Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

Дослідження методів розробки чат-ботів

Текстова частина до курсової роботи

за спеціальністю „Інженерія програмного забезпечення” 121

Керівник курсової роботи

асистент Салата К.В.

_____

(*підпис*)

“____” _____ 2022 р.

Виконав студент БП ІПЗ-3

Дяченко М.Є.

“____” _____ 2022 р.

Календарний план виконання роботи

Тема: «Дослідження методів розробки чат-ботів»

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1	Визначення завдання курсової роботи	19.10.21	
2	Огляд предметної області	30.10.21	
3	Вибір технологій розробки	14.12.21	
4	Формулювання технічного завдання	25.05.22	
5	Написання теоретичної частини	03.06.22	
6	Написання практичної частини	06.06.22	

Зміст

Вступ

Актуальність теми

Дана робота присвячена дослідженню методів розробки чат-ботів та детальному практичному аналізу одного з цих методів. Актуальність теми можна пояснити стрімким розвитком сфери інформаційних технологій. Технічний прогрес до непізнаваності змінив торгівлю, медіапростір, людське спілкування та ще багато інших аспектів нашого життя. Винятком не стали й зв’язки між людьми та брендами, компаніями, громадськими організаціями тощо, що автоматизуються та переходять в онлайн, а з початком пандемії цей перехід значно пришвидшився.

Програми, що забезпечують цей зв’язок і є чат-ботами. За рахунок чат-ботів можна реалізувати такі функції, як інформування клієнтів, продажі та бронювання, відповіді на прості запитання, проведення опитувань. Дане технічне рішення може надавати цілодобову підтримку клієнтів та автоматизує велику частину комунікацій, що приводить до значного збільшення ефективності, і, як наслідок, прибутку. В результаті, використання чат-ботів в будь-якій сфері, що потребує активної комунікації з клієнтами, стало необхідністю. Не зовсім коректно буде говорити що за цією технологією майбутнє, оскільки вона вже стала невід’ємною частиною сьогодення.

Мета курсової роботи

Метою курсової роботи є огляд та аналіз існуючих станом на сьогодні типів чат-ботів та методів їх розробки (мови програмування, патерни дизайну, інші допоміжні технології) та розробка з використанням найбільш підходящих для технічного завдання бота методів.

Для того щоб здійснити зазначену мету необхідно:

- Формування чіткого технічного завдання (далі – ТЗ) для чат-бота
- Огляд типів ботів за моделями спілкування, платформами для реалізації, технологіями та методами розробки, виявлення та опис їх основних переваг та недоліків. Вибір найбільш релевантних методів для встановлених умов ТЗ і використання їх в практичній частині роботи
- Розробка чат-боту що задовольняє вимоги вказані в ТЗ з покроковим описом процесу

Розділ 1. Огляд предметної області

Визначення поняття чат-бота. Загальна класифікація чат-ботів за принципами роботи

Чат-боти є програмами, що створені для імітації розмови з іншою людиною. Термін «чат-бот» був вперше вжитий у 1994 році розробником Майклом Молдингом [11]. За рахунок імітації діалогу, чат-боти можуть бути корисними в багатьох інших сферах, таких як електронні продажі, розваги, освіта. [2] Можна розділити чат-ботів на два основні види:

Декларативні чат-боти, орієнтовані на виконання завдань: Такі програми орієнтовані на виконання одного завдання. В основному, боти цього типу використовують чітко задані команди, на які можна дати одну або декілька відповідей. Також можуть використовувати технологію NLP (Nature Language Processing – обробка природної мови). Окрім чітко заданих команд, такі боти можуть відповідати на прості питання, наприклад, про години роботи закладу.

Предикативні чат-боти, працюючі в режимі діалогу:

Таких ботів часто називають «віртуальними помічниками». Вони мають більш розвинені інтерактивні можливості, та методи персоналізації, ніж декларативні чат-боти. Такі боти можуть враховувати контекст, можуть містити в собі більш розвинені технології NLU(Natural Language Understanding – розуміння природної мови). У своїх відповідях користувачу такі чат-боти використовують процеси NLG(Natural Language Generation – генеруванян природної мови). Також використовують машинне навчання, щоб покращувати свої відповіді в процесі роботи. Вони можуть вивчати вподобання користувача та надавати персоналізовані рекомендації. Окрім

обробки тексту, такі боти можуть мати функції голосового вводу. Прикладами таких чат-ботів є Alexa від Amazon та Siri від Apple. [3]

Декларативні чат-боти є значно простішими у розробці, і є кращим варіантом у випадку, коли необхідно давати відповіді на обмежену кількість питань, з прописаними сценаріями діалогу. Прикладом є перегляд довідкової інформації, заповнення форм з даними. Предикативні чат-боти потребують більше зусиль у розробці та працюють повільніше, оскільки використовують більш складні алгоритми. У випадках, коли існує безліч можливих сценаріїв комунікації, а користувачі вводять текст у довільній формі, необхідно застосовувати саме такий варіант.

Різновиди чат-ботів за функціоналом

Чат боти можуть бути інтегрованими в веб-сайт. З такими ботам стикається ледь не кожен користувач інтернету – на сайті з’являється діалогове вікно з чатом, в якому бот пропонує свою допомогу у вирішенні можливих питань. Також вони можуть бути реалізовані в платформі, підтримуючій розробку ботів (наприклад, Viber, Telegram, Discord та інші).

Станом на сьогодні, існують такі основні напрямки використання чат-ботів:

- Створення допоміжних функцій для платформи, що підтримує розробку чат-ботів. Прикладом є боти в мессенджері Telegram, що додавали можливість залишати реакції на каналах та створювати опитування до того, як цей функціонал був доданий в мессенджер. В геймерській платформі Discord існують боти, які дозволяють транслювати музику в голосовому каналі.

- Інтернет-продажі. В цьому випадку боти можуть надавати рекомандації при покупці, реалізовувати функціонал пошуку, замовлення, покупки товарів, та інша бізнес-логіка, необхідна для ведення комерційної діяльності.
- Перша лінія роботи з клієнтами, телефонія, надання консультацій, відповіді на типові питання. Автоматизація комунікацій з клієнтами дозволяє зберегти гроші на утриманні служби підтримки.
- Обробка контенту, пропонованого користувачами. Даний підхід збору інформації від великої публіки позволяє налаштувати захист від спаму, налаштувати обмеження при вводі даних, схожі на форми, та не показує аудиторії контактні дані людини, що збирає інформацію. Багато телеграм-каналів використовують таких ботів для публікації запропонованого користувачами контенту. 26 лютого 2022 року в Україні було створено чат-бот «@stop_russian_war_bot», через якого збирались дані про пересування російських військових та ДРГ. [12]
- Розваги – такі боту можуть робити що завгодно, від генерації віршів до простих ігор.

