2012.
9. Modrzyk N., Building telegram bots: develop bots in 12 programming languages using the telegram bot API, 2018.
10. Arx T. von, Paterson K.G., On the Cryptographic Fragility of the Telegram Ecosystem. Cryptology ePrint Archive, 2022.
11. Стадниченко А.В. Програмні засоби тестування ботів месенджера. Telegram, 2020.

Додаток А. Лістинг програмного продукту

Файл №1 main.py

```
import os
from sys import path
from time import sleep
from pickle import load
from threading import Thread
from work_database import create_database

# Добавление в патч папки с парсером
path.append('./parser')
from parse import *

# Путь к базе данных
db_patch = './database.sql'

# Путь к ключам телеграмм аккаунтов
acc_patch = './account'

# Путь к файлу с чатами
chats_patch = './parser/new_chats.pk'

# Кол-во аккаунтов
acc_numbers = 9

# Масив рабочих аккаунтов & Отслеживание потоков
stream_acc = [False for i in range(acc_numbers)]

# время таймаутов
timeouts = acc_numbers * 20 + 10


def dump_file(chats_patch):
    with open(chats_patch, 'rb') as chats_file:
        return load(chats_file)


def dialog_window():
    create_database() if int(input(
        '''____Create database:____\n| 0: NO                |\n| 1:
YES               |\n_____________________\n>> ''')) else 0
    parse_from_combot(patch=chats_patch) if int(input(
        '''___Start chat parser:___\n| 0: NO                |\n| 1:
YES               |\n_____________________\n>> ''')) else 0
```

```
def start_stream(chat, acc_id):
    global acc_numbers, stream_acc
    try:
        answer = os.popen(f'python ./parse_chats_to_db_stream.py {chat}
{acc_id}').read()
        if '420 FLOOD_WAIT_X' in answer:
            acc_numbers -= 1
            stream_acc = stream_acc[:-1]
            print(
                f'{chat}, account: {acc_id}, BAN : account will be removed from
the list')
    except Exception as e:
        print(str(e))
    finally:
        stream_acc[acc_id] = False


def start():
    while True:

        temp = 0

        for chat in list(dump_file(chats_patch)):

            while stream_acc[temp % acc_numbers]:
                sleep(1)
                temp += 1

            stream_acc[temp % acc_numbers] = True

            print(stream_acc)

            Thread(target=start_stream, args=(chat, temp % acc_numbers),
daemon=True).start()

            sleep(timeouts - acc_numbers * 20)
            temp += 1

        parse_from_messages_database(chats_patch)


if __name__ == '__main__':
    dialog_window()

    start()
```

Файл № 2 work_database.py

```
import pymysql

host_db = '127.0.0.1'
port_db = 3306
user_db = 'root'
password_db = 'zAlv%uU9t$fMwkbKUvaq^'
database_db = 'tg_bd'
import json


def create_database():
    global database_db

    # Создание базы данных
    sql = f'CREATE DATABASE {database_db}'
    #send_to_mysql(sql)

    sql = '''CREATE TABLE `chats` (
    `id` BIGINT(20) NOT NULL DEFAULT '0',
    `title` TEXT NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
    `username` TEXT NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
    PRIMARY KEY (`id`) USING BTREE
)
COLLATE='utf8mb4_0900_ai_ci'
ENGINE=InnoDB
;
'''
    send_to_mysql(sql)

    # Таблицца коннект
    sql = '''CREATE TABLE `connect` (
    `user_id` BIGINT(20) NOT NULL DEFAULT '0',
    `chat_id` BIGINT(20) NOT NULL DEFAULT '0',
    `status` TEXT NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
    `joined_date` TEXT NOT NULL COLLATE 'utf8mb4_0900_ai_ci'
)
COLLATE='utf8mb4_0900_ai_ci'
ENGINE=InnoDB
;
'''
    send_to_mysql(sql)

    # Таблица сообщения
    sql = '''CREATE TABLE `messages` (
    `user_id` BIGINT(20) NOT NULL DEFAULT '0',
    `chat_id` BIGINT(20) NOT NULL DEFAULT '0',
    `forward_from_chat_id` BIGINT(20) NULL DEFAULT NULL,
    `forward_from_chat_message_id` BIGINT(20) NULL DEFAULT NULL,
    `forward_text` TEXT NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
```

```
    `message_text` TEXT NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
    `message_id` BIGINT(20) NULL DEFAULT NULL,
    `message_date` TEXT NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci'
)
COLLATE='utf8mb4_0900_ai_ci'
ENGINE=InnoDB
;
'''

    send_to_mysql(sql)


    # Таблица пользователей
    sql = '''CREATE TABLE `users` (
    `id` BIGINT(20) NULL DEFAULT NULL,
    `is_bot` TEXT NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
    `is_deleted` TEXT NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
    `first_name` TEXT NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
    `username` TEXT NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
    `status` TEXT NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci'
)
COLLATE='utf8mb4_0900_ai_ci'
ENGINE=InnoDB
;
'''

    send_to_mysql(sql)


def send_to_mysql(sql):
    connection = pymysql.connect(host=host_db, user=user_db,
password=password_db, database=database_db,
                                 port=port_db,
cursorclass=pymysql.cursors.DictCursor)

    with connection:
        with connection.cursor() as cursor:
            cursor.execute(sql)
            connection.commit()
            return cursor.fetchall()


def message_info(app, chat):
    msgs = app.get_chat_history(chat, limit=app.get_chat_history_count(chat))
    for message in msgs:
        print(message)
        try:
            try:
```

```
                forward_from_chat_id =
message.reply_to_message.forward_from_chat.id
                forward_from_chat_message_id =
message.reply_to_message.forward_from_message_id
                forward_text = message.reply_to_message.text.replace('\n',
'').replace("""'""", '')
            except:
                forward_from_chat_id = 0
                forward_from_chat_message_id = 0
                forward_text = '---'
                text = message.text.replace('\n', '')
            sql = f"INSERT INTO messages (user_id, chat_id, forward_from_chat_id,
forward_from_chat_message_id, forward_text, message_text, message_id,
message_date) VALUES ({message.from_user.id}, {message.chat.id},
{forward_from_chat_id}, {forward_from_chat_message_id}, '{forward_text}',
'{text}', {message.id}, '{message.date}');"
            print(sql)
            send_to_mysql(sql)
        except Exception as e:
            print(str(e))


def members_info(app, chat, chat_id):
    cht = app.get_chat_members(chat, limit=app.get_chat_members_count(chat))
    for member in cht:
        try:
            sql = f"INSERT INTO users (id, is_bot, is_deleted,
first_name,  username, status) VALUES ({member.user.id}, '{member.user.is_bot}',
'{member.user.is_deleted}', '{member.user.first_name}', '{member.user.username}',
'{member.user.status}');"
            print(sql)
            send_to_mysql(sql)
        except Exception as e:
            print(member.user.id, str(e))
        sql = f"INSERT INTO connect (user_id, chat_id, status, joined_date )
VALUES ({member.user.id}, {chat_id}, '{member.status}', '{member.joined_date}')"
        print(sql)

        send_to_mysql(sql)


def chat_info(app, chat):
    chat_info = app.get_chat(chat)

    sql = f"INSERT INTO chats (id, title, username) VALUES ({chat_info.id},
'{chat_info.title}', '{chat_info.username}');"
    # print(sql)

    send_to_mysql(sql)
```

```
    return chat_info.id
```

