Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студента *Воробйова Данила Дмитровича*
(ПІБ)

академічної групи *122М-22-2*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *«122 Комп'ютерні науки»*
(назва освітньої програми)

на тему: *Дослідження ефективності використання Telegram-бота для покращення бізнес-процесу*

_____ *Д.Д. Воробйов*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
розділів кваліфікаційної роботи				
спеціальний	*Проф. Мороз Б. І.*			

Рецензент				

Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2023

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2023 року

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

студенту *122м-22-2* (група) *Воробйову Данилу Дмитровичу* (прізвище та ініціали)

Тема кваліфікаційної роботи *Дослідження ефективності використання Telegram-бота для покращення бізнес-процесу*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 09.10.2023 р. № 1227с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – боти для платформи «Telegram», як новітній, популярний, зручний та дуже прибутковий засіб для бізнесу.

Предмет досліджень – дослідження ефективності використання власного створеного бота для Telegram, швидкості його роботи, привабливості, а також можливості конкурувати з іншими продуктами даного типу на ринку.

Мета роботи – розробити бота для платформи Telegram, який буде відповідати усім необхідним нормам для спілкування з клієнтом, надання йому можливості обрати необхідний товар з меню. Дослідити ефективність використання такого бота, порівняти розроблений програмний продукт з іншими існуючими аналогами на ринку та оновити за необхідності, удосконалити.

Вихідні дані для проведення роботи – інформація та результат роботи, що були зібрані під час переддипломної практики. Загально відомі дані про ботів для різних платформ, дані знайдені у мережі Інтернет, відомості з книг і відео у

вільному доступі.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Новизна запропонованих рішень являє собою доведення необхідності використання ботів для бізнесу через необхідність конкурувати з іншими компаніями, покращувати власний продукт і приваблювати нових клієнтів. Відповідно до цього пропонується вирішення даної проблеми.

Практична цінність даної роботи полягає у тому, що розроблений програмний продукт буде надавати можливості взаємодії бізнесу зі своїми клієнтами в онлайн режимі у мережі Інтернет, безпосереднього з ними спілкування, надання можливості користуватися послугами компанії без необхідності кудись йти. Також використання бота буде однозначно підвищувати прибуток фірми. Вивчення недоліків інших аналогічних продуктів дозволить вести конкуренцію та буде надавати перевагу над конкурентами за рахунок цього.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень повинні бути поданими у такому вигляді, аби було можливим використання даного програмного продукту для роботи компанії без необхідності для неї вносити суттєві зміни у програмний код, інтерфейс або налаштування.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Розробка безпосередньо програмного застосунку для подальшого використання, дослідження та покращення.	10.10.23 – 20.10.23
Перевірка програмного продукту на практиці, збір початкових даних про ефективність та можливість конкурувати чи приваблювати клієнтів. Створення плану оновлення та покращення боту для досягнення кінцевої цілі.	20.10.23 – 29.10.23
Надання боту оновлень, що мають покращити роботу та можливість конкурувати. Збір більш детальних даних про можливості бота після оновлення.	02.11.23 – 01.12.23

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект після реалізації проекту роботи очікується позитивний. В першу чергу це пов’язано з тим, що кінцевий продукт буде містити у собі результати спостерігання та покращення програми на значному проміжку часу, це дасть можливість конкурувати. Окрім цього сам по собі проект призначений для того аби приваблювати споживачів так отримувати прибуток, який має зрости після завершення роботи.

Соціальний ефект після реалізації проекту очікується також позитивним. Більш зручний інтерфейс, взяті до уваги побажання клієнтів, надання їм можливості зручно обирати і замовляти що треба, усе це має задовільнити споживача.

Завдання видав _____ *Мороз Б.І.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Воробйов Д.Д.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *10.10.2023 р.*

Термін подання кваліфікаційної роботи до ЕК *11.12.2023*

РЕФЕРАТ

Пояснювальна записка: ___ сторінок, __ рисунки, __ додатки, __ джерела.

Об'єкт дослідження: боти для платформи «Telegram» як новітній, популярний, зручний та дуже прибутковий засіб для бізнесу.

Предмет дослідження: дослідження ефективності використання власного створеного бота для Telegram, швидкості його роботи, привабливості, а також можливості конкурувати з іншими продуктами даного типу на ринку.

Мета роботи: розробити бота для платформи Telegram, який буде відповідати усім необхідним нормам для спілкування з клієнтом, надання йому можливості обрати необхідний товар з меню. Дослідити ефективність використання такого бота, порівняти розроблений програмний продукт з іншими існуючими аналогами на ринку та оновити за необхідності, удосконалити.

Методи дослідження. В ході дослідження використовувалось спостереження за розробленим продуктом, а також порівняння з аналогами. Вивчалась загальна інформація про ботів та бізнес у мережі Інтернет з відеороликів, книжок, статей та документацій.

Новизна отриманих результатів являє собою доведення необхідності використання ботів для бізнесу через необхідність конкурувати з іншими компаніями, покращувати власний продукт і приваблювати нових клієнтів. Відповідно до цього пропонується вирішення даної проблеми.

Практична цінність даної роботи полягає у тому, що розроблений програмний продукт буде надавати можливості взаємодії бізнесу зі своїми клієнтами в онлайн режимі у мережі Інтернет, безпосереднього з ними спілкування, надання можливості користуватися послугами компанії без необхідності кудись йти. Також використання бота буде однозначно підвищувати прибуток фірми. Вивчення недоліків інших аналогічних продуктів дозволить вести конкуренцію та буде надавати перевагу над конкурентами за рахунок цього.

Область застосування. Розроблений бот можна використовувати у бізнесі. В більшості випадків використання приємне для тих компаній, які продають ті чи інші товари та послуги.

Значення роботи та висновки. Розроблений бот можна використовувати для ведення бізнесу, контакту з клієнтами, просування свого товару та збору інформації

Прогнози щодо розвитку досліджень. Даний програмний продукт дозволяє вести подальшу розробку та видозмінення свого наповнення. Сприяє цьому також фактор гнучкої мови Python.

Економіка. Очікується прибуток від використання програми. Це зумовлене популярністю продукції та новаціями. Варіть проекту 23671 грн.

Список ключових слів: Python, Telegram, бот, дослідження, спостереження, framework, бізнес, інформація, оновлення, ефективність.

ABSTRACT

Explanatory note: ___ pages, __ figures, __ appendices, __ sources.

The object of the study: bots for the Telegram platform as the latest, popular, convenient and very profitable tool for business.

The subject of the study: the study of the effectiveness of using a self-created bot for Telegram, its speed of operation, attractiveness, as well as the ability to compete with other products of this type on the market.

The goal of the work: to develop a bot for the Telegram platform that will meet all the necessary standards for communicating with the client, giving him the opportunity to choose the necessary product from the menu. Investigate the effectiveness of using such a bot, compare the developed software product with other existing analogues on the market and, if necessary, update and improve it.

Research methods. During the research, observation of the developed product was used, as well as comparison with analogues. General information about bots and business on the Internet was studied from videos, books, articles and documentation.

The novelty of the obtained results is a justification for the need to use bots for business due to the need to compete with other companies, improve their own product and attract new customers. Accordingly, a solution to this problem is proposed.

The practical value of this work lies in the fact that the developed software product will provide opportunities for businesses to interact with their customers online on the Internet, communicate directly with them, and provide the opportunity to use the company's services without having to go anywhere. Also, using a bot will definitely increase the company's profit. Studying the weaknesses of other analogous products will allow you to compete and give you an advantage over your competitors due to this.

Field of application. The developed bot can be used in business. In most cases, the use is pleasant for those companies that sell certain goods and services.

Value of work and conclusions. The developed bot can be used to conduct business, contact customers, promote your product and collect information

Forecasts regarding the development of research. This software product allows for further development and modification of its content. The factor of the flexible Python language also contributes to this.

Economy. Profit is expected from the use of the program. This is due to the popularity of products and innovations. Cook project 23671 hryvnias.

List of keywords: Python, Telegram, bot, research, observation, framework, business, information, update, efficiency.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ЕОМ – Електронна обчислювальна машина;

ГБ – Гігабайт;

ГЦ – Герц;

SW – Software;

PL – Programming Language;

OS – Operating System;

PY – Python Language;

DB – Data Base;

GUI – Graphical User Interface;

API – Application Programming Interface.

ЗМІСТ

ВСТУП

Актуальність теми. Актуальність теми розробки, покращення та використання ботів у різних соціальних мережах чи «месенджерах» для просування свого бізнесу є дуже високою станом на зараз. Існує дуже багато причин використання та популярності даного типу програмного застосунку, але основні з них – простота використання, легкий і зрозумілий інтерфейс, малий обсяг роботи при створенні та мала кількість необхідних ресурсів. Окрім цього такі проекти зазвичай приносять великий прибуток своїм власникам та допомагають взаємодіяти з клієнтами і просувати свою продукцію, паралельно створюючи великий вплив у «ком'юніті».

Такі програмні застосунки, як Telegram-боти, мають значну роль для обслуговування клієнтів та взаємодії з ними. Дані програми здатні автоматизувати виконання дуже складних завдань, роботи яка потребує багато зусиль, задачі які займають певний час та є кропіткими. Використовуючи такі програмні застосунки, компанії значно прискорюють свою роботу, не витрачають зайвий час на питання від клієнтів про їх товари та послуги, не відповідають на дуже часті запитання. Це дозволяє зменшити кількість роботи, що припадає на персонал, а також значно прискорює процес роботи, дозволяє концентруватись на інших та більш значних задачах і цілях. У більшості випадків компанії мають бота з запропонованими варіантами вирішення питань або вбудовану можливість викликати допомогу онлайн. Звичайно це сприяє більш гарному враженню у клієнта, що може привести до потенційного прибутку у майбутньому. Клієнт буде купувати більше товару, буде йти на контакт, а може і приведе нових покупців.

Звісно не можна забувати про те, що боти здатні дуже сильно і швидко просувати товари, займатися маркетингом. Це значно збільшує кількість продаж і заохочує нових клієнтів. Зокрема бот може відігравати роль на інформаційному полі, створюючи розсилку повідомлень про акції, нові послуги або продукти. Такі боти здатні дати клієнтам пораду щодо покупки та

покращити опит використання. Прикладом популярних і прибуткових ботів можна навести бот від «Нова пошта», який дозволяє відслідковувати замовлення і надає необхідну підтримку, або бот від провайдеру «Фрегат», він дозволяє дізнатися статус свого абонентського пакету чи провести платіж, змінити тарифний план, користуватись іншими послугами і пропозиціями.

Виходячи з усього цього можна зробити висновок, що на даний момент часу боти є дуже популярним і вкрай необхідним «софтом» (software).

Об'єкт досліджень. У якості об'єкта для досліджень була поставлена мета створити бота для платформи «Телеграм» і спостерігати за його ефективністю. Зокрема порівняти його з іншими аналогами на ринку.

Предмет досліджень. Предметом дослідження є власний створений бот, необхідність дослідити його ефективність, швидкість його роботи, можливості конкуренції в порівнянні з іншими аналогами на ринку. Окрім цього детальну увагу було звернено на інформаційні джерела, статті, загальні відомості в галузі та відео.

Мета дослідження. На меті стоїть розробка власного боту з подальшим дослідженням ефективності цього бота, можливість конкуренції, порівняння його зі схожими програмами.

Методи дослідження. В ході дослідження був використаний метод спостереження та порівняння з іншими програмними продуктами на ринку. Окрім цього детально вивчалась документація ботів «Телеграм». Був проведений аналіз інформації з відкритих джерел та літератури, після чого вона отримала узагальнений вид.

Новизна. Рішення та результати дослідження, що були отримані в результаті роботи, доводять необхідність використання ботів у соціальних мережах та «месенджерах», постійно оновлювати програмне забезпечення і додавати нові функції, розробляти прості у використанні застосунки, що можуть спілкуватись з клієнтами. Згідно цього пропонується рішення для цієї проблеми, що було прийнято у результаті аналізу даних, дослідження та

порівняння програмних продуктів. Дане рішення враховує недоліки деяких інших програмних застосунків.

Практичне значення. Практична цінність застосунку полягає у тому, що розроблений бот можна використовувати для ведення бізнесу і продажу великої кількості продукції з широким асортиментом. Програмний засіб має зручний та легкий інтерфейс, який дозволяє легко взаємодіяти з клієнтами та гнучко міняти наповнення бота, додавати нові функції. Перевагою бота є можливість вести конкуренцію з іншим бізнесом, враховані недоліків деяких аналогічних ботів, здатність бота приносити прибуток.

Особистий внесок автора. Особистим внеском автора є розроблений програмний продукт та теоретична частина до нього, а також усієї роботи. Також було проведено аналіз даних та приведення їх в узагальнену форму, дослідження роботи програмного застосунку, використано метод порівняння з іншими продуктами та зроблена кінцева оцінка продукту.

Структура та обсяг роботи магістра. Вся робота містить титульний аркуш, завдання, реферат, умовні позначення, зміст, вступ, чотири розділи, висновок та додатки. Робота складається з ___ сторінок, серед яких ___ основна частина, ___ рисунків, перелік використаних джерел, додатки на __ сторінках.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ

1.1. Зальний стан питання. Теоретичні та методологічні відомості

Станом на сьогодні можна сміливо казати, що з появою ботів для соціальних мереж та месенджерів ці галузі, а також пов'язані з ними, змінились назавжди. З появою таких зручних інструментів, які самі за тебе виконують велику кількість роботи, об'єм роботи на одну людину значно зменшився. Те на що раніше витрачали час робітники, зараз виконують боти. Це в свою чергу дає можливість перерозподілити людські ресурси та використовувати їх у більш доцільних напрямках. Окрім цього такі зміни дали багато вільного часу буквально з повітря.

Поява ботів дала змогу використовувати менеджмент часу більш зручно і ефективніше. Робітникам компаній більше не обов'язково розриватися на дві частини аби усе встигнути, тому що тепер складні завдання за об'ємом праці або багато малих і простих може взяти на себе бот. Відповідно тепер персонал зможе більше концентруватися на поставленій меті та не робити перенавантаження на одиницю часу. Це неминуче зменшить кількість помилок які допускаються і покращить якість роботи.

Окрім виграшу у часі ще одним плюсом автоматизації є онлайн послуги. З появою ботів більше немає сенсу в постійному менеджменті, моніторингу замовлень тощо. Відтепер будь-яка компанія з продажу товару або подібна може просто створити бота та автоматизувати більшу частину процесу роботи. Звичайно такий підхід дуже зручний і для самого клієнта. Навіть просто знаходячись у себе в дома будь-яка людина може взяти в руки телефон та відкрити програмний застосунок і знайти там для себе усю основну та необхідну інформацію. Це стосується як звичайного асортименту товару і послуг, так і більш конкретних питань – часу доставки, способів оплати товару, отримання та вимоги гарантії, а у деяких випадках навіть можливість

примірити товар. Якщо казати про останнє, то саме так зробив магазин «Розетка», прямо у мобільному застосунку є можливість дізнатись чи підходить вам одяг, взуття тощо. Достатньо лише надати необхідні розміри для програмного продукту, а він в свою чергу сам за вас все прорахує.

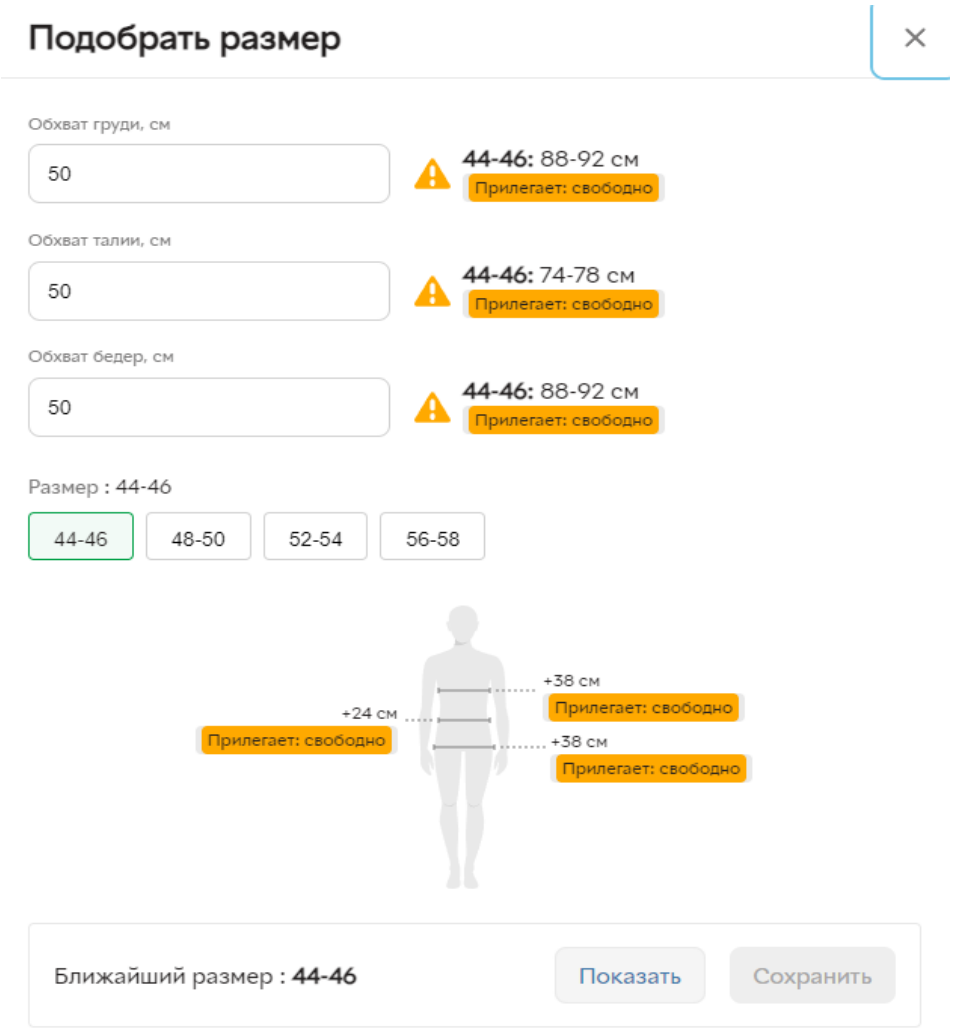

Рис. 1.1. Приклад автоматичного підбору одягу магазину «Розетка»

Усе нове приходить тоді, коли є попит на це. Тому не дивно що з часом у тому ж Telegram почали з’являтись боти призначені для розваги користувача. Ігрові боти досить популярні та займають досить значну частину серед усіх. Проте найпопулярнішими все одно є боти з можливістю замовлення онлайн, допоміжні боти для вирішення найчастіших запитань та боти з рекламою.

Останні приносять найбільший прибуток, особливо якщо мова йде про «гібрид» у якому є абсолютно все.

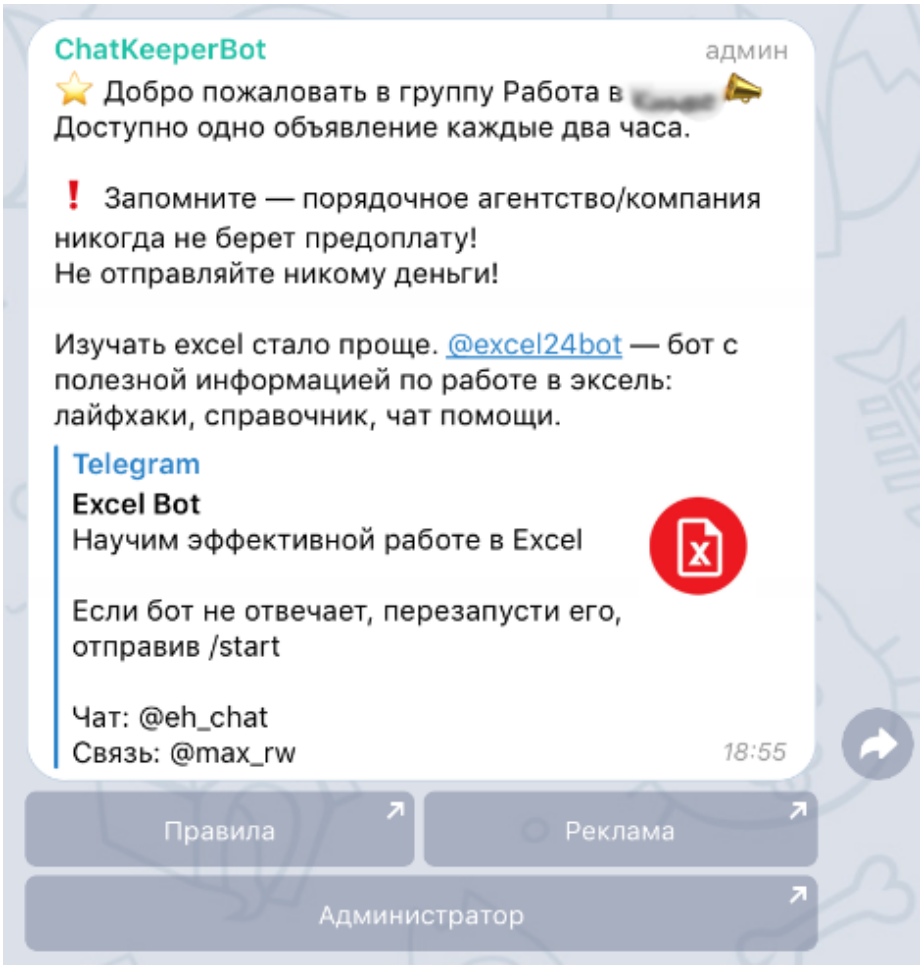

Рис. 1.2. Приклад чат-бота з рекламою.

Через велику кількість ботів відповідно є попит на їх врегулювання, оскільки є боти зі «спамом», а деякі взагалі використовують для шахрайства. У Telegram, як і у значній більшості інших систем, питання «кібербезпеки» було вирішене завдяки спеціальному стандарту «robots.txt». Якщо пояснити швидко і просто, то стандарт регулює кількість інформації та частини сайту, які надаються для обробки. Найпопулярнішими системами, що використовують цей стандарт, є Google, Bing, Yahoo, Baidu, Lycos та деякі інші. У якомусь сенсі вони і самі нагадують ботів, відповідаючи на запитання користувача.

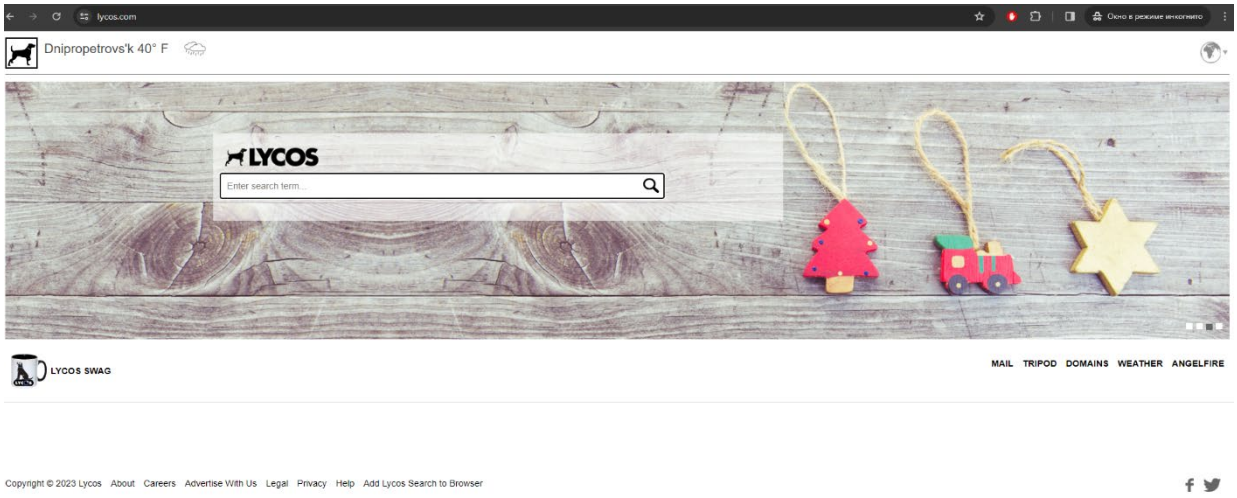

Рис. 1.3. Приклад пошукової системи Lycos.

Слід зазначити, що навіть цей стандарт не може гарантувати нам повний захист, оскільки шкідливі програми зазвичай не слідкують за правилами та обходять їх. Досить простим прикладом є боти, які вдають, ніби вони слідкують правилам, але у результаті до вас у приватний чат може написати спам-бот. Саме тому в першу чергу треба бути самому обережним та дотримуватись правил роботи в мережі Інтернет. Самі ж розробники час від часу випускають оновлення, що мають захищати користувача та його персональні дані, а також завчасно уникати таких проблемних ботів та інших програм. Окрім цього розробники сподіваються на чесне користування від самих людей та інших розробників.

Говорячи про таку платформу як Telegram, слід згадати і інші засоби його захисту. Доволі часто проходить аудит безпеки для виявлення вразливих місць та їх усунення. Серед інших способів захисту є самознищення повідомлень через певний проміжок часу, його користувач встановлює сам. Також є двоетапна «аутентифікація», яка посилює захист вашого облікового запису. Був передбачений також и захист від перехоплення повідомлень, ваше клієнтське повідомлення шифрується програмою, тому отримати і переглянути його зможе лише отримувач та ви. Для цього використовується протокол шифрування MTProto. Він був розроблений спеціально

розробниками Telegram та забезпечує вам шифрування типу «end-to-end». Даний протокол оснований на комбінації асиметричного та симетричного шифрування. У цьому випадку повідомлення будуть шифруватися спеціальним унікальним ключем для кожного окремого повідомлення та діалогу. До цього ж додається асиметричне шифрування, яке додає ще один ключ. Це ще більше ускладнює можливість викрасти персональні дані користувачів.

Доцільним є також використання різних модифікацій та розширень. Наприклад, якщо нам потрібно встановити затримку перед завантаженням сторінок, уникнути перенавантаження системи або прискорити її роботу. Зокрема існують можливості відключати певні елементи та концентруватись на необхідних.

