

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Створення телеграм боту для Молодіжного центру
Житомирщини»**

Виконав студент групи ІПЗ-20-1

НЕТРЕБИЦЬКА Анастасія Володимирівна

Керівник роботи:

ДУНЄВ Сергій Сергійович

Рецензент:

ВАКАЛЮК Тетяна Анатоліївна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **НЕТРЕБИЦЬКА Анастасія Володимирівна**
Керівник роботи: **ДУНЄВ Сергій Сергійович**
Тема роботи: « **Створення телеграм боту для Молодіжного центру Житомирщини**», затверджена Наказом закладу вищої освіти від «23» лютого 2024 р., №74/с Вихідні дані для роботи: **створення телеграм боту для молодіжного центру з автоматичним оновленням розкладу та системою оповіщень**. Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Дунєв С.С.	02.03.2024	02.03.2024
2	Дунєв С.С.	04.04.2024	04.04.2024
3	Дунєв С.С.	02.05.2024	02.05.2024

РЕФЕРАТ

Випускна робота бакалавра складається з комплексної програмної реалізації Телеграм-бота для Молодіжного центру, а також пояснювальної записки. Пояснювальна записка до дипломної роботи містить 55 сторінок, 23 ілюстрації, 5 таблиць, 4 додатки та 8 літературних джерел.

Основною метою даної роботи є розробка алгоритмів та реалізація програмного продукту, що надаватиме широкий спектр інструментів для надсилання сповіщень про майбутні події. В роботі визначено ключові завдання, що пов'язані з розробкою бота, проведено аналіз існуючого інструментарію, подібного до розроблюваного, а також наведено сценарії роботи програмного продукту.

Представлено огляд загальної структури додатку, описано алгоритми взаємодії між окремими компонентами, а також наведено опис об'єктної структури (діаграму класів) додатку. В рамках випускної роботи проводились експерименти по оптимізації процесу надсилання сповіщень, в результаті яких були визначені оптимальні налаштування для швидкого та зручного реагування на події у місті.

КЛЮЧОВІ СЛОВА: ТЕЛЕГРАМ-БОТ, МОЛОДІЖНИЙ ЦЕНТР, СПОВІЩЕННЯ, ПОДІЇ, ПРОГРАМУВАННЯ.

					ІПЗ.КР.Б – 121 – 23 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Створення телеграм боту для Молодіжного центру Житомирщини Пояснювальна записка	Літ.	Арк.	Акрушів
Розроб.		А.В. Нетребицька					2	55
Керівник		С.С. Дунев						
Рецензент.		Т.А. Вакалюк				Житомирська політехніка, група ІПЗ-20-1		
Зав. каф.		Т.А. Вакалюк						

ABSTRACT

The bachelor's thesis consists of a complex software implementation of a Telegram bot for the Youth Center, as well as an explanatory note. The explanatory note to the thesis contains 55 pages, 23 illustrations, 5 tables, 4 appendices, and 8 references.

The main purpose of this work is to develop algorithms and implement a software product that will provide a wide range of tools for sending notifications about upcoming events. The paper identifies key tasks related to the development of the bot, analyzes existing tools similar to the one under development, and provides scenarios for the operation of the software product.

An overview of the general structure of the application is presented, the algorithms of interaction between individual components are described, and the object structure (class diagram) of the application is described. As part of the graduation work, experiments were conducted to optimize the process of sending notifications, as a result of which the optimal settings were determined for a quick and convenient response to events in the city.

KEYWORDS: TELEGRAM BOT, YOUTH CENTER, NOTIFICATIONS, EVENTS, PROGRAMMING.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	5
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ.....	8
1.1 Постановка задачі	8
1.2 Аналіз аналогів програмного продукту	9
1.3 Вибір архітектури при розробці боту	13
1.4 Обґрунтування вибору інструментальних засобів та вимог до апаратного забезпечення	15
Висновки до розділу 1	16
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТЕЛЕГРАМ БОТУ ДЛЯ МОЛОДІЖНОГО ЦЕНТРУ	17
2.1 Визначення варіантів використання та об'єктно орієнтованої структури 17	
2.2 Опис структури даних для телеграм-бота Молодіжного центру.....	22
2.3 Проектування та реалізація алгоритмів роботи системи.....	23
2.4 Реалізація функціоналу телеграм боту для молодіжного центру	30
Висновки до розділу 2	36
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ТЕЛЕГРАМ БОТОМ ДЛЯ МОЛОДІЖНОГО ЦЕНТРУ	38
3.1 Порядок встановлення та налаштування системи.....	38
3.2 Інтерфейс та порядок роботи.....	40
3.3 Тестування роботи телеграм-бота для молодіжного центру.....	49
Висновки до розділу 3	51
ВИСНОВКИ	53
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	55
ДОДАТКИ.....	57
ДОДАТОК А	58
ДОДАТОК Б.....	59
ДОДАТОК В.....	62
ДОДАТОК Г	63

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – Програмне забезпечення;

ОС – Операційна система;

UML – Unified Modeling Language (Уніфікована мова моделювання);

API – Application Programming Interface (Інтерфейс програмування додатку)

SQL – Structured Query Language (Мова структурованих запитів)

FAQ – Frequently Asked Questions (Поширені запитання)

БД – База даних

ВСТУП

У сучасному світі, де інформаційні технології відіграють ключову роль, використання месенджерів, таких як Telegram, стає все більш поширеним. Особливо це стосується молодіжних центрів, які шукають ефективні способи комунікації зі своєю аудиторією. **Актуальність теми полягає в тому**, що створення Telegram-бота для молодіжного центру може значно полегшити процес інформування учасників про майбутні заходи.

Метою кваліфікаційної роботи є розробка та впровадження Telegram-бота, який буде надсилати повідомлення про майбутні події учасникам молодіжного центру. Цей бот допоможе учасникам бути в курсі всіх подій та більш ефективно планувати свій час.

Для досягнення мети необхідно вирішити наступні завдання: проаналізувати вимоги до бота, розробити архітектуру та алгоритми бота, реалізувати бота за допомогою Python та відповідних бібліотек, протестувати та впровадити бота.

Об'єктом дослідження є процес інформування учасників молодіжного центру про майбутні події.

Предметом дослідження є Telegram-бот, який автоматизує процес інформування учасників про майбутні заходи.

Ця робота спрямована на використання сучасних технологій для поліпшення процесу комунікації між молодіжними центрами та їхніми учасниками. Завдяки цьому молодіжні центри зможуть ефективніше взаємодіяти зі своєю аудиторією та забезпечити високий рівень участі в своїх заходах.

Крім того, необхідно врахувати специфіку молодіжного центру, його цілі та потреби аудиторії. Тому ця робота включає не тільки технічну частину, але й аналіз потреб користувачів, проектування інтерфейсу та тестування.

У цій роботі було розглянуто різні аспекти розробки Телеграм-бота, включаючи вибір технологій, проектування архітектури, реалізацію основних функцій та тестування. Крім того, буде проведено аналіз можливих проблем та викликів, з якими може зіткнутися розробник під час розробки бота.

В результаті цієї роботи було отримано функціонального Телеграм-бота, який може бути використаний молодіжним центром для інформування учасників про майбутні події. Це допоможе молодіжному центру покращити комунікацію з учасниками та збільшити їхню участь в заходах центру.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1 Постановка задачі

Основною метою розробки Телеграм-бота для молодіжного центру є надання користувачам інструментів для отримання сповіщень про майбутні події. Впровадження бота дозволить користувачам отримувати актуальну інформацію про події, що відбуваються в молодіжному центрі, та планувати свою участь в них.

Для реалізації поставленого завдання, основними етапами є:

1. Аналіз проекту та збір вимог:

- Проведення ретельного аналізу практики використання Телеграм-ботів;
- Визначення основних характеристик і функціональних можливостей бота;
- Аналіз аналогів програмного продукту.

2. Вибір та інтеграція технологій:

- Дослідження та вибір платформи розробки;
- Інтеграція стандартних інструментів та фреймворків, на яких буде працювати бот;
- Впровадження розгортання програмного продукту на будь-яку платформу, що підтримує Telegram.

3. Розробка та впровадження системи:

- Розробка інтуїтивно зрозумілого та зручного інтерфейсу, який включає різні інструменти для отримання сповіщень про події;
- Реалізація основних функцій: такі як отримання сповіщень про події, підтвердження участі в подіях, відміна участі в подіях;
- Реалізація можливостей до редагування: зміна налаштувань сповіщень, вибір подій для відстеження;

- Обробка даних: зберігання інформації про користувачів та їхні налаштування, обробка запитів користувачів, відправка сповіщень.

4. Тестування та забезпечення якості:

- Проведення ретельного тестування, щоб гарантувати, що програмне рішення працює за призначенням і відповідає вимогам;
- Впровадження процесів забезпечення якості для виявлення та вирішення будь-яких проблем або помилок, які можуть виникнути на етапах розробки та тестування.

5. Розгортання та підтримка:

- Розгортання програмного рішення в середовищі Telegram;
- Надання технічної підтримки та оновлень для вирішення проблем та вдосконалення функціональності.

Успішна розробка та впровадження цього програмного забезпечення дозволить користувачам молодіжного центру отримувати актуальну інформацію про події в зручному та ефективному форматі. Це сприятиме підвищенню рівня участі в заходах центру та поліпшенню комунікації між центром та його учасниками.

1.2 Аналіз аналогів програмного продукту

Для розробки ефективного та корисного Телеграм-бота для молодіжного центру важливо проаналізувати існуючі аналоги. Це допоможе зрозуміти, які функції вже пропонуються на ринку, визначити їхні сильні та слабкі сторони та виявити можливості для покращення.

Один з найпопулярніших ботів в Telegram - це боти для нагадувань. Вони дозволяють користувачам налаштовувати індивідуальні нагадування про події, завдання або дії. У цьому розділі ми проведемо аналіз трьох популярних ботів: ReminderBot, EventBot, ManyBot

ReminderBot - дозволяє користувачам встановлювати нагадування за допомогою простих команд. Однак, цей бот не має спеціалізованих функцій для молодіжних заходів, таких як автоматичне створення нагадувань для подій, що проводяться або інтеграцій з іншими сайтами. Крім того він має доволі маленький функціонал (рис. 1.1).

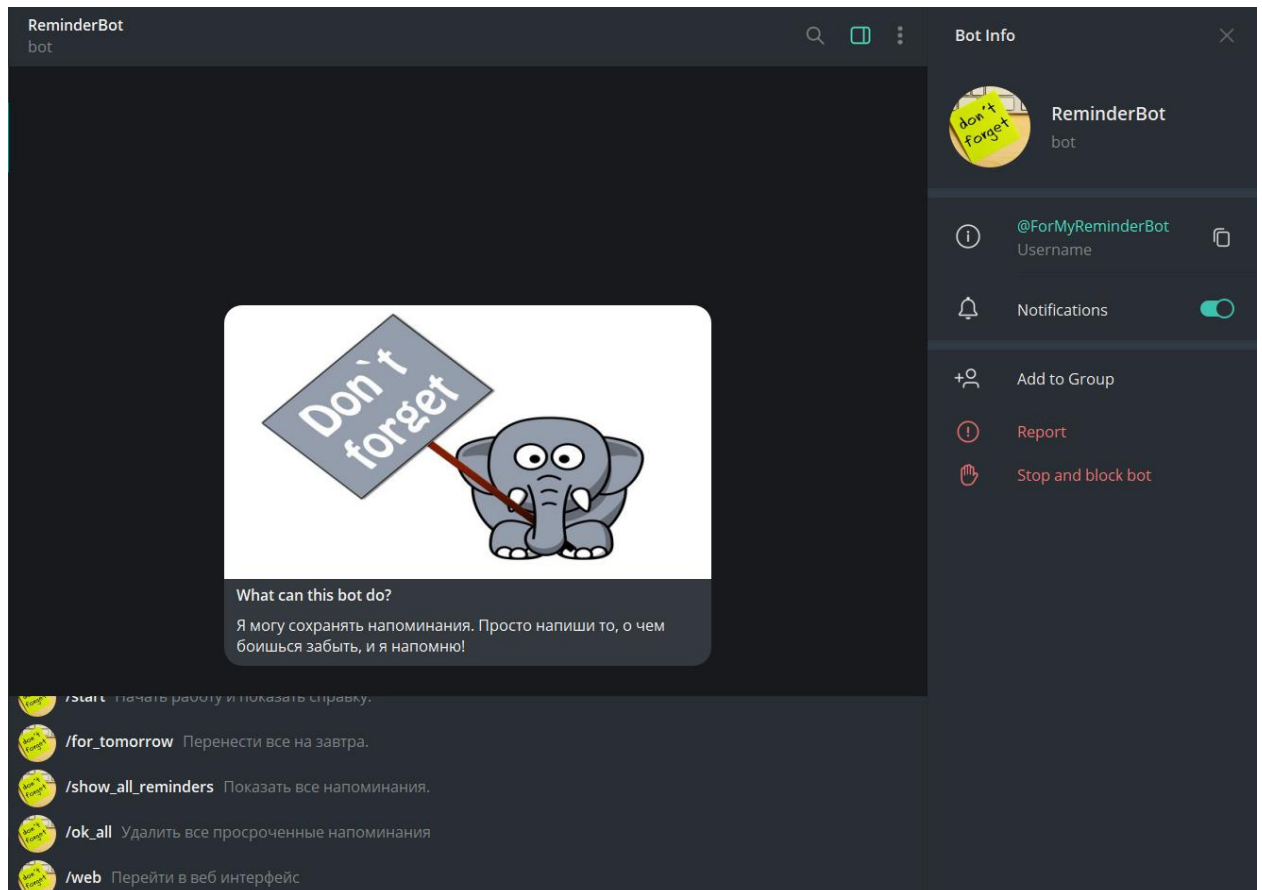


Рис. 1.1. – Інтерфейс ReminderBot.

EventBot - Інший популярний бот який спеціалізується на створенні та керуванні подіями в Telegram. Він має більш розширений функціонал та дозволяє користувачам може інтегруватися з чатами з великою кількістю людей та надсилати нагадування. Однак, цей бот також не має спеціалізованих функцій для молодіжних заходів та автоматичного додавання заходів до бази, всі нагадування потрібно налаштовувати вручну (рис. 1.2).

