**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**
**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**
**КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ**

# КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Telegram-бот для взаємодії з таблицями та можливості їх редагування в реальному часі»

на здобуття освітнього ступеня бакалавр
за спеціальності 126 Інформаційні системи та технології
*(код,найменування спеціальності)*
освітньо-професійної програми Інформаційні системи та технології
*(назва)*

*Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело*

_____ Володимир БЕРБЕНЕЦЬ
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав:
здобувач вищої освіти
група ІСД-41 Володимир БЕРБЕНЕЦЬ
*(ім'я, ПРІЗВИЩЕ)*

Керівник: Олег СЕНЬКОВ
*Доц. каф.* *(ім'я, ПРІЗВИЩЕ)*

Рецензент: _____
*(ім'я, ПРІЗВИЩЕ)*

**Київ 2025**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

**Навчально-науковий інститут Інформаційних технологій**

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти бакалавр
Спеціальність 126 Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

**ЗАТВЕРДЖУЮ**
Завідувач кафедрою ІСТ

_____Каміла СТОРЧАК

«____»_____2025 р.

# З А В Д А Н Н Я
## НА КВАЛІФІКАЦІЙНУ РОБОТУ

Бербенцю Володимиру Едуардовичу
*(прізвище, ім'я, по батькові здобувача)*

1. Тема кваліфікаційної роботи: Telegram-бот для взаємодії з таблицями та можливості їх редагування в реальному часі

керівник кваліфікаційної роботи: Олег СЕНЬКОВ Доц.каф.
*(ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)*

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56

2. Строк подання кваліфікаційної роботи «30» травня 2025 р.

3. Вихідні дані до кваліфікаційної роботи:
   1. Можливості Telegram API та Google Sheets API.
   2. Технології інтеграції чат-ботів з табличними сервісами.
   3. Методологія розробки ботів для обробки даних у реальному часі.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   1. Аналіз аналогічних рішень та вибір технологій.
   2. Проєктування архітектури Telegram-бота.
   3. Реалізація, тестування та підготовка до впровадження системи.

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання «24» лютого 2025 р.

# КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів Кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Формулювання теми роботи та затвердження плану | 18.02.2025 | Підготовка до початку роботи, визначення основних напрямків |
| 2 | Збір та аналіз літератури по темі роботи | 30.02.2025 | Огляд джерел та технічної документації |
| 3 | Опис та вибір інструментів для розробки (Python, бібліотеки) | 23.03.2025 | Вибір відповідних технологій для розробки Telegram-бота |
| 4 | Розробка алгоритму роботи бота | 10.04.2025 | Створення логіки роботи бота, планування функціоналу |
| 5 | Реалізація основної частини бота (API, інтеграція з Google Sheets) | 19.04.2025 | Написання коду для підключення до Google Sheets, обробка запитів |
| 6 | Тестування функціоналу та виправлення помилок | 23.04.2025 | Перевірка роботи бота, тестування на різних даних |
| 7 | Написання звіту про виконану роботу | | Оформлення документу, узагальнення результатів |
| 8 | Захист кваліфікаційної роботи | | Презентація роботи перед комісією |

Здобувач вищої освіти _____ Володимир БЕРБЕНЕЦЬ

*(підпис)* *(ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Олег СЕНЬКОВ

*(підпис)* *(ім'я, ПРІЗВИЩЕ)*

# РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття ступеня магістра: 58 стор., 65 рис., 1 табл., 22 джерела.

Мета роботи – розробити Telegram-бота, здатного у зручному форматі надавати доступ до редагування і перегляду таблиць у режимі реального часу.

Об’єкт дослідження – процес інтеграції Telegram-бота з табличними сервісами, зокрема з Google Sheets.

Предмет дослідження – технології та інструменти, які використовуються для створення Telegram-ботів, а також особливості їхньої реалізації.

Короткий зміст роботи. У кваліфікаційній роботі розглянуто проблему створення Telegram-бота для взаємодії з електронними таблицями Google Sheets у режимі реального часу. Проведено аналіз існуючих рішень і обґрунтовано вибір технологій. Розроблено архітектуру бота, реалізовано функціонал редагування і перегляду даних з мобільних або десктопних клієнтів Telegram. Проведено модульне та інтеграційне тестування. Отримане рішення демонструє зручність у використанні, високу гнучкість та можливість масштабування. Результати можуть бути застосовані в бізнесі, освітніх та інших сферах для автоматизації роботи з табличними даними.

Ключові слова: TELEGRAM-БОТ, GOOGLE SHEETS, ІНТЕГРАЦІЯ, АВТОМАТИЗАЦІЯ, РЕАЛЬНИЙ ЧАС, ОБРОБКА ДАНИХ, API, PYTHON, РОЗРОБКА, ТЕСТУВАННЯ.

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

## Навчально-науковий інститут інформаційних технологій

**ПОДАННЯ**
**ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ**
**ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ**
**на здобуття освітнього ступеня бакалавра**

Направляється здобувач Бербенець В.Е. до захисту кваліфікаційної роботи

за спеціальністю 126 Інформаційні системи та технології

освітньо-професійної програми Інформаційних систем та технологій

на тему: «Telegram-бот для взаємодії з таблицями та можливості їх редагування в реальному часі»**.**

Кваліфікаційна робота і резензія додаються.

Директор ННІІТ _____ Катерина НЕСТЕРЕНКО
*(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

**Висновок керівника кваліфікаційної роботи**

Здобувач Бербенець Володимир Едуардович обрав тему кваліфікаційної роботи, метою якої була розробка Telegram-бота для взаємодії з таблицями та можливості їх редагування в реальному часі. Перелік використаних джерел свідчить про вміння дипломником розбиратись у наукових питаннях та застосовувати їх при дослідженнях. Під час виконання кваліфікаційної роботи Бербенець В.Е. показав добру теоретичну та практичну підготовку, вміння вирішувати самостійно питання та проводити дослідження. Робота виконана сумлінно та вчасно за планом.

Все це дозволяє оцінити виконану кваліфікаційну роботу здобувача Володимира Бербенця на оцінку «відмінно» та присвоїти йому кваліфікацію бакалавр з інформаційних систем та технологій

Керівник кваліфікаційної роботи _____ _____Олег СЕНЬКОВ
*(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

«____»_____2025 року

**Висновок кафедри про кваліфікаційну роботу**

Кваліфікаційна робота розглянута. Здобувач Бербенець В.Е. допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедрою Інформаційних систем та технологій
*(назва)*

_____ Каміла СТОРЧАК
*(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

**ВІДГУК РЕЦЕНЗЕНТА**
**на кваліфікаційну роботу**
**на здобуття освітнього ступеня бакалавра**

здобувача вищої освіти ***Бербенець Володимир Едуардович***
на тему: «Telegram-бот для взаємодії з таблицями та можливості їх редагування в реальному часі».

**Актуальність.**

Обрана тема є актуальною, оскільки питання автоматизації обробки табличних даних через месенджери відповідає сучасним тенденціям цифровізації бізнес-процесів. Реалізація Telegram-бота для взаємодії з Google Sheets дозволяє підвищити зручність та ефективність доступу до даних у реальному часі, що є важливим як у корпоративному середовищі, так і в особистому користуванні. Такі інструменти особливо затребувані в умовах дистанційної роботи, швидкого прийняття рішень та потреби в оперативному редагуванні інформації без використання складного програмного забезпечення. Враховуючи широке розповсюдження Telegram та популярність Google-сервісів, запропоноване рішення має високий прикладний потенціал і може бути масштабовано під різні сфери діяльності — від освіти до малого та середнього бізнесу.

**Позитивні сторони.**

1. Робота відповідає заявленій тематиці та поставленим цілям.
2. Автор продемонстрував вміння працювати з API сторонніх сервісів і реалізовувати інтеграцію з Google Sheets.
3. Висвітлено практичну цінність розробки — мобільність, доступність і простота використання рішення.

**Недоліки.**

1. Функціональність реалізованого рішення обмежена можливостями API Google Sheets, що ускладнює виконання деяких складних операцій.
2. Відсутність графічного інтерфейсу ускладнює виконання складних маніпуляцій з таблицями, що частково обмежує аудиторію користувачів.

Відзначені зауваження не впливають на загальну позитивну оцінку бакалаврської кваліфікаційної роботи.

**Висновок:** *кваліфікаційна робота на здобуття ступеня бакалавра заслуговує оцінку "відмінно", а здобувач Бербенець Володимир Едуардович заслуговує присвоєння кваліфікації: бакалавр з інформаційних систем та технологій.*

Рецензент: _____ _____
*підпис* *Ім'я, ПРІЗВИЩЕ*

# ЗМІСТ

ВСТУП ..... 1
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ЇЇ ПРОБЛЕМ ..... 6
1.1 Месенджери і чат-боти як інструмент цифрової автоматизації ..... 6
1.2 Аналіз існуючих рішень для автоматизації роботи з таблицями ..... 15
1.3 Проблеми та напрямки подальшого розвитку ..... 19
РОЗДІЛ 2 ВИБІР ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ ПРОЄКТУ ..... 22
2.1 Технології для розробки програмного забезпечення ..... 22
2.1.1 Порівняльний аналіз мов програмування ..... 22
2.1.2 Аналіз та обґрунтування вибору середовища програмування ..... 23
2.1.3 Підбір хостингової платформи для розміщення Telegram-бота ..... 24
РОЗДІЛ 3 РОЗРОБКА ТА РЕАЛІЗАЦІЯ TELEGRAM-БОТА ..... 29
3.1 Функціональні можливості та принципи взаємодії бота ..... 29
3.1.1 Внутрішні особливості функціональних можливостей бота ..... 33
3.2 Необхідні бібліотеки та модулі для Telegram-бота ..... 34
3.3 Налаштування та використання API-ключів ..... 34
3.4 Інтеграція створених API ключів ..... 40
3.5 Логіка роботи бота та реалізація основних функцій ..... 41
3.6 Розгортання Telegram-бота на віддаленому сервері ..... 48
3.7 Аналіз стабільності та працездатності бота ..... 51
ВИСНОВКИ ..... 58

# ВСТУП

В наш час все довкола стрімко змінюється й удосконалюється, зокрема й процеси пов’язані з бізнесами, кожен день зростає потреба в інструментах, які допомагають працювати з даними швидко і без зайвих труднощів. Однією з ключових задач для малого та середнього бізнесу є автоматизація обліку клієнтів, товарів та інших не менш важливих даних. Старі способи роботи з електронними таблицями не допомагають, оскільки вони не такі зручні та мають ряд недоліків які я опишу в даній кваліфікаційній роботі.

Зараз месенджери крокнули далеко вперед чим обумовили великий ажіотаж серед розробників. Вони дали можливість розробляти власне програмне забезпечення інтегроване в саму соцмережу і таким чином полегшувати роботу з клієнтами, даними та іншими операціями. І одною із таких соцмереж являється Telegram.

Telegram, як один із найпопулярніших месенджерів, пропонує зручний API для створення власних ботів, які поміж звичних функцій по типу чатування, можуть також виступати посередниками між користувачем та базою даних. Інтеграція Telegram-бота з таблицями дозволяє полегшити керування процесами, що відбуваються в таблицях, їх оновленням, а також забезпечити швидкий доступ до інформації без потреби щоразу відкривати браузер.

Актуальність цієї роботи полягає в тому, що вона вирішує проблему відсутності доступних сервісів для роботи з таблицями для українських малих підприємств. Сервісів які потенційно мають можливість вийти на майже повну автоматизовану роботу та виключити з процесу оновлення інформації в таблицях будь-які помилки. Розроблений бот дозволить зробити роботу з даними набагато простішою, зменшити кількість рутинних операцій та підвищити ефективність управління клієнтською чи будь-якою іншою базою даних.

Аналіз останніх досліджень і публікацій

Проблематика автоматизації обробки даних за допомогою Telegram-ботів частково розглядається у науковій публікації авторів:

- Литвиненко О. В.
- Нафєєв Р. К.

У своїй роботі вони досліджують можливості застосування чат-ботів як помічників на середніх підприємствах, акцентуючи увагу на ефективності, економічності та доступності такого застосування.

Але автори не розглядають тему інтеграції чат-ботів з таблицями. Залишаються відкритими аспекти:

- прямого підключення до таблиць без складного програмування;
- захисту персональних та бізнес-даних при взаємодії через месенджер;
- забезпечення простоти користування ботом для звичайного користувача.

Це й створює необхідність розробки рішення, яке закриє ці прогалини. І таким рішенням може стати функціональний Telegram-бот для роботи з таблицями в реальному часі.

Мета та завдання дослідження

Мета роботи – розробити Telegram-бота, здатного у зручному форматі надавати доступ до редагування і перегляду таблиць у режимі реального часу. Основні завдання включали аналіз існуючих рішень, вибір оптимальних інструментів, проєктування архітектури бота, його розробку, тестування та підготовку до розгортання на хостингу.

Завдання дослідження:

1. Проаналізувати існуючі варіанти Telegram-ботів.
2. Визначити функціональні можливості власного бота.
3. Реалізувати механізми автентифікації та захисту даних.
4. Забезпечити CRUD-функціонал для роботи з таблицями.
5. Протестувати роботу бота.
6. Розгорнути бота на віддаленому сервері для цілодобової роботи.

Об’єкт і предмет дослідження

Об’єктом дослідження є процес інтеграції Telegram-бота з табличними сервісами, такими як Google Sheets

Предмет дослідження – це технології та інструменти, які використовуються для створення таких ботів, а також особливості їхньої реалізації..

Методи дослідження

У роботі використано такі методи:

- Теоретичний аналіз (огляд наукових джерел, документації API);
- Програмна реалізація (Python, бібліотеки telebot, gspread);
- Тестування (перевірка стабільності, обробка помилок);
- Розгортання на хостингу.

Теоретична, практична та методична значущість отриманих результатів

Теоретична значущість полягає в тому, що робота розширює уявлення про використання чат-ботів у бізнесі, зокрема як зручного інструменту доступу до таблиць через Telegram. Запропоновано простий і безпечний підхід до такої інтеграції.

Практична значущість полягає у створенні готового рішення, яке можуть застосовувати малі підприємства для обліку клієнтів і товарів без зайвих витрат та складного налаштування.

Методична значущість проявляється в тому, що весь процес – від задуму до реалізації – може бути використаний як приклад у навчанні або подальших проєктах, пов’язаних з автоматизацією.пр

Структура роботи

Робота складається зі вступу, трьох розділів, висновків та списку використаних джерел.

1. Розділ 1 – аналіз існуючих рішень та вибір технологій.
2. Розділ 2 – проєктування архітектури бота.
3. Розділ 3 – реалізація, тестування та впровадження.

У висновках підсумовано результати та наведено перспективи подальших досліджень.

# РОЗДІЛ 1
# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ЇЇ ПРОБЛЕМ

## 1.1 Месенджери і чат-боти як інструмент цифрової автоматизації

В сучасному світі всі прагнуть дійти до повної автоматизації власних проєктів, бізнесів і повсякденних процесів. Це актуально через постійне зростання кількості інформації, яку потрібно швидко обробляти. Саме на цій основі платформи, що забезпечують зручну комунікацію та інтеграцію з іншими сервісами, стають ключовими інструментами в сучасному цифровому середовищі.

Однією з таких платформ є Telegram – месенджер, який за останні роки перетворився з простої програми для чатування на потужний багатофункціональний інструмент. Telegram підтримує канали, групи, інтеграцію з ботами та можливість створення цікавих рішень для бізнесу чи особистих цілей. Його багатофункціональність та оптимізованість зробили Telegram ідеальним середовищем для розробки інтерактивних сервісів, які в свою чергу вирішують купу проблем.

Згідно з даними, опублікованими на сайті Forbes.ua за 2024 рік, Telegram став одним із головних сервісів для обміну інформацією серед українців [1]. Цей месенджер відіграє важливу роль у щоденному житті 73% громадян нашої країни, оскільки став основним джерелом отримання новин та актуальної інформації. І за даними сайту Marketer.ua станом на вересень 2024 року цей месенджер посів 2-ге місце серед соціальних мереж в Україні (рис. 1.1) [2].

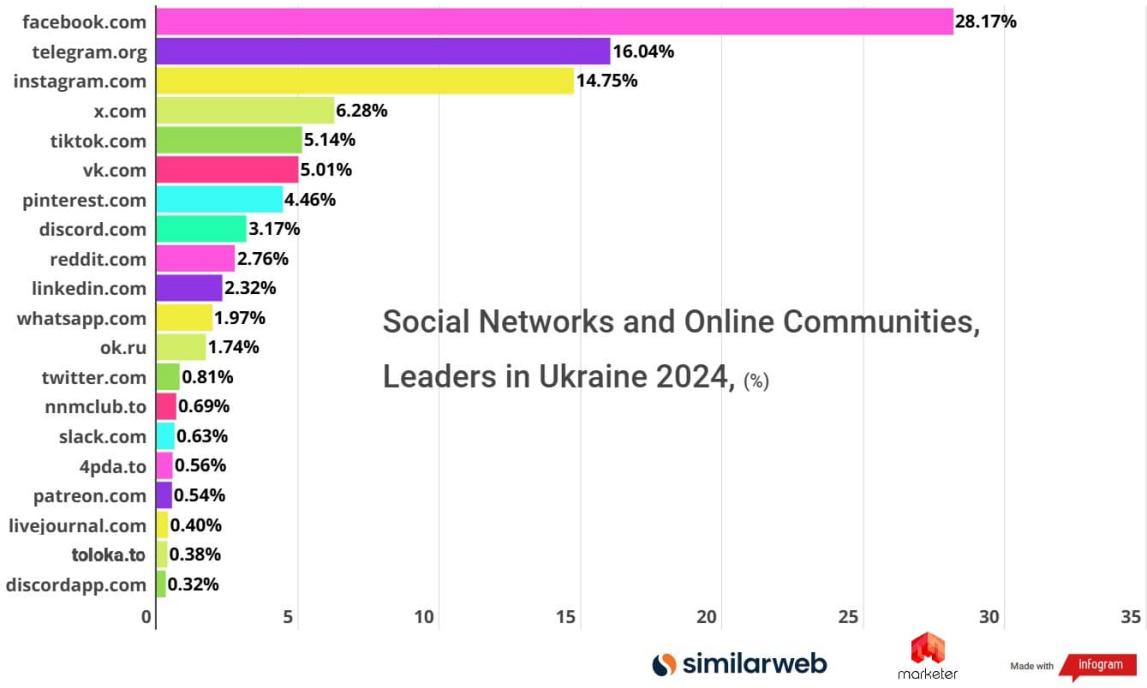



Рис. 1.1 – Діаграма соціальних мереж – лідерів в Україні

Одними з конкурентоспроможних сервісів, які також підтримують можливість створення власних чат-ботів, є Facebook, Instagram, WhatsApp та Viber. Проте їхня конкурентоспроможність закінчується там, де починається сама розробка та інтеграція бота в месенджер. Для визначення кращого месенджера було розроблено порівняльну таблицю:

Таблиця 1.1 – Порівняння месенджерів для роботи з чат-ботами

| Критерій | Telegram | Facebook Messenger | Instagram | Whats App | Viber |
|---|---|---|---|---|---|
| Відкритий API | + | + | – | – | – |
| Легкість створення бота | + | + | – | – | + |
| Доступ до історії повідомлень | + | – | – | – | + |
| Вбудований пошук ботів у додатку | + | – | – | – | ± |

Продовження таблиці 1.1

| Критерій | Telegram | Facebook Messenger | Instagram | Whats App | Viber |
|---|---|---|---|---|---|
| Не вимагає підтвердження бізнесу чи окремої реєстрації | + | – | – | – | – |
| Розсилка повідомлень | + | + | – | + | + |
| Підтримка команд у чаті | + | – | + | – | – |

Таким чином саме Telegram став моїм фаворитом і зайняв перше місце серед соцмереж які надають можливість створювати власних чат-ботів.

Telegram-боти – це програми, які працюють всередині месенджера та можуть взаємодіяти з користувачами. Вони можуть слугувати як простим чат відповідачем, так і повноцінним онлайн магазином, їх можливості в рамках месенджера величезні. А основні переваги полягають у доступності, масштабованості та можливості швидкої інтеграції з іншими сервісами.

У рамках цього дослідження буде розглянуто створення Telegram-бота, який працює з таблицями в реальному часі. Такий функціонал є особливо затребуваним в умовах, коли важливо забезпечити швидкий і зручний доступ до актуальних даних, мати змогу їх швидко редагувати та аналізувати – і все це без потреби залишати межі месенджеру. Це дозволяє суттєво оптимізувати робочі процеси, особливо в командах, які потребують швидкої інтеграції нових працівників в робочий процес.

На сьогоднішній день чат-боти стали важливою частиною цифрового простору. Їх активно використовують не лише для спілкування з клієнтами, а й для автоматизації внутрішніх процесів у компаніях та організаціях. Великі бізнеси щодня створюють нових ботів, які допомагають економити час, знижують навантаження на працівників і підвищують ефективність роботи. Завдяки широким

можливостям інтеграції з різними сервісами, чат-боти перетворилися на універсальний інструмент для вирішення як побутових, так і професійних задач.

