Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота магістра

на тему: «Розробка інформаційної системи продажу товарів з використанням сучасних технологій месенджерів»

Виконав: студент групи К23-1М

Спеціальність 122 Комп'ютерні науки

Куцарський Є.Є.
(прізвище та ініціали)

Керівник к.т.н., доц. Мала Ю.А.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент Дніпровський державний технічний університет
(місце роботи)

доцент кафедри математичного моделювання та системного аналізу
(посада)

к.т.н., доц. Волосова Н.М.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2025

АНОТАЦІЯ

Куцарський Є.Є. Розробка інформаційної системи продажу товарів за допомогою чат-боту.

Кваліфікаційна робота на здобуття освітнього ступеня магістр за спеціальністю 122 «Комп'ютерні науки». – Університет митної справи та фінансів, Дніпро, 2025.

Об'єкт дослідження – інформаційні системи, що використовують чат-ботів для автоматизації бізнес-процесів.

Предмет дослідження – методи та засоби розробки чат-ботів для продажу товарів за допомогою месенджерів.

Метою роботи є розробка інформаційної системи продажу товарів за допомогою чат-бота, що дозволить автоматизувати процес замовлення, зменшити навантаження на операторів та покращити користувацький досвід.

Кваліфікаційна робота присвячена розробці чат-бота для автоматизації продажу товарів у месенджерах. Проведено аналіз сучасних методів створення чат-ботів та обґрунтовано вибір оптимальних технологій. Розроблено архітектуру чат-бота, що включає модулі для обробки запитів користувачів, управління каталогом товарів, оформлення замовлень та адміністрування. Реалізовано систему інтеграції з базою даних для збереження інформації про товари та замовлення. Виконано тестування працездатності та продуктивності системи, що підтвердило її ефективність у реальних умовах експлуатації.

Практична цінність роботи полягає у створенні інформаційної системи, яка може бути використаною підприємцями для автоматизації продажу товарів, що дозволяє покращити якість обслуговування клієнтів, знизити операційні витрати та підвищити швидкість обробки замовлень.

Ключові слова: Чат-бот, Telegram Bot API, Aiogram, Python.

ABSTRACT

Kutsarskyi Ye. Development of an Information System for Selling Goods Using Modern Messenger Technologies.

Diploma project for obtaining a master's degree in speciality 122 "Computer Science." - University of Customs and Finance, Dnipro, 2025.

The object of the study is information systems that utilize chatbots for business process automation.

The subject of the study is methods and tools for developing chatbots for selling goods through messengers.

The purpose of this work is to develop an information system for selling goods via a chatbot, which will automate the ordering process, reduce the workload on operators, and improve the user experience.

This qualification work is dedicated to the development of a chatbot for automating the sale of goods in messengers. An analysis of modern methods for creating chatbots has been conducted, and the selection of optimal technologies has been justified. The chatbot architecture has been developed, including modules for processing user requests, managing the product catalog, handling orders, and administration. A system for database integration has been implemented to store information about products and orders. The functionality and performance of the system have been tested, confirming its efficiency in real-world operating conditions.

The practical value of this work lies in the creation of an information system that can be used by entrepreneurs to automate the sale of goods, thereby improving customer service quality, reducing operational costs, and increasing order processing speed.

Keywords: Chatbot, Telegram Bot API, Aiogram, Python.

ЗМІСТ

ВСТУП

У сучасному світі програмування та автоматизації бізнес-процесів важливим завданням є розробка ефективних та оптимізованих програмних рішень, здатних автоматизувати взаємодію між користувачами та системами. Це особливо актуально у сфері електронної комерції, де швидкість обробки запитів, зручність користування та доступність сервісу безпосередньо впливають на ефективність бізнесу та рівень задоволеності клієнтів.

Актуальність даного дослідження зумовлена необхідністю створення продуктивних алгоритмічних рішень для автоматизації процесу продажу товарів, зокрема за допомогою чат-ботів у месенджерах. Чат-боти дозволяють значно підвищити ефективність взаємодії з клієнтами, забезпечуючи миттєві відповіді на запити, обробку замовлень та персоналізовані рекомендації. Використання таких технологій допомагає компаніям скоротити витрати на обслуговування, оптимізувати бізнес-процеси та підвищити конкурентоспроможність на ринку [1].

Однією з найпопулярніших платформ для розробки чат-ботів є месенджер Telegram. Завдяки відкритому API, широким можливостям інтеграції та зручному інтерфейсу він став вибором багатьох підприємств для автоматизації продажів. Чат-боти у Telegram можуть взаємодіяти з користувачами у режимі реального часу, обробляти платежі, керувати замовленнями та забезпечувати швидке реагування на запити.

Метою дослідження є розробка інформаційної системи продажу товарів за допомогою чат-боту, що дозволить автоматизувати процес замовлення, зменшити навантаження на операторів та покращити користувацький досвід.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Провести аналіз наукових джерел та сучасних підходів до розробки чат-ботів.
2. Дослідити особливості реалізації чат-ботів для електронної комерції.

3. Оцінити можливості Telegram Bot API та вибрати оптимальні інструменти для розробки.

4. Розробити програмну реалізацію чат-бота з підтримкою оформлення замовлень та інтеграції з базою даних.

5. Виконати тестування системи та оцінити її ефективність у реальних умовах використання.

Об'єктом дослідження є інформаційні системи, що використовують чат-ботів для автоматизації бізнес-процесів.

Предметом дослідження є методи та засоби розробки чат-ботів для продажу товарів через месенджери.

Новизна дослідження полягає у використанні сучасних месенджерів із програмними інструментами автоматизації продажів, що дає змогу забезпечити високий ступінь інтеграції між клієнтським інтерфейсом та внутрішніми бізнес-процесами компанії.

Практичне значення роботи полягає у створенні функціонального прототипу інформаційної системи, що може бути використана підприємцями для автоматизації продажу товарів через месенджер.

Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. Обсяг роботи становить 73 сторінок основного тексту, 41 рисунків та 2 таблиці.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАННЯ ДОСЛІДЖЕННЯ

1.1 Аналіз публікацій щодо розробки чат-ботів

Чат-боти або просто боти – це програмні агенти, які виконують різноманітні завдання автоматично, без необхідності постійного втручання людини. Вони створюються для широкого спектру застосувань, таких як спілкування з користувачами на веб-сайтах і в месенджерах, автоматизація рутинних процесів, моніторинг, аналіз даних, оптимізація використання ресурсів тощо. Розробка ботів може здійснюватися за допомогою різних мов програмування та технологій, включно зі штучним інтелектом, машинним навчанням, обробкою природної мови та іншими інноваційними рішеннями. Їх популярність стрімко зростає в таких сферах, як бізнес, маркетинг, комунікація, підтримка клієнтів та багато інших.

Однією з найпоширеніших категорій є боти технічної підтримки. Їхній функціонал охоплює:

- швидкі та точні відповіді на питання;
- переадресацію до спеціалістів для вирішення складніших запитів;
- збір аналітичних даних, корисних для вдосконалення продукту або сервісу.

Ідея створення чат-ботів зародилася ще в середині XX століття. У 1950 році британський математик Алан Тюрінг висунув революційну думку про можливість створення програм, здатних мислити та спілкуватися подібно до людей. Він передбачав, що до 2020 року штучний інтелект досягне такого рівня розвитку, коли відрізнити його від реальної людини стане надзвичайно важко [1].

Перший прототип чат-бота, названий Eliza, був розроблений у 1966 році професором Джозефом Вейценбаумом із Массачусетського технологічного інституту (MIT). Хоча її механізм роботи був доволі примітивним, Eliza

справила неабияке враження на користувачів, імітуючи діалог через аналіз ключових слів у повідомленнях. Наприклад, на фразу «Моя мама добре готує» бот відповідав зустрічним запитанням: «Розкажи більше про свою сім'ю», створюючи ілюзію реальної бесіди.

Ця технологія, заснована на пошуку ключових слів та використанні попередньо визначених шаблонів, стала проривом у комп'ютерних науках і поклала початок новій епосі віртуальної комунікації. У 1994 році термін «чаттербот» увійшов у вжиток, закріпивши за такими програмами унікальну нішу у світі технологій.

Сьогодні чат-боти – це більше, ніж просто інструмент. Вони є символом інтеграції штучного інтелекту в наше повсякденне життя, допомагаючи автоматизувати рутинні процеси, підвищувати ефективність роботи бізнесу та забезпечувати безперервну підтримку для користувачів.

Серед основних переваг чат-ботів – автоматизація повторюваних завдань, доступність у будь-який час доби, висока швидкість виконання, можливість миттєвого спілкування з користувачами, скорочення витрат на персонал і покращення якості обслуговування клієнтів.

Разом із перевагами чат-боти також стикаються з низкою викликів. Процес розробки та підтримки таких систем часто потребує значних фінансових і часових витрат. Особливо це стосується складних ботів, які повинні розуміти неоднозначні або нестандартні запити. Крім того, підтримка актуальності чат-бота вимагає постійного аналізу, корекції та оновлення, оскільки він має адаптуватися до нових вимог користувачів і змін у бізнес-процесах.

Незважаючи на це, потенціал чат-ботів у бізнесі є великим. Вони дозволяють компаніям оптимізувати клієнтське обслуговування, надаючи миттєву підтримку та відповіді на запити без затримок. Завдяки автоматизації повторюваних завдань, співробітники можуть зосередитися на важливіших і стратегічних цілях.

Однією з найважливіших переваг чат-ботів є їх здатність працювати безперервно – 24 години на добу, 7 днів на тиждень, 365 днів на рік. Це дозволяє компаніям забезпечувати стабільний контакт із клієнтами в будь-який час. Крім того, боти пропонують персоналізований підхід: аналізуючи історію взаємодій користувача, адаптують відповіді роблячи спілкування максимально зручним і релевантним.

Створення ефективного чат-бота передбачає чітке визначення завдань, які він повинен виконувати. Універсальність цієї технології дозволяє використовувати її в різних напрямках, таких як:

1. Оптимізація внутрішніх процесів – наприклад, корпоративні HR-боти, автоматизація комунікації між відділами або обробка документів;
2. Клієнтська підтримка – відповіді на поширені запитання (FAQ), онлайн-допомога, технічна підтримка;
3. Маркетинг і реклама – проведення рекламних кампаній, персоналізовані розсилки, повідомлення про знижки та акції;
4. Продажі – автоматизація процесу продажів, обробка замовлень, підтримка клієнтів під час купівлі.

Щоб забезпечити успіх у розробці чат-бота, важливо провести детальний аналіз процесів у компанії, визначивши, які з них є найтрудомісткішими. Залучення експертів дозволить знайти оптимальні шляхи автоматизації, скоротити витрати часу та підвищити ефективність роботи [2].

Таким чином, чат-боти не лише полегшують взаємодію з клієнтами, але й створюють можливості для розвитку бізнесу, забезпечуючи високий рівень сервісу, який відповідає сучасним потребам споживачів.

1.2 Класифікація чат-ботів

Щороку компанії обробляють колосальні обсяги запитів на підтримку клієнтів – близько 265 мільярдів. Це завдає суттєвого фінансового

навантаження, оскільки загальні витрати на їх обслуговування сягають 1,3 трильйона доларів. Проте до 80% запитів можуть бути автоматизовані, що дозволяє значно зменшити витрати та підвищити ефективність роботи. Завдяки впровадженню автоматизації та самообслуговування компанії можуть скоротити витрати на підтримку клієнтів до 30%, дозволяючи персоналу зосередитися на більш складних завданнях.

Перспективи ринку чат-ботів

За прогнозами Grand View Research, до 2025 року ринок чат-ботів досягне 1,25 мільярда доларів. У Великобританії 57% споживачів уже знають, що таке чат-боти, а 35% бажають, щоб їх використовувало більше компаній. Це вказує на зростаючий попит на інтерактивні та доступні рішення, які можуть зробити обслуговування клієнтів більш ефективним і зручним.

Однак впровадження чат-ботів – це не простий процес. Розробка ефективного інструменту з використанням штучного інтелекту вимагає не лише технічної експертизи, але й стратегічного підходу. Нижче розглянемо основні типи чат-ботів, їх особливості та можливості застосування.

1) Кнопкові чат-боти

Fвтоматизовані програми, що взаємодіють із користувачами через попередньо визначені кнопки, а не текстові команди. Вони значно спрощують комунікацію, зменшуючи кількість введених вручну повідомлень та виключаючи ймовірність некоректного розпізнавання тексту. Завдяки інтеграції з бізнес-системами, кнопкові чат-боти активно застосовуються у сфері клієнтської підтримки, електронної комерції, фінансових сервісів та автоматизації процесів.

2) Гібридні чат-боти

автоматизовані системи, що поєднують можливості кнопкових інтерфейсів та розпізнавання природної мови (Natural Language Processing, NLP). Такі боти здатні працювати як зі стандартними сценаріями, так і з відкритими текстовими запитами, забезпечуючи гнучку взаємодію з користувачами. Вони активно застосовуються в електронній комерції,

службах підтримки, фінансових установах та корпоративних рішеннях для підвищення ефективності комунікації.

3) Чат-боти зі штучним інтелектом (ШІ)

Боти зі штучним інтелектом це найсучасніші рішення, здатні розуміти складні запити, обробляти одночасно сотні діалогів і працювати через різні канали комунікації.

4) Багатоканальні екосистеми

Створення багатоканального середовища дозволяє інтегрувати чат-ботів у різні канали комунікації – соціальні мережі, електронну пошту, месенджери. Користувачі можуть почати спілкування в одному каналі, наприклад, у Facebook, і завершити його через телефон або email, без втрати історії діалогу.

5) Голосові чат-боти

Зростання популярності розумних колонок, таких як Siri, Alexa або Google Assistant, відкрило новий напрямок у розвитку голосових ботів. Завдяки технологіям розпізнавання мови вони здатні обробляти складні завдання, використовуючи найбільш природний спосіб взаємодії – голос [3].

Чат-боти це не просто технологія, а динамічний інструмент, який розвивається разом із бізнесом. Створення простого кнопкового бота дозволяє протестувати основні сценарії та поступово перейти до складніших рішень. Інтеграція ШІ та голосових технологій відкриває нові можливості для підвищення ефективності та залучення клієнтів[6].

Вони допомагають автоматизувати рутинні процеси, знижують витрати, забезпечують безперебійний сервіс і зміцнюють лояльність клієнтів. Це стратегічна інвестиція в майбутнє, яка змінює підхід до взаємодії з клієнтами у цифрову епоху.

1.3 Аналіз платформ розробки чат-ботів

Месенджер – це цифровий інструмент для комунікації, який відкриває нові горизонти взаємодії між людьми, дозволяючи обмінюватися текстовими

повідомленнями, мультимедійними файлами, відео та навіть голосовими дзвінками через Інтернет. Месенджери стали невід'ємною частиною сучасного життя, забезпечуючи як індивідуальне спілкування, так і групові чати, створюючи простір для особистих і ділових розмов.

Витоки концепції месенджерів лежать у далекому минулому, коли люди використовували поштових голубів, телеграф і поштові листи для передачі інформації. Сучасна епоха месенджерів почалася з розвитком Інтернету та мобільних технологій, які дозволили людям спілкуватися миттєво, незалежно від відстані.

У 1990-х роках з'явилися перші інтернет-месенджери. Одним із піонерів став AOL Instant Messenger (AIM), який забезпечив спілкування в реальному часі, заклавши основу для подальшого розвитку технології.

