convolutional neural network», in 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT), Dhaka, Bangladesh, 2018, pp. 112-117. **doi: 10.1109/CEEICT.2018.8628078**.
33. F. Siddique, S. Sakib, and M.A.B. Siddique, «Handwritten digit recognition using convolutional neural network in python with tensorflow and observe the variation of accuracies for various hidden layers», *Preprints*, pp. 1-6, 2019. **doi: 10.20944/preprints201903.0039.v1**.
34. Y. LeCun, C. Cortes, and C.J.C. Burges, The MNIST database of handwritten digits [Online]. Available: http://yann.lecun.com/exdb/mnist. Accessed on: May 30, 2023.
35. P.J. Grother, *NIST special database 19 – handprinted forms and characters database*, National Institute of Standards and Technology (NIST), Tech. Rep. Publ., 1995. **doi: 10.18434/T4H01C**.

Рецензент: О.І. Проніна,
канд. техн. наук, доц., ДВНЗ «ПДТУ»

Стаття надійшла 01.10.2023
Стаття прийнята 05.11.2023

УДК 004.896 **doi: 10.31498/2225-6733.47.2023.299990**

© Сергієнко А.В.[1], Єфімов П.С.[2], Обиденний Є.О.[3], Бешта Л.В.[4]

ВИКОРИСТАННЯ ХМАРНИХ ТЕХНОЛОГІЙ ДЛЯ АВТОМАТИЧНОЇ РОЗСИЛКИ ПОВІДОМЛЕНЬ В TELEGRAM

В статті розглянуто питання автоматизації розповсюдження повідомлень в месенджері Telegram за допомогою хмарних технології. Описано загальну проблему розсилки повідомлень декількома каналами. Зроблений огляд останніх досліджень та публікацій сучасних технологій обміну повідомленнями в Telegram, проаналізовані бібліотеки, фреймворки та сучасні патерни. Запропоновано методику використання потужностей AWS EC2 для створення масштабованого та надійного рішення для автоматизованої розсилки повідомлень. Детально описаний процес створення програми за запропонованою методикою. Описано її архітектуру та взаємодію модулів. Архітектура структурована таким чином, щоб інкапсулювати різні функції, кожна з яких служить певній меті в ширшому контексті автоматизованого розповсюдження повідомлень на платформі Telegram. Наводяться характеристики обраного сервісу AWS, обраної операційної системи, мови програмування та використаних бібліотек. Продемонстровано використання автоматичного масштабування в AWS, що автоматизує налаштування ресурсів на основі попередньо визначених критеріїв та забезпечує ефективне рішення для обробки коливань попиту та підтримки оптимальної продуктивності системи. Використані мова програмування Python та бібліотека Telethon. Показано взаємодію AWS EC2 з API Telegram та використання модуля Asynchronous Message Forwarding для організації обробки декількох каналів одночасно. Описані можливості програми – налаштування облікових записів Telegram, пошук повідомлень, автоматизована переадресація на кілька каналів та інші дії. Показані результати проведених тестувань розробленої програми, що показують, що автоматизація процесу пересилки повідомлень збільшує швидкість їх доставки у велику кількість чатів на 120 секунд, а це ефективніше в 40 разів, а її

[1] *канд. техн. наук, ДВНЗ «Приазовський державний технічний університет», м. Дніпро, ORCID: 0000-0003-1328-2572, sergienko_a_v@pstu.edu*
[2] *магістрант, ДВНЗ «Приазовський державний технічний університет», м. Дніпро, efor1999@gmail.com*
[3] *асистент, НТУ «Дніпровська політехніка», м. Дніпро, ORCID: 0000-0001-9065-6369, Obydennyi.Ye.O@nmu.one*
[4] *асистент, НТУ «Дніпровська політехніка», ORCID: 0000-0003-1461-4399, Beshta.l.v@nmu.one*

p-ISSN: 2225-6733; e-ISSN: 2519-271X

ресурсомісткість менша, ніж ресурсомісткість звичайного телеграм клієнту в 5 разів.
Ключові слова: *хмарне середовище, AWS EC2, екземпляр, розсилка повідомлень, API Telegram, Python, автоматизація, Telethon.*