Під час вибору теми для цієї дипломної роботи я проаналізував велику кількість існуючих сервісів і ботів. Моєю головною метою був пошук ідеї для створення практичного та корисного програмного забезпечення, яке мало б цінність в сучасних умовах і вирішувало реальні, повсякденні проблеми користувачів. Особливу увагу я приділив вже популярним Telegram-ботам, що активно використовуються в українському просторі.

Ось декілька з них:

Nova Poshta Tracker Bot – допомагає відстежувати посилки за номером декларації, надсилає сповіщення про зміну статусу доставки (рис. 1.2).

Рис. 1.2 – Інтерфейс Telgram бота Nova Poshta

Robota.ua Bot – надає список актуальних вакансій за параметрами. А також надсилає сповіщення про нові пропозиції роботи (рис. 1.3).

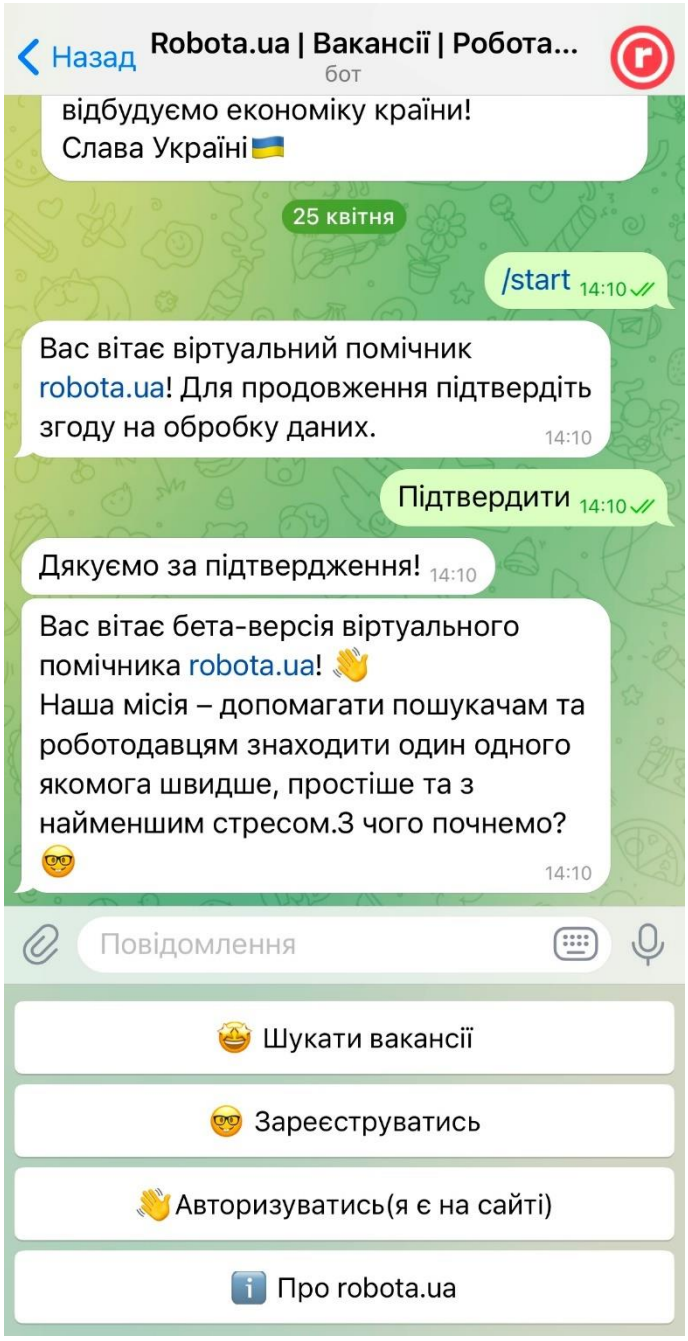



Рис. 1.3 – Інтерфейс Telegram бота Robotaua

Taras_voice_bot – відносно новий сервіс який допомагає конвертувати аудіофайли та голосові повідомлення в текст (рис. 1.4).

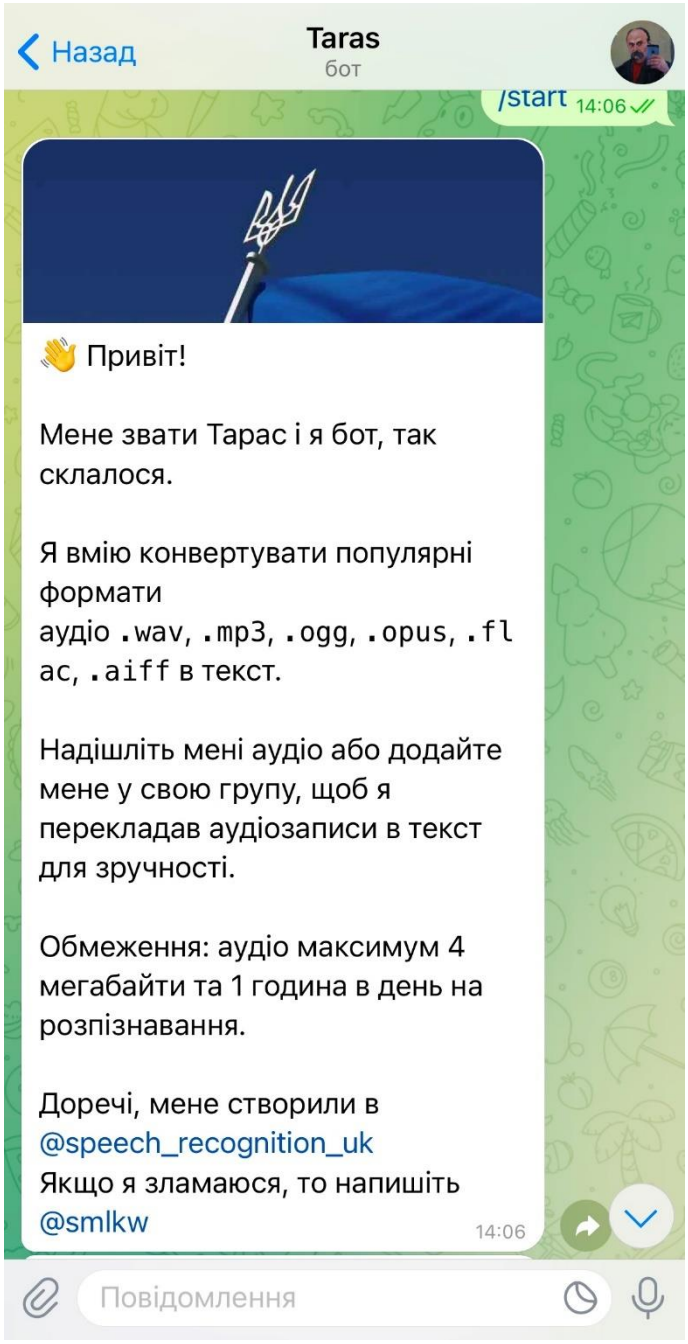



Рис. 1.4 – Інтерфейс Telegram бота Тарас

RailwayBot – бот для пошуку і купівлі квитків на потяги «Укрзалізниці». Користувачі можуть легко перевірити розклад, наявність місць, отримати сповіщення про появу вільних квитків і навіть здійснити покупку в Telegram (рис. 1.5).

Рис. 1.5 – Інтерфейс Telegram бота Railway

OpenDataUABot – надзвичайно корисний інструмент для пошуку відкритих даних про фізичних та юридичних осіб в Україні. Дає змогу швидко отримати інформацію з державних реєстрів, що особливо актуально для журналістів, підприємців та активістів (рис. 1.6).

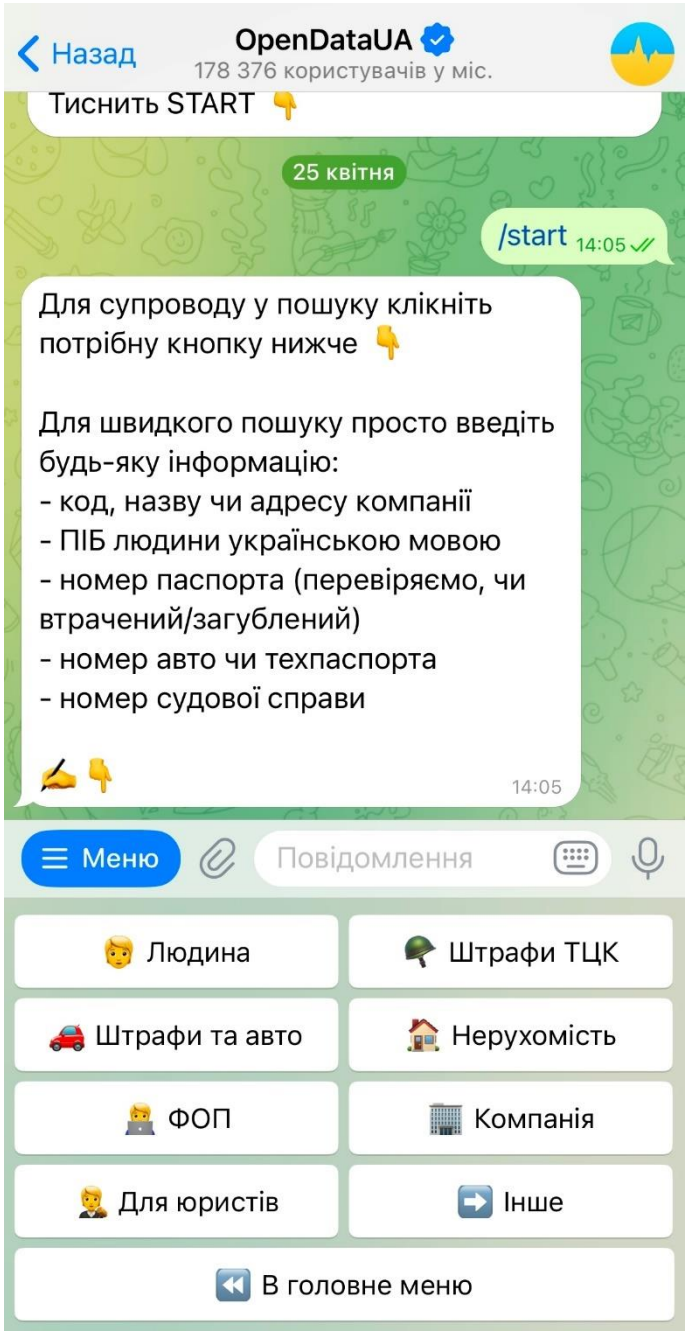



Рис. 1.6 – Інтерфейс Telegram бота OpenDataUA

Ці приклади показують, наскільки цікавими та корисними можуть бути Telegram-боти. Вони вирішують різні завдання й створюють нові варіанти взаємодії з користувачем. Але всі ці боти в більшості орієнтовані на роботу з клієнтами, а не з внутрішнім персоналом компанії. Саме тому я вирішив створити бота, який би вирішував проблему автоматизації та був націлений на спрощення робочих процесів саме для працівників певних компаній чи організацій – Telegram-бот для взаємодії з таблицями в реальному часі. Але тепер основне завдання

полягало у виборі сервісу для роботи з таблицями, який би забезпечив швидку та надійну роботу разом із нашим ботом.

## 1.2 Аналіз існуючих рішень для автоматизації роботи з таблицями

Одним із важливих рішень перед реалізацією проєкту було визначення кращого варіанту для початкової роботи з таблицями. В першу чергу я шукав сервіс, який має легкий доступ до API-ключів для швидкої реалізації підключення таблиць до чат-боту. Далі, не менш важливим пунктом, виступила популярність даних сервісів. За даними сайту Stackby.com [3], Google Sheets має більше 2 мільярдів місячних користувачів, що майже в 3 рази більше за Microsoft Excel, який нараховує близько 750 мільйонів користувачів на місяць. Це дає розуміння того, на боці якого сервісу лежить вибір аудиторії та що буде актуальнішим у найближчий час.

Основний вибір сервісів для роботи з чат-ботом: їхні переваги та недоліки.

Google Sheets. Це онлайн-сервіс електронних таблиць від Google, який повністю безкоштовний для більшості користувачів. Його головні переваги – інтеграція з іншими сервісами Google, підтримка Google Apps Script, робота в реальному часі та простота підключення до Telegram-ботів (рис. 1.7).

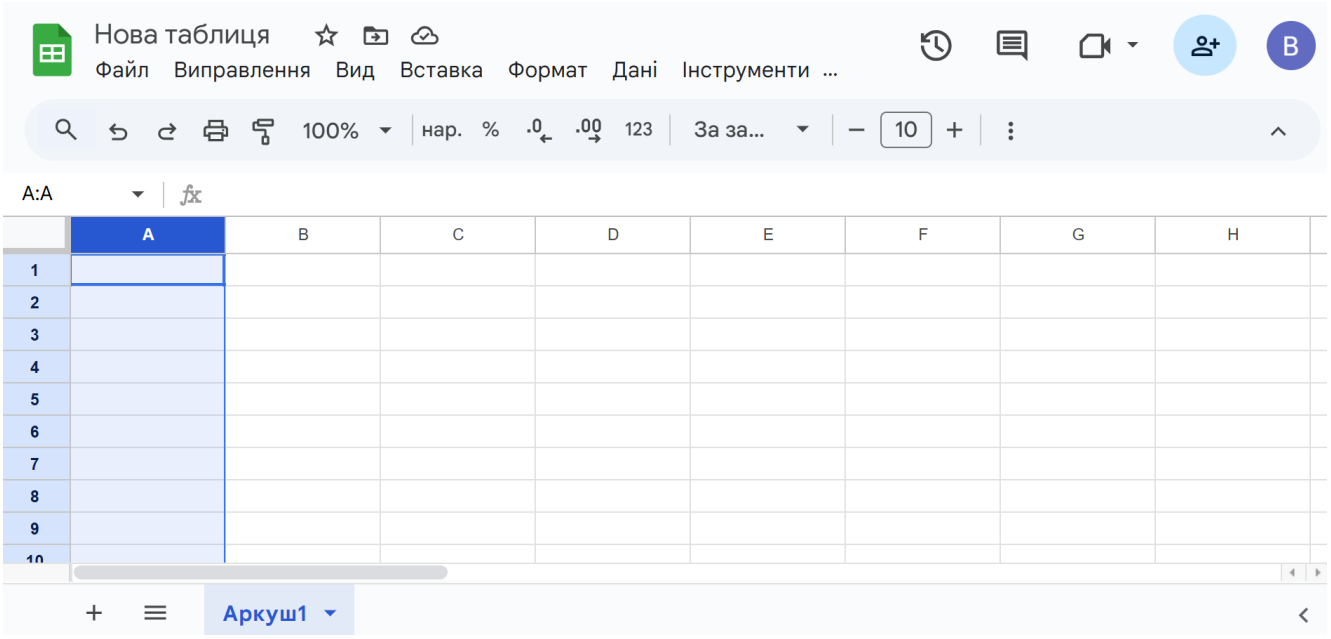



Рис. 1.7 – Інтерфейс сервісу Google Sheets

Microsoft Excel Online. Це хмарна версія Excel. Вона потужна, але трохи важча в освоєнні, і вимагає облікового запису Microsoft. Підключення до ботів можливе, але не таке інтуїтивне, як у Google Sheets (рис. 1.8).

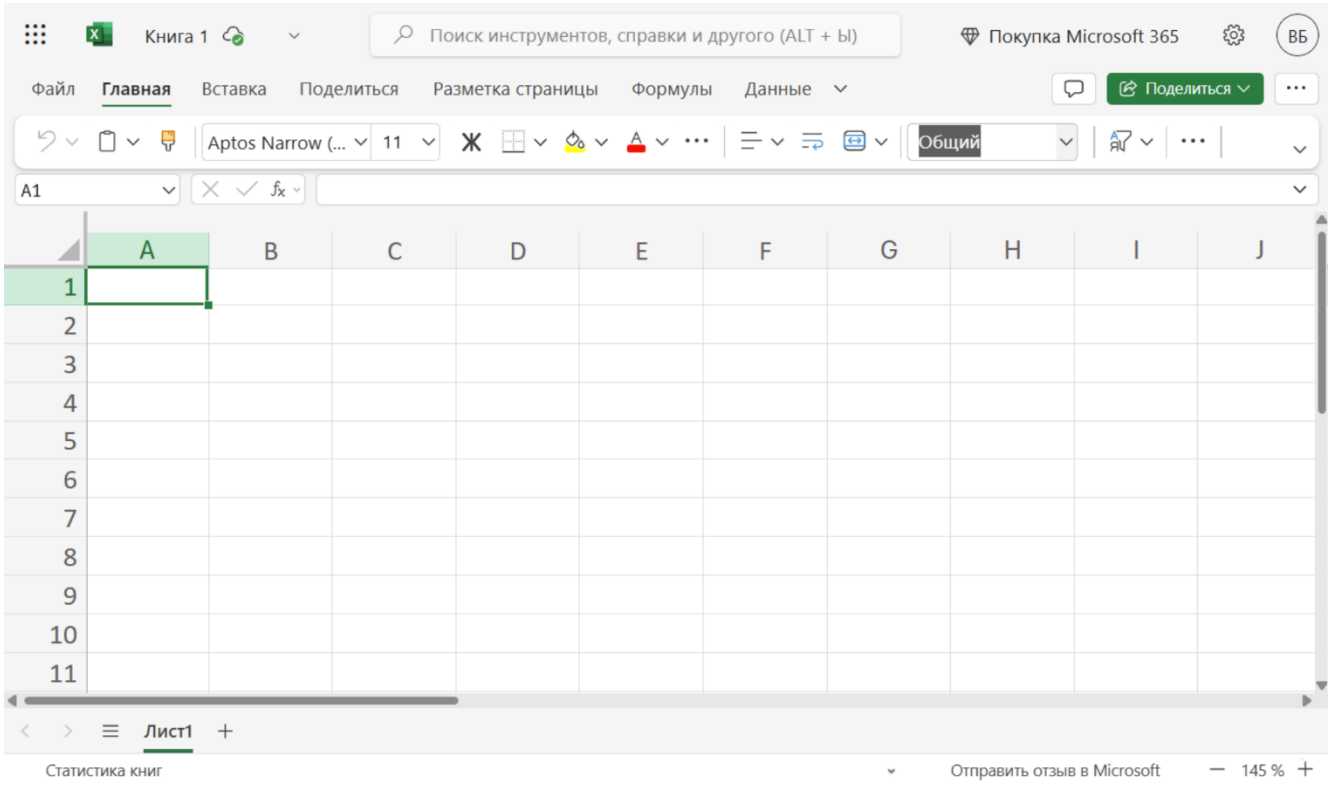



Рис. 1.8 – Інтерфейс сервісу Microsoft Excel

Airtable. Цей сервіс поєднує елементи електронної таблиці та бази даних. Дуже зручно для організації даних і створення власних CRM-систем, але важче інтегрується з Telegram. А також працює виключно за підпискою (рис. 1.9).

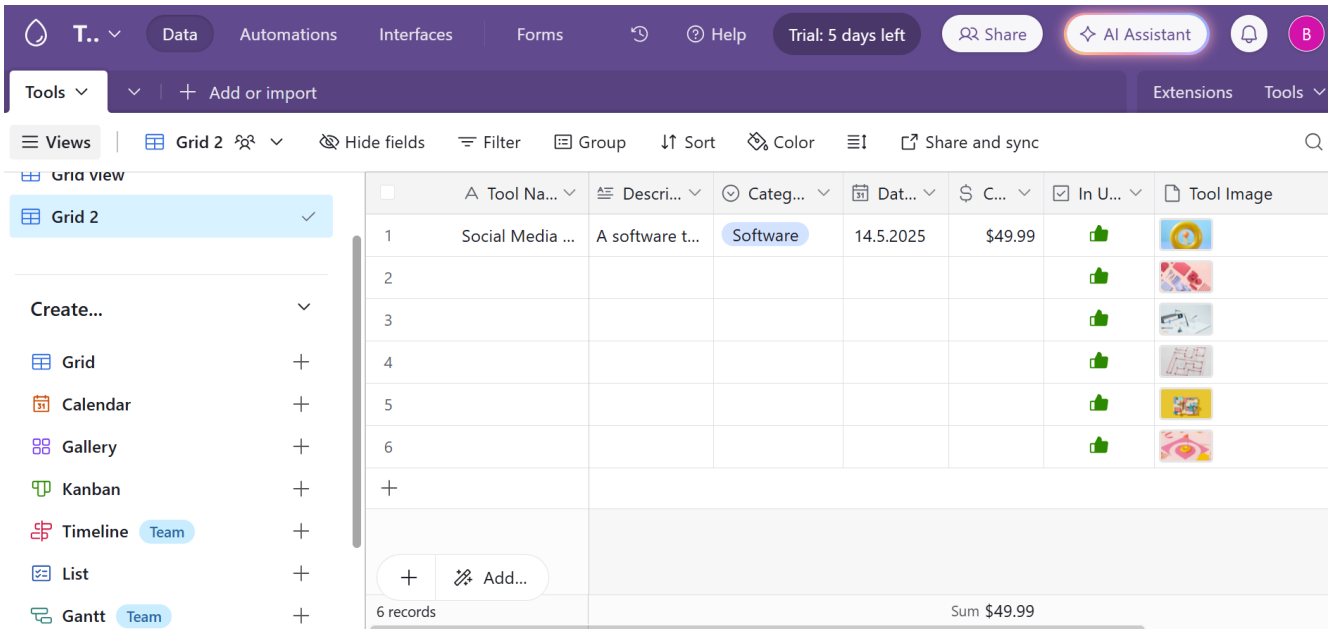



Рис. 1.9 – Інтерфейс сервісу Airtable

Zoho Sheet. Функціонал схожий на Google Sheets, але менш популярний. Є підтримка інтеграцій, проте система не така зручна (рис. 1.10).