На початку 2000-х мобільні месенджери відкрили нову сторінку в історії комунікації. Серед найвпливовіших платформ того періоду виділяється BlackBerry Messenger (BBM), створений виключно для користувачів BlackBerry. Ця платформа запропонувала унікальну можливість обміну миттєвими повідомленнями між користувачами смартфонів.

У 2009 році з’явився WhatsApp, який швидко завоював прихильність завдяки зручному інтерфейсу, функції передачі тексту та медіафайлів через Інтернет. Ще одним значним досягненням став запуск Facebook Messenger у 2011 році, який інтегрувався в екосистему соціальної мережі Facebook, дозволяючи мільйонам користувачів спілкуватися безпосередньо [6].

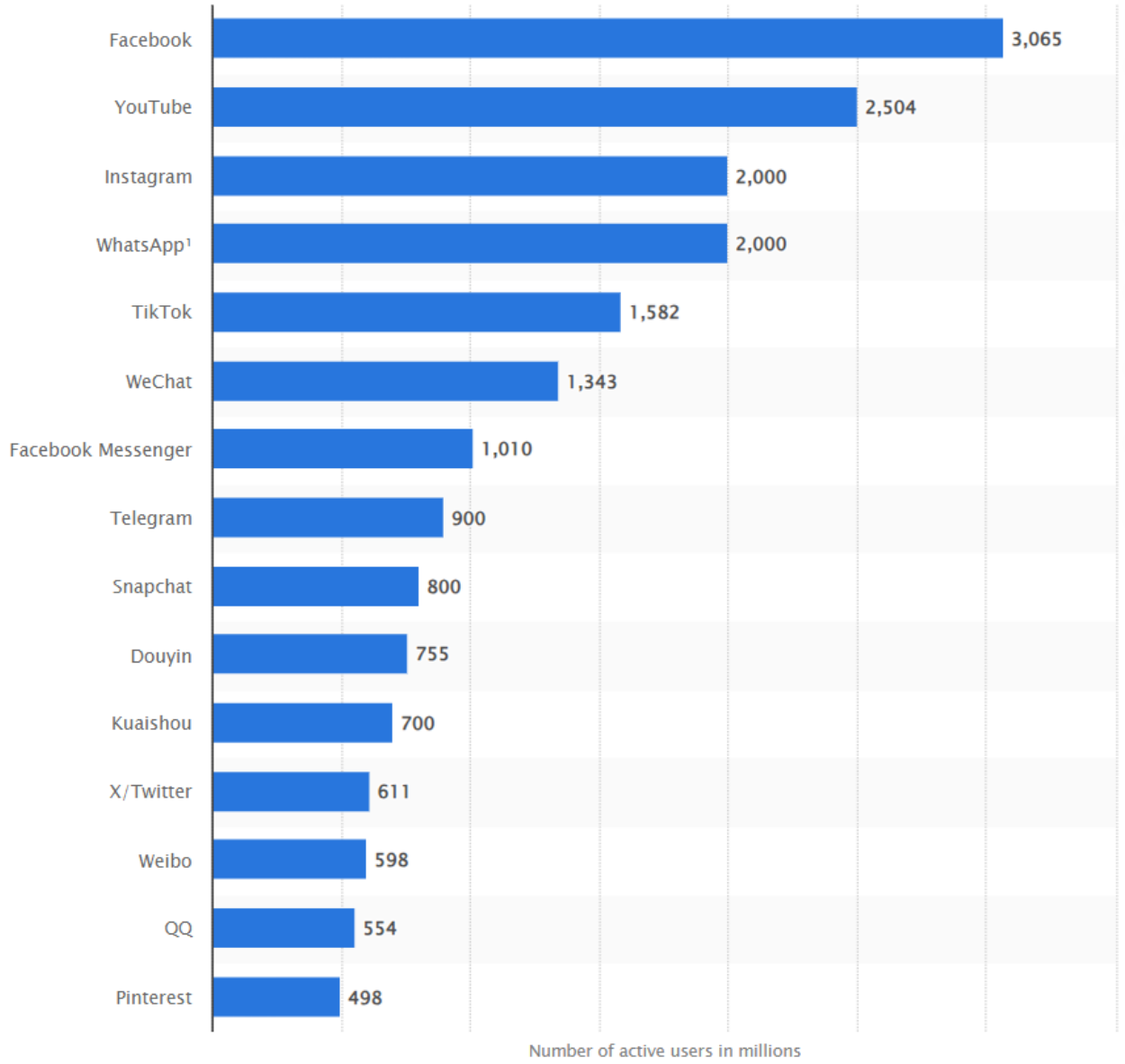


Рисунок 1.1 – Рейтинг месенджерів за даними Statista.com

За даними Statista.com (рис. 1.1), три найпопулярніші месенджери у 2024 році – це WhatsApp, Snapchat і Telegram. Додатково ще розглянемо Viber, який є популярним в Україні [15].

- WhatsApp (рис. 1.2) – беззаперечний лідер завдяки своїй простоті використання, функції наскрізного шифрування повідомлень і здатності працювати навіть при слабкому з'єднанні з Інтернетом. Ця платформа стала синонімом надійної та доступної комунікації.

Рисунок 1.2 – Логотип месенджеру WhatsApp

- Snapchat (рис. 1.3) – месенджер, який пропонує унікальний досвід спілкування через фото та відео, які автоматично зникають після перегляду. Його інтерактивні фільтри та AR-функції перетворили звичайне спілкування на розважальний процес.

Ричунок 1.3 – Логотип месенджеру Snapchat

- Telegram (рис. 1.4) – месенджер відомий своєю швидкістю роботи, захищеними чатами та підтримкою великих груп і каналів. Telegram став вибором для користувачів, які цінують безпеку, масштабованість і гнучкість платформи.

Рисунок 1.4 – Логотип месенджеру Telegram

- Viber (рис. 1.5) – цей месенджер досить популярний у нашому регіоні, виділяється функціями безкоштовних дзвінків, інтеграцією з

локальними сервісами та інтуїтивно зрозумілим інтерфейсом, що робить його доступним для користувачів різного віку.

Рисунок 1.5 – Логотип месенджеру Viber

1.4 Месенджер WhatsApp

Месенджер WhatsApp є одним із найпопулярніших засобів комунікації, що налічує понад два мільярди користувачів у всьому світі. Завдяки своїй широкій аудиторії він стає ефективним інструментом для автоматизації взаємодії між бізнесом та клієнтами.

Для впровадження чат-ботів у WhatsApp розробники використовують офіційні програмні інтерфейси (API), які забезпечують доступ до функціоналу месенджера через серверну взаємодію. Основні способи створення ботів включають:

1. WhatsApp Business API – класична серверна інтеграція, яка потребує розгортання власного сервера для обробки запитів. Доступ до API надається лише офіційно верифікованим бізнесам через партнерів Meta (наприклад, Twilio, 360dialog, WATI).

2. WhatsApp Cloud API – хмарна версія API, що працює безпосередньо через сервери Meta, зменшуючи потребу у власній інфраструктурі.

3. Сторонні сервіси – онлайн-платформи (наприклад, Chatfuel, Landbot, ManyChat), які пропонують графічний інтерфейс для створення ботів без необхідності програмування.

4. Неофіційні бібліотеки – рішення на основі WhatsApp Web (наприклад, Baileys, Venom Bot), які використовуються для створення ботів, проте не відповідають політиці WhatsApp і можуть спричинити блокування акаунта.

Основні функції ботів у WhatsApp включають:

- Автоматизовані відповіді на запити користувачів у режимі 24/7.
- Обробку замовлень, консультацій та інтеграцію з CRM-системами.
- Надсилання транзакційних повідомлень, нагадувань, сповіщень.
- Використання штучного інтелекту для розпізнавання та аналізу тексту (наприклад, Google Dialogflow).
- Впровадження кнопок та інтерактивних меню для спрощення навігації користувачів.

Процес розробки ботів через WhatsApp Business API складається з кількох етапів:

1. Реєстрація бізнес-акаунта у Meta Business Manager.
2. Отримання доступу до API через офіційного постачальника.
3. Інтеграція API зі сторонніми сервісами або власними серверними рішеннями.
4. Розробка логіки бота на основі обробки вхідних повідомлень та автоматичних відповідей [4].

Використання API WhatsApp має певні обмеження. Наприклад, компанії можуть відповідати на вхідні повідомлення користувачів безкоштовно лише протягом 24 годин з моменту отримання запиту. Надсилання масових повідомлень без дозволу користувача заборонене. Крім того, шаблонні повідомлення (template messages) мають бути затверджені WhatsApp перед використанням.

1.5 Месенджер Snapchat

Snapchat – це популярний месенджер, орієнтований на обмін мультимедійним контентом, який використовує механізм самознищення повідомлень після перегляду. Завдяки інтерактивним можливостям, включаючи доповнену реальність (AR), фільтри та лінзи, Snapchat став платформою для брендів, маркетологів та розробників чат-ботів. На відміну від інших месенджерів, Snapchat не пропонує стандартний API для автоматизації текстових чатів, однак забезпечує розширені можливості для інтеграції через Snap Kit та Lens Studio, що дозволяє створювати інтерактивних ботів із використанням AR та штучного інтелекту.

Офіційних текстових ботів у Snapchat немає, оскільки месенджер не підтримує традиційні API для автоматизації чату. Проте існують альтернативні рішення для взаємодії з користувачами через Snapchat:

1. Snap Kit API – набір інструментів, який дозволяє додавати функціонал Snapchat у сторонні додатки, включаючи автентифікацію через Snapchat, обмін контентом та інтеграцію з історіями користувачів.
2. Lens Studio – платформа для створення інтерактивних фільтрів та лінз доповненої реальності, які можуть виконувати функцію ботів через розпізнавання обличчя, голосу або жестів.
3. Інтеграція через сторонні сервіси – використання сторонніх платформ (наприклад, ManyChat, Chatfuel), які дозволяють взаємодіяти з користувачами через історії або рекламу в Snapchat.
4. My AI – офіційний AI-бот від Snapchat на основі OpenAI, який працює у текстових чатах з користувачами та може бути інтегрований у бізнес-рішення.

Основні можливості ботів у Snapchat включають:

- Використання AR-фільтрів для взаємодії з користувачами через доповнену реальність.

- Створення інтерактивних маркетингових кампаній із використанням Snapchat Ads.
- Надсилання персоналізованого контенту через Snap Kit API.
- Використання My AI для автоматизованого обслуговування клієнтів.

Розробка ботів для Snapchat передбачає використання таких інструментів:

1. Snap Kit API – для інтеграції зовнішніх додатків із Snapchat, що дозволяє обмінюватися контентом та отримувати дані про користувачів.
2. Lens Studio – для створення AI-ботів на основі AR-фільтрів, які реагують на міміку, жести або голосові команди.
3. Ad Manager – для запуску інтерактивної реклами, що може залучати користувачів до взаємодії з ботами [5].

Однак використання API Snapchat має певні обмеження. Наприклад, компанія не дозволяє створювати традиційні текстові чат-боти всередині месенджера, а доступ до API можливий лише для партнерських розробників. Крім того, ботів на основі доповненої реальності потрібно адаптувати під політику контенту Snapchat, що може обмежувати їхні можливості.

1.6 Месенджер Viber

Viber – це один із найпопулярніших месенджерів, який забезпечує текстові, голосові та відеозв’язки між користувачами. Завдяки підтримці бізнес-акаунтів і API для інтеграції сторонніх сервісів, Viber став ефективним інструментом для автоматизації взаємодії між компаніями та клієнтами.

Розробка ботів у Viber здійснюється через Viber Bot API, який надає набір інструментів для взаємодії з користувачами. Основні способи створення ботів у Viber включають:

1. Viber Bot API – офіційний інтерфейс для створення ботів, який дозволяє отримувати та надсилати повідомлення, працювати з кнопками, зображеннями та інтерактивними елементами.

2. NoCode платформи – сервіси, такі як Chatfuel, ManyChat та Botsify, дозволяють налаштувати бота без необхідності писати код.

3. Інтеграція з CRM та іншими системами – Viber-боти можуть підключатися до баз даних, штучного інтелекту (наприклад, Google Dialogflow) та сервісів автоматизації (Zapier, Integromat).

Основні можливості Viber-ботів включають:

- Надсилання текстових, графічних та мультимедійних повідомлень.
- Використання інтерактивних кнопок для швидких відповідей та переходів на зовнішні ресурси.
- Створення меню для навігації та структурованого представлення інформації.
- Інтеграцію зі сторонніми API для отримання актуальних даних.
- Автоматизацію обслуговування клієнтів у режимі 24/7.

Створення бота у Viber включає наступні етапи:

1. Реєстрація бота через офіційний портал Viber Admin Panel.

2. Отримання токена доступу, який використовується для інтеграції бота з сервером.

3. Розробка логіки бота на основі Viber Bot API за допомогою мов програмування, таких як Python, Node.js або PHP.

4. Інтеграція з веб-хуками, які дозволяють боту реагувати на вхідні повідомлення.

5. Тестування та запуск бота в публічний доступ.

Попри широкі можливості, боти у Viber мають певні обмеження:

- Взаємодія з ботом можлива лише після підписки користувача на нього.

- Безкоштовне надсилання повідомлень можливе лише протягом 24 годин після останньої взаємодії користувача з ботом.
- Viber вимагає проходження модерації для ботів, що надсилають масові повідомлення.

Viber надає офіційний Viber Bot API, що дозволяє створювати ефективних ботів для автоматизації бізнес-процесів, комунікації з клієнтами та інтеграції з зовнішніми сервісами. Основні переваги Viber-ботів – це підтримка інтерактивних елементів, можливість надсилання мультимедійних повідомлень і зручна інтеграція із CRM та AI-системами. Однак використання ботів у Viber має обмеження, зокрема необхідність підписки користувача та часові рамки для безкоштовного спілкування.

1.7 Месенджер Telegram

Telegram – це популярний месенджер, що відзначається високим рівнем безпеки, швидкодією та розширеним функціоналом для розробників. Однією з ключових можливостей платформи є створення ботів, які можуть автоматизувати процеси, надавати інформаційні послуги, інтегруватися із зовнішніми системами та взаємодіяти з користувачами. Завдяки Telegram Bot API, розробники можуть створювати ботів без необхідності мати окремий сервер для їхнього функціонування.

Боти в Telegram створюються за допомогою Telegram Bot API, який дозволяє взаємодіяти з користувачами через текстові повідомлення, кнопки, команди та мультимедійний контент. Основні способи створення ботів включають:

1. Telegram Bot API – офіційний API, який надає можливість надсилати повідомлення, керувати чатами та обробляти команди користувачів.

2. Telegram Webhooks та Long Polling – методи отримання повідомлень від користувачів, що використовуються для налаштування інтерактивної роботи бота.

3. NoCode платформи – такі сервіси, як ManyBot, Chatfuel, BotFather, які дозволяють створювати ботів без програмування.

4. Інтеграція з нейромережами та базами даних – використання Dialogflow, OpenAI GPT та інших технологій для створення інтелектуальних ботів.

Основні функціональні можливості Telegram-ботів включають:

- Надсилання текстових, графічних та мультимедійних повідомлень.
- Використання клавіатури з кнопками для взаємодії з користувачами.
- Обробку команд (/start, /help, /settings тощо).
- Створення опитувань, квізів та інтерактивних повідомлень.
- Інтеграцію з API сторонніх сервісів, зокрема платіжних систем.

Створення Telegram-бота включає такі кроки:

1. Реєстрація бота через BotFather – спеціальний бот Telegram, який генерує токен доступу для розробки.

