МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Розробка Інтернет-магазина засобами Telegram Bot API**

Виконав: студент 4 курсу, групи 6.1219-пзс
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)
М. К. Фока
(ініціали та прізвище)
Керівник к.ф.-м.н., доцент, доцент кафедри ЕІС та ПЗ
Г.П. Коломоєць
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Дісітел»
П.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти _____перший (бакалавський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Програмне забезпечення систем_____
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри_____Т. В. Критська
“_01_” __березня_____2023_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Фоці Микиті Костянтиновичу
(прізвище, ім’я, по батькові)

1.Тема роботи Розробка Інтернет-магазина засобами Telegram Bot API
керівник роботи Коломоєць Геннадій Павлович, к.ф.-м.н., доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від 29.12.2022 № 1893-с

2. Строк подання студентом кваліфікаційної роботи 14.06.2023
3. Вихідні дані кваліфікаційної роботи бакалавра
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- аналіз сучасних технологій для створення клієнтської частини веб-застосунків;
- аналіз NoSQL СУБД;
- огляд Telegram Bot API
- опрацювання функціональних вимог до системи;
- розробка архітектури системи та проєктування Інтернет-магазину;
- програмна реалізація застосунку та його апробація.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата
		Завдання прийняв

7. Дата видачі завдання 01.03.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	01.03 – 10.03.23	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	11.03 – 12.03.23	виконано
3	Аналіз існуючих методів рішення	13.03 – 14.03.23	виконано
4	Огляд та збір літератури стосовно теми кваліфікаційної роботи	15.03 – 20.03.23	виконано
5	Огляд та аналіз існуючих рішень та аналогів	21.03 – 26.03.23	виконано
6	Аналіз сучасних технологій для створення клієнтської частини веб застосунків	27.03 – 28.03.23	виконано
7	Аналіз NoSQL СУБД	29.03 – 13.04.23	виконано
8	Огляд Telegram Bot API	14.04 – 16.04.23	виконано
9	Опрацювання функціональних вимог до системи	17.04 – 19.04.23	виконано
10	Розробка архітектури системи та проектування Інтернет-магазину	20.04 – 01.05.23	виконано
11	Програмна реалізація застосунку та його апробація	02.05 – 13.05.23	виконано
12	Оформлення звіту	13.05 – 20.05.23	виконано

Студент _____ М.К. Фока
(підпис) (прізвище та ініціали)

Керівник роботи _____ Г.П. Коломоєць
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ І.А. Скрипник
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок – 75

Рисунків – 27

Таблиць – 1

Джерел – 17

Фока М. К. Розробка Інтернет-магазину засобами Telegram Bot API: кваліфікаційна робота бакалавра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник Г. П. Коломоєць. Запоріжжя : ЗНУ, 2023. 75 с.

У роботі проводиться аналіз Telegram Bot API та його можливостей щодо розробки засобів електронної комерції. Досліджуються основні функціональні можливості API, зокрема отримання інформації про користувачів, обробка замовлень, створення каталогу товарів, підтримка платіжних систем тощо.

На основі отриманих знань та аналізу сучасних тенденцій електронної комерції розробляється архітектура Інтернет-магазину з використанням Telegram Bot API. Програмна реалізація розроблюється з використанням популярних технологій та інструментів для веб–розробки.

Проведена апробація працездатності розробленого Інтернет-магазину засобами Telegram Bot API. Виконане порівняння функціональності Інтернет-магазину з традиційними аналогами, а також визначена задоволеність користувачів від використання месенджера Telegram для покупок.

Ключові слова: *Telegram Bot API, Інтернет-магазин, електронна комерція, платіжний шлюз, інтегрована комунікація*.

ABSTRACT

Pages – 75

Drawings – 27

Tables – 1

Sources – 17

Foka M. K. Development of an online store using the Telegram Bot API: bachelor's thesis in specialty 121 "Software Engineering" / supervisor H. P. Kolomoyets. Zaporizhzhia: ZNU, 2023. 75 c.