Файл №3 parse_chats_to_db_stream.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# vim:fileencoding=utf-8

import sys
from configparser import ConfigParser
from pyrogram import Client
from work_database import chat_info, members_info, message_info


chat = sys.argv[1]
acc_id = sys.argv[2]

config = ConfigParser()
config.sections()
config.read(f'./account/{acc_id}/api.ini')

with Client(str(acc_id), config['API']['api_id'], config['API']['api_hash'],
workdir=f'./account/{acc_id}') as app:
    try:
        chat_id = chat_info(app, chat)
        members_info(app, chat, chat_id)
        message_info(app, chat)
        print(f'{chat} add succesfull!')
    except Exception as e:
       print(str(e))
```

Файл №4 parse.py

```
import re
from copy import copy
from requests import get
from pickle import dump, load
from work_database import send_to_mysql

all_chats_patch = './parser/all_chats.pk'


def settings_parser():
    print('''____Choose region:____
|en - English        |
|ru - Русский        |
```

```
|uk - Українська     |
|uz - O‘zbek         |
|tr - Türkçe         |
|id - Indonesia      |
|es - Español        |
|hi - हिंदी          |
|az - Azərbaycan     |
|all - all           |
----------------------''')
    return input('>>> ')


def parse_from_combot(patch):
    limit = 50
    counter = 0
    chats = []
    region = settings_parser()

    print('Parse chats start')

    # requests to site
    while True:
        temp_chats =
get(f'https://combot.org/api/chart/{region}?limit={limit}&offset={limit *
counter}').json()
        if temp_chats == []:
            break
        for chat in temp_chats:
            chats.append(chat['u'])

        counter += 1

    # split chats name '_'
    temp_chats = copy(chats)
    for chat in temp_chats:
        for word in chat.split('_'):
            chats.append(word)

    history(patch, chats)


def history(patch, chats):
    with open(patch, 'wb') as dump_file, open(all_chats_patch, 'rb') as
chat_storage:
        all_chats = load(chat_storage)

        chats = set(chats) - set(all_chats)
        all_chats = list(all_chats) + list(chats)
```

```
        dump(chats, dump_file)

    with open(all_chats_patch, 'wb') as chat_storage:
        dump(all_chats, chat_storage)

    print(f'Parse succesfull! {len(chats)}')


def parse_from_id_database(patch):
    chats = []

    for segment in send_to_mysql('SELECT DISTINCT forward_from_chat_id FROM
messages'):
        chats.append(segment['forward_from_chat_id'])

    history(patch, chats)


def parse_from_messages_database(patch):
    chats = []

    for segment in send_to_mysql("SELECT DISTINCT message_text FROM messages
WHERE message_text LIKE '%joinchat%'"):
        chats.append(re.findall(re.compile("https://t.me/[a-zA-Z0-9]+/[a-zA-Z0-
9_-]+"), segment['message_text']))

    for segment in send_to_mysql("SELECT DISTINCT message_text FROM messages
WHERE message_text LIKE '%t.me%'"):
        chats.append(re.findall(re.compile("https://t.me/([a-zA-Z0-9_-]+)"),
segment['message_text']))

    history(patch, chats)
```