A.V. Serhiienko, P.S. Yefimov, E.O. Obydennyi, L.V. Beshta. Using of a cloud technologies for automatic sending of notifications in Telegram*. The article discusses the issue of automating the distribution of messages in the Telegram messenger using cloud technologies. The general problem of sending messages through several channels is described. An overview of the latest research and publications of modern messaging technologies in Telegram was made, libraries, frameworks and modern patterns were analyzed. A method of using the capabilities of AWS EC2 to create a scalable and reliable solution for automated message distribution is proposed. The process of creating a program according to the proposed method is described in detail. Its architecture and interaction of modules are described. The architecture is structured to encapsulate different functions, each of which serves a specific purpose in the larger context of automated message distribution on the Telegram platform. The characteristics of the selected AWS service, the selected operating system, the programming language and the used libraries are given. The use of automatic scaling in AWS is demonstrated, which automates the configuration of resources based on predefined criteria and provides an effective solution for handling demand fluctuations and maintaining optimal system performance. Python programming language and Telethon library are used. The interaction of AWS EC2 with the Telegram API and the use of the Asynchronous Message Forwarding module to organize the processing of several channels at the same time are shown. The features of the program are described – setting up Telegram accounts, searching for messages, automated forwarding to several channels and other actions. The results of tests of the developed program are shown, which show that the automation of the process of sending messages increases the speed of their delivery to a large number of chats by 120 seconds, which is 40 times more effective, and its resource consumption is less than the resource consumption of ordinary client telegrams in 5 times.*
Key words: *cloud environment, AWS EC2, instance, messaging, Telegram API, Python, automation, Telethon.*

Постановка проблеми. У цьому науковому дослідженні ми вирішуємо нагальну проблему оптимізації комунікаційних процесів на платформі Telegram за допомогою автоматизованої програми Python. Зростаюча база користувачів Telegram зумовила необхідність більш ефективних методів розповсюдження повідомлень, зокрема для розповсюдження інформації кількома каналами. Сучасна практика часто передбачає пересилання вручну, що призводить до неефективності та обмежень у часі. Усвідомлюючи цю гостроту, наше дослідження має на меті розробити автоматизоване рішення, яке використовує AWS EC2, API Telegram і програмування на Python для покращення робочих процесів спілкування [1].

Фундаментальна проблема полягає у відсутності уніфікованого та автоматизованого підходу до розповсюдження повідомлень, що перешкоджає масштабованості та ефективності комунікаційних стратегій у Telegram. Пересилання вручну не тільки споживає дорогоцінний час і ресурси, але й створює ризик помилок і невідповідностей. Це дослідження спрямоване на вирішення цієї проблеми шляхом розробки складної програми на Python, яка здатна асинхронно пересилати повідомлення, забезпечуючи своєчасне та точне розповсюдження інформації різними каналами.

Крім того, відсутність комплексного інструменту, який легко інтегрується з хмарними технологіями та API Telegram, перешкоджає адаптації поточних практик спілкування. Мета полягає в тому, щоб подолати цей розрив, запропонувавши програму, яка використовує потужність AWS EC2 для створення масштабованого та надійного рішення. Приступаючи до цієї роботи, ми очікуємо, що наше дослідження зробить значний внесок у дискусію щодо оптимізації комунікаційних процесів в екосистемі Telegram [2].

p-ISSN: 2225-6733; e-ISSN: 2519-271X

Аналіз останніх досліджень і публікацій. В контексті розробки додатку для автоматичної розсилки повідомлень в Telegram з використанням хмарних технологій AWS EC2, вибір методів для вирішення поставленої задачі є ключовим. Було проаналізовано останні дослідження та публікації, що описують різні методології та підходи для вирішення поставлених завдань дослідження.

Одним з основних методів є реалізація механізмів асинхронного обміну повідомленнями. Асинхронне програмування дозволяє відправляти повідомлення незалежно від інших завдань, підвищуючи швидкість реакції та ефективність системи. Вибір асинхронних бібліотек, фреймворків та патернів буде проаналізовано для визначення найбільш підходящого підходу.

Telegram API – це потужний інструмент, який дозволяє розробникам програмно взаємодіяти з платформою обміну повідомленнями Telegram. Пропонуючи широкий набір функціональних можливостей, він дозволяє бездоганно інтегрувати функції Telegram у зовнішні програми [3]. Розробники можуть використовувати API для надсилання повідомлень, створення ботів, керування групами та доступу до даних користувачів. API Telegram, відомий своєю простотою та універсальністю, підтримує кілька мов програмування, сприяючи різноманітному спектру програм. Його наскрізне шифрування забезпечує безпечне спілкування, що робить його кращим вибором для розробників, які прагнуть включити надійні можливості обміну повідомленнями у свої програмні рішення [4].