The paper analyzes the Telegram Bot API and its capabilities for developing e-commerce tools. The main functionalities of the API are investigated, including obtaining information about users, processing orders, creating a product catalog, supporting payment systems, etc.

Based on the knowledge gained and the analysis of current e-commerce trends, the architecture of an online store using the Telegram Bot API is developed. The software implementation is developed using popular technologies and tools for web development.

The functionality of the developed online store is tested using the Telegram Bot API. A comparison of the functionality of the online store with traditional analogues is made, and user satisfaction with the use of the Telegram messenger for shopping is determined.

Keywords: *Telegram Bot API, online store, e-commerce, payment gateway, integrated communication*.

ЗМІСТ

ВСТУП

Актуальність теми

Розробка Інтернет-магазину з використанням Telegram Bot API є актуальною темою в сучасному світі електронної комерції з кількох причин.

По-перше, клієнти все частіше використовують Telegram-канали для зв'язку зі своїми улюбленими брендами та отримання інформації. Розробка Інтернет-магазину засобами Telegram Bot API надає торговельним організаціям додаткову можливість бути присутніми в цих популярних комунікаційних каналах, а клієнтам – зручний спосіб взаємодії та отримання інформації про товари та послуги.

По-друге, Telegram Bot API дозволяє створити зручний та персоналізований інтерфейс взаємодії з клієнтами, що покращує їх досвід покупок та сприяє залученню нових клієнтів.

По-третє, використання Telegram Bot API дозволяє забезпечити простішу та швидку інтеграцію з іншими сервісами та інструментами, наприклад, з платіжними системами, службами доставлення або аналітичними інструментами, що може значно спростити та автоматизувати процеси замовлення та обробки товарів та послуг.

По-четверте, в порівнянні з розробкою традиційного вебсайту або мобільного застосунку, використання Telegram Bot API дозволяє знизити витрати на розробку та підтримку. Це особливо важливо для малих та середніх організацій, які мають обмежений бюджет.

Загалом, розробка Інтернет-магазину з використанням Telegram Bot API є актуальною темою, оскільки вона відповідає потребам компаній у розвитку електронної комерції, використовує переваги мобільних технологій та месенджерів, є економічно ефективним рішенням і надає доступ до широкої аудиторії користувачів Telegram.

Мета дослідження

Розробка Інтернет-магазину засобами Telegram Bot API.

Завдання дослідження

Розглянути особливості та чинні рішення (аналоги) для розробки Інтернет-магазину засобами Telegram Bot API. Проаналізувати наявні підходи та технології до розробки Інтернет-магазину та вибрати відповідну для реалізації Інтернет-магазину засобами Telegram Bot API.

Об’єкт дослідження

Об’єктом дослідження є процес розробки Інтернет-магазину засобами Telegram Bot API.

Предмет дослідження

Предметом дослідження є технології створення власного застосунку для задоволення потреб комерційних компаній з метою підвищення ефективності їх роботи.

Методи дослідження

Теоретичні – аналіз і порівняння технологій створення електронної комерції; види класифікації та принципів проектування Інтернет-магазинів; порівняльний аналіз для вибору програмного забезпечення.

Практичне значення одержаних результатів

Практичне значення одержаних результатів дослідження полягає у тому, що розроблений Інтернет-магазин з використанням Telegram Bot API надає компаніям зручну та доступну платформу для залучення нових клієнтів. Взаємодія через Telegram дозволяє підприємствам досягти широкої аудиторії, в тому числі в регіонах, де інші онлайн-платформи можуть бути менш поширеними. Результати дослідження можуть бути використані для

створення Інтернет-магазину засобами Telegram Bot API з метою розширення географії бізнесу та підвищення економічної ефективності.

Апробація результатів

Результати роботи було представлено на XVI університетській науково-практичній конференції студентів, аспірантів, докторантів і молодих вчених «Молода наука-2023» [18].

Глосарій

Інтернет-магазин – Електронна торгова платформа, де товари та послуги продаються та купуються через Інтернет.

Telegram – Популярний месенджер, що надає можливості спілкування та обміну повідомленнями між користувачами.

