МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

Ігор ШЕЛЕХОВ

_____

(підпис)

червня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерних наук,

освітньо-професійної програми «Інформатика»

на тему: «Інформаційна система менеджменту універсального онлайн-магазину»

здобувача групи ІН-03 Руденко Владислав Олександрович

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

Владислав
РУДЕНКО

_____

(підпис)

Керівник,
старший викладач комп'ютерних наук

Олег БЕРЕСТ _____

(підпис)

Суми – 2024

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«Затверджую»

В.о. завідувача кафедри

Ігор ШЕЛЕХОВ

_____

(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»

здобувача групи ІН-03 Руденка Владислава Олександровича

1. Тема роботи: «Інформаційна система менеджменту універсального онлайн-магазину»

затверджую наказом по СумДУ від *«» червня 2024 р.*

2. Термін здачі здобувачем кваліфікаційної роботи *до червня 2024 року*

3. Вхідні дані до кваліфікаційної роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

1) Аналіз проблеми та актуальність розробки інформаційної системи менеджменту універсального онлайн-магазину в .месенджері телеграм, постановка та формулювання завдань дослідження. 2) Огляд та вибір програмних засобів. 3) Розробка інформаційної системи менеджменту універсального онлайн-магазину. 4) Тестування роботи функціоналу

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується її

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____

(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми та актуальність розробки інформаційної системи менеджменту універсального онлайн-магазину в .месенджері телеграм, постановка та формулювання завдань дослідження.*	06.05.2024 – 08.05.2024	
2	*Огляд та вибір програмних засобів.*	09.05.2024 – 10.05.2024	
3	*Розробка інформаційної системи менеджменту універсального онлайн-магазину.*	11.05.2024 – 14.05.2024	
4	*Тестування роботи функціоналу.*	15.05.2024 – 16.05.2024	
5	*Оформлення пояснювальної записки до кваліфікаційної роботи.*	17.05.2024 – 20.05.2024	

Здобувач вищої освіти _____ Керівник _____

(підпис) (підпис)

АНОТАЦІЯ

Записка: 71 сторінки, 27 рисунки, 3 додаток, 13 використаних джерел.

Обґрунтування актуальності теми роботи – Актуальність даної роботи полягає у потребі створення зручного та функціонального інструменту для управління онлайн-магазином в месенджері Telegram. Враховуючи популярність Telegram та можливості автоматизації, які надають телеграм-боти, розробка інформаційної системи менеджменту дозволить покращити взаємодію з клієнтами та підвищити ефективність операцій.

Об'єкт дослідження – інформаційні процеси створення та менеджменту систем електронної комерції.

Предмет дослідження – методи розробки телеграм-ботів для управління онлайн-магазинами.

Мета роботи – розробка інформаційної системи менеджменту універсального онлайн-магазину на базі месенджера Telegram, яка забезпечить зручне управління замовленнями, каталогом товарів та взаємодію з клієнтами.

Методи дослідження – аналіз та застосування різноманітних інструментів, які сприяють розробці телеграм-магазинів, а саме Aiogram та SqLite.

Результати – розроблена інформаційна система у вигляді універсального інтернет-магазину, який надає можливість налаштувати себе під продаж будь-яких товарів. Користувачі можуть легко та швидко знайти потрібний їм товар за допомогою пошуку товарів по ціні та назві, можуть додати його до кошику та сплатити. Адміністрація має право редагувати категорії товарів, додавати та видаляти самі товари, робити розсилку, організовувати доставку та інше.

ОНЛАЙН-МАГАЗИН, УНІВЕРСАЛЬНИЙ ТЕЛЕГРАМ-МАГАЗИН,
AIOGRAM, PYTHON, PYCHARM, SQLITE

ЗМІСТ

ВСТУП

Актуальність. У сучасному динамічному світі технології постійно розвиваються і створюють нові можливості та тенденції, які впливають на всі аспекти життя, включаючи IT-індустрію. Розумні будинки, електромобілі, хмарні технології та 3D-друк - це лише деякі з інновацій, які змінюють наше середовище. Серед цих інновацій чат-боти - особливий випадок, який з'явився нещодавно і обіцяє світле майбутнє для комунікації та маркетингу.

Чат-боти (роботизовані додатки) революціонізують наш спосіб взаємодії з технологіями та бізнесом. Спрощуючи процеси комунікації та уможливлюючи виконання рутинних завдань, вони стають не лише незамінними помічниками для користувачів, а й ефективними інструментами для бізнесу.

Зростаюча потреба в оптимізації бізнес-процесів і скороченні витрат особливо важлива в сучасних умовах. Використовуючи чат-ботів, компанії можуть автоматизувати значну частину своїх бізнес-операцій, скоротити витрати і вивільнити ресурси для більш стратегічних завдань.

Ринкове середовище, що постійно змінюється, вимагає від компаній швидкого реагування та гнучкості. Автоматизація бізнес-процесів стає ключовим фактором успіху в умовах постійної конкуренції та мінливих ринків. Тому розробка чат-ботів для електронного бізнесу є не тільки важливою, але й необхідною.

Згідно з експертом зі статистики Емілі Дін, у 2024 році, Telegram є одним із найбільш швидкозростаючих месенджерів в Україні, що також сприяє зростанню популярності та необхідності розробки для нього інноваційних рішень [1]. Зі збільшенням популярності Telegram, розробка інформаційної системи менеджменту для онлайн-магазину в цьому месенджері стає ще більш актуальною. Це дозволяє покращити взаємодію з клієнтами та підвищити ефективність операцій.

Крім того, популярність месенджерів у світі та в Україні зокрема робить створення ефективних інструментів для управління онлайн-магазинами у месенджерах надзвичайно актуальним. Telegram, як один із лідерів серед месенджерів, пропонує широкі можливості для інтеграції та автоматизації, що дозволяє бізнесам ефективніше взаємодіяти зі своїми клієнтами.

Чат-боти у месенджері Telegram можуть значно знизити навантаження на людські ресурси, забезпечуючи автоматизоване оброблення замовлень, надання консультацій та підтримки клієнтам у режимі реального часу. Це особливо актуально для малого та середнього бізнесу, який не завжди має можливість наймати велику кількість персоналу для підтримки клієнтів.

Таким чином, розробка інформаційної системи менеджменту універсального онлайн-магазину на базі месенджера Telegram є не лише актуальною, але й необхідною умовою для підвищення конкурентоспроможності бізнесу у сучасних умовах. Впровадження таких систем сприяє покращенню обслуговування клієнтів, зменшенню витрат та підвищенню ефективності бізнес-процесів, що в свою чергу сприяє стабільному розвитку та зростанню підприємства.

1. ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Дослідження предметної області

Сьогодні буквально весь світ так чи інакше спілкується через соціальні мережі, так звані месенджери. Соціальні мережі стали такими популярними завдяки широкому спектру можливостей для спілкування та розваг. Спілкування з далекими родичами, знайомство з новими людьми, обмін різного роду інформацією тощо.

Що ж являє собою соціальна мережа?

Соціальна мережа – це онлайн-платформа, в які зустрічаються люди для спілкування, розміщення інформації та зображень, залишання коментарів чи надсилання повідомлень. Учасники можуть розширювати свої особисті та ділові контакти, зв’язавшись з іншими на веб-сайтах соціальних мереж та в додатках. В соціальних мережах люди діляться досвідом, емоціями та новинами або просто спілкуються один з одним [2].

Сьогодні існує величезна кількість соціальних мереж, кожна з яких орієнтована на різну аудиторію, має свою специфіку і пропонує свої унікальні функції. За даними статистики, у наш час кількість користувачів соціальних мереж у світі налічує майже 5 мільярдів, що становить близько 60% населення планети. А стандартний користувач, за даними глобального дослідження Statista, в середньому проводить у соціальних мережах близько двох з половиною годин на день [3].

Тоді що ж таке месенджери?

Месенджери - це програми для смартфонів і комп'ютерів, які дозволяють користувачам обмінюватися миттєвими повідомленнями. До миттєвих повідомлень відносяться текстові повідомлення, дзвінки. фотографії, зображення, відео та файли. Месенджери міцно увійшли в наше життя. Вони стали невід'ємним інструментом спілкування, як особистого, так і

професійного. У 2024 році популярність месенджерів продовжує зростати, і на ринку з'являються нові цікаві рішення.

Згідно з даними компанії Statista, у 2024 році найпопулярнішими месенджерами у світі є:

- WhatsApp (2 млрд користувачів)
- Facebook Messenger (1,3 млрд користувачів)
- WeChat (1,2 млрд користувачів)
- QQ (648 млн користувачів)
- Telegram (500 млн користувачів)

Ці месенджери лідирують на ринку вже кілька років. Вони пропонують широкий набір функцій і зручний інтерфейс, що робить їх привабливими для користувачів по всьому світу.

WhatsApp – найпопулярніша у світі програма для обміну повідомленнями; WhatsApp доступний на всіх основних платформах і пропонує різні функції, такі як обмін повідомленнями, дзвінки, відеодзвінки, груповий чат і спільний доступ до файлів.

Facebook Messenger є частиною соціальної мережі Facebook; він пропонує ті ж функції, що і WhatsApp, і дозволяє користувачам надсилати повідомлення користувачам Facebook, які некористуються Messenger.

WeChat – найпопулярніший додаток для обміну повідомленнями в Китаї. Він пропонує широкий спектр функцій, включаючи обмін повідомленнями, дзвінки, відеодзвінки, груповий чат, обмін файлами та платежі.

QQ - другий за популярністю месенджер у Китаї, що пропонує схожі функції з WeChat.

Telegram-один з найбільш швидкозростаючих месенджерів у світі. Він пропонує широкий спектр функцій, включаючи обмін повідомленнями, дзвінки, відеодзвінки, груповий чат, обмін файлами та секретний чат.

В Україні ситуація з популярністю месенджерів трохи відрізняється від глобальної. Згідно з даними компанії InMind, у 2024 році найпопулярнішими месенджерами в Україні є:

- Viber (73,6%)
- Facebook Messenger (42,7%)
- Telegram (31,6%)
- WhatsApp (25,3%)
- Skype (19,2%)

Вже кілька років поспіль Viber є найпопулярнішим додатком для обміну повідомленнями в Україні. Він пропонує простий і зрозумілий інтерфейс та широкий спектр функцій для всіх вікових груп.

Telegram є одним із найбільш швидкозростаючих месенджерів в Україні. Він пропонує підтримку каналів з новинами що дуже вплинуло на його популярність з початком війни [4].

Таблиця 1.1.1 – Порівняння месенджерів

Критерії	Telegram	WhatsApp	Viber	Facebook Messenger
Зручність інтеграції	+ Відкрите API для створення чат-ботів та інтеграції сторонніх додатків	– Обмежені можливості для бізнесу через WhatsApp Business API	+/– Обмежена взаємодія через Viber REST API	+/– Обмежена взаємодія через власну систему Messenger API
Система безпеки	+ Висока: Шифрування "на кінцевому пристрої" для секретних чатів, захист паролем, двофакторна аутентифікація.	+/– Середня: Шифрування "на кінцевому пристрої", двофакторна аутентифікація.	+/– Середня: Шифрування "на кінцевому пристрої", двофакторна аутентифікація	– Низька: Присутнє слабке шифрування, збір та аналіз даних користувачів.
Функціональні можливості	+ Широкі: Секретні чати, голосові та відеодзвінки, групи до 200 000 учасників, канали, боти, хмарне сховище, редагування відправлених повідомлень.	+/– Середні: Групи до 256 учасників, голосові та відеодзвінки, обмін файлами, статус "в мережі", зникнення повідомлень.	+/– Середні: Групи до 250 учасників, голосові та відеодзвінки, обмін файлами, статус "в мережі",	+/– Середні: Групи до 250 учасників, голосові та відеодзвінки, обмін файлами, статус "в мережі", інтеграція з Facebook.

Продовження таблиці 1.1.1 – Порівняння месенджерів

Мовна підтримка	+ Багатомовна: Більше 70 мов інтерфейсу.	+ Багатомовна: Більше 60 мов інтерфейсу.	+ Багатомовна: Більше 50 мов інтерфейсу.	+ Багатомовна: Більше 100 мов інтерфейсу.
Інтерактив-ність	+ Висока: Реакції на повідомлення, опитування, вікторини, гіфки, стікери, анімовані емодзі.та багато іншого.	+/– Середня: Реакції на повідомлення, голосові повідомлення, стікери.	+/– Середня: Реакції на повідомлення, голосові повідомлення, стікери.	+ Середня: Реакції на повідомлення, голосові повідомлення, стікери, інтеграція з Facebook.
Зручність використання	+ Інтуїтивний інтерфейс та висока гнучкість в управлінні чатами та групами	+ Простий та доступний інтерфейс, зручне додавання контактів та використання стандартних функцій	+ Відносно простий інтерфейс, але може бути не таким швидким як Telegram, обмежені можливості налаштувань	+/– Доступний інтуїтивний інтерфейс, але з обмеженими налаштуванням, може здаватися перевантаженим для новачків

Врахування всіх переваг Telegram робить його найкращою платформою для створення універсальної системи управління інформацією інтернет-магазину. Але перед цим необхідно розібратися, що таке телеграм-бот і розглянути його переваги.

Телеграм-бот - це програма, яка автоматизує взаємодію з користувачем у месенджері Telegram за допомогою бот-інтерфейсу. Це своєрідний робот або віртуальний асистент, який може виконувати різноманітні завдання без прямого участі людини.

Боти бувають простими та ускладненими. У першому випадку віртуальний помічник працює винятково на основі чітко прописаних інструкцій: відправляє відповіді тільки на питання та команди користувача, які є в його системі. Якщо дії користувача не збігається зі прописаними в програмі, то боту буде важко підібрати правильний варіант відповіді.

Більш складні чат-боти, що працюють на основі штучного інтелекту (ШІ), розпізнають команди та запити будь-якою мовою. Такі боти з часом розвиваються: обробляючи отриману інформацію, вчаться та стають більш розумними. Щоб отримати відповідь від такого бота, користувачам не обов’язково вводити точне формулювання питання чи команди. Якщо коротко, то в основі роботи таких ботів лежить технологія ШІ, обробка природної мови (NLP) та машинне навчання (ML).

Основна функція бота, окрім надання відповідного функціоналу — збільшити охоплення аудиторії та утримувати потенційного клієнта у воронці продажів за мінімальних внесків. Орієнтуючись на сучасну аудиторію, яка все більше надає перевагу месенджерам, маркетологи все частіше впроваджують ботів для розв’язання цих задач.

Основні переваги телеграм-ботів включають наступне:

- Автоматизація завдань:боти можуть виконувати рутинні завдання автоматично. Наприклад, вони можуть регулярно надсилати оновлення, створювати нагадування, відповідати на запитання та здійснювати операції з акаунтами.

- Підтримка клієнтів: боти можуть надавати користувачам підтримку в режимі реального часу. Вони можуть відповідати на запитання, надавати інформацію про продукти та послуги, а також вирішувати типові проблеми.
- Швидкість і зручність: боти можуть відповідати на запитання користувачів швидко і миттєво в режимі 24/7 без необхідності втручання людини. Тому їх можна використовувати будь-де і будь-коли.
- Персоналізація: боти можуть надавати персоналізовані рекомендації та послуги на основі історії взаємодії з користувачами. Вони можуть бути адаптовані до індивідуальних потреб кожного користувача.
- Інтеграція з іншими сервісами: бот може бути інтегрований з іншими додатками та сервісами, такими як CRM-системи, соціальні мережі та електронні платіжні системи, щоб забезпечити ширший спектр функцій.
- Масштабованість: один бот може обслуговувати велику кількість користувачів одночасно, що робить його придатним для малого, середнього та великого бізнесу.