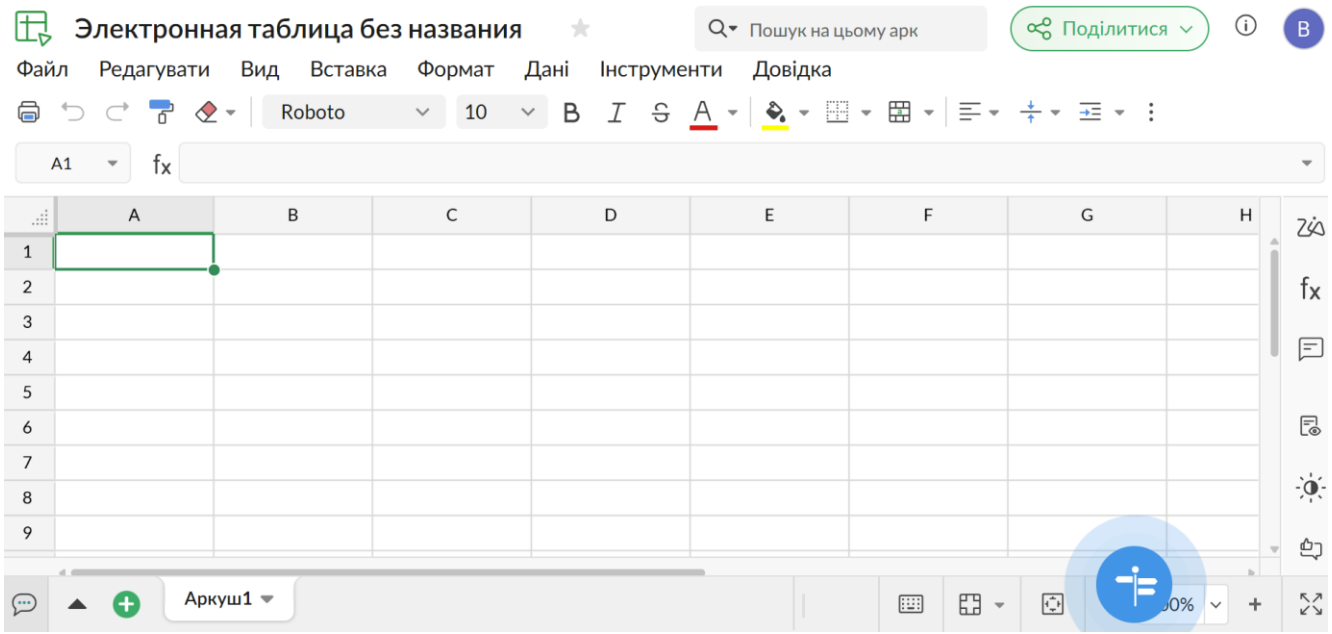



Рис. 1.10 – Інтерфейс сервісу Zoho Sheet

Smartsheet. Потужний інструмент для управління проєктами, але не зовсім підходить для простих таблиць. Добре працює в бізнесі, але вимагає підписки. Для Telegram-ботів використовується рідко, через складнішу інтеграцію (рис. 1.11).

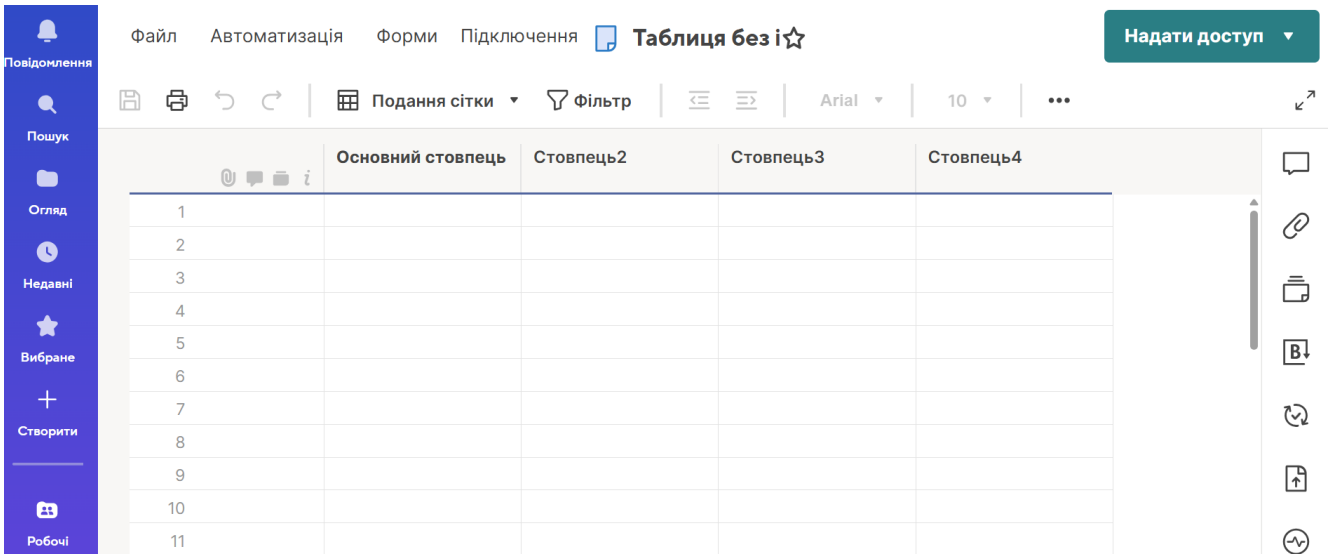



Рис. 1.11 – Інтерфейс сервісу Smartsheet

OnlyOffice. Це рішення більше для локального використання, часто ставлять на корпоративні сервери. З інтеграцією в Telegram усе набагато складніше (рис. 1.12).

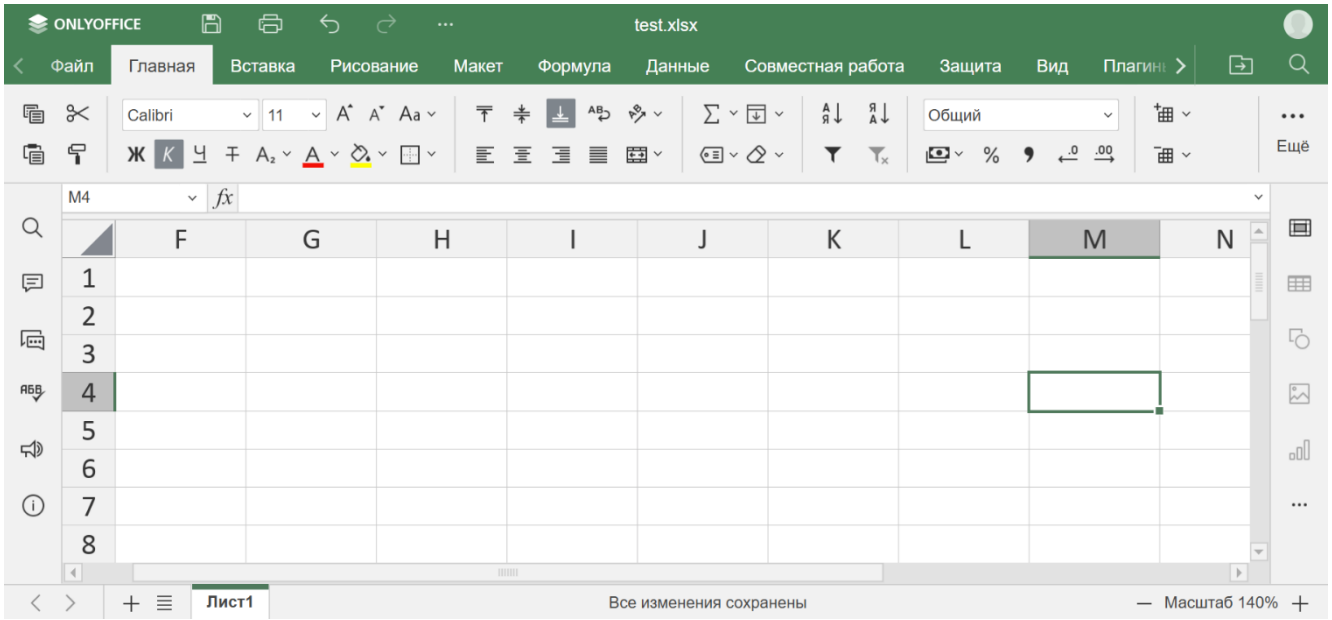



Рис. 1.12 – Інтерфейс сервісу OnlyOffice

Висновок: Google Sheets – найкращий вибір, якщо потрібно зробити Telegram-бота, який буде додавати, редагувати або видаляти дані в таблицях. Його основна сила – це простота інтеграції, шкидкий доступ до гнучного API, підтримка популярних платформ і масове використання, що означає – легко знайти допомогу,

шаблони і готові рішення.

Також варто звернути увагу на те, що наш бот має потенціал не залежити від конкретних сервісів для доступу до таблиць. Завдяки наявності зручного API-підключення, бот може отримати доступ до будь-якої бази даних та продовжити надавати можливість редагувати дані з месенджеру, не втрачаючи при цьому багато часу на реконструкцію коду.

### 1.3 Проблеми та напрямки подальшого розвитку

Незважаючи на чітко визначену ціль по створенню Telegram-бота для роботи з Google Sheets, уже на етапі проєктування стає зрозуміло, що існує низка моменів, які необхідно враховувати. Частина з них пов’язана з технічною реалізацією, інші – з безпекою та можливістю масштабування функціоналу в майбутньому.

Проблеми, що потребують вирішення

Безпека даних. Ще на етапі розробки важливо подумати про захист інформації, адже Google Sheets часто містить конфіденційні дані – фінанси, контакти, аналітику. Зараз планується реалізувати простий захист за допомогою пароля, який зберігатиметься в окремому файлі. У майбутньому варто додати більш надійні рішення: двофакторну автентифікацію, обмеження за IP-адресами або «білий список» користувачів.

Відсутність гнучкої системи прав доступу. Стандартна модель "все або нічого" при доступі до таблиць є неприйнятною для багатьох реальних сценаріїв використання. Уже зараз можна передбачити необхідність розробки ролей, які б дозволяли користувачам взаємодіяти лише з певними стовпцями або рядками, залежно від їхніх завдань.

Продуктивність. Якщо не подбати про оптимізацію ще на початку, то зі збільшенням обсягів таблиць і кількості запитів бот почне працювати повільніше. Це може викликати затримки у відповіді, зависання або дублювання запитів. Щоб цього уникнути, в майбутньому варто проєктувати систему з можливістю

кешування запитів, проміжним збереженням даних у локальній базі та розумною чергою обробки.

Відсутність логування змін. У багатьох задачах важливо знати, хто і коли змінив дані. Якщо бот не фіксуватиме ці дії, виникне плутанина, особливо при колективній роботі. Тому в архітектурі необхідно передбачити реалізацію журналу змін із можливістю його перегляду в Telegram та фільтрації за користувачем чи датою.

Перспективні напрямки розвитку

Інтеграція штучного інтелекту. У майбутньому Telegram-бот може виконувати не лише технічні дії, а й аналітичні. Наприклад:

- знаходити помилки або пропущені значення в таблицях
- прогнозувати зміни (доходи, витрати, динаміку показників)
- голосове керування для базових операцій
- автоматично виправляти найпоширеніші помилки у форматах
- давати підказки користувачам при введенні даних

Автоматичне резервне копіювання. Щоб не втратити важливу інформацію у разі помилки або збою, варто реалізувати систему, яка регулярного буде зберігати копії таблиць у Google Drive з можливістю їх швидкого відновлення.

Масштабування функціональності. Бот має бути гнучким і здатним до розвитку. Бажано врахувати можливість роботи з кількома таблицями та перемиканням між ними в чаті.

Покращення UX. Зручність користувача має стояти на першому місці. У перспективі варто додати:

- інтерактивні підказки при введенні команд
- команду для перегляду останніх змін у таблицях
- систему контролю частоти запитів для зменшення навантаження

Поглиблене логування дій користувачів. Для підвищення безпеки та аналітики важливо зберігати історію запитів кожного користувача – з фіксацією часу, запиту, результату та IP-адреси.

Отже, навіть до запуску Telegram-бота вже зрозуміло, що проєкт має не лише прикладну цінність, а й значний потенціал для розвитку. Грамотно реалізована архітектура дозволить з часом перетворити базовий інструмент на багатофункціонального помічника, який забезпечить швидку, безпечну та зручну роботу з Google Sheets. Головне – закласти правильні основи ще на початку, щоб кожен новий етап розвитку було легко технічно реалізувати.

# РОЗДІЛ 2
# ВИБІР ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ ПРОЄКТУ

## 2.1 Технології для розробки програмного забезпечення

### 2.1.1 Порівняльний аналіз мов програмування

Основним етапом розробки чат-бота став вибір мови програмування, яка мала б найбільше переваг серед інших, була оптимізованою та легкою для вивчення. Під час пошуків я звертав увагу на найпопулярніші мови, з якими вже мав справу раніше. Це були:

- PHP
- Python
- Java
- C++

PHP – це мова, яка в основному використовується для створення вебсайтів. Вона чудово підходить для роботи з серверною частиною сайтів, але в створенні Telegram-ботів має певні недоліки. Також, розробка на PHP для ботів виходить не такою зручною і швидкою.

Java – доволі надійна мова, яку часто використовують у великих проєктах, де потрібна висока стабільність і безпека. Але для невеликих ботів Java – це перебір. Розробка на ній виходить більш складною та важкою, потрібно писати багато коду навіть для простих речей. Тобто для легкого та швидкого створення бота Java взагалі не найкращий варіант.

C++ – це класика програмування, мова для створення максимально продуктивних додатків. Але код на C++ об'ємний, розробка довга, і навіть прості проєкти потребують багато часу на розробку. Для створення Telegram-бота це взагалі не найкращий вибір, оскільки це надто складно і незручно, особливо якщо потрібно швидко щось змінювати або розширювати.

Python – це мова, яка поєднує в собі простоту написання коду, високу швидкість розробки та величезну кількість готових бібліотек, що значно спрощує створення складних проєктів, таких як чат-боти. Крім того, на Python існує безліч зручних фреймворків саме для роботи з Telegram API (наприклад, aiogram або pyTelegramBotAPI). Також важливо, що Python дуже легко масштабувати, що особливо круто, якщо проєкт з часом буде розширюватися.

Проаналізувавши переваги й недоліки різних мов програмування, я вирішив використовувати саме Python, так як на мою думку він є оптимальним вибором для створення Telegram-бота. Він дозволяє працювати швидко, просто й ефективно, не витрачаючи час на зайві технічні складнощі. А велика аудиторія та бібліотеки значно спрощують розробку та підтримку проєкту.

### 2.1.2 Аналіз та обґрунтування вибору середовища програмування

Ще одною важливою частиною розробки є вибір середовища, в якому писатиметься код. Щоб робота була швидкою, середовище має бути зручним, і підтримувати необхідні інструменти. Перед початком створення бота я ретельно продумав, де саме реалізовуватиму проєкт. Основними вимогами були підтримка Python, легкість у використанні та можливість встановлення корисних розширень.

Після аналізу кількох популярних середовищ розробки, я розглянув такі варіанти:

- PyCharm – Дуже функціональний, з купою плюсів для професійної роботи. Але через свою вагу іноді працює повільно, особливо на слабких комп’ютерах.
- Sublime Text – легкий і дуже швидкий текстовий редактор. Однак, без додаткових налаштувань та плагінів – досить обмежений у своїх можливостях.
- Atom – простий та інтуїтивний редактор, який колись був досить популярним. Але останнім часом його майже не оновлюють, і він поступово втрачає актуальність.

- Visual Studio Code – універсальне середовище, яке поєднує все: швидкість, легкість, підтримку Python, зручний інтерфейс і величезний вибір розширень (рис. 2.1).

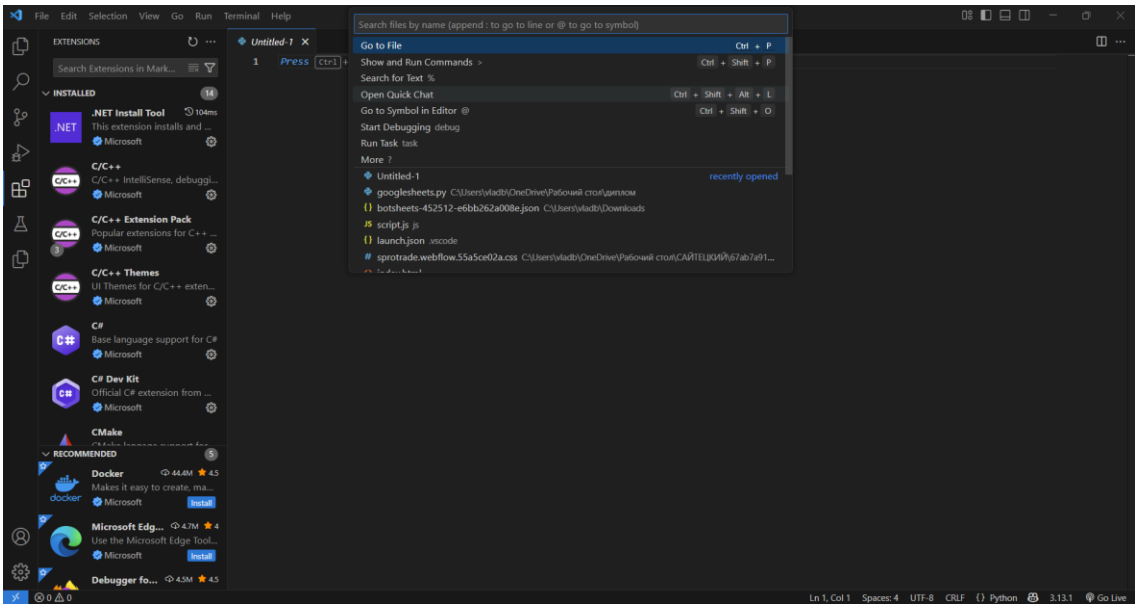



Рис. 2.1 – Інтерфейс середовища Visual Studio Code

У результаті я обрав саме Visual Studio Code. І ось чому:

1. Швидкість і простота – швидкий запуск, інтуїтивний інтерфейс.
2. Гнучкість – можна легко налаштувати «під себе»: змінити вигляд, додати потрібні функції, встановити підтримку форматування, автозаповнення тощо.
3. Величезна кількість розширень – від інтеграції з Git до роботи з API та запуску скриптів у Telegram.

Завдяки цим перевагам, Visual Studio Code став для мене ідеальним робочим інструментом. Писати в ньому приємно, все під рукою, і нічого не гальмує. Саме в цьому середовищі я й буду створювати свого Telegram-бота для роботи з Google Таблицями.

### 2.1.3 Підбір хостингової платформи для розміщення Telegram-бота

Для повноцінної роботи нашого бота він має бути розміщений на онлайн-сервері, який триматиме його постійно активним. Залишати бота увімкненим на Live Server в програмі Visual Studio Code не найкраща ідея. Це не лише незручно, а ще й енергозатратно, бо пристрій, на якому запущений код, також має бути постійно увімкненим. Такий варіант може підходити лише для тестування. Тому я вирішив зайнятися пошуком надійного хостингу, де можна було б розмістити програмне забезпечення нашого бота.

Під час вибору хостингу я визначив для себе кілька основних критеріїв:

1. доступна ціна,
2. ресурси, які надає сервер,
3. популярність і репутація сервісу серед користувачів.

Важливо було обрати не просто розрекламований хостинг, а справді перевірений продукт, який би пропонував найкращі умови для роботи.