2. Розробка логіки бота за допомогою Telegram Bot API.

3. Налаштування серверного середовища – для обробки запитів можна використовувати Long Polling або Webhooks.

4. Інтеграція з базами даних, CRM-системами та зовнішніми API.

5. Запуск та тестування бота перед офіційним використанням.

Обмеження Telegram-ботів включають:

- Відсутність доступу до всіх користувачів месенджера – бот може спілкуватися лише з тими, хто його запустив.
- Обмеження на кількість запитів (до 30 повідомлень на секунду).
- Неможливість додавання бота до груп, якщо він не має відповідних дозволів [7].

Telegram є однією з найзручніших платформ для створення ботів завдяки відкритому Telegram Bot API, простоті налаштування та розширеним функціональним можливостям. Telegram-боти широко використовуються в бізнесі, автоматизації клієнтської підтримки, новинних сервісах, фінансових операціях та інтеграції зі штучним інтелектом. Завдяки можливості обробляти команди, створювати інтерактивні інтерфейси та працювати із зовнішніми API, Telegram залишається однією з найпопулярніших платформ для розробки чат-ботів.

1.8 Висновок до першого розділу

У даному розділі було проведено дослідження предметної області інформаційної систем продажу товарів за допомогою чат-ботів. Розглянуто історію розвитку чат-ботів, їхні ключові функції та переваги, а також класифікацію за рівнем складності та алгоритмічною реалізацією.

Метою дослідження є розробка інформаційної системи продажу товарів за допомогою чат-бота на платформі Telegram, що забезпечить ефективну взаємодію з користувачами, швидке оформлення замовлень та інтеграцію з базою даних товарів.

Для досягнення поставленої мети було визначено та виконано такі завдання:

1. Проведено аналіз наукових та технічних джерел щодо сучасних підходів до розробки чат-ботів.
2. Досліджено основні категорії чат-ботів та їхні можливості в сфері електронної комерції.
3. Проаналізовано популярні месенджери з метою вибору оптимальної платформи для реалізації чат-боту.
4. Оцінено можливості API та інструментів інтеграції чат-ботів із зовнішніми сервісами, такими як CRM-системи, платіжні шлюзи та аналітичні платформи.

Вхідними даними для подальшої розробки є визначені функціональні та нефункціональні вимоги до системи, характеристика цільової аудиторії та особливості інтеграції з обраною платформою. Отримані результати створюють основу для подальшого проєктування та реалізації чат-бота, спрямованого на автоматизацію процесу продажу товарів.

РОЗДІЛ 2 АНАЛІЗ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ ЧАТ-БОТУ

2.1 Вибір платформи для реалізації проекту

Месенджер – це цифровий інструмент для миттєвого обміну повідомленнями, який може бути представлений у вигляді програмного забезпечення, мобільного застосунку або веб-сервісу. Окрім основної функції комунікації, сучасні месенджери часто інтегрують додаткові можливості, такі як голосові та відеодзвінки, підтримка чат-ботів, онлайн-магазини, а також ігровий контент, що робить їх багатофункціональними платформами для взаємодії користувачів .

Зважаючи на популярність месенджерів, варто розглянути три провідні платформи – Facebook Messenger, Snapchat, Telegram та Viber. Щоб оцінити їхні особливості та обрати необхідну платформу, проведемо порівняльний аналіз (табл. 2.1) спираючись на ключові критерії:

- вартість;
- зручність інтерфейсу;
- тип зв’язку;
- функціональність та гнучкість налаштувань;
- основна цільова аудиторія та її потреби.

Таблиця 2.1 – Порівння різних месенджерів

Критерій	Telegram	Viber	Facebook Messenger	Snapchat
Вартість	Повністю безкоштовно	Безкоштовна тільки частина функціоналу	Безкоштовна тільки частина функціоналу	Повністю безкоштовно

Зручність інтерфейсу	Текст, кнопки, голосові повідомлення, веб-застосунки	Текст, кнопки, голосові повідомлення	Текст, кнопки, голосові повідомлення	Текст, кнопки, фото/відео-повідомлення, ефекти та фільтри
Тип зв’язку	Webhook, Polling	Webhook	Webhook	WebRTC, Webhook
Функціональність та гнучкість налаштувань	Листування, опитування, відеодзвінки, передача файлів, веб-застосунки, аудіоповідомлення, шеринґ геолокації, шеринґ номера телефону	Листування, опитування, відеодзвінки, передача файлів, аудіоповідомлення, шеринґ геолокації, шеринґ номера телефону	Листування, опитування, передача файлів, аудіоповідомлення	Листування, зникні фото і відео, відеодзвінки, обмін файлами, фільтри та ефекти, Stories, шеринґ геолокації
Цільова аудиторія	14 – 45 років	28 – 65 років	28 – 65 років	13 – 35 років

На основі проведеного аналізу можна зробити висновок, що Telegram є оптимальним вибором серед сучасних месенджерів завдяки поєднанню розширеного функціоналу, високого рівня безпеки та безкоштовного доступу до всіх можливостей. Відмінною особливістю платформи є її гнучкість, що забезпечується підтримкою веб-застосунків, ботів і можливістю створення

групових чатів із великою кількістю учасників. Використання сучасних технологій передачі даних, зокрема механізмів Webhook та Polling, сприяє стабільності та швидкодії сервісу навіть при високих навантаженнях. Окрім того, месенджер має потужні механізми захисту даних, зокрема наскрізне шифрування в секретних чатах, що підвищує рівень конфіденційності спілкування. Завдяки охопленню широкої вікової аудиторії та підтримці різних платформ Telegram є універсальним рішенням як для особистого, так і для професійного використання.

2.2 Вимоги до програмного забезпечення

Автоматизована система продажу товарів через чат-бот забезпечує швидкий та зручний процес замовлення товарів, а також адміністрування каталогу та замовлень. Формулювання вимог дозволяє визначити відповідність реалізації початковому задуму.

Функціональні вимоги:

1. Основні функції користувача

- Можливість запуску бота в Telegram та взаємодії через текстові команди та кнопки.

- Відображення каталогу товарів із можливістю фільтрації та пошуку за категоріями.

- Оформлення замовлення з вибором товарів, вказанням контактних даних та способу доставки.

- Надсилання користувачеві підтвердження замовлення, статусу виконання та повідомлень про зміну статусу.

- Можливість зв'язку з адміністратором через систему запитань (перегляд питань та надсилання відповідей адміністратором).

2. Функції адміністратора

- Авторизація адміністратора через Telegram ID.

- Додавання, редагування та видалення товарів і категорій через бота.

- Управління замовленнями (перегляд, зміна статусу, видалення замовлень).

- Отримання аналітики щодо продажів (перегляд статистики замовлень).

- Перегляд та модерація запитань користувачів із можливістю відповідати на них.

3. Обробка та збереження даних

- Використання централізованої бази даних для зберігання інформації про товари, користувачів та замовлення.

- Автоматичне оновлення даних у реальному часі при зміні товарів, замовлень чи їхніх статусів.

- Ведення історії змін у статусах замовлень.

4. Система повідомлень

- Автоматичне сповіщення користувачів про підтвердження замовлення, зміну статусу та спеціальні пропозиції.

- Надсилання адміністраторам повідомлень про нові замовлення та запити від користувачів.

Нефункціональні вимоги:

1. Продуктивність

- Система обробляє декілька користувачів одночасно завдяки асинхронній роботі на основі Aiogram.

- Час відповіді бота на запит користувача повинен становить до 2 секунд.

- Швидке виконання запитів до бази даних.

3. Безпека

3.1. Використання Telegram ID для аутентифікації адміністраторів.

4. Масштабованість

- Можливість додавання нових категорії товарів без змін у коді.

- Можливість адаптації під різні бізнес-моделі за рахунок зміни категорій товарів.

5. Юзабіліті

- Інтерфейс побудований за принципом інтуїтивно зрозумілого меню.

- Використання кнопок для швидкої навігації (мінімізовано необхідність введення текстових команд).

2.3 Програмні засоби для реалізації чат-боту

Розвиток цифрових технологій та автоматизації комунікаційних процесів призвів до значного поширення чат-ботів у різних сферах діяльності. Telegram, як один із найпопулярніших месенджерів, надає розробникам широкі можливості для створення ботів, які можуть виконувати як прості, так і складні функції. Завдяки використанню Telegram Bot API, розробники мають змогу інтегрувати ботів із зовнішніми сервісами, керувати повідомленнями та взаємодіяти з користувачами у режимі реального часу [6].

2.3.1 Розробка ботів в Telegram. Telegram Bot API

Telegram Bot API є основним інструментом для розробки ботів, що забезпечує взаємодію між ботом та користувачами через HTTP-запити. Реєстрація нового бота відбувається через спеціального бота BotFather, який генерує унікальний токен доступу до API. Цей токен використовується для автентифікації запитів і є ключовим елементом безпеки [10].

Основні методи API включають:

- sendMessage – надсилання текстових повідомлень;
- sendPhoto, sendDocument – передача мультимедійних файлів;
- setWebhook – встановлення вебхука для автоматичної обробки повідомлень;
- getUpdates – отримання нових повідомлень у режимі опитування сервера.

Telegram підтримує два основні методи отримання повідомлень:

1. Polling – бот регулярно надсилає запит getUpdates до серверів Telegram, отримуючи нові повідомлення. Цей метод простий у реалізації, але

менш ефективний через затримки в отриманні інформації та навантаження на сервер.

2. Webhook – сервер Telegram автоматично надсилає повідомлення на вказаний URL-адрес сервера бота. Використання вебхуків є більш продуктивним способом взаємодії, оскільки повідомлення обробляються у реальному часі.

Для обробки команд та повідомлень застосовуються спеціалізовані бібліотеки, такі як python-telegram-bot або aiogram у середовищі Python, telegraf у Node.js та TeleBot для мови програмування C# [13].

Telegram підтримує кілька видів клавіатур, що дозволяють спростити взаємодію користувачів із ботом:

- ReplyKeyboardMarkup – клавіатура, що з'являється безпосередньо у вікні чату.
- InlineKeyboardMarkup – кнопки, вбудовані в повідомлення, що можуть викликати подальші дії.
- ForceReply – змушує користувача відповісти на конкретне повідомлення.

Застосування таких елементів забезпечує кращу інтерактивність та покращує користувацький досвід.

Захист бота від зловмисних атак є важливим аспектом його розробки. Основні заходи безпеки включають:

- приховування токенів API за допомогою змінних середовища;
- фільтрацію вхідних запитів для запобігання атакам типу SQL-ін'єкцій;
- обмеження доступу до певних функцій лише для авторизованих користувачів;
- шифрування конфіденційних даних під час їх збереження та передавання.

Для збереження стану користувачів та обробки запитів застосовуються різні бази даних:

- SQLite – для невеликих проєктів;
- PostgreSQL, MySQL – для масштабних систем;
- Redis – для кешування тимчасових даних.

Окрім баз даних, бот може інтегруватися із зовнішніми API, такими як OpenAI для генерації текстів, Google Sheets для зберігання інформації або платіжні сервіси (Stripe, LiqPay) для проведення транзакцій.

Розгортання чат-бота на сервері є заключним етапом розробки. Основні платформи для розгортання включають:

- Heroku, Railway.app – для безкоштовного розгортання невеликих ботів;
- VPS (AWS, DigitalOcean, Hetzner) – для комерційних проєктів;
- Serverless-платформи (AWS Lambda, Google Cloud Functions) – для економії ресурсів.

2.3.2 Вибір бібліотеки для оброки команд та повідомлень

Завдяки широкому вибору бібліотек для різних мов програмування, розробники можуть створювати ботів із мінімальними зусиллями, використовуючи вже готові рішення для обробки повідомлень, команд, інтеграції з API та базами даних.

Серед найпопулярніших бібліотек для розробки Telegram-ботів можна виділити:

- aiogram (Python) – асинхронний фреймворк для ефективної обробки запитів.
- Telegraf (Node.js) – подієво-орієнтований фреймворк для розробки ботів у середовищі JavaScript/TypeScript.
- TeleBot (C#) – бібліотека для створення ботів у .NET із підтримкою інтерактивних елементів.

Розглянемо кожен з них більш детально

1. aiogram

Фреймворк aiogram побудований на модульній архітектурі, що включає кілька ключових компонентів:

1) Bot API – забезпечує взаємодію з Telegram, включаючи отримання та надсилання повідомлень, керування чатами та користувачами.

2) Dispatcher – основний механізм, що відповідає за обробку команд, повідомлень та інших подій у боті.

3) Middleware – проміжний шар, який дозволяє перехоплювати та обробляти дані перед передачею у внутрішню логіку бота.

4) Filters – система фільтрації для визначення типу повідомлень (наприклад, текст, фото, відео) та обробки відповідних команд.

5) Keyboards – підтримка інтерактивних елементів, таких як кнопкові меню та inline-клавіатури, що спрощує взаємодію з користувачем.

Основні можливості aiogram включають:

- Підтримку текстових та мультимедійних повідомлень.
- Обробку команд користувачів через кнопку меню або текстовий ввід.
- Використання інтерактивних клавіатур, що дозволяє створювати зручний інтерфейс.
- Інтеграцію з базами даних, CRM-системами та іншими API.
- Масштабованість завдяки підтримці асинхронних запитів.

Для розробки чат-бота за допомогою aiogram необхідно виконати кілька ключових етапів:

1) Налаштування оточення – встановлення бібліотеки aiogram у Python та підключення API Telegram.

2) Реєстрація бота – створення облікового запису бота через офіційний сервіс BotFather, який видає унікальний токен для взаємодії з Telegram API.

3) Розробка логіки – налаштування обробки команд, повідомлень, інтерактивних кнопок та інших елементів взаємодії.

4) Інтеграція з базами даних – збереження інформації про користувачів та їхні запити, що дозволяє створювати персоналізовані відповіді.

5) Розгортання та тестування – запуск бота у локальному середовищі або на хмарному сервері для постійного доступу.

Попри значні переваги, aiogram має певні обмеження:

- Вимагає асинхронного середовища, що може ускладнити інтеграцію зі сторонніми синхронними бібліотеками.
- Підтримує лише Telegram, тому не підходить для створення мультиплатформних ботів.
- Потребує серверного хостингу, оскільки для роботи необхідне постійне з'єднання із Telegram API [12].

2. Telegraf

Фреймворк Telegraf використовує подієво-орієнтовану модель взаємодії з Telegram API. Основні компоненти Telegraf включають:

1) Bot API – забезпечує взаємодію з Telegram, включаючи отримання та надсилання повідомлень.

2) Middleware – проміжний рівень обробки даних, що дозволяє фільтрувати, змінювати або блокувати повідомлення перед їхньою основною обробкою.

3) Командні хендлери – визначення команд (/start, /help, /settings) та відповідних реакцій бота.

4) Інтерактивні клавіатури – підтримка кнопкових меню та inline-клавіатур для взаємодії з користувачем.

5) Фільтрація повідомлень – обробка текстових повідомлень, фото, відео, документів, стікерів тощо.

Основні можливості Telegraf включають:

- Обробку текстових та мультимедійних повідомлень.
- Використання callback-кнопок для створення інтерактивного інтерфейсу.

- Інтеграцію з базами даних та зовнішніми API.
- Розширення функціоналу через middleware та модульну архітектуру.
- Автоматизацію процесів завдяки асинхронній обробці запитів.

