МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Ігор ШЕЛЕХОВ
_____
(підпис)
1 червня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерних наук,

освітньо-професійної програми «Інформатика»

на тему: «Бот для планування та організації робочого процесу»

здобувача групи ІН - 02 Рощиної Анни Сергіївни

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

_____ Анна РОЩИНА
(підпис)

Керівник, доцент, кандидат технічних наук _____ Віктор ОБОДЯК
(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-02 Рощиної Анни Сергіївни

1. Тема роботи: « Бот для планування та організації робочого процесу»
затверджую наказом по СумДУ від *«22» квітня 2024 року № 0414-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 01 червня 2024 року*
3. Вхідні дані до кваліфікаційної роботи
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд технологій, що використовуються у інформаційних системах Телеграм-ботів. 3) Розробка інформаційного та програмного забезпечення телеграм-боту.4) Аналіз результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 2024 р.

Завдання прийняв до виконання _____ (підпис)

Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*		
2	*Огляд технологій, що використовуються у інформаційних системах Телеграм-ботів*		
3	*Розробка інформаційного та програмного забезпечення Телеграм-боту*		
4	*Аналіз отриманих результатів*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ (підпис)

Керівник _____ (підпис)

АНОТАЦІЯ

Записка: 59 ст., 46 рис., 2 табл., 12 використаних джерел

Обґрунтування актуальності теми роботи. Тема кваліфікаційної роботи є актуальною, оскільки присвячена чат-ботам для месенджера Telegram.

Чат-боти для планування виконання задач стають важливим інструментом для ефективного управління робочими та особистими завданнями. Вони забезпечують можливість користувачам швидко додавати, редагувати та видаляти задачі прямо через чат-інтерфейс, без необхідності встановлення додаткових програм або використання складних інтерфейсів.

Об'єкт дослідження – процес створення чат-бота для месенджера Telegram.

Мета роботи – реалізація інформаційного та програмного забезпечення чат-боту Telegram для планування та управління справами користувача.

Методи дослідження – аналіз існуючих рішень, технології існуючих рішень створення чат-ботів.

Результати – розроблено інформаційну систему для запису, редагування, та видалення задач для виконання у мережі Telegram. Створений програмний продукт зручний у використанні, інтерфейс та функціонал інтуїтивно зрозумілий. У результаті авторизовані користувачі соцмережі можуть вести облік виконання своїх справ та не витрачати час на звичайне конспектування їх на паперовий носій або інший сторонній додаток.

ЧАТ-БОТ TELEGRAM, JAVASCRIPT, NEST, POSTGRESQL

ЗМІСТ

ВСТУП

Електронно-обчислювальна техніка призвела до перебудови технічних основ виробництва. Інформаційні технології з'явилися на основі цієї техніки під час четвертої науково-технічної революції, що охопила інтелектуальну діяльність. Інформація в цих процесах є «вихідним матеріалом» і «продукцією». Як виробничі, так і інформаційні технології виникають в результаті модернізації певних процесів, тобто за допомогою впливу людини або групи людей на галузі виробництва і трансформації їх на базі машинної техніки.

Кожна успішна людина повинна уміти раціонально використовувати свій час, тому найкращим варіантом є записування усіх задач на наступний день. Епоха паперових носіїв минула, настала епоха електронних. Для цього є 2 варіанти вирішення даної потреби: стандартна програма «Нотатки», яка є, практично, в кожному смартфоні, або розробка спеціалізованого чат-бота на базі одного із популярних месенджерів.

Чат-бот — це програма, яка імітує справжню розмову з користувачем. За допомогою чат-ботів можна спілкуватися текстовими та аудіо повідомленнями на сайті, в месенджері, мобільному додатку або по телефону [1].

Перевагами використання саме чат-ботів у порівнянні зі звичайними нотатками є:

1. Миттєвість і доступність: користувач може миттєво записувати виконані справи у чат-боті, навіть коли він знаходиться в русі або поза робочим місцем.
2. Зручність і простота використання: чат-інтерфейс є зручним і знайомим для багатьох користувачів, і він дозволяє швидко і легко вводити інформацію про виконані справи.
3. Автоматизація нагадувань: чат-бот може автоматично надсилати нагадування користувачеві про необхідні справи, допомагаючи підтримувати організованість та своєчасність виконання завдань.

4. Інтеграція з іншими сервісами: чат-бот може бути інтегрований з іншими інструментами та додатками, що полегшує обмін даними та використання додаткових функцій.

5. Аналітика та статистика: чат-бот може вести статистику виконаних завдань та надавати аналітичні дані, які можуть бути корисні для оцінки продуктивності та планування часу.

6. Можливість індивідуальних налаштувань: користувач може налаштовувати чат-бота з урахуванням власних потреб та вибору способу ведення записів про справи.

7. Організація та структура: чат-бот може вести систематизовані записи про виконані справи, що допомагає в організації та легкому відслідковуванні історії завдань.

Враховуючи усі вищезазначені факти було вирішено розробити чат-бот для месенджера Телеграм для планування та керування виконанням задач.

Цільова аудиторія – користувачі від 15 до 65 років. Люди саме з цього проміжку найчастіше використовують дану соціальну мережу та мають потребу у плануванні робочого дня (молодші в меншій мірі мають такі обов’язки).

У розробленому продукті користувач зможе виконувати 5 операцій: переглядати список справ, редагувати їх, додавати нові, помічати про виконання та видаляти задачі зі списку.

1 АНАЛІЗ ВІДОМИХ РІШЕНЬ

Для успішної реалізації чат-боту для месенджера Телеграм для планування та керування виконанням задач потрібно зробити огляд готових рішень та аналізуючи недоліки в їх реалізації зробити новий програмний продукт. Для порівняльної характеристики було обрано 3 готових рішення: онлайн-програма Todoist (розробник компанія «Doist»), мобільний додаток Microsoft To Do: Lists & Tasks (розробник корпорація Microsoft) та чат-бот KindReminderBot.

1.1. Огляд готових рішень

1.1.1. Онлайн-програма Todoist

Для того, щоб скористатись даних сервісом потрібно виконати авторизацію одним зі зручних способів: через обліковий запис Google, Facebook або Apple (рис. 1.1).

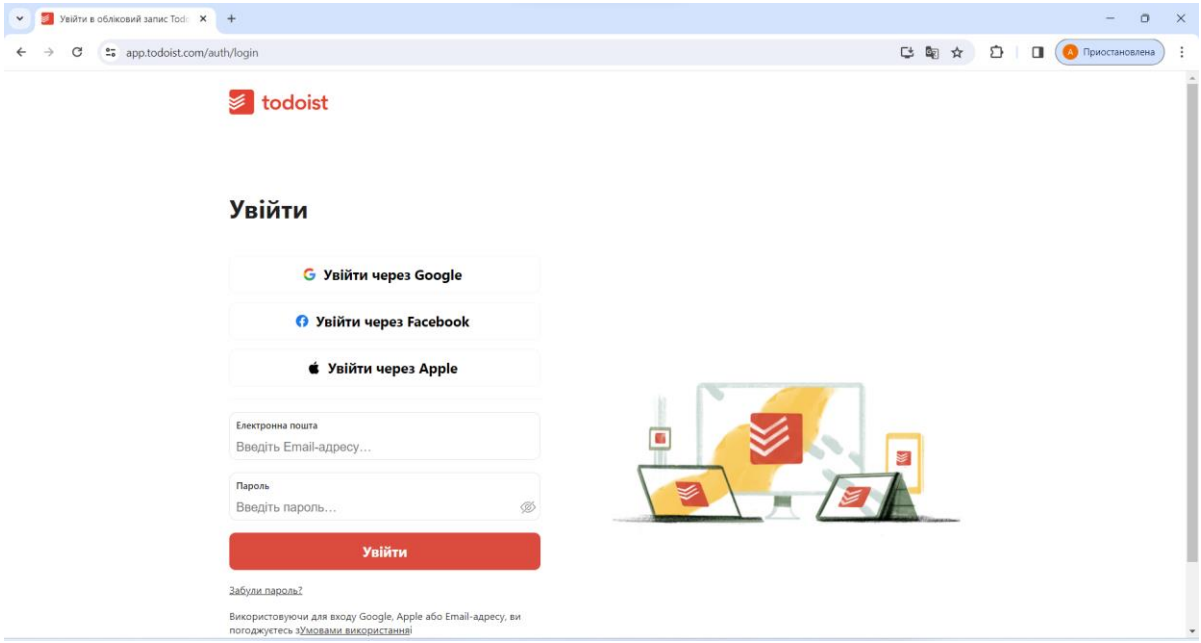

Рисунок 1.1 – Авторизація на сервісі Todoist

Альтернативним способом є безпосередня реєстрація на даному сайті. Для цього потрібно натиснути на кнопку Зареєструватися (рис. 1.2).

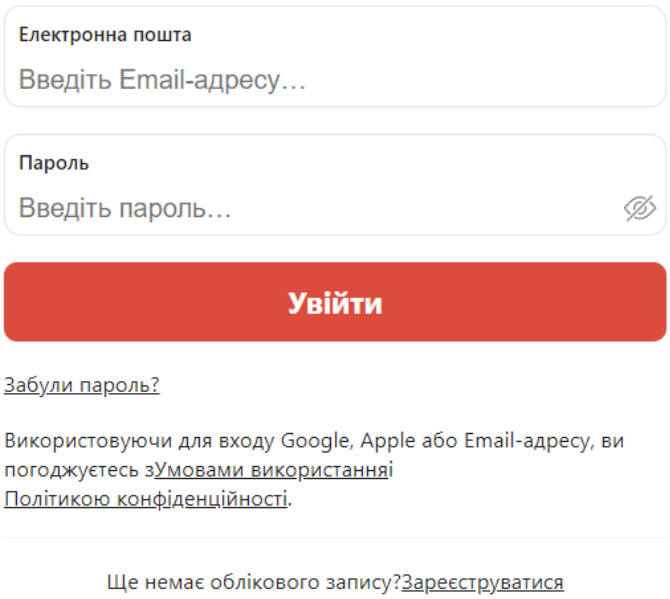

Рисунок 1.2 – Кнопка для реєстрації на сайті

Після даного натискання потрібно увести електронну пошту та пароль і натиснути Увійти. У результаті користувач буде направлений на робоче середовище (рис. 1.3).

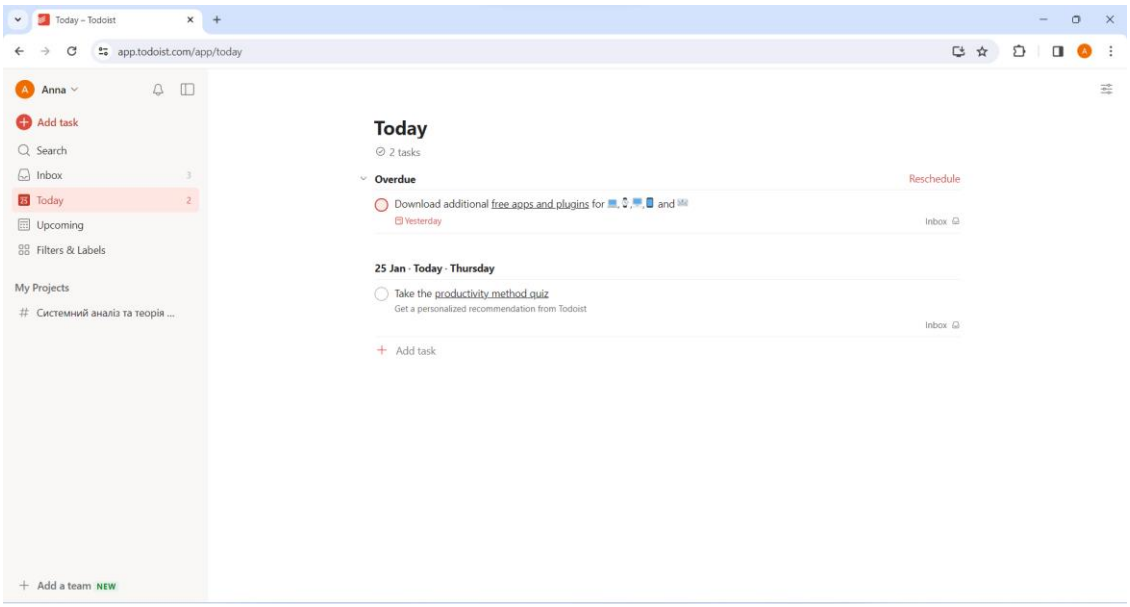

Рисунок 1.3 – Робоче середовище сервісу Todoist

Для управління задачами є декілька способів: роблячи щоденні записи (рис. 1.3) та роблячи окремі проєкти (рис. 1.4).