Наше дослідження підтверджує, що магазини можуть автоматизувати процес замовлення, покращити обслуговування клієнтів та підвищити конверсію, впровадивши Telegram-ботів. Більше того, боти стають інструментом дбайливої та ефективної комунікації з покупцями, що є важливим фактором підтримки лояльності клієнтів.

З огляду на розвиток технологій і тенденцію до цифрової трансформації, можна очікувати, що роль Telegram-ботів в електронній комерції буде зростати. Для магазинів це означає не тільки підвищення конкурентоспроможності, але й можливість створювати більш гнучкі та інноваційні стратегії залучення клієнтів.

Таким чином, на основі нашого дослідження можна стверджувати, що Telegram-боти для магазинів є важливим інструментом підвищення якості обслуговування клієнтів, збільшення продажів і зміцнення позицій на ринку. Їхнє впровадження варто розглядати як стратегічний крок у розвитку сучасного бізнесу.

1.2 Огляд аналогів розроблюваного програмного продукту

Після дослідження значної кількості телеграм-ботів, не було знайдено універсальних, які можна було б налаштувати під продаж будь-яких товарів не покидаючи телеграм, тож було вирішено проаналізувати популярні сайти по продажу товарів, наприклад Rozetka.

Розетка — український інтернет-магазин який був створений в 2004 році як інтернет-магазин з продажу парфумерії та косметики, та вийшов в онлайн з серпня. З червня 2005 почав працювати як інтернет-магазин з продажу техніки - яку можна включити в розетку, а нині на їх сайті можна знайти майже все що завгодно. Розетка має понад 400 відділень у 122 містах і селищах України. У 2020 році був сьомим найвідвідуванішим сайтом в Україні.

Перейшовши на офіційний сайт (рис. 1.2.1), ми можемо побачити класичний інтернет-магазин, в ньому є все що необхідно такому магазинові: категорії товарів, пошук, кошик, акції, пропозиції та інше.

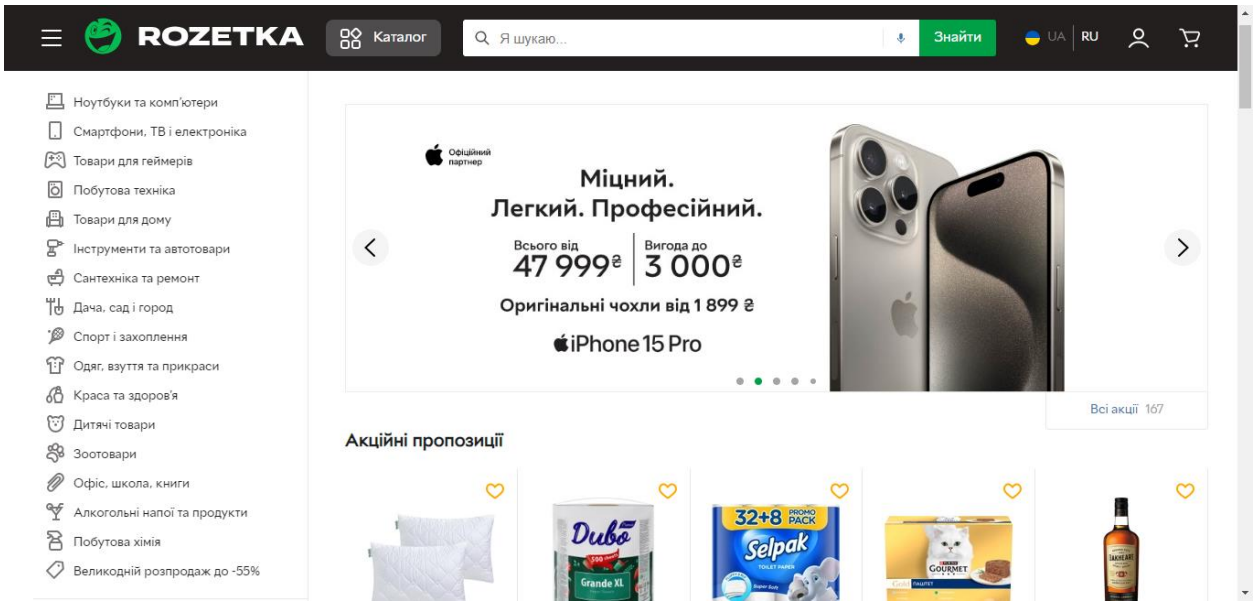

Рисунок 1.2.1 – Головна сторінка

Хотілося б також відзначити зручний та привабливий каталог товарів (рис. 1.2.2), при наведенні на відповідну категорію, по праву частину нам виводиться увесь вміст по групам, а також знижки, які діють на товари в обраній на початку категорії.

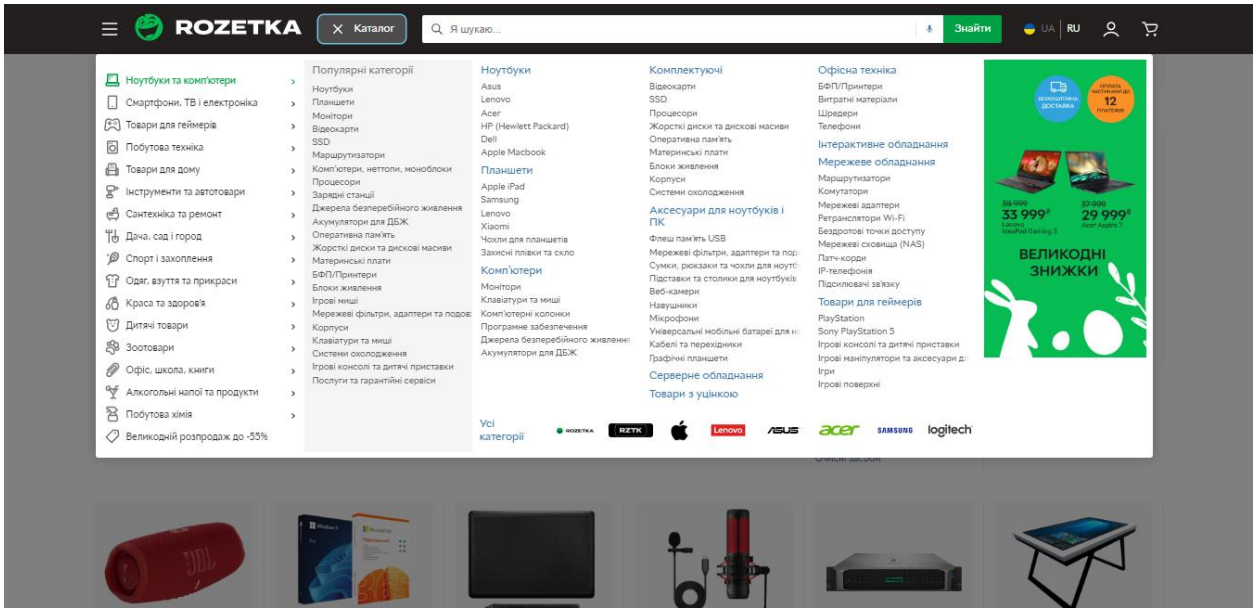

Рисунок 1.2.2 – Каталог товарів

Самі товари мають безліч фільтрів, окрім стандартних (за рейтингом, новинки, за ціною), є також і індивідуальні, які з’являються залежно від обраної категорії, наприклад для комп’ютерної миші можна відфільтрувати за: з’єднанням, розміром, кількістю кнопок, інтерфейсом, особливостями, типом датчика та багато інших.

При оплаті вибраного вами товару, можна обрати будь-який зручний вам спосіб доставки доступний в Україні (рис. 1.2.3), це дуже добре, оскільки ви можете точно бути впевнені, що отримаєте замовлення в будь-якій точці нашої країни.

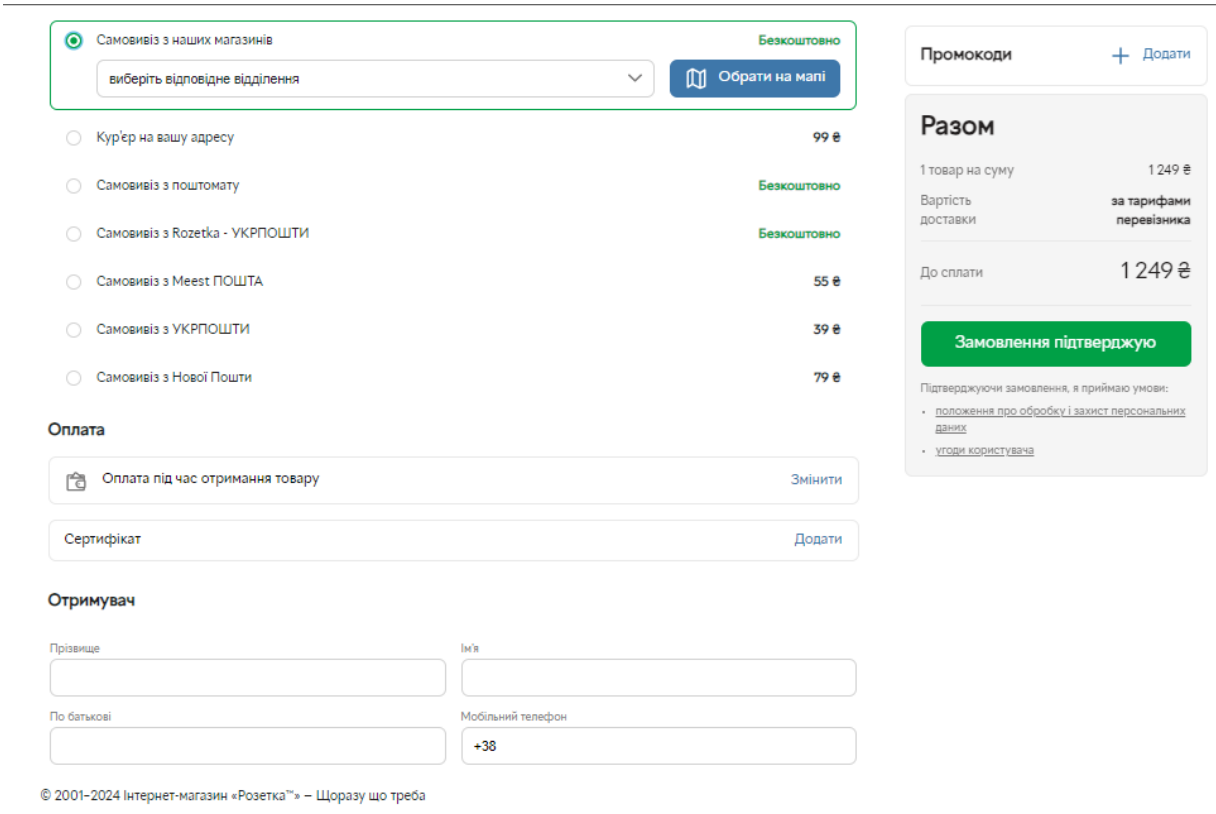

Рисунок 1.2.3 – Оплата замовлення

Загалом інтернет-магазин Rozetka зарекомендував себе як досить популярний та успішний на ринку, пропонує широкий асортимент товарів, зручний інтерфейс та зручну і швидку доставку по всій Україні, що робить його привабливим варіантом для широкого кола покупців. Однак при пошуку Telegram-ботів для цього та подібних магазинів, можна знайти тільки боти підтримки, а повноцінного універсального Telegram-магазину немає. Актуальність таких ботів підтверджується, оскільки ми вже знаємо, що Telegram є однією з найпопулярніших соціальних мереж і може використовуватися для охоплення ширшої аудиторії для продажу товарів.

1.3 Постановка задачі

На основі попереднього аналізу і з врахуванням недоліків розглянутих аналогів постає задача розроблення та впровадження інформаційної системи менеджменту універсального онлайн-магазину з метою покращення комунікації з клієнтами, підвищення задоволеності клієнтів та оптимізації процесів обслуговування замовлень.

Розроблене програмне забезпечення повинно мати:

1) адаптивний дизайн;
2) мінімалістичний інтерфейс;
3) фільтрування товарів по ціновому діапазонові;
4) фільтрування товарів по назві;
5) наявність панелі адміністратора, з можливістю додавати, видаляти категорії товарів та товари робити розсилку;
6) можливість додати будь-які наявні товари до кошика;
7) можливість видалення товарів з кошика та його очищення;
8) можливість оформлення замовлення та його оплата;
9) захист функціоналу адміністратора від несанкціонованого доступу.

1.4 Розроблення дизайну

Метод IDEF0 дозволяє розробнику зобразити уявлення про процес, входи (I), елементи керування (C) над процесом, виходи (O) та механізми (M), що діють на процес (вони разом називаються ICOM).

Кожна з чотирьох сторін функціонального блоку має своє певне значення (роль), при цьому:Вхідні об'єкти (стрілки) діляться на п'ять видів:

– входу (входять в ліву грань роботи) – зображують дані або об'єкти, що змінюються в ході виконання роботи;

– виходу (виходять з правої межі роботи) – зображують дані або об'єкти, що з'являються в результаті виконання роботи;

– управління (входять у верхню грань роботи) – зображують правила і обмеження, згідно з якими виконується робота;

– механізму (входять в нижню межу роботи) – зображують ресурси, необхідні для виконання роботи, але не змінюються в процесі роботи (наприклад, обладнання, людські ресурси тощо);

– виклику (виходять з нижньої межі роботи) – зображують зв'язку між різними діаграмами або моделями, вказуючи на деяку діаграму, де дана робота розглянута більш докладно [5].

Проаналізувавши створення інформаційної системи менеджменту універсального онлайн-магазину, було визначено перелік даних для побудови контекстної діаграми (рис. 1.4.1).