Переваги застосування чат-ботів

Протягом останніх років використання чат-ботів набувало популярності серед компаній що потребують існування лінії зв’язку з людьми. В великій частині випадків, такі застосунки розробляться для автоматизації комунікації компанії з її клієнтами. Значною перевагою чат-ботів є можливіть відповідати на питання багатьох користувачів одночасно. До того ж, більшість питань користувачів сервісів є передбачуваними, шаблонними. Наприклад, найчастішими питаннями до сервісу доставки

будуть питання вигляду «як дізнатися стан посилки?», «скільки необхідно чекати отримання», тощо. Чат-боти можуть легко давати відповіді на такі питання.

Можливий варіант структурування часто задаваних питань (далі - FAQ) в багаторівневу структуру меню з заздалегідь прописаними кнопками. Для більш детального прикладу, розглянемо раніше вказане питання «скільки необхідно чекати отримання», та застосуємо до спрощеної моделі сервісу доставки. На даному прикладі також розглянемо доцільність застосування чат-ботів.

При такому сценарії, шлях до відповіді на питання «скільки необхідно чекати отримання» можна виділити до питань, стосуючихся отримання посилок, де необхідно ввести номер своєї посилки, щоб дізнатися очікуваний час отримання. Більш детальна діаграма приведена на *Рис. 1*.

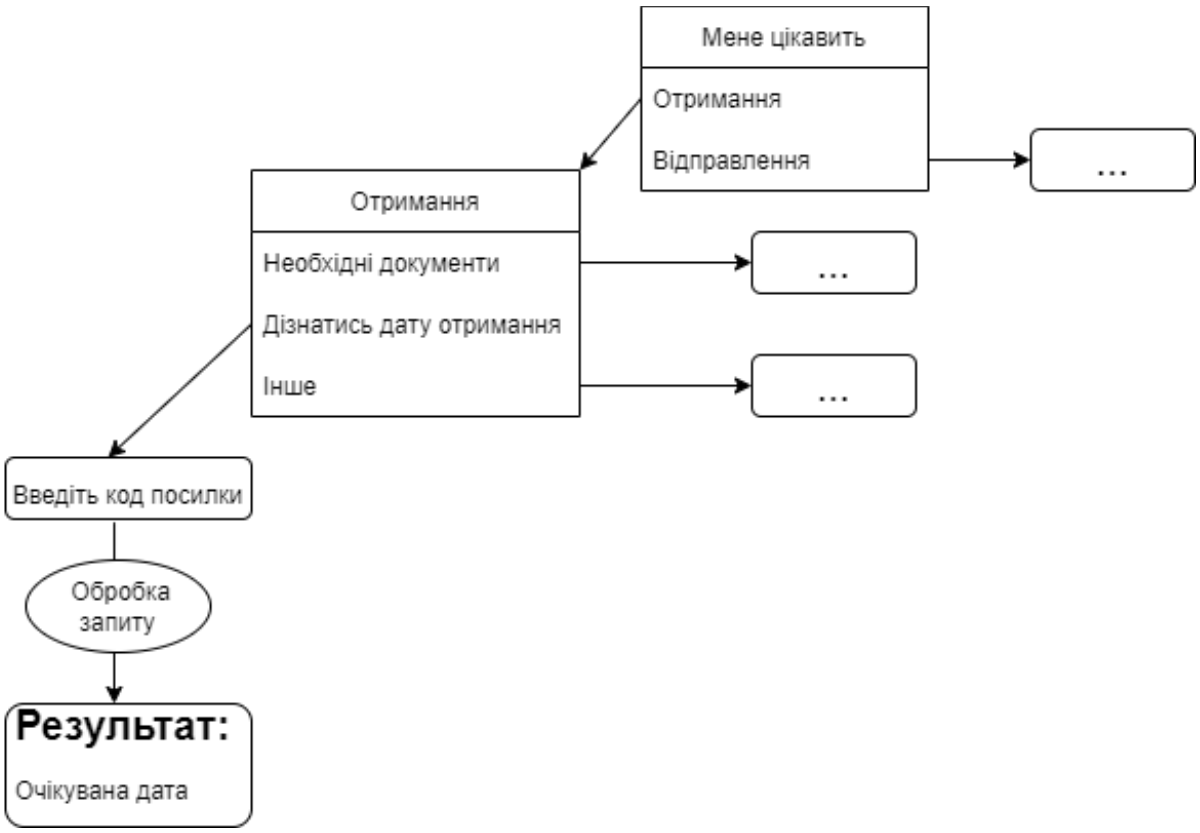

Рисунок 1.1. Схема повідомлень чат-боту

В реальному діалозі, такий запит звучав би приблизно так:
Консультант: «Доброго дня! Що Вас цікавить, отримання чи відправлення?».
Клієнт: «Маю питання щодо отримання посилки».
Консультант: «Що конкретніше Вас цікавить?».
Клієнт: «Хотів би дізнатись, коли прибуде моя посилка».
Консультант: «Добре, вкажіть, будь ласка номер посилки».
Клієнт: «1234».
Консультант: «Ваша посилка №1234 прибуде у відділення 06.06.2022».

Для наведеного уявного прикладу сервісу доставки таке питання є часто задаваним. Додаткові дані від користувача, а саме – номер посилки, досить легко отримати та обробити програмним шляхом. Отже, розробивши чат-бота, що зможе відповідати на таке та схожі питання, можна зекономити на утриманні великого штату служби підтримки. Враховуючи те, що чат-боти все ще не можуть в повній мірі замінити людину, можна залишити невеликий штат людей, що вирішує більш складні питання.

В сучасних реаліях ринку, важливою складовою є створення максимального комфорту для користувачів. З існуванням сльної конкуренції, будь-який крок в сторону покращення якості обслуговування клієнтів має велике значення. Чат-боти є однією зі складових збільшення якості обслуговування. Імітація діалогу робить комунікацію з чат-ботом приємнішою ніж перегляд статичного контенту з FAQ. Для частини людей подібний, наближений до живого спілкування досвід, залишає кращі враження, що може викликати краще ставлення до компанії або організації, що є власником чат-бота. [1]

Ще однією важливою перевагою чат-ботів є те, що в переважній більшості вони є інтегрованими в інші сайти та додатки, та не потребують завантаження додаткового програмного забезпечення, що є зручним для кінцевого користувача, й не потребує розробки графічного інтерфейсу.

Висновок до розділу 1

В цьому розділі був проведений загальний аналіз предметної області чат-ботів. Була надана класифікація чат-ботів за алгоритмами роботи та цілями використання. Наведений кейс з реального життя, на якому пояснюється доцільність використання чат-ботів. Розглянуті переваги використання чат-ботів.

Розділ 2. Технології розробки чат-ботів

Мною було обрано Telegram як платформу реалізації чат бота практичної частини. Даний мессенджер є одним з найпопулярніших в Україні, за даними міжнародного агентсва маркетингових досліджень за грудень 2021 року, Telegram завантажений на 88,5% мобільних пристроїв з ОС Android.