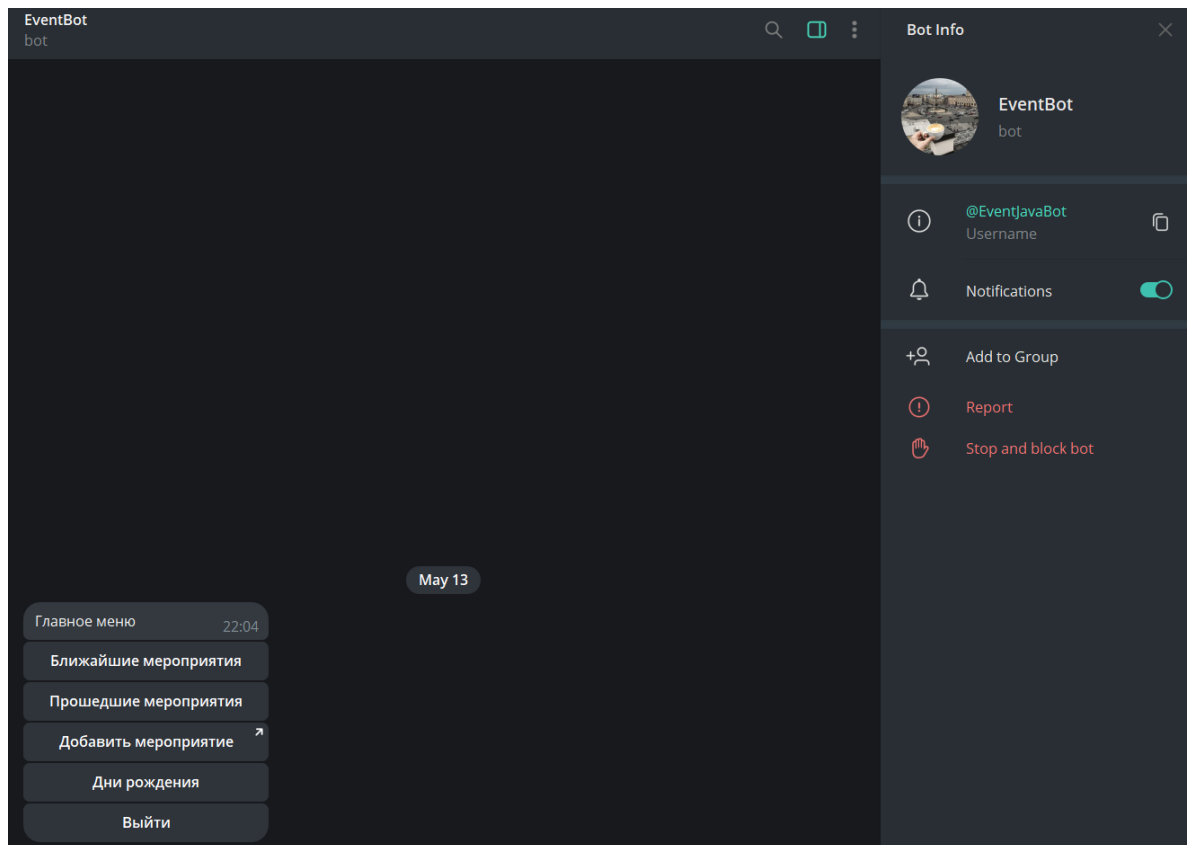


Рис. 1.2. – Интерфейс EventBot

Manybot – є ще одним аналогом. Цей бот дозволяє створювати персоналізовані боти в Telegram без необхідності кодування. Він пропонує різноманітні функції, включаючи надсилання повідомлень, створення меню, проведення опитувань та інше. Однак, він потребує попереднього налаштування користувачем, що може викликати складнощі, і його функціонал буде обмеженим функціями передбаченими розробником цього бота (рис. 1.3).

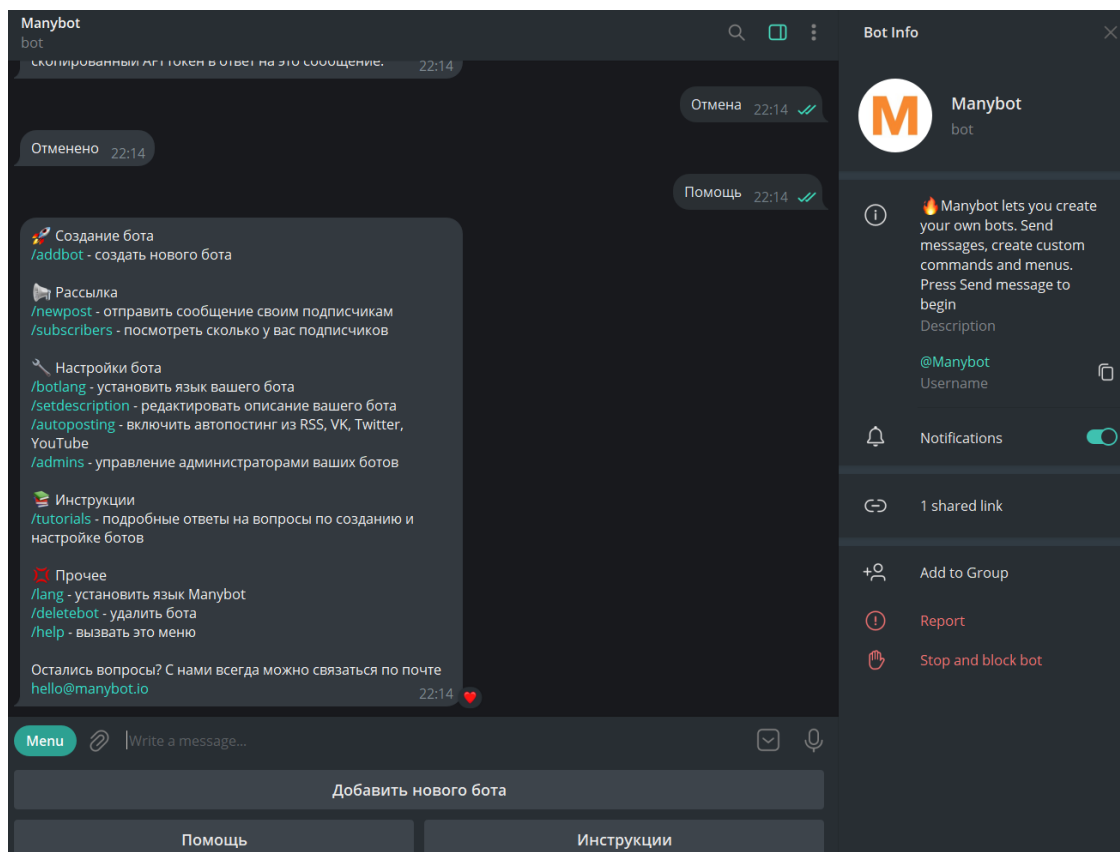


Рис. 1.3. – Интерфейс ManyBot

На основі цього аналізу можна зробити висновок, що існують різні Телеграм-боти, які пропонують широкий спектр функцій для керування подіями та нагадувань. Однак, жоден з них не має спеціалізованих функцій для заходів. Це вказує на наявність прогалини на ринку, яку може заповнити розробка нового Телеграм-бота, спеціально розробленого для молодіжних заходів.

Такий бот може включати функції, такі як автоматичне створення нагадувань для подій, що проводяться, персоналізація нагадувань на основі інтересів користувача, інтеграція з календарем подій сайту та інше. Це допоможе заходам ефективніше взаємодіяти зі своєю аудиторією та забезпечити високий рівень участі в них.

Таким чином, розробка Телеграм-бота для молодіжного центру є актуальною та важливою задачею, яка може принести значну користь як для молодіжних заходів, так і для їхніх учасників.

Таблиця 1.1

Порівняння існуючих аналогів ботів для повідомлень

Конкурент	Плюси	Мінуси
ReminderBot	- Простота використання - Можливість встановлювати індивідуальні нагадування	- Відсутність спеціалізованих функцій для молодіжних центрів
EventBot	- Спеціалізовані функції для керування подіями - Можливість інтеграції за чатами та надсилати нагадування	- Відсутність спеціалізованих функцій для молодіжних центрів
Manybot	- Широкий спектр функцій - Можливість створювати персоналізовані боти без необхідності кодування	- Відсутність спеціалізованих функцій для молодіжних центрів

1.3 Вибір архітектури при розробці боту

При розробці Телеграм-бота для надсилання повідомлень та парсингу заходів з телеграм каналу з застосуванням бази даних важливо обрати правильну архітектуру. Архітектура програмного забезпечення визначає, як програма буде структурована та як її компоненти будуть взаємодіяти. Вона впливає на продуктивність, надійність, масштабованість та здатність до розширення програми.

Основними компонентами архітектури бота є:

Телеграм-бот: Це основний компонент, який відповідає за взаємодію з користувачами через Telegram. Він обробляє вхідні повідомлення, відправляє відповіді та управляє діалогами [1].

Сервіс парсингу заходів: Цей компонент відповідає за отримання інформації про заходи з телеграм каналу. Він використовує бібліотеку Telethon для збору даних та перетворює їх на структурований формат.

База даних: Це місце для зберігання інформації про заходи та користувачів. Вона використовується для зберігання, вибірки та оновлення даних.

Сервіс надсилання повідомлень: Цей компонент відповідає за надсилання сповіщень користувачам. Він використовує інформацію з бази даних та Телеграм-бота для створення та відправки повідомлень (рис. 1.4).

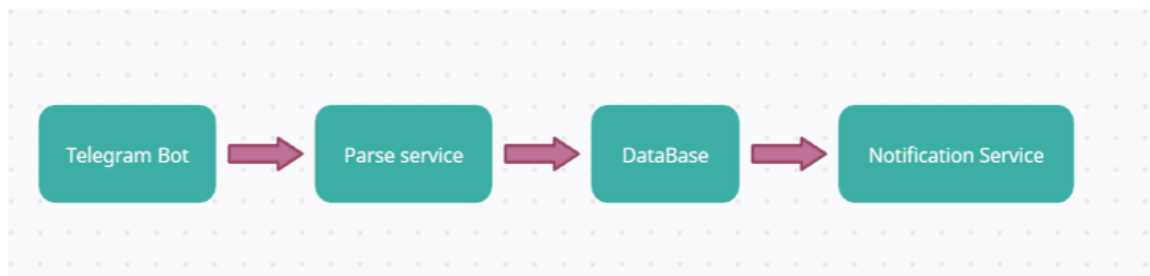


Рис. 1.4. – Архітектура Events Bot

Ці компоненти взаємодіють один з одним для забезпечення правильної роботи бота. Телеграм-бот отримує повідомлення від користувачів та передає їх до відповідних сервісів. Сервіс парсингу заходів збирає дані з телеграм каналу та оновлює базу даних. Сервіс надсилання повідомлень використовує ці дані для створення та відправки повідомлень користувачам.

При виборі архітектури було враховано такі фактори, як масштабованість, продуктивність, надійність та зручність розробки. Для цього проекту використано мікросервісну архітектуру, яка дозволяє кожному компоненту працювати незалежно та масштабуватися відповідно до потреб. Це також спрощує процес розробки, оскільки кожен компонент можна розробляти та тестувати окремо.

Використання такої архітектури, разом з використанням сучасних технологій та практик розробки, допомогло створити ефективний та надійний Телеграм-бот для повідомлень про актуальні заходи у місті.

При розробці Телеграм-бота важливо обрати правильні технології, які дозволять ефективно реалізувати всі необхідні функції та вимоги. Вибір технологій залежить від різних факторів, включаючи вимоги до продуктивності, масштабованості, надійності, а також від досвіду та навичок розробника.

1.4 Обґрунтування вибору інструментальних засобів та вимог до апаратного забезпечення

Ось декілька ключових технологій, які були використані при розробці Телеграм-бота:

Python - це високорівнева мова програмування, яка відома своєю простотою та читабельністю. Вона має велику кількість бібліотек та фреймворків, які значно спрощують розробку. Для розробки Телеграм-бота рекомендується використовувати бібліотеку aiogram, яка є асинхронною та дозволяє ефективно обробляти велику кількість запитів [8].

PostgreSQL - це потужна система управління реляційними базами даних з відкритим вихідним кодом. Вона підтримує широкий спектр типів даних та має гнучкі можливості для запитів. Вона також відома своєю надійністю та продуктивністю [7].

SQLAlchemy - це бібліотека Python для взаємодії з базами даних. Вона дозволяє виконувати SQL-запити за допомогою Python і підтримує різні системи управління базами даних, включаючи PostgreSQL [6][7].

Telethon - це бібліотека для роботи з Telegram API на мові Python. Вона надає зручні інструменти для створення та автоматизації ботів, а також для взаємодії з користувацькими акаунтами. У даному проекті Telethon була використана для розробки та реалізації функціоналу парсингу розкладу з каналу молодіжного центру [4].

AIOGram - це бібліотека Python для створення асинхронних ботів в Telegram. Вона надає високорівневий API для взаємодії з Telegram Bot API та дозволяє створювати складні боти з використанням асинхронного програмування [5].

Ці технології були обрані на основі їхньої надійності, продуктивності, спільноти підтримки та зручності використання. Вони дозволяють розробити зручний Телеграм-бот з використанням сучасних практик розробки та забезпечити його стабільну та ефективну роботу.

Висновки до розділу 1

У першому розділі було проведено детальний аналіз та планування проекту розробки Телеграм-бота для молодіжного центру. Було визначено актуальність теми, поставлено мету кваліфікаційної роботи та обумовлено мету розробки. Об'єкт та предмет дослідження були чітко визначені.

Було проведено аналіз аналогів програмного продукту, що дозволив виявити прогалини на ринку та можливості для покращення. Вибір архітектури був обґрунтований, враховуючи потреби проекту та вимоги до продуктивності, масштабованості та надійності.

Було обрано відповідні інструментальні засоби для розробки бота, включаючи Python, PostgreSQL, SQLAlchemy, Telethon та AIOGram. Вимоги до апаратного забезпечення були визначені на основі обраних технологій [2].

В цілому, перший розділ закладає сильну основу для подальшої розробки Телеграм-бота. Він визначає ключові напрямки роботи та встановлює чіткі цілі для наступних етапів проекту. Результати цього розділу будуть використані як основа для розробки, тестування та впровадження Телеграм-бота в наступних розділах.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТЕЛЕГРАМ БОТУ ДЛЯ МОЛОДІЖНО-ГО ЦЕНТРУ

2.1 Визначення варіантів використання та об'єктно орієнтованої структури

У даному підрозділі досліджується використання та структура бота для управління подіями. Підвищення інтересу до подій та їх ефективна організація стає актуальною задачею в сучасному світі. Використання чат-бота для цих цілей відкриває широкі можливості для спрощення комунікації та залучення цільової аудиторії.

Перед нами постає завдання розглянути функціональні та нефункціональні вимоги до роботи бота, визначити його можливі варіанти використання та описати об'єктно-орієнтовану структуру проекту. Це дозволить зрозуміти, як бот може використовуватися в реальному житті та як він організований всередині. Детальне вивчення цих аспектів допоможе досягти більшої ефективності та користі від використання чат-бота для управління подіями.

Функціональні вимоги:

- Підписка на нагадування про події: Користувач може підписатися на отримання нагадувань про певні події.
- Відписка від подій: Користувач може скасувати свою підписку на певну подію.
- Перегляд доступних подій: Користувач може переглянути список доступних подій та їх розклад.
- Нагадування про наближення подій: Користувач отримує нагадування про наближення подій за тиждень та за день до їх початку.

Нефункціональні вимоги:

- Ефективність: Бот повинен бути достатньо швидким та ефективним у відповіді на запити користувачів.
- Надійність: Система повинна бути надійною та стійкою до помилок.
- Зручність використання: Інтерфейс бота повинен бути зрозумілим та зручним для користувачів.
- Масштабованість: Система повинна бути легко масштабованою для можливості додавання нового функціоналу в майбутньому.
- Безпека: Бот повинен захищати конфіденційність користувачів та використовувати безпечні методи передачі та зберігання даних.

Часові характеристики реакції системи:

- Час відповіді: Бот повинен реагувати на запити користувачів максимум за кілька секунд.
- Періодичність нагадувань: Нагадування про наближення подій мають надсилатися користувачам регулярно, згідно з налаштованим розкладом.

Варіанти використання:

- Підписка на події: Користувачі можуть підписатися на події, які їх цікавлять, отримуючи нагадування про них.
- Скасування підписки: Користувачі можуть скасувати підписку на будь-яку подію, яка їх не цікавить більше.
- Перегляд доступних подій: Користувачі можуть переглянути список доступних подій та їх розклад, щоб обрати ті, які їх цікавлять.