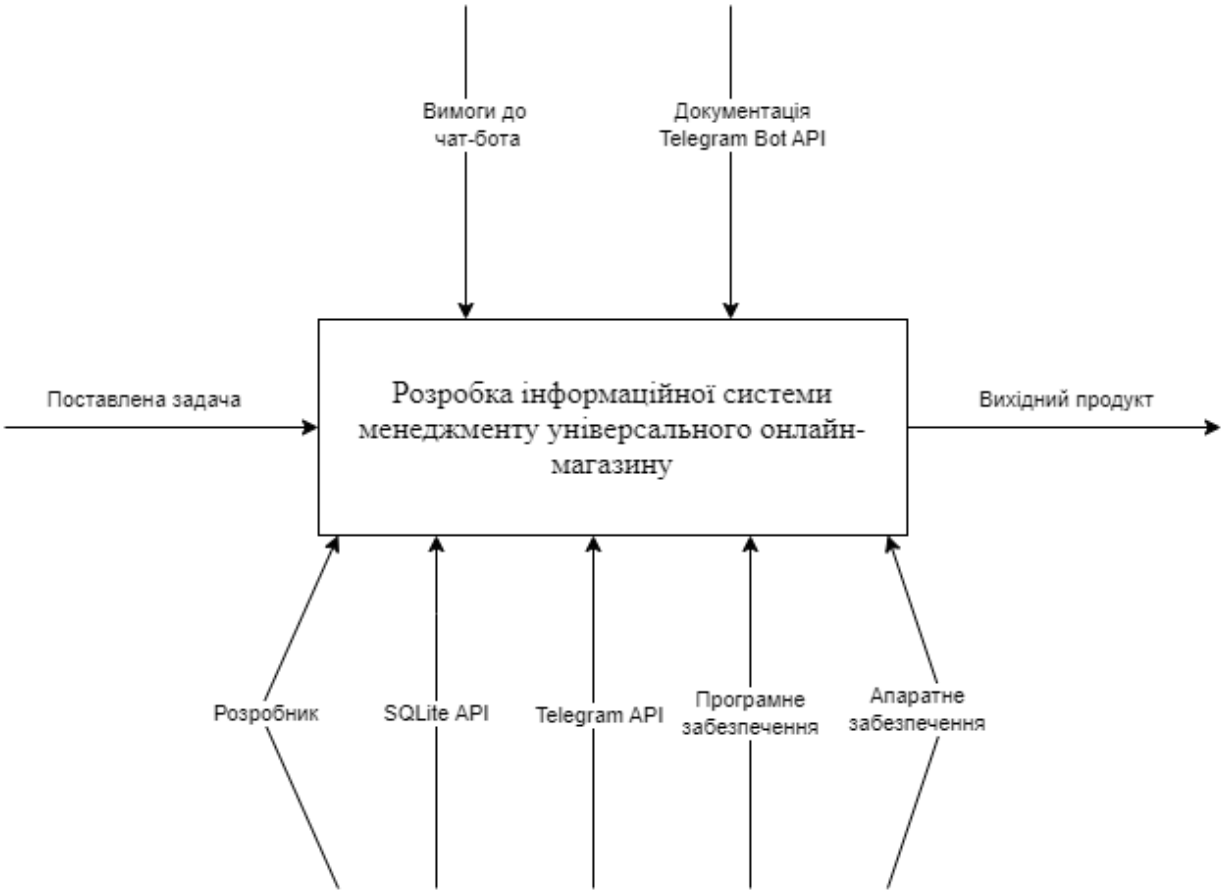

Рисунок 1.4.1 – Контекстна діаграма

Щоб створити бота, також нам знадобиться схема, яка використовує випадки використання, які графічно представляють можливі взаємодії користувача з системою. Такі діаграми показують різні випадки використання програми та різні типи користувачів, які має система. Варіанти використання представляють або колами, або еліпсами. Актори часто зображуються у вигляді фігурок. Щоб розробити діаграми Use Case потрібно було визначити головних акторів системи:

- Admin – розробник або адміністратор. Створений для того щоб маніпулювати ботом та забезпечувати технічну підтримку.
- User – користувач. Який використовує тільки ті функції бота, які відкриті для користувача.

Після того, як ви визначили всіх учасників вашої системи, які можуть виконувати дію, вам потрібно створити список випадків використання. Є відповідні вимоги, яким повинен відповідати бот, вони зазначені у постановці задачі.

Спираючись на сформовані дані про акторів та всі дії для яких можна використовувати бот, створена Use Case діаграма. Вона продемонстрована нижче (рис. 1.4.2).

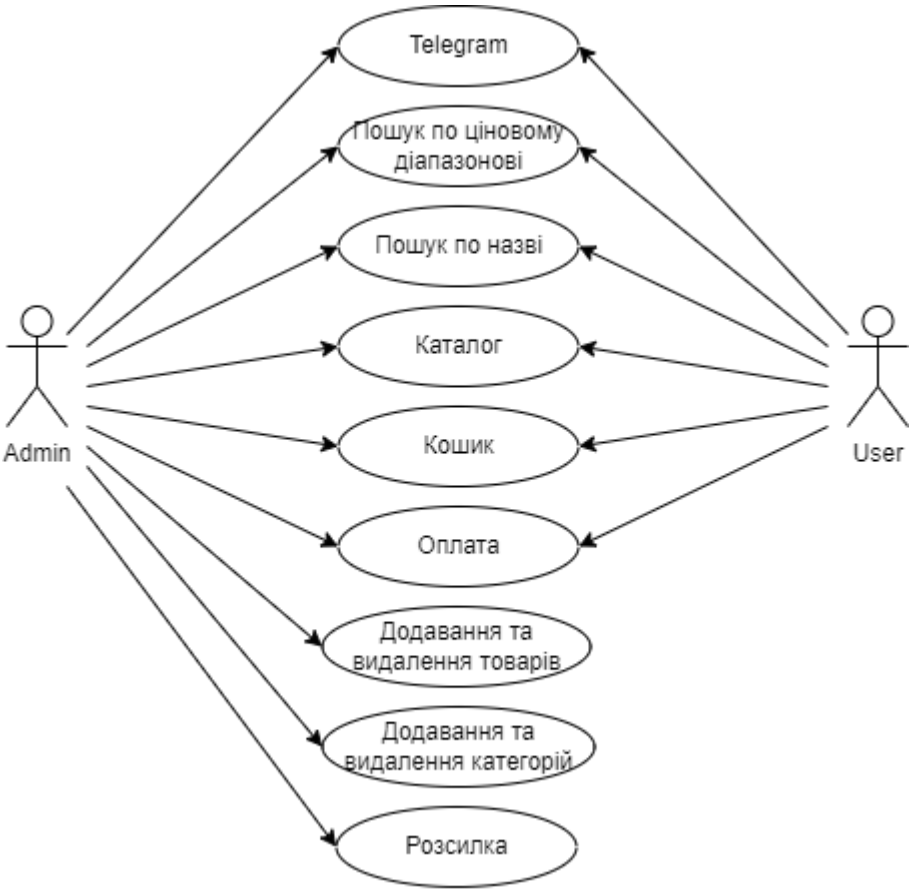

Рисунок 1.4.2 - Діаграма використання бота

2. ВИБІР МЕТОДІВ РІШЕННЯ

Створити бота в Telegram можна кількома способами. Один з них – програмування на одній з мов програмування. Інший спосіб - створення бота без навичок програмування за допомогою конструкторів ботів, наприклад, Manybot.io, Gerabot, Flow XO та інших. За допомогою цих конструкторів користувачі можуть створювати ботів за лічені хвилини, не докладаючи жодних зусиль. Хоча кожен метод має значні переваги, головною проблемою при створенні ботів є їхня функціональність. У цьому випадку метод програмування має велику перевагу, оскільки створення ботів не обмежується доступними інструментами конструктора. Програмування дозволяє створити бота зі складнішою функціональністю, включаючи інтерактивні взаємодії з користувачами, інтеграції з іншими сервісами, створення власних команд та обробку складних завдань.

2.1 Вибір мови програмування для написання бота

Боти можуть бути написані будь-якою мовою програмування, від PHP і Node.js до Java і Go; Python вважається найпопулярнішим, і багато розробників використовують його для написання ботів Telegram з кількох причин:

- простота використання: Python вважається дуже читабельною і простою в освоєнні мовою програмування, що робить її ідеальним вибором для новачків і тих, хто шукає швидкий спосіб створення ботів Telegram.
- багата екосистема: на Python існує безліч бібліотек і фреймворків, що полегшують розробку ботів, таких як python-telegram-bot і aiogram. Ці бібліотеки надають готовий функціонал для взаємодії з Telegram Bot API і спрощують роботу з ним.
- активна спільнота: Python має велику й активну спільноту розробників. Це означає, що в разі виникнення проблем або запитань завжди можна отримати консультацію, документацію та підтримку.

- широкі можливості хостингу: боти на Python можуть бути легко розміщені на багатьох хостингових платформах, що спрощує запуск і підтримку бота в Інтернеті.
- масштабованість і гнучкість: Python - універсальна мова, що дає змогу створювати ботів із різними функціями, від простих команд до інтелектуальних ботів зі штучним інтелектом.

Тому, для розробки телеграм бота, я також буду використовувати Python та бібліотеку aiogram 2.

2.2 Вибір середовища розробки

Для вибору середовища розробки необхідно було дослідити ринок рішень для розробки на Python. В результаті моніторингу увагу привертає продукт компанії JetBrains, а саме продукту PyCharm.

PyCharm – це інтегроване середовище розробки, створене спеціально для програмістів, які працюють з мовою програмування Python. Воно вирізняється потужним функціоналом і зручним інтерфейсом. Під час роботи з PyCharm легко орієнтуватися завдяки інтуїтивно зрозумілому дизайну, що робить процес розробки приємнішим.

Ця IDE забезпечує швидкий доступ до ключових функцій, як-от запуск коду та налагодження, що дає змогу розробникам зосередитися на завданні. PyCharm також підтримує управління версіями, спрощуючи роботу з кодовою базою. Інтеграція можлива в кілька кліків і без командного рядка.

Одним із сильних моментів PyCharm є його продуктивність. IDE має високу швидкість роботи і добре справляється з обробкою великих проєктів. Це особливо важливо для розробників, які працюють над складними і масштабними додатками. А платна версія здатна організовувати проєкти будь-якої структури за допомогою надбудов.

PyCharm тісно інтегрований з особливостями Python. Він надає підтримку динамічної типізації, генераторів, декораторів та інших аспектів мови. Це дає змогу ефективно працювати з особливостями Python, роблячи процес розробки більш гладким.

IDE також відмінно взаємодіє з різними інструментами розробки. PyCharm підтримує системи контролю версій, віртуальні оточення та інструменти тестування, що забезпечує єдиний і зручний робочий простір. І головне, що в останніх версіях практично нічого налаштовувати не потрібно – система сама все підхоплює і дає.

PyCharm надає можливість розширення функціонала з використанням плагінів. Це робить IDE гнучкою та адаптивною до різних потреб розробників [6].

Тому в якості середовища розробки програмного продукту було обрано PyCharm 2023 Community Edition.

2.3 Вибір бази даних

Вибір бази даних є одним із найважливіших аспектів при створенні ботів, тому потрібно проаналізувати найпопулярніші і вибрати найкращий варіант, який підходить під створення нашого бота.

Бази даних (БД) – це структуровані набори даних, організовані та збережені за допомогою спеціальних програм. Бази даних потрібні для ефективного зберігання, управління та доступу до інформації, що робить їх важливою частиною створення програм на Python і багатьох інших мовах.

БД дають змогу програмістам легко додавати, оновлювати, видаляти та витягувати інформацію з програми, а також забезпечують узгодженість даних і безпеку [7].

Навіщо програмістам потрібні бази даних?

Програмістам бази даних здебільшого потрібні з наступних причин:

- Зберігання даних: бази даних можуть зберігати великі обсяги даних, включаючи користувачів, продукти та транзакції.

- Організація даних: бази даних можуть структурувати дані у вигляді таблиць і зв'язків між таблицями. Це дозволяє ефективно організовувати інформацію та керувати нею.
- Швидкий доступ до даних: бази даних спрощують пошук, сортування та фільтрування даних, щоб ви могли швидко знайти потрібну інформацію.
- Масштабованість: база даних може масштабуватися зі збільшенням обсягів даних і зростанням вимог до обробки запитів.
- Інтеграція з іншими системами: багато додатків потребують взаємодії з іншими додатками та сервісами, і база даних є важливим місцем для обміну даними між цими системами.
- Забезпечення цілісності даних: бази даних можуть використовувати різні механізми для забезпечення цілісності даних, такі як обмеження цілісності, транзакції тощо, щоб допомогти запобігти помилкам і втраті даних.
- Автоматизація завдань: багато операцій, пов'язаних з базами даних, можна автоматизувати за допомогою Python, включаючи маніпулювання даними, обробку інформації та створення звітів.

Наразі існує шість найпопулярніших баз даних:

- SQLite
- MySQL
- PostgreSQL
- MongoDB
- Redis
- Oracle Database

Давайте зробимо їх короткий огляд.

2.3.1 SQLite

SQLite - це легка, вбудована реляційна база даних (RDBMS), яка не потребує сервера. Це робить її зручним вибором для невеликих проектів, прототипування та особистих даних. SQLite проста у використанні та налаштуванні, а також портативна, що робить її ідеальною для розробників.

Переваги:

- Легкий та портативний файл бази даних (.db)
- Не потребує сервера
- Простий у використанні та налаштуванні
- Ідеально підходить для розробки

Недоліки:

- Обмежена підтримка одночасних користувачів
- Не підходить для великих проектів з великою кількістю даних

Використання:

- Невеликі веб-сайти та програмні проекти
- Прототипування та тестування
- Зберігання особистих даних
- Вбудовані системи

2.3.2 MySQL

MySQL - це безкоштовна та відкрита RDBMS, яка масштабується, щоб підтримувати великі проекти з великою кількістю користувачів. Її висока

продуктивність та стійкість до збоїв роблять її популярним вибором для веб-сайтів, електронних магазинів та блогів.

Переваги:

- Безкоштовна та з відкритим кодом
- Масштабована для великих проектів
- Висока продуктивність та стійкість до збоїв
- Підтримка одночасних користувачів

Недоліки:

- Складніша установка та налаштування, потребує сервера
- Не така проста у використанні, як SQLite

Використання:

- Веб-сайти з високим трафіком
- Електронні магазини та платформи електронної комерції
- Блоги та форуми
- Інші динамічні веб-застосунки

2.3.3 PostgreSQL

PostgreSQL - це ще одна безкоштовна та відкрита RDBMS, відома своєю надійністю, розширеними можливостями SQL та високою продуктивністю. Її використовують у складних веб-застосунках, аналітиці даних та наукових дослідженнях.

Переваги:

- Безкоштовна та з відкритим кодом
- Дуже масштабована та надійна
- Розширені можливості SQL та функції аналізу даних
- Висока продуктивність

Недоліки:

- Складніша установка та налаштування, потребує сервера
- Не така проста у використанні, як SQLite або MySQL

Використання:

- Складні веб-застосунки та корпоративні системи
- Аналітика даних та машинне навчання
- Наукові дослідження та статистичний аналіз

2.3.4 MongoDB

MongoDB - це NoSQL база даних, яка пропонує гнучкість та масштабованість для неструктурованих або напівструктурованих даних. Її простий формат JSON робить її зручною для роботи з даними, а висока продуктивність робить її популярним вибором для мобільних застосунків та IoT-пристроїв.

Переваги:

- Гнучка та масштабована для неструктурованих даних
- Зберігання даних у форматі JSON, простота роботи з даними
- Висока продуктивність

- Ідеально підходить для мобільних та IoT-застосунків

Недоліки:

- Не така зріла, як реляційні бази даних
- Складніші запити порівняно з SQL
- Не завжди підходить для транзакційних даних

Використання:

- Мобільні застосунки та соціальні мережі
- IoT-пристрої та датчики
- Каталоги продуктів та системи рекомендацій
- Аналітика даних в режимі реального часу

2.3.5 Redis

Redis - це високопродуктивна вбудована база даних NoSQL, яка ідеально підходить для кешування даних, обробки черг повідомлень, реалізації лічильників та інших сценаріїв, що потребують швидкого доступу та маніпуляцій з даними. Вона не призначена для зберігання складних структур даних, але чудово справляється з тим, для чого створена.

Переваги:

- Вбудована, не потребує сервера
- Дуже висока продуктивність
- Ідеально підходить для кешування даних
- Підходить для обробки черг повідомлень

- Підтримує різні структури даних (списки, множини, хеш-таблиці)

Недоліки:

- Не призначена для зберігання складних даних
- Обмежені можливості запитів
- Не підходить для складних транзакцій

Використання:

- Кешування даних для веб-застосунків
- Обробка черг повідомлень в системах реального часу
- Реалізація лічильників, рейтингових систем та ігрових механік
- Сеанси користувачів в веб-застосунках

2.3.6 Oracle Database

Oracle Database - це комерційна реляційна база даних (RDBMS) від корпорації Oracle, відома своєю масштабованістю, надійністю та безпекою. Вона є потужним рішенням для великих підприємств з критично важливими даними.

Переваги:

- Висока масштабованість та продуктивність
- Надійність та безпека даних
- Розширені можливості управління та адміністрування
- Підтримка складних транзакцій та запитів

Недоліки:

- Комерційна ліцензія (платна)
- Складніша установка та налаштування
- Високі вимоги до апаратного забезпечення

Використання:

- Великі підприємства з критично важливими даними
- Фінансові установи та банківські системи
- Управління ланцюгами поставок та ERP-системи
- Системи бізнес-аналітики та звітності

Проаналізувавши всі перераховані вище бази даних, SQLite виявилася найбільш підходящою для нашого проекту. Основними аспектами вибору були відсутність вимог до сервера, простота використання та те що вона уже інтегрована в Python.

