Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)

Кафедра *Системотехніки*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Дослідження та аналіз допоміжних інформаційних технологій для тимчасового прихистку тварин
(тема)

Виконав: студент *2* курсу, групи *ІТПм-22-1*
спеціальності *122 Комп'ютерні науки*
(код і повна назва спеціальності)

Букарєв А.В.
(прізвище, ініціали)

Спеціальність *122 Комп'ютерні науки*
(код і повна назва спеціальності

Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)

Освітня програма *ОПП Інформаційні технології проектування*
(повна назва освітньої програми)

Керівник *доцент Ребезюк Л.М.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ *Гребеннік І.В.*
(підпис) (прізвище, ініціали)

2024 р.

Я як студент ХНУРЕ розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Студент  *Букарєв А.В.*

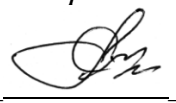
Кваліфікаційна робота не містить відомостей заборонених до відкритого опублікування.

Керівник кваліфікаційної роботи  *доц. Ребезюк Л.М.*

Кваліфікаційна робота виконана у відповідності до діючих стандартів в Україні.

Керівник кваліфікаційної роботи  *доц. Ребезюк Л.М.*

Попередній захист проведено 18 січня 2024 р.

Керівник кваліфікаційної роботи  *доц. Ребезюк Л.М.*

Харківський національний університет радіоелектроніки

(назва закладу вищої освіти)

Факультет *Комп'ютерних наук*

Кафедра *системотехніки*

Рівень вищої освіти *перший (бакалаврський)*

Спеціальність *122 Комп'ютерні науки*

(код і повна назва)

Тип програми *освітньо-професійна*

Освітня програма *Комп'ютерні науки*

(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:

Зав. кафедри *СТ*

проф. Гребеннік І.В.

" _____ " _____ *2023 р.*

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Букарєву Артему Владиславовичу*

(прізвище, ім'я, по батькові)

Тема роботи *Дослідження та аналіз допоміжних інформаційних технологій для тимчасового прихистку тварин*

затверджена наказом по університету від " *14* **"** *грудня* **2023р. №** *1466Ст*

2. Термін подання студентом роботи *17 січня 2024 р.*

3. Вихідні дані до роботи. *Дослідження допоміжних інформаційних технологій для тимчасового прихистку тварин та розробка чат-бота. Перелік використовуваних програмних засобів: ОС Microsoft Windows v.10 або вище, Telegram Bot Api, Python 3, середовище розробки PyCharm. Технічнезабезпечення: IBM-сумісний ПК з МП Pentium II або вище.*

4. Перелік питань, що потрібно опрацювати в роботі

4.1 Вступ. 4.2 Аналіз предметної області. 4.2.1 Постановка задач дослідження та розробки 4.2.2. Аналіз web-сайтів компаній. 4.2.3. Аналіз чат-ботів 4.3 Аналіз моделей, архітектур та інформаційних технологій віртуальних помічників 4.3.1 Аналіз інтелектуальних основ віртуальних помічників в історичному аспекті. 4.3.2 Аналіз архітектур чат-ботів 4.3.3 Аналіз варіантів розробки чат-ботів 4.3.4 Аналіз підходів до розробки чат-ботів 4.4 Структурне проєктування віртуального помічника 4.4.1 Розробка системних та функціональних вимог до віртуального помічника 4.4.2 Розробка моделі потоків даних віртуального помічника 4.5 Програмна реалізація чат-бота тимчасового прихистку тварин 4.5.1 Вибір платформи для створення чат-бота 4.5.2 Вибір мови програмування 4.5.3 Вибір бібліотек 4.5.4 Telegram Bot Api та реєстрація токену 4.5.5 Програмна розробка чат-бота тимчасового прихистку тварин 4.6 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) *5.1 Предмет дослідження та об'єкт розробки, мета роботи (слайд 1). **5.2** Актуальність (слайд 2). **5.3** Постановка задач дослідження та розробки віртуального помічника (слайд 3). **5.4** Функціональні вимоги до чат-бота консультанта (слайд 4). **5.5** Класифікація віртуальних помічників (слайд 5) **5.6** Приклади віртуальних помічників (слайд 6). **5.7** Діаграма галузей, в яких найбільше застосовуються чат-боти (слайд 7). **5.8** Приклад використання різних видів чат-боту в бізнесі (слайд 8). **5.9** Узагальнена архітектура чат-бота (слайд 9). **5.10** Види моделей чат-ботів (слайд 10,11). **5.11** Варіанти розробки чат-ботів (слайд 12,13,14). **5.12** Підходи до розробки чат-ботів (слайд 15,16) **5.13** Схема взаємодії клієнта із чат-ботом (слайд 17) **5.15** Діаграма прецедентів для зареєстрованого користувача (слайд 18) **5.16** Схема даних для платформи MySQL Server (слайд 19) **5.17** Декомпонована схема архітектури чат-бота (слайд 20). **5.18** Генерація токену для чат-бота (слайд 21) **5.19** Розроблений Чат-бот Telegram (слайд 22) **5.20** Висновки (слайд 23)*

6. Дата видачі завдання *20 листопада 2023 р.*

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів роботи	Термін виконання етапів Роботи	Примітка
1.	*Отримання завдання на виконання кваліфікаційної роботи.*	*20.11.23*	
2.	*Аналіз завдання та предметної області*	*21.11 – 23.11.23*	
3.	*Аналіз існуючих web-сайтів та чат-ботів прихистків тварин*	*24.11 – 30.11.23*	
4.	*Аналіз моделей, архітектур та інформаційних технологій віртуальних помічників*	*01.12 – 05.12.23*	
5.	*Структурне проєктування віртуального помічника*	*06.12 – 15.12.23*	
6.	*Вибір архітектури інформаційної системи тимчасового прихистку для тварин*	*16.12 – 24.12.23*	
7.	*Вибір платформи, мови програмування та бібліотек для створення чат-бота*	*25.12 – 31.12.23*	
8.	*Telegram Bot Api та реєстрація токену*	*01.01 – 08.01.24*	
9.	*Програмна розробка чат-бота тимчасового прихистку тварин*		
10.	*Оформлення пояснювальної записки та презентаційних матеріалів комп'ютерного захисту*	*09.01 – 16.01.24*	
11.	*Представлення роботи на рецензування*	*17.01.24*	

Студент _____ *Букарєв А.В.*
(підпис)

Керівник роботи _____ *доц. Ребезюк Л.М.*
(підпис)

ABSRACT

Master's thesis: *92* p., *2* tabl., *29* fig., *1*app., *29* sources.

SHELTER FOR ANIMALS, VIRTUAL ASSISTANT, CHAT-BOT, CHAT-BOT DEVELOPMENT TECHNOLOGY, TELEGRAM, TELEGRAM BOT API, PYTHON.

The object of development is a virtual online assistant in the form of a chatbot that allows automating information processes for animal shelters.

The subject of the research is information technologies for the creation of virtual online assistants in the social sphere in the form of a chatbot.

The purpose of the work is the research and analysis of auxiliary information technologies as virtual assistants, namely chatbots and the development of a chatbot for a temporary animal shelter.

Development methods – methods of functional analysis, information technologies for online mobile communication tools in the form of a chat-bot, flexible development methodology (Agile).

An analysis of the subject area was carried out in the work. As part of the analysis of models, architectures and information technologies of virtual assistants, an analysis of the intellectual foundations of virtual assistants in the historical aspect, analysis of chatbot architectures, options and approaches to the development of chatbots was carried out. As part of the structural design of the IS, an analysis of the interaction of the client of the animal shelter with the chatbot was carried out, the main components of the virtual assistant system were determined and their functionality was analyzed. Developed registered user case diagram and database schema for MySQL Server platform. As part of the software implementation of the chatbot of the temporary animal shelter, the software development of the chatbot was carried out based on the selected tool (platform, programming language and libraries).

РЕФЕРАТ

Кваліфікаційна робота: *92* стор., *2* табл., *29* рис., *1* дод., *29* джерел.

ПРИХИСТОК ДЛЯ ТВАРИН, ВІРТУАЛЬНИЙ ПОМІЧНИК, ЧАТ-БОТ, ТЕХНОЛОГІЯ РОЗРОБКИ ЧАТ-БОТІВ, TELEGRAM, TELEGRAM BOT API, PYTHON.

Об'єкт розробки – віртуальний онлайн-помічник у вигляді чат-боту, що дозволяє автоматизувати інформаційні процеси для прихистку тварин.

Предмет дослідження – інформаційні технології створення віртуальних онлайн-помічників в соціальній сфері у вигляді чат-боту.

Мета роботи – дослідження та аналіз допоміжних інформаційних технологій в якості віртуальних помічників та розробка чат-бота тимчасового прихистку тварин.

Методи розробки –методи функціонального аналізу, інформаційні технології для онлайт-засобів мобільного зв’язку у вигляді чат-боту, гнучка методологія розробки (Agile).

В роботі проведено аналіз предметної області. В рамках аналізу моделей, архітектур та інформаційних технологій віртуальних помічників проведено аналіз інтелектуальних основ віртуальних помічників в історичному аспекті, аналіз архітектур чат-ботів, варіантів та підходів до розробки чат-ботів. В рамках структурного проєктування ІС проведено аналіз взаємодії клієнта прихистку тварин із чат-ботом, визначені основні компоненти системи віртуального помічника та проаналізовано їх функціонал. Розроблено діаграму прецедентів для зареєстрованого користувача та схему бази даних для платформи MySQL Server. В рамках програмної реалізація чат-бота тимчасового прихистку тварин на основі вибраного інструменту (платформи, мови програмування та бібліотек) виконана програмна розробка чат-бота.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

API – Application Programming Interface;

GM – Generative Model;

GPT – Generative Pre-trained Transformer;

IDE – Integrated Development Tnvironment

LSTM – Long Short-Term Memory;

NLU – Natural Language Understanding;

РНМ – рекурентні нейронні мережі (RNN);

ІС – інформаційна система;

ШІ – штучний інтелект

ВСТУП

Інформаційні технології мають неабиякий вплив на розвиток сучасного суспільства. ІТ не тільки кардинально змінили роль інформації для людей, а ще й надали можливість дозволили швидко та ефективно розповсюджувати її.

В Україні та в місті Харків на початок війни закладів тимчасового прихистку для тварин існувало досить мало, оскільки ця ніша ще не настільки популярна, на відміну від закордону. Окрім цього, сайти існуючих закладів прихистку для тварин, не були функціонально пристосовані до масової міграції населення зі своїми тваринами у воєнний час.

З початком воєнної агресії стало актуальним створення онлайн-сервісу, який функціонально дозволить оперативно знайти тимчасовий прихисток для тварин в умовах війни для тих хазяїв, хто не мав змоги забрати із собою улюблену тварину. Подібні веб-додатки повинні мати можливість для розміщення інформації та правила щодо розміщення тварин, як довго вони будуть знаходитися у прихистку та особливі умови утримання. Окрім цього, використання інформаційних технологій в сфері послуг забезпечить функціонування прихистку на вищому рівні внаслідок того, що буде знижено навантаження на адміністратора прихистку.

Розробка інформаційної веб-системи із усуненими аналогічними недоліками веб-системи, була проведена в рамках виконання кваліфікаційної роботи на здобуття ступеня «бакалавр».

Використання засобів мобільного зв’язку дозволяє підтримувати постійний контакт людей між собою, між владою та населенням. Згідно зі статистичними даними, сучасна людина в середньому витрачає 3,4 години часу на добу, тримаючи в руках свій телефон. Понад половина цього часу витрачається на комунікацію через менеджери та соціальні мережі. В остальний час люди займаються переглядом розважального контенту, шукають товари та послуги, або грають в ігри [1,2].

На другому році війни стало зрозумілим, що веб-системи не дозволяють

оперативно отримувати необхідну інформацію користувачам, у тому числі і про своїх тварин. Тому, для отримання населенню України оперативної інформації масово почали створюватися мобільні чат-боти різного направлення.

Таким чином, актуальним стає дослідження можливості створення чат-бота тимчасового прихистку тварин як допоміжної до веб-сайту інформаційної технології.

Об'єкт розробки – віртуальний онлайн-помічник у вигляді чат-боту, що дозволяє автоматизувати інформаційні процеси для тимчасового прихистку тварин.

Предмет дослідження – інформаційні технології створення віртуальних онлайн-помічників в соціальній сфері у вигляді чат-боту.

Мета роботи – дослідження та аналіз допоміжних інформаційних технологій в якості віртуальних помічників, а саме чат-ботів та розробка чат-бота тимчасового прихистку тварин.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка задач дослідження та розробки

Метою роботи є проведення дослідження та аналіз допоміжних інформаційних технологій в якості віртуальних помічників, а саме чат-ботів та розробка чат-бота тимчасового прихистку тварин.

Згідно завданню та меті роботи необхідно вирішити наступні задачі:

– виконати аналіз предметної області, а саме аналіз аналогів web-сайтів та чат-ботів прихистку тварин;

– виконати аналіз інтелектуальних основ віртуальних помічників, моделей та архітектур чат-ботів, варіантів та підходів до розробки чат-ботів;

– виконати структурне проєктування віртуального помічника на основі аналіза системних та функціональних вимог до віртуального помічника та розробити модель потоків даних віртуального помічника;

– обрати інструменти розробки (платформу, мову програмування та бібліотеки) для програмної реалізація;

– розробити чат-бот тимчасового прихистку тварин.

Функціональні вимоги до чат-бота консультанта, як до основного із компонентів розроблюваної система:

– допомога користувачу зі створення заявки на розташування домашньої тварини у тимчасовому прихистку, а саме приймати у користувача інформацію для заявки на прихисток (ПІБ власника, вид тварини, дата розміщення тощо);

– передача інформацію заявки на прихисток для обробки у службу підтримки.

Базуючись на результатах аналізу існуючих моделей, архітектур та підходів для розробки чат-ботів необхідно:

– продумати архітектуру бота, обрати інструменти (платформу, мови програмування та бібліотеки);

– розробити чат-бот тимчасового прихистку тварин, забезпечивши зруч-

ний та логічний діалог з користувачем.

1.2 Аналіз web-сайтів компаній, що надають інформаційні послуги з прихистку тварин

1.2.1 Сайт «Готель для кішок «Найкращий друг» є односторінковим інформаційним сервісом [3]. На сайті надана інформація про вартість розміщення тварини, контакти та розміщення готелю (див.рис.1.1). Також доступні відгуки власників, які вже мали досвід з даним готелем.

До переваг веб-сервісу можна віднести:

– простота та лаконічність UI/UX дизайну;
– адаптивність дизайну до мобільних пристроїв.

До недоліків веб-сервісу можна віднести:

– відсутність можливості онлайн бронювання прихистка;
– відсутність можливості онлайн оплати;
– відсутність особистого кабінету користувача та реєстрації користувача.

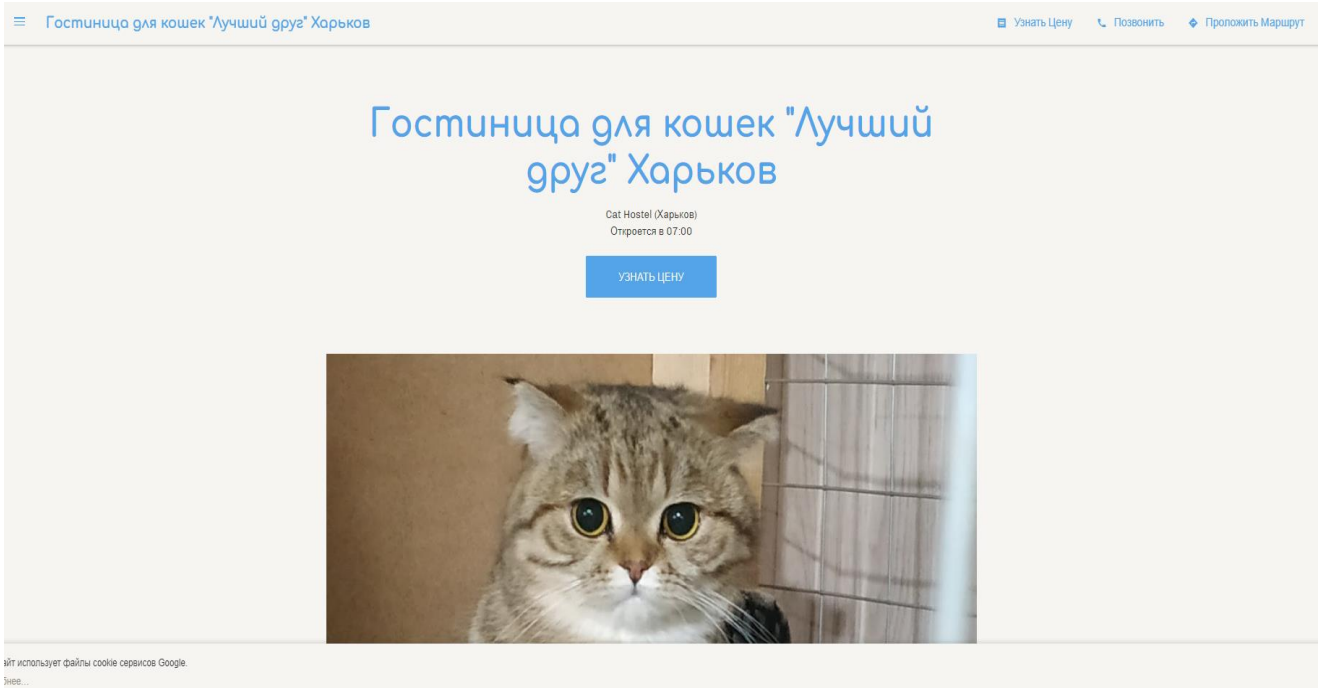


Рисунок 1.1 – Скриншот web-сервісу Готель для кішок «Найкращий друг»

1.2.2 Сайт «Готель для тварин «Darliss» [4] має вже більш розширений функціонал порівняно з попереднім готелем «Найкращий друг» (див.рис.1.2).

На сайті доступна галерея з фото наданих номерів прихистка, їх повним описом, а також повним описом переліку послуг. Також сервіс дозволяє здійснювати онлайн відеоспостереження в режимі реального часу за вихованцями. З доступних для користувачів опцій – вибір міста та онлайн перегляд доступних номерів.