Telegram Bot API – Інтерфейс програмування додатків (API), який надає можливість створення та управління ботами в Telegram.

Бот – Автоматизована програма, яка може взаємодіяти з користувачами через передвизначені команди або штучний інтелект.

API-ключ – Унікальний код, який надається розробникам для доступу до функцій та можливостей API.

Платіжний шлюз – Сервіс або програмне забезпечення, яке дозволяє здійснювати онлайн-платежі за товари та послуги.

Інтегрована комунікація – Взаємодія між користувачем та ботом через обмін повідомленнями та командами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд літературних джерел

В рамках нашого дослідження, розглянемо розуміння терміну "електронної комерції". Згідно із Законом України «Про електронну комерцію», електронна комерція – це відносини, спрямовані на отримання прибутку, які виникають під час вчинення правочинів щодо набуття, зміни або припинення цивільних прав і обов’язків, здійснені дистанційно з використанням інформаційно-телекомунікаційних систем, у наслідок чого в учасників таких відносин виникають права та обов’язки майнового характеру [1].

Електронний договір, який укладений через обмін електронними повідомленнями та підписаний згідно зі статтею 12 даного Закону, має таку юридичну силу, що й договір, укладений у письмовій формі. Кожна копія електронного документа, підписана відповідно до статті 12 цього Закону, є оригіналом такого документа (пункт 12 статті 11 Закону України "Про електронну комерцію") [1].

Розвиток електронної комерції не припиняється навіть під впливом негативних факторів, таких як COVID-19, російсько-українська війна та інфляція. Навпаки, пандемія COVID-19 створила унікальні можливості для розквіту електронної комерції. У цьому секторі відбувся значний зріст попиту, що прискорив розвиток інвестицій у логістику, цифрову рекламу та найми. Важливим фактором стало також швидке розширення цифрового зв'язку, який став ключовим для економічної стійкості під час пандемії, оскільки карантинні обмеження обмежували прямий контакт між людьми. За даними Міжнародного союзу електрозв'язку, до Інтернету у 2021 році під’єднались приблизно 782 мільйони додаткових осіб, що має позитивний вплив на економічне зростання і відновлення [16]. Карантинні обмеження також сприяли впровадженню, розширенню та розвитку систем електронних

закупівель, що забезпечує різні переваги, такі як зменшення адміністративного тягаря та більша прозорість. Деякі уряди також змушені були використовувати цифрові технології для підвищення прозорості, зокрема управління даними та візуалізації даних, у зусиллях боротьби з корупцією. Однак, електронна комерція зіткнулася зі складними викликами на поворотному етапі. Глобальні події, такі як російсько-українська війна, інфляція та проблеми у ланцюзі постачання, призвели до глобальної рецесії та зниження продуктивності багатьох великих гравців. Прибутки компаній у 2022 році виявилися нижчими, ніж очікувалося. Компанії електронної комерції не можуть уникнути звуження людських ресурсів, перерозподілу рекламного бюджету та оптимізації витрат на запаси. Щоб вижити та процвітати в наступному циклі, компаніям необхідно створити стійкість, пристосуватися до нових розробок, таких як соціальна комерція та метавсесвіт, і використовувати певні аспекти вільної комерції. Попри складні події у 2022 році, електронна комерція залишається найбільшим сегментом цифрової економіки [2].