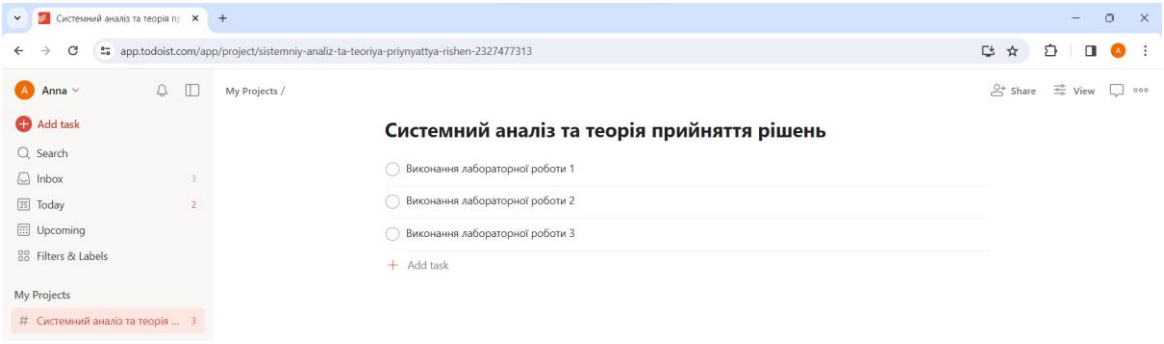

Рисунок 1.4 – Створення окремого проєкту для управління задачами

Створюючи задачу користувач має можливість обрати термін виконання (рис. 1.5), пріоритет (рис. 1.6) та встановити пріоритет (можливе лише з підпискою PRO).

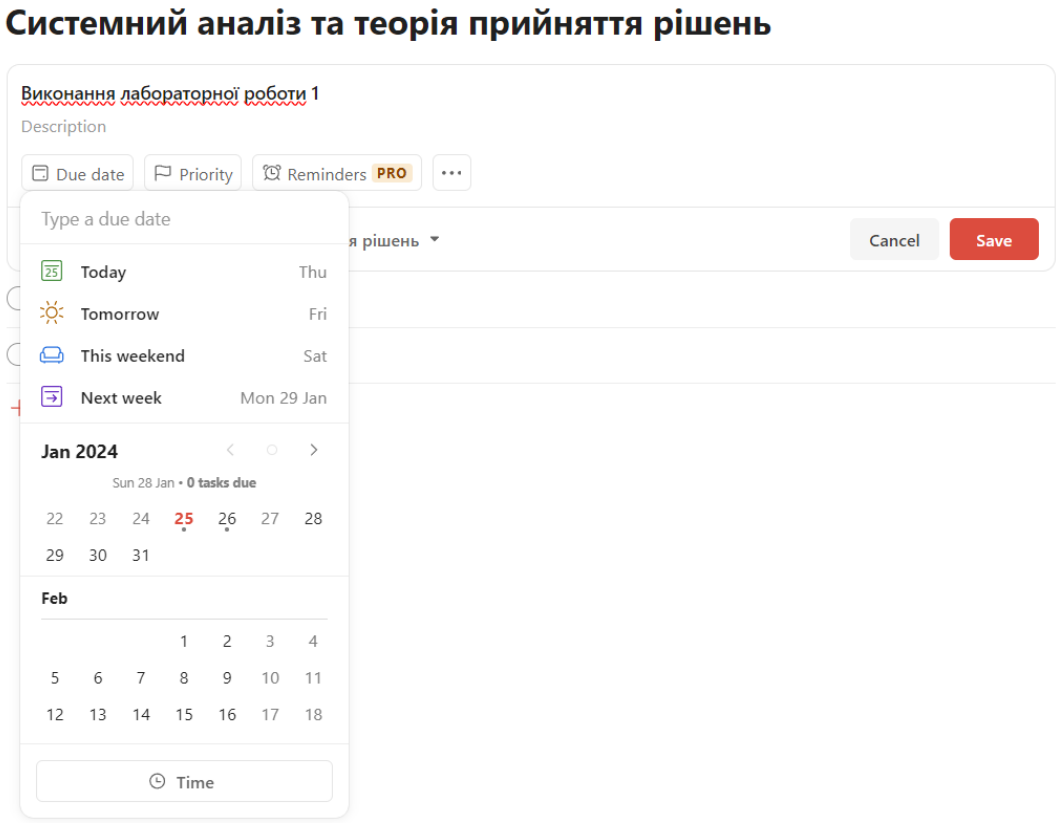

Рисунок 1.5 – Процес обрання терміну виконання задачі

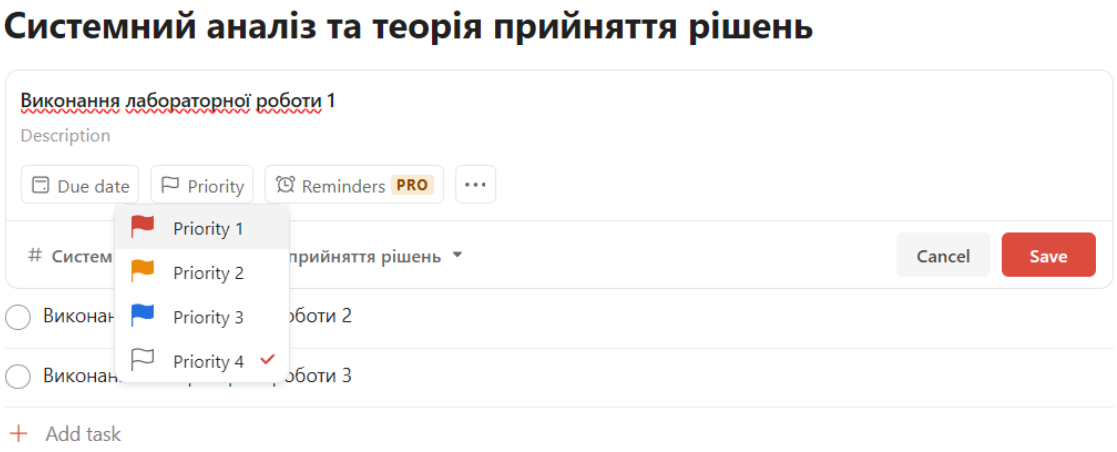

Рисунок 1.6 – Вибір пріоритету виконання користувачем

Варто відзначити, що вищезазначені операції зручно виконувати, оскільки вибір дати виконується безпосередньо на календарі та графічно візуалізовано процес надання задачі пріоритету [2]. Так як за статистикою психологів, 35% людей є візуалами, то графічне відображення відіграє ключову роль при плануванні виконання роботи [3]. Тому це є перевагою даного сервісу.

Серед недоліків варто зазначити обов'язкову авторизацію, тобто користувач повинен відкрити окреме вікно у браузері, аби спланувати роботу протягом наступних днів чи тижнів. Це не завжди зручно, особливо коли виконується багато різних вкладок одночасно.

1.1.2. Мобільний додаток Microsoft To Do: Lists & Tasks

Людина у XXI ст. проводить більшість свого робочого та вільного часу у смартфоні. У такому випадку найоптимальнішим варіантом для планування та управління задачами буде використання мобільного додатку Microsoft To Do: Lists & Tasks. Для початку роботи потрібно авторизуватися за допомогою облікового запису Microsoft (рис. 1.7).

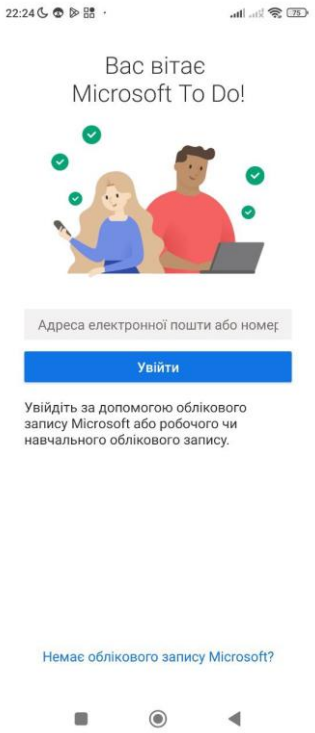

Рисунок 1.7 – Авторизація за допомогою облікового запису Microsoft

Після авторизації користувач потрапляє до головного меню (рис. 1.8). У ньому відображаються задачі на поточний день, важливе (для цього потрібно зробити спеціальну помітку), заплановані задачі на декілька днів вперед та безпосередньо завдання.

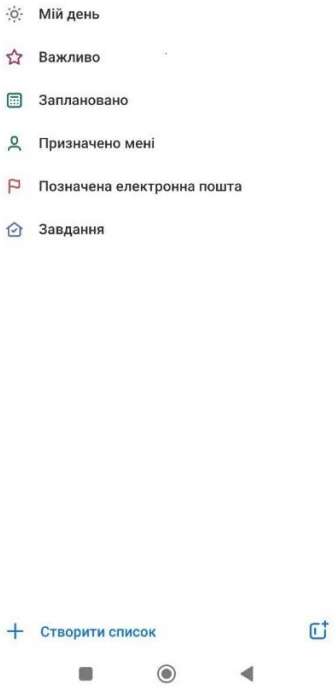

Рисунок 1.8 – Головне меню програми

Варто відзначити мінімалістичний дизайн, але на функціонал це не впливає. Навпаки, зайві графічні елементи не заважають основному призначенню.

Для створення задачі для виконання потрібно натиснути на кнопку Завдання. У відкритому вікні відображається список, поки що він порожній (рис. 1.9).

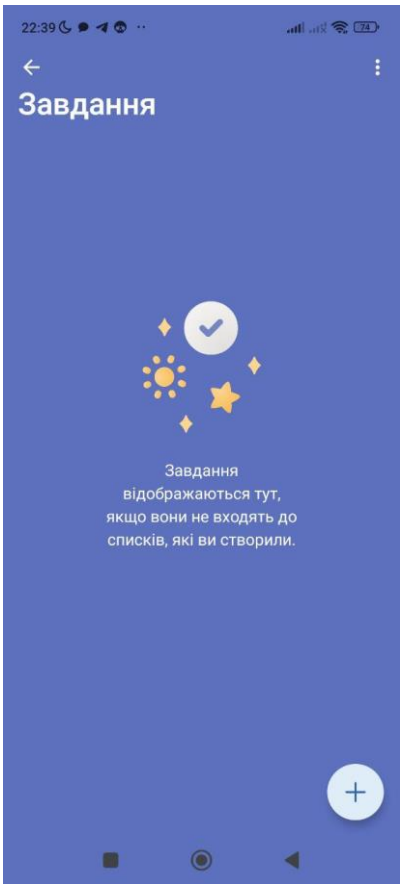

Рисунок 1.9 – Список завдань

Для того, щоб додати новий елемент потрібно натиснути у нижньому правому куті кнопку «+» (рис. 1.10). При створенні встановлюються терміни виконання, за потреби час нагадування. Також за необхідності можливо перенести завдання до списку Важливе.

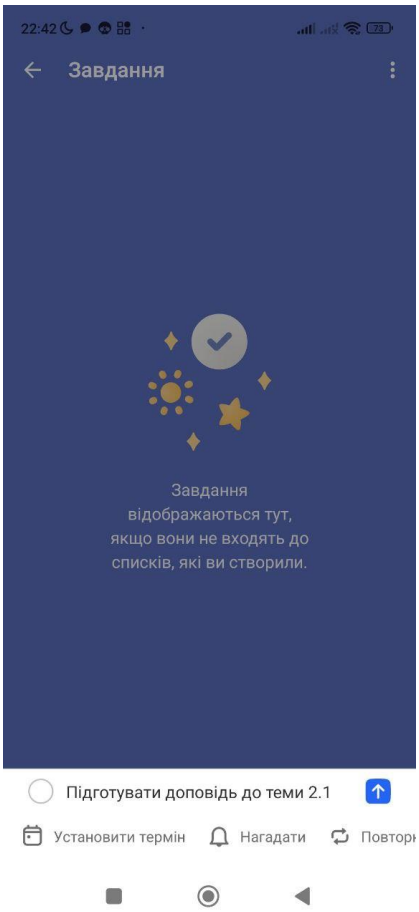

Рисунок 1.10 – Створено завдання

Після виконання завдання користувач робить позначення [4]. Варто відзначити, що задача зі списку не видаляється (рис. 1.11). Це є перевагою, оскільки у такому разі є можливість виконати порівняльний аналіз, стосовно того що вже виконано, а що – ні.

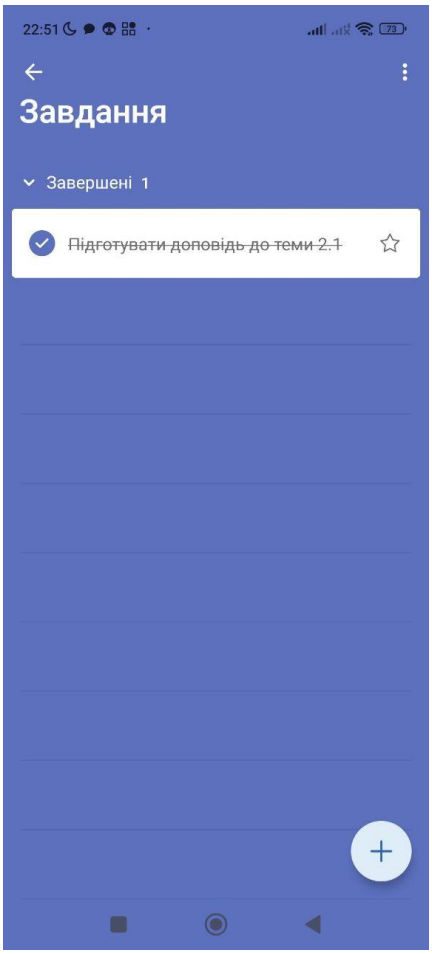

Рисунок 1.11 – Виконано завдання

Аналізуючи роботу із даним мобільним додатком варто відзачити його переваги: мінімалістичний дизайн, простота у використанні та зберігання задач після виконання. Серед недоліків варто згадати обов'язкову авторизацію через обліковий запис Microsoft (немає можливості використання акаунта Google, Facebook або Apple).