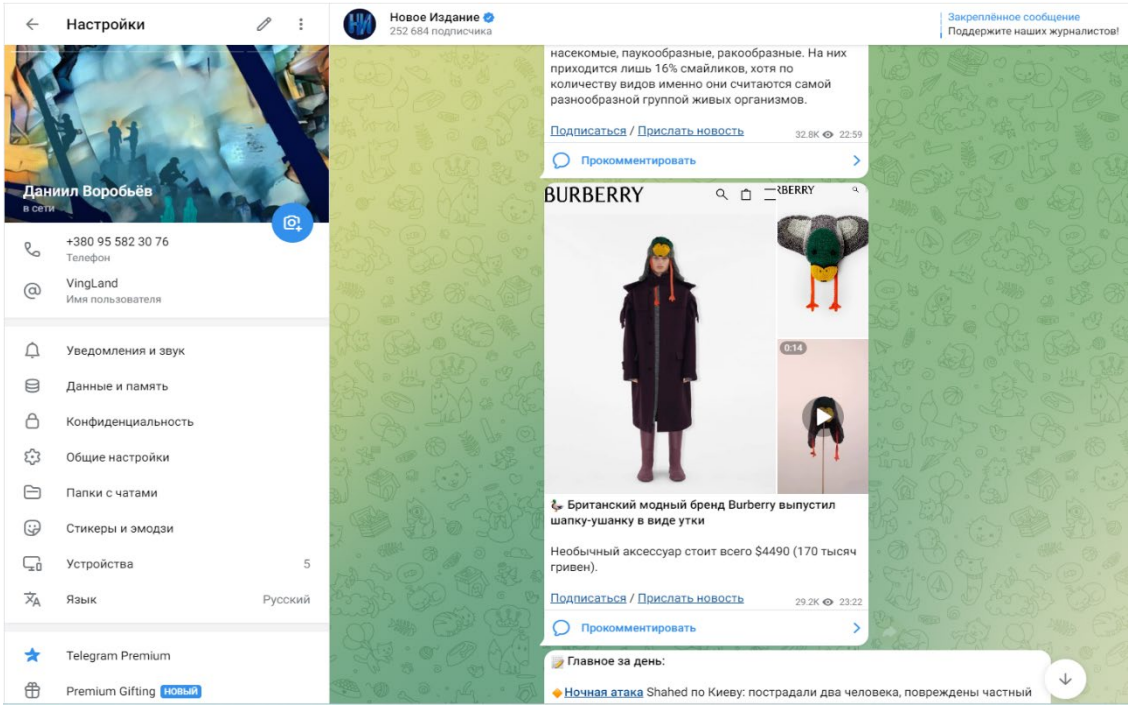

Рис. 1.4. Приклад інтерфейсу «Telegram»

Взагалі слід зауважити, що самі по собі боти є дуже популярними у багатьох сферах, а не тільки в соціальній мережі чи месенджері. Вже згадані раніше ігрові боти є лише малою копією того, що являють собою окремі застосунки. Для прикладу можна взяти ботів які проводять різні тренінги,

допомагають людям тренувати певні навички або можуть розробити для вас розпорядок вашого тренування. Багато з таких здатні розробити цілу тренувальну програму та займатись розпорядком вашого дня чи створити власний план індивідуальних тренувань. Окремо можна відзначити цілі автоматизовані системи, наприклад, ті що здатні стежити за ринком акції, моделювати поведінку та прогнозувати події, зробити аналіз доходу чи збитку, зробити висновки та надати план для подальших дій. Зазначимо, що на цьому сфера діяльності ботів навіть близько не обмежується. По суті боти є універсальним продуктом, які здатні автоматизувати великі обсяги роботи у самих різних сферах. Для прикладу наведемо медицину та програми, які здатні зробити аналіз хвороби пацієнта дослідивши певні показники або зробивши аналіз історії хвороби тощо. Також подібні програми використовують наприклад в гірництві, інженерії і не тільки. Вони дуже допомагають при моделюванні процесів, явищ, предметів, будівель тощо.

Важливо зауважити, що попре схожість на перший погляд з «нейромережами», боти є окремим програмним продуктом, який не може симулювати роботу мозку людини та як правило не здатний навчатись. Окрім цього принциповою різницею є менша придатність до обробки інформації, лише виокремленні алгоритми дії та робота у конкретній сфері. Нейромережі в свою чергу можуть обробляти дуже великі обсяги даних, здатні працювати з різними типами даних у різних сферах діяльності. Моделюючи поведінку людини вони набагато менше залежать від певних параметрів, які були задані на старті роботи. Тобто, якщо зробити підсумок, то боти є продуктом більш малим, не такого високого класу, але нічого не заважає доповнювати свої боти іншими програмними засобами, в тому числі інтеграціями у них нейромереж.

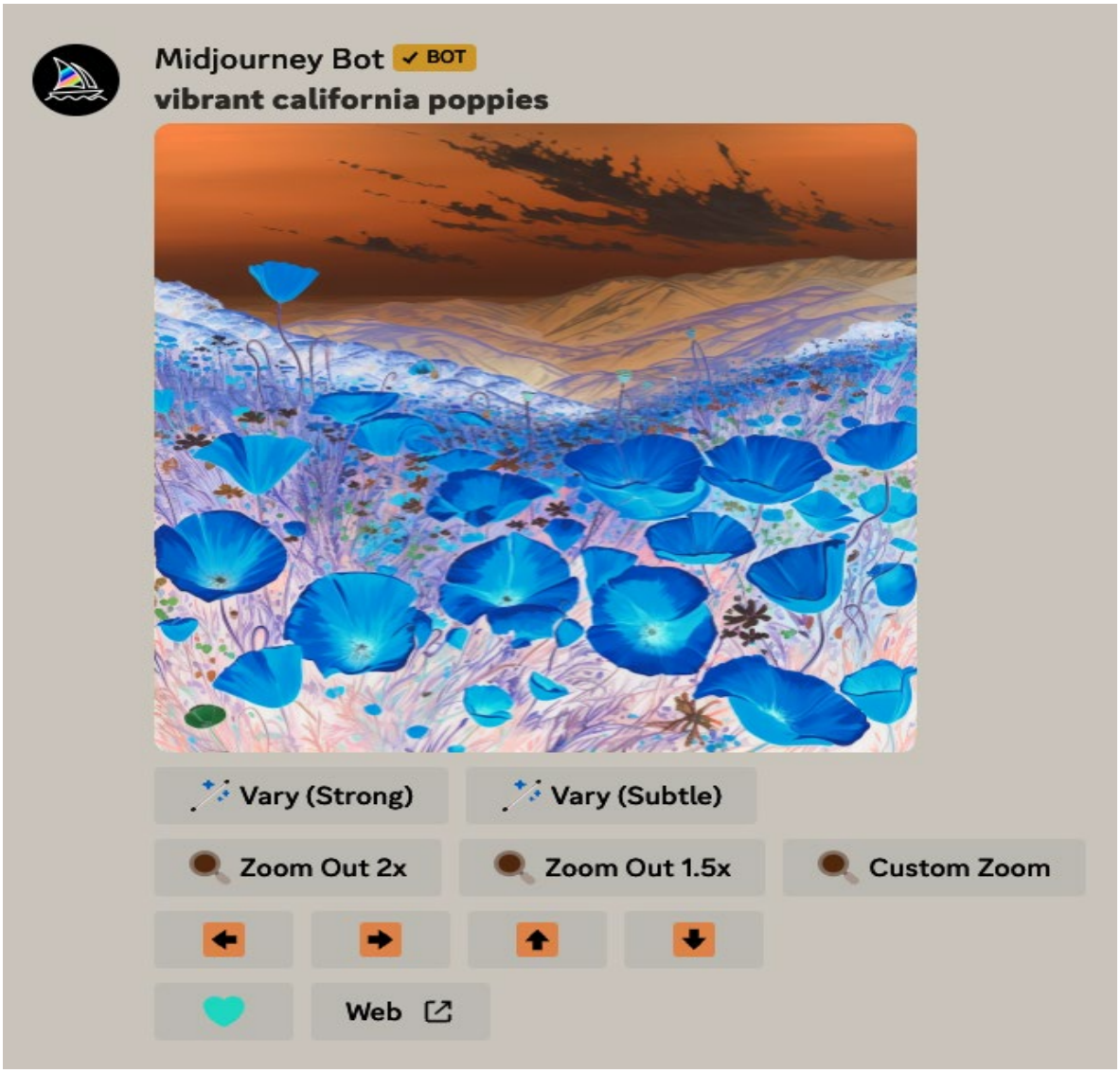

Рис. 1.5. Приклад інтеграції нейромережі «MidJourney» до бота

В незалежності від того, в якій конкретній сфері використовуються боти та інші подібні програми, важливим фактором залишається інтерфейс. Жодний продукт не буде популярним, якщо він буде мати некрасивий або неробочий інтерфейс. При створенні інтерфейсу професійні і досвідчені розробники та компанії акцентують свою увагу на розробку саме інтерфейсу. Важливо балансувати між кольорами та формами об’єктів. Ніхто не захоче дивитись на картинку, якщо вона буде випалювати тобі очі, агресивною, занадто яскравою. В той самий час не вийде привабити своїх потенційних клієнтів кольорами, які є занадто пасивними. Окрім цього важливо звертати увагу на те, для якого регіону призначений продукт, яка його цільова

аудиторія. Є багато народів та національностей, через це в багатьох є популярні кольори і ті які краще не використовувати. Інколи можна відштовхнутися від того, що саме за продукт має вийти на ринок. В середньому популярними є синій, зелений, жовтий, помаранчевий, рожевий та фіолетовий кольори. Інколи їх комбінують з білими, чорними та червоними кольорами для більшого ефекту, адже самі по собі вони бувають зайві та можуть відштовхнути клієнта. У випадку з формами продуктів або деяких візуальним об’єктів застосовується той самий принцип. Зазвичай класичними формами є овали та кола, ромби, квадрати чи трикутники. Далі все залежить від сфери застосування продукту та регіону.

Окремо слід відзначити, що інтерфейс це дещо більше ніж просто візуалізація та форми об’єктів. Тому коли ми кажемо, що інтерфейс повинен працювати і не викликати у клієнта нудьгу та відторгнення, мається на увазі як раз той факт, що інтерфейс є поняттям широким. Сюди входить і безпосередньо швидкість роботи програми, затримки, сам програмний код, пристрої вводу та виводу, набори правил та протоколів, а також багато чого ще.

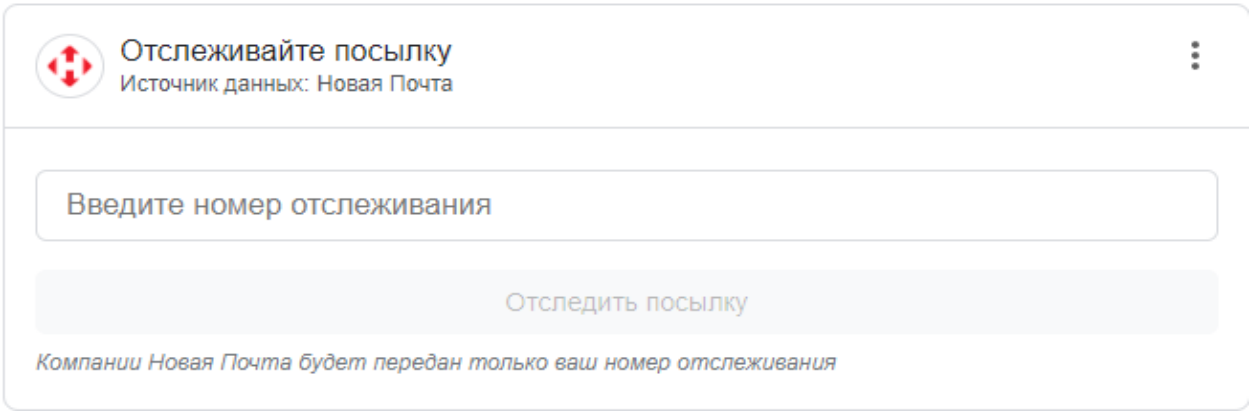

Рис. 1.6. Приклад пошуку посилки від компанії «Нова Пошта»

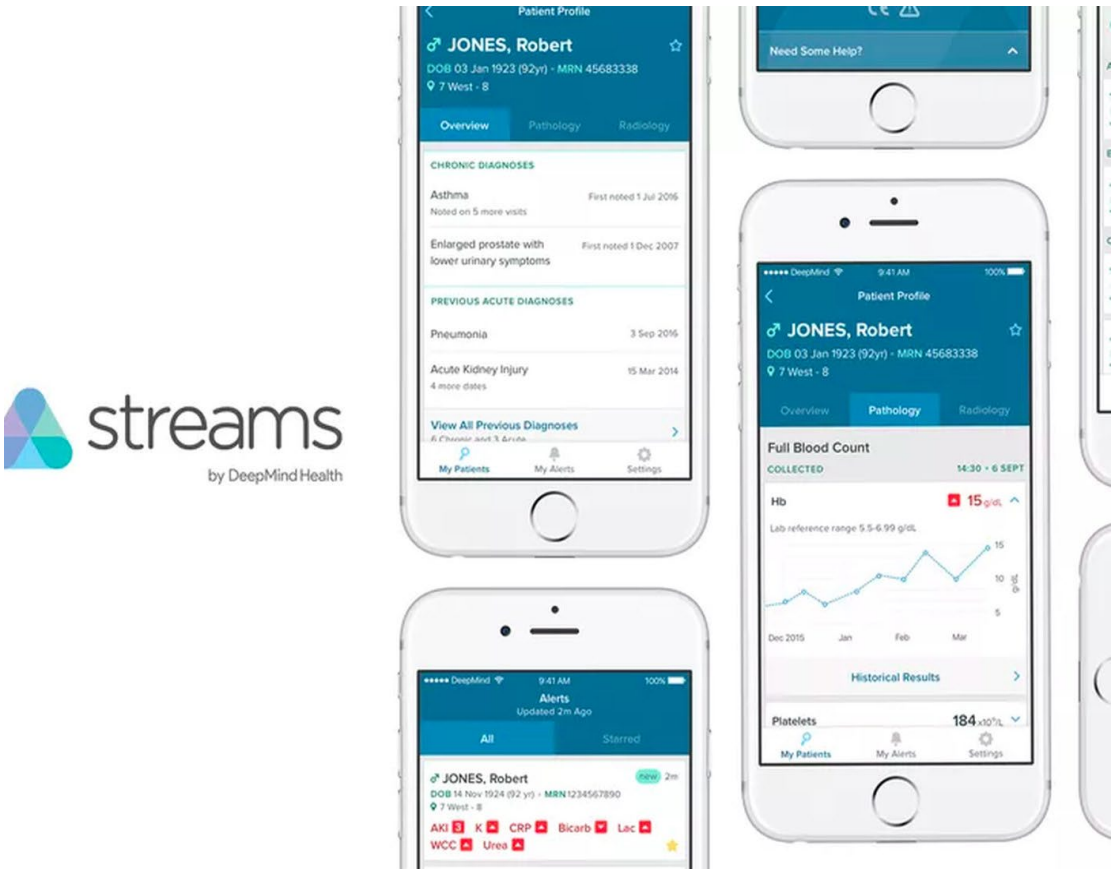

Рис. 1.7. Приклад програми для автоматизованого аналізу «DeepMind Health»

Тобто якщо робити підсумок, то можна побачити, що боти є дуже популярними та корисними. Серед них є безліч тих, які були кудись інтегровані чи щось було інтегровано у них. Використання даних програм для бізнесу є вкрай необхідним для конкуренції та збільшення прибутку. Для створення такого типу програм використовують різні мови програмування, веб-сервіси, штучний інтелект, багато API платформ, фреймворків та купу інших речей.

1.2. Перші інтерфейси та історія їх розвитку

На самому старті, коли комп’ютери та комп’ютерні технології тільки почали свій шлях до розвитку, одними з перших стали комп’ютери ENIAC та UNIVAC і відповідно інтерфейси до них також. ENIAC розшифровується як

«Electronic Numerical Integrator and Computer». Він став першим програмованим комп’ютером у 1945 році та займав площу більше ніж 160 квадратних метрів з масою близько 30 тонн і споживанням електроенергії до 150 кВт. Розробка та вартість такого оцінюється майже в пів мільйона доларів. На меті ж стояли розрахунки та дослідження для балістичних таблиць, але цим він не обмежувався там мав можливість бути перепрограмованим і виконувати інші завдання. Управляти ним можна було за допомогою зміни комутаторів та долучення різних проводів.

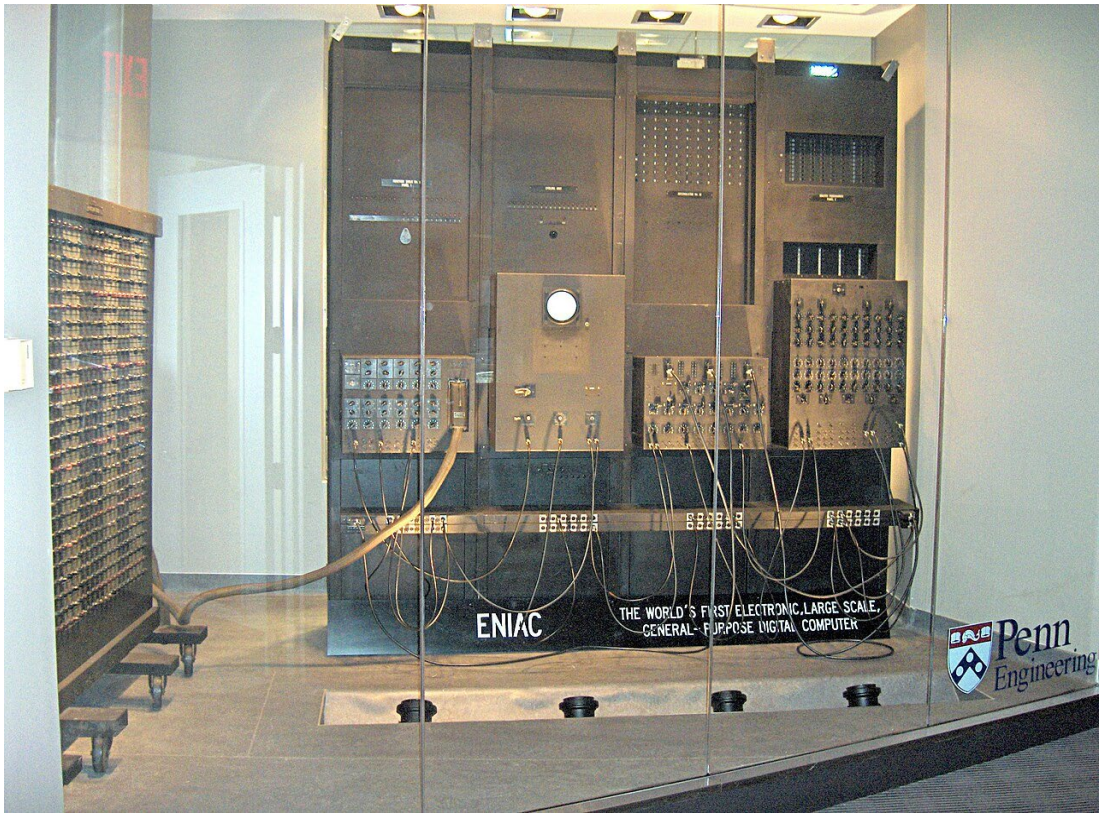

Рис. 1.8. Зображення залишків ENIAC

UNIVAC в свою чергу з’явився з назви самої компанії виробника. Перший такий було створено у 1950 році та називався він «Universal Automatic Computer». Він потребував 125 кВт, його вага була близько 13 тонн, а його площа була приблизно 35 квадратних метрів. Для свого створення він потребував декілька тисяч електровакуумних ламп та працював на частоті рівно 2.25 МГц. Комп’ютер використовував текстовий інтерфейс для використання команд та вводу даних. Для керування машиною був створений

спеціальний пульт, а також використовували стрічкові накопичувачі «UNISERVO», осцилограф та електричну друкарську машинку. Під час створення, основні свої задачі він мав виконувати для звітно-ревізійного офісу управління військово-повітряних сил та топографічної служби армії США.

Рис. 1.9. Фотографія UNIVAC 1

Дані комп'ютери мали занадто примітивні інтерфейси, дуже сильно залежали від технічних навичок виконавця та знання мов програмування. Відповідно і використовували ці машини лише висококваліфіковані спеціалісти.

З появою перших терміналів на початку шістдесятих років робота спеціалістів стало помітно легшою. У 1963 році з'явився перший «Teletype Model 33». Він був дуже доступним та простим. Серед його можливостей була взаємодія з іншими комп'ютерами та передача текстової інформації. Також важливим елементом була його клавіатура, одна з перших. З її допомогою можна було написати текст, а потім ще й роздрукувати його за допомогою вбудованого механізму. Як стає зрозуміло, такі машину з самого початку створювалися для використання у офісах. Попре все це, даний апарат мав також можливість відгукуватися, оскільки його можна було використовувати в комутованих мережах.

Рис. 1.10. Teletype Model 33

На початку 1972 року на ринку почали з’являтися перші «IMB 3270» від компанії IBM. Ці термінали були більш досконалими, в них були вбудовані інтелектуальні функції і вони мали базовий ввід та вивід. Даний термінал міг передавати інформацію «блоковим» типом, а також не залежав від іншого обладнання, що зіграло роль і дало можливість сильно збільшити продуктивність машини. Як результат, центральний «мейнфрейм» даного апарату зміг підтримувати обмін інформацією навіть з декількома тисячами таких машин. Також збільшена продуктивність дала можливість збільшити і відстань розташування від одного ЕОМ до іншого. Згодом машина мала можливість бути модифікованою і отримала собі функціональні клавіші з можливістю їх запрограмувати. Ще була додана графічна модифікація та відповідно кольори.

Саме ця серія терміналів стала початком для безлічі його копій, а також започаткувала новий клас програм, що звуться «емулятори». Певні моделі навіть можуть мати модемне з’єднання, вести сеанс за допомогою TN3270 (окрема модифікація для Telnet, розроблена спеціально для цієї справи) або по послідовному порту. Після цього усі нові версій стали фокусуватися на роботі з іншими платформами, наприклад, Java та Windows.

Рис. 1.11. IBM 3270

Навесні 1973 року робітники компанії Xerox PARC розробили персональний комп'ютер «Alto». Він мав у себе центральний процесор та читач для змінних жорстких дисків. У таких варіантах вже стандартно були клавіатура та миш. Даний апарат мав графічний інтерфейс з вікнами, текстовий процесор у якого були усі основні та найнеобхідніші функції. Сьогодні аналогами такого є «Libre Office Writer» або «Microsoft Word». Також був клієнт електронної пошти з нескладними функціями, чимось схоже на спрощену версію Apple Mail або Outlook. До усього цього додається і інше програмне забезпечення. Це вже був той рівень, коли з'явилась можливість вести розробку інших програм на об'єктно-орієнтованих мовах програмування, від сьогодення мало чим відрізняється. Цей пристрій мав мережевий інтерфейс, завдяки цьому його можна було під'єднати наприклад до принтера. У часи 70-х років це був самий справжній прорив, оскільки більшість пристроїв мали дуже великі розміри і поганий інтерфейс та способи взаємодії.

В подальшому така концепція дала змогу розробити у кінці 1983 року «Macintosh» від Apple, який у ті часи став улюбленцем серед багатьох. Сильно цьому посприяв і його інтерфейс, і вже тільки після цього розробили таку платформу як «Windows».

Рис. 1.12. Xerox Alto

Після того, як з’явились графічні інтерфейси, наступними були сенсорні панелі. Більшість розроблялась у 80-х, 90-х та 00-х роках, але перша сенсорна технологія була створена ще у 1971 році. Це був сенсорний екран з назвою «Elograph» віл компанії CERN.

З самого початку свого існування сенсорні технологія були створені для вирішення простих задач, наприклад, ввід та вивід інформації, керування інтерфейсом. З часом вони стали точнішими та отримали нові функції. Час показав, що є необхідність отримання нових технології, які будуть більш чуткі і матимуть кращий функціонал та точність. Ще у 80-х роках такі технологія були актуальними у банкоматах, на промисловості, кіосках з інформацією та деяких платформах самообслуговування. Майже в той самий час були створені інтеграції до комп’ютерів та мобільних пристроїв.

Рухаючись ближче до 2000-х років, використання сенсорних технологій стало невід’ємною частиною при виробництві смартфонів. Функціональність зросла ще більше, також піднявся і комфорт, у томи числі можна робити одночасно декілька дотиків. Станом на сьогодні ця технологія активно розвивається та має попит у електроніці, автомобілях, медичній сфері тощо.

Рис. 1.13. Сенсорна панель Neets Touch Pannel 10B

В даних момент часом популярними шляхами для розвитку стали технології доповненої реальності (AR/AUG) та віртуальної реальності (VR). Інженери намагаються досягти нових можливостей в плані занурення людини в зовсім інший простір. Вивчається також питання повного занурення.

Серед популярних систем доповненої реальності є Microsoft HoloLens. Це є спеціальна гарнітура, яка дає можливість користувачам взаємодіяти з проекціями віртуальних об’єктів у реальному світі. У промисловості така технологія використовується для створення інтерактивних моделей чи проекцій. Це допомагає при візуалізації та проектуванні об’єктів, що виконують роботу на підприємстві або необхідно розробити. Такий підхід значно покращує швидкість та точність проектування. Популярною технологія є і у медицині. В такій сфері діяльності важлива точна робота і розуміння того, що і коли робити, а також де конкретно. Так можна практикуватися у веденні хірургічних операцій чи навчати молодих лікарів. Також інколи HoloLens використовують при технічному обслуговуванні. Принцип роботи той самий, віртуальні проекції та добрий допоміжний засіб при вирішенні конкретних питань. Сильно допомагають можливості керування жестами та голосом.

Дуже популярним засобом також є і «Oculus». Платформа віртуальної реальності від компанії Meta. За допомогою спеціального пристрою людина

може зробити занурення в віртуальний всесвіт. Основною зоною використання технології є ігрова індустрія. За допомогою Oculus розробники можуть створювати нові ігри та контент. Схоже з HoloLens застосування у сферах медицини, промисловості, навчанні чи де-небудь ще. Окремо застосовують пристрій для віртуальних концертів, перегляду фільмів чи іншої взаємодії в віртуальній реальності. Такий опит запам’ятовується надовго і є незабутнім. Окрім цього є багато модифікацій та засобів менеджменту, наприклад, можливість самому створювати щось в іншому всесвіті.

Рис. 1.14. Microsoft HoloLens

Рис. 1.15. Технологія Oculus від Meta

1.3. Історія розвитку мов програмування

Одними серед найперших мов програмування були «Machine Code» та асемблери. Machine Code взагалі був першою мовою програмування, що складалася з різних інструкцій у двійковому коді, який в свою чергу відповідно був складним і незрозумілим для простої людини, якщо ми порівнюємо зі станом розвитку мов програмування на зараз. Взагалі, основною метою для асемблерів було поліпшення у справі написання програми з використанням мнемонічного коду для створення інструкцій процесорів, після цього було перетворення у код машини.

Серед мов програмування для штучного інтелекту одною з перших та популярних на той час була мова LISP (LISt Processing). Її використовували для роботи зі списками з даних та у розробці штучного інтелекту. Дуже гнучка система мови, можливість використання різних символів та менеджменту даних дуже сильно сприяли цьому процесу. Окрім того, те саме стосується і дерев за даними, символьних обчислень, списків тощо. Не можна не відзначити, що також дана мова використовувалась для спроб створення інших, нових архітектур, алгоритмів та мов програмування. Як не дивно, навіть станом на сьогодні, ця мова досі використовується місцями у промисловості та дослідженнях, не так часто у побудові роботів.