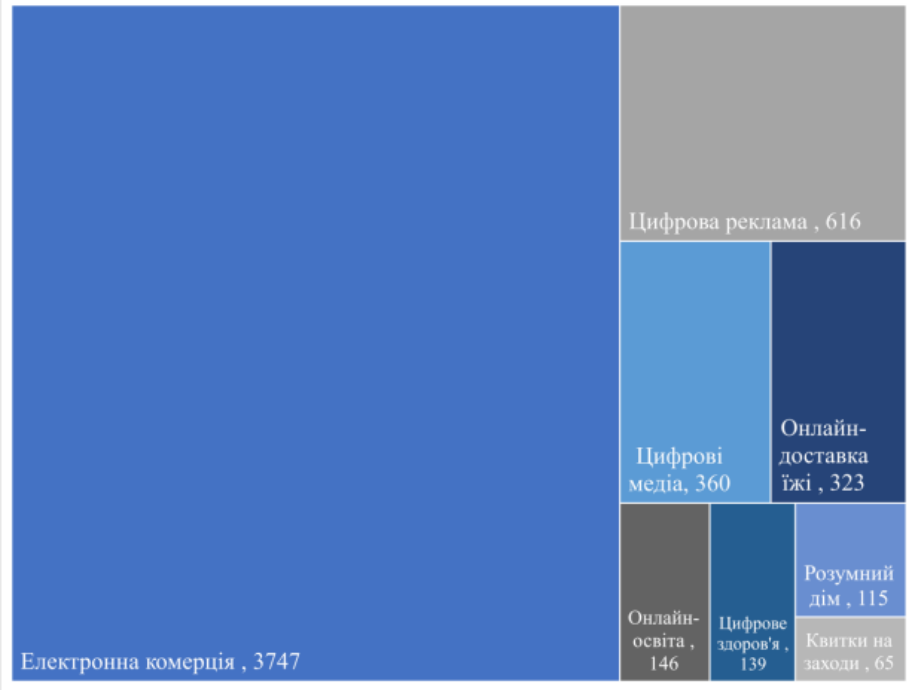

Рис. *1 Дохід окремих ринків цифрової економіки в мільярдах доларів США у 2022 році [2]*

Як ми можемо побачити на рисунку 1. Електронна комерція має найбільшу частку у сфері цифрової торгівлі, що становить 68% від загального обсягу цифрової економіки, що оцінюється у 3747 мільярди доларів США. Натомість найнижча частка спостерігається у секторі квитків на заходи, який становить лише 1,2% або 65 мільярдів доларів США. Аналізуючи динаміку на рисунку 1, можна зробити висновок, що електронна комерція відіграє важливу роль у забезпеченні розвитку цифрової торгівлі у всьому світі.

Проаналізувавши ринок соцмереж, ми бачимо, що протягом року, з 2020 по 2021 рік, аудиторія українських соціальних мереж зросла на 7 мільйонів, досягнувши 26 мільйонів користувачів. Ці дані базуються на дослідженні компанії GlobalLogic, яке використовувало відкриті дані від різних джерел, таких як Data Report, Державна служба статистики, Kantar і PlusOne [17].

Відзначається, що наразі українські соціальні мережі мають 60% населення країни, що порівняно з січнем минулого року, коли цей показник становив незначно більше ніж 40%, є значним зростанням.

Також повідомляється, що з 2019 року кількість українців, які користуються Instagram, збільшилася на 22%, а на Facebook – на 7%. В цей час в Instagram зареєстровано 14 мільйонів українців, а в Facebook – 16 мільйонів.

Найпопулярнішою соціальною мережею залишається YouTube, яка охоплює 96% українських користувачів. В середньому користувачі проводять на цьому сайті близько 40 хвилин щодня.

Результати дослідження також свідчать про зростання кількості українських Інтернет-користувачів на 2 мільйони або 33% в порівнянні з 2019 роком, досягши майже 30 мільйонів на початку 2021 року [3].

Минулого року, у 2022, Telegram входить до переліку п'яти найпопулярніших додатків у всьому світі за кількістю завантажень. За

останній місяць кількість активних користувачів Telegram перевищила 700 мільйонів [4].

Телеграм має значну перевагу у вигляді високоякісної підтримки телеграм-ботів, при цьому функціональність наявних ботів вражає: деякі з них спрямовані на інформування потенційних клієнтів, інші орієнтовані на продажі, а деякі виступають особистими помічниками. Чат-боти використовуються в різних сферах, таких як електронна комерція, кол-центри та галузь ігор. Все залежить від вбудованого функціонала програми [5].

Також ми можемо класифікувати чат-боти як наведено на рисунку 2.