1.1.3. Чат-бот KindReminderBot

Серед телеграм-ботів лише декілька чатів призначені для планування та управління роботою. Одним із таких є чат-бот KindReminderBot від українських розробників Дмитра Дубілета, Андрія Ковтуна та Станіслава Шипцова.

За словами першого розробника: «Прямо в Telegram ви можете ставити завдання виконавцям. Причому робити це можна або в спільному чаті з колегами, або в самому боті. Кожне завдання має виконавця та термін виконання.

За день до терміну виконавцю приходить нагадування, що треба зробити звіт. Якщо звіт не вноситься, то бот нагадує про це щодня».

Для початку роботи із даним програмним продуктом потрібно авторизуватися в месенджері Telegram та в пошуку знайти KindReminderBot. Після відкриття чату користувач бачить привітання (рис. 1.12).

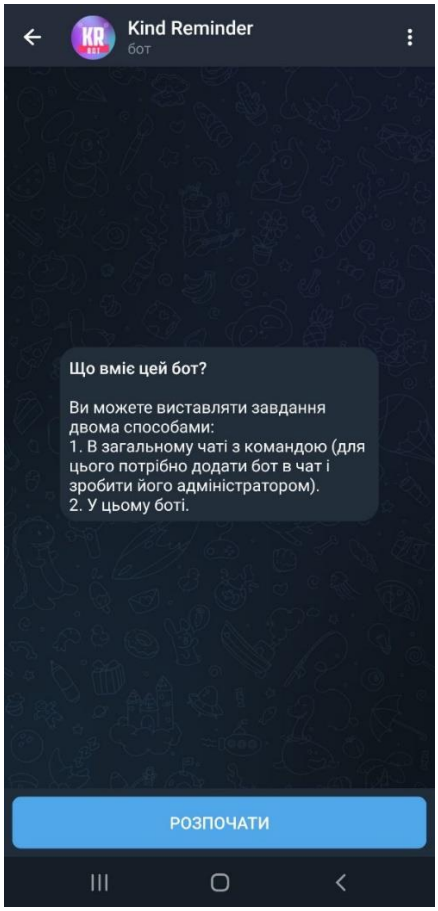

Рисунок 1.12 – Привітання від бота Kind Reminder

Процес роботи із даним рішенням легко налаштовується. Для цього потрібно натиснути кнопку Розпочати. Після цього з’являється детальна інструкція для роботи. Прочитавши її користувач без проблем може починати записувати задачі для виконання (рис. 1.13).

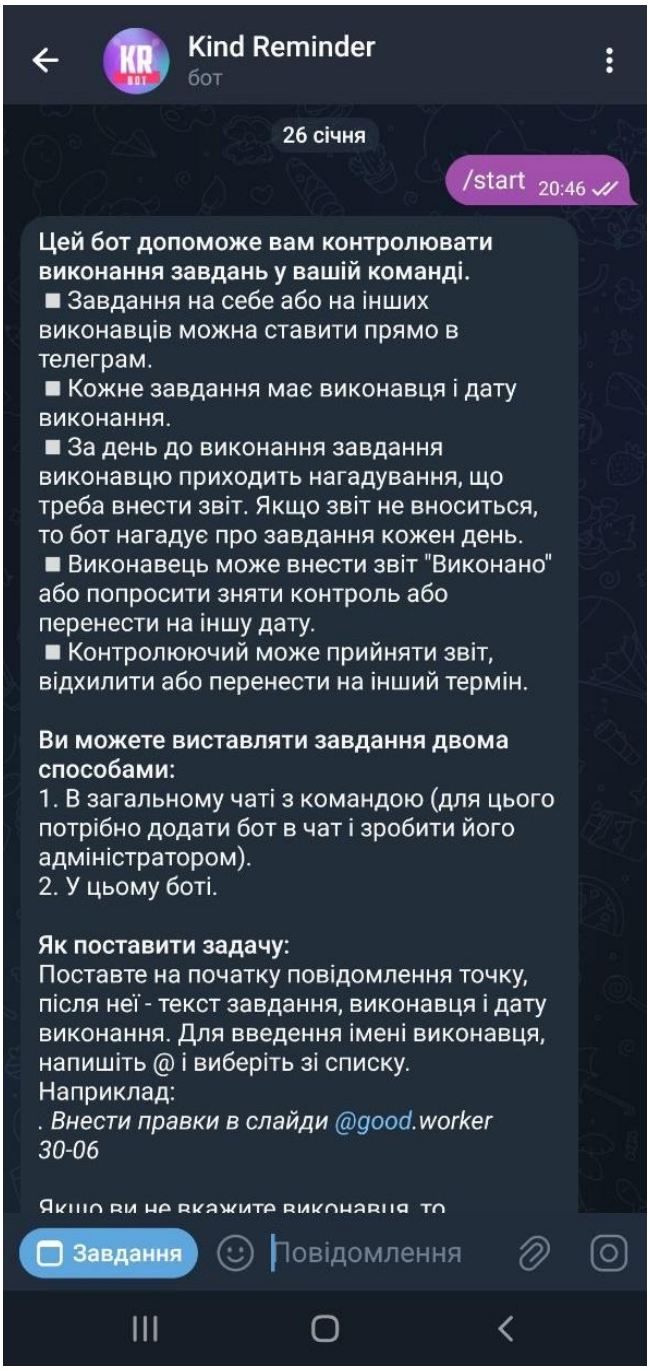

Рисунок 1.13 – Інструкція для користувача

Для перевірки зручності функціоналу потрібно увести задачу згідно вимог у форматі: *.Назва_задачі @Виконавець Кінцева_дата_виконання_задачі (у форматі дд-мм)*. Нижче продемонстровано роботу із уведенням задачі (рис. 1.14).

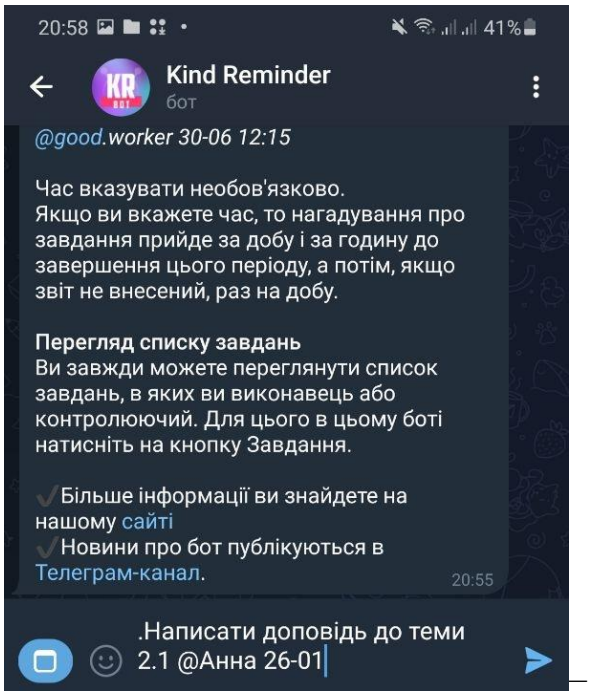

Рисунок 1.14 – Уведення задачі від користувача

Після відправлення задачі бот автоматично виведе інформацію. Для перевірки усього списку завдань потрібно натиснути у лівому нижньому куті на кнопку Завдання. В окремому вікні буде перелік (рис. 1.15).

Рисунок 1.15 – Перелік усіх задач

Для позначення про виконання користувачу слід натиснути на зелене коло, відповідно тоді – на Виконати [5]. У результаті у чаті із ботом буде повідомлення, що користувач завершив виконання задачі (рис. 1.16).

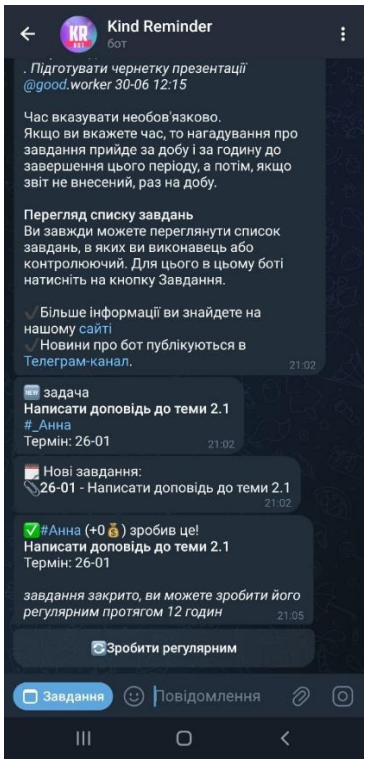

Рисунок 1.16 – Завершено виконання задачі

Аналізуючи роботу із даним продуктом варто відзначити, що простота та легкість у роботі це однозначно його перевага. Недоліком є те, що немає можливості редагувати завдання після відправки його боту, лише одноразово дозволяється увести задачу та відмітити її виконання.

У результаті аналізу існуючих рішень, було складено порівняльну таблицю параметрів, що допоможе під час розробки проєкту.

Таблиця 1.1 Порівняльна характеристика існуючих аналогів

Назва критерію	Назва ресурсу			
	Todoist	Microsoft To Do: Lists & Tasks	KindReminder	Telegram-бот для управління задачами
Обов'язкова авторизація	+	+	+	+
Зручний та зрозумілий інтерфейс	+	-	+	+
Можливість управляти задачами	-	-	+	+
Можливість редагувати задачі у ході їх виконання	+	+	-	+

1.2. Постановка задачі

У результаті проведеного аналізу, поставлено мету роботи – проєктування та розробка Telegram-бота зі зручним інтерфейсом користувача та швидким доступом до всієї інформації. Для досягнення даної мети потрібно завершити такі завдання:

1) створення MindMap для проєктування боту;
2) реєстрація бота в системі;
3) вибір засобів програмування для створення бота;
4) розробка наповнення боту;
5) проведення тестування створеного програмного продукту.

2 ВИБІР ОСНОВНИХ КОМПОНЕНТІВ БОТУ

2.1. Створення MindMap для проєктування боту

Для відображення усього функціоналу боту та роботи кожного модуля окремо відобразимо усе на схемі – MindMap. Mind Map — діаграма, на який зображені елементи і зв'язки між ними [6]. Усе представлення виконано у вигляді деревовидної структури. Основа діаграми зв’язків зображено на рис. 2.1.

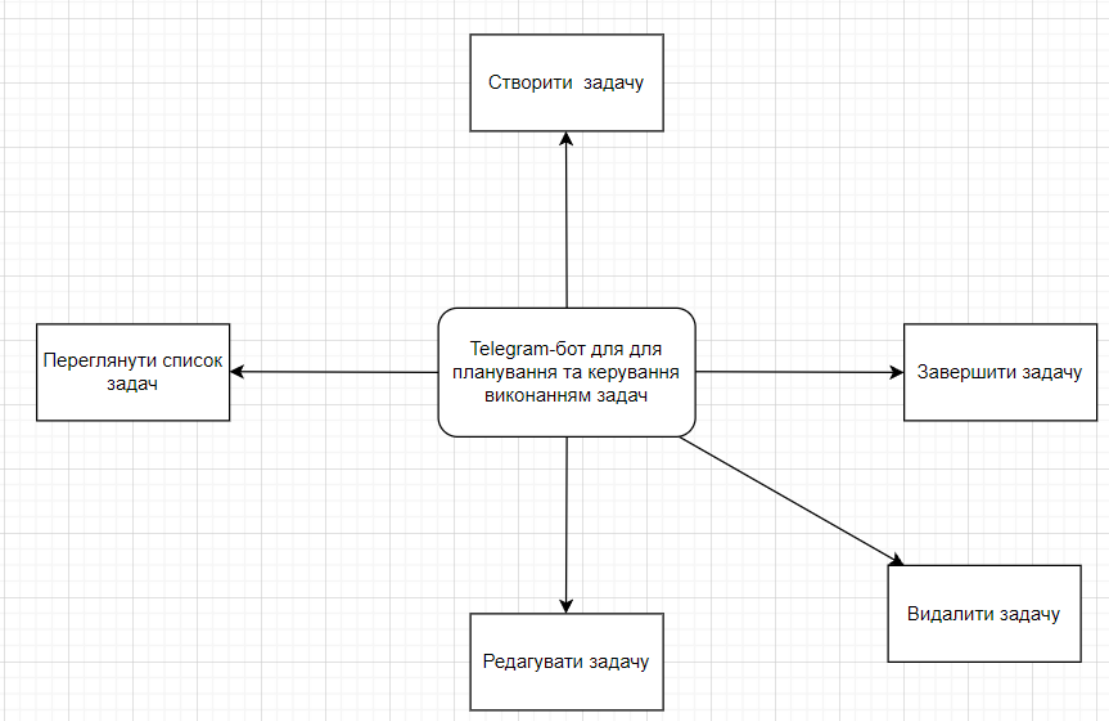

Рисунок 2.1 – Діаграма зв’язків

Користувач починає роботу з ботом з ключового слова /start. Після даної команди з’являється вітальне повідомлення та перелік команд для роботи, які наведені на рис. 2.1.

