МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Telegram бот для пошуку найближчих заходів у містах України»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студентка групи ІТ.м-01 Бирченко А. В.

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «___» грудня 2021 р.

Науковий керівник _____ к.т.н., доц., Шендрик В. В.
(підпис)

Голова комісії _____ Шифрін Д.М.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2021

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. кафедри ІТ

_____ В. В. Шендрик
«___» _____ 2021 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Бирченко Альона Володимирівна
(прізвище, ім'я, по батькові)

1 Тема проекту *Telegram бот для пошуку найближчих заходів у містах України*

затверджена наказом по університету від «29» жовтня 2021 р. № 0787-IV

2 Термін здачі студентом закінченого проекту «10» грудня 2021 р.

3 Вхідні дані до проекту методичні вказівки і вимоги до оформлення кваліфікаційної роботи магістра

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз предметної області, постановка задачі та методи дослідження, проектування чат-бота, розробка чат-бота

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) постановка задачі, порівняння програмних продуктів-аналогів, функціональні вимоги до бота, контекстна діаграма процесу пошуку найближчих заходів у містах України, діаграма декомпозиції процесу пошуку найближчих заходів у містах України, діаграма варіантів використання чат-бота, схема бази даних, засоби реалізації, архітектура чат-бота, демонстрація роботи, висновки.

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускного проекту	Термін виконання етапів проекту	Примітка
1	Аналіз предметної області	15.09.21 – 20.09.21	
2	Аналіз продуктів-аналогів	21.09.21 – 24.09.21	
3	Виявлення проблем предметної області	25.09.21 – 28.09.21	
4	Визначення переліку вимог	29.09.21 – 06.12.21	
5	Визначення технологій розробки	06.10.21 – 08.10.21	
6	Вибір засобів та інструментів розробки	10.10.21 – 14.10.21	
7	Планування змісту структури робіт проекту	15.10.21 – 19.10.21	
	Визначення ризиків при розробці проекту	20.10.21 – 26.10.21	
8	Розробка дизайну бота	27.10.21 – 01.11.21	
9	Розробка БД	04.11.21 – 11.11.21	
10	Розробка бота	11.11.21 – 26.11.21	
11	Тестування бота	26.11.21 – 28.11.21	
12	Оформлення пояснювальної записки	01.12.21 – 11.12.21	
13	Захист роботи	21.12.21	

Магістрант _____ Бирченко А.В.

Керівник роботи _____ к.т.н., доц. Шендрик В.В.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Telegram бот для пошуку найближчих заходів у містах України».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 32 найменувань, додатків. Загальний обсяг роботи − 66 сторінок, у тому числі 44 сторінок основного тексту, 3 сторінки списку використаних джерел, 19 сторінок додатків.

Кваліфікаційну роботу магістра присвячено розробці чат-бота для пошуку найближчих заходів у містах України.

У першому розділі було проведено аналіз продуктів-аналогів, встановлено мету та актуальність проекту.

У другому розділі було описано мету, задачі та методи дослідження,

У третьому розділі було здійснено моделювання чат-бота, а саме було побудовано діаграму варіантів використання, контекстну діаграму та діаграму декомпозиції процесу забезпечення пошуку заходів по містах України у нотації IDEF0, модель бази даних.

У четвертому розділі було описано реалізацію чат-бота, а також здійснено демонстрацію функціоналу бота.

Результатом кваліфікаційної роботи магістра є розроблений бот для пошуку найближчих заходів у містах України

Ключові слова: чат-бот, Telegram, Python, SQLite, пошук заходів, Sublime Text, месенджер.

ЗМІСТ

ВСТУП

Сучасний світ постійно розвивається у різних сферах. У тому числі зміни відбуваються і у сфері інформаційних технологій, де постійно з’являються нові ідеї, які впливають на повсякденне життя людей.

Обмін повідомленнями став одним із найпопулярніших засобів масової інформації в останні роки. На сьогоднішній день окрім додатків та комп’ютерних програм все більшої популярності набуває використання різноманітних чат-ботів.

Чат-бот – це спеціалізований додаток, що дозволяє користувачам взаємодіяти зі сторонніми сервісами через інтерфейс чату. Він є своєрідним помічником, який спілкується з користувачами через повідомлення і має безліч функцій. Так, за допомогою ботів можна шукати необхідну інформацію, бронювати житло, отримати консультацію, здобувати знання та багато іншого.

Кожного дня люди відвідують різноманітні заходи такі як концерти, вистави, виставки, шоу та інші. Часто у людей виникає потреба знайти найближчі заходи у рідному місті, спираючись на конкретні вподобання, місце та час проведення . Гарним рішенням для задоволення даної потреби є розробка чат-бота для пошуку заходів у містах України.

Предметом дослідження є процес функціонування чат-бота з пошуку найближчих заходів у містах України.

Об’єктом дослідження є процес інформаційної підтримки пошуку найближчих заходів у містах України.

Актуальність роботи зумовлена достатньо високою популярністю месенджерів та ботів серед користувачів мережі Інтернет.

Метою роботи є розробка Telegram бота для пошуку найближчих заходів у містах України.

Для досягнення мети необхідно вирішити такі задачі:

– аналіз обраної предметної області;

– порівняння програмних продуктів-аналогів;
– вибір технологій і середовища розробки;
– здійснення проектування чат-бота;
– розробка чат-бота;
– тестування чат-бота;
– оформлення супровідної документації.

Практичним значенням є надання користувачам можливості пошуку та вибору заходів, враховуючи їх вподобання.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1Обгрунтування потреби у створенні чат-ботів

Оскільки, інформаційні технології стрімко розвиваються користувач стає все більш вимогливим до якості роботи технологій та їх функціоналу. Сьогодні використання інформаційних технологій є невід'ємною частиною повсякденного життя. Люди використовують інформаційні технології для комунікації, автоматизації бізнес-процесів, розваг, отримання нової інформації. Крім цього, існує велика кількість різноманітних месенджерів, що прискорюють процес комунікації між людьми. Одним із найбільш популярних та зручних месенджерів є Telegram.

Telegram – безкоштовний месенджер для смартфонів і персональних комп'ютерів, який працює під управлінням всіх найбільш поширених на сьогоднішній день операційних систем. Telegram надає можливість обмінюватися не лише текстовими повідомленнями, але і різними мультимедійними файлами.[1]

Месенджер Telegram використовують більше ніж 500 мільйонів користувачів по всьому світу. Його переваги полягають у шифруванні та підвищеній безпеці інформації. За результатами досліджень сервісу Statista.com за липень 2021 року Telegram входить до рейтингу найбільш популярних месенджерів світу (рис. 1.1). [2]

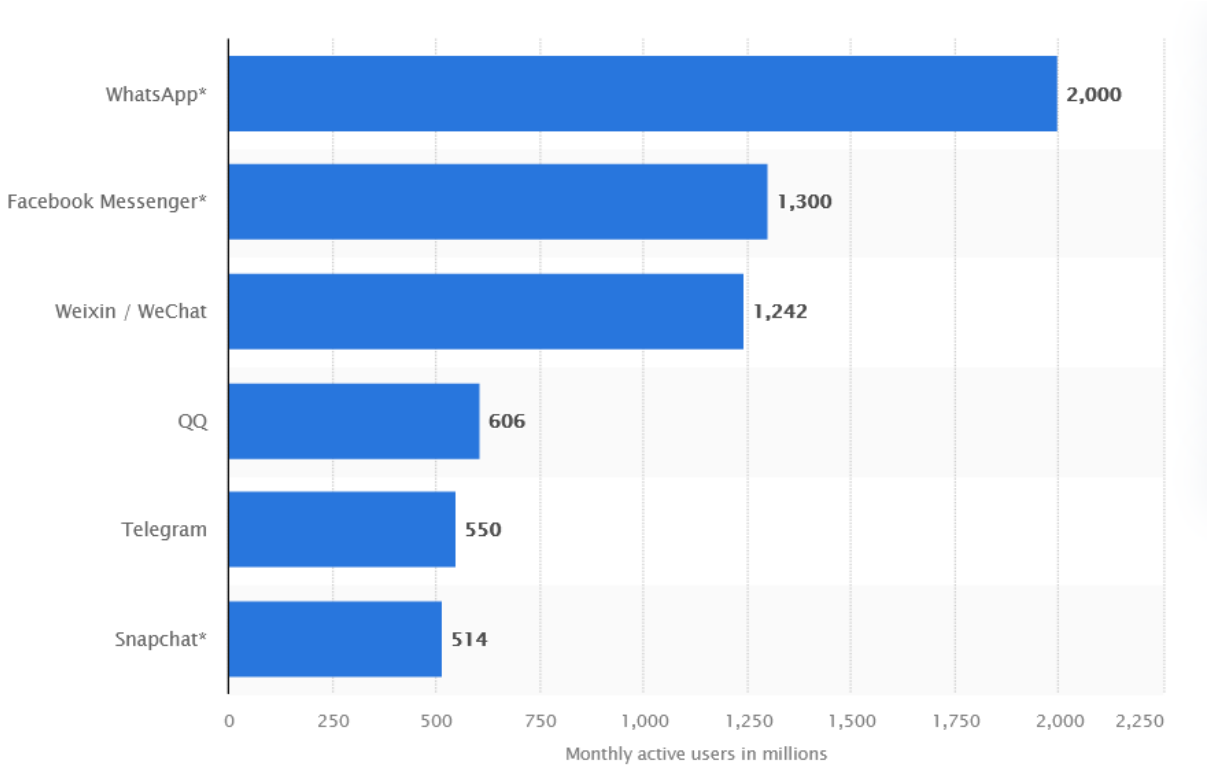


Рисунок 1.1 – Рейтинг найпопулярніших месенджерів світу

Також Telegram відомий своїми чат-ботами, які значно полегшують життя людини.

Чат-бот – це програма, яка імітує реальну розмову з користувачем. Чат-боти дозволяють спілкуватися за допомогою текстових або аудіо повідомлень на сайтах, в месенджерах та мобільних додатках [3]. Чат-боти в основному використовують штучний інтелект для спілкування з користувачами, тому надають релевантний контент і актуальні пропозиції. Вони функціонують на основі набору інструкцій або використовують машинне навчання. Чат-ботів вважають одним із найбільш досконалих та перспективних засобів організації взаємодії користувача з системою. До основних функцій чат-ботів належать:

– підтримка клієнтів, яка здійснюється 24 години на добу;

– клієнтський сервіс. За допомогою чат-бота можна робити покупки та запитувати різні послуги;

– чат-боти допомагають оптимізувати в роботі такі процеси як: бронювання та замовлення квитків, інформування людей, розклад подій і багато іншого [4].

На даний момент існує велика різноманітність чат-ботів, які задовольняють різні потреби користувача:

– чат-боти для бронювання квитків;
– чат- боти для пошуку книг;
– чат-боти для перегляду фільмів;
– чат-боти для перегляду розкладу транспорту;
– чат-боти для замовлення їжі;
– чат-боти для бронювання номерів у готелях;
– чат-боти для відстеження погодних умов;
– чат-боти, які слугують для системної підтримки користувачів

Незважаючи на те, що популярність у використанні ботів зростає, кількість чат-ботів, які б допомагали користувачам у пошуку найближчих заходів, відповідно до їх розташування та побажань не висока. Тому було прийнято рішення створити власного чат-бота, який би дозволив задовольнити потреби користувачів.

1.2 Опис технологій розоробки ботів

У 2016 році в галузі штучного інтелекту відбулося різке зростання інтересу до розумних співрозмовників, чат-ботів. Зростання обумовлено двома факторами: зростання попиту з боку бізнес-замовників та появою достатньої кількості інструментів та технологій для створення чат-ботів [5].

Чат-боти за принципом роботи можна розділити на два види: які працюють за заздалегідь заданим шаблоном і які навчаються в процесі спілкування. Шаблонні (скриптові) боти не розуміють природної мови. Діалог в таких ботах представляє заздалегідь формований шаблон, а скрипт – дерево рішень, в якому відповідь на

питання відкриває новий, заздалегідь запрограмований сценарій. Діалоги в них зазвичай лінійні і структуровані. Шаблонні боти при спілкуванні з користувачем виділяють ключові слова і реагують на них. У таких ботах необхідно прописати команду для кожного слова чи фрази. Якщо при спілкуванні користувач не використовує ключові слова, то бот його не розуміє і виконує дії, передбачені для таких випадків, наприклад, пропонує звернутися до оператора [6].

Для чат-ботів такого типу використовуються технології базової обробки природної мови:

– сегментація (розбиття на речення);
– лематизація або стемінг (процес пошуку основи слова);
– виділення ключових слів та/або іменованих сутностей

І технології створення регулярних граматик (наприклад, AIML):

– регулярні вирази;
– змінні та масиви для запам’ятовування контексту
– умови, цикли, рекурсії та ін.

Боти, які навчаються, розробляються на основі рішень штучного інтелекту і використовують при побудові діалогу оброблення природної мови і машинне навчання [7]. Принцип їх роботи заснований на аналізі діалогу для подальшого удосконалення своїх комунікативних навичок. Такі боти вміють оброблювати природну мову і коректно відповідати на поставлені запитання [8]. Вони можуть ставити уточнюючі питання, щоб дізнатися про мету пошуку, і пропонують товар, виходячи з відповідей покупця. Чат-боти, що навчаються зберігають унікальні пошукові запити від різних користувачів і вдосконалюються, відповідаючи все більш точно.

Наявність великої кількості готових бібліотек спрощує роботу розробника при написанні бота. Завдання, які потрібно вирішувати програмісту, можна узагальнити невеликим списком:

– реалізація розуміння роботом мови;

– створення алгоритмів пошуку оптимальної відповіді;

– інтеграція зі сторонніми API. Написання інших алгоритмів для обробки та видачі даних.

У результаті аналізу технологій створення бота було визначено, що реалізація власного продукту буде здійснюватися за допомогою заздалегідь запрограмованого сценарію, оскільки розроблюваний бот повинен буде виконувати шаблонні команди. Також існує велика кількість різноманітних інструментів та бібліотек, що полегшують розробку бота.

1.3Аналіз існуючих ботів для пошуку найближчих заходів

Перед початком проектування та розробки чат-бота потрібно визначити та описати основні вимоги. Також для то55го, щоб переконатися в необхідності розробки потрібно проаналізувати предметну область, розглянувши вже існуючі рішення.

У ході дослідження було виділено такі аналоги:

– Kyiv City Bot – бот який шукає та рекомендує івенти у місті Київ. Пошук відбувається серед різноманітності концертів, вечірок, лекцій, кінопоказів, вистав та інших заходів, які проходять у місті. Користувач має змогу обрати захід за кількома параметрами:

1. Місце;
2. Час

Kyiv City Bot простий у використанні і максимально швидкий, щоб користувачі з легкістю і не втрачаючи часу шукали улюблені заходи. Недоліками даного чат-бота є те що він шукає заходи лише у межах міста Київ, а також немає можливості пошуку подій за їх видом [9]. Приклад інтерфейсу чат-бота зображено на рисунках 1.2.

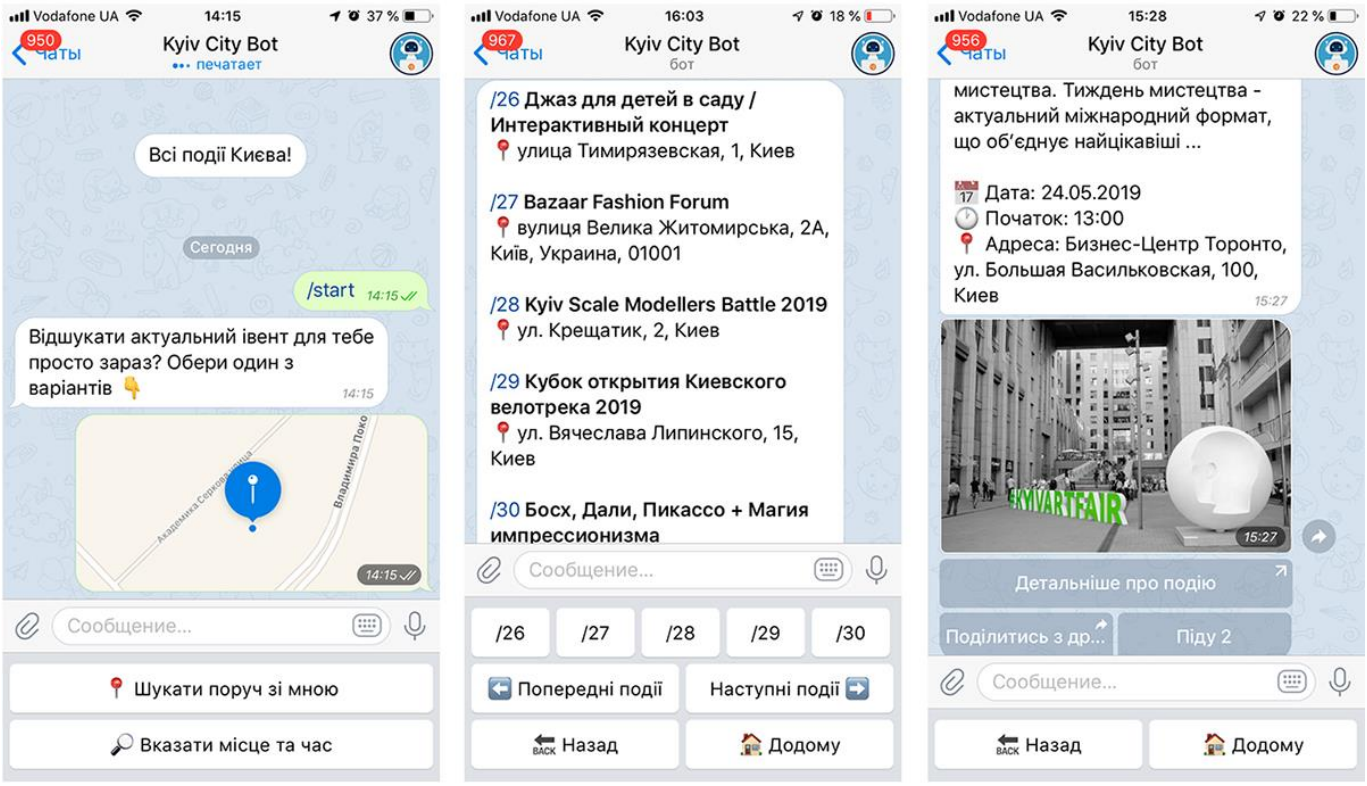


Рисунок 1.2 – інтерфейс чат-бота Kyiv City Bot

– MoeMisto.ua – бот, який надає можливість користувачеві отримувати інформацію про всі заходи у місті. Можливості бота містять пошук та підбір анонсів, подій та заходів по відповідним параметрам, а саме:

1. пошук анонсів по даті проведення заходу;
2. підбір анонсів по тематичним рубрикам, таким як концерти, вечірки, дитячі заходи, кіно, подорожі;
3. афіші розваг та відпочинку у закладах чи по місцях проведення.

