ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Telegram чат-бот на мові Python з інтеграцією в CRM систему»

на здобуття освітнього ступеня магістра
зі спеціальності F3 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Станіслав КОНИК
(підпис) *(Ім'я, ПРІЗВИЩЕ здобувача)*

Виконав:
здобувач вищої освіти Станіслав КОНИК
група КНДМ-61

Керівник: Олена ШИКУЛА
науковий ступінь, вчене звання д.ф.-м.н., професор

Рецензент:
науковий ступінь, вчене звання _____
(Ім'я, ПРІЗВИЩЕ)

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти Магістр

Спеціальність F3 Комп'ютерні науки

Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Коник Станіслав Павлович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Telegram чат-бот на мові Python з інтеграцією в CRM систему

керівник кваліфікаційної роботи Олена ШИКУЛА д.ф.-м.н., професор,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «30» жовтня 2025 р. № 467

2. Строк подання кваліфікаційної роботи «26» грудня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, технології для розробки сайту:мова програмування Python, KeyCRM.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз структури чат-бота .
Огляд CRM систем.
Огляд методів створення Telegram чат-ботів.
Розробка Telegram чат-бота з інтеграцією в CRM систему.

5. Перелік графічного матеріалу: *презентація*

1. Огляд методів створення Telegram чат-ботів
2. Діаграма архітектури взаємодії бота з користувачем.
3. Інтерфейс користувача та оформлення замовлень.

6. Дата видачі завдання«15» вересня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	15.09-27.09.25	виконано
2	Дослідження інструментів створення Telegram чат-бота	28.09-11.10.25	виконано
3	Визначення функціональних вимог чат-бота	12.10-25.10.25	виконано
4	Вибір системи для розробки та інтеграції CRM системи	26.10-04.11.25	виконано
5	Вибір сервісу для онлайн хостингу Telegram чат-бота	05.11-12.11.25	виконано
6	Розробка бота з інтеграцією в CRM систему	12.11-02.12.25	виконано
7	Оформлення роботи: вступ, висновки, реферат	02.12-08.12.25	виконано
8	Розробка демонстраційних матеріалів	09.12-13.12.25	виконано

Здобувачка вищої освіти _____ Станіслав КОНИК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Олена ШИКУЛА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 54 стор., 30 рис., 3 табл., 23 джерела

Наукове завдання – огляд та аналіз методів і технологій створення Телеграм чат-бота на мові Python з інтеграцією в SRM систему; виявлення переваг та недоліків; проектування архітектури обробки замовленнь; розробка програмного забезпечення з метою оптимізації процесів взаємодії з користувачами та підвищення ефективності й надійності роботи системи.

Мета роботи – створення функціонального Телеграм чат-бота на мові Python з інтеграцією в SRM систему, здатного ефективно взаємодіяти з користувачами у реальному часі та автоматизувати обробку їх запитів.

Об'єкт дослідження – Telegram чат-бот, CRM система.

Предмет дослідження – процес розробки та функціонування Телеграм чат-бота на мові Python з інтеграцією в SRM.

Короткий зміст роботи: Робота спрямована на розробку та дослідження Телеграм чат-бота на мові Python з інтеграцією в SRM систему.

У роботі вивчаються та аналізуються існуючі методи й технології розробки чат-ботів, інтеграція їх в CRM систему, створюється програмне забезпечення для функціонування чат-бота та його взаємодії з користувачами через месенджер Telegram.

Особлива увага приділяється інтеграції з SRM системою для автоматизації обробки запитів та управління даними користувачів. Результати дослідження використовуються для вдосконалення та оптимізації функціональності чат-бота з метою підвищення ефективності та зручності його використання.

Ключові слова: Telegram чат-бот, система CRM, середовище програмування Pyth

ABSTRACT

Text part of the master's qualification work: 54 pages, 30 pictures, 3 table, 23 sources.

The purpose of the work is review and analysis of methods and technologies for creating a Telegram chatbot in Python with integration into the SRM system; identification of advantages and disadvantages; design of order processing architecture; software development to optimize user interaction processes and increase the efficiency and reliability of the system.

Object of research – Telegram chatbot, CRM system.

Subject of research – the process of development and operation of a Telegram chatbot in Python with integration into an SRM system.

Summary of the work: The work is aimed at the development and research of a Telegram chatbot in the Python programming language with integration into an SRM system.

The work studies and analyzes existing methods and technologies for chatbot development, creates software for the operation of the chatbot and its interaction with users through the Telegram messenger.

The main emphasis is on integration with an SRM system for automating the processing of user requests and managing data. The results of the research are used to improve and optimize the functionality of the chatbot to enhance its efficiency and user-friendliness.

Keywords: Telegram chat-bot, system CRM, programing environment Python

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення

Telebot, Aiogram, python-telegram-bot - фреймворки для створення телеграм-ботів у Python

CRM – система управління взаємовідносинами з клієнтами

NLP – це технології, які дозволяють комп’ютеру працювати з текстом і мовленням

REST API – це інтерфейс програмування, який дозволяє різним програмам спілкуватися між собою через HTTP-запити

FSM (Finite State Machine) – це кінцевий автомат, тобто модель, яка описує систему через обмежену кількість станів і переходи між ними в залежності від подій або умов

ВСТУП

У сучасних умовах цифрової трансформації підприємства все більше прагнуть підвищити ефективність своєї роботи за рахунок впровадження інноваційних технологічних рішень. Одним із таких рішень є чат-боти, які дозволяють автоматизувати рутинні процеси, пришвидшити обробку запитів та суттєво покращити взаємодію з клієнтами. Особливої популярності набули чат-боти у месенджерах, серед яких Telegram займає провідне місце завдяки своїй надійності, зручності та відкритому API для розробників.

Поширення месенджерів як основного каналу комунікації між користувачами і компаніями призвело до того, що чат-боти стали не просто додатковим сервісом, а повноцінним інструментом, який інтегрується у загальну інфраструктуру бізнесу. Сучасний чат-бот – це не лише автоматична відповідь або меню з кнопками. Це сервіс, який може зберігати дані, збирати замовлення, обробляти заявки, консультувати, проводити реєстрацію, виконувати аналітичні задачі та взаємодіяти з іншими системами компанії.

Одним із ключових елементів автоматизації є CRM-системи. Вони дозволяють структурувати інформацію про клієнтів, вести історію взаємодій, контролювати роботу менеджерів і відстежувати стан виконання заявок. Інтеграція чат-бота з CRM відкриває можливість створення єдиного інформаційного середовища, у якому всі дії користувачів автоматично фіксуються у базі даних компанії. Це значно спрощує роботу співробітників, усуває людський фактор і зменшує ризики помилок [15].

Окрему увагу серед мов програмування для створення чат-ботів привертає Python. Ця мова має простий синтаксис, зручні бібліотеки (зокрема Python-telegram-bot, Aiogram, PyTelegramBotAPI), а також широкі можливості для інтеграції з API зовнішніх сервісів. Python активно використовується у веб-розробці, аналітиці даних та машинному навчанні, що робить його універсальним

інструментом для створення ботів різної складності – від простих консультантів до інтелектуальних систем [1].

У зв’язку з цим розробка Telegram чат-бота з інтеграцією в CRM є актуальною як з технічної, так і з практичної точки зору. Бізнес отримує можливість оптимізувати комунікацію, зменшити навантаження на менеджерів, пришвидшити обробку звернень, проводити автоматичну реєстрацію заявок та збирати дані для подальшої аналітики. Користувачі, своєю чергою, отримують швидкий доступ до послуг без черг, очікування або необхідності дзвонити на гарячу лінію.

Актуальність дослідження також підтверджується тим, що ринок цифрових комунікацій стрімко розвивається. За останні роки частка користувачів, які взаємодіють з компаніями через месенджери, суттєво зросла. У такій ситуації чат-бот стає не просто вигідним рішенням, а необхідною частиною сучасної цифрової інфраструктури. Саме тому тема створення Telegram чат-бота з інтеграцією у CRM є важливою, своєчасною та має значний практичний інтерес.

Метою цієї магістерської роботи є розробка Telegram чат-бота на мові програмування Python із повноцінною інтеграцією в CRM-систему, що дозволяє автоматизувати обробку звернень, забезпечити структуроване збереження даних та покращити якість взаємодії між компанією та користувачами.

Для реалізації поставленої мети необхідно виконати такі завдання:

– Проаналізувати існуючі рішення та сучасні підходи до створення чат-ботів;

– Розглянути архітектуру Telegram Bot API та можливості Python для розробки ботів;

– Дослідити принципи роботи CRM-систем і їхню роль у бізнес-процесах; спроєктувати структуру та логіку взаємодії бота з CRM;

– Розробити Telegram-бота з основними функціями обробки повідомлень та формування заявок;

– Реалізувати інтеграцію з CRM через REST API або інші методи передачі даних;

– Протестувати систему та оцінити її ефективність у практичному використанні.

Структура магістерської роботи включає вступ, чотири розділи, присвячені теоретичним основам, проєктуванню, розробці та впровадженню системи, висновки, список використаних джерел і додатки, які містять фрагменти коду, технічну документацію та схеми взаємодії.

1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ЧАТ-БОТІВ ТА CRM-СИСТЕМ

1.1 Поняття та класифікація чат-ботів

Чат-бот – це програмне забезпечення, яке автоматично взаємодіє з користувачем через текстові або голосові повідомлення. Основна ідея роботи чат-бота полягає в імітації людського діалогу таким чином, щоб користувач міг швидко отримати потрібну інформацію, виконати певну дію або пройти певний процес без безпосередньої участі оператора. Сьогодні чат-боти застосовуються у різних сферах: бізнесі, обслуговуванні клієнтів, освіті, медицині, електронній комерції, фінансових та страхових компаніях, а також у внутрішніх корпоративних процесах. Вони дозволяють автоматизувати рутинні завдання, підвищити ефективність комунікацій та зменшити навантаження на персонал.

Перші чат-боти працювали на основі простих правил: вони реагували на заздалегідь визначені ключові слова або фрази. Такі системи були обмежені у можливостях та не могли обробляти складні сценарії. З розвитком технологій штучного інтелекту та обробки природної мови (NLP – Natural Language Processing) сучасні чат-боти стали здатними розуміти контекст повідомлень, наміри користувачів та логіку їхніх запитів. Завдяки цьому вони можуть підлаштовувати відповіді під конкретну ситуацію та навіть прогнозувати потреби клієнтів на основі аналізу попередніх повідомлень.

Існує кілька підходів до класифікації чат-ботів. Найпоширеніші з них поділяють ботів за принципом роботи, ціллю використання та типом взаємодії з користувачем:

1. За рівнем інтелектуальності:

– правильні (rule-based) чат-боти. Ці боти працюють за заздалегідь прописаними сценаріями та правилами. Вони реагують на конкретні команди, кнопки або ключові слова. Основна перевага – простота у розробці та стабільність роботи. Недоліком є обмеженість сценаріїв: бот не може обробляти нестандартні запити та потребує постійного оновлення бази правил. Приклад: довідкові боти з

меню або кнопками вибору;

– чат-боти на основі штучного інтелекту. Такі боти використовують NLP та машинне навчання для аналізу тексту, розпізнавання намірів користувача та формування адаптивних відповідей. Вони здатні навчатися на основі історії запитів, що підвищує точність і релевантність взаємодії. Приклад: голосові асистенти, чат-боти, що розуміють вільну мову та можуть вести багаторівневі діалоги.

2. За способом взаємодії з користувачем:

– текстові чат-боти. Спілкуються через текстові повідомлення у месенджерах, на сайтах чи в мобільних додатках. Це найбільш поширений тип;

– голосові чат-боти. Працюють через голосові команди та відповіді. Найчастіше інтегруються у «розумні» пристрої;

– гібридні чат-боти. Поєднують текст і голос, або текст і графічні елементи (кнопки, меню, форми).

3. За сферами застосування:

– сервісні чат-боти. Виконують типові задачі: замовлення товарів, бронювання, довідки;

– підтримка клієнтів. Автоматизують відповіді на часті питання, приймають заявки, допомагають вирішувати прості проблеми;

– інформаційні чат-боти. Показують новини, прогнози погоди, нагадування;

– корпоративні чат-боти. Призначені для внутрішнього використання компанією: контроль задач, аналітика, HR-процеси;

– комерційні чат-боти. Використовуються для продажів, збору лідів, консультацій щодо товарів.

4. За платформою роботи:

– чат-боти для месенджерів (Telegram, Viber, WhatsApp);

– чат-боти для сайтів (вбудовані веб-віджети);

– чат-боти у мобільних додатках; голосові боти для «розумних» колонок

(Alexa, Google Home) [2].

Сучасні чат-боти можуть виконувати не лише прості дії, а й складні бізнес-процеси: автоматично реєструвати замовлення, оновлювати інформацію в CRM, відправляти нагадування, проводити опитування та формувати звіти. Використання чат-ботів дозволяє підвищити швидкість обслуговування, зменшити витрати на персонал та підвищити задоволеність клієнтів.

Крім того, розвиток технологій штучного інтелекту відкриває нові можливості: прогнозування поведінки користувачів, персоналізовані пропозиції, інтеграція з аналітичними системами та автоматичне навчання на основі історії взаємодій. Це робить чат-ботів не просто інструментом комунікації, а повноцінним механізмом автоматизації бізнес-процесів.

Отже, сучасні чат-боти є гнучким та масштабованим інструментом, який може мати різний рівень складності та функціоналу залежно від призначення. Вони дозволяють оптимізувати процеси, підвищити ефективність роботи компанії та забезпечити швидке й зручне обслуговування користувачів, що робить їх невід’ємною частиною сучасних інформаційних та бізнес-систем.

1.2 Огляд технологій створення чат-ботів

Створення чат-ботів стало значно простішим завдяки розвитку сучасних технологій, фреймворків та платформ, які надають інструменти для обробки повідомлень, підключення до месенджерів та інтеграції з зовнішніми сервісами. На сьогодні існує велика кількість способів розробки ботів – від простих конструкторів без програмування до комплексних систем на основі штучного інтелекту. У цьому підрозділі розглянемо основні технології, які використовуються для створення сучасних чат-ботів (рис. 1.1).

Одним із найпростіших способів створення чат-бота є використання платформ, що дозволяють налаштовувати логіку роботи без написання коду. Такі системи працюють за принципом блок-схем і сценаріїв, де розробник створює послідовність дій, кнопки, відповіді та переходи між ними.

Найпопулярніші конструктори:

– ManyChat – орієнтований на маркетингові кампанії та продажі, підтримує Telegram, Facebook, WhatsApp.

– Chatfuel – простий у використанні сервіс для створення інформаційних ботів.

– Botsify та Flow XO – підтримують інтеграції з CRM та зовнішніми API.

Перевагою конструкторів є легкість та швидкість розробки, а недоліком – обмеженість функціональності, що не дозволяє створювати складні логічні механізми або повну інтеграцію з бізнес-системами.

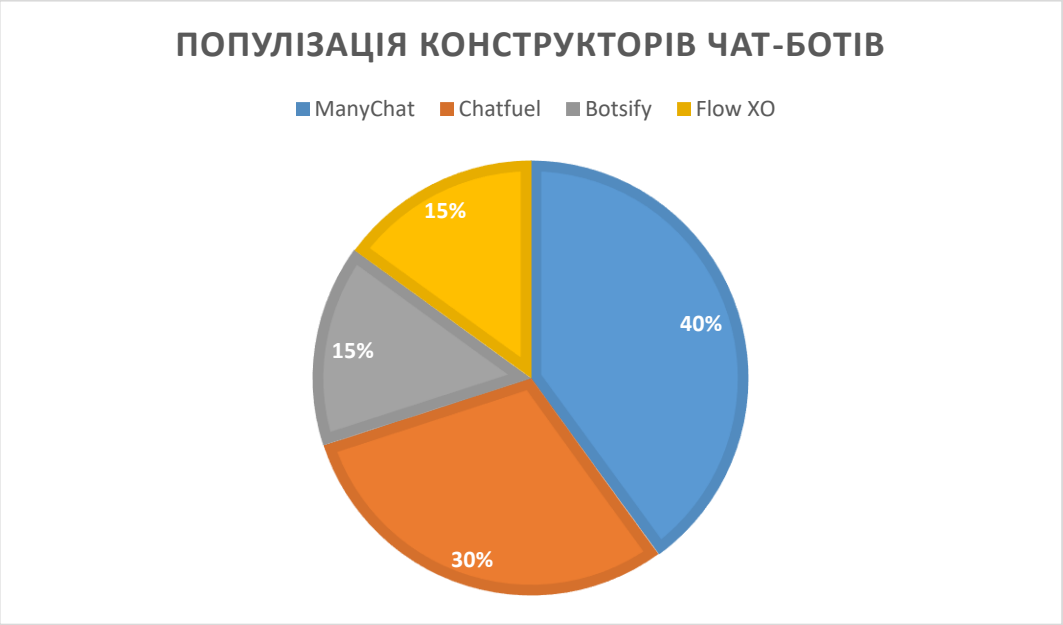


Рисунок 1.1 – Секторна діаграма популізації конструкорів телеграм чат ботів з 2020 по 2025 роки

Для створення складніших ботів розробники використовують фреймворки різними мовами програмування. Вони надають можливість працювати безпосередньо з API месенджера, обробляти дані, підключати бази даних та виконувати інтеграцію зі сторонніми сервісами (рис. 1.2).

Python-фреймворки:

– Aiogram – сучасний, асинхронний фреймворк для Telegram Bot API. Забезпечує швидку обробку великої кількості повідомлень, підтримує FSM, інлайн-кнопки та вебхуки [7].

– PyTelegramBotAPI (Telebot) – простіший у використанні фреймворк,

синхронний, підходить для невеликих або середніх проектів [6].

– python-telegram-bot – один із найстаріших і найстабільніших фреймворків з широкими можливостями [8].

Вибір конкретного фреймворку залежить від цілей розробника: для масштабних та високонавантажених ботів оптимальним буде Aiogram, для простих рішень – Telebot, а для структурованих і функціонально насичених систем – python-telegram-bot. Такий підхід дозволяє ефективно реалізувати чат-бот будь-якого рівня складності.

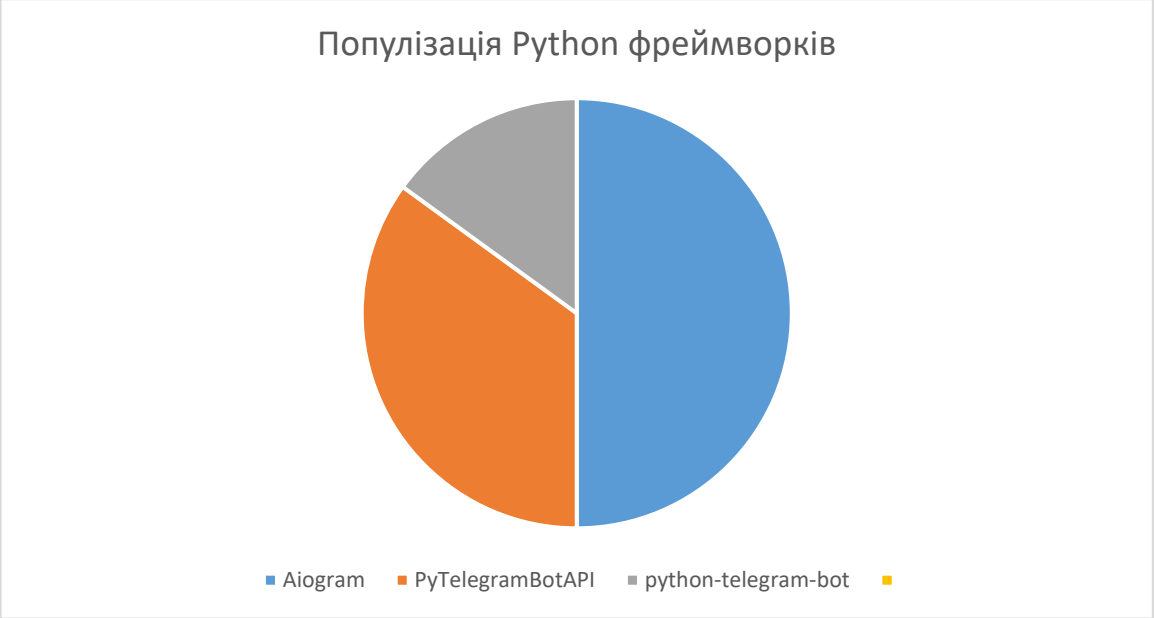


Рисунок – 1.2 Секторна діаграма популізації Python фреймворків

Для стабільної роботи чат-бота необхідно забезпечити цілодобовий доступ до сервера. Найпопулярніші рішення:

– Heroku – простий сервіс для розгортання ботів на Python та Node.js;

– PythonAnywhere – популярний серед студентів і розробників, підтримує вебхуки; Render, Railway, Vercel – сучасні платформи для безкоштовного розгортання;

– AWS, Google Cloud, Azure – для масштабованих комерційних проєктів.

