НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

«До захисту допущено»

Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ

“___” червня 2020 р.

Дипломний проект

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп’ютерні системи та компоненти»

зі спеціальності **123 «Комп'ютерна інженерія»**

на тему: Кросплатформенна система взаємодії з віддаленими об`єктами

Виконав: студент IV курсу, групи КВ-62
(шифр групи)

Остапчук Степан Ігорович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доц. каф. СПіСКС к.т.н., доцент Орлова М.М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) __ (підпис)

Рецензент_____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»

Освітньо-професійна програма «Комп’ютерні системи та компоненти»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___» _____20___ р.

ЗАВДАННЯ
на дипломний проєкт студента
Остапчука Степана Ігоровича

1.Тема проекту «Кросплатформенна система взаємодії з віддаленими об`єктами», керівник проекту доц. каф. СПіСКС, к. т. н., доцент Орлова М. М.
затверджені наказом по університету від «__» _____ 2020 р. №_____

2. Термін подання студентом проекту 22.05.2020

3. Вихідні дані до проекту: технічна література на тему система взаємодії з віддаленими об’єктами, документація мови програмування Python та додаткових бібліотек, документація Telegram Bot API.

4. Зміст пояснювальної записки:

1. Аналіз існуючих рішень та обґрунтування теми проекту

2. Розроблення програмного забезпечення

3. Тестування розробленої системи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) презентація; структурна схема взаємодія модулів програми; структурна схема узагальнена робота системи; блок-схема алгоритму оброблення запитів від користувача; блок-схема алгоритму обробки команд; лістинг програми.

6. Консультанти розділів проекту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
нормоконтроль	к. т. н., доцент Ярослав КЛЯТЧЕНКО		

7. Дата видачі завдання

_____

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Видача завдання на дипломне проектування	12.10.2019	
2	Вивчення літератури за тематикою роботи	30.01.2020	
3	Вибір засобів розробки	10.02.2020	
4	Реалізація програми	20.03.2020	
5	Тестування програмного комплексу	02.04.2020	
6	Підготовка пояснювальної записки	10.05.2020	
7	Підготовка графічних матеріалів	19.05.2020	

Студент _____ Степан ОСТАПЧУК

Керівник проекту _____ Марія ОРЛОВА

_____

АННОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (50 с., 24 рис., 17 таблиць, список використаної літератури з 16 найменувань, 3 додатки).

Мета дипломного проєкту полягає в дослідженні та структуризації теоретичних відомостей про системи доступу до віддалених об’єктів та на основі отриманих знань розробленні власної програми для доступу до віддаленого об’єкту.

Для реалізації поставленої мети проведено аналіз існуючих систем доступу до віддалених об'єктів.

В результаті проведеної роботи було створено телеграм бот для доступу до віддаленого серверу.

Результати дипломного проєкту використовуються користувачами для доступу до віддаленого серверу. Користувач має змогу не лише підєднатись до віддаленого об'єкту, а й використовувати часто вживані команди.

КЛЮЧОВІ СЛОВА: БОТ, ЧАТ-БОТ, РОБОТ, ТЕЛЕГРАМ-БОТ, PYTHON, МЕССЕНДЖЕР, TELEGRAM, СЕРВЕР, КЛІЄНТ.

SUMMARY

Qualification work includes an explanatory note (50 pages, 24 figures, 17 tables, list of references from 16 items, 3 appendices).

The purpose of the diploma project is to study and structure the theoretical information about access systems to remote objects and on the basis of the acquired knowledge to develop their own program for access to remote objects.

To achieve this goal, an analysis of existing access systems to remote objects.

As a result of this work, a telegram bot was created to access the remote server.

The results of the thesis project are used by users to access a remote server. The user can not only connect to a remote object, but also use frequently used commands.

KEY WORDS: BOT, CHAT-BOT, ROBOT, TELEGRAM-BOT, PYTHON, MESSENGER, TELEGRAM, SERVER, CLIENT

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.002 ТЗ	Кросплатформенна система	4		
			взаємодії з віддаленими			
			об’єктами			
			Технічне завдання			
	А4	ІАЛЦ.045490.003 ТП	Кросплатформенна система	2		
			взаємодії з віддаленими			
			об’єктами			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ.045490.004 ПЗ	Кросплатформенна система	50		
			взаємодії з віддаленими			
			об’єктами			
			Пояснювальна записка			
	А4	ІАЛЦ.045490.005 Д1	Взаємодія модулів	1		
			програми.			
			Схема структурна			

Змін	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.001 ОА		
Розробив		Остапчук С. І.			Кросплатформена система взаємодії з віддаленими об`єктами **Опис альбому**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Орлова М. М.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-62	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В. О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.006 Д2	Узагальнена робота	1		
			системи.			
			Схема структурна.			
	А4	ІАЛЦ.045490.007 Д3	Алгоритм оброблення	1		
			запитів від			
			Користувача			
			Схема алгоритму.			
	А4	ІАЛЦ.045490.008 Д4	Алгоритм обробки	1		
			команд.			
			Схема алгоритму.			
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			

					ІАЛЦ.045490.001 ОА	Арк.
Змін	Арк.	№ докум.	Підпис	Дата		2

Зміст

					ІАЛЦ.045490.002 ТЗ		
Змін.	Арк.	№ докум.	Підпис	Дата			
Розробив		Остапчук С. І.			Кросплатформенна система взаємодії з віддаленими об’єктами. **Технічне завдання**	Літ.	Аркуш 1 / Аркушів 3
Перевірила		Орлова М. М.					
Консульт.						КПІ ім. Ігоря Сікорського Кафедра СПіСКС Група КВ-62	
Н. контроль		Клятченко Я.М.					
ав. каф.		Романкевич В.О					

1. НАЙМЕНУВАННЯ І ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування роботи – дипломний проект на тему «Кросплатформенна система взаємодії з віддаленими об’єктами».

Область дослідження: взаємодія з віддаленими об’єктами та Телеграм боти.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. ЦІЛЬ І ПРИЗНАЧЕННЯ РОБОТИ

Метою даної роботи є дослідження доступу до віддалених об’єктів та створення бота для доступу до віддалених об’єктів на базі месенджеру Телеграм. Реалізація програмного продукту.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації в періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до до програмного продукту, що розробляється:

- можливість підключення до віддаленого об’єкту;
- доступ для конкретних осіб;
- можливість додавати часто використовуванні команди;
- можливість доступу до командного рядка віддаленого об’єкту.

5.2 Вимоги до апаратного забезпечення:

- Оперативна пам’ять: 1 Гб;
- наявність доступу до мережі Wi-Fi (IEEE 802.11 b/g/n).

5.3 Вимоги до програмного та апаратного забезпечення користувача:

- Операційна система, яка підтримує Telegram.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Видача завдання на дипломне проектування	12.10.2019	
2	Вивчення літератури за тематикою роботи	30.01.2020	
3	Вибір засобів розробки	10.02.2020	
4	Реалізація вибраних алгоритмів	20.03.2020	
5	Розробка графічного інтерфейсу	02.04.2020	
6	Тестування програмного комплексу	10.05.2020	
7	Підготовка пояснювальної записки	19.05.2020	
8	Підготовка графічних матеріалів	23.05.2020	

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045492.004 ПЗ	Кросплатформенна система	50		
			взаємодії з віддаленими			
			об’єктами			
			Пояснювальна записка			
	А4	ІАЛЦ.045490.005 Д1	Взаємодія модулів	1		
			програми.			
			Схема структурна			
	А4	ІАЛЦ.045490.006 Д2	Узагальнена робота	1		
			системи.			
			Схема структурна.			
	А4	ІАЛЦ.045490.007 Д3	Алгоритм оброблення	1		
			запитів від			
			Користувача			
			Схема алгоритму.			
	А4	ІАЛЦ.045490.008 Д4	Алгоритм обробки	1		
			команд.			
			Схема алгоритму.			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.003 ТП		
Розробив		Остапчук С. І.			Кросплатформена система взаємодії з віддаленими об`єктами **Відомість технічного проєкту**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Орлова М. М.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-62	
Н.контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В. О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.003 ТП	Арк. 2

Пояснювальна записка
до дипломного проекту

на тему: Кросплатформенна система взаємодії з віддаленими об`єктами

Київ – 2020 року

ЗМІСТ

ІАЛЦ.045490.004 Взаємодія модулів програми. Схема структурна

ІАЛЦ.045490.004 Узагальнена робота системи. Схема структурна

ІАЛЦ.045490.004 Алгоритм оброблення запитів від користувача. Схема алгоритмічна

ІАЛЦ.045490.004 Алгоритм обробки команд. Схема алгоритмічна

Додаток 2. Лістинг програми.

Додаток 3. Презентація.

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Операційна система (ОС) - це комплекс програм, що виконує керування складовими комп'ютера або віртуальної машини; організовує взаємодію з користувачем і керує обчислювальними процесами.

Персональний комп’ютер (ПК) – електронна машина, що служить для обробки та зберігання інформації.

Програмне забезпечення (ПЗ) – це сукупність програм та документів для роботи цих програм.

Токен – компактний пристрій для забезпечення безпечного з’єднання, ідентифікації користувача або для забезпечення інформаційної безпеки

Чат-бот –розроблене на основі нейромереж програмне забезпечення призначене для спілкування з користувачем .

API - Application Programming Interface (API)

Https - схема URI, що синтаксично ідентична http: схемі, яка зазвичай використовується для доступу до ресурсів Інтернет. Використання https: URL вказує, що протокол HTTP має використовуватися, але з іншим портом за замовчуванням (443) і додатковим шаром шифрування/автентифікації між HTTP і TCP. Ця схема була винайдена у компанії Netscape Communications Corporation для забезпечення автентифікації та шифрування комунікацій і широко використовується в Інтернеті у програмному забезпеченні.[9]

SSH - Secure Shell

ВСТУП

Досить часто відбуваються ситуації, коли доступ до віддаленого об’єкта потрібен тут і зараз. Однак, не завжди підключення по SSH є найбільш зручним способом, тому що під рукою може не виявитися SSH клієнта, адреси сервера або зв'язки «користувач/пароль». Звичайно, є SSH клієнти, які спрощують адміністрування, але вони не дають моментальний доступ до об’єкта.

Тому вирішено реалізувати просту та ефективну систему для підключення до віддаленого об’єкту, яка б працювала на будь якому пристрої, не залежно від операційної системи.

Оскільки доступ потрібно встановити з конкретним сервером на якому знаходяться автоматичні тести. Ще однією вимогою до системи, окрім звичайних функцій SSH клієнта, є спеціальні команди для простішої і швидшої взаємодії.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1. Аналіз існуючих систем підключення до віддалених об’єктів

Розглянемо існуючі рішення.

Мобільний додаток ConnectBot дозволяє з'єднатися з віддаленим сервером через SSH, або Telnet, або навіть до вашого власного пристрою для роботи з консолі. Для цього необхідно ввести ім'я користувача, ім'я хоста, до якого потрібно підключитися, і (необов'язково) використовуваний порт. Параметри підключення зберігаються, так що відкривши додаток наступного разу, можна підключитися швидше. ConnectBot намагається встановити вибране з'єднання, а потім просить ввести пароль. Даний додаток дає можливість підключатись до декількох серверів одночасно і чергувати ці підключення.