Серед варіантів українських хостингів, які я розглядав, були такі:

1. Tuthost – один із найстаріших українських провайдерів. Пропонує гарний вибір тарифів, стабільні сервери та адекватну підтримку (рис. 2.2).

| | Плюс<br>4.00 $ /міс. | Бізнес<br>6.00 $ / міс. | Великий<br>8.00 $ /міс. | Люкс<br>10.00 $ /міс. |
|---|---|---|---|---|
| Місце на диску SSD | 3 Гб | 10 Гб | 20 Гб | 30 Гб |
| Сайтів на обліковому записі | 5 | 15 | 30 | 100 |
| Бази MySQL | Необмежено | Необмежено | Необмежено | Необмежено |
| Ліміт оперативної пам'яті | 256 Мб | 512 Мб | 1024 Мб | 1536 Мб |
| Ліміт процесора | 1600 Мгц | 2200 Мгц | 3000 Мгц | 3000 Мгц |
| PHP 5.2 – 8.2 | ✓ | ✓ | ✓ | ✓ |
| Python 2.7 - 3.8 | ⊘ | ✓ | ✓ | ✓ |
| Домен безкоштовно | ✓ | ✓ | ✓ | ✓ |
| SSL сертифікат безкоштовно | ✓ | ✓ | ✓ | ✓ |
| Конструктор сайтів та 200 шаблонів | ✓ | ✓ | ✓ | ✓ |
| Інсталятор скриптів Installatron | ✓ | ✓ | ✓ | ✓ |
| Розташування хостингу (на вибір) | | | | |
| Безкоштовний тестовий період | Тест 30 днів | Тест 30 днів | Тест 30 днів | Тест 30 днів |

Рис. 2.2 – Ресурси та тарифи хостингу Tuthost

2. Ukraine.com.ua – дуже популярний сервіс з дешевими цінами та гнучкою системою налаштувань (рис. 2.3).

| | Швидкий<br>Базовий план для невеликих сайтів | Кращий<br>Оптимально для кількох сайтів | Якісний<br>Готове рішення для веб студій і великих проектів |
|---|---|---|---|
| Місце на NVMe-диску | 10 GB | 20 GB | 30 GB |
| Безкоштовний SSL-сертифікат | ✓ | ✓ | ✓ |
| Конструктор сайтів | ✓ | ✓ | ✓ |
| Apache, OpenLiteSpeed | ✓ | ✓ | ✓ |
| MySQL 5.7 / 8.0 * | ✓ | ✓ | ✓ |
| Сайтів | 1 | 5 | ∞ |
| Memory limit (RAM) | 512 MB | 1024 MB | 1536 MB |
| Розташування серверів | | | |
| 30-денне повернення грошей | від 190 ₴/міс<br>~~291 ₴/міс~~<br>Замовити<br>5 днів на тест | від 354 ₴/міс<br>~~540 ₴/міс~~<br>Замовити<br>5 днів на тест | від 507 ₴/міс<br>~~774 ₴/міс~~<br>Замовити<br>5 днів на тест |

Рис. 2.3 – Ресурси та тарифи хостингу Ukraine.com.ua

3. Cityhost – надійний хостинг із гарною репутацією, простим управлінням і хорошою тех-підтримкою (рис. 2.4).

| Місце на диску NVME | 10 Gb | 20 Gb | 40 Gb | 60 Gb |
|---|---|---|---|---|
| Вартість продовження | 121.86 грн/міс. | 171.90 грн/міс. | 256.68 грн/міс. | 322.56 грн/міс. |
| Сайтів/Піддоменів | ∞ | ∞ | ∞ | ∞ |
| Розташування сервера. | 🇩🇪 🇫🇮 🇺🇦 | 🇩🇪 🇫🇮 🇺🇦 | 🇩🇪 🇫🇮 🇺🇦 | 🇩🇪 🇫🇮 🇺🇦 |
| Поштових скриньок | ∞ | ∞ | ∞ | ∞ |
| Безкоштовний домен 🎁 | до 250 грн | до 350 грн | до 450 грн | до 450 грн |
| SSL сертифікат | FREE | FREE | FREE | FREE |
| PHP memory_limit | 512 MB | 1024 MB | 1024 MB | 1536 MB |
| MySQL 5.7, 8.0 | ∞ | ∞ | ∞ | ∞ |
| PHP 5.2-8.4 | ✓ | ✓ | ✓ | ✓ |
| Node.js | ✓ | ✓ | ✓ | ✓ |
| HTTP/2 | ✓ | ✓ | ✓ | ✓ |
| Конструктор сайтів | ✓ | ✓ | ✓ | ✓ |
| Технічна підтримка | 24/7 | 24/7 | 24/7 | 24/7 |
| Тестовий період | 10 днів | 10 днів | 10 днів | 10 днів |

Рис. 2.4 – Ресурси та тарифи хостингу Cityhost.ua

4. Hostpro – більше орієнтований на бізнес-рішення, трохи дорожчий, але з дуже стабільною роботою та великим вибором послуг (рис. 2.5).

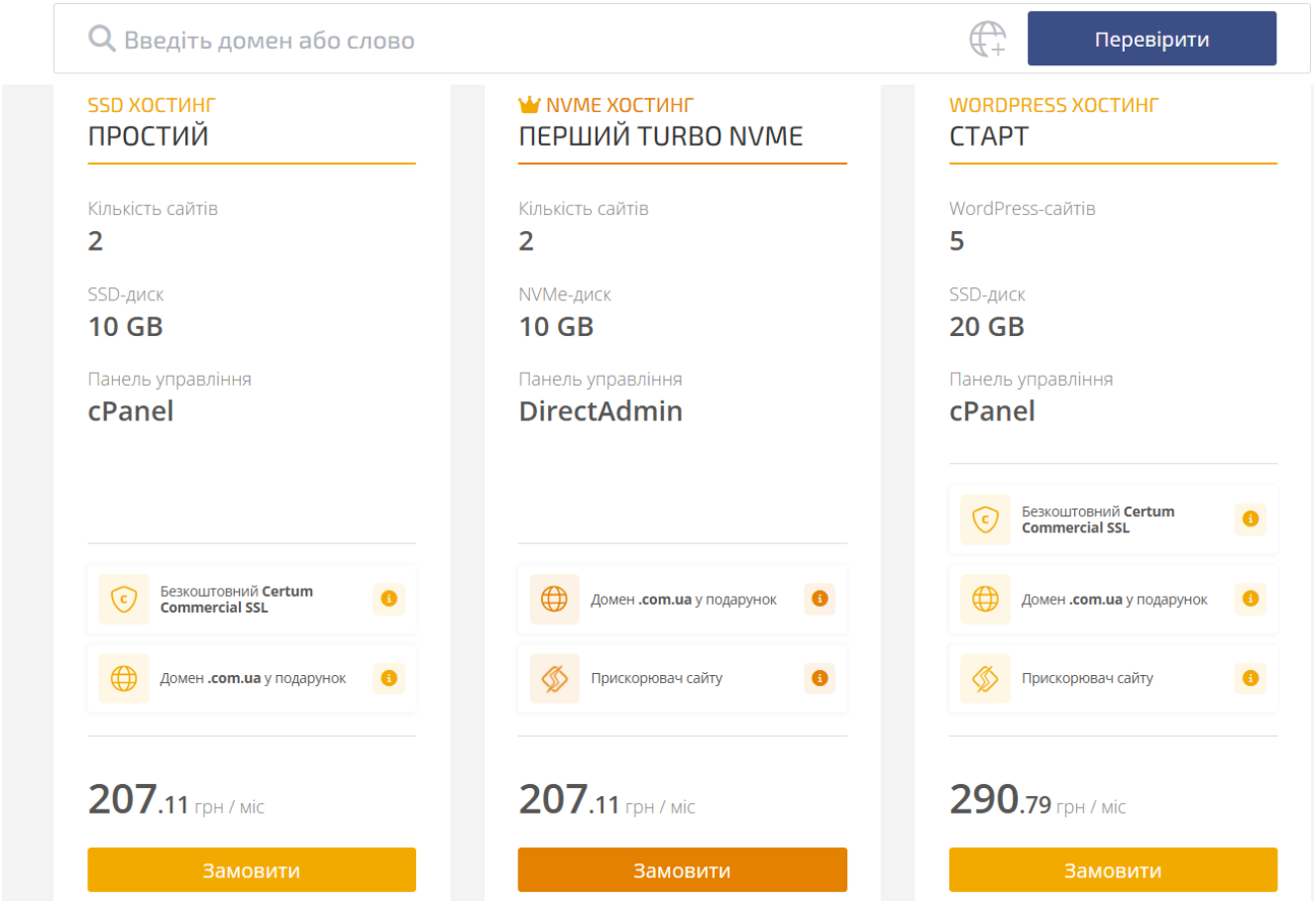



Рис. 2.5 – Ресурси та тарифи хостингу Hostpro.ua

Після порівняння всіх варіантів я обрав Ukraine.com.ua. Тому що він показав чудове співвідношення ціни й якості, дозволяє швидко розгорнути програмне забезпечення, має підтримку Docker-контейнерів і гарантує високу стабільність роботи, що дуже важливо для постійно активного бота.

Висновок: для стабільної та професійної роботи наш Telegram-бот повинен бути розміщений на надійному хостингу. Вибравши Ukraine.com.ua, я отримав оптимальне рішення з хорошою швидкістю, доступною ціною та впевненістю, що наш бот буде працювати без збоїв.

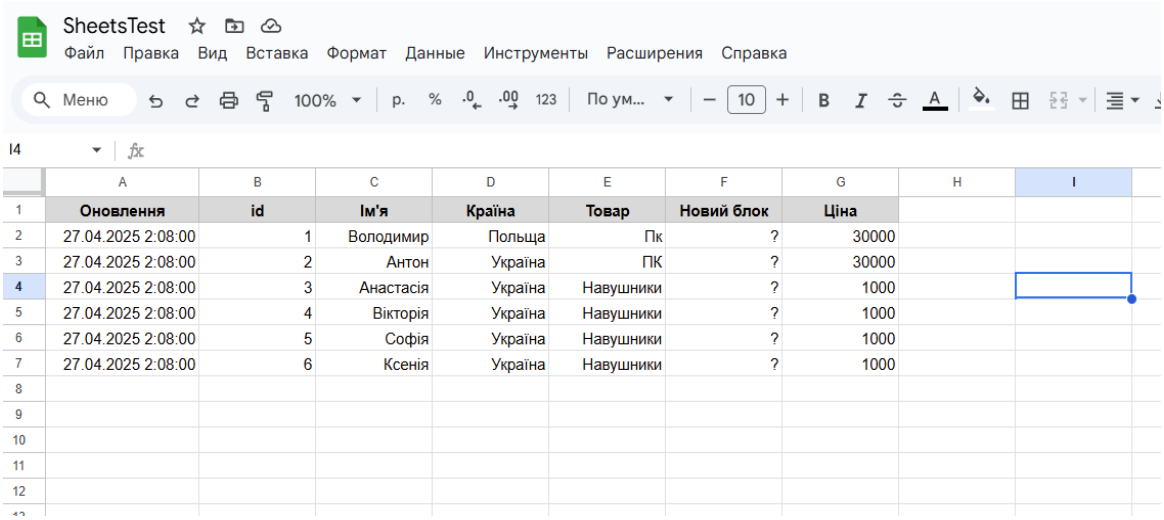
# РОЗДІЛ 3
# РОЗРОБКА ТА РЕАЛІЗАЦІЯ TELEGRAM-БОТА

## 3.1 Функціональні можливості та принципи взаємодії бота

Одним із найважливіших кроків під час створенні нашого сервісу було детальне продумування всіх його можливостей. Основна задача бота – робота з таблицями в реальному часі, але важливо також показати переваги використання саме месенджера для оновлення інформації. Тому я одразу обрав таке середовище, яке повністю розкриє потенціал бота та його необхідність в сучасному світі.

Таким середовищем став бізнес, що включає облік клієнтів та менеджера, який у процесі роботи вручну заповнює базу даних. Це може бути магазин, поштове відділення, white-list для певних організацій або навіть особисті замітки. Особливо актуальним рішення буде для бізнесів, що тільки розпочинають свою діяльність і потребують ефективного та недорогого програмного забезпечення для автоматизації робочих процесів. Саме для таких випадків і створений наш бот.

Бот повинен працювати із шаблоном клієнтської бази даних у Google Sheets, який буде поступово заповнюватися через месенджер (рис. 3.1).


SheetsTest

Файл Правка Вид Вставка Формат Данные Инструменты Расширения Справка

Меню 100% р. % .0 .00 123 По ум... 10

I4

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Оновлення | id | Ім'я | Країна | Товар | Новий блок | Ціна | | |
| 2 | 27.04.2025 2:08:00 | 1 | Володимир | Польща | Пк | ? | 30000 | | |
| 3 | 27.04.2025 2:08:00 | 2 | Антон | Україна | ПК | ? | 30000 | | |
| 4 | 27.04.2025 2:08:00 | 3 | Анастасія | Україна | Навушники | ? | 1000 | | |
| 5 | 27.04.2025 2:08:00 | 4 | Вікторія | Україна | Навушники | ? | 1000 | | |
| 6 | 27.04.2025 2:08:00 | 5 | Софія | Україна | Навушники | ? | 1000 | | |
| 7 | 27.04.2025 2:08:00 | 6 | Ксенія | Україна | Навушники | ? | 1000 | | |
| 8 | | | | | | | | | |
| 9 | | | | | | | | | |
| 10 | | | | | | | | | |
| 11 | | | | | | | | | |
| 12 | | | | | | | | | |

Рис. 3.1 – Шаблон заповненої таблиці в Google Sheets

Наш шаблон клієнтської бази даних орієнтований на онлайн-магазин. Він міститиме такі стовпці:

- Оновлення – (завжди перший стовпець) – відображає дату та час створення або оновлення інформації в рядку
- Id – (другий стовпець) – унікальний ідентифікатор клієнта, що призначається вручну
- Ім’я – ім'я клієнта
- Країна – країна чи місце доставки товару
- Товар – замовлений товар
- Новий блок – призначений для додаткової інформації
- Ціна – вартість замовлення

Після визначення цілі нашого бота та його взаємодії з таблицею, потрібно також реалізувати базові елементи комунікації з користувачем. Для цього я створив прості діаграми послідовності у сервісі PlantUML, які показують основні сценарії роботи бота:

Автентифікація (рис. 3.2). Ця діаграма демонструє сценарій авторизації користувача у Telegram-боті. Після введення команди /start, бот запитує пароль. Якщо користувач вводить правильний пароль – відбувається успішна авторизація, і бот запрошує ввести ID клієнта, надсилаючи також кнопку для відкриття таблиці. У разі введення неправильного пароля бот просить спробувати ще раз.

Рис. 3.2 – Діаграма послідовності процесу автентифікації

Пошук даних (рис. 3.3). Ця діаграма описує процес пошуку даних клієнта за його id. Користувач надсилає ідентифікатор, бот звертається до Google Sheets API для отримання інформації. Якщо дані знайдено – бот надсилає користувачу відповідну інформацію та підказку для оновлення. Якщо дані не знайдено – повідомляє, що id не знайдено, і пропонує створити новий запис.

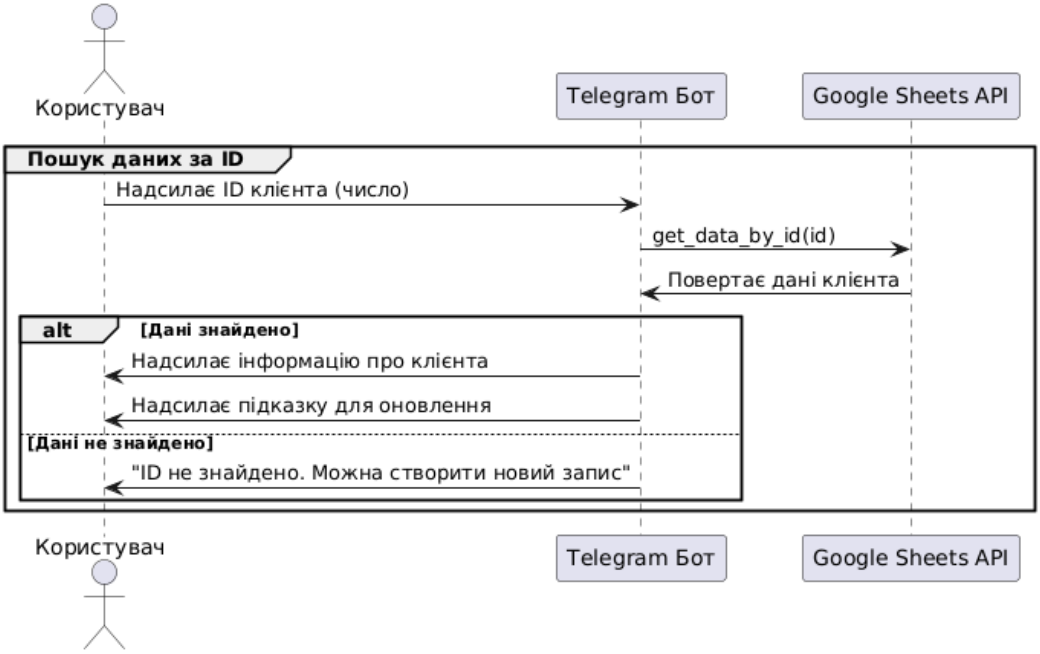



Рис. 3.3 – Діаграма послідовності процесу пошуку даних

Оновлення даних (рис. 3.4). Ця діаграма демонструє процес оновлення або створення даних у Telegram-боті. Користувач надсилає повідомлення з id і новими даними. Якщо вказано слово "видалити", бот сприймає це як команду на видалення стовпця і передає запит до Google Sheets. В іншому випадку бот або оновлює існуючий запис (якщо id знайдено), або створює новий (якщо такого запису ще немає), після чого повідомляє користувача про результат.

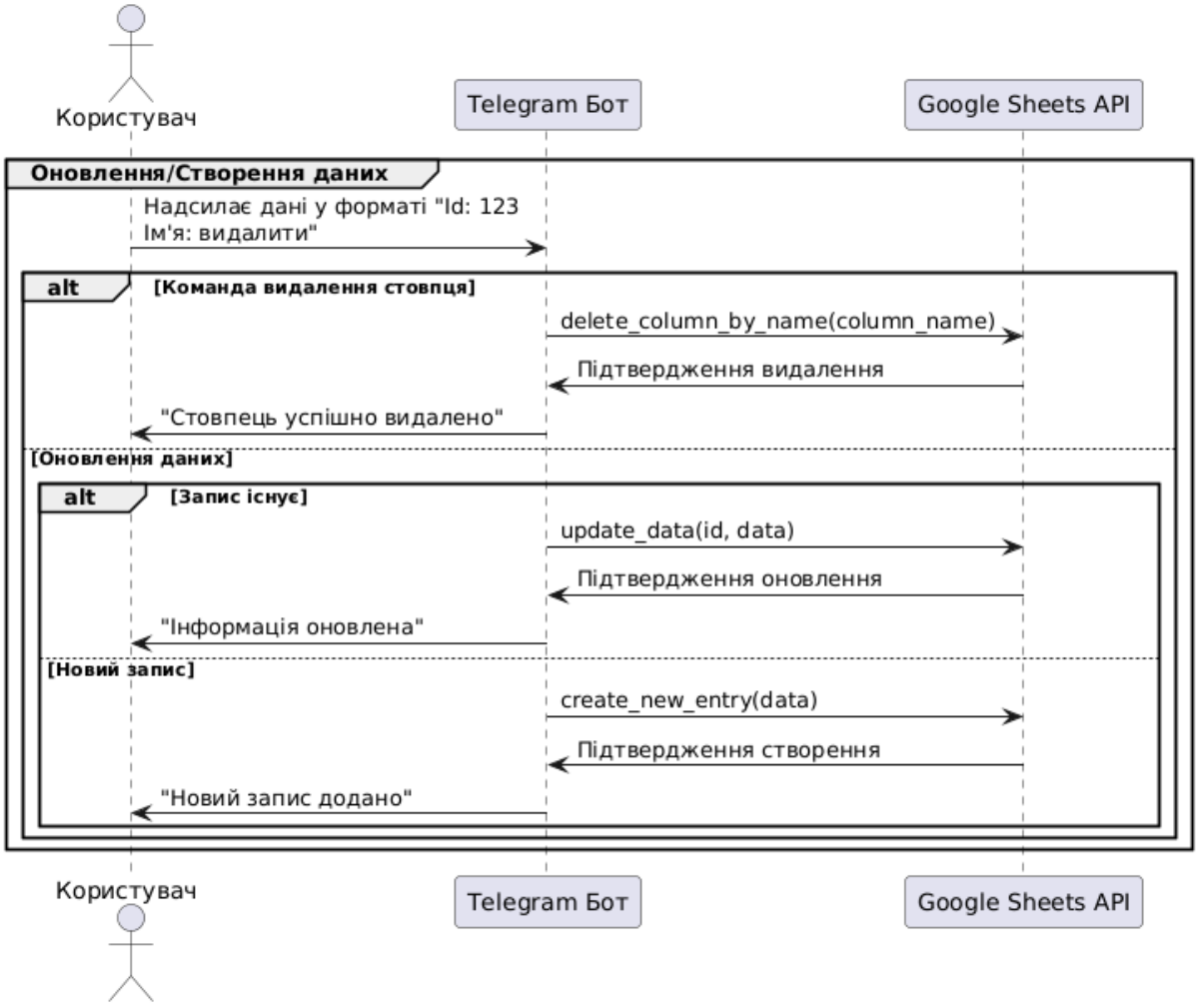



Рис. 3.4 – Діаграма послідовності процесу оновлення даних

### 3.1.1 Внутрішні особливості функціональних можливостей бота

Щоб убезпечити дані від несанкціонованого доступу, бот повинен мати етап обов'язкової авторизації. Перед початком роботи кожен користувач повинен ввести правильний пароль, який зберігається в окремому текстовому файлі password.txt. Це рішення буде корисне тим, що дозволить змінювати пароль, не вносячи зміни в код.