Технології створення чат-ботів охоплюють широкий спектр інструментів та підходів – від простих візуальних конструкторів до потужних програмних фреймворків і платформ штучного інтелекту. Прості конструктори дозволяють

швидко створювати бота без глибоких знань програмування, налаштовувати сценарії діалогу через графічний інтерфейс та підключати базові інтеграції. Вони ефективні для швидкого запуску проектів, проте обмежені у можливостях персоналізації, масштабування та аналітики.

Більш складні та гнучкі рішення реалізуються за допомогою мов програмування та спеціалізованих фреймворків. Вони дозволяють створювати багаторівневі сценарії взаємодії, підключати бази даних, автоматизувати складні бізнес-процеси та інтегруватися з різними системами, включно з CRM, ERP та аналітичними платформами. При цьому можна реалізувати повністю кастомізовану поведінку бота, забезпечити безпеку даних та збирати детальну аналітику про користувачів.

Вибір конкретної технології залежить від низки факторів: складності проекту, необхідності інтеграцій з існуючими системами, бюджету, очікуваного навантаження та вимог до функціональності. Для розробки Telegram чат-бота з інтеграцією в CRM найчастіше обирають мову програмування Python через її простоту, доступність бібліотек та підтримку сучасних фреймворків, таких як Aiogram, PyTelegramBotAPI чи python-telegram-bot. Використання цих інструментів забезпечує гнучкість у налаштуванні, надійність роботи, масштабованість системи та можливість швидкого додавання нових функцій або змін у сценаріях взаємодії.

Таким чином, комбінування правильно обраних технологій і фреймворків дозволяє створювати ефективні та адаптивні чат-боти, які не лише автоматизують взаємодію з користувачами, а й підвищують ефективність бізнес-процесів та покращують якість обслуговування клієнтів. Розумний підхід до вибору технології дозволяє оптимізувати ресурси, мінімізувати витрати часу на розробку та забезпечити стабільну роботу системи навіть при високому навантаженні.

1.3 Особливості роботи Telegram Bot AP

Telegram Bot API – це офіційний програмний інтерфейс, призначений для створення чат-ботів у месенджері Telegram. Він надає розробникам повний набір інструментів для автоматизованої взаємодії з користувачами через текстові повідомлення, кнопки, меню, мультимедійний контент та інтерактивні елементи. Основною особливістю Telegram Bot API є те, що він базується на стандартних HTTP-запитах до серверів Telegram, що дозволяє ботам працювати незалежно від клієнтського додатка користувача. Це забезпечує високу гнучкість розробки та можливість інтеграції ботів у різні системи і платформи [22].

Однією з ключових характеристик Telegram Bot API є підтримка асинхронного обміну повідомленнями, що дозволяє обробляти велику кількість запитів одночасно. Завдяки цьому бот може швидко реагувати на повідомлення користувачів навіть при високому навантаженні, що особливо важливо для комерційних проєктів, сервісів підтримки клієнтів та масових інформаційних кампаній. Telegram надає два основних способи отримання повідомлень від користувачів: полінг (long polling) та вебхуки (webhooks) [16].

1. Полінг (Long Polling) – етод, при якому бот періодично надсилає запит до серверів Telegram та очікує на нові повідомлення. Якщо повідомлень немає, сервер утримує запит до певного часу, після чого повертає порожню відповідь. Цей підхід простий у реалізації, не потребує складної серверної інфраструктури та підходить для невеликих проектів або тестових середовищ. Однак при великій кількості користувачів він може створювати невелику затримку у відповіді, а також збільшувати навантаження на сервер через постійні запити.

2. Вебхуки (Webhooks) – метод, при якому сервер Telegram автоматично надсилає дані про нові повідомлення на вказаний URL бота. Це дозволяє миттєво отримувати інформацію про запити користувачів, зменшує навантаження на сервер бота і підвищує ефективність обробки повідомлень. Вебхуки вважаються більш сучасним і масштабованим рішенням для реальних проєктів, особливо коли бот обслуговує тисячі користувачів одночасно.

Крім того, Telegram боти можуть взаємодіяти з іншими сервісами через HTTP-запити та REST API, що дозволяє реалізовувати складну логіку роботи: підключення до баз даних, інтеграцію з CRM-системами, зовнішніми веб-сервісами, аналітичними платформами або фінансовими сервісами. Це робить Telegram Bot API універсальним інструментом не лише для створення простих інформативних ботів, а й для побудови повноцінних систем автоматизації бізнес-процесів.

Ще однією важливою особливістю Telegram Bot API є високий рівень безпеки та контролю доступу. Кожен бот має унікальний токен авторизації, що гарантує, що лише власник бота може надсилати або отримувати повідомлення від користувачів. Це забезпечує надійну та захищену комунікацію, зменшує ризик несанкціонованого доступу та дозволяє реалізувати багаторівневу систему прав доступу у складних проєктах.

Telegram Bot API також підтримує масштабування та асинхронну роботу. Боти можуть обробляти одночасно тисячі запитів, підключатися до черг повідомлень, балансувати навантаження на сервери та інтегруватися з хмарними рішеннями. Це особливо актуально для проєктів, де велика кількість користувачів надсилає запити одночасно, наприклад, у службах підтримки, інформаційних каналах або e-commerce.

Таким чином, Telegram Bot API є потужним та гнучким інструментом для розробки автоматизованих сервісів будь-якого рівня складності. Його особливості, такі як асинхронна обробка повідомлень, підтримка вебхуків, інтеграція з зовнішніми сервісами, багатий набір типів повідомлень та високий рівень безпеки, роблять його ефективним засобом для створення як простих інформативних ботів, так і складних систем автоматизації бізнес-процесів. Використання цього API дозволяє компаніям оптимізувати комунікацію з клієнтами, підвищити швидкість обробки запитів і знизити витрати на обслуговування користувачів.

1.4 CRM-системи: призначення, структура та функціональні можливості

CRM-системи (Customer Relationship Management) є ключовим інструментом для організацій, які прагнуть ефективно управляти взаємодією з клієнтами, оптимізувати внутрішні бізнес-процеси та підвищити якість обслуговування. Основне призначення CRM полягає у централізованому зборі, зберіганні та аналізі даних про клієнтів. Це дозволяє компаніям не лише контролювати комунікації з користувачами, а й розуміти їхні потреби, прогнозувати поведінку, швидко реагувати на запити та пропонувати персоналізовані рішення. Використання CRM сприяє ефективному управлінню заявками, замовленнями та угодами, дозволяє вести історію всіх взаємодій із клієнтами і аналізувати результати роботи співробітників.

Структура сучасної CRM-системи включає кілька взаємопов’язаних модулів, що забезпечують комплексну автоматизацію бізнес-процесів:

1. Модуль управління контактами та даними клієнтів. Цей модуль дозволяє зберігати детальну інформацію про користувачів: контактні дані, історію звернень, уподобання, замовлення та комунікації через різні канали (телефон, email, месенджери). Він забезпечує централізоване сховище даних і створює основу для персоналізації пропозицій, сегментації клієнтів та аналітики поведінки користувачів.

2. Модуль управління продажами. Відповідає за автоматизацію процесу укладання угод: формування комерційних пропозицій, ведення етапів угоди, контроль виконання завдань і фіксація результатів. CRM дозволяє менеджерам відстежувати кожну угоду на всіх етапах, прогнозувати обсяги продажів і планувати роботу команди.

3. Модуль обробки звернень і підтримки клієнтів. Забезпечує автоматичний розподіл запитів між співробітниками, контроль термінів їх виконання, ведення статистики та формування звітів. Інтеграція з чат-ботами та месенджерами дозволяє швидко реагувати на запити клієнтів, скорочувати час обробки звернень та підвищувати задоволеність користувачів.

4. Аналітичний модуль. Генерує звіти, аналізує продажі та ефективність

співробітників, прогнозує розвиток бізнесу та допомагає приймати стратегічні рішення. Сучасні CRM-системи часто підтримують побудову візуальних дашбордів і графіків, що дозволяє керівникам оперативно оцінювати стан бізнес-процесів і коригувати стратегії.

5. Модуль маркетингу. Дозволяє налаштовувати розсилки, акції, рекламні кампанії та персоналізовані пропозиції. Використання аналітики і сегментації клієнтів підвищує ефективність маркетингових заходів і забезпечує точкове впливання на аудиторію.

6. Модуль інтеграції. Забезпечує взаємодію CRM із зовнішніми сервісами: чат-ботами, платіжними системами, ERP-платформами, аналітичними системами та іншими програмними продуктами. Завдяки цьому організація може створити єдину екосистему обробки даних і взаємодії з клієнтами.

Функціональні можливості CRM-систем досить широкі. Вони дозволяють не лише вести облік контактів і взаємодій, але й автоматизувати повторювані процеси, наприклад, обробку замовлень або заявок. Завдяки аналітичним інструментам компанія може відстежувати ефективність роботи менеджерів, формувати статистику продажів, оцінювати ефективність маркетингових кампаній і прогнозувати попит на товари чи послуги. Інтеграція CRM із зовнішніми каналами зв’язку, включно з месенджерами та чат-ботами, дозволяє централізувати комунікацію, мінімізувати ризик втрати важливої інформації та підвищити швидкість обробки запитів. Сучасні CRM-системи також підтримують можливість сегментації клієнтів і персоналізації пропозицій, що сприяє підвищенню рівня задоволеності користувачів і лояльності до компанії.

Таким чином, CRM-система виступає фундаментом для організації ефективної взаємодії з клієнтами. Вона дозволяє структурувати інформацію, автоматизувати процеси, підвищувати ефективність продажів і забезпечувати високий рівень обслуговування. Інтеграція CRM із Telegram чат-ботами створює єдину екосистему взаємодії з користувачами. Це поєднує автоматичну обробку запитів, накопичення даних, аналітику в реальному часі та персоналізацію взаємодії. Наприклад, бот може автоматично реєструвати замовлення, оновлювати

інформацію в CRM, надсилати підтвердження клієнтам та генерувати звіти для менеджерів.

Таким чином, CRM-система є фундаментом для організації ефективної взаємодії з клієнтами. Вона дозволяє структурувати інформацію, автоматизувати бізнес-процеси, підвищувати ефективність продажів та забезпечувати високий рівень обслуговування. Інтеграція CRM із сучасними цифровими каналами комунікації, зокрема Telegram чат-ботами, дозволяє компаніям підвищити швидкість обробки запитів, зменшити витрати часу на рутинні завдання та зосередитися на стратегічному розвитку бізнесу, підвищуючи при цьому задоволеність та лояльність клієнтів [20].

1.5 Взаємодія чат-ботів із CRM у бізнес-процесах

Інтеграція чат-ботів із CRM-системами сьогодні є ключовим елементом автоматизації бізнес-процесів у сучасних компаніях. Вона дозволяє поєднати зручність комунікації через популярні месенджери, такі як Telegram, Viber чи WhatsApp, із потужними можливостями управління клієнтською базою, аналітики та автоматизації. Основна ідея такої взаємодії полягає у тому, що чат-бот виступає першою точкою контакту з користувачем. Він приймає запити, обробляє стандартні питання, фіксує замовлення та передає інформацію безпосередньо до CRM-системи для подальшої обробки співробітниками компанії, що забезпечує цілісність даних та зручність управління ними.

Завдяки інтеграції автоматизується значна кількість рутинних операцій, що дозволяє зменшити навантаження на персонал. Наприклад, замість того, щоб оператор вручну вводив дані про клієнта в CRM, чат-бот автоматично збирає необхідну інформацію, перевіряє її коректність, а потім передає до системи. Це не лише економить час співробітників, а й значно знижує ризик помилок у записах, забезпечує оперативну обробку запитів та підвищує швидкість реагування на потреби клієнтів. Крім того, автоматизація дозволяє співробітникам зосередитися на більш складних та стратегічно важливих завданнях, таких як персональні консультації або оптимізація бізнес-процесів.

Інтеграція чат-ботів із CRM також забезпечує постійний обмін даними в режимі реального часу. Бот може автоматично надсилати підтвердження замовлень, нагадування про зустрічі чи оплату, формувати персоналізовані пропозиції на основі даних, збережених у CRM, а також реагувати на зміну статусу заявки без участі людини. Це дозволяє компанії ефективно керувати процесами продажу, обслуговування та маркетингу, а користувачеві надавати швидкі, точні та персоналізовані відповіді, що підвищує рівень задоволеності клієнтів та їхню лояльність [19].

Особливу увагу слід приділити аналітичним можливостям інтегрованої системи. Дані, що надходять від чат-бота, можуть використовуватися для формування детальних звітів у CRM, оцінки ефективності маркетингових кампаній, аналізу поведінки користувачів, прогнозування попиту та виявлення проблемних точок у процесі обслуговування. Таким чином, чат-бот стає не лише інструментом автоматизації, а й джерелом цінної аналітичної інформації, яка дозволяє приймати управлінські рішення на основі реальних даних, а не лише на інтуїції або суб’єктивних оцінках.

Інтеграція також створює уніфіковану екосистему комунікацій. Чат-бот може збирати інформацію з різних каналів, централізовано передавати її до CRM, а система, у свою чергу, обробляє ці дані та забезпечує їх доступність для менеджерів та аналітиків. Це мінімізує ризик втрати важливої інформації, скорочує час на обробку запитів і дозволяє формувати персоналізовані пропозиції для різних сегментів клієнтів. Завдяки цьому компанія отримує змогу швидко реагувати на зміни ринку, запускати цільові маркетингові кампанії та оптимізувати процеси обслуговування.

Ще одним важливим аспектом інтеграції є покращення користувацького досвіду. Користувачі отримують можливість взаємодіяти з системою у зручний для себе час і формат: через текстові повідомлення, кнопки меню, інлайн-формуляри та інші інтерактивні елементи. Це підвищує швидкість комунікації, робить її більш зручною та інтуїтивною, а також дозволяє компанії пропонувати персоналізовані

рішення та акції на основі історії взаємодій, що значно підвищує рівень задоволеності клієнтів.

Отже, інтеграція чат-ботів із CRM-системами є важливим стратегічним інструментом для сучасного бізнесу. Вона дозволяє об’єднати канали комунікації, автоматизувати рутинні операції, забезпечити оперативне реагування на запити клієнтів та збирати аналітичні дані для прийняття ефективних управлінських рішень. Використання таких рішень особливо актуальне для компаній, які прагнуть підвищити ефективність бізнес-процесів, покращити якість обслуговування, скоротити витрати часу на обробку звернень користувачів і створити конкурентні переваги на ринку за рахунок швидкої, персоналізованої та автоматизованої взаємодії з клієнтами.

1.6 Аналіз існуючих рішень та їхніх недоліків

Сучасний ринок чат-ботів та CRM-систем представлений великою кількістю готових комерційних та відкритих рішень. Серед популярних платформ для створення чат-ботів виділяють сервіси KeyCRM, ManyChat, Chatfuel, Botsify, а також програмні фреймворки на Python – Aiogram, PyTelegramBotAPI та python-telegram-bot. Для CRM найбільш поширеними є KeyCRM, Bitrix24, HubSpot, Salesforce, Zoho CRM та AmoCRM. Кожна з цих систем має свої сильні сторони і підходить для певного типу бізнесу та задач.

Конструктори чат-ботів забезпечують швидке створення сценаріїв взаємодії, інтерфейсних кнопок та простих діалогів без програмування. Вони ефективні для малого та середнього бізнесу, де важлива швидкість розробки та мінімальні технічні знання. Проте такі платформи мають низку обмежень: складні логічні сценарії та автоматизація бізнес-процесів реалізовані частково, інтеграція з CRM часто потребує додаткових платних модулів, а можливості аналітики обмежені стандартними звітами. Крім того, готові рішення не завжди дозволяють гнучко налаштовувати поведінку бота під специфічні потреби компанії, що знижує ефективність автоматизації.

Фреймворки для програмування на Python надають значно більшу свободу. Використання Aiogram, PyTelegramBotAPI або python-telegram-bot дозволяє створювати багаторівневі сценарії взаємодії, підключати бази даних, реалізовувати складну бізнес-логіку та інтегрувати бота з CRM. При цьому є можливість персоналізувати досвід користувача, збирати деталізовану аналітику та підлаштовувати систему під будь-які вимоги бізнесу. Основними недоліками таких рішень є необхідність знань програмування, витрати часу на розробку, тестування та забезпечення стабільного серверного середовища, моніторингу та резервного копіювання, що створює додаткові витрати ресурсів.

Комерційні CRM-системи, такі як Salesforce або HubSpot, забезпечують широкий функціонал, високу надійність та готові аналітичні інструменти. Однак їхня вартість може бути значною для малого бізнесу, а гнучкість у налаштуванні складних бізнес-процесів обмежена стандартними модулями. Безкоштовні або локальні CRM, наприклад Bitrix24, AmoCRM або KeyCRM, мають спрощений функціонал інтеграції та обмежені можливості автоматизації, що ускладнює синхронізацію даних і налаштування індивідуальних процесів. Більшість CRM-систем не забезпечують повної автоматизації комунікацій через месенджери без підключення чат-ботів або розробки кастомних модулів [18].

Одним із головних недоліків готових рішень є недостатня інтеграція між чат-ботами та CRM. Це проявляється у дублюванні даних, затримках при передачі інформації та обмежених аналітичних можливостях. Крім того, технічні обмеження платформ ускладнюють масштабування системи при збільшенні кількості користувачів або повідомлень. Інші проблеми включають обмежену персоналізацію взаємодії з клієнтами, складність інтеграції з сторонніми сервісами, обмежені можливості аналітики та ризики щодо безпеки даних при використанні хмарних рішень (таб. 1.1).

Аналіз існуючих рішень показує, що жодна платформа або фреймворк не забезпечує одночасно повної гнучкості, масштабованості, надійності роботи та інтеграції з усіма бізнес-процесами. Для отримання комплексного та ефективного інструменту автоматизації комунікацій та управління клієнтами доцільно

розробляти кастомізований Telegram чат-бот на Python із повноцінною інтеграцією в CRM. Такий підхід дозволяє поєднати гнучкість програмного рішення, можливість масштабування, високий рівень безпеки даних та деталізовану аналітику, що забезпечує повний контроль над бізнес-процесами.