Ще одним клієнтом для підключення до віддалених об’єктів є PuTTY. PuTTY - це клієнт для мережевих протоколів SSH, Telnet і Rlogin. Всі ці протоколи використовуються для запуску віддаленого сеансу на комп'ютері, через мережу. PuTTY реалізує клієнтську частину сеансу: ту частину, на якій сеанс відображається, але не ту, на якій він працює. Простий приклад: якщо запущене PuTTY на машині з ОС Windows, і повідомлене їй, наприклад, з'єднатися з машиною Unix. PuTTY відкривається вікно терміналу, і все, що набрано в командному рядку, відправляється прямо на Unix, і все, що віддалений термінал посилає у відповідь, відобразиться на локальному комп’ютері. Таким чином, можна працювати з віддаленим Unix-сервером так, як ніби модуль-ініціатор знаходиться безпосередньо в його консолі.

З початку PuTTY розроблявся для ОС Windows, але згодом був портований на інші ОС, зокрема наявні порти для Unix подібних систем та для Mac OS.

Prompt 2 – SSH клієнт для техніки IOS. Продукт позиціонує себе як легкий в користуванні і паралельно безпечний SSH клієнт. Основні переваги додатку:

1. Доступ до сервера по відбитках пальців
2. Можливість зберігати логін та пароль для швидкого доступу до сервера
3. Можливість створювати кнопки для дій які часто повторюються
4. Генератор приватних ключів
5. Зручний менеджер для керування приватними ключами

Основними недоліками даного продукту є його вартість та не зовсім зручний і зрозумілий інтерфейс.

1.2. Обґрунтування теми проекту та обраних засобів розроблення

Проаналізувавши вже існуючі рішення з'ясувалось, що немає програного забезпечення за допомогою якого можна було б швидко та з будь-якого пристрою підключитись до віддаленого об'єкту. Також розглянуті SSH клієнти не дають змоги настроїти їх на виконання базових команд, які часто виконуються.

Отож прийнято рішення розробити свою систему. Проаналізувавши вимоги до майбутньої системи прийнято рішення реалізувати її у вигляді чат-боту в одному з популярних месенджерів.

Для початку визначимо, що таке чат-бот. Чат-бот – це додаток, який дозволяє користувачам взаємодіяти зі сторонніми сервісами у вигляді всім відомого інтерфейсу чату. Основна мета чат-боту допомогти користувачам. Кожен чат-бот має свої певні функції, тобто якщо боту написати спеціальну команду, яку в свою чергу він розпізнає певним чином, ми отримаєм потрібну нам інформацію. Так можна швидко виконувати різноманітні дії: переводити, коментувати, знаходити, тестувати, шукати, навчати, транслювати, вбудовуватися в інші сервіси і платформи, взаємодіяти з датчиками і речами, підключеними до мережі Інтернет.

Платформою для створення чат-боту обрано месенджер Telegram. Telegram – один з найпопулярніших додатків для обміну повідомленнями. Крім версій для мобільних ОС, є версії для ОС Windows, Linux та MacOS, є також браузерна версія даного месенджера.

Окрім можливості обмінюватись повідомлення Telegram дає можливість створювати ботів, які можуть взаємодіяти з користувачем, реагувати на його повідомлення, команди, а також відправляти користувачу повідомлення-відповідь. При цьому повідомлення, команди і запити, надіслані користувачами, передаються безпосередньо на сервер, а саме до розробників пошукових роботів. Зауважимо, що сервер Telegram є тільки посередником і надає свій API для розробників, та доступний за допомогою HTTPS-інтерфейсу, який пропонує спрощену версію API Telegram, так званий «Bot API». Для того, щоб розпочати взаємодію з ботом, користувач повинен спочатку додати його в свій в чат або в свою групу. Після цього відправляються команди з чату безпосередньо якомусь конкретному боту, використовуючи символ «@» перед його іменем в повідомленні з командою. Більш детальна інформація про боти наведена на офіційному сайті Telegram. В даному проєкті необхідно розглянути тільки можливість додавання бота в чат або групу, після чого він зможе надсилати туди повідомлення при будь-яких відхиленнях аналізованих метрик.

Для початку роботи слід визначити етапи створення нового Телеграм бота. Для створення бота згідно з інструкції на офіційному сайті Телеграм API, необхідно виконати такі настійпні пункти.

а) Потрібно додати головного Телеграм бота до себе в чат, його можна знайти ввівши в пошуку «@BotFather».

б) Наступним кроком починється робота з «@BotFather», для чого надсилається команда «/newbot», далі потрібно ввести ім’я, після чого той запросить коротке ім'я створюваного бота (для посилання швидких команд). Якщо немає співпадінь то він видасть токен ідентифікації, який потрібен при відправці запитів до Телеграм API.

Проаналізувавши варіанти створення чат-бота було виявлено, що для написання цього бота потрібно використати одну з таких мов програмування: PHP, Ruby, Node.JS, або Python. Ці мови найкраще підходять для серверного

програмування. Критеріями для обрання мови є вміння працювати з REST (Representational State Transfer) API (Application Programming Interface), в даному випадку з TelegramBotAPI. З вище перелічених мов був обраний Python. І його модуль TeleBot для взаємодії з TelegramBotAPI.

Типи ботів:

а)боти, які розуміють природну мову. Зазвичай на основі нейронної мережі яка логічно та структурно відповідає на повідомлення користувача;

б) заскриптовані боти – це боти, котрі не розуміють природної мови а відповідають лише на певний перелік команд. Діалоги з такими ботами зазвичай чіткі, лаконічні та структуровані.

Для нашого випадку краще підходить заскриптований бот.

Висновки до розділу

Після огляду можливих рішення завдання, вирішено зробити чат-бот для Telegram. Для розробки бота буде використовуватись мова Pyhon v3.6, а також будуть використанні окремі її модулі CherryPy, який відповідає за обробку запитів TeleBot для роботи з телеграм API та Subproces для роботи з сервером.

Для того, щоб написати програму чат-бота, потрібно ознайомитись з Telegram Bot API, мовою програмування Python, та способами підключення до віддаленого серверу.

2. РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Реєстрація чат-бота в месендежрі Telegram

Щоб розпочати розробку бота потрібно у месенджері Telegram знайти спеціального чат-бота «BotFather». Для початку реєстрації потрібно ввести команду «/newbot». Після чого «BotFather» запропонує ввести назву чат-бота обов'язковою умовою: будь-який чат-бот має закінчуватись на «bot», або «_bot»(регістр не має значення). Якщо дотримано всіх вимог, то «BotFather» видає спеціальний ключ(токен) . Токен – спеціальний набір символів для доступу до Telegram Bot API за допомогою звичайного HTTPS інтерфейсу. Всі ці дії зображенні на рисунку 2.1.

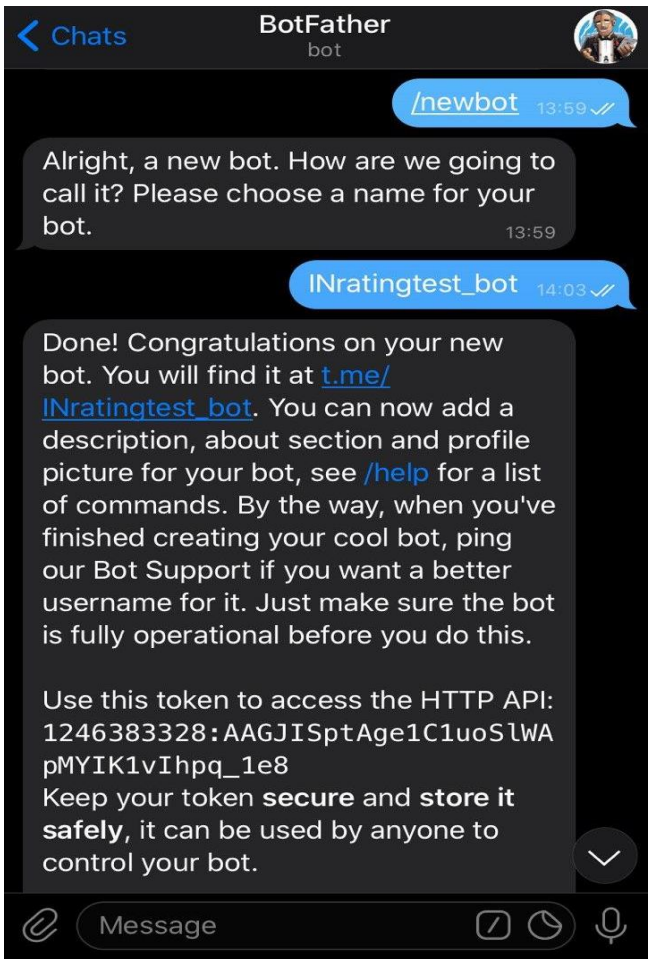

Рисунок 2.1 – Приклад створення нового боту

Після створення чат-боту «BotFather» дає можливість встановити додаткові параметри за допомогою спеціальних команд (таблиця 2.1).

Таблиця 2.1 - Команди для зміни парамерів чат-бота

Команда	Опис
/setname	Заміна імені чат-бота
/setdescription	Встановлює текст, який буде показаний користувачу перед початком роботи з чат-ботом
/setabouttex	Встановлює текст, який відображається в полі «Про чат-бот»
/setuserpic	Дає можливість встановити аватар чат-бота
/setcommands	Дає можливість створити список доступних команд чат-бота
/deletebot	Видаляє чат-бота

Окрім можливості змінити основні параметри, є команди для додаткового налаштування чат-бота, які наведенні в таблиці 2.2.

Таблиця 2.2 – Команди для додаткової настройки чат-боту

Команда	Опис
/token	Повертає ключ(токен) боту
/revoke	Анулює ключ(токен) боту
/setinline	Дозволяє ввімкнути або вимкнути виклик бота з інших чатів
/setinlinefeedback	Дає можливість отримати інформацію про кількість обраних користувачем програм
/setjoingroup	Показує чи може бот бути доданий в груповий чат

Після завершення налаштувань чат-боту за допомогою «BotFather» і отриманні токена приступимо до розробки програмної частини бота.

2.2 Розміщення чат боту на віддаленому сервері

Однією з проблем, яку потрібно вирішити, це цілодобова робота бота. Щоб він працював, потрібно не вимикати машину, на якій він розміщений. Домашній ПК не підходить, тому вирішено розмістити його на віддаленому сервері. Якщо розмістити його на сервері, до якого потрібен доступ, це спростить реалізацію даного завдання.

Для розгортання чат-бота на сервері була використана програма контролю версій git. Git є однією з найбільш ефективних, надійних і високопродуктивних систем керування версіями і надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок. При цьому для забезпечення цілісності та стійкості до змін використовуються криптографічні методи. Крім того, можлива прив'язка цифрових підписів розробників до тегів і комітів. Це зручно, адже можна допрацьовувати бота локально на комп’ютері і з допомогою git оновлювати код на сервері.[7]

Етапи розгортання програми на сервері є наступними.

1. Реєстрація в git.
2. git add – додаємо потрібні нам файли.
3. git commit - комітимо всі файли, які додали попередньою командою.
4. git push – пушимо всі файли, в сховище git.
5. На віддаленому сервері залогінюємось під користувачем, котрий має доступ до проєкту на git.
6. За допомогою команди git pull викачуємо проєкт.