До переваг веб-сервісу можна віднести:

– простота та лаконічність UI/UX дизайну;
– адаптивність дизайну до мобільних пристроїв;
– реалізація онлайн спостереження.

До недоліків веб-сервісу можна віднести:

– відсутність можливості онлайн бронювання прихистка;
– відсутність можливості онлайн оплати;
– відсутність особистого кабінету користувача та реєстрації користувача.

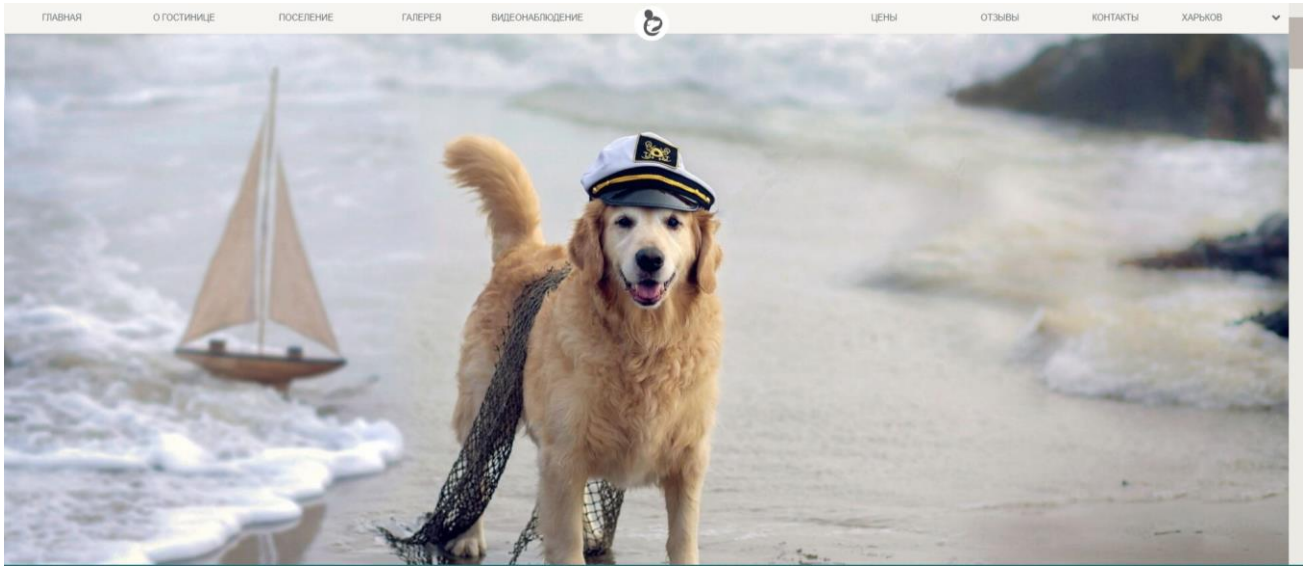


Рисунок 1.2 – Скриншот web-сервісу «Готель для тварин «Darliss»

1.2.3 Сайт «Готель для тварин «vet-domik» ветеринарної клініки, скриншот якого показано на рисунку 1.3, на поєднує у собі можливості притулку для

тварин та ветеринарної консультації [5].

На сайті доступна інформація щодо догляду за вихованцем з можливістю запису на консультацію.

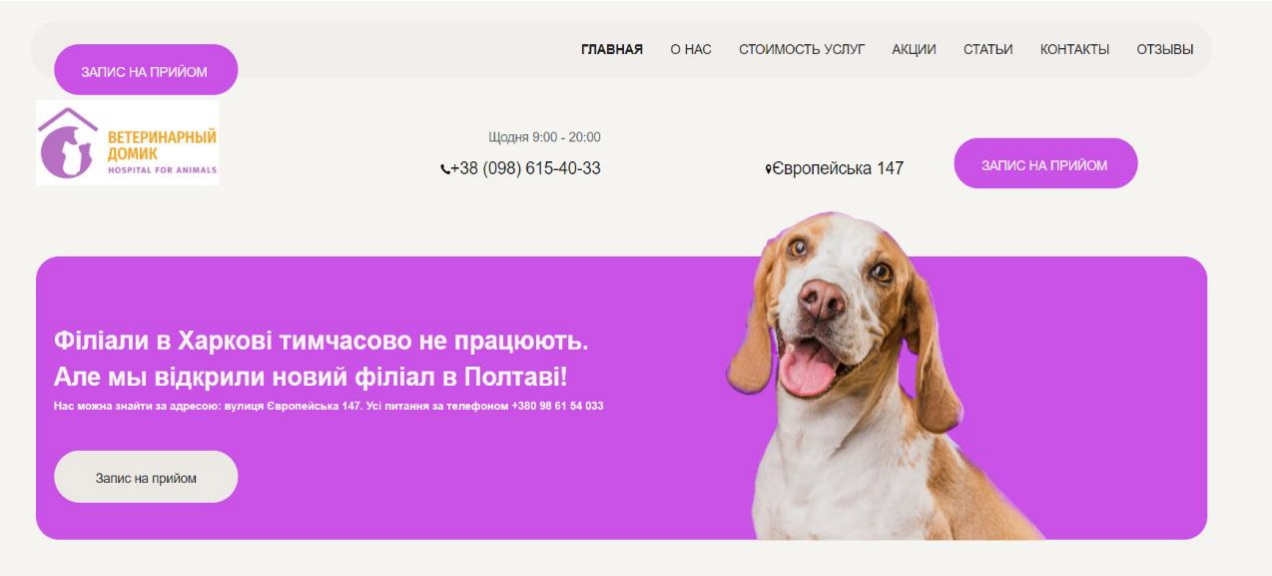


Рисунок 1.3 – Скриншот web-сервісу «Готель для тварин «vet-domik»

Запис на консультацію та відправка заявки на тимчасове притулок здійснюється за допомогою діалогового вікна, де користувачеві необхідно вказати контактні дані (телефон), ініціали та вид вихованця (див.рис.1.4).

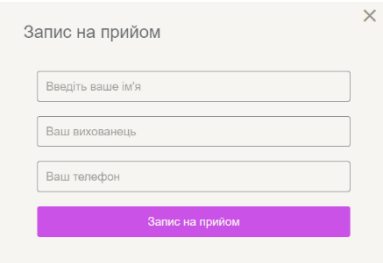


Рисунок 1.4 – Скриншот діалогового вікна на запит консультації/прихистку

Решта функціоналу сайту зводиться до інформаційної частини, а саме надання інформації:

– інформація про саму клініку та прилисток;

– вартість послуг;

– акції;

– статті;

– контакти;

– відгуки.

Також доступні активні посилання на соціальні мережі.

До переваг веб-сервісу можна віднести:

– простота та лаконічність UI/UX дизайну;

– адаптивність дизайну до мобільних пристроїв.

До недоліків веб-сервісу можна віднести:

– відсутність можливості онлайн бронювання прихистку;

– відсутність можливості онлайн оплати;

– відсутність особистого кабінету користувача та реєстрації користувача.

В результаті проведеного аналізу веб-сервісів були виявлені критичні у воєнний час наступні недоліки:

– неможливість створення облікового запису у системі;

– неможливість оформлення онлайн замовлення на бронювання номеру;

– замала кількість інформації для оформлення замовлення;

– неможливість оплати сервісних послуг.

– неможливість оформити доставку тварини до тимчасового прихистку.

1.3 Аналіз чат-ботів, що надають інформаційні послуги з прихистку тварин

Інтернет-маркетинг пристосувався до нової реальності, впроваджуючи адаптивно-медійні оголошення у мобільній версії. Тому виникнення чат-ботів було очікуваним кроком. З їхньою допомогою бізнес вирішує різні задачі, такі як консультації, збільшення кількості продажів та підтримка звернень клієнтів. Чат-бот для користувачів представляє собою віртуальний асистент, що сприяє задоволенню потреб користувача. Чат-ботах інформація надається у формі тексту, інтерактиву або мультимедійного контенту [6].

Серед основних переваг чат-ботів можна виділити наступні переваг [7,8]:

– надання сервісного обслуговування цілодобово. За результатами опиту-

вання 64% респондентів відзначили цілодобовий сервіс як одну з основних переваг використання чат-ботів, що підвищує якість обслуговування та сприяє успішності бізнесу;

– допомога у лідогенерації клієнтів. Згідно з дослідженням, 69% користувачів для взаємодії віддають перевагу чат-ботам перед іншими сервісами. У бізнесі це дозволяє досягти широкої аудиторії з інформацією про свої продукти і послуги, що має економічний зміст [6].

На теперішній час впровадження віртуального співрозмовника є успішним у державних організаціях. Наприклад, для спрощення навігації на веб-сайтах, допомоги у заповненні форм та поданні документів. На сьогоднішній день в Україні, як і в усьому світі, відбувається процес автоматизації різних аспектів людського життя. Соціальна сфера не є винятком, і в різних державних установах з'являються свої інформаційні сервіси/сайти та віртуальні помічники [8].

Чат-боти отримали широке застосування в наступнах державних установах [8]:

– газові служби та електроспоживання (для передачі показників лічильника),

– паспортні відділи (для отримання інформації щодо надання паспортних послуг),

– СБУ (запустили чат-бота, через якого можна повідомити про мародерів або підозрілих осіб),

– УБКІ (з чат-ботом, де можна перевірити кредитну історію та інші операції, пов'язані з кредитами),

– Держстат (для дізнання статистики та роботи з ЄДРПОУ),

– Мінцифри (чат-бот застосунку "Дія");

– освіта. Вже розроблено безліч віртуальних помічників для вивчення, повторення та закріплення інформації. Існують дослідження щодо ефективності використання чат-ботів в освіті.

Використання чат-ботів у бізнесі, а саме у маркетингу, надають наступні переваги:

– ефективна взаємодія: програма дозволяє налаштовувати контент для кожного користувача, надаючи їм лише ту інформацію, яку вони запитали. Чат-боти сприяють залученню як потенційних покупців, так і існуючих клієнтів, що призводить до збільшення продажів. Вони відрізняються від традиційних методів обслуговування, надаючи релевантну інформацію без перевантаження аудиторії, що сприяє довшому утриманню інтересу аудиторії завдяки автоматизованому ланцюжку повідомлень [7,8];

– економія бюджету компанії та легкість експлуатації: використання чат-ботів є одноразовою інвестицією, яка не потребує подальших витрат на персонал, оскільки вони можуть відповідати на прості запитання потенційних покупців та передавати складніші менеджерам з обслуговування замовлень. Чат-боти дозволяють компаніям обслуговувати клієнтів на декількох мовах, що допомагає розширювати діяльність бренду на нових ринках;

– можливість вимірювання даних: легко відстежити доставку контенту, залученість користувачів та проводити сегментацію аудиторії на основі дій користувачів. Бот також забезпечує збір зворотного зв'язку від клієнтів для подальшого удосконалення послуг;

– генерація, кваліфікація та "прогрів" лідів: чат-боти можуть ефективно працювати з аудиторією на всіх етапах воронки продажів, включаючи нічний час. Вони генерують ліди, переконують потенційних покупців та допомагають визначити некваліфіковані ліди за допомогою ключових показників результативності. Окрім цього забезпечують інформацією, яка дозволяє персоналізувати розсилку повідомлень клієнтам на різних етапах воронки продажів, а також допомагають відсікати "важких" клієнтів, які вимагають значних зусиль на комунікацію;

– додаткові можливості: поступове розширення функціоналу чат-бота дозволяє компанії привертати нову аудиторію, наприклад, шляхом введення нових мов інтерфейсу для доступу до міжнародних ринків.

Чат-боти можуть бути впроваджені в різних сферах, лише потрібно придумати ідею та реалізувати її відповідним чином.

Коли за вікном іде війна, що визвала масштабні міграційні процеси, надзвичайно актуальним стали питання прихистку тварин, яких не можливо було взяти з собою іммігрантам, та створення чат-ботів для оперативної цілодобової інформаційної підтримки прихистку тварин.

Проаналізуємо чат-боти для прилаштуванню тварин з різним функціоналом, які почали з’являтися у 2023 року.

1.3.1 Благодійна організація «UAnimals» запустила чат-бот «Прихисток із твариною» [9]. Це бот для тих, хто шукає прихисток із тваринами, та тих, хто готовий його надати (див.рис.1.4).

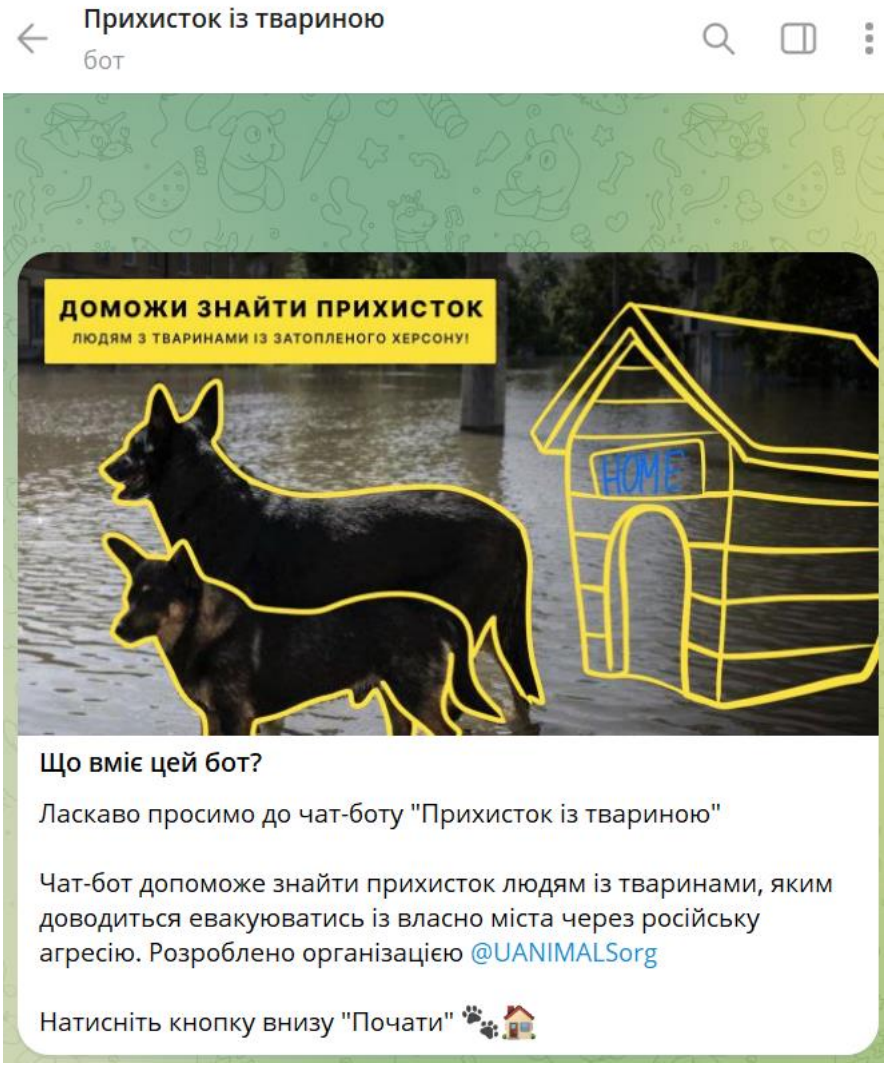


Рисунок 1.4 – Зовнішний вигляд сторінки Telegram-бота «Прихисток із твариною»

Чат-бот покликаний допомогти постраждалим людям та їхнім чотирилапим з Херсонщини. Як працює допомога через чат-бот:

– заповнюєте інформацію у боті про те, що ви шукаєте прихисток або готові прихистити;

– система знаходить анкету, що відповідає вашому запиту;

– отримуєте контакт людини, яка готова прихистити вас і ваших тварин АБО отримуєте контакт людини, яка має запит прихистити, відповідний вашим можливостям.

Недоліком Telegram-бота «Прихисток із твариною» є перевантаженість інформацією, що не стосується безпосередньо прихистку.

1.3.2 Фахівці КП «Київська міська лікарня ветеринарної медицини» створили Telegram-бот TinPet, який допомогає обрати домашнього улюбленця з притулку (див.рис.1.5) [10]. З його допомогою сподіваються розв’язати проблему безпритульних тварин у столиці.

Для того, щоб скористатись, необхідно зареєструватись в чат-боті TinPet і створити акаунт для пошуку улюбленця або пошуку домівки для тварини.

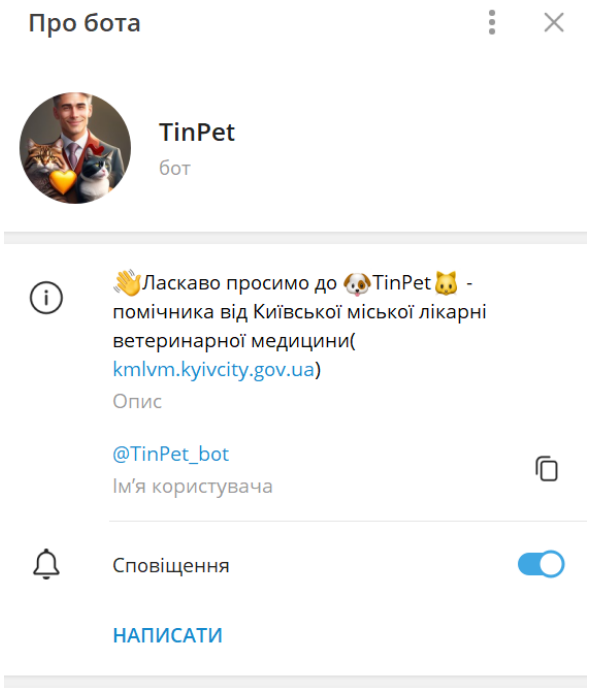


Рисунок 1.5 – Вигляд інформаційної сторінки Telegram-бота TinPet

Важливо, що сервісом можуть скористатися представники інших притулків, яким сервіс доступний. Через чат-бот вони зможуть поширювати інформацію про своїх підопічних, що значно спрощує прилаштування тварин,

Наразі чат-бот працює в тестовому режимі, тому через технічні особливості один користувач може завантажити лише 3 анкети.