Ще одним практичним рішенням є автоматичне зміщення стовпців при видаленні одного із раніше створених. Це дозволяє зберегти зрозумілу структуру таблиці та зменшує ймовірність виникнення помилок.

Також однією з важливих функцій бота має стати фіксація дати та часу під час оновлення рядків. Така функція логування стане корисною для відстеження

часу створення або редагування замовлення, що значно спростить пошук помилок у таблиці для користувача.

Не менш важливим буде також впровадження функції нескінченного опитування повідомлень, яка забезпечить безперервну взаємодію користувачів із ботом без потреби у ручному перезапуску після збоїв або перезавантаження серверу. Завдяки цьому бот зможе працювати у фоновому режимі, постійно перевіряючи наявність нових повідомлень.

## 3.2 Необхідні бібліотеки та модулі для Telegram-бота

На початку роботи над ботом я імпортував усі необхідні бібліотеки (рис. 3.5). Для взаємодії з Telegram використовується бібліотека telebot, яка дозволяє просто організувати обробку повідомлень і команд. Для роботи з Google Sheets я підключив бібліотеку gspread, а для авторизації в Google API – oauth2client. Також я імпортував InlineKeyboardMarkup та InlineKeyboardButton для створення інтерактивних кнопок в чаті бота і модуль datetime, який потрібен для фіксації дати та часу змін у таблиці (рис. 3.5).

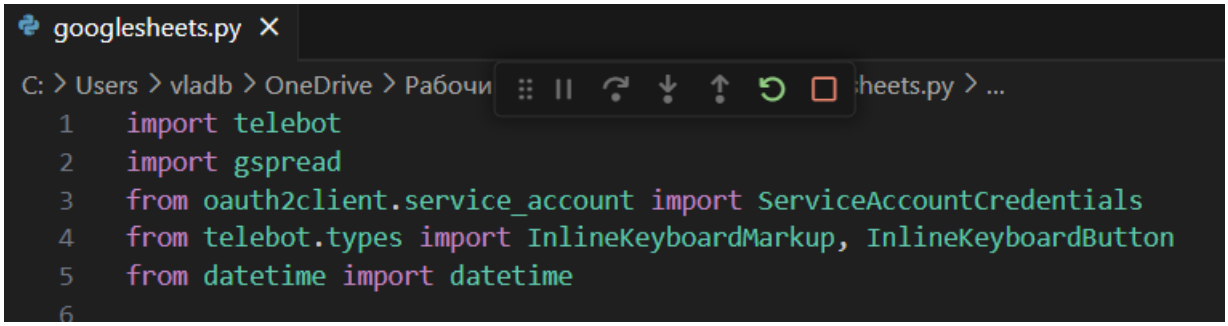



Рис. 3.5 – Бібліотеки та модулі майбутнього бота

## 3.3 Налаштування та використання API-ключів

API-ключі – як для підключення бота в Telegram, так і для підключення таблиць до нашого програмного забезпечення – є надзвичайно важливою деталлю. Без них неможливо забезпечити безпечну та стабільну роботу системи.

У першу чергу, щоб отримати API-ключ для роботи нашого чат-бота в Telegram, потрібно скористатися внутрішнім інструментом месенджера. Як я вже

писав у першому розділі під час порівняння месенджерів, створення Telegram API-ключів доступне кожному користувачу. Для цього не потрібні додаткові реєстрації, підтвердження бізнесу чи перехід на спеціальний тип акаунта – все відбувається в самому месенджері.

Етапи створення API-ключа в Telegram

1. Пошук офіційного Telegram-бота BotFather (рис. 3.6). Саме через нього Telegram дозволяє створювати та керувати своїми ботами.

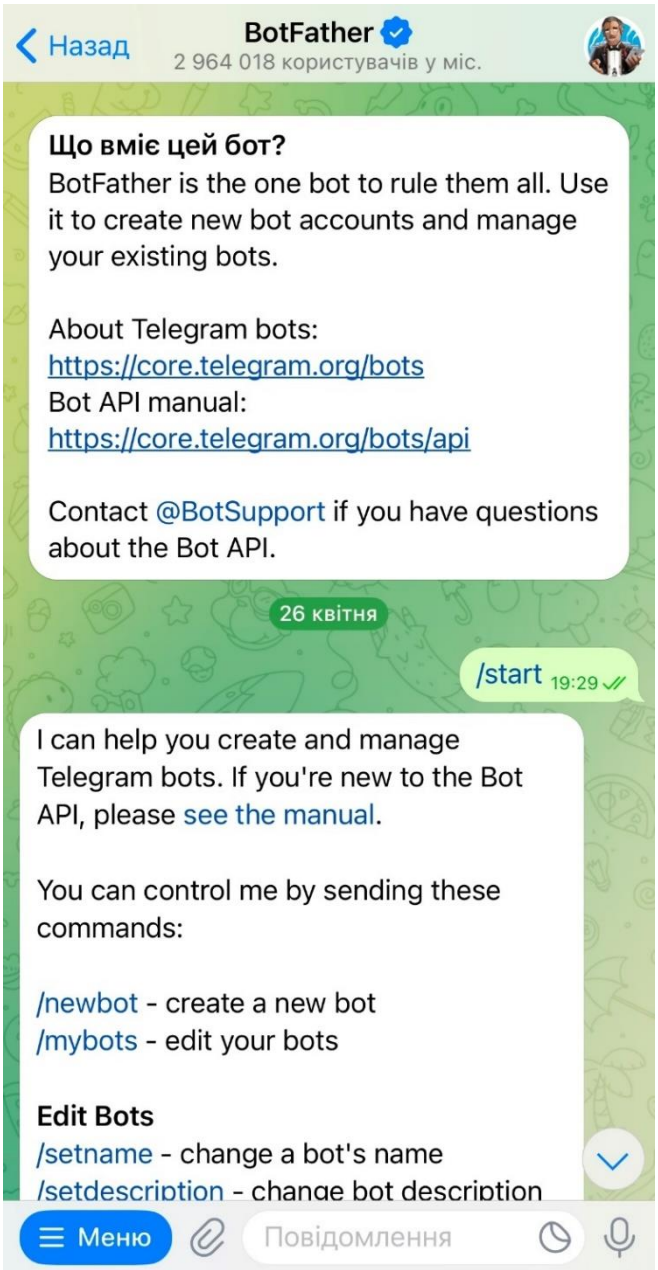



Рис. 3.6 – Інтерфейс BotFather

2. Після знаходження BotFather потрібно почати з ним діалог

3. Для створення нового бота обрати команду /newbot.

4. BotFather запропонує ввести ім'я для нового бота, а також унікальний нікнейм, який має закінчуватись на bot

5. Після цього бот згенерує унікальний API Token – саме цей токен нам і потрібно буде використовувати в нашому коді для підключення до Telegram (рис. 3.7).

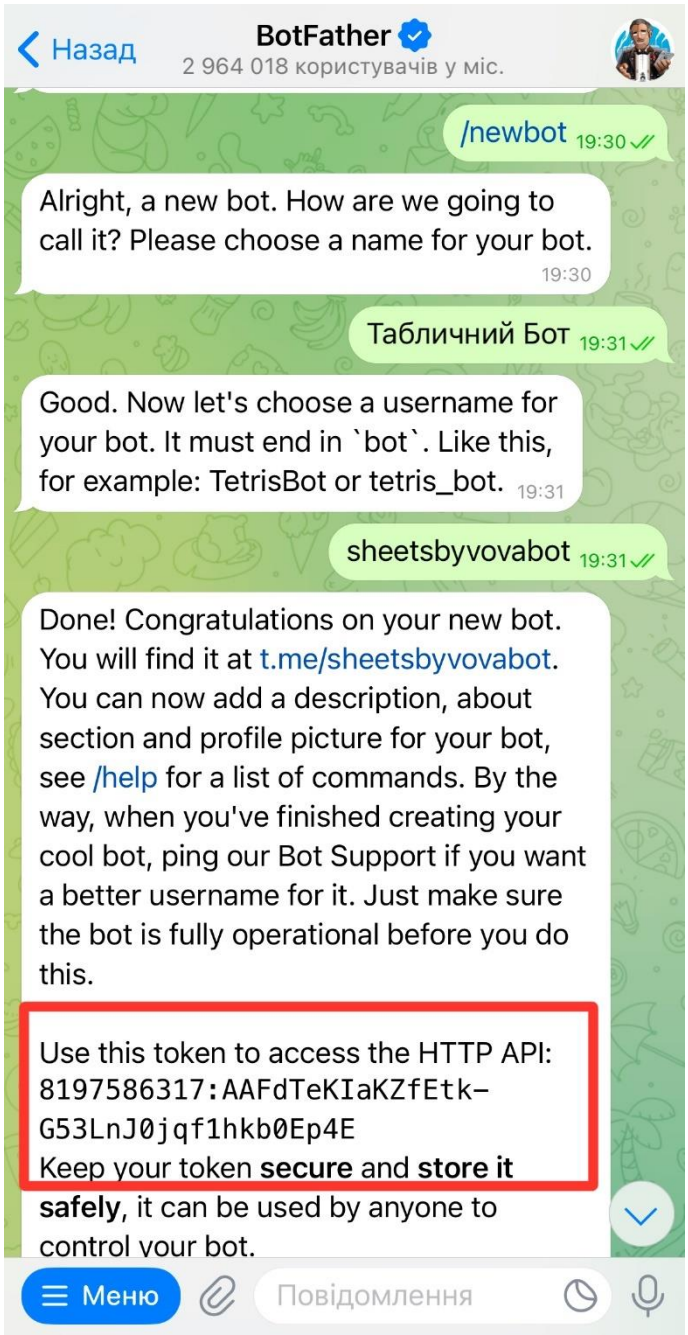



Рис. 3.7 – Приклад процесу генерації API ключа через BotFather

Отримавши API для Telegram-бота, можна перейти до більш складного завдання – створення API-ключів Google Sheets для подальшої інтеграції з нашим ботом. Для цього я буду використовувати сервіс Google Cloud.

Процес налаштування виглядає так:

1. Переходимо на офіційний сайт Google Cloud Console.
2. Створюємо новий проєкт
3. Далі у розділі "API та сервіси" потрібно увімкнути дві важливі бібліотеки: Google Sheets API (рис. 3.8) та Google Drive API (рис. 3.9). Перша дозволить нам працювати з таблицями, а друга надасть права на доступ до файлів на Google Диску.

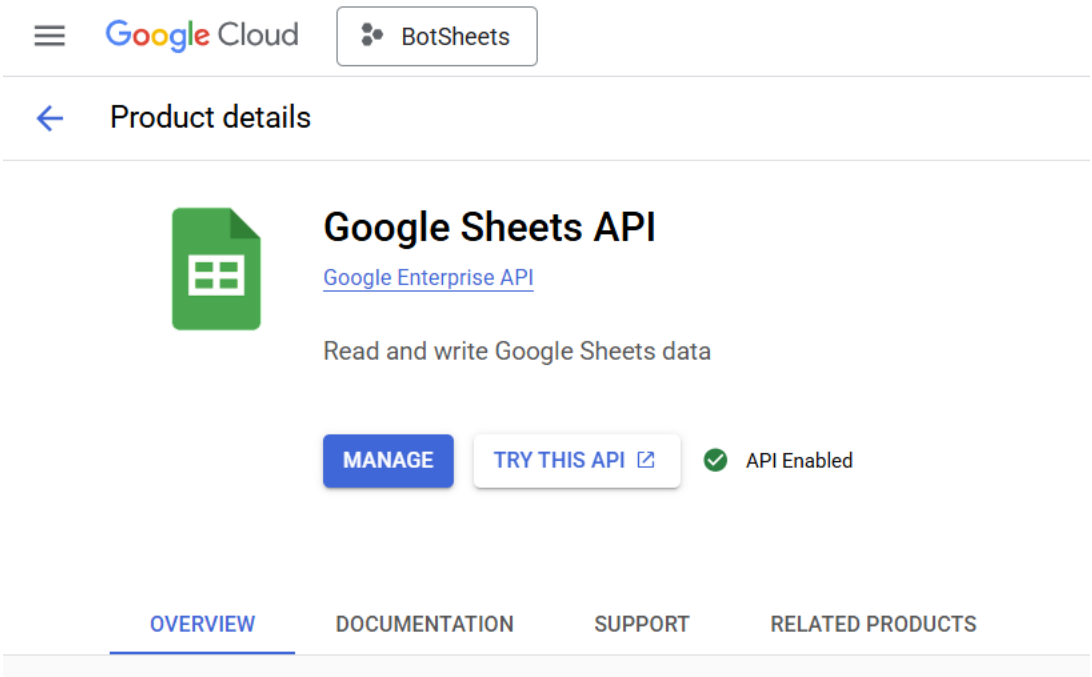



Рис. 3.8 – Бібліотека Google Sheets API

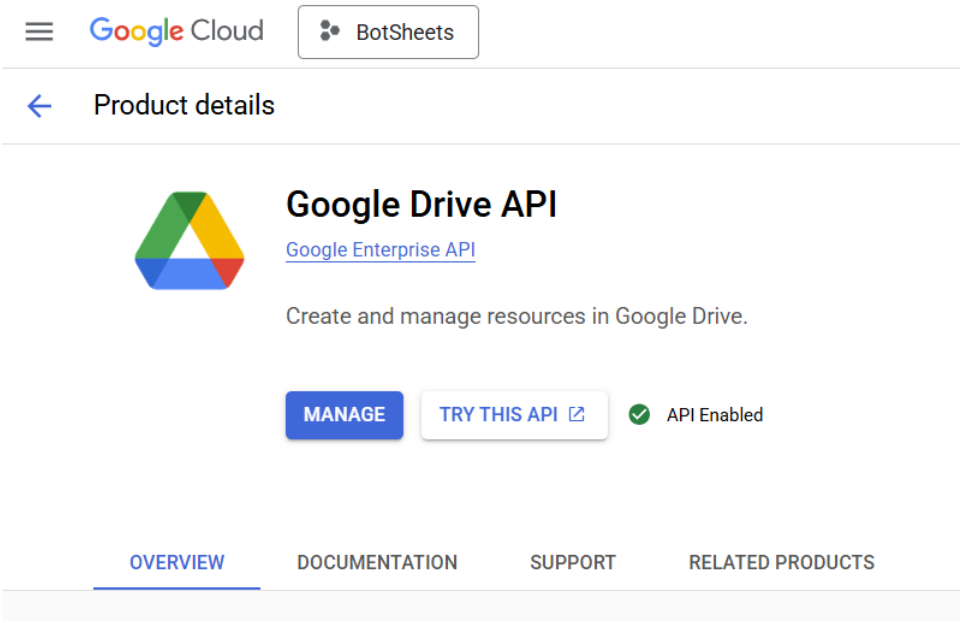



Рис. 3.9 – Бібліотека Google Drive API

4. Після активації API необхідно створити облікові дані. Для інтеграції нашого бота найкраще створити сервісний акаунт (рис. 3.10).

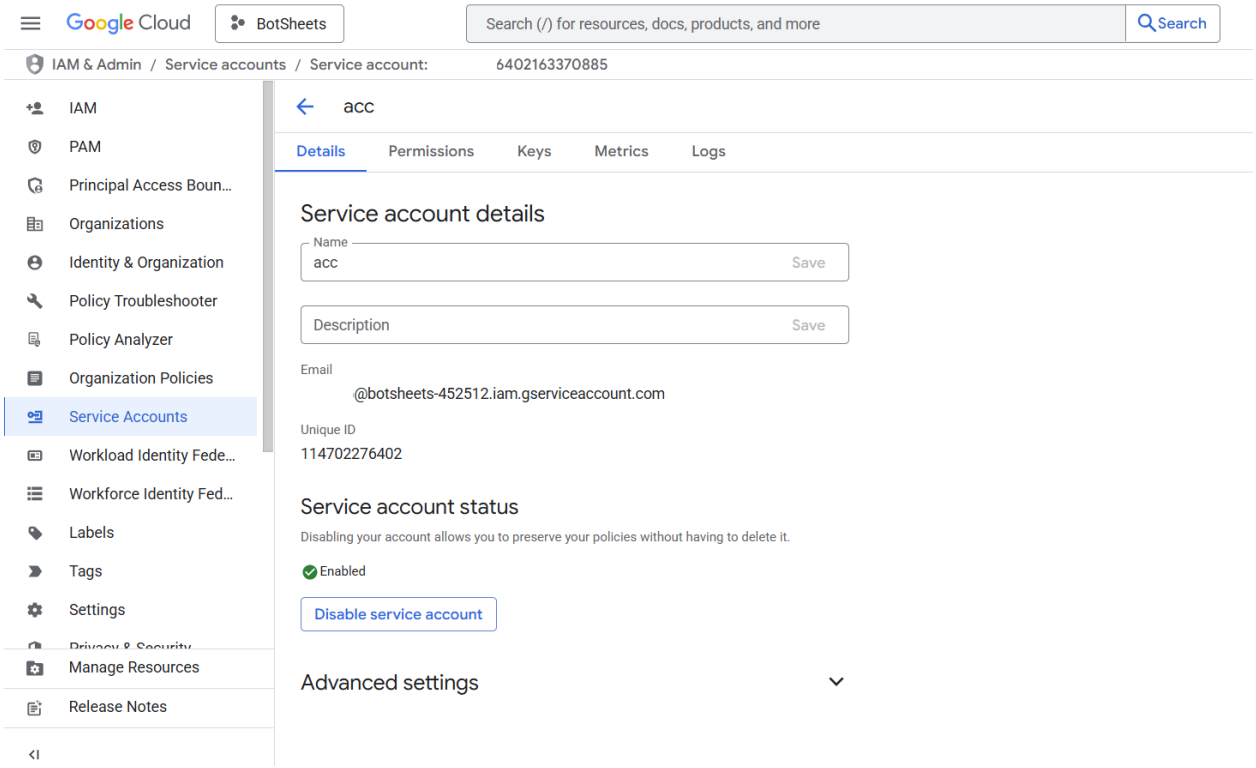



Рис. 3.10 – Бібліотека Google Drive API

5. Під час створення сервісного акаунту потрібно задати назву та опис, після чого обрати роль. Саме для нашої роботи з таблицями потрібно додати роль

"Редактор" а не "Читач" (рис. 3.11).

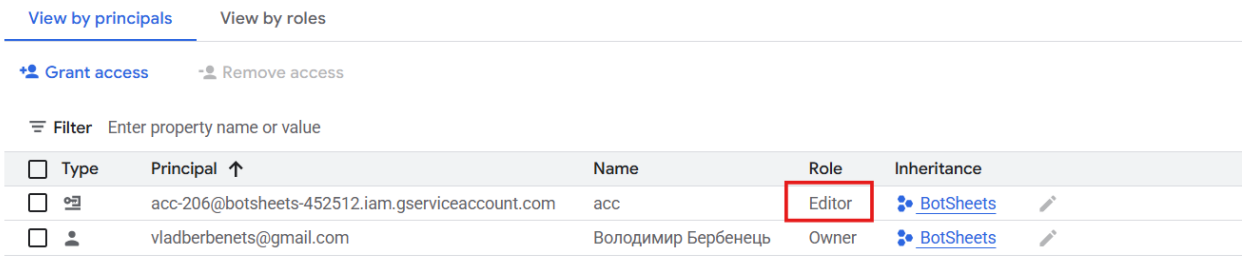



Рис. 3.11 – Налаштування доступу до таблиці

6. Після створення сервісного акаунта Google запропонує завантажити JSON-файл який буде в собі містити ключі та ідентифікатори, його ми будемо використовувати у нашому коді для авторизації (рис. 3.12).

Рис. 3.12 – Вміст завантаженого файлу JSON

Важливий момент: щоб бот міг працювати з конкретною таблицею, потрібно вручну надати доступ на редагування цьому сервісному акаунту. Для цього просто відкриваємо потрібну таблицю в Google Sheets, натискаємо "Надати доступ" і вписуємо пошту сервісного акаунту, яка буде вказана в отриманому JSON-файлі (рис. 3.13).