Для розробників існує Telegram API, з багатим функціоналом і можливістю використовувати весь спектр можливостей мессенджера. На його основі було створено багато бібліотек для різних мов програмування. На офіційному сайті Telegram є список з п'ятдесяти бібліотек, розроблених спільнотою для таких мов, як PHP, Node.js, Python, Rust, Go, Java, C#, C++. Для написання чат-бота було обране середовище розробки Node.js та фреймворк для розробки ботів “telegraf.js”, що є зручним, в повній мірі реалізує функції Telegram API, та станом на час написання роботи активно підтримується авторами.

Створення Telegram-бота

Спочатку нам потрібно створити власного чат-бота в Telegram, для цього необхідно скористатись іншим ботом під назвою «BotFather». Ввівши команду «/newbot», з’являється можливіть ввести ім’я бота, та його юзернейм (унікальний ідентифікатор, що використовуєть у URL-посиланні на нового бота). Після цих двох кроків, ми отримуємо унікальний токен бота (приклад токену – «123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11»), що дає доступ до API. Через «BotFather» відбувається й кастомізація бота, наприклад, встановлення зображення профілю, опису, зміна імені тощо. [10]

NodeJS та JavaScript

JavaScript - мова програмування з динамічною типізацією, є реалізацією стандарту ECMAScript. Поєднує у собі риси об'єктно-орієнтованої мови програмування, базованої на прототипах, та функціонально-орієнтованої мови (об'єкти – списки, анонімні функції, функції – об'єкти першого порядку).

Node.js – кросплатформенне середовище виконання з відкритим кодом, що позволяє виконувати код написаний на мові JavaScript. За рахунок виористання Node.js, можна використовувати JavaScript не лише у браузерах, а ще й як мову для розробки серверних застосунків.

Використання PaaS Heroku для хостингу Telegram-ботів

Створення серверів є невід'ємною частиною сучасного програмування. Чат-бот, що розробляється в рамках цієї роботи, по суті є сервером, який обробляє запити з повідомленнями, отримані від Telegram. Для багатьох проектів розгортання власних серверних потужностей є занадто дорогим чи просто нераціональним рішенням. Для подібних випадків, існує альтернатива у вигляді використання хмарного хостингу. Однією з подібних платформ для хмарних обчислень є Heroku. Однією з переваг використання Heroku для невеликих pet-проектів є те, що вона надає безкоштовний доступ до невеликого серверу з 512MB оперативної пам'яті.

Крім цього, в Heroku додано багато додаткових можливостей, що допомагають розробникам:

- Інтеграція з популярною системою контролю версій GitHub. На Heroku існує можливість під'єднати існуючий репозиторій на GitHub,

код якого буде розгортатися на сервері. Існує можливість автоматичного розгортання з гілки «main» при кожній новій зміні коду.

- Велика кількість аддонів (доповнень), що виконують різноманітні функції, такі як створення або підключення баз даних, пошукових систем, логування, тестування, сканування додатку на предмет наявності вразливостей, та багато інших. [8]
- Зручне налаштування та безпечне зберігання змінних середовища в рамках хмарного хостингу.
- Можливіть автоматичного налаштування SSL-сертифікатів за допомогою сервісу «Let’s Encrypt».

Система керування базами даних PostgreSQL

Для написання бота необхідно створити базу даних, в якій зберігатиметься інформація про події, дані про користувачів, оцінки відвідувачів про організаторів. В розробленій моделі дані будуть зберігатись в чітко структурованому форматі, для якого підходить використання реляційних баз даних. СКБД PostgreSQL має великий набір функцій, повністю реалізовує стандарт SQL [7]. В згаданому раніше хмарному хостингу Heroku існує можливість створення PostgreSQL-бази даних. Для безкоштовного використання, розробникам надається можливість створювати базу даних об’ємом до 10000 рядків, що повністю задовольняє потреби невеликих проектів на етапі розробки.

Розробка чат-ботів на базі low-code рішень

Ця частина роботи містить короткий теоретичний огляд процесу розробки чат-ботів за допомогою візуальних мов програмування. Такі програмні рішення користуються популярністю, тому вважаю за необхідне окремо згадати їх.

Як і будь-яка інша програма, чат-боти можуть бути представлені у вигляді блок-схеми. При чому, в випадку чат-бота така схема здебільшого складатиметься з простих та повторюваних кроків, коли програма очікує введення даних, щоб надати відповідь в залежності від питання, його місця на

Спираючись на блок-схему, можна умовно поділити процеси роботи чат-боту на такі етапи:

- Очікування повідомлення користувача.
- Обробка повідомлення користувача, генерація відповіді.
- Зміна стану.

Для реалізації подібного функціоналу можуть підійти мови візуального програмування. Візуальне програмування – це мова програмування, в якій для передачі використовується більше ніж один вимір семантики. Прикладами таких додаткових вимірів є використання просторових елементів та зв'язків, використання часового виміру [5]. Просторові елементи можна перетягувати, і їх розміщення викорустовується як елемент синтаксису.

При такому підході, чат-бот може бути спроектований як діаграма, яка визначає всі можливі стани та переходи між ними. Стан користувача зберігається в системі, і єдиним місцем, що потребує використання коду, є етап обробки користувацького повідомлення. Такі програми як правило

містять блоки умови, що слугують альтернативою «if else» блоків коду, блоки виводу повідомлень від бота та блоки очікування відповіді користувача. Також існує можливість використовувати вебхуки, надсилати запити на сервер та отримувати відповідь.

Прикладами платформ для розробки є Corezoid Bot Platform та ChatBot. Corezoid Bot Platform дозволяє додавати блоки JS коду для внутрішньої обробки даних, що є дуже важливим доповненням. За рахунок цього розробник має можливість обробляти дані, отримані з уже існуючих ендпоінтів. Розглянемо випадок, коли бот продуктової мережі має надсилати список відкритих закладів у місті проживання користувача. Оскільки цей список може оновлюватись, ми не можемо задати статичні дані. Натомість, ми можемо робити GET-запит, що поверне JSON(JavaScript Object Notation) з актуальними даними, та перетворити його в зручний для читання текст, обробивши дані в блоці з кодом.