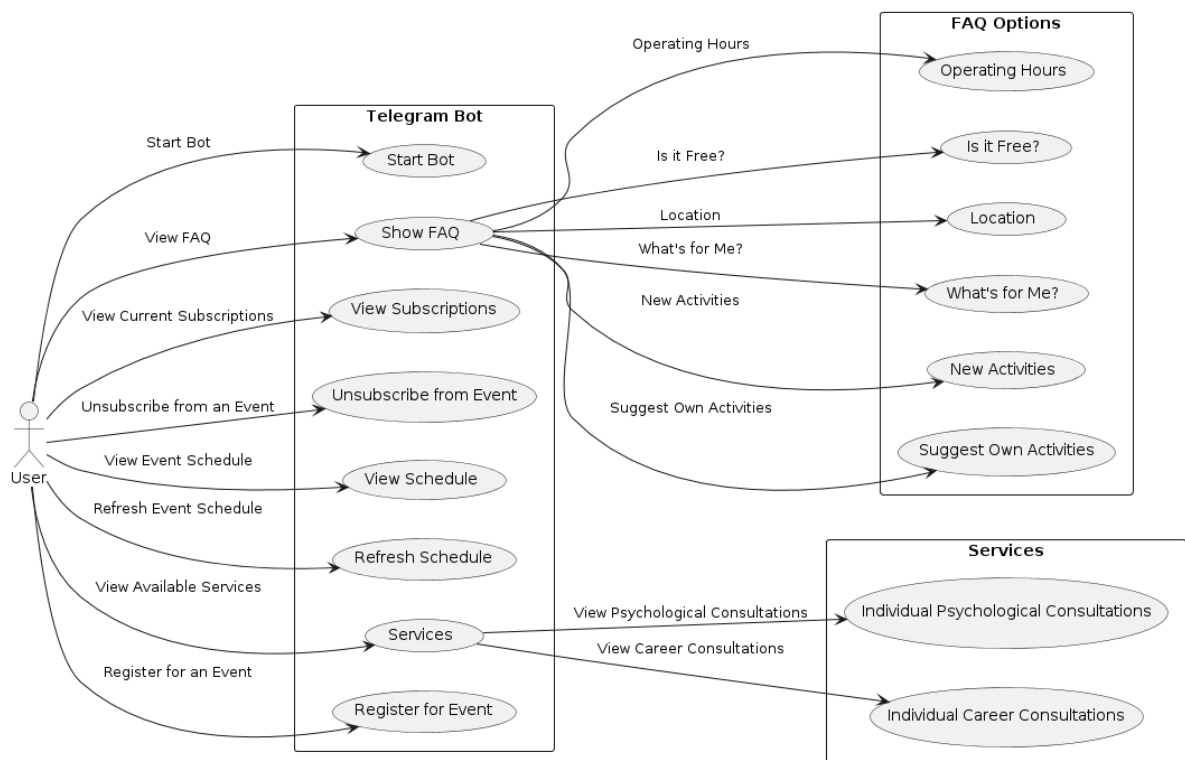


Рис. 2.1. – Діаграма варіантів використання

Діаграма варіантів використання (Use Case Diagram) є графічним зображенням того, як користувачі (актори) взаємодіють з системою для досягнення певних цілей. Вона допомагає зрозуміти функціональність системи та ідентифікувати всі можливі варіанти використання, що забезпечуються цією системою (рис 2.1).

Основні елементи діаграми варіантів використання:

- Актори: Це сутності або системи, які взаємодіють з системою і використовують її функціональність для досягнення своїх цілей.
- Варіанти використання (Use Cases): Це конкретні функції або можливості системи, які задовольняють потреби акторів.
- Взаємодія між акторами та варіантами використання: Вона відображає, як актори взаємодіють з варіантами використання для досягнення своїх цілей.
- Система: Це сама система, яка надає функціональність для акторів.

Об'єктно-орієнтована структура:

					ІПЗ.КР.Б – 121 – 23 – ІЗ	
						19

- Клас User: Представляє користувача бота зі збереженими інформацією про його ідентифікатор та ім'я.
- Клас Event: Представляє подію, яка може бути доступна для підписки.
- Клас Subscription: Відображає підписку користувача на певну подію.
- Клас Form: Використовується для зберігання станів форм та інтерактивної взаємодії з користувачем.
- Бізнес-логіка: Функціональність бота реалізована в окремих функціях та обробниках подій, які взаємодіють з класами користувачів, подій та підписок.

Принципи ООП які застосовуються:

- Класи: У проєкті визначено класи User, Event і Subscription, які представляють сутності користувачів, подій і підписок відповідно. Кожен клас містить свої атрибути і методи для роботи з даними.
- Успадкування: У цьому проєкті немає прямих прикладів успадкування, але SQLAlchemy, бібліотека для роботи з базами даних, використовує успадкування для визначення моделей даних [6].
- Інкапсуляція: Класи User, Event і Subscription інкапсулюють дані та методи, пов'язані з відповідними сутностями.
- Поліморфізм: У цьому проєкті може бути реалізовано поліморфізм через використання методів з однаковими іменами для різних класів. Наприклад, метод send_notification може працювати з різними типами сутностей (користувачі, підписки тощо), надсилаючи їм повідомлення.

Таким чином, хоча основний фокус у цьому проєкті скоріше на функціональному програмуванні, об'єктно-орієнтовані концепції застосовують для впорядкування та структурування коду (рис. 2.2).

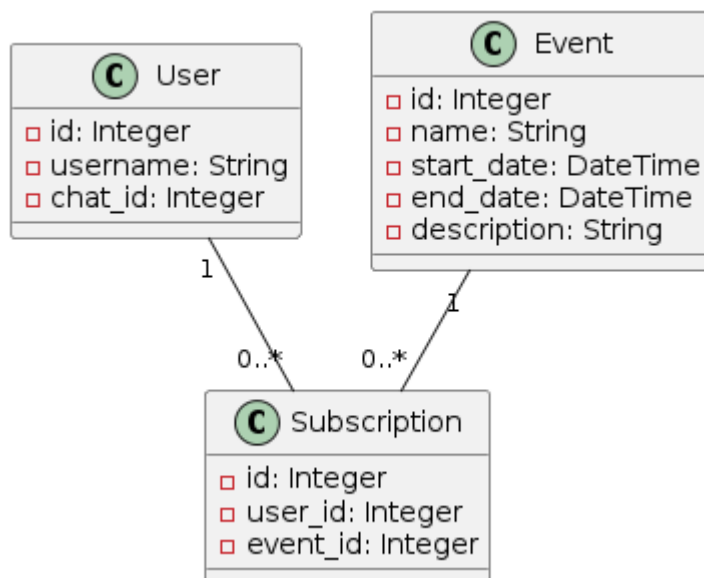


Рис. 2.2. Діаграма класів CDatabase

Діаграма класів (Class Diagram) є графічним зображенням структури системи, що використовується для відображення класів, їх атрибутів, методів та зв'язків між класами. Ця діаграма дозволяє візуалізувати структуру програми та її компонентів, що полегшує розуміння та розробку системи.

В діаграмі класів для бота управління подіями реалізовані такі класи:

- Клас User: Представляє користувача, який використовує бота і має ідентифікатор, ім'я та інші атрибути.
- Клас Event: Представляє подію, яка має назву, дату, опис та інші атрибути.
- Клас Subscription: Представляє підписку користувача на певну подію.

2.2 Опис структури даних для телеграм-бота Молодіжного центру

Структура даних, що використовується в телеграм-боті Молодіжного центру, включає в себе ряд ключових компонентів, які сприяють ефективному функціонуванню бота та забезпечують зручну взаємодію з користувачами. Нижче розглянемо кожен з цих компонентів докладніше:

Користувачі (Users):

Кожен користувач, що звертається до бота Молодіжного центру, зберігається як окремий об'єкт користувача. Кожен об'єкт користувача містить наступну інформацію:

- Ідентифікатор чату (Chat ID): Унікальний ідентифікатор, що визначає чат користувача з ботом.
- Ім'я користувача: Відображається в боті для ідентифікації користувача.
- Статус підписки: Вказує, чи підписаний користувач на отримання нагадувань та повідомлень від бота.

Події (Events):

Кожна подія, яка оголошується або редагується через бота Молодіжного центру, представляється об'єктом події. Кожен об'єкт події містить наступну інформацію:

- Назва події: Ідентифікатор події, який відображається в боті та ідентифікує конкретну подію.
- Дата та час проведення: Інформація про час та дату початку та завершення події.
- Опис: Докладний опис події, що допомагає користувачам зрозуміти її суть та значення.

Підписки (Subscriptions):

- Інформація про підписки користувачів на певні події зберігається в об'єктах підписки. Кожен об'єкт підписки включає:

- Користувача: Посилання на об'єкт користувача, який підписаний на подію.
- Подію: Посилання на об'єкт події, на яку користувач підписаний.

Стан (State):

Для відстеження поточного стану роботи бота та забезпечення логіки переходів між різними етапами використовується система станів (State Machine). Кожен об'єкт стану визначає поточний контекст роботи бота та можливі подальші кроки.

Інтерфейс (Interfaces):

Для взаємодії з користувачами через бота використовуються різні інтерфейси, такі як текстові повідомлення, клавіатура, кнопки тощо. Інтерфейси забезпечують користувачам зручний спосіб взаємодії з ботом та його функціями.

Ця детально структурована модель даних дозволяє ефективно зберігати та обробляти інформацію про користувачів, події та їх підписки, забезпечуючи оптимальне функціонування та взаємодію телеграм-бота Молодіжного центру з користувачами.

2.3 Проектування та реалізація алгоритмів роботи системи

У цьому розділі ми детально розглянемо алгоритми та процеси, які були використані для проектування та реалізації системи управління подіями за допомогою телеграм-бота для Молодіжного центру. Розглянемо кожну з ключових функціональних можливостей бота та їх вплив на роботу системи.

Автоматизований збір даних

Одним із важливих аспектів системи є забезпечення актуальності та повноти інформації про події. Для цього ми використовуємо модуль `parse.py`, який використовує бібліотеку `Telethon` для автоматизованого збору даних з телеграм каналу, де публікуються події. Цей модуль дозволяє нам отримувати

ти інформацію про події, таку як назва, місце, дата та час проведення, опис, тощо. Після отримання цих даних вони зберігаються у базі даних (рис. 2.3).



Рис. 2.3. - Діаграма активності для автоматизованого збору даних

На основі діаграми активності для збору даних було створено наступну таблицю специфікації:

Таблиця 2.1

Ім'я варіанту використання: Автоматизований збір даних		Рівень: високий
Основний актор:		користувач
Короткий опис:		Автоматизований збір даних
Передумови:		Запуск бота, реєстрація
Відносини:		
Включення:		Команда /refresh
Основний потік:	отримувати інформацію про події, таку як назва, місце, дата та час проведення, опис, тощо. Після отримання цих даних вони зберігаються у базі даних.	

Керування підписками

Однією з ключових можливостей бота є можливість користувача підписатися на певні події та отримувати сповіщення про них. Для цього був реалізований алгоритм керування підписками. Користувач може додавати та видаляти підписки на події, що дозволяє йому отримувати або припиняти отримання сповіщень (рис. 2.4).

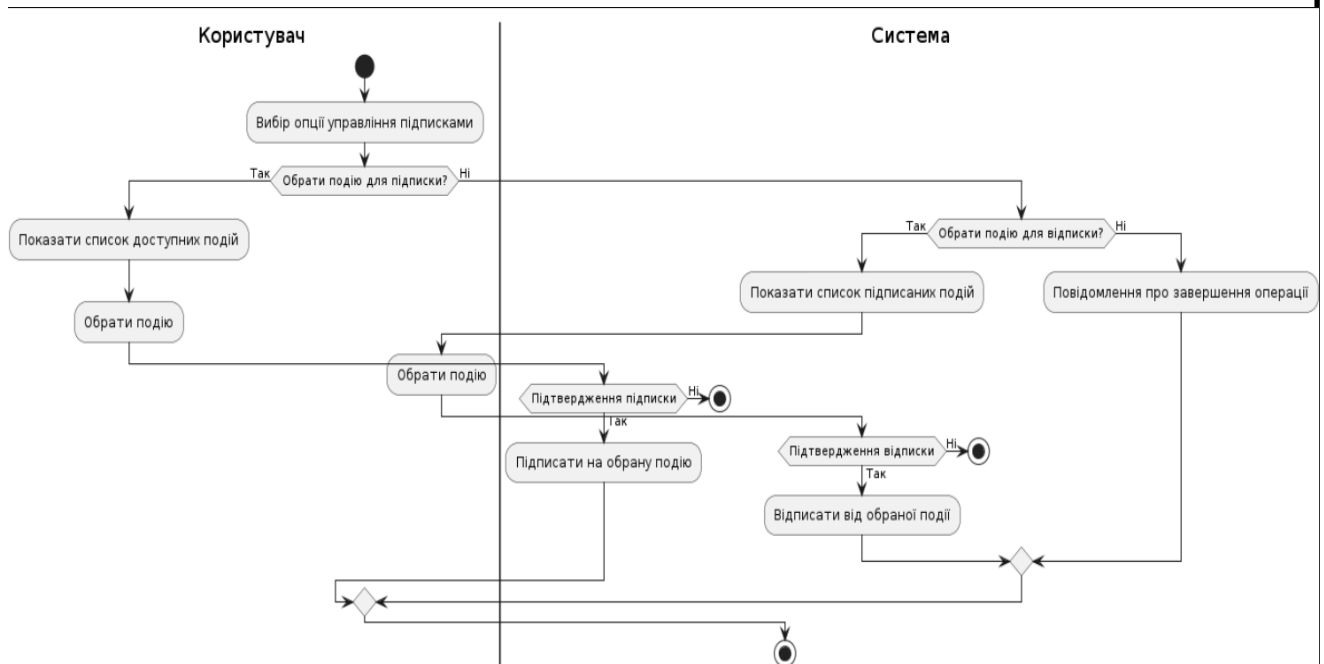


Рис. 2.4. Діаграма активності для керування підписками

Ця діаграма активності моделює процес керування підписками користувача на події в системі. Починаючи зі стартового вузла, користувач обирає опцію управління підписками. Далі, якщо користувач хоче підписатися на нову подію, система показує список доступних подій, з яких він може обрати. Після вибору події система запитує підтвердження від користувача і підписує його на обрану подію. Якщо користувач хоче відписатися від певної

події, він обирає відповідну опцію, після чого система показує список підписаних подій. Після вибору події користувач підтверджує своє бажання відписатися,

і система виконує цю операцію. У кінці користувач отримує повідомлення про завершення операції.

На основі діаграми активності для керування підписками було створено наступну таблицю специфікації:

Таблиця 2.2

Ім'я варіанту використання: Керування підписками		Рівень: Середній
Основний актор:		користувач
Короткий опис:		Керування підписками користувача на події
Передумови:		Запустити програму
Відносини:		
Включення:		Команда /subscriptions
Основний потік:	Користувач обирає опцію управління підписками, користувач обирає подію для підписки або відписки система підтверджує операцію і здійснює відповідні дії.	
Альтернативні/виключні потоки:		Користувач не обирає подію для підписки або відписки; Користувач не підтверджує операцію.