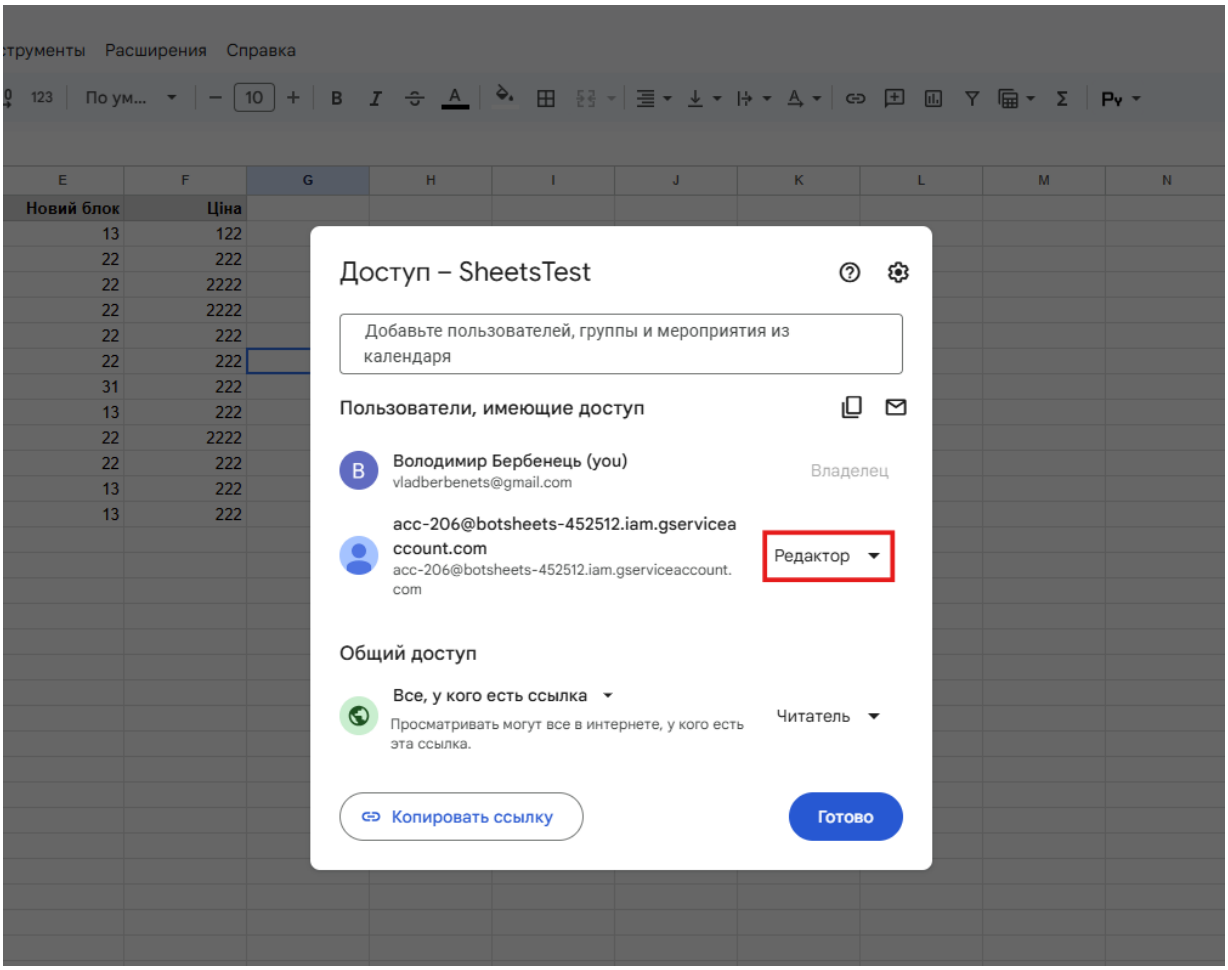



Рис. 3.13 – Надання прав доступу в Google Sheets

Таким чином, налаштувавши обидва API – Telegram та Google Sheets – ми можемо перейти безпосередньо до розробки самого бота та інтегрування вже існуючих ресурсів.

## 3.4 Інтеграція створених API ключів

Першим ділом я підключив токен бота – спеціальний унікальний ключ, який раніше створив через BotFather. За допомогою цього токену код буде керувати ботом, отримувати повідомлення та відправляти відповіді (рис. 3.14).

```
6
7   bot = telebot.TeleBot("8136592296:AAElw-KFSwaKuPqaA_Qgc3Ol6GjQX46ugoU")
8
```

Рис. 3.14 – Підключення API ключів Telegram

Для того, щоб бот міг працювати з таблицями Google Sheets, спочатку

потрібно налаштувати доступ через API. Я вже виконав усі дії з налаштування та завантажив файл із потрібними ключами.

Далі у коді я вказав шлях до цього файлу. Для авторизації використав певний перелік дозволів (SCOPE), які дають боту право не тільки читати, а й змінювати вміст таблиць (рис. 3.15).

```
8
9   SCOPE = ["https://spreadsheets.google.com/feeds", "https://www.googleapis.com/auth/drive"]
10  CREDS = ServiceAccountCredentials.from_json_keyfile_name("C:/Users/vladb/Downloads/botsheets-452512-e6bb262a008e.json", SCOPE)
11  CLIENT = gspread.authorize(CREDS)
12  SHEET = CLIENT.open("SheetsTest").sheet1
13
```

Рис. 3.15 – Підключення API ключів Telegram

Тепер бот має змогу автоматично зчитувати дані з таблиці, додавати нові записи або змінювати наявні, що дозволяє ефективно автоматизувати обробку інформації.

## 3.5 Логіка роботи бота та реалізація основних функцій

Після підключення всіх необхідних API-ключів я перейшов до налаштування автентифікації користувачів у боті. Для цього спочатку створив окремий текстовий файл password.txt, в який записав пароль. Далі в коді я відкрив цей файл, а за допомогою конструкції with open зчитується його вміст у змінну correct-password (рис. 3.16).

```
13
14  with open("C:/Users/vladb/OneDrive/Рабочий стол/диплом/password.txt", "r") as file:
15      CORRECT_PASSWORD = file.read().strip()
16
```

Рис. 3.16 – Підключення файлу з паролем

Це дозволяє безпечно зберігати пароль окремо від основного коду бота та спрощує його оновлення в майбутньому.

Для того, щоб бот знав, хто з користувачів уже пройшов авторизацію та в подальшому давав доступ до боту без заповнення паролю, я створив словник user-authenticated. У цьому словнику зберігаються id користувачів, які ввели правильний пароль (рис. 3.17).

```
16
17  user_authenticated = {}
18
```

Рис. 3.17 – Словник для зберігання статусу автентифікованих користувачів

Щоб підтримувати структуру таблиці у правильному вигляді, я створив спеціальну функцію ensure-columns-order. Вона слідкує, щоб перший стовпець завжди був "Оновлення", а другий – "id". Якщо таких стовпців немає, або вони в неправильному порядку, функція автоматично їх додає чи переміщує (рис. 3.18).

```
19
20  def ensure_columns_order():
21      headers = SHEET.row_values(1)
22
23      if not headers:
24          SHEET.append_row(["Оновлення", "id"])
25          return
26
27      if "Оновлення" not in headers:
28          SHEET.insert_cols([["Оновлення"]], 1)
29          headers = ["Оновлення"] + headers
30
31      if headers[0] != "Оновлення":
32          update_col = headers.index("Оновлення") + 1
33          SHEET.insert_cols([["Оновлення"]], 1)
34          SHEET.delete_columns(update_col + 1)
35          headers = SHEET.row_values(1)
36
37      if "id" in headers and headers.index("id") != 1:
38          id_col = headers.index("id") + 1
39          id_values = SHEET.col_values(id_col)
40          SHEET.insert_cols([[id] for id in id_values], 2)
41          SHEET.delete_columns(id_col + 1)
42
```

Рис. 3.18 – Функція корегування порядку стовпців

Оскільки під час роботи користувачі будуть надсилати дані з новими полями, була створена функція add-new-columns. Вона додає нові стовпці у таблицю, якщо їх ще немає (рис. 3.19).

```python
def add_new_columns(data_dict):
    headers = SHEET.row_values(1)
    headers_lower = [h.lower() for h in headers]
    new_columns = []

    for key in data_dict.keys():
        if key.lower() not in headers_lower and key.lower() not in ["оновлення", "id"]:
            new_columns.append(key)

    if new_columns:
        last_col = len(headers)
        SHEET.insert_cols([[col] for col in new_columns], last_col + 1)
```

Рис. 3.19 – Функція додавання нових стовпців

Для збереження часу оновлень в таблиці я створив окрему допоміжну функцію get-current-datetime. Вона повертає дату та час у форматі "день.місяць.рік година:хвилина" (рис. 3.20).

```python
def get_current_datetime():
    return datetime.now().strftime("%d.%m.%Y %H:%M")
```

Рис. 3.20 – Функція для отримання поточного часу

Щоб знаходити потрібні записи в таблиці за id, я створив функцію get-data-by-id. Вона читає всі записи і повертає той, у якому міститься потрібний "id" (рис. 3.21).

```python
def get_data_by_id(id):
    try:
        id = int(id)
    except ValueError:
        return None

    ensure_columns_order()

    records = SHEET.get_all_records()
    for record in records:
        if str(record.get("id", "")).strip() == str(id):
            return record
    return None
```

Рис. 3.21 – Пошук запису в таблиці за id

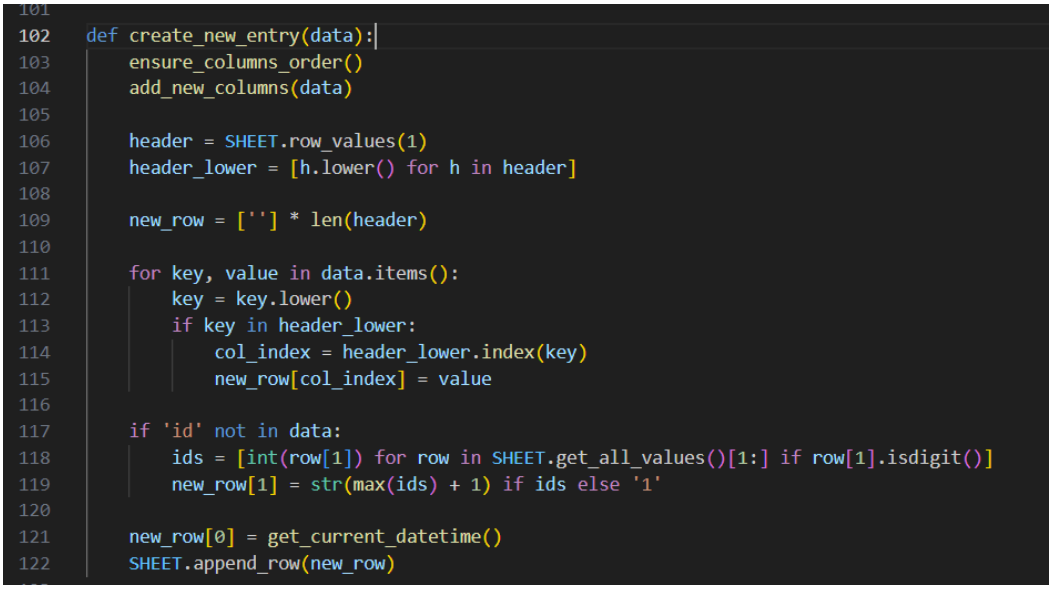
Функція update-data оновлює існуючий запис у таблиці. Вона знаходить

рядок за id і замінює потрібні значення, а також автоматично проставляє нову дату і час оновлення (рис. 3.22).

```
def update_data(id, data):
    try:
        ensure_columns_order()
        add_new_columns(data)

        cell = SHEET.find(str(id))
        if not cell:
            return False

        header = SHEET.row_values(1)
        header_lower = [h.lower() for h in header]

        for key, value in data.items():
            key = key.strip().lower()
            if key in header_lower:
                col_index = header_lower.index(key) + 1
                SHEET.update_cell(cell.row, col_index, value)

        SHEET.update_cell(cell.row, 1, get_current_datetime())
        return True
    except Exception as e:
        print(f'Помилка оновлення даних: {e}')
        return False
```

Рис. 3.22 – Оновлення запису за id

Якщо в таблиці ще немає запису з потрібним id, бот може створити новий запис. Для цього я реалізував функцію create-new-entry (рис. 3.23).



```
def create_new_entry(data):
    ensure_columns_order()
    add_new_columns(data)

    header = SHEET.row_values(1)
    header_lower = [h.lower() for h in header]

    new_row = [''] * len(header)

    for key, value in data.items():
        key = key.lower()
        if key in header_lower:
            col_index = header_lower.index(key)
            new_row[col_index] = value

    if 'id' not in data:
        ids = [int(row[1]) for row in SHEET.get_all_values()[1:] if row[1].isdigit()]
        new_row[1] = str(max(ids) + 1) if ids else '1'

    new_row[0] = get_current_datetime()
    SHEET.append_row(new_row)
```

Рис. 3.23 – Створення нового запису в таблиці

Щоб користувачі могли очищати таблицю від непотрібних полів, я реалізував функцію delete-column-by-name яка видаляє стовпець за його назвою, за винятком базових стовпців "Оновлення" та "id" а також функцію delete-row-by-id для видалення цілий рядків із таблиці за id. (рис. 3.24).

```
125  def delete_column_by_name(column_name):
126      if column_name.lower() in ["оновлення", "id"]:
127          return False
128
129      ensure_columns_order()
130
131      header = SHEET.row_values(1)
132      column_name_lower = column_name.strip().lower()
133      matched_column = next((col for col in header if col.strip().lower() == column_name_lower), None)
134
135      if matched_column:
136          col_index = header.index(matched_column) + 1
137          SHEET.delete_columns(col_index)
138          return True
139      return False
140
141  def delete_row_by_id(id):
142      try:
143          ensure_columns_order()
144          cell = SHEET.find(str(id))
145          if cell:
146              SHEET.delete_rows(cell.row)
147              return True
148          return False
149      except Exception as e:
150          print(f'Помилка видалення рядка: {e}')
151          return False
```

Рис. 3.24 – Видалення стовпця з таблиці

Коли користувач надсилає команду /start, бот запитує пароль для авторизації (рис. 3.25).

```
154
155  @bot.message_handler(commands=['start'])
156  def send_welcome(message):
157      bot.send_message(message.chat.id, "Введіть пароль для авторизації.")
```

Рис. 3.25 – Видалення рядка за id

Якщо користувач ввів правильний пароль, бот додає його id у словник user-authenticated і показує йому інтерактивну кнопку для відкриття таблиці (рис. 3.26).

```python
@bot.message_handler(func=lambda message: message.text and not user_authenticated.get(message.chat.id, False))
def handle_password(message):
    if message.text.strip() == CORRECT_PASSWORD:
        user_authenticated[message.chat.id] = True
        markup = InlineKeyboardMarkup()
        markup.add(InlineKeyboardButton("Відкрити таблицю",
                   url="https://docs.google.com/spreadsheets/d/1BYnMGTOvJrA7o80sE9OL-MR20UbwOri2vOKXpbiz4Q8/edit?usp=sharing"))
        bot.send_message(message.chat.id,
                         "Авторизація успішна. Тепер ви можете використовувати бота.\nНапишіть id клієнта.",
                         reply_markup=markup)
    else:
        bot.send_message(message.chat.id, "Неправильний пароль. Спробуйте ще раз.")
```

Рис. 3.26 – Запит пароля при старті та перевірка введеного пароля

Після авторизації користувач може надіслати повідомлення, що містить тільки id. Бот тоді шукає відповідний запис у таблиці (рис. 3.27).

```python
    record = get_data_by_id(message.text)
    if record:
        response = '\n'.join([f'{key.capitalize()}: {value}'
                              for key, value in record.items()
                              if key.lower() != "оновлення"])

        bot.send_message(message.chat.id, response)
```

Рис. 3.27 – Відображення даних за id

Після виведення інформації бот також пропонує користувачу можливі наступні кроки (рис. 3.28).

```python

        update_prompt = "Ви можете:\n"
        update_prompt += "1. Змінити інформацію (відправте у форматі 'Id: X\nПоле: X')\n"
        update_prompt += "2. Видалити рядок (відправте 'Id: X\nвидалити')\n"
        update_prompt += "3. Видалити стовпець (відправте 'Id: X\nНазва стовпця: видалити')"

        bot.send_message(message.chat.id, update_prompt)
    else:
        bot.reply_to(message, "ID не знайдено. Ви можете створити новий запис, відправивши дані у форматі:\nId: X\nІм'я: X\n...")
```

Рис. 3.28 – Пропозиція варіантів редагування запису

Визначення типу запиту: оновлення чи видалення рядка
Перед початком обробки даних бот аналізує надіслане повідомлення користувача. Якщо другий рядок містить слово "видалити" українською або російською "видалити ", бот розпізнає це як запит на видалення рядка (рис. 3.29).

```
196     try:
197         data_lines = [line.strip() for line in message.text.split('\n') if line.strip()]
198 
199         # Перевіряємо чи є команда видалення рядка
200         if len(data_lines) >= 2 and data_lines[1].lower() in ["видалити"]:
201             id = data_lines[0].split(': ')[1].strip()
202             if delete_row_by_id(id):
203                 bot.reply_to(message, f"Рядок з ID {id} успішно видалено.")
204             else:
205                 bot.reply_to(message, f"Не вдалося знайти рядок з ID {id}.")
206             return
```

Рис. 3.29 – Визначення запиту на видалення рядка

Обробка запиту на видалення окремих стовпців Якщо користувач хоче видалити окремі стовпці, а не весь запис, бот аналізує усі рядки повідомлення після id. Стовпці, які слід видалити, збираються у список columns-to-delete (рис. 3.30).

```
210         data = {}
211         columns_to_delete = []
212 
213         for line in data_lines[1:]:
214             if ': ' in line:
215                 key, value = line.split(': ', 1)
216                 if value.strip().lower() in ["видалити"]:
217                     columns_to_delete.append(key.strip())
218                 else:
219                     data[key.lower()] = value.strip()
```

Рис. 3.30 – Видалення окремих стовпців

Якщо жодна команда видалення не була розпізнана, бот розглядає запит як оновлення існуючого запису або створення нового (рис. 3.31).

```
237 
238         if get_data_by_id(id):
239             if update_data(id, data):
240                 bot.reply_to(message, f"Інформація для ID {id} оновлена.")
241             else:
242                 bot.reply_to(message, "Помилка оновлення даних.")
243         else:
244             data["id"] = id
245             create_new_entry(data)
246             bot.reply_to(message, f"Новий запис з ID {id} додано.")
247     except Exception as e:
248         bot.reply_to(message, f"Помилка обробки даних: {e}")
```

Рис. 3.31 – Оновлення існуючого або створення нового запису

Наприкінці програми я викликав функцію ensure-columns-order і запустив нескінченне опитування повідомлень (рис. 3.32).

```
248
249 if __name__ == '__main__':
250     ensure_columns_order()
251     bot.polling(none_stop=True, interval=0)
```

Рис. 3.32 – Запуск опитування бота

Це забезпечує безперервну роботу бота без необхідності перезапуску вручну.

Висновок: у цьому підрозділі було детально розглянуто всі елементи обробки даних нашого бота та його контакт з користувачем. Розроблений код вирізняється високим рівнем оптимізації та логічною структурою.

## 3.6 Розгортання Telegram-бота на віддаленому сервері

Для забезпечення безперервної роботи нашого бота необхідно розмістити його на сервері. Аналізуючи різні хостинг-провайдери у другому розділі, я визначив найбільш оптимальний варіант онлайн-хостингу – Ukraine.com.ua, який пропонує великий обсяг ресурсів за досить доступною ціною.

Для керування ботом на сервері я використовую SSH – це мережевий протокол, який дозволяє безпечно підключатись до віддаленої машини і керувати нею через командний рядок. Окрім того, для зручності переносу файлів на сервер я використовую програму FileZilla. Вона дає змогу швидко з'єднуватися з сервером через SFTP і легко копіювати файли або цілі директорії без зайвих складнощів.

Процес налаштування виглядає наступним чином: спершу потрібно зайти в адмін-панель хостингу і перейти до розділу SSH-доступ (рис. 3.33). Там відображаються всі необхідні дані для з'єднання: хост, порт, логін і пароль.