Після того як проєкт розгорнуто на сервері, встановлюємо там потрібну версію Python. Скачуємо потрібні бібліотеки. Для запуску бота використаємо утиліту screen. Для встановлення даної утиліти потрібно в терміналі ввести

команду «sudo apt-get install screen». Після становлення утиліти залишається запустити програму «screen -dmS ServerBot python3 bot.py»

2.3 Мова програмування Python

Python - це динамічна програма програмування високого рівня, інтерпретована та загальноприйнята, що фокусується на читанні коду. Синтаксис в Python допомагає програмістам робити кодування за кілька кроків порівняно з Java або C ++. Мова, заснована в 1991 році розробником Гвідо Ван Россумом, з її використанням програмується легко та цікаво.

Python широко використовується у великих організаціях через її багаторазові парадигми програмування. Зазвичай вони передбачають імперативне та об'єктно-орієнтоване функціональне програмування. Вона має комплексну та велику стандартну бібліотеку, яка має автоматичне управління пам'яттю та динамічні функції.[3]

Компанії, що розробляють програмне забезпечення, віддають перевагу мові Python через її універсальні можливості та меншу кількість програмних кодів. Майже 14% програмістів використовують її в таких операційних системах, як UNIX, Linux, Windows та Mac OS. Програмісти великих компаній використовують Python, оскільки вона створила собі позначку в розробці програмного забезпечення з характерними функціями, такими як:

- інтерактивність;
- інтерпретованість;
- модульність;
- динамічність;
- об’єктно-орієнтованість;
- портативність;
- високорівневість;
- можливість розширення на C ++ та C.

Якщо говорити простою мовою, Python дає можливість написати практично все: веб/настільні додатки, ігри, автоматизовані інформаційні системи, комплексні системи, системи управління життєзабезпеченням і багато іншого. Крім того, поріг входження низький, а код багато в чому небагатослівний і зрозумілий навіть для того, хто ніколи на ньому не писав. За рахунок простоти коду, майбутній супровід програм, які були написані на Python, стає легшим і доступнішим по відношенню до Java або C. А з точки зору бізнесу це дає скорочення витрат і підвищення ефективності трудових ресурсів.[3]

Також, важлива риса – розширюваність мови, цьому надається велике значення і, як пише сам автор Гвідо ван Россум, мова була задумана саме як розширювана. Це означає, що є можливість вдосконалення мови усіма зацікавленими розробниками. Інтерпретатор був розроблений на мові програмування C і вихідний код доступний для будь-яких змін.[3]

Наступна значна перевага – наявність великого числа модулів, які можна підключити, що забезпечують різні додаткові можливості. Такі модулі зазвичай пишуться на C, на самому Python і можуть бути створені більш досвідченими програмістами. Як приклад можна навести такі модулі:

а) Numerical Python – надає програмісту більш розширені математичні можливості, такі як маніпуляції з цілими векторами і матрицями;

б) Tkinter – дає змогу будувати додатки з використанням графічного інтерфейсу користувача (GUI);

в) OpenGL – це велика бібліотека графічного моделювання дво- та тривимірних об'єктів Open Graphics Library.

Єдиним недоліком, на який звертав увагу сам автор, є порівняно невисока швидкість виконання Python програми. Однак, це не відграє велику роль у порівнянні з перевагами мови при написанні програм не дуже критичних до швидкості виконання. З використаних модулів можна виділити: TeleBot, CherryPy, Subprocess, які більш детально будуть описані нижче.[3]

2.3.1 Модуль TeleBot

Модуль telebot – це проста, але розширювана реалізація Python для API Telegram Bot. Він використовується для мінімізації та спрощення коду програми. Існує два способи встановлення.

1. Встановлення за допомогою pip (менеджер пакунків Python):
   pip install pyTelegramBotAPI
2. Встановлення з джерела (цей метод потребує встановлений git):
   - git clone
   - cd pyTelegramBotAPI
   - python setup.py install

Клас TeleBot визначений у _init_.py і включає в собі всі методи Telegram API в одному класі. Щоб відповідати всім стандартам Python, методи перейменовані. Наприклад: sendMessage - send_message, editMessageText -edit_message_text. Функція, прикрашена декоратором примірника TeleBot – обробник повідомлень. Обробники повідомлень складаються з одного або декількох фільтрів. Кожен фільтр повертає True або False для певного повідомлення і обробник отримує дозвіл на обробку повідомлення, якщо повертається True. Приклад обробника повідомлень зображено на рисунку 2.2. Цей обробник спрацьовує на команду «/start».

Рисунок 2.2 – Код обробника команд

Будь-яке ім'я функції дозволено в обробниках повідомлень. Функція повинна приймати щонайменше один аргумент, який буде повідомленням, яке функція повинна обробляти. Фільтр оголошуються в такий спосіб – ім’я = аргумент (в прикладі на рисунку 2.1 commands=[‘start’]). Один обробник може

мати кілька фільтрів. TeleBot підтримує фільтри наведенні в таблиці 2.3.

Таблиця 2.3 – Фільтри обробника в модулі TeleBot.

Назва	Аргументи	Опис
content_types	Список рядків (за замовчуванням [‘text’])	True, якщо message.content_type знаходтться у списку
regexp	Регулярний вираз у вигляді рядка	True якщо re.search повертає True
commands	Список рядків	True, якщо message.content_type == ‘text’ і message.text починається з команди, яка знаходиться у списку
func	функція(лямбда або посилання на функцію)	True, якщо лямбда або функція повертають True

2.3.2 Модуль Subprocess

Модуль subprocess дозволяє створювати нові процеси. При цьому він може підключатися до стандартних потоків введення/виведення/помилок і отримувати код повернення.

За допомогою subprocess можна, наприклад, виконувати будь-які команди Linux з скрипта. І в залежності від ситуації отримувати висновок або тільки перевіряти, що команда виконалася без помилок.

Функція subproces.run(‘команда’) – основний спосіб роботи з модулем subprocess. Subproces.run() - повертає спеціальний об’єкт CompletedProcess, з цього об’єкту можна отримати інформацію про виконання процесу. Ще однією особливістю функції є те, що вона очікує на завершення виконання команди.

Ще однією корисною функцією модуля Subprocess є функція chekck_output(). Вона виконує команду і повертає результат її виконання, якщо код

повернення є не нульовим. Функції run() і check_output() можуть приймати однакові аргументи, найважливіші з яких показані в таблиці 2.4.

Таблиця 2.4 – Аргументи для функцій

Команда	Опис
cmd	Команда для породження дочірнього процесу
timeout	Час очікування в секундах
output	Вихід з дочірнього процесу, якщо він був захоплений
stdout	Псевдонім для виводу, для симетрії з stderr
stderr	Екстрений вивід дочірнього процесу якщо він був захоплений

Функція check_output() використовується в реалізації даного чат боту. Функція main приймає текстове повідомлення. Текстове повідомлення відправляється на віддалений сервер за допомогою check_output(), якщо у відповідь від сервера приходить нормальна відповідь, то ця відповідь у вигляді текстового повідомлення надходить користувачу. У випадку негативної відповіді від сервера користувачу надсилається повідомлення «Invalid input. Please enter terminal command». Даний алгоритм написаний мовою Python і зображений на рисунку 2.3.

```
@bot.message_handler(content_types=["text"])
def main(message):
    if user_id == message.chat.id:
        comand = message.text
        try:
            bot.send_message(message.chat.id, check_output(comand, shell=True))
        except:
            bot.send_message(message.chat.id, "Invalid input. Please enter terminal command")
```

Рисунок 2.3 – Алгоритм відправки команд на сервер

2.4 Rest API

REST (RESTful) - це загальні принципи організації взаємодії додатка сайту з сервером за допомогою протоколу HTTP. Особливість REST в тому, що сервер не запам'ятовує стан користувача між запитами - в кожному запиті передається інформація, що ідентифікує користувача (наприклад, token, отриманий через OAuth-авторизацію) і всі параметри, необхідні для виконання операції.

REST визначає 6 архітектурних обмежень, які роблять будь-який веб-сервіс - справжнім RESTful AP:

Клієнт-сервер - це по суті означає, що клієнтська програма та серверне додаток обов’язково мати можливість розвиватися окремо, не залежно одне від одного. Клієнт повинен знати лише URI-ресурси ресурсів, і це все. Сьогодні це звичайна практика в розробці веб-сторінок, тому нічого фантазійного з вашого боку не потрібно. Не ускладнювати.

Уніфікований інтерфейс - вирішує інтерфейс API для ресурсів всередині системи, які піддаються споживачам API та дотримуються їх релігійно. Ресурс у системі повинен мати лише один логічний URI, який повинен забезпечувати спосіб отримання пов'язаних або додаткових даних. Завжди краще синонімізувати ресурс із веб-сторінкою. Будь-який єдиний ресурс не повинен бути занадто великим і містити кожен і все у своєму поданні. Також представлення ресурсів у всій системі повинно дотримуватися конкретних вказівок, таких як конвенції іменування, формати посилань або формат даних (XML або / та JSON).Усі ресурси повинні бути доступними через загальний підхід, такий як HTTP GET та аналогічно модифіковані за допомогою послідовного підходу.

Без громадянства - Рой Філдінг отримав натхнення від HTTP, тому він відображається в цьому обмеженні. Зробіть усі взаємодії клієнт-сервер без стану. Сервер нічого не зберігатиме про останній запит HTTP, який зробив клієнт. Він розгляне кожен запит як новий. Ні сесії, ні історії. Якщо клієнтській

програмі потрібно стати додатковою заявою для кінцевого користувача, де користувач входить в систему одночасно та виконує інші санкціоновані операції після цього, то кожен запит від клієнта повинен містити всю інформацію, необхідну для обслуговування запиту - включаючи автентифікацію та авторизацію деталі.

Кешований - У сучасному світі кешування даних та відповідей є надзвичайно важливим, де б вони не застосовувалися / були можливі. Веб-сторінка, яку ви читаєте тут, також є кешованою версією HTML-сторінки. Кешування приносить поліпшення продуктивності для клієнта та покращує масштабність для сервера, оскільки навантаження зменшилось.

Багатошарова система - REST дозволяє використовувати шарувату системну архітектуру, де ви розгортаєте API на сервері A, а також зберігаєте дані на сервері B та аутентифікуєте запити, наприклад, на сервері C. Клієнт не може звичайно сказати, підключений він безпосередньо до кінцевого сервера чи до посередника по шляху.

Код на вимогу (необов'язково) - це обмеження необов'язкове. Більшу частину часу ви будете надсилати статичні уявлення про ресурси у вигляді XML або JSON. Але коли вам потрібно, ви return executable codeможете підтримати частину вашої програми, наприклад, клієнти можуть зателефонувати на ваш API, щоб отримати код відтворення інтерфейсу інтерфейсу користувача. Це дозволено REST дозволяє розширити функціональність клієнта, завантаживши та виконавши код у вигляді аплетів чи сценаріїв. Це спрощує клієнтів, зменшуючи кількість функцій, необхідних для попередньої реалізації.

Ще одна важлива річ, пов'язана з REST, - це ресурсні методи, які використовуються для виконання бажаного переходу. Велика кількість людей неправильно ставляться до ресурсних методів до методів HTTP GET / PUT / POST / DELETE .