Для створення більш формальних, алгоритмічних та структурованих методів у програмуванні, використовували мову ALGOL (ALGOrithmic Language). Дана мова давала можливість читання коду та була зрозумілою. Окрім цього, до мови входили різні цикли, умовні оператори, блоки чи підпрограми. Така структура була зрозумілою та легкою для розробників. Мова ALGOL дала старт для інших мов програмування, серед популярних були різні мови «C», Pascal, об’єктно-орієнтовані мови та функціональні. Важливим чинником стала стандартизація мови на міжнародному рівні. Сфера використання мови включала у себе і інженерні чи наукові розрахунки, і різні дослідження, можливість навчати інших людей програмуванню. В історії

програмування дана мова цілком заслуговую на статус ключової, що дозволила сильно розвинути усю сферу в цілому, даючи дорогу для інших мов, структури та формальних методів.

ALGORITHM 35
SIEVE
T. C. Wood
RCA Digital Computation and Simulation Group, Moorestown, New Jersey

```
procedure  Sieve (Nmax) Primes: (p) ;
           integer Nmax; integer array p ;
comment    Sieve uses the Sieve of Eratosthenes to find all prime
             numbers not greater than a stated integer Nmax
             and stores them in array p. This array should be
             of dimension 1 by entier (2 × Nmax/ℓn (Nmax)) ;
begin integer n, i, j ;
           p[1] := 1 ; p[2] := 2 ; p[3] := j := 3 ;
           for n := 3 step 2 until Nmax do
begin      i := 3 ;
           L1: go to if p[i] ≤ sqrt (n) then a1 else a2 ;
           a1: go to if n/p[i] = n ÷ p[i] then b1 else b2 ;
           b2: i := i + 1 ; go to L1 ;
           a2 : p[j] := n ; j := j + 1 ;
           b1: end end
```

Рис. 1.16. Вирішення задачі за допомогою мови ALGOL

```
# Klisp interpreter v0.1.
> (+ 1 2 3)
6
> (reduce (seq 100) + 0)
4950
> (join (list 1 2 3) (list 'a' 'b' 'c'))
(1 2 3 'a' 'b' 'c')
> (def count-to (fn (max) (range 1 (inc max) 1)))
(function)
> (count-to 20)
(1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20)
> (prod (count-to 20))
2432902008176640000
>
```

Рис. 1.17. Використання мови LISP

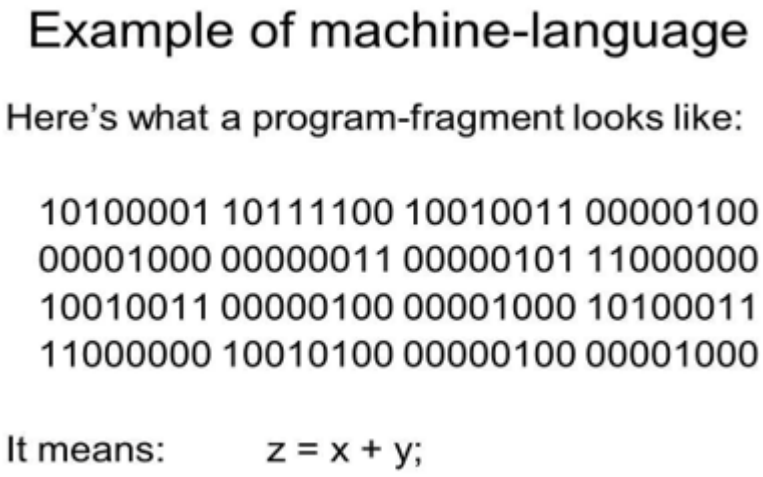

Рис. 1.18. Приклад старої «мови» Machine Code

Близько до 1970-х та 1980-х років популярними стали Fortran, COBOL, BASIC та Pascal серед мов програмування.

Не дивлячись на те, що Fortran (FORmula TRANslator) був розроблений ще у 50-х роках для інженерних та наукових дослідів і розрахунків, популярним він став пізніше. Серед його особливостей була робота з рівняннями та формулами. З точки зору ефективності він ідеально підходив для розрахунків так інших операцій з числами. Існує багато різних версій цієї мови, постійно додається щось нове, покращується, розробляються нові можливості. Серед них є окремі бібліотеки для наукових дослідів чи наприклад паралельне програмування.

COBOL, інакше кажучи, «Common Business Oriented Language». Ця мова також була розроблена у 50-х роках і стала популярною потім. Його використовували для роботи бізнес-процесів та вирішення комерційних питань. Вирішення різних задач у бізнесі та обробка даних є його основними задачами. У цьому випадку зрозумілість коду також була важливою при розробці. Мова була розроблена так, аби аналітики могли з цим працювати, а вчити інших людей було легше. Як стає зрозумілим, використання цієї мови було поширене у фінансових організаціях, сферах банку та деяких державних підприємствах.

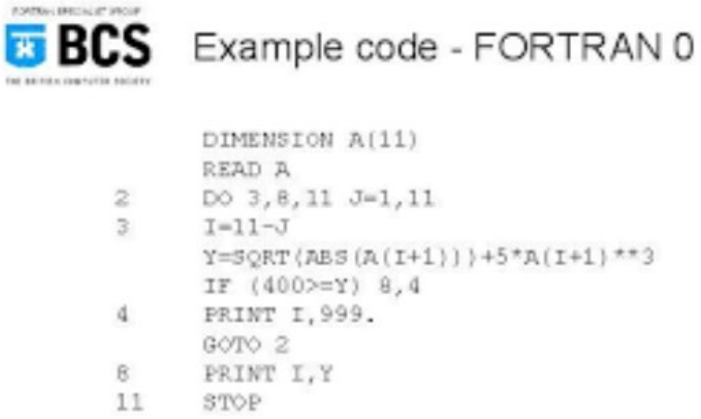

Рис. 1.19. Приклад коду «фортран»

```
def add_numbers():
    print("here is the first number.")
    first_number = 8
    print("Let's add 20 to that number.")
    first_number += 20
    print(first_number)
    print("create a second number.")
    second_number = 30
    print(second_number)
    result = first_number + second_number
    print("the result is: " + str(result))
add_numbers()
```

Рис. 1.20. Код COBOL

Ще однією популярною мовою був BASIC. Розшифровується як «Beginners all-purpose symbolic instruction code». Мова була створена в 60-х роках та, як стає зрозумілим з назви, використовується для навчання інших людей та створення дуже легких програм. Безумовним плюсом для мови є низький рівень необхідних початкових знань та простота розуміння. Будучи мовою, що запрограмована для початківців має зрозумілу інструкцію та нескладний синтаксис. Свою популярність даний проект набрав завдяки інтерактивному режиму для роботи, це значно спрощує процес навчання програміста.

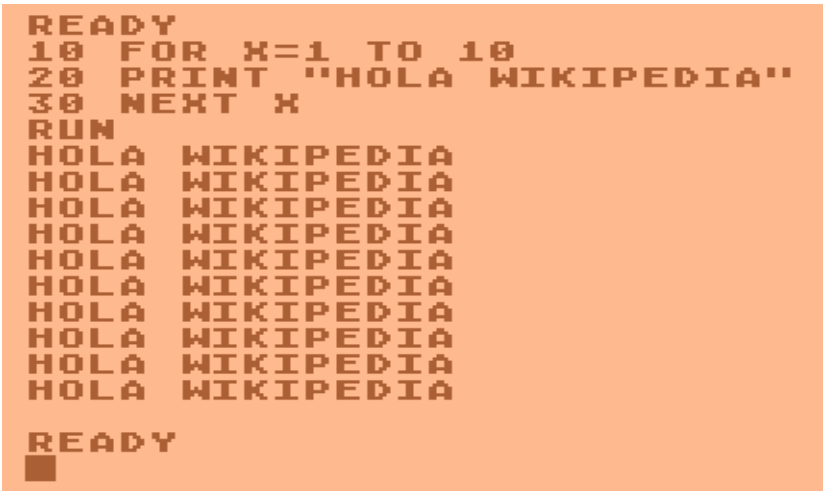

Рис. 1.21. Мова «BASIC»

Також, у ці самі часи став популярним Pascal. Основною метою було створення мови програмування, яка могла б навчати структурованому програмуванню та буди не такою складною. Серед плюсів є підтримка структурованого програмування для створення програм з керуванням та можливістю читання. Pascal має досить високий рівень безпеки, це зумовлене перевіркою границі масиву та суворою типізацією даних.

В цілому, кожна з мов у свій час створила великий вплив у своїй сфері розташування та дала майбутнє. Зараз це дозволяє нам легше вивчати мови програмування, мати красиві і зручні програми, а також сильні персональні комп’ютери.

```
while a <> b do  WriteLn('Waiting');

if a > b then WriteLn('Condition met')   {no semicolon allowed!}
          else WriteLn('Condition not met');

for i := 1 to 10 do  {no semicolon for single statements allowed!}
  WriteLn('Iteration: ', i);

repeat
  a := a + 1
until a = 10;

case i of
  0 : Write('zero');
  1 : Write('one');
  2 : Write('two');
  3,4,5,6,7,8,9,10: Write('?')
end;
```

Рис. 1.22. Мова Pascal

Наближаючись до 90-х та 2000-х років, з'являються більш сучасні і відомі нам мови програмування. Такими є мови «C», Java та наш улюблений Python.

Мова «C» була розроблена у 70-і роки та була низькорівневою і одною з перших у даному плані. Її використовували для створення OS (операційних систем) і відповідно системного програмування в тому числі. При створенні мови було важливо мати інструменти створення програм, що могли б бути перенесені з однієї платформи на іншу і при цьому були ефективними. Вплив на розвиток інших мов був дуже значним, з цієї мови були узяті основні концепції та синтаксис. З часом з'явилися такі стандарти як ANSI C, які допомагали при створенні легких програм, покращували мову і її можливості. Ближче до початку 80-х років з'являється розширення C++ для об'єктно-орієнтованого програмування. Це розширення додало поліморфізму, концепцій класів, абстракцій та спадкоємство. Плюсом стали великі бібліотеки, можливості розробки високопродуктивних програм там програмування з шаблонами. Постійно додавались і оновлювались різні стандарти ISO, це значно збільшувало оптимізацію, додавало можливостей та покращення.

```cpp
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    cout << "Hello, SkillFactory";
    for (int i = 0; i <= 16; i++)
    {
        cout << "2 ^ " << i << " = " << pow(2, i);
    }

    return 0;
}
```

Рис. 1.23. Мова C++

Мова Java створена в 1990-х роках для розробки крос-платформний програм. Це означає, що в незалежності від типу платформи, якщо вона має встановлену Java машину, то на цьому пристрої можна вести розробки або запускати застосунок. Зазначимо, що для об'єктно орієнтованого програмування Java також має функції. В процесі своєї еволюції, мова постійно отримує оновлення, які додають безпеки, новітні функції та роблять продуктивність вищою. JavaScript був розроблений близько у тому ж році. Його використовують, як правило, для розробки інтерактивних елементів на сайті та динамічних елементів сайту. Дуже поширена ця мова у «браузерах». Мова має типи даних, які визначаються під час роботи програми, оскільки вона має динамічну типізацію. Пізніше, коли з'явився стандарт ECMAScript, мова постійно наповнюється новим функціоналом, розширюється, додаючи можливостей.

Мова «TypeScript» створена компанією Microsoft в 2012 році. Це був «суперсет», який додав статичну типізацію. Він дозволяє спростити розробку застосунків та підвищує надійність програм, все завдяки визначенню типів даних. Дану мову можна назвати розширенням для JavaScript, вона дозволяє з часом додавати типізацію і використовувати вже існуючий код. Дане розширення постійно покращується розробниками. Вони випускають нові інструменти для інших розробників, покращують типізацію, додають інших можливостей. Кожна з представлених мов є різноманітною та надає широкий асортимент для розробника, а це сприяє розробці, навчанню та досвіду.

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  myBool: boolean = false
  title = 'angular-app';

  onBtnClick(event) {
    const reportContainer = document.getElementById('reports');
    this.myBool = !this.myBool;
    event.target.value = 'Send this data to the homepage'
  }
}
```

Рис. 1.24. Розширення «TypeScript»

Всім відомий Python було створено у 1991 році, його розробив програміст з Нідерландів. Основні задачі, які мав виконувати Python – зручне та ефективне програмування, яке буде зрозумілим і не викликатиме складнощів. Перш за все робиться акцент на зрозумілий прозорий синтаксис. Для початківців це вкрай необхідна річ. В мові присутні одразу декілька різних сфер програмування – функціональне, процедурне та об'єктно-орієнтоване програмування. Використовувати і програмувати на Python можна на таких платформах як macOS, Linux, Windows тощо. Кількість прихильників даної мови велика, це посприяло створенню великої кількості модифікацій, бібліотек та модулей, що значно спрощують розробку. Мова використовується для створення веб-застосунків. Для цього є такі популярні фреймворки як Pyramid, Flask або Django. Популярна ця мова і серед інженерів та вчених. За допомогою неї робляться наукові розрахунки, розробляється штучний інтелект, обробляється велика купа даних. В машинному навчанні є бібліотеки з широким використанням NumPy, Pandas, TensorFlow тощо. Дуже часто Python застосовують для автоматизації різних задач (наприклад, в ботах), створення «скриптів», для мережевого чи системного адміністрування. Завдяки простоті Python та здатності до читання, він підходить для створення прототипів програм та навчання. Як бачимо, Python знаходиться серед найпопулярніших мов програмування не просто так, його універсальність, потужність дають можливість широкого застосування.

Рис. 1.25. Створення гри на Python

1.4. Поява ботів та подальший розвиток в мережі Інтернет

Перші боти почали з'являтись в 1960-х роках. Прикладом одного з ранніх ботів є «ELIZA». Бот був створений Джозефом Вайзенбаумом у Массачусетському інституті технології 1966 року. Це була програма, яка була здатна імітувати розмови з психотерапевтом. Можна було задавати питання, надаючи відповідь основану на одному з шаблонів.

Впродовж розвитку штучного інтелекту і технологій, боти почали поширюватись і отримали більший функціонал. В кінці двадцятого століття з'явились «чат-боти» і підштовхнуло всю галузь вперед. В 2001 році був випущений «SmarterChild» бот. Він є одним з самих відомих чат-ботів свого часу. Він мав можливість спілкуватися з користувачами через месенджери.

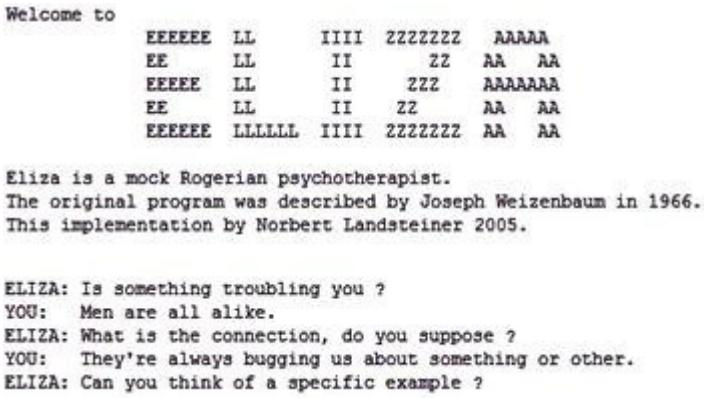

Рис. 1.26. Бот «ELIZA»

Рис. 1.27. «SmarterChild» бот

При появі месенджерів та соціальних мереж Facebook, WhatsApp, Instagram чи Telegram, боти стали ще більш популярними і отримали широку аудиторію. Програмні інтерфейси (API) від соціальних платформ дають змогу створювати, вдосконалювати та використовувати ботів з різним функціоналом. Прості відповіді на запитання, автоматизовані сервіси надання інформації і послуг, інші послуги та розваги, все це є в сучасних ботах. Більш детальні приклади вже були наведені в пункті 1.1.

Безумовним проривом в сфері ботів та чат-ботів став «ChatGPT». Цей чат-бот оснований на штучному інтелекті і здатен робити аналіз та генерацію тексту, надаючи людям підтримку в питаннях та необхідну інформацію. Призначенням програми є допомога людям, систематизація інформації та покращення її розуміння. Щодня бот надає допомогу мільйонам людей, спілкується з ними та надихає нових розробників. Станом на сьогодні вже відомі декілька випадків, коли даний бот допоміг врятувати життя людей, а також став корисним для медичних працівників. Саме тому ця технологія безумовно зробила колосальний внесок в розвиток сфери і життя людей.

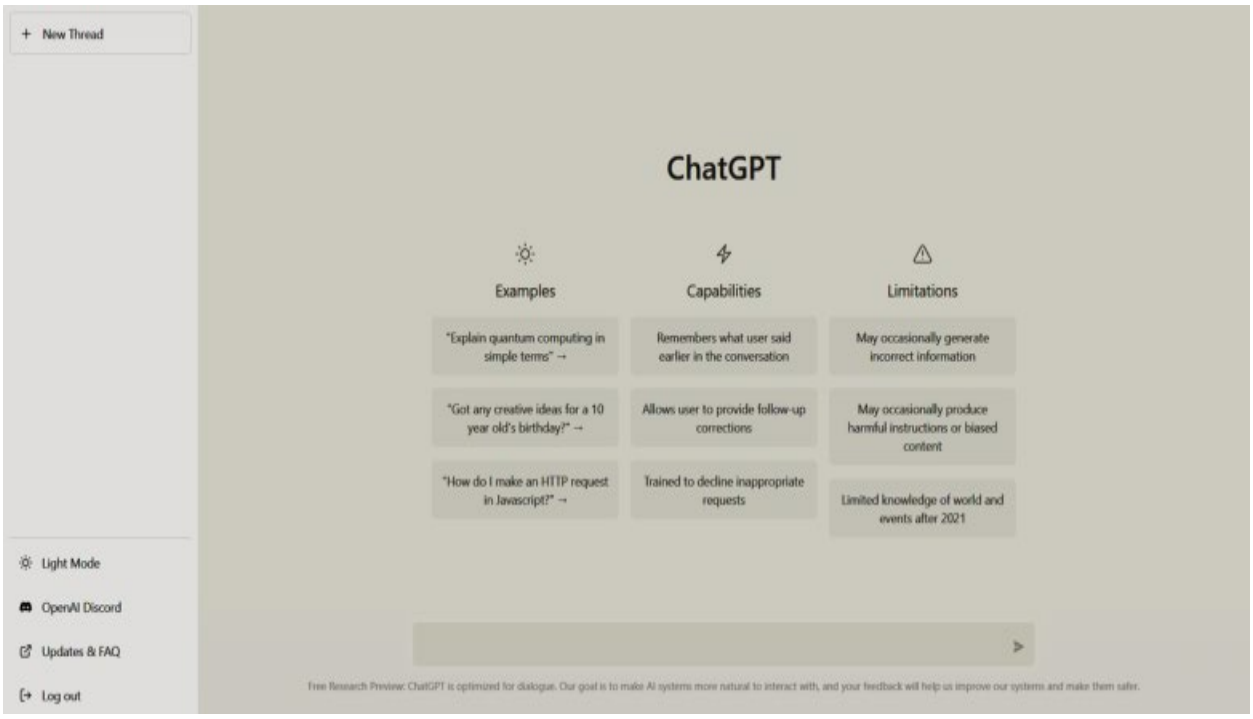

Рис. 1.28. Інтерфейс чат-боту «ChatGPT»

1.5. Формулювання задачі в дослідженні ефективності використання ботів Telegram для ведення бізнес-процесів

В якості головних задач магістерської кваліфікаційної роботи стоїть розробка власного Telegram бота для бізнесу на мові програмування Python з використанням самого популярного і зручного для цього фреймворку «Aiogram», а також подальшого дослідження ефективності роботи даного бота. А саме, дослідження швидкості роботи бота, в тому числі, швидкості запуску та реакції бота на команди в чаті. Окрім цього, необхідно порівняти можливості бота, його швидкість та ефективність з вже існуючими аналогами на ринку. Дослідити прибуток від використання бота. Зібрати початкові дані після введення бота в експлуатацію. В разі необхідності зробити оновлення бота для урахування усіх недоліків та збільшення можливості вести конкуренцію, приваблювати клієнтів та отримувати прибуток.

Для проведення даного дослідження, збору інформації та створення висновків було проведено аналіз даних. В зборі інформації допоміжними засобами є платформи «Айко» та «аналітика» від Google. В процесі дослідження використовувались методи спостереження та порівняння.

1.6. Вимоги до програмного продукту для розробки та дослідження

Серед заявлених характеристик у бота обов’язково повинні бути – блокування повідомлень які не проходять цензуру, привабливі, прості для розуміння кнопки на клавіатурі бота, швидкі відповіді на команди бота, запитання до бота чи контрольні слова. Даний бот повинен мати можливість збирати статистику про використання сервісу компанії та його самого. Після вводу бота в експлуатацію, потрібно зробити оновлення для бота, аби видалити недоліки та вирішити проблему з тим, чого бракує.

Для того аби створити сам бот, необхідно отримати «токен» для розроблюваного боту від менеджера ботів Telegram. Потрібно скористатись веб-версією сайту або використати версію для робочого столу. Потім в поле

для пошуку груп і контактів ввести «@BotFather». В знайденому боті натиснути кнопку «старт» або ввести команду «/start». Серед усіх команд, які запропонує бот, необхідно обрати «створити нового бота». Треба буде дати йому його ім’я та адресу. Після закінчення налаштування необхідно буде скопіювати отриманий токен бота та додати його до розробленого застосунку. Даний токен необхідно буде використати для створення файлу через який бот буде працювати.

Саму розробку продукту необхідно вести за допомогою мови програмування Python та виконувати інструкції з документації до ботів від розробників Telegram. Використовувати фреймворк «Aiogram». Написавши кодову структуру для бота, треба переконатись, що бот спокійно виходить в режим роботи.

За зберігання персональних даних та конфіденційність можна не хвилюватись. Розробники API Telegram заздалегідь про все подбали та розробили безпечний режим «private». Його налаштуванням може займатись і сам користувач. Коли такий режим працює, бот реагує лише на обмежену кількість команд та повідомлень, не обробляє зайву інформацію. Такий підхід дозволить максимально захистити клієнта від втрати його особистих даних. Технічно залишається лише один варіант втрати даних, у випадку коли клієнт сам надає свої дані та те як цими даними користується бот, куди їх поширює. Ваш особовий токен знаходиться в повній безпеці, оскільки ніхто його не бачить, а API сайту не дозволить ніяк його викрасти. Рівень кібербезпеки платформи постійно зростає, а розробники навіть видають винагороду тим, хто зможе зламати цей захист.

Для розробки бота необхідно мати встановлену версію Python на своєму комп’ютері або ноутбуку. Бажано користуватись версією телеграму для робочого столу та веб-версією. Також бажано мати версію Windows 10 або вище. Не рекомендується використання старих версій Python (нижче за 3-і версії). Необхідним буде редактор коду, для прикладу можна взяти розповсюджений «Sublime Text» або програмувати у «Visual Studio».

Серед технічних показників вам необхідно мати процесор з 2 ядрами або більше, оперативну пам’ять бажано від 8 ГБ. Встановлену у слот материнської плати або інтегровану відеокарту. Знадобляться і звичайні клавіатура та миш, окрім цього вільне місце на жорсткому диску, в залежності від кількості коду, може знадобитися від 10 ГБ вільного місця. Звичайно необхідним є швидкий доступ до мережі Інтернет. Перелічених показників буде достатньо для початку роботи.

Для використання бота, його необхідно завантажити на «хостинг» або запускати через файл з розширенням .bat. В разі використання і запуску програми з вашого комп'ютера, вище перелічених показників та software буде достатньо. Працювати з Python можна на більшості платформ, оскільки він є крос-платформний. Але важливо мати на увазі, що на «Windows xp» у вас не буде працювати Python, якщо його версія буде вищою за 3.3, а в системі «Vista» вище за 3.7.

Сам бот повинен мати інтерфейс, що буде зручним для користувача, легким, приємним і зрозумілим. Також бот має надавати саму необхідну інформацію, мати відповіді на часті питання і виконувати свої головні функції.

1.7. Висновки

Розробка ботів для соціальних мереж та месенджерів є дуже поширеною на сьогодні. Гарно спроектований бот дозволить бізнесу підвищувати кількість своїх клієнтів, а значить і збільшувати доходи. При створенні програмних застосунків такого типу важливо враховувати цільову аудиторію та сферу застосування боту. Також важливо використовувати ті мови програмування, які націлені саме на такі нескладні програми, а також фреймворки та розширення, що були створені спеціально для цього. За великий проміжок часу в процесі еволюції програмісти постійно покращували різноманітні бібліотеки, розробляли все більш потужні фреймворки та мови програмування. Сьогодні це дає можливість розробляти дуже продуктивні додатки з простою візуальною складовою, а також великим функціоналом.

РОЗДІЛ 2
РОЗРОБКА БОТА ДЛЯ ПЛАТФОРМИ TELEGRAM З ЦІЛЛЮ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ БОТІВ ДЛЯ ПОКРАЩЕННЯ БІЗНЕС-ПРОЦЕСУ

2.1. Призначення розробки

Під час виконання роботи був створений програмний застосунок, а конкретно бот для платформи «Telegram» в мережі Інтернет. Для створення даного боту була використана мова програмування «Python» та був обраний найбільш підходящий для цього фреймворк – «Aiogram». Бот можна використовувати для комунікації зі своїми клієнтами у бізнесі, надання можливості перевірити асортимент та послуги, які пропонує компанія, зробити замовлення, надати необхідну інформацію про заклад чи часті питання, окрім цього є і інші другорядні можливості. Також даний бот збирає інформацію про клієнта та самого себе, останнє необхідне для аналізу даних, спостережень і дослідження. Ця інформація збирається лише за згоди користувача та захищена від зайвого втручання програмним інтерфейсом самого сайту.

Конкретні можливості бота:

1) Для зберігання інформації та виклик її під час необхідності, була створена data base(db). База даних створена на основі «sqlite» третьої версії. Це досить просто у використанні та комфортна система для керуванням сховищами даних. Плюсом використання є можливість легко її інтегрувати в систему.

2) За допомогою спеціальних команд, які були закладені у розроблений програмний продукт під час створення, даний бот може надавати допомогу клієнту, показати своє меню, просто почати розмову, запросити номер телефону, спитати місцезнаходження або показати своє, видати часи роботи компанії, показує інші команди тощо. Для більш сприятливого опиту

користування, був розроблений лист команд і вкладений у бота, бот під час використання сам нагадує про існуючи команди, наприклад, команда «/help».

3) Вище описані в пункті 2 можливості бот має завдяки використанню фреймворку «Aiogram» та створеним за допомогою нього кнопкам на клавіатурі для бота під час спілкування. Кількість кнопок залежить від необхідних команд, загальних можливостей бота та безпосереднього налаштування бота і програмного коду. Під час розробки було прийнято рішення використати команди «row» та «add», суть яких лежить в формуванні розташування кнопок на клавіатурі, наприклад, в ряд чи в строку.

4) Також на протязі роботи створено клавіатуру для менеджерів чи, наприклад, адміністраторів компанії. Її призначенням є створення нових елементів у меню, а також видалення зайвих елементів, які більше непотрібні. Доступ до даної клавіатури мають лише працівники компанії, вхід виконується за допомогою окремої команди. Про команду не знають користувачі, її не видно через інтерфейс та мають доступ до неї лише працівники.

5) Оскільки одним з розділів роботи є дослідження, прийнято рішення про створення кнопки з посиланням. Суть такої функції полягає в можливості збору даних про використання клієнтами конкретної частини інтерфейсу бота з ціллю подальшого аналізу зібраної інформації. Окрім цього, конкретно ця кнопка призначена для популяризації боту з ціллю конкуренції, за допомогою відповідного відео на «хостингу», який надає такі послуги. Коротке відео здатне детальніше показати бота та пояснити суть його роботи. Значно легше буде пояснити принципи керування ботом на практиці, аніж намагатися пояснити через текст. Окрім цього, це додає свого роду опит. Одною з основних функцій бота, відповідно, є збір інформації за допомогою інших програм і сервісів, вже згаданих раніше – «Айко» та «аналітика» від компанії «Google».

6) Окрім вище вказаних функції та характеристик, боту дана можливість для аналізу конкретних слів. Бот здатний робити аналіз тексту та

знаходити ключові слова. З використанням цієї особливості бот краще розуміє, що необхідно клієнту: буде надавати йому потрібну інформацію, яку шукає користувач, або допоможе знайти її. На цьому ж принципі заснована ще одна особливість бота. В якості додаткової «фішки» бота, він буде реагувати на певні слова і видавати стислі відповіді в тему, чи надсилати картинки.

Умови створення бота та коректної його роботи:

1) Чітке, покрокове виконання вимог, що вже зазначені в першому розділі, в пункті шість.

2) Для коректної роботи необхідне використання власного «токену» бота та слідування за усіма правилами, що описані в документах від розробників месенджеру.

3) Вся необхідна програмна та кодова реалізація з використанням усіх необхідних сервісів. Обов’язкова перевірка здатності бота до запуску і з’єднанням з платформою.

4) Використання мови Python та фреймворку «Aiogram» для створення і якісної роботи. Створення з їх допомогою та інтеграція бази даних «sqlite».

5) Обладнання бота механізмами збору даних з подальшим їх збереженням та аналізом.

6) Зокрема, необхідним є відключення реакції бота на повідомлення, надіслані в момент коли бот не працював, це потрібно аби уникнути «флуду» в чаті.

7) Порушення правил може призвести до некоректної роботи бота.

2.2. Використання математичних формул та методів

Під час виконання роботи по створенню програмного застосунку та подальшого дослідження його роботи і ефективності, збору даних було використано декілька математичних методів та формул.

Серед конкретних формул були застосовані економічні формули. І хоча вони мають більше відношення до економічного розділу, де і будуть описані, але згадати їх також треба, оскільки з їх допомогою була визначена ціна розробки програми, її потенційний прибуток, час витрачений на розробку та обсяг роботи.

З використаних функції можна виділити lambda-функцію, що працює з будь-якими типами даних. Конкретно в програмуванні її називають анонімною або без імені. Сама вона бере початок з математики, там вона використовується у обчисленнях «лямбда». Таке обчислення є формальною системою. Ця система надає абстракції різних функцій, а також використання цих функцій. В процесі використовують спеціальні символи, а самі функції імені не мають. Так були створені анонімні функції у конкретних програмних мовах. В Python така функція може бути передана як аргумент до іншої і використовується в програмуванні під назвою «функціональне».

Була використана асинхронна функція під час створення бота. Прямого відношення вона звісно не має, але прийшла у програмування також від математичних концептів і засобів зі спільними елементами. В математиці є думки про паралельні обчислення. Це ті обчислення, які виконуються або незалежно одне від одного, або в один самий час. За аналогією, в програмуванні можуть виконуватись декілька задач асинхронно або паралельно, і для цього не потрібно очікувати завершення інших операцій. Також є певні концепції послідовності виконання та часу. В асинхронному програмуванні певні задачі можуть виконуватись в певному порядку, по різному або є залежними від інших факторів та вимог.

2.3. Використані мови програмування і технології при розробці

При написанні коду для боту в процесі розроби використовувались декілька мов та технологій.

Буквально один раз було використано один раз було використано JavaScript. Дана мова знадобилась для написання дуже простого скрипту перетворення звичайних слів у їх кодовий формат, що є більш зрозумілим для машини. Це знадобилось для реалізації цензурування тексту, яке в свою чергу необхідно для того аби зайва інформація видалялась з чату та було припинено використання поганих слів. JavaScript є дуже розповсюдженою мовою. Він користується популярністю у всьому світі і є однією з основних технологій для мережі Інтернет. Схожу популярність мають, наприклад, CSS та HTML. Зараз більшість сайтів, а конкретно більше ніж 97%, користуються даною мовою. Часто використовують різні бібліотеки та розширення при використанні JavaScript. Більшість браузерів можуть виконувати ті чи інші завдання на пристроях клієнтів за допомогою «скрипту». Це мова високого рівня, вона постійно слідкує за трендами та часом, тому і відповідає такому стандарту як ECMAScript, його вже було згадано раніше у першому розділі. Типи даних у мові є динамічними, сама мова об’єктно-орієнтована та заснована на прототипі, має першокласні функції. Присутня мультипарадигма, що здатна підтримувати функціональний стиль програмування, керований подіями та імперативний. Також є інтерфейс прикладного програмування для оброки тексту, популярними виразами, стандартними структурами даних, датами чи об’єктною моделлю документу.

ECMAScript як стандарт не включає ніяких засобів виводу чи вводу, таких як сховища, графічні засоби або мережеві. На практиці маємо ситуацію коли кожна система для виконання дає вам JavaScript API для виведення чи введення.

Механізми мови на старті використовували для веб-браузерів. Зараз це основні компоненти різних програм та на серверах. Одним з самих популярних є Node.js.

При тому, що звичайно Java схожа по назві, деякими бібліотеками та синтаксисом до JavaScript, вони є різними мовами, мають різне застосування та інтерфейс.

При створенні сховища даних використана технологія SQLite. Вона написана на програмній мові «С» та являється системою баз даних. Окремою програмою її важко назвати. Спосіб її використання більше схожий на бібліотечний. Це зумовлене тим, що це база даних для вбудовування в системи, тобто розробник має її «додати» до свого продукту. Серед багатьох механізмів баз даних, цей є одним з найпоширеніших. Користується багатьма операційними системами, мобільними телефонами, веб-браузерами та іншими системами, в тому числі і для вбудовування.

Велика кількість програмних мов з самого початку мають вбудовану SQLite. Як правило вони працюють за синтаксисом PostgreSQL, а за замовчуванням не перевіряють тип. В такому випадку можна вставити рядок у стовп в якості цілого числа.

Плюсом її використання є можливість все записати у файл на робочій машині без під'єднання до серверів. Таким чином ми автоматично отримуємо можливість використовувати менше ресурсів комп'ютера і збільшуємо швидкість роботи за рахунок меншого часу звернення. Є можливість отримувати інформацію від кількох потоків даних. Маємо підтримку динамічних даних з п'ятьма варіантами значень. Такі типи як txt чи blob не мають обмеженого розміру. Присутня константа максимального розміру в 1000000000 і вона лише одна. При оголошенні типу його початковий стан зберігається для випадків коли виникають проблеми при зміні типу і зі схожими назвами. Окреме значення має можливість бути будь-якого типу та в будь-якому полі, це залежить від того, що було вказано при оголошенні.