На сайті Corezoid вказано, що його використовують такі крупні компанії, як «METRO», «АТБ», «ПУМБ», «NOVUS», «VARUS» та інші. [4]

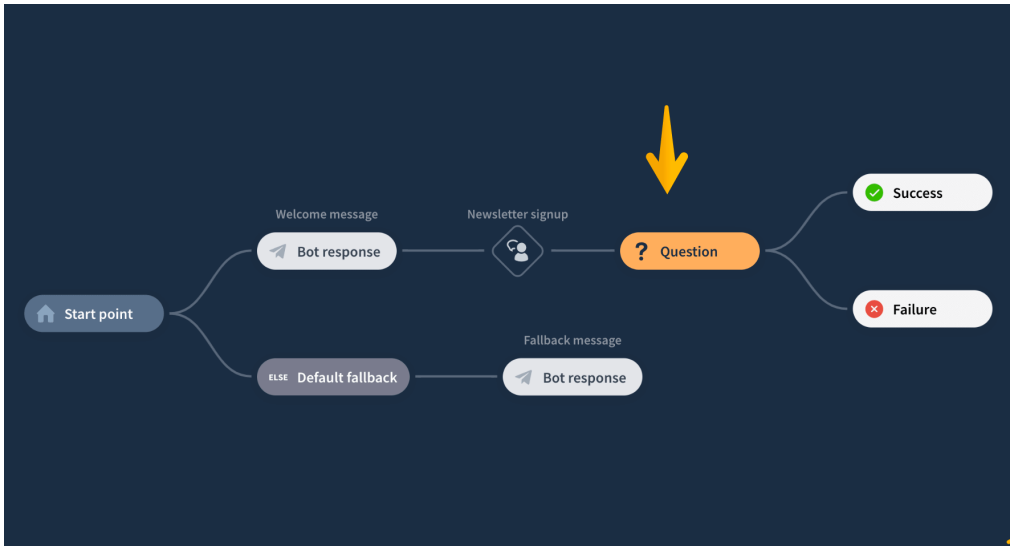

Рисунок 2. Інтерфейс засобу візуального програмування «ChatBot»

Переваги використання візуального програмування у розробці чат ботів:

- До розробки можна підключати людей з невеликим досвідом програмування.
- Прискорює час розробки.
- Зазвичай, такі середовища розробки є мультиплатформенними, тобто в них розробляється одна універсальна логіка для всіх мессенджерів.

Недоліки:

- Відсутність можливості використання систем контролю версій.
- Через обмежені можливості візуального програмування, для роботи чат-бота скоріше за все будуть необхідні додаткові зовнішні сервіси. Як наслідок, відсутня можливість використання монолітної архітектури.

Висновок до розділу 2

Наданий теоретичний опис технологій, що використовуватимуться в процесі розробки, розписані їх основні переваги. Додана інформація про альтернативний спосіб створення чат-ботів за допомогою візуальних мов програмування.

Розділ 3. Розробка власного чат-боту в Telegram

Технічне завдання чат-бота

Бот повинен реалізувати дошки оголошень з подіями в рамках університетського життя. Бот працює в парі з телеграм-каналом, на який публікується інформація про події, які створюються користувачами через бота. Має бути реалізований такий функціонал:

1. Публікація інформації про нові події. Є можливіть додати назву, опис, та дату проведення події.
2. Можливість зберегти назву організації, що проводить подію.
3. Додати функціонал для адміністраторів, що перевіряють запропоновані події перед публікацією.
4. Додати можливість вибору мови інтерфейсу чат-бота.

Користувачів чат-боту можна умовно розділити на 2 типи:

1. Організатор події. В ролі організатора може виступати будь-який користувач програми. Такі користувачі мають можливість створювати події, що мають бути схвалені адміністратором перед публікацією.
2. Адміністратор – користувач, який має право схвалювати або скасовувати події, в залежності від того чи не містять вони неприйнятного для публікації контенту. У випадку схвалення, інформація про подію публікуватиметься на канал. В випадку схвалення та скасування організатору прийде додаткове повідмлення з інформацією про те, що подія була схвалена або скасована відповідно.

Основні завдання розробки:

1. Створити модель даних, розробити базу даних на її основі.
2. Розробити програмне рішення для отримання інформації з бази даних у коді чат-боту.
3. Реалізувати всі необхідні сценарії діалогу.

Спираючись на теоретичний матеріал з першого розділу, бота з подібним функціоналом можна віднести до декларативних, він підходить під цю категорію по всім ознакам, а саме – заточений під виконання однієї функції (публікація подій на Telegram-канал), та має відповідати на чітко прописані сценарії діалогу, а саме – навігація по меню та обробка повідомлень, написаних користувачем.

Налаштування проекту

Перед початком процесу написання коду, необхідно створити проект та провести всі попередні налаштування проекту, зокрема – завантажити усі необхідні нам бібліотеки/фрейморки. В проектах на Node.js використовується менеджер пакунків npm. Використання npm дозволяє легко завантажувати, видаляти та оновляти сторонні бібліотеки і фреймворки, що значно спрощують розробку. В рамках розробки чат-боту використовуються такі додаткові пакунки:

- «express» (версія 4.17.1) - фреймворк для написання бекенд застосунків на Node.js.
- «pg-promise» (версія 10.5.6) - бібліотека для роботи з СКБД PostgreSQL.
- «telegraf» (версія 3.38.0) – фреймвокр для розробки чат-ботів в Telegram.

Щоб завантажити пакунок, необхідно ввести команду «npm install назва_пакунку». Наприклад, для завантаження «telegraf» використовується команда «npm install telegraf».

Для ініціалізації проекту, використовується команда «npm init». Ця команда генерує файл «package.json», у якому вказується назва проекту, сторонні пакунки та їх версії, тип ліцензії (MIT, NSC тощо), а також скрипти для запуску та тестування проекту. Хмарний хостинг Heroku використовує команду «npm start» для запуску проекту, отже необхідно написати скрипт «start», який запускатиме сервер (в нашому випадку, файл «index.js».

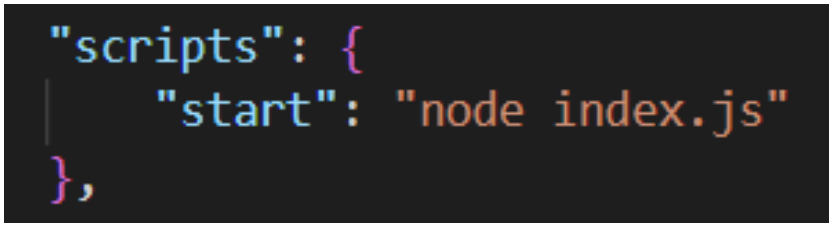

Рисунок 3.1. Скрипт для запуску проекту

Написання коду для взаємодії з базою даних

Таблиці в базі даних, що використовується в проекті, створенні за моделлю, зображеною на *Рис. 3.2.*