Здебільшого інформація щодо зоомагазинів, волонтерські служби та онлайн-ветеринарів надається у регіональних новинних Telegram-ботах, як показано на рисунку 1.6 [11].

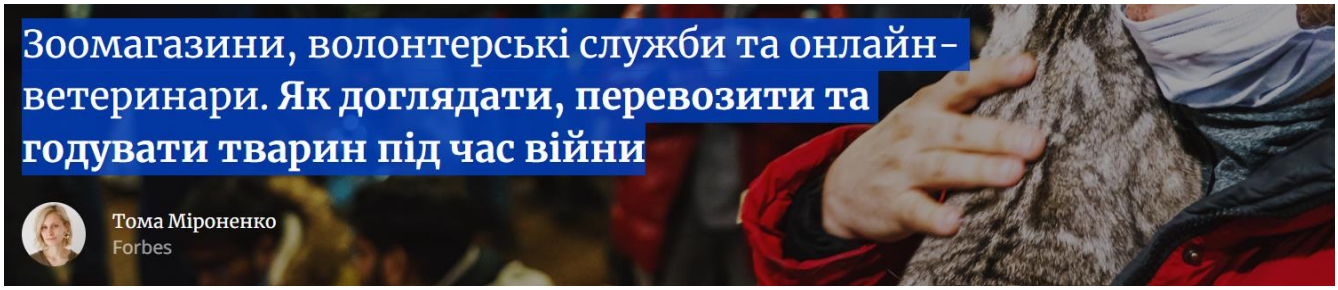


Рисунок 1.6 – Приклад інформації, що надається у новонних регіональних Telegram-ботах

2 АНАЛІЗ МОДЕЛЕЙ, АРХІТЕКТУР ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ВІРТУАЛЬНИХ ПОМІЧНИКІВ

2.1 Аналіз інтелектуальних основ віртуальних помічників в історичному аспекті

Саме на початку 2000-х років відбувся піковий інтерес до теперішнього маловідомого віртуального помічника. Деякі впроваджували його безпосередньо в основну програму, інші створювали його як окремий додаток до основного. Програми почали заміщувати консультантів та відповідати на десятки тисяч питань користувачів щоденно, що дозволило деяким компаніям заощаджуючи отримувати значні фінансові вигоди.

Ще однією революцією стала поява месенджерів та соціальних мереж, які дозволили людям спілкуватись в інтернеті, незалежно від місцезнаходження на планеті. Поширення віртуальних користувачів в цих платформах стало лише питанням часу. Важливо відзначити, що для використання чат-боту не потрібно завантажувати додаткові програми, так як, якщо ви вже присутні в соціальній мережі чи месенджері, ви вже є потенційним користувачем віртуального консультанта.

Історія створення чат-ботів визначається численними подіями та інноваціями в області інформаційних технологій. Ось загальна історія та кілька ключових моментів:

1) коріння чат-ботів можна віднести до робіт у галузі штучного інтелекту та обробки природної мови, які виникли в середині 20-го століття. Алан Тюрінг, в своїй статті "Computing Machinery and Intelligence" (1950), поставив питання про можливість машинного мислення та розглядав можливості для створення програм, що можуть розмовляти з людьми [12];

2) у 60-і відбувались перші спроби створення систем обробки природної мови, а також поява ранніх програм, які намагалися імітувати розмову. Створена Джозефом Вайзенбаумом у 1966 році ELIZA була призначена для імітації роз-

мови з психотерапевтом та використовувала шаблони для визначення відповідей на введені користувачем фрази. По суті ELIZA – це один з перших та найвідоміших чат-ботів. При цьому ELIZA використовувала метод "розбиття" (pattern matching), що означає визначення ключових слів у введеному тексті та відповідь на них відповідним чином, тобто використовувала прості шаблони та правила для визначення відповідей на введені користувачем фрази. Цей метод був досить ефективний, але програма була обмеженою у своїх можливостях та розумінні. Програма стала важливим кроком у розвитку обробки природної мови та віртуальних асистентів. Багато з ідеї ELIZA покладено в основу подальших досліджень та розробок в галузі штучного інтелекту та чат-ботів [13 – 15];

3) у період 1990-2010 рр. спостерігалося збільшення інтересу до області чат-ботів, але технологічний прогрес був обмеженим. Розвиток Інтернету та обчислювальних ресурсів дав певний поштовх в розвитку чат-ботів;

4) у період 2010-2015 рр. з'явилося кілька платформ та інструментів для створення чат-ботів, що дозволило розробникам створювати їх без значного володіння програмуванням. Популярні платформи, такі як Chatfuel, Dialogflow та Wit.ai, забезпечили можливість створювати ботів без глибоких знань в галузі програмування;

5) з 2015 р. зростає зацікавленість до чат-ботів внаслідок введення популярних месенджерів, таких як Telegram, WhatsApp, Viber та Slack, які стали платформами для розповсюдження та використання чат-ботів. Технології штучного інтелекту, такі як нейронні мережі та глибоке навчання, значно покращили здатність ботів до розпізнавання мови та взаємодії з користувачами [14, 15];

6) у майбутньому, з розвитком технологій штучного інтелекту, аналізу даних та машинного навчання буде і далі вдосконалювати здатність чат-ботів до ефективної взаємодії з користувачами. Очікується, що чат-боти будуть використовувати більше інтелектуальних та адаптивних стратегій для надання персоналізованої інформації та обслуговування, ефективність та придатність яких залежать від конкретного використання та умов впровадження. Окремо варто відмітити віртуальних помічників із аудіо підтримкою, наприклад, Siri та Google

Assistant. Так, функціонал Siri (див.рис.2.1) дуже обширний: вона вміє шукати інформацію в інтернеті, відправляти електронну пошту, набирати вказаного абонента, підказувати список своїх команд та відповідати на певні прості питання [14, 15].

Чат-боти головним чином використовують штучний інтелект для взаємодії з користувачами, що робить їх здатними надавати релевантний контент та актуальні пропозиції. Вони можуть працювати на основі набору інструкцій або використовувати машинне навчання. Функціональність чат-бота, що використовує інструкції, може бути обмеженою і, головним чином, спрямованою на відповіді на конкретні питання. Таким чином, якщо користувач ставить питання не відповідно до програми, бот може виявити обмежену здатність відповісти [15].

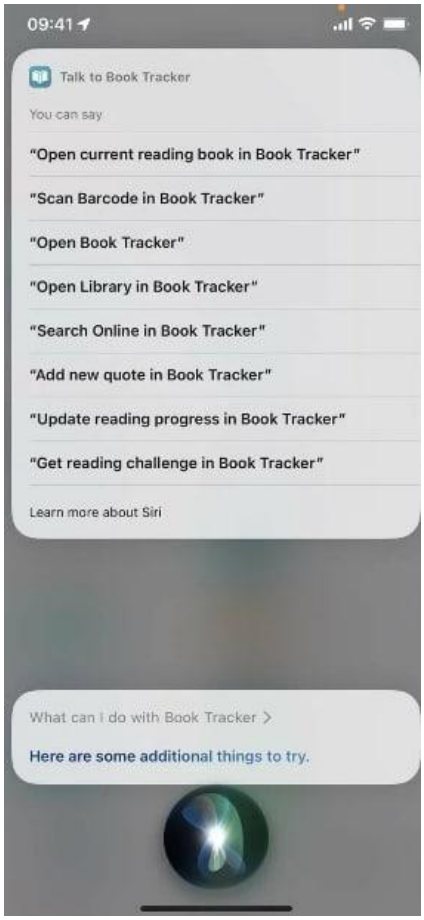


Рисунок 2.1 - Інтерфейс голосового помічника Siri

Рівень інтелекту чат-бота повністю залежить від його програмування. Чат-бот на основі машинного навчання проявляє кращу ефективність, оскільки він розуміє не лише конкретні команди, а й природну мову. Це означає, що користувач може отримувати відповіді, не вводячи точні слова. Крім того, бот навчається взаємодіяти з клієнтами і легко вирішувати подібні ситуації, які можуть виникнути – з кожним діалогом поліпшується інтелект чат-бота [15].

Окрім чат-бота на базі штучного інтелекту, існує інший різновид, який стане у пригоді маркетологам без технічного досвіду. Цей чат-бот є більш простим та спрощує масові розсилки, і бренди використовують його для розширення можливостей електронної пошти, а також стратегій веб-повідомлень [6]. Кампанії у Facebook, з використанням таких ботів, дозволяють збільшити охоплення аудиторії, підвищити продажі і покращити якість обслуговування клієнтів. З сервісом SendPulse можна розпочати використання чат-бота, навіть якщо немає знань в галузі кодування [16].

2.2 Аналіз архітектур чат-ботів

Архітектура чат-ботів може бути різноманітною залежно від призначення та завдань, які вони повинні виконувати, і використовуваних алгоритму та технологій [17,18].

Чат-боти можна класифікувати за кількома ознаками [17]:

а) алгоритмом роботи:

1) обмежені, які реагують на конкретні запити користувачів заздалегідь визначеним сценарієм і обмеженою кількістю можливих відповідей;

2) саморозвиваючі, які розроблені на основі штучних нейронних мереж та які можуть розуміти сутність розмови, проводити реалістичні діалоги з користувачем та поступово навчатися, щоб надавати більш релевантні відповіді на запити;

б) форматом взаємодії з користувачем;

в) призначенням.

За форматом взаємодії з користувачем існують наступні типи чат-ботів [17]:

а) кнопковий, які використовуються в месенджерах і в яких взаємодія із користувачем відбувається через кнопки з варіантами дій. Бот реагує на них як на команди, надаючи уточнювальні кнопки або відповідаючи на поставлені запитання (див.рис.2.2);

Рисунок 2.2 – Приклад кнопкового чат-боту

б) текстовий, який є найбільш функціональним типом віртуального співрозмовника. Взаємодія з ним подібна до мовлення людини, оскільки робот розпізнає запит, проводить аналіз інформації та обирає найбільш релевантну відповідь із заздалегідь підготовлених варіантів (див.рис.2.3);

в) вбудований (inline), який з'являється всередині розмови у месенджері після виклику та пропонує різні опції дій. Результати можна обмінюватися зі співрозмовником, з яким тривав діалог. Зазвичай використовується для пошуку

відповідних місць, замовлення їжі та інших подібних послуг (див.рис.2.4).

Рисунок 2.3 – Приклад текстового чат-боту

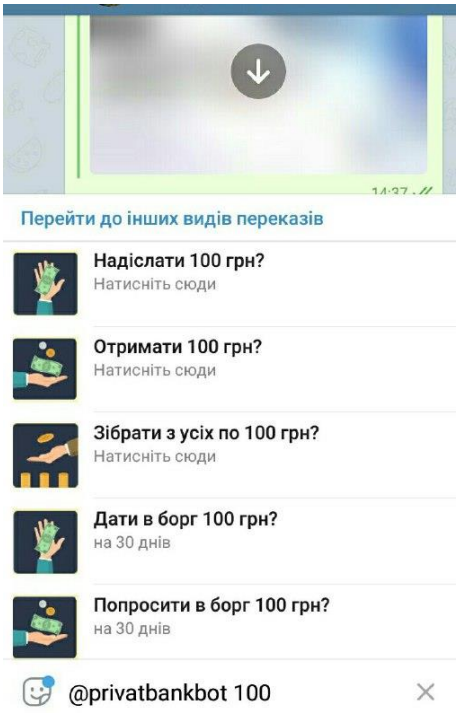


Рисунок 2.4 – Приклад вбудованого чат-боту

Також чат-боти можна класифікувати за призначенням на [17]:

а) комунікаційні чат-боти, які слідують своїй назві та забезпечують взаємодію компанії з клієнтами, включаючи відповіді на стандартні запитання, пропозиції зворотного зв'язку та переадресацію на живого менеджера. Крім того, комунікаційні чат-боти можуть мати рекламний характер, надаючи інформацію про послуги та акції;

б) функціональні чат-боти, які, натомість, виступають в ролі повноцінних мобільних додатків, які об'єднують у себе можливості пошуку, консультування, бронювання, покупок та банківських операцій в одному вікні. Ці чат-боти також вміють надавати інтерактивні функції та персоналізовані відповіді, а їхні можливості можна нескінченно розширювати.

Архітектура чат-ботів ґрунтується на тому, що вони або генерують відповіді з нуля, використовуючи моделі машинного навчання, або визначають відповіді на основі конкретного набору правил.

Загальний вигляд архітектури чат-бота [18] представлено на рисунку 2.5.

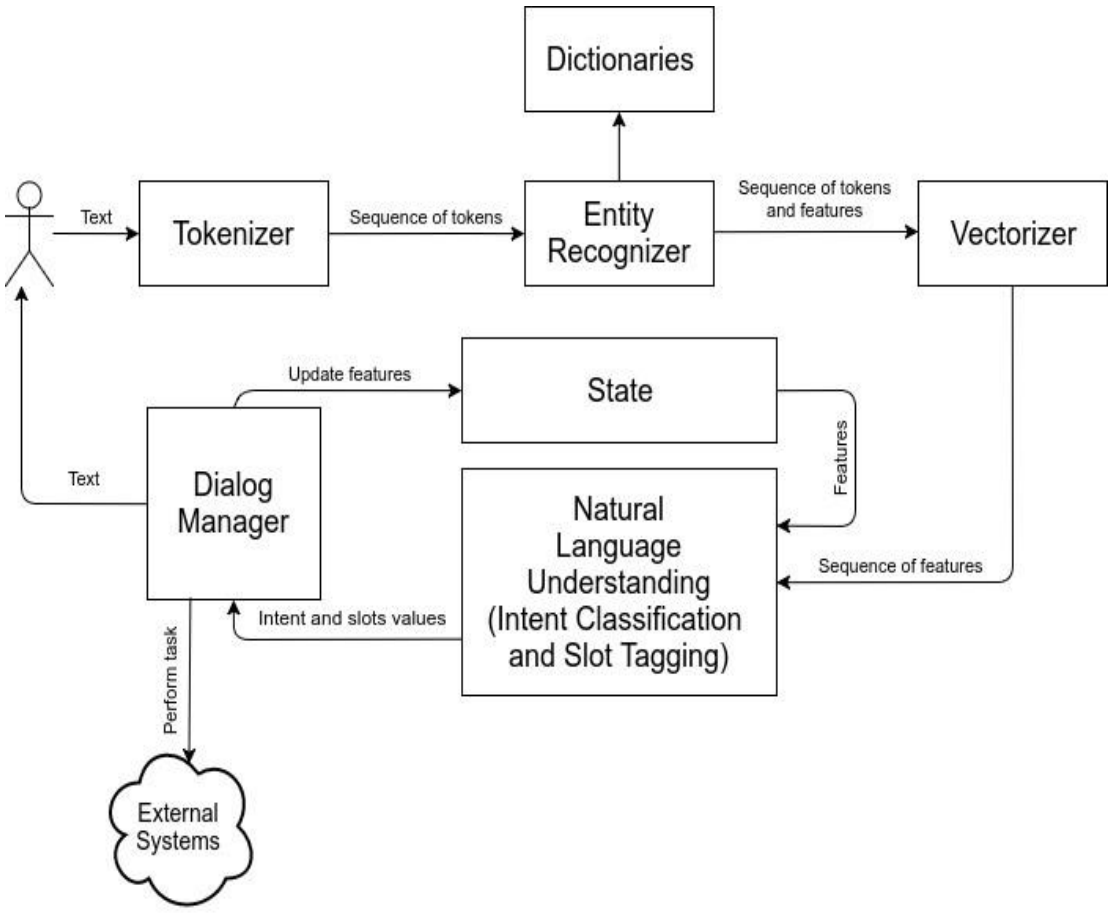


Рисунок 2.5 – Приклад архітектури чат-боту

При різноманітності архітектур чат-ботів можна узагальнити архітектуру, що буде включати наступні основні компоненти [18]:

1) модуль обробки природної мови (Natural Language Processing, NLP), який відповідає за розпізнавання та розуміння мови користувача і який може включати в себе різноманітні техніки, такі як токенізація, аналіз синтаксису, визначення іменованих сутностей, визначення інтентів та інше;

2) діалоговий модуль:

– правила та шаблони: деякі чат-боти використовують прості правила та шаблони для визначення відповідей на конкретні запитання чи команди;

– машинне навчання: інші чат-боти, що використовують складніший підхід, можуть використовувати методи машинного навчання для навчання моделей, як реагувати на різні сценарії;

3) система управління діалогом:

– контекст: збереження контексту розмови дозволяє чат-боту краще розуміти та відповідати на послідовні запитання користувача.

– станові машини: використання станових машин для моделювання різних станів діалогу та переходів між ними;

4) модуль взаємодії з користувачем:

– генерація відповідей: створення текстових відповідей для користувача залежно від розпізнаного інтенту та контексту;

– відображення та інтерфейс: подача відповідей на платформі чат-бота або взаємодія з іншими інтерфейсами;

5) система збереження даних та аналітики:

– збереження стану: зберігання стану діалогу та інших корисних даних для поліпшення якості обслуговування в майбутньому;

– аналіз взаємодії: використання аналітики для вдосконалення взаємодії та адаптації до потреб користувачів.