HTTP метод GET використовується для отримання (або читання) уявлення ресурсу. У разі "вдалого" (або не містять помилок) адреси, GET повертається уявлення ресурсу у форматі XML або JSON в поєднанні з кодом стану HTTP 200 (OK). У разі наявності помилок зазвичай повертається код 404 (NOT FOUND) або 400 (BAD REQUEST).[9]

Метод PUT зазвичай використовується для надання можливості поновлення ресурсу. Тіло запиту при відправленні PUT-запиту до існуючого ресурсу URI має містити оновлені дані оригінального ресурсу.[9]

POST запит найбільш часто використовується для створення нових ресурсів. На практиці він використовується для створення вкладених ресурсів. Іншими словами, при створенні нового ресурсу, POST запит відправляється до батьківського ресурсу і, таким чином, сервіс бере на себе відповідальність на встановлення зв'язку створюваного ресурсу з батьківським ресурсом, призначення нового ресурсу ID.[9]

DELETE запит вкрай простий для розуміння. Він використовується для видалення ресурсу, ідентифікованого конкретним URI (ID).[9]

На рисунку 2.4 зображена узагальнена робота REST API.

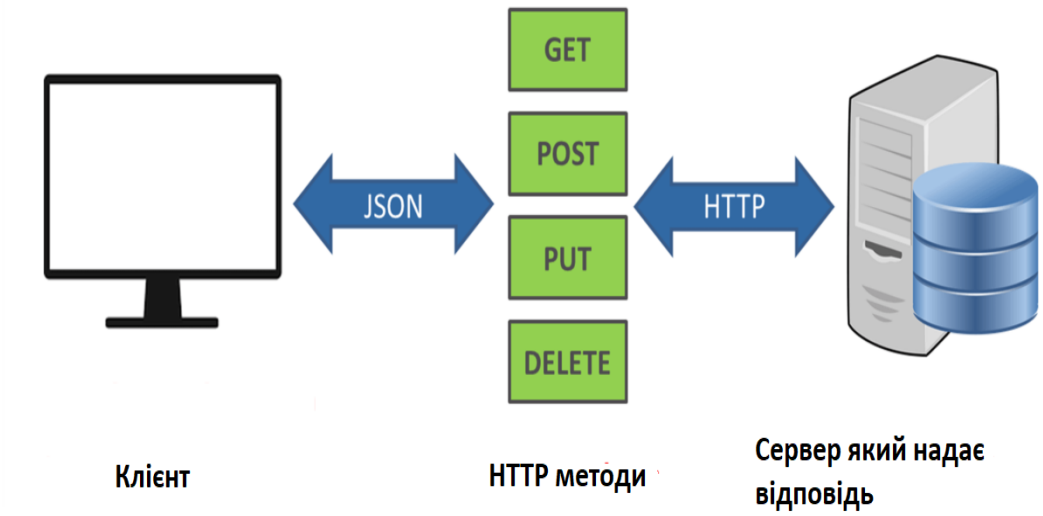

Рисунок 2.4 – Робота REST API

2.5 Telegram Bot API

Telegram Bot API - HTTP-інтерфейс для роботи з ботами в Telegram. Бот – спеціально створений акаунт в мережі Телеграм. Використовується для відправки, оброблення та приймання повідомлень .

В документації Telegram Bot API є два протилежних способи отримання оновлень:

а) періодичні запити на оновлення;

б) установка веб-хуків.

Оновлення, які приходять зберігаються до моменту, доки не будуть обробленні на сервері, проте оновлення не зберігаються більше 24 годин. У разі коректного оброблення оновлень отримуємо об'єкт Update, який приходить у вигляді JSON, метод оновлення не впливає на результат оброблення оновлень.

Перший метод є найбільш простим. Він періодично опитує сервер Telegram на наявність оновлень. Якщо на сервері Телеграм є оновлення, встановлюється з'єднання на короткий період часу. Доки встановлене з'єднання, всі оновлення відразу відправляються боту, все це здійснюється через зв'язок Long Polling. Перевагою цього методу є простота, проте цей метод є дуже не надійним.

Другий метод кардинально відрізняється від першого. Коли повідомлення від користувача надходить на сервер, то Телеграм сам одразу сповіщає про це. Відпадає необхідність опитувати сервери, завдяки цьому, зникає причина помилок пошукових ботів. Проте є і мінус для установки і нормальної роботи таких ботів потрібно розгортати повноцінний веб-сервер на пристрої де запускається бот.

Окрім запуску веб-серверу для коректної роботи потрібно мати валасний SSL сертифікат, адже веб-хуки в Телеграм нормально працють лише з використаням HTTPS запитів.Більш детально роботацх методів зоржена на рисунку 2.5.

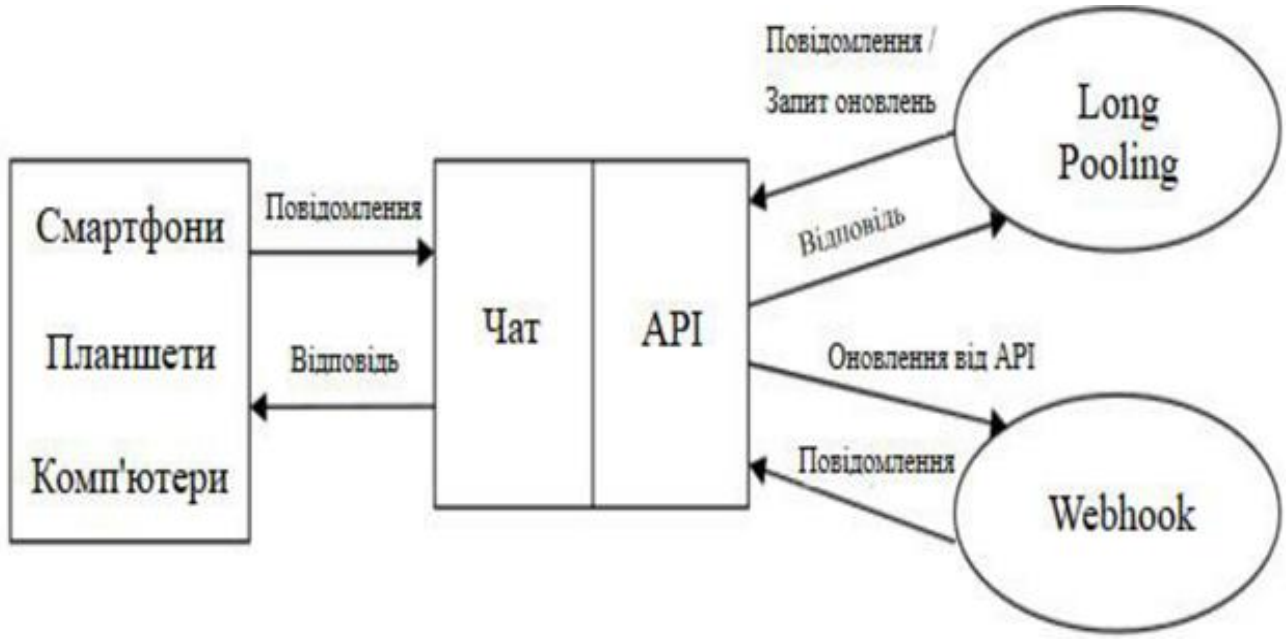

Рисунок 2.5 – Принцип роботи бота у платформі Telegram

Нижче наведенні приклади методів для TELEGRAM BOT API:

а) метод getUpdates використовується для оновлення повідомлень, за допомогою принципів long polling;

б) метод setWebhook – дозволяє присвоїти боту url домену, на якому запущений цей бот;

в) метод sendMessage дозволяє відправляти повідомлення в Телеграм клієнт;

г)метод sendLocation дозволяє відправити координати місцезнаходження пристрою повідомленням.[2]

д) метод getFile повертає файли.

Параметри до Telegram Bot API можна передавати 4 способами:

а) запит в URL;

б) application / x-www-form-urlencoded;

в) application / json;

г) multipart / form-data.[2]

Для роботи з Telegram Bot API була вивчена документація, в якій описані всі методи і параметри, було встановлено, що всі відповіді приходять у JSON-форматі. В ході написання чат-бота були протестовані і використані наступні методи і типи:

getUpdates – метод використовується для отримання оновлень через long polling. Результатом роботи цього методу є масиву об'єктів Update. В таблиці 2.5 наведенні параметри даного методу.[2]

Таблиця 2.5 – Параметри getUpdates

Параметри	Тип	Обов'язковий	Опис
Offset	Integer	Ні	Зсув відносно початку масиву Update, дозволяє повідомлення, ігноруючи старі.
Limit	Integer	Ні	Обмежує Update. Приймає значення від 1 до 100.
Timeout	Integer	Ні	Таймаут в секундах для тривалого опитування. За умовчуванням 0, тобто так званий короткий запит.

setWebhook – метод для url веб-хука, на який будуть відправлятись оновлення боту. При оновленні на цей url зразу відправляється HTTPS POST запит з перетвореним в JSON об’єктом Update.[2]

Якщо запит на сервер не пройшов, то спроба підключитись буде повторена декілька разів. Для збільшення надійності потрібно включити токен боту в url веб-хука, наприклад: https://youserver/ <token>. Завдяки таким діям точно відомо, що запит веб-хуку відправляється з Телеграму, оскільки більше ніхто не знає токен. Параметри методу описанні в таблиці 2.6.[2]

Інформацію про поточний стан веб-хук підключення можна отримати методом getWebhookInfo. В таблиці 2.7 показані параметри даного методу.[2]

Таблиця 2.6 – Параметри setWebhook

Параметри	Тип	Обов'язковий	Опис
url	String	Hi	HTTPS url на яий відправляються запити.
Sertificate	InputFile	Hi	Використовується для перевірки сертифіката підключається публічний ключ .

Таблиця 2.7 - Параметри які повертає getWebhookIn

Параметри	Тип	Опис
url	String	url веб-хука, необов’язковий параметр.
has_custom_certificate	Boolean	Повертає True, якщо -хук використовує свій само завірений сертифікат.
pending_update_count	Integer	Кількість оновлень в черзі.
last_error_date	Integer	Час останньої помилки, яка виникла в результаті неправильної доставки на даний веб-хук. Не обов’язковий.
last_error_message	String	Опис останньої помилки доставки оновлення на вказаний веб-хук.

Відправка повідомлень користувачу відбувається за допомогою методу getMessage. Параметри методу наведені в таблиці 2.8.

Таблиця 2.8 – параметри SendMessage

Параметри	Тип	Обов'язковий	Опис
chat_id	Integer або String	Так	Унікальний ідентифікатор цільового чату або імені цільового каналу.
Text	String	Так	Текстповідомлення

Для відправки фото використовується Метод sendPhoto. Параметри методу наведені в таблиці 2.9.

Таблиця 2.9 - Параметри sendPhoto

Параметри	Тип	Обов'язковий	Опис
chat_id	Integer або String	Так	Унікальний ідентифікатор цільового чату або імені цільового каналу.
Photo	InputFile або String	Так	Фото для відправки. Даний метод дозволяє передавати file_id, якщо дане зображення вже є на серверах Телеграм, або додати нову фотографію
Параметри	Тип	Обов'язковий	Опис
Caption	String	Ні	Тема фотографії. Ліміт 200 символів

Продовження таблиці 2.9

Параметри	Тип	Обов'язковий	Опис
Replay_markup	InlineKeyboard Markup or ReplyKeyboard Markup or ReplyKeyboard Hide or ForcekReply	Ні	Розширені можливості пошуку інтерфейсу. Являє собою JSON-серіалізований об'єкт для вбудованої клавіатури, яка призначена для користувача, певні інструкції які використовуються, для приховування клавіатури користувача або примусової відповіді від користувача.[2]

Метод editMessageText використовується для редагування текстових повідолень. Параметри методу наведеніу таблиці 2.10.

