МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ УПРАВЛІННЯ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри

_____ Литвиненко О.Є.

«___» _____2022 р.

ДИПЛОМНИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬО–КВАЛІФІКАЦІЙНОГО РІВНЯ "БАКАЛАВР"

Тема: «Телеграм бот для менеджменту фінансів»

Виконавець: Даньшов Д.О.

Керівник: Масловський Б.Г.

Нормоконтролер: Тупота Є.В.

Київ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютеризованих систем управління

Спеціальність 123 «Комп'ютерна інженерія»

Спеціалізація «Системне програмування»

ЗАТВЕРДЖУЮ
Завідувач кафедри КСУ

_____ Литвиненко О.Є.

«___» _____2022 р.

ЗАВДАННЯ
на виконання дипломного проекту

Даньшова Дмитра Олексійовича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема проекту: Телеграм бот для менеджменту фінансів

затверджена наказом ректора від " 15 " лютого 2022 року №251/ст.

2. Термін виконання проекту: з 15.05.2022 до 19.06.2022

3. Вихідні дані до проекту:

Загальні правила побудови телеграм ботів використання мови Python при створенні телеграм боту

ДСТУ 3008-95 Документація. Звіти у сфері науки і техніки. Структура і правила оформлення

Положення про дипломні роботи(проекти) випускників НАУ

СМЯ НАУ П 03.01(10) – 02 – 2017

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):

1) Аналіз сучасних засобів створення бота
2) Програмна реалізація Телеграм бота
3) Телеграм бот для менеджменту фінансів

5. Перелік обов'язкового графічного матеріалу:

1) Телеграм бот для менеджменту фінансів(інтерфейс бота)
2) Телеграм бот для менеджменту фінансів(схема алгоритму)
3) Алгоритм обробки команд "допомога" "вибір"
4) Алгоритм обробки команд "прибуток" "витрати"
5) Алгоритм обробки команди "історія"
6) Алгоритм обробки команди "видалити запис""

6. Календарний план

№ п/п	Етапи виконання дипломної роботи	Термін виконання етапів	Примітка
1	Здобуток інформації на тему проектування телеграм ботів на мові програмування Python	15.05.22-16.05.22	
2	Дослідження гайдів по створенню ботів	17.05.22-19.05.22	
3	Встановлення програмного середовища VS Code та завантаження стандартної бібліотеки Python	20.05.22-21.05.22	
4	Завантаження бібліотек Python що спроектовані спеціально для роботи з телеграм ботами aiogram та pyTelegramBotAPI	22.05.22-23.05.22	
5	Реєстрація свого бота у додатку телеграм, та зв'язок його зі своїм програмним кодом	24.05.22-26.05.22	
6	Написання базового exo бота для перевірки роботи бота у додатку	27.05.22-29.06.22	
7	Завантаження програмного середовища для створення баз данних SQLite Studio	30.06.22-01.06.22	
8	Створення бази даних для збереження інформації про користувачів та про їх витрати які вони занесли до бази даних для менеджменту фінансів	02.06.22-04.06.22	
9	Написання логіки бота щоб він міг відповідати користувачам, та вносив їх інформацію до бази даних	05.05.22-10.06.22	
10	Оформлення звіту про проходження технологічної практики.	11.06.22-13.06.22	

7. Дата видачі завдання _____

Керівник дипломної роботи _____ Масловський Б.Г
(підпис)

Завдання прийняв до виконання _____ Даньшов Д.О.
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Представлений дипломний проект складається з 3 розділів

– Аналіз сучасних засобів створення бота

– Програмна реалізація Телеграм бота

– Телеграм бот для менеджменту фінансів

Актуальність теми. Сьогодні, в роки розвитку всіх можливих інтернет-технологій, а також розвитку мобільних програм з різними можливостями, постає проблема з необхідністю завантажити на свій смартфон десятки додатків, які допомагають нам у різних сферах життя. Вони значно полегшують вирішення різноманітних задач користувачам, але оскільки додаток має лише одну функцію і вирішує конкретну проблему в одній області, користувачам доводиться завантажувати велику кількість одноцільових програм.

Telegram Bot - це багатофункціональний застосунок для вирішення проблем і надання послуг користувачам. Крім того, цей додаток безкоштовний. Бот — це такий тип програми, створений у сучаній програмі для обміну повідомленнями і файлами Telegram з використанням технології штучного інтелекту, і бот може виконувати багато різних функцій.

Метою цього дипломного проекту буде проектування Телеграм бота для менеджменту фінансів. Цей бот повинен вміти працювати з базою даних, та опрацьовувати данні у базі.

Розробка. У функції бота повинні входити: занесення операцій у базу даних, зчитання історії операцій за період, видалення операції.

При створенні фінансової операції бот має давати користувачу вибір типу операції: прибуток, витрати; якщо це витрати то обрати тип витрат.

Предметом дослідження є технології створення телеграм-ботів на мові програмування Python. Дослідження автоматизації процесів підтримки клієнтів.

Вихідним результатом дипломного проекту є працюючий Телеграм бот для менеджменту фінансів. Завдяки якому користувачі можуть відслідковувати свою фінансову активність.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

VoIP — технологія передачі медіа-даних у реальному часі за допомогою сімейства протоколів TCP/IP.

Бот — спеціальна програма, що виконує автоматично і/або за заданим розкладом які-небудь дії через ті ж інтерфейси, що й звичайний користувач.

Чат — мережевий засіб для швидкого обміну текстовими повідомленнями між користувачами інтернету в системі реального часу.

Інтерфейс — сукупність засобів, методів і правил взаємодії (керування, контролю, тощо) між елементами системи.

Токен — унікальний ключ для зв'язку та передачі інформації.

ВСТУП

Телеграм — багатоплатформовий клауд-месенджер з функціями VoIP для смартфонів, планшетів та ПК, який дозволяє обмінюватися текстовими, голосовими та відеоповідомленнями, наліпками та фотографіями, файлами багатьох форматів. Також має функції відео і аудіодзвінків, організації відеконференцій у групах і каналах. Клієнтські програми Telegram доступні для Android, iOS, Windows Phone, Windows, macOS і GNU / Linux . Кількість щомісячних активних користувачів сервісу станом на січень 2021 року становить близько 500 млн осіб.

У Telegram працює платформа чатботів. Боти можуть виконувати різноманітні завдання, такі як пошук в інтернеті чи держреєстрах, покупки, платежі, розваги, модерація груп тощо. У спілкуванні беруть участь користувач Telegram та комп'ютерна програма від стороннього розробника.

Користувач може взаємодіяти з ботом за допомогою елементів інтерфейсу месенджера: надсилання повідомлень, натискання на команди та кнопки, використання інлайн-режиму. Telegram надає три способи взаємодії користувача з ботом: приватний чат (класичний спосіб), група й так званий інлайн-режим:

Найпоширеніший спосіб — приватний чат. Здебільшого бот не може ініціювати діалог із користувачем.

Деякі боти можуть бути учасниками груп. Наприклад, у групах бот може підтримувати розмову, модерувати повідомлення або бути ведучим гри.

Інлайн-режим нагадує інтерфейс пошукової системи. Користувач уводить у поле для введення повідомлень запит, що починається з короткого імені бота. Далі користувач може вибрати й надіслати один з результатів.

Деякі боти можуть бути учасниками каналів. У каналах ботів застосовують здебільшого для запланованого розміщення повідомлень. З такими ботами користувач взаємодіє через приватний чат або вебінтерфейс.

РОЗДІЛ 1

Аналіз сучасних засобів створення бота

Перший крок у роботі з будь-якою мовою програмування полягає у налаштуванні необхідних інструментів. Більшість розробників використовують редактор коду, наприклад Visual Studio Code, оскільки ці інструменти пропонують автозаповнення коду та виділяють синтаксичні помилки. Редактори коду також надають доступ до великого набору надбудов для подальшого покращення процесу розробки.

1.1 Середа розробки Visual Studio Code

Visual Studio Code — засіб для створення, редагування та зневадження сучасних вебзастосунків і програм для хмарних систем. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X. За основу для Visual Studio Code використовуються напрацювання вільного проєкту Atom, що розвивається компанією GitHub. Зокрема, Visual Studio Code є надбудовою над Atom Shell, що використовує браузерний рушій Chromium і Node.js.

Python – це високорівнева об'єктно-орієнтована та структурна мова програмування загального призначення, яка відноситься до категорії інтерпретованих мов і не потребує компіляції. Це є скриптова мова і відрізняється високим ступенем універсальності. Завдяки цьому, оптимально підходить для безлічі платформ та завдань, від серверних ОС до мобільних додатків під iOS/Android.

Рис.1.1.1 встановлення Python

Після вибору середи для написання та мови, наступним кроком стане під'єднання бібліотек. Під'єднання у «Python» проводиться через команду «import». Щоб наш «Python» розумів, що ми хочемо робити з "незрозумілими" для нього командами, ми прописуємо список бібліотек, які він буде використовувати.

Для програмування бота потрібно ще встановити одну з бібліотек які спеціально розроблені для такої задачі: aiogram або Telegram Bot API. Для їх встановлення треба прописати команди у теміналі VS Code

pip install -U aiogram

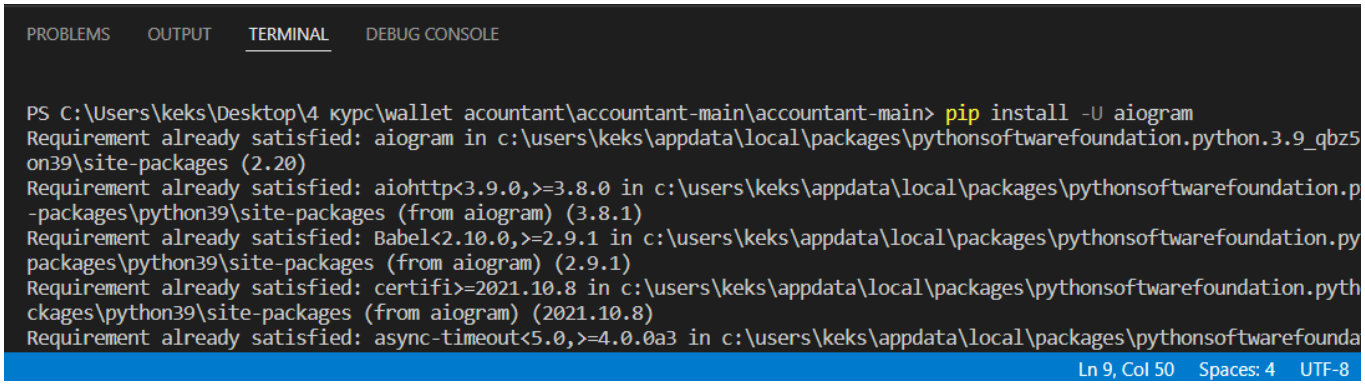


Рис.1.1.2 всановлення бібліотеки aiogram

1.2 Інструмент створення баз даних SQLiteStudio

SQLite — полегшена реляційна система керування базами даних. Втілена у вигляді бібліотеки, де реалізовано багато зі стандарту SQL-92. Сирцевий код SQLite поширюється як суспільне надбання, тобто може використовуватися без обмежень та безоплатно з будь-якою метою. Фінансову підтримку розробників SQLite здійснює спеціально створений консорціум, до якого входять такі компанії, як Adobe, Oracle, Mozilla, Nokia, Bentley і Bloomberg.

Інструмент SQLiteStudio — це безкоштовний інструмент з графічним інтерфейсом для керування базами даних SQLite. Він безкоштовний, портативний, інтуїтивно зрозумілий і кросплатформний. Інструмент SQLite також надає деякі з найважливіших функцій для роботи з базами даних SQLite, такі як імпорт, експорт даних у різних форматах, включаючи CSV, XML та JSON.

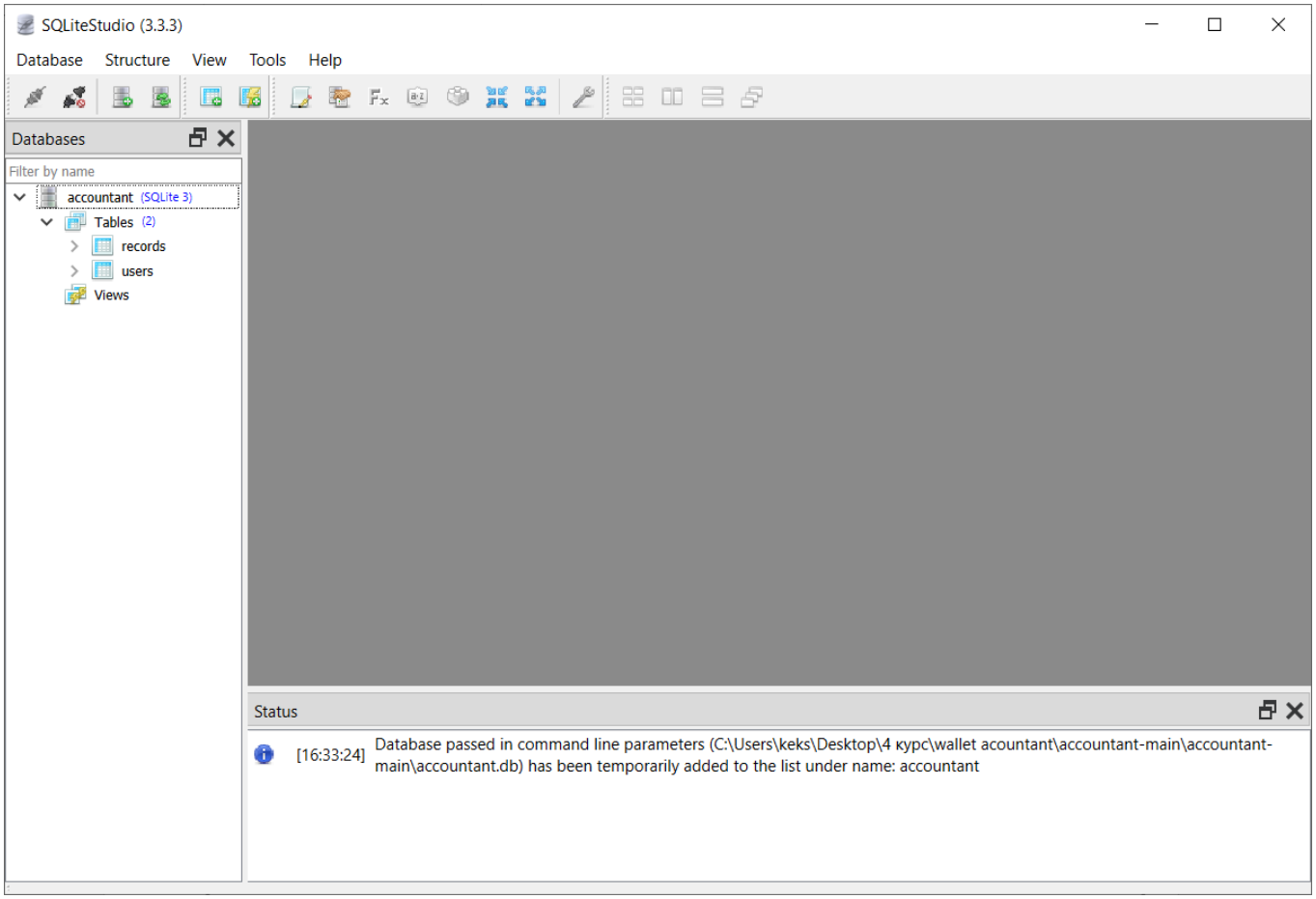


Рис.1.2.1 вигляд інтерфейсу SQLiteStudio

1.3 Висновки до розділу

Отже, в даному розділі було описано програмне забезпечення, що використовується для реалізації проекту.