Існує декілька різновидів моделей чат-ботів, які можна класифікувати залежно від способу їхньої побудови та роботи [19-22]:

2.2.1 **Правила та Шаблони**: це прості моделі, які використовують фіксовані правила або шаблони для визначення, як реагувати на конкретні запитання чи команди. Зазвичай, ці боти ефективні в областях з обмеженим набором можливих варіантів взаємодії. До переваг слід віднести: простота, легкість розгортання, а до недоліків – обмежена гнучкість та неефективність у вирішенні складних завдань. Модель у таких чат-ботах побудована на правилах та шаблонах, зазвичай є простою та обмеженою, але вона може бути ефективною для конкретних сценаріїв взаємодії, де можна передбачити варіанти запитань і відповідей. Такий чат-бот визначає відповіді на основі фіксованих правил чи використання шаблонів. Один з прикладів – чат-бот, який надає інформацію про розклад автобусів. Розглянемо простий приклад на мові Python, використовуючи бібліотеку python-telegram-bot, яку можна встановити, використовуючи pip install python-telegram-bot.

```
    from telegram import Update
    from telegram.ext import Updater, CommandHandler, MessageHandler, Filters, CallbackContext

    # Список шаблонів та відповідей
    templates = {
        'розклад': 'Розклад автобусів: Понеділок - П'ятниця: 8:00, 12:00, 16:00. Субота - Неділя: вихідний.',
        'контакт': 'Наш контактний номер: +123456789.',
        'привіт': 'Привіт! Я бот з розкладом автобусів. Як я можу вам допомогти?'
    }

    # Обробник команди /start
    def start(update: Update, context: CallbackContext) -> None:
        update.message.reply_text(templates['привіт'])
    # Обробник повідомлень
    def handle_messages(update: Update, context: CallbackContext) -> None:
        user_input = update.message.text.lower()

        # Пошук відповіді за шаблоном
        for keyword, response in templates.items():
            if keyword in user_input:
                update.message.reply_text(response)
                return

        # Відповідь, якщо шаблон не знайдено
        update.message.reply_text('Вибачте, я не розумію ваш запит. Спробуйте інший запитання.')

    def main() -> None:
```

```
    # Створення об'єкта Updater та встановлення токену бота
    updater = Updater("YOUR_BOT_TOKEN")

    # Отримання об'єкта Dispatcher
    dp = updater.dispatcher

    # Додавання обробників команд та повідомлень
    dp.add_handler(CommandHandler("start", start))
    dp.add_handler(MessageHandler(Filters.text & ~Filters.command, handle_messages))

    # Запуск бота
    updater.start_polling()

    # Зупинка бота при натисканні Ctrl+C
    updater.idle()

if __name__ == '__main__':
    main()
```

У цьому прикладі бот розпізнає команду /start і вітає користувача. Крім того, він обробляє повідомлення, шукає ключові слова (шаблони) та відповідає відповідно до них. Якщо введене повідомлення не відповідає жодному шаблону, бот повідомляє користувача, що не розуміє запитання.

2.2.2 **Моделі на основі правил та машинного навчання**: використовують комбінацію правил та методів машинного навчання для кращого розуміння та обробки введених користувачем запитань. До переваг слід віднести: покращена гнучкість порівняно з простими правилами, а до недоліків – потребують великої кількості навчальних даних та ручної роботи при налаштуванні. Модель чат-бота, яка використовує комбінацію правил та машинного навчання, може бути більш гнучкою, оскільки вона може вивчати зразки відповідей з навчальних даних. Давайте розглянемо приклад простого чат-бота на основі правил та машинного навчання, використовуючи бібліотеку ChatterBot у мові Python.

```
from chatterbot import ChatBot
from chatterbot.trainers import ChatterBotCorpusTrainer

# Створення об'єкта чат-бота
bot = ChatBot('MyBot')

# Створення об'єкта тренера та вибір навчального джерела (ChatterBot корпус)
trainer = ChatterBotCorpusTrainer(bot)
trainer.train('chatterbot.corpus.english')  # Навчання на англійському корпусі
```

```
# Додавання додаткових правил
bot.set_trainer(ChatterBotCorpusTrainer)
bot.train([
    'How are you?', 'I am good, thank you!',
    'What is your name?', 'My name is MyBot.'
])

# Взаємодія з користувачем
while True:
    user_input = input("You: ")
    response = bot.get_response(user_input)
    print("Bot:", response)
```

У цьому прикладі ChatterBot використовує корпус для навчання базових відповідей. Також можна додавати власні правила для більш специфічної відповіді. Цей підхід дозволяє створювати чат-бота, який може взаємодіяти з користувачем на основі вивчених взорів, але він залишається обмеженим тим, що він може вивчити з навчальних даних і вручну доданих правил. Цей код представляє лише простий приклад, і в реальних застосуваннях можливі використання більш складних моделей та штучних нейронних мереж для покращення якості відповідей чат-бота.

2.2.3 **Моделі на основі глибокого навчання**: використовують глибокі нейронні мережі для розуміння та генерації текстової інформації. Можуть включати рекурентні нейронні мережі (RNN), трансформери та інші архітектури. До переваг слід віднести: здатність до автоматичного вивчення вищих рівнів представлення та контексту, а до недоліків – вимагають значних обчислювальних ресурсів та великої кількості даних для навчання. Модель чат-бота на основі глибокого навчання може використовувати глибокі нейронні мережі, такі як рекурентні нейронні мережі (RNN), трансформери чи інші архітектури. Розглянемо приклад простого чат-бота, який використовує глибокі нейронні мережі для генерації відповідей, використовуючи бібліотеку TensorFlow у мові Python.

```
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Дані для навчання
conversations = [
    ("Hello", "Hi there!"),
```

```
        ("What's your name?", "I am a chatbot."),
        ("How are you?", "I'm good, thank you."),
        ("Bye", "Goodbye!")
    ]

    # Розпаковка даних
    questions, answers = zip(*conversations)

    # Ініціалізація токенізатора
    tokenizer = Tokenizer(oov_token="<OOV>")
    tokenizer.fit_on_texts(questions + answers)

    # Кількість унікальних слів
    vocab_size = len(tokenizer.word_index) + 1

    # Перетворення тексту в послідовності
    questions_sequences = tokenizer.texts_to_sequences(questions)
    answers_sequences = tokenizer.texts_to_sequences(answers)

    # Доповнення послідовностей до одного розміру
    max_len = max(max(map(len, questions_sequences)), max(map(len, answers_sequences)))
    padded_questions = pad_sequences(questions_sequences, padding='post', maxlen=max_len)
    padded_answers = pad_sequences(answers_sequences, padding='post', maxlen=max_len)

    # Модель глибокого навчання
    model = tf.keras.Sequential([
        tf.keras.layers.Embedding(vocab_size, 64, input_length=max_len),
        tf.keras.layers.LSTM(128),
        tf.keras.layers.Dense(vocab_size, activation='softmax')
    ])

    # Компіляція та навчання моделі
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', met-
rics=['accuracy'])
    model.fit(padded_questions, padded_answers.argmax(axis=2), epochs=50)

    # Функція для генерації відповіді
    def generate_response(input_text):
        input_seq = tokenizer.texts_to_sequences([input_text])
        padded_input = pad_sequences(input_seq, padding='post', maxlen=max_len)
        predicted_index = model.predict_classes(padded_input, verbose=0)[0]
        return tokenizer.index_word.get(predicted_index, "<OOV>")

    # Взаємодія з користувачем
    while True:
        user_input = input("You: ")
        if user_input.lower() == 'exit':
            break
        response = generate_response(user_input)
        print("Bot:", response)
```

Цей приклад використовує Embedding-шар, LSTM (Long Short-Term Memory) шар і Dense-шар для створення моделі глибокого навчання. Зверніть увагу, що цей код - простий приклад і може бути вдосконалений для отримання кращих результатів. Також, для створення більш складних чат-ботів, може бути використана модель типу GPT (Generative Pre-trained Transformer) або інші сучасні архітектури.

2.2.4 **Генеративні моделі**: використовують глибокі нейронні мережі для генерації текстового контенту відповідно до запитань чи команд користувача. Зазвичай використовують трансформери та механізми уваги. До переваг слід віднести: здатність генерувати творчий та контекстуальний текст, а до недоліків – великі вимоги до обчислювальних ресурсів, можливість генерації неправильної чи небажаної інформації. Блок-схема роботи генеративної моделі представлена на рисунку 2.6

Рисунок 2.6 – Блок-схема роботи генеративної моделі

Генеративна модель (Generative Model): Це клас моделей в машинному навчанні, які призначені для генерації нових даних, що схожі на ті, які вони бачили під час навчання. Одним з прикладів генеративних моделей є рекурентні нейронні мережі (RNN), які використовуються для генерації послідовностей тексту.

У контексті чат-ботів, генеративні моделі можуть використовуватися для створення текстуальних відповідей на запитання користувачів, особливо в ситу-

аціях, де може бути кілька правильних або відкритих відповідей. Однак, такі моделі можуть вимагати значних обчислювальних ресурсів та великих обсягів даних для навчання.

Однією з відомих генеративних моделей є GPT (Generative Pre-trained Transformer) від OpenAI, яка демонструє високий рівень текстового генерування та розуміння контексту.

2.2.5 **Асистенти з розпізнавання інтентів**: спрямовані на визначення інтентів (намірів) користувача та надання відповідей на їх запитання. Можуть використовувати як правила, так і машинне навчання. До переваг слід віднести: ефективні в розпізнаванні основного наміру, а до недоліків – можуть бути обмеженими в розумінні контексту та неочікуваних запитань. Модель чат-бота, яка спрямована на розпізнавання інтентів (намірів) користувача, може використовувати комбінацію правил і машинного навчання для визначення того, що користувач хоче від чат-бота.

Розглянемо приклад простого чат-бота на основі правил та машинного навчання для розпізнавання інтентів, модель якого представлена на рисунку 2.7.

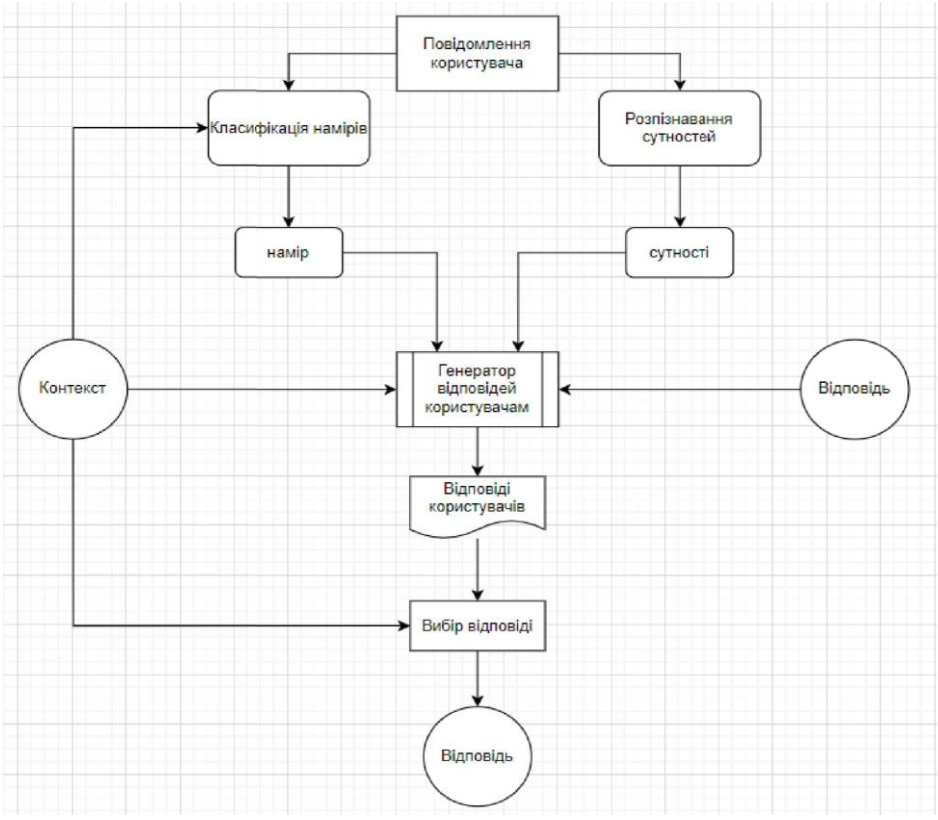


Рисунок 2.7 – Модель чат-бота, яка спрямована на розпізнавання інтентів

Введені текстові повідомлення можете і бачити, як чат-бот розпізнає інтенти, та як надає відповіді відповідно до вашої конфігурації та тренувальних даних.

Ці моделі можуть використовуватися як індивідуально, так і в комбінації, залежно від потреб конкретного чат-бота та сценаріїв взаємодії з користувачами.

2.3 Аналіз варіантів розробки чат-ботів

Існує декілька методів для створення чат-ботів, і вони можуть варіюватися в залежності від рівня технічної компетентності, бажаного рівня контролю та величини проекту [19-22]. Розглянемо існуючі на теперішній час підходи.

2.3.1 Використання **платформ** для створення ботів [23]:

– Telegram Bot API, BotFather: Telegram надає власний API для створення ботів. За допомогою BotFather, ви можете швидко створити базового бота та отримати токен для використання в API;

– Chatfuel, Dialogflow, Wit.ai: Платформи, такі як Chatfuel, Dialogflow від Google, та Wit.ai від Facebook, дозволяють створювати чат-ботів з використанням інтерфейсу без коду. Вони використовуються для розпізнавання мови та створення інтелектуальних ботів. Основною перевагою такого способу є максимальна висока швидкість створення, а головним недоліком – обмежений функціонал. Приклад створення бота за шаблоном показано на рисунку 2.8;

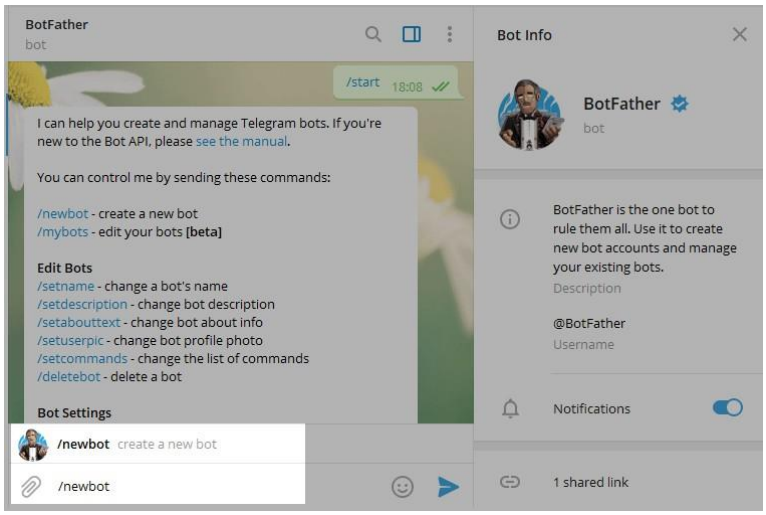


Рисунок 2.8 – Приклад створення бота за шаблоном

2.3.2 Створення за допомогою **конструктора** (можуть виконувати певні бізнес-функції). Це набори готових інструментів, які дозволяють створювати та налаштовувати ботів без технічних навичок та розробників – будь-який користувач може зайти на сервіс конструктора та розробити в ньому віртуального помічника. Конструктори бувають low-code (де трохи треба розумітись на програмуванні) та no-code (де взагалі не треба навичок програмування) [19,23].

Для створення сценарію використовуються візуальні блоки, з яких користувач становить логіку роботи свого рішення. У блоках вказуються дії бота і очікувані дії користувачів, наприклад, наміри користувачів (інтенти), і це з'єднується між собою у потрібній послідовності. Коли сценарій та логіка готові, бота можна підключити до соцмереж та месенджерів, інтегрувати з CRM та зовнішніми системами, навчати потрібним діям: наприклад, перекладати діалог на оператора за певної події.

Перевагою такого способу є можливість створювати ботів для бізнесу, а основними недоліками є те, що прийдеться щомісячно платити за бота, функціонал може бути обмежений конструктором. Конструктори мають низький поріг входу в розробку, для роботи з ними не треба вміти програмувати. Швидкість створення боту залежить від того, що саме необхідно користувачу та обсяг необхідних функцій. У однієї людини це може бути декілька запитань, в іншої - декілька сотень блоків з прописанною логікою, або переліком товарів.

Завдяки конструкторам зазвичай створюють простих F.A.Q.-ботів, або ботів, що реалізують певні бізнес-ідеї (збір інформації про клієнтів, розсилка повідомлень, продаж товарів, тощо). Приклад сервісу-конструктора чат-ботів наведено на рисунку 2.9.

2.3.3 Створення через **діалогові платформи.** Це більш складна версія конструктора, вона вміщує в себе сервіси з: прототипування, розробки, тестування, розгортання, контролю якості, інтеграції з зовнішніми системами, зберіганням логів та розміщенням на сервері (див.рис.2.10). Дані платформи можна назвати об’єднаним компонентом всього чат-боту, вони використовуються тими, хто вже має навички програмування або працює в команді, що виконує проект .

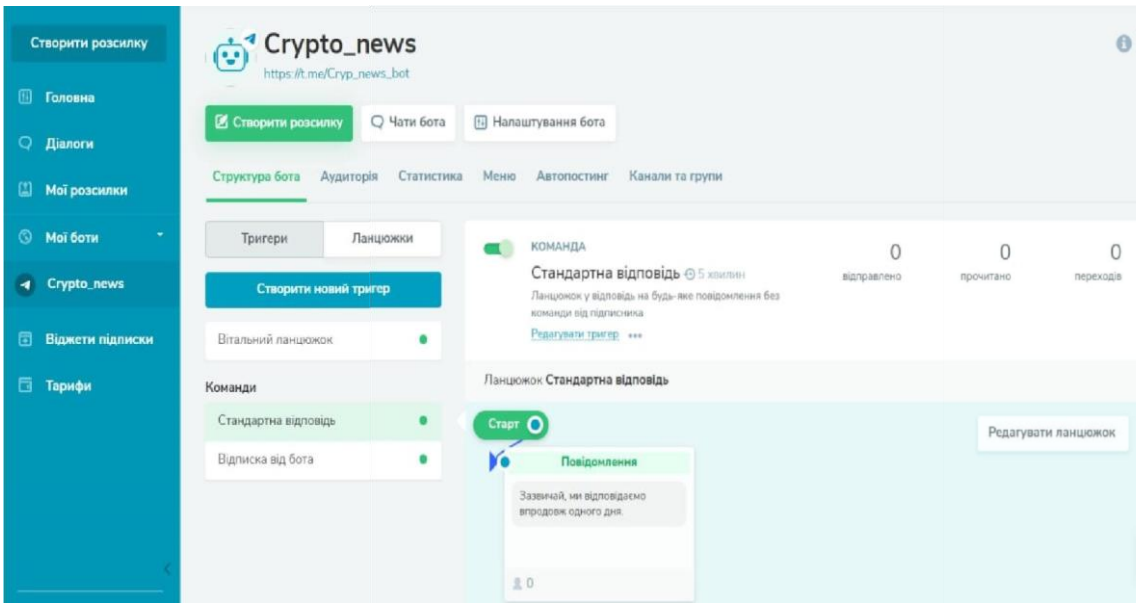


Рисунок 2.9 – Приклад сервісу-конструктора чат-ботів

Від конструкторів вони відрізняються підвищеним функціоналом, а саме: є можливість пропрацювати більш складні сценарії, вирішувати нестандартні ситуації та створювати чат-ботів з кращою логікою. Ця можливість є завдяки тому, що такі платформи комбінують в собі нижчий та вищий рівні розробки: конструктора та написання коду власноруч [19,23].