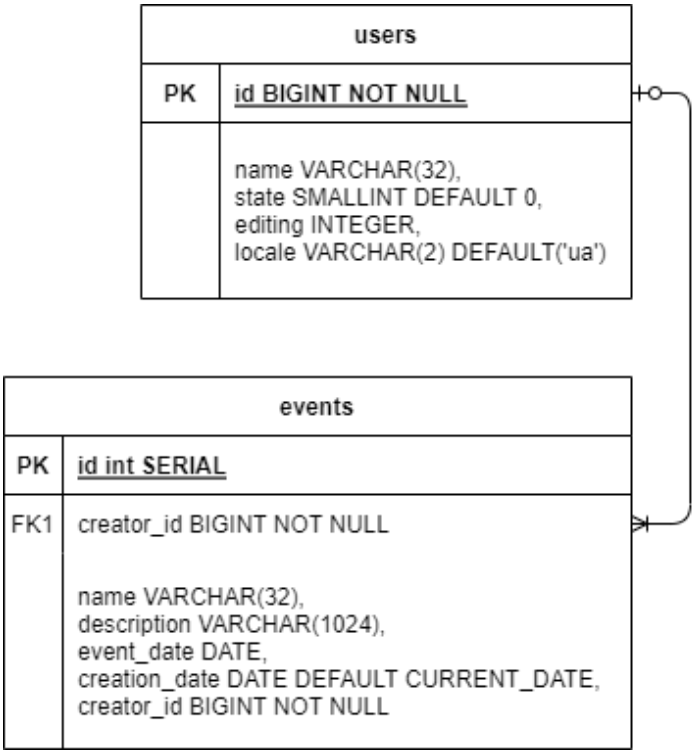

Рисунок 3.1. ER-модель бази даних для чат-боту

Пояснення значень атрибутів таблиці «users»:

- id – ключ, значенням якого є ідентифікатор користувача в Telegram.
- name – назва організації, не є обов'язковим атрибутом, користувач може публікувати події і без неї.
- state – стан користувача, від якого залежить якою функцією-обробником чат-бот буде обробляти та давати відповідь користувачу.
- editing – ідентифікатор останньої події, що редагувалась користувачем.
- locale – мова інтерфейсу, за замовчуванням українська.

Пояснення значень атрибутів таблиці «events»:

1. id – ідентифікатор події, генерується шляхом автоматичної інкрементації в базі даних.
2. creator_id – ідентифікатор користувача, що створив подію.
3. name – назва події, довжина до 32 символів.

4. description – опис події, довжина до 1024 символів.
5. event_date – дата проведення події, визначається користувачем, не може бути меншою за сьогоднішню дату.
6. creation_date – дата створення події, генерується автоматично.

Як було зазначено раніше, для взаємодії з базою даних використовуватиметься бібліотека «pg-promise». Для підключення до бази даних в Heroku, використовується подібний код:

```
const config = {
    connectionString: process.env.DATABASE_URL,
    ssl: { rejectUnauthorized: false }
};
let DB = pgp(config);
```

Рисунок 3.3. Підключення до бази даних

Розглянемо також код взаємодії з базою даних, на прикладі методу «getUser(userId)». Цей метод перевіряє чи є користувач зареєстрованим в боті, роблячи SELECT-запит в базу даних користувачів. Повертає JSON з даними про користувача, якщо він є в базі даних, або булеве значення false, якщо користувач відсутній.

```
/**
 * is user in DB
 * @param {number} userId
 * @return {boolean|Object} db row, representing user if he is in db, false otherwise - false
 */
exports.getUser = async function (userId) {
    let res = false;
    await DB.result(`SELECT * FROM "users" WHERE "id"=${userId}`)
        .then(function (data) {
            if(data.rows.length >= 1) res = data.rows[0];
        })
        .catch(function (error) {
            logDBError(error);
        });
    return res;
}
```

Рисунок 3.4. Метод getUser

Сервер для роботи чат-бота

Для роботи чат-бота створити сервер з використанням фреймворку «express». В цьому фрагменті коду відбувається налаштування серверу, ініціалізується express-застосунок та проміжне програмне застосування боту. Метод «app.use» застосовує проміжні налаштування для всього серверу, таким чином ми можемо створити шлях, за яким запити надходитимуть на вебхук бота. «app.use(express.json())» визначає тип даних тіла запиту, в цьому випадку тіло POST-запитів буде парситись як JSON. В свою чергу, «bot» є об’єктом, у якому зберігаються усі проміжні обробники. [9]

```
const app = express();
const bot = new Telegraf(process.env.BOT_TOKEN);

app.use(bot.webhookCallback('/bot'));
app.use(express.json());
```

Рисунок 3.5. Налаштування вебхуку та парсеру POST-запитів

Вебхук — це запит HTTP, який ініціюється подією у вихідній системі та надсилається в систему призначення, часто з корисним навантаженням даних. Вебхуки автоматизовані, іншими словами, вони автоматично надсилаються, коли їх подія запускається у вихідній системі. [6]. В нашому випадку, Telegram надсилатиме POST-запит на сервер після кожного нового повідомлення користувача. Таким чином, чат-бот отримує інформацію про всі оновлення.

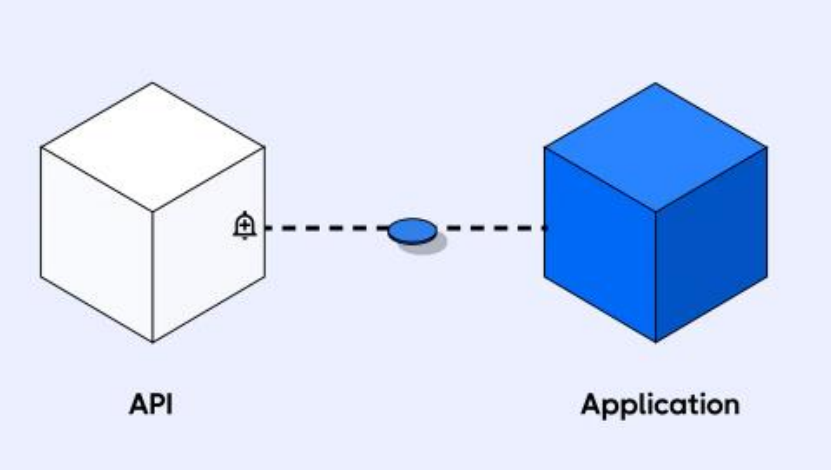

Рисунок 3.6. Схема роботи вебхуків [6]

Обробка команд та різних видів подій

Крім простого налаштування вебхуків в рамках express-серверу, бібліотека «telegraf» містить багато корисних методів для розробки чат-ботів. Середних є метод «start» що позволяє окремо обробляти команду «/start», що вводиться користувачем при першому використанні чат-бота або при повторному використанні після видалення. В цьому коді метод «start» додає нових користувачів в базу даних.