Створення чат-бота за допомогою Telegraf передбачає кілька основних етапів:

1) Налаштування середовища – встановлення Node.js та підключення бібліотеки Telegraf.

2) Реєстрація бота через BotFather – отримання унікального токена для доступу до Telegram API.

3) Розробка логіки бота – налаштування команд, обробки повідомлень та кнопок.

4) Інтеграція з базами даних та зовнішніми сервісами – збереження історії взаємодій та персоналізація відповідей.

5) Розгортання на сервері – запуск бота у хмарному середовищі для забезпечення його безперервної роботи.

Попри значні переваги, Telegraf має деякі обмеження:

- Працює лише з Telegram, тому не підходить для мультиплатформних ботів.
- Вимагає хостингу або серверного розгортання, оскільки не підтримує автономний режим роботи.
- Для обробки великої кількості користувачів потребує масштабованої серверної інфраструктури [9].

3. TeleBot

Бібліотека TeleBot працює за принципом обробки вхідних подій та команд через подієво-орієнтовану модель. Основні компоненти TeleBot включають:

1) Telegram Bot API – забезпечує взаємодію бота з користувачами через відправку та отримання повідомлень.

2) Командні хендлери – дозволяють обробляти текстові команди (/start, /help, /menu).

3) Подієвий механізм обробки повідомлень – бот реагує на введені користувачем запити та автоматично виконує відповідні дії.

4) Інтерактивні клавіатури – можливість створення меню для вибору дій, що значно покращує користувацький досвід.

5) Інтеграція з базами даних – дозволяє зберігати інформацію про користувачів, історію повідомлень та персоналізовані налаштування.

Основні можливості TeleBot включають:

- Обробку текстових повідомлень, фото, відео, голосових повідомлень.
- Використання callback-кнопок для створення інтерактивного інтерфейсу.
- Роботу з inline-клавіатурами для швидкого вибору опцій.
- Інтеграцію з базами даних (SQL Server, MySQL, SQLite).
- Автоматизацію завдань за допомогою таймерів та фонового виконання процесів.

Процес створення чат-бота за допомогою TeleBot у середовищі C# включає наступні етапи:

1) Реєстрація бота через BotFather у Telegram та отримання токена доступу.

2) Налаштування середовища – встановлення бібліотеки TeleBot через NuGet.

3) Розробка логіки бота – визначення команд, обробка повідомлень, створення інтерактивного меню.

4) Інтеграція з базами даних – для збереження та аналізу даних користувачів.

5) Розгортання на сервері – запуск бота у середовищі Windows Server, Azure або Docker для безперервної роботи.

Обмеження TeleBot:

- Працює лише з Telegram, що не дозволяє використовувати його для мультиплатформних рішень.
- Вимагає серверного хостингу, оскільки для роботи необхідне постійне з’єднання із Telegram API.
- Для обробки великої кількості користувачів потребує оптимізації ресурсів [8].

Для наглядного представлення в таблиці 2.2 представлено порівняння трьох бібліотек для створення Telegram-ботів: aiogram, Telegraf та TeleBot.

Таблиця 2.2 – Порівняння бібліотек реалізації Telegram-ботів

Характеристика	aiogram	Telegraf	TeleBot
Мова програмування	Python	Node.js	C#
Підтримка асинхронності	Так	Так	Так
Простота використання	Середня	Висока	Середня
Масштабованість	Висока	Висока	Середня
Інтерактивні клавіатури	Так	Так	Так
Підтримка мультимедіа	Так	Так	Так
Інтеграція з базами даних	Так	Так	Так

Вимоги до серверного хостингу	Так	Так	Так
Сфера застосування	Автоматизація, AI, клієнтська підтримка	Веб-додатки, інтеграція з сервісами	Бізнес-рішення, корпоративні системи

Кожна з цих бібліотек пропонує зручний API, що дозволяє швидко реалізувати ключові функції бота, включаючи обробку команд, роботу з кнопками, збереження даних користувачів та інтеграцію з іншими сервісами.

Для реалізації проекту було обрано aiogram та мову програмування Python.

2.4 Висновок до другого розділу

У другому розділі було проведено комплексний аналіз засобів для реалізації чат-боту, що включає вибір платформи, визначення функціональних та нефункціональних вимог до програмного забезпечення, а також огляд бібліотек для його розробки. Було детально розглянуто особливості та можливості популярних месенджерів, зокрема Telegram, Viber, Facebook Messenger і Snapchat, з акцентом на їхню зручність, функціональність, гнучкість налаштувань та підтримку інтеграцій.

За результатами порівняльного аналізу, Telegram було обрано як оптимальну платформу для розробки чат-боту завдяки його розширеному функціоналу, високому рівню безпеки, підтримці Webhook та Polling, а також можливості інтеграції з різними зовнішніми сервісами. Важливим фактором вибору також стала доступність API та активна спільнота розробників, що дозволяє легко адаптувати бот під конкретні бізнес-завдання.

Було сформульовано вимоги до програмного забезпечення, включаючи функціональні вимоги (основні функції для користувачів і адміністраторів, механізми обробки запитів, збереження та обробка даних) та нефункціональні вимоги (продуктивність, безпека, масштабованість та юзабіліті). Основна увага була приділена ефективному управлінню замовленнями, автоматизації взаємодії з клієнтами та інтеграції чат-бота з базою даних.

Окрім того, було проведено детальний огляд програмних засобів для розробки Telegram-ботів, включаючи бібліотеки aiogram (Python), Telegraf (Node.js) та TeleBot (C#). Аналіз показав, що aiogram є найкращим вибором завдяки його асинхронній архітектурі, зручному API, можливості гнучкої інтеграції з базами даних та зовнішніми сервісами, а також ефективному управлінню великою кількістю запитів у реальному часі.

Таким чином, проведений аналіз програмних засобів, визначення вимог до системи та порівняння бібліотек дозволили обґрунтовано обрати Telegram як основну платформу для розробки чат-бота, а aiogram – як ключову бібліотеку для його реалізації. Ці рішення забезпечують оптимальну продуктивність, масштабованість та безпеку, що робить їх ефективним вибором для автоматизації комунікацій у бізнес-середовищі.

РОЗДІЛ 3. РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ПРОДАЖУ ТОВАРІВ З ДОПОМОГОЮ ЧАТ-БОТУ

3.1 Актуальність задачі

Розробки автоматизованої системи продажу товарів з допомогою чат-боту обумовлена сучасними тенденціями у сфері електронної комерції та стрімким розвитком цифрових технологій. Сьогодні телеграм-боти набувають все більшого поширення завдяки своїй доступності, зручності використання та інтеграції з різними платформами. Вони стають ефективним інструментом автоматизації бізнес-процесів, що дозволяє компаніям значно зменшити витрати на обслуговування клієнтів, оптимізувати операції та забезпечити цілодобовий доступ до своїх товарів і послуг.

Попит на швидкі та зручні способи здійснення покупок зростає разом із популярністю мобільних додатків та месенджерів. Телеграм, як одна з найбільш поширених платформ для комунікації, надає можливість створення ботів, які можуть виконувати функції повноцінного магазину: надавати інформацію про товари, приймати замовлення, здійснювати оплату та навіть координувати доставку. Це значно підвищує зручність як для покупців, так і для бізнесу, дозволяючи уникнути необхідності створення окремого веб-сайту чи мобільного додатку.

Особливо актуальним впровадження такої системи є для малого та середнього бізнесу, який прагне знайти недорогі, але ефективні способи виходу на ринок. Використання телеграм-ботів дозволяє швидко адаптуватися до змін ринкового середовища, тестувати нові стратегії продажу та аналізувати дані про взаємодію клієнтів із системою. Крім того, бот може бути налаштований для персоналізованого підходу до кожного користувача, що сприяє підвищенню лояльності клієнтів та зростанню продажів.

З урахуванням сучасних викликів, таких як пандемія COVID-19 та війна в Україні, які стимулювали розвиток онлайн-торгівлі, автоматизовані системи

продажу стали не лише зручністю, а й необхідністю для багатьох бізнесів. Телеграм-боти пропонують просте і водночас потужне рішення для організації продажу в умовах обмеженого доступу до фізичних магазинів, забезпечуючи можливість безконтактного обслуговування клієнтів.

Отже, розробка автоматизованої системи продажу товарів за допомогою телеграм-боту є актуальною не лише через її економічну ефективність, але й через її здатність забезпечити сучасний рівень обслуговування клієнтів, гнучкість у роботі та зручність для користувачів. Така система сприяє підвищенню конкурентоспроможності бізнесу та створює нові можливості для розвитку в умовах цифрової економіки.

Мета розробки інформаційної системи продажу товарів за допомогою телеграм-боту полягає у створенні зручного, доступного та ефективного інструменту для автоматизації процесів онлайн-продажів, який забезпечить швидку взаємодію з клієнтами, полегшить управління замовленнями та сприятиме розвитку бізнесу. Основною ціллю є впровадження інноваційного рішення, яке поєднує функціонал традиційних інтернет-магазинів із можливостями месенджера Telegram, створюючи таким чином інтуїтивно зрозумілий інтерфейс для кінцевих користувачів та оптимізуючи процес продажу з боку бізнесу. Реалізація цього проекту має забезпечити доступність для широкої аудиторії, високу швидкість обробки даних, можливість персоналізації взаємодії з клієнтами та легку інтеграцію з існуючими системами обліку чи платіжними сервісами.

Завдання, які повинен поставити перед собою розробник, охоплюють декілька ключових напрямів. Перш за все, необхідно розробити зручний та зрозумілий інтерфейс взаємодії користувача з ботом. Це включає реалізацію можливості перегляду каталогу товарів, пошуку за ключовими словами, фільтрації та сортування продукції за різними критеріями. Важливим є також забезпечення функції оформлення замовлення, що передбачає інтеграцію з платіжними системами для проведення онлайн-оплат та можливість вибору способу доставки.

Додатково розробник має створити систему обробки замовлень, яка буде автоматично фіксувати дані клієнтів, замовлені товари та статус виконання замовлення. Для цього потрібно реалізувати серверну частину з базою даних, що дозволить зберігати, обробляти та аналізувати інформацію. Не менш важливим завданням є розробка системи сповіщень, яка забезпечить оперативне інформування користувачів про підтвердження замовлення, статус доставки чи спеціальні пропозиції.

Також варто передбачити можливість адміністрування бота, що включає управління каталогом товарів, оновлення цін та інформації, а також моніторинг статистики продажів. Розробник повинен забезпечити високий рівень безпеки системи, що охоплює захист особистих даних користувачів, безпечні транзакції та запобігання несанкціонованому доступу до адміністративного інтерфейсу.

3.2 Структура проекту

Структура чат-боту, яка складається з бази даних і додатку (що поділяється на частини для користувача та адміністратора (рис. 3.1). Кожен із цих компонентів виконує окремі завдання, які в сукупності формують цілісну систему, здатну задовольнити як потреби користувачів, так і вимоги бізнесу.

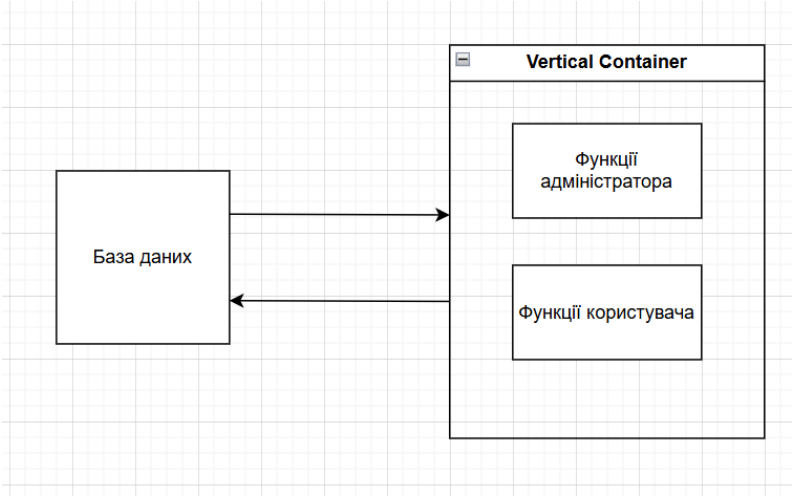


Рисунок 3.1 – Структура додатку

База даних є центральним елементом, який забезпечує зберігання, організацію та обробку всіх необхідних даних. Вона включає таблиці, що містять інформацію про товари, їх характеристики, наявність на складі, ціни, замовлення, а також дані про користувачів і адміністраторів. Ключовим завданням бази даних є забезпечення швидкого доступу до інформації, її актуальність і цілісність. Наприклад, при запиті користувача на перегляд певного товару, бот звертається до бази даних для отримання необхідної інформації. Окрім того, база даних зберігає історію взаємодій користувачів із ботом, що може бути використано для аналізу поведінки клієнтів і покращення якості обслуговування.

Додаток поділяється на два основні сегменти: користувацький і адміністративний.

Користувацька частина додатка відповідає за взаємодію з кінцевими користувачами. Ця частина включає інтерфейс для перегляду каталогу товарів, пошуку за категоріями або ключовими словами, вибору товарів, оформлення замовлення та оплати. Інтерфейс має бути максимально інтуїтивним і дружнім до користувача, щоб забезпечити комфортну навігацію та швидкий доступ до потрібних функцій. Чат-бот у цьому випадку виступає посередником, який спілкується з користувачем у текстовому форматі, надсилаючи запити до бази даних і повертаючи відповідні результати.

Адміністративна частина додатка створена для управління системою і є інструментом для адміністраторів. Вона включає функціонал для управління товарним каталогом (додавання нових товарів, оновлення інформації, видалення товарів), моніторингу замовлень, управління статусами їх виконання, а також перегляду аналітичних даних, таких як статистика продажів або поведінкові звіти користувачів. Адміністративна частина повинна мати високий рівень безпеки, включаючи функції автентифікації та авторизації, щоб уникнути несанкціонованого доступу до управлінських функцій.

3.3 Розробка системи

Компонент config.py є ключовою частиною програми для роботи з телеграм-ботом (рис. 3.2), забезпечуючи ініціалізацію важливих параметрів та завантаження конфігурацій із зовнішнього файлу .env. Він відповідає за отримання секретних даних, таких як токен бота, які використовуються для взаємодії з Telegram API [11], а також задає структуру для зберігання ідентифікаторів адміністраторів, що мають доступ до спеціальних функцій бота.

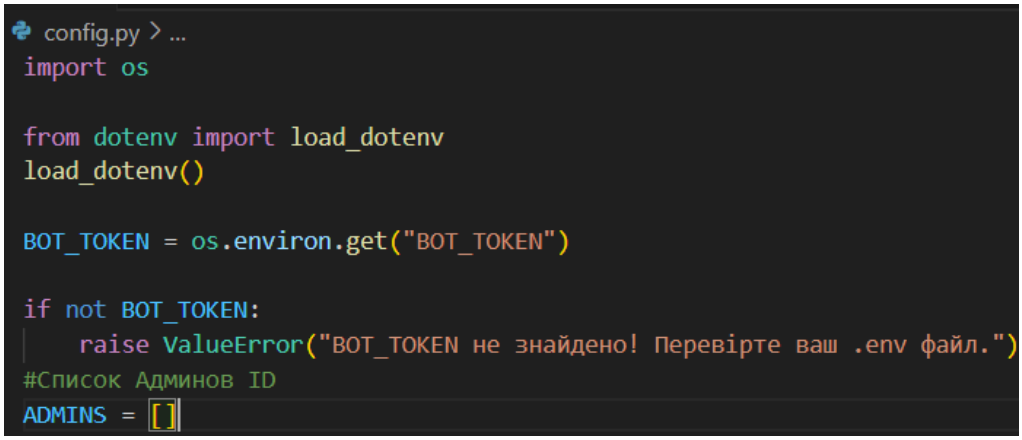


Рисунок 3.2 – Компонент config.py

Для роботи компонента використовується бібліотека os, яка є стандартною частиною Python і забезпечує взаємодію з операційною системою [14]. У цьому випадку вона використовується для доступу до змінних середовища. Функція os.environ.get() дозволяє отримати значення змінної середовища за її назвою. Це зручно для зберігання конфіденційних даних, таких як токени чи ключі API, оскільки вони не включаються безпосередньо в код, а зберігаються в зовнішніх файлах чи налаштуваннях операційної системи. Такий підхід підвищує безпеку програми, оскільки зменшує ризик випадкового витоку секретної інформації.