Чат-бот надає можливість пошуку подій у таких містах України як: Вінниця, Житомир, Тернопіль, Одеса, Хмельницький, Київ, Львів, Кропивницький [10].

Приклад інтерфейсу бота MoeMisto.ua зображено на рисунку 1.3

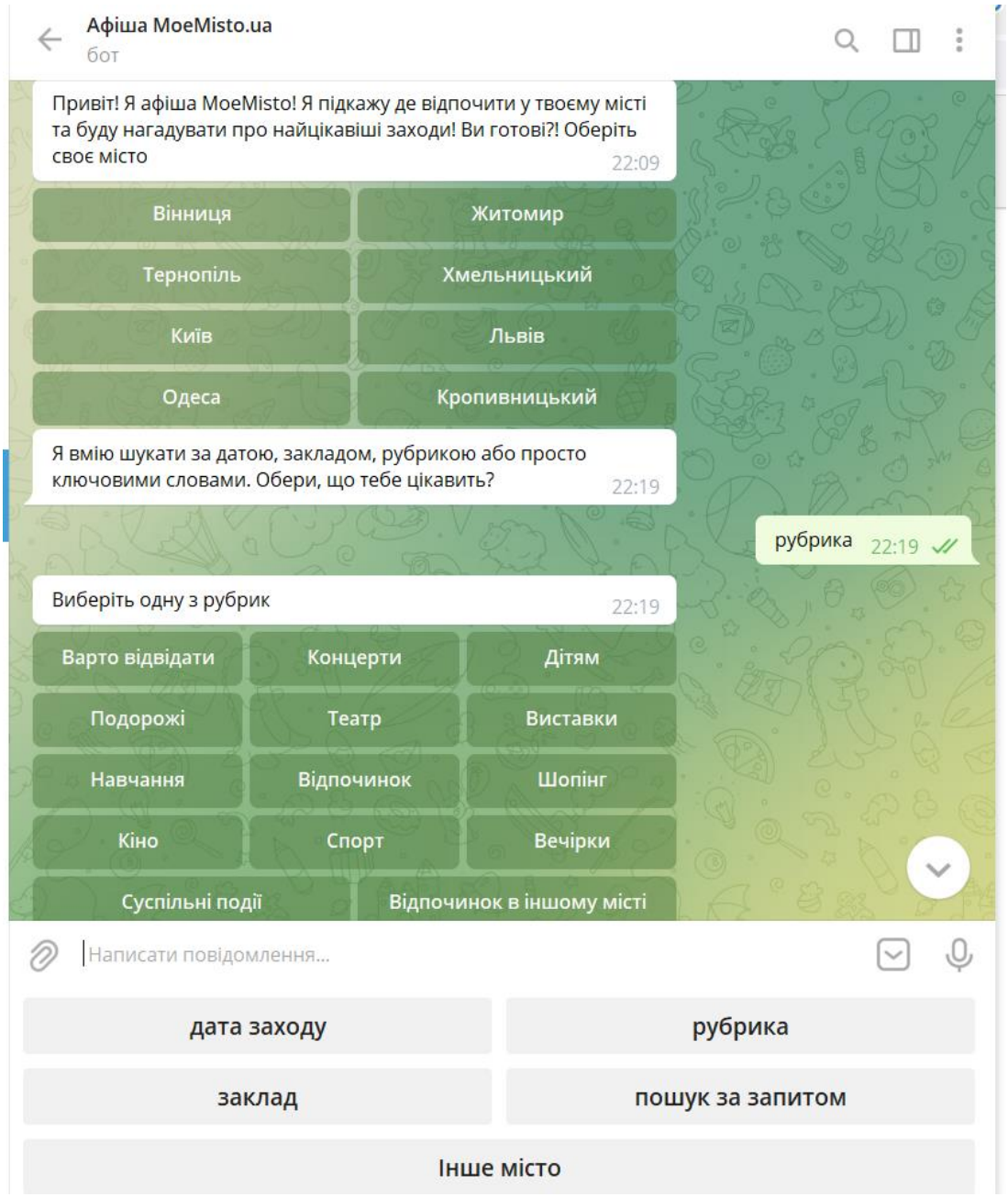


Рисунок 1.3 – Приклад інтерфейсу бота MoeMisto.ua

– Music Trip Planner – бот для пошуку концертів музичних гуртів. Пошук здійснюється по назві гурту або місці проведення концерту [11]. На рисунку 1.4 зображено приклад інтерфейсу чат-бота.

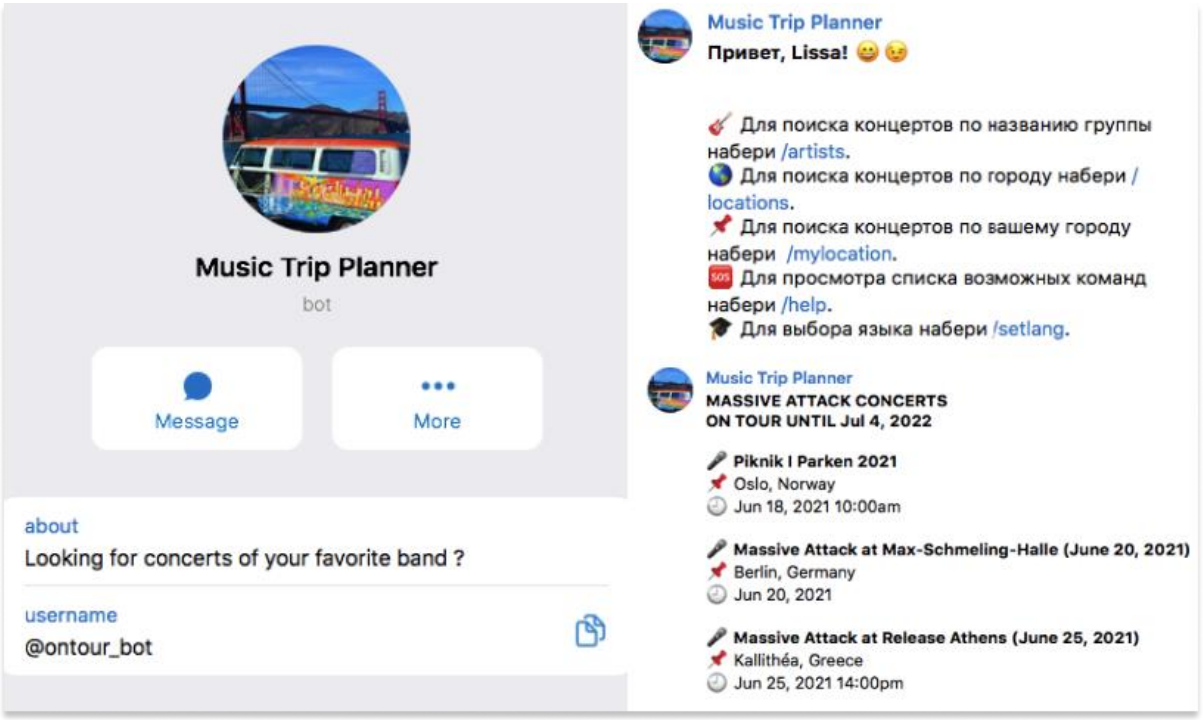


Рисунок 1.4 – Інтерфейс Music Trip Planner бота

Порівняльну характеристику розглянутих чат-ботів зображено в таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика чат-ботів

Критерій	Kyiv City Bot	MoeMisto.ua	Music Trip Planner
Пошук заходів по місцю проведення	+	+	+
Пошук за типом заходу	-	+	-
Пошук заходів за датою	+	+	-
Можливість зміни мови	-	-	+
Перегляд детальної інформації про події	+	+	-

Продовження таблиці 1.1 – Порівняльна характеристика чат-ботів

Критерій	Kyiv City Bot	MoeMisto.ua	Music Trip Planner
Додавання власних заходів	-	-	-
Додавання заходів до списку обране	-	-	-
Додавання нагадувань про заходи	-	-	-

Отже, в результаті аналізу аналогів було визначено, що для задоволення потреб користувача необхідно розробити чат-бот, що матиме такі функції: пошук заходів по місцю проведення, пошук за типом заходу, пошук заходів за датою, можливість зміни мови, перегляд детальної інформації про події.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1Мета та задачі роботи

Головною метою роботи є розробка Telegram бота для пошуку найближчих заходів у містах України. Даний проект буде мати цінність для великої кількості користувачів, які бажають знайти заходи у їхньому рідному місті.

Для досягнення мети роботи були визначені такі задачі:

- здійснити аналіз предметної області;
- здійснити аналіз вимог;
- проаналізувати технології розробки ботів;
- обрати методи та засоби реалізації;
- здійснити проектування бота;
- розробка бота;
- тестування бота;
- розробка супровідної документації.

Після встановлення мети та задач проекту було визначено, що чат-бот повинен мати такі функціональні можливості:

- мати команди для інтерактивного спілкування з ботом;
- мати команди для запуску роботи;
- мати можливість пошуку заходів за місцем проведення;
- мати можливість пошуку заходів за їх видами;
- мати можливість пошуку заходів за датою проведення;
- мати можливість пошуку заходів за ключовими словами;
- мати можливість перегляду детальної інформації про захід;
- мати можливість зміни мови інтерфейсу бота;
- мати можливість встановлювати нагадування про наближення вибраних заходів.

- мати можливість збереження заходів до списку бажань;
- мати можливість фільтрації заходів.

Після виконання аналізу було встановлено що, чат-бот буде корисним для широкого кола користувачів для пошуку заходів за різними категоріями.

2.2Вибір засобів реалізації мобільного чат-бота

Перед початком реалізації чат-бота потрібно обрати засоби реалізації. В якості мови програмування було обрано Python.

Python — це високорівнева мова програмування загального призначення, яка використовується навіть для розробки веб-додатків та чат-ботів. Мова орієнтована на підвищення продуктивності розробника і читання коду [12].

Python підтримує кілька парадигм програмування: структурне, об'єктно-орієнтоване, функціональне, імперативне та аспектно-орієнтоване. У мові є динамічна типізація, автоматичне управління пам'яттю, повна інтроспекція, механізм обробки винятків, підтримка багатопоточних обчислень та зручні високорівневі структури даних. Програмний код Python організовується в функції і класи, які можуть об'єднуватися в модулі, а вони в свою чергу можуть бути об'єднані в пакети. Python зазвичай використовується як інтерпретований, але також може бути скомпільований у байт-код Java та в MSIL (в рамках платформи .NET).

Python — одна з основних мов програмування, які застосовують у галузі машинного навчання та штучного інтелекту (Machine Learning та Artificial Intelligence). Python займає друге місце у списку найпопулярніших мов програмування. Він випереджає JavaScript, PHP, Swift та інші поширені мови, поступаючись лише C.

У якості редактору коду було обрано Sublime Text. Sublime Text Editor – це повнофункціональний текстовий редактор для редагування локальних файлів або бази коду. Він включає різні функції для редагування бази коду, яка допомагає

розробникам відстежувати зміни. До функцій, що підтримуються в Sublime належать:

- підсвічування синтаксису;
- автовідступ;
- розпізнавання типів файлів;
- бічна панель із файлами вказаного каталогу;
- макрос;
- плагін та пакети [13].

Sublime Text editor використовується як інтегрований редактор розробки (IDE), як Visual Studio і NetBeans. Поточна версія редактора Sublime Text – 3.0 та сумісна з різними операційними системами, такими як Windows, Linux та MacOS.

SQLite – це система управління базами даних, відмінною особливістю якої є її вбудовування в додатки. Використовуючи високоефективну інфраструктуру, SQLite може працювати в невеликому обсязі пам’яті, що виділяється для неї, набагато меншому, ніж у будь-яких інших системах БД. Це робить SQLite дуже зручним інструментом з можливістю використання практично в будь-яких завданнях, що покладаються на базу даних. До переваг SQLite відносяться: надійність, відкритість вихідних кодів, наявність консольної утиліти для роботи з базами, популярність [14].

Також для розробки бота для месенджеру Телеграм знадобиться пакет python-telegram-bot – оболонка для API від Telegram. Написати бота за допомогою цієї бібліотеки дуже просто, оскільки вона повністю сумісна з Python 3.6+ [15]. Також для створення чат-бота необхідний Telegram канал @BotFather, який відповідає за реєстрацію нових ботів. Він також відповідає за базове налаштування (опис, фото профілю, вбудована підтримка тощо).

3 ПРОЕКТУВАННЯ ЧАТ-БОТА

3.1 Діаграми процесу забезпечення пошуку заходів по містах України у нотаціях IDEF0

Перед розробкою програмного продукту потрібно визначити структуру та функції чат-бота та побудувати функціональну модель. Функціональна модель призначена для опису існуючих бізнес-процесів, у яких використовуються як природна, так і графічна мови [16]. З цією метою було побудовано діаграми процесу забезпечення пошуку заходів по містах України у нотаціях IDEF0.

IDEF0 – методологія функціонального моделювання. За допомогою наочної графічної мови IDEF0, система, що вивчається, постає перед розробниками та аналітиками у вигляді набору взаємопов’язаних функцій (функціональних блоків – у нотаціях IDEF0). Як правило, моделювання засобами IDEF0 є першим етапом вивчення будь-якої системи [17].

Для відображення структурно-функціональної моделі процесів роботи чат-бота спочатку було побудовано контекстну діаграму, що відображає процес забезпечення пошуку заходів по містах України. Для її побудови було виділено: вхідні дані, вихідні дані, механізми та керуючі елементи.

Вхідними даними процесу є запит користувача.

Вихідними даними є:

- результати пошуку заходів;
- заходи, додані до списку «Обране»;
- налаштовані нагадування про заходи;
- нові додані заходи до БД.

Керуючими елементами процесу в даному випадку є:

- функціональні вимоги до бота;
- інструкції користувача;

Механізмами є:

– мобільний чат-бот;

– база даних;

– Telegram.

Контекстну діаграму зображено на рисунку 3.1

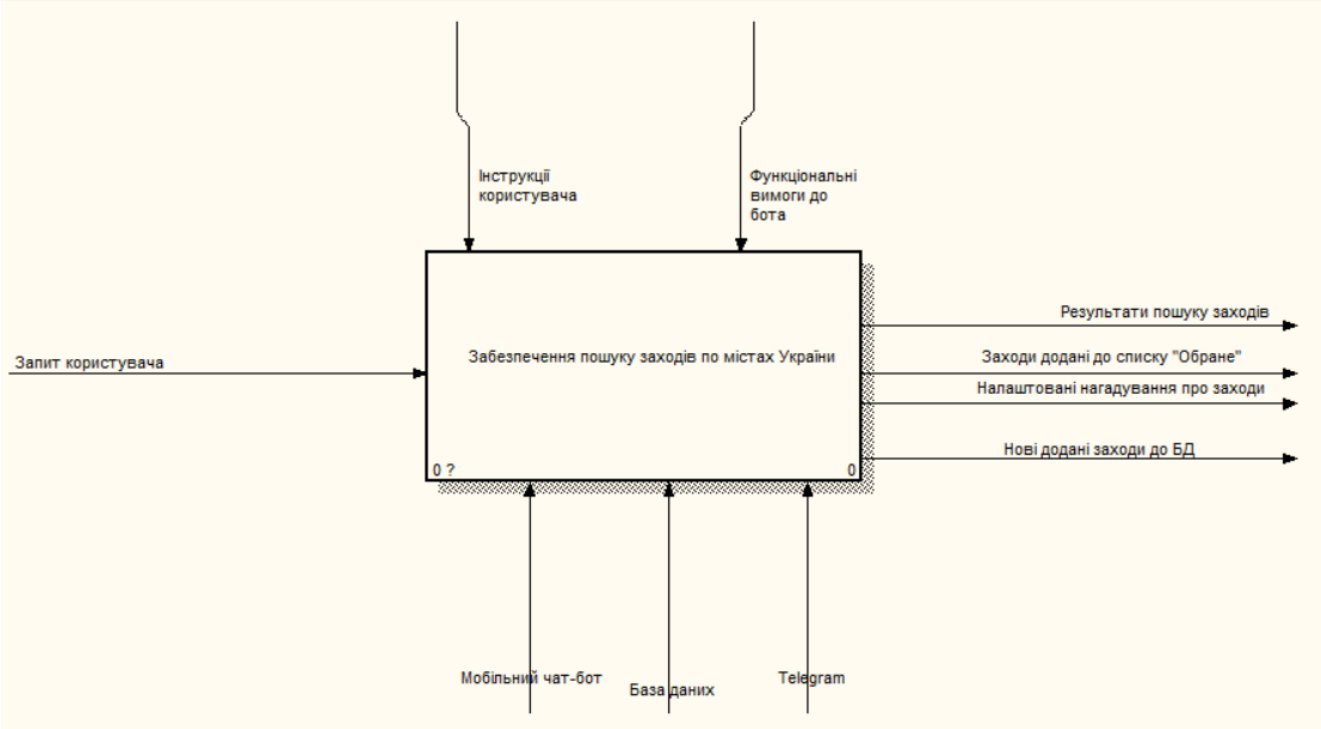


Рисунок 3.1 – Контекстна діаграма процесу забезпечення пошуку заходів по містах України

Після побудови контекстної діаграми потрібно здійснити функціональну декомпозицію. Для цього систему потрібно розбити на підсистеми і описати кожну з них окремо (діаграма декомпозиції).