Система нагадувань

Щоб забезпечити своєчасне інформування користувачів про наближення подій, була реалізована система нагадувань. Ця система автоматично перевіряє наближення дати проведення подій та надсилає користувачам повідомлення про це. Нагадування може бути відправлене за декілька днів або за один день до події, щоб користувачі мали достатньо часу на підготовку (рис. 2.5).



Рис. 2.5. Діаграма активності для системи нагадувань

Ця діаграма активності представляє процес нагадування користувачам про наближення певних подій. Починаючи зі стартового вузла, система перевіряє наявність подій, для яких потрібно надсилати нагадування. Якщо такі події є, система надсилає нагадування користувачам. У випадку, якщо немає подій для нагадування, система повідомляє про це.

На основі діаграми активності для системи нагадувань було створено наступну таблицю специфікації:

Таблиця 2.3

Ім'я варіанту використання: Система нагадувань		Рівень: Високий
Основний актор:		система
Короткий опис:		Нагадування користувачам про наближення певних подій
Передумови:		Робота системи
Відносини:		
Включення:		
Основний потік:	Система перевіряє наявність подій для нагадування; Якщо є такі події, система надсилає нагадування користувачам; У протилежному випадку, система повідомляє про відсутність подій для нагадування.	
Альтернативні/виключні потоки:		Немає

Взаємодія з користувачами

Бот забезпечує зручний інтерфейс для взаємодії з користувачами. Він реагує на команди, надіслані користувачами, і надає необхідну інформацію про події та їх розклад. Користувач може з легкістю підписатися на певні події, переглянути свої підписки, отримати розклад подій, а також скасування підписок на події (рис. 2.6).

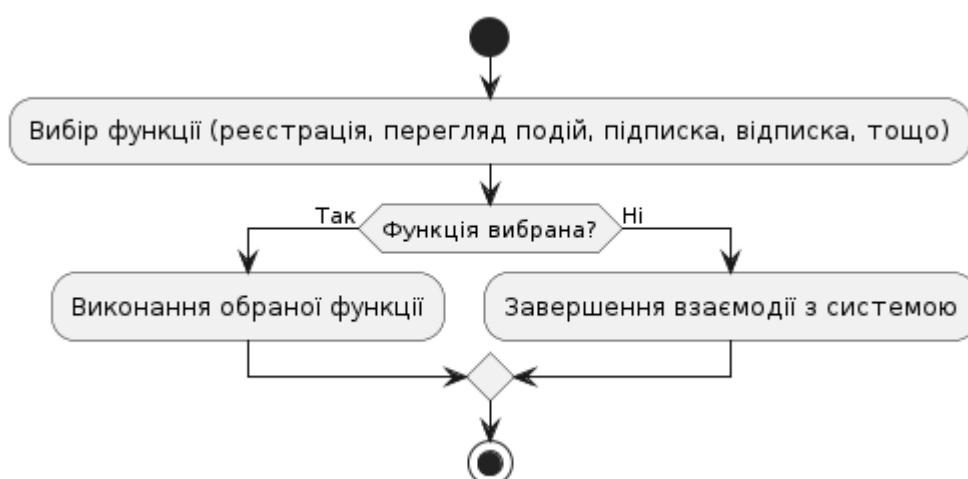


Рис. 2.6. Діаграма активності для взаємодії з користувачами

Ця діаграма активності представляє процес взаємодії користувача з системою. Користувач починає з вибору функції (наприклад, реєстрація, перегляд подій, підписка на події, відписка від подій тощо). Якщо користувач вибрав функцію, система виконує її. У протилежному випадку, взаємодія з системою завершується.

На основі діаграми активності для системи нагадувань було створено наступну таблицю специфікації:

Таблиця 2.4

Ім'я варіанту використання: Система нагадувань	Рівень: Високий
Основний актор:	Користувач
Короткий опис:	Взаємодія користувача з системою
Передумови:	Робота системи

Відносини:		
Включення:		
Основний потік:	Користувач обирає функцію (реєстрація, перегляд подій, підписка на події, відписка від подій тощо); Якщо функція вибрана, система виконує її; У протилежному випадку, взаємодія з системою завершується.	
Альтернативні/виключні потоки:		Немає

Було розглянуто основні функціональні можливості системи, такі як створення нових проектів, керування підписками на події, нагадування користувачам про наближення подій, і взаємодія з користувачами через телеграм-бот.

Для кожної функції були розроблені та реалізовані відповідні алгоритми, використовуючи мову програмування Python та бібліотеки, такі як aiogram для роботи з телеграм-ботом і SQLAlchemy для взаємодії з базою даних. Також були побудовані діаграми активності та діаграми послідовності, які відображають взаємодію користувачів з системою та внутрішній процес роботи системи [8].

Реалізація цих алгоритмів дозволяє користувачам зручно керувати подіями, на які вони підписані, отримувати нагадування про них та взаємодіяти з системою у найбільш зручний для них спосіб.

Діаграма послідовності - це тип діаграми UML, яка відображає послідовність обміну повідомленнями між об'єктами в рамках конкретного сценарію взаємодії. Вона показує порядок виконання різних елементів системи та взаємодію між ними протягом певного процесу або функції. Діаграми послідовностей допомагають уявити, як об'єкти взаємодіють між собою в рамках конкретного сценарію та як кожен крок цього процесу впливає на інші елементи системи.

Діаграма послідовності, що описує дії користувача, який взаємодіє з телеграм ботом. (Додаток А)

Ця діаграма послідовності представляє взаємодію користувача з телеграм-ботом для управління подіями. Користувач спочатку надсилає команду /start для початку спілкування з ботом. Бот вітає користувача і пропонує зареєструватися. Після отримання команди /register від користувача, бот зберігає нового користувача в базі даних і повідомляє про це. Потім, після отримання команди /events, бот шукає доступні події та пропонує користувачу вибрати подію для підписки. Користувач обирає подію, і бот показує інформацію про неї, а потім підтверджує підписку. Бот зберігає підписку в базі даних і повідомляє користувача про успішну підписку.

2.4 Реалізація функціоналу телеграм боту для молодіжного центру

У цьому розділі ми розглянемо процес реалізації функціоналу телеграм-боту для Молодіжного центру. Телеграм-бот створено з метою полегшення процесу управління подіями та сповіщенням користувачів про актуальні події, що відбуваються.

Ми детально розглянемо кожен аспект реалізації функціоналу телеграм-боту для Молодіжного центру. Ми розглянемо процес створення та інтеграції різних компонентів, таких як обробники повідомлень, база даних, парсер даних про події та інші. Кожен крок розробки буде детально проаналізовано, починаючи від структури бази даних та закінчуючи реалізацією важливих функцій, таких як керування підписками та система нагадувань.

Ми також вивчимо технічні деталі реалізації, такі як використання бібліотеки aiogram для роботи з API телеграму, парсинг даних з телеграм каналів для отримання інформації про події та організацію асинхронних процесів для нагадування користувачів про майбутні події.

Вибір мови програмування Python

Мову програмування Python було обрано для реалізації телеграм бота з кількох причин:

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						30

Простота та зручність синтаксису: Python має простий та зрозумілий синтаксис, що дозволяє швидко розробляти та тестувати код. Це особливо важливо для швидкої розробки прототипів та реалізації функціональності бота [8].

Розширена екосистема бібліотек: Python має багату екосистему бібліотек, що підтримує інтеграцію з багатьма сервісами та платформами. Зокрема, існує велика кількість бібліотек для роботи з API Telegram [2].

Підтримка асинхронного програмування: Python підтримує асинхронне програмування через бібліотеки, такі як asyncio, що дозволяє ефективно обробляти велику кількість одночасних запитів, зменшуючи затримки.

Вибір бібліотеки aiogram

Для реалізації телеграм бота було обрано бібліотеку aiogram через такі переваги:

Асинхронність: aiogram побудована на основі asyncio, що дозволяє писати асинхронний код, який є більш ефективним у плані ресурсів та продуктивності при обробці численних запитів від користувачів [5].

Гнучкість та розширюваність: Бібліотека aiogram має модульну архітектуру, що дозволяє легко розширювати функціональність бота та інтегрувати додаткові можливості без значних змін у коді.

Широкі можливості для налаштування: aiogram підтримує різні способи обробки повідомлень та команд, що робить її зручною для створення бота з широкими можливостями та складною логікою.

Приклади реалізації з коду програми

Ініціалізація бота та обробка команд:

```
# Ініціалізація бота та диспетчера
```

```
API_TOKEN = 'YOUR_API_TOKEN'
```

```
bot = Bot(token=API_TOKEN)
```

```
storage = MemoryStorage()
```

```

dp = Dispatcher(bot, storage=storage)
logging.basicConfig(level=logging.INFO)

@dp.message_handler(commands=['start'])
async def send_welcome(message: types.Message):
    await message.reply("Привіт! Я твій телеграм-бот для управління подіями. \nНадішли /help, щоб отримати допомогу")

@dp.message_handler(commands=['help'])
async def send_help(message: types.Message):
    await message.reply("Команди:\n/events - Зареєструватися на подію\n/subscriptions - Показати підписані події/відписатися\n/schedule - Розклад івентів\n/refresh - Оновити дані про події")

```

Наданий код описує:

- Імпорт необхідних бібліотек: Імпортуються бібліотеки aiogram для роботи з Telegram API, asyncio для асинхронного програмування та інші корисні модулі [1].
- Ініціалізація бота та диспетчера: Ініціалізується об'єкт Bot з використанням токена API Telegram. Створюється диспетчер Dispatcher, який буде керувати обробкою повідомлень.
- Обробка команди /start: Створюється обробник для команди /start, який надсилає привітальне повідомлення користувачу.
- Обробка команди /help: Створюється обробник для команди /help, який надсилає користувачу список доступних команд.

Реалізація команди для перегляду подій:

```
@dp.message_handler(commands=['events'])
async def show_events(message: types.Message):
    events = session.query(Event).all()
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for event in events:
        keyboard.add(types.KeyboardButton(event.name))
    await message.reply("Обери подію, на яку хочеш підписатися:",
reply_markup=keyboard)
```

Наданий код описує:

- Обробка команди /events: Створюється обробник для команди /events, який виконує запит до бази даних для отримання всіх подій.
- Створення клавіатури: Створюється клавіатура з кнопками для кожної події, використовуючи назву події як текст кнопки.
- Відправка повідомлення: Користувачу надсилається повідомлення з клавіатурою, що дозволяє вибрати подію для підписки.

Функція для парсингу розкладу:

```
def parse_schedule(message):
    events = []
    lines = message.message.split('\n')
    current_date = None

    for line in lines:
        line = line.strip()
        day_info = re.match(r'(\d{2}\.\d{2})', line)
        if day_info:
```

```

        current_date = day_info.group(2)

        current_date = datetime.strptime(current_date + '.' +
str(datetime.now().year), '%d.%m.%Y')
        elif '-' in line and current_date:
            time_info = re.match(r'(\d{2}:\d{2})\s*-\s*(.*)',
line)

            if time_info:
                start_time = time_info.group(1)
                name = time_info.group(2)
                start_datetime = datetime.combine(current_date,
datetime.strptime(start_time, '%H:%M').time())

                events.append({
                    'name': name,
                    'location': '',
                    'description': None,
                    'start_date': start_datetime
                })
    return events

```

Наданий код описує:

- Парсинг розкладу: Функція `parse_schedule` отримує повідомлення і розбиває його на рядки.
- Визначення дати: Визначає поточну дату на основі рядка з датою.
- Визначення подій: Парсить час та назву події, комбінуючи їх з датою для створення об'єктів подій, які потім додаються до списку `events`.

Збереження подій в базу даних:

```
def save_to_database(event_data):
```

```

event = session.query(Event).filter(Event.name ==
event_data['name'], Event.start_date ==
event_data['start_date']).first()

```

```

if not event:
    event = Event(
        name=event_data['name'],
        description=event_data['description'],
        location=event_data['location'],
        start_date=event_data['start_date']
    )
    session.add(event)
else:
    event.description = event_data['description']
    event.location = event_data['location']

session.commit()

```

Наданий код описує:

- Перевірка наявності події: Перед додаванням нової події до бази даних, перевіряється, чи вже існує подія з таким же ім'ям та датою.
- Додавання або оновлення події: Якщо подія не знайдена, створюється новий запис. Якщо подія вже існує, її опис та локація оновлюються.
- Збереження змін: Зміни зберігаються в базі даних за допомогою session.commit().

Використання Telethon у поєднанні з Python та асинхронними можливостями бібліотеки aiogram дозволяє створити ефективний та гнучкий бот, який може швидко обробляти велику кількість запитів, інтегруватися з різ-

ними сервісами, забезпечувати багатофункціональність і розширюваність додатка.

Висновки до розділу 2

У розділі "Проектування телеграм боту для молодіжного центру" було докладно розглянуто процес вибору технологій, архітектурні рішення та реалізаційні аспекти створення телеграм боту. Основним завданням було забезпечити зручний, швидкий та надійний інструмент для взаємодії молодіжного центру з його відвідувачами.

Вибір мови програмування та бібліотек:

Python був обраний завдяки його простоті, багатій екосистемі бібліотек та активній спільноті розробників. Він ідеально підходить для швидкої розробки та прототипування.

Aiogram та Telethon були обрані для реалізації бота через їхню асинхронну природу, що дозволяє ефективно обробляти велику кількість запитів одночасно. Aiogram забезпечує простий і зрозумілий інтерфейс для створення бота, тоді як Telethon надає додаткові можливості для роботи з Telegram API, включаючи обробку повідомлень від імені користувача.

Архітектурні рішення:

Було обрано підхід з використанням станів (FSMContext), що дозволяє легко керувати послідовністю дій користувача та забезпечує інтуїтивно зрозумілий інтерфейс.

Застосування бази даних для зберігання інформації про користувачів, події та підписки дозволяє ефективно управляти даними та забезпечує швидкий доступ до необхідної інформації.

Реалізаційні аспекти:

Реалізація основних команд, таких як /start, /help, /events, /subscriptions, /schedule, забезпечує користувачам швидкий доступ до необхідної інформації та функціоналу.

Використання асинхронних функцій для отримання та обробки даних, зокрема для парсингу розкладу подій з Telegram-каналу, дозволяє боту працювати швидко та без затримок.

Додавання інтерактивних клавіатур та кнопок полегшує навігацію та робить взаємодію з ботом більш зручною та інтуїтивно зрозумілою.

В цілому, реалізація телеграм боту для молодіжного центру демонструє, як сучасні технології можуть бути використані для створення корисних та ефективних інструментів для взаємодії з аудиторією. Використання Python, aiogram та Telethon дозволило створити гнучкий, масштабований та надійний бот, який відповідає всім вимогам молодіжного центру.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ТЕЛЕГРАМ БО- ТОМ ДЛЯ МОЛОДІЖНОГО ЦЕНТРУ

3.1 Порядок встановлення та налаштування системи

Даний розділ містить покрокові інструкції щодо встановлення та налаштування телеграм бота для молодіжного центру. Виконуючи ці кроки, ви зможете розгорнути бота на вашому сервері та налаштувати його для взаємодії з користувачами.

Вимоги до системи:

- Операційна система: Linux, Windows, або macOS.
- Python: Версія 3.7 або новіша.
- PostgreSQL: Для зберігання даних користувачів, подій та підписок.
- Telegram API Token: Отримайте токен для вашого бота від BotFather.

Кроки встановлення

1. Встановлення Python та необхідних бібліотек

Переконайтеся, що у вас встановлено Python. Якщо ні, завантажте та встановіть його з офіційного сайту Python [8].