Для розробки теми дипломної роботи необхідно наступне:

1) Розробити схему алгоритму

2) Провести програмну реалізацію алгоритму

3) Провести тестування програми

РОЗДІЛ 2

Програмна реалізація Телеграм бота

2.1 Реєстрація бота у додатку Телеграм

У месенджері Telegram є бот BotFather(рис.2.1.1). - помічник, який допомагає користувачам створити свого власного бота та управляти ботами.

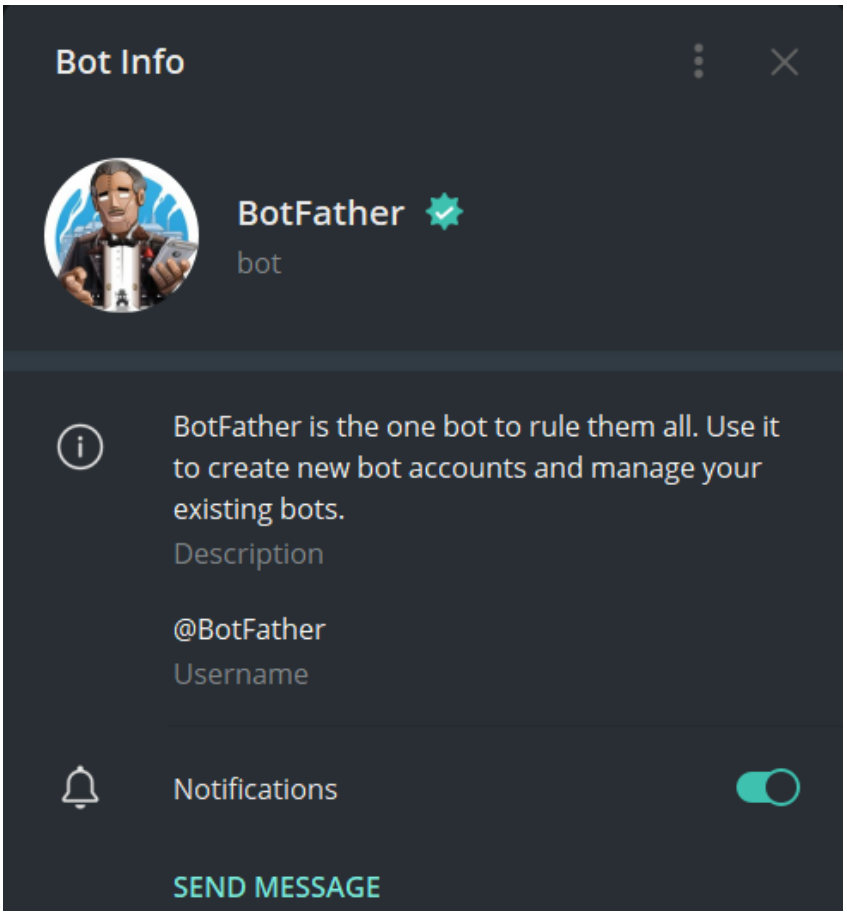


Рис.2.1.1 Інтерфейс бота BotFather

У BotFather є 2 основні команди:

– /newbot дозволяє створити новий бот.

– /mybots запускає редактор налаштувань ваших роботів.

6 команд для редагування:

– /setname для зміни назви бота.

– /setdescription для зміни опису бота. Короткий опис бота. Дозволяється не більше 512 символів. Цей текст буде видно користувачеві під час ініціалізації діалогу.

– /setabouttext — запускає редактор інформації "про бот". Інформація про робота відображається в його профілі, максимум 120 символів.

– /setuserpic — відповідає за запуск редактора фотографії профілю бота. Розмір фотографії профілю повинен бути не меншим ніж 200×200 пікселів. Також бажано завантажувати квадратні зображення.

– /setcommands – запускає редактор списку команд бота. Команди потрібні для управління та взаємодії з ботом. Кожна команда починається з косої межі "/", можна використовувати латинські літери, цифри, а також підкреслення. Але не більше ніж 32 символи. Також необхідно вказати опис команди. Цей список команд можна побачити, написавши косу межу «/» у діалозі з ботом.

– /deletebot - команда для видалення бота

7 команд для налаштування бота:

– /token — команда для генерації токена.

– /revoke — команда зміни токена.

– /setinline — команда увімкнення inline mode.

– /setinlinegeo — увімкнення розташування.

– /setinlinefeedback — відкриває налаштування зворотного зв'язку.

– /setjoingroups — дозволяє увімкнути та вимкнути можливість додавання ботів у групи.

– /setprivacy — команда для активації режиму конфіденційності у групах. Можна вказати, які повідомлення бот може отримувати в групах.

Для створення бота потрібно ввести у чат із BotFather команду /newbot. Далі ввести назву. Підтримується кирилиця та латиниця, наприклад: Студент bot. Ім'я відображатиметься в заголовку та в інформації про бота.

Першим кроком у програмуванні чат-бота є отримання його персонального токену. Персональний токен – це унікальний номер, за яким працює чат-бот, без нього програма не дізнається через кого будуть оброблятися запити. Щоб отримати персональний тег для бота у сервісі Telegram, необхідно поспілкуватись з «BotFather». У результаті ми отримаємо нашого чат-бота, його токен, задамо йому ім’я та оберемо картинку.

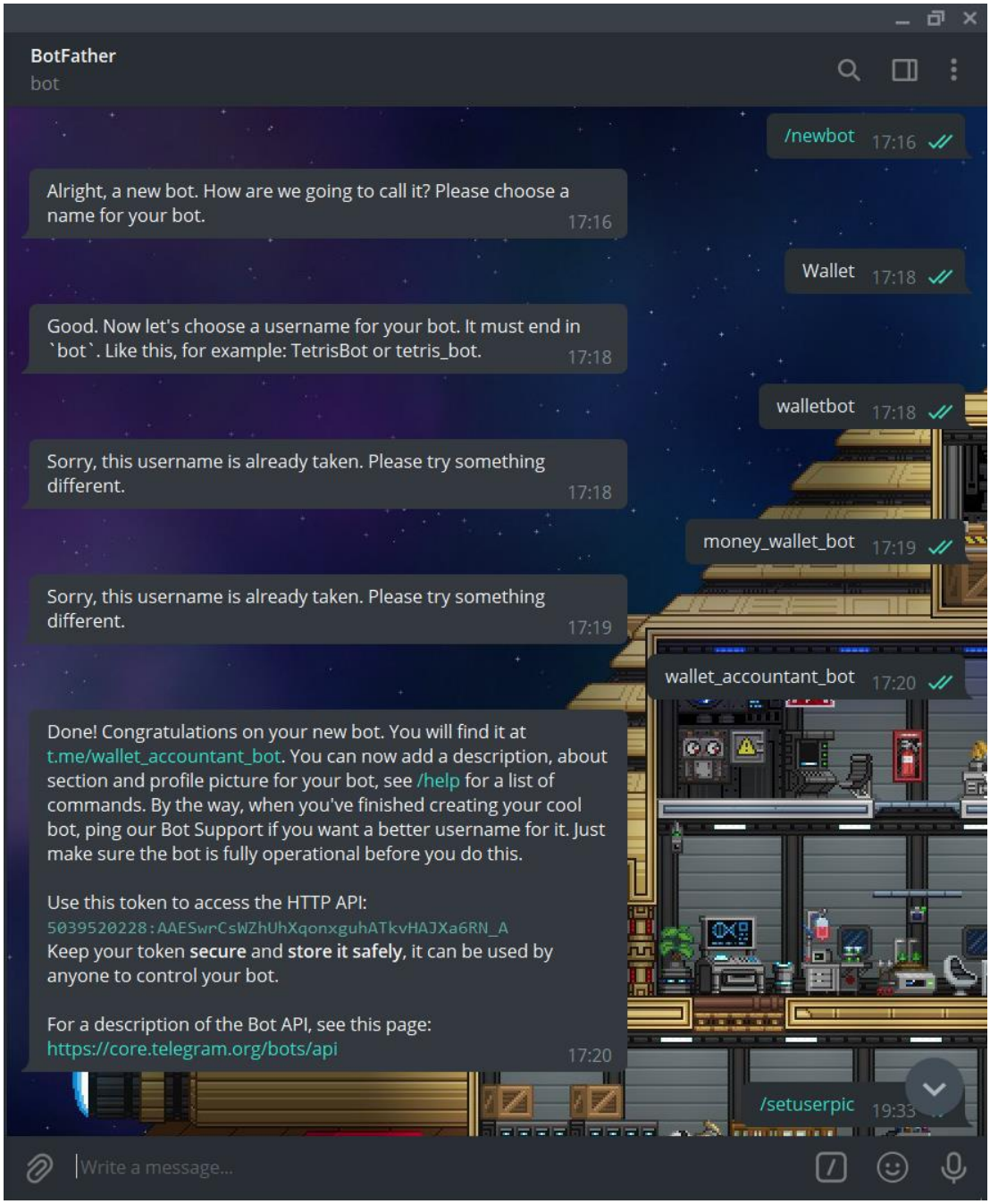


Рис 2.1.2 процес реєстрації бота

2.2 Реалізація най простішого бота

За для перевірки працездатності бота потрібно написати його найпростішу форму, ехо-бот – бот який буде як папуга відправляти нам наші ж повідомлення, таким чином ми зможемо переконатися що бот працює.

«Import telebot» – бібліотека для розпізнавання рядків коду пов'язаного зі створенням телеграм бота.

Після реєстрації бота ми отримали токен. Цей токен можна використовувати для авторизації бота та надсилання запитів до Bot API.

Якщо токен було втрачено, можна згенерувати його знову. Для цього потрібно написати боту BotFather команду /token, вибираємо потрібний бот, отримуємо новий токен.

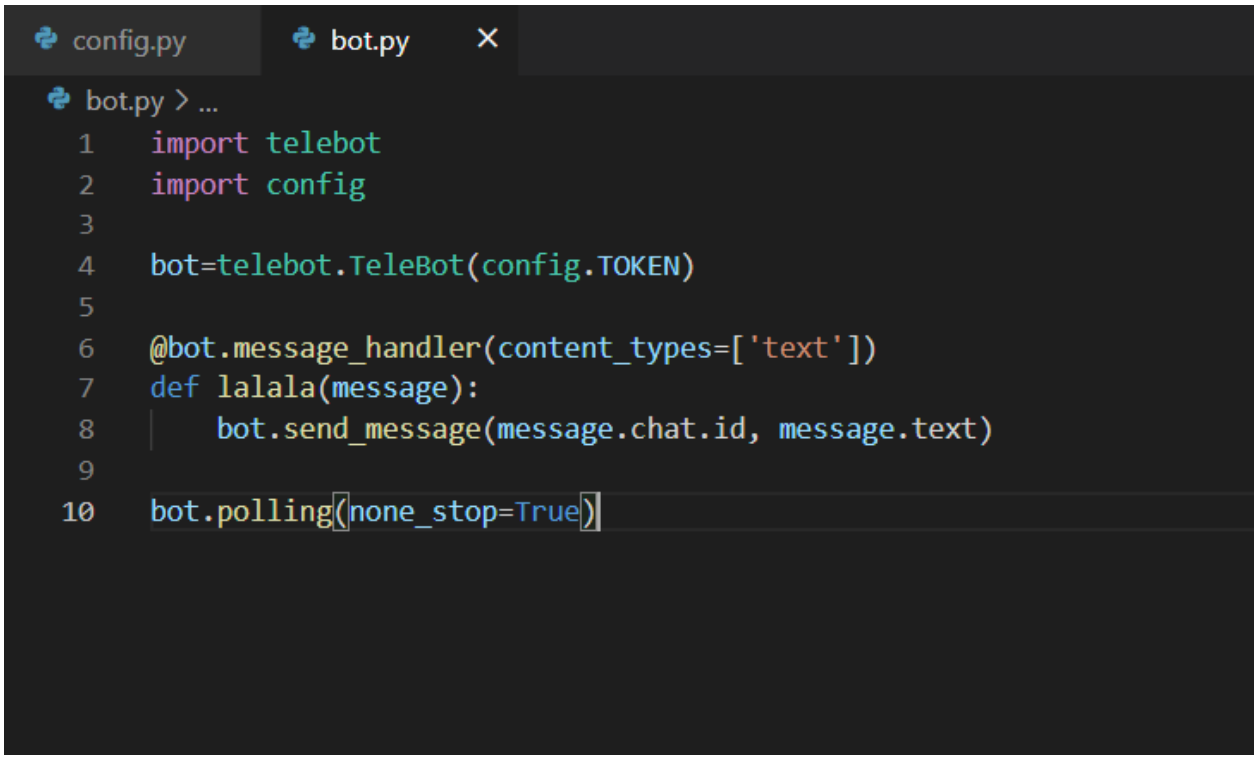


Рис 2.2.1 лістинг коду єхо-бота

Пропис токену здійснюється за командою «bot = telebot.TeleBot("*")», де "*" – ваш токен. Даною командою задається токен для нашого телеграм бота, без нього він не буде функціонувати.