Таблиця 1.1 – Переваги та недоліки існуючих варіантів для створення чат-бота

Категорія	Готові конструктори чат-ботів	Фреймворки для програмування	Комерційні CRM	Безкоштовні або локальні CRM
Інтеграція з CRM та іншими системами	Часткова, часто платна	Повна, через API	Вбудовані інтеграції, але не завжди під специфічні потреби	Обмежена, потребує кастомних рішень
Можливості аналітики	Стандартні звіти	Повна аналітика, кастомні звіти	Широкий набір звітів	Обмежені, часто базові
Масштабованість	Обмежена	Висока, залежить від сервера	Висока, але потребує додаткових ресурсів	Середня, обмеження на кількість користувачів та даних
Безпека та конфіденційність	Дані зберігаються у хмарі, контроль обмежений	Повний контроль, можна зберігати на власних серверах	Висока, стандарти безпеки платформи	Середня, залежить від налаштувань і серверів
Необхідність технічних знань	Мінімальна	Висока, потрібні знання програмування	Середня, потрібне навчання	Середня, потрібне налаштування та адміністрування

Аналіз таблиці показує, що готові конструктори чат-ботів підходять для швидкого запуску та невеликих бізнес-процесів, але обмежені у гнучкості,

аналітиці та інтеграції. Фреймворки для програмування забезпечують максимальну свободу та масштабованість, проте потребують високого рівня технічних знань і додаткових ресурсів для підтримки. Комерційні CRM-системи пропонують широкий функціонал та стабільність, але їхня вартість і обмежена персоналізація можуть бути проблемними для малого бізнесу. Безкоштовні або локальні CRM-системи економічні, проте мають обмежені можливості інтеграції та складніші налаштування для складних процесів.

2 ПРОЄКТУВАННЯ ТЕЛЕГРАМ ЧАТ-БОТА З ІНТЕГРАЦІЄЮ В CRM-СИСТЕМУ

2.1 Постановка задачі та вимоги до системи

Проєктування Telegram чат-бота з інтеграцією в CRM-систему передбачає створення інструменту, здатного автоматизувати ключові процеси взаємодії з клієнтами та забезпечити оперативну передачу даних для подальшої обробки. На сучасному етапі цифровізації бізнесу саме чат-боти у месенджерах стають одним з найзручніших засобів комунікації, оскільки вони доступні користувачам у будь-який час та не потребують додаткової установки програмного забезпечення. У свою чергу, CRM-система забезпечує структуроване зберігання інформації про клієнтів, історію звернень та процеси обробки заявок, що дозволяє підвищити ефективність роботи компанії.

Постановка задачі полягає у розробці Telegram чат-бота, який здатний приймати звернення користувачів, обробляти їх, взаємодіяти з CRM та автоматизувати частину бізнес-процесів. Бот повинен виконувати функції первинного збору інформації, реєстрації запитів, відправлення даних до CRM, а також отримання відповідей або оновлень зі сторони CRM-системи. Він має забезпечувати швидкий канал комунікації між користувачем і компанією, зменшуючи навантаження та покращуючи якість обслуговування [21].

Для успішної реалізації проєкту необхідно сформувати вимоги до системи, які визначають її функціональність, надійність. (таб 2.1).

Таблиця 2.1 – Функціональні та нефункціональні вимоги до створення чат-бота

Функціональні вимоги	**Нефункціональні вимоги**
Можливість обробки текстових повідомлень, команд та кнопок Telegram.	Цілодобову доступність роботи бота без збоїв та простоїв.

Продовження таблиці 2.1

Підтримка меню, інлайн-кнопок та інтерактивних сценаріїв.	Стабільність при високому навантаженні, коли надходить велика кількість заявок одночасно.
Збір та валідація контактних даних користувача.	Можливість додавання нового функціоналу та модулів без порушення роботи системи.
Створення та передача заявок/замовлень у CRM-систему;	Швидкість обробки запитів, що забезпечує комфортну взаємодію користувачів з ботом.
Отримання відповіді від CRM.	Логування дій, зберігання історії звернень та помилок для подальшого аналізу.
Авторизація або ідентифікація користувачів.	Надійність і безпека, включаючи захист токена Telegram і ключів CRM.

Також важливими є вимоги до інтеграції з CRM, яка має здійснюватися через API або вебхуки, що забезпечують двосторонній обмін інформацією у режимі, наближеному до реального часу. Взаємодія між ботом та CRM повинна бути стабільною, а дані, що передаються, — правильними та структурованими, що дозволить зберігати повноцінну історію комунікацій.

Таким чином, буде сформовано повний набір функціональних та технічних вимог, на основі яких буде розроблено архітектуру майбутнього Telegram чат-бота. Визначені параметри створюють фундамент для наступних етапів проєктування, зокрема моделювання структури, розробки логіки роботи та інтеграційних механізмів із CRM-системою. Крім того, чітке визначення вимог дозволить уникнути помилок на етапах реалізації, забезпечити узгодженість між усіма компонентами системи та зробити бот більш надійним і зручним у використанні. Це також дасть можливість заздалегідь оцінити необхідні ресурси та визначити

послідовність виконання робіт, що значно підвищить ефективність всього процесу розробки.

2.2 Архітектура чат-бота

Архітектура Telegram чат-бота з інтеграцією в CRM-систему повинна забезпечувати стабільну роботу, швидку обробку запитів користувачів, масштабованість та безпечний обмін даними між усіма компонентами. Такий бот функціонує як частина більшого програмного комплексу, де ключовими елементами виступають модулі обробки повідомлень, логіка бізнес-процесів, інструменти взаємодії з базою даних та окремий блок для інтеграції зі сторонньою CRM.

Загалом архітектуру чат-бота можна описати як багаторівневу систему, де кожен рівень виконує свою чітку роль. Першим основним компонентом є Telegram API, який забезпечує прийом та передачу повідомлень. Через нього бот отримує текст, команди, натискання кнопок та інші події, що надходять від користувачів. Наступним елементом є обробник повідомлень – модуль, який інтерпретує дії користувача та визначає, який сценарій або функція повинні бути виконані.

У центральній частині архітектури знаходиться бізнес-логіка, яка реалізує всі можливості бота: реєстрацію користувачів, створення заявок, отримання інформації, обробку форм та маршрутизацію даних. Саме цей блок відповідає за послідовність дій після кожної взаємодії з користувачем.

Важливим елементом є модуль доступу до бази даних CRM, у якому зберігаються службові відомості про користувачів, активні сесії, дані про заявки, статуси та записи, необхідні для роботи сценаріїв. База даних дозволяє відстежувати історію взаємодій, підтримувати коректну роботу FSM (машини станів) та забезпечувати збереження інформації навіть у випадку перезапуску застосунку.

Окрему увагу займає блок інтеграції з CRM-системою. Цей модуль забезпечує зв’язок між ботом та зовнішньою платформою через API або Webhook. Він відповідає за передачу даних про користувачів, створення, оновлення чи

перегляд записів у CRM, синхронізацію статусів, а також обробку відповідей, які надходять від CRM назад у бот. Така інтеграція дозволяє автоматизувати рутинні бізнес-процеси: оформлення заявок, ведення клієнтської бази, контроль етапів обслуговування тощо (рис. 2.1).

Для стабільної роботи та можливості розширення система повинна мати рівень логування та моніторингу, який дозволяє контролювати помилки, швидкість обробки запитів, статистику використання та інші важливі показники. Це дає можливість швидко реагувати на збої та вдосконалювати роботу бота [11].

Завершальним елементом архітектури є інфраструктура розгортання, яка включає сервер, середовище виконання, налаштування вебхуків або механізму long polling. Для безперервної роботи необхідно використовувати хостинг з цілодобовою доступністю, підтримкою Python та можливістю масштабування у разі збільшення навантаження.

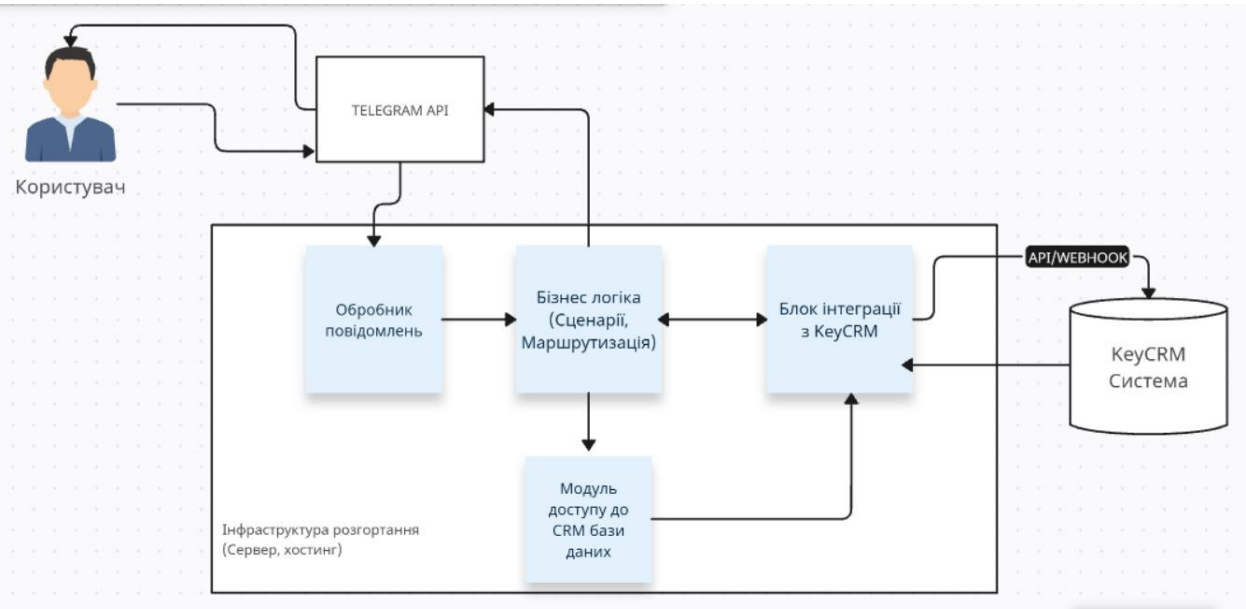


Рисунок 2.1 – Діаграма архітектури

Таким чином, архітектура чат-бота є комплексною системою, у якій кожен модуль відповідає за певну функцію. Завдяки правильному проєктуванню та інтеграції з CRM чат-бот здатний працювати стабільно, безпечно та виконувати широкий спектр бізнес-завдань, підвищуючи ефективність компанії та якість взаємодії з клієнтами.

2.3 Модель бізнес-процесів взаємодії чат-бота з CRM

Модель бізнес-процесів дозволяє зрозуміти, як саме Telegram чат-бот взаємодіє з CRM-системою в межах роботи підприємства. Таке моделювання дає змогу візуально відобразити послідовність дій, точки прийняття рішень, інформаційні потоки та ролі учасників процесу. Для цього зазвичай використовують нотації BPMN та UML, які зручні для опису як бізнес-процесів, так і технічної логіки системи.

Основним бізнес-процесом у взаємодії чат-бота з CRM є цикл обробки звернення клієнта від моменту першого повідомлення до фіксації результату в системі. У стандартній ситуації процес стартує з того, що користувач надсилає команду або повідомлення боту. Telegram API передає це звернення у бекенд системи, де модуль обробки запитів визначає тип дії: чи це питання, реєстрація, створення заявки, уточнення деталей або запит на інформацію.

У моделі BPMN це відображається як подія «Отримання повідомлення». Після цього система переходить до етапу «Аналіз контексту та визначення наміру». Тут працює логіка машини станів, яка визначає, у якому сценарії знаходиться користувач та яку саме дію необхідно виконати.

Подальший етап – це взаємодія з CRM-системою. Якщо користувач оформлює заявку або залишає контактні дані, модуль інтеграції надсилає запит у CRM через API (рис. 2.2). Це може бути створення нового ліда, оновлення картки клієнта, зміна статусу або отримання додаткової інформації. У BPMN-діаграмі це відображається як «Виклик зовнішнього сервісу». CRM відповідає системі або інформацією, або статусом обробки, який далі повертається у бот для відображення користувачеві.

З точки зору UML, механізм взаємодії можна описати у вигляді діаграми послідовності, де чітко показано, які об’єкти беруть участь: користувач, Telegram API, бот, база даних, CRM.

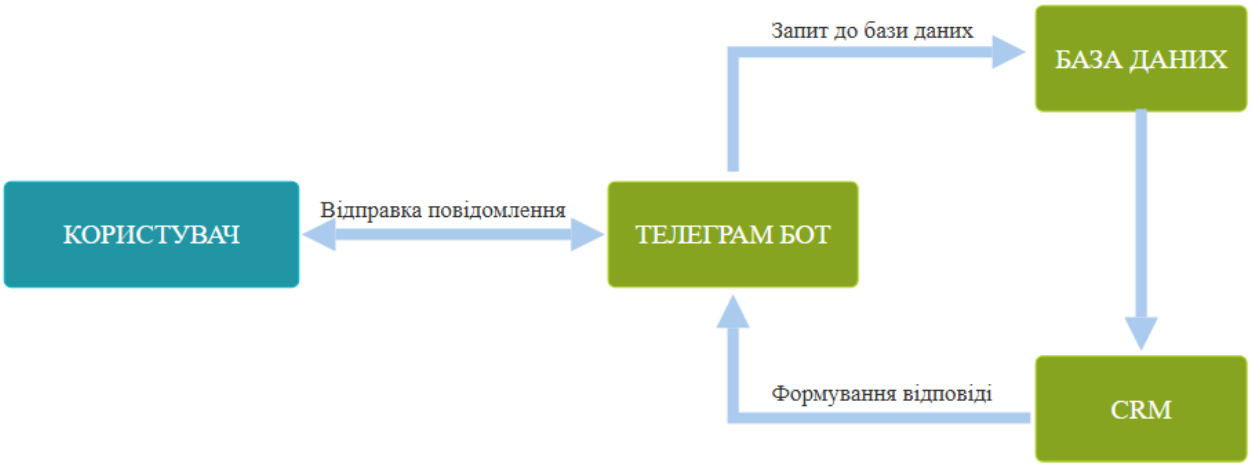


Рисунок 2.2 – Діаграма UML взаємодії користувача, Telegram-бота, бази даних та CRM

Діаграма UML також дозволяє показати можливі альтернативні сценарії: наприклад, коли CRM недоступна, бот може повідомити про технічні роботи або запропонувати альтернативний канал зв'язку. Усі ці варіанти враховуються у вигляді гілок на діаграмі послідовності (рис. 2.3).

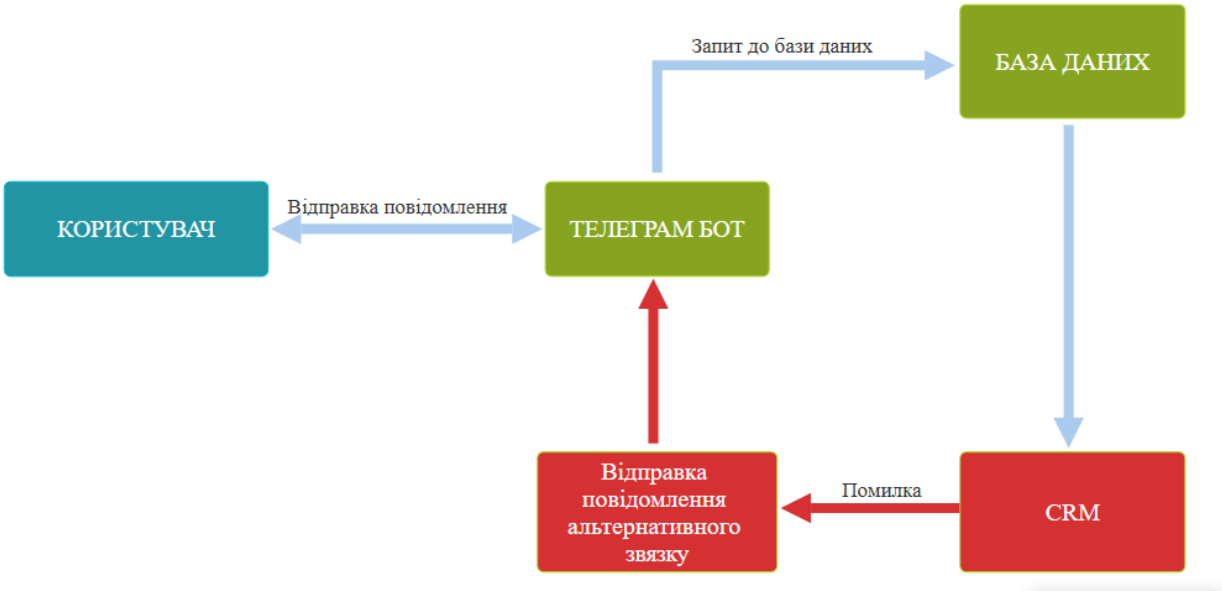


Рисунок 2.3 – Діаграма UML взаємодії користувача в разі помилки CRM системи

Ще один важливий аспект – це моделювання життєвого циклу заявки. Діаграма станів UML може показати, як змінюється статус звернення користувача:

«Створено», «Передано в CRM», «Опрацьовується», «Завершено». Це дає змогу краще зрозуміти логіку переходів між етапами та зв’язок між діями бота й оператора CRM.

Завдяки використанню BPMN та UML можна чітко визначити, які саме процеси має реалізувати система, як вони пов’язані між собою та які дані передаються між компонентами. Така модель дозволяє не тільки структурувати проєкт, але й полегшує його реалізацію, тестування та подальшу підтримку. Описані діаграми виступають своєрідною "дорожньою картою", яка забезпечує узгоджену та логічно побудовану взаємодію між Telegram чат-ботом і CRM-системою в рамках бізнес-процесів підприємства.

2.4 Проєктування структури даних та логіки бот–CRM

Проєктування структури даних та логіки взаємодії між Telegram-ботом і CRM-системою є одним із ключових етапів створення повноцінної інформаційної інфраструктури. На цьому етапі визначаються формати збереження даних, структура записів у базі, способи їх обробки, а також механізми передачі даних між ботом і CRM. Коректно спроєктована структура дозволяє забезпечити узгодженість інформації, швидку обробку запитів і стабільну роботу всього комплексу.

Основним завданням цього етапу є формування даних у такому вигляді, щоб і Telegram-бот, і CRM могли використовувати їх без додаткових перетворень. Зазвичай структура включає кілька сутностей: інформацію про користувача, історію звернень, активні сесії, дані заявок та службові журнали. У базі даних Telegram-бота повинні зберігатися такі ключові параметри, як Telegram ID користувача, його контактні дані, а також попередні запити.

Іншою важливою частиною структури є заявка або звернення, яке формується ботом і передається в CRM. Такий об’єкт включає інформацію, зібрану від користувача: ім’я, номер телефону, опис проблеми чи потреби, час створення та статус. Після передавання в CRM цей запис отримує унікальний ідентифікатор, який також зберігається в базі даних бота для подальшої синхронізації.

Логіка взаємодії бот–CRM базується на кількох типових операціях: створення нового запису, оновлення існуючих даних, запит інформації та отримання статусів. Бот формує структурований JSON-об’єкт і передає його через API CRM-системи. У разі успішної операції CRM обробляє інформацію, що містить ідентифікатор створеного запису або додаткову інформацію. Усі отримані дані зберігаються в базі для подальшої роботи бота (таб. 2.2).

Особлива увага приділяється обробці помилок та нестандартних ситуацій. Якщо CRM недоступна, бот повинен зберегти інформацію локально та ініціювати повторну спробу пізніше. Це забезпечує безперервність бізнес-процесів і запобігає втраті важливих даних. Також логіка взаємодії передбачає ведення журналу подій: кожен обмін даними між ботом і CRM фіксується разом із часом, результатом операції та можливими помилками. Журнал допомагає в діагностиці проблем і оптимізації роботи [24].

Таблиця 2.2 – Логіка взаємодії та бізнес-процеси

Етап / Процес	Механізм реалізації	Опис логіки
Формат обміну	API та JSON	Бот формує структурований JSON-об’єкт і передає його через API CRM-системи.
Типові операції	CRUD (Create, Read, Update)	• Створення нового запису (заявки) • Оновлення існуючих даних • Перевірка статусу заявки
Успішна операція	Відповідь від CRM	CRM повертає ідентифікатор (ID) створеного запису, який бот зберігає для подальшої синхронізації.
Обробка помилок	Локальне збереження + Retry	Якщо CRM недоступна, бот зберігає дані локально та ініціює повторну спробу пізніше (черга запитів).

Продовження таблиці 2.2

Масштабування	Модульна структура	Гнучка структура дозволяє додавати нові поля або сценарії без переробки всієї системи.