```
kousekip@ako-kaede-mirai ~ $ sqlite3
SQLite version 3.39.4 2022-09-29 15:55:41
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .show
        echo: off
         eqp: off
     explain: auto
     headers: off
        mode: list
   nullvalue: ""
      output: stdout
colseparator: "|"
rowseparator: "\n"
       stats: off
       width:
    filename: :memory:
```

Рис. 2.1. sqlite в командной строке

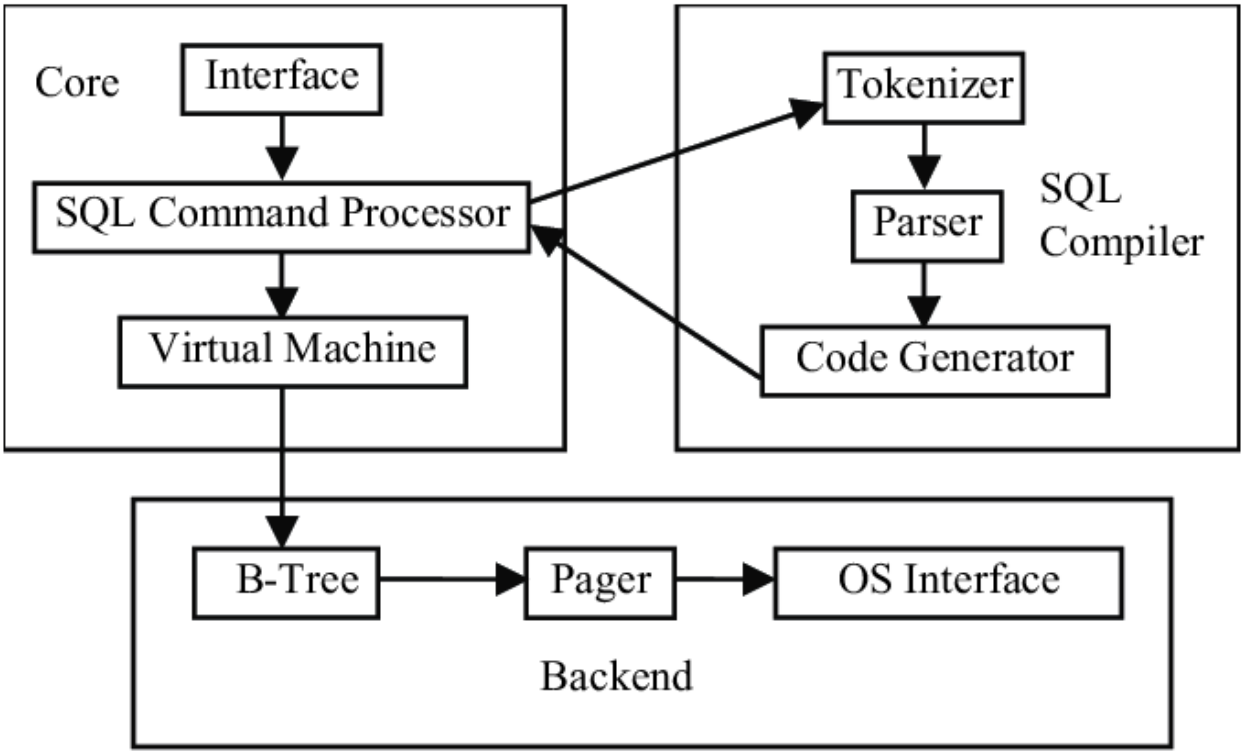

Рис. 2.2. Побудова SQL

```
15
16  const LOCALE = globalThis.navigator.language
17
18  const div = document.body.appendChild(document.createElement('div'))
19  const list = div.appendChild(document.createElement('ol'))
20
21  const dayNames = new Map()
22
23  for (let i = 0; i < 7; ++i) {
24      const d = Temporal.PlainDate.from({
25          year: Temporal.Now.plainDateISO().year,
26          month: 1,
27          day: i + 1,
28      })
29
30      dayNames.set(d.dayOfWeek, d.toLocaleString(LOCALE, { weekday: 'long' }))
31  }
32
33  for (const num of [ ...dayNames.keys()].sort((a, b) => a - b)) {
34      list.appendChild(Object.assign(
35          document.createElement('li'),
36          { textContent: dayNames.get(num) },
37      ))
38  }
39
```

Рис. 2.3. Приклад коду JS

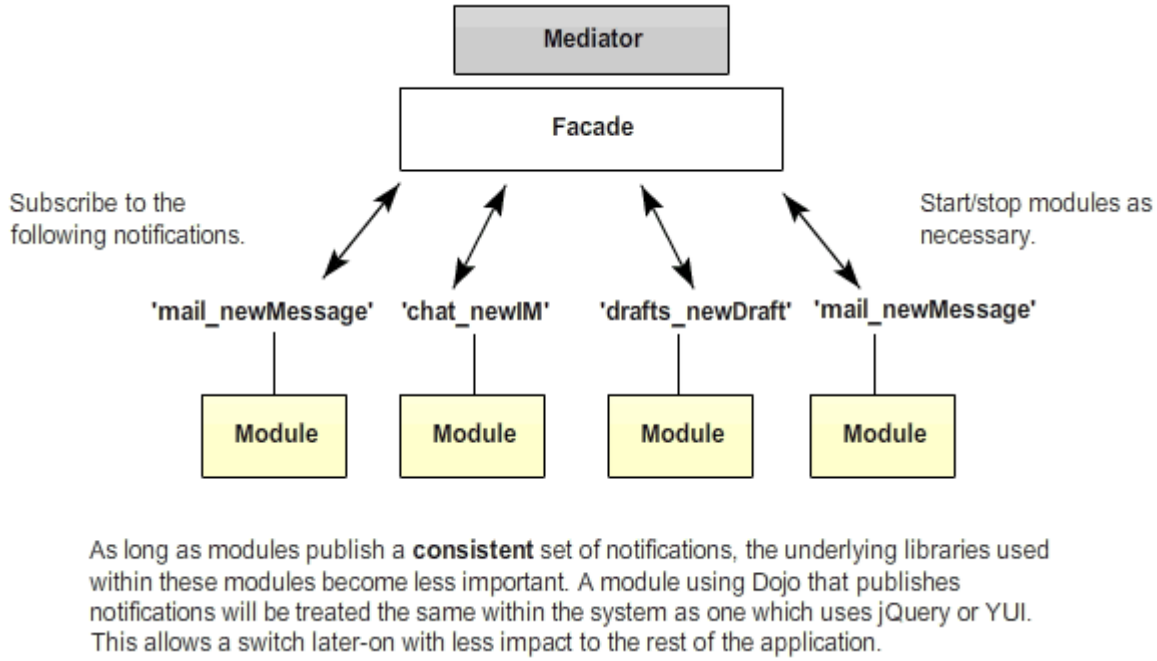

Рис. 2.4. JS модульна архітектура

Також, при розробці були використані безпосередньо інструменти та технології від самого «Телеграм». В матеріалі, поданому вище, вже було наведено приклад налаштування бота на сайті «Telegram». Окрім цих налаштувань, існують і інші. Всі вони знаходяться в інтерфейсі сайту да сприяють легкому користуванню.

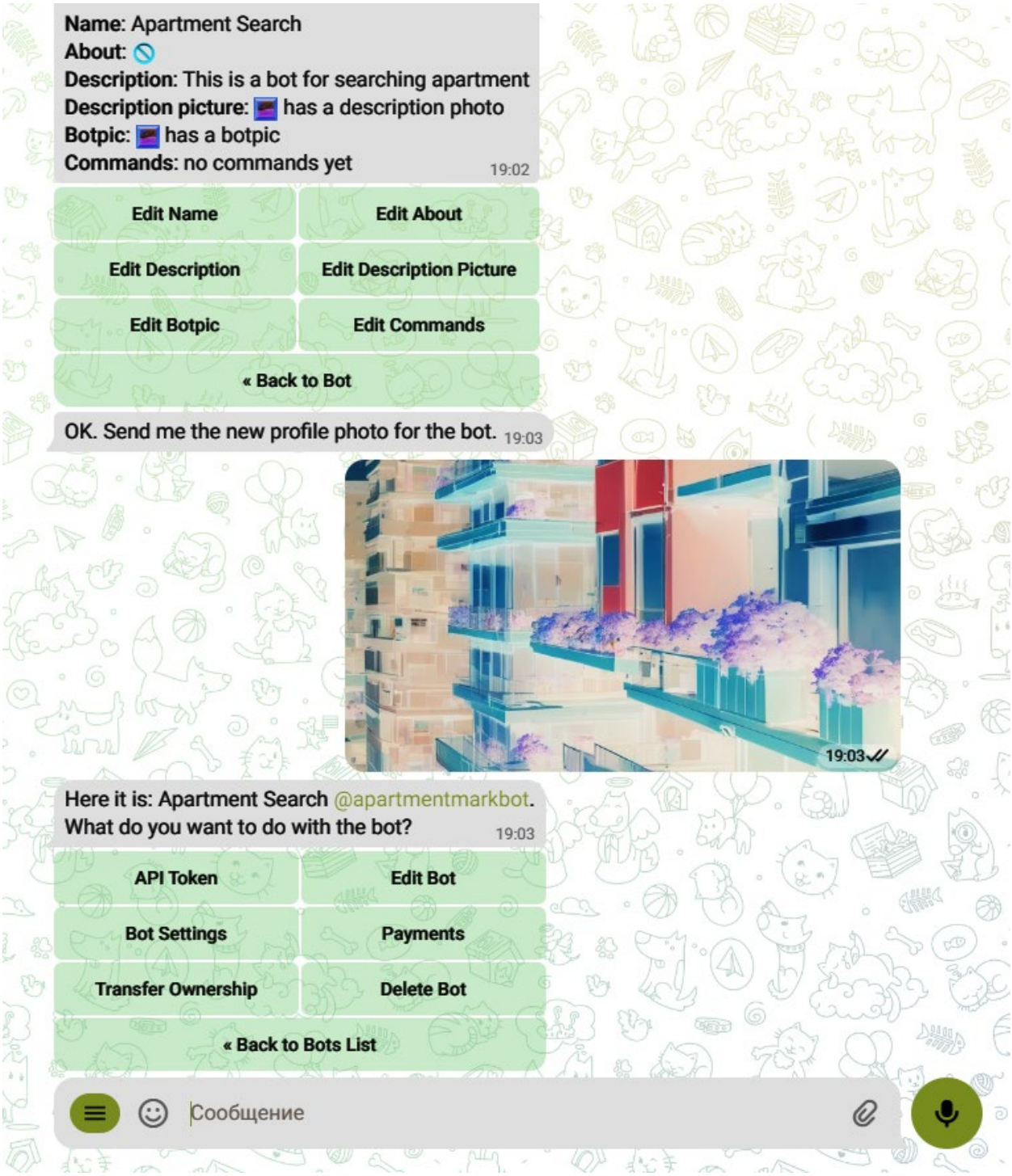

Рис. 2.5. Налаштування від «Телеграм»

Для створення та налаштування боту був використаний фреймворк «Aiogram». Він є спеціально створений для програмування ботів розробниками платформи «Телеграм», використовується фреймворк конкретно для цієї справи.

З моменту своєї появи на ринку в 2017 році, він дуже сильно змінився та отримав покращення. Сьогодні він активно розвивається та отримує більші можливості. В меті та необхідних умовах для створення бота вже було зазначено, що бот повинен швидко працювати, бути зручним та легким. Даний фреймворк було розроблено саме для цього. Aiogram дозволяє легко створити бота для Telegram, який буде швидким у використанні і процесі своєї роботи. Також, дана технологія спрощує сам процес розробки, це скорочує час розробки та дозволяє скористатися унікальними можливостями і функціями, потребує менших знань і навичок від програміста, за рахунок чого популяризується та має розвиток. Великим плюсом є широкий асортимент інструментів та асинхронний інтерфейс взаємодії з Telegram API.

Фреймворк володіє великою кількістю функцій, серед них є обробка файлів мультимедіа, керування за допомогою клавіатури, обробка текстових повідомлень і підтримка асинхронності. Останнє дозволить швидко обробляти запити і працювати в мережі.

Серед плюсів використання можна виділяти легкий, зручний, зрозумілий інтерфейс. За рахунок асинхронності покращується відклик від бота, продуктивність, тому це також плюс. Серед інших плюсів можна назвати велику кількість різних інструментів. В цілому фреймворк є дуже гнучким і дозволяє досягти потрібного результату.

Деякими мінусами можна назвати велику документацію та необхідність робити уточнення. Для того, аби зрозуміти всю цю складну систему, новим програмістам і може бути складно та потрібно багато часу, але не завжди.

Серед інших використаних технологій є звичайно сам Python. Про цю мову можна сказати, що вона є високого рівня. За рахунок цього новачкам легше її вчити, працювати з нею, створювати проекти. Також це означає, що у вас буде великий і зручний функціонал, багато бібліотек, додатків тощо. Зокрема, мова «пітон» має динамічну строгу типізацію. Тому ви можете легко змінювати типи змінних, вони будуть отримані тільки в той час, коли ви будете давати значення. Якщо ви розробник, для вас це буде великим плюсом, такий

підхід значно спрощує роботу, менше потребує від розробника та прискорює процес.

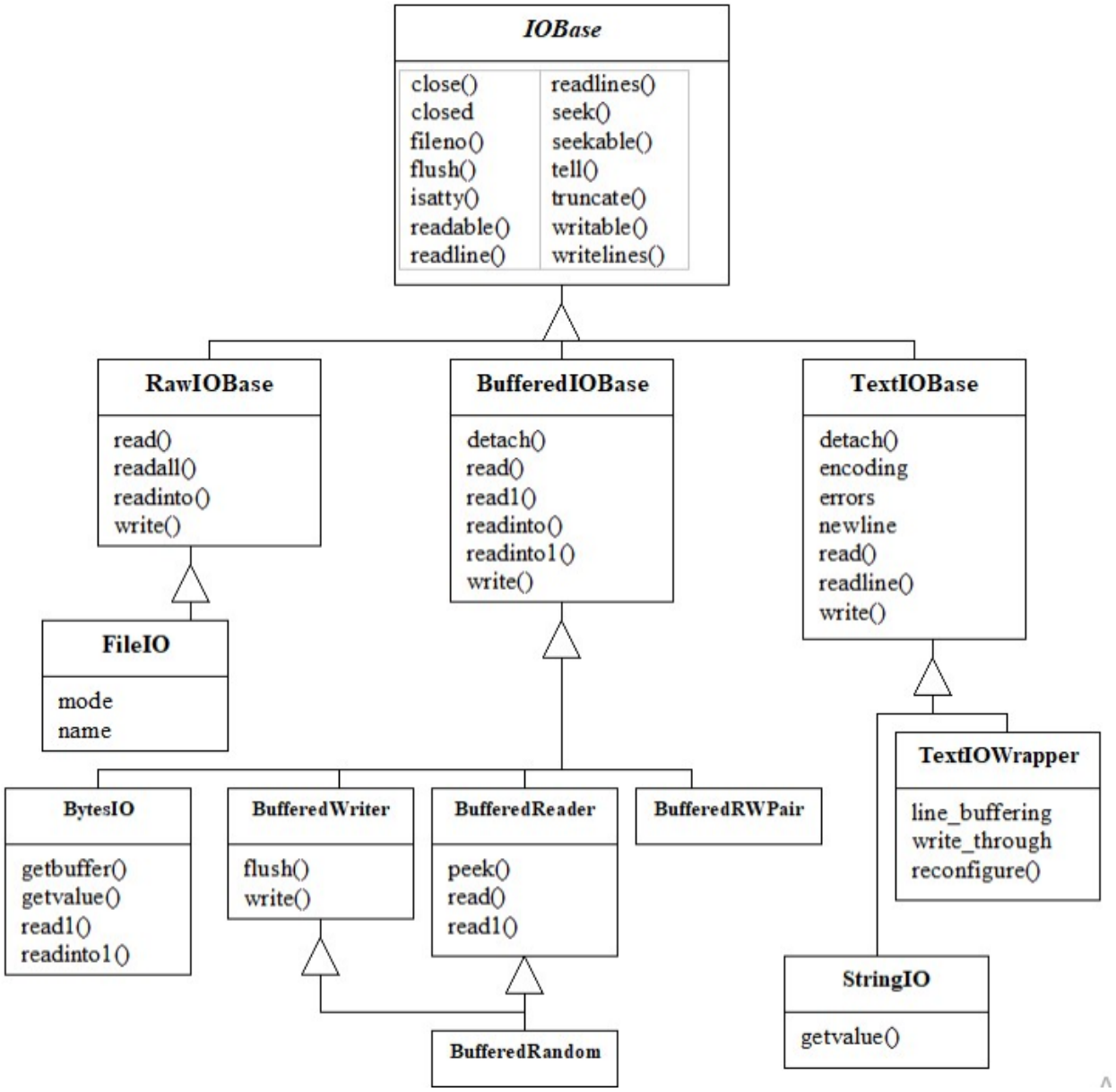

Рис. 2.6. Властивості і методи

Щоб зробити процес розробки ще простішим і швидким, в мові Python вам не потрібно займатись менеджментом пам’яті, як в прикладі з C++, вона сама все за вас зробить. Про будь які проблеми з нестачею чи ще чимось можна більше не хвилюватись. В самій мові буквально кожна частина це об’єкт. Окрім цього, на даній мові можна написати «script». Не дуже складно, коли мова для програмування інтерпретована і все робиться лінійно без необхідності щось до цього компілювати. Більш нові версії мови

використовують менше пам’яті і працюють швидше. У мові є підтримка програмування об’єктно-орієнтованого, функціонального, структурованого і не тільки. Наприклад, ви можете розробити нову програму, написавши при цьому не дуже великий код. Декоратори «пітона» відкриваються доступ і до інших видів. Сильно в нагоді стають і інші фреймворки. Є підтримка одразу декількох потоків та можливість аналізувати об’єкти під час виконання коду. Такі інструменти дозволять пришвидшити розробку. Мова звичайно має свій власний інтерпретатор «CPython». Свого роду це стандарт та інструмент для підтримки великої купи популярних сервісів. Завдяки ньому віртуальна машина може вам скомпілювати стартовий текст в код з байтів.

Як не дивно, але Python має також свої аналоги або ж додатки для інших PL. Для написання веб-додатків, математичних моделювань, роботи з мережами чи для створення розважальних застосунків PY також підходить. Використання мови є і у інших сферах, в тому числі деякі вже готові роботи до яких можна інтегрувати PY. Допомагає в цьому те, що він портований на велику кількість інших платформ. Такими є Android, UNIX, macOS та інші платформи. Як і більшість мов програмування, «пітон» не полюбляє свою підтримку на старих OS. Мова використовується і для створення GUI чи роботи з масивом.

Рис. 2.7. Основні типи даних

Рис. 2.8. Сфери застосування

2.4. Алгоритми функціонування програми і її структура

Розроблений програмний продукт є ботом для месенджеру «Телеграм». В процесі розробки програми було використано систему управління базами даних SQLite, дуже сучасну і популярну мову Python, для створення мінімалістичного «скрипту» було використано JavaScript, а також передовий фреймворк по створенню ботів Aiogram.

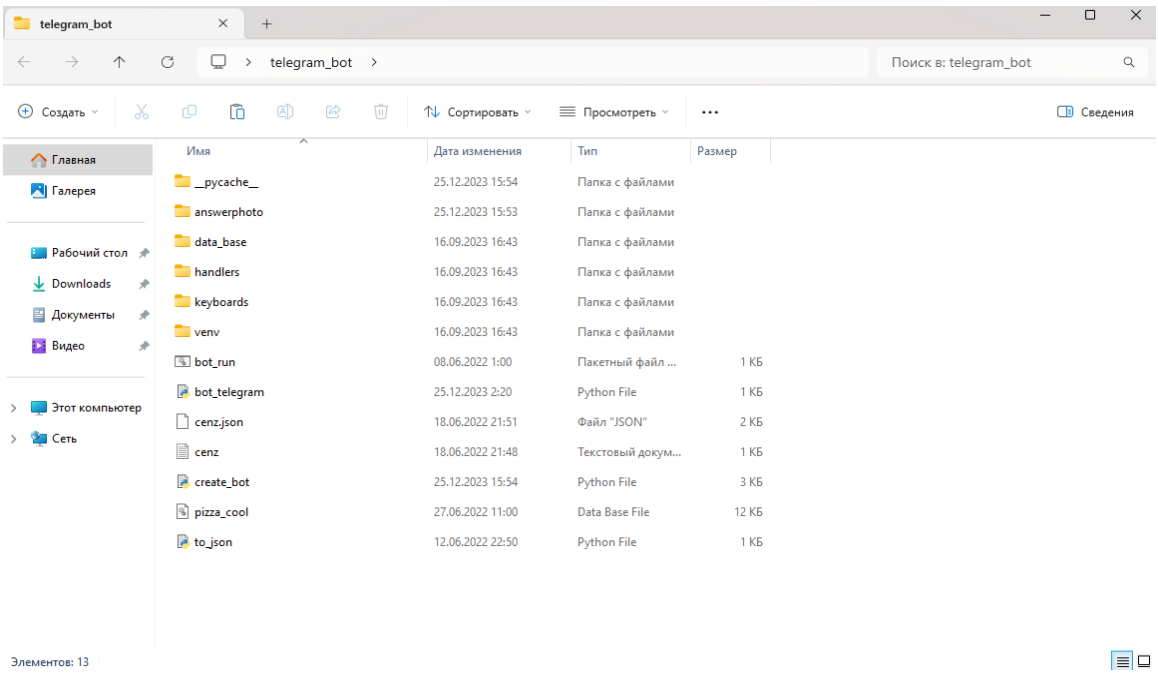

Рис. 2.9. Папка з ботом

Для більш зручної розробки, сегрегації файлів та належного підходу було прийняте рішення помістити весь проект в один файл. Для зрозумілого, зручного і швидкого пошуку елементів, а також можливості мобільно змінювати код, його було роздроблено по частинах та підписано. Для окремих частин коду є свої папки та коментарі. Усі файли системи пов’язані між собою да мають змогу взаємодіяти завдяки тому, що були об’явлені, отримали своє ім’я або були імпортовані чи викликані з одного файлу в інший.

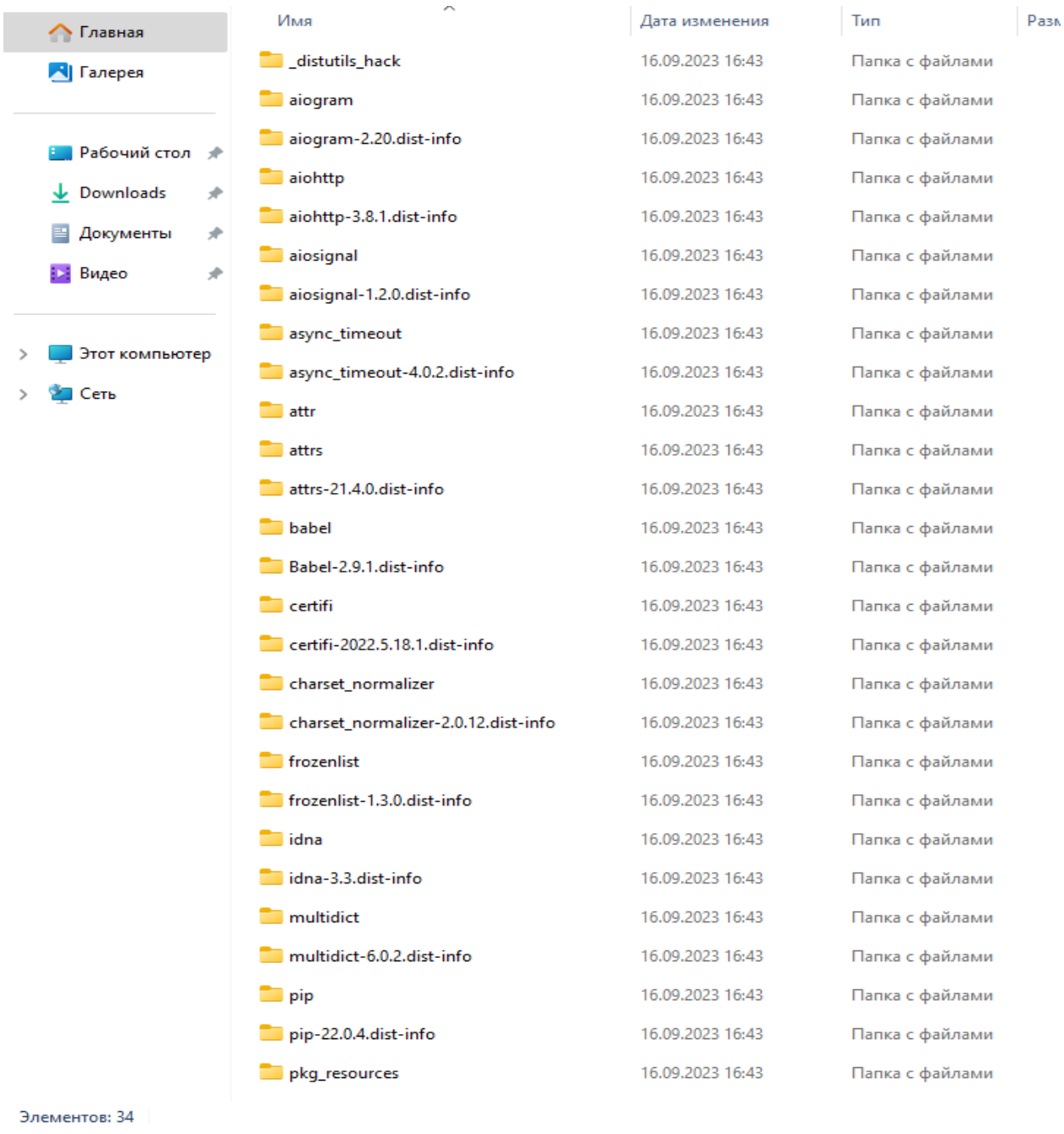

Рис. 2.10. Пакети файлів в боті

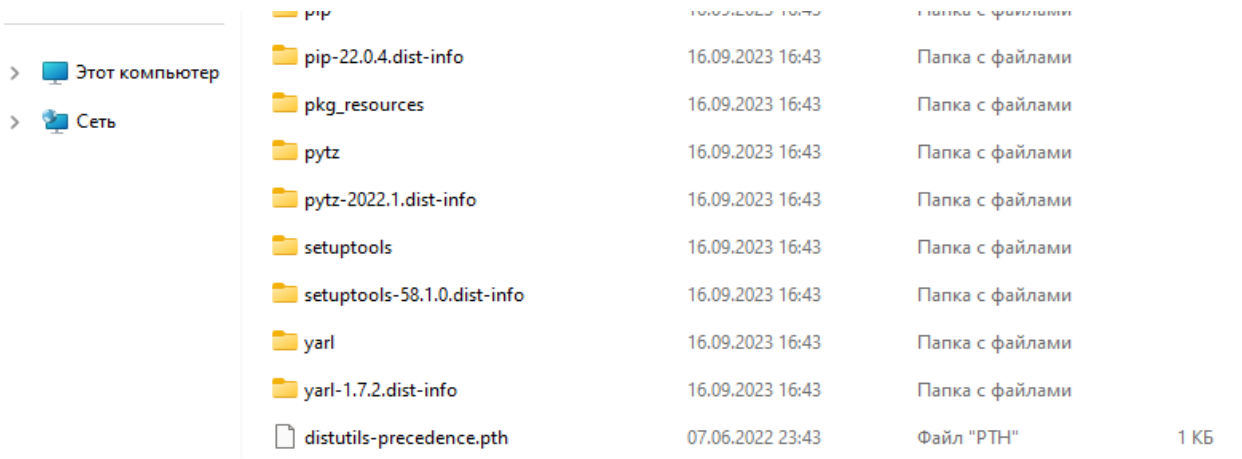

Рис. 2.11. Менеджер pip

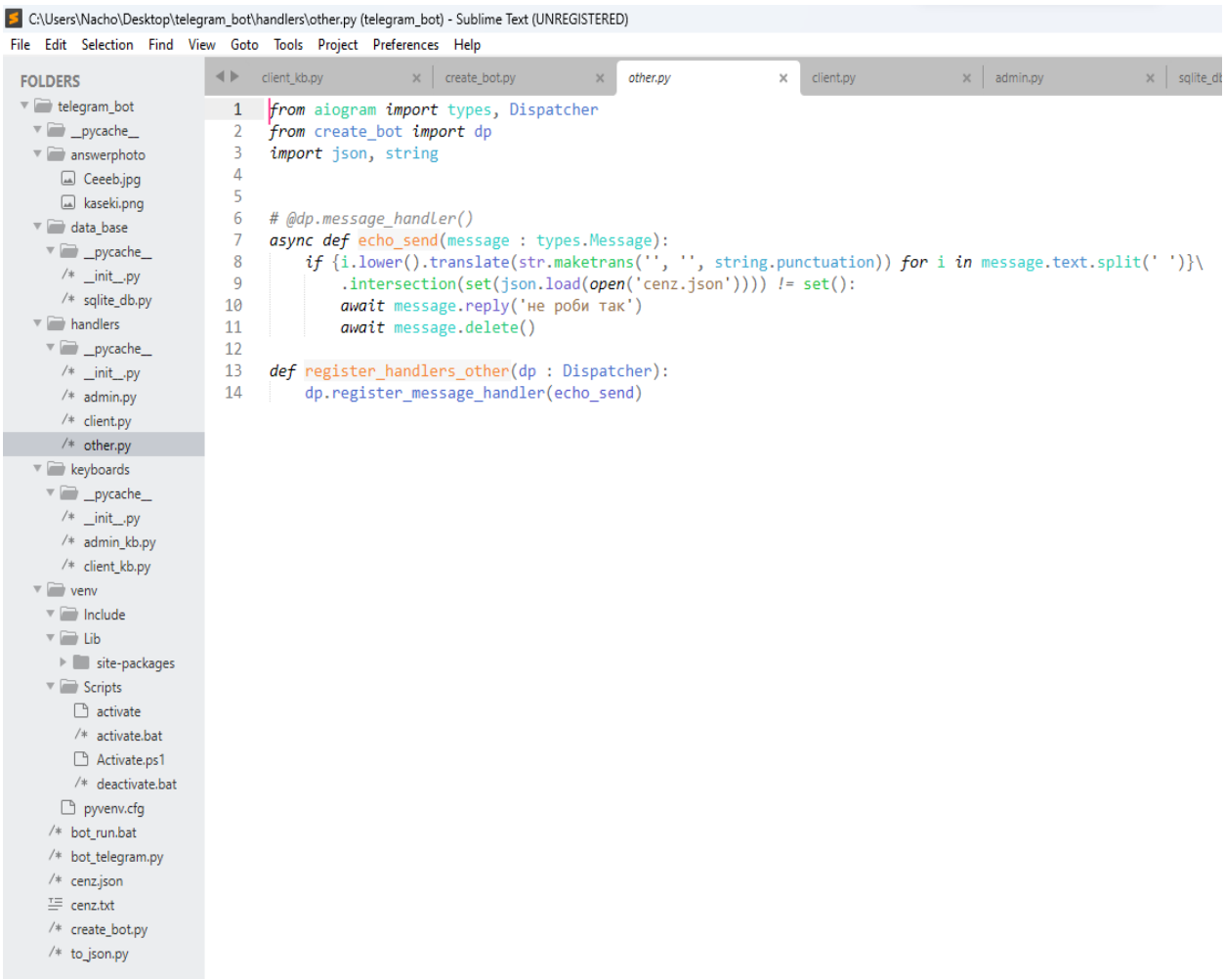

Рис. 2.12. Програмний вигляд (код)

Аби була змога використовувати меню товарів і сервісу створеного бота, до нього була інтегрована база даних SQLite. Файл з базою даних знаходиться також в основній папці з проектом. Програма розроблена таким чином, що DB

буде створено автоматично. Для такого випадку була написана за допомогою JavaScript команда створення файлу з даними. Відповідно, у разі зникнення цього файлу (видалення навмисно чи через необережність), його буде відновлено. Як вже було зазначено, база з даними використовується компанією для зберігання інформації своєї продукції та редагування товару.

Для використання бази даних необхідно ввести необхідні «імпорти». Окрім цього, треба створити та налаштувати кінцевий автомат FSM (Finite state machine).