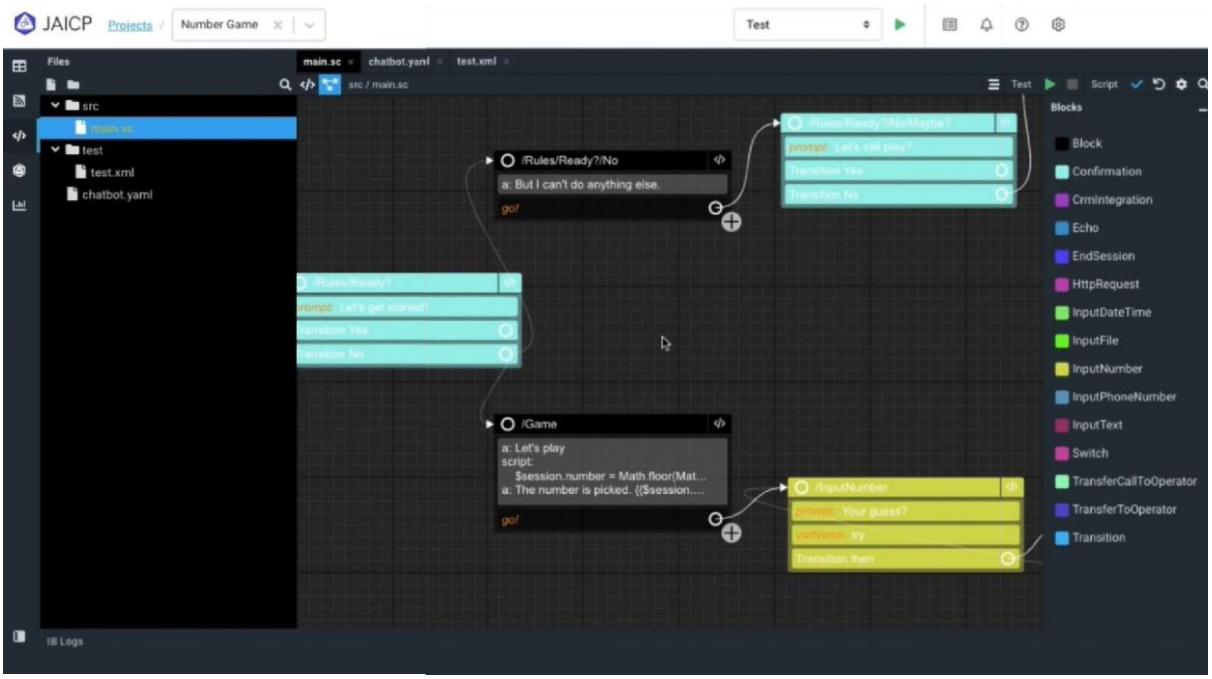


Рисунок 2.10 – Приклад діалогової платформи

В платформах можна розробити багато ботів різного рівня складності, але найчастіше вони використовуються для створення голосових асистентів або консультантів, які можуть обдзвонити базу користувачів або продиктувати код для підтвердження [19,23].

2.3.4 **Програмування** власного бота:

– **Python (з використанням бібліотеки Telebot, python-telegram-bot)**. Якщо є досвід програмування, то можна створити свого бота використовуючи мову програмування Python та відповідні бібліотеки для роботи з Telegram API.

– **Node.js (з використанням бібліотеки Telegraf).** Інша популярна мова програмування для створення чат-ботів – JavaScript. З використанням бібліотеки Telegraf для Node.js, ви можете легко розробляти Telegram-ботів [19,23].

2.3.5 **Фреймворки** – це самий популярний засіб створити чат-бота. Це набір інструментів з відкритим вихідним кодом та високим ступенем контролю над розробкою, який інтегрується з NLU (Natural Language Understanding) для створення складного сценарію в рамках власного розмовного рішення та розв'язання різних завдань. Їх відмінність від діалогових вікон в тому, що вони мають відкритий вихідний код, який дозволяє реалізувати ще більш обширний функціонал, тобто розробка часто обмежується можливостями мови програмування.

Код, написаний за допомогою фреймворку, необхідно розмістити в середовищі виконання самостійно. Для цього можна використовувати власні сервери в хмарі або контурі або скористатися платформою, яка візьме на себе всі завдання з хостингу, масштабування і балансування. Те саме стосується NLU-моделей. Якщо сценарій віртуального асистента використовує у своїй роботі розуміння природної мови, такі NLU-моделі також необхідно розміщувати серед виконання і, відповідно, масштабувати і балансувати навантаження.

За допомогою цього способу можна реалізувати будь-який функціонал бота:

а) вдосконалення за допомогою **ШІ (Штучного Інтелекту)**:

– **Natural Language Processing (NLP)**: додавання можливостей розпізнавання природної мови дозволяє ботам розуміти та відповідати на складні

команди та запитання користувачів. Популярні сервіси для NLP: Dialogflow, Wit.ai, Microsoft LUIS;

– **Machine Learning**: застосування методів машинного навчання може покращити роботу бота, особливо в області аналізу та передбачення поведінки користувачів.

– **чат-боти з підтримкою інтелектуальних обчислень**: використання методів ШІ для вдосконалення взаємодії та вирішення складних завдань.

Незалежно від методу, важливо продумати архітектуру бота, забезпечити зручний та логічний діалог з користувачем, а також захищати дані та забезпечувати безпеку взаємодії.

Щодо вдосконалення чат-ботів за допомогою ШІ, важливо вдосконалювати алгоритми розпізнавання мови, оптимізувати обробку великих обсягів даних, та впроваджувати технології машинного навчання для покращення розуміння та реакції на запитання користувачів. Також, можливості аналітики можуть допомогти виробникам ботів розуміти попит та вдосконалювати функціональність.

Отже, способи розробки ботів залежать від задач, які ставляться до цих самих ботів. Чим складніше функціонал – тим більше ресурсів повинен мати його власник, і навпаки. Але це не значить те, що деякі способи не повинні мати права на існування. Вони забезпечують економію часу, грошей та можуть допомогти людині створити віртуального помічника для власного бізнесу, тощо.

2.4 Аналіз підходів до розробки чат-ботів

На теперішній час існує три методи створення чат-ботів [19-22]:

1) засновані на **бізнес-правилах (заготований сценарій)**:

– підхід використовує заздалегідь визначені сценарії та правила для взаємодії з користувачем;

– бізнес-правила визначають, як бот повинен реагувати на конкретні запитання або команди користувачів;

2) засновані на **NLP (штучний інтелект)**:

– цей підхід використовує штучний інтелект та обробку природної мови (NLP) для розуміння та відповіді на запитання користувачів;

– бот може аналізувати мовлення, розпізнавати інтенції та взаємодіяти з користувачем більш природоподібно;

3) **гібридні (комбінація обох видів)**:

– цей підхід поєднує в собі елементи бізнес-правил і технології штучного інтелекту;

– гібридні чат-боти можуть використовувати стандартні правила для простих запитань і відповідей, а також використовувати NLP для більш складних взаємодій.

Детальніше проаналізуємо кожний з цих підходів, щоб краще зрозуміти їхню сутність.

2.4.1 Підхід заснований на бізнес-правилах, ґрунтується на тому, що чат-боти використовують структуру діалогу у вигляді дерева (див.рис.2.11). Це означає, що їхнє спілкування із користувачем обмежується заздалегідь визначеним сценарієм, який був розроблений програмістом.

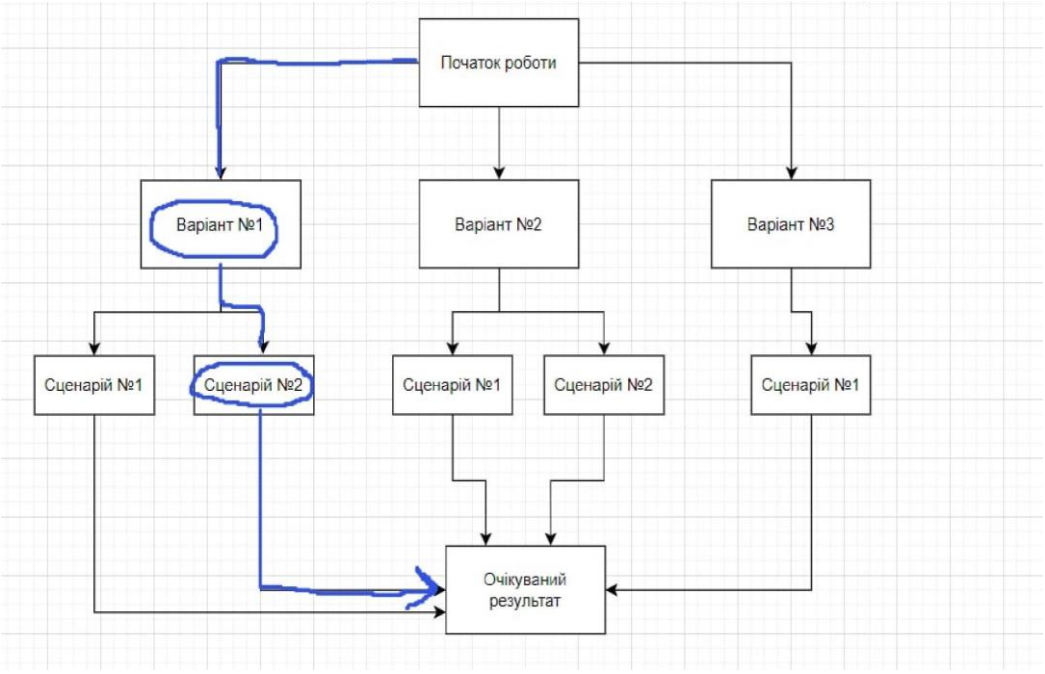


Рисунок 2.11 – Приклад способу розробки чат-боту, заснованого на бізнес правилах

Користувач, виступаючи як головний учасник, приймає рішення, але не може відхилитися від певного шляху. Зазвичай такі чат-боти уникають від відповідей у вільній формі та надають безліч кнопок як альтернативний спосіб взаємодії [19-23].

2.4.2 Підхід, заснований на NLP (Natural Language Processing, обробка природної мови) є галуззю інформатики та штучного інтелекту, яка вивчає, як комп'ютери аналізують природні (людські) мови. NLP дозволяє використовувати алгоритми машинного навчання для обробки тексту та мовлення. Наприклад, це може бути застосовано для створення систем, які розпізнають мову, узагальнюють документи, виконують машинний переклад, виявляють спам, розпізнають іменовані сутності, відповідають на питання, надають автокомплітацію, забезпечують предиктивний ввід тексту тощо [19,22].

В сучасний час у багатьох людей є смартфони, які використовують розпізнавання мови з використанням NLP для розуміння нашої мови. Крім того, багато людей користуються ноутбуками з вбудованим у операційну систему розпізнаванням мови.

Нові, більш розумні чат-боти використовують глибоке навчання як для аналізу введення людської мови, так і для генерування відповідей. Глибоке навчання застосовується у декодуванні введення та генеруванні відповідей, забезпечуючи аналіз та формування відповідей. Крім того, NLP переводить введення та виведення у текстовий формат, який зрозумілий і комп'ютеру, і людині.

При розгляді простіших чат-ботів будь-яка надана відповідь буде лінгвістично бездоганною, оскільки бот користується готовими пропозиціями з бази даних. Такий чат-бот не може обробити інформацію, якщо користувач допустив граматичні помилки або висловився так, що не збігається із заданими шаблонами. Сучасніші та розумніші чат-боти проходять "навчання" для розпізнавання природної мови та реагування відповідно до конкретних ситуацій. На жаль, здатність відповідати природним чином вимагає значних витрат часу на навчання і великої кількості даних для освоєння різноманітних можливих запитань [19,22].

Подальше навчання визначить, наскільки чат-боти можуть впоратися з серйозними викликами, які представляють виклик для їхньої роботи порівняно з простішими моделями. Залежно від контексту розмов, вони можуть бути як короткими, так і тривалими. Розмови, що тривають довше, зазвичай мають більше глибини і включають численні питання, які чат-бот повинен врахувати для створення комплексного зображення (див.рис.2.12).

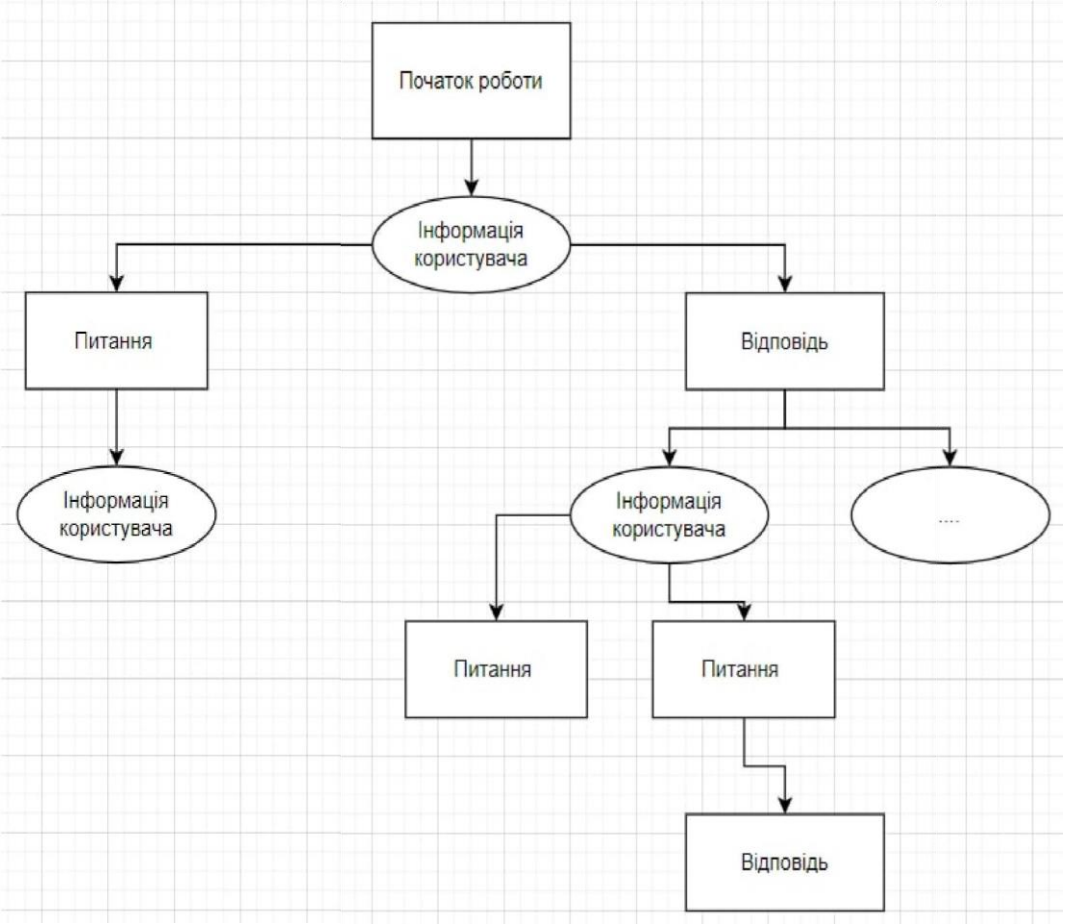


Рисунок 2.12 – Приклад блок-схеми чат-боту, заснованого на NLP

До завдань, які потрібно вирішувати за допомогою штучної обробки мови, причому багато з них може бути пов'язано із розпізнаванням тексту, мови та навіть зображень, входять:

– реферування (створення реферату чи резюме з обширного тексту);

– відкриті та закриті запитання (здатність надавати на них відповіді);

– зіставлення (можливість бота асоціювати об'єкти зі словами та розуміти,

коли кілька слів вказують на один об'єкт);

– двозначність;

– морфологія (здатність бота розкладати слова на морфеми, враховуючи особливості морфології кожної мови);

– семантика (забезпечення бота здатністю розуміти людську мову, аналізувати запити та генерувати відповіді для них);

– структура тексту;

– тональність (чат-бот повинен розуміти відношення людини до певного об'єкту за її емоційним вираженням).

2.4.3 Комбінований спосіб, що поєднує в собі два вище описаних способи, має найбільш обширний функціонал через те, що людина може як сама задавати дані, так і вибрати необхідну кнопку (див.рис.2.13).

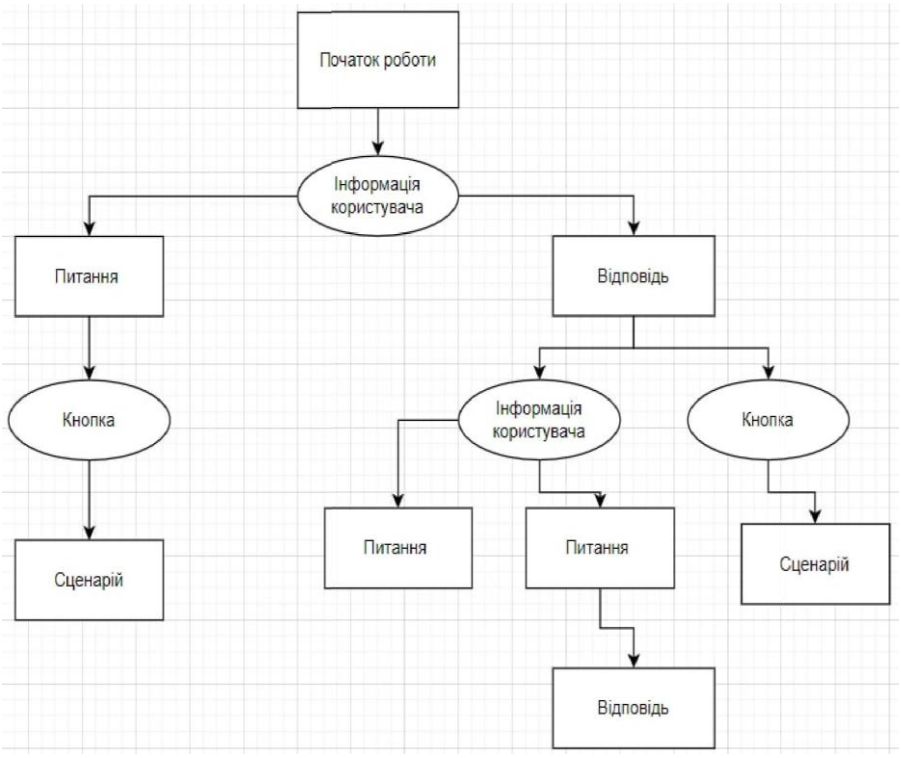


Рисунок 2.13 – Приклад блок-схеми комбінованого чат-боту

Підводячи підсумок, можна сказати що найбільш оптимальним способом є метод, заснований на бізнес-правилах. Він здатен реалізувати більшість задач, не

потребує обширного багажу знань та може створюватись достатньо швидко. Перевагами двох інших методів є гнучкість, здатність вирішувати складні завдання та інтерактивність. Але при використанні цих способів треба витратити значну кількість ресурсів.