Токен бота прописаний у іншому файлі для полегшення доступу до нього та за для оптимізації програмного процесу у майбутньому, щоб не шукати його по всьому файлу з кодом коли бот стане більшим та складнішим.

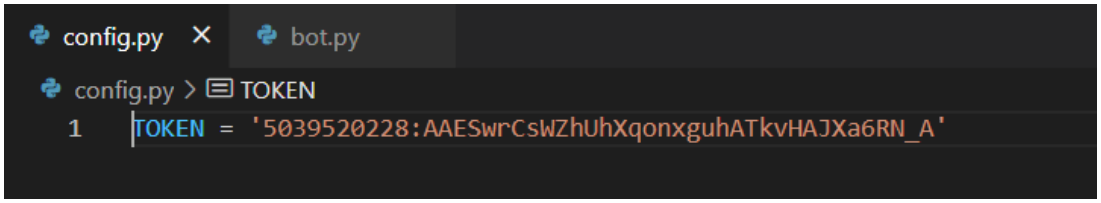


Рис.2.2.2 прописаний токен у конфігу бота

Після програмування бота, його можна запустити обравши файл bot.py та натиснувши на кнопку стрілки у правому верхньому куту

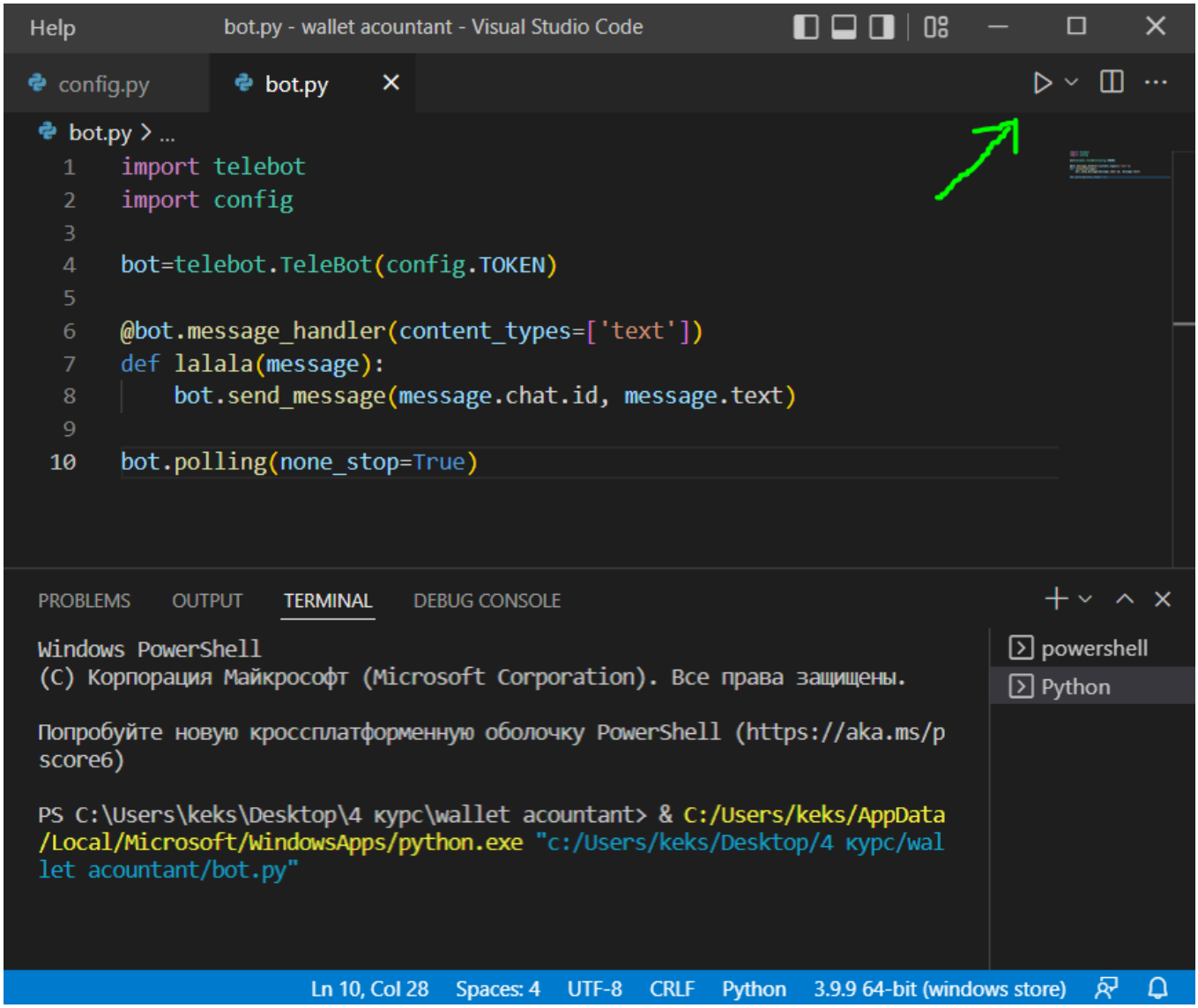


Рис.2.2.3 запуск бота у VS Code

Як можемо бачити бот працює та копіює і повертає на наше повідомлення. Це означає що токен бота прописаний правильно та помилок у коді не було.

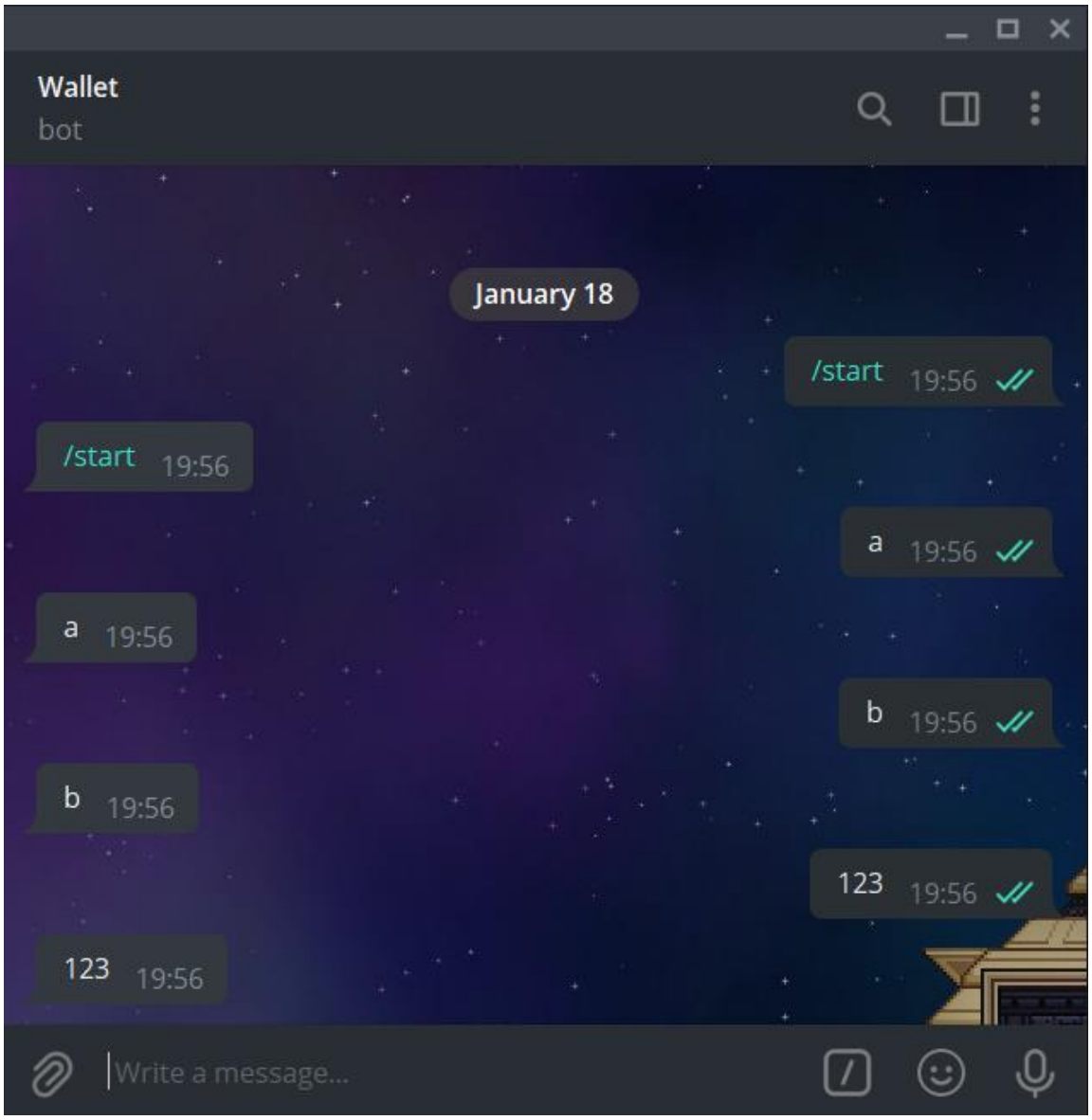


Рис 2.2.3 відпрацювання ехо-бота

2.3 Висновки до розділу

Створено першого бота який працює справно, це був перший крок по створенню більш складних ботів. Тепер на цій основі буде легше працювати над додатком функціоналу до бота.

РОЗДІЛ 3

Телеграм бот для менеджменту фінансів

Так як тема дипломного проекту «Телеграм бот для менеджменту фінансів» то він має уміти зберігати та обробляти інформацію яку йому надають користувачі. До функціоналу бота має входити:

- Привітання користувача при першому підключенні бота
- Пояснення користувачу командою /help як працювати із ботом
- Додавання фінансового запиту до бази даних користувача
- Видалення фінансового запиту з бази даних користувача
- Виведення історії транзакцій за певний період часу користувача

Щоб зберігати інформацію про користувачів та їх фінансові операції буде створена база даних на основі SQLite у додатку SQLiteStudio з таблицями користувачів та їх транзакцій.

3.1 Створення таблиць баз даних

Перша таблиця бази даних users, її описання:

- Id - номер користувача у базі
- user_id – id користувача у Телеграмі
- join_date – дата занесеня користувача у базу даних