Встановіть необхідні бібліотеки, виконавши наступні команди в терміналі:

```
pip install aiogram telethon sqlalchemy psycopg2-binary
```

2. Налаштування бази даних PostgreSQL

Встановіть PostgreSQL, якщо він ще не встановлений. Інструкції можна знайти на офіційному сайті PostgreSQL [7] (Рис 3.1).

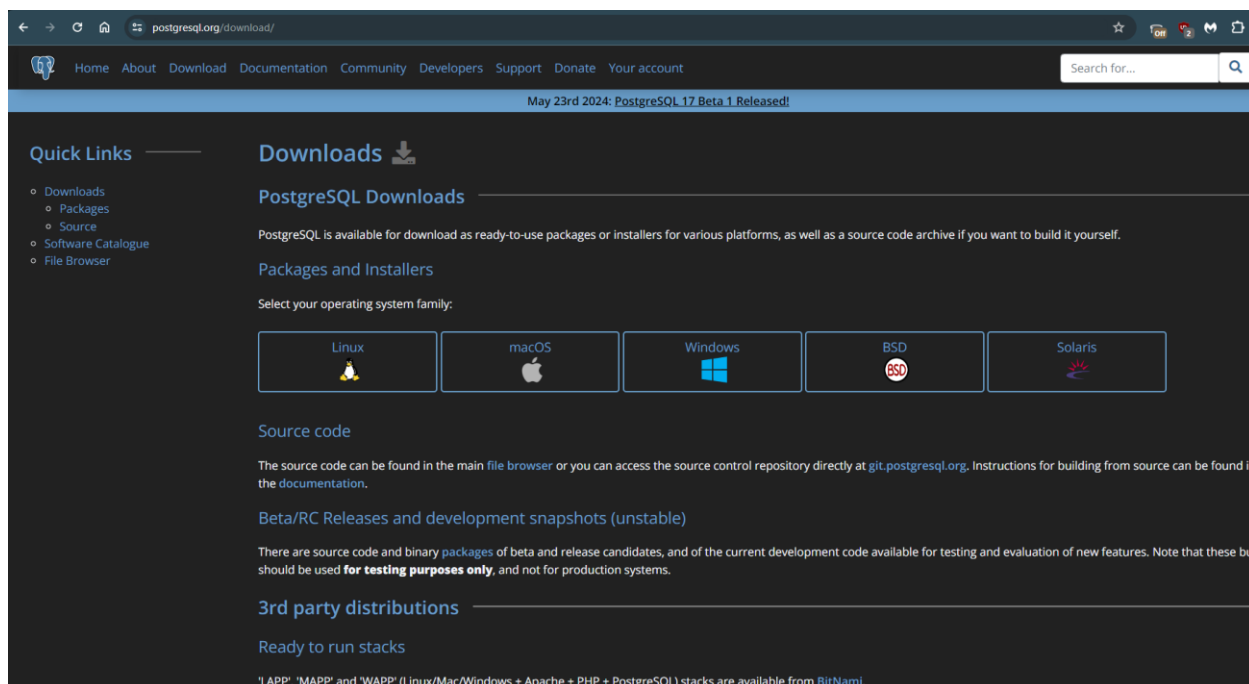


Рис. 3.1. Офіційний сайт PostgreSQL

Створіть нову базу даних для вашого бота:

```
CREATE DATABASE youth_center_bot;
```

Створіть таблиці у вашій базі даних, виконавши наступні SQL-запити:

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(255) UNIQUE NOT NULL,
    chat_id BIGINT UNIQUE NOT NULL
);

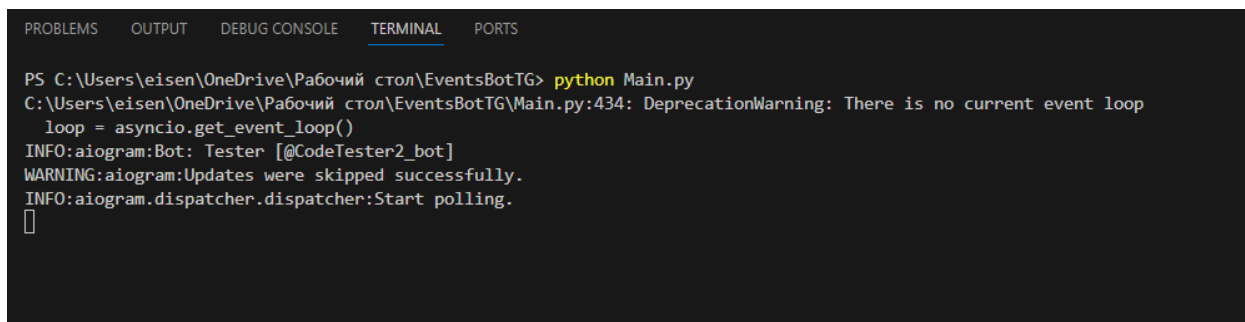
CREATE TABLE events (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    location VARCHAR(255),
    start_date TIMESTAMP NOT NULL
);

CREATE TABLE subscriptions (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(id),
    event_id INTEGER REFERENCES events(id) );
```

1. Запуск бота

Запустіть бота, виконавши наступну команду в терміналі (рис. 3.2):

`python Main.py`



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\eisen\OneDrive\Рабочий стол\EventsBotTG> python Main.py
C:\Users\eisen\OneDrive\Рабочий стол\EventsBotTG\Main.py:434: DeprecationWarning: There is no current event loop
  loop = asyncio.get_event_loop()
INFO:aiogram:Bot: Tester [@CodeTester2_bot]
WARNING:aiogram:Updates were skipped successfully.
INFO:aiogram.dispatcher.dispatcher:Start polling.
█
```

Рис. 3.2. Запуск бота через термінал

Дотримуючись цих інструкцій, ви зможете успішно встановити та налаштувати телеграм бота для молодіжного центру. Бот забезпечить зручний інтерфейс для взаємодії з користувачами, дозволяючи їм переглядати розклад подій, підписуватися на події та отримувати нагадування.

3.2 Інтерфейс та порядок роботи

Телеграм бот для молодіжного центру розроблений для надання зручного та інтуїтивного інтерфейсу для користувачів. Основні функції бота включають перегляд розкладу подій, підписку на події, отримання нагадувань про майбутні події, та доступ до часто запитуваної інформації

Після запуску бота користувачі можуть взаємодіяти з ним за допомогою команд та кнопок у головному меню. Головне меню містить наступні кнопки (рис 3.3):

- Розклад подій
- Мої підписки
- Оновити розклад
- Допомога
- FAQ

- Послуги
- Підписатися на нагадування
- Відписатися від нагадувань

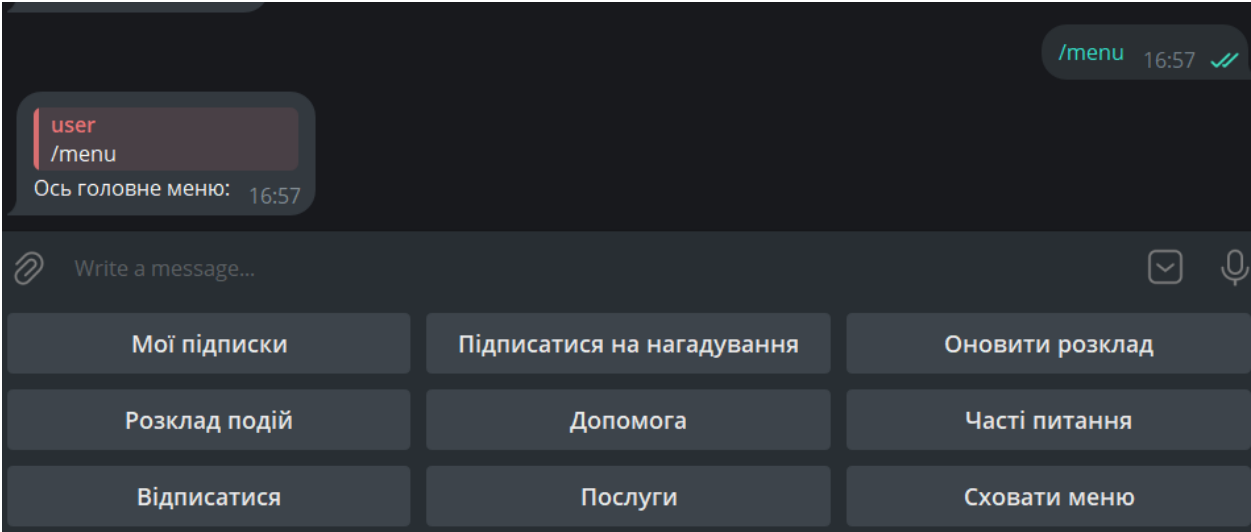


Рис. 3.3 Головне меню бота

Користувач може натиснути кнопку «Розклад подій» для перегляду списку всіх доступних подій. Бот відобразить список подій з датами та описами (рис. 3.4).

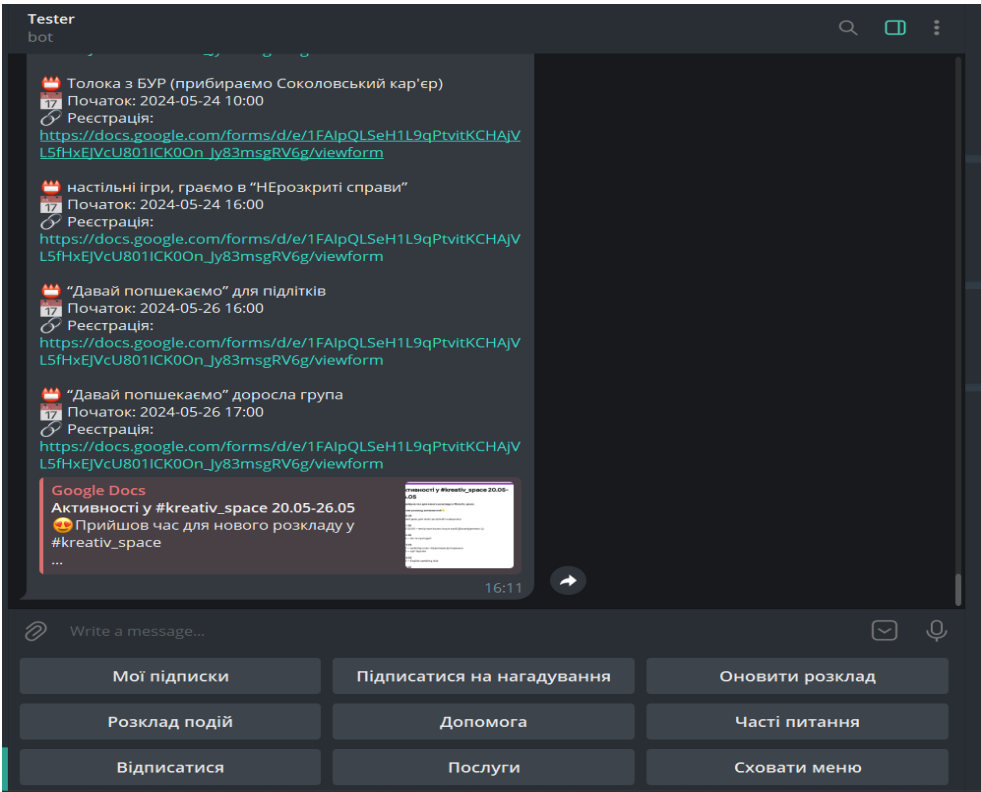


Рис. 3.4 Розклад подій молодіжного центру

«**Мої підписки**» дозволяє користувачам переглядати свої активні підписки та відписуватися від них. Після натискання кнопки «Мої підписки», бот відобразить список подій, на які користувач підписаний.(рис 3.5).

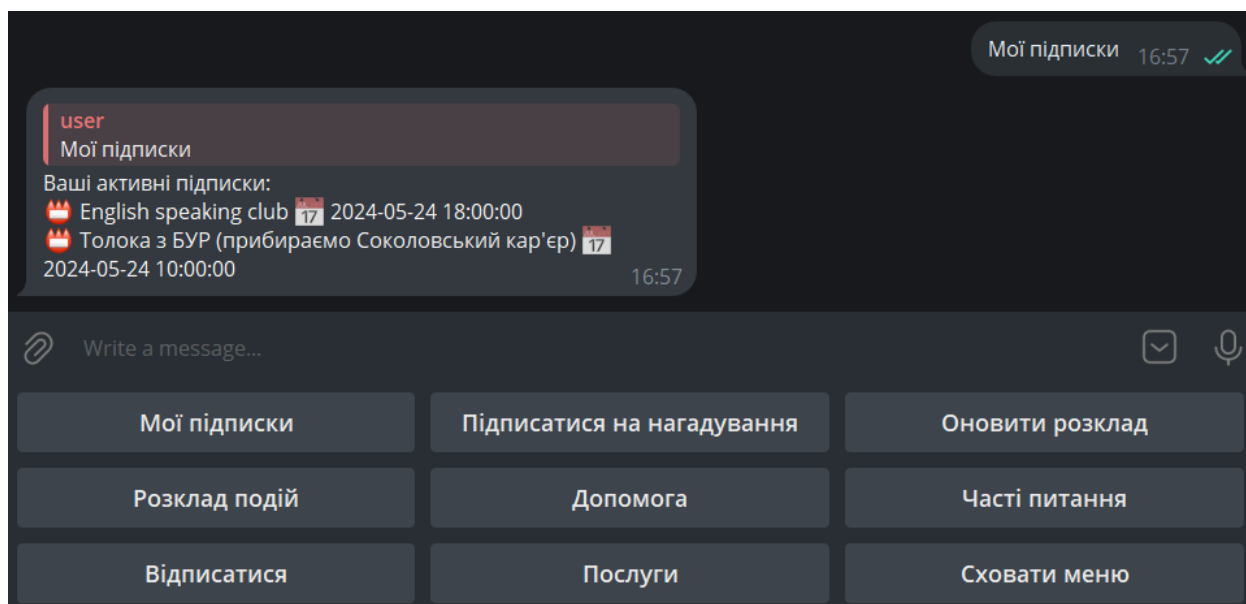


Рис. 3.5. Відображення підписок користувача.

Кнопка «Відписатися» дозволяє користувачам відмовитися від підписки на події, на які вони раніше зареєструвалися. Ця функція є важливою для користувачів, які бажають зменшити кількість отримуваних сповіщень або вже не зацікавлені в певних подіях.

Користувач натискає кнопку «Відписатися» в головному меню. Бот отримує та відображає список всіх подій, на які користувач підписаний. Для цього використовується SQL-запит, що отримує дані з таблиці subscriptions для поточного користувача (рис. 3.6).