```
ID = None

class FSMAdmin(StatesGroup):
    photo = State()
    name = State()
    description = State()
    price = State()

#Current moderator id receiving
# @dp.message_handler(commands=['moder'], is_chat_admin=True)
async def make_changes_command(message: types.Message):
    global ID
    ID = message.from_user.id
    await bot.send_message(message.from_user.id, 'Що треба', reply_markup=admin_kb.button_case_admin)
    await message.delete()

# Start new menu chapter downloading
# @dp.message_handler(commands='Download', state=None)
async def cm_start(message : types.Message):
    if message.from_user.id == ID:
        await FSMAdmin.photo.set()
        await message.reply('загрузи фото')

# Status Exit
# @dp.message_handler(state="*", commands='revoke')
# @dp.message_handler(Text(equals='revoke', ignore_case=True), state="*")
async def cancel_handler(message: types.Message, state: FSMContext):
    if message.from_user.id == ID:
        current_state = await state.get_state()
        if current_state is None:
            return
        await state.finish()
        await message.reply('ok')

# 1st Answer Catch up
# @dp.message_handler(content_types=['photo'], state=FSMAdmin.photo)
async def load_photo(message: types.Message, state: FSMContext):
```

Рис. 2.13. Налаштування FSM

Для користування ботом, його базою даних та клавіатурою треба зробити «import» ключових класів, декоратора, під’єднати папки, файли тощо. Після створення DB та завершення усіх налаштувань, необхідно наповнити отримане сховище даних тими товарами, які є в асортименті у компанії. Ключ, який було отримано від «Телеграм», необхідно вбудувати в код, замаскувати

його з питань безпеки, дати йому ім’я. Останнє є необхідним фактором для використання його та цього ім’я в іншій частині програми. Для розробки бота застосовані функції асинхронності, класи, інші види функцій та опцій від фреймворку та PY.

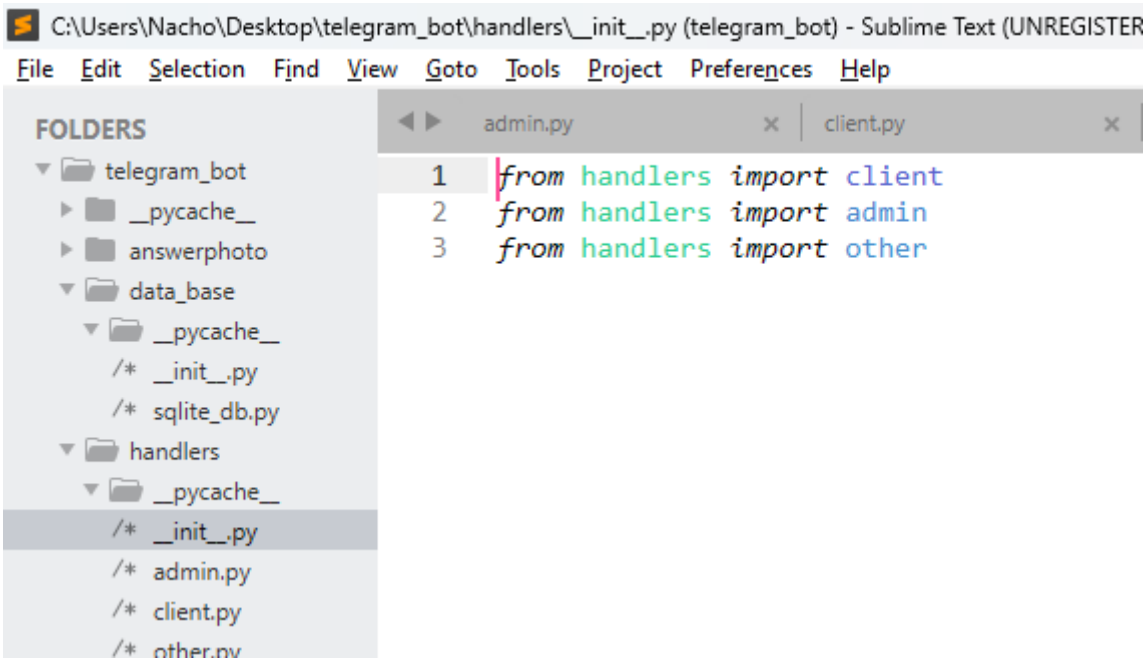

Рис. 2.14. Налаштування та виклики

Рисунок 2.14, який наведено вище, демонструє виклик з іншої частини коду компонентів, що призначені для роботи з клієнтом, виконання опцій менеджерів та адміністраторів, а також усіх інших властивостей та функцій, якими володіє бот.

Рисунок 2.15, що розташований нижче, показує як виконувались виклики і додавання ключових класів, декоратора та компоновки всієї програми.

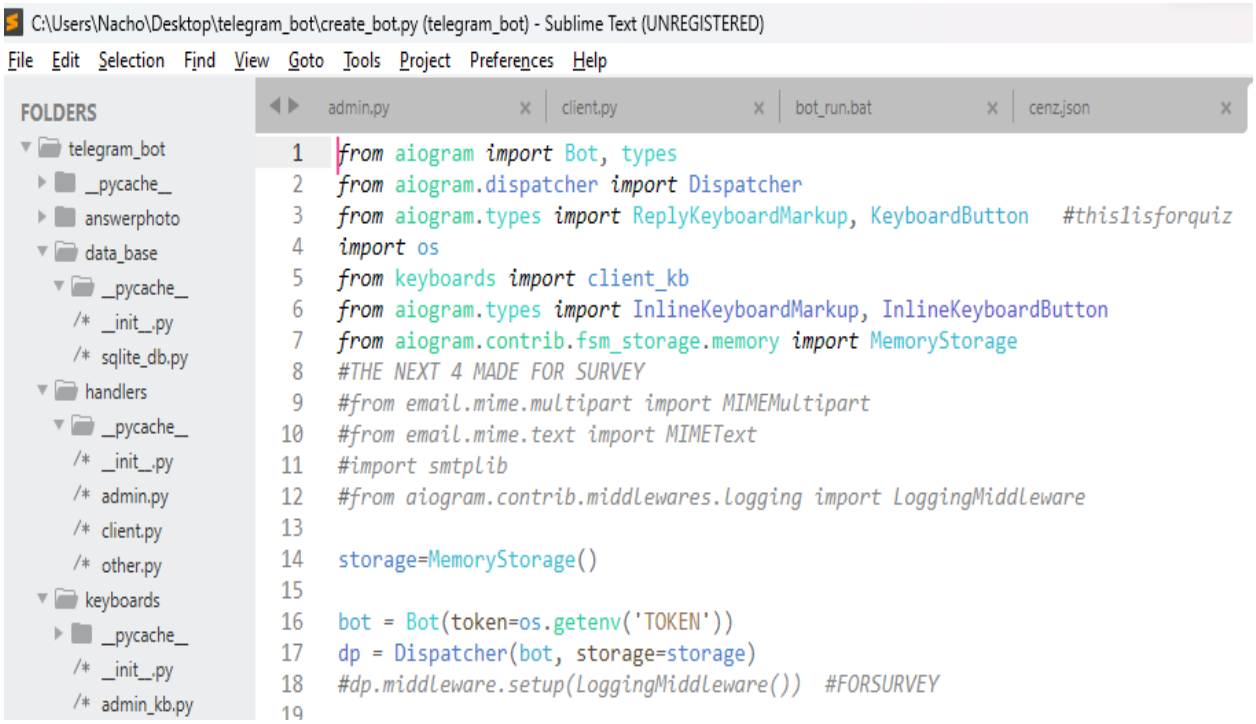

Рис. 2.15. Виклик декоратора

Для старту роботи програми необхідним елементом є файл запуску та скрипт запуску. Ця особливість була реалізована за допомогою файлу з розширення .bat. В цьому файлі вказаний шлях до функції активації бота, вона знаходиться в папці з усіма файлами. В процесі виклику використовується назва папки та скрипта на мові PY. Після цього вводиться токен, що вже був раніше отриманий. За допомогою двох команд «echo» та «pause» були реалізовані можливості побачити заданий текст в консолі, який має свідчити про активацію та пропуск усіх зайвих повідомлень, та блокування функції відлуння.

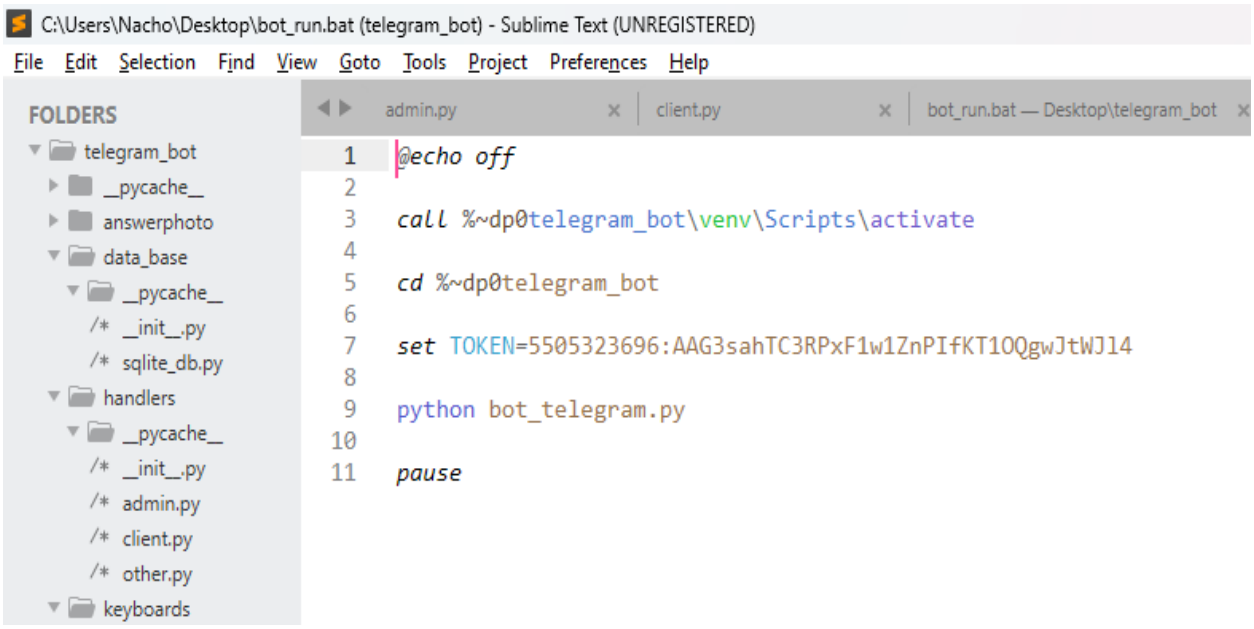

Рис. 2.16. Запуск бота

Для того аби бот став активований та не реагував на повідомлення, які були надіслані в момент простою боту, використовується команда початку режиму «пулінгу», де в якості параметрів вказується створений екземпляр боту, команда пропуску «апдейтів» та ввімкнення робочого режиму програми. Детально це показано на рисунку нижче.

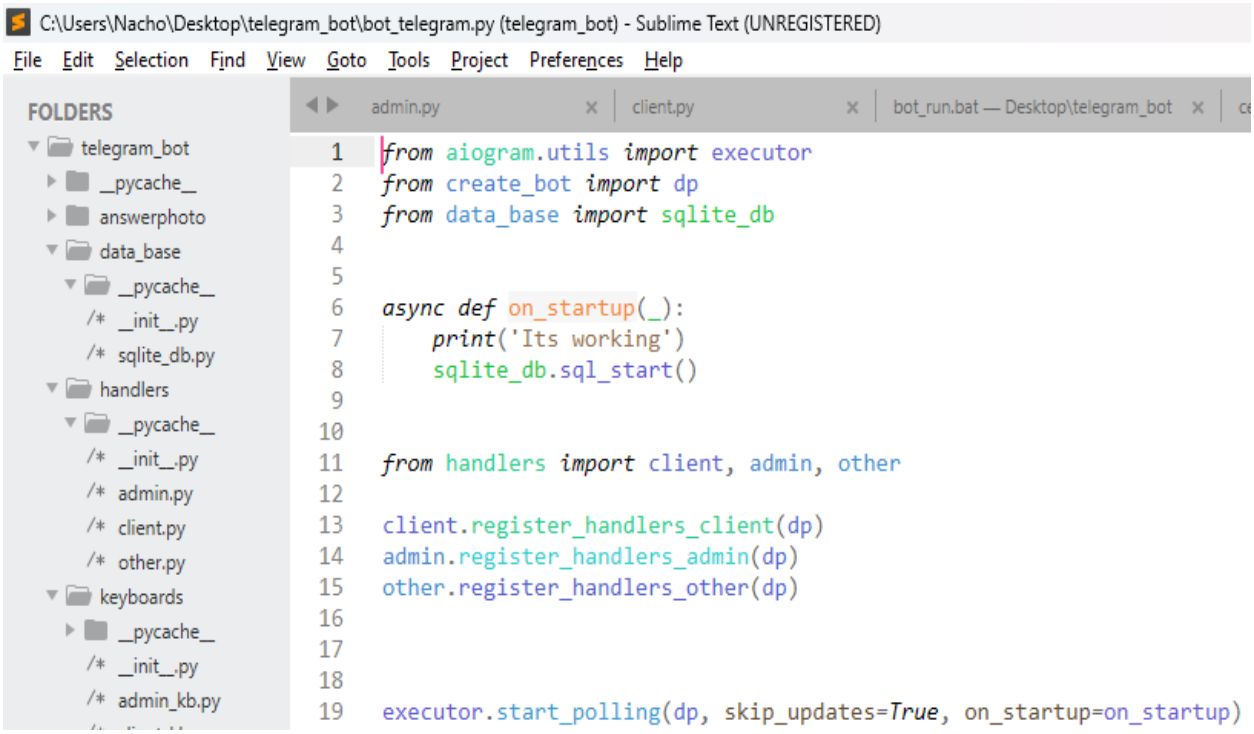

Рис. 2.17. Пропуск «апдейтів»

Такі звичні речі як адрес бота, робота з його API у «Телеграм», отримання секретного ключу для запуску, використання та налаштування різного роду транзакцій та інших речей, все це проходить на самому сайті.

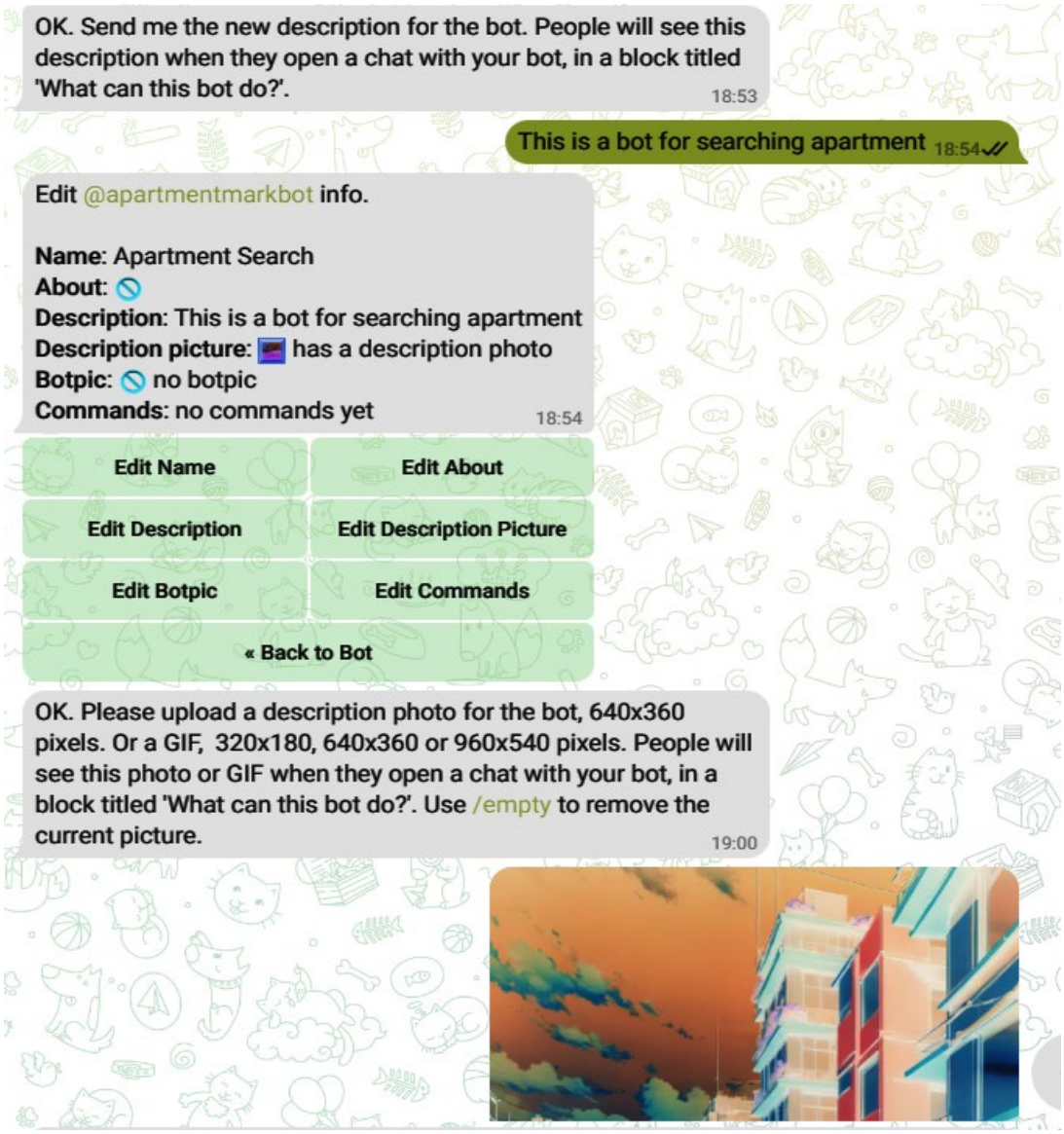

Рис. 2.18. Детальне налаштування в «Телеграм»

Рисунок 2.18 зверху відображає можливості корегувати бота на власний розсуд та побажання.

Для запуску бота і його подальшої роботи зазвичай використовують метод «Webhook». При виконанні завдання та для демонстрації роботи бота використовується менш популярний метод, який більше направлений саме на тестування та економію грошей, аби зайвий раз не переплачувати за «хостинг»

свого застосунку на віддаленому сервері. Цей метод зветься «Polling». За необхідності, є можливість використовувати «Webhook», виконавши мінімальну кількість дій.

Webhook використовується для надання можливості веб-застосунку автоматично отримати повідомлення від сервера у реальному часі і не потребує другорядних запитів. По суті це «callback» для ініціалізації сервером у випадку настання певних змін та подій. В порівнянні з іншим методом, цей зменшує навантаження на сервер за рахунок того, що відправляє запити тільки у разі актуальних подій. Технологія використовується для повідомлень, моніторингу будь-яких змін, у веб-розробці, в чат-ботах і інших випадках при потребі отримувати актуальну інформацію. Працює все це наступним чином: програміст має зареєструвати URL-адресу, куди будуть йти повідомлення, оновлення, при виконанні подій на сервері, він починає ініціацію запиту до адреси, відправляючи «payload» (інакше кажучи інформацію) про те, що трапилось. Застосунок, який отримав, опрацьовує їх та робить відповідну реакцію. Технологія має попит серед месенджерів типу Facebook і Telegram. Використовується в блогах, автоматизації реакцій на події, для керування контентом, електронних комерційних питаннях, покупках, реєстрації на сайті тощо. Webhook виконує свою роботу моментально, зменшує навантаження, прискорює обробку даних та є більш ефективним.

В свою чергу метод «Polling» взаємодіє з сервером та клієнтом. Клієнт час від часу робить запити до сервера для отримання актуальних даних та активно для цього використовується. Також дозволяє отримувати оновлення від сервера, пости, повідомлення та інше. Дозволяє підтримувати зв’язок з допомогою запитів. Клієнт періодично відправляє запити на сервер. Ці запити обробляються сервером та надається відповідь в залежності від присутності чи відсутності змін. Polling має види «short» і «long». В одному з них робляться запити в певні проміжки у часі, а в іншому підтримується зв’язок відкритого типу. Бува дуже популярним раніше, зараз використовується в застосунках для чату, здатний вести моніторинг повідомлень, а також в іграх, наприклад,

оновлення ситуації на ігровій мапі і отримання даних про інших гравців. Це надійний та легкий метод взаємодії сервера з клієнтом, але менш швидкий та витрачає зайві ресурси.

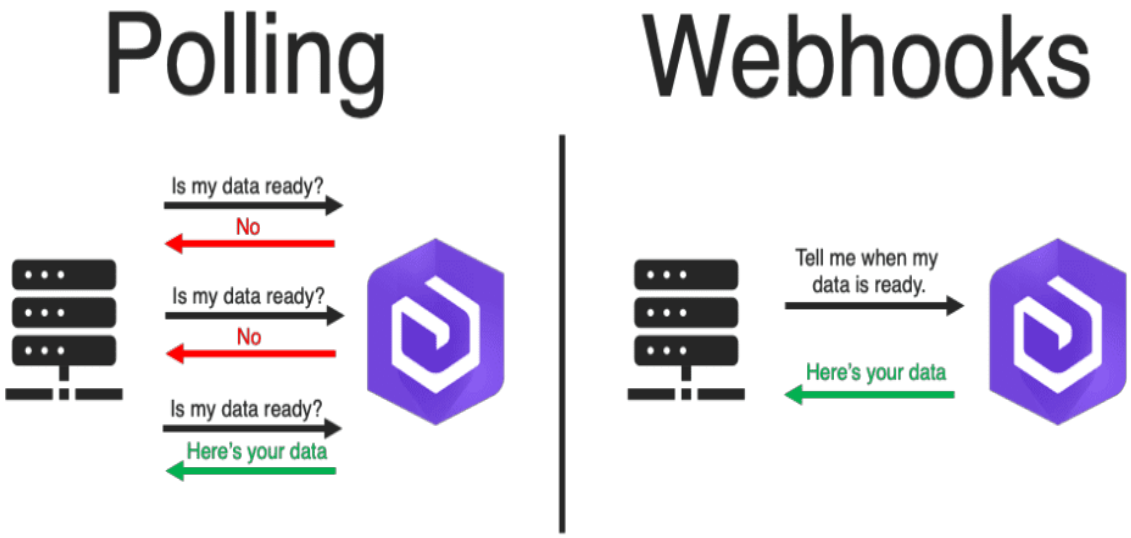

Рис. 2.19. Методи «пулінгу» та «вебхук»

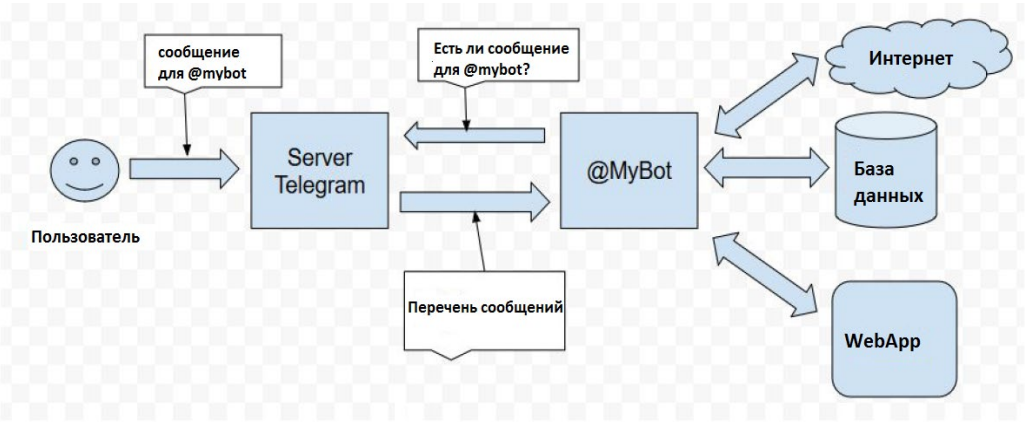

Рис. 2.20. Спрощена версія роботи API платформи

2.5. Опис створеної програми та принципі її роботи

Створена програма є ботом для платформи «Телеграм». Її призначенням є поліпшення роботи персоналу в компанії, автоматизація частини процесів. Одними з завдань є надання послуг купівлі товару, допомоги з вирішення питань, перегляд асортименту компанії і послуг. Паралельним завданням є збір інформації для проведення дослідження ефективності роботи. Також бот

має відповідати на питання клієнтів, реагувати на ключові слова. В якості другорядного завдання, застосунок має нести розважальний характер, відповідаючи популярними фразами та надсилаючи зображення. В такий спосіб можна привернути більшу увагу до бота, налаштувати клієнтів на використання сервісу, потенційно привабити нових користувачів, які можуть стати майбутніми клієнтами.

Програма була створена на персональному комп'ютері з наступним переліком використаних технічних засобів: SSD диск Kingstone A400 з об'ємом пам'яті 480 ГБ, оперативна пам'ять Fury четвертого покоління з частотою 3200 ГЦ та об'ємом 16 ГБ, блок живлення RZTK 600W, материнська плата ASUS TUF, процесор Intel Core I9-13900, та монітор HPX24C.