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Реєстрація та налаштування бота

Спершу в пошуковій стрічці Telegram потрібно знайти BotFather – чат-бот, за допомогою якого можна створити власного бота та надати йому всі необхідні параметри та функції. BotFather надає можливість назвати створеного бота, створити опис і надати список всіх команд, доступних всім користувачам. Щоб користувачам було легко і зручно користуватися телеграм-ботом, необхідно вказати правильну інформацію. Під час створення видається унікальний токен, який необхідний для зв'язку з ботом та їхньої взаємодії.

Відправляємо команду /newbot і вводимо ім'я нашого бота та його псевдонім (нікнейм), за допомогою якого його шукатимуть інші користувачі (рис. 3.1.1). Надалі всі налаштування, крім нікнейму, можна буде поміняти. У разі потреби також можна отримати новий унікальний ключ.

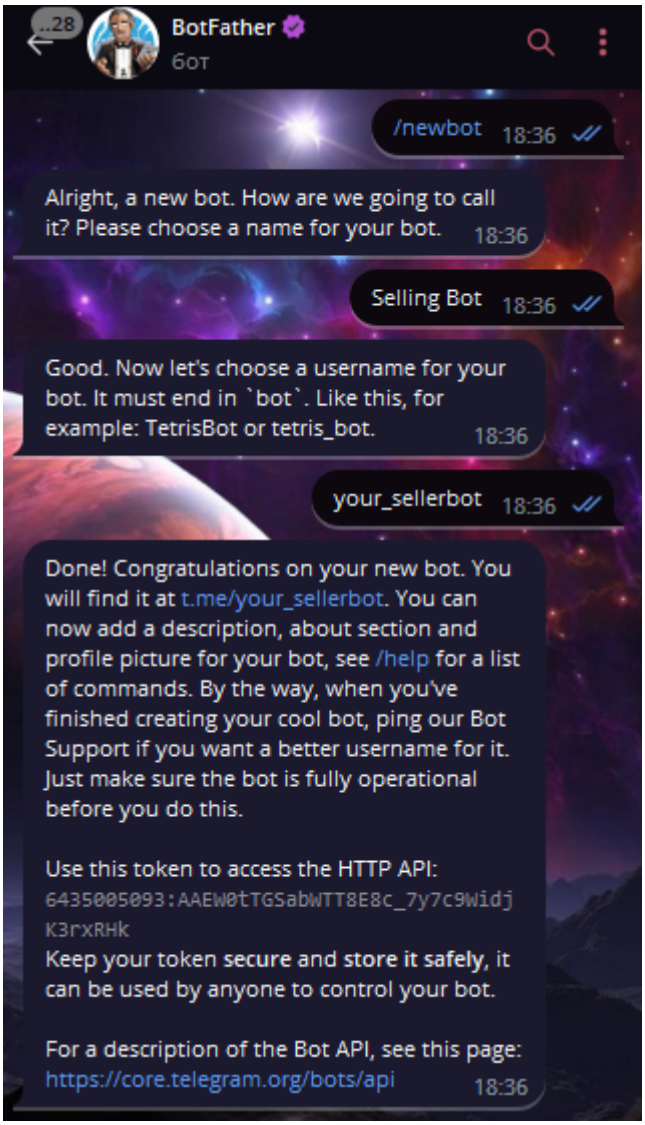

Рисунок 3.1.1 – Реєстрація

Далі можна додати опис бота, який користувач отримуватиме спочатку діалогу, надати команди, які допоможуть відправляти повідомлення швидше та завантажити боту зображення для спрощення пошуку та інше [8], перелік основних та додаткових команд наведено в таблицях 3.1.1 та 3.1.2 [9].

Таблиця 3.1.1 – Основні команди

Команди	Опис
/start	Початок роботи з BotFather, запуск бота
/help	Отримання допомоги та інформації про доступні команди
/newbot	Створення нового бота
/setname	Зміна назви бота
/setdescription	Зміна опису бота
/setuserpic	Встановлення аватара для бота
/setcommands	Додавання або видалення команд бота
/deletebot	Видалення бота

Таблиця 3.1.2 – Додаткові команди

Команди	Опис
/token	Виводить отриманий раніше токен
/revoke	Видаляє токен
/setinline	Налаштування функції Inline-режиму
/setinlinegeo	Функція місцезнаходження
/setinlinefeedback	Налаштування функції відгуків в Inline-режимі
/setjoingroups	Надавання дозволу на додавання бота до груп
/setprivacy	Налаштування приватності бота

3.2 Опис структури бота

Для зберігання даних, потрібно створити базу даних для нашого бота. Для цього було створено чотири таблиці:

- accounts – вміщує в себе інформацію про користувача, який скористався ботом;
- categories – містить інформацію про наявні назви категорії товарів;
- items – зберігає інформацію про наявні товари.
- cart – містить інформацію про додані користувачем до кошика товари.

Таблиця «accounts» зберігає інформацію про всіх користувачів, які хоть раз заходили у бота, а саме їх телеграм айді, айді їх кошика та статус активності, за допомогою якого проводиться розсилка.

Таблиця «categories» містить всі назви категорій товарів, які задані в боті.

Таблиця «items» зберігає інформацію про наявні в боті товари, а саме до якої категорі належить товар, його назву, опис, ціну та фото.

Таблиця «cart» зберігає айді користувача, та айді товару, який від додав до кошика.

Загальний вигляд бази даних бота показано на рисунку 3.2.1, а детально можна побачити на рисунку 3.2.2.

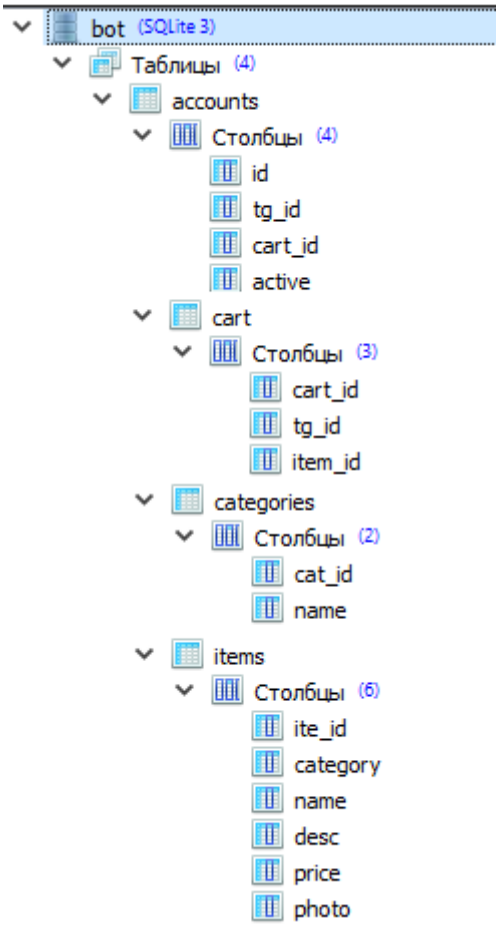

Рисунок 3.2.1 – Загальний вигляд створеної бази даних

```
CREATE TABLE accounts (
    id      INTEGER PRIMARY KEY AUTOINCREMENT,
    tg_id   INTEGER,
    cart_id INTEGER,
    active  BOOLEAN DEFAULT (TRUE)
);
```

```
CREATE TABLE categories (
    cat_id INTEGER PRIMARY KEY,
    name   TEXT
);
```

```
CREATE TABLE items (
    ite_id   INTEGER PRIMARY KEY AUTOINCREMENT,
    category TEXT,
    name     TEXT,
    desc     TEXT,
    price    INTEGER,
    photo    TEXT
);
```

```
CREATE TABLE cart (
    cart_id INTEGER PRIMARY KEY,
    tg_id   INTEGER,
    item_id INTEGER
);
```

Рисунок 3.2.2 – Детальний вигляд створеної бази даних

3.3 Опис функціоналу бота

Структура бота поділяється на клієнтську і адміністраторську. Клієнтська частина вміщую відповідне вітальне повідомлення та три активних кнопки (“Переглянути товари”, “Кошук” та “Інфо”), а частина адміністратора має своє вітальне повідомлення і додаткову кнопку “Панель адміністратора” (рис. 3.3.1), також має додатковий функціонал при перегляді товарів та в “Інфо”, про це пізніше.

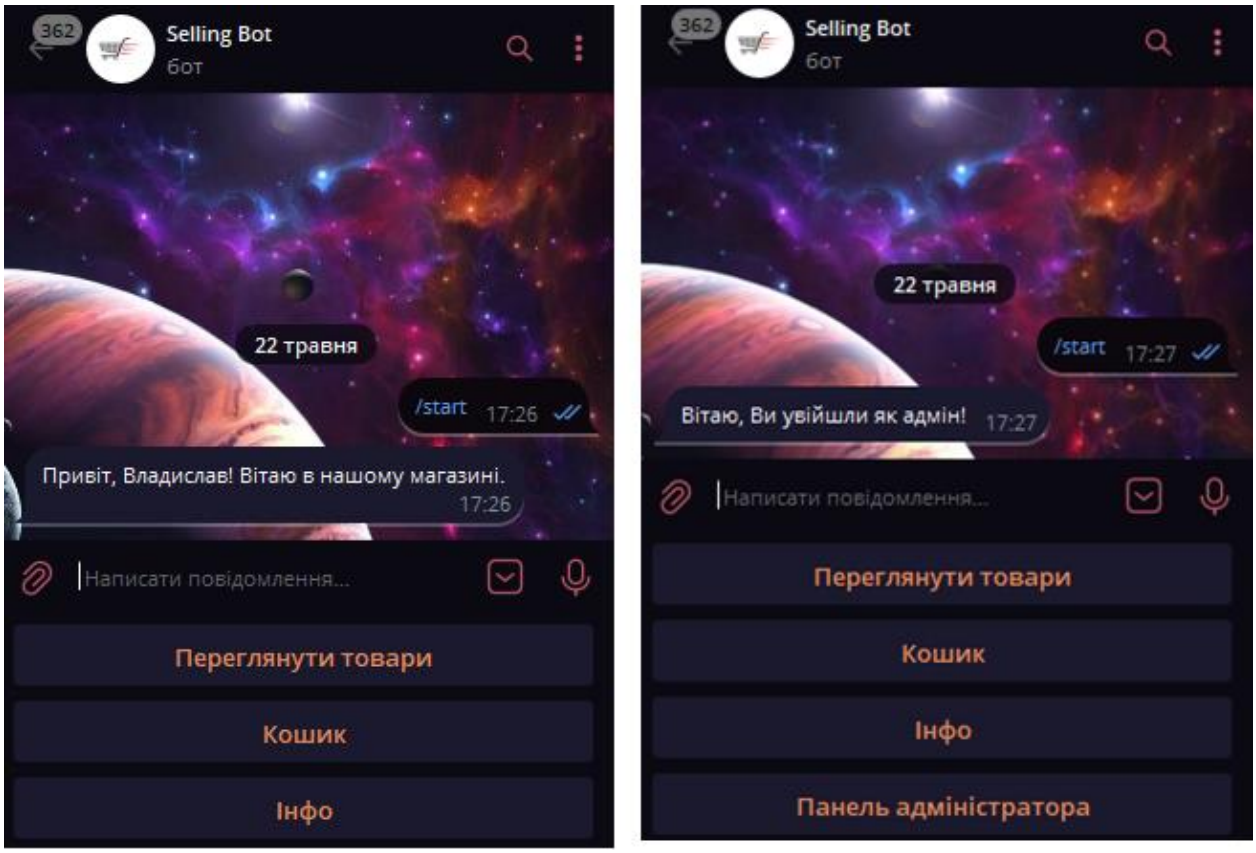

Рисунок 3.3.1 – Початковий функціонал клієнта та адміністратора

Відповідний функціонал та клавіатури надається в залежності від айді користувача, якщо айді збігається з адмінським, який заданий у боті, то видається відповідна клавіатура, якщо ні, то видається клавіатура і функціонал звичайного користувача (рис 3.3.2).

```python
@dp.message_handler(commands=['start'])
async def cmd_start(message: types.Message):
    global save_id
    save_id = message.from_user.id

    await db.add_user(message.from_user.id)

    if message.from_user.id == int(os.getenv('ADMIN_ID')):
        await message.answer('Вітаю, Ви увійшли як адмін!', reply_markup=kb_admin)
    else:
        await message.answer(
            f"Привіт, {message.from_user.first_name}! Вітаю в нашому магазині.",
            reply_markup=kb_client)
```

Рисунок 3.3.2 – Надання відповідного функціоналу та клавіатур

3.3.1 Перегляд товарів

Натиснувши “Переглянути товари” ми побачемо ще 4 активних кнопки (“Підбір товарів по ціні”, “Пошук по назві”, “Каталог ” та “Повернутися назад”).

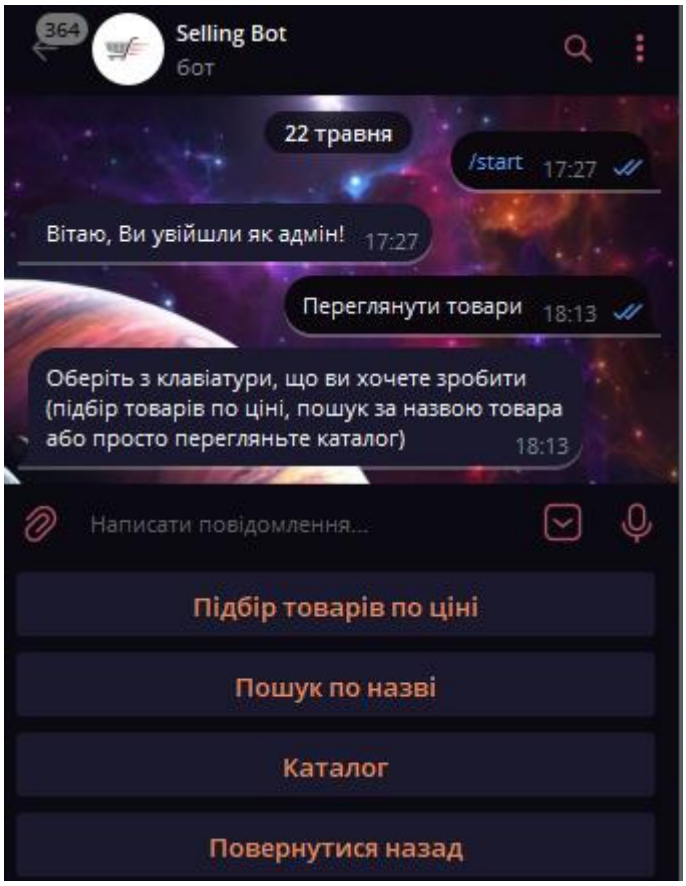

Рисунок 3.3.1.1 – Вміст кнопки “Переглянути товари”

Почнемо по порядку, натиснувши на “Підбір товарів по ціні”, потрібно вибрати категорію товарів, з доступних, в якій ми будемо шукати, потім ввести бажаний ціновий діапазон, і якщо є товари, які відповідають обраним значенням, то ми їх побачимо, зможемо додати до кошика, або переміщуватися між товарами за допомогою кнопок навігації, якщо їх більше чим один. Якщо підбір по ціні робить адміністратор, то він одразу може тут і видалити товар.