Після декомпозиції процесу було виділено такі блоки:

– авторизація користувача у системі;

– вибір мови інтерфейсу;

– додавання власних заходів;

– пошук заходів;

– додавання заходів до списку «Обрані»;

– створення нагадувань про обраний захід.

Діаграму декомпозиції процесу зображено на рисунку 3.2

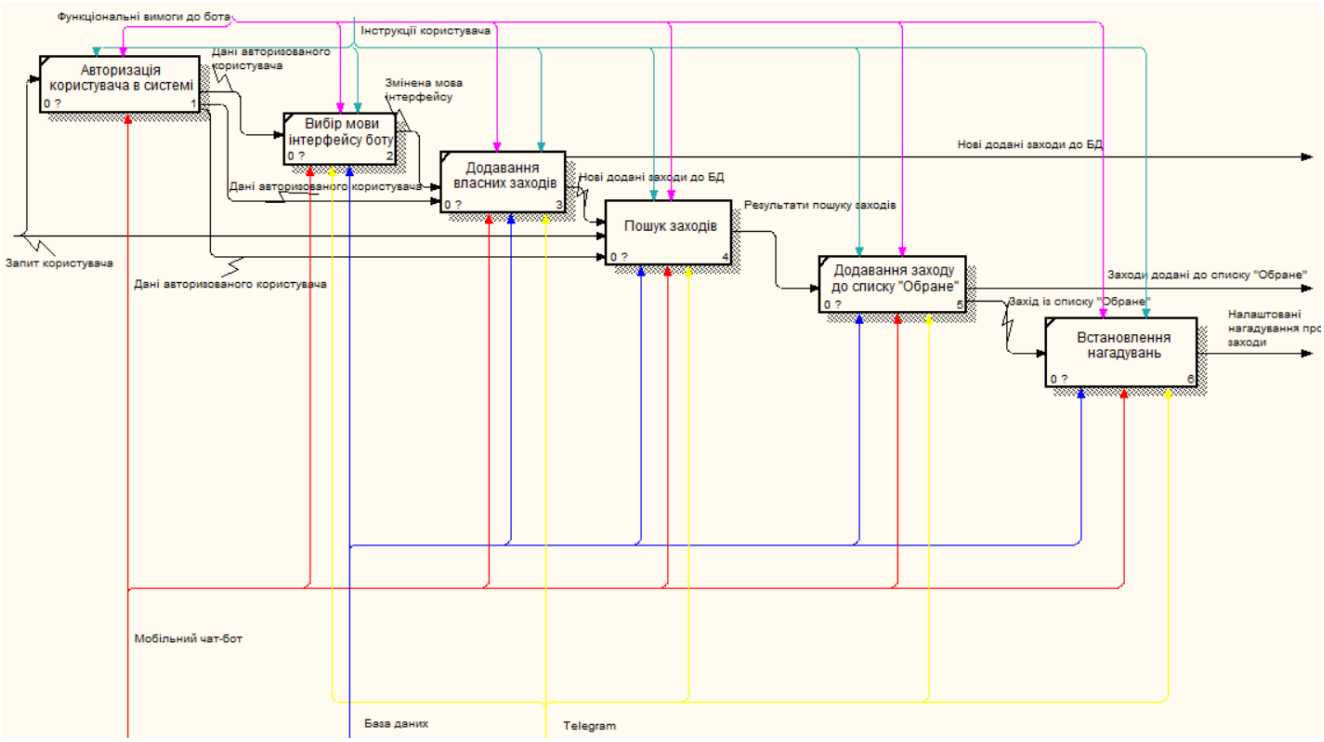


Рисунок 3.2 – Діаграма декомпозиції процесу забезпечення пошуку заходів по містах України

Далі було здійснено декомпозицію другого рівня, де було декомпозовано безпосередньо сам процес пошуку заходів.

У результаті декомпозиції процесу пошуку заходів було виділено такі блоки:

- вибір міста;
- вибір критеріїв для подальшого пошуку;
- вибір категорії заходу;
- обробка введених даних системою.

Діаграму декомпозиції другого рівня зображено на рисунку 3.3

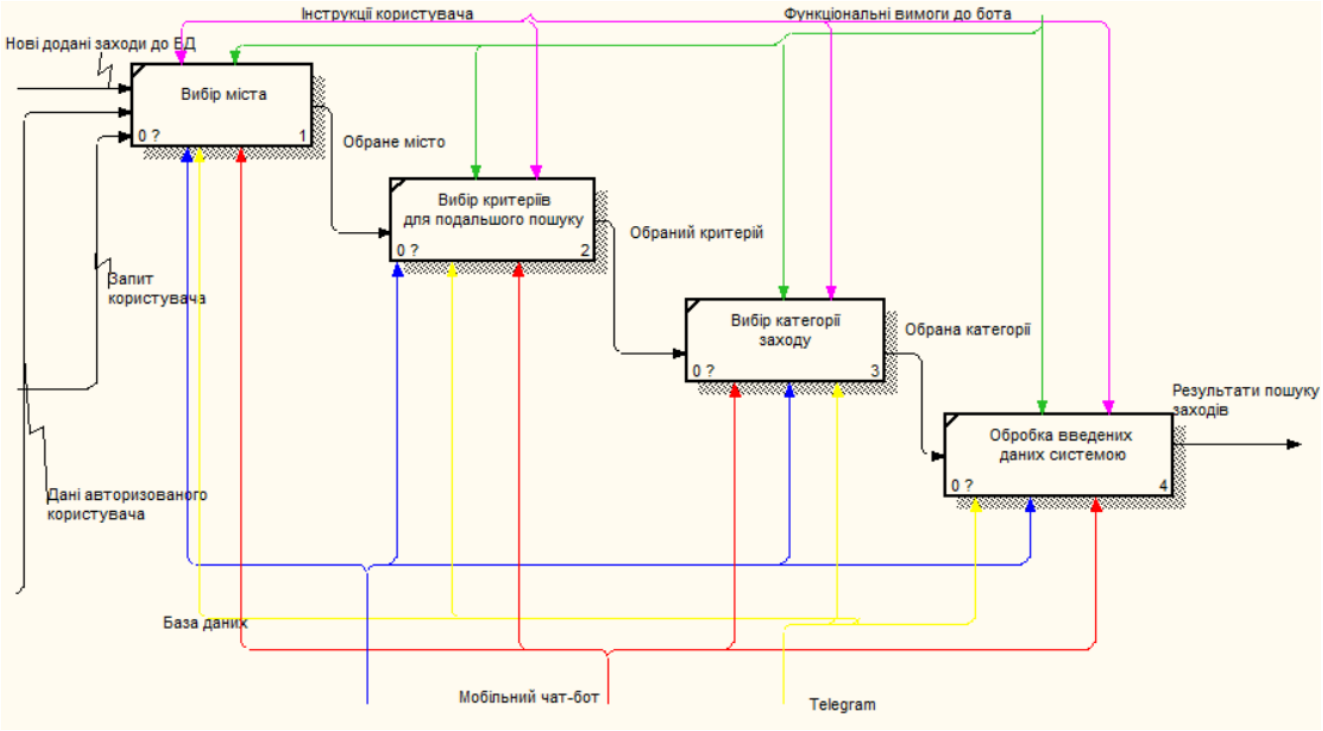


Рисунок 3.3 – Діаграма декомпозиції другого рівня

3.2 Моделювання варіантів використання чат-бота

На наступному етапі було створено діаграму варіантів використання чат-бота.

Діаграма варіантів використання – діаграма, яка описує, який функціонал розробленої програмної системи, доступний кожній групі користувачів [18].

На діаграмі використання зображаються:

– актори – групи осіб або систем, що взаємодіють із розроблюваною системою;

– варіанти використання (прецеденти) – послуги, які розроблювана система надає акторам;

– коментарі;

– відносини між елементами діаграми [19].

Діаграму варіантів використання зображено на рисунку 3.4.

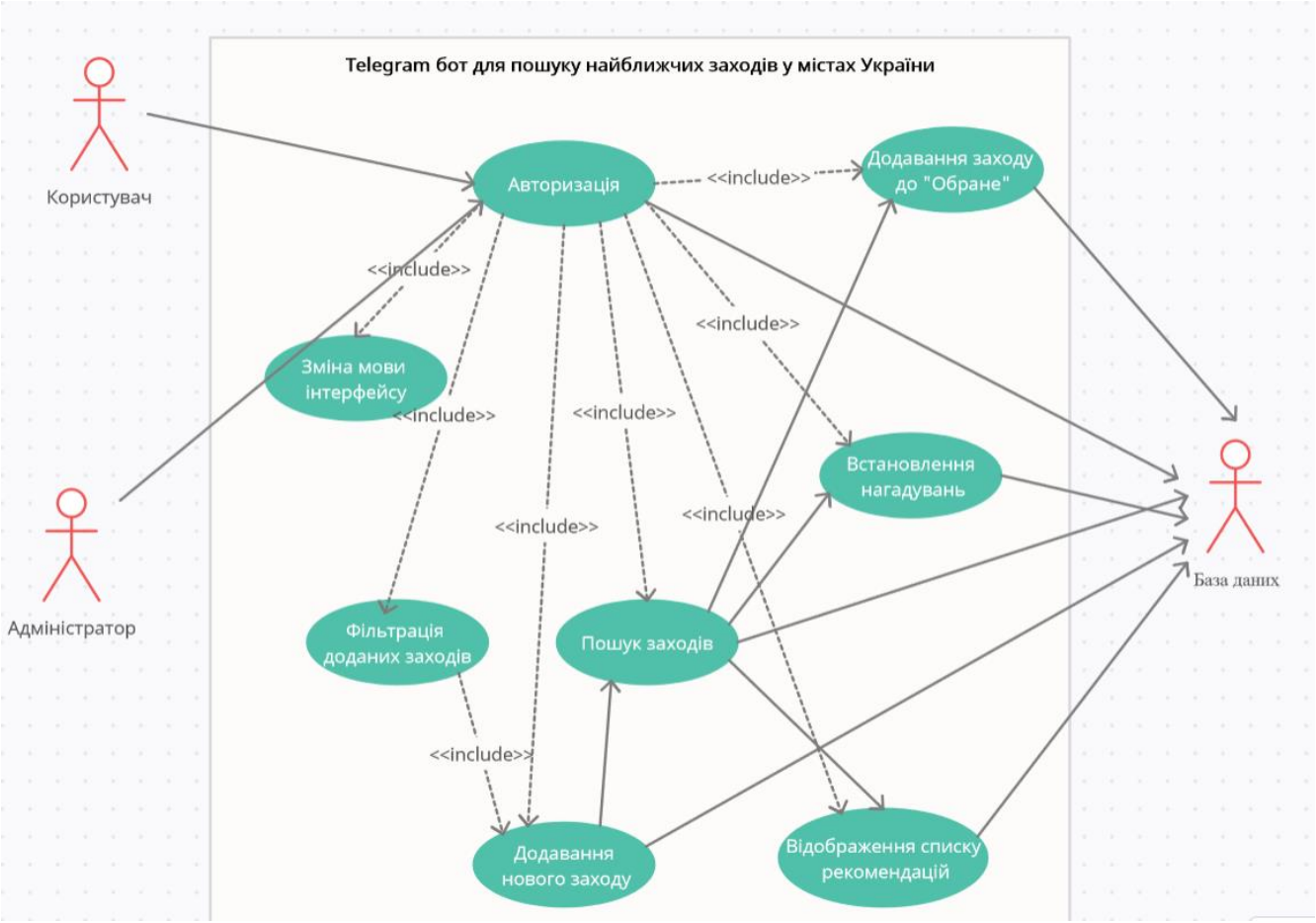


Рисунок 3.4 – Діаграма варіантів використання

За актора було виділено користувача чат-бота, який здійснює пошук заходів, адміністратора, який відповідає за фільтрацію заходів та базу даних, що зберігає дані, які використовуються або додаються впродовж роботи з ботом.

Варіанти використання:

– ВВ авторизація – надає можливість авторизації користувача в системі;

– ВВ зміна мови інтерфейсі – надає можливість зміни мови інтерфейсу чат-бота;

– ВВ пошук заходів – надає можливість здійснювати пошук заходів за містом, датою, категорією заходу, місцем проведення, за ключовими словами;

– ВВ додавання нового заходу – надає можливість користувачу додати власний захід до бази даних. Всі додані заходити будуть доступні для перегляду будь-яким користувачам бота ;

– ВВ додавання заходів до списку «Обране» – надає можливість додавання обраного користувачем заходу до окремого списку. Користувач має можливість перегляду даного списку, обравши в основному меню пункт «Обране»;

– ВВ створення нагадувань про обраний захід – надає можливість користувачу додати нагадування про наближення заходу. Користувач обирає дату та час для встановлення нагадування та має можливість перегляду даного списку, обравши в основному меню пункт «Мої нагадування»;

– ВВ відображення списку рекомендованих заходів – надає можливість користувачу переглядати список рекомендованих особисто для нього заходів;

– ВВ фільтрація заходів – здійснення фільтрації заходів перед додаванням в базу даних.

3.3 Проектування моделі бази даних

ER-діаграма – це різновид блок-схеми, де показано, як різні сутності пов’язані між собою всередині системи. ER-діаграми найчастіше застосовуються для проектування та налагодження реляційних баз даних у сфері освіти, дослідження та розробки програмного забезпечення та інформаційних систем для бізнесу [20]. ER-діаграми (або ER-моделі) покладаються на стандартний набір символів, включаючи прямокутники, ромби, овали та сполучні лінії для відображення сутностей, їх атрибутів та зв'язків. Ці діаграми влаштовані за тим самим принципом, як і граматичні структури [21].

Далі було здійснено проектування бази даних та було створено ER-діаграму.

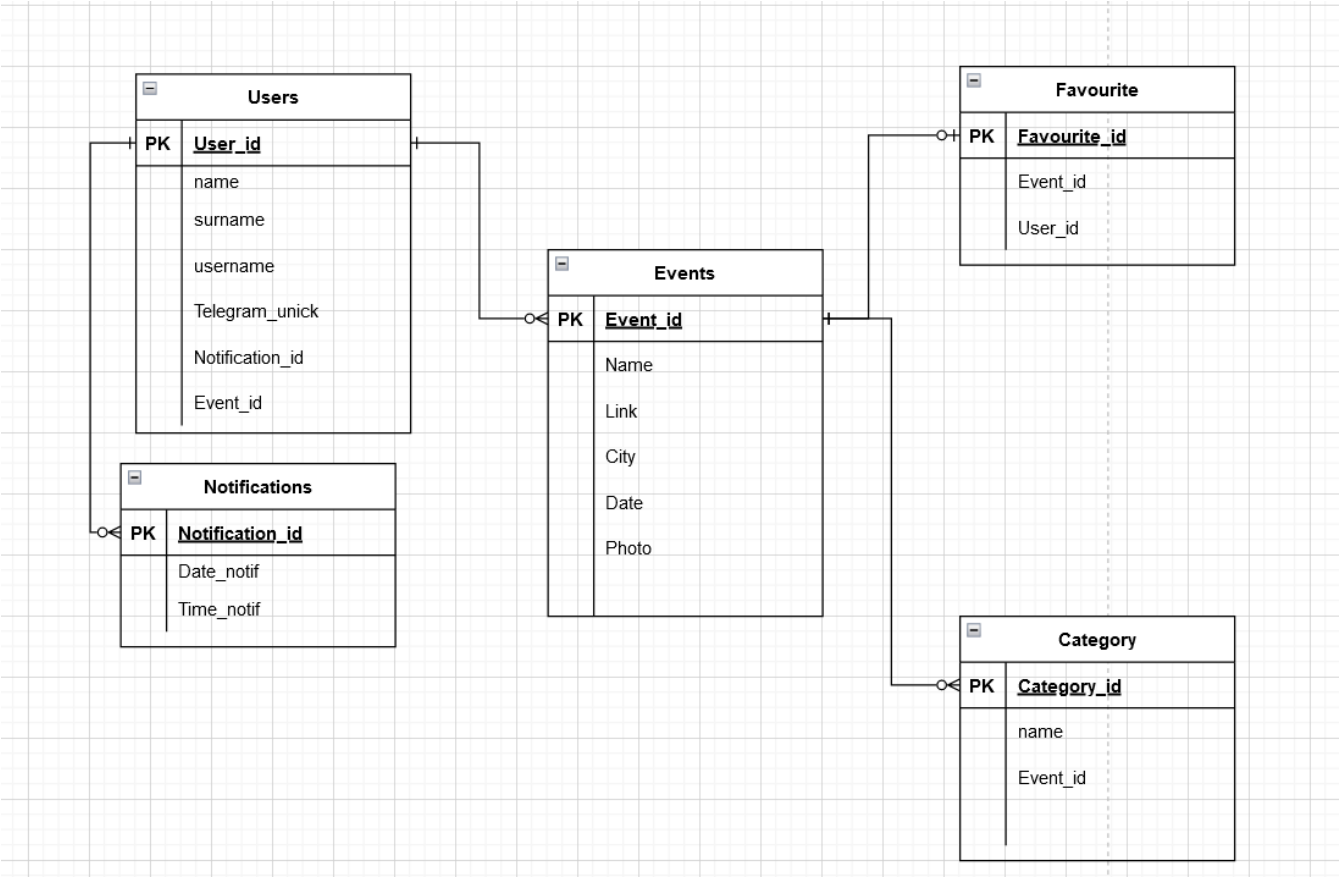


Рисунок 3.5 – ER-діаграма

Дана база даних містить 6 таблиць, що мають певні атрибути:

– Users – таблиця, що містить дані про користувачів чат-бота. Таблиця містить такі атрибути: User_id (ідентифікатор користувача), name (ім’я користувача в месенджері Телеграм), surname (прізвище користувача в месенджері Телеграм) та username (нік користувача в месенджері Телеграм), Notification_id (ідентифікатор нагадування), Event_id (ідентифікатор заходу);

– Events – таблиця, що містить дані про існуючі заходи. Таблиця містить такі атрибути: Event_id (ідентифікатор заходу), Name (назва заходу), Photo (фото заходу), City (місце проведення), Date (дата заходу), Link (посилання);

– Category – таблиця, що містить назви категорій заходів. Таблиця містить такі атрибути: Category_id (ідентифікатор категорії), Name (назва категорії), Event_id (ідентифікатор заходу);

– Notifications – таблиця, що містить список із створеними нагадуваннями. Таблиця містить такі атрибути: Notification_id (ідентифікатор нагадування), Date_notif (дата нагадування), Time_notif (час нагадування);

– Favourite – таблиця, що містить заходи із списку «Обране». Таблиця містить такі атрибути: Favourite_id (ідентифікатор обраного заходу), Event_id (ідентифікатор заходу), User_id (ідентифікатор користувача).