Правильно спроєктована структура даних дозволяє не лише ефективно реалізувати поточний функціонал, а й безболісно розширювати його у майбутньому. Додавання нових модулів, сценаріїв або полів у заявках не вимагає глобальних змін системи, якщо структура спершу була створена гнучкою та логічно організованою. Таким чином, етап проєктування структури даних та логіки взаємодії бот–CRM є фундаментом для побудови стабільної, масштабованої та ефективної системи.

2.5 Вибір програмних засобів (Python, фреймворк для бота, тип CRM)

Вибір програмних засобів є важливим етапом у проєктуванні Telegram чат-бота з інтеграцією в CRM-систему, оскільки від нього залежить продуктивність, масштабованість, стабільність та можливість подальшого розширення функціоналу. Правильно підібраний набір технологій забезпечує ефективну роботу системи та спрощує процес розробки.

Основною мовою програмування для реалізації чат-бота було обрано Python. Такий вибір зумовлений його простотою, наявністю великої кількості бібліотек, хорошою читабельністю коду та активною спільнотою розробників. Python також добре підходить для інтеграції з зовнішніми сервісами завдяки розвиненим можливостям роботи з HTTP-запитами та JSON-форматом, що є стандартом у взаємодії з більшістю CRM-систем.

Для створення Telegram-бота було обрано фреймворк python-telegram-bot, який є одним із найпопулярніших інструментів для розробки ботів на Python. Цей фреймворк працює на основі системи обробників (Handlers), що дозволяє чітко структурувати логіку програми, розділивши її на окремі компоненти — команди, текстові повідомлення, callback-запити та інші події. Бібліотека підтримує роботу

з усіма сучасними можливостями Telegram API: кнопками, інлайн-меню, медіафайлами, пересиланням локації, контактів, документів, а також дозволяє реалізовувати складні сценарії взаємодії з користувачем.

Важливою перевагою є наявність вбудованої машини станів (ConversationHandler), що значно спрощує створення послідовних діалогів – наприклад, коли користувач спочатку обирає товар, потім вводить номер телефону, а далі підтверджує замовлення.

Щодо вибору CRM-системи, ключовим критерієм було забезпечення доступу до API, можливість створення, оновлення та читання записів, а також підтримка вебхуків. Системи на кшталт KeyCRM, AmoCRM, Bitrix24 або Zoho CRM є найпопулярнішими для інтеграції з Telegram ботами, оскільки вони мають зрозумілу документацію, стабільні REST API та підтримують складні сценарії автоматизації. Конкретний вибір CRM залежить від бізнес-завдань: наприклад, Bitrix24 підходить для компаній із великою кількістю відділів і складним документообігом, тоді як KeyCRM більш орієнтована на роботу з лідами та продажами [9].

Таким чином, вибір програмних засобів формується на основі вимог проєкту, очікуваного навантаження та необхідної функціональності. Python у поєднанні з python-telegram-bot забезпечує гнучкість та швидкість розробки, а KeyCRM з API гарантує надійну інтеграцію та автоматизацію бізнес-процесів. Це створює міцну технічну основу для розробки ефективного та масштабованого Telegram чат-бота.

2.6 Сценарії взаємодії користувача з чат-ботом

Для ефективної роботи чат-бота важливо продумати сценарії взаємодії, тобто як саме користувач буде спілкуватися з ботом і які кроки виконуватиме під час роботи. Сценарії визначають логіку переходу між командами, відповіді бота та можливі варіанти поведінки користувача.

У системі передбачено такі основні сценарії:

1. Початок взаємодії

– Користувач відкриває чат з ботом і натискає кнопку "Start".

– Бот надсилає привітальне повідомлення та коротко пояснює свої можливості.

– Пропонується головне меню з основними розділами.

2. Реєстрація або авторизація користувача

– Користувач обирає пункт який йому потрібен.

– Бот запитує необхідні дані (ім’я, номер телефону, e-mail).

– Після успішної реєстрації бот створює або оновлює картку клієнта в CRM.

3. Задання запиту чи отримання консультації

– Користувач обирає опцію “Підтримка”, “Каталог товарів” або інший відповідний пункт.

– Бот пропонує вибрати тему або описати проблему текстом. Повідомлення зберігається у CRM як новий запит.

– Користувач отримує відповідь автоматично або від оператора

4. Оформлення замовлення / запис / створення заявки

– Користувач обирає необхідну послугу чи товар.

– Бот уточнює додаткові дані (Як звязатися з клієнтом для підтвердження даних).

– Після підтвердження бот реєструє замовлення в CRM і надсилає користувачу статус "створено".

5. Завершення взаємодії

– Користувач може повернутися до головного меню або завершити роботу командою /stop.

– Бот надсилає коротке прощальне повідомлення.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА TELEGRAM ЧАТ-БОТА

3.1 Особливості програмної реалізації модулів бота

Програмна реалізація чат-бота виконана мовою програмування Python 3.1x, яка сьогодні вважається одним із найзручніших та найпопулярніших інструментів для створення серверних застосунків. Python має велику кількість готових бібліотек, простий синтаксис і добре підходить для швидкої розробки та подальшої підтримки коду. Саме тому ця мова часто використовується як у комерційних проектах, так і в навчальних роботах.

Основою архітектури бота є асинхронний підхід (Asynchronous Programming). Це означає, що програма не виконує всі завдання послідовно й не чекає завершення кожної операції, а може працювати з кількома подіями одночасно. Завдяки цьому бот здатен обробляти велику кількість повідомлень паралельно, не перевантажуючи основний потік виконання. Асинхронність дозволяє ефективно працювати навіть з тисячами користувачів, забезпечуючи швидку реакцію бота на кожен запит без затримок та «підвисань» [3].

Крім того, такий підхід зменшує навантаження на сервер і робить систему більш гнучкою та масштабованою. У разі збільшення кількості користувачів немає потреби кардинально змінювати архітектуру – достатньо додати додаткові ресурси або оптимізувати окремі модулі.

Додатково в роботі використовується архітектура модульного типу, коли логіка бота поділена на окремі файли: обробники повідомлень, функції взаємодії з API, налаштування тощо. Такий підхід робить код більш зрозумілим, дозволяє легко знаходити й виправляти помилки, а також спрощує розширення функціоналу. У майбутньому можна без проблем додати нові команди, інтеграції або адміністративний інтерфейс [4].

Для взаємодії з Telegram Bot API застосовується спеціальний фреймворк Telebot, який значно спрощує роботу з запитами, повідомленнями, inline-кнопками й станами користувачів. Завдяки цьому більшість рутинних операцій автоматизовані, а розробник може зосередитись на реалізації бізнес-логіки.

Таким чином, програмна реалізація чат-бота поєднує сучасні підходи до серверної розробки, асинхронну архітектуру та гнучку модульну структуру, що забезпечує надійність, масштабованість та зручність подальшого розвитку системи.

3.1.1 Вибір бібліотек та інструментарію

Для реалізації функціонала чат-бота було підібрано набір бібліотек, які забезпечують стабільну роботу системи, простоту розробки та можливість подальшого масштабування. Основна ідея полягала в тому, щоб створити гнучку структуру, яку можна розширювати без суттєвої переробки всього проєкту.

1. python-telegram-bot (версія 20+)

Це головна бібліотека, на основі якої реалізована взаємодія між ботом і Telegram. Дана версія повністю підтримує асинхронний режим роботи (через синтаксис async/await), що дає змогу обробляти багато повідомлень одночасно.

Важливою перевагою є наявність ConversationHandler – механізму, який дозволяє будувати складні сценарії діалогів, розбиваючи їх на окремі стани. Завдяки цьому логіка спілкування з користувачем стає набагато зрозумілішою: кожен етап (отримання даних, уточнення інформації, підтвердження) відокремлений у власний блок коду.

Також ця бібліотека має добре структуровану документацію і активну спільноту розробників, що спрощує вирішення можливих проблем під час написання коду.

2. Requests

Для надсилання HTTP-запитів до зовнішнього API CRM-системи (у нашому випадку – KeyCRM) використовується бібліотека requests. Вона дозволяє легко виконувати запити типу GET, POST, передавати параметри, заголовки, JSON-дані та аналізувати відповіді серверу. Бібліотека проста у використанні, має мінімалістичний інтерфейс і не потребує складних налаштувань. Це робить її оптимальним вибором для інтеграції чат-бота з будь-якими сторонніми сервісами [12].

3. Logging

Модуль logging використовується для фіксації інформації про роботу бота. Логи допомагають контролювати такі процеси:

– надсилання та отримання даних;

– стан з’єднання з CRM-системою;

– помилки під час виконання команд;

– події, пов’язані з діями користувачів.

Завдяки логуванню система стає більш прозорою, а процес виправлення помилок – значно простішим. У разі збою лог-файли дозволяють швидко знайти причину проблеми та відновити роботу сервісу. Крім того, логування є важливим етапом під час розгортання бота на хостингу, оскільки дозволяє стежити за його роботою в реальному часі.

4. Інші допоміжні бібліотеки

У проєкті також використовуються додаткові стандартні модулі Python, такі як:

– json – для серіалізації та десеріалізації даних;

– os – для роботи з системними змінними середовища (наприклад, зберігання токенів у захищеному вигляді);

– asyncio – як основа асинхронної логіки програми.

Комбінація цих бібліотек забезпечує не лише коректну взаємодію з Telegram і CRM, а й стабільну роботу всієї архітектури чат-бота.

3.1.2 Реалізація логіки діалогу

Однією з ключових задач у розробці Telegram-ботів, що працюють у сфері електронної комерції, є збереження контексту розмови з користувачем. Це необхідно для того, щоб бот «розумів», у якому місці сценарію діалогу знаходиться користувач, яку дію він виконує та які дані очікуються на наступному кроці. Наприклад, якщо користувач обрав товар, то бот має пам’ятати цей вибір до моменту отримання номера телефону або підтвердження замовлення.

Для вирішення цієї задачі було використано механізм станів діалогу, реалізований через клас ConversationHandler бібліотеки python-telegram-bot. Даний інструмент працює за принципом скінченного автомата (Finite State Machine), де кожен стан визначає конкретний етап взаємодії, а переходи між станами виконуються на основі отриманих подій: натискання кнопок, введення тексту, надсилання контактів тощо.

У створеному боті виділено декілька основних станів, що відповідають логічним етапам взаємодії:

– MENU – головне меню, яке відображає стартові кнопки та дає можливість перейти до каталогу, перевірити статус замовлення або зв’язатися з менеджером.

– CATALOG – стан, у якому користувач обирає категорії або конкретні позиції. Тут працюють callback-кнопки, що дозволяють швидко навігувати між елементами каталогу.

– LEAD_WAIT_PHONE – стан, у якому бот очікує на номер телефону користувача. Після вибору товару бот пропонує залишити контактні дані, щоб менеджер міг зв’язатися для уточнення замовлення.

– LEAD_WAIT_TEXT – очікування текстового повідомлення, яке користувач надсилає.

– STATUS_WAIT_NUMBER – стан для введення номера замовлення, який використовується для перевірки його статусу через API CRM-системи.

Перехід між станами здійснюється за допомогою обробників подій – CallbackQueryHandler (для натискання кнопок у меню та каталозі) та MessageHandler (для текстових повідомлень і надсилання контактів). Кожен обробник визначає, яка саме подія трапилася, і переводить бота у відповідний наступний стан.

Такий підхід дозволяє структурувати логіку роботи, уникнути плутанини між різними гілками сценарію та забезпечити більш природну взаємодію користувача з ботом. Завдяки чітко визначеним станам діалог виглядає послідовним і контрольованим, що особливо важливо у комерційних застосунках, де кожний крок користувача повинен бути передбачуваним та коректно обробленим [5].

3.2 Налаштування та створення Telegram чат-бота, інтеграції та обміну даними з CRM

Інтеграція розробленого Telegram-бота з системою KeyCRM реалізована за архітектурним стилем REST (Representational State Transfer). Обмін даними відбувається через захищений протокол HTTPS з використанням формату JSON для серіалізації даних. Це забезпечує високу швидкість передачі інформації та універсальність взаємодії між сервером бота та хмарною CRM.

Процес налаштування інтеграції можна розділити на три етапи: конфігурація на стороні KeyCRM, налаштування транспортного рівня (API) та реалізація логіки синхронізації даних.

Попереднє налаштуваннння та конфігурація KeyCRM

Для коректної роботи програмного комплексу була виконана попередня конфігурація облікового запису в системі KeyCRM, оскільки саме ця CRM відповідає за зберігання товарів, обробку замовлень та взаємодію з API (рис. 3.1). Першим етапом стало створення облікового запису адміністратора на платформі KeyCRM. Після реєстрації та підтвердження електронної пошти система відкриває доступ до адміністративної панелі, де здійснюються усі подальші налаштування.

Окрім цього, необхідно активувати всі модулі та інструменти, які будуть задіяні під час інтеграції з Telegram-чат-ботом. Сюди входить робота з товарами, замовленнями, клієнтами та доступ до API. Кожен із цих модулів відповідає за свій набір функцій і безпосередньо впливає на те, як бот взаємодіятиме з CRM.

Наприклад, модуль товарів дозволяє боту отримувати актуальну інформацію про наявність продукції, ціни та характеристики. Модуль замовлень забезпечує можливість автоматично створювати нові заявки, фіксувати вибрані товари та відстежувати статус виконання. Доступ до API виступає як міст між ботом та CRM-системою, завдяки чому всі дані передаються швидко й без помилок.

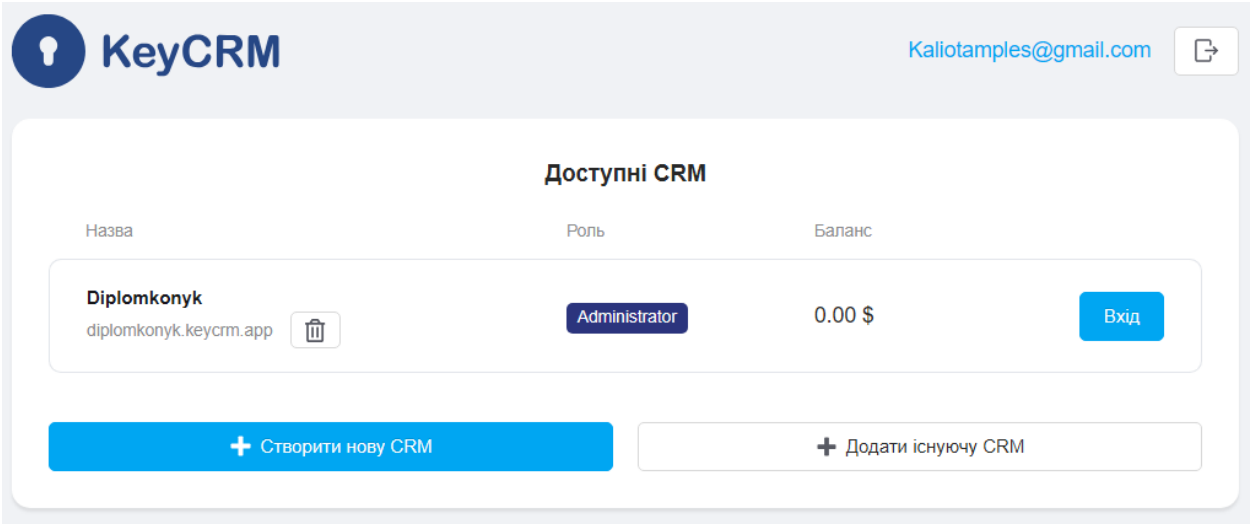


Рисунок 3.1 – Створення CRM панелі з ролью адміністратора

У панелі управління було проведено базову підготовку каталогу товарів: додано необхідні категорії, створено картки товарів з описами, зображеннями та цінами (рис. 3.2). Це важливо, оскільки чат-бот у подальшому отримує інформацію саме з цього каталогу через API, а користувачеві відображаються дані, які зберігаються у CRM [10].

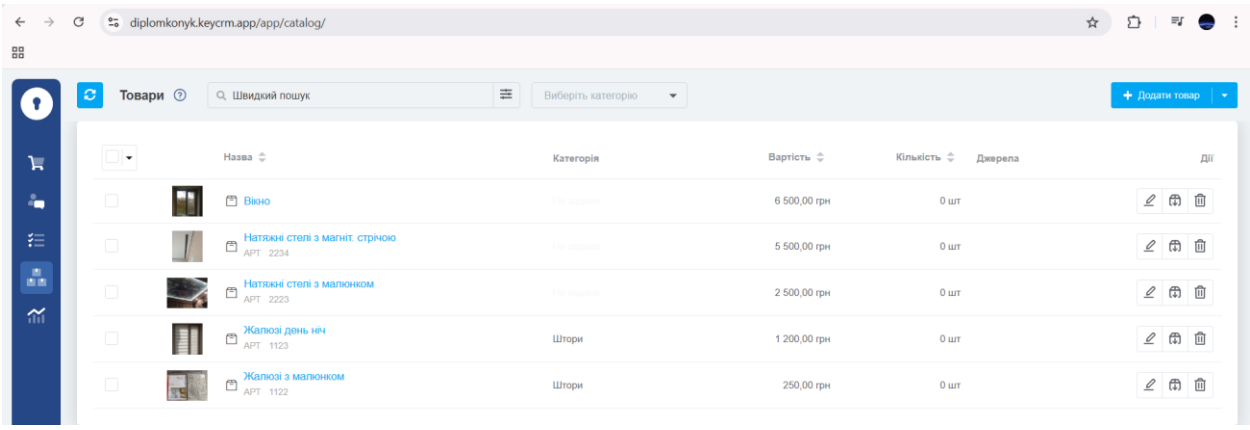


Рисунок 3.2 – Додавання товару на панель CRM

Наступним кроком стало налаштування API-доступу. У розділі інтеграцій було згенеровано унікальний API-ключ, який використовується для авторизації запитів із боку чат-бота (рис. 3.3). Цей ключ дозволяє системі ідентифікувати джерело запиту і гарантує, що доступ до даних отримує саме авторизований сервіс. Також у профілі компанії були перевірені та за потреби оновлені параметри

профілю: валюта, логотип компанії та інші поля, що можуть впливати на формування замовлення через бота.

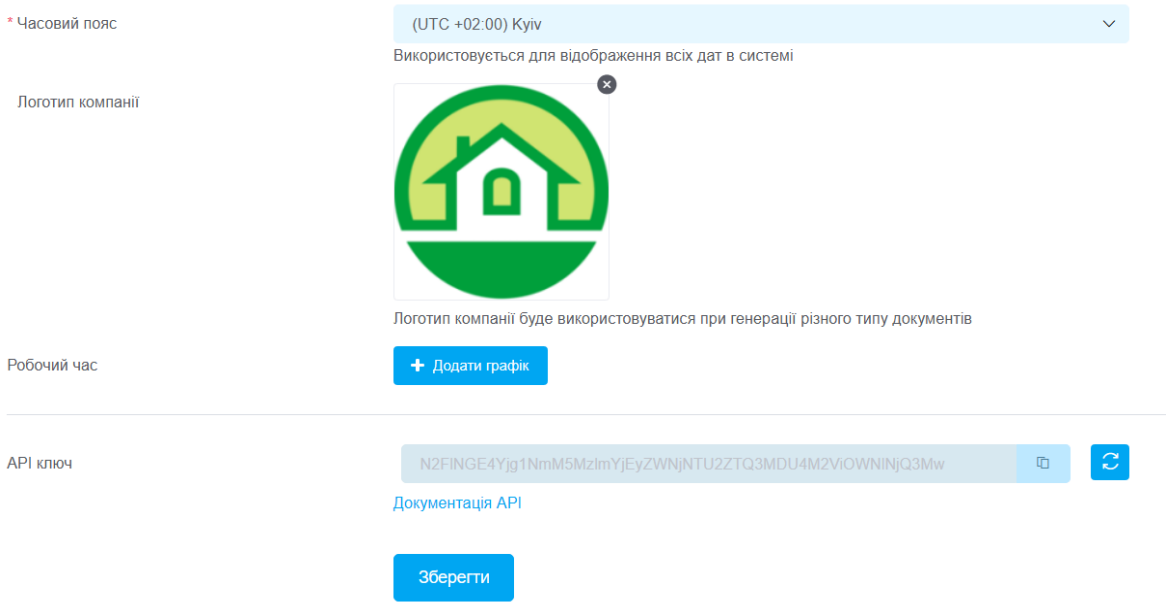


Рисунок 3.3 – Налаштування профілю та створення унікального API ключа в KeyCRM

Після завершення налаштувань API-ключ був внесений у конфігураційний файл бота, що забезпечило можливість виконання запитів до KeyCRM: отримання списку товарів, створення нового ліда, передавання контактних даних користувача та інші запити через бота. Завдяки цьому процес вибору товару в Telegram до створення замовлення у CRM – відбувається автоматично та без необхідності ручного втручання адміністратора.