Таблиця 2.10 - Параметри методу editMessageText

Параметри	Тип	Обов'язковий	Опис
chat_id	Integer або String	Ні	Необхідний параметр, якщо inline_message_id не вказано. Унікальний ідентифікатор цільового чату або імені каналу.
inline_message_id	String	Ні	Обов'язково, якщо chat_id і message_id не вказані. Ідентифікатор вбудованого повідомлення

Продовження таблиці 2.10

Параметри	Тип	Обов'язковий	Опис
Text	String	Так	Новий текст повідомлення.
parse_mode	String	Ні	Потрібно надіслати Markdown або HTML, щоб додатки Telegram відображали напівжирний, курсивний, текст з фіксованою шириною або вбудовані URL-адреси в повідомленні нашого бота.
disable_web_page_preview	Boolean	Ні	Відключає попередній перегляд посилань для посилань в цьому повідомленні.
replay_markup	InlineKeyboardMarkup або ReplyKeyboardMarkup або ReplyKeyboardHide	Ні	Розширені можливості пошуку інтерфейсу.

Інформацію про користувача в Телеграм надає поле ти User. Поля об’єкту наведені в таблиці 2.11.

Таблиця 2.11 – Поля об’єкту user

Поле	Тип	Опис
Id	Integer	Унікальний ідентифікатор користувача (бота)
first_name	String	Ім'я користувача/бота
last_name	String	Прізвище користувача/бота (не обов’язково)
Username	String	Username користувача/бота (не обов’язково)

Інформація про чат міститься в обєкті типу Chat. В таблиці 2.12 наведені поля даного типу.

Таблиця 2.12 – Поля об’єкту chat

Поле	Тип	Опис
Id	Integer	Унікальний ідентифікатор чату. Абсолютне значення не перевищує 1e13[2]
Type	Enum	Тип чату: а) private; б) group; в) supergroup; г) channel;[2]
Title	String	Назва, для каналів/груп (не обов’язково)
Username	String	Username, для чатів і деяких каналів (не обов’язково) [2]
first_name	String	Ім'я співрозмовника в чаті (не обов’язково) [2]
last_name	String	Прізвище співрозмовника в чаті (не обов’язково)[2]
all_members_are_administrators	Boolean	True, якщо всі учасники чату є адміністраторами. (не обов’язково)[2]

Об’єкт типу Message представляє собою інформацію про повідомлення. Поля типу наведені в таблиці 2.13.

Таблиця 2.13 – Поля об’єкту Message

Поле	Тип	Опис
message_id	Integer	Унікальний ідентифікатор повідомлення [2]
From	User	Не обов’язково. Відправник. Може бути порожнім в каналах.[2]
Date	Integer	Дата відправлення повідомлення (Unix time)[2]
Chat	Chat	Діалог, в якому було відправлено повідомлення [2]
forward_from	User	Не обов’язково. Для пересланих повідомлень: відправник оригінального повідомлення[2]
forward_date	Integer	Не обов’язково. Для пересланих повідомлень: дата відправки повідомлення.[2]
reply_to_message	Message	Не обов’язково. Для відповідей: оригінальне повідомлення. Варто звернути увагу, що об'єкт Message в цьому полі не буде містити додаткові поля reply_to_message, навіть якщо він сам є відповіддю.[2]
Text	String	Не обов’язково. Для текстових повідомлень: текст повідомлення, 0-4096 символів.[2]

Продовження таблиці 2.13

Поле	Тип	Опис
Entities	Масив з MessageEntity	Не обов'язково. Для текстових повідомлень: особливі суті в тексті повідомлення.[2]
Document	Document	Не обов'язково. Інформація про Фото[2]
Photo	масив з PhotoSize	Не обов'язково. Доступні розміри Фото[2]
Sticker	Sticker	Не обов'язково. Інформація про Стікери[2]
Video	Video	Не обов'язково. Інформація про Відеозапис[2]
Voice	Voice	Не обов'язково. Інформація про голосове повідомлення[2]
Caption	String	Не обов'язково. Підпис до файлу, фото або відео, 0-200 символів[2]
Contact	Contact	Не обов'язково. Інформація про відправлений контакт.[2]
Location	Location	Не обов'язково. Інформація про місцезнаходження.[2]
Venue	Venue	Не обов'язково. Інформація про місце на карті.[2]
new_chat_member	User	Не обов'язково. Інформація про користувача, доданому до групи.[2]
left_chat_member	User	Не обов'язково. Інформація про користувача, якого видалили.[2]

Об'єкт типу ReplyKeyboardMarkup являє собою клавіатуру з опціями відповіді. Поля типу наведені в таблиці 2.14.

Таблиця 2.14 – Поля об'єкта ReplyKeyboardMarkup

Поле	Тип	Опис
Keyboard	масив масивів з KeyboardButton	Масив рядів кнопок, кожен з яких є масивом об'єктів KeyboardButton[2]
resize_keyboard	Boolean	Не обов’язково. Вказує клієнту підігнати висоту клавіатури під кількість кнопок (зробити її менше, якщо кнопок мало). За замовчуванням False, тобто клавіатура завжди такого ж розміру, як і стандартна клавіатура пристрою. [2]
one_time_keyboard	Boolean	Не обов’язково. Вказує клієнту приховати клавіатуру після використання (після натискання на кнопку). Її як і раніше можна буде відкрити через іконку в поле вводу повідомлення. За замовчуванням False[2].
Selective	Boolean	Не обов’язково. Цей параметр потрібен, щоб показувати клавіатуру тільки деяким користувачам. Приклад: користувач відправляє запит на зміну мови бота. Бот відправляє клавіатуру зі списком мов, видиму тільки цьому користувачеві.[2]

Об'єкт типу KeyboardButton є ще однією кнопкою в клавіатурі відповіді. Для звичайних текстових кнопок цей об'єкт може бути змінений на рядок, що містить текст на кнопці. Поля типу наведені в таблиці 2.15.

Таблиц 2.15 – Поля об'єкта KeyboardButton

Поле	Тип	Опис
Text	String	Текст на кнопці. Якщо жодне з опціональних полів не використано, то при натисканні на кнопку цей текст буде відправлений боту як просте повідомлення.[2]
request_contact	Boolean	Не обов’язково. Якщо значення True, то при натисканні на кнопку боту відправиться контакт користувача з його номером телефону. Доступно тільки в діалогах з ботом.[2]
request_location	Boolean	Не обов’язково. Якщо значення True, то при натисканні на кнопку боту відправиться розташування користувача. Доступно тільки в діалогах з ботом.[2]

Об'єктом типу InlineKeyboardMarkup є вбудована клавіатура, яка з'являється під відповідним повідомленням. Поля типу наведені в таблиці 2.16.

Об'єкт типу InlineKeyboardButton це ще одна кнопка вбудованої клавіатури. Тоді обов'язково необхідно задіяти рівно одне опціональне поле, типи та параметри якого наведено в таблиці 2.17.

Таблиця 2.16 - Поля об'єкта InlineKeyboardMarkup

Поле	Тип	Опис
inline_keyboard	Масив масивів з InlineKeyboardButton	Масив рядків, кожна з яких є масивом об'єктів InlineKeyboardButton.[2]

Таблиця 2.17 - Поля об'єкта Inline KeyboardButton

Поле	Тип	Опис
Text	String	Текст на кнопці[2]
url	String	Не обов’язково. URL, який відкриється при натисканні на кнопку[2]
callback_data	String	Не обов’язково. Дані, які будуть відправлені в callback_query при натисканні на кнопку.[2]
switch_inline_query	String	Не обов’язково. Якщо цей параметр заданий, то при натисканні на кнопку додаток запропонує користувачеві вибрати будь-який чат, відкриє його і вставить В поле вводу повідомлення юзернейм бота і певний запит для вбудованого режиму. Якщо відправляти пусте поле, то буде вставлений тільки юзернейм бота. [2]

Продовження таблиці 2.17

Поле	Тип	Опис
switch_inline_query_current_chat	String	Не обов’язково. Якщо встановлено, натискання кнопки введе ім'я бота і вказаний вбудований запит в поле вводу поточного чату. Може бути порожнім, і в цьому випадку буде вставлено тільки ім'я користувача бота.[2]
callback_game	CallbackGame	Не обов’язково Опис гри, яка буде запущена, коли користувач натисне кнопку. Примітка: Цей тип кнопки завжди повинен бути першою кнопкою в першому рядку.[2]

На рисунку 2.6 зображено використання об’єкту InlineKeyboard в розробленому чат-боті.

```
keyboardMainMenu.add(buttonRunTest, buttonResult, buttonSettings)
keyboardMenuRunTest.row(buttonRunTestApi, buttonRunTestDesctop, buttonRunTestMobileWeb)
KeyboardTestResult.add(APIResult, DesctopResult, MobileWEBResult)
keyboardSettings.row(buttonTime)
keyboardBack.add(buttonBack)
```

Рисунок 2.6 – Використання об’єкту InlineKeyboard

На рисунку 2.7 приклад використання об'єкту InlineKeyboardButton.

```
buttonRunTest = telebot.types.InlineKeyboardButton('Run test', callback_data='run test')
buttonResult = telebot.types.InlineKeyboardButton('Test Result', callback_data='test result')
buttonSettings = telebot.types.InlineKeyboardButton('Settings', callback_data='settings')

buttonRunTestApi = telebot.types.InlineKeyboardButton('API', callback_data='test api')
buttonRunTestDesctop = telebot.types.InlineKeyboardButton('Desctop', callback_data='test api')
buttonRunTestMobileWeb = telebot.types.InlineKeyboardButton('Mobile Web', callback_data='test api')

buttonBack = telebot.types.InlineKeyboardButton('<- Back', callback_data='back')

APIResult = telebot.types.InlineKeyboardButton('API', callback_data='test result api')
DesctopResult = telebot.types.InlineKeyboardButton('Desctop', callback_data='test result api')
MobileWEBResult = telebot.types.InlineKeyboardButton('MobileWEB', callback_data='test result api')
```

Рисунок 2.7 - Використання об'єкту InlineKeyboardButton.

Висновки до розділу

В даному розділ було проведено дослідження технологій для розробки телеграм бота та технологій для його нормальної та безперервної роботи. Отже для розробки чат бота було використано мову програмування Python, модуль TeleBot для роботи з Телеграм API, модуль CherryPy, модуль Subproces для роботи з віддаленим об’єктом, та Telegram API. Також наведено визначення Rest API та сформульовані його основні принципи.

3 Тестування розробленої системи

Тестування розробленого боту відбувалось вручну. Тестування працездатності відбувалось після кожного етапу розробки за підготовленими тест-кейсами. Тест-кейси були складенні заздалегідь, при визначенні цілей розробки і інструментів, котрі використовувались в ході розробки.

Основні тест-кейси:

а)швидкість відклику чат-бота на команду;

б)коректна обробка повідомлень від користувача.

в)коректне відображення діалогу;

г)обробка натискання на клавішу відправки повідомлення;

д)швидкість відображення відповіді після того, як користувач, написав повідомлення;

е)оцінювання коректності відповідей.

Тестування чат боту відбувалось на «Iphone 7»(мобільний пристрій). Технічні характеристики:

а) екран: 4,6 ", IPS LCD, 1920x1080;

б) процесор: Apple A9 Fusion;

в) операційна система: iOS 13.3.2;

г) оперативна пам'ять: 4 ГБ;

д) вбудована пам'ять: 64 ГБ.