Для уведення списку задач використовується функція «Створити задачу». Механізм роботи її полягає в наступному: користувач натискає на кнопку «Створити задачу». У відповідь на це бот виводить повідомлення: «Опишіть задачу». Юзер друкує свою задачу. Після цього інформація вноситься до бази даних та виводиться на екран перелік. Даний принцип роботи відображено на рис. 2.2.

Рисунок 2.2 – Принцип роботи функції «Створити задачу»

Для виведення списку задач використовується функція «Список задач». Механізм роботи її полягає в наступному: користувач натискає на кнопку «Список задач». У відповідь на це бот виводить інформацію на екран про список задач для виконання із бази даних. Даний принцип роботи відображено на рис. 2.3.

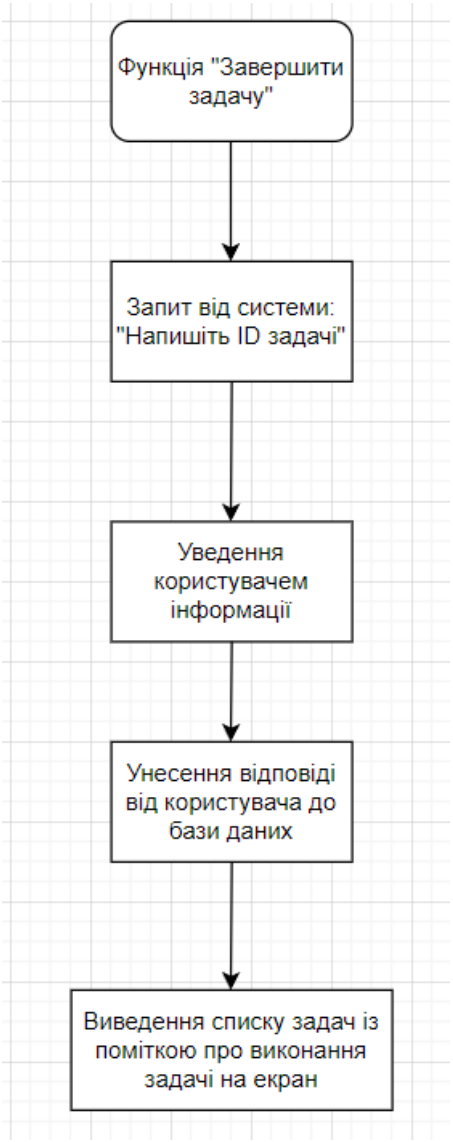

Рисунок 2.3 – Принцип роботи функції «Список задач»

Для редагування задач використовується функція «Редагувати задачу». Механізм роботи її полягає в наступному: користувач натискає на кнопку «Редагувати задачу». У відповідь на це виводить повідомлення на екран «Напишіть ID задачі на нову назву задачі». Наступним кроком юзер повинен увести ID задачі на нову назву задачі. У результаті на екран буде виведено відредагований список та змінено дані у базі даних. Даний принцип роботи відображено на рис. 2.4.

Рисунок 2.4 – Принцип роботи функції «Редагувати задачу»

Для видалення задачі використовується функція «Видалити задачу». Програмно окремо передбачено процес завершення задачі (задача залишається у списку, але з іншою позначкою) та видалення задачі (у такому разі взагалі видаляється інформація зі списку та бази даних). Для повноцінного видалення потрібно натиснути на кнопку «Видалити задачу». У відповідь система виведе повідомлення «Напишіть ID задачі». Користувачу необхідно написати ID. У разі правильного введення на екрані з’явиться список задач після видалення необхідної. Даний принцип роботи відображено на рис. 2.5.

Рисунок 2.5 – Принцип роботи функції «Видалити задачу»

Для завершення задачі використовується функція «Завершити задачу». Механізм її роботи полягає в наступному: користувач натискає на кнопку «Завершити задачу». З’являється запит від системи «Напишіть ID задачі». Після цього необхідно увести ID. У результаті виведеться список задач, біля відповідної команди зміниться позначка про виконання. Також зміниться дана інформація у базі даних. Даний принцип роботи відображено на рис. 2.6.

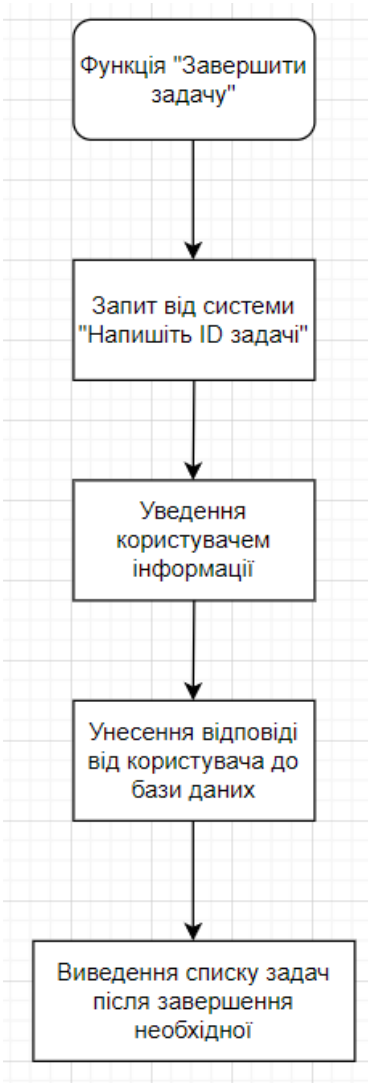

Рисунок 2.6 – Принцип роботи функції «Завершити задачу»

2.2. Реєстрація бота в системі

Для реалізації роботи бота в месенджері Telegram потрібно його зареєструвати через головний (батьківський) бот у системі. Процес максимально автоматичний. Алгоритм реєстрації наведено на рисунках 2.7 – 2.12.

У месенджері Telegram передбачено окремий акаунт @BotFather для створення та реалізації ботів. Дане рішення значно полегшує процес практичної реалізації у месенджері для інших користувачів. Чат-бот — це автоматизований багатофункціональний помічник, який може збирати та показувати інформацію підписникам за їхнім запитом згідно зі заздалегідь підготовленим сценарієм [7]. На рис. 2.1 наведено процес початку роботи з ним (коротке привітання).

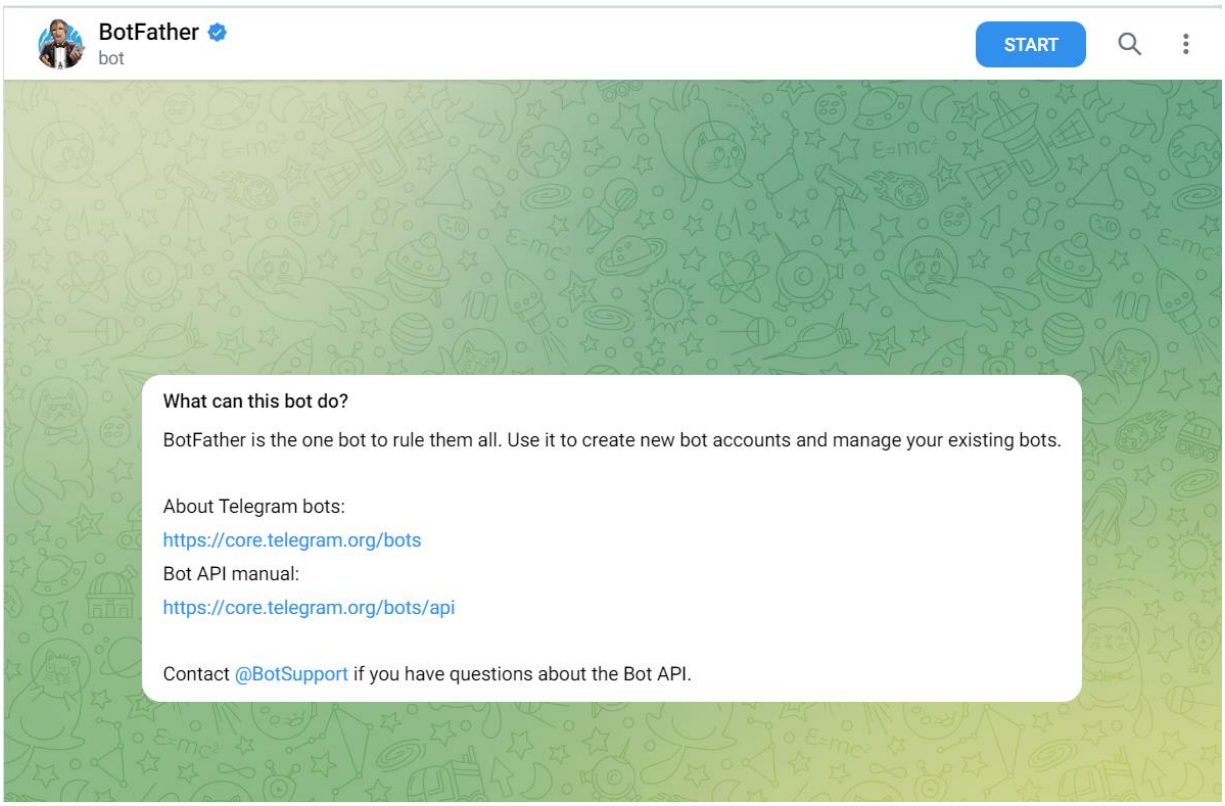

Рисунок 2.7 – Початок роботи з BotFather

Для початку спілкування потрібно увести команду /start вручну або скористатись кнопками швидкого набору (рис. 2.8). Після цього з'являється перелік основних команд та короткий опис їх функціоналу. Так як потрібно створити нового бота, то потрібно увести команду /newbot (рис. 2.9).