Сам бот був написаний на мові високого рівня Python. Для стабільної та швидкої роботи програми був застосований «топовий» фреймворк «Aiogram». При створенні бази даних для бота було обрано SQLite, це пов'язано з тим, що дана DB вбудована в мову програмування, є дуже просто і зручною у використанні та позбавляє необхідності розробляти великі обсяги коду. Також всі дані зберігаються в папці з файлами, що дозволило не використовувати віддалений сервер зі збереженими даними у DB. На момент створення бота було використано останню версію Windows 11 з «білдом» 2H22. Окрім цього, використаний звичайний редактор коду Text, текстовий редактор Word, остання версія браузеру Edge. Окрім згаданих програмних засобів, було використано стандартні «утиліти» які вбудовані та є в Windows. Використано програмний застосунок для робочого столу «Телеграм», скачаний з офіційного сайту компанії. Для створення окремих скриптів використовувався JavaScript. Для зв'язку з сервером та організації роботи бота Telegram була використана технологія «polling».

Коли бот активований і знаходиться в мережі до нього може звернутись будь-яка людина за допомогою додатку для робочого столу від Telegram або на їх сайті. Для цього буде достатньо увійти в профіль та у пошуку ввести ім'я вказаного бота. Потрібно обрати його зі списку та натиснути або ввести /start.

З ціллю виклику програми був написаний спеціальний скрипт та створений файл запуску. Файл має розширення типу bat. Це стандартне розширення для файлів запуску в операційній системі Windows. Такий спосіб запуску бота вже було продемонстровано на рисунку 2.16.

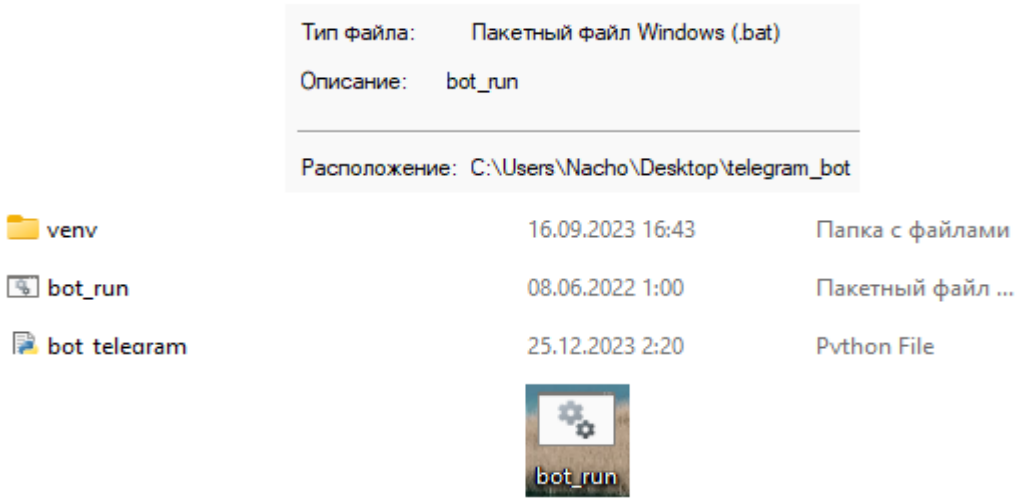

Рис. 2.21. Файл запуску бота

Нові користувачі є необізнаними об’єктами для бота, тому для першого використання проведено налаштування опису привітального повідомлення від бота та його пропозиція перейти до нього та написати йому. Для цього написано окрему частину коду, що викликає це повідомлення, при виконанні необхідних дій, бот зможе отримати унікальний код користувача.

```
# Налаштування кнопки та команди старт для початку користування ботом і спілкування зним
async def command_start(message : types.Message):
    try:
        await bot.send_message(message.from_user.id,'Добрий день, смачного', reply_markup=kb_client)
        await message.delete()
    except:
        await message.reply('Пиши йому -> напиши команду допомоги /help :\n https://t.me/DogmatixBot')

# НАЛАШТУВАННЯ КОМАНДИ ДОПОМОГИ
async def command_help(message : types.Message):
    await bot.send_message(message.from_user.id, 'бот має команди: \nhelp, \nновини, \nРозклад, \nLocati
    await message.delete()
```

Рис. 2.22. Перше повідомлення бота

Унікальний код може знадобитися для як для тривіального спілкування з клієнтами, так і для надання альтернативних повідомлень. Останнє наступає у разі, якщо щось іде не так, або бот вас не знає. В іншому випадку можна вести бесіду с ботом та спокійно користуватись його функціоналом.

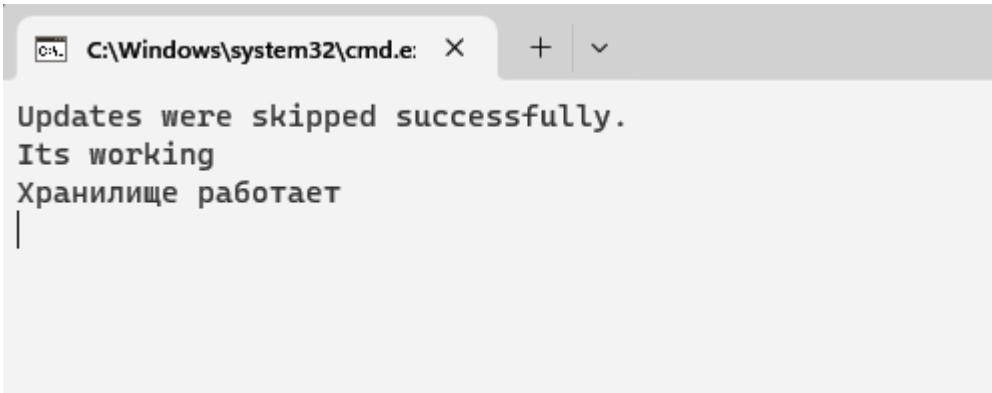

Рис. 2.23. Старт боту і пропуск оновлень

Рисунок 2.23 демонструє повідомлення у консолі. Вона дає зрозуміти, що усі «оновлення» були пропущені. Також можна побачити напис про активізацію боту та вихід його у мережу. Окрім цього, бачимо, що база даних бота також стає активною.

Привітання від бота та його опис виглядають так, як це показано на рисунку 2.24.

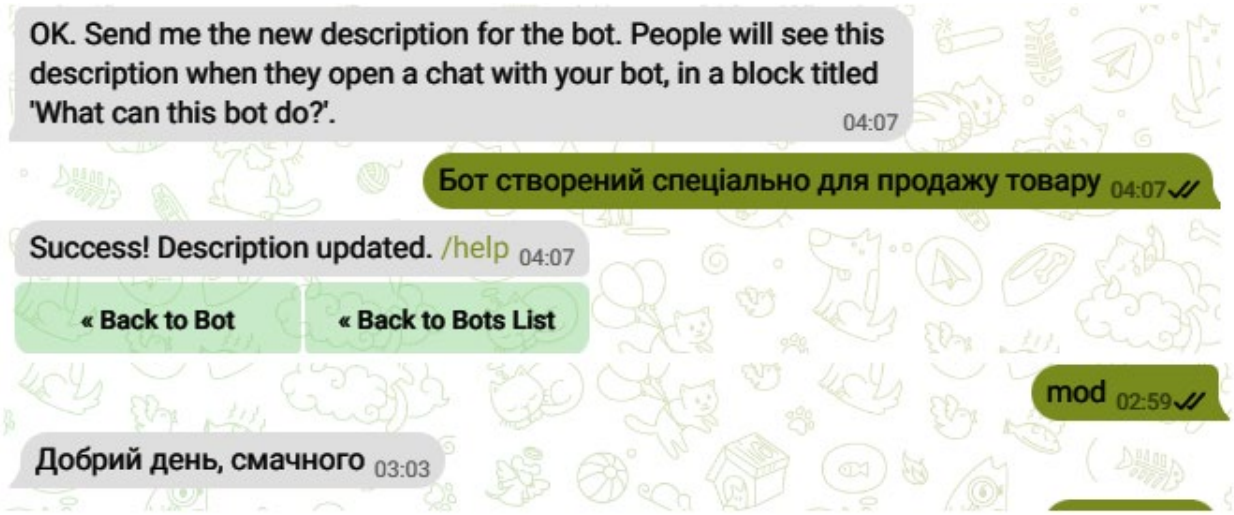

Рис. 2.24. Опис бота та початок розмови

Наступний рисунок 2.25 виводить нам налаштування фотографії над описом для бота.

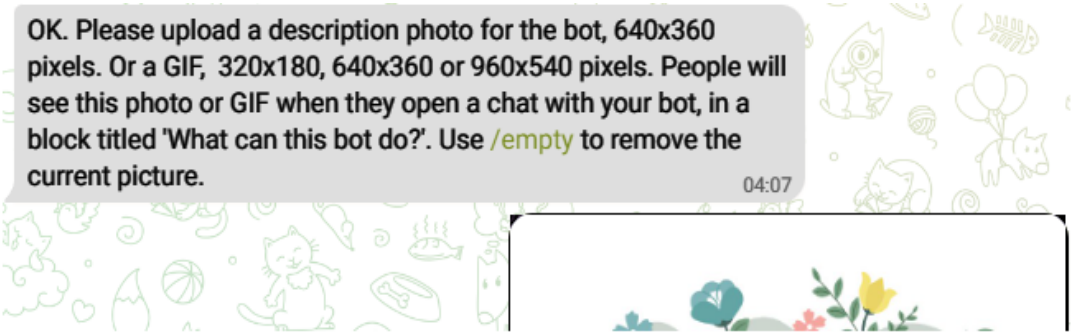

Рис. 2.25. Завантаження та зміна фото

Привітання після опису в боті спрацьовує, якщо виконується команда «старт». Бот відправляє спланований месендж до клієнта в чат. Це було продемонстровано на рисунку 2.24.

Для покращення опиту використання клієнтом бота, йому був наданий список з команд, окрім цього, бот сам пропонує ввести допоміжну команду. Коли виконується команда «help», бот надає список доступних йому команд, але не всіх, команди для менеджера, адміністратора і розробника ніхто не бачить.

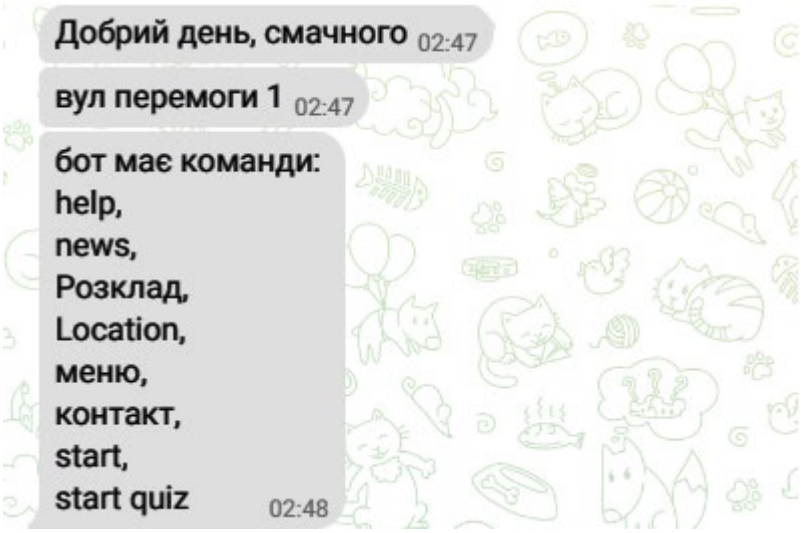

Рис. 2.26. Доступні публічні команди

В якості прикладу роботи команд з клавіатури бота, використаємо самі популярні, отримаємо відповідь від бота. Результат можна бачити на рисунку 2.27. Також наведено збір інформації ботом (телефонні дані).

Рис. 2.27. Популярні команди бота

Пошук необхідної інформації реалізований за допомогою клавіатури бота та реакції на ключові слова.

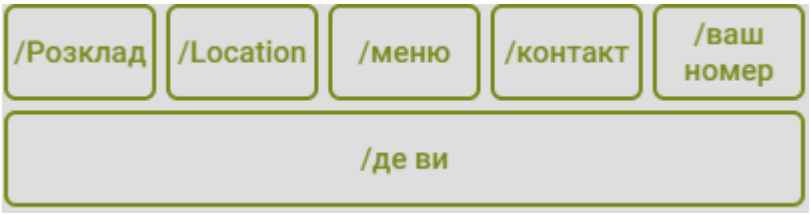

Рис. 2.28. Клавіатура команд

В якості прикладу реакції на певні слова та вирази, був зроблений рисунок 2.29. Суть використання полягає в тому, аби бот міг зрозуміти побажання клієнта, надавав моментальні відповіді по темі та блокував слова, що не пройшли цензуру.

Рис. 2.29. Робота цензури

Ще однією командою для збору інформації є команда «локації», яка може надати дані про місцезнаходження клієнта з його згоди. Отримана інформація піде у використання в процесі дослідження. Як вона працює видно на рисунку 2.30.

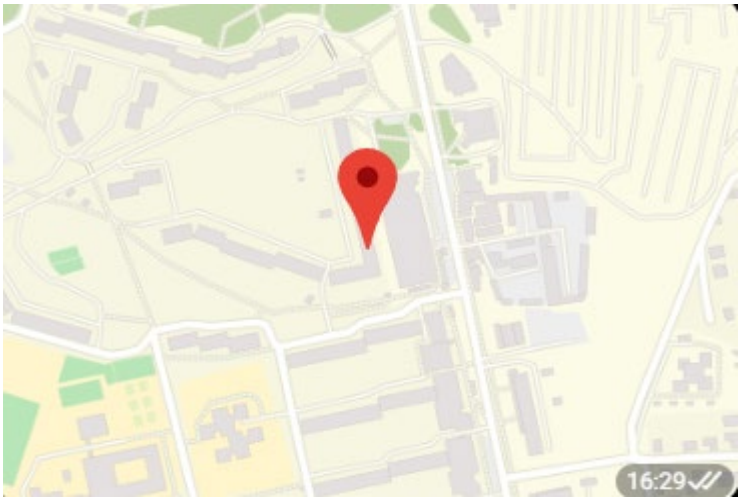

Рис. 2.30. Визначення локації

Реалізація каталогу товарів бота наведена на рисунках 2.31 та 2.32. Серед пунктів бачимо напої та харчову продукцію.

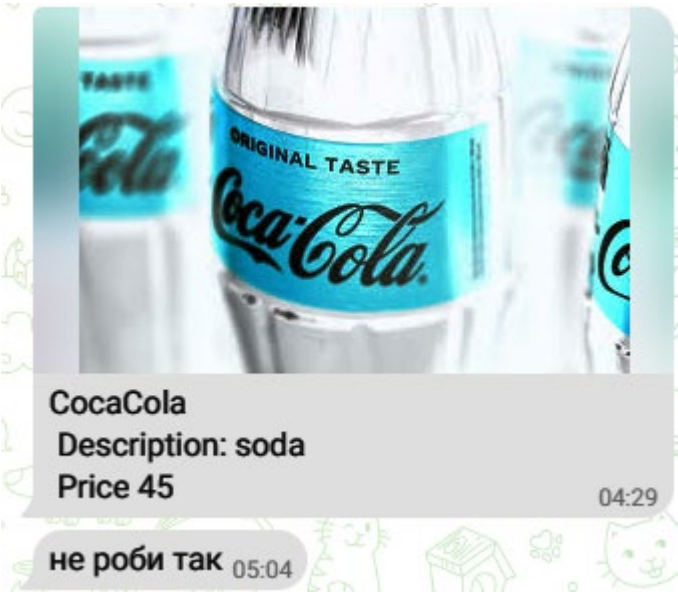

Рис. 2.31. Кола в меню

Рис. 2.32. Їжа у меню

На випадок, якщо ботом користується хтось без досвіду роботи з ботами, та для випадку, коли необхідно швидко увімкнути клавіатуру бота, була створена окрема допоміжна команда. Для її запуску достатньо ввести простий символ скісної риски. Сама команда не відобразиться в чаті, але її наслідки будуть видними у новому повідомленні.

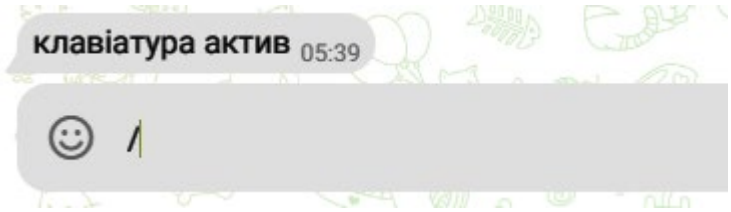

Рис. 2.33. Швидка активація

В процесі виконання роботи вже згадувалась інтерактивна кнопка з посиланням. Вона застосовується для збору інформації там веде на навчальне відео користування ботом.

Як зроблено клавіатуру для менеджера товарів бота та її робочий варіант, можна знайти на рисунку 2.35. Вхід в режим для корекції асортименту товарів виконувався окремою командою «mod», яка потім була замінена на іншу в цілях безпеки доступу до меню.

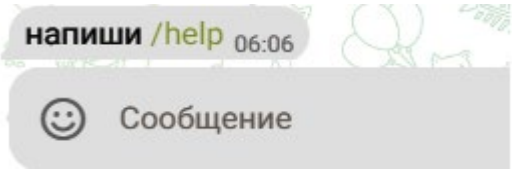

Рис. 2.34. Бот відреагував на ключове слово

Рисунок 2.34 показує, що бот здатний помічати певний контекст та слова в реченнях. Така особливість потрібна для більш швидкого пошуку у боті.

Рис. 2.35. Клавіатура для менеджера

При використанні клавіатури менеджера можна налаштувати каталог з товарами, додати новий товар, видалити старий.

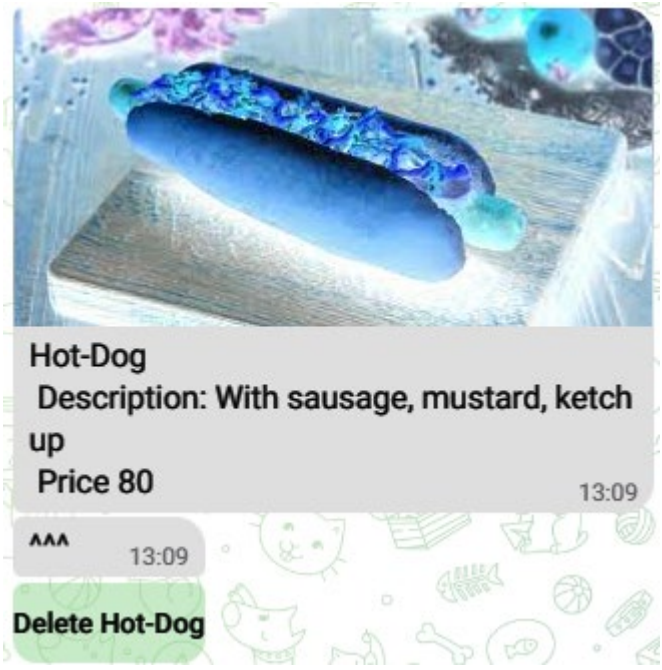

Рис. 2.36. Спосіб видалення

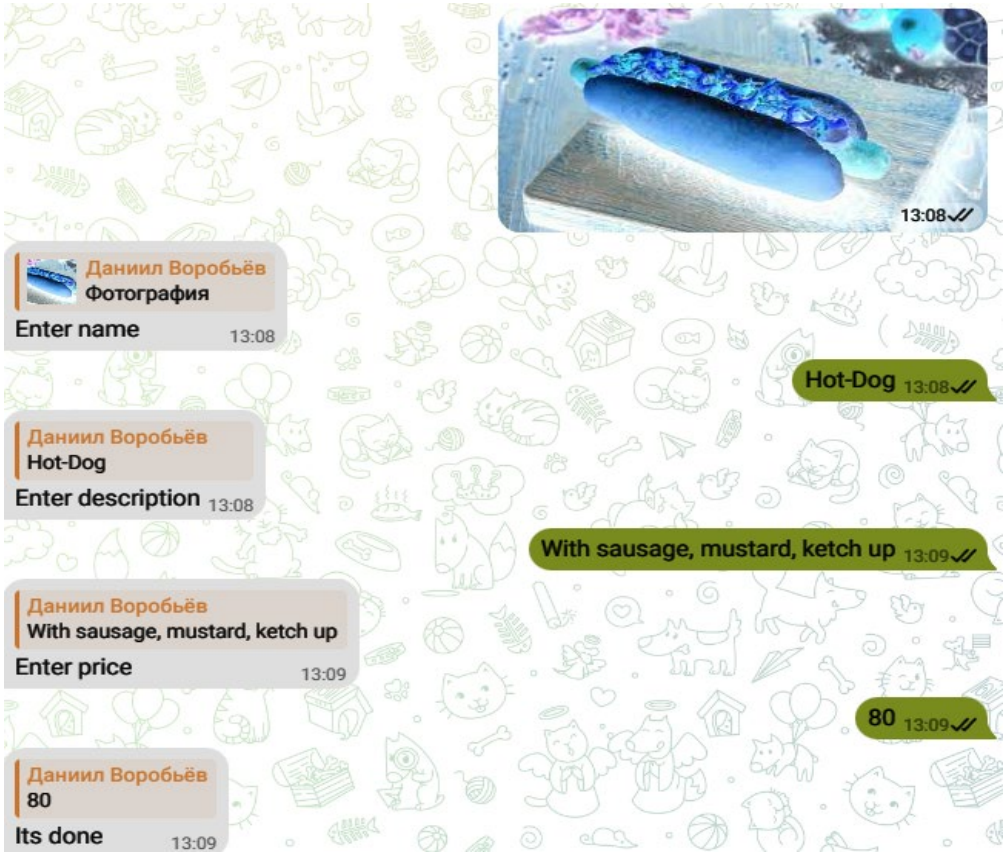

Рис. 2.37. Додавання нового товару

Рисунок 2.37 відображає як можна додати товар до каталогу. Для цього використовується раніше створена машина FSM. При створені нового товару в каталозі необхідно вказати необхідні дані для бота та надати йому фотографію продукції. Потрібно зафіксувати ім’я товару, його вартість, коментар до товару, завантажити фото.

Трапляються ситуації, коли інформація викладена невірно або необхідно відхилити новий товар. Для цього створено команду, що виходить з режиму FSM. Її приклад знаходиться на рисунку 2.38.

Аби мати змогу фільтрувати чат та не допускати використання нецензурованих слів, був розроблений метод блокування таких слів, якщо вони є у «забороненому» списку слів. Лист зі словами створюється автоматично в результаті виконання написаного скрипту на JavaScript. Він аналізує записані слова та перетворює їх в інший формат.