4 РЕАЛІЗАЦІЯ ЧАТ-БОТА

4.1 Архітектура чат-бота

Архітектура чат-бота представляє концепцію, яка визначає структуру і взаємозв’язки між його компонентами [22].

До компонентів архітектури чат-бота відносяться:

– Telegram – месенджер, що є платформою для розробки чат-бота;

– Telegram Bot API – є посередником між Telegram та обробником команд, саме він отримує команди, у вигляді повідомлень, від користувача, а також відправляє відповідь на ці команди користувачу;

– Обробник команд отримує команди від Telegram Bot API і за допомогою прописаних сценаріїв та бази даних відправляє відповідь до Telegram Bot API.

Архітектура чат-бота зображена на рисунку 4.1

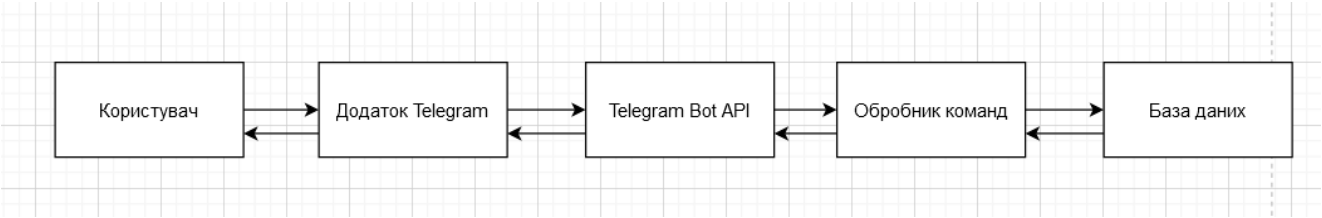


Рисунок 4.1 – Архітектура чат-бота

4.2 Реалізація чат-бота

Перед початком розробки чат-бота необхідно необхідно зареєструвати його за допомогою спеціального бота @BotFather [23]. Реєстрацію чат-бота зображено на рисунку 4.2

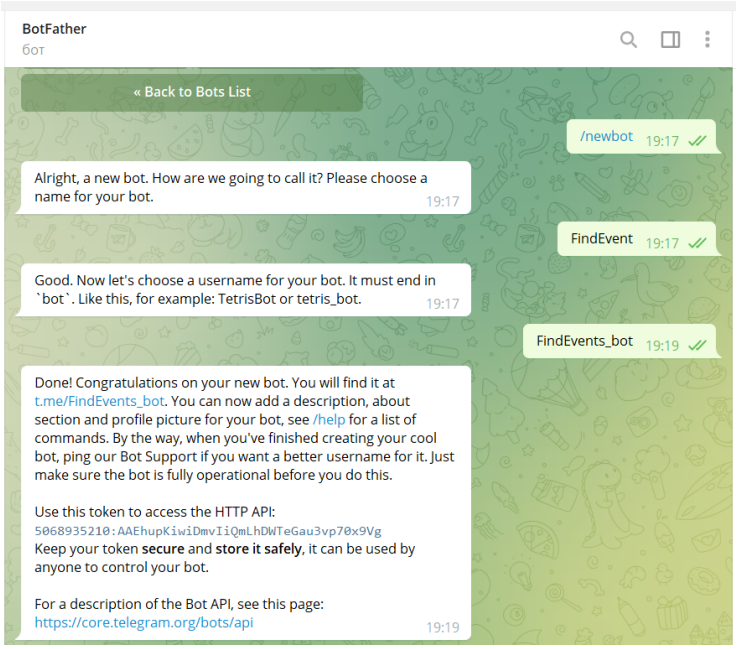


Рисунок 4.2 – Реєстрація бота

Далі було встановлено фото профілю та опис бота.

Фото профілю та опис бота зображено на рисунку 4.3.

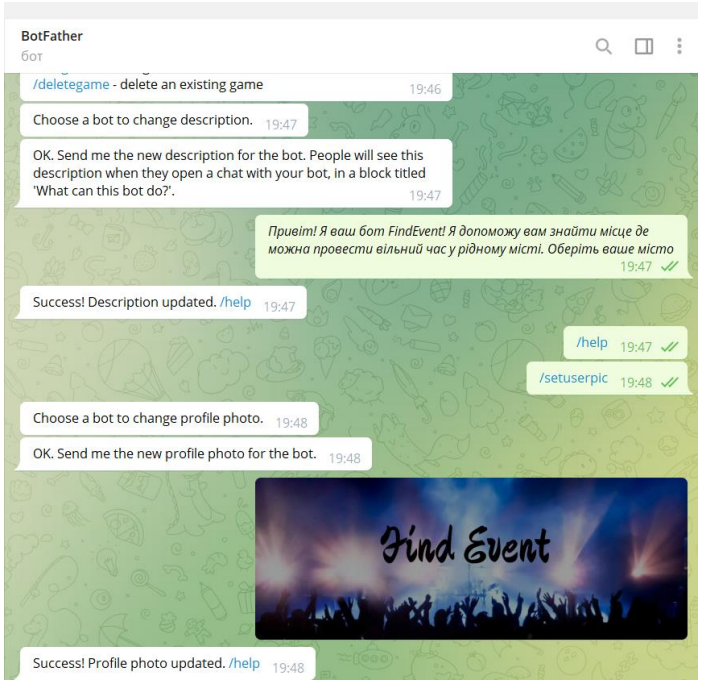


Рисунок 4.3 – Встановлення фото прфілю та опису бота

Наступним кроком було імпортовно необхідні бібліотеки та підключено токен отриманий при реєстрації бота [24].

Для розробки Telegram бота було підключено такі бібліотеки [25]:

– sqlite3;

– telebot;

– random.

Фрагмент даного коду зображено на рисунку 4.4

```
1   #!/usr/bin/python
2   # coding: utf-8
3   import sqlite3
4   import telebot
5   import random
6   from keyboards import *
7
8   database = "database.db"
9   statebase = "state.db"
10
11  bot = telebot.TeleBot("5082705820:AAGTATaDZalIAXTsGQkUCmLbcbP1Ehipd0w")
```

Рисунок 4.4 – Імпорт бібліотек та підключення токену

Далі було створено обробник для команди Start [26]. Фрагмент коду обробника команди Start зображено на рисунку 4.5

```
32  @bot.message_handler(commands=['start'])
33  def welcome(message):
34      con = sqlite3.connect(database)
35      cur = con.cursor()
36      cur.execute("SELECT id FROM users WHERE id = ?",(message.chat.id,))
37      user = cur.fetchone()
38      if user == None:
39          con = sqlite3.connect(database)
40          cur = con.cursor()
41          cur.execute(f"INSERT INTO users (id)"f"VALUES ({message.chat.id})")
42          con.commit()
43      bot.send_message(message.chat.id,"<i>Привіт! Я ваш бот FindEvent! Я допоможу вам знайти місце де можна провести вільний час у рідному місті. Оберіть ваше
44
45
```

Рисунок 4.5 – Обробник події Start

Далі було створено базу даних за допомогою середовища SQLite Studio [27]. Структура бази даних зображена на рисунку 4.6

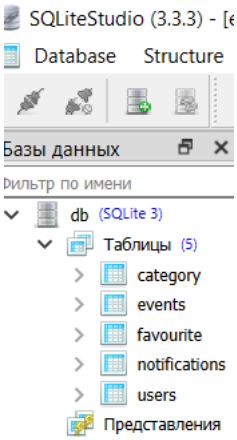


Рисунок 4.6 – Структура бази даних

Також було створено основне меню та інлайн кнопки. Фрагмент коду, що відповідає за створення основного меню зображено на рисунку 4.7

```
from telebot import types


def menu():

    markup= types.ReplyKeyboardMarkup(one_time_keyboard=False, resize_keyboard=True)
    btn1 = types.KeyboardButton("Пошук заходу")
    btn2 = types.KeyboardButton("Змінити мову")
    btn3 = types.KeyboardButton("Обране")
    btn4 = types.KeyboardButton("Додати захід")
    btn5 = types.KeyboardButton("Мої нагадування")
    btn6 = types.KeyboardButton("Рекомендації")

    markup.row(btn1, btn4)
    markup.row(btn2)
    markup.row(btn3, btn6)
    markup.row(btn5)

```

Рисунок 4.7 – Створення основного меню

Після цього було описано сценарії для кожної команди. Фрагмент коду, що відповідає за пошук заходів зображено на рисунку 4.8

```
    if search != "zapyt":
        poisk = message.text[:1].upper()+message.text[1:].lower()
        print(poisk)
        con = sqlite3.connect(database)
        cur = con.cursor()
        cur.execute(f"SELECT * FROM events WHERE {search} = ?",(poisk,))
        result = cur.fetchall()

        for result in result:
            bot.send_photo(message.chat.id,f"{result[7]}", f"{result[1]}\n{result[2]}\n {result[4]} {result[5]}\n<a href='{result[3]}'>Купити білет</a>",

    else:
        con = sqlite3.connect(database)
        cur = con.cursor()
        cur.execute("SELECT * FROM events WHERE name = ? or date = ? or city = ? or name = ? or category = ?",(message.text, message.text, message.text, m
        result = cur.fetchall()

        if result != None:
            bot.edit_message_text(chat_id=msg.chat.id, message_id=msg.message_id,text="<i>Результати пошуку:</i>", parse_mode = "HTML")

            for result in result:
                bot.send_photo(message.chat.id,f"{result[7]}", f"{result[1]}\n{result[2]}\n {result[4]} {result[5]}\n<a href='{result[3]}'>Купити білет</a
        else:
            bot.edit_message_text(chat_id=msg.chat.id, message_id=msg.message_id,text="<i>Нічого не знайдено</i>", parse_mode = "HTML")
```

Рисунок 4.8 – Фрагмент коду для пошуку заходів

4.3 Демонстрація роботи чат-бота

Для початку роботи з ботом користувачу необхідно ввести команду /start. Після цього відбудеться запуск бота та з’явиться коротке повідомлення та список міст для пошуку заходів. Процес запуску бота зображено на рисунку 4.9.

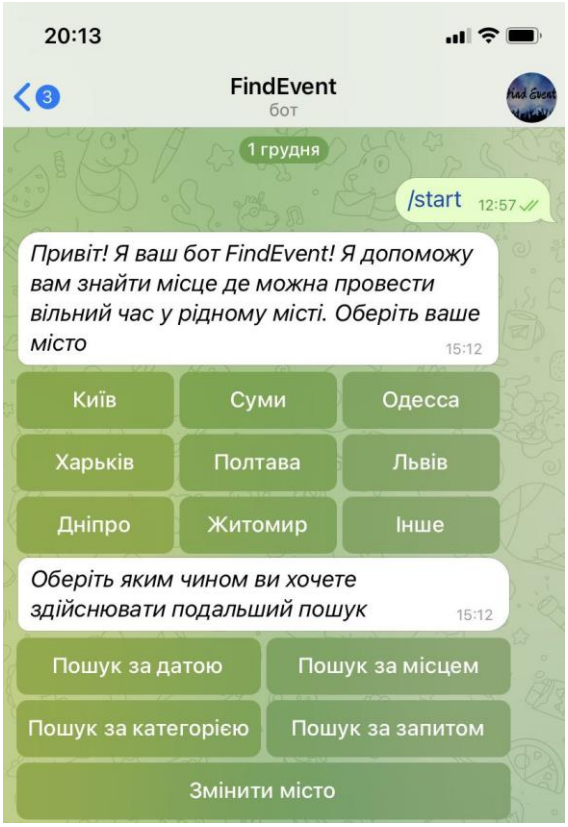


Рисунок 4.9 – Запуск бота

Крім цього користувачу доступне основне меню бота, яке складається з таких пунктів:

– Пошук заходу;
– Додати захід;
– Змінити мову;
– Обране;
– Рекомендації;
– Мої нагадування.

Вигляд головного меню бота зображено на рисунку 4.10

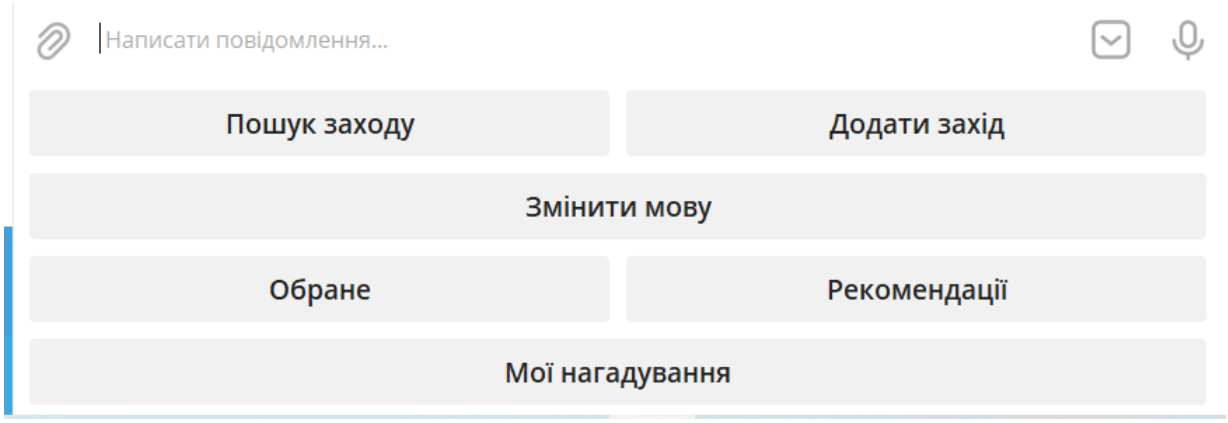


Рисунок 4.10 – Головне меню бота

При виборі пункту «Пошук заходу» користувачу відобразиться список міст. Список міст зображено на рисунку 4.11

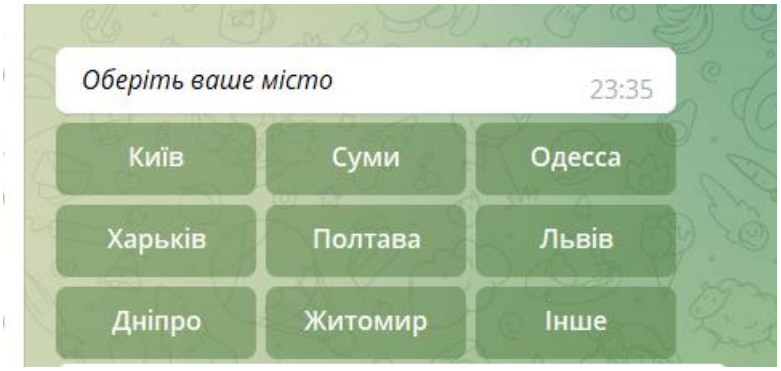


Рисунок 4.11 – Список міст

Далі користувач обирає місто для пошуку або вводить власне місто, вибравши пункт «Інше». Після цього користувачу відображується список критеріїв для подальшого пошуку. До критеріїв пошуку належать:

– Пошук за датою;
– Пошук за категорією;
– Пошук за місцем;
– Пошук за запитом.