Ще однією важливою бібліотекою є python-dotenv, яка надає зручний спосіб завантаження змінних середовища з файлу .env. Файл .env зазвичай

використовується для зберігання конфігураційних параметрів у вигляді пар "ключ-значення". Наприклад, у цьому файлі можна вказати токен бота (BOT_TOKEN=ваш_токен). Бібліотека dotenv дозволяє легко зчитувати цей файл і автоматично додавати вказані в ньому змінні до середовища. Функція load_dotenv() зчитує файл .env і додає його вміст до змінних середовища, що дозволяє доступ до них через модуль os.

У наведеному коді реалізовано перевірку наявності токена бота. Якщо змінна BOT_TOKEN не знайдена в середовищі, програма виведе помилку з повідомленням про те, що токен не знайдено, і попросить перевірити файл .env. Такий підхід дозволяє уникнути ситуацій, коли програма запускається без необхідних даних, що могло б призвести до некоректної роботи або краху програми.

Окремий елемент коду — список ADMINS, який призначений для зберігання ідентифікаторів користувачів-адміністраторів. Цей список може бути використаний для надання спеціальних привілеїв певним користувачам бота, таким як доступ до розширеного функціоналу, моніторинг або адміністрування. Поки що список порожній, але він може бути заповнений ідентифікаторами Telegram ID адміністраторів.

Наступний компонент is_admin.py Даний компонент є реалізацією фільтра для телеграм-бота, створеного на основі бібліотеки Aiogram, яка є потужним інструментом для створення асинхронних телеграм-ботів. Цей фільтр забезпечує перевірку, чи належить користувач до списку адміністраторів, перед виконанням певних дій або обробкою повідомлення. Завдяки цьому компоненту можна легко обмежити доступ до адміністративного функціоналу бота, надаючи його лише певній групі користувачів.

Клас IsAdmin наслідує BoundFilter, який є базовим класом у бібліотеці Aiogram для створення власних фільтрів. Фільтри в Aiogram дозволяють визначати логіку перевірки повідомлень або інших подій, забезпечуючи зручний механізм управління обробниками. У цьому випадку IsAdmin реалізує

спеціальний фільтр для перевірки, чи належить ID користувача, який надіслав повідомлення, до списку адміністраторів, визначеного в конфігураційному файлі.

Ключовим методом класу є check (рис. 3.3), який виконується під час перевірки повідомлення. Цей метод отримує об'єкт повідомлення Message як аргумент і перевіряє, чи знаходиться ідентифікатор користувача (message.from_user.id) у списку адміністраторів (ADMINS). Якщо ID користувача є в списку, метод повертає True, що дозволяє обробнику виконати відповідну дію. Якщо ID не знайдено, метод повертає False, і обробник не виконується. Такий підхід забезпечує простий спосіб розмежування доступу до функцій бота.

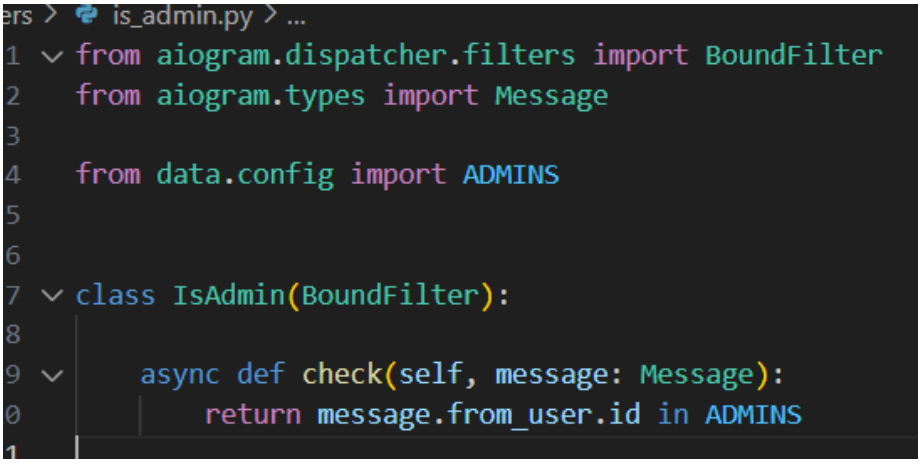


Рисунок 3.3 – Компонент is_admin.py

Список ADMINS, на який посилається компонент, зберігається у конфігураційному файлі data.config і містить ID користувачів, які мають адміністративні права. Це забезпечує централізоване управління правами доступу, дозволяючи легко додавати або видаляти адміністраторів без необхідності змінювати код самого бота.

Компонент IsUser (рис. 3.3) має аналогічну структуру до IsAdmin, але з протилежною логікою перевірки: він визначає, чи користувач не є адміністратором. Він також наслідує клас BoundFilter бібліотеки Aiogram і

перевіряє, чи ID користувача відсутній у списку ADMINS. Це дозволяє розмежувати функціонал бота між звичайними користувачами та адміністраторами, забезпечуючи просте і централізоване управління доступом через конфігураційний файл.

```python
from aiogram.dispatcher.filters import BoundFilter
from aiogram.types import Message

from data.config import ADMINS


class IsUser(BoundFilter):

    async def check(self, message: Message):
        return message.from_user.id not in ADMINS
```

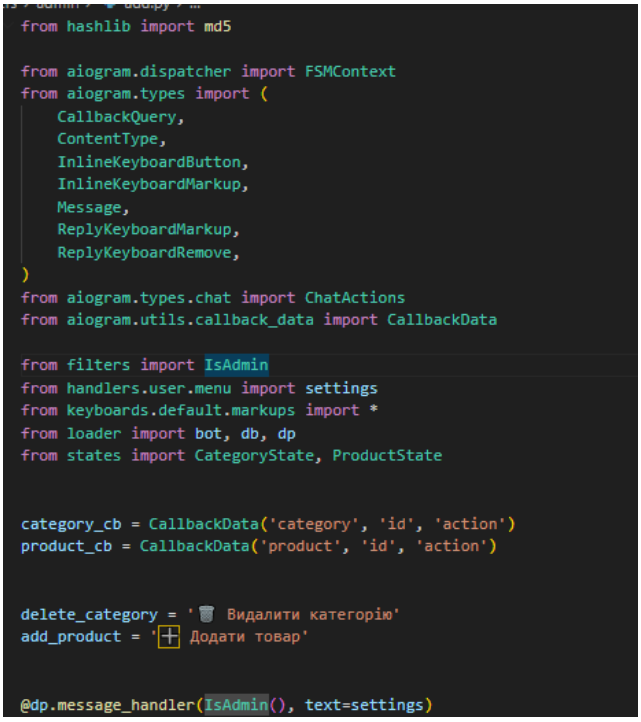
Рисунок 3.4 – Компонент IsUser

Add.py компонент є частиною телеграм-бота, що реалізує функціонал для адміністрування категорій товарів. Він використовує бібліотеку Aiogram для обробки повідомлень, а також містить інтеграцію з базою даних для роботи з категоріями. Код демонструє створення інтерактивного інтерфейсу за допомогою інлайн-клавіатур та використання станів для управління діями користувача.

На початку імпортуються необхідні модулі та бібліотеки (рис. 3.5). Зокрема, hashlib для роботи з хешуванням, модулі з Aiogram для обробки повідомлень, а також кастомні фільтри, обробники та машини станів (FSM). У компоненті використовуються попередньо визначені фільтри (IsAdmin) для перевірки доступу, а також дані з бази (db) для роботи з категоріями.

CallbackData використовується для створення унікальних обробників телеграм-кнопок. Дві основні структури category_cb і product_cb містять поля для передачі параметрів, таких як ID категорії чи товару та дії, які необхідно виконати. Це спрощує обробку подій, пов'язаних із натисканням кнопок.

Функція process_settings обробляє повідомлення, надіслані адміністраторами, якщо текст повідомлення відповідає константі settings. У функції створюється інлайн-клавіатура, що відображає список категорій, завантажених із бази даних через запит db.fetchall('SELECT * FROM categories'). Для кожної категорії додається кнопка, що відповідає її назві, і прикріплюється callback-запит для перегляду деталей (action='view'). Додатково створюється кнопка для додавання нової категорії.



```
from hashlib import md5

from aiogram.dispatcher import FSMContext
from aiogram.types import (
    CallbackQuery,
    ContentType,
    InlineKeyboardButton,
    InlineKeyboardMarkup,
    Message,
    ReplyKeyboardMarkup,
    ReplyKeyboardRemove,
)
from aiogram.types.chat import ChatActions
from aiogram.utils.callback_data import CallbackData

from filters import IsAdmin
from handlers.user.menu import settings
from keyboards.default.markups import *
from loader import bot, db, dp
from states import CategoryState, ProductState


category_cb = CallbackData('category', 'id', 'action')
product_cb = CallbackData('product', 'id', 'action')


delete_category = '🗑 Видалити категорію'
add_product = '➕ Додати товар'


@dp.message_handler(IsAdmin(), text=settings)
```

Рисунок 3.5 – Компонент Add.py

У кінці функції бот надсилає повідомлення з текстом "Налаштування категорії:" та прикріплює створену клавіатуру. Це дозволяє адміністраторам легко переглядати категорії та виконувати базові операції, такі як додавання нових категорій.

Наступна функція add_category_callback_handler обробляє подію натискання кнопки "Додати категорію" в інтерфейсі бота, яка доступна лише адміністраторам завдяки використанню фільтру IsAdmin. Коли адміністратор натискає цю кнопку, бот видаляє попереднє повідомлення, щоб очистити чат

від зайвої інформації та уникнути плутанини. Далі бот надсилає повідомлення із запитом "Назва категорії?" для введення адміністратора. Після цього функція встановлює стан CategoryState.title, який сигналізує машині станів, що бот очікує від користувача введення тексту, який буде інтерпретований як назва нової категорії.

```python
@dp.callback_query_handler(IsAdmin(), text='add_category')
async def add_category_callback_handler(query: CallbackQuery):
    await query.message.delete()
    await query.message.answer('Назва категорії?')
    await CategoryState.title.set()
```

Рисунок 3.6 – Функція add_category_callback_handler

Функція set_category_title_handler (рис. 3.7) відповідає за обробку текстового повідомлення, яке вводить адміністратор як назву нової категорії. Вона перевіряє машину станів, щоб переконатися, що бот знаходиться в режимі очікування назви категорії. Після отримання тексту функція генерує унікальний ідентифікатор для категорії, використовуючи алгоритм MD5, який хешує назву категорії в унікальне значення. Цей ідентифікатор разом із назвою зберігається в базі даних через SQL-запит INSERT INTO categories. Після успішного збереження даних машина станів завершується, а функція викликає process_settings для оновлення адміністративного меню.

```python
@dp.message_handler(IsAdmin(), state=CategoryState.title)
async def set_category_title_handler(message: Message, state: FSMContext):

    category = message.text
    idx = md5(category.encode('utf-8')).hexdigest()
    db.query('INSERT INTO categories VALUES (?, ?)', (idx, category))

    await state.finish()
    await process_settings(message)
```

Рисунок 3.7 – Функція set_category_title_handler

Машина станів (Finite State Machine, FSM) — це концепція в програмуванні, яка використовується для моделювання процесів, які можуть перебувати в різних станах і переходити між ними залежно від певних умов або подій. У контексті телеграм-ботів, зокрема з бібліотекою Aiogram, FSM дозволяє ефективно організовувати логіку роботи бота, коли необхідно обробляти послідовність дій користувача.

FSM виконує роль контролера, який зберігає поточний стан користувача та визначає, яку дію виконувати далі. Наприклад, бот може знаходитися в стані очікування введення тексту, вибору категорії або підтвердження дії. Завдяки FSM бот "розуміє", що саме робить користувач у конкретний момент, і реагує на його дії відповідно до поточного стану.

Основні етапи роботи машини станів в даному коді:

- Стан (State): конкретний етап, на якому знаходиться користувач під час взаємодії з ботом. У даному компоненті, наприклад, стан CategoryState.title означає, що бот очікує від адміністратора введення назви нової категорії.
- Перехід між станами (State Transition): Це процес переходу бота з одного стану до іншого залежно від дій користувача. Наприклад, після введення тексту користувачем бот може перейти від стану очікування введення категорії (CategoryState.title) до стану завершення (state.finish()).
- Контекст стану (FSMContext): Aiogram використовує цей контекст для зберігання тимчасових даних, які стосуються конкретного стану. У цьому контексті можна зберігати введені дані, наприклад, назву категорії, щоб використовувати їх у наступних діях.

Для хешування, що зазначалося вище, використовується алгоритм MD5 (Message-Digest Algorithm 5) — це криптографічна хеш-функція, яка використовується для отримання фіксованого розміру хешу (128-бітного значення) із вхідних даних будь-якої довжини. MD5 був створений у 1991 році Роном Рівестом і широко застосовується для перевірки цілісності даних,

створення унікальних ідентифікаторів та інших цілей, де потрібне компактне уявлення великих даних.

Наступні дві функції додають функціонал для перегляду та управління товарами в конкретній категорії в телеграм-боті. Вони забезпечують адміністраторам можливість переглядати список товарів, пов'язаних із вибраною категорією, видаляти товари або керувати категорією через зручний інтерфейс.