3.3 Створення Telegram чат-бота

Для того щоб інтегрувати чат-бота з платформою Telegram, необхідно попередньо створити окремий бот-акаунт. Ця процедура виконується за допомогою офіційного сервісного бота BotFather, який відповідає за реєстрацію нових ботів та керування їх налаштуваннями

Процес створення бота складається з кількох кроків:

1. Пошук BotFather у Telegram. У полі пошуку достатньо ввести назву BotFather та відкрити офіційний профіль

2. Створення нового бота командою /newbot. Після натискання кнопки "Start" необхідно виконати команду /newbot (рис. 3.4). BotFather запропонує ввести: назву бота – відображається в списку чатів користувачів; унікальний username – повинен закінчуватися на bot (наприклад, mystore_helper_bot).

3. Отримання токена доступу (API Token). Після успішного створення бота BotFather генерує унікальний токен: 8310434855:AAFgnwTsiBpneXFjcdaQpAZdWMuZEvlxH9U. Цей токен використовується у Python-коді для підключення до Telegram Bot API. Він був збережений у конфігураційному файлі проекту та передається в програму при запуску.

4. Перевірка працездатності бота. Після налаштування бот стає доступним за посиланням виду http://t.me/KONYKSPCRM_bot. На цьому етапі можна протестувати, чи бот відповідає на команду /start. (Можна додати зображення першого запуску бота.)

Створений бот після цього готовий до інтеграції з програмною частиною. З отриманим токеном Python-скрипт може виконувати обробку команд, надсилати повідомлення, працювати з кнопками та виконувати логіку, описану у програмному модулі.

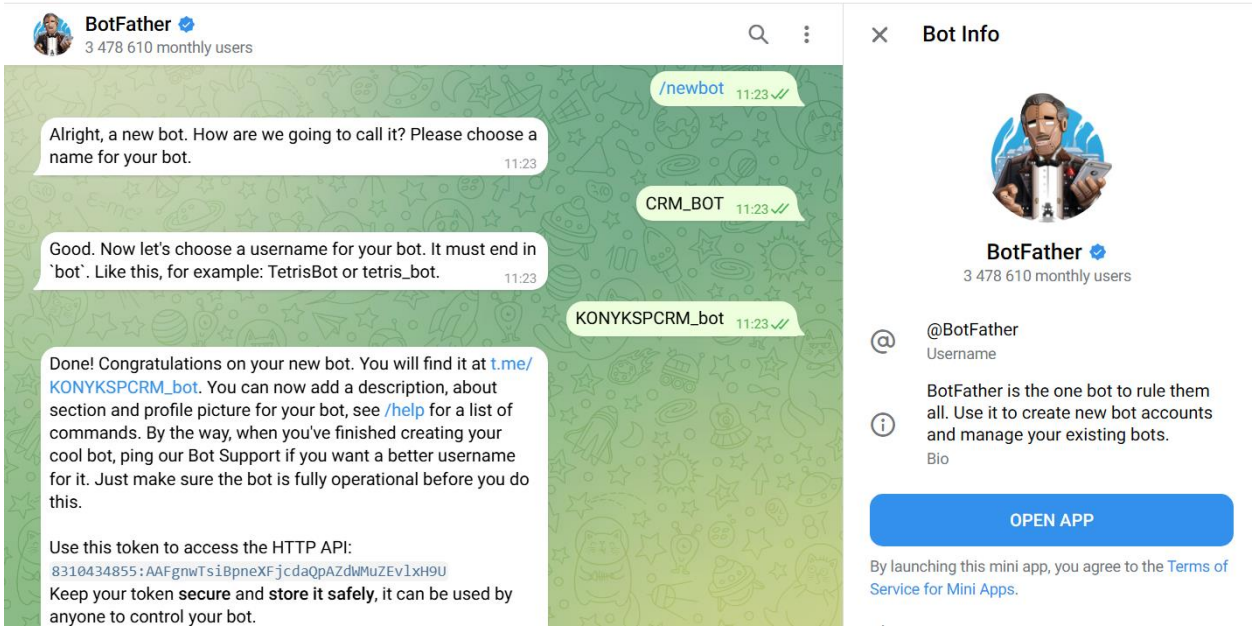


Рисунок 3.4 – Створення Telegram чат-бота та отримка унікального API ключа

Крім того, токен забезпечує можливість підключення зовнішніх сервісів, таких як CRM-система, дозволяючи боту передавати дані у KeyCRM або отримувати інформацію з каталогу товарів. Усі подальші програмні модулі бота базуються саме на цьому механізмі автентифікації, що робить токен ключовим елементом у всьому процесі інтеграції та обміну даними.

Після отримання посилання ми зможемо переглянути створеного чат-бота (рис.3.5). На даному етапі він ще не має активного функціоналу, команд або можливості обробляти інформацію. У подальшому цей бот відіграватиме ключову роль. Варто відзначити, що перший запуск будь-якого Telegram-бота здійснюється за допомогою команди /start. У нашому випадку відповіді від бота не буде, оскільки він ще не налаштований на обробку HTTP-запитів. Цю здатність він отримає пізніше після написання відповідного коду.

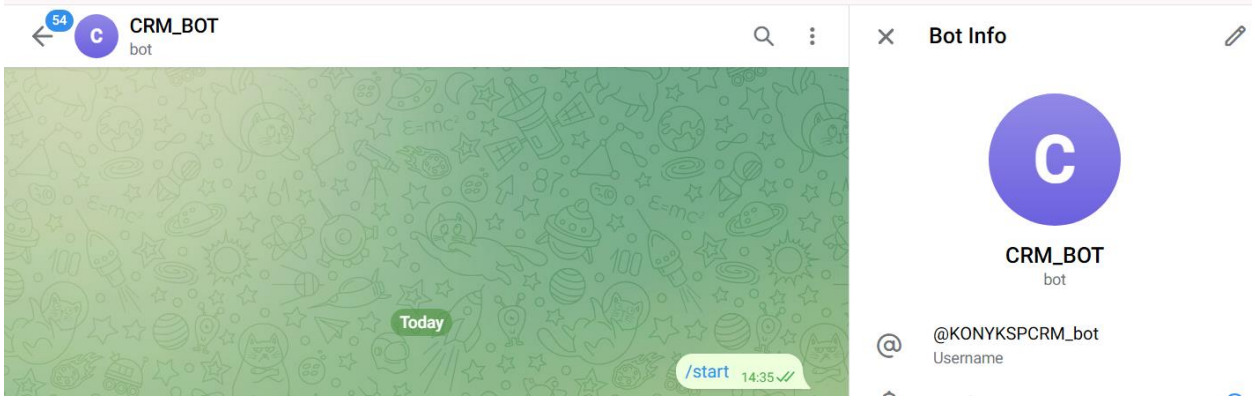


Рисунок 3.5 – Вікно чату з створеним чат-ботом

Під час огляду цього інформаційного блоку помітно, що він практично не відрізняється від звичайного чату з живим користувачем. Насправді, чатове вікно виконує роль для взаємодії з користувачем: увесь код, який виконується у середовищі розробки, обробляється та відображається у боті у вигляді текстових повідомлень, зображень або інтерактивних елементів, таких як кнопки. Таким чином, це специфічний інтерфейс для комунікації з програмою.

Водночас варто зазначити, що всі внутрішні зв’язки та механізми шифрування не пов’язані з протоколом безпеки Telegram. Для передачі відповідей на запити користувача використовується протокол HTTPS, що забезпечує

захищену передачу даних. Структура таких запитів виглядає наступним чином: api.telegram.org/bot’токен_бота’/назва_метода. Цей формат дозволяє боту отримувати повідомлення та надсилати відповіді, забезпечуючи ефективну взаємодію з користувачем через Telegram

Увага до візуального оформлення та професійний підхід до цього аспекту можуть суттєво впливати на залучення користувачів. Естетично продуманий дизайн бота здатен не лише викликати інтерес у користувачів, але й стимулювати їх до активної взаємодії. Тому при розробці чат-бота важливо враховувати не тільки його функціональні можливості, а й зовнішній вигляд, щоб забезпечити приємний та зручний досвід для користувачів.

Змінити дизайн чат-бота можна за допомогою спеціальних інструментів, таких як BotFather, використовуючи відповідні команди. Через BotFather можно налаштувати зовнішній вигляд бота, наприклад, змінити його ім’я, аватарку, опис та інструкції для користувача, а також додати інтерактивні елементи, які впливають на зручність взаємодії (рис. 3.6).

Рисунок 3.6 – Оновлення фотографії чат-бота

Після завершення налаштування робочого середовища та встановлення всіх необхідних бібліотек був створений головний файл у форматі “.py”. У цьому файлі були підключені основні бібліотеки та модулі, що забезпечують коректну роботу всіх компонентів і команд під час написання коду (рис. 3.7). Імпорт модулів здійснювався безпосередньо у середовищі розробки у вигляді стандартного коду Python. Такий підхід дозволив уникнути помилок під час компіляції та виконання програми, забезпечуючи стабільну роботу бота на всіх етапах розробки.

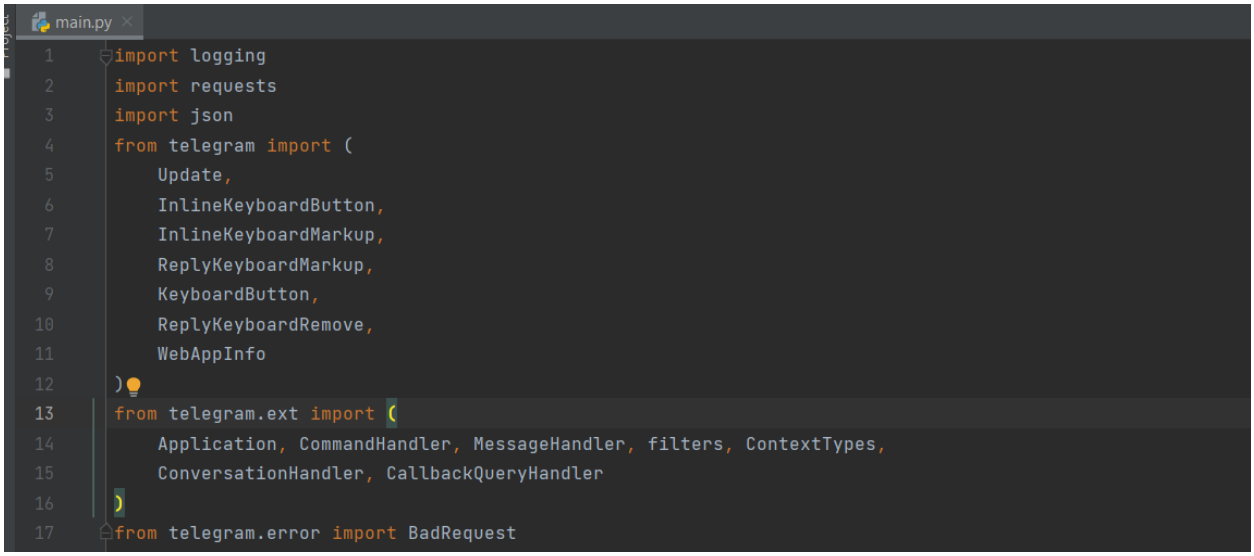


Рисунок 3.7 – Імпорт бібліотек в Python

Щоб чат-бот почав реагувати на повідомлення користувачів та міг пропонувати свої функції та послуги, необхідно створити команду "/start". Після її активації бот автоматично надсилає користувачеві меню з доступними варіантами взаємодії та інформаційними блоками.

Для роботи бота був доданий унікальний токен, який надається під час створення бота через BotFather (рис. 3.8). Цей токен забезпечує ідентифікацію саме цього бота та гарантує, що команди та запити, які генеруються в коді програми, обробляються правильно і надсилаються до потрібного облікового запису Telegram.

Крім того, для інтеграції бота з CRM-системою використовувався окремий ключ доступу до CRM. Він дозволяє боту безпечно обмінюватися даними з CRM, отримувати інформацію про клієнтів, фіксувати нові заявки та оновлювати статус

ліда. Завдяки цьому бот стає не лише інструментом комунікації з користувачем, а й важливою частиною автоматизованої системи управління клієнтськими даними, забезпечуючи точність та надійність роботи всієї екосистеми.

```
TG_TOKEN = "8388020438:AAE2t4jtLXhNPFlYNtIfLzAUjBBHufqtwkA"
KEYCRM_TOKEN = "N2FlNGE4Yjg1NmM5MzlmYjEyZWNjNTU2ZTQ3MDU4M2ViOWNlNjQ3Mw"
KEYCRM_SOURCE_ID = 1
```

Рисунок 3.8 – Інтеграція токенів Telegram та CRM

Після того як ключи доступу були додані до програмного середовища, з'являється можливість напряму взаємодіяти з ботом, використовуючи команди та функції, що надаються встановленими віртуальним середовищем бібліотеками. Це значно полегшує процес розробки, оскільки дозволяє звертатися до бота без додаткових налаштувань, а всі запити та повідомлення обробляються автоматично. Завдяки цьому бот може оперативно реагувати на дії користувачів, аналізувати отримані дані та формувати відповідні відповіді, що забезпечує стабільну й ефективну комунікацію в реальному часі.

Важливим етапом у розробці є підключення Telegram-бота до CRM-системи. Саме CRM виступає центральним сховищем даних, у якому зберігається інформація про користувачів, їхні звернення, замовлення та історія взаємодії. Інтеграція бота з CRM дозволяє забезпечити автоматизовану передачу даних та формування персоналізованих відповідей на основі вже накопиченої інформації.

Для повноцінної роботи чат-бота недостатньо лише запустити команду /start – важливо також організувати правильний механізм обробки всіх подальших дій користувача. Саме це забезпечує спеціальний модуль керування діалогами. Після першого звернення користувача бот не просто показує меню, а й переходить у певний «стан», у якому очікує наступну дію. Це дозволяє будувати логічні ланцюжки та спілкуватися з користувачем у кілька етапів.

Код із використанням ConversationHandler відповідає саме за це (рис. 3.9). Він визначає, у які стани може переходити бот та які типи повідомлень повинен обробляти в кожному з них [13]. Наприклад:

– після команди /start бот відкриває головне меню;

– у станах MENU та CATALOG бот реагує на натискання кнопок у меню;

– у стані LEAD_WAIT_PHONE бот очікує на контактний номер користувача та зберігає його в базу даних;

– у стані FEEDBACK_TEXT бот приймає текст відгуку;

– у стані LOCATION_CHECK бот може обробляти геолокацію або дозволяти користувачу скасувати дію.

```
def main():
    print("🚀 BOT STARTED...")
    application = Application.builder().token(TG_TOKEN).build()

    conv_handler = ConversationHandler(
        entry_points=[CommandHandler("start", start)],
        states={
            MENU: [CallbackQueryHandler(handle_menu_clicks)],
            CATALOG: [CallbackQueryHandler(handle_menu_clicks)],
            LEAD_WAIT_PHONE: [MessageHandler(filters.CONTACT, handle_contact)],
            FEEDBACK_TEXT: [MessageHandler(filters.TEXT & ~filters.COMMAND, handle_feedback)],
            LOCATION_CHECK: [
                MessageHandler(filters.LOCATION, handle_location),
                MessageHandler(filters.Regex('^(❌ Скасувати)$'), cancel_location)
            ]
        },
        fallbacks=[CommandHandler("start", start_over)]
    )

    application.add_handler(conv_handler)
    application.run_polling()
```

Рисунок 3.9 – Структура початку кода для взаємодії з користувачем

Таким чином, цей фрагмент коду забезпечує керування всіма можливими етапами взаємодії з користувачем: від стартового меню до отримання номера телефону, текстових відповідей чи геолокації. Завдяки цьому діалог стає структурованим, контрольованим і максимально зручним як для користувача, так і для подальшої обробки даних ботом.

Важливим етапом є отримання актуального каталогу товарів із CRM-системи. Для цього використовується спеціальна функція fetch_products_from_crm(), яка відправляє запит до API CRM та формує локальний список товарів для подальшого використання в боті. На початку функція встановлює необхідні заголовки, зокрема токен авторизації, після чого надсилає запит до ендпоінта /offers, що повертає перелік пропозицій разом із базовою

інформацією про товар. Далі код перебирає отримані дані, коректно опрацьовує кожен товар: витягує його ідентифікатор, ціну, артикул, назву, опис та зображення. Якщо деякі поля відсутні, функція обробляє це без помилок і пропускає некоректні елементи (рис. 3.10). У результаті формується словник products_map, у якому всі товари структуровано зберігаються у зручному форматі (назва, опис, фото, ціна, SKU). Ця функція відповідає за синхронізацію каталогу бота з CRM, забезпечуючи актуальні дані для відображення користувачам.

```
# пулінг товарів
def fetch_products_from_crm():
    print("\n🔄 Оновлюю каталог...")
    headers = {"Authorization": f"Bearer {KEYCRM_TOKEN}", "Accept": "application/json"}
    products_map = {}

    # Етап 1 /offers
    try:
        r = requests.get("https://openapi.keycrm.app/v1/offers", headers=headers,
                         params={"limit": 40, "include": "product"})
        if r.status_code == 200:
            for item in r.json().get('data', []):
                try:
                    p_id = str(item.get('id'))
                    price = float(item.get('price', 0.0))
                    sku = item.get('sku', '')

                    # Назва
                    info = item.get('product')
                    if info:
                        base = info.get('name', 'Товар')
                        var = item.get('name', '')
                        name = f"{base} {var}" if var and var != base else base
                        desc = info.get('description', '')  # Опис
                        img = info.get('thumbnail_url')  # Фото
                    else:
                        name = item.get('name', 'Товар')
                        desc = ""
```

Рисунок 3.10 – Приклад функції оброблення товару з CRM системи

Другим етапом у формуванні каталогу товарів є створення підвантаження інформації безпосередньо з ендпоінта /products CRM-системи. Цей блок коду виконується у тому випадку, якщо попередній запит до /offers не повернув жодної позиції. Спочатку відбувається виведення службового повідомлення про перехід до резервного джерела даних. Далі формується GET-запит до API /products, у якому передаються заголовки з токеном доступу та параметр limit, що обмежує кількість отримуваних товарів. Якщо CRM успішно відповідає, програма проходить по

всьому списку товарів та вилучає ключову інформацію: унікальний ID, назву, артикул, опис та URL зображення. Крім того, код перевіряє ціну товару, підставляючи мінімальну ціну або нуль, якщо основне значення відсутнє. Кожен елемент додається до словника products_map, у якому зберігається повний, структурований каталог. Усі можливі помилки під час обробки окремих товарів безпечно перехоплюються, що дає змогу уникнути зупинки роботи через некоректні дані (рис. 3.11). Після завершення обробки система виводить кількість отриманих позицій та повертає сформований каталог для подальшого використання чат-ботом [14].

```
# Етап 2: /products
if not products_map:
    print("⚠️  Offers пусто, пробую /products (чистий запит)...")
    try:
        # !!!!!!без include='offers', бо CRM блокує!
        r = requests.get("https://openapi.keycrm.app/v1/products", headers=headers, params={"limit": 40})
        if r.status_code == 200:
            for item in r.json().get('data', []):
                try:
                    p_id = str(item.get('id'))
                    name = item.get('name', 'Товар')
                    sku = item.get('sku', '')
                    desc = item.get('description', '')
                    img = item.get('thumbnail_url')

                    price = float(item.get('price') or item.get('min_price') or 0.0)

                    products_map[p_id] = {"name": name, "price": price, "sku": sku, "id": p_id, "desc": desc,
                                          "img": img}
                except:
                    pass
    except Exception as e:
        print(f"Error Products: {e}")

print(f"✅ Каталог: {len(products_map)} позицій.")
return products_map
```

Рисунок 3.11 – Реалізація резервного запиту до CRM API для отримання каталогу товарів

Наступним етапом розробки стало формування головного меню Telegram чат-бота за допомогою інтерактивних інлайн-кнопок. Це меню є одним із ключових елементів взаємодії користувача з ботом, адже саме через нього він може переходити до каталогу товарів, відкривати сайт, переглядати профіль, оформлювати доставку або залишати відгук. Для реалізації цього інтерфейсу був створений окремий блок коду (рис 3.12), який формує структуру клавіатури та

визначає дії, що виконуються після натискання відповідної кнопки. Кожна кнопка містить або callback-дані для подальшої обробки ботом, або пряме посилання на зовнішній ресурс, наприклад Instagram чи сайт магазину. Фрагмент коду нижче демонструє побудову цього меню та повертає готову розмітку клавіатури, яку бот надсилає користувачеві у відповідь на команду /start.

```
def kb_main_menu():
    keyboard = [
        [InlineKeyboardButton("🛒 Каталог товарів", callback_data='catalog')],
        [
            InlineKeyboardButton("🌐 Сайт", web_app=WebAppInfo(url=WEBSITE_URL)),
            InlineKeyboardButton("📸 Instagram", url=INSTAGRAM_URL)
        ],
        [
            InlineKeyboardButton("👤 Профіль", callback_data='profile'),
            InlineKeyboardButton("📍 Доставка", callback_data='delivery_calc')
        ],
        [
            InlineKeyboardButton("⭐ Відгук", callback_data='feedback'),
            InlineKeyboardButton("📞 Підтримка", url=MANAGER_USERNAME)
        ]
    ]
    return InlineKeyboardMarkup(keyboard)
```

Рисунок 3.12 – Створення кнопок взаємодії користувача з чат-ботом

Ще одним важливим етапом розробки стало створення та візуалізація інтерактивних кнопок, які формують зручний та привабливий інтерфейс Telegram-бота. Саме кнопки є ключовим елементом взаємодії користувача з системою, адже вони дозволяють швидко переходити між розділами, відкривати каталог товарів, переглядати профіль, залишати відгуки або переходити на зовнішні ресурси (рис. 3.13).