3 СТРУКТУРНЕ ПРОЄКТУВАННЯ ВІРТУАЛЬНОГО ПОМІЧНИКА

3.1 Розробка системних та функціональних вимог до віртуального помічника

Є клієнт тимчасового прихистку для тварин, який хоче влаштувати свою тварину на перетримку на певний час. Для автоматизації надання інформації від користувача для оформлення заявки, також для пришвидшення цієї процедури та гарантування правильності введення даних вирішено створити чат-бот віртуального помічника (див.рис.3.1).

Рисунок 3.1 – Схема взаємодії клієнта із чат-ботом

Чат-бот повинен отримувати необхідну інформацію для консультацій з різних джерел. Розробник додає обмежену кількість даних в меню чи кнопки, також може вручну вводити список номерів чи видів тварин. Для обсягової або змінюваної інформації рекомендується використовувати парсинг. Наприклад, парсер може автоматично оновлювати розклад бронювання номерів або інші дані за заданим періодом або за запитом користувача. Таким чином, два основні джерела інформації в боті – ручне додавання розробника та можливість оновлення через парсер.

Основні компоненти системи віртуального помічника включають в себе:

– чат-бот;

– парсер;

– сховище даних;

– адмін-панель.

Кожен із перелічених компонентів виконує своє власне завдання в системі:

– чат-бот виконує основний функціонал, який визначає користувач. В даному випадку він приймає дані від користувача, які необхідні для оформлення заявки на надання прихистку, та перевіряє дані на коректність. Також бот надає інформацію, як зв’язатися із самим прихистком;

– парсер (див.рис.3.2) — це програма або скрипт, який використовується для обробки та аналізу структурованого або неструктурованого текстового контенту. Основна мета парсера - виділення конкретних даних з тексту, щоб їх можна було легко обробити або використовувати в інших програмах. У контексті розробки чат-ботів, парсер може використовуватися для збору та систематизації інформації з різних джерел, таких як веб-сайти чи бази даних, наприклад, чи зайняті номери та чи є наявні вільні номери в прихистку;

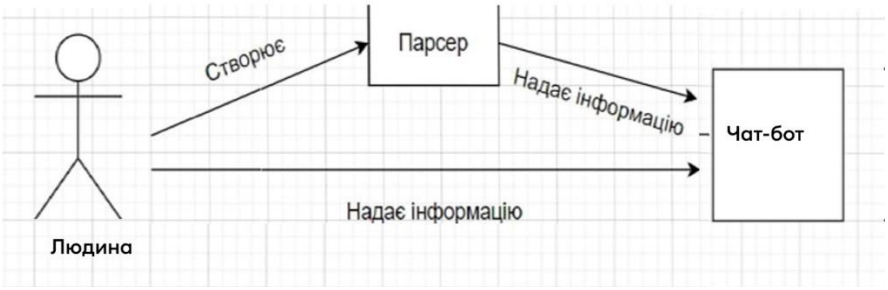


Рисунок 3.2 – Більш детальне відображення взаємодії клієнта з компонентами віртуального помічника

– сховище даних (або БД) містить всю інформацію, що надходить до чат-боту, та зберігає її;

– адмін-панель реалізує інтерфейс управління, який надається адміністраторам або уповноваженим користувачам для керування та контролю за певною системою, програмою чи веб-сайтом. У випадку створення системи

чат-бота, адмін-панель може надавати можливості керування вмістом, встановлення правил і налаштувань, аналізу даних та інші адміністративні функції, необхідні для ефективного управління чат-ботом.

3.2 Розробка моделі потоків даних віртуального помічника

Наступним кроком необхідно зрозуміти, що повинен містити в собі чат-бот, для того, щоб надавати необхідну консультацію користувачу. Для того, щоб оформити заявку на прихисток тварини, необхідно надати такі дані з боку користувача:

– ПІБ власника;

– Ім’я тварини;

– вид тварини;

– дата заселення;

– кількість днів заселення;

– особливі коментарі.

Більшість цієї інформації варто додавати способом вручну, та певну інформацію (наприклад дати та вільні номери) додавати за допомогою парсингу даних з БД сайту прихистка. Також слід зазначити, що тип номеру залежить від виду тварини, який обумовлює розмір номеру.

Для розроблюваної ІС були виділені три типи дійових осіб:

1) клієнт н/р (користувач без облікового запису в ІС);

2) клієнт з/р (користувач з обліковим записом в ІС);

3) адміністратор (робітник прихистку, який регулює процеси у системі).

Замовлення може оформлювати виключно зареєстрований користувач, тому надалі будемо розглядати виключно ролі Клієнт з/р та Адміністратор. Діаграма прецедентів для зареєстрованого користувача представлена на рисунку 3.3.

Для ІС прихисток тварин вже була розроблена у кваліфікаційній роботі на здобуття ступеня «бакалавр» база даних, яка представлена на рисунку 3.4.

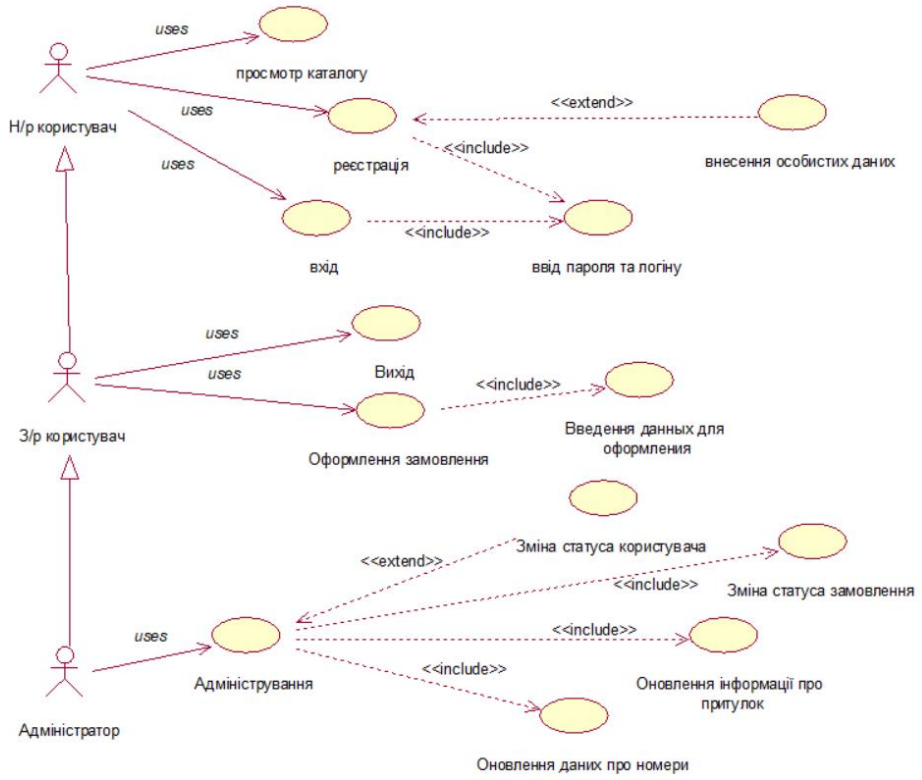


Рисунок 3.3 – Діаграма прецедентів для зареєстрованого користувача

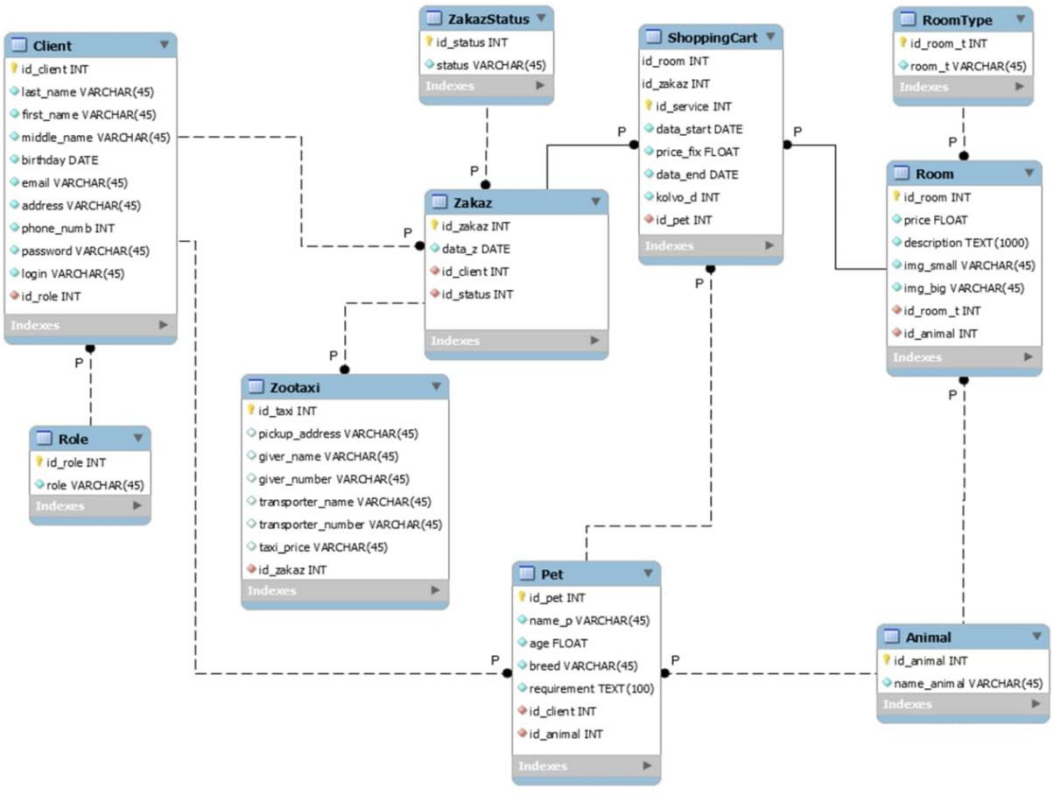


Рисунок 3.4 — Схема бази даних для платформи MySQL Server

За допомогою чат-бота заповнюються наступні атрибути сутності:

1) дані користувача, зареєстрованого в системі, підтягуються з сутності «Client» та заповнюються при реєстації користувача в системі, як і дані про тварину – у сутність «Pet». Дані в Сутность «Status», яка призначена для зберігання даних про статус замовлення в системі, заповнюються адміністратором після схвалення заявки на прихисток;

2) дані, які безпосередньо надає клієнт при оформленні заявки, заносяться у сутність «Zakaz», яка призначена для зберігання даних про замовлення в системі;

3) вид тварини обумовлює можливі значення атрибуту Сутність «id_room_t» (домен «Int» – унікальний ID типу номеру сутності «Room», яка призначена для зберігання даних про номери), серед яких: великий номер для середніх ссавців, середній номер для середніх ссавців, малий номер відповідно для малих ссавців, тераріум великий для змій, тераріум малий для черепах та ящірок, велика клітка для великих та середніх папуг, мала клітка для малих птахів та малих папуг.

Опис прецеденту «Оформлення замовлення за допомогою чат-бота» та прецеденту «Зміна статусу замовлення» за методологією RUP надано відповідно у таблиці 3.1 та таблиці 3.2.

Таблиця 3.1 – Опис прецеденту «Оформлення замовлення за допомогою чат-бота» за методологією RUP

Прецедент	«Оформлення замовлення»
Дійові особи	Клієнт. Бажає швидко оформити заявку на прихисток тварини та отримати підтвердження. Адміністратор. Бажає внести точні дані без помилок.
Цілі	Оформлення замовлення на бронь номеру.
Передумови	Користувач успішно авторизувався в системі.
Успішний сценарій: 1. Зареєстрований користувач відкриває чат-бот для швидкого подання заявки на прихисток тварини.	

Продовження таблиці 3.1

2. Користувач вносить дані про тварину, свої ПІБ та доставку. 3. Система перевіряє дані про дату початку прихистку тварини. 3.1 Якщо дані вірні, далі продовжується введення даних користувачем 3.2 Якщо дані не вірні, то система видає помилку у вигляді текстового повідомлення «Дата невірна, оберіть іншу» 4. Користувач вносить дані про тип тварини. 5. Система, в залежності від видц тварини та дати прихистка, перевіряє, чи є вільні номери.5.1 Якщо номер вільний, система видає всі введені дані про заявку на прихисток. 5.1.1 Якщо користувач згоден, натискає «Підтвердити». 5.1.2 Якщо користувач не згоден, натискає «Помилка». 5.2 Якщо номер зайнятий, то система видає помилку у вигляді текстового повідомлення «Номери зайняті, оберіть іншу або зв'яжіться з адміністратором» 6. Система формує текстовий файл із даними про замовлення та надсилає на пошту прихистку.	
Результат	Дані про замовлення збережені. Користувач успішно створив замовлення на бронювання.
Розширення:	
*а	При кожній помилці введених даних: 1. Система виводить повідомлення про помилку 2. Система відновлює попередній стан. 3. Користувач вводить дані заново.

Таблиця 3.2 – Опис прецеденту «Зміна статусу замовлення» за методологією RUP

Прецедент	«Зміна статусу замовлення»
Дійові особи	Адміністратор. Бажає внести точні дані без помилок.
Цілі	Змінити статус замовлення.
Передумови	Адміністратор успішно авторизувався в системі.
Успішний сценарій: 1. Адміністратор перевіряє пошту на наявність нових замовлень. 1. Адміністратор заходить в свій акаунт на сайті прихистка. 2. Адміністратор переходить на панелі за посиланням «Адміністрування». 3. Адміністратор редагує таблицю з замовленнями. Всі нові замовлення за замовченням мають статус «Нове». 4. Адміністратор зв'язується з клієнтом та дані. Якщо користувач підтвердив замовлення, Адміністратор змінює статус замовлення на «Підтверджене».	

Продовження таблиці 3.2

5. Після закінчення терміну прихистку та тварину вже забрали, адміністратор має змінити статус замовлення на «Закінчено». 6. Адміністратор виходить з системи.	
Результат	Статус замовлення на бронювання змінений успішно без помилок.
Розширення:	
4а	Адміністратор уточнює дані замовлення з клієнтом. Якщо клієнт відмовив, адміністратор видаляє замовлення зі списку замовлень.
4б	Клієнт робить запит для скорочення строку прихистку. Адміністратор видаляє замовлення зі списку замовлень. Клієнт заново оформлює замовлення з вірними датами.
4в	Клієнт робить запит для збільшення строку прихистку. Адміністратор скасовує старе замовлення. Клієнт заново оформлює замовлення.
5а	Клієнт робить запит для дострокового повернення тварини. У цьому випадку адміністратор попереджає, що кошти клієнтові не повертаються. Статус замовлення після виїзду змінюється на «Закінчено».

В результаті цього з’являються дещо інший функціонал компонентів:

а) розробник створює парсер;

б) база даних зберігає інформацію від розробника та статичну інформацію (незмінювану) від парсеру;

в) чат-бот отримує запити від користувача та обмінюється необхідною інформацією з базою даних

в) чат-бот отримує запити від користувача та починає задіювати парсер, у випадку якщо інформація відсутня в базі даних.

З урахуванням вищенаведеного отримаємо декомпоновану схему ІС, що представлена на рисунку 3.5.

Для реалізації схеми слід виконати декілька етапів:

– заповнити попередньо дані про різновиди номерів та види тварин;

– ретельно продумати логіку чат-бота для зручної навігації;

– розробити максимально можливу кількість сценаріїв для обраних пунктів та детально проробити їх.

Рисунок 3.5 – Декомпонована схема ІС

Ефективна логіка чат-бота допомагає користувачеві швидко переміщатися між пунктами меню. Можливі два варіанти обробки логіки чат-боту: поганий та добрий. Щодо самих сценаріїв, вони повинні бути чітко сформульовані, мати конкретну мету та включати альтернативні варіанти вирішення.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ЧАТ-БОТА ТИМЧАСОВОГО ПРИХИСТКУ ТВАРИН

4.1 Вибір платформи для створення чат-бота

Обираючи платформу для створення чат-бота, важливо враховувати цільову аудиторію, функціональні вимоги та зручність використання інтерфейсу розробки. Існує багато платформ та сервісів, які надають можливості для створення чат-ботів, серед них [19,23]:

1) Facebook Messenger: Чат-боти для Facebook Messenger дозволяють підприємствам та розробникам створювати ботів для взаємодії з користувачами на платформі Facebook;

2) WhatsApp: WhatsApp Business API також дозволяє підприємствам створювати ботів для обслуговування клієнтів та ведення бізнес-комунікацій;

3) Slack: Боти в Slack призначені для автоматизації робочого процесу та спільної роботи команд в месенджері для бізнесу;

4) Microsoft Teams: Так само, як і в Slack, Microsoft Teams має можливість використовувати ботів для полегшення робочих задач та взаємодії;

5) Viber: Платформа Viber також надає інтерфейс для створення чат-ботів.

6) Telegram: також дозволяє підприємствам та користувачам створювати ботів для обслуговування клієнтів та ведення бізнес-комунікацій.