Amazon Elastic Compute Cloud (AWS EC2) є ключовим компонентом сучасних хмарних обчислень, що забезпечує масштабовані віртуальні сервери в хмарі. За допомогою AWS EC2 користувачі можуть розгортати віртуальні сервери та керувати ними для розміщення програм і обробки різноманітних робочих навантажень. Його еластичність дозволяє його легко масштабувати, регулюючи обчислювальну потужність відповідно до потреб. Ця гнучкість у поєднанні з різними попередньо налаштованими образами машин сприяє швидкому розгортанню різноманітних програм. Крім того, екземпляри EC2 створені для забезпечення надійності та безпеки, пропонуючи комплексне рішення для підприємств, яким потрібні економічно ефективні та безпечні хмарні обчислювальні ресурси.

Проведення тестування продуктивності має важливе значення для оцінки ефективності різних методів. Навантажувальне тестування, стрес-тестування та бенчмаркінг будуть використовуватися для оцінки здатності програми обробляти різні робочі навантаження та взаємодію з користувачами.

Впровадження надійних методів безпеки, включаючи шифрування, автентифікацію та авторизацію, має вирішальне значення при роботі з конфіденційними даними повідомлень. Механізми управління ідентифікацією та доступом (IAM) і шифрування AWS будуть перевірені на предмет їхньої ефективності в забезпеченні цілісності та конфіденційності даних.

Для оцінки переваг і недоліків різних методів буде використано метод порівняльного аналізу. Це включає порівняння синхронних та асинхронних підходів до відправки повідомлень, фреймворків для розробки ботів та конфігурацій AWS EC2 [5].

Масштабування є критично важливим аспектом архітектури системи, що забезпечує адаптивність до різних навантажень і вимог користувачів. У контексті хмарних обчислень масштабування передбачає динамічне налаштування ресурсів для оптимізації продуктивності. Вертикальне масштабування передбачає збільшення потужності наявного обладнання, а горизонтальне масштабування розподіляє навантаження між кількома машинами. Це підвищує стійкість, покращує час відгуку та враховує зростаючу базу користувачів. Ефективні стратегії масштабування, такі як автоматичне масштабування в AWS, автоматизують налаштування ресурсів на основі попередньо визначених критеріїв, забезпечуючи ефективне рішення для обробки коливань попиту та підтримки оптимальної продуктивності системи [6].

Метою даної роботи є проведення поглибленого дослідження технології, яка використовується для розробки програми для автоматичної розсилки повідомлень, використовуючи можливості хмарної технології Amazon Web Services (AWS) EC2. Дослідження охоплює комплексне дослідження різних аспектів життєвого циклу розробки програми, включаючи архітектурний дизайн, реалізацію та розгортання в інфраструктурі AWS EC2. Дослідження спрямоване на виявлення тонкощів розробки надійної та масштабованої програми, яка автоматизує розсилку повідомлень, враховуючи динамічну та розподілену природу AWS EC2.

p-ISSN: 2225-6733; e-ISSN: 2519-271X

Виклад основного матеріалу. Основні функції системи зосереджуватимуться на безперебійній відправці повідомлень на платформі Telegram. Ці функції включають реєстрацію користувача, авторизацію, перегляд списків продуктів. Важливим елементом є розширення цих функціональних можливостей для користувачів-адміністраторів, дозволяючи їм переглядати, редагувати та створювати продукти, отримувати доступ до профілів користувачів і деталей облікового запису, завершувати сеанси користувачів і переглядати історію транзакцій [7].

Оскільки для забезпечення роботи програми необхідно, щоб програма працювала постійно, було обрано сервіс AWS. Було обрано Instance формату t2.micro з характеристиками центрального процесору, зображеного рис. 1.