Щоб інтерфейс виглядав сучасно і заохочував користувача продовжувати взаємодію з ботом, кнопки були оформлені з використанням емодзі та логічної структуризації. Кожна кнопка має зрозумілу назву, яка підказує користувачеві її призначення, а також callback-дані або URL-посилання для виконання відповідної дії. Такий підхід робить роботу з ботом інтуїтивною та більш комфортною, зменшуючи кількість зайвих повідомлень і спрощуючи навігацію.

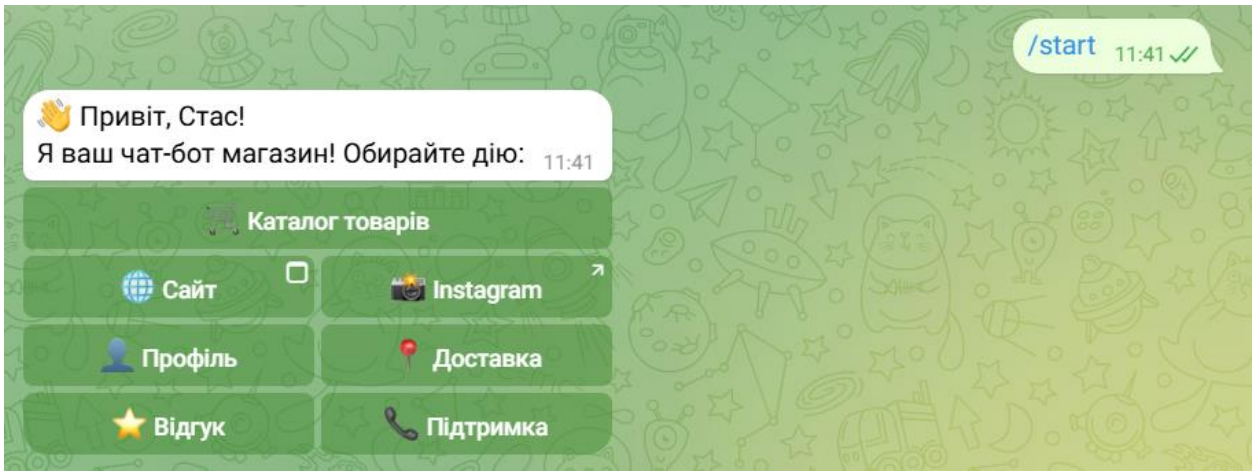


Рисунок 3.13 – Навігаційні кнопки в телеграм чат-боті

Після реалізації запитів до CRM та створенням кнопок для взяємодії корисувача з ботом наступним важливим кроком стало впровадження механізму автоматичного створення лідів (рис. 3.14), тобто передачі інформації, отриманої від користувача в Telegram-боті, безпосередньо в CRM-систему. Це дозволяє зберігати всі звернення, запити й потенційні замовлення у структурованому вигляді.

Створена функція не просто надсилає дані – вона спочатку визначає, який саме тип взаємодії відбувся: звичайний текстовий відгук від користувача, запит щодо доставки чи запит, пов’язаний із конкретною товарною позицією з каталогу. Залежно від цього формується відповідний заголовок, опис та зміст ліда, після чого інформація надсилається у CRM, де менеджер може оперативно її опрацювати.

```python
def create_crm_lead(user_data, product_id=None, text_msg=None, type="ORDER"):
    headers = {"Authorization": f"Bearer {KEYCRM_TOKEN}", "Content-Type": "application/json"}

    full_name = user_data.get('full_name', 'Client')
    phone = user_data.get('phone')
    username = user_data.get('username_tg', '')

    if type == "FEEDBACK":
        title = "★ Відгук з Telegram"
        comment = f"Клієнт залишив відгук:\n{text_msg}"

    elif product_id:
        product = CACHED_PRODUCTS.get(product_id)
        if product:
            title = f"Замовлення: {product['name']}"
            comment = (
                f"🛒 **ХОЧЕ КУПИТИ**\n"
                f"Товар: {product['name']}\n"
                f"Ціна: {product['price']} грн\n"
                f"Артикул: {product['sku']}\n"
                f"-----------------\n"
                f"Клієнт: @{username}"
            )
```

Рисунок 3.14 – Приклад створення заявки від користувача в вигляді ліда

Якщо користувач залишає відгук, система формує окремий запис із відповідним заголовком та текстом повідомлення (рис. 3.15).

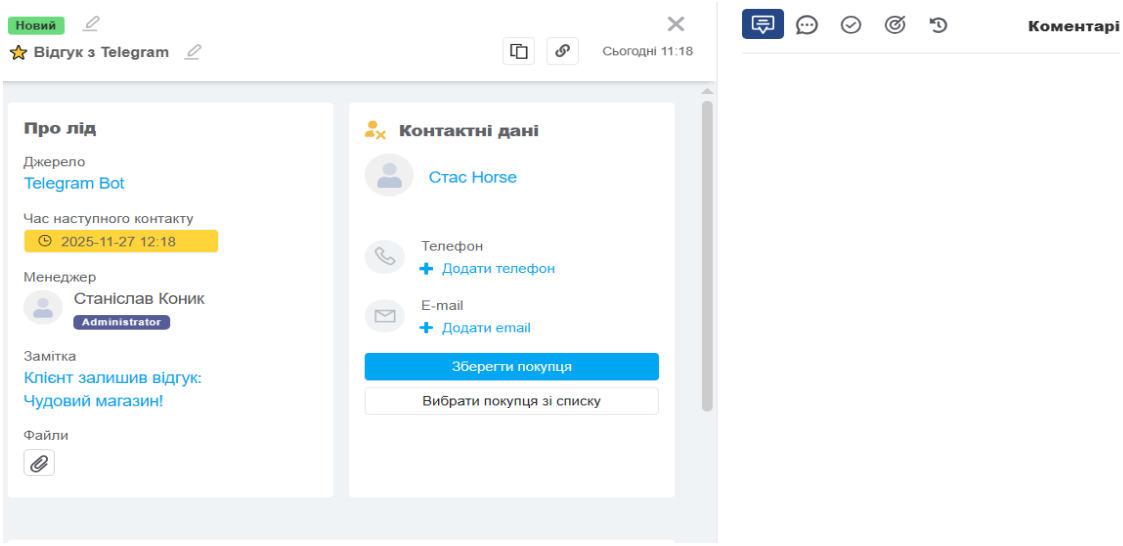


Рисунок 3.15 – Приклад ліда в який приходить відгук від клієнта

У випадку, коли в запит передається ідентифікатор товару, бот завантажує інформацію про цей продукт із кешованого каталогу та створює структурований опис замовлення, що включає назву товару, його ціну, артикул та контактні дані клієнта (рис.3.16).

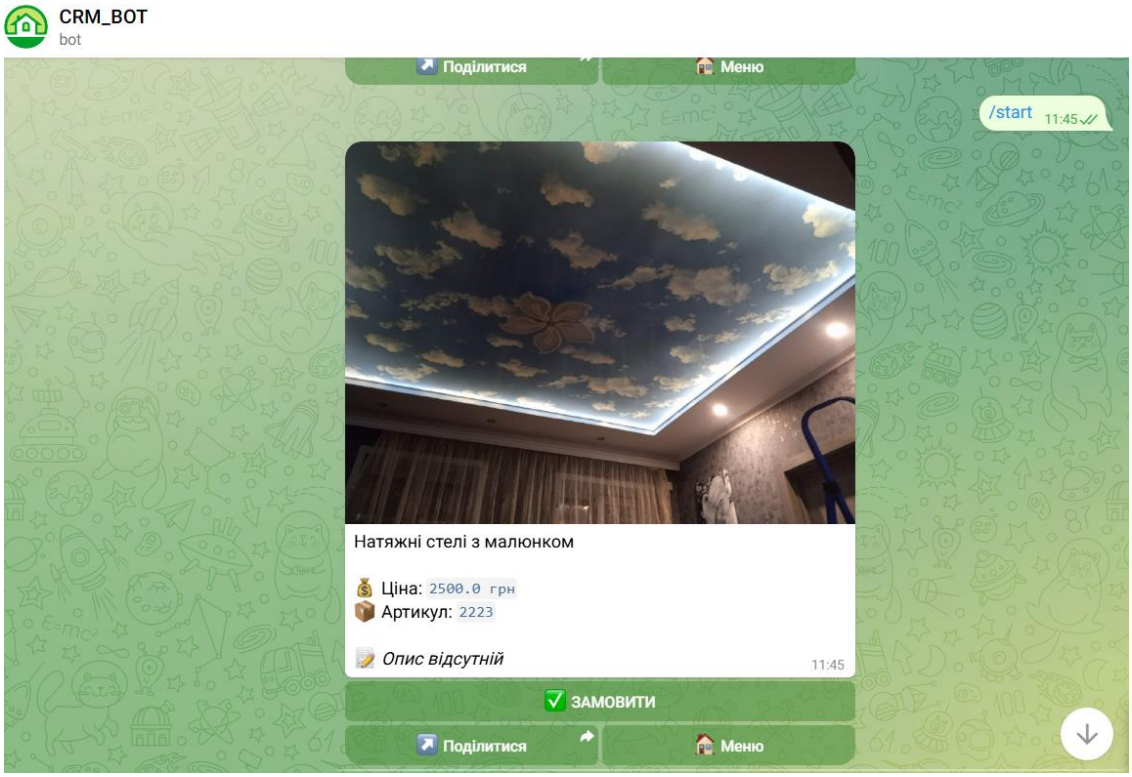


Рисунок 3.16 – Приклад товару який знаходиться в БД CRM

Після натискання користувачем інтерактивної кнопки «Замовити» відбувається відправлення даних на API CRM за допомогою HTTP POST-запиту. Для автоматизації збору даних користувач може натиснути кнопку «Поділитися номером», що спрощує отримання інформації від клієнта (рис. 3.17). Це дозволяє створити ліда автоматично, без будь-якого ручного втручання з боку менеджера. У результаті всі звернення користувачів із Telegram миттєво потрапляють у CRM у формі ліда, де їх можна опрацьовувати, призначати відповідальних, вести історію комунікацій та контролювати подальшу роботу з клієнтом.

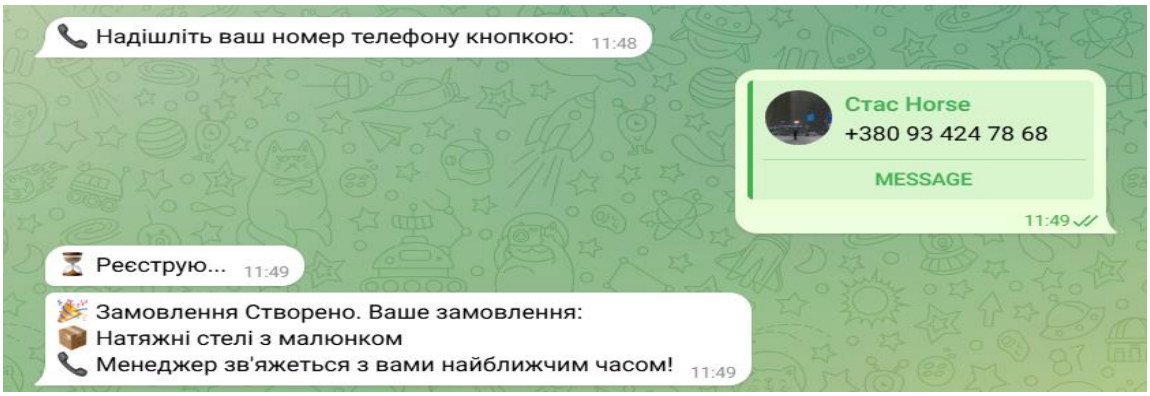


Рисунок 3.17 – Приклад створення замовлення в Telegram чат-боті

Такий підхід суттєво підвищує ефективність бізнес-процесів, оскільки мінімізує ризик втрати даних, пришвидшує обробку запитів і забезпечує повну синхронізацію між Telegram-ботом та CRM-системою (рис. 3.18).

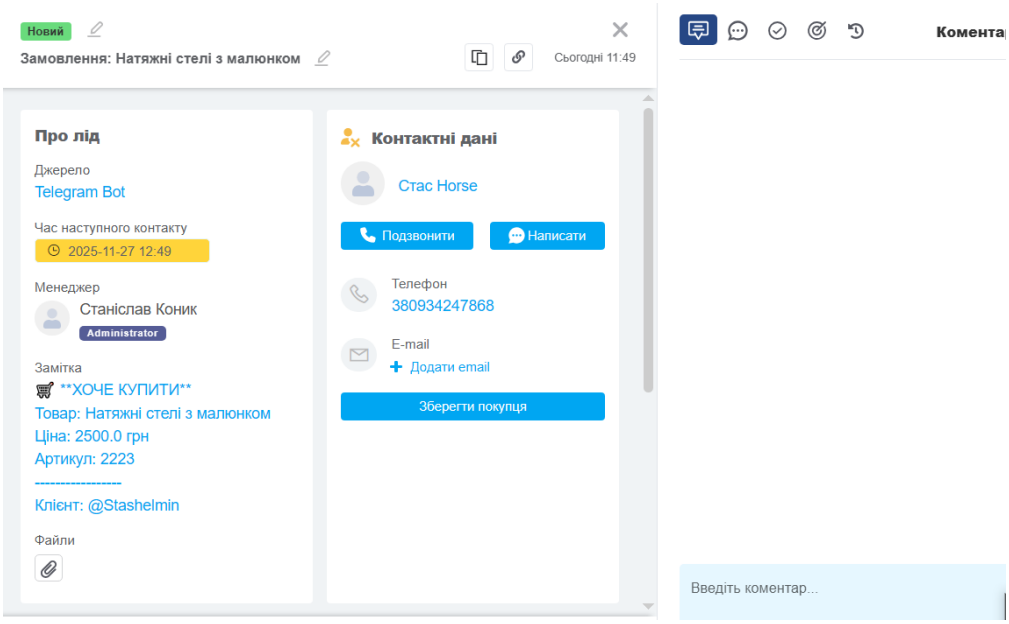


Рисунок 3.18 – Створений лід на KeyCRM з замовленим товаром

Після впровадження механізму автоматичного створення лідів менеджер отримує можливість оперативно зв’язатися з клієнтом для уточнення деталей замовлення або надання консультації. Це можна зробити як у телефонному режимі, так і за допомогою текстових повідомлень. Для зручності комунікації в самому ліді передбачена інтерактивна кнопка «Написати» (рис. 3.19), натискання на яку дозволяє менеджеру миттєво відкрити чат із клієнтом і розпочати обмін повідомленнями без необхідності шукати контактні дані окремо. Такий підхід спрощує взаємодію, економить час та забезпечує більш ефективну обробку замовлень і запитів клієнтів.

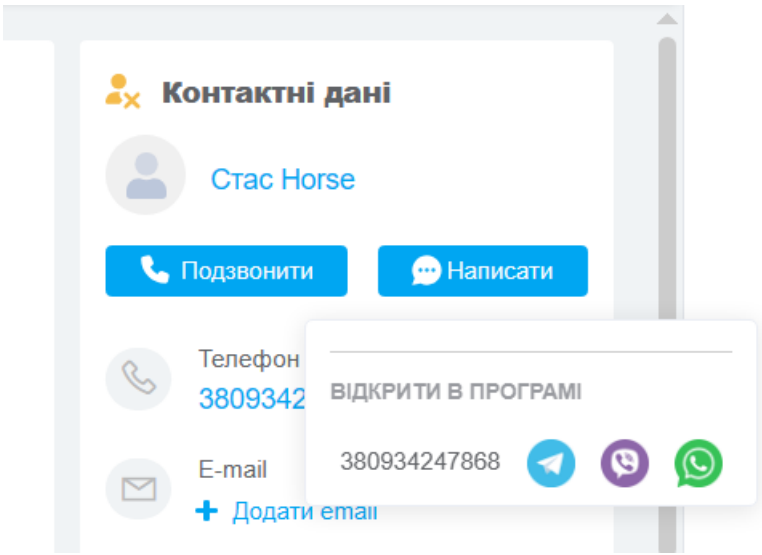


Рисунок 3.19 – Приклад звязку з клієнтом через CRM систему

Також у ліді реалізована функція додавання коментарів окремо від безпосередньої взаємодії з користувачем (рис. 3.20). Це дозволяє менеджерам фіксувати важливі деталі, уточнення або внутрішні замітки, що стосуються клієнта, без впливу на поточну комунікацію з ним. Завдяки цьому можна систематизувати інформацію про кожного клієнта, відокремлюючи внутрішні примітки від повідомлень, які отримує або надсилає сам клієнт. Такий підхід забезпечує зручну структуризацію даних, дозволяє вести повну і детальну історію взаємодій, а також підвищує загальну прозорість інформації для всієї компанії.

Наявність коментарів допомагає співробітникам швидко орієнтуватися у специфіці кожного клієнта, об’єднувати зусилля у командній роботі, планувати

подальші дії та приймати обґрунтовані рішення на основі актуальних і систематизованих даних. Крім того, це дозволяє аналізувати поведінку клієнтів, відслідковувати ефективність взаємодій і покращувати процес обслуговування, забезпечуючи більш високу якість підтримки та підвищення лояльності клієнтів.