Функція category_callback_handler (рис. 3.8) обробляє запит, коли адміністратор натискає кнопку для перегляду категорії. Завдяки використанню фільтру IsAdmin, вона переконується, що запит надходить від адміністратора. Вхідні дані обробляються через CallbackData, що дозволяє отримати ідентифікатор категорії (category_idx) із callback-запиту. Після цього функція виконує SQL-запит, щоб отримати всі товари, пов'язані з цією категорією, з бази даних. Дані про товари витягуються через вкладений запит, який знаходить товари за тегом, що відповідає назві категорії. Далі функція видаляє попереднє повідомлення адміністратора (меню категорій), надсилає коротке сповіщення про те, що всі товари цієї категорії будуть відображені, і зберігає ідентифікатор категорії в контексті стану (FSMContext). Це дає змогу функціоналу залишатися послідовним і використовувати ідентифікатор у подальших операціях. Наприкінці викликається функція show_products, яка відображає список товарів.

```python
@dp.callback_query_handler(IsAdmin(), category_cb.filter(action='view'))
async def category_callback_handler(query: CallbackQuery, callback_data: dict,
                                    state: FSMContext):
    category_idx = callback_data['id']

    products = db.fetchall('''SELECT * FROM products product
    WHERE product.tag = (SELECT title FROM categories WHERE idx=?)''',
                           (category_idx,))

    await query.message.delete()
    await query.answer('Все добавленные товары в эту категорию.')
    await state.update_data(category_index=category_idx)
    await show_products(query.message, products, category_idx)
```

Рисунок 3.8 – Функція category_callback_handler

Функція show_products (рис. 3.9) займається безпосереднім відображенням товарів у чаті. Спочатку бот надсилає повідомлення про "набір тексту" (ChatActions.TYPING), що інформує користувача про обробку запиту. Потім бот проходить по списку товарів, витягнутому з бази даних. Для кожного товару формується текстове повідомлення, що включає назву товару, опис, ціну, а також кнопку для видалення товару. Кнопка використовує product_cb для передачі ID товару та дії "delete", що дозволяє обробляти запити на видалення. Крім тексту, бот відправляє фотографію товару.

```python
async def show_products(message, products, category_idx):
    await bot.send_chat_action(message.chat.id, ChatActions.TYPING)

    for idx, title, body, image, price, tag in products:
        text = f'<b>{title}</b>\n\n{body}\n\nЦена: {price} рублей.'

        markup = InlineKeyboardMarkup()
        markup.add(InlineKeyboardButton(
            '🗑 Удалить',
            callback_data=product_cb.new(id=idx, action='delete')))

        await message.answer_photo(photo=image,
                                   caption=text,
                                   reply_markup=markup)

    markup = ReplyKeyboardMarkup()
    markup.add(add_product)
    markup.add(delete_category)

    await message.answer('Хочете що-небудь додати або видалити?',
                         reply_markup=markup)
```

Рисунок 3.9 – Функція show_products

Після відображення всіх товарів функція show_products додає елементи інтерфейсу з опціями для подальших дій адміністратора: "Додати товар" (add_product) і "Видалити категорію" (delete_category).

Наступна функція delete_category_handler (рис. 3.10) викликається, коли адміністратор натискає кнопку "Видалити категорію". Використовуючи

фільтр IsAdmin, вона гарантує, що цей функціонал доступний лише адміністраторам. Спочатку функція звертається до контексту стану (FSMContext) і перевіряє, чи існує у ньому ключ category_index, що відповідає ідентифікатору вибраної категорії. Якщо такий ключ знайдено, виконується послідовне видалення всіх товарів, пов'язаних із цією категорією, через SQL-запит до бази даних. Після цього видаляється сама категорія за її ідентифікатором. Адміністратору надсилається повідомлення "Готово!" як підтвердження успішного виконання операції, а клавіатура видаляється, щоб уникнути помилкових дій. В кінці, функція викликає process_settings, оновлюючи інтерфейс налаштувань категорій.

```
@dp.message_handler(IsAdmin(), text=delete_category)
async def delete_category_handler(message: Message, state: FSMContext):
    async with state.proxy() as data:
        if 'category_index' in data.keys():
            idx = data['category_index']

            db.query(
                'DELETE FROM products WHERE tag IN (SELECT '
                'title FROM categories WHERE idx=?)',
                (idx,))
            db.query('DELETE FROM categories WHERE idx=?', (idx,))

            await message.answer('Готово!', reply_markup=ReplyKeyboardRemove())
            await process_settings(message)
```

Рисунок 3.10 – Функція delete_category_handler

Функція process_add_product (рис. 3.11) запускає процес додавання нового товару, починаючи з запиту назви. Вона переводить машину станів у стан ProductState.title, сигналізуючи, що наступний крок адміністратора — введення назви товару. У функції створюється клавіатура з однією кнопкою "Скасувати" (cancel_message), що дає можливість адміністраторам перервати процес у будь-який момент. Потім бот надсилає запит із текстом "Назва?" і прикріплює клавіатуру до повідомлення.

```python
@dp.message_handler(IsAdmin(), text=add_product)
async def process_add_product(message: Message):
    await ProductState.title.set()

    markup = ReplyKeyboardMarkup(resize_keyboard=True)
    markup.add(cancel_message)

    await message.answer('Назва?', reply_markup=markup)
```

Рисунок 3.11 – Функція process_add_product

Функція process_cancel (рис. 3.12) викликається, коли адміністратор натискає кнопку "Скасувати" під час введення назви товару. Функція завершує поточний стан машини станів, повідомляючи адміністратора текстом "Добре, відмінено!", і видаляє клавіатуру для уникнення подальших дій. Після цього викликається функція process_settings, яка повертає адміністратора до меню налаштувань, забезпечуючи плавний перехід до інших дій.

```python
@dp.message_handler(IsAdmin(), text=cancel_message, state=ProductState.title)
async def process_cancel(message: Message, state: FSMContext):
    await message.answer('Добре, відмінено!', reply_markup=ReplyKeyboardRemove())
    await state.finish()

    await process_settings(message)
```

Рисунок 3.12 – Функція process_cancel

Функція process_title (рис. 3.13) обробляє введення назви товару, коли бот знаходиться в стані ProductState.title. Вона використовує контекст стану (FSMContext), щоб зберегти введену назву товару в тимчасовому сховищі (data['title']). Це забезпечує збереження даних між етапами процесу, дозволяючи боту обробляти їх на наступних кроках. Після цього функція переводить машину станів у наступний стан (ProductState.next()), сигналізуючи, що наступний крок — введення опису товару. Адміністратору надсилається повідомлення "Опис?" разом із клавіатурою, яка містить кнопку "Назад" (back_markup()).

```python
@dp.message_handler(IsAdmin(), state=ProductState.title)
async def process_title(message: Message, state: FSMContext):
    async with state.proxy() as data:
        data['title'] = message.text

    await ProductState.next()
    await message.answer('Опис?', reply_markup=back_markup())
```

Рисунок 3.13 – Функція process_title

Функція process_title_back обробляє запит адміністратора повернутися до попереднього етапу під час введення назви товару. Якщо адміністратор натискає кнопку "Назад" (back_message) у стані ProductState.title, функція викликає process_add_product, щоб повернути користувача до початкового етапу додавання товару, де запитується назва.

Функція process_body_back виконує аналогічну задачу, але для стану ProductState.body, коли адміністратор вводить опис товару. Під час натискання кнопки "Назад" бот переводить стан машини станів у ProductState.title, що повертає адміністратора до етапу введення назви товару. За допомогою контексту стану (FSMContext) бот отримує раніше введену назву товару (data['title']) і відображає її в повідомленні, пропонуючи змінити.

Функція process_body відповідає за обробку введеного адміністраторами опису товару, коли бот перебуває у стані ProductState.body. Після отримання текстового повідомлення функція зберігає введений опис у контексті стану (data['body']). Потім машина станів переходить до наступного етапу (ProductState.next()), де бот запитує у адміністратора завантаження фото товару.

Функція process_image_photo обробляє завантаження фотографії товару, коли бот перебуває у стані ProductState.image. Вона перевіряє, чи повідомлення адміністратора містить зображення, і отримує ідентифікатор фото (file_id) із повідомлення. Бот завантажує файл із серверів Telegram, читає його вміст і зберігає отримані дані у контексті стану (data['image']). Після цього

функція переводить машину станів на наступний етап (ProductState.next()), де бот запитує у адміністратора введення ціни товару.

Наступна функція process_price (рис. 3.14) відповідає за обробку введеної адміністратором ціни товару. Вона активується, якщо введений текст є числом (перевіряється через умовну функцію lambda message: message.text.isdigit()) і бот перебуває у стані ProductState.price. Ціна зберігається у контексті стану (data['price']), разом із раніше введеними даними: назвою (data['title']) і описом (data['body). Далі бот переходить у стан ProductState.confirm, де він готує підсумкове повідомлення. Це повідомлення містить назву, опис і ціну товару, а також додає до нього зображення, отримане на попередньому етапі.

```
@dp.message_handler(IsAdmin(), lambda message: message.text.isdigit(),
                    state=ProductState.price)
async def process_price(message: Message, state: FSMContext):
    async with state.proxy() as data:
        data['price'] = message.text

        title = data['title']
        body = data['body']
        price = data['price']

        await ProductState.next()
        text = f'<b>{title}</b>\n\n{body}\n\nЦіна: {price} гривень.'

        markup = check_markup()

        await message.answer_photo(photo=data['image'],
                                   caption=text,
                                   reply_markup=markup)
```

Рисунок 3.14 – Функція process_price

Функція process_confirm (рис. 3.15) обробляє підтвердження збереження товару після того, як адміністратор натискає кнопку "Все правильно" (all_right_message) у стані ProductState.confirm. Вона отримує всі введені раніше дані (назву, опис, фото, ціну) з контексту стану і зчитує тег категорії з бази даних на основі ідентифікатора категорії (data['category_index). Для

створення унікального ідентифікатора товару використовується алгоритм MD5, який хешує об’єднані дані товару (назву, опис, ціну та тег).

Потім функція зберігає товар у базі даних через SQL-запит, де кожен товар отримує унікальний ідентифікатор, що забезпечує цілісність і унікальність записів. Після успішного збереження бот завершує стан машини станів (state.finish()), надсилає адміністратору повідомлення "Готово!" і видаляє клавіатуру, що використовувалась раніше. Нарешті, викликається функція process_settings, яка повертає адміністратора до меню налаштувань категорій.

```python
@dp.message_handler(IsAdmin(), text=all_right_message,
                    state=ProductState.confirm)
async def process_confirm(message: Message, state: FSMContext):
    async with state.proxy() as data:
        title = data['title']
        body = data['body']
        image = data['image']
        price = data['price']

        tag = db.fetchone(
            'SELECT title FROM categories WHERE idx=?',
            (data['category_index'],))[0]
        idx = md5(' '.join([title, body, price, tag]
                           ).encode('utf-8')).hexdigest()

        db.query('INSERT INTO products VALUES (?, ?, ?, ?, ?, ?)',
                 (idx, title, body, image, int(price), tag))

    await state.finish()
    await message.answer('Готово!', reply_markup=ReplyKeyboardRemove())
    await process_settings(message)
```

Рисунок 3.15 – Функція process_confirm

Функція delete_product_callback_handler обробляє подію, коли адміністратор натискає кнопку видалення товару в інтерфейсі бота. Вона активується за допомогою callback-запиту з дією delete, визначеною у product_cb. Функція отримує ідентифікатор товару (product_idx) із даних callback-запиту (callback_data['id']) і виконує SQL-запит для видалення товару з бази даних. Після успішного видалення бот надсилає адміністратору коротке сповіщення "Видалено!" через query.answer() і видаляє повідомлення з інформацією про товар із чату.

Функція process_confirm_back (рис. 3.16) дозволяє адміністратору повернутися до етапу зміни ціни під час підтвердження введених даних про товар. Якщо адміністратор натискає кнопку "Назад" (back_message) у стані ProductState.confirm, бот переводить машину станів у попередній стан (ProductState.price), що відповідає введенню ціни. Потім бот отримує поточну ціну з контексту стану (data['price']) і надсилає повідомлення з текстом, який пропонує змінити цю ціну, додаючи клавіатуру для подальших дій (back_markup()).

```python
@dp.message_handler(IsAdmin(), text=back_message, state=ProductState.confirm)
async def process_confirm_back(message: Message, state: FSMContext):
    await ProductState.price.set()

    async with state.proxy() as data:
        await message.answer(f"ЗМінити ціну з <b>{data['price']}</b>?",
                             reply_markup=back_markup())
```

Рисунок 3.16 – Функція process_confirm_back

Компонент orders.py (рис. 3.17) реалізує функціонал для адміністраторів, пов'язаний із переглядом замовлень у телеграм-боті. Він дозволяє отримувати інформацію про всі замовлення, збережені в базі даних, і відображати їх у зручному текстовому форматі. Компонент складається з двох функцій, які працюють разом, забезпечуючи динамічний і структурований інтерфейс.

Функція process_orders відповідає за обробку запиту адміністратора для перегляду замовлень. Вона активується, коли адміністратор надсилає повідомлення, яке відповідає тексту orders, фільтр IsAdmin, гарантує використання функціоналу тільки для адміністратора.

```python
from aiogram.types import Message

from filters import IsAdmin
from handlers.user.menu import orders
from loader import db, dp

@dp.message_handler(IsAdmin(), text=orders)
async def process_orders(message: Message):
    orders = db.fetchall('SELECT * FROM orders')
    if len(orders) == 0:
        await message.answer('У вас немає замовлень.')
    else:
        await order_answer(message, orders)


async def order_answer(message, orders):
    res = ''
    for order in orders:
        res += f'Замолвення <b>№{order[3]}</b>\n\n'

    await message.answer(res)
```

Рисунок 3.17 – Компонент orders.py

Функція виконує SQL-запит до бази даних для отримання всіх замовлень (SELECT * FROM orders) і перевіряє, чи існують записи. Якщо замовлень немає, адміністратору надсилається повідомлення "У вас немає замовлень". У разі наявності замовлень викликається допоміжна функція order_answer, яка формує і надсилає текст із переліком замовлень.

Функція order_answer отримує список замовлень і формує текстове повідомлення, яке містить базову інформацію про кожне замовлення. Для цього вона інтегрується по списку замовлень і додає рядок із номером замовлення (order[3]) до змінної res. Після формування тексту він надсилається адміністратору через message.answer().

Компонент questions.py (рис. 3.18) реалізує функціонал обробки питань користувачів і надання відповідей адміністраторами в телеграм-боті. Він забезпечує зручний інтерфейс для перегляду, вибору та відповіді на питання через інтерактивні клавіатури, використовуючи систему станів (FSM) для організації послідовності дій адміністратора.

```python
from aiogram.dispatcher import FSMContext
from aiogram.types import (
    CallbackQuery,
    InlineKeyboardButton,
    InlineKeyboardMarkup,
    Message,
    ReplyKeyboardRemove,
)
from aiogram.types.chat import ChatActions
from aiogram.utils.callback_data import CallbackData

from filters import IsAdmin
from handlers.user.menu import questions
from keyboards.default.markups import (
    all_right_message,
    cancel_message,
    submit_markup,
)
from loader import bot, db, dp
from states import AnswerState


question_cb = CallbackData('question', 'cid', 'action')
```

Рисунок 3.18 – Компонент questions.py

Функція process_questions (рис. 3.19) активується, коли адміністратор надсилає команду, яка відповідає тексту questions. Вона виконує SQL-запит до бази даних для отримання списку всіх питань, що зберігаються в таблиці questions. Якщо список порожній, адміністратор отримує повідомлення "Питання відсутні". Якщо ж питання є, кожне з них відображається окремо, разом із кнопкою "Відповісти", створеною за допомогою InlineKeyboardMarkup. Натискання цієї кнопки викликає функцію process_answer, яка переводить адміністратора до етапу введення відповіді.

```python
@dp.message_handler(IsAdmin(), text=questions)
async def process_questions(message: Message):
    await bot.send_chat_action(message.chat.id, ChatActions.TYPING)
    questions = db.fetchall('SELECT * FROM questions')

    if len(questions) == 0:

        await message.answer('Питання відсутні.')

    else:

        for cid, question in questions:
            markup = InlineKeyboardMarkup()
            markup.add(InlineKeyboardButton(
                'Відповісти',
                callback_data=question_cb.new(cid=cid, action='answer')))

            await message.answer(question, reply_markup=markup)
```

Рисунок 3.19 – Функція process_questions

Функція process_answer (рис. 3.20) обробляє натискання кнопки "Ответить" і зберігає у стані (FSMContext) унікальний ідентифікатор питання (cid), отриманий із callback-запиту. Далі бот надсилає адміністратору запит "Надрукуй відповідь", видаляючи попередню клавіатуру (ReplyKeyboardRemove), і переводить машину станів у стан AnswerState.answer, сигналізуючи про необхідність введення текстової відповіді.