Список критеріїв зображено на рисунку 4.12

Рисунок 4.12 – Список критеріїв для здійснення пошуку

В залежності від вибору критерію, користувачу надається можливість вибору категорії заходу або введення дати, місця, запиту пошуку. Приклад пошуку заходу за датою зображено на рисунку 4.13

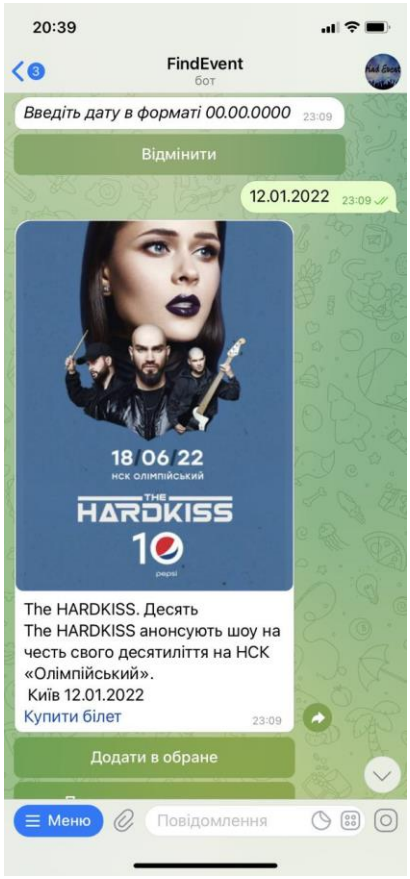


Рисунок 4.13 – Пошук заходів за датою

Після отримання результатів пошуку користувач має можливість додати захід до списку «Обране» або «Нагадування». Додавання заходу до списку «Обране» зображено на рисунку 4.14

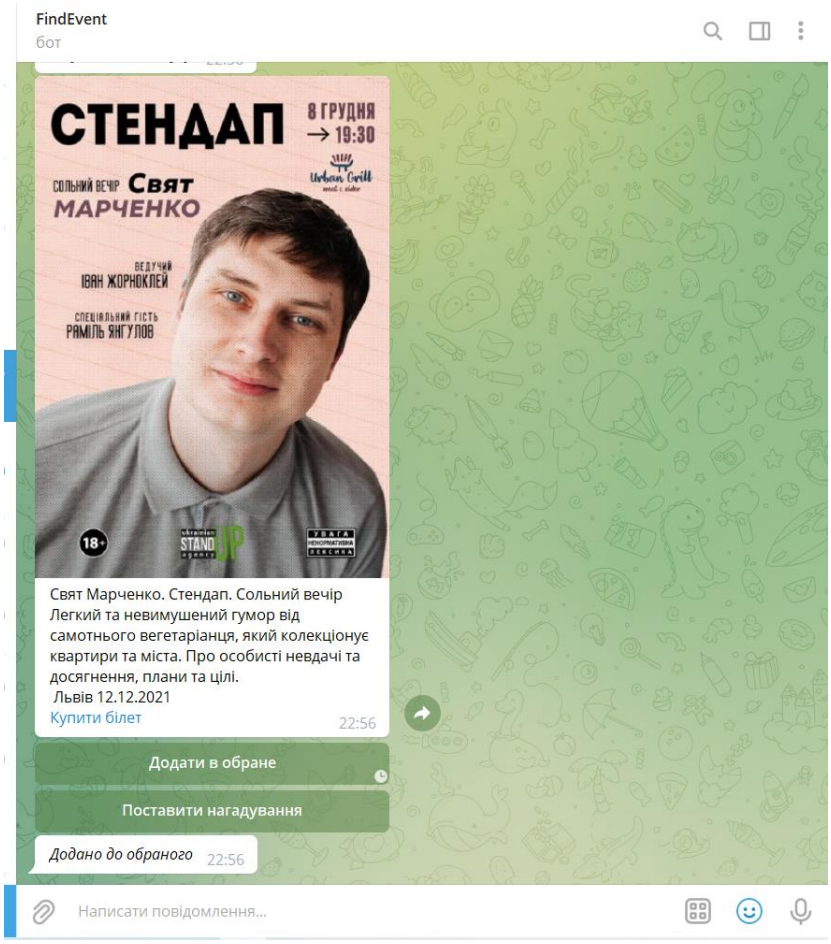


Рисунок 4.14 – Додавання заходу до списку «Обране»

Для перегляду списку та можливості видалення заходу з списку користувач повинен обрати відповідний пункт меню. Перегляд та видалення заходу із списку «Нагадування» зображено на рисунку 4.15

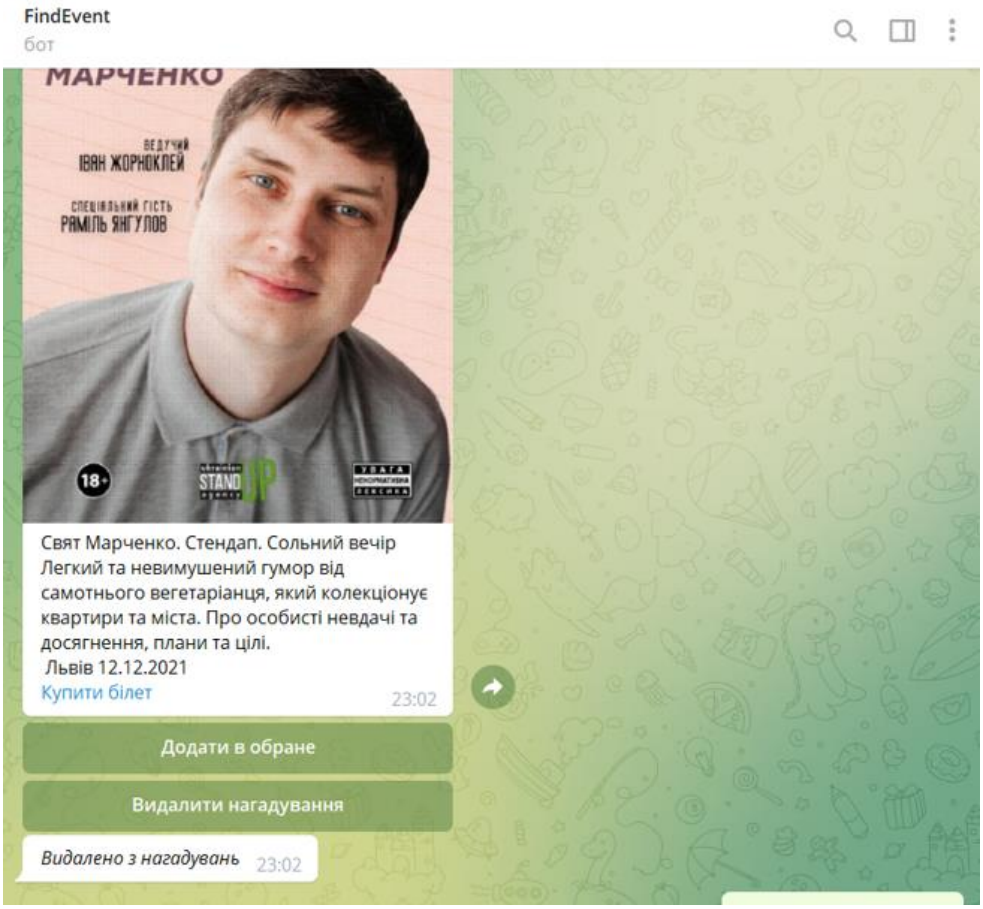


Рисунок 4.15 – Видалення заходу із списку «Нагадування»

Крім цього користувачі бота мають змогу додавання власних заходів. Для цього необхідно вибрати пункт «Додати захід» в головному меню. Додавання заходу користувачем здійснюється в декілька етапів:

- введення назви заходу;
- введення опису заходу;
- введення категорії заходу;
- введення дати проведення заходу;
- введення міста проведення заходу;
- введення посилання на захід;
- додавання посилання на фото.

Введення назви заходу зображено на рисунку 4.16

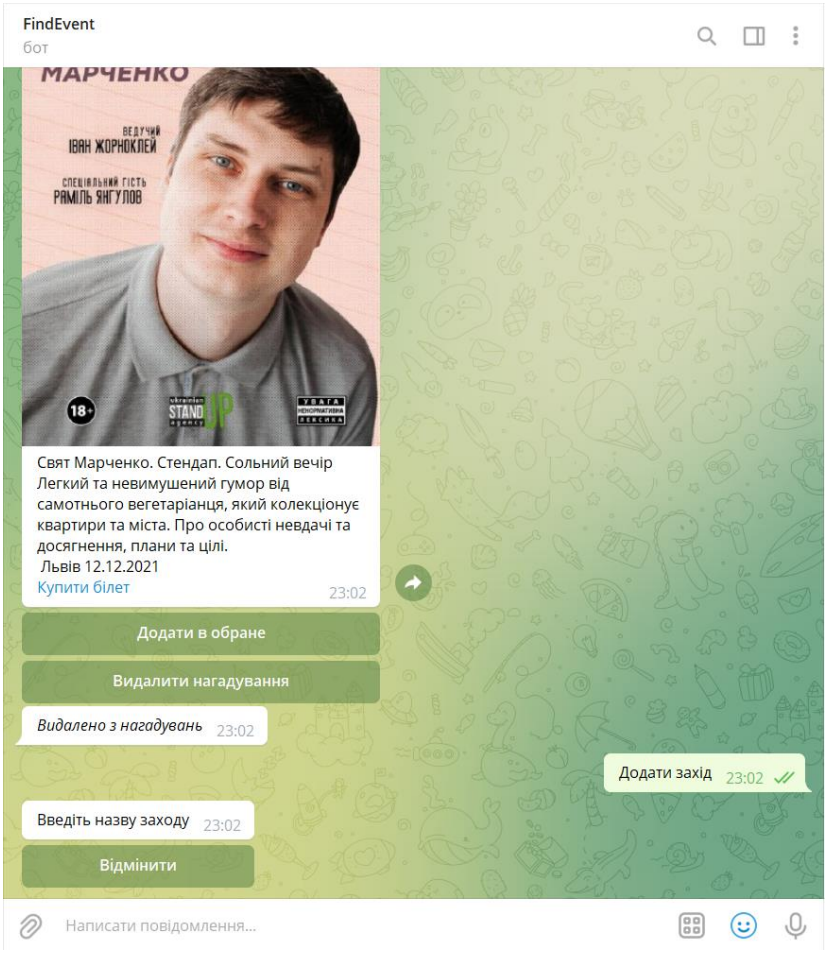


Рисунок 4.16 – Процес додавання заходу

Після цього користувач повинен підтвердити додавання заходу. Даний процес зображено на рисунку 4.17

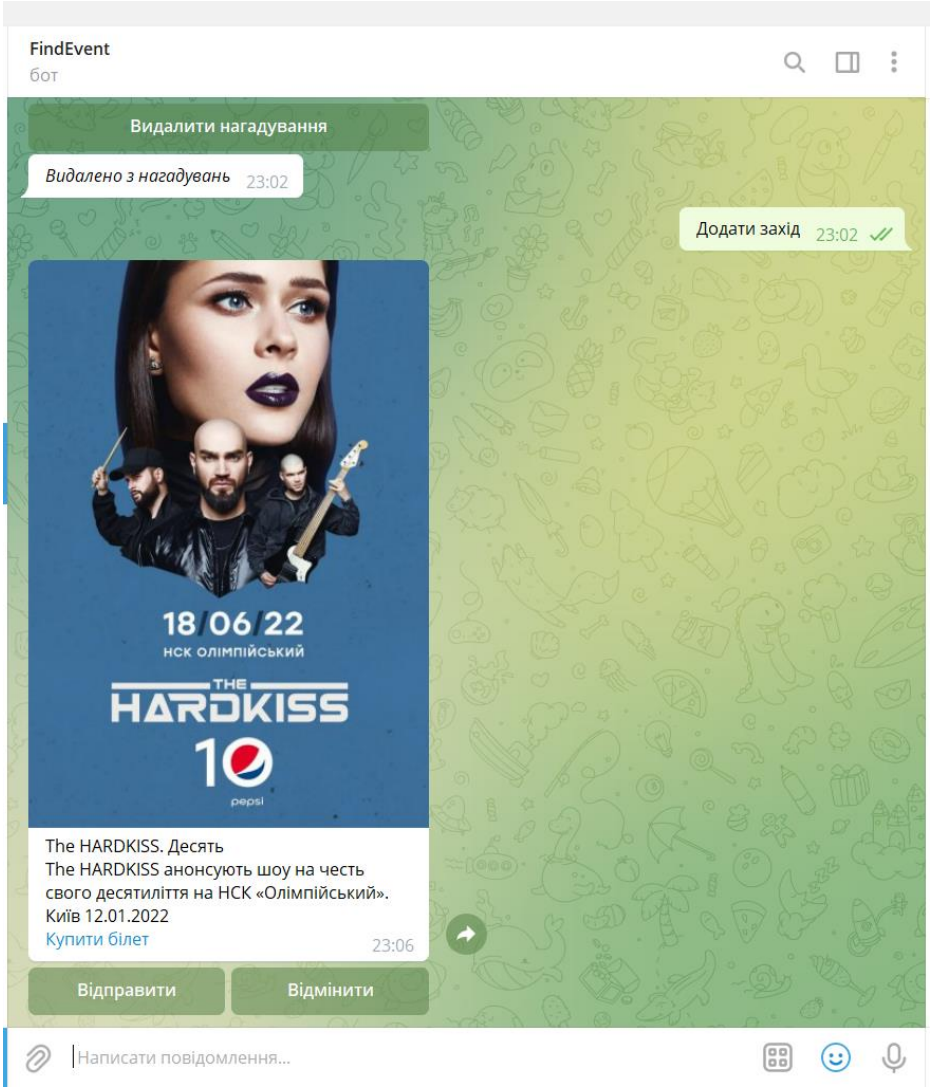


Рисунок 4.17 – Додавання заходу користувачем

Після того як користувач відправив захід він надсилається адміністратору на схвалення. Схвалення заходу зображено на рисунку 4.18.

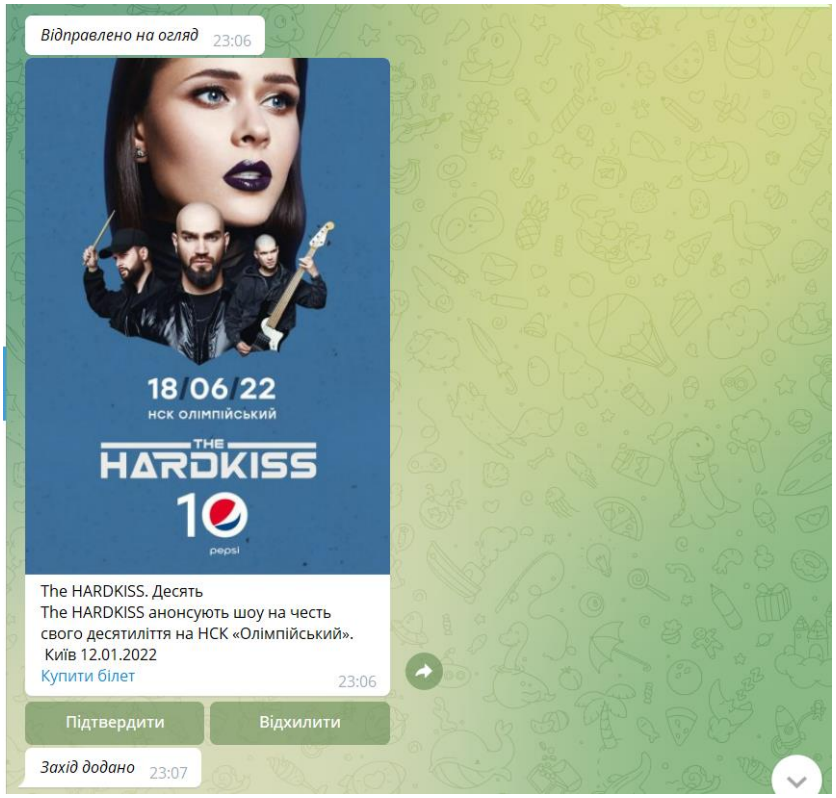


Рисунок 4.18 – Підтвердження додавання заходу адміністратором

Також користувачі мають змогу зміни мови інтерфейсу користувача, обравши пункт «Змінити мову» в головному меню. Зміна мови інтерфейсу користувача зображено на рисунку 4.19

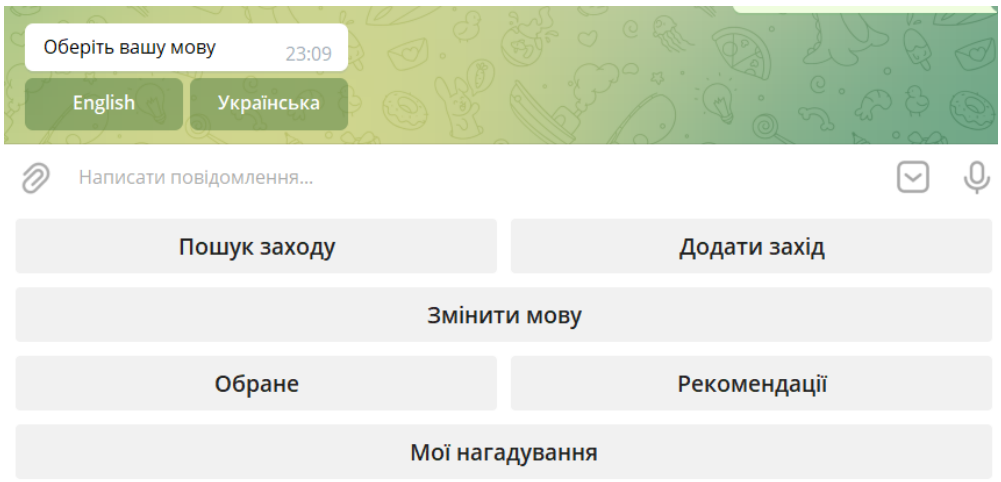


Рисунок 4.19 – Зміна мови інтерфейсу користувача

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи магістра, було реалізовано чат-бот. За мету роботи було визначено розробку чат-бота для месенджеру Telegram, який допоможе користувачам у пошуку найближчих заходів у містах України.

Для досягнення поставленої мети було визначено такі задачі: аналіз предметної області та вимог, порівняння вже існуючих рішень-аналогів, вибір технологій і середовища розробки, розробка чат-боту, розробка супровідної документації .

Також у ході дослідження було обгрунтовано підставу для розробки програмного рішення, було здійснено деталізацію мети та задач проекту, визначено функціональні можливості чат-бота. Для розробки чат-бота було обрано такі технологій та засоби як Python, в якості мови програмування, Sublime Text як середовище розробки.

На етапі проектування було побудовано діаграми, які показують функціональні можливості чат-боту, а саме було побудовано діаграми процесу забезпечення пошуку заходів по містах України у нотаціях IDEF0, діаграму варіантів використання чат бота та модель бази даних.