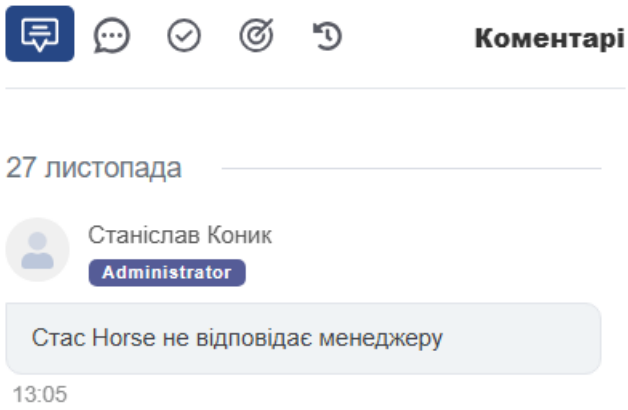


Рисунок 3.20 – Приклад додавання коментарів в ліді CRM

Таким чином, створений Telegram-чат-бот повністю інтегрований із CRM-системою та здатний автоматично обробляти всі запити користувачів, формувати ліди і передавати їх у базу даних CRM без ручного втручання менеджера. Це забезпечує оперативну обробку замовлень, зменшує ризик помилок і втрати даних, а також підвищує ефективність роботи компанії в цілому.

Інтерактивні кнопки, інтуїтивний інтерфейс і актуальний каталог товарів дозволяють користувачам легко взаємодіяти з ботом, швидко знаходити потрібну інформацію, оформляти замовлення або залишати відгуки. Водночас бот забезпечує структуроване збереження даних у CRM, що дає змогу менеджерам призначати відповідальних, відстежувати історію комунікацій та контролювати процес обробки кожного запиту.

Крім того, автоматична синхронізація з CRM і можливість обробки різних типів запитів від текстових повідомлень до контактних даних та замовлень з каталогу роблять бота ефективним інструментом не лише для комунікації з користувачами, але й для збору аналітичної інформації, оптимізації бізнес-процесів та підвищення рівня обслуговування клієнтів. Такий підхід дозволяє забезпечити

прозорість роботи, швидкий доступ до даних і гнучку адаптацію під потреби компанії, роблячи чат-бота важливою складовою сучасної системи управління клієнтськими даними.

3.4 Розгортання бота в робоче середовище

Для забезпечення безперервної роботи чат-бота необхідно розгорнути його на віддаленому сервері, який дозволяє виконувати Python-скрипт цілодобово, 24/7. Це гарантує, що бот завжди буде доступним для користувачів, своєчасно оброблятиме повідомлення та коректно передаватиме дані у CRM-систему.

Оптимальним рішенням є хмарний сервіс PythonAnywhere (рис. 3.20). Ця платформа надає безкоштовний тариф, підтримує роботу Python-додатків і дозволяє розгорнути скрипти без складних налаштувань серверної інфраструктури. PythonAnywhere також пропонує зручний веб-інтерфейс для керування проєктом, перегляду логів виконання, налаштування планувальника завдань і швидкого внесення змін у код без зупинки роботи бота [17].

Використання такого хмарного середовища забезпечує стабільність роботи бота, дозволяє ефективно тестувати та оновлювати функціонал, а також гарантує постійну доступність сервісу для користувачів. Це робить PythonAnywhere зручним та практичним рішенням для реалізації навчального проєкту з мінімальними витратами часу та ресурсів.

Рисунок 3.21 – Онлайн хостинг для розготання бота PythonAnywhere

Процес розгортання складається з декількох етапів:

1. Створення облікового запису (рис 3.22). На офіційному сайті PythonAnywhere створюється безкоштовний акаунт, який одразу відкриває доступ

до робочої консолі та файлової системи.

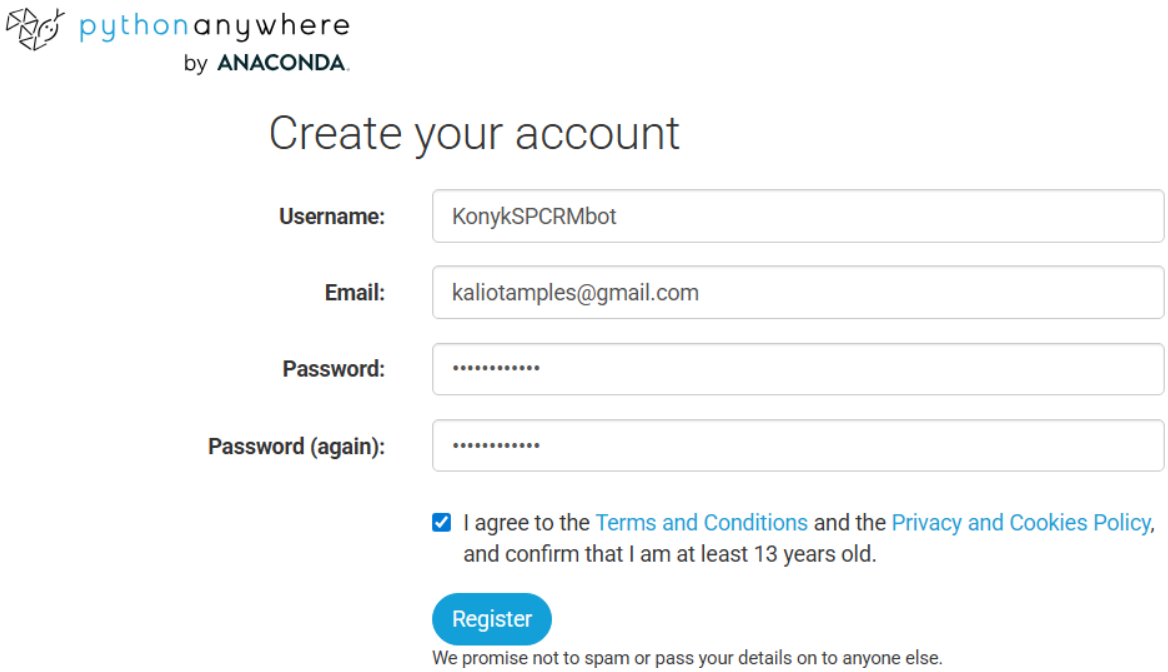


Рисунок 3.22 – Створення облікового запису на платформі PythonAnywhere

2. Завантаження файлів проєкту (рис. 3.23). В інтерфейсі PythonAnywhere є файловий менеджер, де можна завантажити всі python-скрипти, конфігураційні файли та залежності.

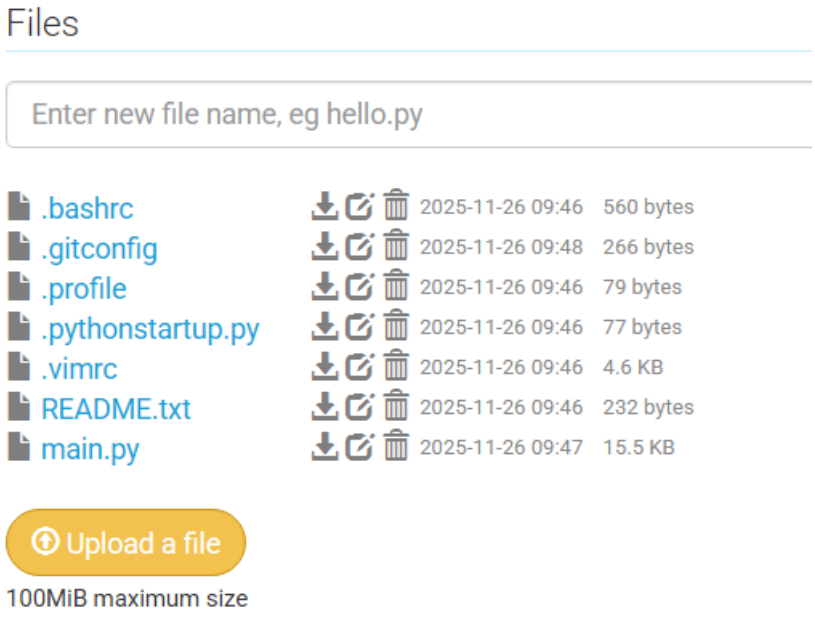


Рисунок 3.23 – Файловий менеджер PythonAnywhere

3. Запуск бота у режимі постійної роботи (рис. 3.24). На безкоштовному тарифі PythonAnywhere підтримує запуск скриптів у режимі Background Tasks. У цьому режимі бот працює постійно, отримуючи та обробляючи запити користувачів у реальному часі [23].

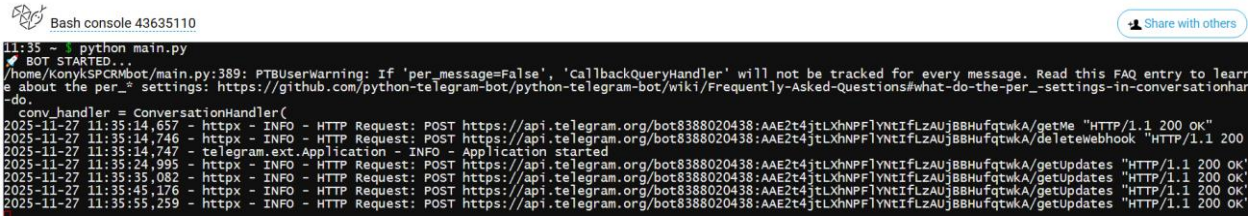


Рисунок 3.24 – Відображення Bash консолі в PythonAnywhere для запуску чат-боту

Використання PythonAnywhere забезпечує стабільну та безперервну роботу Telegram-бота без необхідності оренди фізичного сервера або складних DevOps-налаштувань. Такий варіант розгортання є повністю достатнім для малого бізнесу, навчальних проєктів, тестування та демонстрації працездатності програмного продукт

На рисунку 3.25 демонструється, що бот успішно запускається у робочому середовищі, коректно приймає запити від користувача та стабільно виконує закладені функції без помилок, що підтверджує правильність налаштування середовища та готовність системи до подальшої експлуатації.

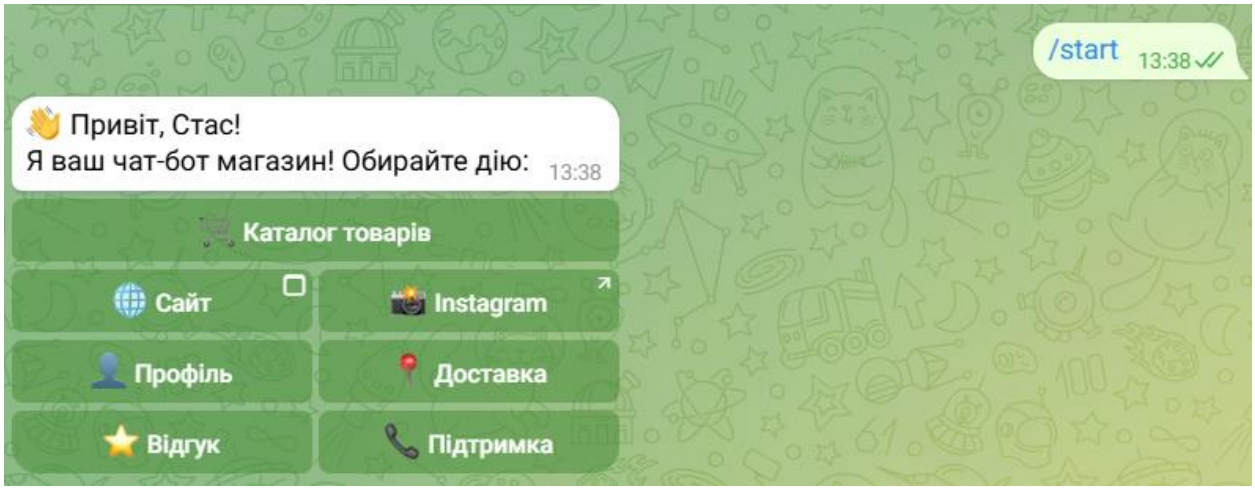


Рисунок 3.25 – Перевірка працездатності бота після розгортання його на сервісі.

ВИСНОВКИ

Метою даної роботи стало розробленно Telegram чат-бота на мові програмування Python з інтеграцією в CRM систему для автоматизації обробки звернень користувачів та оптимізації бізнес-процесів. У ході аналізу сучасних тенденцій було встановлено, що чат-боти відіграють ключову роль у цифровій трансформації компаній, забезпечуючи швидку комунікацію, зниження навантаження на операторів та підвищення якості сервісу.

У процесі дослідження вивчено можливості сучасних CRM-платформ та способи їх інтеграції з Telegram-ботами. Це дало змогу вибрати оптимальну CRM систему, яка підтримує роботу з API, дозволяє приймати замовлення, зберігати дані клієнтів та взаємодіяти з ботом у режимі реального часу. Створений бот автоматизує передавання даних у CRM, забезпечує формування лідів, обробку замовлень та фіксацію історії взаємодій. Інтеграція системи дозволила досягнути високого рівня автоматизації та мінімізувати ручну роботу менеджерів

Реалізація проекту здійснювалася з використанням Python та сучасного інструментарію для роботи з Telegram Bot API. Для забезпечення інтеграції застосовано HTTP-запити та модульну архітектуру, що дозволило створити гнучку та масштабовану систему. Усі функціональні модулі бота були протестовані як під час розробки, так і після розгортання на хостингу, включаючи перевірку API-запитів, стабільності роботи та коректності обміну даними з CRM.

У результаті успішної реалізації чат-бот став ефективним інструментом для взаємодії з клієнтами через Telegram, забезпечивши швидке оформлення замовлень, автоматичну передачу інформації до CRM та можливість моніторингу всіх процесів у режимі реального часу. Досягнуті результати підтверджують актуальність та практичну цінність розробленої системи, а також її потенціал для подальшого розширення та впровадження у різних бізнес-сферах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.Офіційна документація Python 3.12. [Електронний ресурс]- https://docs.python.org/3/

2.PEP 8 — Style Guide for Python Code. [Електронний ресурс] - https://peps.python.org/pep-0008/

3.AsyncIO: Asynchronous I/O. [Електронний ресурс] // Python Documentation. [Електронний ресурс] - https://docs.python.org/3/library/asyncio.html

4.Real Python. Async IO in Python: A Complete Walkthrough. [Електронний ресурс] - https://realpython.com/async-io-python/

5.Pydantic Documentation: Data validation using Python type hints. [Електронний ресурс] - https://docs.pydantic.dev/latest/

6.Telegram Bot API: Офіційна документація. [Електронний ресурс] [Електронний ресурс] - https://core.telegram.org/bots/api

7.Aiogram 3.x Documentation: Modern and fully asynchronous framework for Telegram Bot API. [Електронний ресурс] - https://docs.aiogram.dev/en/latest/8.

8.Python-telegram-bot Documentation. [Електронний ресурс] - https://docs.python-telegram-bot.org/en/stable/

9.What is a REST API? [Електронний ресурс] - https://www.redhat.com/en/topics/api/what-is-a-rest-api

10.KeyCRM API Documentation. [Електронний ресурс] - https://docs.keycrm.app/

11.Telegram Bot UI/UX Best Practices. [Електронний ресурс]. - https://core.telegram.org/bots/features

12.Бібліотека Requests: HTTP for Humans. [Електронний ресурс] - https://requests.readthedocs.io/en/latest/

13. ConversationHandler. [Електронний ресурс] - https://docs.python-telegram-bot.org/en/v21.8/telegram.ext.conversationhandler.html

14.HTTP Methods: GET vs. POST. [Електронний ресурс] - https://www.w3schools.com/tags/ref_httpmethods.asp

15.CRM-системи: що це таке і навіщо вони потрібні бізнесу. [Електронний ресурс] - https://dou.ua/

16.Webhooks vs. Polling. [Електронний ресурс] - https://www.google.com/search?q=https://zapier.com/blog/webhook-vs-polling/

17.PythonAnywhere Help & Documentation. [Електронний ресурс] - https://help.pythonanywhere.com/

18. Топ 10 найкращих CRM систем для України. [Електронний ресурс] - https://www.livebusiness.com.ua/ua/tools/crm/

19.Інтеграція CRM з іншими інструментами: як створити єдину екосистему .[Електронний ресурс] - https://cases.media/article/integraciya-crm-z-inshimi-instrumentami-yak-stvoriti-yedinu-ekosistemu?srsltid=AfmBOooc8ZkB9KSaeLzBYEknrT8Z5ZvGSs9sQUtagAsX77_TjhnFnX_I

20. NetHunt CRM: Інтеграція з месенджерами як інструмент омніканальності .[Електронний ресурс] - https://nethunt.ua/blog/intieghratsiia-z-miesiendzhierami/

21. CRM with Telegram: The Ultimate Guide to Integration. [Електронний ресурс] - https://controlhippo.com/blog/telegram/telegram-crm/

22.Introduction to JSON Web Tokens (JWT) and API Security. [Електронний ресурс] - https://jwt.io/introduction