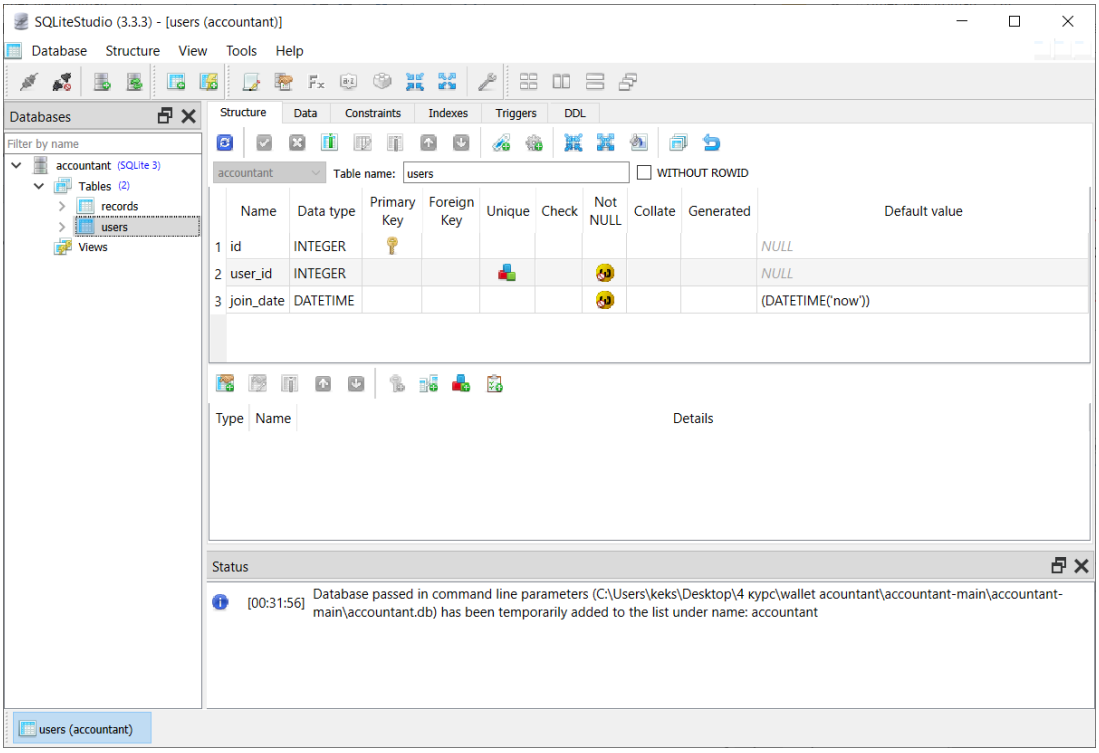


Рис.3.1.1 таблиця users

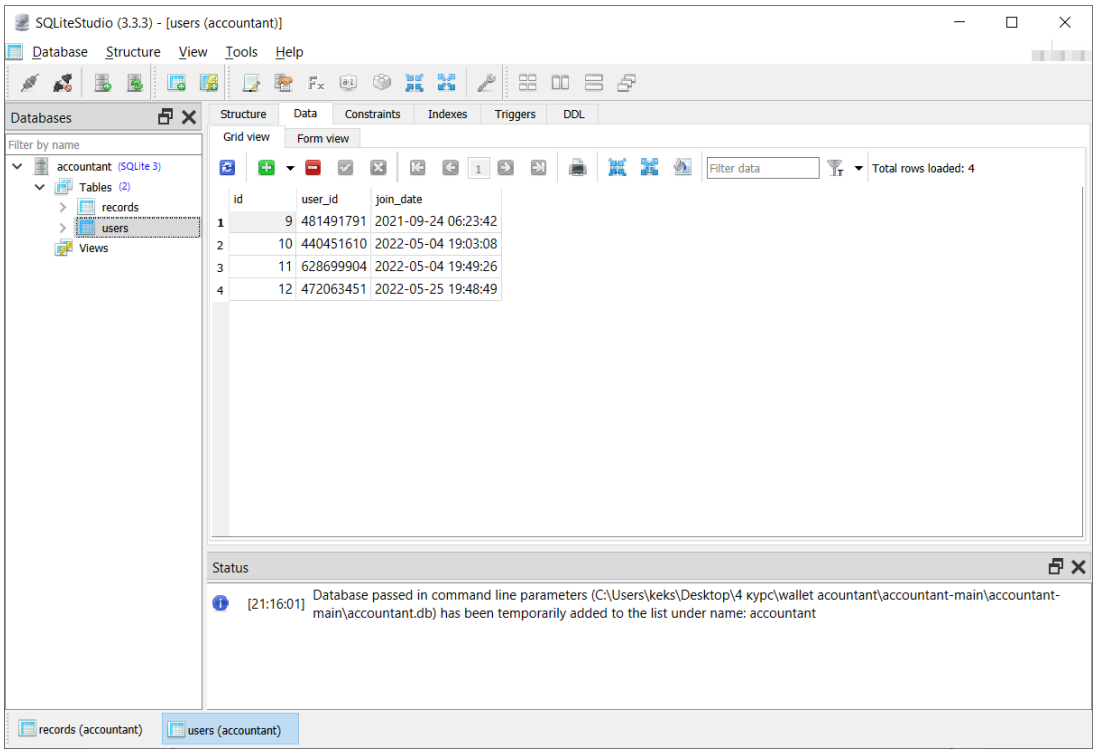


Рис.3.1.2 вид таблиці users заповненою інформацією

На кожного користувача створюється таблиця бази даних records, її описання:

- Id - номер операції користувача у базі
- user_id – id користувача у Телеграмі
- operation – прибуток(true)/витрати(false)
- record_type – тип операції(на що пішли гроші)
- value – сума грошей в операції
- date – дата операції

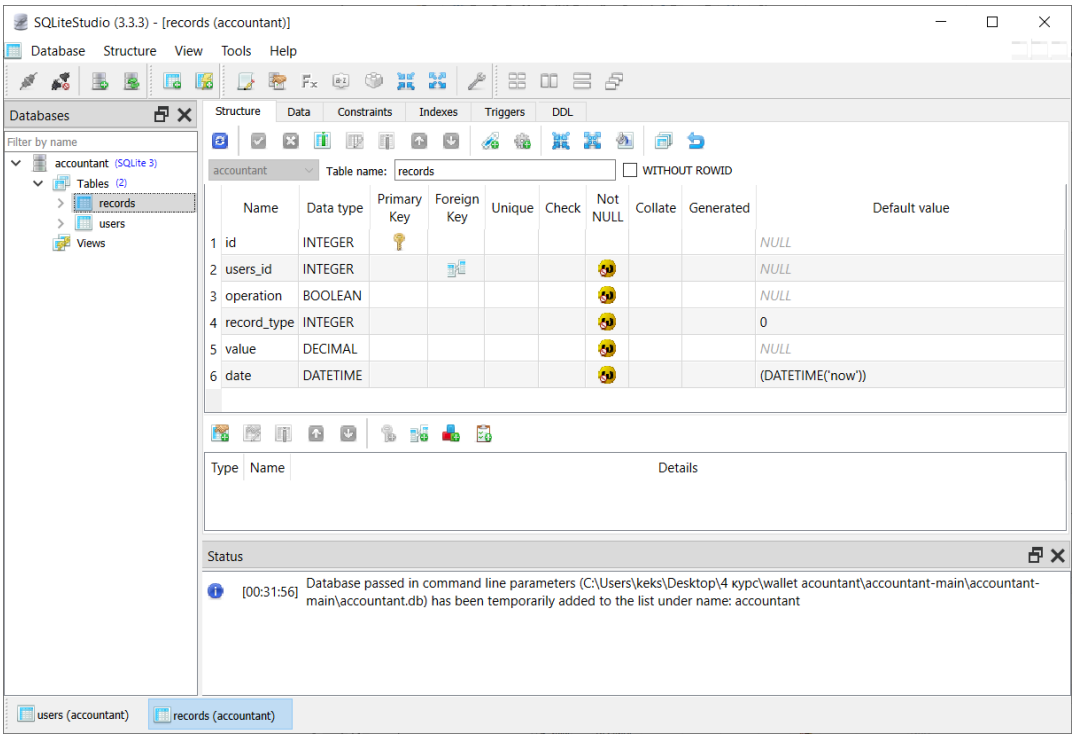


Рис.3.1.3 таблиця records

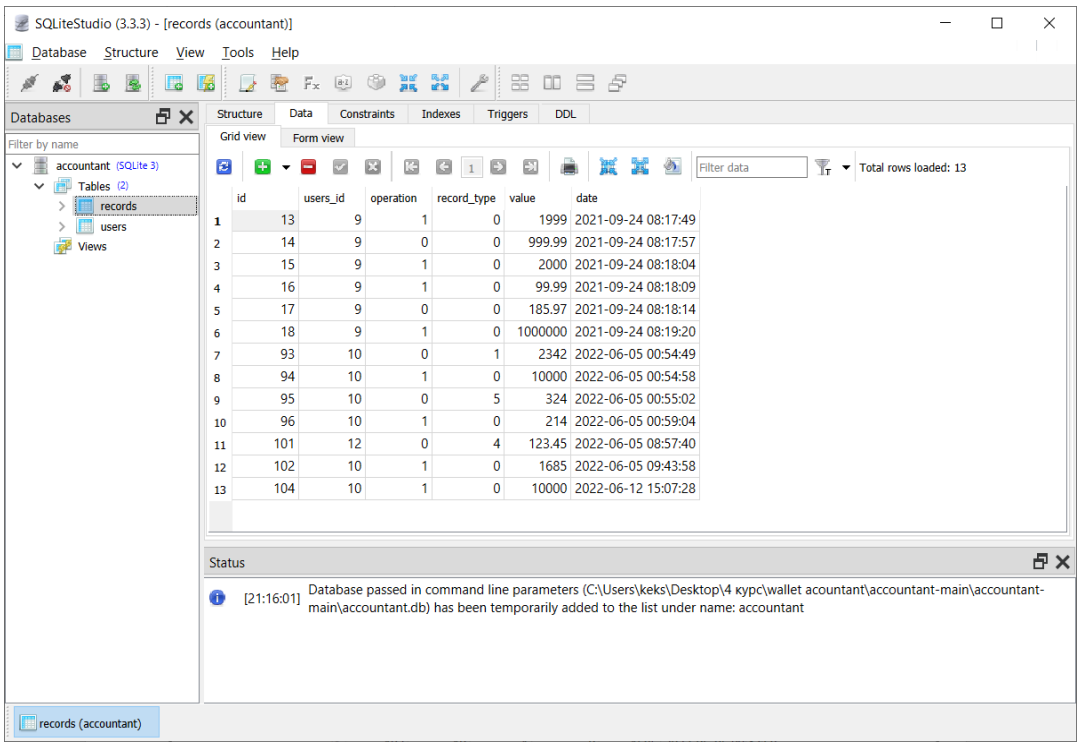


Рис.3.1.4 вид таблиці records заповненою інформацією

3.2 Описання функцій класу для взаємодії з базою даних

Щоб бот міг заносити та зчитувати дані з бази даних потрібно описати клас взаємодії бота з даними у базі.

Підключення бази даних

```
    def __init__(self, db_file):
        self.conn = sqlite3.connect(db_file)
        self.cursor = self.conn.cursor()
```

Перевірка чи існує юзер

```
    def user_exists(self, user_id):
        result = self.cursor.execute("SELECT `id` FROM `users` WHERE `user_id` = ?", (user_id,))
        return bool(len(result.fetchall()))
```

Отримання id юзера у базі даних по його id у Телеграмі

```
    def get_user_id(self, user_id):
        result = self.cursor.execute("SELECT `id` FROM `users` WHERE `user_id` = ?", (user_id,))
        return result.fetchone()[0]
```

Занесення юзера у базу даних

```
    def add_user(self, user_id):
        self.cursor.execute("INSERT INTO `users` (`user_id`) VALUES (?)", (user_id,))
        return self.conn.commit()
```

Занесення фінансової операції юзера у базу даних

```
    def add_record(self, user_id, operation, value):
        self.cursor.execute("INSERT INTO `records` (`users_id`, `operation`, `value`) VALUES (?,
?, ?)",
            (self.get_user_id(user_id),
            operation == "+",
            value))
        return self.conn.commit()
```

Зміна типу фінансової операції

```
    def change_record_type(self, user_id, record_type):
        self.cursor.execute("UPDATE `records` set `record_type` = ?, `users_id` = ?
WHERE `date` = (SELECT MAX(`date`) FROM `records`) ",
            (record_type,
            self.get_user_id(user_id)))
        return self.conn.commit()
```

Видалення останньої фінансової операції

```
    def remove_record(self,user_id):
        self.cursor.execute("DELETE FROM `records` WHERE `date` = (SELECT
MAX(`date`) FROM (SELECT * FROM `records` WHERE `users_id` = ?)) AND `users_id` =
?",
            (self.get_user_id(user_id),self.get_user_id(user_id),))
```

Виведення історії операцій юзера

```
    def get_records(self, user_id, within = "all"):
        if(within == "day"):
            result = self.cursor.execute("SELECT * FROM `records` WHERE `users_id` = ? AND `date`
BETWEEN datetime('now', 'start of day') AND datetime('now', 'localtime') ORDER BY `date`",
                (self.get_user_id(user_id),))
        elif(within == "week"):
             result = self.cursor.execute("SELECT * FROM `records` WHERE `users_id` =
? AND `date` BETWEEN datetime('now', '-6 days') AND datetime('now', 'localtime')
ORDER BY `date`",
                (self.get_user_id(user_id),))
        elif(within == "month"):
            result = self.cursor.execute("SELECT * FROM `records` WHERE `users_id` = ? AND `date`
BETWEEN datetime('now', 'start of month') AND datetime('now', 'localtime') ORDER BY `date`",
                (self.get_user_id(user_id),))
        else:
            result = self.cursor.execute("SELECT * FROM `records` WHERE `users_id` = ? ORDER BY
`date`",
                (self.get_user_id(user_id),))

        return result.fetchall()
```

Відключення від бази даних

```
    def close(self):
        self.connection.close()
```

Після створення бази даних та методу зв’язку її з кодом бота, можна приступити до проектування логіки бота.

3.3 Схема роботи бота

За такою логікою бот буде обробляти команди користувача. Так як схема велика, вона розбита на менші в яких описується алгоритм обробки певної команди яку обрав користувач.

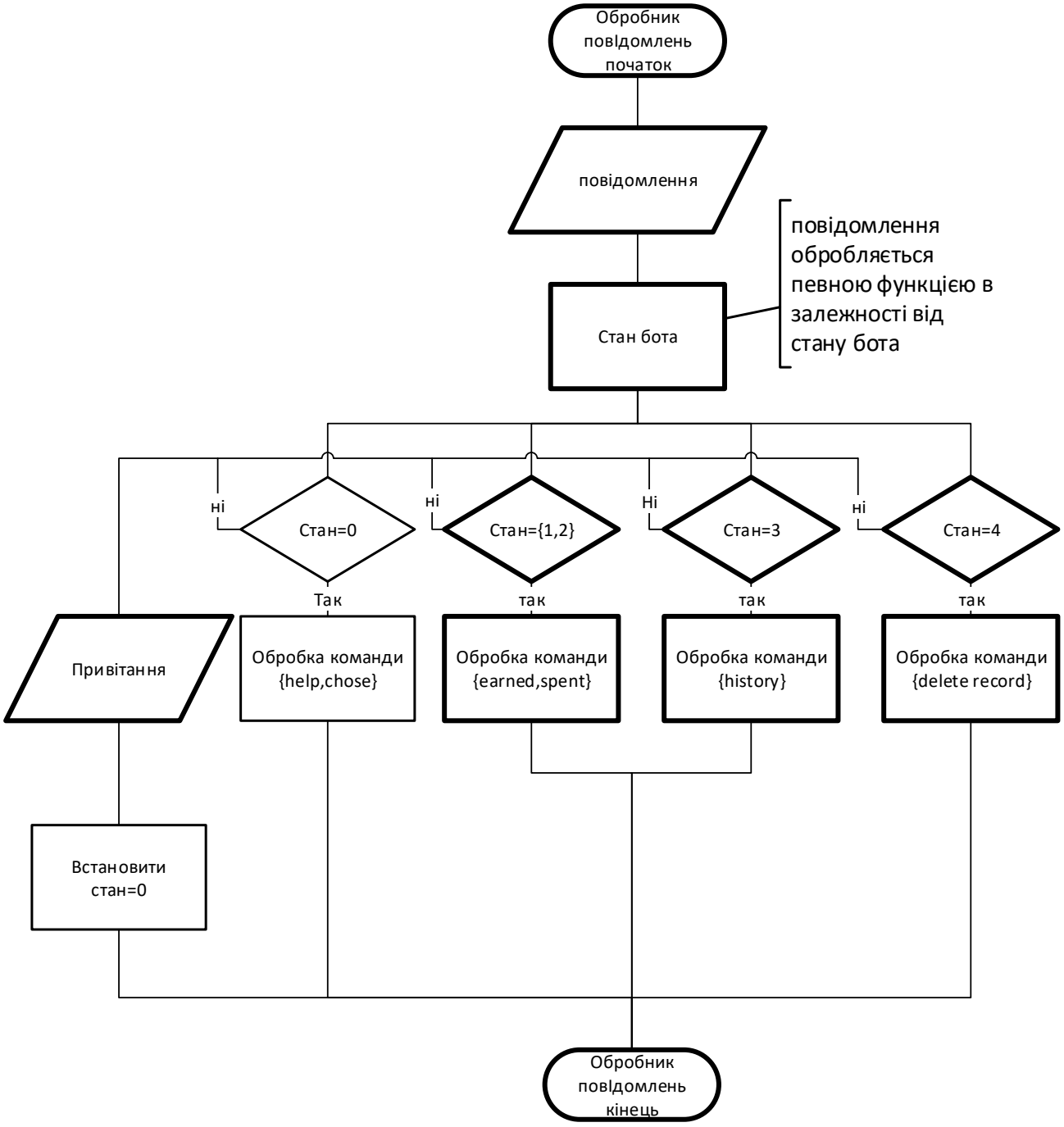


Рис.3.3.1 загальна схема роботи бота

Стан {0} це стан бота за замовчуванням, в якому користувачу буде представлений вибір функцій бота представлених на його клавіатурі. Також у цьому стані працює команда /help для ознайомлення з ботом.