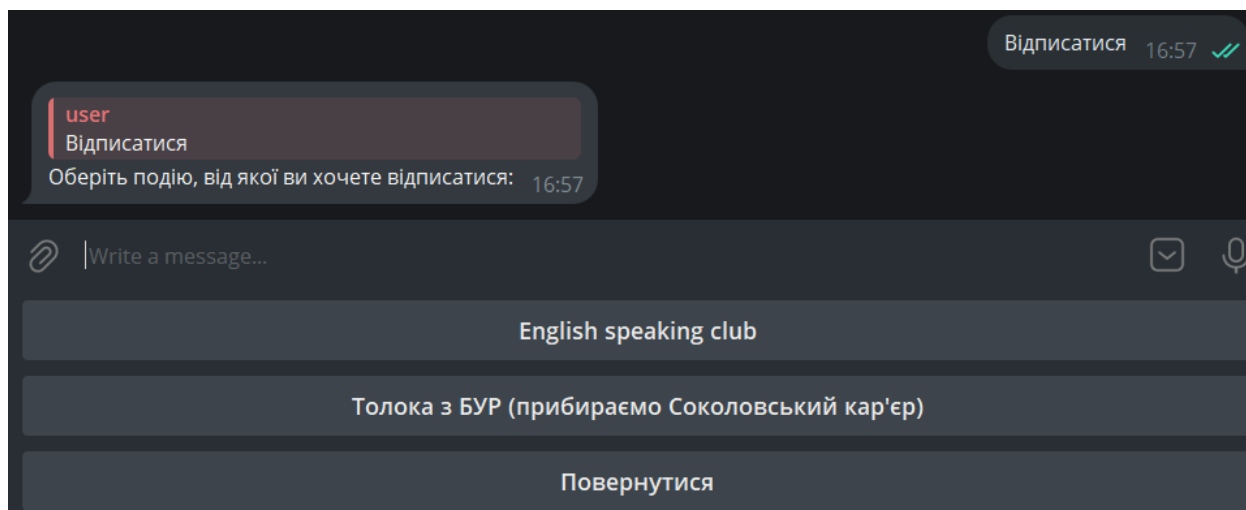


Рис. 3.6 Меню «Відписатися»

Користувач вибирає подію зі списку, яку бажає відписати. Бот отримує назву події та перевіряє наявність підписки в базі даних для поточного користувача. Якщо підписка існує, вона видаляється з таблиці subscriptions

Якщо користувач передумав, він може натиснути кнопку «Повернутися», щоб скасувати дію відписки. Бот видаляє клавіатуру та повертається до попереднього стану.

Кнопка "Оновити розклад" дозволяє користувачам оновити дані про події, щоб бути в курсі найсвіжіших змін та доповнень. Це особливо корисно в середовищі, де події можуть часто змінюватися або додаватися нові.

Коли користувач натискає кнопку "Оновити розклад", викликається обробник команди /refresh. Бот надсилає повідомлення про початок процесу оновлення та викликає функцію для оновлення даних про події (рис 3.7).

Після завершення процесу оновлення бот повідомляє користувача про успішне завершення або про помилку, якщо вона сталася.

Використання кнопки "Оновити розклад" дозволяє користувачам бути впевненими, що вони мають найактуальнішу інформацію про всі події молодіжного центру, що сприяє кращій організації часу та участі в подіях (рис. 3.7).

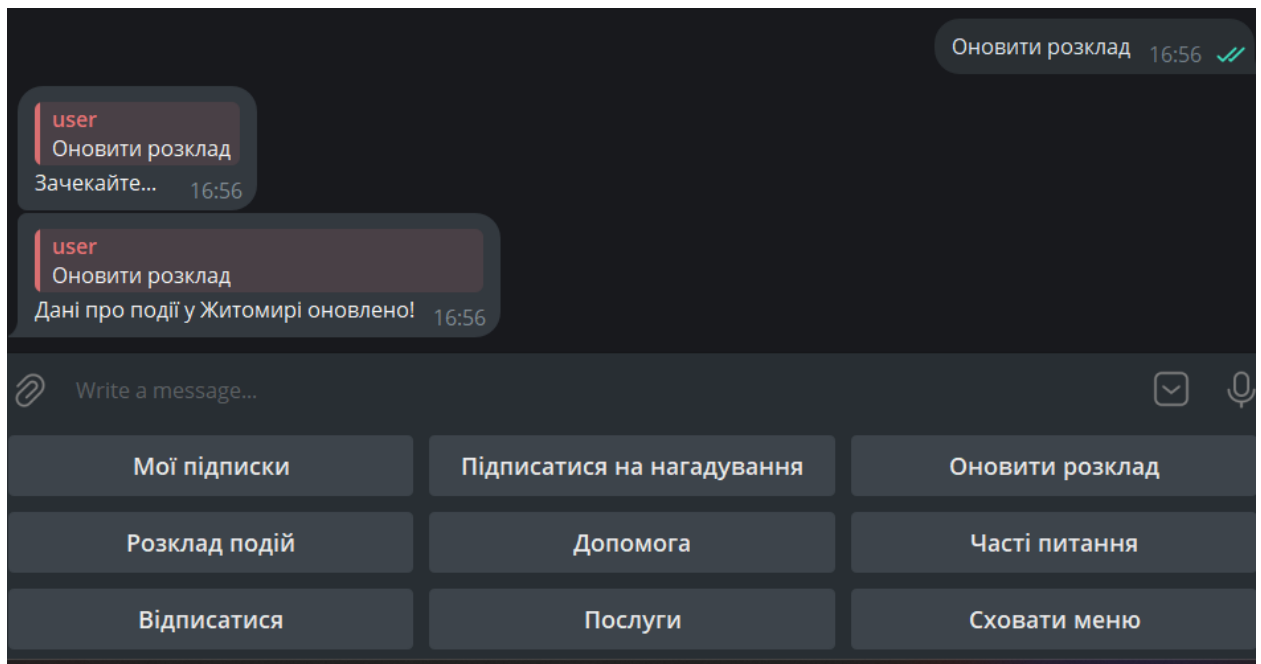


Рис. 3.7. Виконання функції оновлення розкладу

Функція "Часті питання" забезпечує користувачів швидким доступом до найпоширеніших запитань та відповідей на них. Це допомагає швидко знайти необхідну інформацію без необхідності звертатися до адміністрації молодіжного центру.

Коли користувач натискає кнопку "Часті питання", викликається обробник команди /faq. Бот надсилає повідомлення з кнопками для вибору конкретного питання (рис 3.8).

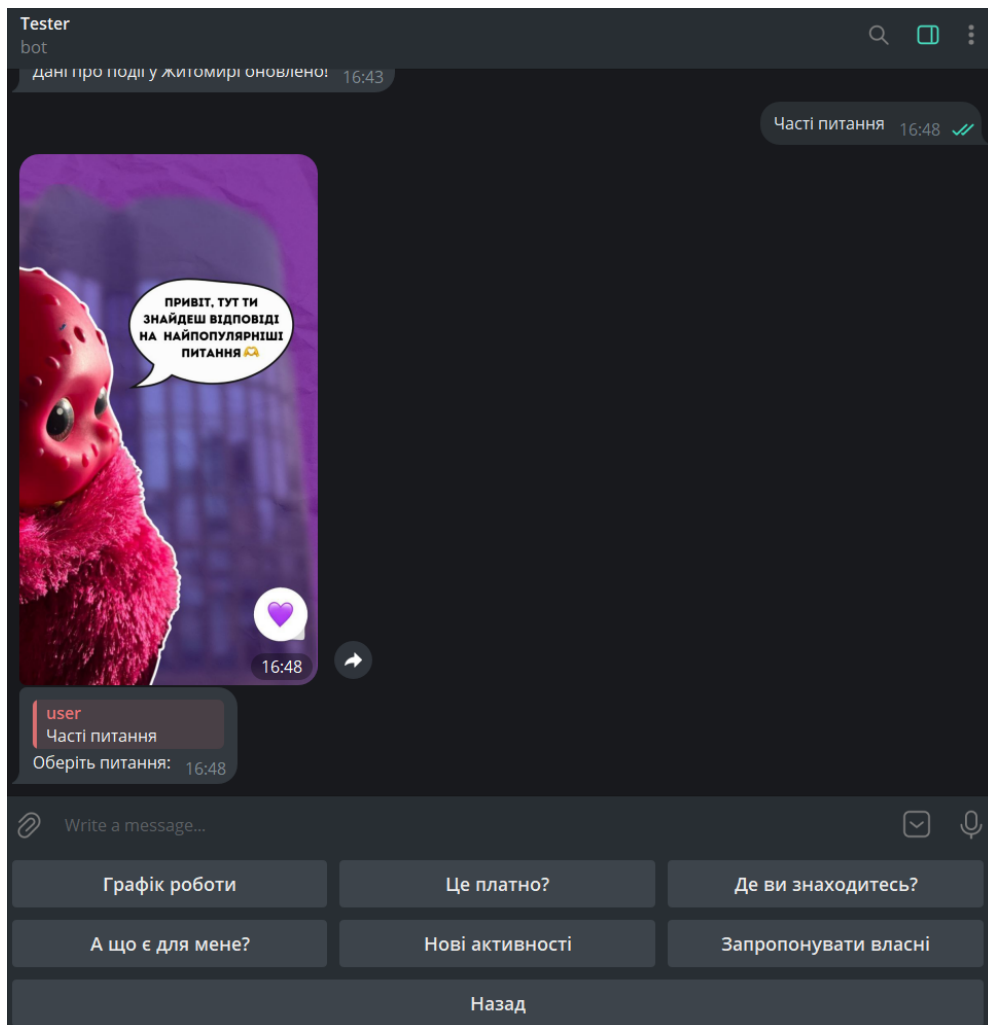


Рис. 3.8. Виконання функції «Часті питання»

Користувач бачить кнопки: «Графік роботи», «Це платно?», «Де ви знаходитесь?», «А що є для мене?», «Нові активності», «Запропонувати власні», «Назад».

Користувач обирає одне з питань, бот відправляє відповідне зображення з відповіддю (Рис 3.9).

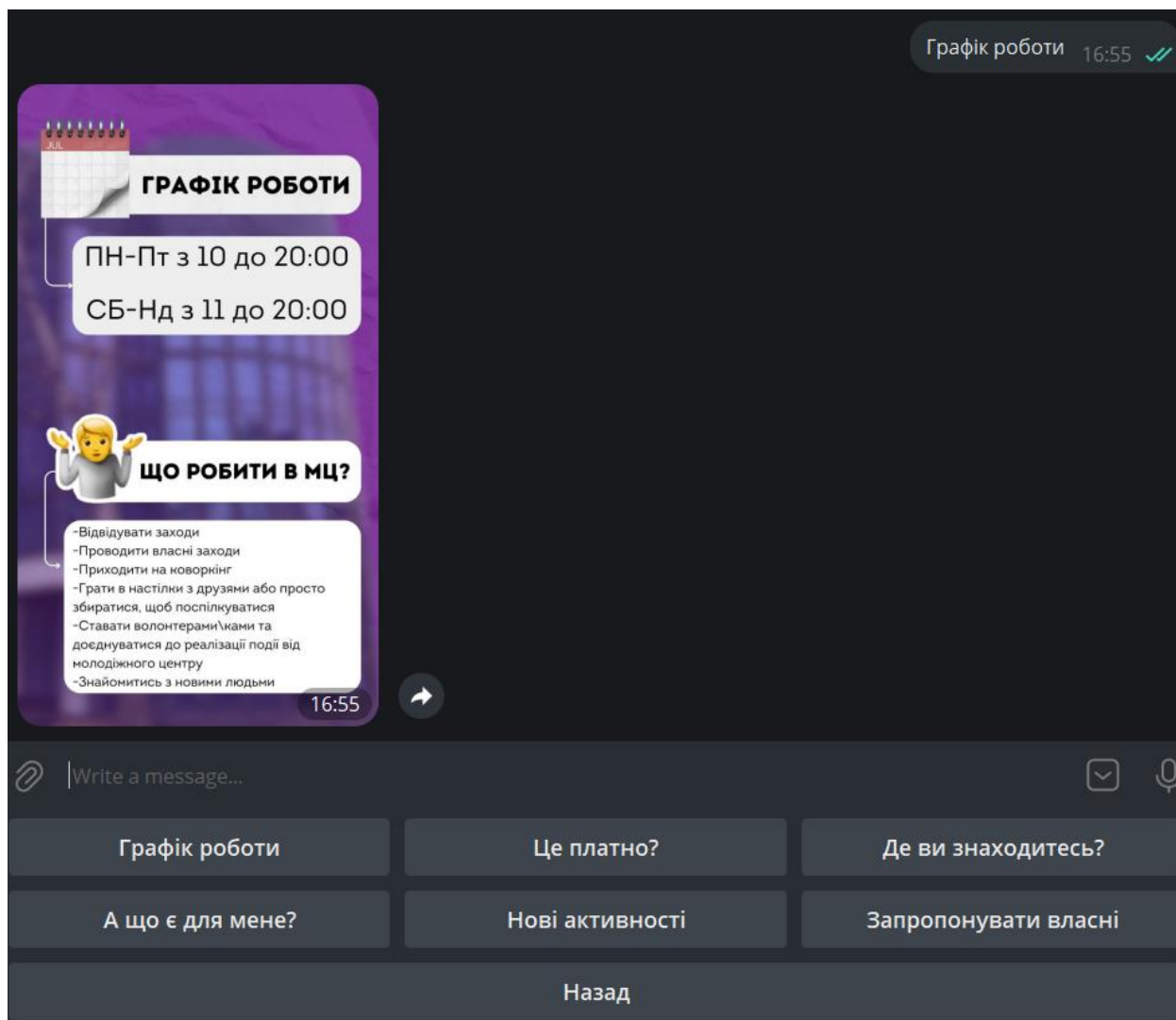


Рис. 3.9. Відповідь бота на одне з частих питань

Ця функція значно спрощує процес отримання важливої інформації для користувачів молодіжного центру. Завдяки інтуїтивно зрозумілому інтерфейсу та зручному доступу до часто запитуваної інформації, користувачі можуть швидко знайти відповіді на свої запитання без зайвих зусиль.

Функція "Підписатися на нагадування" дозволяє користувачам телеграм-бота молодіжного центру підписуватися на події, щоб отримувати сповіщення про них. Це забезпечує можливість бути в курсі майбутніх заходів і не пропустити важливі події.

Коли користувач надсилає команду /events, бот показує список доступних подій, на які можна підписатися (рис 3.10).