```python
import json

ar = []

with open('cenz.txt', encoding='utf-8') as r:
    for i in r:
        n = i.lower().split('\n')[0]
        if n != '':
            ar.append(n)

with open('cenz.json', 'w', encoding='utf-8') as e:
    json.dump(ar, e)
```

```
"\u043f\u0438\u0434\u043e\u0440", "\u043f\u0438\u0434\u0430\u0440"
```

Рис. 2.38. Перетворення слів і створення «забороненого» листа

Рис. 2.39. Інтерактивна кнопка з посиланням на відео

Концепт роботи FSM зображено на рисунку 2.40.

```python
async def cancel_handler(message: types.Message, state: FSMContext):
    if message.from_user.id == ID:
        current_state = await state.get_state()
        if current_state is None:
            return
        await state.finish()
        await message.reply('ok')

# додавання фотографії для товару

async def load_photo(message: types.Message, state: FSMContext):
    if message.from_user.id == ID:
        async with state.proxy() as data:
            data['photo'] = message.photo[0].file_id
        await FSMAdmin.next()
        await message.reply("название")

# Заповнення другої відповіді

async def load_name(message: types.Message, state: FSMContext):
    if message.from_user.id == ID:
        async with state.proxy() as data:
            data['name'] = message.text
        await FSMAdmin.next()
        await message.reply("описание")

# Заповнення форми з ціною продукту

async def load_description(message: types.Message, state: FSMContext):
    if message.from_user.id == ID:
```

Рис. 2.40. Машина FSM

2.6. Висновки

В ході розробки програмного застосунку, а конкретно бота для сайту Telegram, була отримана працююча модель, яка виконує поставлені задачі. Для спілкування клієнтів з ботом його навчили реагувати на слова і контекст, користувачам були надані допоміжні команди та клавіатура бота. Отримана версія програмного продукту може збирати необхідну інформацію для досліджень. При розробці були додані усі заплановані функції та призведені до робочого стану.

РОЗДІЛ 3
ДОСЛІДНИЦЬКА ЧАСТИНА

3.1. Методи досліджень

В ході дослідження ефективності використання бота для платформи Telegram використані кілька методів, кожен з яких буде описаний в цьому розділі.

Обробка природної мови NLP або Natural Language Processing. Мова йде про розділ комп'ютерної науки та штучного інтелекту, який займається взаємодією між комп'ютерами та природними мовами людини, серед них є російська, англійська, німецька, іспанська та інші.

Ключовими аспектами NLP є:

1) Токенізація - процес поділу тексту на окремі слова або фрази (токени) для подальшої обробки;

2) Частинна розмітка (part of speech tagging) - визначення частин мови для кожного слова в реченні (іменник, дієслово, прикметник та інші);

3) Синтаксичний аналіз - розуміння структури речень та зв'язків між словами для визначення змісту та контексту;

4) Семантичний аналіз - аналіз змісту тексту розуміння значення слів, фраз та його взаємозв'язків;

5) Вилучення інформації - витяг структурованих даних з тексту, таких як іменовані сутності (імена, дати, розташування) або факти;

6) Генерація мови - створення тексту або мови природними мовами з використанням комп'ютерних алгоритмів.

Для NLP є свої застосунки:

1) Машинний переклад - переклад тексту з однієї мови іншою з використанням комп'ютерних алгоритмів;

2) Обробка текстів - аналіз та обробка великих обсягів текстових даних для отримання інформації або проведення аналітики;

3) Автоматизований аналіз тексту - пошук ключових слів, порівняльний аналіз, кластеризація тексту та тематичне моделювання;

4) Розпізнавання мовлення - перетворення мовлення в текст, що дозволяє вводити команди голосом і робити інші дії.

Також NLP має не мало інструментів та технологій для роботи. Серед таких є моделі глибокого навчання. Це нейромережі, що за допомогою методу глибинного навчання дозволяють створити моделі для вирішення питань NLP. Сприяють цьому бібліотеки NLTK або «Natural Language Toolkit», Gensim, Transformers, SpaCy тощо. Використовуються такі інструменти як IBM Watson Speech to Text , PocketSphinx чи Google Cloud Speech to Text тощо. API та послуги для NLP: Amazon Comprehend, IBM Watson NLP, Google Cloud NLP та інші хмарні послуги для обробки природної мови.

NLP є потужним інструментом, який використовується в різних галузях, таких як медицина, фінанси, освіта, інформаційні технології та інші для обробки, аналізу та розуміння природної людської мови комп'ютерами.

Метод спостереження. Метод спостереження є фундаментальним методом дослідження та отримання інформації про явища, об'єкти або процеси шляхом безпосереднього сприйняття або реєстрації їх властивостей за допомогою органів чуття, інструментів чи технічних пристроїв. Основна мета методу спостереження - отримання інформації про предмет дослідження для здобуття знань, розуміння його властивостей, характеристик, поведінки чи змін. Метод спостереження ґрунтується на прямому сприйнятті феноменів, тобто спостерігач повинен бачити, чути, відчувати чи реєструвати об'єкт чи явище самостійно. Важливо, щоб спостереження були об'єктивними та незалежними від упередженості чи суб'єктивних оцінок. Це може бути досягнуто стандартизацією процедур спостереження та використанням різних технічних засобів. Отримана інформація або дані можуть фіксуватися у вигляді нотаток, записів, фотографій, відео або інших форматів та аналізуватись для виявлення закономірностей, зв'язків чи висновків.

Існують два основних типи спостереження: пряме та «індиректне». Перше відбувається у разі, коли спостерігач самостійно фіксує явища, об'єкти чи події без втручання чи модифікації. В другому випадку спостереження проводяться з використанням технічних пристроїв, приладів або засобів, наприклад телескопів, мікроскопів, датчиків та інших. У багатьох галузях науки, таких як біологія, фізика, соціологія, психологія, метод спостереження відіграє ключову роль у збиранні даних та перевірці гіпотез. У навчальних цілях метод спостереження часто використовується для демонстрації явищ та розуміння процесів. У медицині, техніці, екології та інших галузях метод спостереження допомагає спостерігати та вивчати процеси, явища та об'єкти для прийняття рішень чи покращення процедур.

Метод спостереження є потужним інструментом для отримання інформації та знань про світ навколо нас та відіграє важливу роль у наукових дослідженнях, освіті та повсякденному житті.

Для виконання роботи, проведення дослідження та отримання нових даних використовувався метод «збору» інформації. Такий метод пропонує систематичне отримання інформації, фактів, постійне спостереження для подальшого аналізу, інтерпретації та прийняття рішень. Дані методи є широким спектром технік, що використовуються в різних областях для збору різноманітної інформації.

Такий метод має цілі категорії інших методів:

1) Опитування - збір інформації за допомогою питань, які проводяться через опитування, анкети, інтерв'ю або телефонні опитування;

2) Експерименти - контрольовані процедури, що проводяться для вивчення впливу певних факторів на об'єкт або явище;

3) Спостереження - безпосереднє спостереження та реєстрація подій, процесів або поведінки без втручання;

4) Аналіз даних (Data Analysis). Це використання існуючих даних для отримання інформації та формулювання висновків, наприклад, статистичні методи та машинне навчання;

5) Документи та історичні джерела. Використання архівних даних, літератури, документів для аналізу та отримання інформації.

Серед етапів можна виділити:

1) Планування для визначення цілей, формулювання питань та цілей дослідження, розробка методології збору даних;

2) Вибір методу для збору даних в залежності від цілей дослідження, доступних ресурсів та характеру даних;

3) Збір інформації відповідно до обраного методу, спостереження, анкетування, експерименти тощо;

4) Обробка та аналіз зібраних даних, їх структурування, аналіз, інтерпретація та формулювання висновків;

5) Інтерпретація та використання результатів, ухвалення рішень, розробка стратегій або рекомендацій на основі отриманих даних.

У науці методи збору даних використовуються для перевірки гіпотез, експериментів та отримання нових знань. Опитування, інтерв'ю та аналіз даних використовуються для вивчення переваг споживачів та громадської думки. Збір даних для дослідження захворювань, ефективності лікування та попередження захворювань. Аналіз даних для розробки алгоритмів, машинного навчання та покращення процесів.

Методи збору даних є потужним інструментом для отримання інформації, вивчення явищ і прийняття обґрунтованих рішень у багатьох сферах досліджень і практичного застосування.

Наступним є метод аналізу даних. Метод аналізу даних це процес систематичного вивчення, обробки та інтерпретації інформації з метою отримання сенсу, виявлення закономірностей та виявлення тенденцій. Він включає різноманітні підходи та інструменти, що застосовуються для роботи з даними в різних сферах. Аналіз даних починається з підготовки інформації: збору даних, їх очищення від помилок та пропущених значень, а також приведення в зручний для аналізу формат. Потім проводиться дослідницький аналіз, який дозволяє отримати уявлення про дані через візуалізацію та

виявлення патернів. Після цього йде моделювання даних, що включає використання статистичних методів чи алгоритмів машинного навчання створення моделей, які можуть прогнозувати, класифікувати чи узагальнювати інформацію. Результати аналізу інтерпретуються з метою виявлення важливих факторів, формулювання висновків та прийняття рішень.

Ці методи широко застосовують у різних областях. У бізнесі та маркетингу, аналіз даних допомагає у прогнозуванні продажів, розумінні споживчих переваг та оптимізації стратегій. У охороні здоров'я дані використовуються для аналізу захворювань, передбачення ризиків і поліпшення лікування. У фінансовій сфері методи аналізу даних застосовуються для прогнозування ринків, управління ризиками та прийняття інвестиційних рішень. Важливість аналізу даних полягає у можливості вилучення цінної інформації з великих обсягів даних, що допомагає приймати більш обґрунтовані рішення та отримувати цінні інсайти у різних сферах діяльності.

Останнім використаним методом є метод порівняння. Тобто підхід, що дозволяє аналізувати та зіставляти різні об'єкти, дані чи явища з метою виявлення їх подібності, відмінностей та особливостей. Цей метод використовується в багатьох областях, починаючи від наукових досліджень та освіти і закінчуючи бізнесом і аналізом даних. Суть методу порівняння полягає в аналізі характеристик об'єктів чи явищ з метою виділення їх основних особливостей та відмінностей. Це може включати порівняння якісних або кількісних даних, а також оцінку функціональних аспектів. Часто використовувані підходи методі порівняння включають порівняння по парам, коли об'єкти аналізуються попарно виявлення відмінностей і подібності, множинне порівняння, заснований на зіставленні безлічі об'єктів, і статистичні методи порівняння виявлення статистично значимих відмінностей.

Метод порівняння застосовується у наукових дослідженнях зіставлення різних теорій чи концепцій, освіти для аналізу різних методів навчання чи теорій, у бізнесі порівняння конкурентів чи стратегій. Він є важливим

інструментом для розуміння та виявлення закономірностей, виявлення візерунків та прийняття обґрунтованих рішень у різних сферах діяльності.

3.2. Використані програмні засоби для дослідження

Для проведення дослідження, збору інформації та її аналізу, була використана програма «Айко»(iiko).

Айко є програмним рішенням для автоматизації управління та оптимізації бізнес-процесів у ресторанній сфері. Її функціонал включає управління замовленнями, облік продуктів, персоналом, фінансовий контроль, аналітику та багато іншого. Вона забезпечує можливість прийняття замовлень, формування меню, керування столами та доставки їжі. Також веде облік товарів складі, контролює надходження і списання товарів, аналізує ефективність роботи персоналу. Програма дозволяє вести фінансовий облік, включаючи облік витрат, доходів та формування звітності. Її аналітика надає різноманітні звіти з метою оцінки ефективності роботи, аналізу популярності страв, фінансових показників тощо. Застосовується у ресторанному бізнесі для оптимізації процесів обслуговування та обліку, у «фастфуді» для керування доставкою та кухнею, у барах для обліку напоїв, у готелях для керування ресторанними службами. Вона забезпечує контроль та аналіз даних, необхідних для прийняття стратегічних рішень та підвищення якості обслуговування клієнтів.

Окрім програмного застосунку «Айко», була використана платформа для аналітики Google Analytics. Це безкоштовна веб-аналітична платформа, створена Google для аналізу та відстеження даних про відвідування сайтів та програм. Вона надає інформацію про відвідувачів, їх дії на сайті, джерела трафіку, конверсії та багато іншого. Ця платформа допомагає власникам сайтів зрозуміти, як користувачі взаємодіють з їхніми ресурсами, забезпечуючи дані про кількість відвідувань, їх походження, тривалість сесій та взаємодію з контентом. Вона також надає можливість відстежувати цілі та оцінювати

ефективність маркетингових кампаній. Аналітика дозволяє генерувати різноманітні звіти та проводити аналітику, що корисно для оптимізації веб-сайтів, покращення користувальницького досвіду, моніторингу маркетингових зусиль та розуміння цільової аудиторії. Це важливий інструмент для прийняття обґрунтованих рішень у розвитку онлайн-проектів та покращення їх ефективності.

3.3. Порівняння розробки з існуючими аналогами

Для того аби порівняти функціонал створеного програмного продукту з іншими, які вже існують на ринку, були обрані декілька ботів в Telegram.

В чат-боті «Nova Poshta», що створений компанією з відповідною назвою, надається підтримка клієнтів в режимі «онлайн». С початку за допомогою кнопки «start» починається розмова з ботом. Він відправляє початкові повідомлення і одразу кличе оператора. В розробленому додатку є можливість для реалізації такої функції та все сприяє цьому, але компанія використовує лише пошту та роботу з клієнтами «offline». В коді програми окремим коментарем буде наведена можливість використання такої системи.

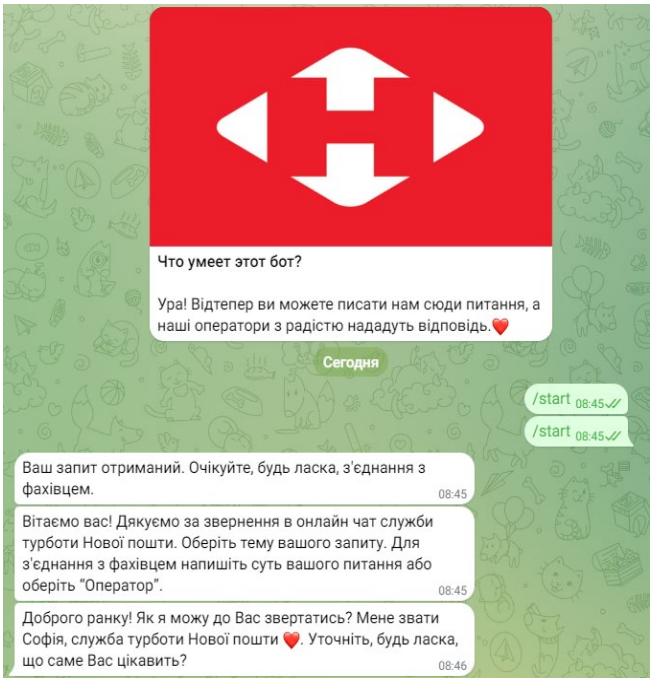

Рис. 3.1. Чат-бот від «Нова Пошта»

Також, як і в створеному боті, аналог має клавіатуру з запропонованими функціями та відповідями на запитання. Проте функції оплати він не має, тому в цьому плані створений власний бот є кращим.

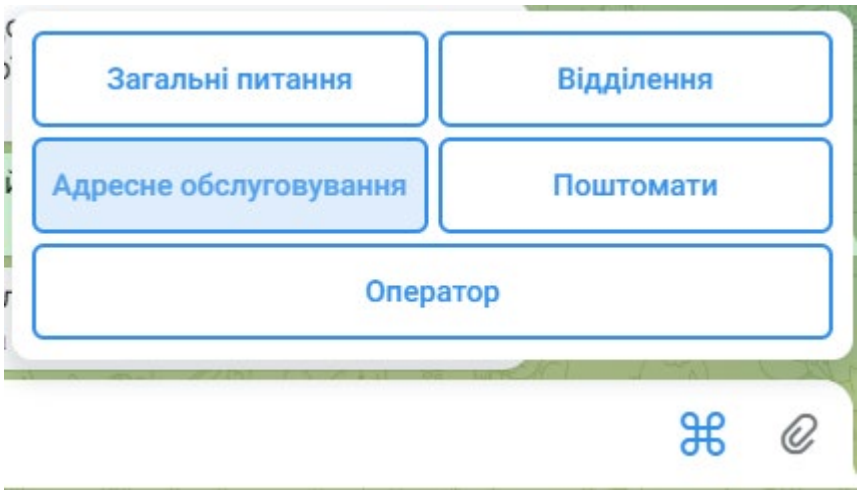

Рис. 3.2. Клавіатура бота-аналога

Наступним прикладом став бот від «Фрегат». Він також має звичне для ботів привітання та одразу пропонує свої послуги.

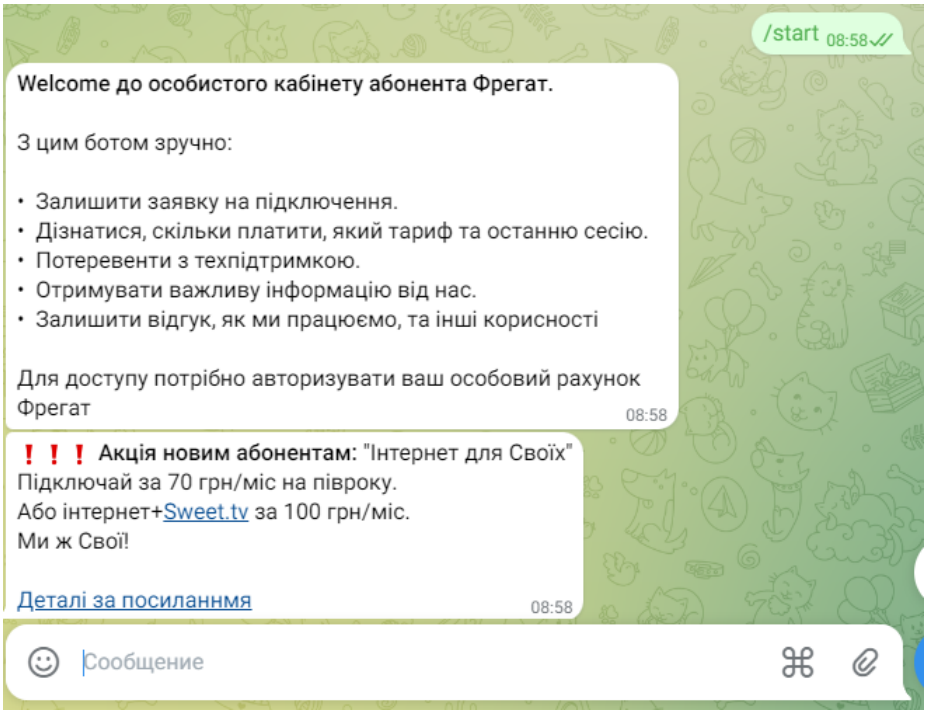

Рис. 3.3. Бот «Фрегат»

За допомогою бота пропонують послуги, а також він розповідає свої особливості.

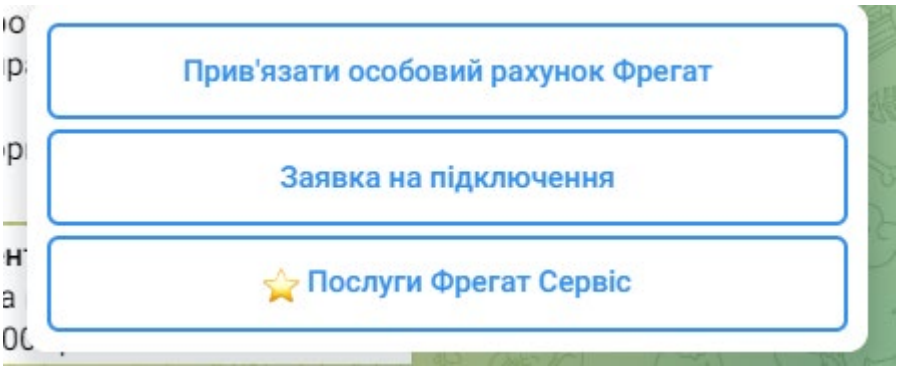

Рис. 3.4. Клавіатура бота від «Фрегат»

Як видно, на клавіатурі присутня кнопка інтеграції свого рахунку. Якщо ви проведете цей процес, то можна буде поповнювати рахунок прямо в боті. Тобто бот надає послугу оплати, як і у випадку зі створеною програмою.

Більшість інших аналогів, наприклад, боти з покупки меблів, або поповнення рахунку у інших провайдерів мають схожі особливості. Звідси можна зробити висновок, що мета роботи була досягнута і створений бот може конкурувати на ринку. В якості мінімального висновку можна сказати, що частина дослідження пройшла успішно.

3.4. Аналіз інформації протягом дослідження

На початку роботи, після створення бота та введення його в експлуатацію, фіксувались такі показники як кількість переходів за посиланням у боті, кількість людей які відвідують бота щодня, використання команди допомоги, зроблені замовлення та прибуток. Перший етап дослідження проводився десять днів. Дослідження було розпочато 20.10.23, після цього був збір інформації про клієнтів, використання бота та його ефективність. Наприкінці етапу отримано висновок, що середня кількість переглядів боту на день сягає 73, з них 25 чоловік користуються посиланням на відео для короткого «гайду» по боту, 38 людей користувались кнопкою

допомоги з викликом відомих команд. В середньому на день було до 11 замовлень у боті. Середня вартість покупок в день склала 1594 гривні з урахуванням комісії. Звідси можна зробити висновок, що бот приніс прибуток за 10 днів в 15940 гривень.

Після закінчення першого етапу було прийняте рішення про модифікацію бота, оскільки певна частина клієнтів залишила відгуки і внесла пропозиції. В результаті оновлення бота він отримав можливість розважати клієнтів відповідями та картинками, якщо він помічає певні слова. Було збільшене меню бота. Певна частина коду була скорегована. Авторським рішенням було додати в якості ще однієї новації погодний «віджет». Таке рішення прийняте для того, аби клієнт думав про те, чи варто йому бути присутнім «offline». Також був доданий QR-код для збільшення популярності.

По закінченню модифікації почався другий етап в 30 днів. Початок був 2 листопада.

Рис.3.5. QR-код з посиланням на створений бот

Після закінчення другої частини дослідження були отримані наступні результати: кількість відвідувачів зросла до 131 на день, навчальним відео користувались 43 людини, ще 70 використала команду для допомоги. Середнє значення замовлень на день зросло до 26. Прибуток майже не змінився, але за рахунок замовлень став більшим – 3341 гривня. Відповідно місячний здобуток став 100230 гривні за допомогою використання бота.

3.5. Висновки

В ході дослідження були зібрані дані про клієнтів. Зафіксовані показники кількісті переходів за посиланням у боті, кількість людей які відвідують бота щодня, використання команди допомоги, зроблені замовлення та прибуток. На кінці першого етапу дослідження зроблений висновок про необхідність оновлення бота до необхідного рівня конкуренції. Для цього були враховані показники ефективності бота, швидкості, доходу, а також зроблене порівняння з аналогами на ринку. В результаті оновлення бот став ефективніше та приніс прибуток. На першому етапі зафіксовано прибуток в 15940 гривень. На другому етапі він зріс до 33410 за 10 днів та відповідно 100230 за місяць.

Спираючись на ці дані можна зробити висновок про успішне завершення дослідження ефективності розробленого програмного застосунку, а також ефективне його використання.

РОЗДІЛ 4
ЕКОНОМІЧНИЙ РОЗДІЛ

4.1 Розрахунок трудомісткості і вартості розробки програмного продукту

Розробка Telegram-bot безумовно потребує від себе розрахунків витрат на неї, адже програмний застосунок має власну ціну та трудомісткість.

Отримали наступні дані:

1) Заробітна плата (з/п) розробника 100 гривень за одну годину.

2) Передбачені оператори 500.

3) Машино-годинні витрати техніки – 30грн/година.

4) Коефіцієнт корекції при розробці 0.2.

5) Коефіцієнт кваліфікації розробника в залежності від опиту – 1.

6) Коефіцієнт збільшення витрат в разі нечіткої постанови задачі – 1.1.

7) Коефіцієнт складності програми 1.

Рівень нормалізації при створені програми залежний від багатьох чинників, в тому числі непередбачених, моральних тощо. Спираючись на ці факти будемо розраховувати трудомісткість таким чином:

Трудомісткість розробки, формула у людино годинах(л/г):

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{ л/г.} \quad (3.1)$$

t_u – витрати праці (в/п) дослідження алгоритму рішення задачі.

t_a - в/п розробки блок схеми алгоритму.

t_n - в/п програмування за блок схемою.

$t_{отл}$ - в/п налагодження програми.

$t_д$ - в/п підготовки документації.

$t_о$ - в/п підготовки і опису поставленої задачі, зазвичай 50.

Через умовну кількість операторів програмного забезпечення визначаються складові витрати праці.

Умовна кількість операторів:

$$Q = q * C * (1 + p), \tag{3.2}$$

де q – передбачуване число операторів

C – коефіцієнт складності програми

p – коефіцієнт корекції програми в ході її розробки

$$Q = 500 * 1 *(1 + 0,2) = 600$$

В/п дослідження алгоритму розв’язку задачі:

$$t_u = \frac{Q * B}{(75…85) * k} \text{ л/г} \tag{3.3}$$

B – Коефіцієнт збільшення витрат в разі нечіткої постанови задачі

k – коефіцієнт кваліфікації розробника в залежності від опиту

$$t_u = \frac{600 * 1,1}{75 * 1} = 8.8 \text{ л/г}$$

Витрати розробки блок схеми алгоритму:

$$t_a = \frac{Q}{(20…25) * k} \text{ л/г} \quad (3.4)$$

$$t_a = \frac{600}{24 * 1} = 25 \text{ л/г}$$

В/п програмування за блок схемою:

$$t_n = \frac{Q}{(20…25) * k} \text{ л/г} \quad (3.5)$$

$$t_n = \frac{600}{25 * 1} = 24 \text{ л/г}$$

Витрати налагодження програми:

1) Налагодження звичайного завдання:

$$t_{отл} = \frac{Q}{(4..5) * k} \text{ л/г} \quad (3.6)$$

$$t_{отл} = \frac{600}{5 * 1} = 120 \text{ л/г}$$

2) Налагодження комплексного завдання:

$$t^k_{отл} = 1{,}5 * t_{отл}, \text{ л/г} \quad (3.7)$$

$$t^{k}_{отл} = 1,5 * 120 = 180, \text{ л/г}$$

Витрати на підготовку документації:

$$t_{д} = t_{др} + t \quad \text{л/г} \tag{3.8}$$

$t_{др}$ – трудомісткість підготовки матеріалів

$$t_{др} = \frac{Q}{15..20 * k} \text{ л/г} \tag{3.9}$$

$$t_{др} = \frac{600}{20 * 1} = 30 \text{ л/г}$$

$t_{до}$ – загальна трудомісткість оформлення документації:

$$t_{до} = 0,75 * t_{др} \text{ л/г} \tag{3.10}$$

$$t_{до} = 0,75 * 30 = 22.5 \text{ л/г}$$

$$t_{д} = 30 + 22.5 = 52.5 \text{ л/г}$$

Трудомісткість розробки бота:

$$t = 50 + 8,8 + 25 + 24 + 120 + 52,5 = 280,3 \text{ л/г}$$

Отримано трудомісткість програми.

3.2. Розрахунок витрат на створення програми

Необхідні гроші при розробці програмного забезпечення:

$$К_{по} = З_{зп} + З_{мв} \text{ гривень} \quad (3.11)$$

З/п розробника програми:

$$З_{зп} = t * С_{пр} \text{ гривень} \quad (3.12)$$

t – загальна трудомісткість програми

$С_{пр}$ – середня з/п (у гривнях на годину)

$$З_{зп} = 280,3 * 100 = 28030 \text{ гривень}$$

Необхідний час налагодження програми виражений у вартості:

$$З_{мв} = t_{отл} * С_{мч} \text{ гривень} \quad (3.13)$$

$t_{отл}$ – трудомісткість налагодження програми

$С_{мч}$ – вартість машинних годин (у гривнях на годину)

$$З_{мв} = 180 * 30 = 5400 \text{ гривень}$$

Затрати на створення ПЗ є частиною ОКВ в даному випадку:

$$К_{по} = 28030 + 5400 = 33530 \text{ гривень}$$

Очікуваний період створення програми:

$$T = \frac{t}{B_k * F_p} \text{ місяць} \quad (3.14)$$

B_k – Число виконавців, зазвичай береться один

F_p – Часова норма роботи на місяць 176 годин

$$T = \frac{280,3}{1 * 176} = 1,5 \text{ місяць}$$

4.3. Висновок

Розроблений програмний продукт був призначений для спілкування з користувачами, збору інформації та своєї безпосередньої задачі ведення бізнес-процесу. Важливо було надати зрозумілий і легкий інтерфейс, а також основні можливості бота, розробити код для збору інформації і провести дослідження. В результаті необхідними стали оновлення та збільшення ефективності і продуктивності бота. Згідно оцінки, вартість програмного продукту склала 17999 гривень за умови часу розробки в 10 робочих днів. Окрім цього зафіксовано сумарний прибуток за 40 днів в 116170 гривень.

ВИСНОВКИ

В результаті ведення розробки та дослідження був розроблений програмний продукт. Цей продукт уявляє з себе бот для платформи Telegram. Для розробки була використана мова Python та один з кращих фреймворків – Aiogram. Бот був призначений для збору інформації про клієнтів, надання їм послуг компанії, можливостей купувати товар. Окрім цього необхідно було дослідити ефективність боту та його можливість вести конкуренцію на ринку праці.

Після кінця дослідження та отримання даних маємо такий результат:

- Зроблено замір ефективності бота.

- Зроблене порівняння з вже існуючими аналогами.

- Зроблений замір прибутковості бота.

- Бот отримав оновлення по завершенню першої частини дослідження.

- Бот приніс прибуток для компанії.

Під час дослідження бот за допомогою бота отримана інформація про кількість клієнтів на день в середньому, кількість натискань на кнопку з посиланням, кількість введених команд допомоги, кількість зроблених замовлень та отриманий прибуток.

Враховуючи ці та інші фактори, можна зробити висновок, що в результаті роботи було розроблено бот з усіма виконаними критеріями, після цього отримані дані про клієнтів та зроблені оновлення. На кінці дослідження стає зрозуміли факт можливості цього бота вести конкуренцію.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лекція про чат-ботів / URL: https://youtu.be/1iKM1pNP4Aw (дата звернення 10.10.2023).
2. Створення ботів / URL: https://youtu.be/qDGHNYFZFgQ (дата звернення 10.10.2023).
3. Лекція про бізнес на ботах / URL: https://youtu.be/q7oHY7Va9Pw (дата звернення 10.10.2023).
4. Лекція про збільшення прибутку / URL: https://youtu.be/ke7LP4LDXSg (дата звернення 10.10.2023).
5. Система ENIAC / URL: https://uk.wikipedia.org/wiki/ENIAC (дата звернення 10.10.2023).
6. Система UNIVAC / URL: https://ru.wikipedia.org/wiki/UNIVAC_I (дата звернення 11.10.2023).
7. Teletype model 33 / URL: https://en.wikipedia.org/wiki/Teletype_Model_33 (дата звернення 11.10.2023).
8. IBM 3270 / URL: https://ru.wikipedia.org/wiki/IBM_3270 (дата звернення 11.10.2023).
9. Документація Python / URL: https://docs.python.org/uk/3/ (дата звернення 11.10.2023).
10. Оператори в Python / URL: https://acode.com.ua/operators-python/ (дата звернення 12.10.2023).
11. JavaScript / URL: https://ru.wikipedia.org/wiki/JavaScript (дата звернення 12.10.2023).
12. API / URL: https://en.wikipedia.org/wiki/API (дата звернення 12.10.2023).
13. Документація фреймворку aiogram / URL: https://docs.aiogram.dev/en/latest/ (дата звернення 12.10.2023).

14. Економічний розділ / URL: https://jak.bono.odessa.ua/articles/ php (дата звернення 13.10.2023).

15. David E. Lundstrom. A Few Good Men from Univac. — Mit Press, 1987. — 300 p. (дата звернення 13.10.2023).

16. Scott McCartney. ENIAC: The Triumphs and Tragedies of the World's First Computer. — Berkley Books, 2001. — 262 p. (дата звернення 13.10.2023).

17. Herman H. Goldstine. The Computer from Pascal to von Neumann. — Princeton University Press, 1980. — 365 p. (дата звернення 13.10.2023).

18. Nancy B. Stern. From Eniac to UNIVAC: An Appraisal of the Eckert-Mauchy Computers. — Digital Press, 1981. — 286 p. (дата звернення 13.10.2023).

19. William Aspray. John von Neumann and the Origins of Modern Computing. — MIT Press, 1990. — 394 p. (дата звернення 13.10.2023).

20. Scott McCartney. ENIAC: The Triumphs and Tragedies of the World's First Computer. — Berkley Books, 2001. — 262 p. (дата звернення 14.10.2023).

21. Raúl Rojas, Ulf Hashagen. The First Computers: History and Architectures. — MIT Press, 2002. — 471 p. (дата звернення 14.10.2023).

22. Kristine C. Harper. Weather by the Numbers: The Genesis of Modern Meteorology. — MIT Press, 2008. — 320 p. (дата звернення 14.10.2023).

23. Thomas Haigh, Mark Priestley, Crispin Rope. ENIAC in Action: Making and Remaking the Modern Computer. — The MIT Press, 2016. — 360 p. (дата звернення 15.10.2023).

24. Starr, Samuel S. (July 1977). "Inside the Amazing ASR 33" (PDF). Kilobaud Microcomputing. pp. 98–100. (дата звернення 15.10.2023).

25. Tomasi, Wayne. "Electronic Communications Systems: fundamentals through advanced", Prentice Hall, 2001, p. 531. (дата звернення 15.10.2023).

26. Smith, D. K. Fumbling the Future : How Xerox Invented, Then Ignored, the First Personal Computer : [англ.] / D. K. Smith, R. C. Alexander. — New York : William Morrow and Company, 1999. — 274 p. (дата звернення 15.10.2023).

27. Rheingold, H. Tools for Thoughtruen : [англ.]. — MIT Press, 2000. — 336 p. (дата звернення 15.10.2023).

28. Thomas A Wadlow. The Xerox Alto Computer : [англ.] // Byte[en]. — 1981. — Т. 06, № 09 (September). — С. 58−68. (дата звернення 15.10.2023).

29. Лекція з Teletype / URL: https://jak.bono.odessa.ua/articles/ php (дата звернення 16.10.2023).

ДОДАТОК А

КОД ПРОГРАМИ

create_bot.py # основні функції