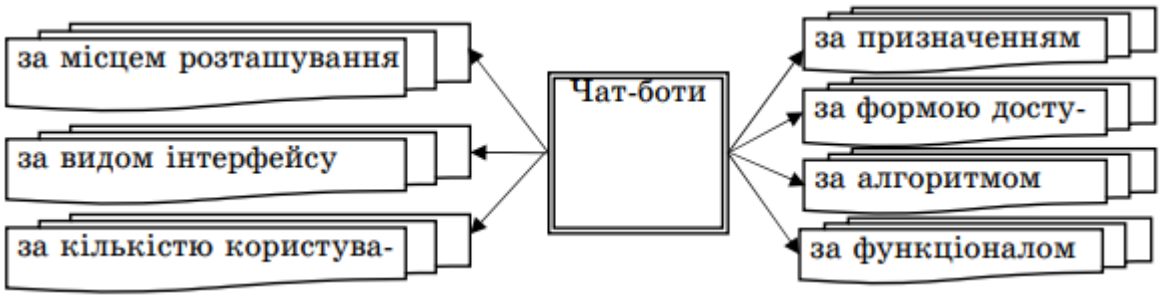

Рис.*2 Класифікація чат-ботів*

Цей огляд літературних джерел підкреслює значення "маркетплейсів" як платформ для ефективної комерції між бізнес-партнерами. Вивчення підходів та інформації, наданої в літературних джерелах, дозволить нам розширити наші знання та розуміння щодо розробки Інтернет-магазина з використанням Telegram Bot API та використовувати ці знання для подальшої реалізації дослідження.

1.2 Аналіз програмних продуктів-аналогів

Електрона комерція з часом стає все більшою, з’являється все більше нових продуктів на цьому ринку, і навіть події 2019 – 2022 років не завадили

їй продовжувати зростати та залишатись актуальною. Розглянемо деякі приклади Інтернет-магазинів.

Rozetka – це один з найбільших Інтернет-магазинів в Україні, який пропонує широкий асортимент товарів в різних категоріях. Заснований у 2005 році, Rozetka став відомим брендом і популярним місцем для покупки електроніки, побутової техніки, комп'ютерних товарів, одягу, взуття, косметики, товарів для дому та багато іншого [6].

Один з ключових принципів Rozetka – це надання високоякісного обслуговування та задоволення потреб клієнтів. Вони стежать за якістю товарів, які представлені в магазині, і співпрацюють зі відомими брендами, щоб забезпечити своїм клієнтам лише надійні та оригінальні продукти.

Інтерфейс магазину можемо побачити на рисунку 3.