Чат-бот в Телеграмі — це програма, яка автоматично взаємодіє з користувачами через платформу Телеграм. Вона може виконувати різноманітні завдання, від відповіді на запитання та обробки команд до виконання складніших завдань, таких як взаємодія з базою даних або використання зовнішніх сервісів.

Основні характеристики чат-ботів в Телеграмі:

а) **текстова взаємодія**, яка відбувається з чат-ботом за допомогою текстових повідомлень, які користувачі вводять в чат;

б) **автоматизована обробка команд**: чат-боти реагують на спеціальні

команди або текстові запитання користувачів та виконують певні дії відповідно до програмованої логіки;

в) **інтеграція з зовнішніми сервісами**: багато чат-ботів можуть взаємодіяти з зовнішніми сервісами та API для надання користувачам різноманітних послуг та інформації;

г) **використання клавіатур та кнопок**: чат-боти можуть використовувати клавіатури та кнопки для спрощення взаємодії та навігації користувачів;

д) **можливість відправлення медіа**: чат-боти можуть обробляти та надсилати користувачам різноманітні медіафайли, такі як зображення, відео та аудіо.

Переваги використання Телеграм-бота включають:

– велика аудиторія: Телеграм має велику кількість користувачів по всьому світу, що робить його привабливим для досягнення широкої аудиторії;

– простота використання: створення Телеграм-ботів відбувається швидко та легко. Інтерфейс для розробки є зручним та дружелюбним:

– розширені можливості: Телеграм надає розширені можливості для створення ботів, такі як inline-кнопки, відправка медіафайлів, клавіатури та інше;

– безкоштовність: використання Телеграм-ботів безкоштовне, що робить його доступним для широкого кола користувачів та розробників;

– відкритий API: Телеграм надає відкритий API, який дозволяє розробникам створювати різноманітні застосунки та сервіси для взаємодії з ботами.

Для створення чат-бота в Телеграмі, щоб забезпечити успішний старт та оптимальну взаємодію з користувачами, необхідно дотримуватись наступних рекомендацій:

– визначте ціль та завдання: з'ясуйте, що саме ви хочете досягти за допомогою чат-бота. Це може бути надання інформації, обслуговування клієнтів, автоматизація завдань тощо;

– розумійте свою аудиторію: зрозумійте потреби та очікування вашої

цільової аудиторії. Розробіть чат-бота, який відповідає їхнім потребам та забезпечує високий рівень користувацького досвіду.

– обирайте відповідний інтерфейс взаємодії: використовуйте кнопки, клавіатури та інші елементи взаємодії для спрощення комунікації з користувачами;

– створюйте зрозумілий та приємний діалог: важливо мати зрозумілу та приємну конверсацію. Визначте основні кроки та забезпечте зрозумілі відповіді на питання користувачів;

– використовуйте inline-функції: якщо це можливо, використовуйте inline-функції для швидкого відповідання на запитання чи пошуку інформації прямо в тексті повідомлення;

– робіть бота інтерактивним: включайте елементи гри, опитування та інтерактивні елементи, які зроблять взаємодію з ботом цікавою для користувачів;

– будьте готові до помилок: забезпечте бота можливістю правильно реагувати на невірні або нерозпізнані команди, видаючи відповідні повідомлення чи надаючи додаткові опції;

– захищайте конфіденційність: забезпечте захист конфіденційності користувачів та дотримуйтеся стандартів безпеки;

– тестування та вдосконалення: проводьте тестування чат-бота перед випуском та вносьте вдосконалення залежно від отриманих результатів та відгуків користувачів;

– надавайте корисні функції: забезпечте чат-бота корисними та актуальними функціями для вашої аудиторії. Можливо, ви зацікавите користувачів унікальними можливостями;

– підтримуйте регулярну комунікацію: забезпечте регулярну комунікацію з користувачами через сповіщення, оновлення чи новини.

Слід пам'ятати, що розуміння потреб та вимог вашої цільової аудиторії є ключовим елементом успішного створення та використання чат-бота в Телеграмі.

Розробка чат-бота для Telegram може бути розділена на декілька етапів, які виконуються у наступній послідовності для Telegram:

1) **розуміння вимог та цілей**:

– визначте, для чого вам потрібен чат-бот і які завдання він має вирішити;

– визначте метрики успішності та функціональні вимоги;

2) **вибір технологій**:

– визначте мову програмування та технології, які ви хочете використовувати для розробки;

– оберіть фреймворк чи бібліотеку для роботи з Telegram API;

3) **створення бота в Telegram**:

– створіть нового бота в Telegram за допомогою BotFather, слідуючи інструкціям;

– отримайте токен доступу для вашого бота;

4) **налаштування оточення розробки**:

– встановіть та налаштуйте необхідне програмне забезпечення для розробки;

– зробіть ініціалізацію проекту та встановіть залежності;

5) **взаємодія з Telegram API**: використовуйте бібліотеку для мови програмування вашого вибору для взаємодії з Telegram API, використовуючи отриманий токен;

6) **реалізація основної логіки**:

– розробіть основну логіку вашого чат-бота

– обробляйте вхідні повідомлення від користувачів та надсилайте відповіді;

7) **навчання моделей (за необхідності):** якщо ви використовуєте моделі машинного навчання, навчіть їх на відповідні дані;

8) **тестування: п**роведіть тестування вашого чат-бота, переконайтеся, що він працює коректно та відповідає на очікувані повідомлення;

9) **розгортання**: розгорніть свого чат-бота на сервері чи хмарному сервісі;

10) **налаштування безпеки**:

– забезпечте безпеку чат-бота, обмежте доступ до функцій, якщо це необхідно;

– захистіть важливі дані, такі як токени доступу;

11) **підтримка та вдосконалення**:

– надавайте підтримку користувачам та вносьте необхідні виправлення та вдосконалення;

– розгляньте можливості розширення функціоналу чат-бота на майбутнє.

Цей алгоритм може слугувати загальним напрямком для розробки чат-бота в Telegram, але кожен проект може мати свої специфічні особливості та вимоги.

4.2 Вибір мови програмування

Для розробки чат-ботів використовують кілька мов програмування, таких як Python, JS, C# та інші. Вибір мови програмування для створення телеграм-бота залежить від конкретних вимог проєкту та знань, навичок і особистих вподобань розробника [23].

Порівняльний аналіз Python, JavaScript (JS) та C# з погляду їхньої придатності для розробки телеграм-ботів показує:

1) **Python**:

– *простота та легкість*: Python вважається однією з найпростіших мов програмування для вивчення та використання. Вона має простий синтаксис та велику кількість готових бібліотек, спрощуючи процес розробки;

– *бібліотеки та фреймворки*: Python має багато бібліотек, призначених спеціально для розробки чат-ботів (такі як python-telegram-bot, aiogram);

– *спільнота*: Python має активну та велику спільноту, що сприяє доступності ресурсів та підтримці;

2) **JavaScript (JS):**

– *веб-застосунки та взаємодія*: JS є основною мовою для розробки веб-застосунків, і вона широко використовується для взаємодії з елементами сторінки, включаючи телеграм-боти. Node.js дозволяє використовувати JS для серверної розробки;

– *асинхронність*: JS володіє асинхронним характером, що може бути корисним для опрацювання багатофункціональних запитів у часі реального обсягу.

3) **C#:**

– *розробка під Windows*: C# зазвичай пов'язана з платформою Microsoft, і вона є відмінним вибором для розробки під Windows. Для крос-платформеної розробки може використовуватися .NET Core;

– *інтеграція з .NET*: C# добре інтегрується з .NET, що може бути важливим для компаній, які вже використовують цей стек технологій.

Враховуючи це для вирішення завдання розробки чат-бота для прихистку тварин виберемо мову програмування Python. Python є популярним та ефективним вибором для багатьох сценаріїв чат-ботів, також ця мова програмування має достатньо інструментів для реалізації системи. Недоліком є низька швидкість роботи, але вона буде компенсована завдяки асинхронному програмуванню, який значно пришвидшить обробку інформації. На користь вибору вплинула простота в розумінні та легкість освоєння. Основною перевагою є те, що Python є "користувацькою" мовою, з багатьма зручностями, розробленими користувачами, і володіє великою кількістю бібліотек для створення чат-ботів, що дозволяє вибрати оптимальні рішення [24,25].

4.3 Вибір бібліотек

Чат-бот буде створюватись у месенджері Telegram. Він має власне Telegram Bot API, котрий дозволяє створювати ботів.

Для створення чат-боту існує багато бібліотек, найбільш популярні з них: Aiogram,TeleBot,pyTelegramBot [26,27].

Для створення згідно завданню телеграм-бота на Python, який приймає від користувача вказані дані, використовувалися наступні бібліотеки:

1) ***python-telegram-bot***. Для роботи з цією бібліотекою вам потрібно буде встановити її за допомогою ***pip install python-telegram-bot***;

2) ***smtplib*** для відправлення готового текстового файлу з даними на пошту після введення цих данних у бот користувачем. Так, для створення нових функцій для формування текстового файлу та відправки електронної пошти:

```
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

# ...

def create_text_file(data):
    # Створення текстового файлу з даними
    file_content = f"Дані про тварину:\n" \
                   f"Ім'я власника: {data['owner_name']} {data['owner_surname']}\n" \
                   f"Порода тварини: {data['animal_breed']}\n" \
                   f"Вид тварини: {data['animal_type']}\n" \
                   f"Дата розташування в прихистку: {data['shelter_location']}\n" \
                   f"Потрібна доставка: {'Так' if data['delivery_needed'] else 'Ні'}\n" \
                   f"Стан здоров'я тварини: {data['health_condition']}\n" \
                   f"Побажання по розташуванню: {data['placement_preference']}"

    with open('animal_data.txt', 'w') as file:
        file.write(file_content)

def send_email(data, to_email):
    # Підключення до серверу електронної пошти (замініть на свої дані)
    smtp_server = "your_smtp_server"
    smtp_port = 587
    smtp_username = "your_email@gmail.com"
    smtp_password = "your_email_password"

    # Формування текстового файлу з даними
    create_text_file(data)

    # Налаштування електронного листа
    subject = "Дані про тварину"
    sender_email = "your_email@gmail.com"

    msg = MIMEMultipart()
    msg['From'] = sender_email
    msg['To'] = to_email
    msg['Subject'] = subject
```

```
    # Додаємо текстовий файл до листа
    with open('animal_data.txt', 'r') as file:
        file_content = file.read()

    body = MIMEText(file_content, 'plain')
    msg.attach(body)

    # Відправка електронного листа
    with smtplib.SMTP(smtp_server, smtp_port) as server:
        server.starttls()
        server.login(smtp_username, smtp_password)
        server.sendmail(sender_email, to_email, msg.as_string())

# ...

def handle_text(update: Update, context: CallbackContext) -> None:
    # ...

    elif 'placement_preference' not in animal_data:
        animal_data['placement_preference'] = text

        # Виведення збережених даних
        update.message.reply_text(f"Дані про тварину:\n"
                                  f"Ім'я власника: {animal_data['owner_name']} {ani-
mal_data['owner_surname']}\n"
                                  f"Порода тварини: {animal_data['animal_breed']}\n"
                                  f"Вид тварини: {animal_data['animal_type']}\n"
                                  f"Дата розташування в прихистку: {animal_data['shelter_location']}\n"
                                  f"Потрібна доставка: {'Так' if animal_data['delivery_needed'] else 'Ні'}\n"
                                  f"Стан здоров'я тварини: {animal_data['health_condition']}\n"
                                  f"Побажання по розташуванню: {animal_data['placement_preference']}")

        # Отримання поштової адреси від користувача
        update.message.reply_text("Введіть вашу електронну адресу для відправлення даних:")
        return

    elif 'user_email' not in animal_data:
        animal_data['user_email'] = text
        send_email(animal_data, text)
        update.message.reply_text(f"Дані були відправлені на адресу {text}.")

        # Очищення збережених даних
        del context.user_data['user_id']
        animal_data.clear()
        update.message.reply_text("Дякуємо за введення даних! Інформація була збережена та
відправлена на вашу електронну адресу.")
```

3) додаткові бібліотеки, такі як ***requests*** та ***beautifulsoup4***, які допоможуть отримувати та обробляти дані з веб-сайту. Вони використовуються для

реалізації функції отримання переліку вільних місць у прихистку. Встановити їх можна командами, як представлено на рисунку 4.1;

Рисунок 4.1 – Виконання команди для підключення бібліотек ***requests*** та ***beautifulsoup4***

4) бібліотека для роботи з БД ***SQLAlchemy***, та відповідний драйвер БД, наприклад, SQLite або PostgreSQL. Вони необхідні для взаємодії з базою даних. Для їх встановлення використовуються запити, які показані на рисунку 4.2;

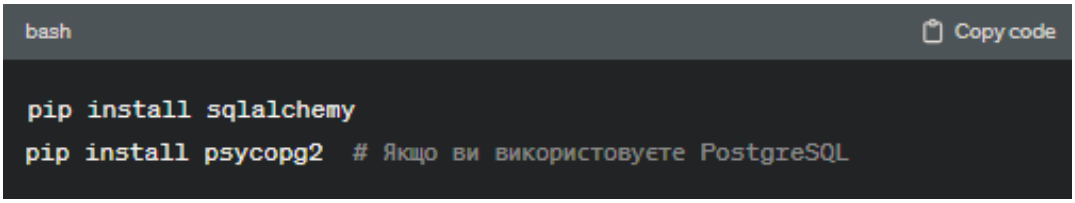


Рисунок 4.2 – Виконання команди для виконання запитів на Python для створення чат-боту Телеграм

5) бібліотеки ***requests*** для отримання вмісту веб-сайту та ***beautifulsoup4*** для парсингу HTML-коду та пошуку вільних місць у прихистку.

Усі ці бібліотеки володіють подібним набором функціоналу та прямо взаємодіють з Telegram API. Aiogram виділяється завдяки реалізації більш складних опцій та своєї асинхронної природи (тобто здатності виконувати процеси паралельно, не зачіпаючи порядок виконання, що призводить до покращення продуктивності чат-боту). Також, використання бібліотеки Asyncio дозволяє асинхронно впроваджувати навіть базові функції, що додатково прискорює роботу чат-бота [28].

4.4 Telegram Bot Api та реєстрація токену

Загальний алгоритм розробки чат-боту для Телеграм виглядає так [26,27]:

1) **створення бота в Телеграм**:

– відкрийте Телеграм та зайдіть у розділ "Боти" за допомогою пошуку;

– напишіть @BotFather, щоб створити нового бота;

– дотримуйтеся інструкцій BotFather, надайте ім'я та користувальне ім'я для вашого бота;

– отримайте токен бота від BotFather;

2) **створення сервера для бота**:

– щоб бот працював, вам потрібно створити сервер, на якому буде виконуватися код бота;

– ви можете використовувати хмарні платформи, такі як Heroku, AWS, або будь-яку іншу, яку ви вподобаєте;

3) налаштування оточення для бота: встановіть мову програмування та необхідні бібліотеки для роботи з API Телеграм (наприклад, python-telegram-bot для Python);

4) **написання коду бота**: створіть код бота, використовуючи отриманий токен та відповідні бібліотеки. Приклад для Python:

Telegram Bot API (Application Programming Interface) – це інтерфейс програмування застосунків, який надає Telegram для взаємодії з ботами в їхній платформі месенджера Telegram.

Основні характеристики та функції Telegram Bot API включають:

а) **отримання оновлень (Updates)**: За допомогою API бот може отримувати оновлення про події, які стаються в чатах, такі як нові повідомлення, зміни в учасниках чату тощо;

б) **надсилання повідомлень**: Бот може відправляти текстові повідомлення, зображення, відео, аудіо, файлові документи та інші медіафайли в чати або іншим користувачам;

в) **керування клавіатурою**: Бот може створювати і взаємодіяти з інтерактивними клавіатурами для полегшення взаємодії з користувачами;

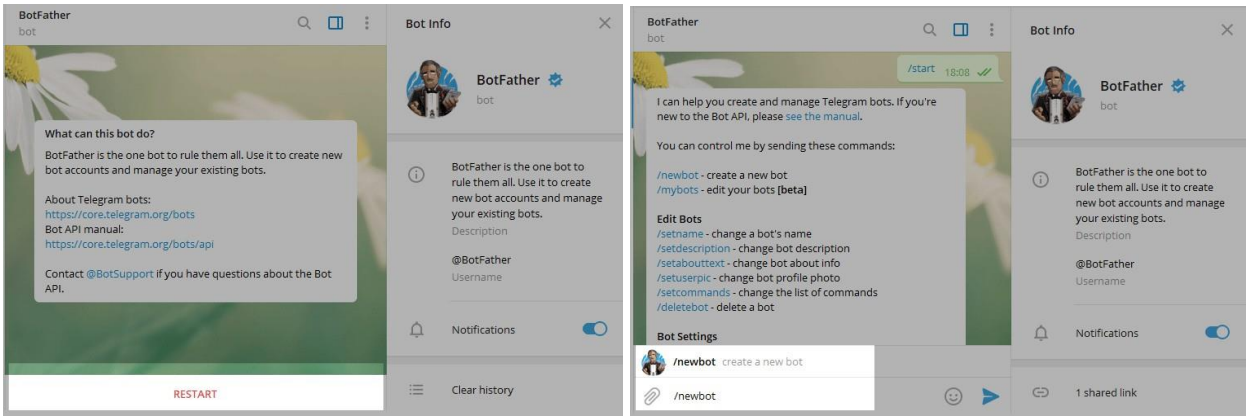
г) **робота з Inline-запитами**: Бот може обробляти Inline-запити, що дозволяє відправляти результати пошуку або іншу інформацію прямо в текстових полях чату;

д) **авторизація та ідентифікація**: API надає можливості для визначення та перевірки ідентифікаційних даних користувачів;

е) **керування чатами та учасниками**: Бот може отримувати інформацію про чати, керувати учасниками, змінювати налаштування чатів тощо.

З використанням Telegram Bot API розробники можуть створювати різноманітні чат-боти для автоматизації взаємодії з користувачами в Telegram, надаючи їм зручний та функціональний інтерфейс.