Також існує метод «help» для обробки команди з відповідною назвою. Як правило, відповіддю на цю команду є статична інформація з підказками в використанні чат-бота. Обидва меторжи приймають функцію-обробник повідомлення. [9]

```
//Команда start
bot.start(async (ctx) => {
    //Запис нового користувача в базу даних
    let user = await DB.getUser(ctx.chat.id);
    await ctx.reply(BM.getMessageLocale(user.locale,'START_MSG'),
    { reply_markup: BM.menu[user.locale]['MENU_MAIN_BUTTONS'] });
    if (user === false) {
        user = await DB.addUserToDB(ctx.chat.id);
    }
})

//Команда help
bot.help(async (ctx) => {
    let user = await DB.getUser(ctx.chat.id);
    ctx.reply(BM.getMessageLocale(user.locale,'HELP_MSG'))
});
```

Рисунок 3.7. Реалізація команд /start та /help

Для обробки команди з будь-якою назвою, існує метод «command», що кірм функції-обробника, приймає й стрічку з назвою команди.

```
//Команда для повернення в головне меню
bot.command('main',async (ctx) => {
    let user = await DB.changeUser(ctx.chat.id,'state',0);
    await ctx.reply(BM.getMessageLocale(user.locale,'MAIN_MENU'),
    { reply_markup: BM.menu[user.locale]['MENU_MAIN_BUTTONS'] });
});
```

Рисунок 3.8. Використання методу command

Як правило, команди працюють досить відокремлено від інших функцій чат-боту. Тому виокремлення команд у окремі методи є гарним рішенням, що відділяє логіку їх обробки від інших видів обробки повідомлень. Context (ctx) зберігає в собі всі оновлення, що були отримані в результату webhook-запиту на сервер від Telegram. Даний бот підтримує два типи оновлень, що вказані в методі «bot.on», а саме – «message» (повідомлення) та «callback_query» (колбек-запит). Далі, за наявность callback-запиту, використовується обробник з модуля «callbackHandler». В іншому випадку, через обробник «privateChatHandler» обробляються повідомлення в приватному чаті з ботом. В груповому чаті бот не відповідатиме на повідомлення. Метод «bot.launch()» запускає бота на сервері. [9]

```
bot.on(['message', 'callback_query'], async (ctx) => {
    if (ctx.callbackQuery) {
        callbackHandler.handle(ctx, bot.telegram);
    } else if (ctx.chat && ctx.chat.type == 'private') {
        privateChatHandler.handle(ctx, bot.telegram);
    }
});

bot.launch();
```

Рисунок 3.9. Розподіл обробки оновлень в залежності від типу

Збереження відповідей чат-бота

При розробці, відповіді чат-бота на різні повідомлення користувача необхідно зберігати в одному файлі. Це необхідно задля підтримки чистоти коду, можливості легко відредагувати повідомлення чат-боту, що використовується більш ніж у одному місці. В рамках цього проекту, такий функціонал реалізований в файлі «botMessages.js». Повідомлення зберігатимуться у JSON-об’єкті з парами «ключ-значення». Таким чином, ми можемо використовувати ключі повідомлень в декількох місцях коду, та змінювати значення лише в одному JSON.

```
const msg = {
    ua: {
        'HELP_MSG': 'Інструкція по використанню бота',
        'MAIN_MENU': 'Ви у головному меню.',
        //...more messages
```

Рисунок 3.10. Структура JSON-об’єкту з повідомленнями

Кнопки меню зберігаються теж зберігаються в JSON-форматі в файлі «botMessages.js» в форматі масиву з масивів.

Такий формат збереження повідомлень чат-бота позволяє легко імплементувати зміну мови інтерфейсу. Пари «ключ-значення» з повідомленнями можуть бути полями об’єкту з скороченням мови інтерфейсу. Отримати будь-яке повідомлення на потрібній нам мові можна за рахунок подібного методу:

```
exports.getMessageLocale = function (lang,message) {
    try{
        let botMsgs = msg[lang][message];
        if (botMsgs === undefined) return message;
        if (botMsgs === null) return message;
        if (typeof botMsgs === 'object') {
            if (botMsgs.length === 1) {
                return botMsgs[0];
            } else if (botMsgs.length > 1) {
                let random = Math.floor(Math.random() * botMsgs.length);
                return botMsgs[random];
            }
        }
        return botMsgs;
    } catch (error) {
        return message;
    }
}
```

Рисунок 3.11. Метод для отримання повідомлень за їх ключем

Також, ми можемо зберігати масив з повідомленнями для одного ключа, та повертати випадкове повідомлення з цього масиву. За рахунок цього, спілкування з ботом може відчуватися більш «живим», наближеним до реального.

Обробник оновлень privateChatHandler

Очевидним є факт, що написання всього коду додатку в одному файлі зробить дуже незручною роботу з кодом у випадку значного розширення функціоналу. Саме тому необхідне розбиття обробників повідомлень на різні модулі. Одним з них є js-модуль «privateChatHandler». Основний метод модуля – «handle», який обробляє текст, введений користувачем, в залежності від стану, збереженого в базі даних. Оскільки інформація про оновлення, що надходить від Telegram, містить лише загальну інформацію про повідомлення, запам’ятовувати місце, де був користувач на етапі

відправки повідомлення, ми маємо самотужки. В цьому чат-боті є такі стани:

- «0» - користувач знаходиться в головному меню.
- «1» - користувач вводить назву події.
- «2» - користувач вводить опис події.
- «3» - користувач вводить дату події.
- «4» - користувач має підтвердити або скасувати публікацію.
- «10» - користувач змінює назву організації, під якою публікуються події.
- «20» - користувач знаходиться в меню вибору мови.

Перевіряється стан в методі «handle», використовуючи вираз «switch».

```
switch (currentState) {
    case 0:
        await startMenuActions(ctx, bot, userId, userInput, user.name);
        break;
    case 1:
        await eventNameInputActions(ctx, bot, userId, userInput);
        break;
    case 2:
        await eventDescriptionInputActions(ctx, bot, userId, userInput, user.editing);
        break;
    case 3:
        await eventDateInputActions(ctx, bot, userId, userInput, user.editing);
        break;
    case 4:
        await eventPublishActions(ctx, bot, userId, userInput);
        break;
    case 10:
        await orgNameInputActions(ctx, bot, userId, userInput);
        break;
    case 20:
        await localeChangeActions(ctx, bot, userId, userInput);
        break;
    default:
        ctx.reply(BM.getMessageLocale(USER_LOCALE, 'NICHT_FERSTEIN'));
}
```

Рисунок 3.12. Розбиття на обробники повідомлень кожного стану

Кожна з цих функцій має реалізувати перходи по меню, валідацію введених даних перед збереженням, змінювати стан, та давати відповідь, в залежності від повідомлення. Розглянемо один з обробників на прикладі

обробки вводу назви організації. Спочатку перевіряємо, чи натиснув користувач на кнопку меню, для повернення в головне меню. У випадку, якщо користувач таки ввів нову назву, проводимо валідацію введених даних (у цьому випадку – лише перевірка на довжину стрічки), зберігаємо назву в базу даних та змінюємо стан користувача. Також відправляємо відповідь за допомогою методу «ctx.reply». [9]

```
async function orgNameInputActions(ctx, bot, userId, userInput) {
    if (userInput === BM.menu[USER_LOCALE]['GOTO_MAIN'].keyboard[0][0]) {
        ctx.reply(BM.getMessageLocale(USER_LOCALE, 'MAIN_MENU'),
         { reply_markup: BM.menu[USER_LOCALE]['MENU_MAIN_BUTTONS'] });
        await DB.changeUser(userId, 'state', 0);
    } else {
        if (userInput.length > 32) {
            ctx.reply(BM.getMessageLocale(USER_LOCALE, 'ORG_MUST_BE_1_TO_32_CHARS'));
        } else {
            await DB.changeUser(userId, ['name', 'state'], [userInput, 0]);
            ctx.reply(BM.getMessageLocale(USER_LOCALE, 'NAME_CHANGED')(userInput),
            { reply_markup: BM.menu[USER_LOCALE]['MENU_MAIN_BUTTONS'] });
```