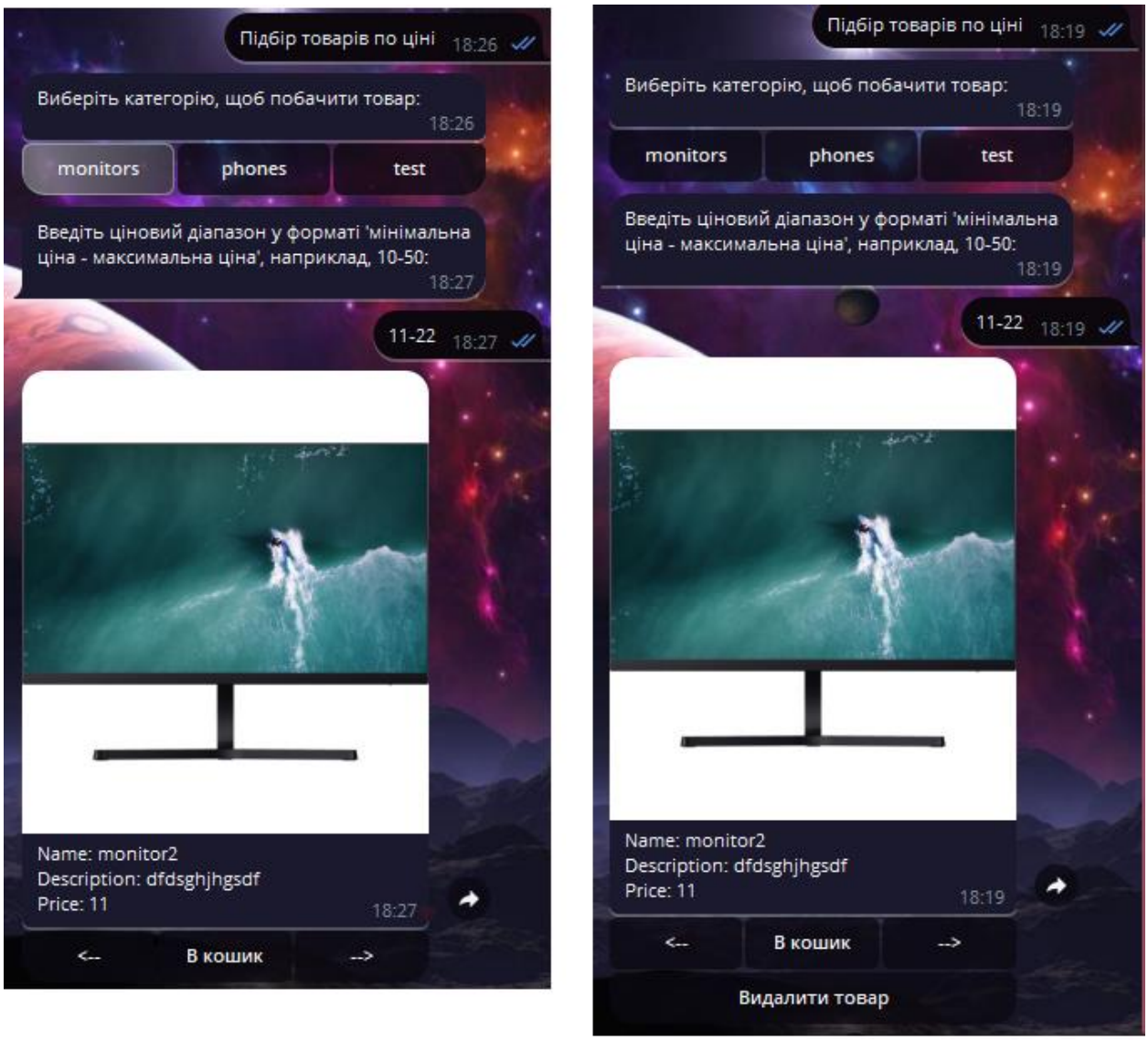

Рисунок 3.3.1.2 – Підбір товар по ціні для клієнта та адміністратора

Далі розглянемо пошук по назві, в цілому тут все аналогічно, також вибираємо категорію товарів, в якій ми хочемо шукати товар, вводимо назву товару, не обов’язково знати повну назву, можна ввести тільки фрагмент, наприклад початок чи кінець назви, якщо такий товар є, то він покажеться на екрані, якщо їх декілька, то між ними можна переміщуватися за допомогою кнопок навігації. Якщо пошук по назві робить адміністратор, то він одразу може тут і видалити товар.

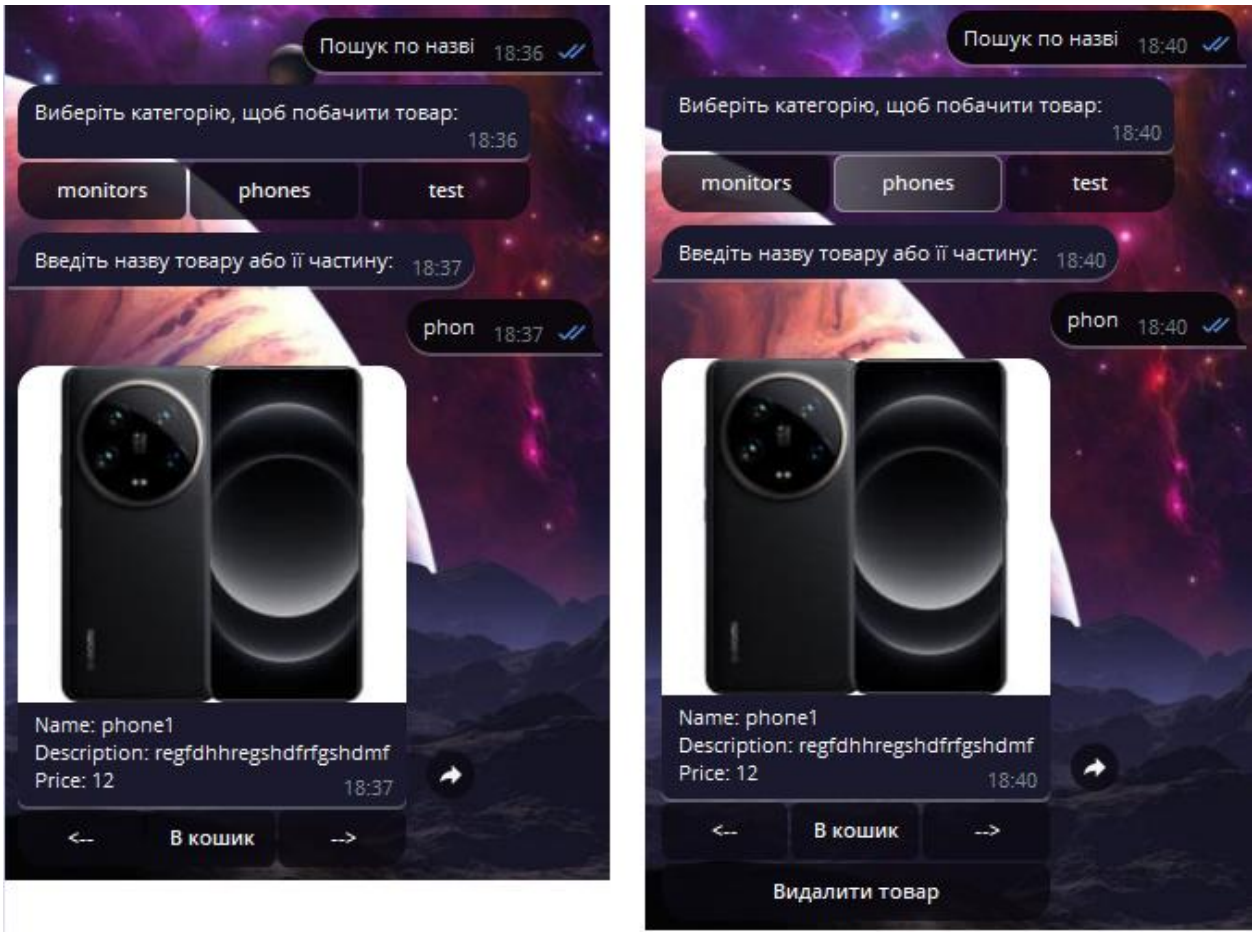

Рисунок 3.3.1.3 – Пошук товарів по назві для клієнта та адміністратора

Ну і тепер про каталог. В ньому ви обираєте доступну категорію і вам показуються всі товари, які вона містить, між ними можна переміщуватися за допомогою кнопок навігації. Знову ж таки, товари можна додавати в кошик, та адміністратор може видаляти товар.

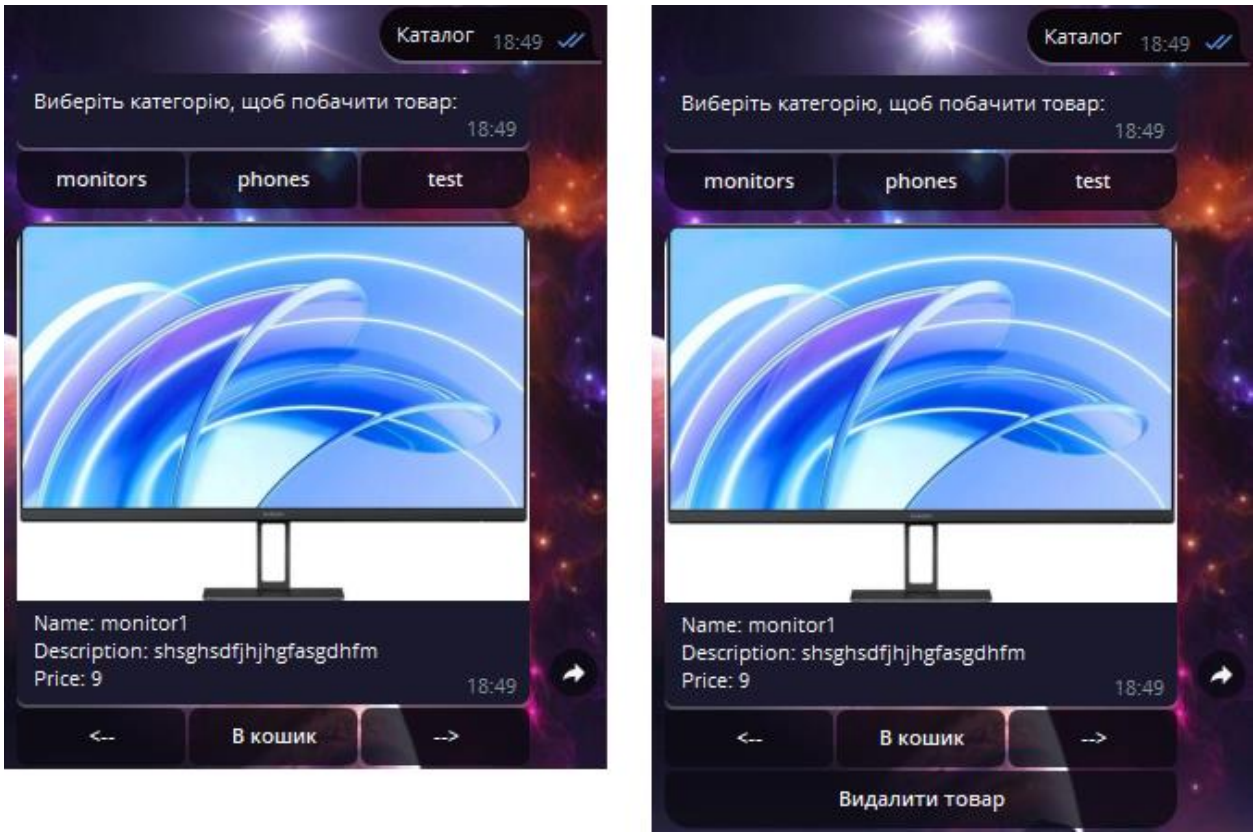

Рисунок 3.3.1.4 – Каталог товарів для клієнта та адміністратора

3.3.2 Кошик

В кошик потрапляють товари, які користувач обрав при пошуку по ціні, назві або просто з каталогу (рис. 3.3.2.1). В ньому присутні три додаткових кнопки (“Прибрати товар”, “Очистити кошик” та “Сплатити”), назви говорять самі за себе, за допомогою “Прибрати товар” ми можемо прибирати по 1 товару з кошика (рис. 3.3.2.2), “Очистити кошик” видаляє всі товари з кошика, а за допомогою “Сплатити” можна заплатити за обрані товари (рис. 3.3.2.3).