На останньому етапі було визначено архітектуру чат-бота, здійснено реалізацію чат-бота, а також було продемонстровано основний функціонал бота.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Что такое Телеграмм (Telegram), зачем он мне и как им пользоваться? . [Електронний ресурс]. – Точка доступу: URL: https://protelegram.ru/telegram_faq/
2. Most popular global mobile messenger apps as of July 2021, based on number of monthly active users [Електронний ресурс]. – Точка доступу: URL: https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/
3. Чат-боты - кто они и что умеют? [Електронний ресурс]. – Точка доступу: URL: https://efsol.ru/articles/messendzhery-i-chat-boty-dlya-biznesa-dostavki.html
4. Как использовать чат-боты [Електронний ресурс]. – Точка доступу: URL: https://vc.ru/services/93850-kak-ispolzovat-chat-boty-v-biznese-5-idey-i-5-keysov
5. Как создать Telegram-бота с помощью библиотеки python-telegram-bot [Електронний ресурс]. – Точка доступу: URL: https://highload.today/kak-sozdat-telegram-bot-na-python-poshagovoe-rukovodstvo/
6. Обзор технологий создания чат-ботов [Електронний ресурс]. – Точка доступу: URL: http://masters.donntu.org/2018/fknt/overchenko/library/article4.htm
7. How do chatbots really work? [Електронний ресурс]. – Точка доступу: URL: https://www.itechart.com/blog/how-do-chatbots-really-work/
8. Підходи до створення інтелектуальних чат-ботів [Електронний ресурс]. – Точка доступу: URL: http://www.hups.mil.gov.ua/periodic-app/article/19330
9. Афіша Києва у твоєму смартфоні: з'явився новий Телеграм-бот для пошуку івентів [Електронний ресурс]. – Точка доступу: URL: https://kyivmaps.com/ua/blog/afisa-kieva-v-tvoem-smartfone-poavilsa-novyy-telegramm-bot-dla-poiska-iventov
10. Telegram-бот MoeMisto.ua - твій помічник для вибору дозвілля у Києві! [Електронний ресурс]. – Точка доступу: URL: https://moemisto.ua/kiev/blog/telegram-bot-moemistoua---tviy-pomichnik-dlya-viboru-dozvillya-u-kievi-295.html

11. Ontour bot в Telegram: как быстро найти концерт [Електронний ресурс]. – Точка доступу: URL: https://highload.today/lyublyu-kontserty-i-hotel-bystro-najti-gde-igraet-lyubimaya-gruppa-kak-ya-sozdal-telegram-bot-dlya-poiska-vystuplenij/

12. Язык программирования Python [Електронний ресурс]. – Точка доступу: URL: https://web-creator.ru/articles/python

13. Documentation Sublime Text [Електронний ресурс]. – Точка доступу: URL: https://www.sublimetext.com/docs/completions.html

14. SQLite Documentation [Електронний ресурс]. – Точка доступу: URL: https://www.sqlite.org/docs.html

15. Как создать Telegram-бота с помощью библиотеки python-telegram-bot [Електронний ресурс]. – Точка доступу: URL: https://highload.today/kak-sozdat-telegram-bot-na-python-poshagovoe-rukovodstvo/

16. UML для бізнес-моделювання: для чого потрібні діаграми процесів Architecture [Електронний ресурс]. – Точка доступу: URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html

17. Основные методологии обследования организаций. Стандарт IDEF0. [Електронний ресурс]. – Точка доступу: URL: https://www.cfin.ru/vernikov/idef/idef0.shtml

18. Использование диаграммы вариантов использования UML при проектировании программного обеспечения [Електронний ресурс]. – Точка доступу: URL: https://habr.com/ru/post/566218/

19. Основы UML — диаграммы использования [Електронний ресурс]. – Точка доступу: URL: https://pro-prof.com/archives/2594

20. Что такое ER-диаграмма и как ее создать? [Електронний ресурс]. – Точка доступу: URL: https://www.lucidchart.com/pages/ru/erd-диаграмма

21. Поняття ER-моделі. Поняття сутності (entity). Атрибути. Види атрибутів [Електронний ресурс]. – Точка доступу: URL:

22. Understanding The Conversational Chatbot Architecture [Електронний ресурс]. – Точка доступу: URL: https://blog.vsoftconsulting.com/blog/understanding-the-architecture-of-conversational-chatbot

23. Building Your First Telegram Bot: A Step by Step Guide [Електронний ресурс]. – Точка доступу: URL: https://www.toptal.com/python/telegram-bot-tutorial-python

24. Telegram Bot API [Електронний ресурс]. – Точка доступу: URL: https://core.telegram.org/bots/api

25. Python Telegram Bot's documentation [Електронний ресурс]. – Точка доступу: URL: https://python-telegram-bot.readthedocs.io/en/stable/

26. Bots: An introduction for developers [Електронний ресурс]. – Точка доступу: URL: https://core.telegram.org/bots

27. Instructions for SQLiteStudio [Електронний ресурс]. – Точка доступу: URL: http://www.cse.hut.fi/fi/opinnot/CSE-A1200/K2016/harjoitukset/studio.html

28. Цели по SMART: подробный обзор [Електронний ресурс]. – Точка доступу: URL: http://powerbranding.ru/marketing-strategy/smart-celi/

29. Что такое WBS проекта, и зачем она нужна [Електронний ресурс]. – Точка доступу: URL: https://upravlenie-proektami.ru/chto-takoe-wbs-proekta-i-zachem-ona-nuzhna

30. What is a Work Breakdown Structure? [Електронний ресурс]. – Точка доступу: URL: https://www.workbreakdownstructure.com/

31. Организационная структура компании [Електронний ресурс]. – Точка доступу: URL: https://studfile.net/preview/9650542/page:20/

32. Что такое диаграмма Ганта? [Електронний ресурс]. – Точка доступу: URL: https://www.atlassian.com/ru/agile/project-management/gantt-chart

ДОДАТОК А. ПЛАНУВАННЯ РОБІТ

Ідентифікація мети ІТ-проекту

Метою проекту є розробка бота для пошуку найближчих заходів у містах України. Для деталізації мети зручно використовувати методологію SMART. SMART – це метод опису мети, що включає: конкретність, вимірність, досяжність, реалістичність і визначеність за термінами [28]. Система постановки smart-цілей дозволяє узагальнити всю наявну інформацію, встановити прийнятні терміни роботи, визначити кількість необхідних ресурсів, надати всім учасникам процесу ясні, точні, конкретні завдання.

Деталізації мети методом SMART розміщені у таблиці А.1.

Таблиця А.1 – Деталізація мети інформаційної системи методом SMART

Specific (конкретна)	Розробка бота для пошуку заходів по містах України
Measurable (вимірювана)	Оскільки, проект не є комерційним, то результатом роботи є оцінка проекту замовником.
Achievable (досяжна)	Мету реально досягнути, розробник проекту володіє необхідними навичками з програмування.
Relevant (реалістична)	Всі необхідні технічні та програмні засоби є у наявності. Достатньо кваліфіковані розробники для виконання поставлених задач
Time-framed (обмежена у часі)	Обмеженість в часі зумовлена рішенням замовника, проект повинен бути розробленим в оговорені терміни. Проект необхідно виконати згідно з календарним планом.

Планування змісту структури робіт IT-проекту

Планування являє собою процес розробки та прийняття цільових установок кількісного та якісного характеру та визначення шляхів найбільш ефективного їх досягнення. Ці установки, що розробляються найчастіше у вигляді дерева цілей, характеризують бажане майбутнє і по можливості чисельно виражаються набором показників, ключових для рівня управління.

Основним інструментом для планування змісту структури робіт служить WBS-діаграма. WBS проекту – це розбиття проекту на конкретні результати, які мають бути досягнуті для досягнення цілей проекту [29].

WBS забезпечує виявлення робіт, необхідних для досягнення цілей проекту. За такого підходу проект визначається в термінах ієрархічно взаємопов’язаних орієнтованих на результат елементів (пакетів робіт — комплексів робіт, згрупованих по заданим критеріям) [30]. Кожен наступний рівень декомпозиції забезпечує послідовну деталізацію змісту проекту, що дозволяє проводити оцінку виконаних обсягів робіт, бюджету проекту та виконання за строками. Запропонований підхід декомпозиції робіт формує необхідну основу для визначення вимірних показників (трудомісткості, вартості), а також дозволяє з високим ступенем достовірності говорити про те, що цілі, пов’язані з цим пакетом робіт, можуть і будуть досягнуті. Ієрархічна структура робіт зображена на рисунку А.1

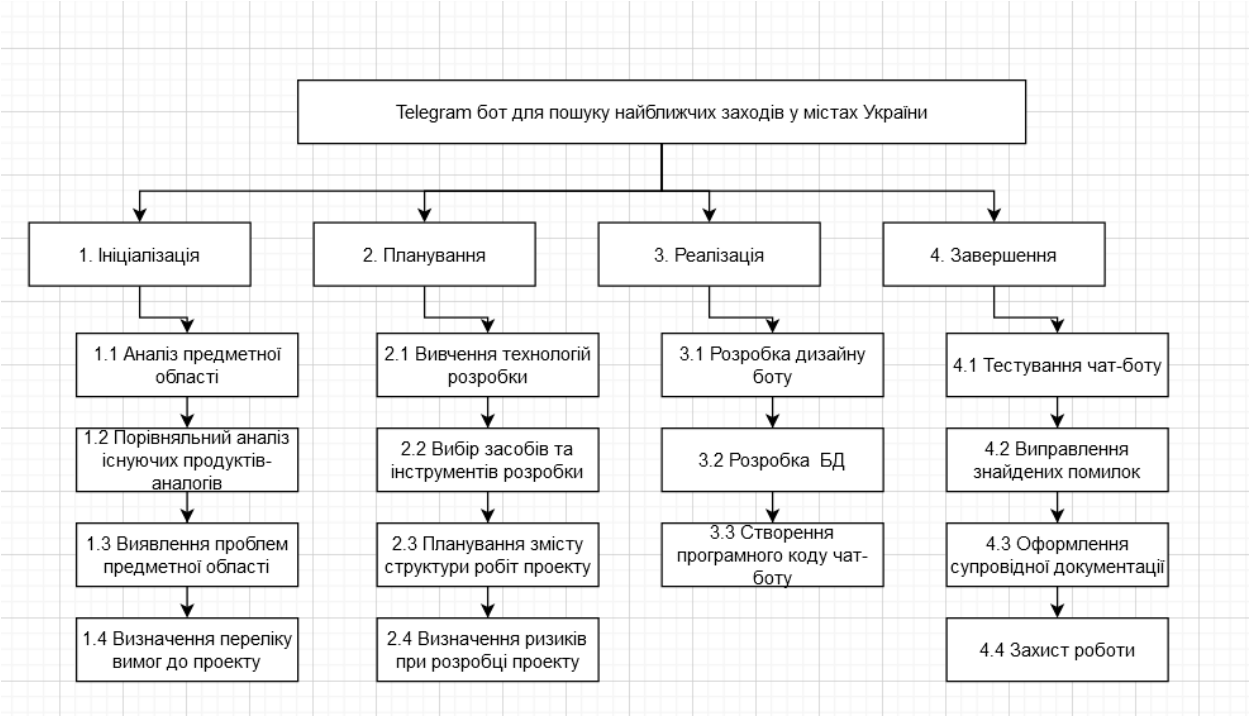


Рисунок А.1 – Ієрархічна структура робіт (WBS)

Після побудови WBS-діаграми було розроблено організаційну структуру виконавців (OBS), що за своєю структурою відповідає WBS-діаграмі, але її елементами є виконавці робіт [31]. Організаційна структура проекту (OBS) є ієрархічною структурою управління проектом і показує відносини між учасниками проекту. OBS діаграма зображена на рисунку А.2.

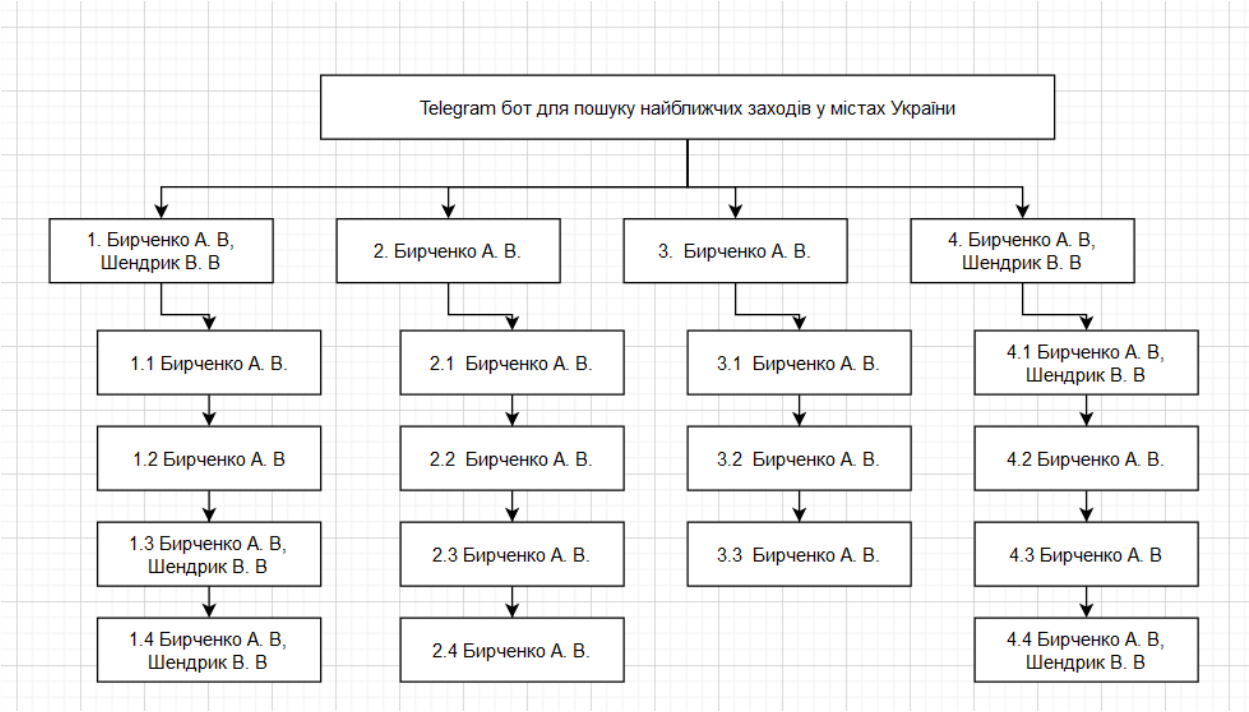
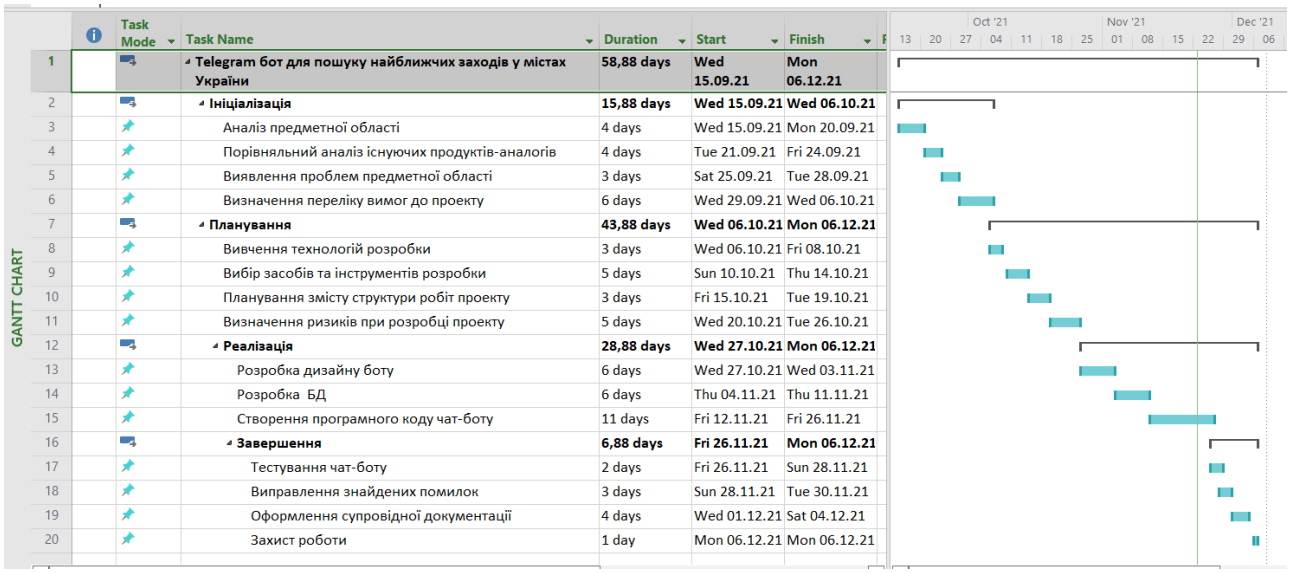


Рисунок А.2 – OBS структура системи

Побудова календарного графіку виконання ІТ-проекту

Далі було розроблено календарний план виконання дипломного проекту. Найпоширенішим форматом календарного плану є діаграма Ганта. Діаграма Ганта – це інструмент управління проектами, що ілюструє план проекту. Зазвичай вона складається із двох частин: у лівій частині наведено список завдань, а у правій — тимчасова шкала зі смугами, що зображають роботу. Діаграма Ганта також може включати дати початку та завершення завдань, контрольні точки, залежності між завданнями та виконавцями [32]. Діаграма Ганта та список робіт зображені на рисунку А.3.