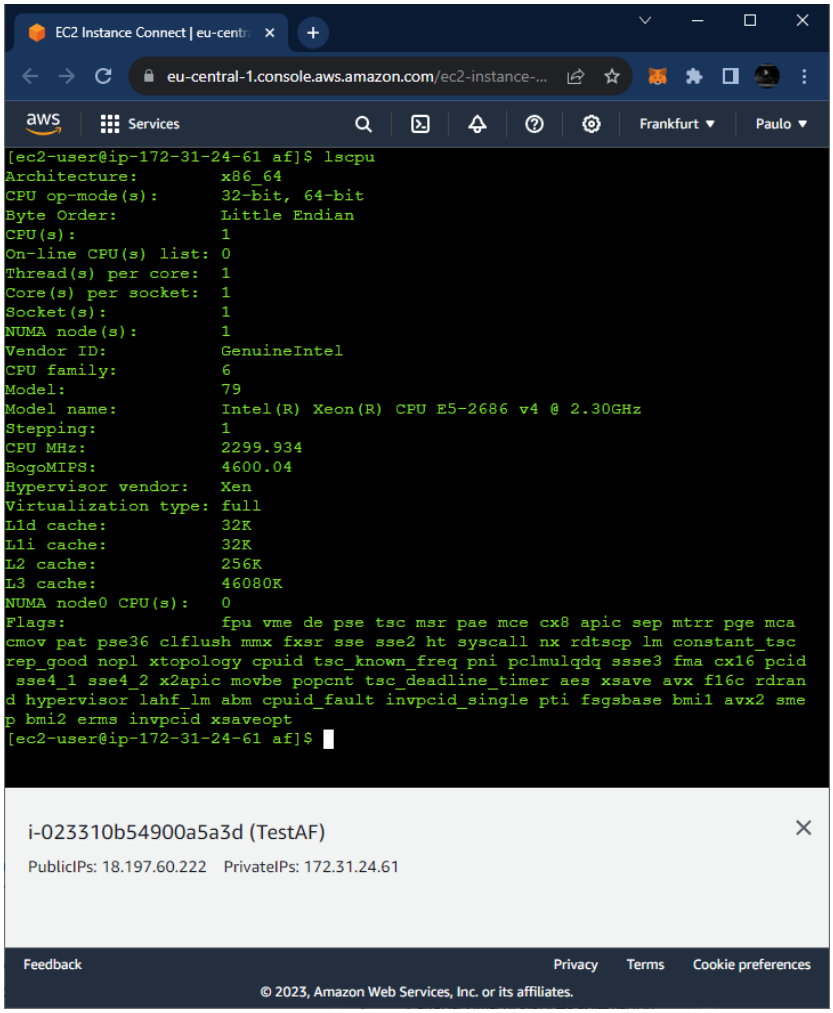


Рис. 1 – Характеристики CPU

Обов’язково повинно бути забезпечення стабільним інтернетом, дані про швидкість мережі Інтернет зображено за допомогою моніторингових служб AWS на рис. 2.

Програма повинна мати пакет програмного забезпечення, щоб працювати. Таблиця 1 містить перелік програмного забезпечення. Для підключення необхідний комп’ютер, оснащений мережевим адаптером і підключений до мережі Інтернет.

Таблиця 1

Програмне забезпечення

Назва програмного забезпечення	Версія (не нижче)
версія ядра Linux	5.10.186-179.751.amzn2.x86_64
Python 2.7.18	2.7.18
tmux	3.a

p-ISSN: 2225-6733; e-ISSN: 2519-271X

В описаній системі ядро Linux слугує фундаментальним рівнем, забезпечуючи основні функції операційної системи. Python, версія 2.7.18, виступає основною мовою програмування для скрипту автоматизації Telegram. Універсальність і простота використання Python робить її ідеальним вибором для взаємодії з API Telegram. Крім того, наявність tmux версії 3.a покращує виконання скрипту, полегшуючи мультиплексування терміналів, що дозволяє одночасно керувати кількома сесіями терміналів. Ці програмні компоненти формують надійне середовище, що забезпечує надійне виконання інструменту автоматизації Telegram на зазначеній системі.