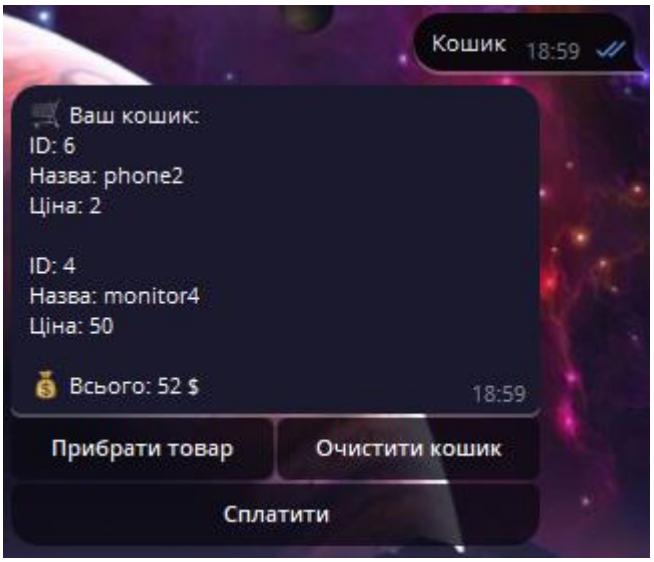

Рисунок 3.3.2.1 - Кошик

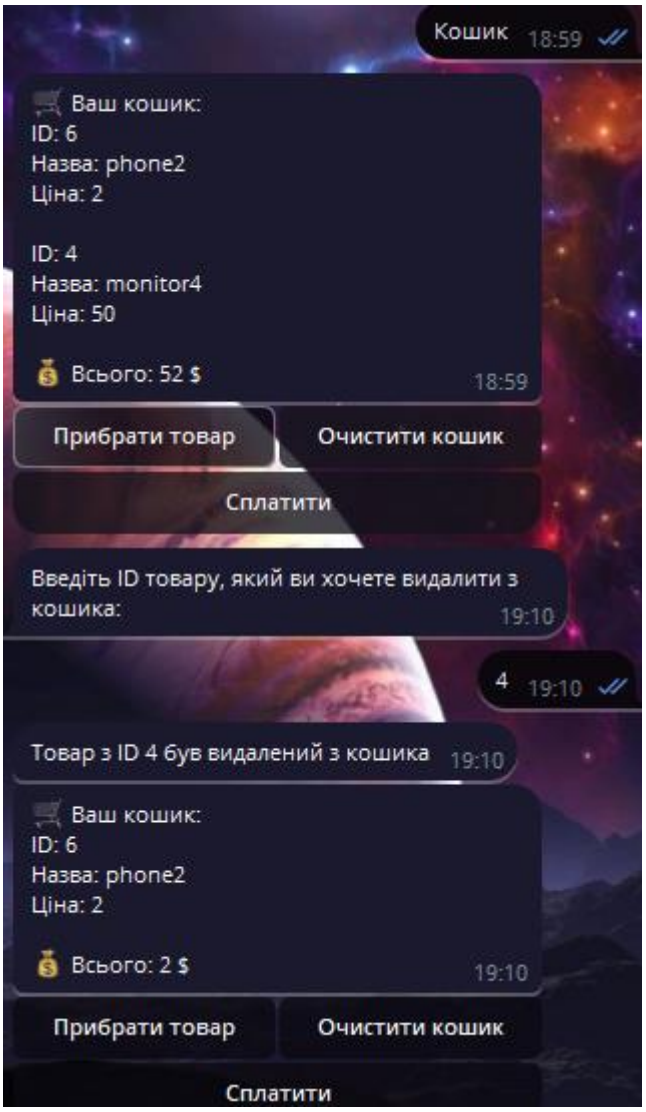

Рисунок 3.3.2.2 – Видалення товару з кошика

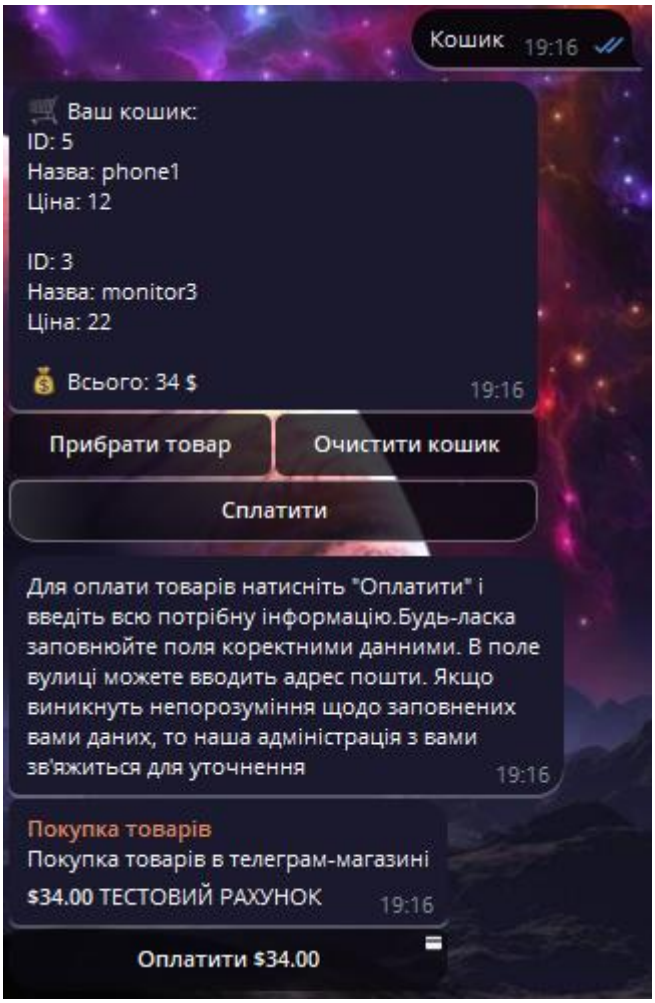

Рисунок 3.3.2.3 – Перехід до оплати

При натисканні на кнопку “Оплатити” з’явиться спливаюче вікно, в якому потрібно ввести дані карти, адресу доставки, електронну пошту та номер телефона (рис. 3.3.2.4), якщо все введено коректно, то пройде оплата (рис. 3.3.2.5), власник магазину отримає гроші, а дані користувача разом з обраними товарами запишуться в ексель таблицю, за допомогою якої, адміністрація зможе організувати доставку або зв’язатися з клієнтом, якщо виникнуть якісь непорозуміння.

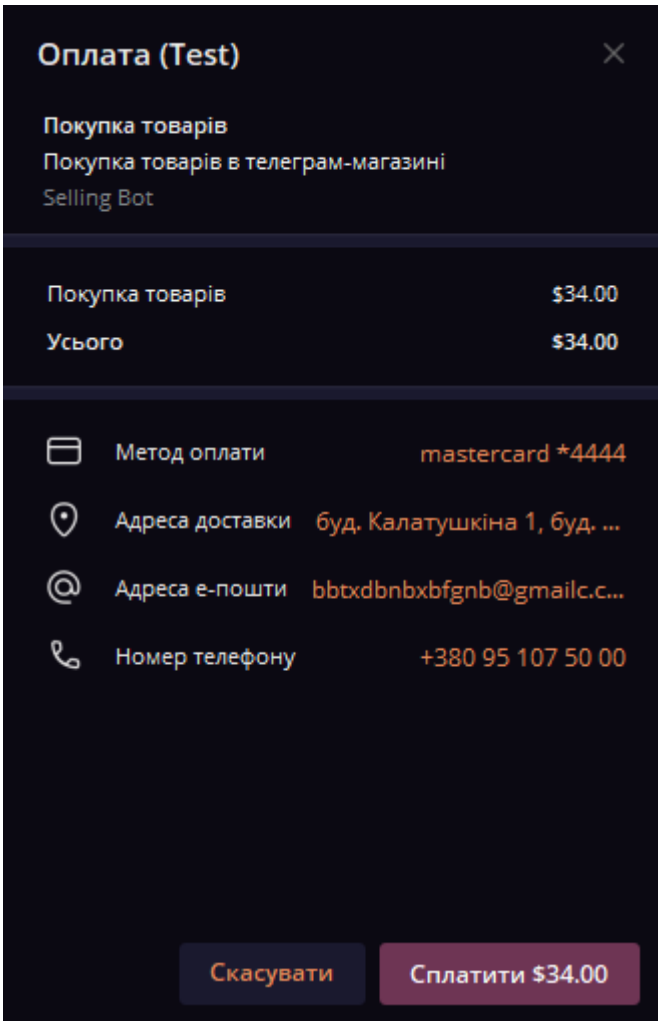

Рисунок 3.3.2.4 – Оплата

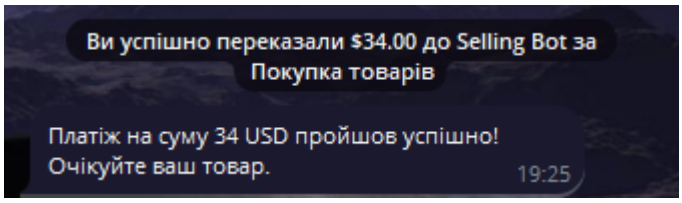

Рисунок 3.3.2.5 – Успішна оплата

3.3.3 Інфо

В “Інфо” може міститися якась важлива інформацію, наприклад, про місцезнаходження офісу або контактна інформації підтримки (рис. 3.3.3.1).

Адміністратор може його змінювати прямо в боті і ці зміни відразу побачать усі користувачі (рис. 3.3.3.2).

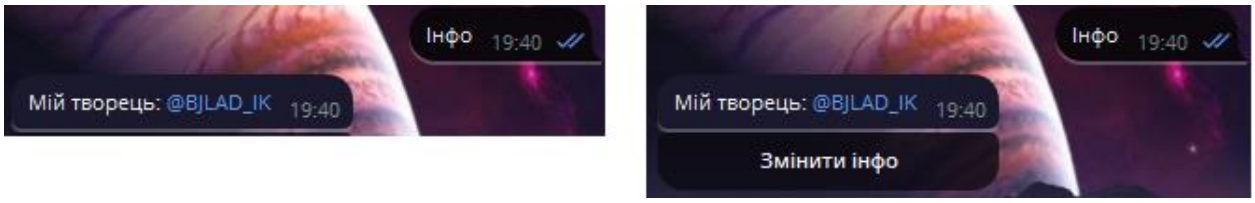

Рисунок 3.3.3.1 - Інфо

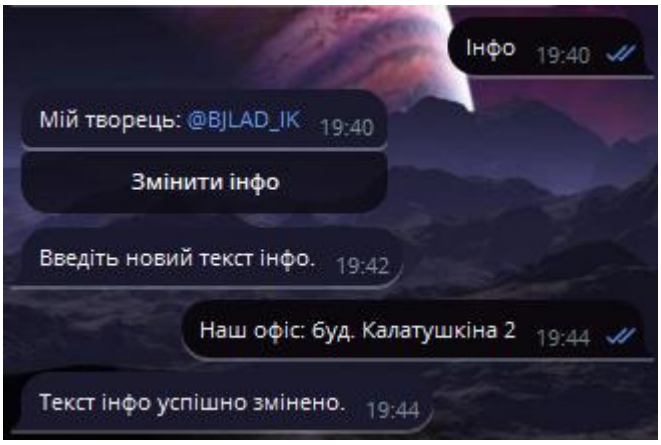

Рисунок 3.3.3.2 – Зміна інфо

3.3.4 Панель адміністратора

Як уже було сказано раніше, панель адміністратора доступна тільки для адміністратора бота, вона містить п’ять кнопок (рис. 3.3.4.1), функціонал яких говорить сам за себе (“Додати категорію”, “Видалити категорію”, “Додати товар”, “Зробити розсилку”, “Повернутися назад”).

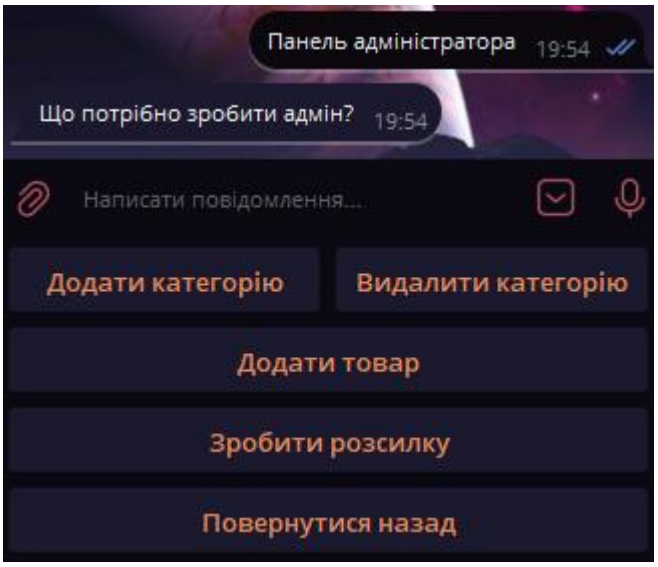

Рисунок 3.3.4.1 – Панель адміністратора

Почнемо з додавання нової категорії, після натискання кнопки, вас попросять ввести назву нової категорії, якщо такої категорії не існує, вона буде додана (рис. 3.3.4.2) і її можна буде побачити в каталозі та при підборі товарів по ціні або назві.

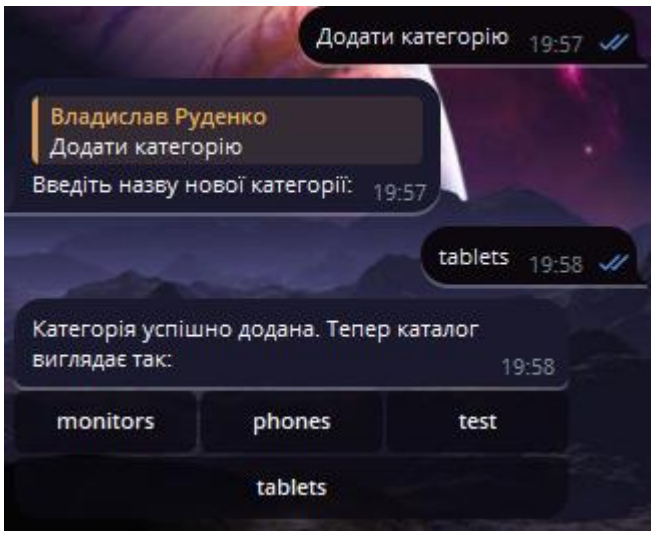

Рисунок 3.3.4.2 – Додавання категорії

При видаленні категорії потрібно натиснути на категорію з списку і вона одразу видалиться (рис. 3.3.4.3), але потрібно бути обережним і пам’ятати, що при видаленні категорії, товари які їй належали, також будуть видалені.

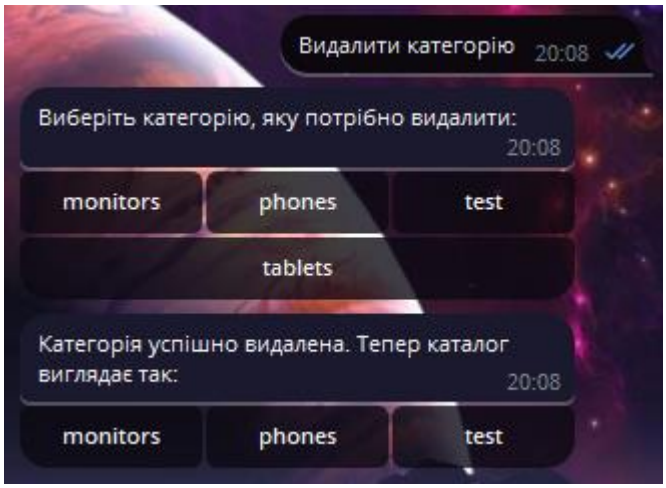

Рисунок 3.3.4.3 – Видалення категорії

Щоб додати товар, потрібно спочатку обрати категорію, до якої він належить, або попередньо її створити, потім ввести назву товару, далі задати опис цього товару, ввести вартість та відправити фото. Якщо ви все зробили правильно, то товар буде доданий (рис. 3.3.4.4) і його можна буде побачити в каталозі, при підборі по ціні або пошуку по назві.

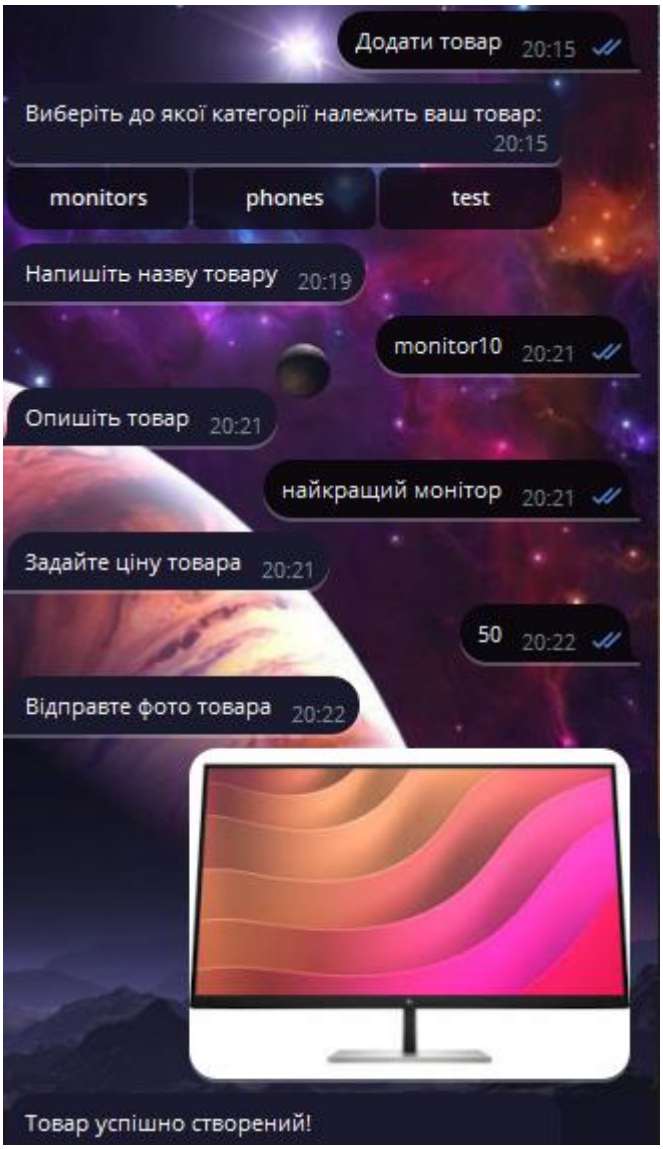

Рисунок 3.3.4.4 – Додавання нового товару

Ну і на останок, розсилка. За допомогою неї можна розіслати повідомлення, з фото (рис. 3.3.4.5) або без (рис. 3.3.4.6), усім користувачам бота, у яких наявний статус активності, якщо повідомлення дійшло до користувача, то статус активності встановлюється активний, якщо ні, наприклад, користувач заблокував бота, то статус активності зміниться на неактивний і повторно йому не будуть надсилатися повідомлення, а якщо користувач розблокує бота, то статус знову стане активним і він знову буде отримувати повідомлення з розсилок.