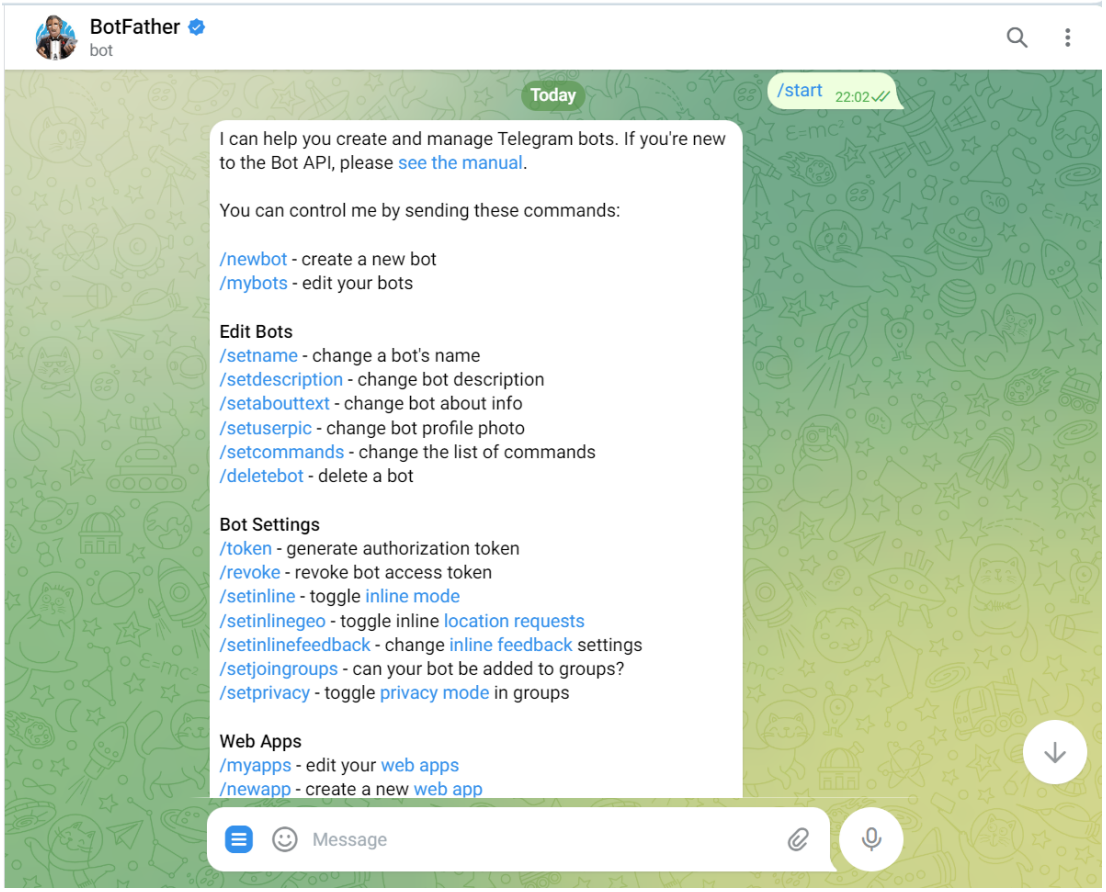

Рисунок 2.8 – Початок спілкування з помічником

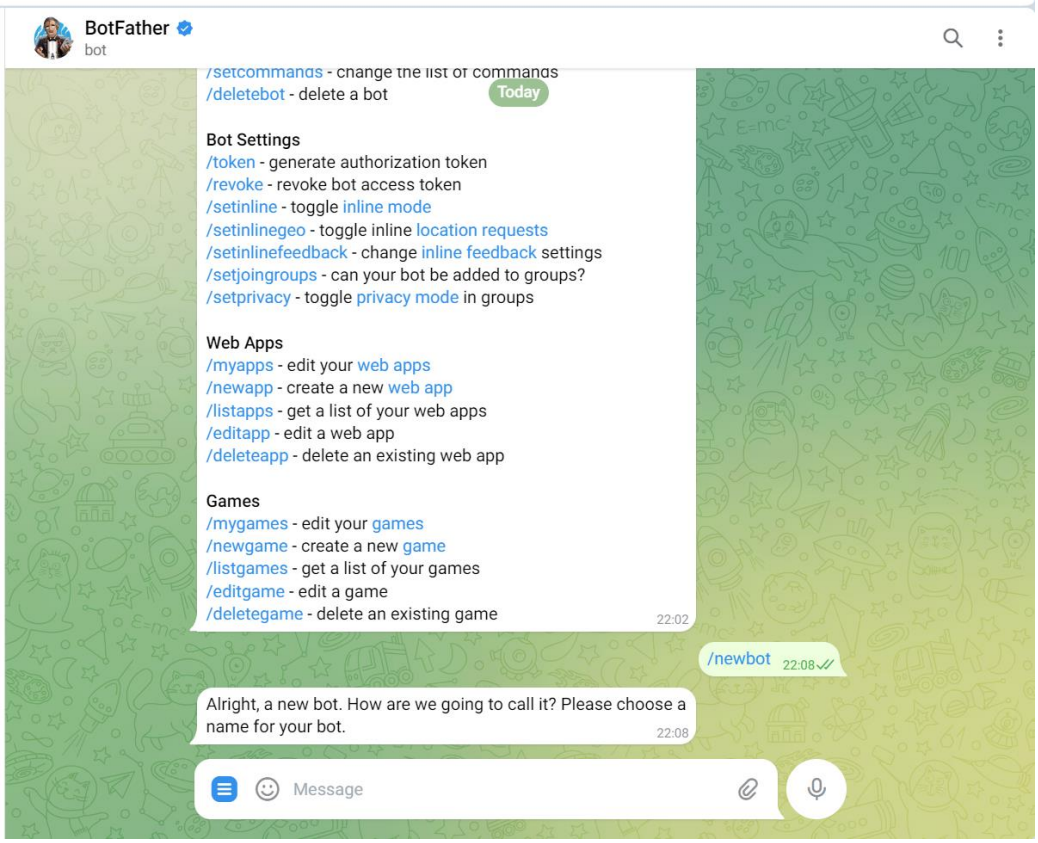

Рисунок 2.9 – Команда для створення нового бота

Наступним кроком потрібно увести назву нового програмного продукту. Мною вирішено назвати його TaskCommander. Юзернейм повинен закінчуватися на ‘bot’ (рис. 2.10).

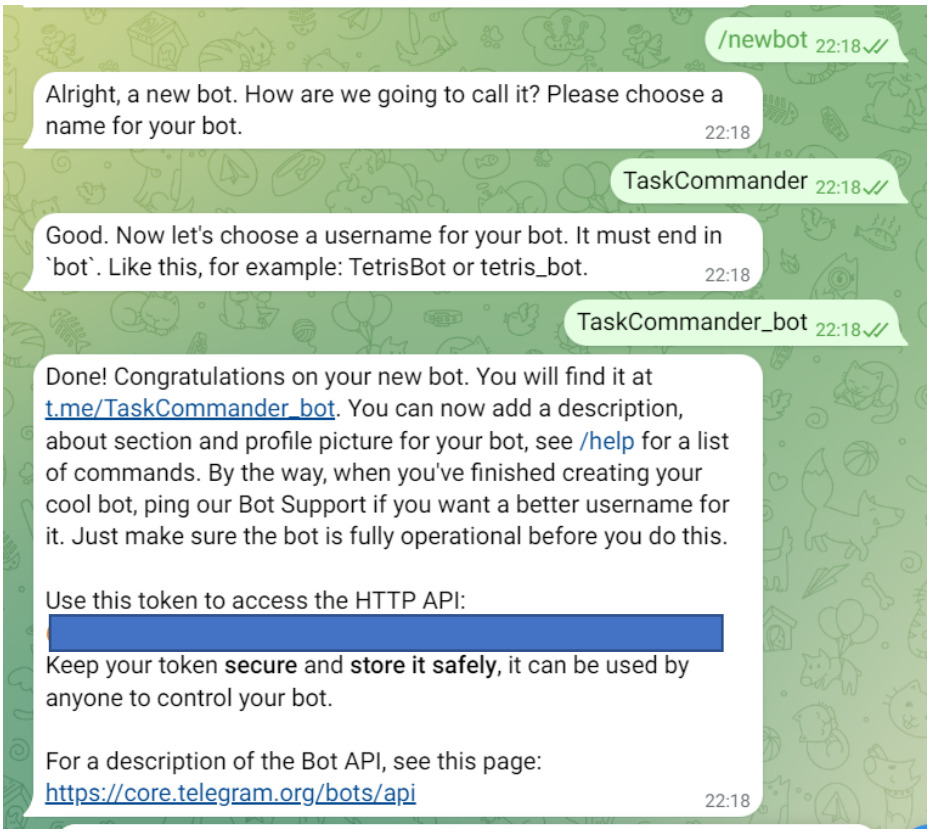

Рисунок 2.10 – Уведення назви та юзернейму для програмного продукту

У результаті отримано унікальний токен, що дає доступ до HTTP API. Токен – це ключ, який складається з набору символів, та використовується для зв`язку webhook-у та серверу месенджера Telegram [8]. Різниця між ним та звичайним паролем полягає у наступному: цифровий ключ постійно обновлюється (генерується нова послідовність символів) і, як наслідок, токен може знати лише розробник бота у момент його авторизації.

HTTP протокол використовується не тільки для передачі гіпертекстових документів, але і для передачі зображень та відео або для відправки контенту на сервери [9].

API — це набір компонентів, за допомогою яких одна комп’ютерна програма (бот або сайт) може використовувати іншу. Тобто це механізми, які

дозволяють двом програмним компонентам взаємодіяти один з одним, використовуючи набір визначень та протоколів [10].

Тепер залишається додати короткий опис для зручності інформування нового користувача про призначення та функціонал бота. Для цього потрібно увести команду /setdescription та обрати ім’я бота. Після цього уводжу короткий опис (рис. 2.11).

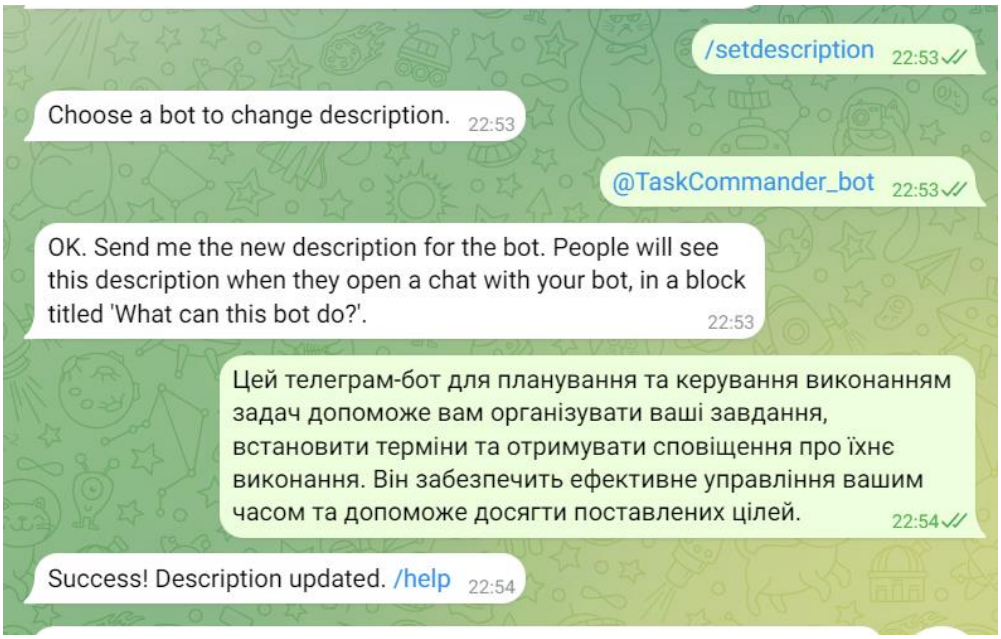

Рисунок 2.11 – Уведення короткого опису для бота

Останнім кроком слід обрати та встановити зображення профілю. Даний крок реалізовується уведенням команди /setuserpic та вибору назви бота. Результат налаштування наведено на рис. 2.12.

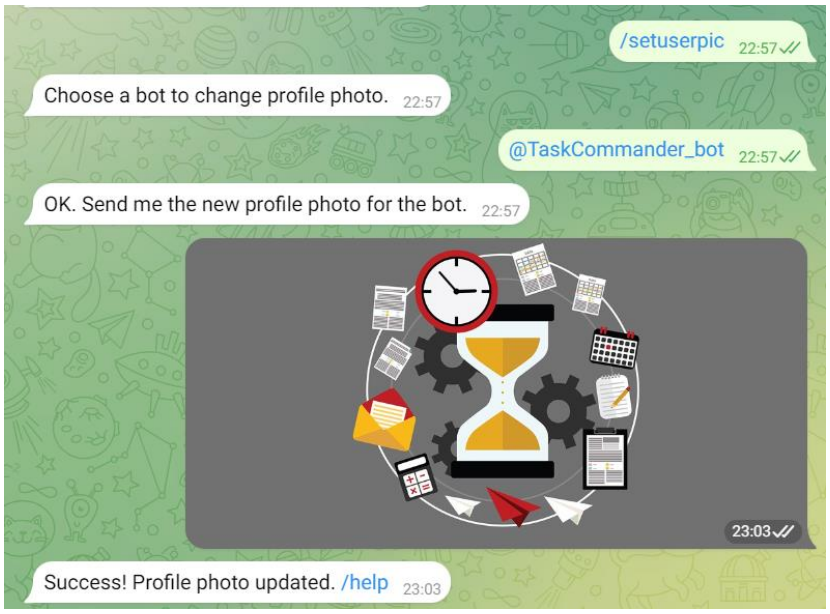

Рисунок 2.12 – Налаштування фото профілю бота

2.3. Вибір засобів програмування для створення бота

Існує широкий спектр використання мов програмування для створення Telegram-бота. Згідно дослідження Івана Юркевича у статті «Яка у Telegram мова програмування?», Telegram-API підтримує такі мови програмування: Python, Java, JavaScript, PHP, Ruby, Swift, Kotlin [11]. Використання кожної із них оптимально для специфічних задач.

Нижче детально проаналізовано кожну із вижчезазначених мов.

1. Перевагою використання мови програмування Python є легкість вивчення мови (Python вважається однією з найбільш доступних мов програмування, особливо для початківців. Його зрозумілий синтаксис і чистий код дозволяють швидко вчитися і створювати програми), широка спільнота та підтримка (Python має велику та активну спільноту розробників, що означає наявність багатьох ресурсів, документації та форумів для отримання допомоги) та готові бібліотеки для роботи з API телеграма (у Python існують багато готових бібліотек, таких як python-telegram-bot, які спрощують розробку телеграм-ботів). Недоліком даної мови є відносно великий час виконання програми (Python може бути повільнішим у порівнянні з іншими мовами програмування, такими як Java або C++, через інтерпретовану природу мови).

2. Перевагою використання мови програмування JavaScript (фреймворк Node.js) є велика можливість JavaScript (JavaScript є однією з найпопулярніших

мов програмування, а Node.js дозволяє виконувати JavaScript на сервері, що робить його привабливим вибором для створення веб-застосунків та серверних програм, також і для телеграм-ботів) та простота розгортання (за допомогою платформ хостингу, таких як Heroku або AWS Lambda, можна легко розгорнути та використовувати телеграм-бота на Node.js). Недоліком є необхідність в додаткових етапах налаштування.