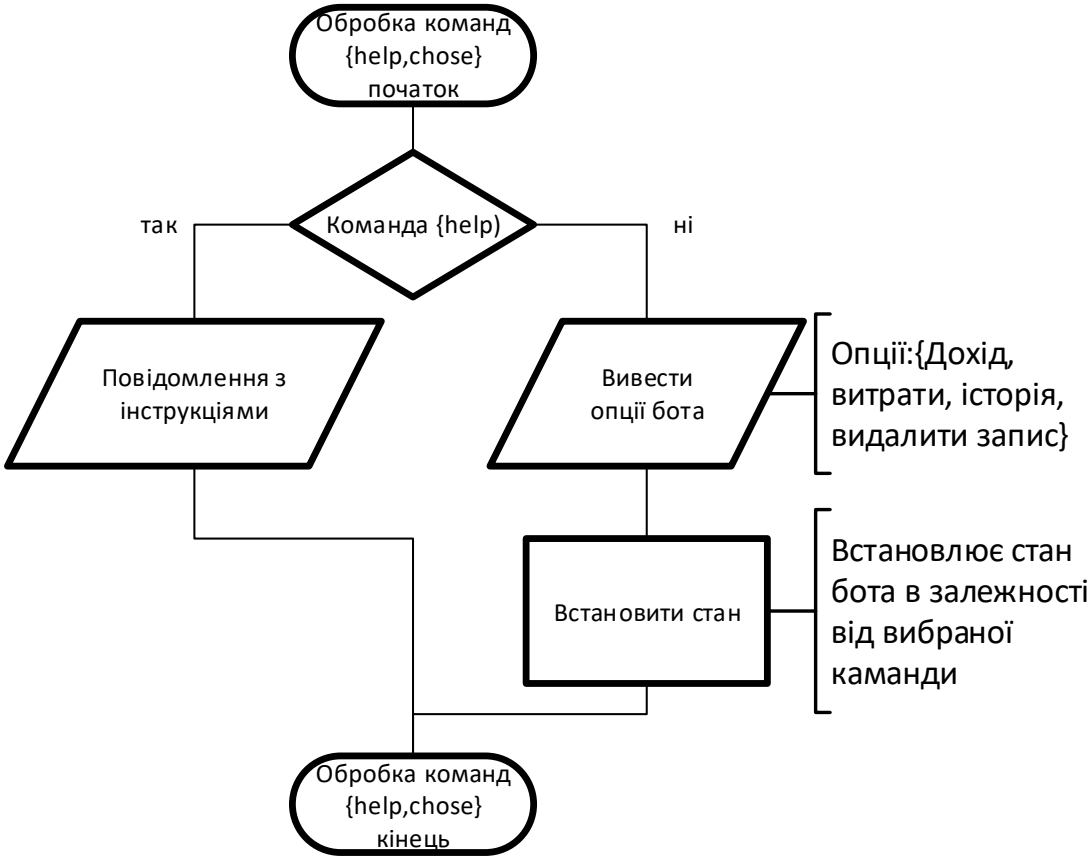


Рис.3.3.2 алгоритм обробки команд “допомога” “вибір”

Стан {1,2} це стани в яких виконуються функції занесення даних фінансових операцій у базу даних. Бот переходить у ці стани при виборі функцій “прибуток” “витрати”.

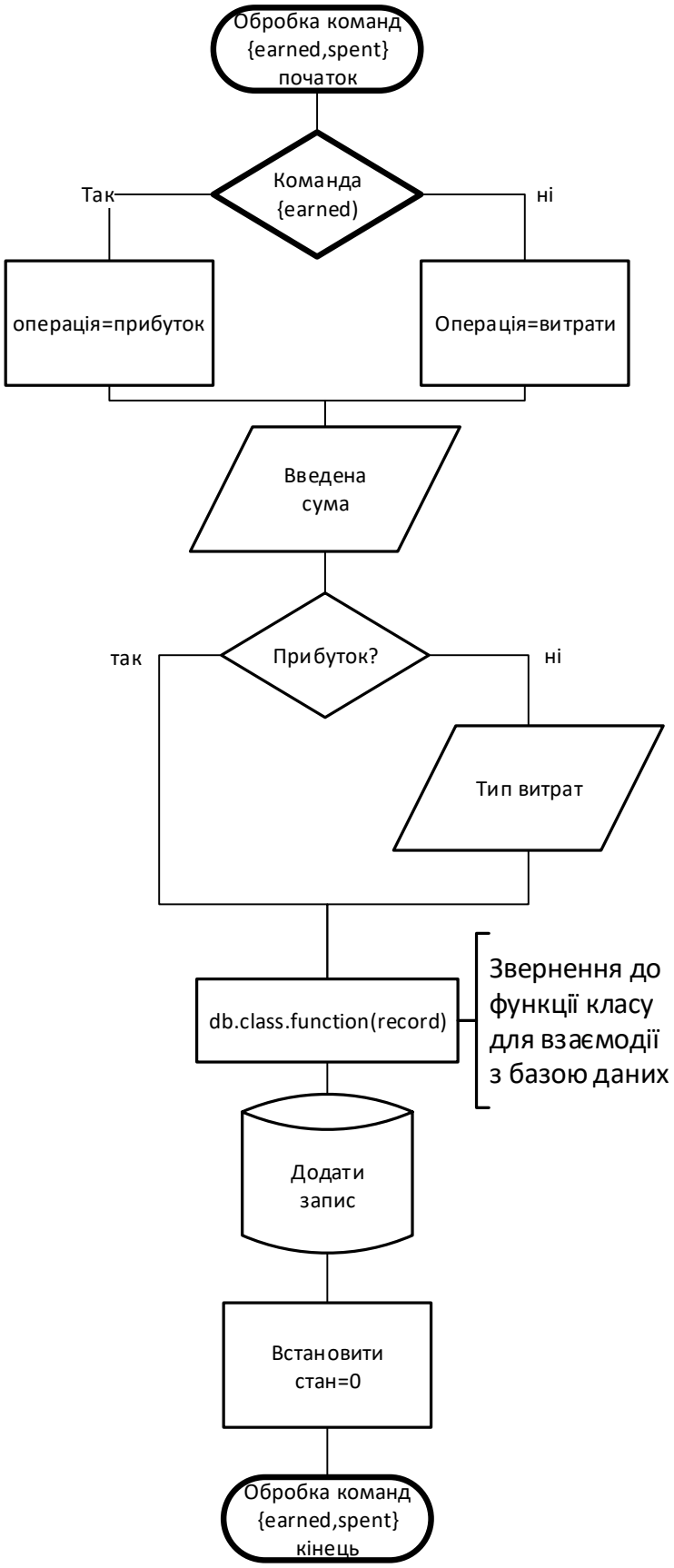


Рис.3.3.3 алгоритм обробки команд “прибуток” “витрати”

Бот переходить у стан {3} коли було обрано функцію “історія”, для виводу фінансової інформації за період часу.

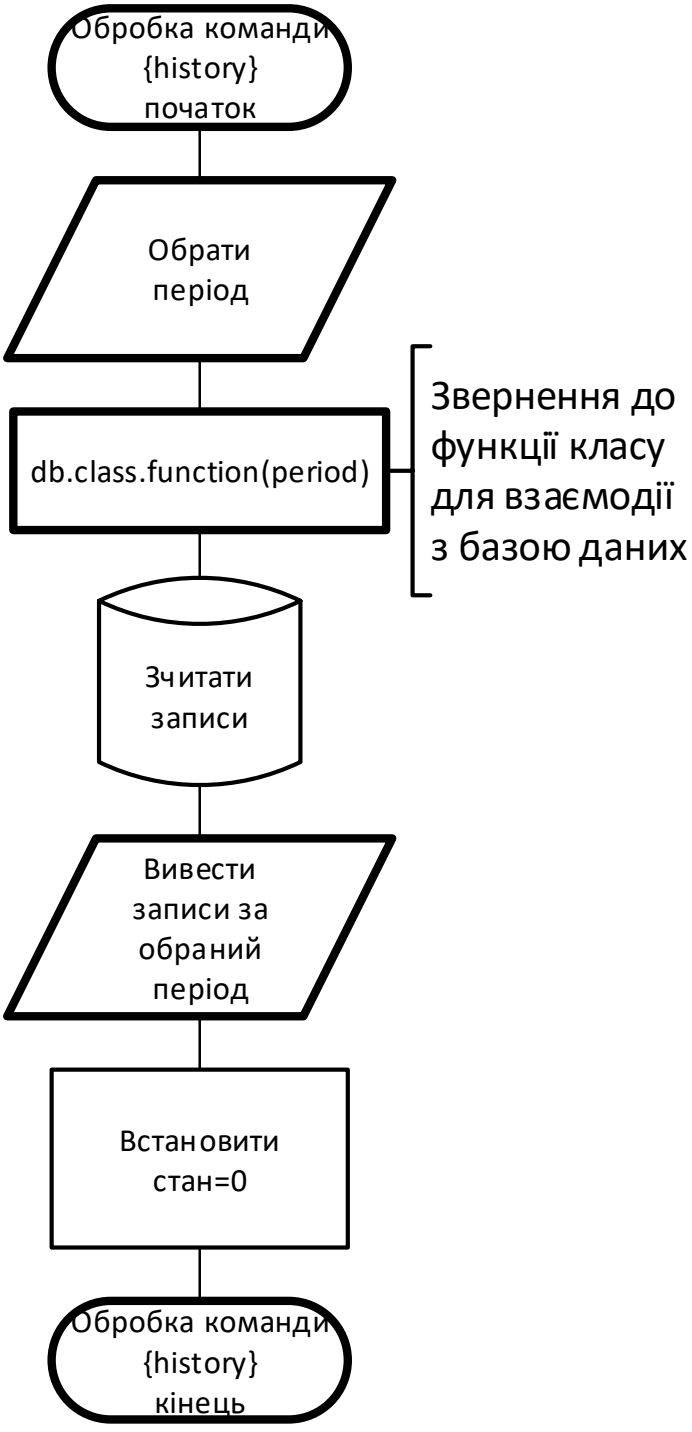


Рис.3.3.4 алгоритм обробки команди “історія”

Бот переходить у стан {4} коли було обрано функцію “видалити запис”, при обробці цієї команди буде видано запит користувачу для підтвердження видалення останньої фінансової операції цього користувача.

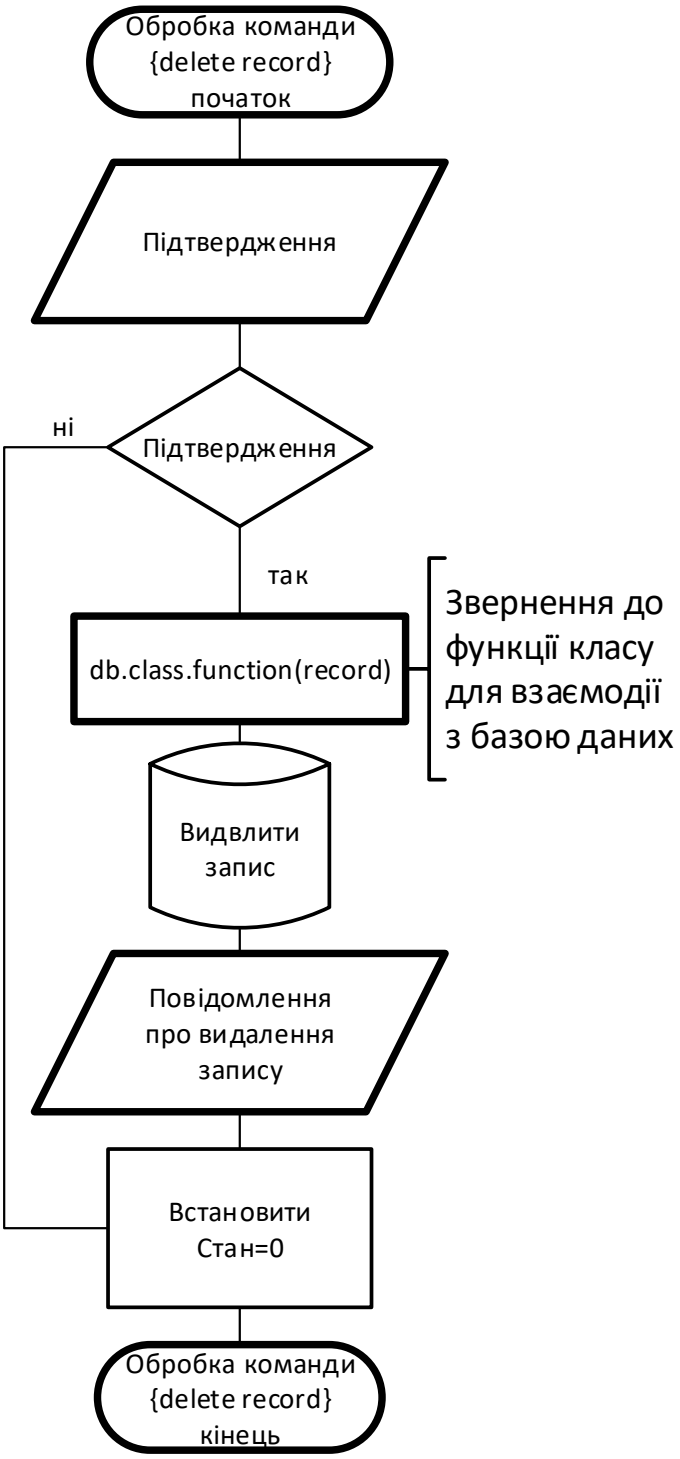


Рис.3.3.5 алгоритм обробки команди “видалити запис”

3.4 Тестування бота

3.4.1 відпрацювання функції /start

Бот відпрацював справно та відправив привітальне повідомлення, та запросив обрати одну з функцій на кнопках клавіатури яку він створив.