23. How to Deploy a Telegram Bot to PythonAnywhere. [Електронний ресурс] – https://medium.com/

24. Building a Telegram Bot with Python and CRMsystem. [Електронний ресурс] - https://realpython.com/

ДОДАТОК А. ФРАГМЕНТИ ПРОГРАМНОГО КОДУ

Код програми main.py

```
import logging
import requests
import json
from telegram import (
    Update,
    InlineKeyboardButton,
    InlineKeyboardMarkup,
    ReplyKeyboardMarkup,
    KeyboardButton,
    ReplyKeyboardRemove,
    WebAppInfo
)
from telegram.ext import (
    Application, CommandHandler, MessageHandler, filters, ContextTypes,
    ConversationHandler, CallbackQueryHandler
)
from telegram.error import BadRequest

# кей апі
TG_TOKEN = "8388020438:AAE2t4jtLXhNPFlYNtIfLzAUjBBHufqtwkA"
KEYCRM_TOKEN =
"N2FlNGE4Yjg1NmM5MzlmYjEyZWNjNTU2ZTQ3MDU4M2ViOWNlNjQ3Mw"
KEYCRM_SOURCE_ID = 1

# Посилання на соц мережі
INSTAGRAM_URL = "https://instagram.com"
WEBSITE_URL = "https://google.com"
MANAGER_USERNAME = "https://t.me/Stashelmin"

# кеш
CACHED_PRODUCTS = {}

logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
level=logging.INFO)
logger = logging.getLogger(__name__)

# стани
MENU, CATALOG, LEAD_WAIT_PHONE, LEAD_WAIT_TEXT,
FEEDBACK_TEXT, LOCATION_CHECK = range(6)
```

```
# пулінг товарів
def fetch_products_from_crm():
    print("\n🔄 Оновлюю каталог...")
    headers = {"Authorization": f"Bearer {KEYCRM_TOKEN}", "Accept":
"application/json"}
    products_map = {}

    # Етап 1 /offers
    try:
        r = requests.get("https://openapi.keycrm.app/v1/offers", headers=headers,
                         params={"limit": 40, "include": "product"})
        if r.status_code == 200:
            for item in r.json().get('data', []):
                try:
                    p_id = str(item.get('id'))
                    price = float(item.get('price', 0.0))
                    sku = item.get('sku', '')

                    # Назва
                    info = item.get('product')
                    if info:
                        base = info.get('name', 'Товар')
                        var = item.get('name', '')
                        name = f"{base} {var}" if var and var != base else base
                        desc = info.get('description', '')  # Опис
                        img = info.get('thumbnail_url')  # Фото
                    else:
                        name = item.get('name', 'Товар')
                        desc = ""
                        img = None

                    products_map[p_id] = {"name": name, "price": price, "sku": sku, "id":
p_id, "desc": desc,
                                          "img": img}
                except:
                    pass
    except Exception as e:
        print(f"Error Offers: {e}")

    # Етап 2: /products
    if not products_map:
        print("⚠️ Offers пусто, пробую /products (чистий запит)...")
        try:
```

```
        # !!!!!!без include='offers', бо CRM блокує!
        r = requests.get("https://openapi.keycrm.app/v1/products", headers=headers,
params={"limit": 40})
        if r.status_code == 200:
            for item in r.json().get('data', []):
                try:
                    p_id = str(item.get('id'))
                    name = item.get('name', 'Товар')
                    sku = item.get('sku', '')
                    desc = item.get('description', '')
                    img = item.get('thumbnail_url')

                    price = float(item.get('price') or item.get('min_price') or 0.0)

                    products_map[p_id] = {"name": name, "price": price, "sku": sku, "id":
p_id, "desc": desc,
                                          "img": img}
                except:
                    pass
        except Exception as e:
            print(f"Error Products: {e}")

    print(f"✅ Каталог: {len(products_map)} позицій.")
    return products_map


# Створення ліда
def create_crm_lead(user_data, product_id=None, text_msg=None, type="ORDER"):
    headers = {"Authorization": f"Bearer {KEYCRM_TOKEN}", "Content-Type":
"application/json"}

    full_name = user_data.get('full_name', 'Client')
    phone = user_data.get('phone')
    username = user_data.get('username_tg', '')

    if type == "FEEDBACK":
        title = "⭐ Відгук з Telegram"
        comment = f"Клієнт залишив відгук:\n{text_msg}"

    elif product_id:
        product = CACHED_PRODUCTS.get(product_id)
        if product:
            title = f"Замовлення: {product['name']}"
            comment = (
```

```
            f"🛒 **ХОЧЕ КУПИТИ**\n"
            f"Товар: {product['name']}\n"
            f"Ціна: {product['price']} грн\n"
            f"Артикул: {product['sku']}\n"
            f"----------\n"
            f"Клієнт: @{username}"
        )
    else:
        title = "Замовлення (помилка)"
        comment = "Товар не знайдено в кеші"
else:
    title = "Питання з Telegram"
    comment = f"💬 Питання:\n{text_msg}\n@{username}"

lead_data = {
    "title": title,
    "source_id": KEYCRM_SOURCE_ID,
    "contact": {
        "full_name": full_name,
        "phone": phone,
        "social_name": f"@{username}" if username else ""
    },
    "manager_comment": comment
}

try:
    requests.post("https://openapi.keycrm.app/v1/leads", json=lead_data,
headers=headers)
    return True
except:
    return False


# Кнопки меню
def kb_main_menu():
    keyboard = [
        [InlineKeyboardButton("🛒 Каталог товарів", callback_data='catalog')],
        [
            InlineKeyboardButton("🌐 Сайт",
web_app=WebAppInfo(url=WEBSITE_URL)),
            InlineKeyboardButton("📸 Instagram", url=INSTAGRAM_URL)
        ],
        [
            InlineKeyboardButton("👤 Профіль", callback_data='profile'),
```

```
            InlineKeyboardButton("📍 Доставка", callback_data='delivery_calc')
        ],
        [
            InlineKeyboardButton("⭐ Відгук", callback_data='feedback'),
            InlineKeyboardButton("📞 Підтримка", url=MANAGER_USERNAME)
        ]
    ]
    return InlineKeyboardMarkup(keyboard)


def kb_catalog():
    keyboard = []
    global CACHED_PRODUCTS
    if not CACHED_PRODUCTS: CACHED_PRODUCTS =
fetch_products_from_crm()

    if not CACHED_PRODUCTS:
        keyboard.append([InlineKeyboardButton("⚠ Каталог порожній",
callback_data='menu')])
    else:
        for p_id, val in CACHED_PRODUCTS.items():
            name = (val['name'][:20] + '..') if len(val['name']) > 20 else val['name']
            btn = f"{name} • {val['price']} грн"
            keyboard.append([InlineKeyboardButton(btn, callback_data=f"view_{p_id}")])

    keyboard.append([InlineKeyboardButton("🔄 Оновити",
callback_data='refresh_catalog')])
    keyboard.append([InlineKeyboardButton("🏠 Назад", callback_data='menu')])
    return InlineKeyboardMarkup(keyboard)


def kb_product_actions(p_id, p_name):
    return InlineKeyboardMarkup([
        [InlineKeyboardButton("✅ ЗАМОВИТИ", callback_data=f"buy_{p_id}")],
        [
            InlineKeyboardButton("↗ Поділитися", switch_inline_query=p_name),
            InlineKeyboardButton("🏠 Меню", callback_data='catalog')
        ]
    ])


def kb_phone():
    return ReplyKeyboardMarkup(
```

```
        [[KeyboardButton("📱 Надіслати номер", request_contact=True)]],
        resize_keyboard=True, one_time_keyboard=True
    )


def kb_location():
    return ReplyKeyboardMarkup(
        [[KeyboardButton("📍 Надіслати геолокацію", request_location=True)],
[KeyboardButton("🔙 Скасувати")]],
        resize_keyboard=True, one_time_keyboard=True
    )


async def safe_edit(update, text, reply_markup):
    try:
        await update.callback_query.edit_message_text(text=text,
reply_markup=reply_markup, parse_mode='Markdown')
    except BadRequest:
        pass


# обробники функцій
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int:
    user = update.effective_user
    context.user_data['username_tg'] = user.username
    context.user_data['full_name'] = user.full_name

    await update.message.reply_text(
        f"👋 Привіт, **{user.first_name}**!\nЯ ваш чат-бот магазин! Обирайте дію:",
        reply_markup=kb_main_menu(),
        parse_mode='Markdown'
    )
    return MENU


async def handle_menu_clicks(update: Update, context:
ContextTypes.DEFAULT_TYPE) -> int:
    query = update.callback_query
    await query.answer()
    data = query.data

    if data == 'catalog':
        await safe_edit(update, "□ Завантажую...", InlineKeyboardMarkup([]))
        await safe_edit(update, "🛒 **Каталог товарів:**", kb_catalog())
```

```
    return CATALOG

elif data == 'refresh_catalog':
    global CACHED_PRODUCTS
    CACHED_PRODUCTS = {}
    await safe_edit(update, "□ Оновлення...", InlineKeyboardMarkup([]))
    await safe_edit(update, "🛒 **Каталог (оновлено):**", kb_catalog())
    return CATALOG

elif data.startswith('view_'):
    p_id = data.split('_')[1]
    product = CACHED_PRODUCTS.get(p_id)
    context.user_data['view_id'] = p_id

    # Перевірка на None, щоб не було помилки
    desc = product.get('desc') or "Опис відсутній"
    if len(desc) > 200: desc = desc[:200] + "..."

    text = (
        f"👟 **{product['name']}**\n\n"
        f"💰 Ціна: `{product['price']} грн`\n"
        f"🎁 Артикул: `{product['sku']}`\n\n"
        f"📝 _{desc}_"
    )

    await query.delete_message()

    if product.get('img'):
        try:
            await context.bot.send_photo(
                chat_id=update.effective_chat.id,
                photo=product['img'],
                caption=text,
                reply_markup=kb_product_actions(p_id, product['name']),
                parse_mode='Markdown'
            )
        except:
            await context.bot.send_message(update.effective_chat.id, text,
                                           reply_markup=kb_product_actions(p_id, product['name']),
                                           parse_mode='Markdown')
    else:
        await context.bot.send_message(update.effective_chat.id, text,
                                       reply_markup=kb_product_actions(p_id, product['name']),
                                       parse_mode='Markdown')
```

```
        return CATALOG

    elif data.startswith('buy_'):
        p_id = data.split('_')[1]
        context.user_data['selected_product_id'] = p_id
        context.user_data['is_order'] = True

        try:
            await query.delete_message()
        except:
            pass

        await context.bot.send_message(
            update.effective_chat.id,
            "📞 Надішліть ваш номер телефону кнопкою:",
            reply_markup=kb_phone()
        )
        return LEAD_WAIT_PHONE

    elif data == 'profile':
        user = update.effective_user
        info = f"👤 **Профіль:**\nID: `{user.id}`\nІм'я: {user.full_name}"
        await safe_edit(update, info, InlineKeyboardMarkup([[InlineKeyboardButton("🔙 Назад", callback_data='menu')]]))
        return MENU

    elif data == 'feedback':
        await safe_edit(update, "✍️ Напишіть ваш відгук:",
                        InlineKeyboardMarkup([[InlineKeyboardButton("🔙 Скасувати", callback_data='menu')]]))
        return FEEDBACK_TEXT

    elif data == 'delivery_calc':
        await query.delete_message()
        await context.bot.send_message(
            update.effective_chat.id,
            "📍 Надішліть геолокацію для розрахунку доставки (використайте кнопку):",
            reply_markup=kb_location()
        )
        return LOCATION_CHECK

    elif data == 'menu':
```

```
        try:
            await query.message.delete()
        except:
            pass
        await context.bot.send_message(update.effective_chat.id, "🏠 Меню:",
reply_markup=kb_main_menu())
        return MENU

    return MENU


async def handle_contact(update: Update, context: ContextTypes.DEFAULT_TYPE) ->
int:
    context.user_data['phone'] = update.message.contact.phone_number

    if context.user_data.get('is_order', False):
        p_id = context.user_data.get('selected_product_id')
        prod = CACHED_PRODUCTS.get(p_id)

        await update.message.reply_text("□ Реєструю...",
reply_markup=ReplyKeyboardRemove())

        if create_crm_lead(context.user_data, product_id=p_id):
            await update.message.reply_text(f"🎉 **Замовлення прийнято. Ваше
замовлення:**\n🎁 {prod['name']}\n📞 Менеджер зв'яжеться з вами найближчим
часом!",
                                            parse_mode='Markdown')
        else:
            await update.message.reply_text("❌ Помилка CRM.")

    await update.message.reply_text("🏠 Меню:", reply_markup=kb_main_menu())
    return MENU


async def handle_location(update: Update, context: ContextTypes.DEFAULT_TYPE) -
> int:
    await update.message.reply_text(
        f"📍 Локацію отримано!\n🚚 Доставка: ~80-120 грн (Нова Пошта)",
        reply_markup=ReplyKeyboardRemove()
    )
    await update.message.reply_text("🏠 Меню:", reply_markup=kb_main_menu())
    return MENU
```

```
async def cancel_location(update: Update, context: ContextTypes.DEFAULT_TYPE) -
> int:
    await update.message.reply_text("Скасовано.",
reply_markup=ReplyKeyboardRemove())
    await update.message.reply_text("🏠 Меню:", reply_markup=kb_main_menu())
    return MENU


async def handle_feedback(update: Update, context: ContextTypes.DEFAULT_TYPE)
-> int:
    text = update.message.text
    create_crm_lead(context.user_data, text_msg=text, type="FEEDBACK")
    await update.message.reply_text("⭐ Дякуємо за відгук!",
reply_markup=kb_main_menu())
    return MENU


async def start_over(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int:
    await start(update, context)
    return MENU


def main():
    print("🚀 BOT STARTED...")
    application = Application.builder().token(TG_TOKEN).build()

    conv_handler = ConversationHandler(
        entry_points=[CommandHandler("start", start)],
        states={
            MENU: [CallbackQueryHandler(handle_menu_clicks)],
            CATALOG: [CallbackQueryHandler(handle_menu_clicks)],
            LEAD_WAIT_PHONE: [MessageHandler(filters.CONTACT, handle_contact)],
            FEEDBACK_TEXT: [MessageHandler(filters.TEXT & ~filters.COMMAND,
handle_feedback)],
            LOCATION_CHECK: [
                MessageHandler(filters.LOCATION, handle_location),
                MessageHandler(filters.Regex('^(🔙 Скасувати)$'), cancel_location)
            ]
        },
        fallbacks=[CommandHandler("start", start_over)]
    )

    application.add_handler(conv_handler)
    application.run_polling()
```

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

ДОДАТОК Б. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра Комп'ютерних Наук

ДИПЛОМНА РОБОТА
на ступінь вищої освіти магістр
із спеціальності F3 Комп'ютерні науки

Telegram чат-бот на мові Python з інтеграцією в CRM систему

Виконав: студент 6 курсу, групи КНДм-61
Коник Станіслав Павлович
Керівник:професор кафедри Комп'ютерних наук
д.ф.-м.н., професор Шикула О.М

Київ - 2025

Слайд 1

Актуальність та Мета роботи

- Актуальність:

1. Стрімкий розвиток електронної комерції та перехід бізнесу в месенджери.
2. Необхідність автоматизації рутинних процесів обробки замовлень.
3. Важливість створення єдиного інформаційного середовища (Бот + CRM)

- Мета роботи:

1. Розробка Telegram чат-бота на мові Python із повноцінною інтеграцією в CRM-систему.
2. Автоматизація обробки звернень та структурування даних про клієнтів.

Слайд 2

Об’єкт, Предмет та Завдання

- Об’єкт дослідження:

  Telegram чат-бот та CRM система як інструменти автоматизації бізнесу
- Предмет дослідження:

  Процес розробки та функціонування чат-бота на Python з інтеграцією в CRM.
- Основні завдання:

  Проаналізувати методи створення ботів та CRM-систем.

  Спроєктувати архітектуру та логіку взаємодії.

  Розробити програмне забезпечення бота

  Реалізувати інтеграцію з KeyCRM через API.

Слайд 3

Переваги чат-бот розробки

- Програмне забезпечення автоматизовує взаємодію з користувачем
- Доступність 24/7, миттєва реакція, зручність інтерфейсу
- Збір бази клієнтів, облік замовлень, аналітика продажів.
- Автоматизація продажів

Слайд 4

Теоритичні основи

Чат-бот:

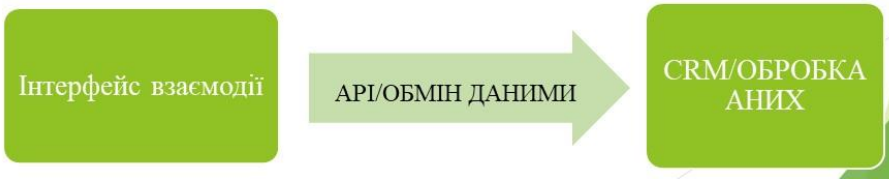
1. Програмне забезпечення, що іітує діалог з користувачем за заданим алгоритмом. Класифікація:
2. Rule-based (сценарні) боти — працюють за чіткими правилами та командами, що забезпечує стабільність бізнес-логіки.
3. Використання Telegram Bot API: підтримка асинхронного обміну повідомленнями та вебхуків

CRM-система:

1. Стратегія та інструмент для централізованого збору, зберігання та аналізу даних про клієнтів.
2. Ключові модулі: управління контактами, обробка замовлень (Leads), аналітика продаж

«Бот — CRM»:

1. Створення єдиного інформаційного середовища, де бот виступає інтерфейсом (Front-end), а CRM — базою даних та аналітичним центром (Back-end).
2. Автоматизація рутинних операцій: реєстрація заявок, перевірка статусів, збір зворотного зв'язку.



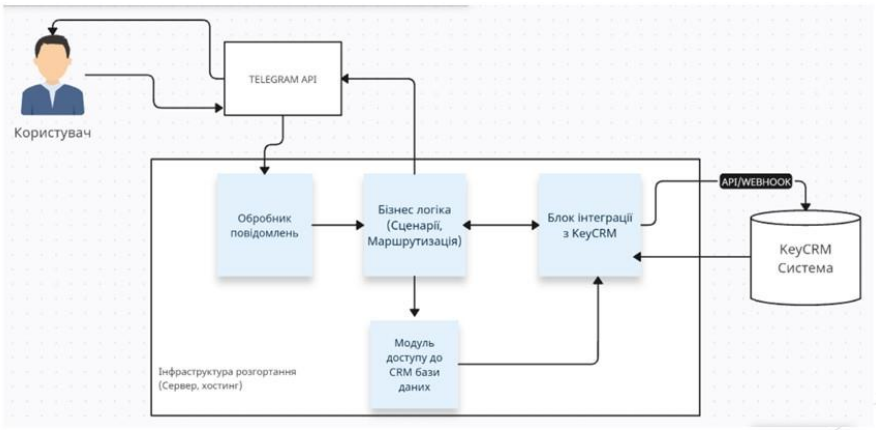
Слайд 5

Процес розробки

1. Створення кабінету адміністратора в KeyCRM та налаштування каталогу товарів. Генерація API-ключа для доступу до зовнішніх запитів
2. Використання BotFather для створення профілю бота та отримання API Token. Налаштування візуального оформлення (опис, аватар, команди)
3. Написання модулів обробки повідомлень та логіки станів (FSM). Реалізація функцій запиту до API CRM
4. Завантаження коду на хмарний сервіс PythonAnywhere. Налаштування безперервної роботи

Слайд 6

Діаграма структури чат-бота

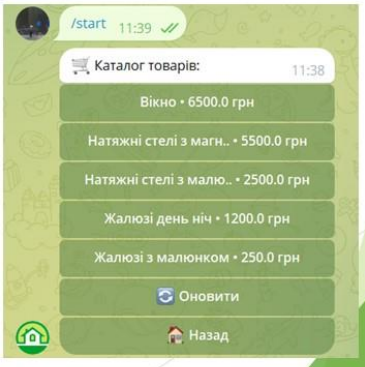


Слайд 7

Взаємодія користувача з ботом на практиці

1. Перша взаємодія користувача з чат ботом (Бот має привітне повідомлення та Інтерактивний інтерфейс. Користувач знайомиться з функціоналом бота).

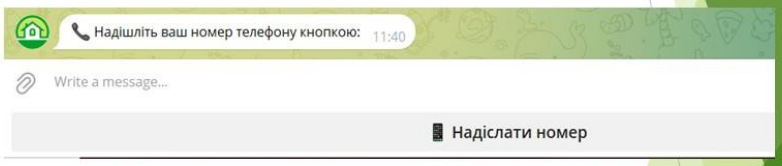
2. Користувач взаємодіє з «каталогом товарів» та обирає що він хоче замовити



Слайд 8

Взаємодія користувача з ботом на практиці

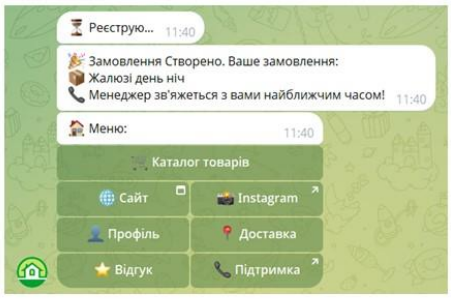
3. Після вибору товару бот надсилає фото товару, ціну та артикул. Також з'явилась кнопка для підтвердження замовлення.



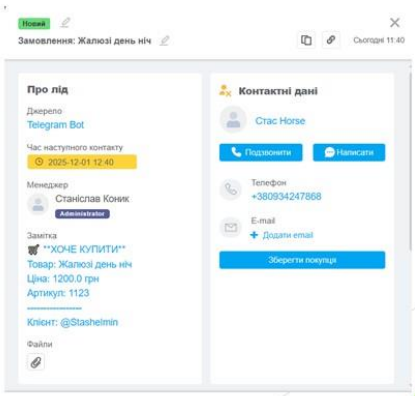
4. Для підтвердження замовлення бот запросить номер телефону користувача для подальшої взаємодії менеджера з клієнтом.

Слайд 9

Взаємодія користувача з ботом на практиці



5. Після підтвердження замовленя користувач отримує повідомлення про створене замовлення та його товар



6. В цей момент, для подальшої взаємодії, менеджеру приходить створений лід на KeyCRM з контактними даними про користувача та інформація про його замовлення.

Слайд 10

ВИСНОВКИ

- Розроблено Telegram чат-бот на Python та реалізовано його повну інтеграцію з CRM-системою через API.
- Застосовано модульну архітектуру та HTTP-запити для гнучкості системи.
- Використано сучасний інструментарій Telegram Bot API для стабільної роботи.
- Реалізовано автоматичну передачу лідів, обробку замовлень та збереження історії взаємодій.

Слайд 11

ВИСНОВКИ

- Забезпечено швидку комунікацію з клієнтами та оформлення замовлень у реальному часі.
- Система протестована та готова до практичного використання і масштабування.
- Система є гнучкою та готовою до подальшого масштабування.
- Автоматизовано ключові бізнес-процеси. Зокрема, реалізовано автоматичну передачу лідів, обробку замовлень та збереження історії взаємодій у картці клієнта в CRM.

Слайд 12