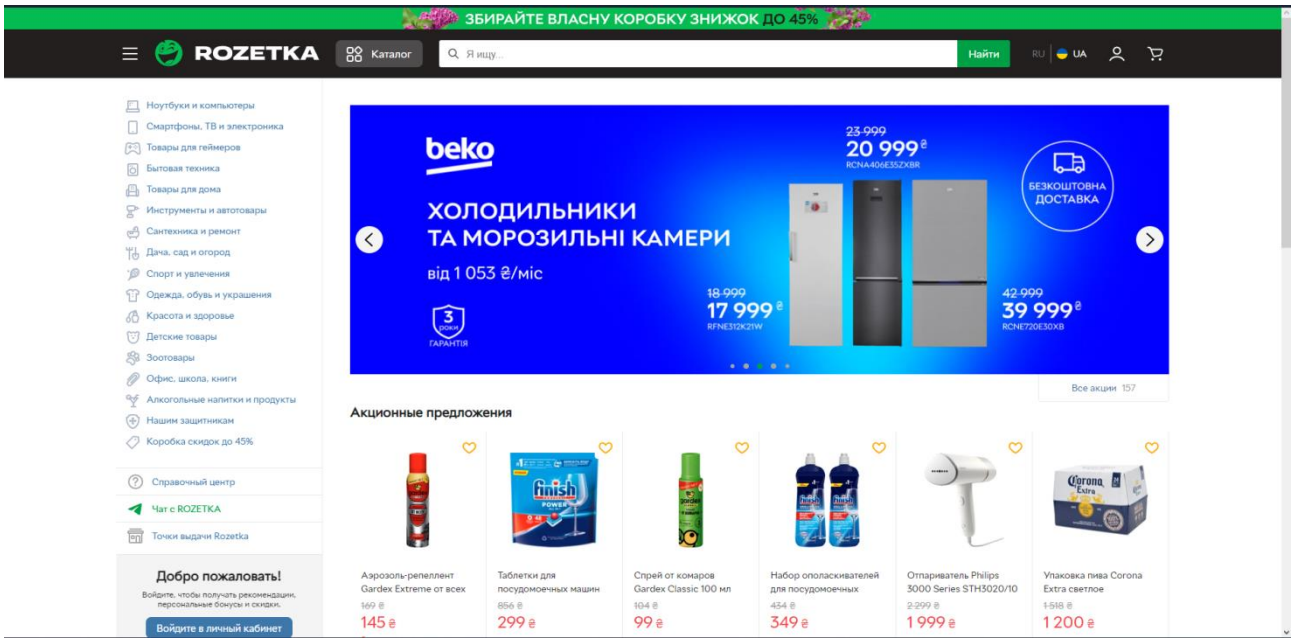

Рис.*3 Інтерфейс Rozetka*

Переваги:

- Розетка має велику базу клієнтів і відоме ім'я на ринку електронної комерції.
- Вебсайт та мобільний додаток Rozetka є добре організованими та зручними у використанні.

- Покупці можуть легко знайти потрібний товар за допомогою розширеної системи фільтрації та пошуку.
- Користувачі можуть залишати відгуки про товари, що дозволяє іншим клієнтам отримувати додаткову інформацію перед покупкою.

Недоліки:

- Складність навігації: Деякі користувачі можуть зазнавати складнощів зі знаходженням конкретних товарів або переходу між різними категоріями.
- Відсутність свого бота в різних месенджерів.
- Розетка може бути перевантажена великою кількістю товарів, що може погано відобразитись на користувацькому досвіді сервісом.
- Відсутність однозначної розмітки акцій: Деякі користувачі відзначають, що акції та знижки на Rozetka можуть бути заплутані або неоднозначними. Вони можуть мати проблеми з розумінням, які товари є дійсно зниженими або які акції дійсні в певний час.

Foxtrot – це відомий український Інтернет-магазин, спеціалізований на продажі електроніки, побутової техніки, комп'ютерних товарів, гаджетів та аксесуарів. Заснований у 1994 році, Foxtrot став одним з найбільших роздрібних торгових мереж в Україні й відомим брендом для багатьох покупців [7].

- Foxtrot пропонує широкий вибір товарів в різних категоріях, включаючи телевізори, смартфони, планшети, ноутбуки, комп'ютери, фото – та відеотехніку, побутову техніку, аудіо – та відеоаксесуари, кухонну техніку, гаджети, ігрові консолі та багато іншого. Каталог товарів Foxtrot охоплює відомі бренди як національного, так і світового рівня, що дозволяє покупцям знайти оптимальний варіант для своїх потреб та бюджету.

Інтерфейс сайту бачимо на рисунку 4.

Переваги:

- Foxtrot відомий своїм широким асортиментом продуктів у сфері електроніки та комп'ютерної техніки.
- Мобільний додаток Foxtrot має простий та інтуїтивно зрозумілий інтерфейс, що сприяє зручному користуванню.
- Користувачі можуть зберігати список обраних товарів та отримувати сповіщення про знижки та акції.
- Foxtrot має розвинутий сервіс підтримки клієнтів, який допомагає розв'язувати проблеми та відповідає на запитання клієнтів.

Недоліки:

- Деякі користувачі скаржаться на тривалість доставлення товарів та проблеми з обміном або поверненням товару.
- Зауважено, що деякі товари на веб-сайті Foxtrot можуть бути недоступні або неактуальні.