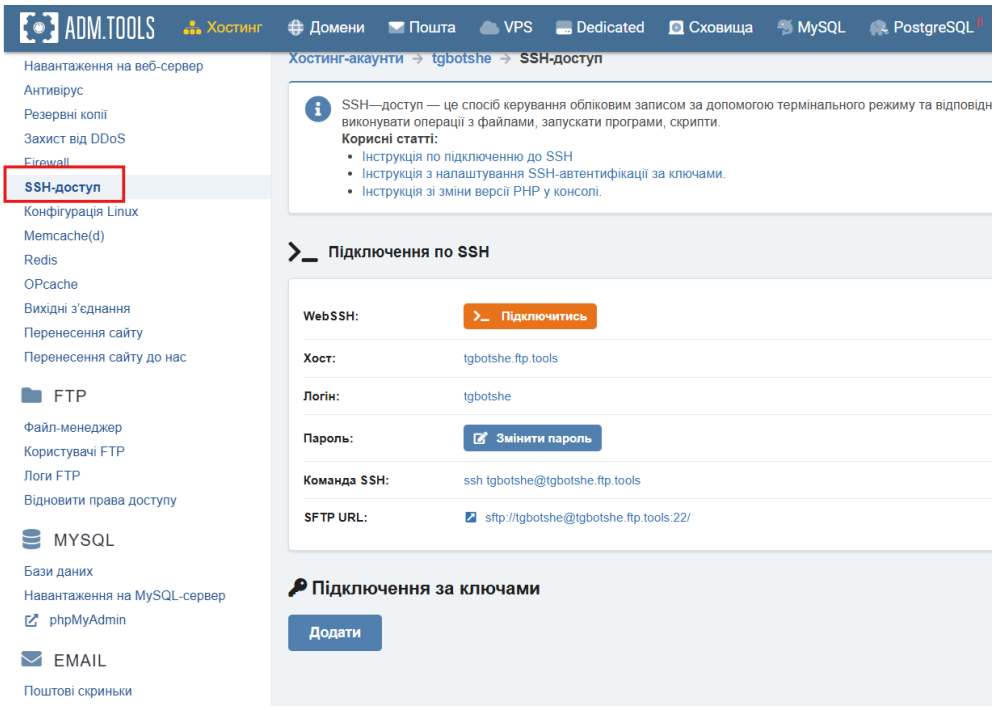



Рис. 3.33 – Дані для підключення до серверу

Далі відкриваємо FileZilla і вказуємо параметри підключення (рис. 3.34).

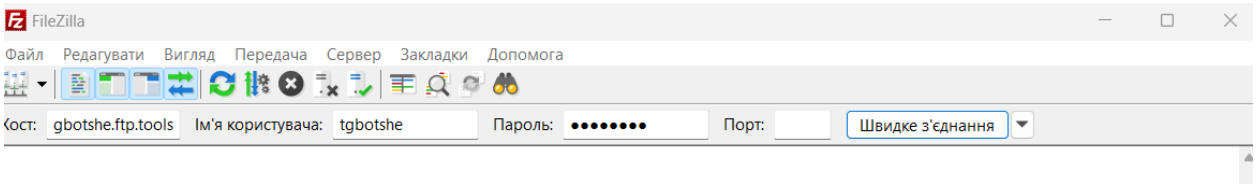



Рис. 3.34 – Запис даних для підключення до серверу

Після підключення створюємо окрему директорію для нашого проєкту, куди завантажуємо усі необхідні файли бота: файл з основним кодом бота (googlesheets1.py), файл password.txt із паролем для авторизації користувачів, JSON-файл з API-ключами для роботи з Google Sheets (рис. 3.35).

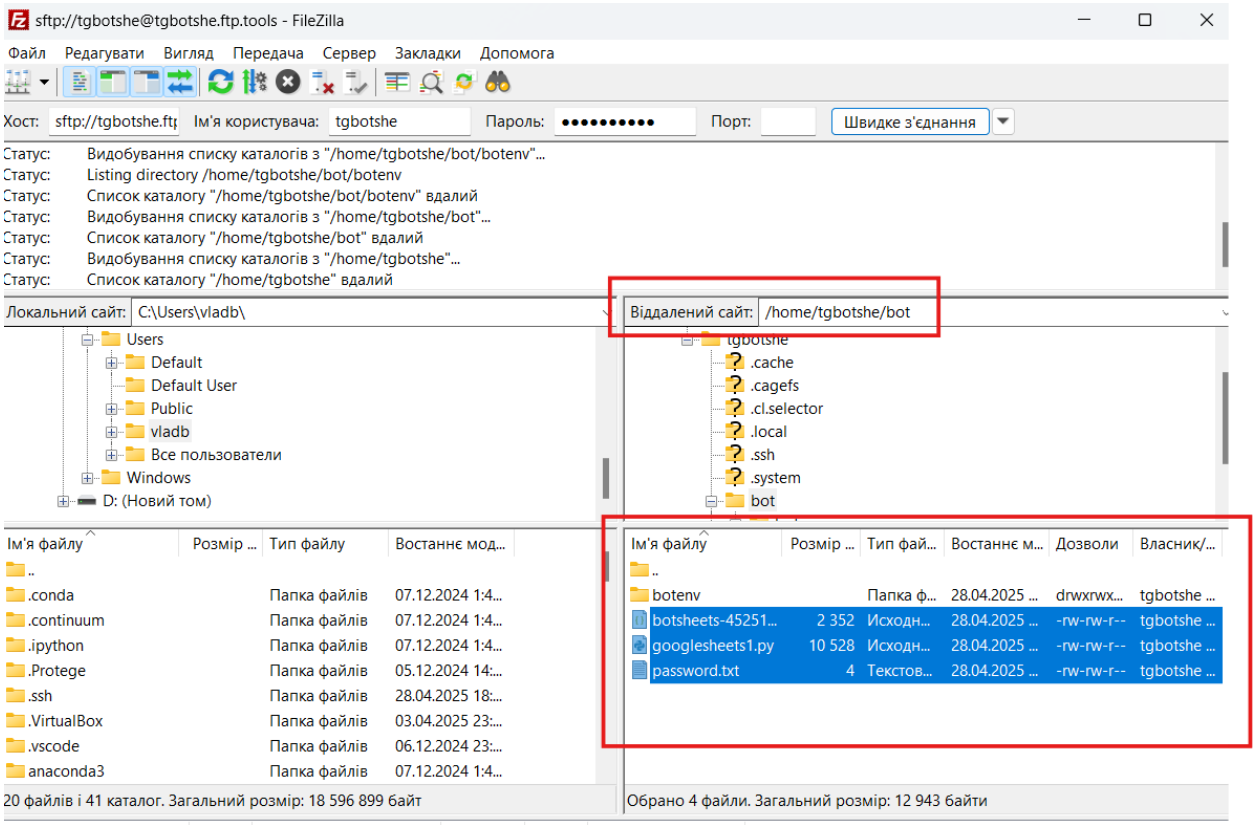



Рис. 3.35 – Створення директорії та завантаження файлів

Після завантаження всіх необхідних файлів на сервер потрібно відредагувати шляхи до ключових файлів у коді бота. Необхідно змінити шляхи до файлів password.txt та json файлу з API-ключами для роботи з Google Sheets, щоб вони відповідали розташуванню файлів на сервері (рис. 3.36).

```
8
9   SCOPE = ["https://spreadsheets.google.com/feeds", "https://www.googleapis.com/auth/drive"]
10  CREDS = ServiceAccountCredentials.from_json_keyfile_name("/home/tgbotshe/bot/botsheets-452512-e6bb262a008e.json", SCOPE)
11  CLIENT = gspread.authorize(CREDS)
12  SHEET = CLIENT.open("SheetsTest").sheet1
13
14  with open("/home/tgbotshe/bot/password.txt", "r") as file:
15      CORRECT_PASSWORD = file.read().strip()
```

Рис. 3.36 – Редагування шляхів до ключових файлів

Також не буде зайвим створити файл requirements.txt, в якому будуть перелічені всі бібліотеки, необхідні для роботи бота (рис. 3.37).

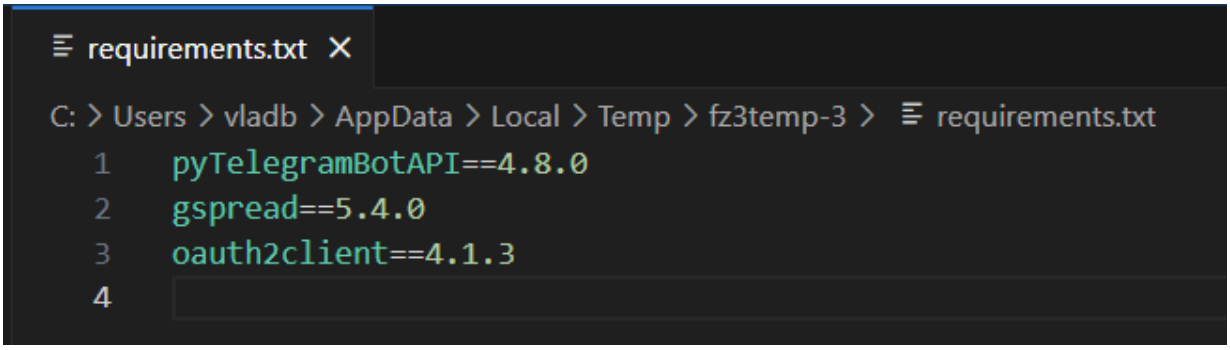



Рис. 3.37 – Запис потрібних бібліотек для швидкого встановлення

Це спростить процес встановлення залежностей за допомогою команди яку ми впишемо в консоль сервера (рис. 3.38).

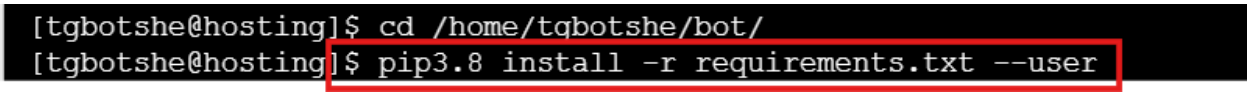



Рис. 3.38 – Команда для завантаження всіх бібліотек в файлі requirements.txt

Після виконання цих етапів необхідно запустити бота та переконатися в його коректній роботі. Це робиться за допомогою команди: python3.8 googlesheets1.py

### 3.7 Аналіз стабільності та працездатності бота

Після довгого пошуку та підбору потрібних компонентів для проєкту – таких як мова програмування, середовище розробки, вибір хостингу, інтеграція API-ключів та визначення функціональних можливостей нарешті настав етап тестування його роботи.

Тестування включило в себе: авторизацію користувача, повноцінну реалізацію CRUD-операцій (створення, читання, оновлення, видалення), а також перевірка системи у разі виникнення помилок або незрозумілих дій з боку користувача.

Перевірка авторизації. Перевірка починалась з авторизації. Після запуску бота користувач має ввести пароль – тільки після цього надсилається inline-кнопка з посиланням на Google Таблицю та запитом на введення id (Рис. 3.39).

1. Введення правильного пароля відкриває доступ до функцій бота

2. Бот надсилає коректне inline-посилання на Google Таблицю
3. При неправильному паролі доступ блокується

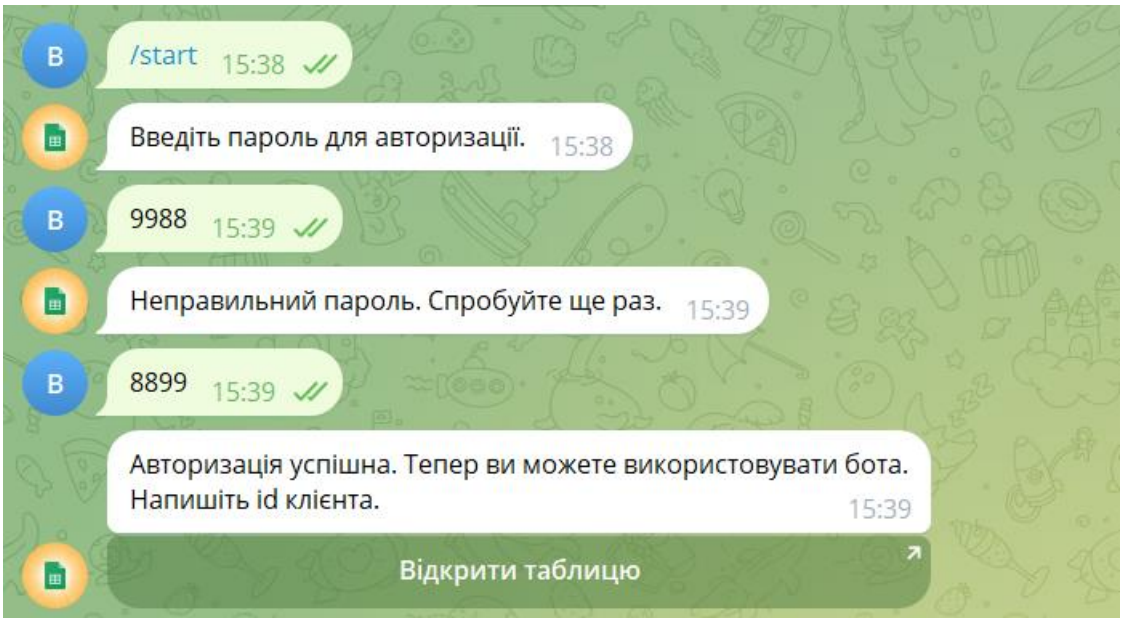
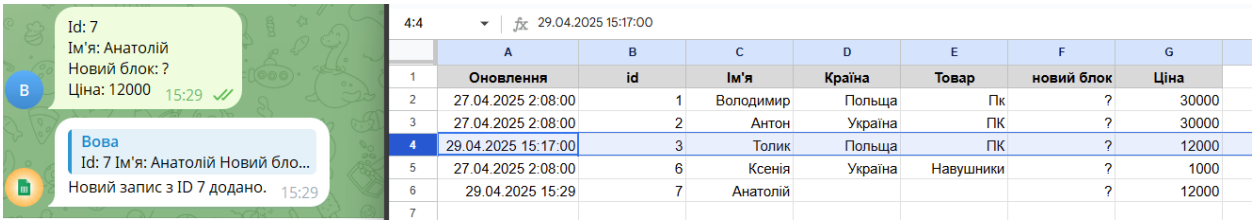



Рис. 3.39 – Перевірка коректності авторизації

CRUD-операції з даними. Create – створення запису. Функціонал створення передбачає можливість додати новий рядок за введеним id. Якщо деякі поля не заповнені, вони залишаються порожніми, що дозволяє створювати неповні записи без помилок (Рис. 3.40).




| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | Оновлення | id | Ім'я | Країна | Товар | новий блок | Ціна |
| 2 | 27.04.2025 2:08:00 | 1 | Володимир | Польща | Пк | ? | 30000 |
| 3 | 27.04.2025 2:08:00 | 2 | Антон | Україна | ПК | ? | 30000 |
| 4 | 29.04.2025 15:17:00 | 3 | Толик | Польща | ПК | ? | 12000 |
| 5 | 27.04.2025 2:08:00 | 6 | Ксенія | Україна | Навушники | ? | 1000 |
| 6 | 29.04.2025 15:29 | 7 | Анатолій | | | ? | 12000 |
| 7 | | | | | | | |

Рис. 3.40 – Створення нового запису з частково заповненими даними

Read – читання даних. Бот виводить повну інформацію за правильним id. Якщо ж ввести некоректний або порожній id, користувач отримує відповідне повідомлення про помилку (Рис. 3.41).

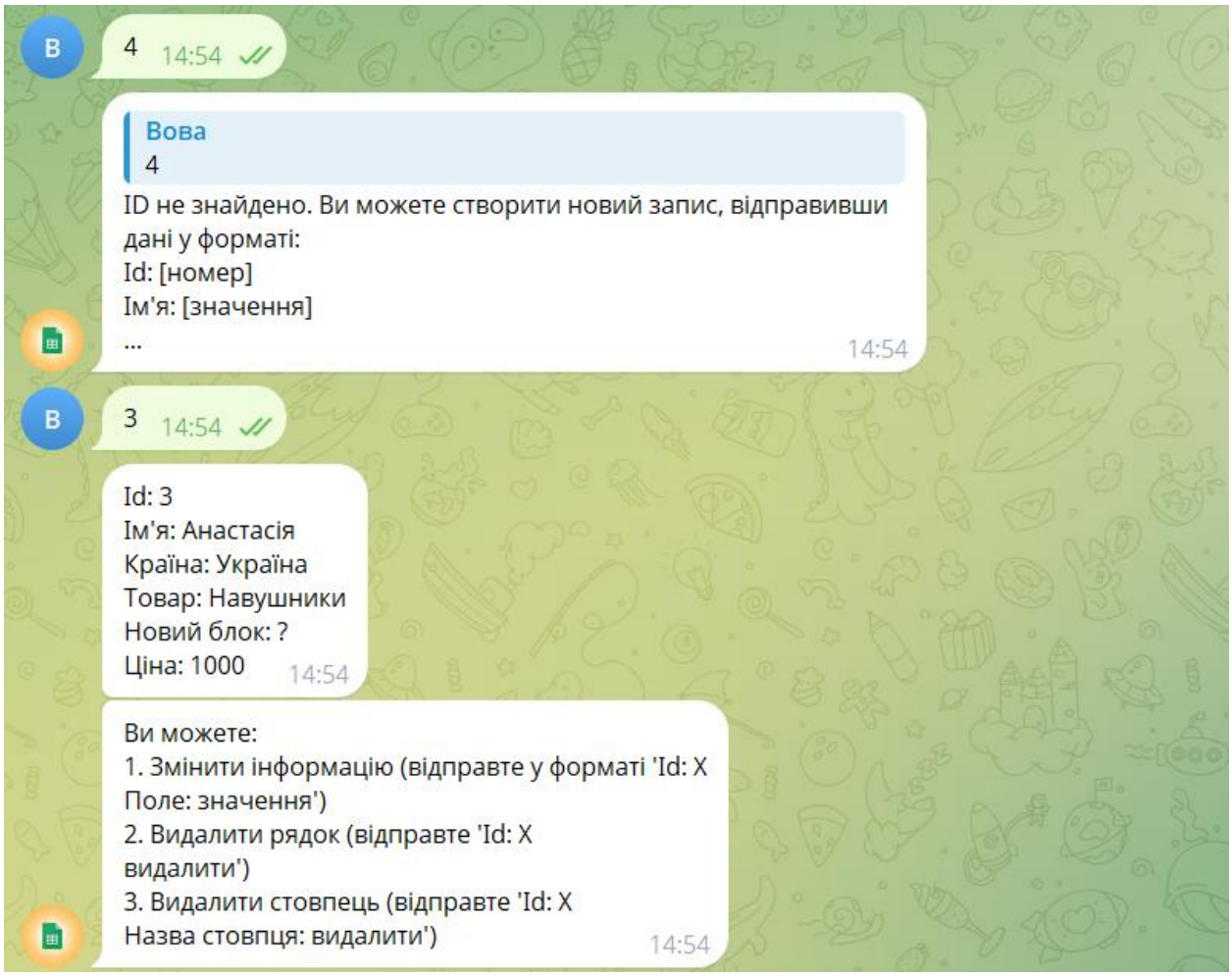



Рис. 3.41 – Виведення та обробка правильного та неправильного id

Update – оновлення даних. Оновлення реалізовано гнучко: можна змінювати як усі дані, так і лише окремі колонки. Якщо деякі поля залишити порожніми, бот зберігає попередні значення, що дозволяє уникнути випадкового стирання важливої інформації (Рис. 3.42).

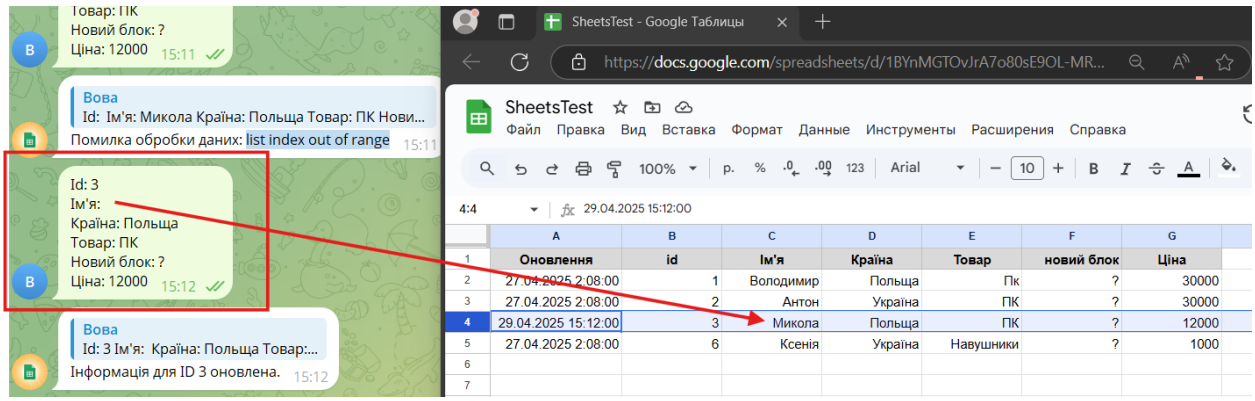



Рис. 3.42 – Збереження попереднього значення при порожньому введенні

При кожному оновленні фіксується час останньої зміни (Рис. 3.43).