Версія месенджера Telegram 6.1.2.

Після початку чату з ботом на екрані з’являється опис та команда «Start» (рисунок 3.1). При натисканні на клавішу «Start» отримуємо основну інформацію, про роботу боту та головне меню (рисунок 3.2), або отримаємо повідомлення «Hello! Wrong user id. You can`t use me. Please contact @stepa1234». Дане повідомлення генерується, якщо користувач не може користуватись даним ботом. Тобто його немає в списку користувачів.

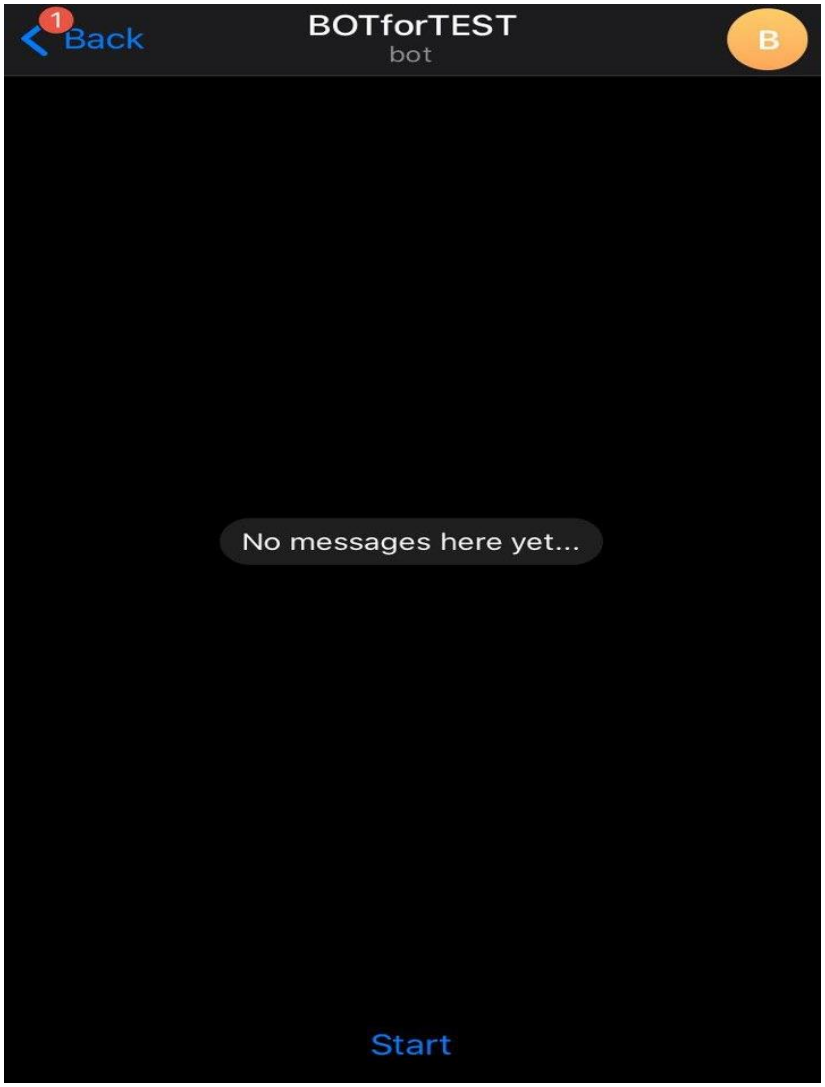

Рисунок 3.1 – Початок роботи з ботом

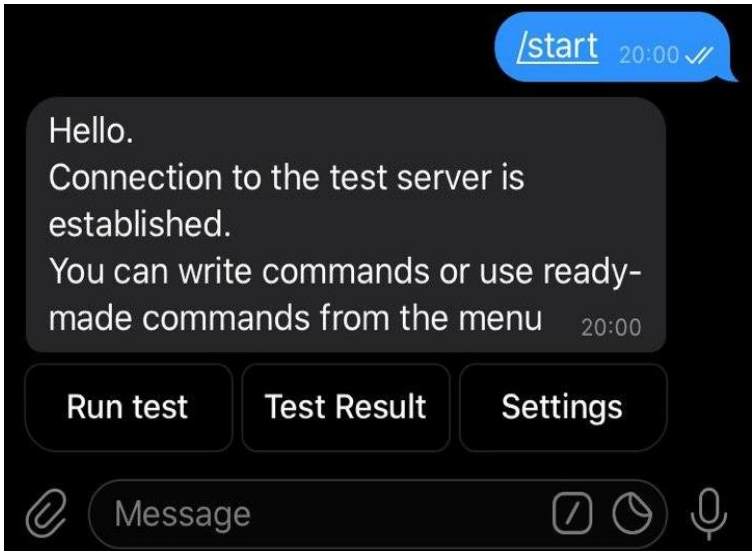

Рисунок 3.2 – Привітальне повідомлення

У головному меню три пункти (рисунок 3.2):

1. «Run test».
2. «Test Result».
3. «Settings».

При натисканні на пункт «Run test», з’являєтеся меню, де можна обрати вид тесту, який потрібно запустити (рисунок 3.3).

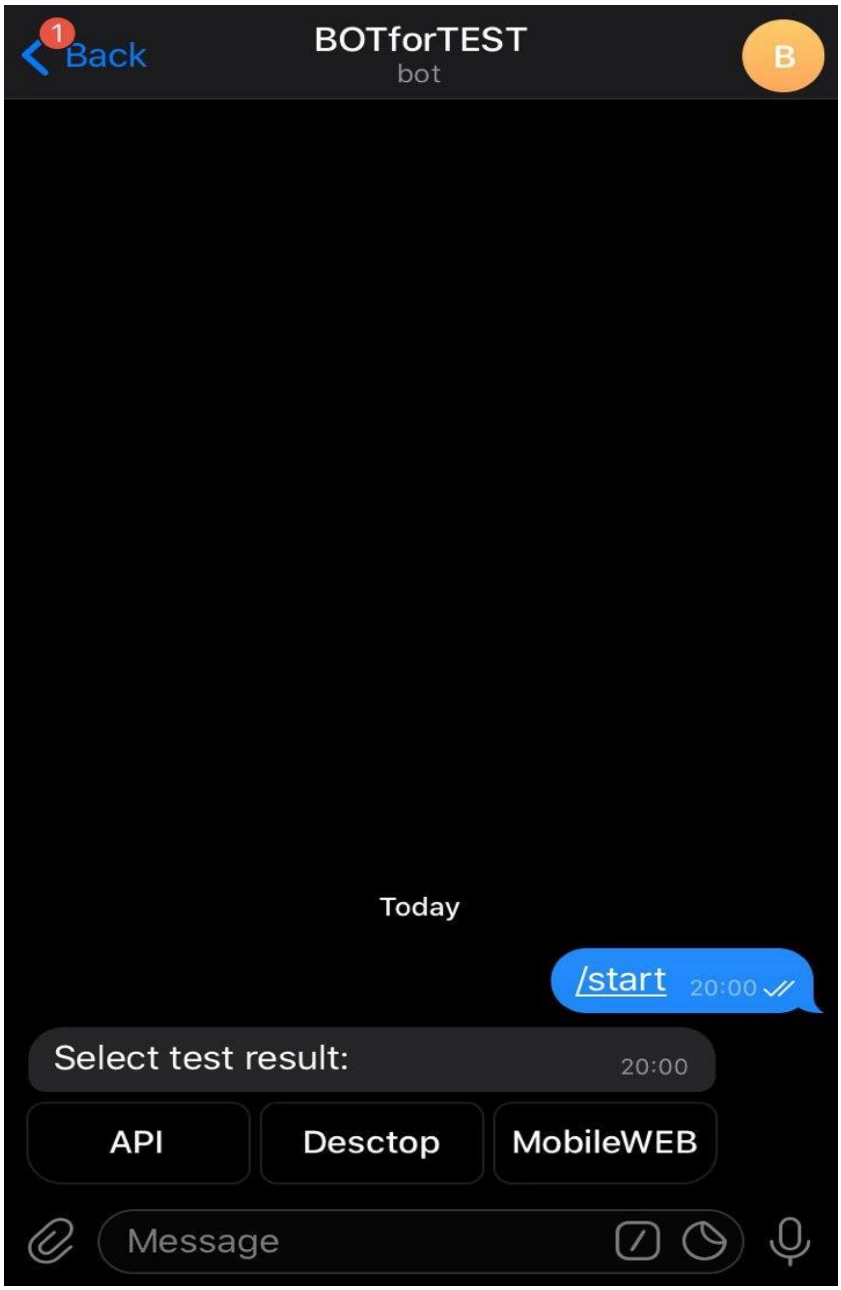

Рисунок 3.3 – Головне меню

Обравши тест серед «API», «Desktop», «Mobile Web» після його виконання приходить повідомлення, про завершення тесту і кнопка повернення в головне меню «← Back» (рисунок 3.4).

Рисунок 3.4 – Меню після запуску тесту

Якщо обрати пункт меню «Test Result», на екрані з’явиться меню(рисунок 3.5).

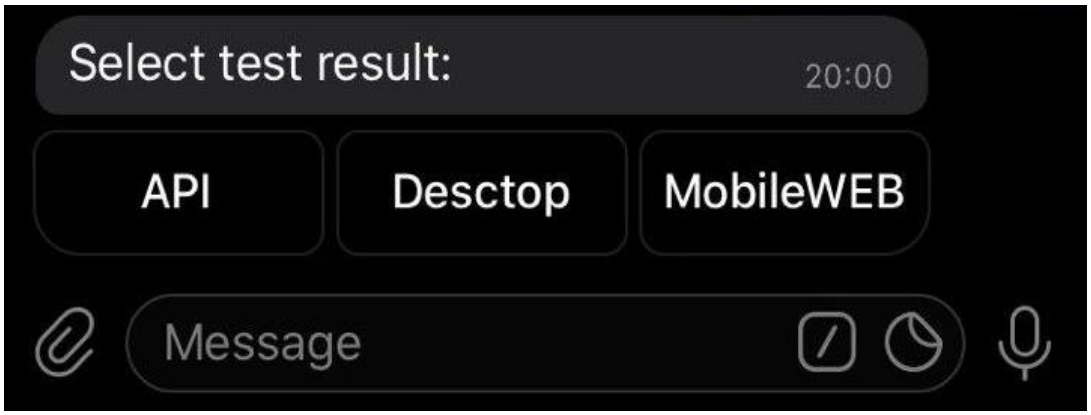

Рисунок 3.5 – Меню «Test Result»

В даному меню можна обрати результати якого тестування потрібно отримати. Коли вибір зроблено, бот пришле в повідомленні файл з потрібним результатом (рисунок 3.6).