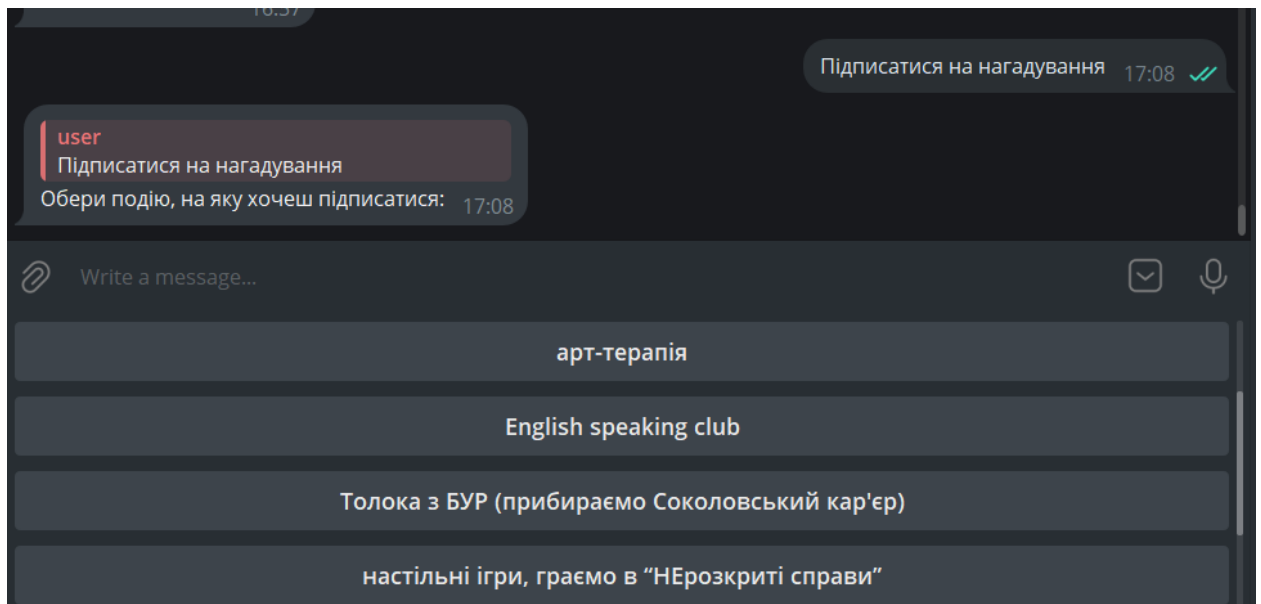


Рис. 3.10. Обробник підписки на нагадування

Користувач обирає подію зі списку, і бот надсилає запит на підтвердження участі в події (рис. 3.11)

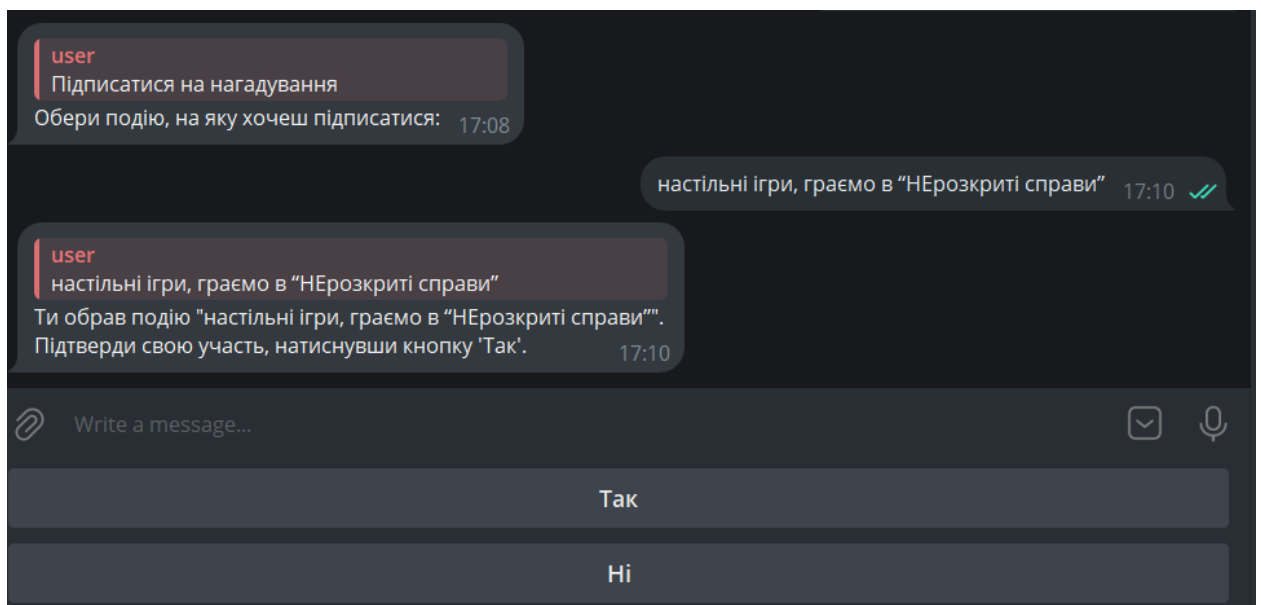


Рис. 3.11. Підтвердження підписки на подію

Якщо користувач підтверджує свою участь, створюється запис підписки в базі даних. Якщо користувач вже підписаний на цю подію, бот інформує його про це.

Функція "Послуги" дозволяє користувачам телеграм-бота молодіжного центру ознайомитися з наданими послугами, такими як індивідуальні психологічні консультації та індивідуальні кар'єрні консультації. Це забезпечує зручний доступ до інформації про послуги, що надаються центром, та допомагає користувачам приймати обґрунтовані рішення про те, чи скористатися цими послугами.

Кнопка "Послуги" додається до головного меню, що дозволяє користувачам легко отримати доступ до інформації про доступні послуги.

Коли користувач натискає кнопку "Послуги", бот показує список доступних послуг (рис 3.12).

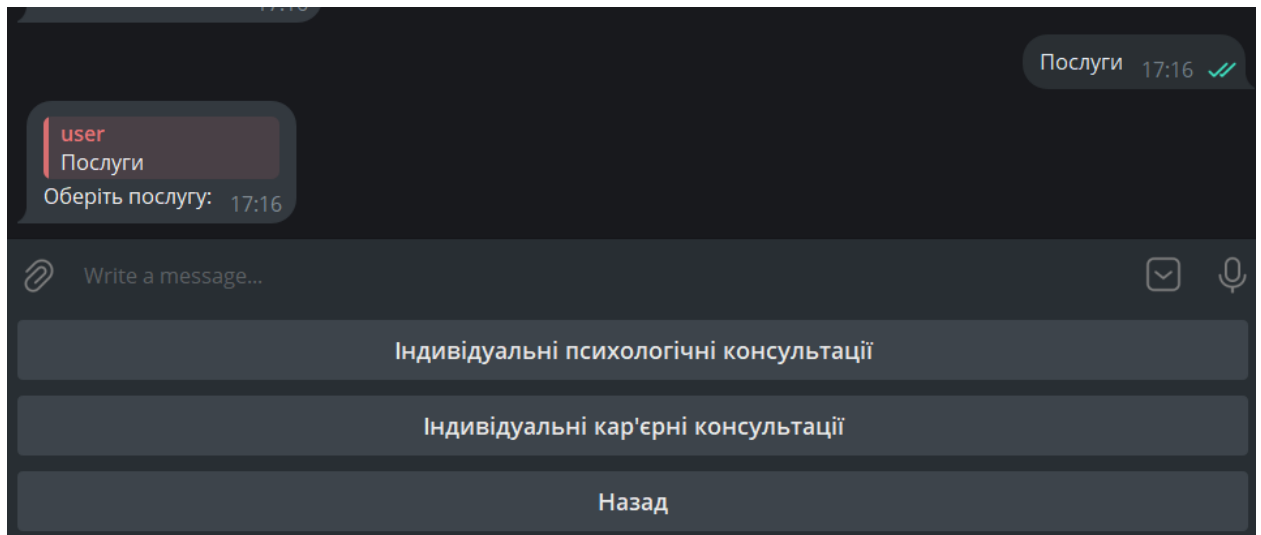


Рис. 3.12. Список доступних послуг молодіжного центру

Коли користувач обирає одну з послуг, бот надсилає інформацію про цю послугу (рис 3.13).

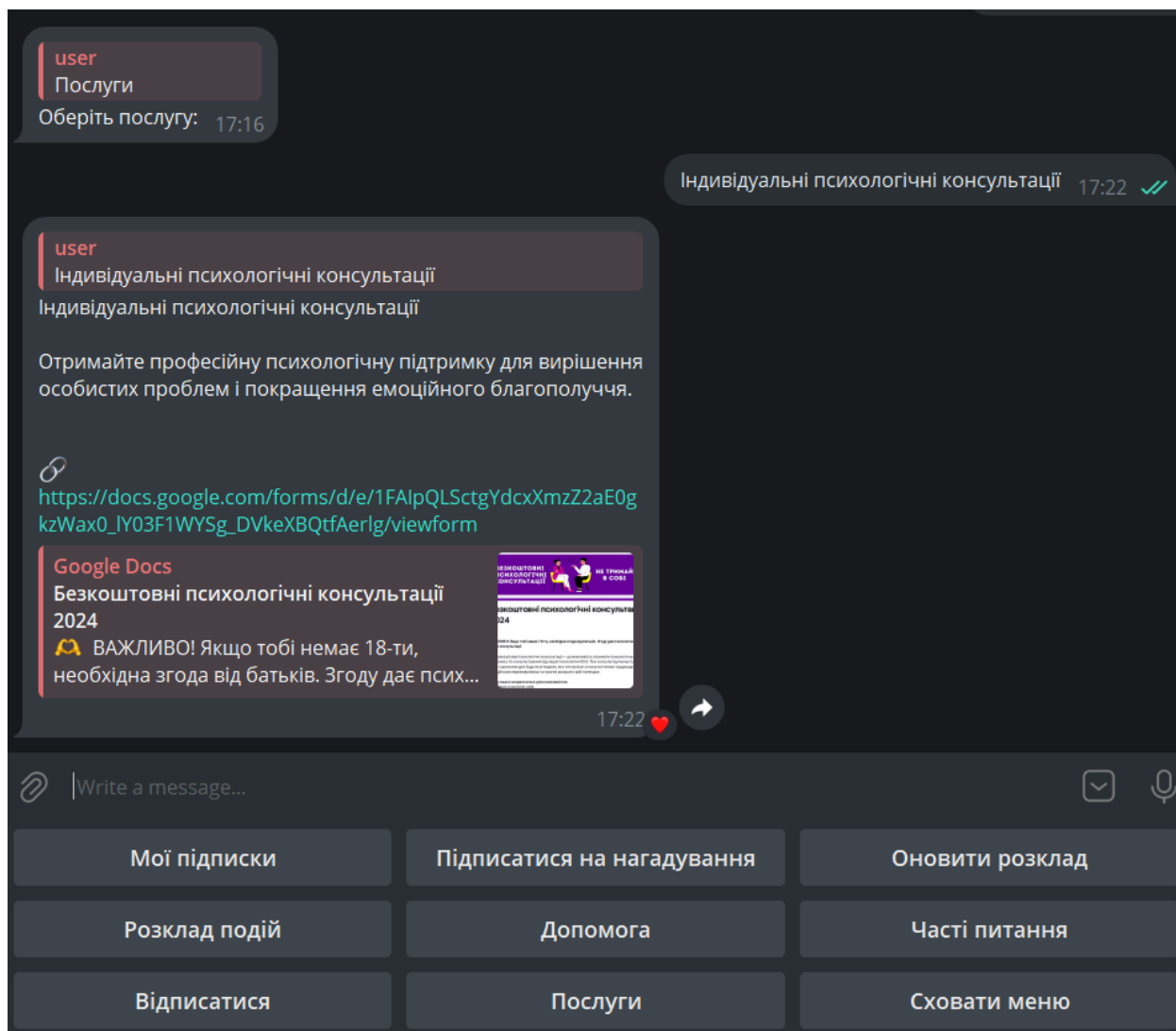


Рис. 3.13. Надання інформації про послугу

Функція "Послуги" дозволяє користувачам легко ознайомитися з доступними послугами молодіжного центру, такими як індивідуальні психологічні та кар'єрні консультації. Це сприяє підвищенню обізнаності користувачів про послуги та допомагає їм приймати обґрунтовані рішення щодо використання цих послуг.

3.3 Тестування роботи телеграм-бота для молодіжного центру

Тестування роботи телеграм-бота для молодіжного центру включає кілька етапів, щоб перевірити різні аспекти його функціональності та продуктивності. Важливі аспекти, які потрібно врахувати під час тестування:

1. Функціональність основних команд

Перевірка коректності роботи основних команд бота:

- Команда реєстрації: перевірка, чи можуть користувачі успішно зареєструватися за допомогою команди /register.
- Команда перегляду подій: перевірка, чи можуть користувачі переглядати доступні події за допомогою команди /events та підписуватися на них.

2. Перевірка підписки та відписки

Перевірка функцій підписки на події та відписки від них:

- Підписка на подію: тестування процесу підписки на подію, включаючи перевірку, чи не є користувач вже підписаним на ту саму подію.
- Відписка від події: тестування можливості користувачів відписатися від подій.

3. Перевірка оновлення та відображення розкладу подій

Тестування команд для оновлення та перегляду розкладу подій:

- Оновлення розкладу: перевірка, чи можуть користувачі оновлювати розклад подій за допомогою команди /refresh.
- Перегляд розкладу: тестування команди /schedule для відображення актуального розкладу подій.

4. Перевірка функціональності FAQ

Тестування функціоналу частих питань:

- Команда частих питань: перевірка, чи можуть користувачі отримати відповіді на поширені питання за допомогою спеціальної команди.

5. Тестування користувацького інтерфейсу

Перевірка зручності користування ботом, зокрема, тестування інтуїтивності інтерфейсу та простоти виконання команд:

- Реакція на команди: перевірка швидкості реакції бота на команди користувачів.
- Відображення повідомлень та меню: перевірка коректного відображення повідомлень та клавіатур з кнопками.

Проведене тестування показало, що телеграм-бот для молодіжного центру функціонує коректно та задовольняє основні вимоги користувачів. Були виявлені та виправлені деякі помилки, що дозволило підвищити стабільність роботи бота. Завдяки тестуванню було забезпечено високу якість роботи бота та його готовність до використання в реальних умовах.

Висновки до розділу 3

В процесі встановлення та налаштування системи було визначено ключові етапи, необхідні для успішної інтеграції телеграм-бота для молодіжного центру. Встановлення програмного забезпечення та налаштування середовища виконання забезпечили основу для стабільної роботи бота. Крім того, детальні інструкції з налаштування допомогли спростити процес впровадження для користувачів та адміністраторів. Успішне налаштування підтвердило відповідність системи вимогам та забезпечило готовність до подальшого тестування та експлуатації.

В ході розробки інтерфейсу та порядку роботи телеграм-бота для молодіжного центру було створено інтуїтивно зрозумілий користувацький інтерфейс. Чітка структура меню та логічна послідовність команд дозволили забезпечити зручність використання бота. Користувачі можуть легко отримувати інформацію про події, підписуватися на них та відписуватися, переглядати розклад та отримувати відповіді на часті питання. Загалом, розроблений інтерфейс сприяє підвищенню залученості користувачів та ефективному використанню бота.

Проведене тестування телеграм-бота для молодіжного центру дозволило оцінити його функціональність, продуктивність та стабільність

роботи. Тестування основних команд та функціоналу, таких як реєстрація, підписка на події, перегляд розкладу та функція FAQ, показало, що бот відповідає заявленим вимогам та забезпечує необхідний рівень користувацького досвіду. Виявлені під час тестування проблеми були оперативно усунені, що дозволило покращити якість роботи бота. Загалом, результати тестування підтвердили готовність системи до впровадження та експлуатації.

ВИСНОВКИ

Розробка та впровадження телеграм-боту для молодіжного центру виявилася складним, але цікавим завданням, що вимагало комплексного підходу та уважного аналізу потреб цільової аудиторії. Під час виконання дипломної роботи було здійснено значний обсяг роботи з розробки програмного забезпечення, тестування та оптимізації його функціональності та інтерфейсу користувача.

Результатом дипломної роботи є створення ефективного та зручного інструменту для комунікації з молоддю. Телеграм-бот молодіжного центру дозволяє швидко та зручно отримувати інформацію про події, послуги та інші корисні ресурси, сприяючи поліпшенню взаємодії між молоддю та центром.

У процесі виконання дипломної роботи було проведено дослідження потреб та вимог цільової аудиторії. Виявлено, що молодь виявляє значний інтерес до використання месенджерів для отримання інформації та комунікації. Це стало основою для розробки та впровадження телеграм-боту як ефективного інструменту для взаємодії з молоддю.

Розробка програмного забезпечення включала в себе розробку архітектури системи, реалізацію функціональності та інтерфейсу користувача, а також тестування та вдосконалення продукту. Завдяки комплексному підходу та використанню сучасних технологій, було досягнуто високої якості та ефективності програмного забезпечення.