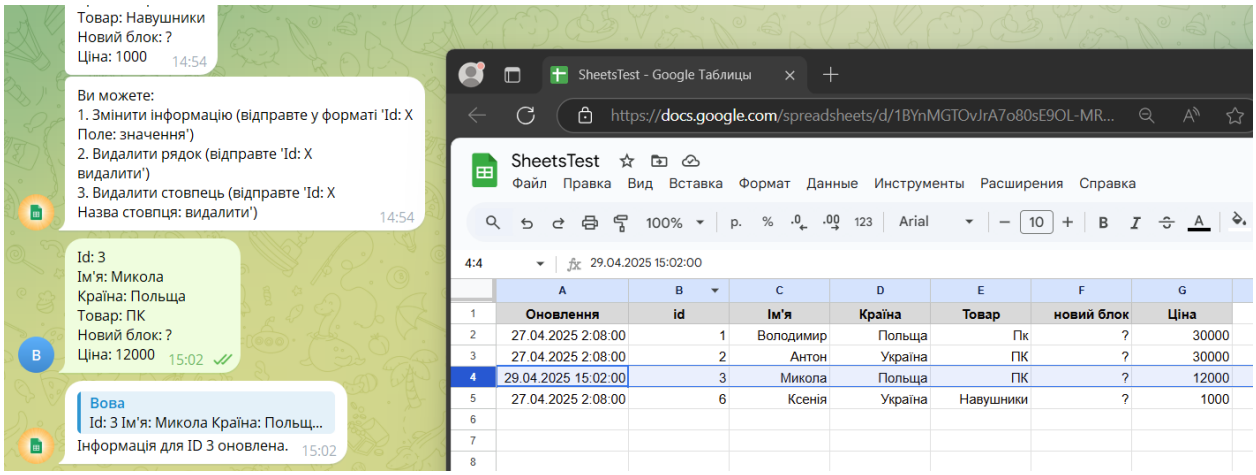



Рис. 3.43 – Оновлення даних у таблиці та фіксація часу

Оновлення можливе навіть для частини полів (Рис. 3.44).

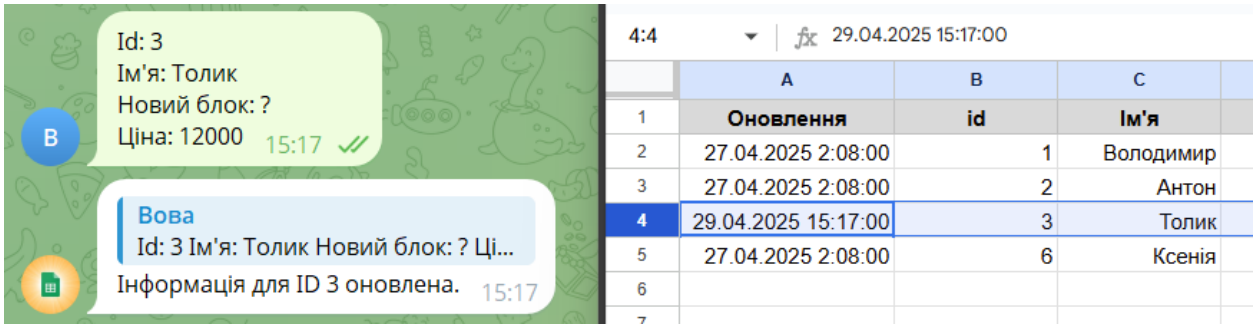



Рис. 3.44 – Часткове оновлення даних у записі

Якщо id не вказано, бот повідомляє про помилку та не виконує змін (Рис. 3.45).

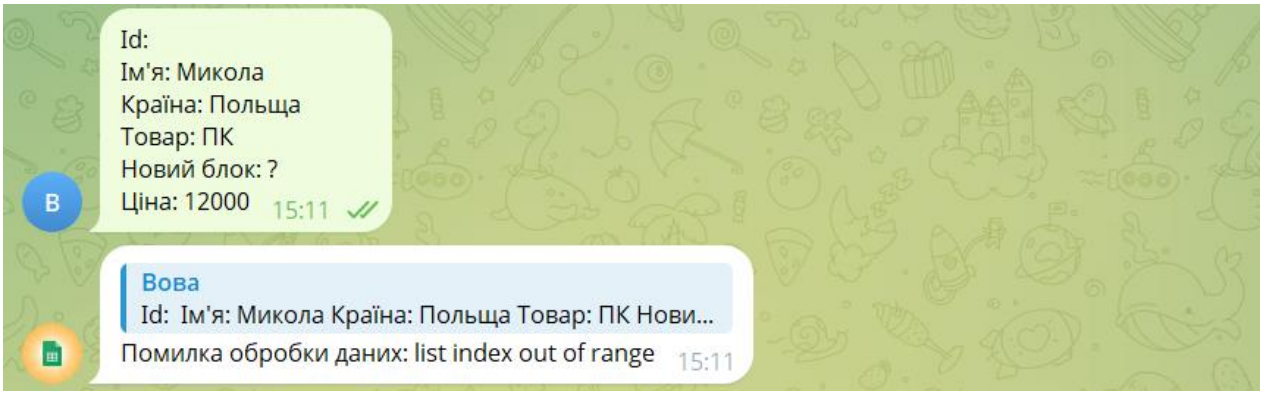



Рис. 3.45 – Повідомлення про помилку при спробі оновлення порожнього id

Delete – видалення даних. Видалення можливо як для окремих рядків, так і для стовпців:

При видаленні рядка поле "оновлення" не змінюється, щоб уникнути плутанини (Рис. 3.46).

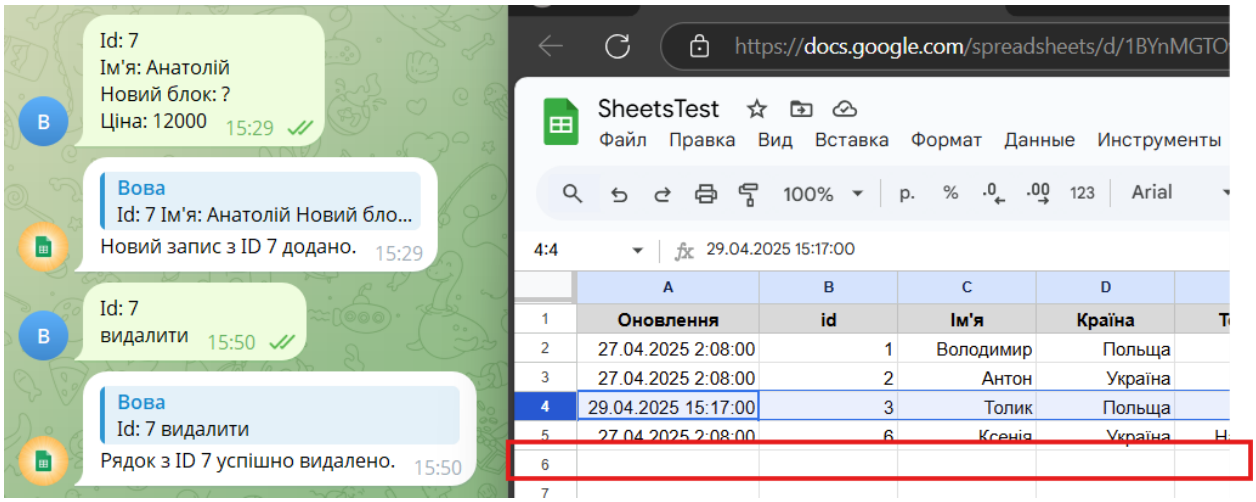



Рис. 3.46 – Видалення рядка без зміни часу оновлення

Видалення стовпців можливе лише після підтвердження id. Після видалення структура таблиці зберігається: дані зсуваються вліво (Рис. 3.47).

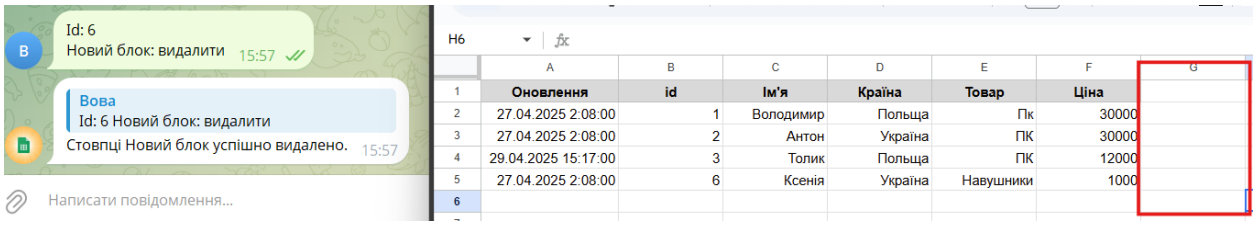



Рис. 3.47 – Видалення стовпця та зсування даних

Базові колонки, як id та оновлення, є захищеними – бот блокує спроби їх видалення та повідомляє про це користувача (Рис. 3.48).

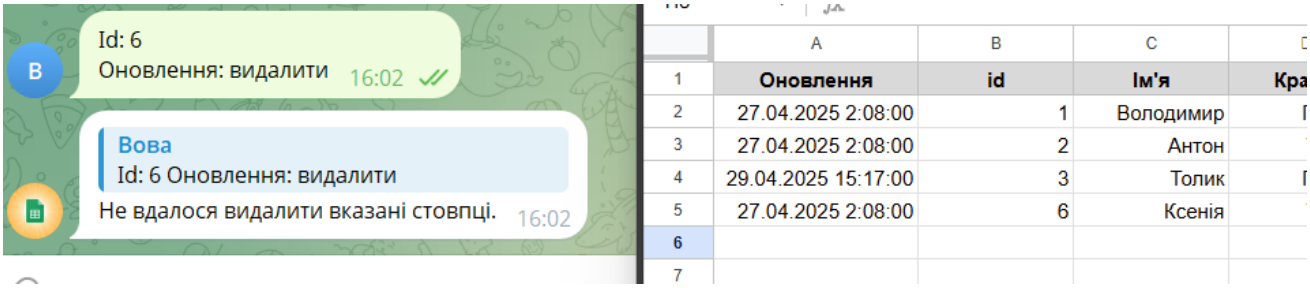



Рис. 3.48 – Виведення помилки при спробі видалення стовпця "оновлення"

Обробка помилок. Бот демонструє стійкість до помилок та некоректних

сценаріїв введення:

1. Повідомляє про порожній або неправильний id
2. Інформує про недопустимі дії, наприклад, спробу видалити службовий стовпець
3. Стабільно працює навіть при нетипових сценаріях користувача

Перевіривши роботу бота, можна зробити висновок, що він працює швидко, стабільно та без помилок. Система демонструє хороший потенціал для обробки даних і взаємодії з таблицями в реальному часі. Такий бот може ефективно задовольнити потреби невеликого бізнесу і стати частиною його автоматизації. При цьому на його розробку витрачено мінімум ресурсів, що робить рішення особливо привабливим. У підсумку ми отримали зручного та функціонального помічника для роботи з базами даних.

# ВИСНОВКИ

В ході розробки цієї кваліфікаційної роботи мені вдалося створити функціонального Telegram-бота, здатного взаємодіяти з Google-таблицями в режимі реального часу. Цей бот забезпечує доступ користувача до таблиць без необхідності відкривати браузер. Основні функції програми включають в себе авторизацію по паролю, пошук записів за id, оновлення даних, додавання нових рядків, видалення записів і редагування таблиці з автоматичним збереженням її виду за замовченням.

Однією з ключових переваг розробленого сервісу стало його розміщення на віддаленому сервері, що забезпечило цілодобовий доступ до бота та його стабільну роботу. Також було проведено детальний аналіз різноманітних хостинг-сервісів, інструментів для роботи з таблицями, а також середовищ розробки.

Також не мало уваги було приділено захисту даних. Для авторизації було реалізовано перевірку пароля, який зберігається у текстовому файлі – це дозволяє зручно змінювати його без втручання в код програми. Також реалізовано систему логування змін в таблиці. Telegram-бот має інтуїтивний інтерфейс, а також є стабільним, швидким та зручним у користуванні. Такий інструмент буде корисним не тільки для особистого користування, а й для користування власниками малого чи середнього бізнесу, які прагнуть в майбутньому автоматизувати свої бізнес-процеси та полегшити роботу персоналу при мінімальних витратах.

У результаті, створений бот повністю відповідає поставленим завданням кваліфікаційної роботи. Він гнучкий, масштабований і практичний, а також має великі перспективи для подальшого розвитку, інтеграції нових функцій та адаптації під потреби користувача чи бізнесу.

# ПЕРЕЛІК ПОСИЛАНЬ


1. [1] Forbes. Україна. URL: https://cutt.ly/Drj1SeQU
2. [2] Marketer.ua. Соціальні мережі 2024: Рейтинг найпопулярніших платформ в Україні та світі. URL: https://cutt.ly/Qrk0isyQ
3. [3] Stackby.com. Порівняння Google Sheets та Microsoft Excel URL: https://cutt.ly/KrcID1cH
4. Prometheus. Telegram-боти України: як українці використовують месенджер для бізнесу та освіти. URL: https://prometheus.org.ua/blog/ukrainiantelegrambots/
5. Сервіси для роботи з таблицями. URL: https://cutt.ly/Zrj1ul61
6. Що відрізняє мови програмування: детальний огляд. URL: https://cutt.ly/drj1qUrr
7. Python. Офіційна документація. Підручник Python 3.13. URL: https://docs.python.org/uk/3.13/tutorial/index.html
8. Visual Studio Code. Часті запитання. URL: https://code.visualstudio.com/docs/supporting/faq
9. Порівняння хостингів для вебсайтів. URL: https://www.websiteplanet.com/uk/web-hosting/
10. Використаний хостинг для розгортання Telegram-бота. URL: https://cutt.ly/arj1Pc71
11. Telegram. BotFather – інструмент для створення ботів. URL: https://telegram.org/blog/bot-revolution
12. Telegram Bot API. Офіційна документація. URL: https://core.telegram.org/bots/api
13. Telegram API. Технічна документація. URL: https://core.telegram.org/api
14. Google Sheets API. Документація. URL: https://developers.google.com/sheets/api
15. Основи розробки Telegram-ботів. URL: https://cutt.ly/NrjM6Gcq
16. Telebot – бібліотека для створення Telegram-ботів мовою Python. URL: https://pytba.readthedocs.io/en/latest/index.html

17. gspread – бібліотека для роботи з Google Sheets API. URL: https://docs.gspread.org/en/v6.1.3/

18. datetime – модуль Python для роботи з датою та часом. URL: https://docs.python.org/uk/3.13/library/datetime.html

19. Requests у Python – документація. URL: https://docs.python-requests.org/en/latest/

20. JSON – офіційна специфікація формату. URL: https://www.json.org/json-uk.html

21. Python Requests – бібліотека для HTTP-запитів. URL: https://requests.readthedocs.io/en/latest/

22. Запуск Telegram-бота на хостингу. URL: https://ukrline.com.ua/wiki/ua/telegram-bot.php

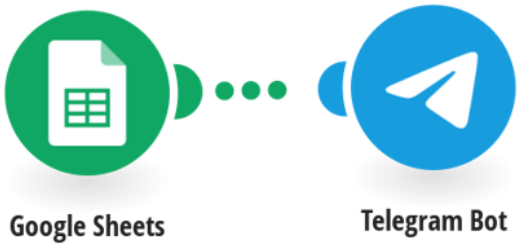
**ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)**

# Розробка Telegram-бота для роботи з таблицями у реальному часі

**Виконав: Бербенець Володимир, ІСД-41**
**Керівник: Олег СЕНЬКОВ**

## Мета та завдання

Створити Telegram-бота для взаємодії з таблицями в реальному часі.




1. Проаналізувати рішення
2. Обрати технології
3. Реалізувати функції авторизації, оновлення, видалення
4. Підключити Google Sheets API
5. Протестувати стабільність

# Проблематика та актуальність сервісу

Чим же може бути корисний наш сервіс в сучасних умовах?

### Легкість

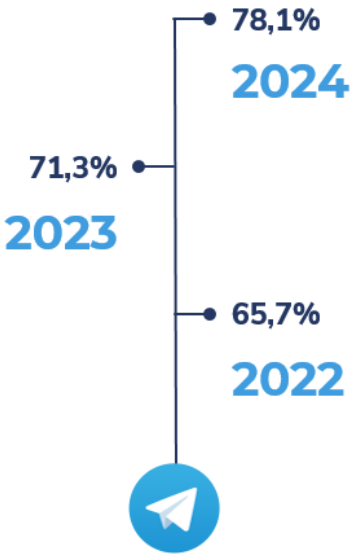
Бот може полегшити процес інтеграції колективу в робочий процес

### Логування

Може надавати детальну інформацію для керівників про оновлення в базі даних

# Чому саме Telegram?

Telegram є одним із найпопулярніших засобів комунікації в Україні. За даними сайту vctr.media на 2024 рік цим месенджером користуються понад 78% українців



1

## Мова

**Python — проста і швидка**

2

## Середовище

**Visual Studio Code**

3

## Табличний сервіс

**Google Sheets**

4

## Хостинг

**Ukraine.com.ua — стабільний і доступний.**

# Чому саме Google Sheets?

### Масове використання

**2 мільярди користувачів на місяць.**

### Швидкий доступ до API

**Google Sheets легко підключається до Telegram-бота**

### Вартість

**На відміну від Airtable, Smartsheet чи OnlyOffice, більшість функцій доступні безкоштовно**

# Як працює бот?

### Авторизація

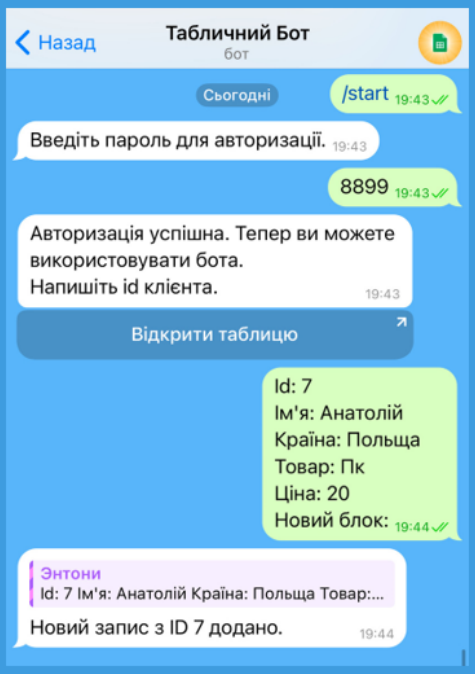
Після запуску бота користувач вводить пароль. Якщо він правильний, бот надає доступ до функцій. Стан авторизації зберігається для кожного користувача.

### CRUD операції

- Create: додавання нових записів до таблиці
- Read: отримання інформації за ID
- Update: зміна існуючих записів
- Delete: видалення рядків або стовпців

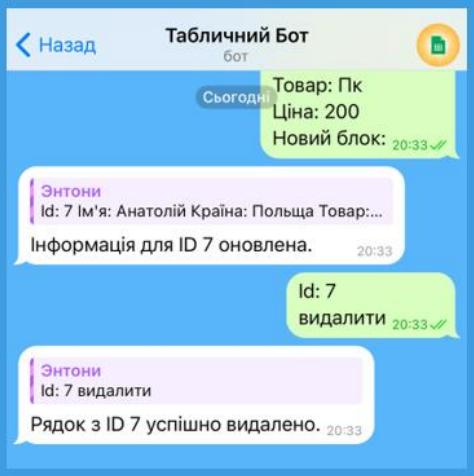
### Оновлення часу

Після кожної зміни бот автоматично оновлює стовпець "Оновлення" з поточною датою та часом




## Процес додавання нових записів

| Оновлення | id | Ім'я |
|---|---|---|
| 27.04.2025 2:08:00 | 1 | Володимир |
| 27.04.2025 2:08:00 | 2 | Антон |
| 29.04.2025 15:17:00 | 3 | Толик |
| 27.04.2025 2:08:00 | 6 | Ксенія |
| 18.05.2025 19:44:00 | 7 | Анатолій |




# Процес видалення записів

| Оновлення | id | Ім'я |
|---|---|---|
| 27.04.2025 2:08:00 | 1 | Володимир |
| 27.04.2025 2:08:00 | 2 | Антон |
| 29.04.2025 15:17:00 | 3 | Толик |
| 27.04.2025 2:08:00 | 6 | Ксенія |

# Перспективи розвитку

**Можливі покращення які збільшать функціонал і актуальність нашого сервісу**

- **Підвищення безпеки**
- **Інтеграція AI (підказки, аналіз помилок)**
- **Підтримка кількох таблиць**
- **Автоматичне резервне копіювання**
- **Голосове введення**

# Висновки

1. Бот спрощує щоденну роботу з таблицями
2. Підходить для малого бізнесу
3. Реалізовано базові функції
4. Є потенціал для масштабування

# Дякую за увагу