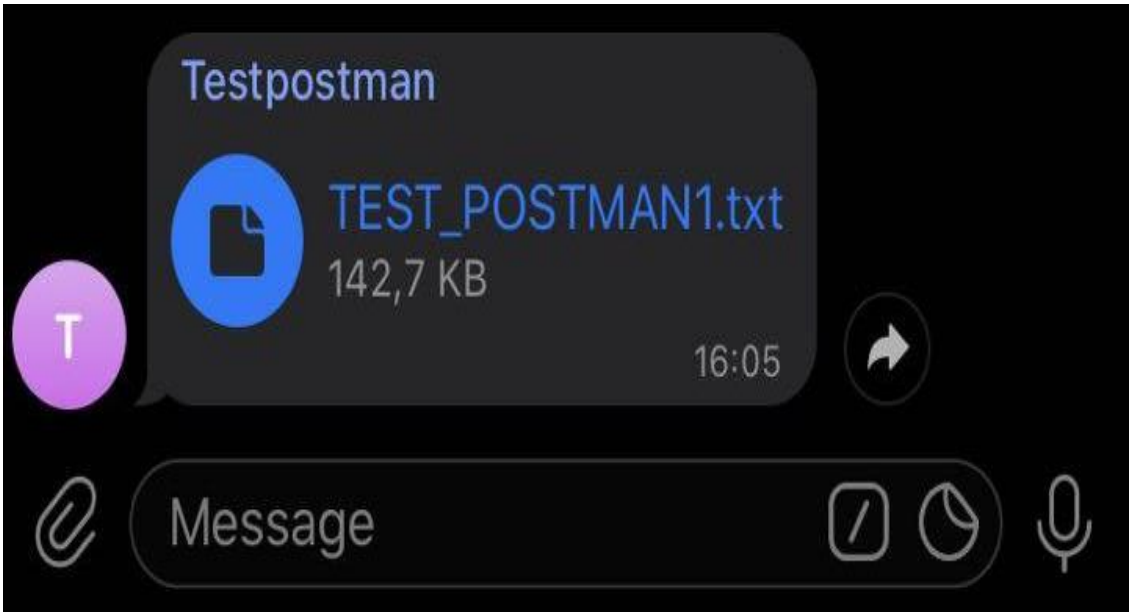

Рисунок 3.6 – Результати тестування

Якщо обрати останній пункт меню то ми перейдемо в меню налаштувань серверу (рисунок 3.7).

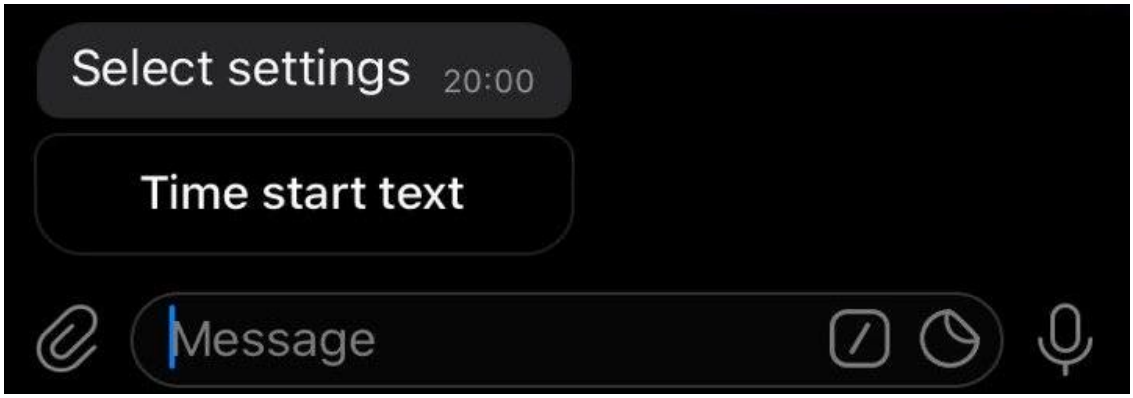

Рисунок 3.7 – Меню «Settings»

В меню налаштувань є пункт «Time start test». Вибір цього пункту дозволяє встановити час автоматичного запуску авто тестів на сервері (рисунок 3.8).

Рисунок 3.8 – Робота «Time start test»

Час потрібно ввести повідомленням через двокрапку (рисунок 3.9). При позитивному результаті у відповідь приходить повідомлення з текстом «ok».

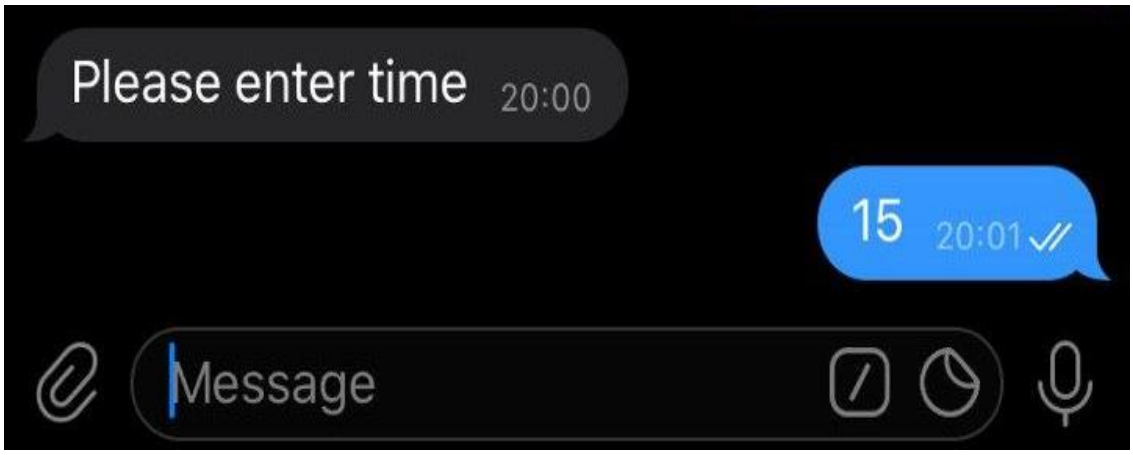

Рисунок 3.9 – Результат роботи «Time start test»

Окрім меню в даному чат-боті є зарезервовані команди:

7. «/start» - робота даної команди зображена на початку цього розділу.
8. «/help» - виводить підказки для роботи з ботом (рисунок 3.10).

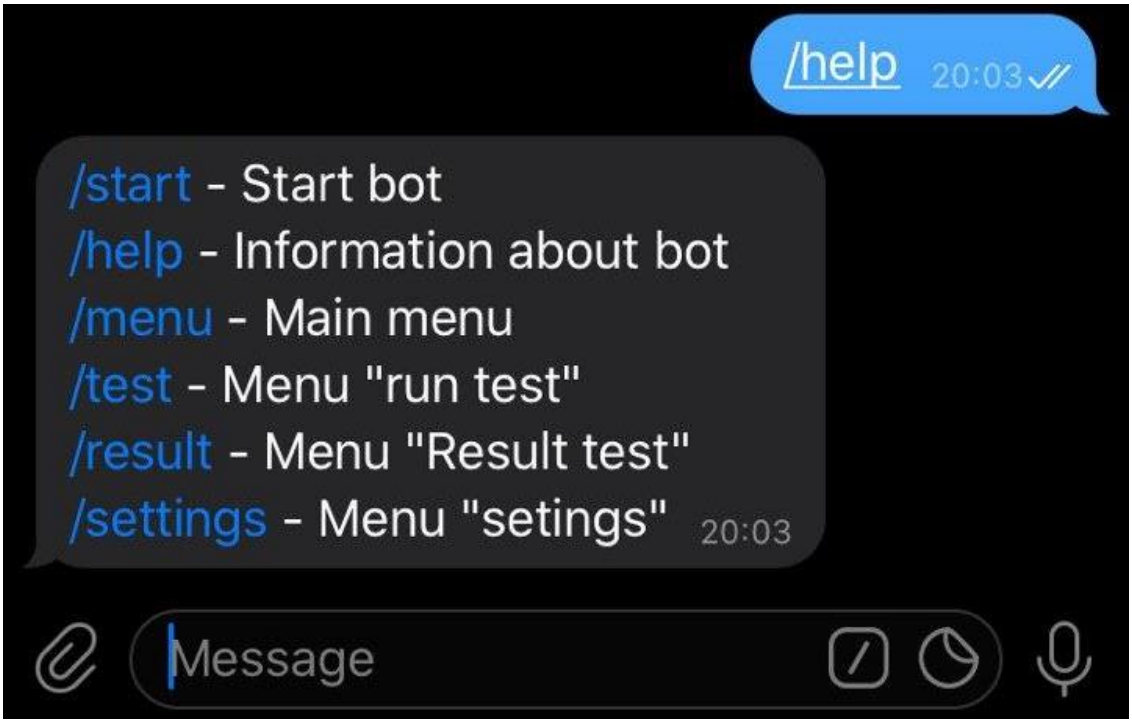

Рисунок 3.10 – Робота команди «/help»

9. «/menu» - виводить головне меню (рисунок 3.11)

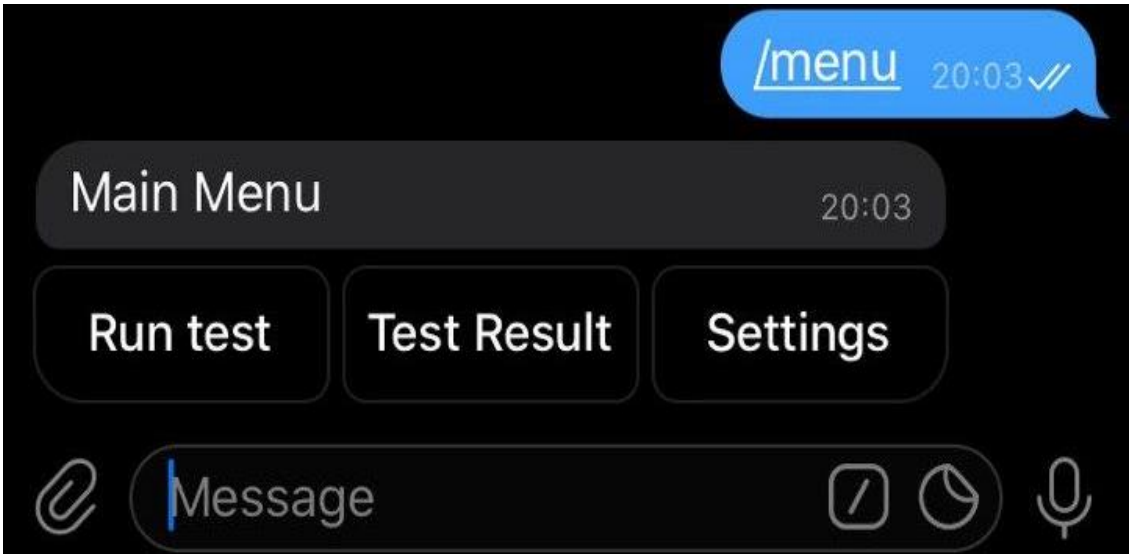

Рисунок 3.11- Робота команди «/menu»

10.«/test» - вводить меню для запуску тестів (рисунок 3.12)

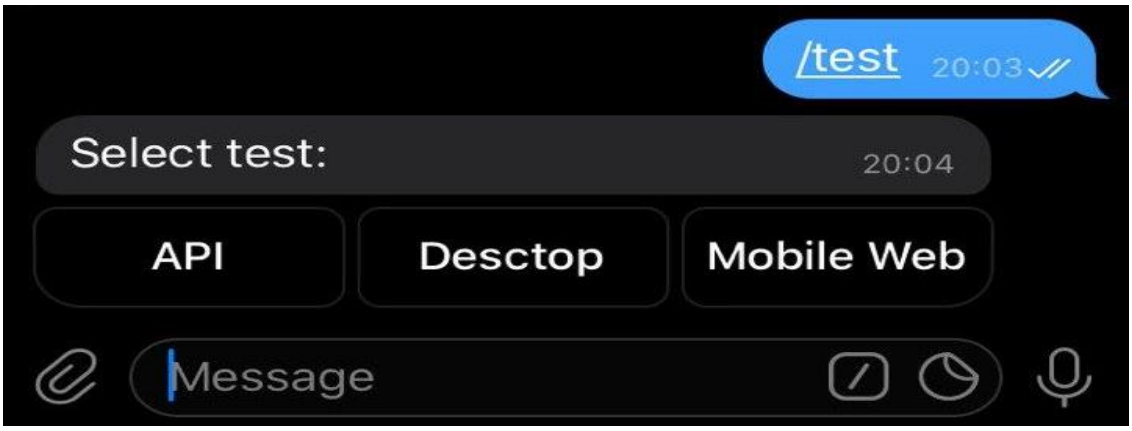

Рисунок 3.12 -Робота команди «/test»