	Task Mode	Task Name	Duration	Start	Finish
1		**Telegram бот для пошуку найближчих заходів у містах України**	**58,88 days**	**Wed 15.09.21**	**Mon 06.12.21**
2		**Ініціалізація**	**15,88 days**	**Wed 15.09.21**	**Wed 06.10.21**
3		Аналіз предметної області	4 days	Wed 15.09.21	Mon 20.09.21
4		Порівняльний аналіз існуючих продуктів-аналогів	4 days	Tue 21.09.21	Fri 24.09.21
5		Виявлення проблем предметної області	3 days	Sat 25.09.21	Tue 28.09.21
6		Визначення переліку вимог до проекту	6 days	Wed 29.09.21	Wed 06.10.21
7		**Планування**	**43,88 days**	**Wed 06.10.21**	**Mon 06.12.21**
8		Вивчення технологій розробки	3 days	Wed 06.10.21	Fri 08.10.21
9		Вибір засобів та інструментів розробки	5 days	Sun 10.10.21	Thu 14.10.21
10		Планування змісту структури робіт проекту	3 days	Fri 15.10.21	Tue 19.10.21
11		Визначення ризиків при розробці проекту	5 days	Wed 20.10.21	Tue 26.10.21
12		**Реалізація**	**28,88 days**	**Wed 27.10.21**	**Mon 06.12.21**
13		Розробка дизайну боту	6 days	Wed 27.10.21	Wed 03.11.21
14		Розробка БД	6 days	Thu 04.11.21	Thu 11.11.21
15		Створення програмного коду чат-боту	11 days	Fri 12.11.21	Fri 26.11.21
16		**Завершення**	**6,88 days**	**Fri 26.11.21**	**Mon 06.12.21**
17		Тестування чат-боту	2 days	Fri 26.11.21	Sun 28.11.21
18		Виправлення знайдених помилок	3 days	Sun 28.11.21	Tue 30.11.21
19		Оформлення супровідної документації	4 days	Wed 01.12.21	Sat 04.12.21
20		Захист роботи	1 day	Mon 06.12.21	Mon 06.12.21

Рисунок А.3 – Діаграма Ганта

Планування ризиків проекту

Далі було проведено аналіз ризиків проекту.

Ризик проекту – це певна подія або умова, яка у разі виникнення має позитивний або негативний вплив щонайменше на одну з цілей проекту, наприклад, терміни, вартість, утримання або якість.

Основною стратегією управління ризиками вважається їх мінімізація. Управління ризиками містить у собі планування ризиків, реалізацію дій по реагуванню, контроль, роботу з новими ризиками, або ризиками, що залишилися.

В таблицях А.3 – А.5 зображено реєстрацію ризиків.

Таблиця А.3 – Risk Register

	Risk	Impact	Probability	Mitigation	RV	How to solve
1	Часта зміна вимог	3	3		Виправданий	Погодити всі питання з замовником та внести необхідні зміни і поправки.
2	Неоптимальний розподіл часу	5	3		Недопустимий	Змінити порядок пріоритетів робіт. Знайти способи оптимізації із уже існуючою розтановкою.
3	Проблеми в комунікації з замовником	1	2		Прийнятний	Вияснити, що саме стало причиною непорозуміння та обговорити її
4	Не оптимальне планування бюджету	5	3		Недопустимий	Компенсувати витрати за рахунок інших етапів проекту
5	Некоректна робота апаратного забезпечення	5	2		Недопустимий	Проводити регулярну перевірку апаратного забезпечення

Відповідно до таблиці ризики поділяються на:

– зелений колір – прийнятні ризики;

– жовтий колір – виправданні ризики;

– червоний колір – недопустимі ризики.

Оцінка ризиків здійснюється за допомогою двох критеріїв: ступінь впливу (impact) та ймовірність виникнення (probability).

Таблиця А.4 – Шкала оцінювання ймовірності виникнення та впливу ризику на виконання проекту

Оцінка	Ймовірність виникнення	Ступінь впливу
1	very low	negligible
2	Low	minor
3	Medium	moderate
4	High	serious
5	very high	critical

Таблиця А.5 – Probability/Impact matrix

Probability	Probability/Impact matrix				
5 very high					
4 high					
3 medium			1		2, 4
2 low	3			6	5
1 very low					
Impact	1 negligible	2 minor	3 moderate	4 serious	5 critical

ДОДАТОК Б. ПРОГРАМНИЙ КОД

Файл створення таблиць бази даних

```
PRAGMA foreign_keys = off;
BEGIN TRANSACTION;

-- Таблиця: category
DROP TABLE IF EXISTS category;

CREATE TABLE category (
    category_id INT,
    name        VARCHAR (255),
    event_id    INT
);


-- Таблиця: events
DROP TABLE IF EXISTS events;

CREATE TABLE events (
    event_id        INTEGER       PRIMARY KEY
                                  UNIQUE,
    name            VARCHAR (255),
    photo           VARCHAR (255),
    city            VARCHAR (255),
    date            DATE,
    notification_id INTEGER,
    ecity           VARCHAR (255),
    link            VARCHAR (255)
);


-- Таблиця: favourite
DROP TABLE IF EXISTS favourite;

CREATE TABLE favourite (
    favourite_id INTEGER PRIMARY KEY
                         UNIQUE,
    event_id     INTEGER REFERENCES events (event_id) ON DELETE NO ACTION
                                                      ON UPDATE NO ACTION
                                                      MATCH SIMPLE,
    user_id      INTEGER REFERENCES users (id) ON DELETE NO ACTION
                                               ON UPDATE NO ACTION
                                               MATCH SIMPLE
);


-- Таблиця: notifications
DROP TABLE IF EXISTS notifications;

CREATE TABLE notifications (
    notification_id INTEGER PRIMARY KEY,
    date            DATE,
    time            TIME
);
```

```
-- Таблиця: users
DROP TABLE IF EXISTS users;

CREATE TABLE users (
    id              INTEGER       PRIMARY KEY,
    name            VARCHAR (255),
    surname         VARCHAR (255),
    username        VARCHAR (255),
    notification_id INTEGER       REFERENCES notifications (notification_id) ON
DELETE NO ACTION
                                                                               ON
UPDATE NO ACTION
                                                                               MATCH
SIMPLE,
    event_id        INTEGER       REFERENCES events (event_id) ON DELETE NO ACTION
                                                               ON UPDATE NO ACTION
                                                               MATCH SIMPLE
);


COMMIT TRANSACTION;
PRAGMA foreign_keys = on;
```

Файл keyboards.ry

```
from telebot import types


def menu():

      markup= types.ReplyKeyboardMarkup(one_time_keyboard=False,
resize_keyboard=True)
      btn1 = types.KeyboardButton("Пошук заходу")
      btn2 = types.KeyboardButton("Змінити мову")
      btn3 = types.KeyboardButton("Обране")
      btn4 = types.KeyboardButton("Додати захід")
      btn5 = types.KeyboardButton("Мої нагадування")
      btn6 = types.KeyboardButton("Рекомендації")

      markup.row(btn1, btn4)
      markup.row(btn2)
      markup.row(btn3, btn6)
      markup.row(btn5)

      return markup
def citykb():
      markup = types.InlineKeyboardMarkup()
      btn1 = types.InlineKeyboardButton(text="Київ", callback_data="city Київ")
      btn2 = types.InlineKeyboardButton(text="Суми", callback_data="city Суми")
      btn3 = types.InlineKeyboardButton(text="Одесса", callback_data="city Одеса")
      btn4 = types.InlineKeyboardButton(text="Харьків", callback_data="city
Харків")
      btn5 = types.InlineKeyboardButton(text="Полтава", callback_data="city
Полтава")
      btn6 = types.InlineKeyboardButton(text="Львів", callback_data="city Львів")
      btn7 = types.InlineKeyboardButton(text="Дніпро", callback_data="city
Дніпро")
      btn8 = types.InlineKeyboardButton(text="Житомир", callback_data="city
Житомир")
```

```
      btn9 = types.InlineKeyboardButton(text="Інше", callback_data="other")

      markup.row(btn1, btn2, btn3)
      markup.row(btn4, btn5, btn6)
      markup.row(btn7, btn8, btn9)

      return markup

def otmenac():
      markup = types.InlineKeyboardMarkup()
      btn1 = types.InlineKeyboardButton(text="Відмінити", callback_data="otmenac")

      markup.row(btn1)

      return markup

def otmenap():
      markup = types.InlineKeyboardMarkup()
      btn1 = types.InlineKeyboardButton(text="Відмінити", callback_data="otmenap")

      markup.row(btn1)

      return markup

def otmena():
      markup = types.InlineKeyboardMarkup()
      btn1 = types.InlineKeyboardButton(text="Відмінити", callback_data="otmena")

      markup.row(btn1)

      return markup

def poisk():
      markup = types.InlineKeyboardMarkup()
      btn1 = types.InlineKeyboardButton(text="Пошук за датою",
callback_data="poisk date")
      btn2 = types.InlineKeyboardButton(text="Пошук за місцем",
callback_data="poisk city")
      btn3 = types.InlineKeyboardButton(text="Пошук за категорією",
callback_data="poisk category")
      btn4 = types.InlineKeyboardButton(text="Пошук за запитом",
callback_data="poisk zapyt")
      btn5 = types.InlineKeyboardButton(text="Змінити місто",
callback_data="changecity")

      markup.row(btn1, btn2)
      markup.row(btn3, btn4)
      markup.row(btn5)

      return markup

def category():
      markup = types.InlineKeyboardMarkup()
      btn1 = types.InlineKeyboardButton(text="Концерт", callback_data="category
Концерт")
      btn2 = types.InlineKeyboardButton(text="Театр", callback_data="category
Театр")
      btn3 = types.InlineKeyboardButton(text="Виставка", callback_data="category
Виставка")
      btn4 = types.InlineKeyboardButton(text="Спорт", callback_data="category
Спорт")
```

```
      btn5 = types.InlineKeyboardButton(text="Вечірка", callback_data="category
Вечірка")
      btn6 = types.InlineKeyboardButton(text="Гумор", callback_data="category
Гумор")
      btn7 = types.InlineKeyboardButton(text="Цирк", callback_data="category
Цирк")
      btn8 = types.InlineKeyboardButton(text="Екскурсія", callback_data="category
Екскурсія")
      btn9 = types.InlineKeyboardButton(text="Фестиваль", callback_data="category
Фестиваль")

      markup.row(btn1, btn2, btn3)
      markup.row(btn4, btn5, btn6)
      markup.row(btn7, btn8, btn9)

      return markup

def eventkb(id):
      markup = types.InlineKeyboardMarkup()
      btn1 = types.InlineKeyboardButton(text="Додати в обране",
callback_data=f"izbr {id}")
      btn2 = types.InlineKeyboardButton(text="Поставити нагадування",
callback_data=f"nagad {id}")

      markup.row(btn1)
      markup.row(btn2)
      return markup

def eventizb(id):
      markup = types.InlineKeyboardMarkup()
      btn1 = types.InlineKeyboardButton(text="Видалити з обраного",
callback_data=f"delizbr {id}")
      btn2 = types.InlineKeyboardButton(text="Поставити нагадування",
callback_data=f"nagad {id}")

      markup.row(btn1)
      markup.row(btn2)
      return markup

def eventnagad(id):
      markup = types.InlineKeyboardMarkup()
      btn1 = types.InlineKeyboardButton(text="Додати в обране",
callback_data=f"izbr {id}")
      btn2 = types.InlineKeyboardButton(text="Видалити нагадування",
callback_data=f"delnagad {id}")

      markup.row(btn1)
      markup.row(btn2)
      return markup



def lang():
      markup = types.InlineKeyboardMarkup()
      btn1 = types.InlineKeyboardButton(text="English", callback_data=f"eng")
      btn2 = types.InlineKeyboardButton(text="Українська", callback_data=f"ukr")

      markup.row(btn1, btn2)

      return markup
```