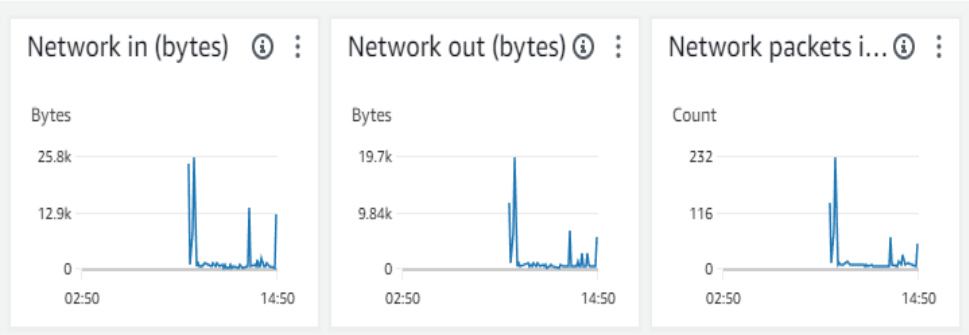


Рис. 2 – Швидкість мережі

В основі програми Python лежить її ретельно розроблена модульна архітектура, критичний аспект, що сприяє її універсальності та масштабованості. Архітектура структурована таким чином, щоб інкапсулювати різні функції, кожна з яких служить певній меті в ширшому контексті автоматизованого розповсюдження повідомлень на платформі Telegram.

Програма починає свою роботу з модуля налаштування облікового запису, що дозволяє користувачам легко налаштувати свої облікові дані Telegram. Цей модуль розроблено зі зручним інтерфейсом, який збирає важливу інформацію, таку як ідентифікатор API, хеш API та номер телефону. Бібліотека Telethon забезпечує безпечну взаємодію з API Telegram, забезпечуючи цілісність процесу налаштування облікового запису користувача.

Після налаштування облікового запису в роботу вступає модуль отримання повідомлень. Цей модуль використовує асинхронні механізми для ефективного отримання повідомлень із визначених каналів. Використовуючи потужність бібліотеки Telethon, програма взаємодіє з API Telegram для отримання повідомлень на основі визначених користувачем параметрів, забезпечуючи динамічний і адаптований підхід до отримання повідомлень [8].

Наріжним каменем функціональності програми є модуль Asynchronous Message Forwarding. Асинхронна робота дозволяє програмі обробляти декілька каналів одночасно без шкоди для продуктивності. Цей модуль використовує асинхронну бібліотеку Python, забезпечуючи оптимальне використання ресурсів і оперативність. Повідомлення, отримані з модуля отримання повідомлень, плавно пересилаються на кілька каналів, підвищуючи ефективність зв’язку за різними каналами.

Основним елементом модульної архітектури є бездоганна інтеграція з бібліотекою Telethon. Створений для взаємодії з Telegram, Telethon діє як міст між програмою Python і API Telegram. Його широкий функціонал спрощує складні операції, такі як пересилання повідомлень і взаємодія з об’єктами Telegram. Модульна архітектура гарантує, що кожен модуль ефективно використовує бібліотеку Telethon, сприяючи згуртованому та спрощеному процесу розробки [9].

Модульний дизайн не тільки підвищує ефективність програми, але й сприяє її адаптивності. Користувачі можуть вибірково розгортати конкретні модулі відповідно до своїх вимог, забезпечуючи адаптоване та ресурсоефективне виконання. Цей модульний підхід закладає основу для майбутніх удосконалень і доповнень, гарантуючи, що програма може розвиватися в тандемі з динамічним ландшафтом Telegram і вимогами користувачів.

Створюємо сесію «tmux» для того, щоб код працював постійно, за допомогою команди «tmux new-session -s autoforward» рис. 3 [10].

p-ISSN: 2225-6733; e-ISSN: 2519-271X

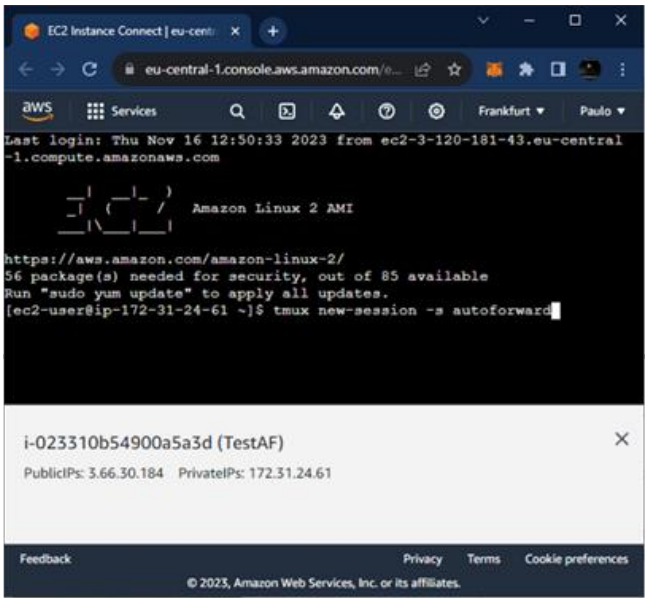


Рис. 3 – Створення сесії з назвою «autoforward»