11.«/result» - виводить на екран меню для вибору результатів тестів (рисунок 3.13)

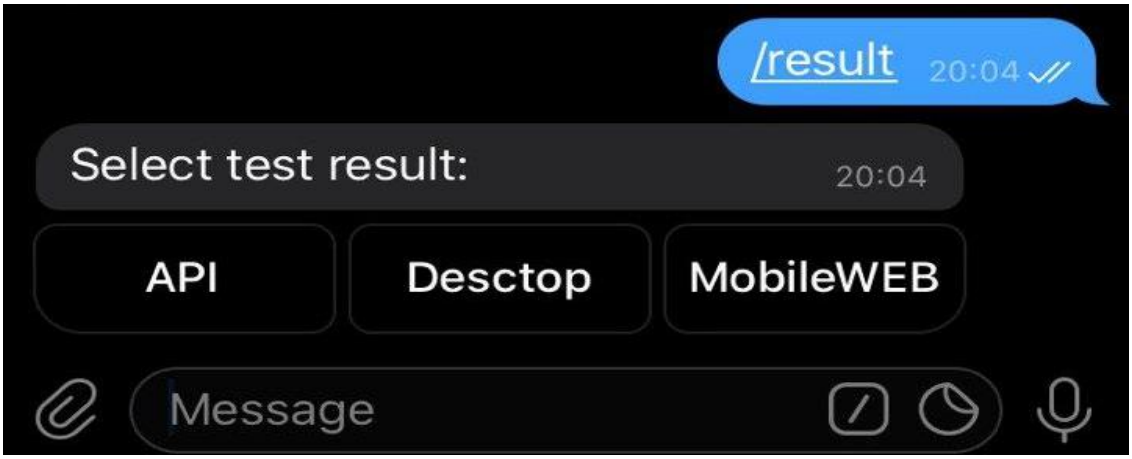

Рисунок 3.13 - Робота команди «/result»

12.«/settings» - виводить на екран меню настройок (рисунок 3.14)

Рисунок 3.14 - Робота команди «/settings»

Однією з основних функцій даного чат-бота є відправка команд на сервер в формі текстового повідомлення боту, котрий обробляє його та відсилає на сервер. Потім присилає нам відповідь від серверу (рисунок 3.16).

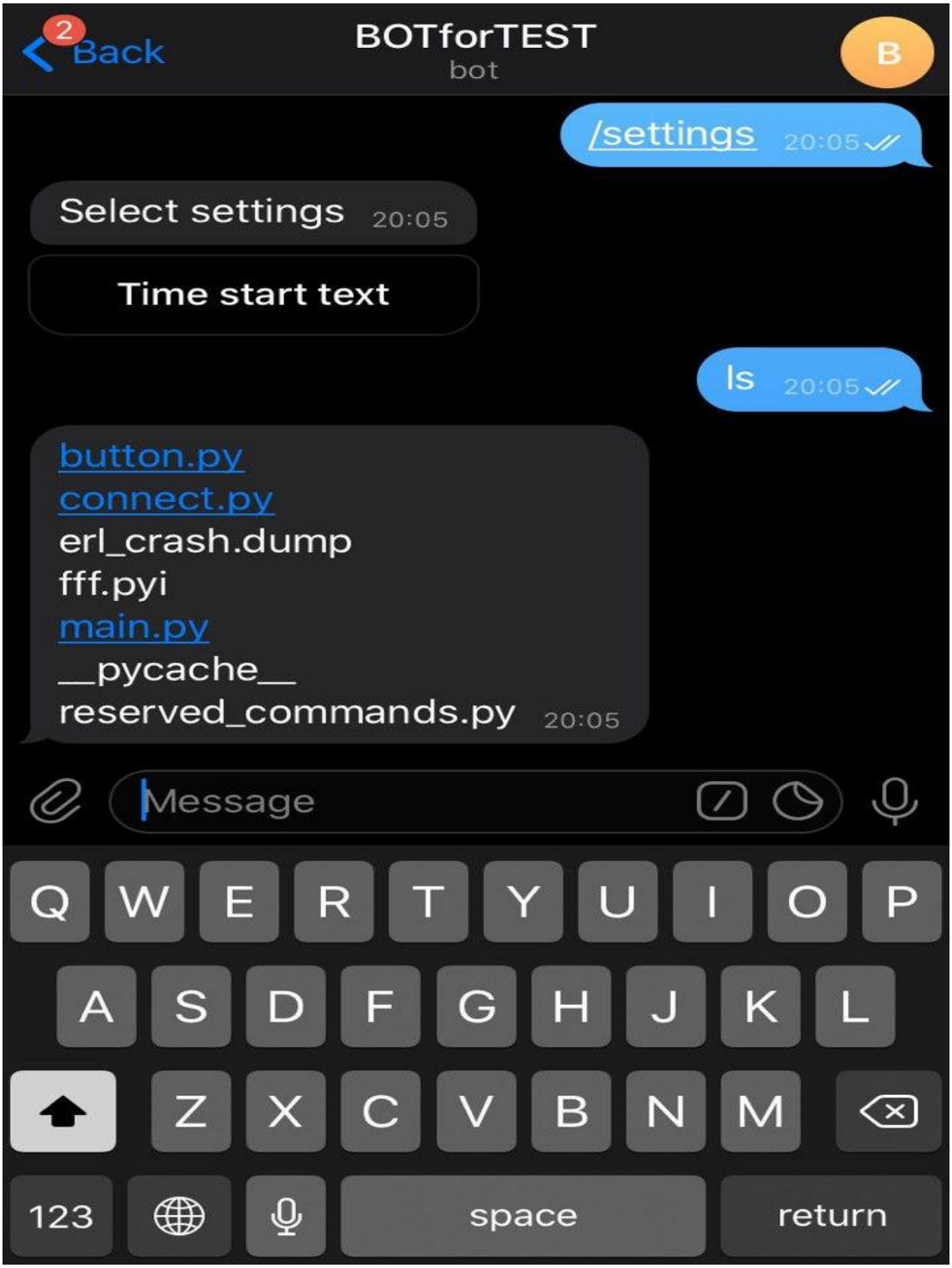

Рисунок 3.16 – Приклад роботи команди

Якщо команда виконалась та помилку «Invalid input. Please enter terminal command», якщо команда не виконалась (рисунок 3.17).

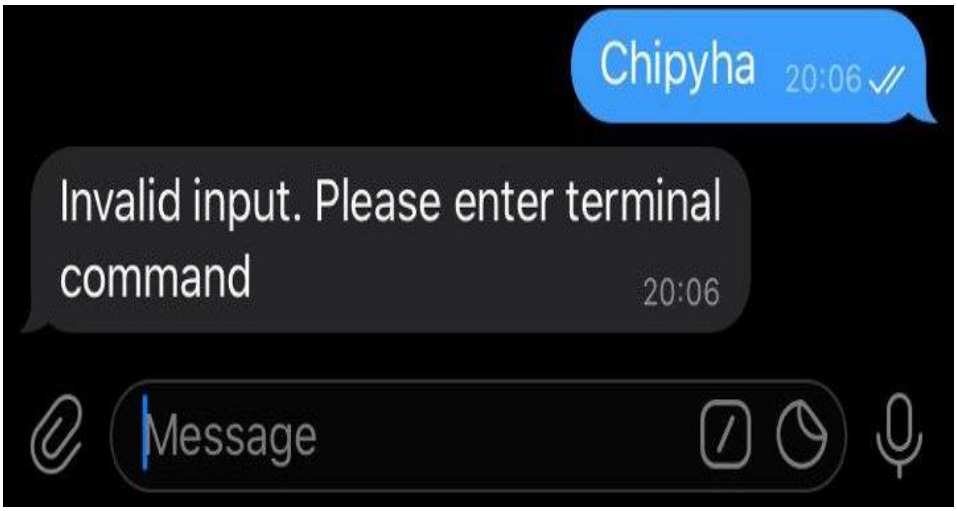

Рисунок 3.17 – Повідомлення про помилку

Висновки до розділу

Тестування є одним з найголовніших аспектів в розробці програмного забезпечення. Тестування програми проводилось на кожному етапі розробки (unit тести), та проводився інтеграційний тест наприкінці розробки ПЗ.

Під час тестування не виявлено критичних багів, які б впливали на роботу системи. Це значить, що розроблена система надійна та стійка.

Висновки

В ході виконання дипломного проєкту було визначено головні вимоги та бізнес-процеси системи, а також проведено дослідження предметної області.

Проаналізувавши процеси та функції, котрі повинні виконувати система, були обрані: мова програмування, модулі для роботи з Телеграм API, та технології для роботи з віддаленим об’єктом.

Результатом дипломного проєкту стала розробка кросплатформеної системи для доступу до віддалених об’єктів на базі месенджеру Телеграм, а саме Телеграм бот @INratingtest_bot. Дана система є сучасно і зручною. Вона задовольняє всім вимогам щодо функціональності, логіки та простоти користування.

Список використаної літератури

1. Топ-10 корисних Telegram-ботів для українських користувачів: URL: https://www.epravda.com.ua/publications/2017/08/7/627822 (дата звернення 10.02.2020)
2. Telegram Bot API: URL: https://core.telegram.org/bots/api (дата звернення 15.03.2020)
3. Марк Лутц, Learning Python :2019, 720с.
4. Telegram FAQ: URL: https://telegram.org/faq (дата звернення 20.03.2020)
5. Telegram-bot: URL: https://habr.com/ru/post/442800/ (дата зверення 22.03.2020)
6. Телеграм бот за допомогою TeleBot: URL: https://habr.com/ru/post/448310/ (дата звернення 22.03.2020)
7. Git: URL: https://uk.wikipedia.org/wiki/Git (дата звернення 05.04.2020)
8. Python Requests : URL : https://2.python-requests.org/en/master/ (дата звернення 10.04.2020)
9. Огляд протоколу HTTP : URL: https://developer.mozilla.org/ru/docs/Web/HTTP/Overview (дата звернення 13.04.2020)
10. Черга повідомлень : URL: https://aws.amazon.com/ru/message-queue/ (дата звернення 13.04.2020)
11. What is REST: URL: https://restfulapi.net/ (дата звернення 15.04.2020)
12. REST : URL: https://habr.com/ru/post/351890/ (дата звернення 19.04.2020)
13. JSON: URL: https://www.json.org/json-ru.html (дата звернення 26.04.2020

14. Тестування телеграм-бота: URL: https://habr.com/ru/post/322816/ (дата звернення 08.05.2020)
15. What is Artificial Intelligence :URL: https://www.techopedia.com/definition/190/artificial-intelligence-ai (дата звернення 12.05.2020)
16. Subproces management: URL: https://docs.python.org/3/library/subprocess.html (дата звернення 15.05.2020)