Рисунок 3.3.4.6 – Розсилка з фото

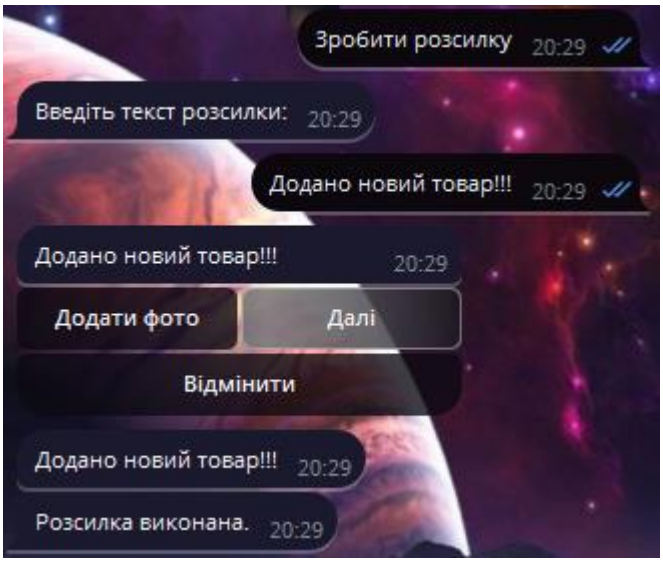

Рисунок 3.3.4.5 – Розсилка без фото

Як бачимо, увесь базовий функціонал інтернет-магазину було розроблено, також було реалізовано універсальність телеграм-магазину шляхом його налаштування під продаж будь-якої продукції. Основні частини коду наведені у додатках А, Б та В. У додатку А міститься клієнтська частина, у додатку Б – функціонал адміністратора, а у додатку В знаходиться база даних та її запити.

ВИСНОВКИ

В даній роботі було проведено дослідження зі створення та впровадження інформаційної системи менеджменту універсального онлайн-магазину для месенджера Telegram. Дослідження включало в себе дослідження та аналіз існуючих подібних систем, вибір мови програмування, вибір середовища розробки, вибір найбільш підходящої бази даних, розробку ботів та тестування його функціоналу.

Дослідження показало, що існуючі на ринку рішення не в повній мірі задовольняють потреби користувачів універсального інтернет-магазину в месенджері Telegram. Тому було прийнято рішення розробити власного бота, який би надавав широкий спектр послуг, включаючи вибір, замовлення та оплату товару.

При виборі технологічного стеку було враховано продуктивність, масштабованість, безпеку та простоту розробки. В результаті для розробки бота було обрано мову програмування Python, середовище розробки PyCharm, базу даних SQLite та бібліотеку aiogram для взаємодії з Telegram API.

Створений бот успішно пройшов тести на функціональність та коректність роботи. Бот відповідає вимогам, зазначеним у цій статті, і може бути використаний для створення інформаційної системи менеджменту універсального онлайн-магазину в месенджері Telegram.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dean E. 100+ Telegram Statistics for 2024 [Електронний ресурс]. *Whop*. URL: https://whop.com/blog/telegram-statistics/ (дата звернення: 23.05.2024).

2. Що таке соціальні мережі? (види, класифікація, безпека...) [Електронний ресурс]. *FutureNow*. URL: https://futurenow.com.ua/shho-take-sotsialni-merezhi-vydy-klasyfikatsiya-bezpeka/ (дата звернення: 28.08.2023).

3. ТОП соціальних мереж - Wizeclub Education [Електронний ресурс]. *Wizeclub Education*. URL: https://wizeclub.education/blog/top-sotsialnih-merezh/ (дата звернення: 01.05.2024).

4. Рейтинг популярних месенджерів в Україні та світі у 2024 році, який обрати [Електронний ресурс]. *Sitecat - Блог для розробників сайтів та IT | Новини, поради, технології*. URL: https://sitecat.net/review/top-10-popular-messengers/ (дата звернення: 01.05.2024).

5. Нотація IDEF0 [Електронний ресурс]. *Elib LNTU*. URL: https://elib.lntu.edu.ua/sites/default/files/elib_upload/Кондіус%202%20готовва/page9.html (дата звернення: 26.05.2024).

6. Найкращі IDE для Python [Електронний ресурс]. *DAN-IT*. URL: https://dan-it.com.ua/uk/blog/luchshie-ide-dlja-python/ (дата звернення: 01.05.2024).

7. База даних Python: огляд сумісних БД та як їх підключити [Електронний ресурс]. *FoxmindEd*. URL: https://foxminded.ua/baza-danykh-python/ (дата звернення: 02.05.2024).

8. Як створити телеграм бота через BotFather? - Ukr-Bot [Електронний ресурс]. *Ukr-Bot*. URL: https://ukr-bot.com/yak-stvoryty-telehram-bota-cherez-botfather/ (дата звернення: 25.05.2024).

9. BotFather. Можливості, команди та функціонал [Електронний ресурс]. *Gerabot*.

URL: https://gerabot.com/article/botfather_mozhlivosti_ta_funkcional (дата звернення: 25.05.2024).

10. aiogram 3.0.1 documentation [Електронний ресурс]. URL:https://docs.aiogram.dev /uk_UA/dev-3.x/ (дата звернення: 29.08.2023).

11. The Python Tutorial – Python 2.7.18 documentation [Електронний ресурс]. *2.7.18 Documentation*. URL: https://docs.python.org/2.7/tutorial/index.html (дата звернення 29.08.2023).

12. Яковенко А. В. Основи програмування. Python. Частина 1 [Електронний ресурс]. Київ, 2018. 195 с. URL: https://ela.kpi.ua/bitstream/123456789/25111/1/Python.pdf (дата звернення 29.08.2023).

13. sqlite3 DB-API 2.0 interface for SQLite databases [Електронний ресурс]. *Python documentation*. URL: https://docs.python.org/3/library/sqlite3.html (дата звернення: 26.05.2024).

Додаток А. Функціонал клієнта

```
import os
from aiogram import types
from aiogram.dispatcher import FSMContext
from openpyxl import load_workbook, Workbook
from create_bot import dp, bot
from keyboards.admin_kb import kb_admin, item_btn_admin
from keyboards.client_kb import kb_client, item_btn_client
from keyboards.catalog_kb import create_keyboard_catalog
from keyboards.cart_kb import cart_btn
from keyboards.info_kb import info_change_btn
from keyboards.view_products_kb import products_kb
from database import database as db
from states.client_states import SearchPrice,
DeleteItemFromCart, SearchName

save_id = None
current_category_messages = {}
min_max = {'min': None, 'max': None, 'category_id': None}
item_data = {"item_id": None, "category_id": None}
summa = 0
info_text = "Мій творець: @BJLAD_IK"


@dp.message_handler(commands=['start'])
async def cmd_start(message: types.Message):
    global save_id
    save_id = message.from_user.id

    await db.add_user(message.from_user.id)

    if message.from_user.id == int(os.getenv('ADMIN_ID')):
        await message.answer('Вітаю, Ви увійшли як адмін!',
reply_markup=kb_admin)
    else:
        await message.answer(
            f"Привіт, {message.from_user.first_name}! Вітаю в
нашому магазині.",
            reply_markup=kb_client)


# --- View Product Handlers ---
@dp.message_handler(text='Переглянути товари')
async def view_products(message: types.Message):
    await message.answer('Оберіть з клавіатури, що ви хочете
зробити (підбір товарів по ціні, пошук за назвою товара '
                         'або просто перегляньте каталог)',
reply_markup=products_kb)


@dp.message_handler(text='Повернутися назад')
```

```
async def back_to_start_kb(message: types.Message):
    if message.from_user.id == int(os.getenv('ADMIN_ID')):
        keyboard = kb_admin
    else:
        keyboard = kb_client

    await message.answer('Повернення до головної клавіатури',
reply_markup=keyboard)


# --- Search by Price Handlers ---
@dp.message_handler(text='Підбір товарів по ціні')
async def search_by_price(message: types.Message):
    await SearchPrice.waiting_category.set()
    await show_catalog(message)


@dp.callback_query_handler(state=SearchPrice.waiting_category)
async def handle_category_selection(callback_query:
types.CallbackQuery, state: FSMContext):
    category_id = int(callback_query.data)
    await state.update_data(category_id=category_id)
    await ask_price_range(callback_query.message)


async def ask_price_range(message: types.Message):
    await message.answer("Введіть ціновий діапазон у форматі
'мінімальна ціна - максимальна ціна', наприклад, 10-50:")
    await SearchPrice.waiting_price_range.set()


@dp.message_handler(state=SearchPrice.waiting_price_range)
async def handle_price_range_input(message: types.Message,
state: FSMContext):
    global min_max
    current_category_messages.clear()
    price_range = message.text.strip()
    min_price, max_price = map(int, price_range.split('-'))
    min_max['min'] = min_price
    min_max['max'] = max_price
    async with state.proxy() as data:
        category_id = data.get('category_id')
        min_max['category_id'] = category_id

        items = await
db.get_items_by_price_range_and_category(min_price, max_price,
category_id)

        if items:
            await show_item(message, items[0])
        else:
            await message.answer("Товарів у заданому ціновому
діапазоні не знайдено.")
```

```
        await state.finish()

# --- Search by Name Handlers ---


@dp.message_handler(text='Пошук по назві')
async def search_by_name(message: types.Message):
    await SearchName.waiting_category.set()
    await show_catalog(message)


@dp.callback_query_handler(state=SearchName.waiting_category)
async def handle_category_selection_name(callback_query:
types.CallbackQuery, state: FSMContext):
    category_id = int(callback_query.data)
    await state.update_data(category_id=category_id)
    await ask_name(callback_query.message)


async def ask_name(message: types.Message):
    await message.answer("Введіть назву товару або її частину:")
    await SearchName.waiting_name.set()


@dp.message_handler(state=SearchName.waiting_name)
async def handle_name_input(message: types.Message, state:
FSMContext):
    current_category_messages.clear()
    name = message.text.strip()
    async with state.proxy() as data:
        category_id = data.get('category_id')
        items = await db.search_items_by_name(category_id, name)

        if items:
            await show_item(message, items[0])
        else:
            await message.answer("Товарів із заданою назвою не
знайдено.")

        await state.finish()

# --- View Catalog Handlers ---


@dp.message_handler(text='Каталог')
async def catalog(message: types.Message):
    global current_category_messages
    current_category_messages = {}

    for key in min_max:
        min_max[key] = None
```

```
    await show_catalog(message)


async def show_catalog(message: types.Message):
    categories = await db.get_all_categories()

    if categories:
        keyboard = await create_keyboard_catalog(categories)
        await message.answer('Виберіть категорію, щоб побачити
товар: ', reply_markup=keyboard)
    else:
        await message.answer('Немає доступних категорій')


async def show_item(message: types.Message, item):
    global item_data
    item_data["item_id"] = item[0]
    item_data["category_id"] = item[1]

    item_info = f"Name: {item[2]}\n"
    item_info += f"Description: {item[3]}\n"
    item_info += f"Price: {item[4]}\n"

    keyboard = item_btn_admin if save_id ==
int(os.getenv('ADMIN_ID')) else item_btn_client

    if message.chat.id in current_category_messages:
        await bot.edit_message_media(chat_id=message.chat.id,
message_id=current_category_messages[message.chat.id],

media=types.InputMediaPhoto(media=item[5], caption=item_info),
                                     reply_markup=keyboard)
    else:
        sent_message = await
bot.send_photo(chat_id=message.chat.id, photo=item[5],
caption=item_info,

reply_markup=keyboard)
        current_category_messages[message.chat.id] =
sent_message.message_id


async def show_items_in_categories(message: types.Message,
category: str):
    items = await db.get_items_by_category(category)

    if items:
        await show_item(message, items[0])
    else:
        if message.chat.id in current_category_messages:
            await bot.delete_message(message.chat.id,

current_category_messages[message.chat.id])
```

```
            del current_category_messages[message.chat.id]
        # await message.answer('В цій категорії товарів не
знайдено')


@dp.callback_query_handler(lambda query: query.data.isdigit())
async def show_category_items(callback_query:
types.CallbackQuery):
    category = callback_query.data
    await show_items_in_categories(callback_query.message,
category)


@dp.callback_query_handler(text=['previous_item', 'next_item'])
async def navigate_items(callback_query: types.CallbackQuery):

    if min_max['min'] is not None and min_max['max'] is not
None:
        items = await
db.get_items_by_price_range_and_category(min_max['min'],
min_max['max'], min_max['category_id'])
    else:
        items = await
db.get_items_by_category(item_data["category_id"])

    current_index = next((index for index, item in
enumerate(items) if item[0] == item_data["item_id"]), None)

    if current_index is not None:
        if callback_query.data == 'previous_item' and
current_index > 0:
            await show_item(callback_query.message,
items[current_index - 1])
        elif callback_query.data == 'next_item' and
current_index < len(items) - 1:
            await show_item(callback_query.message,
items[current_index + 1])


@dp.callback_query_handler(text='add_to_cart')
async def add_item_to_cart(callback_query: types.CallbackQuery):
    cart_items = await db.get_items_in_cart(save_id)

    if item_data["item_id"] in cart_items:
        await callback_query.answer('Товар вже знаходиться в
кошику')
    else:
        await db.add_to_cart(save_id, item_data["item_id"])
        await callback_query.answer('Товар додано до кошика')


# --- Basket Handlers ---
```

```
async def display_basket(message: types.Message):
    global summa
    summa = 0
    cart_items = await db.get_items_in_cart(save_id)

    if not cart_items:
        await message.answer('Кошик пустий!')
    else:
        response = "🛒 Ваш кошик:\n"
        for item_id in cart_items:
            item = await db.get_item(item_id)
            response += f"ID: {item_id}\nНазва: {item[2]}\nЦіна:
{item[4]}\n\n"
            summa += int(item[4])
        response += f"💰 Всього: {summa} $"
        await message.answer(response, reply_markup=cart_btn)


@dp.message_handler(text='Кошик')
async def basket(message: types.Message):
    await display_basket(message)


@dp.callback_query_handler(text='clear_cart')
async def clear_cart(callback_query: types.CallbackQuery):
    await db.delete_items_from_cart(save_id)
    await callback_query.answer('Кошик очищено')

    cart_item = await db.get_items_in_cart(save_id)
    if not cart_item:
        await callback_query.message.delete()
        await callback_query.message.answer('Ваш кошик пустий')


@dp.callback_query_handler(text='delete_item_from_cart')
async def delete_item_query(callback_query: types.CallbackQuery,
state: FSMContext):
    await callback_query.message.answer('Введіть ID товару, який
ви хочете видалити з кошика:')
    await DeleteItemFromCart.item_id.set()


@dp.message_handler(state=DeleteItemFromCart.item_id)
async def delete_item(message: types.Message, state:
FSMContext):
    item_id = int(message.text)
    await db.delete_item_from_cart(save_id, item_id)
    await message.answer(f'Товар з ID {item_id} був видалений з
кошика')
    await state.finish()
    await display_basket(message)
```

```
# --- Payment Handlers ---


@dp.callback_query_handler(text='to_pay')
async def to_pay(callback_query: types.CallbackQuery):
    await callback_query.message.answer('Для оплати товарів
натисніть "Оплатити" і введіть всю потрібну інформацію.'
                                        'Будь-ласка заповнюйте
поля коректними даними. '
                                        'В поле вулиці можете
вводить адрес пошти. Якщо виникнуть непорозуміння щодо '
                                        'заповнених вами даних,
то наша адміністрація з вами зв\'яжиться для уточнення')
    await bot.send_invoice(callback_query.message.chat.id,
'Покупка товарів', 'Покупка товарів в телеграм-магазині',
                           'invoice',
os.getenv('PAYMENT_TOKEN'), 'USD',
                           [types.LabeledPrice('Покупка
товарів', summa * 100)],
                           need_email=True,
need_phone_number=True, need_shipping_address=True)


@dp.pre_checkout_query_handler(lambda query: True)
async def pre_checkout_query(pre_checkout_q:
types.PreCheckoutQuery):
    await bot.answer_pre_checkout_query(pre_checkout_q.id,
ok=True)


@dp.message_handler(content_types=types.ContentType.SUCCESSFUL_P
AYMENT)
async def successful_payment(message: types.Message):
    payment_info = message.successful_payment.to_python()

    payment_info['total_amount'] =
payment_info.get('total_amount', 0) / 100

    try:
        workbook = load_workbook('payment_info.xlsx')
        sheet = workbook.active
    except FileNotFoundError:
        workbook = Workbook()
        sheet = workbook.active
        sheet.append(
            ['currency', 'total_amount', 'ordered_items',
'invoice_payload', 'phone_number', 'email', 'country_code',
             'state', 'city', 'street_line1', 'street_line2',
'post_code', 'telegram_payment_charge_id',
             'provider_payment_charge_id'])

    ordered_items = ''
```

```
    cart_items = await db.get_items_in_cart(save_id)
    for item_id in cart_items:
        item = await db.get_item(item_id)
        ordered_items += f"{item[2]} ({item[4]}), "

    order_info = payment_info.pop('order_info')
    order_info_values = [order_info.get('phone_number', ''),
                         order_info.get('email', ''),
                         order_info.get('shipping_address',
{}).get('country_code', ''),
                         order_info.get('shipping_address',
{}).get('state', ''),
                         order_info.get('shipping_address',
{}).get('city', ''),
                         order_info.get('shipping_address',
{}).get('street_line1', ''),
                         order_info.get('shipping_address',
{}).get('street_line2', ''),
                         order_info.get('shipping_address',
{}).get('post_code', '')]

    sheet.append([payment_info.get('currency', ''),
                  payment_info.get('total_amount', ''),
                  ordered_items.rstrip(', '),
                  payment_info.get('invoice_payload', '')] +
order_info_values +
                 [payment_info.get('telegram_payment_charge_id',
''),
                  payment_info.get('provider_payment_charge_id',
'')])

    workbook.save('payment_info.xlsx')

    await bot.send_message(message.chat.id,
                           f"Платіж на суму
{message.successful_payment.total_amount // 100} "

f"{message.successful_payment.currency} пройшов успішно!
Очікуйте ваш товар.")

# --- Info Handlers ---


@dp.message_handler(text='Інфо')
async def info(message: types.Message):
    global info_text

    if message.from_user.id == int(os.getenv('ADMIN_ID')):
        await message.answer(info_text,
reply_markup=info_change_btn)
    else:
        await message.answer(info_text)
```