Тепер запускаємо програму за допомогою «python3 main.py», обираємо почати пересилку «2: Start Forwarding». Вводимо «Channel ID» та «ID MESSAGE» і програма успішно запущена, відображаються ID каналів, куди було відправлено повідомлення (рис. 4).

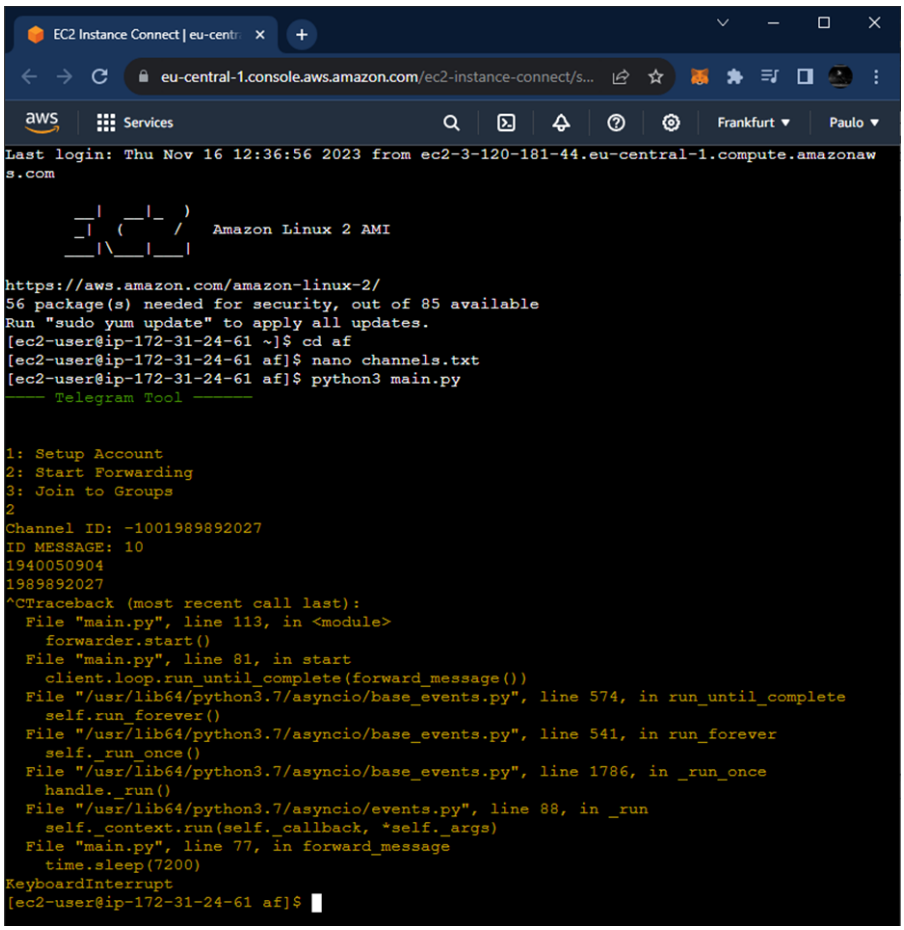
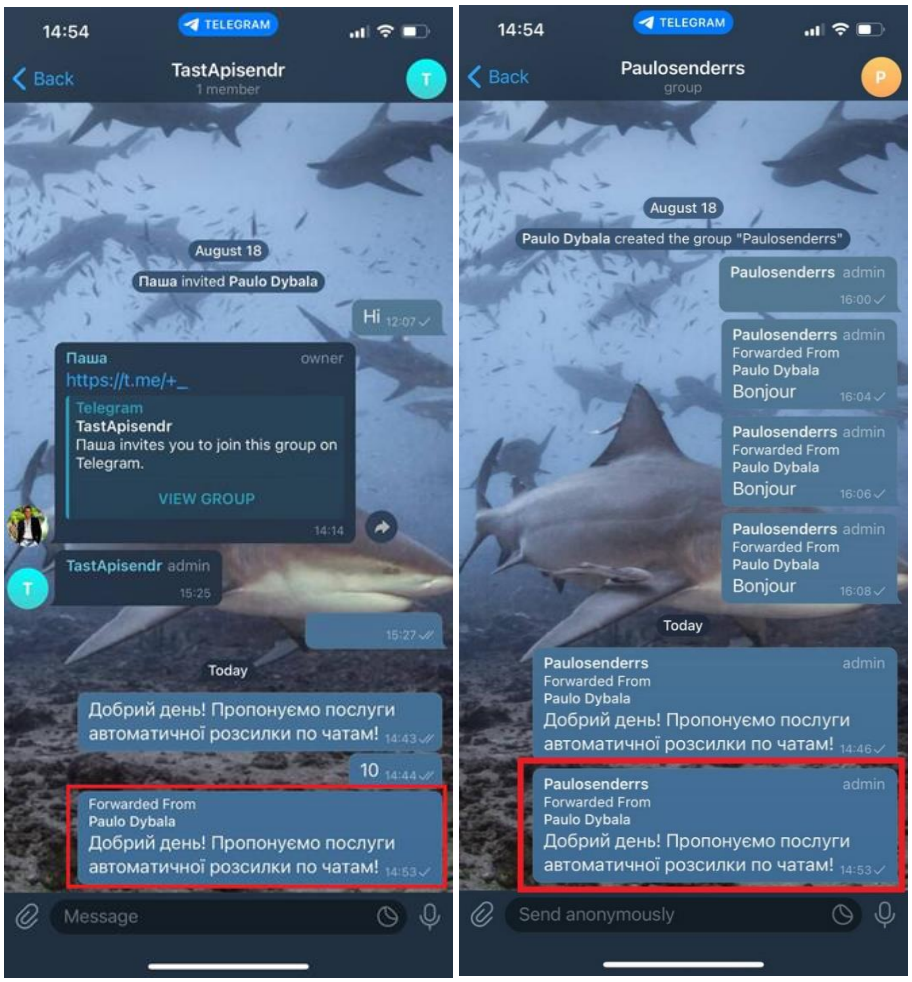