По-перше, перед тим як налаштовувати функціонал, необхідно отримати унікальний токен для вашого бота. Це можна зробити, відкривши Telegram та написавши команду /newbot користувачу @BotFather [26,27]. Після цього вам слід вказати ім'я для вашого бота (має бути унікальним) та його користувацьке ім'я (унікальний ідентифікатор, за яким можна знаходити бота в месенджері, також має бути унікальним). Після виконання цих кроків буде згенерований токен, який служить ключем для управління вашим ботом (див. рис. 4.3).



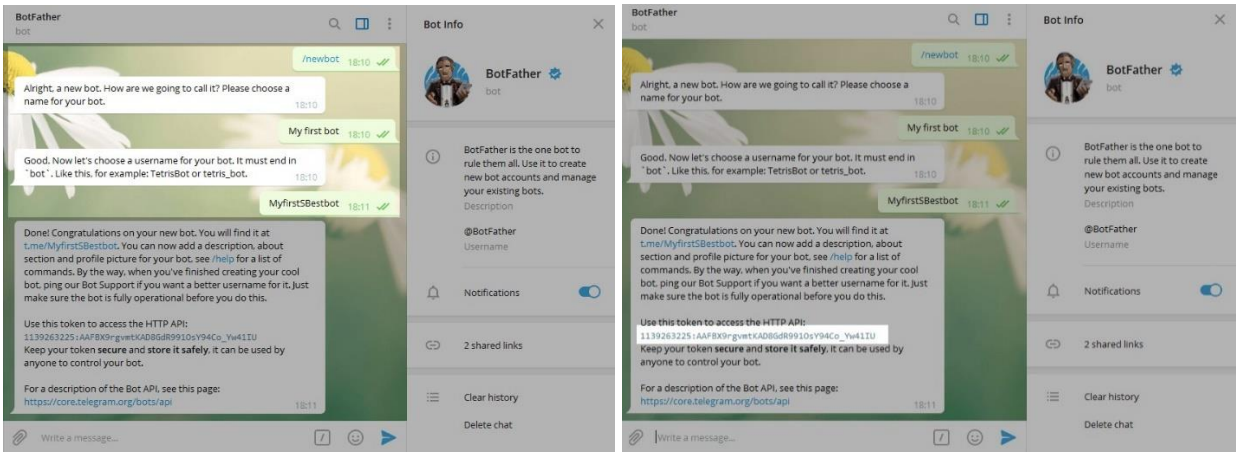


Рисунок 4.3 – Генерація токену для чат-боту

4.5 Програмна розробки чат-бота тимчасового прихистку тварин

Для того щоб почати робити бота, необхідно відкрити PyCharm, створити новий проект та додати до проекту описані вище бібліотеки [28]. Встановлення виконується в термінал вводом команд (див.рис.4.4):

```
ip install requests
pip install beautifulsoup4
```

Рисунок 4.4 – Встановлення додаткових бібліотек на прикладі requests та beautifulsoup4, які допоможуть отримувати та обробляти дані з веб-сайта

Встановлення пакетів бібліотек відображені на рисунку 4.5.

Рисунок 4.5 – Встановлення бібліотек

Після цього рекомендується створити файл **config.py**, в якому будуть зберігатися дві важливі речі: токен та ідентифікатор адміністратора, які слід тримати конфіденційними.

```
token = "5014918283:AaAaAaAaaaaaaaaaa_kkkkkk4"
admin_id = 111222333
```

У головний файл боту рекомендується включити імпорт завантажених бібліотек, а також файлів конфігурації та парсера. Далі, важливим етапом буде виконання двох завдань: передача конфігурації боту та повідомлення обробнику, що це є ботом:

```
bot = Bot(token=token, parse_mode=types.ParseMode.HTML)
dp = Dispatcher(bot)
```

Зчитування інформації від користувача відбувається наступним кодом:

```
if 'owner_name' not in context.user_data:
        context.user_data['owner_name'] = text
        update.message.reply_text("Введіть прізвище власника тварини:")
    elif 'owner_surname' not in context.user_data:
        context.user_data['owner_surname'] = text
```

Для написання чат-боту використовуються наступні функції:

а) щоб додати перевірку дати та виведення повідомлення про помилку при некоректній даті функція **handle_text**, яка виконує перевірку дати, яку вводить користувач, щоб вона не була раніше дати введення даних. І, якщо дата раніш сьогоденної, то виводить текстове повідомлення про помилку "Дата некоректна, оберіть іншу". Данні про дату вводяться до тих пір, допоки не буде введене коректне число:

```
def handle_text(update: Update, context: CallbackContext) -> None:
    user_id = update.message.from_user.id

    if 'user_id' not in context.user_data:
        context.user_data['user_id'] = user_id
```

В наведеному нижче коді використовується **datetime.strptime** для перетворення введеного тексту у об'єкт **datetime**. Якщо це вдається, проводиться порівняння дати розташування в прихистку з поточною датою. Якщо дата раніша за поточну, виводиться повідомлення про помилку, і користувач повинен ввести іншу дату. При помилковому форматі дати також виводиться відповідне повідомлення;

```
elif 'shelter_location' not in context.user_data:
        try:
            # Перевірка, чи дата розташування в прихистку не раніше сьогоднішньої
            placement_date = datetime.strptime(text, '%Y-%m-%d').date()
            today_date = datetime.now().date()

            if placement_date < today_date:
                update.message.reply_text("Дата некоректна, оберіть іншу.")
                return
```

б) функція **send_email** для відправлення повідомлення про те, що заявка схвалена та надсилання цього повідомлення користувачеві після відправлення листа на пошту:

```
def send_email(user_data, user_email):
    # Код для надсилання електронного листа на пошту user_email

    # Ваш код для надсилання електронного листа

    # Після успішного надсилання листа, відправте повідомлення користувачеві через Telegram
    chat_id = user_data['user_id']
    message_text = "Ваша заявка прийнята в обробку, очікуйте на дзвінок оператора."
    context.bot.send_message(chat_id=chat_id, text=message_text)

# ...
```

У цьому коді, після успішного надсилання листа на пошту користувача, використовуючи ваш власний код для надсилання листа, бот відправляє користувачеві повідомлення про те, що заявка була прийнята в обробку.

Бот після перегляду даних показує дві кнопки: кнопка "Все вірно" , яка підтверджує правильність даних перед їх відправкою на пошту, кнопка

"видалити заявку", після натискання якої остання заявка від користувача з даними видаляється;

в) функція **button_callback** для обробки цих натискань кнопок. Вона реалізує можливість підтверджувати вірність даних та видаляти заявку за допомогою кнопок. Функція видаляє збережені дані після відправки заявки або видалення її користувачем:

```
def button_callback(update: Update, context: CallbackContext) -> None:
    query = update.callback_query
    query.answer()
    user_id = query.from_user.id

    if query.data == 'confirm':
        # Видалення збережених даних після підтвердження користувачем
        del context.user_data[user_id]
        query.edit_message_text("Дякуємо! Ваша заявка була успішно відправлена.")
    elif query.data == 'cancel':
        # Видалення збережених даних при відмові користувача
        del context.user_data[user_id]
        query.edit_message_text("Ваша заявка була видалена.")

# ...

def main() -> None:
    # ...

    dp.add_handler(CallbackQueryHandler(button_callback))

    updater.start_polling()

    updater.idle()

# ...
```

г) функції отримання переліку вільних місць у прихистку, для якої спочатку були встановлені додаткові бібліотеки, такі як **requests** та **beautifulsoup4**, які допоможуть отримувати та обробляти дані з веб-сайту. Функція отримання переліку вільних місць у прихистку: за запитом користувача парсер заходить на сайт, шукає там необхідну інформацію, які номери вже зайняті, та, у разі відсутніх вільних номерів видає повідомлення користувачу "Вільних місць на вказаний термін немає". Тоді функція для обробки текстових

повідомлень так, щоб бот міг обробляти запити користувачів та шукати вільні місця на вказаний термін виглядає так:

```
import requests
from bs4 import BeautifulSoup

# ...

def handle_text(update: Update, context: CallbackContext) -> None:
    # ...

    elif 'placement_preference' not in animal_data:
        animal_data['placement_preference'] = text

        # Отримання переліку вільних місць на вказаний термін
        free_places = get_free_places(animal_data['placement_preference'])

        # Виведення збережених даних та переліку вільних місць
        update.message.reply_text(f"Дані про тварину:\n"
                                  f"Ім'я власника: {animal_data['owner_name']} {ani-
mal_data['owner_surname']}\n"
                                  f"Порода тварини: {animal_data['animal_breed']}\n"
                                  f"Вид тварини: {animal_data['animal_type']}\n"
                                  f"Дата розташування в прихистку: {animal_data['shelter_location']}\n"
                                  f"Потрібна доставка: {'Так' if animal_data['delivery_needed'] else 'Ні'}\n"
                                  f"Стан здоров'я тварини: {animal_data['health_condition']}\n"
                                  f"Побажання по розташуванню: {ani-
mal_data['placement_preference']}\n\n"
                                  f"Перелік вільних місць:\n{free_places}")

        # Очищення збережених даних
        del context.user_data['user_id']
        animal_data.clear()
        update.message.reply_text("Дякуємо за введення даних та перевірку вільних місць! Інфор-
мація була збережена та перелік вільних місць вказано вище.")

def get_free_places(shelter_location):
    # URL прихистку, замініть на реальний URL
    shelter_url = "https://example.com/shelter"

    # Отримання вмісту веб-сайту
    response = requests.get(shelter_url)
    soup = BeautifulSoup(response.text, 'html.parser')

    # Пошук елементів, які містять інформацію про вільні місця
    free_places_list = soup.find_all('div', class_='free-place')

    # Перевірка, чи є вільні місця
    if not free_places_list:
        return "Вільних місць на вказаний термін немає"
```

```
    # Формування переліку вільних місць
    free_places = ""
    for place in free_places_list:
        place_number = place.find('span', class_='place-number').text
        free_places += f"Місце {place_number}\n"

    return free_places

# ...
```

У наведеному коді функція **get_free_places** використовує бібліотеку **requests** для отримання вмісту веб-сайту та **beautifulsoup4** для парсингу HTML-коду та пошуку вільних місць. Пам'ятайте про те, що ви повинні замінити **"https://example.com/shelter"** на реальний URL веб-сайту прихистку, який ви хочете перевірити;

д) функція для формування текстового файлу та відправки електронної пошти:

```
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

# ...

def create_text_file(data):
    # Створення текстового файлу з даними
    file_content = f"Дані про тварину:\n" \
                   f"Ім'я власника: {data['owner_name']} {data['owner_surname']}\n" \
                   f"Порода тварини: {data['animal_breed']}\n" \
                   f"Вид тварини: {data['animal_type']}\n" \
                   f"Дата розташування в прихистку: {data['shelter_location']}\n" \
                   f"Потрібна доставка: {'Так' if data['delivery_needed'] else 'Ні'}\n" \
                   f"Стан здоров'я тварини: {data['health_condition']}\n" \
                   f"Побажання по розташуванню: {data['placement_preference']}"

    with open('animal_data.txt', 'w') as file:
        file.write(file_content)

def send_email(data, to_email):
    # Підключення до серверу електронної пошти (замініть на свої дані)
    smtp_server = "your_smtp_server"
    smtp_port = 587
    smtp_username = "your_email@gmail.com"
    smtp_password = "your_email_password"
```

```
    # Формування текстового файлу з даними
    create_text_file(data)

    # Налаштування електронного листа
    subject = "Дані про тварину"
    sender_email = "your_email@gmail.com"

    msg = MIMEMultipart()
    msg['From'] = sender_email
    msg['To'] = to_email
    msg['Subject'] = subject

    # Додаємо текстовий файл до листа
    with open('animal_data.txt', 'r') as file:
        file_content = file.read()

    body = MIMEText(file_content, 'plain')
    msg.attach(body)

    # Відправка електронного листа
    with smtplib.SMTP(smtp_server, smtp_port) as server:
        server.starttls()
        server.login(smtp_username, smtp_password)
        server.sendmail(sender_email, to_email, msg.as_string())

# ...
```

Таким чином, була прописана певна логіка чат-боту, що дозволяє формувати заявку на прихисток домашньої тварини, з урахуванням вида тварини та наявної кількості вільних номерів, та перевіряти дату оформлення прихистку. Якщо відправити боту повідомлення, яке він не може ідентифікувати, він поверне помилку. Для поліпшення логіки можна створити словник або базу схожих слів і фраз. Це суттєво підвищить ефективність бота, але для більш комфортного взаємодії рекомендується користуватися клавіатурою.

ВИСНОВКИ

Згідно завданню необхідно провести дослідження та аналіз допоміжних інформаційних технологій в якості віртуальних помічників, а саме чат-ботів та розробити чат-бот тимчасового прихистку тварин.

Для вирішення поставленої задачі проведено аналіз предметної області, в рамках якого виконана постановка задач дослідження та розробки, проведено аналіз аналогів web-сайтів та чат-ботів прихистку тварин.

В рамках аналізу моделей, архітектур та інформаційних технологій віртуальних помічників проведено аналіз інтелектуальних основ віртуальних помічників в історичному аспекті, аналіз архітектур чат-ботів, варіантів та підходів до розробки чат-ботів.

В рамках структурного проєктування ІС проведено аналіз взаємодії клієнта прихистку тварин із чат-ботом, визначені основні компоненти системи віртуального помічника та проаналізовано їх функціонал. Визначена вхідна інформація заявки користувача чат-бота тимчасового прихистку тварин та способи її подачі. Розроблена діаграма прецедентів для зареєстрованого користувача та визначені атрибути сутності, а також розроблена схема бази даних для платформи MySQL Server та виконано опис прецеденту «Оформлення замовлення за допомогою чат-бота» за методологією RUP.

В рамках програмної реалізація чат-бота тимчасового прихистку тварин на основі вибраного інструменту (платформи, мови програмування та бібліотек) виконана програмна розробка чат-бота.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Як чат-боти змінили IT-індустрію. URL: bit.ly/3GdYNAl (дата звернення: 01.10.2023).

2. Кількість користувачів у соц-мережах. URL: https://www.sostav.ru/ publication/ we-are-social-i-hootsuite-52472.html (дата звернення: 18.10.2023).

3. Готель для кішок «Найкращий друг»: веб-сайт. URL: https://cat-hostel-100.business.site/ (дата звернення: 05.10.2023).

4. Готель для тварин «Darliss»: веб-сайт. URL: https://darliss.com/kharkov/ (дата звернення: 07.10.2023).

5. Готель для тварин «vet-domik»: веб-сайт. URL: https://www.vet-domik.com/ (дата звернення: 09.10.2023).

6. Чат-боти в маркетингу та бізнесі. URL: https://www.epochta.ru/ blog/articles/chat-bots/ (дата звернення: 19.10.2023).

7. Переваги та недоліки користування чат-ботами. URL: https://продвижение.kz/pljusy-i-minusy-ispolzovanija-chat-botov-v-biznese/ (дата звернення: 20.10.2023).

8. Чому чат-боти стали популярні. URL: https://valmaxdigital.com.ua/ blog/why-chat-bots-became-popular-and-how-to-usethem-for-your-business(дата звернення: 18.10.2023).

9. «Прихисток із твариною»: Telegram -бот. URL: https://t.me/PrykhystyBot (дата звернення: 14.10.2023).

10. TinPet: Telegram-бот. URL: https://t.me/TinPet_bot (дата звернення: 16.10.2023).

11. «Pro100media. Краматорськ новини»: Telegram-бот. https://t.me/pro100_media (дата звернення: 16.10.2023).

12. A.M. Turing. Computing Machinery and Intelligence. Mind. – 1959. Vol. 59, N. 236. – P. 433-460. URL: www.jstor.org/stable/2251299

13 A.L.I.C.E. Artificial Intelligence Foundation. URL: http://www.alicebot.org

14. Обробка звичайної мови та машинне навчання. URL:

https://singularika.com/ru/chatbots/обработка-естественного-языка-и-маши/

15. Обробка природної мови. URL: https://uk.wikipedia.org/wiki/Обробка _природної_мови.

16. Приклад чат-боту з фейсбуку. URL: https://gdetraffic.com/img/ news_3304/tp/1080x0__pasted_image_0_(87).png (дата звернення: 18.10.2023).

17. Основні задачі чат-ботів. URL: https://helpcrunch.com/blog/ru/chtotakoe-chat-bot/

18. Архітектура чат-ботів. URL: https://digitrain.ru/articles/512418/

19. Технологии создания и применения чат ботов. URL: https://cyberleninka.ru/article/v/tehnologii-sozdaniya-i-primeneniya-chatbotov

20. Що таке чат-бот та як його створити. URL: https://serpstat.com/ru/ blog/kak-sozdat-chat-bota/ (дата звернення: 18.10.2023).

21. Як створити свого чат-бота. URL: https://habr.com/ru/company/ just_ai/blog/656801/ (дата звернення: 18.10.2023).

22. Все, про що повинен знати розробник чат-ботів. URL: https://habr.com/ ru/post/543676/ (дата звернення: 18.10.2023).

23. На чому робити чат-боти новачку та профі. URL: https://vc.ru/services/440634-shablony-konstruktory-dialogovye-platformy-i-opensource-freymvorki-na-chem-delat-bota-novichku-i-profi

24. Лутц М. Вивчаємо Python, 4-е видання. – 1280 с.

25. Как создать чат-бота с нуля на Python: подробная инструкция URL:https://www.dcstyling.ru/robots/kaksozdat-chat-bota-s-nulia-na-pythonpodrobnaia-instruktsiia/

26. Документація Telegram Bot Api. URL: https://tlgrm.ru/docs/bots/api#authorizing-your-bot

27. Офіційний сайт Telegtam API. URL: https://core.telegram.org/api

28. Документація Aiogram. URL: https://docs.aiogram.dev/en/latest/ 27. Asyncio для розробника. URL: https://habr.com/ru/post/337420/

28. How to make a responsive telegram bot. URL: https://www.sohamkamani.com/blog/2016/09/21/making-atelegram-bot/

29. ДСТУ 3008-2015. Документація. «Звіти у сфері науки і техніки. Структура та правила оформлення» Чинний від 2017-07-01. – Київ: ДП «УкрНДНЦ», 2016. – 26 с.