```
from aiogram import Bot, types
from aiogram.dispatcher import Dispatcher
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton   #this1isforquiz
import os
from keyboards import client_kb
from aiogram.types import InlineKeyboardMarkup, InlineKeyboardButton
from aiogram.contrib.fsm_storage.memory import MemoryStorage
#THE NEXT 4 MADE FOR SURVEY
#from email.mime.multipart import MIMEMultipart
#from email.mime.text import MIMEText
#import smtplib
#from aiogram.contrib.middlewares.logging import LoggingMiddleware

storage=MemoryStorage()

bot = Bot(token=os.getenv('TOKEN'))
dp = Dispatcher(bot, storage=storage)
#dp.middleware.setup(LoggingMiddleware())  #FORSURVEY

#Кнопка инлайн
urlkb = InlineKeyboardMarkup(row_width=1)
urlButton = InlineKeyboardButton(text='новини', url='https://youtu.be/ePy-MZb6rUg')
urlkb.add(urlButton)

@dp.message_handler(commands='новини')
async def url_command(message : types.Message):
        await message.answer('новини:', reply_markup=urlkb)
        await message.delete()
```

```
@dp.message_handler(lambda message: 'команди' in message.text)
async def commands(message: types.Message):
        await message.answer('напиши /help')
        await message.delete()


@dp.message_handler(commands=['video'])
async def video_command(message : types.Message):
        await message.answer('https://youtu.be/PDjTOX0Ws9A')    #D
        await message.delete()


@dp.message_handler(lambda message: 'gachi' in message.text)
async def gachi_command(message: types.Message):
        await message.answer('https://youtu.be/NdqbI0_0GsM')
        await message.delete()


@dp.message_handler(lambda message: 'гачи' in message.text)
async def gachi_command(message: types.Message):
        await message.answer('https://youtu.be/NdqbI0_0GsM')
        await message.delete()


#@dp.message_handler(lambda message: 'дед' in message.text)       # Отправка картинки
#async def answer_command(message: types.Message):
#        await bot.send_photo(message.chat.id, photo=open('D:\Arts\kaseki.png', 'rb'))

@dp.message_handler(lambda message: 'дед' in message.text)       # Отправка картинки
async def answer_command(message: types.Message):

    await message.reply("\/")
    await bot.send_photo(message.chat.id, photo=open('answerphoto\kaseki.png', 'rb'))

@dp.message_handler(lambda message: 'Ceb' in message.text)       # Отправка картинки
async def answer_commandtwo(message: types.Message):
```

```
    await message.reply("ready")
    await bot.send_photo(message.chat.id, photo=open('answerphoto\Ceeeb.jpg', 'rb'))

#Location
```

to_json.py # перетворення файлів в інший формат

```
import json

ar = []

with open('cenz.txt', encoding='utf-8') as r:
    for i in r:
        n = i.lower().split('\n')[0]
        if n != '':
            ar.append(n)

with open('cenz.json', 'w', encoding='utf-8') as e:
    json.dump(ar, e)
```

bot_telegram.py # bot start

```
from aiogram.utils import executor
from create_bot import dp
from data_base import sqlite_db


async def on_startup(_):
            print('Its working')
            sqlite_db.sql_start()


from handlers import client, admin, other

client.register_handlers_client(dp)
admin.register_handlers_admin(dp)
other.register_handlers_other(dp)
```

executor.start_polling(dp, skip_updates=True, on_startup=on_startup)

bot_run.bat # BOT TOKEN

@echo off

call %~dp0telegram_bot\venv\Scripts\activate

cd %~dp0telegram_bot

set TOKEN=5505323696:AAG3sahTC3RPxF1w1ZnPIfKT1OQgwJtWJl4

python bot_telegram.py

pause

client_kb.py # CLIENT KEYBOARD

from aiogram.types import ReplyKeyboardMarkup, KeyboardButton

#Кнопки для клієнта

b1 = KeyboardButton('/Розклад')

b2 = KeyboardButton('/Location')

b3 = KeyboardButton('/меню')

b4 = KeyboardButton('/контакт')

b5 = KeyboardButton('/ваш номер', request_contact=True)

b6 = KeyboardButton('/де ви', request_location=True)

kb_client = ReplyKeyboardMarkup(resize_keyboard=True)

kb_client.row(b1, b2, b3, b4, b5).add(b6)

admin_kb.py # admin keyboard

from aiogram.types import ReplyKeyboardMarkup, KeyboardButton

```
#Кнопки для адміністратора закладу чи менеджерів завантаження та розвантаження
button_load = KeyboardButton('/Завантажуй')
button_delete = KeyboardButton('/Видаляй')

button_case_admin = ReplyKeyboardMarkup(resize_keyboard=True).add(button_load)\
        .add(button_delete)
```

Other.py # other functions

```
from aiogram import types, Dispatcher
from create_bot import dp
import json, string



# Регістрація для використання цензурування
async def echo_send(message : types.Message):
        if {i.lower().translate(str.maketrans('', '', string.punctuation)) for i in message.text.split(' ')}\
            .intersection(set(json.load(open('cenz.json')))) != set():
            await message.reply('не роби так')
            await message.delete()

def register_handlers_other(dp : Dispatcher):
    dp.register_message_handler(echo_send)
```

client.py # client keyboard

```
from aiogram import types, Dispatcher
from create_bot import dp, bot
from keyboards import kb_client
from keyboards import client_kb
from data_base import sqlite_db



# Налаштування кнопки та команди старт для початку користування ботом і спілкування зним
async def command_start(message : types.Message):
        try:
            await bot.send_message(message.from_user.id,'Добрий день, смачного', reply_markup=kb_client)
```

```
            await message.delete()
        except:
            await message.reply('Пиши йому -> напиши команду допомоги /help :\n https://t.me/DogmatixBot')

# НАЛАШТУВАННЯ КОМАНДИ ДОПОМОГИ
async def command_help(message : types.Message):
        await bot.send_message(message.from_user.id, 'бот має команди: \nhelp, \nновини, \nРозклад, \nLocation, \nменю, \nконтакт, \nstart, \nstart quiz')
        await message.delete()



# Створення команди для перевірки часу роботи закладу
async def pizza_shedule_command(message : types.Message):
        await bot.send_message(message.from_user.id, 'пн-пт з 10 до 9, будні 10-9')
        await message.delete()

# Збір персональної інформації від користувача про його місцезнаходження
async def pizza_location_command(message : types.Message):
        await bot.send_message(message.from_user.id, 'вул перемоги 1')
        await message.delete()

# Інформація необхідна для клієнтів про контакт з компанією
async def pizza_contact_command(message : types.Message):
        await bot.send_message(message.from_user.id, '38076****95')
        await message.delete()

# допоміжна функція, необхідна для розробника при швидкому виклику клавіатури
async def test_command(message : types.Message):
        await bot.send_message(message.from_user.id, 'клавіатура актив', reply_markup=kb_client)
        await message.delete()

# Функція з допомогою якої реалізється меню бота
async def pizza_menu_command(message : types.Message):
        await sqlite_db.sql_read(message)
        await message.delete()

# Команда яка дозволить зібрати інформацію про знаходження клієнта
```

```
@dp.message_handler(commands=['Share_location'])
async def Share_location(message: types.Message):
    await message.answer("Please share your location:", reply_markup=kb_client)

# ПОЛЕ РЕГІСТРАЦІЇ КОМАНД ТА ФУНКЦІЙ

def register_handlers_client(dp : Dispatcher):
        #dp.register_message_handler(Share_location, commands=['Share_location'])
    dp.register_message_handler(command_start, commands=['start'])
    dp.register_message_handler(command_help, commands=['help'])
    dp.register_message_handler(pizza_shedule_command, commands=['Розклад'])
    dp.register_message_handler(pizza_location_command, commands=['Location'])
    dp.register_message_handler(pizza_contact_command, commands=['контакт'])
    dp.register_message_handler(test_command, commands=[''])
    dp.register_message_handler(pizza_menu_command, commands=['меню'])
```

admin.py #створення FSM

```
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters.state import State, StatesGroup
from aiogram import types, Dispatcher
from create_bot import dp, bot
from aiogram.dispatcher.filters import Text
from data_base import sqlite_db
from keyboards import admin_kb
from aiogram.types import InlineKeyboardMarkup, InlineKeyboardButton



ID = None

class FSMAdmin(StatesGroup):
        photo = State()
        name = State()
        description = State()
        price = State()

# Вхід в режим продавця
```

```
async def make_changes_command(message: types.Message):
        global ID
        ID = message.from_user.id
        await bot.send_message(message.from_user.id, 'Що треба', reply_markup=admin_kb.button_case_admin)
        await message.delete()

# Початок створення нового меню

async def cm_start(message : types.Message):
        if message.from_user.id == ID:
            await FSMAdmin.photo.set()
            await message.reply('загрузи фото')

# Статус для виходу з режиму



async def cancel_handler(message: types.Message, state: FSMContext):
        if message.from_user.id == ID:
            current_state = await state.get_state()
            if current_state is None:
                    return
            await state.finish()
            await message.reply('ok')

# додавання фотографії для товару

async def load_photo(message: types.Message, state: FSMContext):
        if message.from_user.id == ID:
            async with state.proxy() as data:
                    data['photo'] = message.photo[0].file_id
            await FSMAdmin.next()
            await message.reply("название")

# Заповнення другої відповіді

async def load_name(message: types.Message, state: FSMContext):
```

```
        if message.from_user.id == ID:
            async with state.proxy() as data:
                    data['name'] = message.text
            await FSMAdmin.next()
            await message.reply("описание")

# Заповнення форми з ціною продукту

async def load_description(message: types.Message, state: FSMContext):
        if message.from_user.id == ID:
            async with state.proxy() as data:
                    data['description'] = message.text
            await FSMAdmin.next()

#           await sqlite_db.sql_add_command(state)
            await message.reply("цена")
#           await state.finish()

# Останній стан для машини

async def load_price(message: types.Message, state: FSMContext):
        if message.from_user.id == ID:
            async with state.proxy() as data:
                    data['price'] = message.text
#           await FSMAdmin.next()

            await sqlite_db.sql_add_command(state)
            await message.reply("готово")
            await state.finish()

# реалізація колбек функції
@dp.callback_query_handler(lambda x: x.data and x.data.startswith('del '))
async def del_callback_run(callback_query: types.CallbackQuery):
        await sqlite_db.sql_delete_command(callback_query.data.replace('del ', ''))
        await callback_query.answer(text=f'{callback_query.data.replace("del ", "")} Deleted.', show_alert=True)

# Реалізація кнопки видалення для менеджерів та адміну
```

```
@dp.message_handler(commands='Delete')
async def delete_item(message: types.Message):
    if message.from_user.id == ID:
        read = await sqlite_db.sql_read2()
        for ret in read:
            await bot.send_photo(message.from_user.id, ret[0], f'{ret[1]}\n Description: {ret[2]}\n Price {ret[-1]}')
            await bot.send_message(message.from_user.id, text='^^^', reply_markup=InlineKeyboardMarkup().\
                add(InlineKeyboardButton(f'Delete {ret[1]}', callback_data=f'del {ret[1]}')))


# місце регістрації команд та хендлерів
def register_handlers_admin(dp : Dispatcher):
    dp.register_message_handler(cm_start, commands=['Download'], state=None)
    dp.register_message_handler(cancel_handler, state="*", commands='revoke')
    dp.register_message_handler(cancel_handler, Text(equals='revoke', ignore_case=True), state="*")
    dp.register_message_handler(load_photo, content_types=['photo'], state=FSMAdmin.photo)
    dp.register_message_handler(load_name, state=FSMAdmin.name)
    dp.register_message_handler(load_description, state=FSMAdmin.description)
    dp.register_message_handler(load_price, state=FSMAdmin.price)
    dp.register_message_handler(make_changes_command, commands=['vendor'], is_chat_admin=True)
# другі можливі варіанти для хендлерів вище, призначених для видалення
#   dp.register_message_handler(del_callback_run, lambda x: x.data and x.data.startswith('del '))
#   dp.register_message_handler(delete_item, commands='Delete')
```

Sqlite_db.py # база даних сервісу

```
import sqlite3 as sq
from create_bot import bot
# зєднання з базою даних
def sql_start():
    global base, cur
    base = sq.connect('pizza_cool.db')
    cur = base.cursor()
    if base:
        print('Хранилище работает')
    base.execute('CREATE TABLE IF NOT EXISTS menu(img TEXT, name TEXT PRIMARY KEY, description TEXT, price TEXT)')
    base.commit()
```

```
async def sql_add_command(state):
        async with state.proxy() as data:
                cur.execute('INSERT INTO menu VALUES (?, ?, ?, ?)', tuple(data.values()))
                base.commit()


async def sql_read(message):
        for ret in cur.execute('SELECT * FROM menu').fetchall():
           await bot.send_photo(message.from_user.id, ret[0], f'{ret[1]}\n Description: {ret[2]}\n Price {ret[-1]}')

async def sql_read2():
        return cur.execute('SELECT * FROM menu').fetchall()


async def sql_delete_command(data):
        cur.execute('DELETE FROM menu WHERE name == ?', (data,))
        base.commit()
```

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Воробйов_магдип.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Воробйов_магдип.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
telegram_bot	Папка. Містить код програми та файли
Презентація	
Воробйов_презентація.ppt	Презентація кваліфікаційної роботи