Рис. 4 – Знімок екрана, коли пересилка повідомлень відбулась

p-ISSN: 2225-6733; e-ISSN: 2519-271X

Таймер встановлений пересилати це повідомлення через кожні дві години, це вказано в коді програми «time.sleep(7200)». Перевіримо чати в самому телеграмі, перший та другий чати зображені відповідно на рис. 5 а) та рис. 5 б).



а) б)

Рис. 5 – Успішна пересилка повідомлення: а) – в першому чаті; б) – в другому чаті

В рамках експериментальних досліджень інструмент Telegram пройшов ретельне тестування для перевірки його надійності, функціональності та відповідності визначеним вимогам. Особливістю інструменту є можливість асинхронного пересилання повідомлень, що забезпечує автоматичне та ефективне розповсюдження інформації кількома каналами.

Модульна структура інструменту складається з різних програмних модулів, включаючи ядро Linux версії 5.10.186-179.751.amzn2.x86_64, Python 2.7.18 та tmux 3.a. Ці модулі в сукупності сприяють безперебійній роботі інструменту. AWS EC2 Instance слугує хостинговою платформою, що відповідає найкращим галузевим практикам масштабованості, безпеки та надійності.

Висновки

В рамках даної роботи було проведено дослідження предметної області, аналіз потенційних методів вирішення завдання. Були визначені вимоги до інформаційної системи, її функцій, складено технічне завдання. Проведені експериментальні дослідження розробленої інформаційної системи, що показують, що автоматизація процесу пересилки повідомлень збільшує швидкість їх доставки у велику кількість чатів на 120 секунд, а це ефективніше в 40 разів.

Було проведено дослідження, що дозволяє зробити висновок, що розроблена за запропонованою технологією програма має меншу в 5 разів ресурсомісткість, ніж звичайний телеграм клієнт. Програма протестована у реальних умовах та доводить її надійність та безпеку

використання. Програма задовольняє виявлений ринковий попит на автоматизацію, але й має продуманий дизайн, пріоритети користувацького досвіду та дотримується найкращих практик у програмуванні на Python та хмарній інфраструктурі.