Додаток Б. Функціонал адміністратора

```
import os
from asyncio import sleep
from aiogram import types
from aiogram.dispatcher import FSMContext
from create_bot import dp
from database import database as db
from handlers.client import show_items_in_categories, item_data
from keyboards.admin_kb import admin_panel, kb_admin
from keyboards.catalog_kb import create_keyboard_catalog
from keyboards.mailing_kb import kb_mailing,
btn_next_and_cancel, btn_cancel
from states.admin_states import ManagerCategories, NewOrder,
BotMailing, ChangeInfoText
import handlers


@dp.message_handler(text='Панель адміністратора')
async def admin_btn(message: types.Message):

    if message.from_user.id == int(os.getenv('ADMIN_ID')):
        await message.answer('Що потрібно зробити адмін?',
reply_markup=admin_panel)
    else:
        await message.reply('Вам не доступна панель
адміністрування')


# --- Add/Delete Category Handlers ---

@dp.message_handler(text='Додати категорію')
async def add_category(message: types.Message):
    await ManagerCategories.adding.set()
    await message.reply('Введіть назву нової категорії: ')


@dp.message_handler(text='Видалити категорію')
async def delete_category(message: types.Message):
    await ManagerCategories.deleting.set()
    categories = await db.get_all_categories()
    keyboard = await create_keyboard_catalog(categories)
    await message.answer('Виберіть категорію, яку потрібно
видалити: ', reply_markup=keyboard)


@dp.message_handler(state=ManagerCategories.adding)
async def process_add_category(message: types.Message, state:
FSMContext):
    async with state.proxy() as data:
        data['category_name'] = message.text
    await db.add_category(data['category_name'])
```

```
    categories = await db.get_all_categories()
    keyboard = await create_keyboard_catalog(categories)
    await message.answer('Категорія успішно додана. Тепер
каталог виглядає так: ', reply_markup=keyboard)
    await state.finish()


@dp.callback_query_handler(lambda callback_query: True,
state=ManagerCategories.deleting)
async def process_delete_category_callback(callback_query:
types.CallbackQuery, state: FSMContext):
    category_id_del = callback_query.data
    await db.delete_categories(category_id_del)
    categories = await db.get_all_categories()
    keyboard = await create_keyboard_catalog(categories)

    if not categories:
        await callback_query.message.delete()
        await callback_query.message.answer('Категорій більше не
існує, всі категорії видалені')
    else:
        await callback_query.message.answer('Категорія успішно
видалена. Тепер каталог виглядає так:', reply_markup=keyboard)

    await db.delete_item_from_category(category_id_del)
    await state.finish()

# --- Add Item Handlers ---


@dp.message_handler(text='Додати товар')
async def add_item(message: types.Message):
    if message.from_user.id == int(os.getenv('ADMIN_ID')):
        await NewOrder.type.set()
        categories = await db.get_all_categories()
        keyboard = await create_keyboard_catalog(categories)
        await message.answer('Виберіть до якої категорії
належить ваш товар: ', reply_markup=keyboard)
    else:
        await message.reply('Додавати товар може тільки
адміністратор')


@dp.callback_query_handler(state=NewOrder.type)
async def add_item_type(call: types.CallbackQuery, state:
FSMContext):
    async with state.proxy() as data:
        data['category'] = call.data
    await call.message.answer('Напишіть назву товару')
    await NewOrder.next()


@dp.message_handler(state=NewOrder.name)
```

```
async def add_item_name(message: types.Message, state:
FSMContext):
    async with state.proxy() as data:
        data['name'] = message.text
    await message.answer('Опишіть товар')
    await NewOrder.next()


@dp.message_handler(state=NewOrder.desc)
async def add_item_desc(message: types.Message, state:
FSMContext):
    async with state.proxy() as data:
        data['desc'] = message.text
    await message.answer('Задайте ціну товара')
    await NewOrder.next()


@dp.message_handler(state=NewOrder.price)
async def add_item_desc(message: types.Message, state:
FSMContext):
    async with state.proxy() as data:
        data['price'] = message.text
    await message.answer('Відправте фото товара')
    await NewOrder.next()


@dp.message_handler(lambda message: not message.photo,
state=NewOrder.photo)
async def add_item_photo(message: types.Message, state:
FSMContext):
    await message.reply('Це не фото!')


@dp.message_handler(content_types=['photo'],
state=NewOrder.photo)
async def add_item_photo(message: types.Message, state:
FSMContext):
    async with state.proxy() as data:
        data['photo'] = message.photo[0].file_id
        item_info = f"Item category: {data['category']}\n"
        item_info += f"Item Name: {data['name']}\n"
        item_info += f"Item Description: {data['desc']}\n"
        item_info += f"Item Price: {data['price']}\n"
        item_info += f"Item Photo: {data['photo']}"

    await db.add_item(state)
    await message.answer(f'Товар успішно створений!',
reply_markup=admin_panel)
    await state.finish()

# --- Delete Item Handlers ---
```

```
@dp.callback_query_handler(text='delete_item')
async def delete_item_callback(callback_query:
types.CallbackQuery):
    await db.delete_item(item_data["item_id"])
    await show_items_in_categories(callback_query.message,
item_data["category_id"])


# --- Mailing Handlers ---

@dp.message_handler(text='Зробити розсилку')
async def start_mailing(message: types.Message):
    if message.from_user.id == int(os.getenv('ADMIN_ID')):
        await message.answer('Введіть текст розсилки:')
        await BotMailing.text.set()
    else:
        await message.reply('Вам не доступна розсилка')


@dp.message_handler(state=BotMailing.text)
async def mailing_text(message: types.Message, state:
FSMContext):
    answer = message.text
    await state.update_data(text=answer)
    await message.answer(text=answer, reply_markup=kb_mailing)
    await BotMailing.state.set()


@dp.callback_query_handler(text='next', state=BotMailing.state)
async def start(call: types.CallbackQuery, state: FSMContext):
    users = await db.get_user()
    data = await state.get_data()
    text = data.get('text')
    await state.finish()

    for user_id, active_status in users:
        try:
            if active_status != 1:
                await db.set_active(user_id, 1)
            await dp.bot.send_message(chat_id=user_id,
text=text)
            await sleep(0.33)
        except:
            await db.set_active(user_id, 0)

    await call.message.answer('Розсилка виконана.',
reply_markup=admin_panel)


@dp.callback_query_handler(text='add_photo',
state=BotMailing.state)
async def add_photo(call: types.CallbackQuery):
    await call.message.answer('Відправте фото')
```

```
    await BotMailing.photo.set()


@dp.message_handler(state=BotMailing.photo,
content_types=types.ContentType.PHOTO)
async def mailing_text(message: types.Message, state:
FSMContext):
    photo_file_id = message.photo[-1].file_id
    await state.update_data(photo=photo_file_id)
    data = await state.get_data()
    text = data.get('text')
    photo = data.get('photo')
    await message.answer_photo(photo=photo, caption=text,
reply_markup=btn_next_and_cancel)


@dp.callback_query_handler(text='next', state=BotMailing.photo)
async def start(call: types.CallbackQuery, state: FSMContext):
    users = await db.get_user()
    data = await state.get_data()
    text = data.get('text')
    photo = data.get('photo')
    await state.finish()

    for user_id, active_status in users:
        try:
            if active_status != 1:
                await db.set_active(user_id, 1)
            await dp.bot.send_photo(chat_id=user_id,
photo=photo, caption=text)
            await sleep(0.33)
        except:
            await db.set_active(user_id, 0)

    await call.message.answer('Розсилка виконана.',
reply_markup=admin_panel)


@dp.message_handler(state=BotMailing.photo)
async def no_photo(message: types.Message):
    await message.answer_photo('Відправте фото!!!',
reply_markup=btn_cancel)


@dp.callback_query_handler(text='cancel',
state=[BotMailing.text, BotMailing.photo, BotMailing.state])
async def cancel(call: types.CallbackQuery, state: FSMContext):
    await state.finish()
    await call.message.answer('Розсилка скасована.',
reply_markup=admin_panel)


# --- Change Info Handlers ---
```

```
@dp.callback_query_handler(text="change_info")
async def change_info_text(callback_query: types.CallbackQuery):
    await ChangeInfoText.new_info_text.set()
    await callback_query.message.answer("Введіть новий текст
інфо.")


@dp.message_handler(state=ChangeInfoText.new_info_text)
async def process_new_info_text(message: types.Message, state:
FSMContext):

    handlers.client.info_text = message.text
    await state.finish()
    await message.answer("Текст інфо успішно змінено.")


@dp.message_handler(text='Повернутися назад')
async def back_to_kb_admin(message: types.Message):
    await message.answer('Повернення до головної клавіатури',
reply_markup=kb_admin)
```

Додаток В. База даних та її запити

```
import sqlite3 as sq

db = sq.connect('database/bot.db')
cur = db.cursor()


async def db_start():
    cur.execute('CREATE TABLE IF NOT EXISTS accounts('
                'id INTEGER PRIMARY KEY AUTOINCREMENT,'
                'tg_id INTEGER,'
                'cart_id INTEGER,'
                'active BOOLEAN DEFAULT(TRUE))')
    cur.execute('CREATE TABLE IF NOT EXISTS items('
                'ite_id INTEGER PRIMARY KEY AUTOINCREMENT,'
                'category TEXT,'
                'name TEXT,'
                'desc TEXT,'
                'price INTEGER,'
                'photo TEXT)')
    cur.execute('CREATE TABLE IF NOT EXISTS categories('
                'cat_id INTEGER PRIMARY KEY,'
                'name TEXT)')
    cur.execute('CREATE TABLE IF NOT EXISTS cart('
                'cart_id INTEGER PRIMARY KEY,'
                'tg_id INTEGER,'
                'item_id INTEGER)')
    db.commit()


async def add_user(user_id):
    user = cur.execute('SELECT * FROM accounts WHERE tg_id ==
{key}'.format(key=user_id)).fetchone()
    if not user:
        cur.execute('INSERT INTO accounts (tg_id) VALUES
({key})'.format(key=user_id))
        db.commit()


async def add_item(state):
    async with state.proxy() as data:
        cur.execute("INSERT INTO items (category, name, desc,
price, photo) VALUES (?, ?, ?, ?, ?)",
                    (data['category'], data['name'],
data['desc'], data['price'], data['photo']))
        db.commit()


async def get_item(item_id):
    item = cur.execute("SELECT * FROM items WHERE ite_id=?",
(item_id,)).fetchone()
```

```
    return item


async def delete_item(item_id):
    cur.execute("DELETE FROM items WHERE ite_id=?", (item_id,))
    db.commit()


async def set_active(tg_id, active):
    cur.execute("UPDATE accounts set active=? WHERE tg_id=?",
(active, tg_id,))
    db.commit()


async def get_user():
    users = cur.execute("SELECT tg_id, active FROM
accounts").fetchall()
    return users


async def get_items_by_category(category):
    items = cur.execute("SELECT * FROM items WHERE category=?",
(category,)).fetchall()
    return items


async def add_category(name):
    cur.execute('INSERT INTO categories (name) VALUES (?)',
(name,))
    db.commit()


async def delete_categories(cat_id):
    cur.execute('DELETE FROM categories WHERE cat_id=?',
(cat_id,))
    db.commit()


async def delete_item_from_category(category):
    cur.execute('DELETE FROM items WHERE category=?',
(category,))
    db.commit()


async def get_all_categories():
    categories = cur.execute('SELECT * FROM
categories').fetchall()
    return categories


async def add_to_cart(tg_id, item_id):
    cur.execute('INSERT INTO cart (tg_id, item_id) VALUES (?,
?)', (tg_id, item_id))
```

```
    db.commit()


async def get_items_in_cart(tg_id):
    items = cur.execute("SELECT item_id FROM cart WHERE
tg_id=?", (tg_id,)).fetchall()
    return [item[0] for item in items]


async def delete_items_from_cart(tg_id):
    cur.execute('DELETE FROM cart WHERE tg_id=?',
(tg_id,)).fetchall()
    db.commit()


async def delete_item_from_cart(tg_id, item_id):
    cur.execute('DELETE FROM cart WHERE tg_id=? AND item_id=?',
(tg_id, item_id))
    db.commit()


async def get_items_by_price_range_and_category(min_price,
max_price, category_id):
    items = cur.execute("SELECT * FROM items WHERE category=?
AND price BETWEEN ? AND ?", (category_id, min_price,
max_price)).fetchall()
    return items


async def search_items_by_name(category_id, name):
    items = cur.execute("SELECT * FROM items WHERE category=?
AND name LIKE ?", (category_id, '%' + name + '%',)).fetchall()
    return items
```