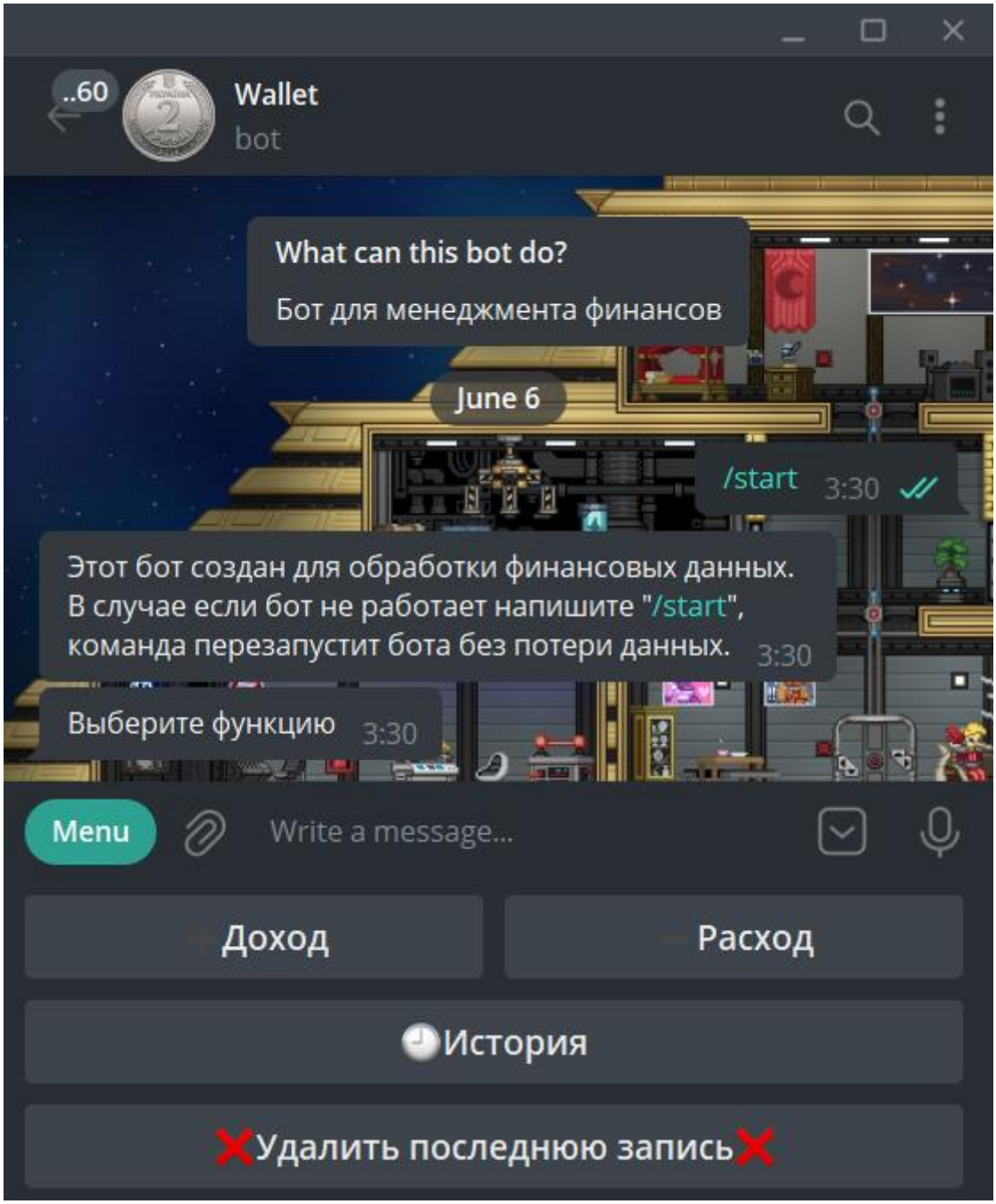


Рис.3.4.1 відпрацювання функції /start

3.4.2 Відпрацювання функції занесення запису

Спочатку було обрано функцію витрати, після чого бот запросив користувача ввести суму, і так як це витрати у боті було додано можливість вибору витрат:

– На будинок
– Продукти
– Транспорт
– Покупки
– Розваги

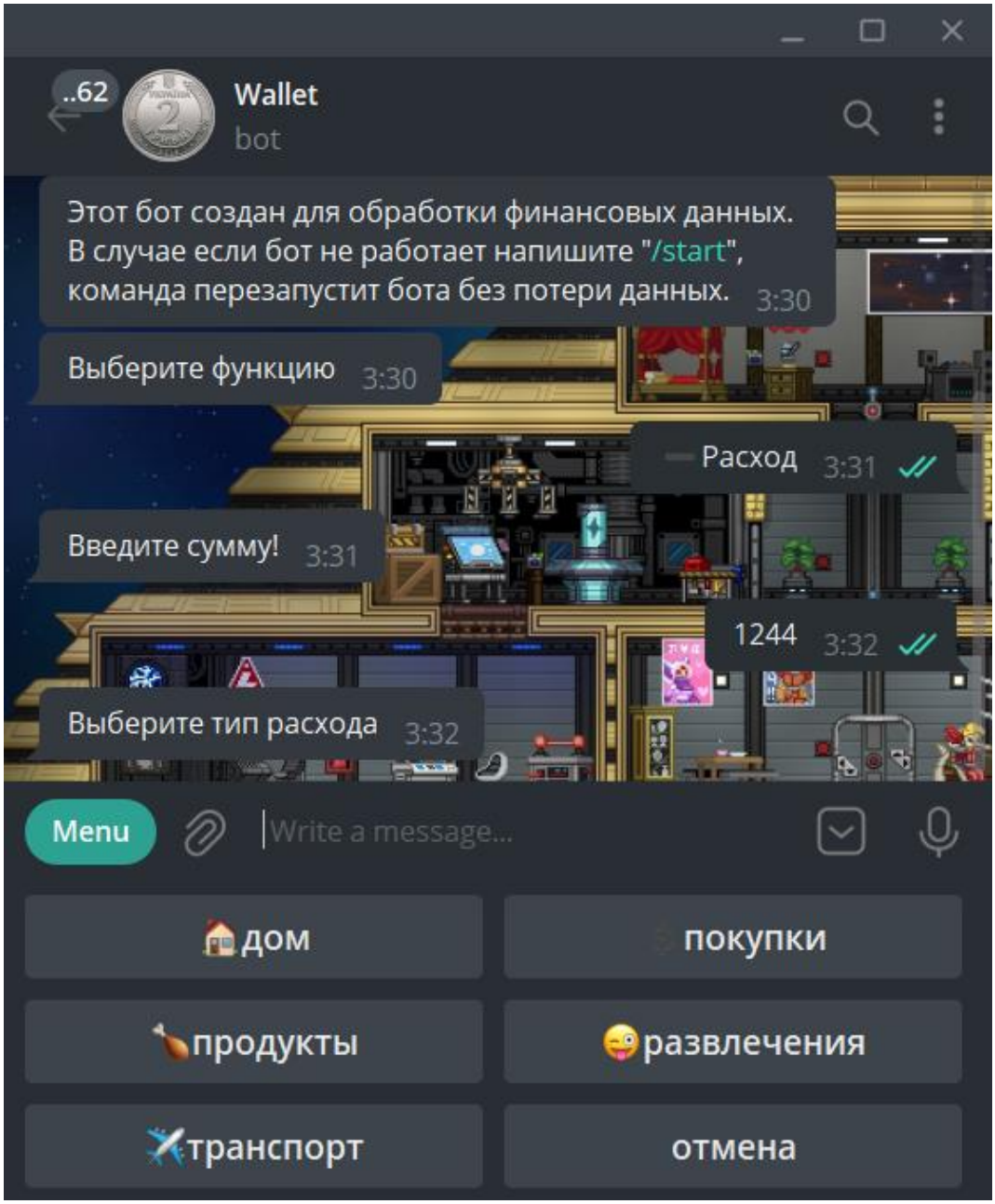


Рис.3.4.2 Відпрацювання функції занесення запису про витрати

Бот обробив запит користувача та відповів про успішно виконану операцію

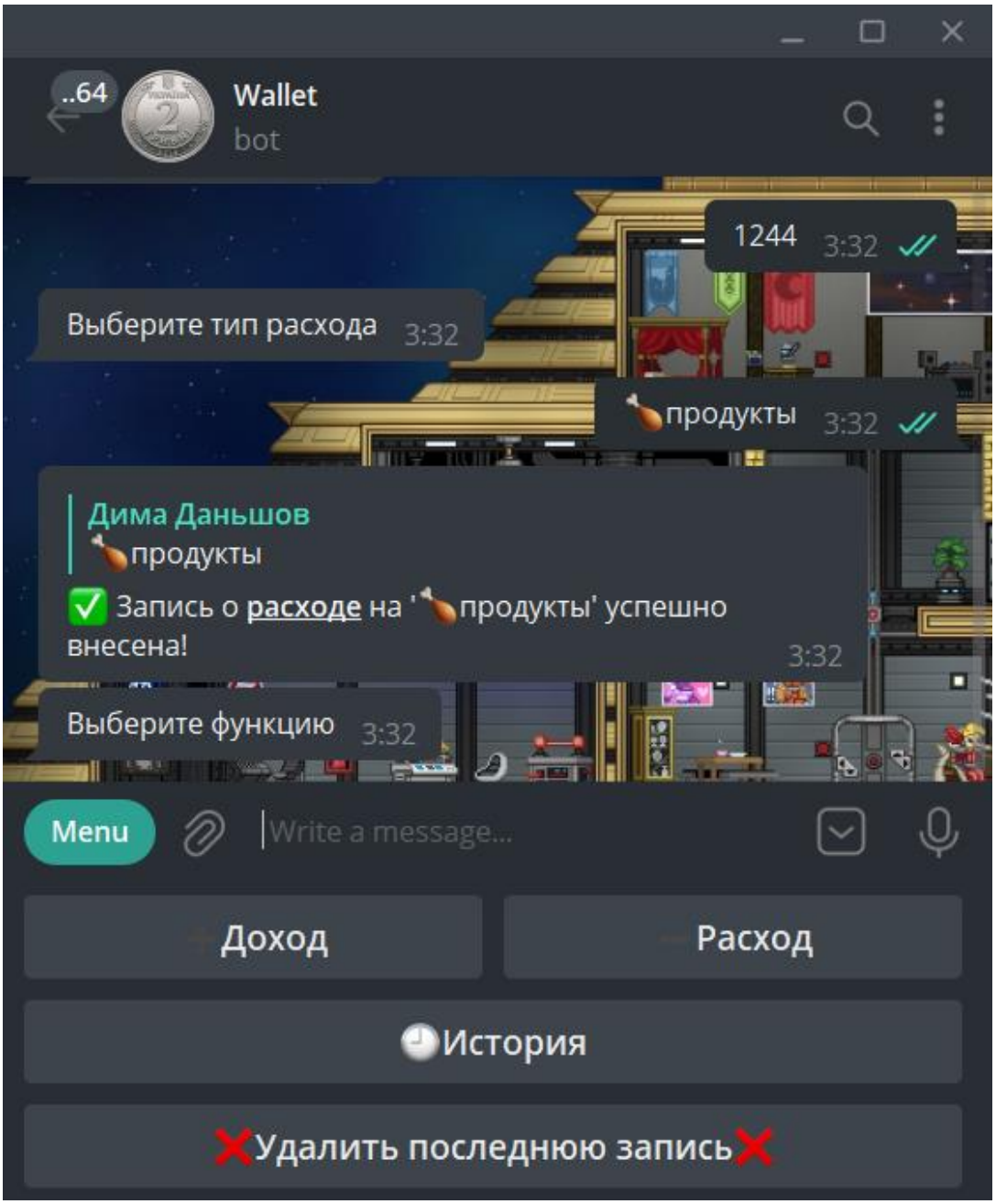


Рис.3.4.2 Відпрацювання функції занесення запису про витрати(відповідь бота)

3.4.3 Відпрацювання функції виводу історії операцій

Користувач обравши цю функцію отримає інформацію про витрати та прибуток за обраний період часу. Було обрано період за день, і так як було занесено лише один запис за цей період, бот її і відобразив користувачу.