3. Перевагою використання мови програмування Java є велика ефективність (Java відома своєю високою продуктивністю та ефективністю, що робить її привабливим вибором для розробки великих та надійних систем), потужна система типізації (система типізації Java допомагає уникнути багатьох помилок на етапі розробки, що робить код більш надійним) та широкий вибір сторонніх бібліотек для роботи з API телеграма. Недоліком є складніший вивід на продакшн (розгортання Java-застосунків може бути складнішим порівняно з іншими мовами).

4. Перевагою використання мови програмування Ruby є чистий синтаксис (Ruby має чистий та легко зрозумілий синтаксис, що сприяє швидкому розвитку програм) та фреймворк Rails (фреймворк Ruby on Rails має багато вбудованих інструментів, що дозволяють швидше розробляти та розгортати веб-додатки, включаючи телеграм-ботів). Недоліком є відносно невелика кількість сторонніх бібліотек: Ruby може мати меншу кількість сторонніх бібліотек порівняно з іншими мовами, що може ускладнити розробку деяких функцій.

5. Перевагою використання мови програмування PHP є широке поширення та підтримка хостингів (PHP є однією з найпоширеніших мов програмування для веб-розробки, і багато хостингів підтримують PHP) та наявність декількох бібліотек для роботи з API телеграму (наявність деяких бібліотек, таких як Telegram Bot API, які спрощують роботу з API телеграма в PHP). Недоліком використання є статичність типізації та менша ефективність (PHP має статичну систему типізації та може бути менш ефективним порівняно з іншими мовами, такими як Java або C++).

Вищезазначені твердження узагальнено та систематизовано у таблиці 2.1.

Таблиця 2.1 Порівняльний аналіз мов програмування для розробки Telegram-боту

Мова програмування	Переваги	Недоліки
Python	Легко вивчити Широко підтримується у спільноті Багато готових бібліотек для роботи з API телеграма.	Час виконання може бути повільним у порівнянні з деякими іншими мовами.
JavaScript (фреймоврк Node.js)	Потужність JavaScript та екосистема Node.js Простота розгортання за допомогою платформи хостингу, такої як Heroku або AWS Lambda	Потребує більше додаткових кроків для встановлення та налаштування.
Java	Велика ефективність Потужна система типізації Широкий вибір сторонніх бібліотек для роботи з API телеграма	Складніший вивід на продакшн порівняно з іншими мовами.
Ruby	Чистий синтаксис Rails має деякі корисні вбудовані інструменти для розробки ботів	Менша кількість сторонніх бібліотек порівняно з іншими мовами.
PHP	Широко поширений та підтримується хостингами Є деякі бібліотеки, такі як Telegram Bot API для спрощення роботи з API телеграма	Статичність типізації та менша ефективність порівняно з іншими мовами

Аналізуючи усю вищезазначену інформацію із табл. 2.1 вирішено для розробки Telegram-боту обрати мову програмування JavaScript.

Отже, у результаті виконання вищенаведених кроків було створено і налаштовано у BotFather телеграм-бот для планування та керування виконанням задач та обрано засіб програмування для створення бота. Наступним кроком буде вибір середовища програмування та програмна реалізація.

3 КОМП'ЮТЕРНА РЕАЛІЗАЦІЯ ПРОЄКТУ ТА ТЕСТУВАННЯ

3.1. Проєктування бази даних

Після того як зареєстровано бот у системі Telegram через Father-bot, створено Mind-map, на якій відображено зв'язки основних блоків, та обрано мову програмування потрібно розробити місце для зберігання інформації, отриманої від користувача. База даних Telegram-боту TaskCommander складається із 1 таблиці Tasks (рис. 3.1).

Tasks	
PK	Id_tasks
	Name (varchar(255)) Status(boolean)

Рисунок 3.1 – ERD-діаграма Telegram-бота

Дана таблиця має 3 поля: Id-tasks (порядковий номер задачі, ключове поле PK), Name (назва + короткий опис завдання від користувача, тип даних – varchar(255)), Status (значення виконана задача чи ні, тип даних boolean). Саме по значенню стовпця Status будуть відображатися мітки про статус виконання.

3.2. Програмна реалізація бота

Для програмної реалізації, як було вище зазначено, обрано мову програмування JavaScript та її фреймворк Nest.js, середовище програмування – WebStorm від компанії Intellij Idea. Перевагою використання фреймворку Nest.js є виконання коду поза браузером, яке дозволяє писати серверний код для веб-сторінок і веб-застосунків, а також для програм командного рядка [12].

Telegram-бот TaskCommander виконано за наступною структурою:

app.buttons.ts – файл для налаштування кнопок зв'язку користувача та бота під час роботи;

app.module.ts – файл для під'єднання до бази даних PostrgeSQL;

app.service.ts – файл для організації методів для редагування, видалення та завершення виконання завдання за запитом користувача;

app.update.ts – файл для організації роботи боту. У ньому використовуються файли app.buttons.ts та app.service.ts;

app.utils.ts – файл для виведення списку задач;

config.ts – файл для збереження змінної, яка містить інформацію про API боту. Дана інформація видається при реєстрації у Father-bot та є конфіденційною.

Спершу потрібно у файлі app.update.ts прописати команду Start, саме з неї починається користування додатком (рис. 3.2).

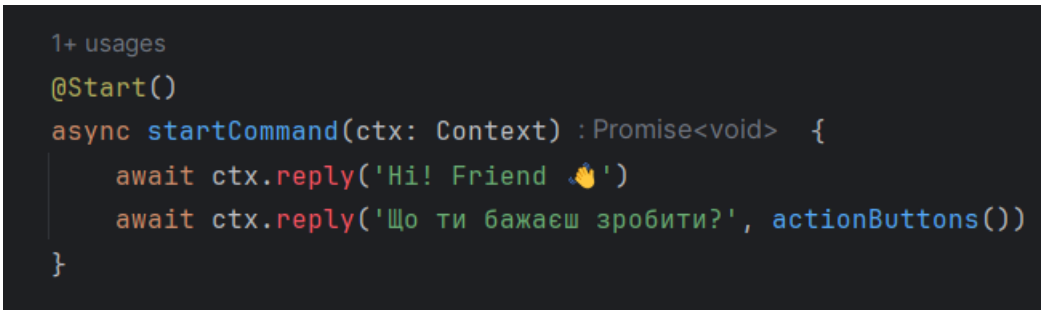

Рисунок 3.2 – Створення команди Start

Наступним етапом потрібно розробити кнопки «Створити задачу», «Список задач», «Завершити», «Редагування», «Видалення», за допомогою яких буде відбуватися взаємодія з користувачем. Для цього у файлі app.buttons.ts створюються наступні рядки коду (рис. 3.3).

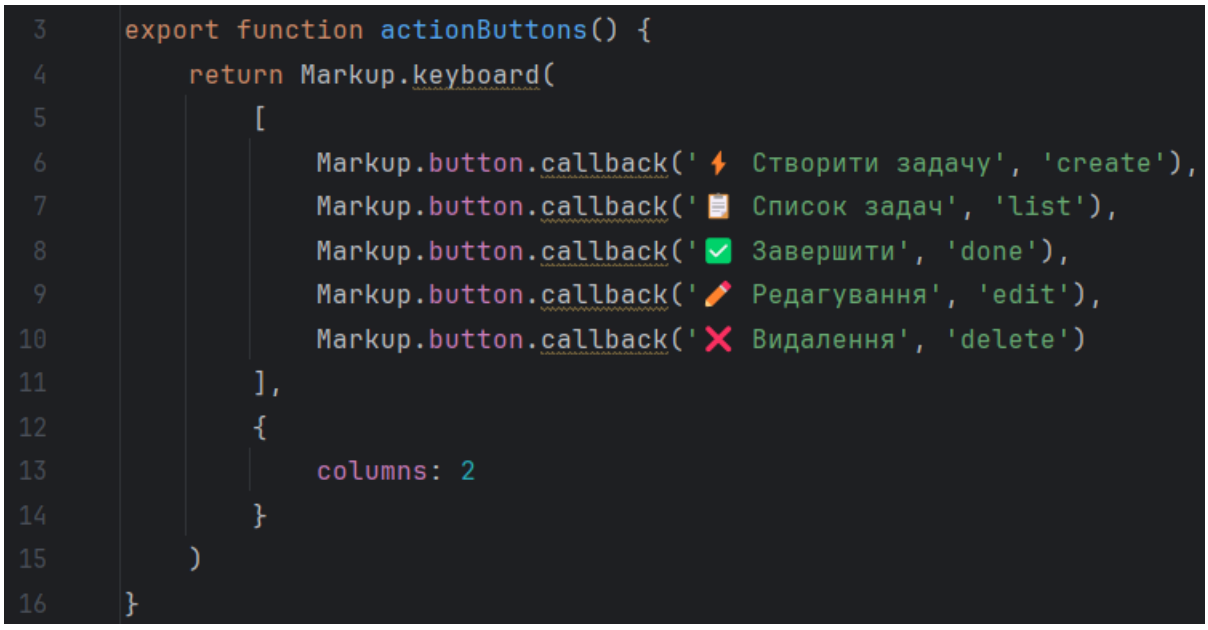

Рисунок 3.3 – Створення кнопок для боту

На рис. 3.3 використано вбудовану функцію callback. У якості аргументів вона приймає 2 значення: напис, який буде виводитися на кнопці, та внутрішній ідентифікатор для використання необхідної функції (для внесення задачі у БД, виведення списку задач, завершення, редагування та видалення). Дані ідентифікатори будуть використані у файлі app.update.ts при створенні та налаштуванні окремо функції.

Безпосередня робота кожної із 5 вищезазначених функцій розміщена у файлі app.update.ts. Спочатку потрібно імпортувати дефолтний клас AppUpdate та створити конструктор (рис. 3.4).

```
16  @Update()
17  export class AppUpdate {
        no usages
18      constructor(
19          @InjectBot() private readonly bot: Telegraf<Context>,
20          private readonly appService: AppService
21      ) {}
```

Рисунок 3.4 – Імпортування класу AppUpdate та створення конструктора

Нижче описуємо функціонал кожної із кнопок, створюємо однойменні функції (рис. 3.5-3.6).

```
29      @Hears('⚡ Створити задачу')
30      async createTask(ctx: Context) : Promise<void>  {
31          ctx.session.type = 'create'
32          await ctx.reply('Опиши задачу: ')
33      }
34
        1+ usages
35      @Hears('📋 Список задач')
36      async listTask(ctx: Context) : Promise<void>  {
37          const todos = await this.appService.getAll()
38          await ctx.reply(showList(todos))
39      }
40
        1+ usages
41      @Hears('✅ Завершити')
42      async doneTask(ctx: Context) : Promise<void>  {
43          ctx.session.type = 'done'
44          await ctx.deleteMessage()
45          await ctx.reply('Напиши ID задачі: ')
46      }
47
```

Рисунок 3.5 – Створення функцій для кнопки «Створити задачу», «Список задач», «Завершити»

```
48      @Hears('🖍 Редагування')
49      async editTask(ctx: Context) : Promise<void>  {
50          ctx.session.type = 'edit'
51          await ctx.deleteMessage()
52          await ctx.replyWithHTML(
53              'Напиши ID і нову назву задачі: \n\n' +
54                  'У форматі - <b>1 | Нова назва</b>'
55          )
56      }
57
        1+ usages
58      @Hears('❌ Видалення')
59      async deleteTask(ctx: Context) : Promise<void>  {
60          ctx.session.type = 'remove'
61          await ctx.deleteMessage()
62          await ctx.reply('Напиши ID задачі: ')
63      }
64
```

Рисунки 3.6 – Створення функцій для кнопки «Редагування», «Видалення»

Для функції «Редагування», «Видалення» та «Завершити» потрібно передбачити додаткову перевірку на наявність значення ID (порядкового номеру) задачі, так як може не бути запису під таким номером. Для цього використовується оператор розгалудження if – else (рис. 3.7-3.8).

```
65      @On('text')
66      async getMessage(@Message('text') message: string, @Ctx() ctx: Context) : Promise<void>  {
67          if (!ctx.session.type) return
68
69          if (ctx.session.type === 'create') {
70              const todos = await this.appService.createTask(message)
71              await ctx.reply(showList(todos))
72          }
73
74          if (ctx.session.type === 'done') {
75              const todos = await this.appService.doneTask(Number(message))
76
77              if (!todos) {
78                  await ctx.deleteMessage()
79                  await ctx.reply('Задачі з таким ID не знайдено!')
80                  return
81              }
82
83              await ctx.reply(showList(todos))
84          }
```

Рисунок 3.7 – Перевірка наявності ID задачі для позначення задачі про завершення

```
        if (ctx.session.type === 'edit') {
            const [taskId : string , taskName : string ] = message.split( separator: ' | ')
            const todos = await this.appService.editTask(Number(taskId), taskName)

            if (!todos) {
                await ctx.deleteMessage()
                await ctx.reply('Задачі з таким ID не знайдено!')
                return
            }

            await ctx.reply(showList(todos))
        }

        if (ctx.session.type === 'delete') {
            const todos = await this.appService.deleteTask(Number(message))

            if (!todos) {
                await ctx.deleteMessage()
                await ctx.reply('Задачі з таким ID не знайдено!')
                return
            }

            await ctx.reply(showList(todos))
        }
    }
}
```

Рисунок 3.8 – Перевірка наявності ID задачі для редагування та видалення задачі зі списку

Отримані дані від користувача вносяться у файл session_db.json, який має наступну структуру:

Id_tasks – primary key, порядковий номер

Name – varchar (255), опис задачі

Status – boolean, статус виконання

Файли із програмним кодом знаходяться у додатках А-Е.