Рисунок 3.12. Реалізація обробки повідомлень в стані «10»

Обробник оновлень callbackHandler

Крім повідомлень, в Telegram існують інші типи оновлень, одним з них є колбек. Колбеки є подіями, що надходять після натискання кнопки, що прикріплена до повідомлення. Є моливість передати додаткову інформацію про подію в змінній стрічці «callback_data». Обробка колбеків працює за принципом, схожим на обробку повідомлень. Але, на відміну від повідомлень, ми можемо зберігати власні типи колбеку в змінній «callback_data». Також, крім того щоб опційно відповісти на повідомлення, ми маємо відповісти на колбек за допомогою методу «ctx.answerCbQuery».

```
reply_markup: {
    inline_keyboard: [
        [{
            text: BM.getMessageLocale(USER_LOCALE,'CB_YES'),
            callback_data: `approve_${eventId}_${userId}`
        },{
            text: BM.getMessageLocale(USER_LOCALE,'CB_NO'),
            callback_data: `remove_${eventId}_${userId}`
        }]
    ]
},
```

Рисунок 3.13. Приклад кнопок, прикріплених до повідомлення

Розглянемо приклад роботи одного з колбеків. Даний код відповідає за скасування публікації події адміністратором. На початку стрічки «callback_data» вказаний тип колбеку, а саме «remove» (див *Рис 3.13*). За ним ми можемо ідентифікувати колбек через функцію «startsWith». Також ми можемо передати необхідні для обробки повідомлення дані, в цьому випадку – ідентифікатор користувача, що публікував замовлення. Маючи цей ідентифікатор, ми можемо сповітити користувача що публікація події була скасована, уточнивши її назву. Для відправки повідомлення не у відповідь в той самий діалог, використовується метод «bot.sendMessage», що приймає ідентифікатор чату, куди потрібно відправити повідомлення.

```
if(callback.data.startsWith("remove")) {
    let eventId = callback.data.split("_")[1];
    let orgId = callback.data.split("_")[2];
    let event = await DB.getEvent(eventId);
    ctx.editMessageReplyMarkup({inline_keyboard: [[]]});
    bot.sendMessage(orgId,BM.getMessageLocale(USER_LOCALE,'PUB_DISAPPROVE')(event.name));
    ctx.answerCbQuery(BM.getMessageLocale(USER_LOCALE,'CB_NO'));
}
```

Рисунок 3.14. Обробка колбеку «remove»

Приклади сценаріїв діалогу

В цьому розділі будуть наведені скріншоти з діалогами в різних ситуаціях.