Функція process_submit (рис. 3.20) обробляє текстову відповідь адміністратора, коли бот перебуває в стані AnswerState.answer. Введений текст зберігається у контексті стану як відповідь (data['answer]). Після цього бот переходить у стан AnswerState.submit, де адміністратору надсилається повідомлення з проханням перевірити введену відповідь на помилки. До повідомлення додається клавіатура з кнопками підтвердження ("Все правильно") або скасування ("Відмінено"), які дають змогу адміністратору прийняти або відхилити відповідь.

```python
@dp.callback_query_handler(IsAdmin(), question_cb.filter(action='answer'))
async def process_answer(query: CallbackQuery, callback_data: dict,
                         state: FSMContext):
    async with state.proxy() as data:
        data['cid'] = callback_data['cid']

    await query.message.answer('Надрукуй відповідь.',
                               reply_markup=ReplyKeyboardRemove())
    await AnswerState.answer.set()


@dp.message_handler(IsAdmin(), state=AnswerState.answer)
async def process_submit(message: Message, state: FSMContext):
    async with state.proxy() as data:
        data['answer'] = message.text

    await AnswerState.next()
    await message.answer('Переконайтеся, що ви не помилилися у відповіді.',
                         reply_markup=submit_markup())
```

Рисунок 3.20 – Функції process_answer, process_submit

Функція process_send_answer, активована кнопкою "Все правильно", завершує процес. Вона отримує відповідь і ідентифікатор питання зі стану, видаляє це питання з бази даних і формує текст повідомлення, яке включає

запитання та відповідь. Потім це повідомлення надсилається користувачеві, який поставив питання (cid). Після цього стан завершується, а адміністратору надсилається повідомлення "Відправлено!" разом із очищенням клавіатури. Функція process_send_answer, активована кнопкою "Відмінено", завершує стан і інформує адміністратора, що операцію було скасовано.

```python
@dp.message_handler(IsAdmin(), text=all_right_message, state=AnswerState.submit)
async def process_send_answer(message: Message, state: FSMContext):

    async with state.proxy() as data:

        answer = data['answer']
        cid = data['cid']

        question = db.fetchone(
            'SELECT question FROM questions WHERE cid=?', (cid,))[0]
        db.query('DELETE FROM questions WHERE cid=?', (cid,))
        text = f'Вопрос: <b>{question}</b>\n\nОтвет: <b>{answer}</b>'

        await message.answer('Відправлено!', reply_markup=ReplyKeyboardRemove())
        await bot.send_message(cid, text)

    await state.finish()
```

Рисунок 3.21 – Функція process_send_answer

Markups.py відповідає за створення кнопок для взаємодії з користувачами в телеграм-боті. Він надає кілька функцій для генерації різних варіантів кнопок, які використовуються на різних етапах роботи бота, забезпечуючи зручність і інтуїтивність інтерфейсу. Усі клавіатури створюються за допомогою класу ReplyKeyboardMarkup із бібліотеки Aiogram (рис. 3.22).

```python
from aiogram.types import ReplyKeyboardMarkup


back_message = '👈 Назад'
all_right_message = '✅ Все вірно'
cancel_message = '🚫 Відмінити'
confirm_message = '✅ Підтвердити замовлення'

def back_markup():
    markup = ReplyKeyboardMarkup(resize_keyboard=True, selective=True)
    markup.add(back_message)

    return markup
```

Рисунок 3.22 – Компонент Markups.py

Функція back_markup створює одну кнопку "Назад". Вона використовується для надання можливості користувачеві повернутися до попереднього етапу або меню. Ця клавіатура проста й інтуїтивно зрозуміла, оскільки дозволяє уникнути помилкових дій і дає змогу швидко виправити введену інформацію.

Функція check_markup створює клавіатуру з двома кнопками, розташованими в одному рядку: "Назад" і "Все вірно". Ця клавіатура зазвичай використовується на етапах підтвердження введених даних, надаючи користувачеві вибір між поверненням для виправлення і підтвердженням правильності введеної інформації.

```
def back_markup():
    markup = ReplyKeyboardMarkup(resize_keyboard=True, selective=True)
    markup.add(back_message)

    return markup

def check_markup():
    markup = ReplyKeyboardMarkup(resize_keyboard=True, selective=True)
    markup.row(back_message, all_right_message)

    return markup
```

Рисунок 3.23 – Функції back_markup, check_markup

Функція confirm_markup (рис. 3.24) формує кнопки для підтвердження замовлення. Вона включає дві кнопки, розташовані в стовпчик: " Підтвердити замовлення" і "Назад".

Функція submit_markup (рис. 3.24) створює 2 кнопки: " Відмінити" і "Все вірно". Ця клавіатура призначена для моментів, коли користувач повинен остаточно прийняти рішення або скасувати операцію. Розташування кнопок в одному рядку робить вибір більш компактним і зручним для користувача.

```python
def confirm_markup():
    markup = ReplyKeyboardMarkup(resize_keyboard=True, selective=True)
    markup.add(confirm_message)
    markup.add(back_message)

    return markup

def submit_markup():
    markup = ReplyKeyboardMarkup(resize_keyboard=True, selective=True)
    markup.row(cancel_message, all_right_message)

    return markup
```

Рисунок 3.24 – Функції confirm_markup, submit_markup

Файл app.py (рис. 3.25) є головним компонентом програми, який забезпечує запуск і основну функціональність телеграм-бота. Він обробляє команду /start, надаючи користувачам привітальне повідомлення з описом можливостей бота та інтерактивну клавіатуру для вибору режиму "Користувач" або "Адмін". Адміністративний режим дозволяє користувачам отримувати розширені права, додаючи їх до списку адміністраторів (ADMINS), а користувацький режим обмежує їхній доступ, видаляючи їх із цього списку. Файл також ініціалізує базу даних через функцію on_startup, створюючи необхідні таблиці під час запуску. Завдяки інтеграції з бібліотекою Aiogram, програма забезпечує плавний і стабільний процес обробки команд і повідомлень, використовуючи обробники для взаємодії з користувачами, а також виконує налаштування логування для моніторингу роботи бота. Усе це робить компонент центральною частиною системи, яка об'єднує її функції й забезпечує ефективну взаємодію між користувачами та ботом.

```python
from logging import INFO, basicConfig

from aiogram import executor, types
from aiogram.types import ReplyKeyboardMarkup, ReplyKeyboardRemove

import handlers

from data.config import ADMINS
from loader import bot, db, dp


user_message = 'Користувач'
admin_message = 'Адмін'


@dp.message_handler(commands='start')
async def cmd_start(message: types.Message):
    markup = ReplyKeyboardMarkup(resize_keyboard=True)

    markup.row(user_message, admin_message)

    await message.answer('''Привіт! 👋

🤖 Я бот-магазин з продажу товарів будь-якої категорії.

🛍️ Щоб перейти до каталогу і вибрати
товари, що сподобалися скористайтесь командою /menu.

❓ Виникли питання? Чи не проблема! Команда /sos допоможе
зв'язатися з адмінами, які постараються якнайшвидше відгукнутися.
    ''', reply_markup=markup)
```

Рисунок 3.25 – Компонент app.py

3.4 Тестування чат боту

Основною метою тестування є виявлення та усунення помилок, перевірка працездатності основного функціоналу, таких як обробка команд, відповідність інтерактивного інтерфейсу очікуванням користувачів, а також коректна взаємодія з базою даних.

Процес тестування включає перевірку коректності обробки повідомлень, відповідності логіки ботів до вимог, перевірку захищеності даних і здатності бота виконувати свої задачі при високих навантаженнях. Особливу увагу приділяють тестуванню різних режимів роботи (користувача та адміністратора), інтерактивних клавіатур, обробки команд, таких як /start, /menu, /sos, і динамічної взаємодії з користувачами.

Спочатку завантажимо бота, та відкриємо його в месенджері (рис. 3.26):

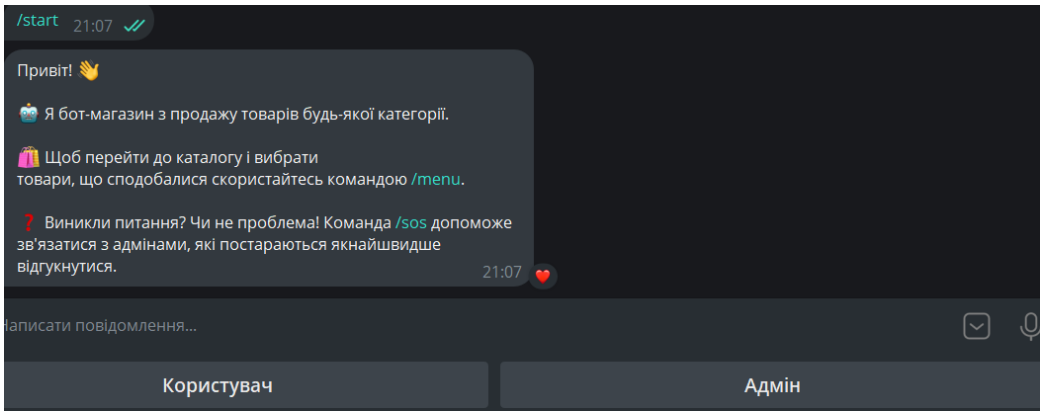


Рисунок 3.26 – Відкриття боту

Перед користувачем з’явилося меню в якому він може обрати режим роботи користувача або адміністратора. Для початку потрібно перевірити базові команди, для прикладу «/sos» (рис. 3.27).

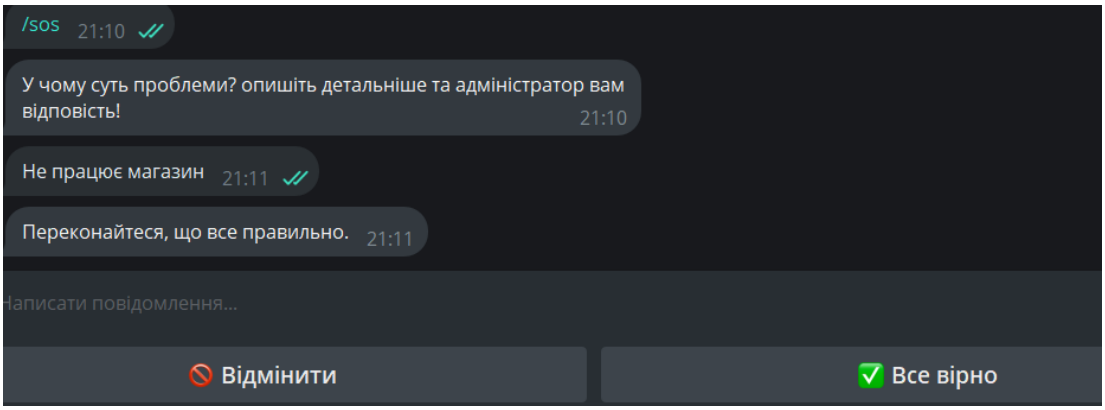


Рисунок 3.27 – Робота системи відгуків

За допомогою даної команди користувач може зв’язатися з технічною підтримкою та відправити власний відгук про проблеми даного додатку.

Для повернення для стартового режиму просто введемо команду «/start», і процес користування додатком відновиться (рис. 3.28):

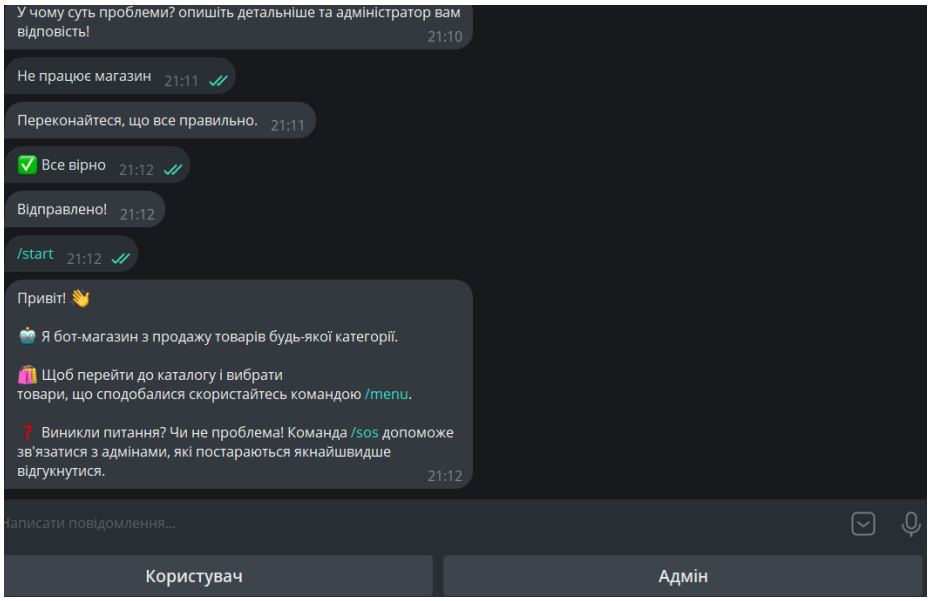


Рисунок 3.28 – Початок роботи з ботом

Після ввімкнення режиму користувача, чат-бот пропонує наступні вкладки, де користувач може замовити товар, дізнатися яка кількість товарів лежить на кошику або отримати статус замовлення (рис. 3.29):

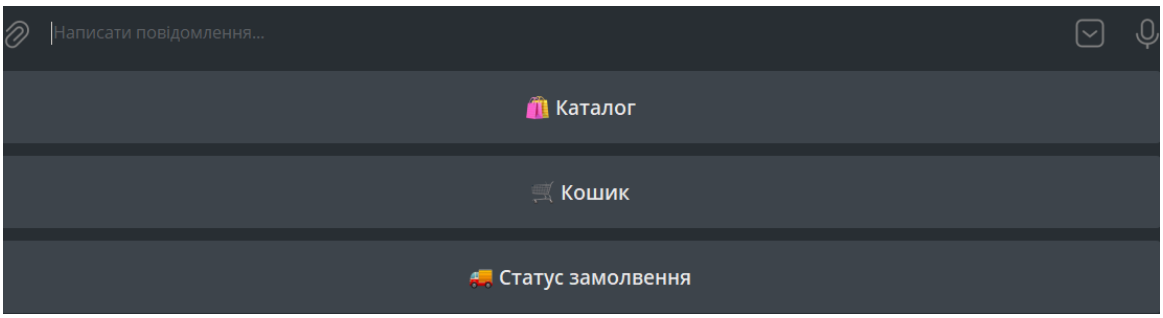


Рисунок 3.29 – Меню користувача

Аналогічно в адміністратора теж є меню, але з більш розширеними правами редагування наприклад товарів, категорій (рис. 3.30):

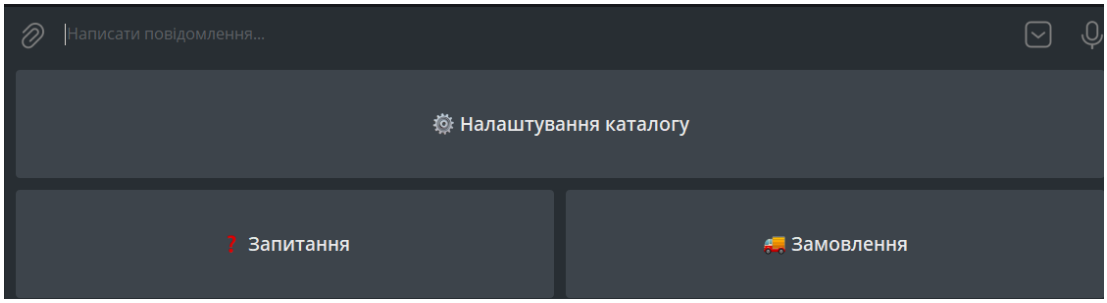


Рисунок 3.30 – Меню адміністратора

Спробуємо натиснути на налаштування каталогу:

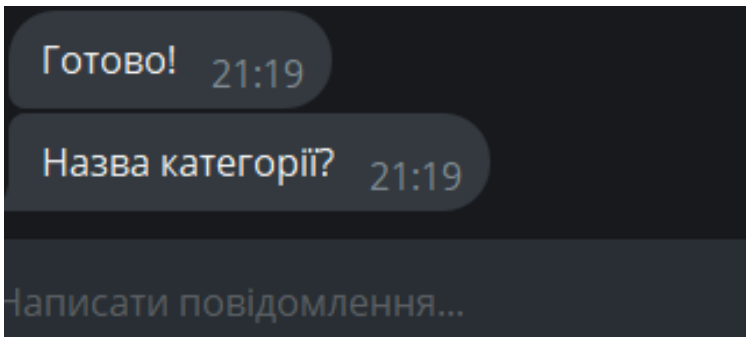


Рисунок 3.31 – Налаштування категорії

Користувачу пропонується додати за його бажанням категорію, до прикладу введемо все що від нас вимагає бот:

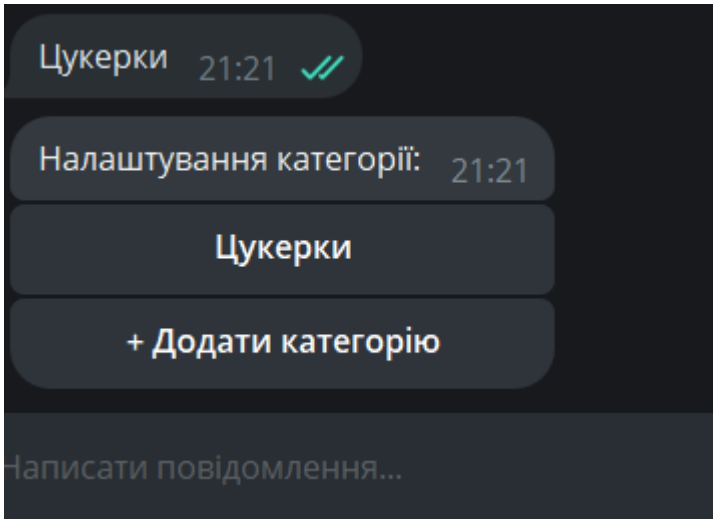


Рисунок 3.32 – Додавання нової категорії

В результаті сформувалася нова категорія в яку адміністратор може додавати нові товари (рис. 3.33):

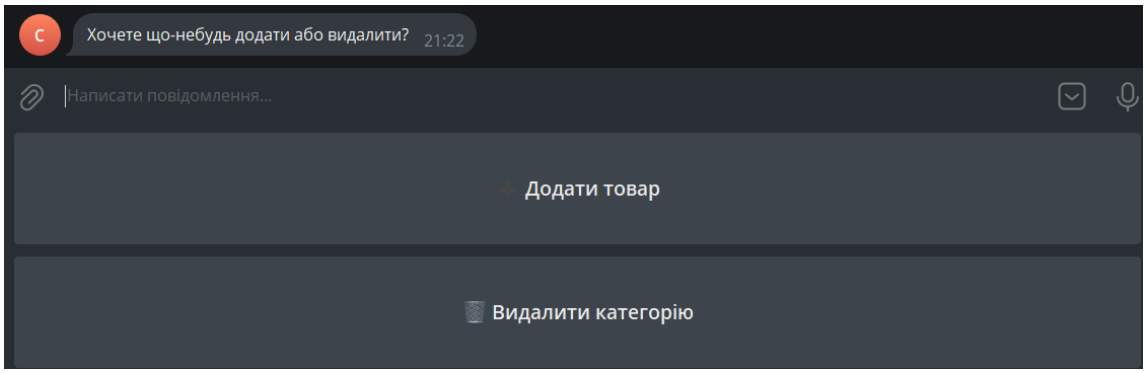


Рисунок 3.33 – Меню для додавання або видалення

В результаті виконуючи всі вимоги перед якими ставить чат-бот, створився новий товар категорії цукерок (рис. 3.34):

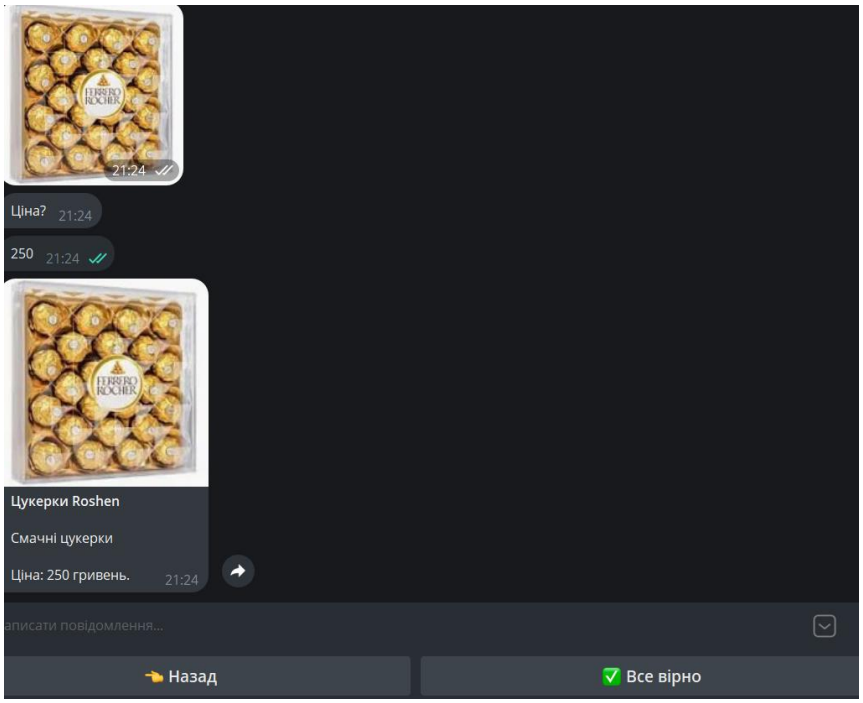


Рисунок 3.34 – Створення нового товару

Зі сторони користувача, розглянувши каталог, він теж зможе побачити новий товар та замовити його (рис. 3.35):

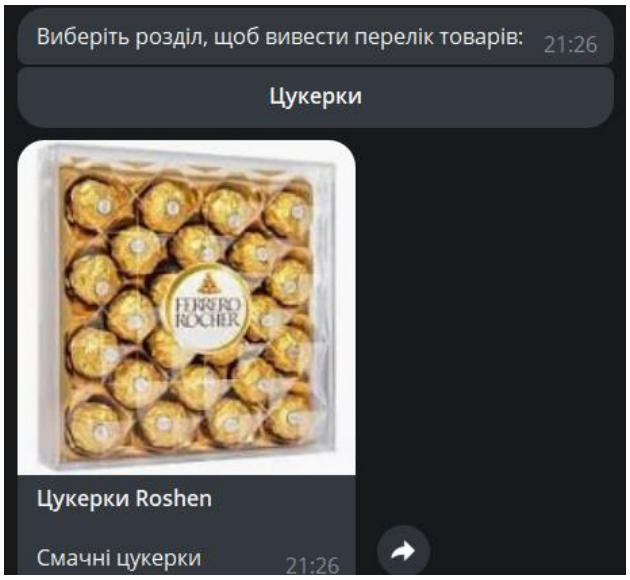


Рисунок 3.35 – Додавання товарів в корзину

Далі після додавання користувачем товару, є можливість оформити замовлення для доставки, для цього потрібно відповісти на всі питання боту:

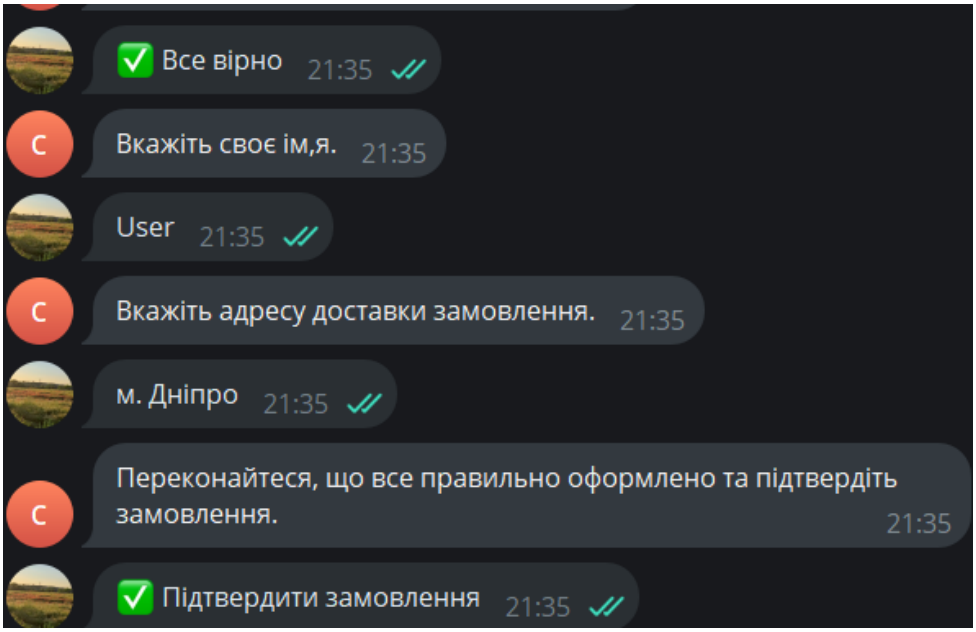


Рисунок 3.36 – Оформлення замолвення

3.5 Висновок третього розділу

У результаті роботи було розроблено функціональний чат-бот, який забезпечує автоматизацію процесів взаємодії з користувачами та адміністрування. Чат-бот реалізує широкий спектр функціональності, включаючи обробку команд, перегляд та управління товарами, опрацювання запитань користувачів, а також роботу з замовленнями. Використання бібліотеки Aiogram дозволило створити гнучку архітектуру з чітким поділом ролей користувачів (адміністраторів і клієнтів) та забезпечити інтуїтивно зрозумілий інтерфейс.

Особливу увагу було приділено динамічному управлінню базою даних, створенню інтерактивних кнопок і впровадженню системи станів (FSM), що дозволило зробити процеси більш послідовними й зручними. Інтеграція адміністративних функцій, таких як додавання, редагування та видалення категорій і товарів, спрощує управління ботом і забезпечує ефективність його використання.

Розроблений чат-бот не лише забезпечує зручний інтерфейс для кінцевих користувачів, але й відповідає потребам бізнесу, автоматизуючи рутинні операції, покращуючи якість обслуговування та оптимізуючи робочі процеси. Така система може бути легко адаптована до конкретних вимог і розширена для додаткових сценаріїв використання, що робить її універсальним рішенням для автоматизації в сфері онлайн-продажів або інших галузях.

ВИСНОВОК

У процесі виконання кваліфікаційної роботи на тему «Розробка інформаційної системи продажу товарів за допомогою чат-бота» було здійснено комплексний аналіз методів створення та впровадження чат-ботів, а також розроблено програмне рішення на базі месенджера Telegram. Дослідження охоплювало вивчення історії розвитку чат-ботів, огляд класифікації за рівнем складності й алгоритмічною реалізацією, обґрунтування вибору платформи та інструментів розробки, а також проєктування й тестування прототипу інформаційної системи.

Основні завдання, вирішені в межах дослідження, були такими:

1. Проведено огляд технологій та підходів до розробки чат-ботів, зокрема аналіз особливостей популярних месенджерів (WhatsApp, Viber, Snapchat, Telegram).
2. Обґрунтовано вибір платформи Telegram з урахуванням її масштабованості, безпеки та зручності інтеграції із зовнішніми сервісами.
3. Визначено функціональні та нефункціональні вимоги до системи: структуровано набір сервісів для клієнтів (перегляд каталогу, оформлення замовлення тощо) та інструментів для адміністраторів (управління товарним каталогом, замовленнями, аналітикою).
4. Розроблено архітектуру чат-бота з поділом на користувацький і адміністративний модулі, а також базу даних для зберігання інформації про товари та замовлення.
5. Здійснено програмну реалізацію системи з використанням бібліотеки aiogram для асинхронної обробки запитів, що забезпечує високу продуктивність та гнучкість масштабування.
6. Проведено тестування та верифікацію реалізованого рішення, підтверджено відповідність вимогам щодо функціональності, продуктивності й безпеки.

За підсумками дослідження створено ефективну інформаційну систему продажу товарів, яка дає змогу автоматизувати рутинні процеси, оперативно реагувати на запити користувачів та керувати базою даних у режимі реального часу. Застосування чат-бота як каналу взаємодії з клієнтами розширює можливості бізнесу, зокрема, підвищує якість обслуговування та спрощує процедуру замовлень. Отримані результати роботи можуть бути використані для подальшого розвитку системи (впровадження платіжних сервісів, розширення аналітичних модулів), а також знайти практичне застосування у сфері електронної комерції, маркетингу й автоматизації бізнес-процесів.

Таким чином, поставлені цілі щодо розробки чат-бота для продажу товарів у Telegram успішно досягнуто, а створене рішення має високий потенціал для вдосконалення та адаптації до різноманітних бізнес-сценаріїв.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Smith D. Building Telegram Bots: Develop Bots with Zero Coding Knowledge. Packt Publishing. 2023. P. 75–88.

2. Smith M. Cybersecurity in the Digital Age: Challenges and Solutions. Cambridge University Press. 2022.

3. Johnson S. Digital Communication Skills: Navigating the Modern Landscape. Routledge, 2021. P. 60–73

4. Платформа WhatsApp Business [Електронний ресурс] – Режим доступу: https://developers.facebook.com/docs/whatsapp/

5. Snap for Developers [Електронний ресурс] – Режим доступу: https://developers.snap.com/api/home

6. Bots: An introduction for developers [Електронний ресурс] – Режим доступу: https://core.telegram.org/bots

7. Telegram Bot API [Електронний ресурс] – Режим доступу: https://core.telegram.org/bots/api

8. TeleBot [Електронний ресурс] – Режим доступу: https://github.com/TelegramBots/Telegram.Bot

9. Telegraf.js - v4.16.3 [Електронний ресурс] – Режим доступу: https://telegraf.js.org

10. Bot API Library Examples [Електронний ресурс] – Режим доступу: https://core.telegram.org/bots/samples

11. Python Telegram Bot’s documentation [Електронний ресурс] – Режим доступу: https://python-telegram-bot.readthedocs.io/en/stable 39

12. Асинхронний Telegram бот мовою Python 3 з використанням бібліотеки aiogram [Електронний ресурс]. URL: https://opensourcelibs.com/lib/aiogram-lessons

13. python-telegram-bot [Електронний ресурс] – Режим доступу: https://github.com/python-telegram-bot/python-telegram-bot

14. Python 3.13.2 documentation [Електронний ресурс] – Режим доступу: https://docs.python.org/3/

15. Most popular social networks worldwide [Електронний ресурс] – Режим доступу: https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/