3.3. Тестування Telegram-боту

Після завершення розробки та публікації бота було виконано його тестування розробником та сторонніми користувачами. У результаті тестування помилок не було знайдено.

Для початку роботи з додатком потрібно у пошуковому полі месенджера Telegram написати TaskCommander (рис. 3.9).

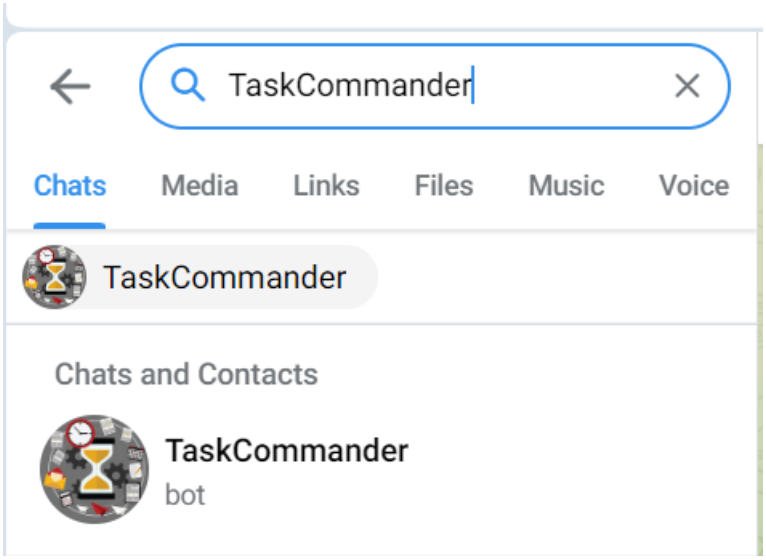

Рисунок 3.9 – Результат пошуку бота TaskCommander

При відкритті з’являється коротке повідомлення, яке містить інформацію ознайомлювального характеру, аби пояснити призначення програмного продукту користувачу (рис. 3.10). Для початку необхідно натиснути на кнопку «Start».

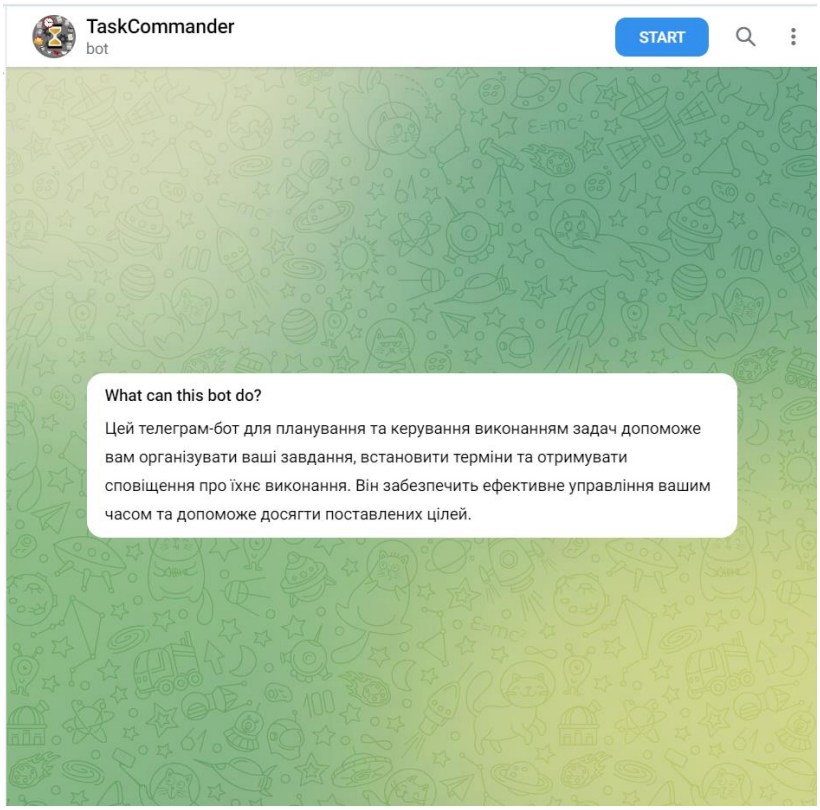

Рисунок 3.10 – Коротке вікно перед початком роботи із ботом

Користувачу потрібно обрати 1 із 5 кнопок для роботи на вибір (рис. 3.11).

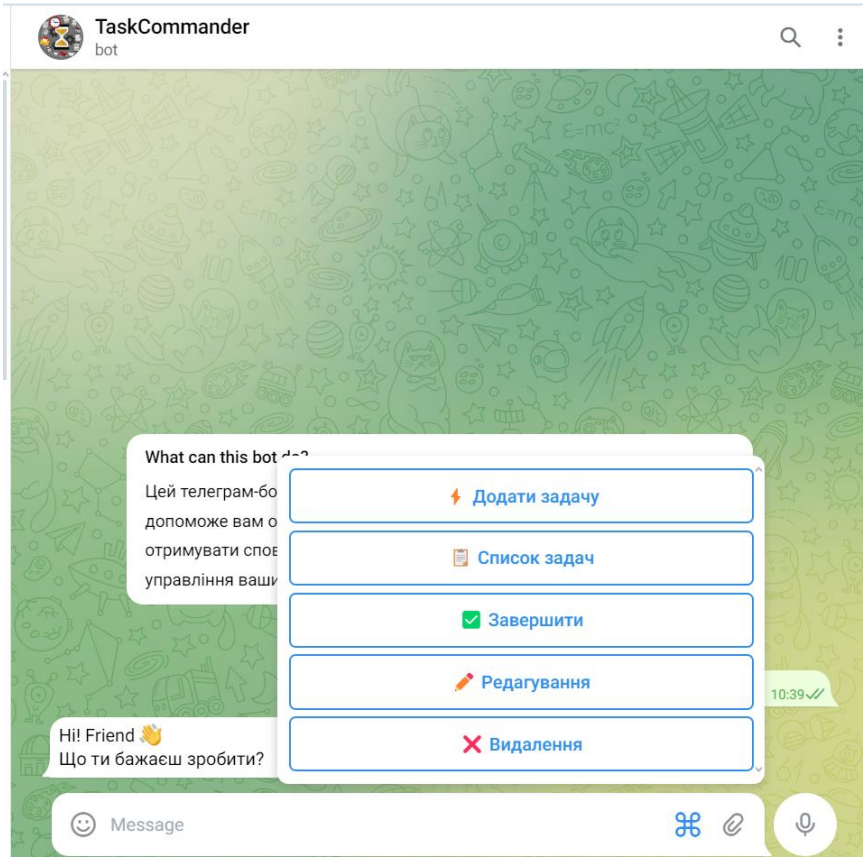

Рисунок 3.11 – Вибір кнопок для роботи

Для додавання задачі потрібно натиснути кнопку «Додати задачу». Після цього увести назву завдання та натиснути кнопку «Надіслати». Приклад наведено на рис. 3.12. Уведена інформація вноситься до бази даних.

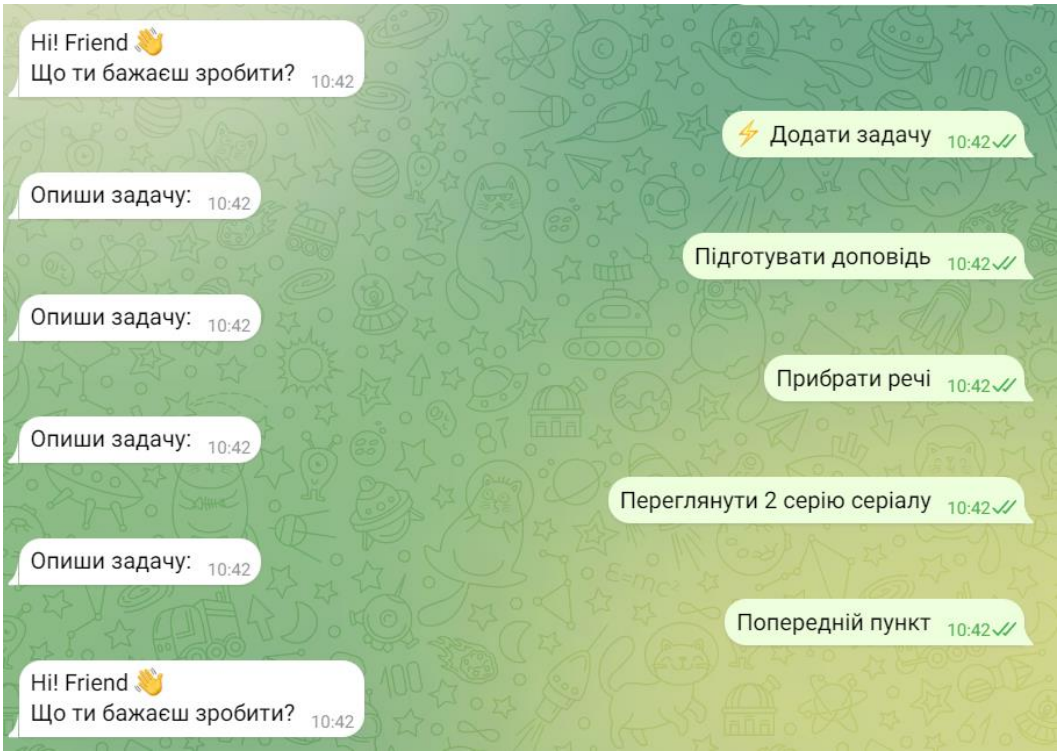

Рисунок 3.12 – Додавання задач

Після того як усі задачі додано, то у користувача є можливість переглянути список задач. Для цього необхідно натиснути кнопку «Переглянути список» (рис. 3.13). Даному етапі не повинно виникати ніяких помилок, оскільки система виводить інформацію, отриману від користувача. Так як жодна задача не відмічена користувачем як виконана, то перед її назвою позначка у вигляді білого круга.

Рисунок 3.13 – Виведення списку задач

Для позначення про завершення задачі користувачу необхідно натиснути на кнопку «Завершити», попередньо повернувшись у Головне меню. Для перевірки чи відбулись зміни у списку задач потрібно повернутись у Головне меню та натиснути на кнопку «Список задач». На рис. 3.14 наведено перевірка роботи кнопки «Завершити».

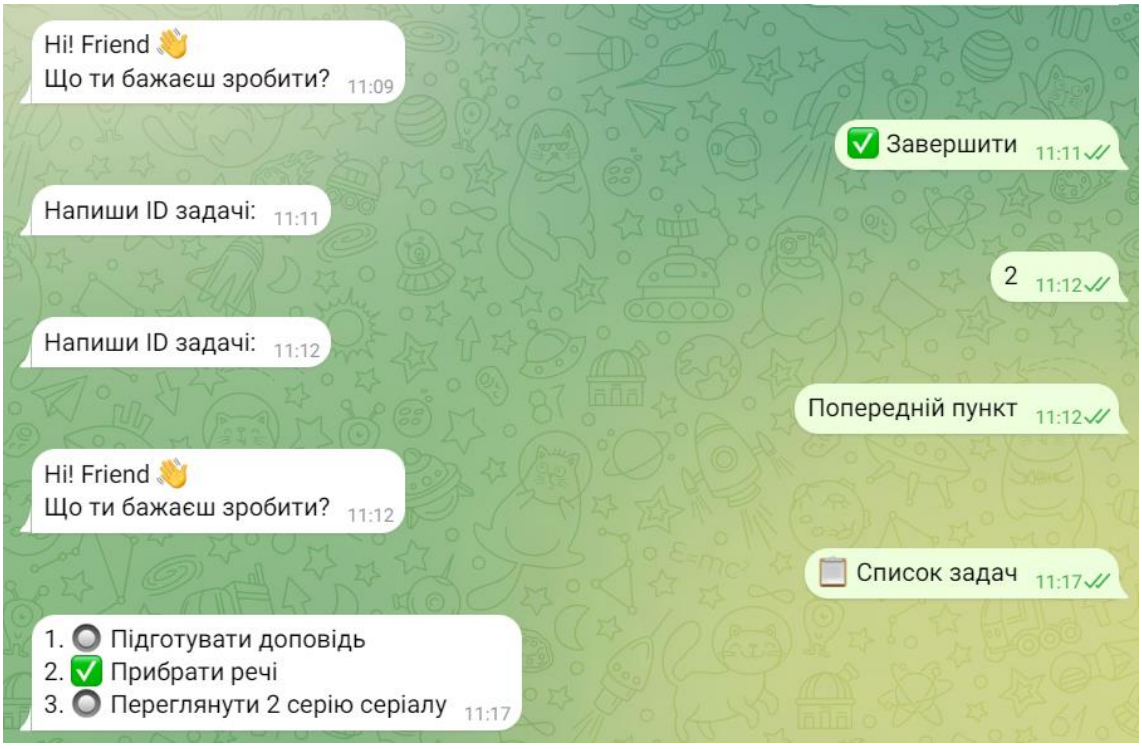

Рисунок 3.14 – Перевірка кнопки «Завершити»

Якщо користувач уведе значення ID більше, ніж кількість задач або будь-який інший формат крім, то система виведене інформацію про помилку (рис. 3.15).