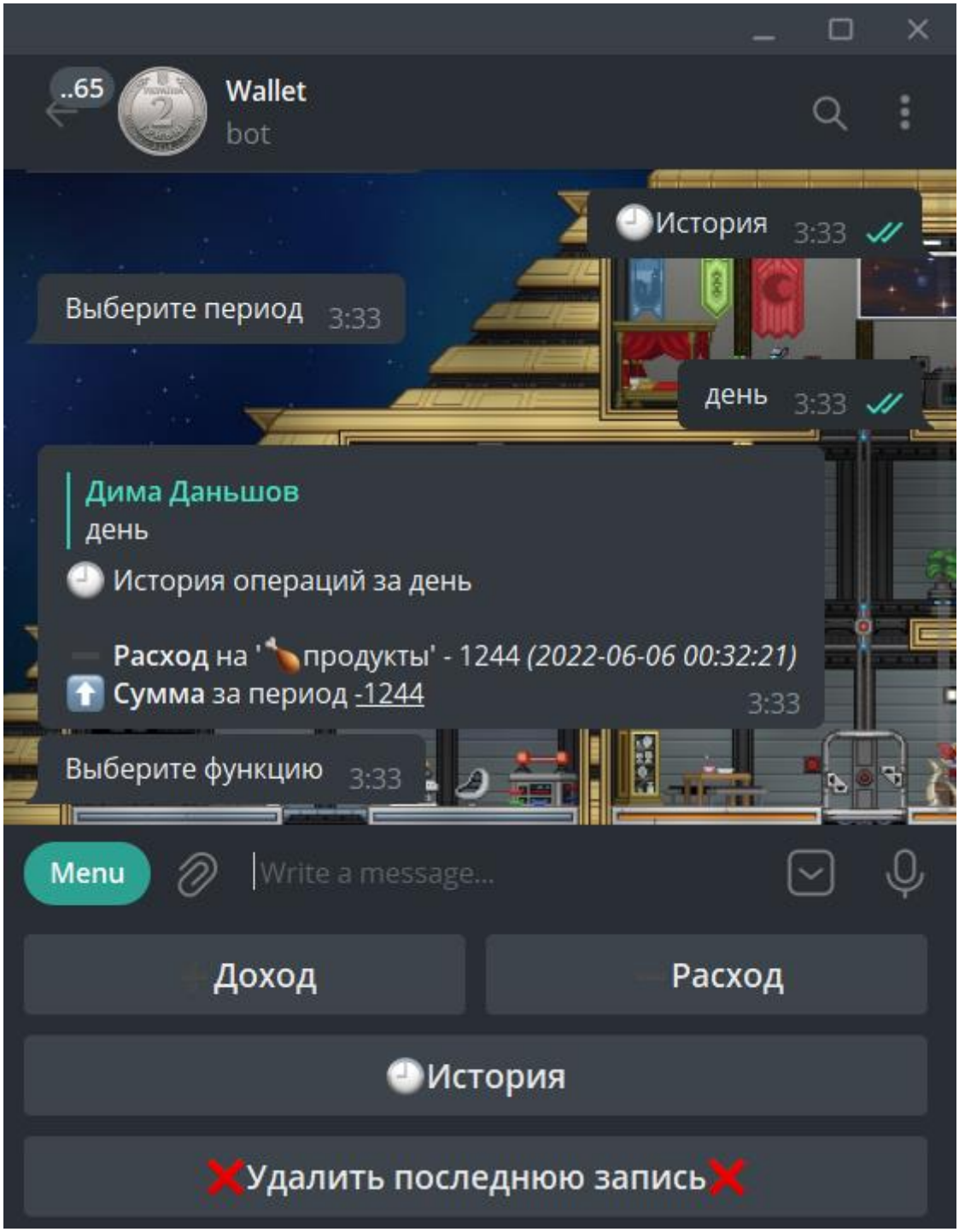


Рис.3.4.3 Відпрацювання функції виводу історії операцій за день

3.4.4 Відпрацювання функції видалення запису

Обравши цю функцію буде видалено останній запис що занесено у базу даних фінансових операцій користувача. Перед видаленням бот запитає підтвердження у користувача.

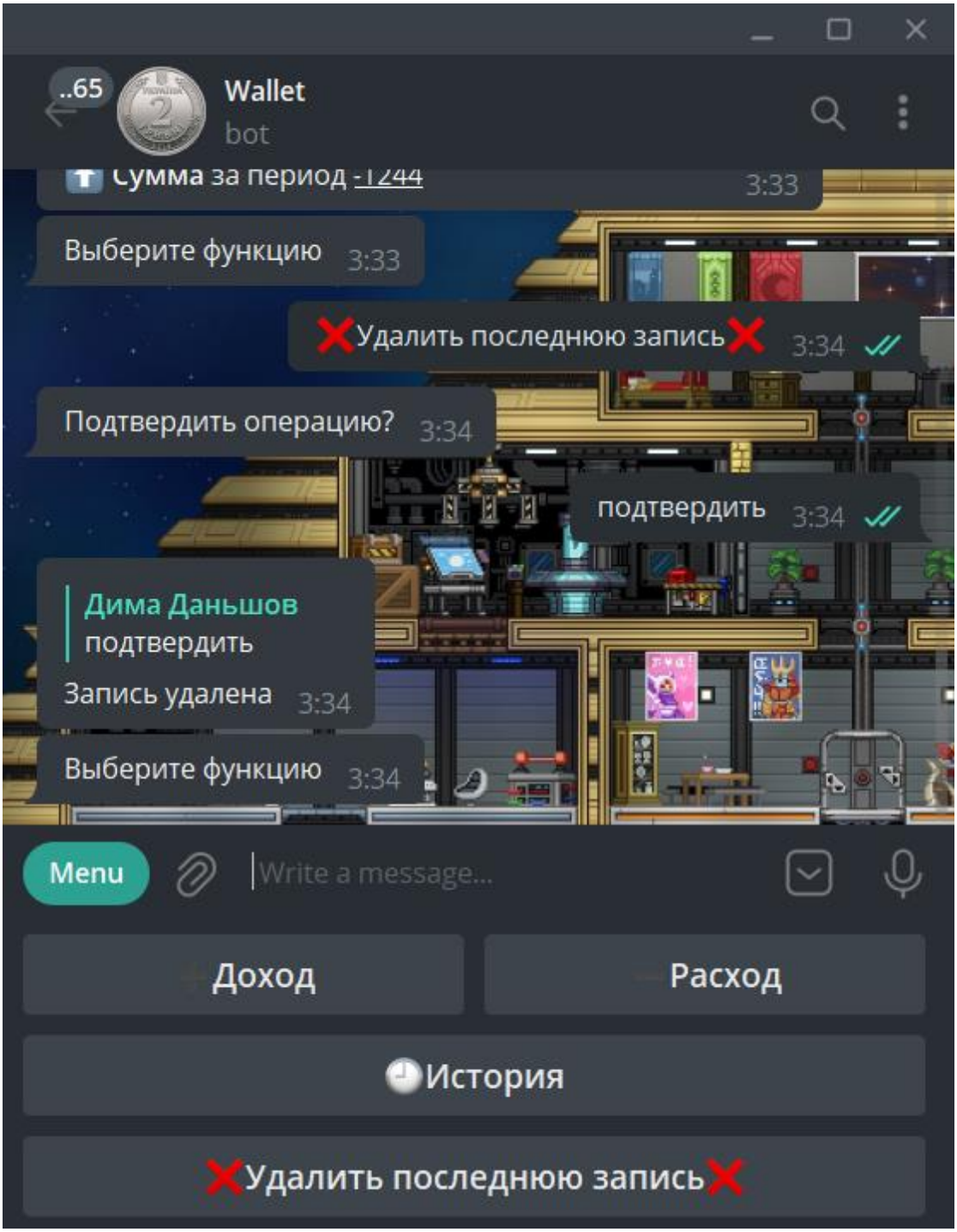


Рис.3.4.4 успішне видалення запису

Після видалення обравши функцію “історія” бот нам відповість що не знайдено даних фінансових операцій так як у базі був лише один запис який і було видалено. Бот відпрацював справно та сповістив користувача про відсутність записів у базі.

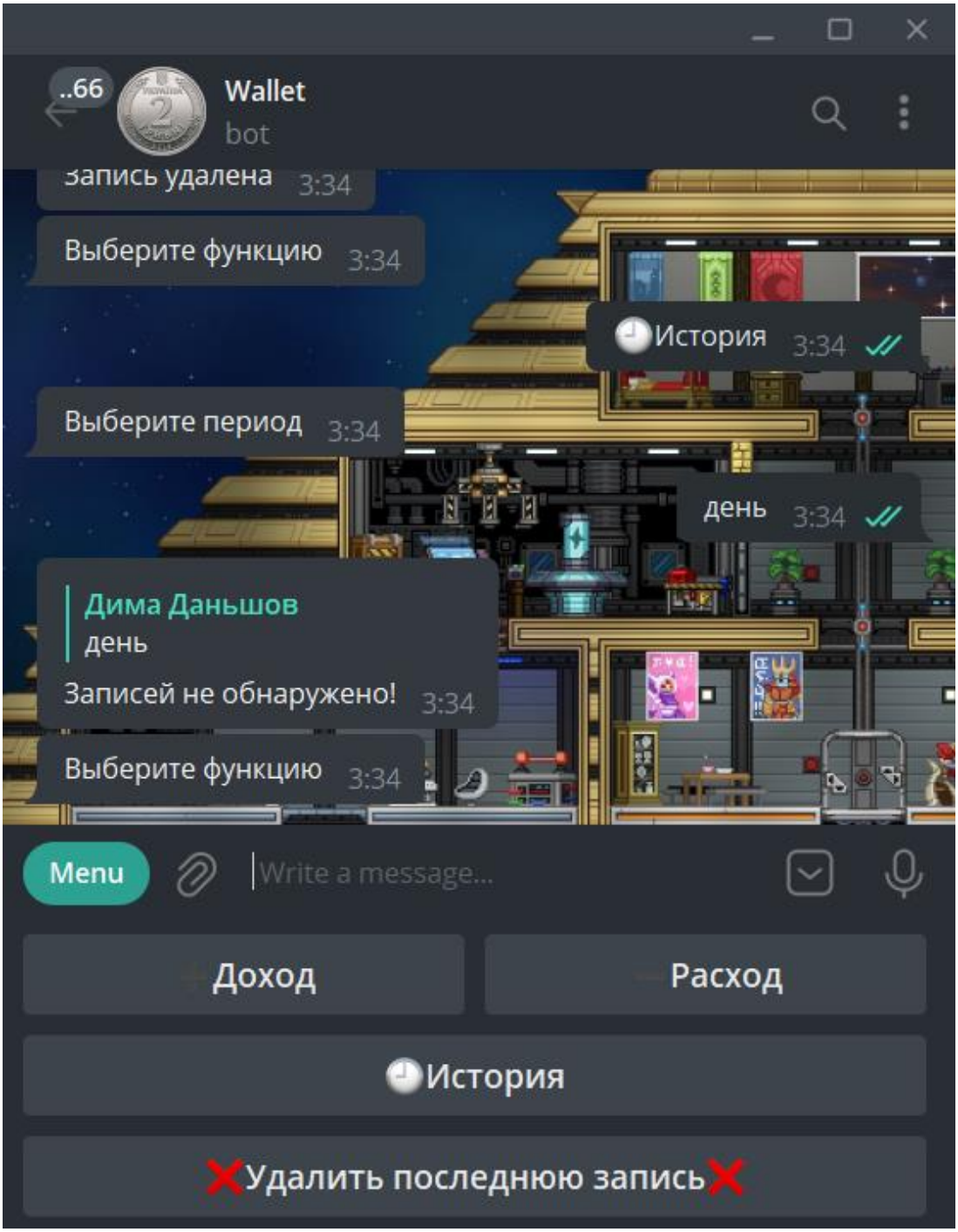
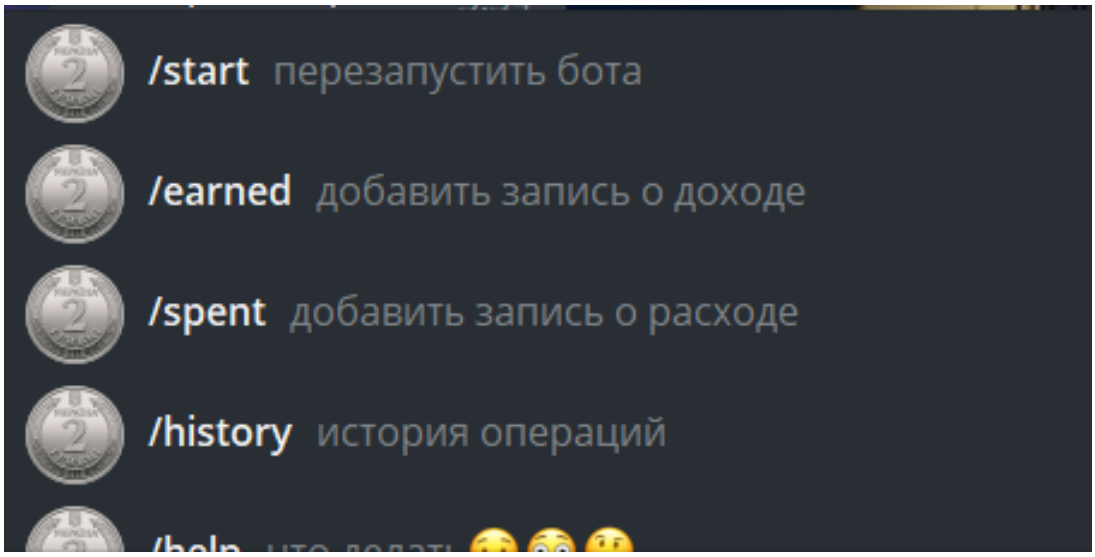


Рис.3.4.4 перевірка справного видалення запису

3.4.5 Команди якими може користуватися юзер

У меню бота описаний список команд та їх функціонал. Користувач може користуватися ними як альтернатива кнопкам інтерфейсу бота.



3.5 Висновки до розділу

Спроектовано та запрограмовано функціонального бота. При тестуванні помилок не виявлено, дані заносяться та зчитуються з бази даних коректно.

ВИСНОВКИ

В ході виконання дипломного проекту здобуто практичні навички по створенню бота для додатку Телеграм. Цей продукт доведений до більш презентабельного вигляду та з покращеним функціоналом може бути конкурентоспроможним на ринку ботів.

Підсумок дипломного проекту:

1) Розроблено схему алгоритму

2) Проведено програмну реалізацію алгоритму

3) Проведено тестування програми бота

В процесі створено базу даних у яку заносяться дані користувачів, та здобуто навички роботи з нею, реалізовано зв'язок бази даних з програмним кодом бота.