Перелік використаних джерел:

1. Liu K., Dong L.-J. Research on cloud data storage technology and its architecture implementation. *Procedia Engineering*. 2012. Vol. 29. Pp. 133-137. DOI: https://doi.org/10.1016/j.proeng.2011.12.682.
2. Amazon Web Services. AWS Elastic Compute Cloud (EC2): Documentation and User Guides. 2021. URL: https://aws.amazon.com/ec2/documentation/ (дата звернення: 18.04.2023).
3. What is Telegram Messenger. 2018. URL: https://medium.com/@telegramguide/what-is-telegram-messenger-c079418e1f10 (дата звернення: 25.04.2023).
4. TelegramClient. URL: https://doc.esdoc.org/github.com/dot-build/telegram-js/class/src/telegram-client.js~TelegramClient.html (дата звернення: 05.02.2023).
5. amazon.aws.ec2_instance module – Create & manage EC2 instances. AWS Documentation. URL: https://docs.ansible.com/ansible/latest/collections/amazon/aws/ec2_instance_module.html (дата звернення: 23.04.2023).
6. Artificial intelligence in information systems research: A systematic literature review and research agenda / Collins C., Dennehy D., Conboy K., Mikalef P. *International Journal of Information Management*. 2021. Vol. 60. Pp. 1-17. DOI: https://doi.org/10.1016/j.ijinfomgt.2021.102383.
7. Faisandier A., Roedler G., Adcock R. System Architecture. SEBoK. URL: https://sebokwiki.org/wiki/System_Architecture (дата звернення: 30.04.2023).
8. Telethon's Documentation. URL: https://docs.telethon.dev/en/stable (дата звернення: 07.05.2023).
9. Python Standard Library. Python 3.12.2 documentation. URL: https://docs.python.org/3/library/ (дата звернення: 07.05.2023).
10. Tmux. Ubuntuusers. URL: https://wiki.ubuntuusers.de/tmux/ (дата звернення: 15.05.2023).

References:

1. K. Liu, and L.-J. Dong, «Research on cloud data storage technology and its architecture implementation», *Procedia Engineering*, vol. 29, pp. 133-137, 2012. **doi: 10.1016/j.proeng.2011.12.682**.
2. Amazon Web Services. AWS Elastic Compute Cloud (EC2): Documentation and User Guides, 2021 [Online]. Available: https://aws.amazon.com/ec2/documentation/. Accessed on: April 18, 2023.
3. What is Telegram Messenger, 2018 [Online]. Available: https://medium.com/@telegramguide/what-is-telegram-messenger-c079418e1f10. Accessed on: April 25, 2023.
4. TelegramClient [Online]. Available: https://doc.esdoc.org/github.com/dot-build/telegram-js/class/src/telegram-client.js~TelegramClient.html. Accessed on: May 5, 2023.
5. amazon.aws.ec2_instance module – Create & manage EC2 instances. AWS Documentation [Online]. Available: https://docs.ansible.com/ansible/latest/collections/amazon/aws/ec2_instance_module.html. Accessed on: April 23, 2023.
6. C. Collins, D. Dennehy, K. Conboy, and P. Mikalef, «Artificial intelligence in information systems research: A systematic literature review and research agenda», *International Journal of Information Management*, vol. 60, pp. 1-17, 2021. **doi: 10.1016/j.ijinfomgt.2021.102383**.
7. Faisandier A., Roedler G., Adcock R. System Architecture. SEBoK [Online]. Available: https://sebokwiki.org/wiki/System_Architecture. Accessed on: April 30, 2023.
8. Telethon's Documentation [Online]. Available: https://docs.telethon.dev/en/stable. Accessed on: May 7, 2023.
9. Python Standard Library. Python 3.12.2 documentation [Online]. Available: https://docs.python.org/3/library/. Accessed on: May 7, 2023.
10. Tmux. Ubuntuusers [Online]. Available: https://wiki.ubuntuusers.de/tmux. Accessed on: May 15, 2023.

Рецензент: О.І. Проніна,
канд. техн. наук, доц., ДВНЗ «ПДТУ»

Стаття надійшла 13.09.2023
Стаття прийнята 17.10.2023