Рисунок 3.15 – Уведення значення ID більшого, ніж є в системі

У такому разі необхідно знову увести порядковий номер та повторно переглянути список задач.

Для редагування списку задач потрібно натиснути на кнопку «Редагувати». Алгоритм роботи частково схожий на алгоритм роботи із кнопкою «Завершити»: користувачу необхідно обрати дану кнопку та увести у відповідному форматі нову назву задачі. Результат роботи зображено на рис. 3.16.

Рисунок 3.16 – Результат роботи кнопки «Редагувати»

Після уведення нової задачі потрібно переглянути увесь список. Задача змінена – отже працює додаток коректно (рис. 3.17).

Рисунок 3.17 – Список змінений після редагування

Помилка може виникнути, якщо користувач уводить ID задачі, якого не існує (більше, ніж кількість задач є в базі даних або некоректне значення: з буквою чи символом). У такому разі потрібно повторити спробу уведення, але вже із виправленою помилкою.

Для видалення задачі потрібно натиснути на кнопку «Видалити». Дана операція необхідна у тому випадку, якщо задача уже виконана і не потрібно, щоб заважала зайва інформація, або змінилися плани користувача та не потрібно її виконувати. У такому разі необхідно натиснути на кнопку «Видалити». Після натискання система пропонує увести ID задачі. Необхідно увести порядковий номер та переглянути повторно список задач (рис. 3.18).

Рисунок 3.18 – Видалення задачі зі списку

Під час роботи може виникнути помилка, якщо користувачем буде уведено некоректне значення порядкового номера або задачі з таким ID не існує в системі. Система повторно виведе повідомлення і потрібно увести коректне значення порядкового номеру.

ВИСНОВКИ

У процесі виконання даної кваліфікаційної бакалаврської роботи було створено Telegram-бота для організації та планування виконання списку справ користувача. Метою проєкту було створити додаток у популярному месенджері. Основним завданням було створення комфортного та інтуїтивно зрозумілого інтерфейсу для організації вирішення справ.

У процесі розробки реалізовано кілька етапів. На першому етапі виконано аналіз вимог та огляд готових рішень. Другий етап полягав у створенні карти-проєкту (MindMap) та реєстрації бота в системі за допомогою чату FatherBot у месенджері Telegram. На третьому етапі було обрано засіб програмування для створення бота, порівняно та обрано мову програмування, спроєктовано та зареєстровано базу даних у системі, безпосередньо програмно реалізовано чат-бот та протестовано після публікації у відкритому доступі. Для тестування обиралось ручне (мануальне) тестування, адже це найшвидший спосіб перевірити коректну роботу розробленого додатку.

Після успішного завершення тестування було завантажено та надано доступ до створеного програмного продукту усім бажаючим із мережі Telegram.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сайт SendPulse: Що таке чат-бот. [Електронний ресурс]. – Режим доступу: https://sendpulse.ua/support/glossary/chatbot. 20.01.2024
2. Сайт Todoist: Авторизація. [Електронний ресурс] – Режим доступу: https://todoist.com/. 25.01.2024
3. Сайт Політехнічний ліцей НТУУ «КПІ» м. Києва: Візуал, аудіал, кінестетик, дискрет – хто це? Або особливості сприйняття інформації. [Електронний ресурс] – Режим доступу: https://pl.kpi.ua/2098-2/. 25.01.2024
4. Сайт Microsoft: Програма Microsoft To Do. [Електронний ресурс] – Режим доступу: https://www.microsoft.com/uk-ua/microsoft-365/microsoft-to-do-list-app. 25.01.2024
5. Сайт MC.Today: Дмитро Дубілет запустив Telegram-бот для управління завданнями. Ним можна користуватися безкоштовно. [Електронний ресурс] – Режим доступу: https://mc.today/dmitrij-dubilet-zapustil-telegram-bot-dlya-upravleniya-zadachami-im-mozhno-polzovatsya-besplatno/. 25.01.2024
6. Сайт Bakertilly: Mind Map: як створити карту ваших ідей (+ сервіси для цього). [Електронний ресурс] – Режим доступу: https://bakertilly.ua/mind-map-як-створити-карту-ваших-ідей-сервіс/ 04.02.2024
7. Сайт SendPulse: Як створити чат-бот у Telegram. [Електронний ресурс] – Режим доступу: https://sendpulse.ua/knowledge-base/chatbot/telegram/create-telegram-chatbot 04.02.2024
8. Сайт Gerabot : Як отримати токен телеграм та створити бота? [Електронний ресурс] – Режим доступу: https://gerabot.com/article/yak_otrimati_token_telegram_ta_stvoriti_bota#:~:text=Токен%20-%20це%20ключ%2C%20який%20складається,знайти%20бота%20з%20назвою%20BotFather 04.02.2024
9. Сайт AQTestLab training center : HTTP ПРОТОКОЛ. ЩО І ДЕ ТЕСТУВАТИ? [Електронний ресурс] – Режим доступу:

https://training.qatestlab.com/blog/technical-articles/http-protocol-what-and-where-to-test/ 04.02.2024

10. Сайт Nic.ua : Що таке API і де їх шукати [Електронний ресурс] – Режим доступу: https://info.nic.ua/uk/blog-uk/api-2/ 04.02.2024
11. Сайт Lemon.school : Яка в Telegram мова програмування? [Електронний ресурс]. – Режим доступу: https://lemon.school/blog/yaka-u-telegram-mova-programuvanya. 28.02.2024
12. Сайт Хесклет : Node.js: навіщо писати бекенд на JavaScript і скільки за це платять [Електронний ресурс]. – Режим доступу: https://ru.hexlet.io/blog/posts/zachem-izuchat-node-js-ili-o-perspektivah-bekenda-na-javascript#:~:text=js-,Node.,использования%20JavaScript%20на%20стороне%20сервера. 28.02.2024

ДОДАТКИ

Додаток А. Налаштування кнопок зв’язку користувача та бота під час роботи

Файл app.buttons.ts

```
import { Markup } from 'telegraf'

export function actionButtons() {
	return Markup.keyboard(
		[
			Markup.button.callback(' ⚡ Створити задачу', 'create'),
			Markup.button.callback(' 📋 Список задач', 'list'),
			Markup.button.callback('☑ Завершити', 'done'),
			Markup.button.callback(' 🖍 Редагування', 'edit'),
			Markup.button.callback(' ❌ Видалення', 'delete')
		],
		{
			columns: 2
		}
	)
}
```

Додаток Б. Під’єднання до бази даних PostrgeSQL

Файл app.module.ts

```
import { Module } from '@nestjs/common'
import { TypeOrmModule } from '@nestjs/typeorm'
import { TelegrafModule } from 'nestjs-telegraf'
import { join } from 'path'
import * as LocalSession from 'telegraf-session-local'
import { AppService } from './app.service'
import { AppUpdate } from './app.update'
import { TG_TOKEN } from './config'
import { TaskEntity } from './task.entity'

const sessions = new LocalSession({ database: 'session_db.json' })

@Module({
	imports: [
		TelegrafModule.forRoot({
			middlewares: [sessions.middleware()],
			token: TG_TOKEN
		}),
		TypeOrmModule.forRoot({
			type: 'postgres',
			host: 'localhost',
			port: 5421,
			database: 'todo-tg-bot',
			username: 'postgres',
			password: '',
			entities: [join(__dirname, '**', '*.entity.{ts,js}')],
			migrations: [join(__dirname, '**', '*.migration.{ts,js}')],
			synchronize: true
```

```
            }),
            TypeOrmModule.forFeature([TaskEntity])
        ],
        providers: [AppService, AppUpdate]
})
export class AppModule {}
```

Додаток В. Організація методів для редагування, видалення та завершення виконання завдання за запитом користувача

Файл app.service.ts

```
import { Injectable } from '@nestjs/common'
import { InjectRepository } from '@nestjs/typeorm'
import { Repository } from 'typeorm'
import { TaskEntity } from './task.entity'

@Injectable()
export class AppService {
	constructor(
		@InjectRepository(TaskEntity)
		private readonly taskRepository: Repository<TaskEntity>
	) {}

	async getAll() {
		return this.taskRepository.find()
	}

	async getById(id: number) {
		return this.taskRepository.findOneBy({ id })
	}

	async createTask(name: string) {
		const task = await this.taskRepository.create({ name })

		await this.taskRepository.save(task)
		return this.getAll()
	}
```

```
    async doneTask(id: number) {
        const task = await this.getById(id)
        if (!task) return null

        task.isCompleted = !task.isCompleted
        await this.taskRepository.save(task)
        return this.getAll()
    }

    async editTask(id: number, name: string) {
        const task = await this.getById(id)
        if (!task) return null

        task.name = name
        await this.taskRepository.save(task)

        return this.getAll()
    }

    async deleteTask(id: number) {
        const task = await this.getById(id)
        if (!task) return null

        await this.taskRepository.delete({ id })
        return this.getAll()
    }
}
```

Додаток Г. Організація роботи боту

Файл app.update.ts

```
import {
	Ctx,
	Hears,
	InjectBot,
	Message,
	On,
	Start,
	Update
} from 'nestjs-telegraf'
import { Telegraf } from 'telegraf'
import { actionButtons } from './app.buttons'
import { AppService } from './app.service'
import { showList } from './app.utils'
import { Context } from './context.interface'

@Update()
export class AppUpdate {
	constructor(
		@InjectBot() private readonly bot: Telegraf<Context>,
		private readonly appService: AppService
	) {}

	@Start()
	async startCommand(ctx: Context) {
		await ctx.reply('Hi! Friend 👋')
		await ctx.reply('Що ти бажаєш зробити?', actionButtons())
	}
```

```
@Hears('⚡ Створити задачу')
async createTask(ctx: Context) {
	ctx.session.type = 'create'
	await ctx.reply('Опиши задачу: ')
}

@Hears('📋 Список задач')
async listTask(ctx: Context) {
	const todos = await this.appService.getAll()
	await ctx.reply(showList(todos))
}

@Hears('☑ Завершити')
async doneTask(ctx: Context) {
	ctx.session.type = 'done'
	await ctx.deleteMessage()
	await ctx.reply('Напиши ID задачі: ')
}

@Hears('🖍 Редагування')
async editTask(ctx: Context) {
	ctx.session.type = 'edit'
	await ctx.deleteMessage()
	await ctx.replyWithHTML(
		'Напиши ID і нову назву задачі: \n\n' +
			'У форматі - <b>1 | Нова назва</b>'
	)
}

@Hears('❌ Видалення')
async deleteTask(ctx: Context) {
```

```
        ctx.session.type = 'remove'
        await ctx.deleteMessage()
        await ctx.reply('Напиши ID задачі: ')
    }

    @On('text')
    async getMessage(@Message('text') message: string, @Ctx() ctx: Context) {
        if (!ctx.session.type) return

        if (ctx.session.type === 'create') {
            const todos = await this.appService.createTask(message)
            await ctx.reply(showList(todos))
        }

        if (ctx.session.type === 'done') {
            const todos = await this.appService.doneTask(Number(message))

            if (!todos) {
                await ctx.deleteMessage()
                await ctx.reply('Задачі з таким ID не знайдено!')
                return
            }

            await ctx.reply(showList(todos))
        }

        if (ctx.session.type === 'edit') {
            const [taskId, taskName] = message.split(' | ')
            const todos = await this.appService.editTask(Number(taskId),
taskName)

            if (!todos) {
```

```
                        await ctx.deleteMessage()
                        await ctx.reply('Задачі з таким ID не знайдено!')
                        return
                    }

                    await ctx.reply(showList(todos))
                }

                if (ctx.session.type === 'delete') {
                    const todos = await this.appService.deleteTask(Number(message))

                    if (!todos) {
                        await ctx.deleteMessage()
                        await ctx.reply('Задачі з таким ID не знайдено!')
                        return
                    }

                    await ctx.reply(showList(todos))
                }
            }
        }
```

Додаток Д. Виведення списку задач

Файл app.utils.ts

```
export const showList = todos =>
    `Твій список задач: \n\n${todos
        .map(todo => (todo.isCompleted ? '☑' : '⭕') + ' ' + todo.name + '\n\n')
        .join('')}`
```

Додаток Е. Збереження змінної, яка містить інформацію про API боту

Файл config.ts

```
export const TG_TOKEN = ''
```