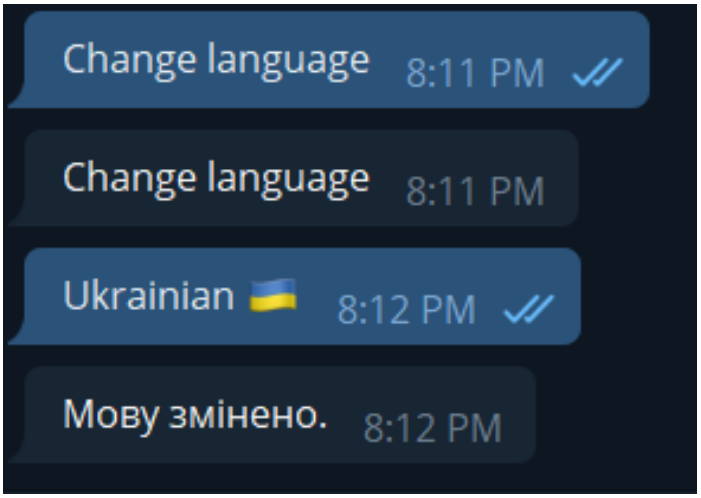

Рисунок 3.15. Зміна мови інтерфейсу

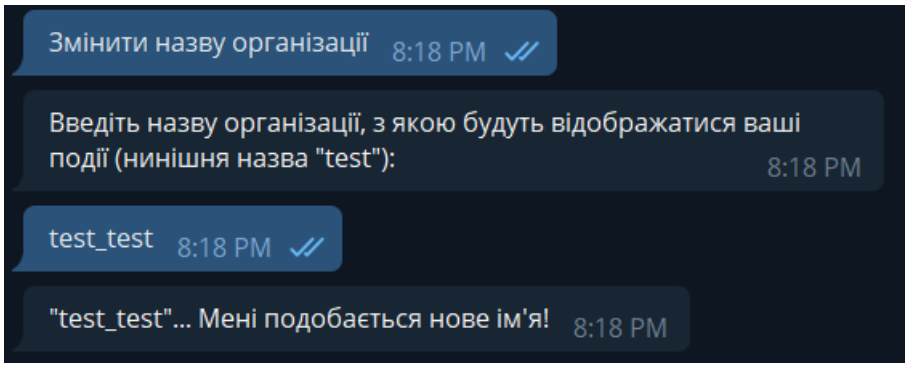

Рисунок 3.16. Зміна назви організації

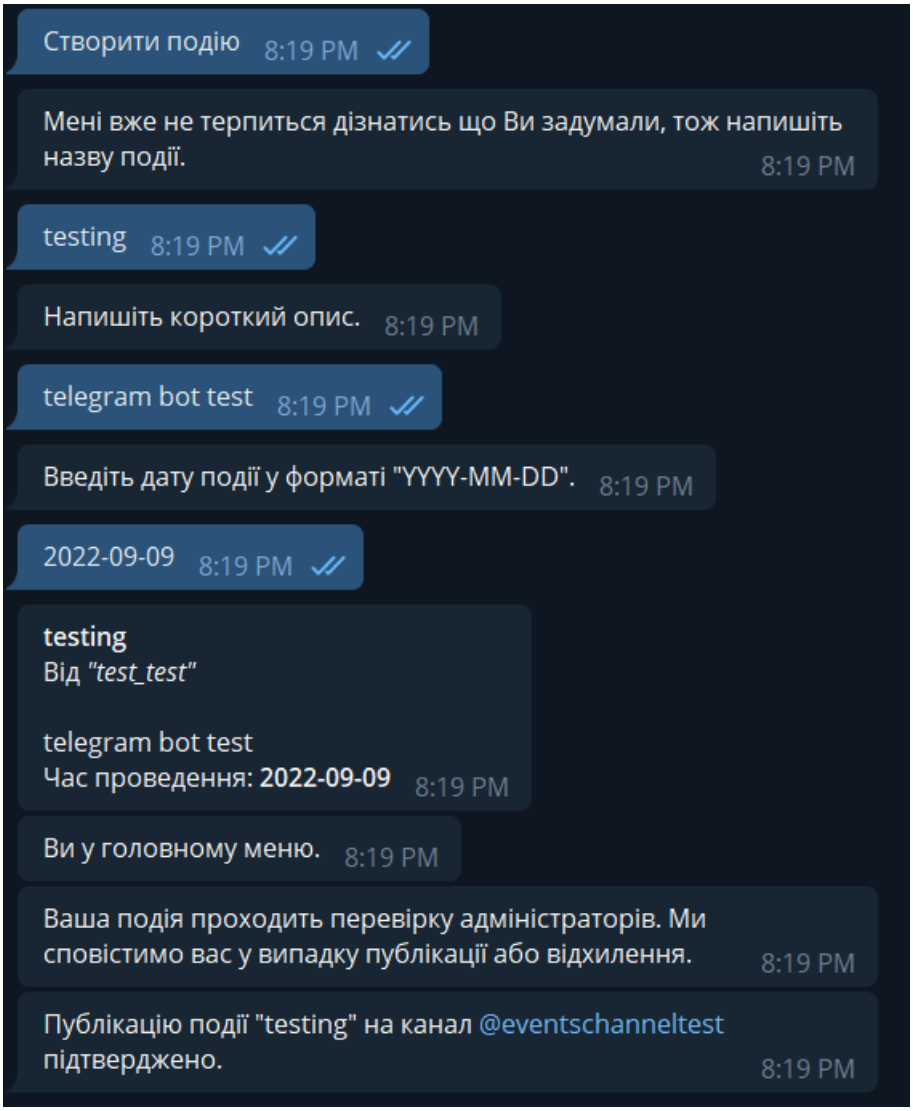

Рисунок 3.17. Публікація події

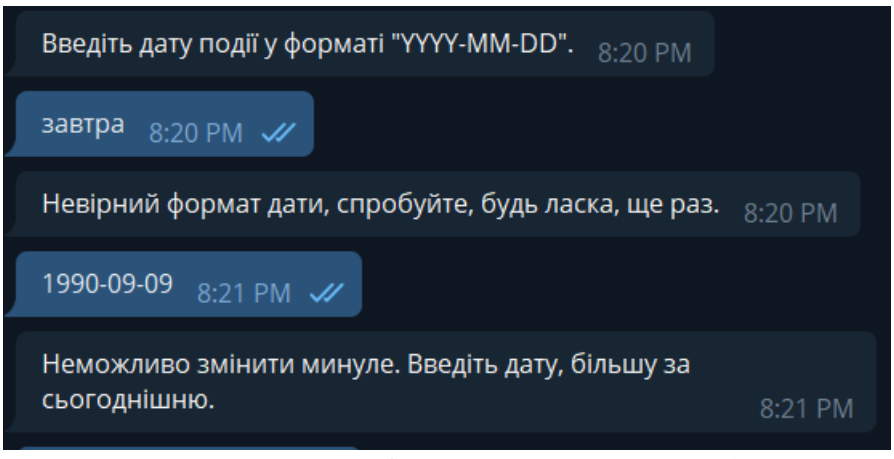

Рисунок 3.18. Валідація дати за форматом та часом (підходять дати від завтрашньої включно)

Висновок до курсової роботи

В першому розділі курсової роботи були розглянуті теоретичні поняття про чат-ботів. Було надане чітке визначення та класифікація чат-ботів, як за алгоритмами роботи (декларативні, предикативні) так і за їх застосуваннями. Були наведені практичні ситуації з життя, в яких було б доцільно використовувати чат-ботів, та реальні приклади використання чат-ботів у різних сферах. Також були наведені переваги використання чат-ботів в сьогоднішніх умовах, що обгрунтовує їх актуальність.

Другий розділ був присвячений опису методів розробки чат-ботів. У цьому розділі був описаний процес створення чат-боту в мессенджері Telegram за допомогою бота @BotFather, наданий опис набору технологій, що використовувалися у практичній частині, а саме:

- Мова програмування JavaScript.
- Середовище виконання Node.js.
- Сервіс хмарного хостингу Heroku.
- СКБД PostgreSQL.
- Фреймворк для розробки телеграм-ботів «telegraf».

В третьому розділі, базуючись на наведеній раніше теоретичній інформації, детально описується процес розробки чат-боту. Наводяться інформація про пакетний менеджер «npm» для середовища виконання Node.js, розглядаються методи фреймворку «telegraf» розробки чат-ботів у Telegram. Наводиться ER-модель бази даних, що використовується в роботі бота, приклади коду з описом принципу роботи усіх складових чат-боту для обробки повідомлень та колбеків, роботи з базою даних. Приведені скріншоти з прикладами різних сценаріїв діалогу фінального продукту. Бот доступний для тестування в Telegram за унікальним іменем: **@eventboard_test_bot**.

Список джерел

[1] - Chatbots: History, technology, and applications https://reader.elsevier.com/reader/sd/pii/S2666827020300062?token=B161F51B04F160462626EA9015ABDD4531702C08524E2E2F1AB7F962F720E047FD81B1EF7085407788E98FD765A67802&originRegion=eu-west-1&originCreation=20220530192319

[2] - Review of Chatbots Design Techniques https://www.researchgate.net/profile/Nahdatul-Akma-Ahmad/publication/327097910_Review_of_Chatbots_Design_Techniques/links/5b77cf3e4585151fd11cd905/Review-of-Chatbots-Design-Techniques.pdf

[3] – What is a chatbot https://www.oracle.com/chatbots/what-is-a-chatbot/

[4] – Corezoid Bot Platform https://corezoid.com/bot-platform/

[5] - Visual Programming Margaret M.Burnett https://www.cs.auckland.ac.nz/courses/compsci732s1c/archive/2005/lectures/WhatIsVP.pdf

[6] – When to use webhooks https://hookdeck.com/webhooks/guides/when-to-use-webhooks/

[7] - The benefits of PostgreSQL https://www.prisma.io/dataguide/postgresql/benefits-of-postgresql

[8] -Heroku Add-ons https://elements.heroku.com/addons

[9] – Документація Telegraf.js https://telegraf.js.org

[11] – Стаття про чат-ботів на Вікіпедії https://en.wikipedia.org/wiki/Chatbot

[10] – Telegram API https://core.telegram.org/api

[12] -Stop Russian War Bot https://www.ukrinform.ua/rubric-ato/3413790-v-ukraini-zavivsa-catbot-kudi-mozna-povidomlati-pro-vorozu-tehniku-ta-diversantiv-sbu.html