Створений телеграм-бот успішно впроваджено в роботу молодіжного центру і отримав позитивний відгук від користувачів. Його функціональність, зручний інтерфейс та висока продуктивність сприяють покращенню комунікації з молоддю та забезпеченню доступу до необхідної інформації.

У майбутньому планується подальше розвиток та вдосконалення телеграм-боту. Планується розширення функціональності, вдосконалення

інтерфейсу користувача, а також впровадження нових технологій та сервісів для поліпшення якості та ефективності взаємодії з молоддю.

Дипломна робота є важливим кроком у розвитку і впровадженні програмного забезпечення для молодіжного центру. Створення телеграм-боту відкриває нові можливості для покращення комунікації та взаємодії з молоддю, сприяючи подальшому розвитку та популяризації центру серед молоді.

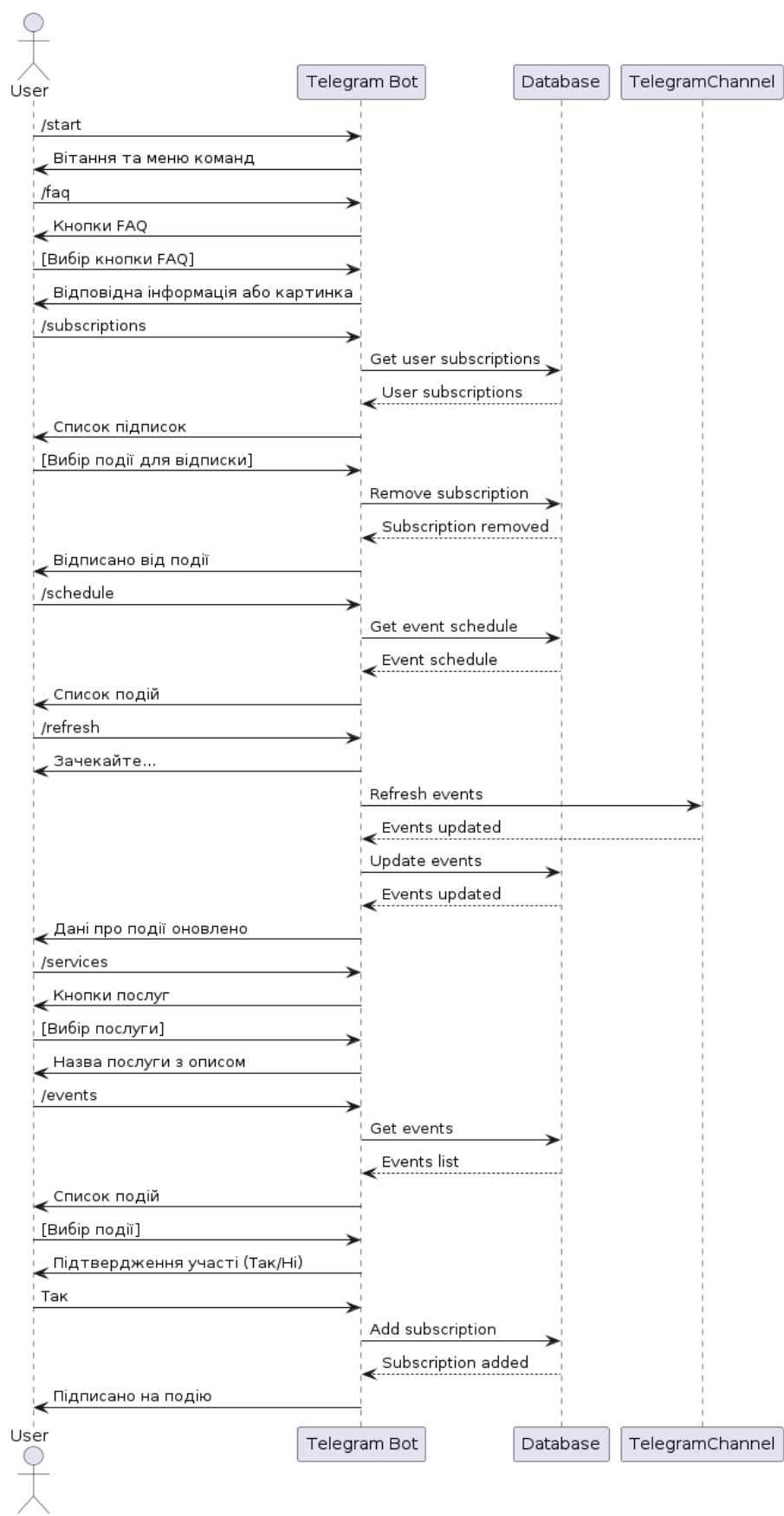
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Aiogram Documentation. [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.aiogram.dev/en/latest/index.html#>
2. Build Your Own Python Chatbot with the Telegram API. [Електронний ресурс] – Режим доступу до ресурсу: <https://jokiruiz.com/python/python-telegram-bot-tutorial-create-powerful-bots/>
3. How to Make a Telegram Bot in Python. [Електронний ресурс] – Режим доступу до ресурсу: <https://thepythoncode.com/article/make-a-telegram-bot-in-python>
4. PostgreSQL 16.3 Documentation. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/index.html>
5. SQLAlchemy Documentation. [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.sqlalchemy.org/en/20/dialects/postgresql.html>
6. Telegram Bot API. [Електронний ресурс] – Режим доступу до ресурсу: <https://core.telegram.org/bots/api>
7. Telethon's Documentation. [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.telethon.dev/en/stable/index.html>
8. The Python Tutorial. [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.python.org/3/tutorial/index.html>
9. Онлайн редактор JSON-коду. [Електронний ресурс]. – Режим доступу до ресурсу: <https://jsoneditoronline.org/>
10. Requestsв Python – приклади виконання HTTP запитів. [Електронний ресурс]. – Режим доступу до ресурсу: <https://python-scripts.com/>
11. Use-case diagrams [Електронний ресурс]. – Режим доступу до ресурсу: <https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams>
use-case

12. Добірка: 30 корисних ботів [Електронний ресурс]. – Режим доступу до ресурсу: <https://ain.ua/2017/08/14/30-telegram-botov/>
13. Використання мови Python для розробки науково-технічного програмного забезпечення [Електронний ресурс]. – Режим доступу до ресурсу: <https://dou.ua/lenta/articles/python-for-science>
14. Асинхронний Telegram бот мовою Python 3 з використанням бібліотеки aiogram [Електронний ресурс]. – Режим доступу до ресурсу: <https://opensource4u.com/lib/aiogram-lessons>
15. Welcome to Python Telegram Bot's documentation [Електронний ресурс] - Режим доступу: <https://python-telegram-bot.readthedocs.io/en/stable>
16. Learn to build your first bot in Telegram with Python [Електронний ресурс] – Режим доступу до ресурсу: <https://medium.freecodecamp.org/learn-to-buildyourfirst-bot-in-telegram-with-python-4c99526765e4>
17. PyCharm: the Python IDE for Professional Developers by JetBrains [Електронний ресурс] – Режим доступу до ресурсу: <https://www.jetbrains.com/pycharm>
18. python-telegram-bot [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/python-telegram-bot/python-telegram-bot>

ДОДАТКИ

					ПЗ.КР.Б – 121 – 23 – ПЗ	
						57



Додаток А. Діаграма послідовності для взаємодії з ботом

Реалізація парсингу розкладу з телеграм каналу.

Лістинг fbparse.py

```
import re
import asyncio
from telethon import TelegramClient
from datetime import datetime
from CDatabase import Event, Subscription, session

# Вкажіть свої параметри Telegram API
api_id = 'your_api_id'
api_hash = 'your_api_hash'
channel_username = 'mol_centra_zt'

async def parse_tg():
    client = TelegramClient('anon', api_id, api_hash)

    await client.start()
    channel = await client.get_entity(channel_username)

    found_message = None

    async for message in client.iter_messages(channel):
        if message.message and "Лови розклад активностей" in
message.message:
            found_message = message
            break

    if found_message:
        print("Знайдене повідомлення:")
        print(found_message.text)

        events = parse_schedule(found_message)
```

```

        existing_events = {event.name: event for event in
session.query(Event).all()}
        new_event_names = [event['name'] for event in events]

        for event_name, event in existing_events.items():
            if event_name not in new_event_names:
                delete_event_and_unsubscribe(event)

        for event in events:
            save_to_database(event, existing_events)

    else:
        print("Повідомлення з текстом 'Лови розклад активностей'
не знайдено.")

    await client.disconnect()

def parse_schedule(message):
    print("Знайдена ссылка для реєстрації")
    reg_link = find_first_link(message.text)
    print(reg_link)
    events = []
    lines = message.message.split('\n')
    current_date = None

    for line in lines:
        line = line.strip()
        day_info = re.match(r'(ПН|ВТ|СР|ЧТ|ПТ|СБ|НД)
(\d{2}\.\d{2})', line)
        if day_info:
            current_date = day_info.group(2)
            current_date = datetime.strptime(current_date + '.' +
str(datetime.now().year), '%d.%m.%Y')
            elif '-' in line and current_date:

```

```

        time_info = re.match(r'(\d{2}:\d{2})\s*-\s*(.*)',
line)

        if time_info:
            start_time = time_info.group(1)
            name = time_info.group(2)
            start_datetime = datetime.combine(current_date,
datetime.strptime(start_time, '%H:%M').time())

            events.append({
                'name': name,
                'location': '',
                'description': reg_link,
                'start_date': start_datetime
            })

        return events

def find_first_link(text):
    link_pattern = r'\[.*?\]\((.*?)\)' # Знаходимо текст у фор-
маті [текст](посилання)
    match = re.search(link_pattern, text)

    if match:
        print(match.group(1))
        return match.group(1) # Повертаємо перше знайдене поси-
лання
    else:
        return None # Якщо посилання не знайдено, повертаємо
None

def save_to_database(event_data, existing_events):
    event = existing_events.get(event_data['name'])
    if event:
        event.description = event_data['description']
        event.location = event_data['location']

```

```

        event.start_date = event_data['start_date']
    else:
        event = Event(
            name=event_data['name'],
            description=event_data['description'],
            location=event_data['location'],
            start_date=event_data['start_date']
        )
        session.add(event)

    session.commit()

def delete_event_and_unsubscribe(event):
    subscriptions =
session.query(Subscription).filter_by(event_id=event.id).all()
    for subscription in subscriptions:
        session.delete(subscription)
    session.delete(event)
    session.commit()

```

ДОДАТОК В

Реалізація сутностей бази даних, підключення до бази даних

Лістинг CDataBase.py

```

from sqlalchemy import create_engine, Column, Integer, String,
DateTime, ForeignKey
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import relationship, sessionmaker
from datetime import datetime, timedelta

# Підключення до бази даних
engine = create_engine('postgresql://postgres:adminpass@localhost:5432/EventsDatabase')

Session = sessionmaker(bind=engine)

```

```

session = Session()

Base = declarative_base()
# Визначення моделі користувача
class User(Base):
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True)
    username = Column(String)
    chat_id = Column(Integer)

# Визначення моделі події
class Event(Base):
    __tablename__ = "events"

    id = Column(Integer, primary_key=True, index=True)
    name = Column(String)
    description = Column(String)
    location = Column(String)
    start_date = Column(DateTime)

# Визначення моделі підписки
class Subscription(Base):
    __tablename__ = 'subscriptions'
    id = Column(Integer, primary_key=True)
    user_id = Column(Integer, ForeignKey('users.id'))
    event_id = Column(Integer, ForeignKey('events.id'))
    user = relationship("User", backref="subscriptions")
    event = relationship("Event", backref="subscriptions")

```

ДОДАТОК Г

Реалізація інтерфесу, основного функціоналу програми

Лістинг Main.py:

```

import asyncio
import aiogram

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						63

```

import logging
from aiogram import Bot, Dispatcher, types
from aiogram.contrib.middlewares.logging import LoggingMiddleware
from aiogram.contrib.fsm_storage.memory import MemoryStorage
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters.state import State, StatesGroup
from aiogram.types import ParseMode, ReplyKeyboardMarkup,
KeyboardButton

from aiogram.utils import executor
from datetime import datetime, timedelta

from CDatabase import User, Subscription, Event, Base, session
from fbparse import parse_tg

# Ініціалізація бота та диспетчера
API_TOKEN = '6457359930:AAE3_Pp306BqCfUAAXhoMhP8SL9uaeqRM-U' #
Замініть це на свій токен
bot = Bot(token=API_TOKEN)
storage = MemoryStorage()
dp = Dispatcher(bot, storage=storage)
logging.basicConfig(level=logging.INFO)

# Функція для створення клавіатури з однією кнопкою
def create_keyboard(button_text):
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add(KeyboardButton(button_text))
    return keyboard

# Функція для створення клавіатури з двома кнопками
def create_keyboard(button_text, button_text2):
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add(KeyboardButton(button_text))
    keyboard.add(KeyboardButton(button_text2))
    return keyboard

```

```

# Функція для створення головного меню
def create_main_menu():
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
    buttons = ["Мої підписки", "Підписатися на нагадування",
"Оновити розклад", "Розклад подій", "Допомога", "Часті питання",
"Відписатися", "Послуги", "Сховати меню"]
    keyboard.add(*buttons)
    return keyboard

def create_faq_keyboard():
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
    buttons = ["Графік роботи", "Це платно?", "Де ви знаходи-
тесь?", "А що є для мене?", "Нові активності", "Запропонувати власні",
"Назад"]
    keyboard.add(*buttons)
    return keyboard

# Обробник команди /menu
@dp.message_handler(commands=['menu'])
async def show_menu(message: types.Message):
    await message.reply("Ось головне меню:",
reply_markup=create_main_menu())

# Функція для показу FAQ
@dp.message_handler(lambda message: message.text == "Часті питан-
ня")
async def show_faq(message: types.Message):
    await bot.send_photo(message.chat.id,
photo=open('faq_main.jpg', 'rb'))
    await message.reply("Оберіть питання:",
reply_markup=create_faq_keyboard())

# Функції-обробники для кожного питання

```

```
@dp.message_handler(lambda message: message.text == "Графік робо-  
ти")
```

```
    async def show_schedule_info(message: types.Message):  
        await bot.send_photo(message.chat.id,  
photo=open('work_time.jpg', 'rb'))
```

```
@dp.message_handler(lambda message: message.text == "Це платно?")  
    async def show_pricing_info(message: types.Message):  
        await bot.send_photo(message.chat.id,  
photo=open('how2find_hm.jpg', 'rb'))
```

```
@dp.message_handler(lambda message: message.text == "Де ви знахо-  
дитесь?")  
    async def show_location_info(message: types.Message):  
        await bot.send_photo(message.chat.id,  
photo=open('how2find_hm.jpg', 'rb'))
```

```
@dp.message_handler(lambda message: message.text == "А що є для  
мене?")  
    async def show_for_me_info(message: types.Message):  
        await bot.send_photo(message.chat.id,  
photo=open('for_me.jpg', 'rb'))
```

```
@dp.message_handler(lambda message: message.text == "Нові актив-  
ності")  
    async def show_new_activities_info(message: types.Message):  
        await bot.send_photo(message.chat.id,  
photo=open('new_or_own.jpg', 'rb'))
```

```
@dp.message_handler(lambda message: message.text == "Запропонува-  
ти власні")  
    async def show_suggest_own_info(message: types.Message):  
        await bot.send_photo(message.chat.id,  
photo=open('new_or_own.jpg', 'rb'))
```