Зареєстровано бота у додатку Телеграм, де він буде справно функціонувати через унікальний токен зв'язку, та буде обробляти запити декількох користувачів.

Створено інтерфейс бота для комунікації програми з користувачем. Зв'язок з користувачем може бути проведений через текстове поле чату бота з використанням команд, або через натискання кнопок клавіатури бота.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Найханова, Л.В. Методы и алгоритмы трансляции естественно-языковых запросов к базе данных в *SQL*-запросы [Текст]: Монография / Л. В. Найханова, И. С. Евдокимова. – Улан-Удэ: Изд-во ВСГТУ, 2006. – 148 с.
2. Кубенский, А.А. Создание и обработка структур данных в примерах на *Java*. Серия "Мастер" [Текст] / А. А. Кубенский. – СПб.: БХВ-Петербург, 2006. – 336 с.
3. Эккель, Б. Философия *Java* [Текст] / Б. Эккель. – СПб.: Питер, 2007. – 976 с. – *ISBN* 5-88782-105-1
4. *SyTech* – разработка программного обеспечения: аналитические системы, электронный документооборот, корпоративные системы, информационные порталы //*http*://*sytech.ru*.
5. Бухараев Р.Г., Сулейманов Д.Ш. Семантический анализ в вопросно-ответных системах. – Казань: Казан, 1990. – 124 с.
6. ДСТУ 3008-95 Документація. Звіти у сфері науки і техніки. Структура і правила оформлення.
7. VS Code https://docs.microsoft.com/ru-ru/learn/modules/python-install-vscode/1-introduction
8. Telegram BOT API https://core.telegram.org/bots/api
9. Aiogram https://docs.aiogram.dev/en/latest/index.html
10. SQLite https://www.sqlitetutorial.net/download-install-sqlite/
11. VS Code wiki https://uk.wikipedia.org/wiki/Visual_Studio_Code

Додаток А

Лістинг вихідного коду

```
import telebot
from dispatcher import dp
import config
import re
from bot import BotDB
from telebot import types

import logging

from aiogram import Bot, types
from utils import TestStates
from messages import MESSAGES

logging.basicConfig(format=u'%(filename)+13s [ LINE:%(lineno)-4s] %(levelname)-8s
[%(asctime)s] %(message)s',
                    level=logging.DEBUG)

bot = Bot(token=config.BOT_TOKEN)

StartKeybord = types.ReplyKeyboardMarkup(resize_keyboard=True,one_time_keyboard=True)
item1 = types.KeyboardButton("➕Доход")
item2 = types.KeyboardButton("➖Расход")
item3 = types.KeyboardButton("❌Удалить последнюю запись❌")
item4 = types.KeyboardButton("🕘История")
StartKeybord.add(item1,item2).add(item4).add(item3)

within_als = {
    "day": ('today', 'day', 'сегодня', 'день'),
    "week": ('week','неделю'),
    "month": ('month', 'месяц'),
    "year": ('year', 'год'),
    "all": ('all','все время')
}

record_types=["доход","🏠дом","💲покупки","🍗продукты","😁развлечения","✈транспорт"]

async def history(message: types.Message,within):
    records = BotDB.get_records(message.from_user.id, within)
    if(len(records)):
        answer = f"🕘 История операций за {within_als[within][-1]}\n\n"
        sum=0
        for r in records:
            answer += "<b>" + ("➖ Расход" if not r[2] else "➕ Доход") + "</b>"
            answer += f" на '{record_types[r[3]]}'" if not r[2] else ""
            answer += f" - {r[4]}"
            answer += f" <i>({r[5]})</i>\n"
```

```
            if r[2]:
                sum+=r[4]
            else:
                sum-=r[4]
        answer+=f"⬆ <b>Сумма</b> за период <u>{str(sum)}</u>"
        await message.reply(answer)
    else:
        await message.reply("Записей не обнаружено!")

@dp.message_handler(state='*', commands = "start")
async def start(message: types.Message):
    state = dp.current_state(user=message.from_user.id)
    await state.set_state(TestStates.all()[0])
    if(not BotDB.user_exists(message.from_user.id)):
        BotDB.add_user(message.from_user.id)
    await message.reply(MESSAGES['help'],reply=False)
    await message.reply(MESSAGES['start'],reply=False,reply_markup=StartKeybord)

@dp.message_handler(state='*',commands=['help'])
async def process_help_command(message: types.Message):
    await message.reply(MESSAGES['help'])

@dp.message_handler(state='*', commands=['setstate'])
async def process_setstate_command(message: types.Message):
    argument = message.get_args()
    state = dp.current_state(user=message.from_user.id)
    if not argument:
        await state.set_state(TestStates.all()[0])
        return await message.reply(MESSAGES['state_reset'])

    if (not argument.isdigit()) or (not int(argument) < len(TestStates.all())):
        return await message.reply(MESSAGES['invalid_key'].format(key=argument))

    await state.set_state(TestStates.all()[int(argument)])
    await message.reply(MESSAGES['state_change'], reply=False)

@dp.message_handler(state=TestStates.TEST_STATE_0, commands = ("spent", "earned", "s",
"e"), commands_prefix = "/!")
async def start(message: types.Message):
    cmd_variants = (('/spent', '/s', '!spent', '!s'), ('/earned', '/e', '!earned',
'!e'))
    operation = '-' if message.text.startswith(cmd_variants[0]) else '+'
    state = dp.current_state(user=message.from_user.id)
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True,one_time_keyboard=True)
    item1 = types.KeyboardButton("отмена")
    item2 = types.KeyboardButton("🏠дом")
    item3 = types.KeyboardButton("💲покупки")
    item4 = types.KeyboardButton("🍗продукты")
    item5 = types.KeyboardButton("😁развлечения")
    item6 = types.KeyboardButton("✈транспорт")
    value = message.text
    for i in cmd_variants:
```

```
        for j in i:
            value = value.replace(j, '').strip()

    if(len(value)):
        if value[0]=='-':
            operation='-'
        elif value[0]=='+':
            operation='+'
        x = re.findall(r"\d+(?:.\d+)?", value)
        if(len(x)):
            value = float(x[0].replace(',', '.'))
            if(operation == '-'):
                markup.add(item2,item3).add(item4,item5).add(item6,item1)
                BotDB.add_record(message.from_user.id, operation, value)
                await state.set_state(TestStates.all()[4])
                await message.reply("Выберите тип
расхода",reply=False,reply_markup=markup)
            else:
                markup.add(item1)
                await message.reply("✅ Запись о <u><b>доходе</b></u> успешно
внесена!")
                await state.set_state(TestStates.all()[0])
                await
message.reply(MESSAGES['start'],reply=False,reply_markup=StartKeybord)
        else:
            await message.reply("Не удалось определить сумму!")
    else:
        markup.add(item1)
        await message.reply("Введите сумму!",reply=False, reply_markup=markup)
        if operation=='-':
            await state.set_state(TestStates.all()[2])
        elif operation=='+':
            await state.set_state(TestStates.all()[1])

@dp.message_handler(state=TestStates.TEST_STATE_0,commands = ("history", "h"),
commands_prefix = "/!")
async def start(message: types.Message):
    cmd_variants = ('/history', '/h', '!history', '!h')

    cmd = message.text
    for r in cmd_variants:
        cmd = cmd.replace(r, '').strip()

    within = "day"
    if(len(cmd)):
        for k in within_als:
            for als in within_als[k]:
                if(als == cmd):
                    within = k

    await history(message,within)
```

```
@dp.message_handler(state=TestStates.TEST_STATE_0, content_types=['text'])
async def start(message: types.Message):
    state = dp.current_state(user=message.from_user.id)
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True,one_time_keyboard=True)
    item1 = types.KeyboardButton("отмена")
    item2 = types.KeyboardButton("день")
    item3 = types.KeyboardButton("неделя")
    item4 = types.KeyboardButton("месяц")
    item5 = types.KeyboardButton("год")
    item6 = types.KeyboardButton("за все время")
    item7 = types.KeyboardButton("подтвердить")
    if message.text=="➕Доход":
        await state.set_state(TestStates.all()[1])
        markup.add(item1)
        await message.reply("Введите сумму!",reply=False, reply_markup=markup)
    elif message.text=="➖Расход":
        await state.set_state(TestStates.all()[2])
        markup.add(item1)
        await message.reply("Введите сумму!",reply=False, reply_markup=markup)
    elif message.text=="🕘История":
        if(BotDB.records_exists(message.from_user.id)):
            await state.set_state(TestStates.all()[3])
            markup.add(item2,item3).add(item4,item5).add(item6).add(item1)
            await message.reply("Выберите период",reply=False, reply_markup=markup)
        else:
            await message.reply("Записей не обнаружено!")
            await state.set_state(TestStates.all()[0])
            await
message.reply(MESSAGES['start'],reply=False,reply_markup=StartKeybord)
    elif message.text=="❌Удалить последнюю запись❌":
        if(BotDB.records_exists(message.from_user.id)):
            await state.set_state(TestStates.all()[5])
            markup.add(item7,item1)
            await message.reply("Подтвердить операцию?",reply=False,
reply_markup=markup)
        else:
            await message.reply("Записей не обнаружено!")
            await state.set_state(TestStates.all()[0])
            await
message.reply(MESSAGES['start'],reply=False,reply_markup=StartKeybord)
    else:
        await message.reply("Неизвестная команда",reply=False)

@dp.message_handler(state=TestStates.TEST_STATE_1 | TestStates.TEST_STATE_2,
content_types=['text'])
async def start(message: types.Message):
    state = dp.current_state(user=message.from_user.id)
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True,one_time_keyboard=True)
    item1 = types.KeyboardButton("отмена")
    item2 = types.KeyboardButton("🏠дом")
    item3 = types.KeyboardButton("💲покупки")
    item4 = types.KeyboardButton("🔧продукты")
```

```
    item5 = types.KeyboardButton("😁развлечения")
    item6 = types.KeyboardButton("✈транспорт")
    markup.add(item2,item3).add(item4,item5).add(item6,item1)
    if message.text=='отмена':
        await state.set_state(TestStates.all()[0])
        await message.reply(MESSAGES['start'],reply=False,reply_markup=StartKeybord)
    else:
        operation= '-' if message.text[0]=='-' else '+'
        if ((message.text[0]!= '-')==(message.text[0]!= '+')):
            operation= '-' if await state.get_state()=='test_state_2' else '+'
        value = message.text
        if(len(value)):
            x = re.findall(r"\d+(?:.\d+)?", value)
            if(len(x)):
                value = float(x[0].replace(',', '.'))
                if(operation == '-'):
                    BotDB.add_record(message.from_user.id, operation, value)
                    await state.set_state(TestStates.all()[4])
                    await message.reply("Выберите тип
расхода",reply=False,reply_markup=markup)
                else:
                    BotDB.add_record(message.from_user.id, operation, value)
                    await message.reply("✅ Запись о <u><b>доходе</b></u> успешно
внесена!")
                    await state.set_state(TestStates.all()[0])
                    await
message.reply(MESSAGES['start'],reply=False,reply_markup=StartKeybord)
            else:
                await message.reply("Не удалось определить сумму!")
        else:
            await message.reply("Введите сумму!",reply=False)

@dp.message_handler(state=TestStates.TEST_STATE_3, content_types=['text'])
async def start(message: types.Message):
    state = dp.current_state(user=message.from_user.id)
    if message.text=="день":
        await state.set_state(TestStates.all()[0])
        await history(message,"day")
        await message.reply(MESSAGES['start'],reply=False,reply_markup=StartKeybord)
    elif message.text=="неделя":
        await state.set_state(TestStates.all()[0])
        await history(message,"week")
        await message.reply(MESSAGES['start'],reply=False,reply_markup=StartKeybord)
    elif message.text=="месяц":
        await state.set_state(TestStates.all()[0])
        await history(message,"month")
        await message.reply(MESSAGES['start'],reply=False,reply_markup=StartKeybord)
    elif message.text=="год":
        await state.set_state(TestStates.all()[0])
        await history(message,"year")
        await message.reply(MESSAGES['start'],reply=False,reply_markup=StartKeybord)
    elif message.text=="за все время":
```

```
        await state.set_state(TestStates.all()[0])
        await history(message,"all")
        await message.reply(MESSAGES['start'],reply=False,reply_markup=StartKeybord)
    elif message.text=="отмена":
        await state.set_state(TestStates.all()[0])
        await message.reply(MESSAGES['start'],reply=False,reply_markup=StartKeybord)
    else:
        await message.reply("Неизвестная команда",reply=False)

@dp.message_handler(state=TestStates.TEST_STATE_4, content_types=['text'])
async def start(message: types.Message):
    state = dp.current_state(user=message.from_user.id)
    try:
        if message.text=="отмена":
            BotDB.remove_record(message.from_user.id)
            await state.set_state(TestStates.all()[0])
            await
message.reply(MESSAGES['start'],reply=False,reply_markup=StartKeybord)
        else:
            index=record_types.index(message.text)
            BotDB.change_record_type(message.from_user.id, index)
            await state.set_state(TestStates.all()[0])
            await message.reply("✅ Запись о <u><b>расходе</b></u> на
'"+record_types[index]+"' успешно внесена!")
            await
message.reply(MESSAGES['start'],reply=False,reply_markup=StartKeybord)
    except ValueError:
        await message.reply("Неизвестная команда",reply=False)

@dp.message_handler(state=TestStates.TEST_STATE_5, content_types=['text'])
async def start(message: types.Message):
    state = dp.current_state(user=message.from_user.id)
    if message.text=="подтвердить":
        BotDB.remove_record(message.from_user.id)
        await state.set_state(TestStates.all()[0])
        await message.reply("Запись удалена")
        await message.reply(MESSAGES['start'],reply=False,reply_markup=StartKeybord)
    elif message.text=="отмена":
        await state.set_state(TestStates.all()[0])
        await message.reply(MESSAGES['start'],reply=False,reply_markup=StartKeybord)
    else:
        await message.reply("Неизвестная команда",reply=False)
```