Файл bot.py

```
#!/usr/bin/python
# coding: utf-8
import sqlite3
import telebot
import random
from keyboards import *

database = "database.db"
statebase = "state.db"

bot = telebot.TeleBot("2122531068:AAHqFT1oeoGH1pQpGoY9gPsvpAtlRBnGq4M")

admin = 394134197

def statushandler(id, value):
      con = sqlite3.connect(statebase)
      cur = con.cursor()
      cur.execute("SELECT id FROM users WHERE id = ?",(id,))
      user = cur.fetchone()
      if user != None:
            con = sqlite3.connect(statebase)
            cur = con.cursor()
            cur.execute("UPDATE users SET state = ? WHERE id = ?",(value, id,))
            con.commit()
      else:
            con = sqlite3.connect(statebase)
            cur = con.cursor()
            cur.execute("INSERT INTO users (id, state) VALUES "f"({id}, {value})")
            con.commit()


@bot.message_handler(commands=['start'])
def welcome(message):
      con = sqlite3.connect(database)
      cur = con.cursor()
      cur.execute("SELECT id FROM users WHERE id = ?",(message.chat.id,))
      user = cur.fetchone()
      if user == None:
            con = sqlite3.connect(database)
            cur = con.cursor()
            cur.execute(f"INSERT INTO users (id)"f"VALUES ({message.chat.id})")
            con.commit()
      bot.send_message(message.chat.id,"<i>Привіт! Я ваш бот FindEvent! Я допоможу
вам знайти місце де можна провести вільний час у рідному місті. Оберіть ваше
місто</i>", parse_mode = "HTML", reply_markup = citykb())


@bot.message_handler(content_types=['text'])
def menuhandler(message):

      if message.text == "Обране":
            con = sqlite3.connect(database)
            cur = con.cursor()
            cur.execute("SELECT id_favourite FROM events WHERE id_favourite =
?",(message.chat.id,))
            izb = cur.fetchone()[0]

            if len(izb) >= 1:
                  izb = izb.split(" ", maxsplit = int((len(izb)+1)/2))
```

```
                  for izb in izb:
                        con = sqlite3.connect(database)
                        cur = con.cursor()
                        cur.execute("SELECT * FROM events WHERE id_favourite =
?",(izb,))
                        result = cur.fetchall()[0]
                        bot.send_photo(message.chat.id,f"{result[7]}",
f"{result[1]}\n{result[2]}\n {result[4]} {result[5]}\n<a href='{result[3]}'>Купити
білет</a>", parse_mode = "HTML", reply_markup = eventizb(result[0]))
            else:
                  bot.send_message(message.chat.id, "Нічого не додано", parse_mode
= "HTML")

      if message.text == "Пошук заходу":
            bot.send_message(message.chat.id,"<i>Оберіть ваше місто</i>",
parse_mode = "HTML", reply_markup = citykb())

      if message.text == "Мої нагадування":
            con = sqlite3.connect(database)
            cur = con.cursor()
            cur.execute("SELECT id_notification FROM notification WHERE
id_notification  = ?",(message.chat.id,))
            nagad = cur.fetchone()[0]

            if len(nagad) >= 1:
                  nagad = nagad.split(" ", maxsplit = int((laen(nagad)+1)/2))
                  for nagad in nagad:
                        con = sqlite3.connect(database)
                        cur = con.cursor()
                        cur.execute("SELECT * FROM events WHERE id = ?",(nagad,))
                        result = cur.fetchall()[0]
                        bot.send_photo(message.chat.id,f"{result[7]}",
f"{result[1]}\n{result[2]}\n {result[4]} {result[5]}\n<a href='{result[3]}'>Купити
білет</a>", parse_mode = "HTML", reply_markup = eventnagad(result[0]))
            else:
                  bot.send_message(message.chat.id, "Нічого не додано", parse_mode
= "HTML")

      if message.text == "Змінити мову":
            bot.send_message(message.chat.id, "Оберіть вашу мову", parse_mode =
"HTML", reply_markup = lang())

      if message.text == "Рекомендації":
            bot.send_message(message.chat.id, "Список ваших рекомендацій",
parse_mode = "HTML")
            con = sqlite3.connect(database)
            cur = con.cursor()
            cur.execute("SELECT id_events FROM users WHERE id =
?",(message.chat.id,))
            eventsid = cur.fetchone()[0]
            eventsid = eventsid.split(" ", maxsplit = int((len(eventsid)+1)/2))

            for eventsid in eventsid:
                  print(eventsid)
                  con = sqlite3.connect(database)
                  cur = con.cursor()
                  cur.execute("SELECT city, category FROM events WHERE id =
?",(eventsid,))
                  result = cur.fetchall()[0]
                  print(result)
                  con = sqlite3.connect(database)
```

```
                  cur = con.cursor()
                  cur.execute("SELECT * FROM events WHERE city = ? or category =
?",(result[0], result[1],))
                  result = cur.fetchall()[0]
                  print(result)
                  #bot.send_photo(message.chat.id,f"{result[7]}",
f"{result[1]}\n{result[2]}\n {result[4]} {result[5]}\n<a href='{result[3]}'>Купити
білет</a>", parse_mode = "HTML", reply_markup = eventkb(result[0]))
```

```
      if message.text == "Додати захід":
            msg = bot.send_message(message.chat.id, "Введіть назву заходу",
parse_mode = "HTML", reply_markup = otmena())
            stat = random.choice(range(0, 999999))
            statushandler(message.chat.id, stat)
            bot.register_next_step_handler(message, zahid, msg, stat)
```

```
@bot.callback_query_handler(func=lambda call: True)
def answer(call):
      print(call.data)
      if call.data == "other":
            msg = bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.message_id,text="<i>Будьласка введіть ваше місто</i>",
parse_mode = "HTML", reply_markup=otmenac())
            stat = random.choice(range(0, 999999))
            statushandler(call.message.chat.id, stat)
            bot.register_next_step_handler(call.message, cityhandler, msg ,stat)

      if call.data == "otmenac":
            bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.message_id,text="<i>Привіт! Я ваш бот FindEvent! Я
допоможу вам знайти місце де можна провести вільний час у рідному місті. Оберіть
ваше місто</i>", parse_mode = "HTML", reply_markup = citykb())
            statushandler(call.message.chat.id, 0)

      if call.data == "otmenap":
            bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.message_id,text="<i>Оберіть яким чином ви хочете
здійснювати подальший пошук</i>", parse_mode = "HTML", reply_markup = poisk())
            statushandler(call.message.chat.id, 0)

      if call.data == "otmena":
            bot.delete_message(call.message.chat.id, call.message.message_id)
            bot.send_message(call.message.chat.id, "<i>Відмінено</i>", parse_mode
= "HTML")
            statushandler(call.message.chat.id, 0)

      if call.data[:4] == "city":
                  con = sqlite3.connect(database)
                  cur = con.cursor()
```

```
                  cur.execute("UPDATE users SET city = ? WHERE id =
?",(call.data[5:], call.message.chat.id,))
                  con.commit()
                  bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.message_id,text="<i>Оберіть яким чином ви хочете
здійснювати подальший пошук</i>", parse_mode = "HTML", reply_markup = poisk())

      if call.data == "changecity":
            bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.message_id,text="<i>Виберіть ваше місто</i>", parse_mode =
"HTML", reply_markup=citykb())

      if call.data[:5] == "poisk":
            if call.data[6:] == "city":
                  msg = bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.message_id,text="<i>Введіть ваше місто</i>", parse_mode =
"HTML", reply_markup=otmenap())
                  stat = random.choice(range(0, 999999))
                  statushandler(call.message.chat.id, stat)
                  bot.register_next_step_handler(call.message, poiskhandler, msg,
stat, call.data[6:])

            if call.data[6:] == "date":
                  msg = bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.message_id,text="<i>Введіть дату в форматі
00.00.0000</i>", parse_mode = "HTML", reply_markup=otmenap())
                  stat = random.choice(range(0, 999999))
                  statushandler(call.message.chat.id, stat)
                  bot.register_next_step_handler(call.message, poiskhandler, msg,
stat, call.data[6:])

            if call.data[6:] == "zapyt":
                  msg = bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.message_id,text="<i>Введіть ваш запит</i>", parse_mode =
"HTML", reply_markup=otmenap())
                  stat = random.choice(range(0, 999999))
                  statushandler(call.message.chat.id, stat)
                  bot.register_next_step_handler(call.message, poiskhandler, msg,
stat, call.data[6:])

            if call.data[6:] == "category":
                  bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.message_id,text="<i>Виберіть категорію</i>", parse_mode =
"HTML", reply_markup=category())

      if call.data[:8] == "category":
            cat = call.data[9:]
            con = sqlite3.connect(database)
            cur = con.cursor()
            cur.execute("SELECT city FROM events WHERE id = ?
",(call.message.chat.id,))
            city = cur.fetchone()[0]
            con = sqlite3.connect(database)
            cur = con.cursor()
            cur.execute("SELECT * FROM events WHERE category = ? AND city = ?
",(cat, city,))
            result = cur.fetchall()


            if result != None:
```

```
                  bot.send_message(call.message.chat.id,"<i>Результати
пошуку</i>", parse_mode = "HTML", reply_markup = menu())
                  for result in result:
                        bot.send_photo(call.message.chat.id,f"{result[7]}",
f"{result[1]}\n{result[2]}\n {result[4]} {result[5]}\n<a href='{result[3]}'>Купити
білет</a>", parse_mode = "HTML", reply_markup = eventkb(result[0]))
            else:
                  bot.edit_message_text(chat_id=msg.chat.id,
message_id=msg.message_id,text="<i>Нічого не знайдено</i>", parse_mode = "HTML")

      if call.data[:4] == "izbr":
            con = sqlite3.connect(database)
            cur = con.cursor()
            cur.execute("SELECT id_notification FROM users WHERE id = ?
",(call.message.chat.id,))
            result = cur.fetchone()[0]

            if len(result) != 0:

                  if call.data[5:] in result:
                        bot.send_message(call.message.chat.id,"<i>Цю подію вже
додано</i>", parse_mode = "HTML")
                  else:

                        eventid = f"{result} {call.data[5:]}"
                        con = sqlite3.connect(database)
                        cur = con.cursor()
                        cur.execute("UPDATE users SET izb = ? WHERE id =
?",(eventid, call.message.chat.id,))
                        con.commit()
                        bot.send_message(call.message.chat.id,"<i>Додано до
обраного</i>", parse_mode = "HTML")

            else:
                  eventid = call.data[5:]

                  con = sqlite3.connect(database)
                  cur = con.cursor()
                  cur.execute("UPDATE users SET izb = ? WHERE id = ?",(eventid,
call.message.chat.id,))
                  con.commit()
                  bot.send_message(call.message.chat.id,"<i>Додано до
обраного</i>", parse_mode = "HTML")

      if call.data[:5] == "nagad":
            con = sqlite3.connect(database)
            cur = con.cursor()
            cur.execute("SELECT id_notification FROM users WHERE id = ?
",(call.message.chat.id,))
            result = cur.fetchone()[0]

            if len(result) != 0:

                  if call.data[6:] in result:
                        bot.send_message(call.message.chat.id,"<i>Цю подію вже
додано</i>", parse_mode = "HTML")
                  else:

                        eventid = f"{result} {call.data[6:]}"
                        con = sqlite3.connect(database)
                        cur = con.cursor()
```

```
                        cur.execute("UPDATE users SET nagad = ? WHERE id =
?",(eventid, call.message.chat.id,))
                        con.commit()
                        bot.send_message(call.message.chat.id,"<i>Додано до
нагадувань</i>", parse_mode = "HTML")

            else:
                  eventid = call.data[6:]

                  con = sqlite3.connect(database)
                  cur = con.cursor()
                  cur.execute("UPDATE users SET nagad = ? WHERE id = ?",(eventid,
call.message.chat.id,))
                  con.commit()
                  bot.send_message(call.message.chat.id,"<i>Додано до
нагадувань</i>", parse_mode = "HTML")


      if call.data[:7] == "delizbr":
            con = sqlite3.connect(database)
            cur = con.cursor()
            cur.execute("SELECT id_notification FROM users WHERE id = ?
",(call.message.chat.id,))
            result = cur.fetchone()[0]
            id = call.data[8:]
            strid = result.find(id)
            if strid == 0:
                  result = result[result.find(id)+2:]
            else:
                  result = result[:result.find(id)-1]+result[result.find(id)+1:]
            con = sqlite3.connect(database)
            cur = con.cursor()
            cur.execute("UPDATE users SET izb = ? WHERE id = ?",(result,
call.message.chat.id,))
            con.commit()
            bot.delete_message(call.message.chat.id, call.message.message_id)
            bot.send_message(call.message.chat.id,"<i>Видалено з обраного</i>",
parse_mode = "HTML")

      if call.data[:8] == "delnagad":
            con = sqlite3.connect(database)
            cur = con.cursor()
            cur.execute("SELECT id_notification FROM users WHERE id = ?
",(call.message.chat.id,))
            result = cur.fetchone()[0]
            id = call.data[9:]
            strid = result.find(id)
            if strid == 0:
                  result = result[result.find(id)+2:]
            else:
                  result = result[:result.find(id)-1]+result[result.find(id)+1:]
            con = sqlite3.connect(database)
            cur = con.cursor()
            cur.execute("UPDATE users SET nagad = ? WHERE id = ?",(result,
call.message.chat.id,))
            con.commit()
            bot.delete_message(call.message.chat.id, call.message.message_id)
            bot.send_message(call.message.chat.id,"<i>Видалено з нагадувань</i>",
parse_mode = "HTML")

      if call.data == "zanovo":
```

```
            msg = bot.send_message(message.chat.id, "Введіть назву заходу",
parse_mode = "HTML", reply_markup = otmena())
            stat = random.choice(range(0, 999999))
            statushandler(message.chat.id, stat)
            bot.register_next_step_handler(message, zahid, msg, stat)

      if call.data[:6] == "otprav":
            con = sqlite3.connect(database)
            cur = con.cursor()
            cur.execute("SELECT * FROM events WHERE id_events = ?
",(call.data[7:],))
            result = cur.fetchall()[0]

            markup = types.InlineKeyboardMarkup()
            btn1 = types.InlineKeyboardButton(text="Підтвердити",
callback_data=f"confirm {call.data[7:]}")
            btn2 = types.InlineKeyboardButton(text="Відхилити",
callback_data="otmena")
            markup.row(btn1,btn2)
            bot.delete_message(call.message.chat.id, call.message.message_id)
            bot.send_message(call.message.chat.id, "<i>Відправлено на огляд</i>",
parse_mode = "HTML")
            bot.send_photo(admin,f"{result[7]}", f"{result[1]}\n{result[2]}\n
{result[4]} {result[5]}\n<a href='{result[3]}'>Купити білет</a>", parse_mode =
"HTML", reply_markup = markup)

      if call.data[:7] == "confirm":
            con = sqlite3.connect(database)
            cur = con.cursor()
            cur.execute("SELECT * FROM events WHERE id_events = ?
",(call.data[7:],))
            event = cur.fetchall()[0]
            con = sqlite3.connect(database)
            cur = con.cursor()
            cur.execute(f"INSERT INTO events (id, name, desk, link, city, date,
category, foto) VALUES ({event[0]}, \"{event[1]}\", \"{event[2]}\",
\"{event[3]}\", \"{event[4]}\", \"{event[5]}\", \"{event[6]}\", \"{event[7]}\")")
            con.commit()
            bot.send_message(call.message.chat.id, "<i>Захід додано</i>",
parse_mode = "HTML")



@bot.message_handler(content_types=['text'])
def cityhandler(message, msg, stat):
      con = sqlite3.connect(statebase)
      cur = con.cursor()
      cur.execute("SELECT * FROM users WHERE id = ?",(message.chat.id,))
      state = cur.fetchone()[0]

      if state == stat:
            con = sqlite3.connect(database)
            cur = con.cursor()
            cur.execute("SELECT city FROM events WHERE city = ?",(message.text,))
            citybase = cur.fetchone()

            if citybase != None:
                  con = sqlite3.connect(database)
                  cur = con.cursor()
                  cur.execute("UPDATE events SET city = ? WHERE id =
?",(message.chat.id, message.text,))
```

```
                  con.commit()
                  bot.edit_message_text(chat_id=msg.chat.id,
message_id=msg.message_id,text="<i>Оберіть яким чином ви хочете здійснювати
подальший пошук</i>", parse_mode = "HTML", reply_markup = poisk())

            else:
                  bot.delete_message(message.chat.id, message.message_id)
                  bot.edit_message_text(chat_id=msg.chat.id,
message_id=msg.message_id,text="<i>Заходів в даному місті немає в нашій базі
даних. Оберіть інше місто</i>", parse_mode = "HTML", reply_markup = otmenac())
                  stat = random.choice(range(0, 999999))
                  statushandler(message.chat.id, stat)
                  bot.register_next_step_handler(message, cityhandler, msg ,stat)


@bot.message_handler(content_types=['text'])
def poiskhandler(message, msg, stat, search):
      con = sqlite3.connect(statebase)
      cur = con.cursor()
      cur.execute("SELECT * FROM users WHERE id = ?",(message.chat.id,))
      state = cur.fetchone()[0]

      if state == stat:

            if search != "zapyt":
                  poisk = message.text[:1].upper()+message.text[1:].lower()
                  print(poisk)
                  con = sqlite3.connect(database)
                  cur = con.cursor()
                  cur.execute(f"SELECT * FROM events WHERE {search} = ?",(poisk,))
                  result = cur.fetchall()

                  for result in result:
                        bot.send_photo(message.chat.id,f"{result[7]}",
f"{result[1]}\n{result[2]}\n {result[4]} {result[5]}\n<a href='{result[3]}'>Купити
білет</a>", parse_mode = "HTML", reply_markup = eventkb(result[0]))

            else:
                  con = sqlite3.connect(database)
                  cur = con.cursor()
                  cur.execute("SELECT * FROM events WHERE name = ? or date = ? or
city = ? or name = ? or category = ?",(message.text, message.text, message.text,
message.text, message.text,))
                  result = cur.fetchall()

                  if result != None:
                        bot.edit_message_text(chat_id=msg.chat.id,
message_id=msg.message_id,text="<i>Результати пошуку:</i>", parse_mode = "HTML")

                        for result in result:
                              bot.send_photo(message.chat.id,f"{result[7]}",
f"{result[1]}\n{result[2]}\n {result[4]} {result[5]}\n<a href='{result[3]}'>Купити
білет</a>", parse_mode = "HTML", reply_markup = eventkb(result[0]))
                  else:
                        bot.edit_message_text(chat_id=msg.chat.id,
message_id=msg.message_id,text="<i>Нічого не знайдено</i>", parse_mode = "HTML")


@bot.message_handler(content_types=['text'])
def zahid(message, msg, stat):
      con = sqlite3.connect(statebase)
```

```
      cur = con.cursor()
      cur.execute("SELECT * FROM users WHERE id = ?",(message.chat.id,))
      state = cur.fetchone()[0]

      if state == stat:
            event = message.text
            bot.delete_message(message.chat.id, message.message_id)
            msg = bot.edit_message_text(chat_id=msg.chat.id,
message_id=msg.message_id,text=f"<i>Введіть опис заходу</i>", parse_mode = "HTML",
reply_markup = otmena())
            stat = random.choice(range(0, 999999))
            statushandler(message.chat.id, stat)
            bot.register_next_step_handler(message, zahid2, msg, stat, event)

@bot.message_handler(content_types=['text'])
def zahid2(message, msg, stat, event):
      con = sqlite3.connect(statebase)
      cur = con.cursor()
      cur.execute("SELECT * FROM users WHERE id = ?",(message.chat.id,))
      state = cur.fetchone()[0]

      if state == stat:
            event = [event, message.text]
            bot.delete_message(message.chat.id, message.message_id)
            msg = bot.edit_message_text(chat_id=msg.chat.id,
message_id=msg.message_id,text=f"<i>Введіть місто проведення заходу</i>",
parse_mode = "HTML", reply_markup = otmena())
            stat = random.choice(range(0, 999999))
            statushandler(message.chat.id, stat)
            bot.register_next_step_handler(message, zahid3, msg, stat, event)

@bot.message_handler(content_types=['text'])
def zahid3(message, msg, stat, event):
      con = sqlite3.connect(statebase)
      cur = con.cursor()
      cur.execute("SELECT * FROM users WHERE id = ?",(message.chat.id,))
      state = cur.fetchone()[0]

      if state == stat:
            event.append(message.text)
            bot.delete_message(message.chat.id, message.message_id)
            msg = bot.edit_message_text(chat_id=msg.chat.id,
message_id=msg.message_id,text=f"<i>Введіть дату провведення заходу</i>",
parse_mode = "HTML", reply_markup = otmena())
            stat = random.choice(range(0, 999999))
            statushandler(message.chat.id, stat)
            bot.register_next_step_handler(message, zahid4, msg, stat, event)

@bot.message_handler(content_types=['text'])
def zahid4(message, msg, stat, event):
      con = sqlite3.connect(statebase)
      cur = con.cursor()
      cur.execute("SELECT * FROM users WHERE id = ?",(message.chat.id,))
      state = cur.fetchone()[0]

      if state == stat:
            event.append(message.text)
            bot.delete_message(message.chat.id, message.message_id)
            msg = bot.edit_message_text(chat_id=msg.chat.id,
message_id=msg.message_id,text=f"<i>Введіть категорію заходу</i>", parse_mode =
"HTML", reply_markup = otmena())
```

```
            stat = random.choice(range(0, 999999))
            statushandler(message.chat.id, stat)
            bot.register_next_step_handler(message, zahid5, msg, stat, event)

@bot.message_handler(content_types=['text'])
def zahid5(message, msg, stat, event):
      con = sqlite3.connect(statebase)
      cur = con.cursor()
      cur.execute("SELECT * FROM users WHERE id = ?",(message.chat.id,))
      state = cur.fetchone()[0]

      if state == stat:
            event.append(message.text)
            bot.delete_message(message.chat.id, message.message_id)
            msg = bot.edit_message_text(chat_id=msg.chat.id,
message_id=msg.message_id,text=f"<i>Введіть посилання на захід</i>", parse_mode =
"HTML", reply_markup = otmena())
            stat = random.choice(range(0, 999999))
            statushandler(message.chat.id, stat)
            bot.register_next_step_handler(message, zahid6, msg, stat, event)

@bot.message_handler(content_types=['text'])
def zahid6(message, msg, stat, event):
      con = sqlite3.connect(statebase)
      cur = con.cursor()
      cur.execute("SELECT * FROM users WHERE id = ?",(message.chat.id,))
      state = cur.fetchone()[0]

      if state == stat:
            event.append(message.text)
            bot.delete_message(message.chat.id, message.message_id)
            msg = bot.edit_message_text(chat_id=msg.chat.id,
message_id=msg.message_id,text=f"<i>Введіть посилання на фото для заходу</i>",
parse_mode = "HTML", reply_markup = otmena())
            stat = random.choice(range(0, 999999))
            statushandler(message.chat.id, stat)
            bot.register_next_step_handler(message, zahid7, msg, stat, event)

@bot.message_handler(content_types=['text'])
def zahid7(message, msg, stat, event):
      con = sqlite3.connect(statebase)
      cur = con.cursor()
      cur.execute("SELECT * FROM users WHERE id = ?",(message.chat.id,))
      state = cur.fetchone()[0]

      if state == stat:
            event.append(message.text)
            eventid = random.choice(range(0, 999999))
            con = sqlite3.connect(database)
            cur = con.cursor()
            cur.execute(f"INSERT INTO newevent (id, name, desk, link, city, date,
category, foto) VALUES ({eventid}, \"{event[0]}\", \"{event[1]}\", \"{event[5]}\",
\"{event[2]}\", \"{event[3]}\", \"{event[4]}\", \"{event[6]}\")")
            con.commit()
            markup = types.InlineKeyboardMarkup()
            btn1 = types.InlineKeyboardButton(text="Відправити",
callback_data=f"otprav {eventid}")
            btn2 = types.InlineKeyboardButton(text="Відмінити",
callback_data="otmena")
            markup.row(btn1,btn2)
```

```
            bot.delete_message(message.chat.id, message.message_id)
            bot.delete_message(msg.chat.id, msg.message_id)
            bot.send_photo(message.chat.id,f"{event[6]}",
f"{event[0]}\n{event[1]}\n{event[2]} {event[3]}\n<a href='{event[5]}'>Купити
білет</a>", parse_mode = "HTML", reply_markup = markup)

            statushandler(message.chat.id, 0)









bot.polling(none_stop=True)
```