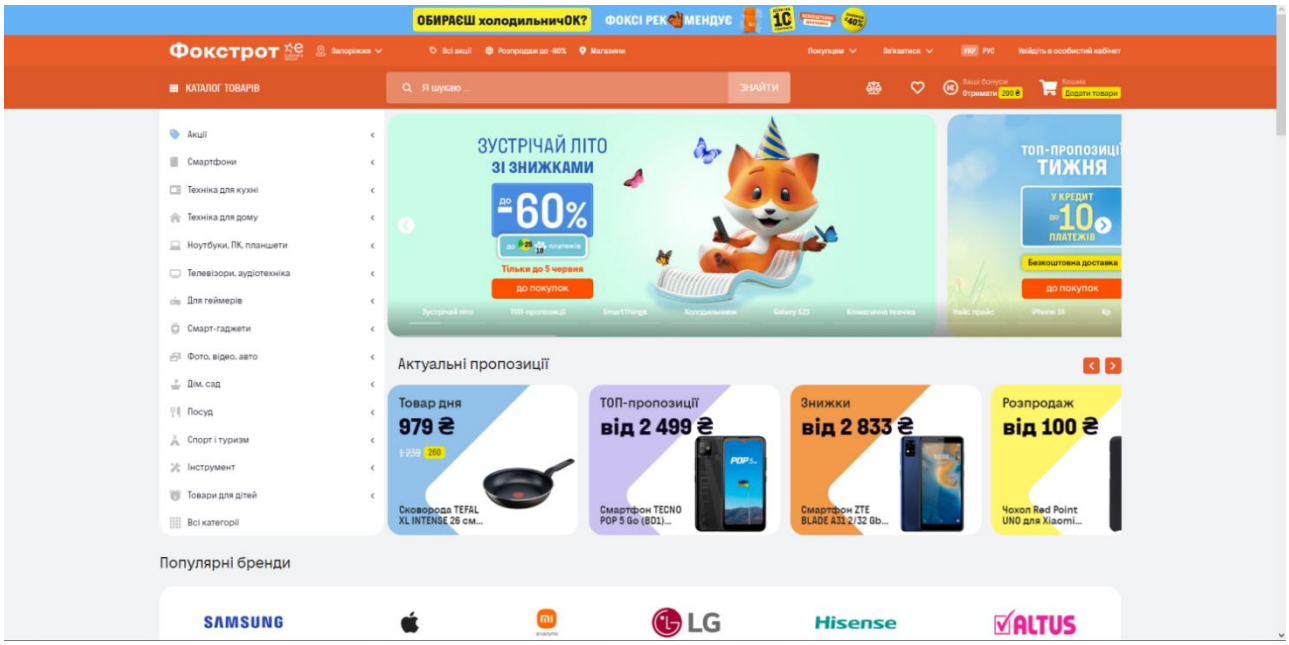

Рис.*4 Інтерфейс Foxtrot*

Allo – це відомий український Інтернет-магазин, спеціалізований на продажі електроніки, мобільних пристроїв, комп'ютерної техніки, побутової техніки, аксесуарів та інших товарів для технологій. Заснований у 1992 році,

Allo завоював довіру та популярність серед споживачів українського ринку [8].

Один з головних принципів Allo – це поєднання високоякісного товару з надійним обслуговуванням. Вони співпрацюють зі світовими виробниками й брендами, щоб пропонувати своїм клієнтам тільки оригінальні та сертифіковані товари. Асортимент Allo охоплює широкий спектр товарів, включаючи смартфони, планшети, ноутбуки, телевізори, фото – та відеотехніку, побутову техніку, аудіо – та відеоаксесуари та багато іншого.

Інтерфейс магазину можемо побачити на рисунку 5.

Переваги:

- Allo пропонує широкий вибір товарів, зокрема мобільних пристроїв, що привертає увагу клієнтів.
- Веб-сайт та мобільні додатки Allo мають сучасний дизайн та легку навігацію.
- Користувачі можуть скористатися сервісом онлайн-чату для отримання консультацій в режимі реального часу.

Недоліки:

- Деякі клієнти зазначають, що деякі товари можуть бути дорожчими на вебсайті Allo порівняно з конкурентами.
- Іноді замовлення можуть затримуватися або мати певні проблеми з доставленням.

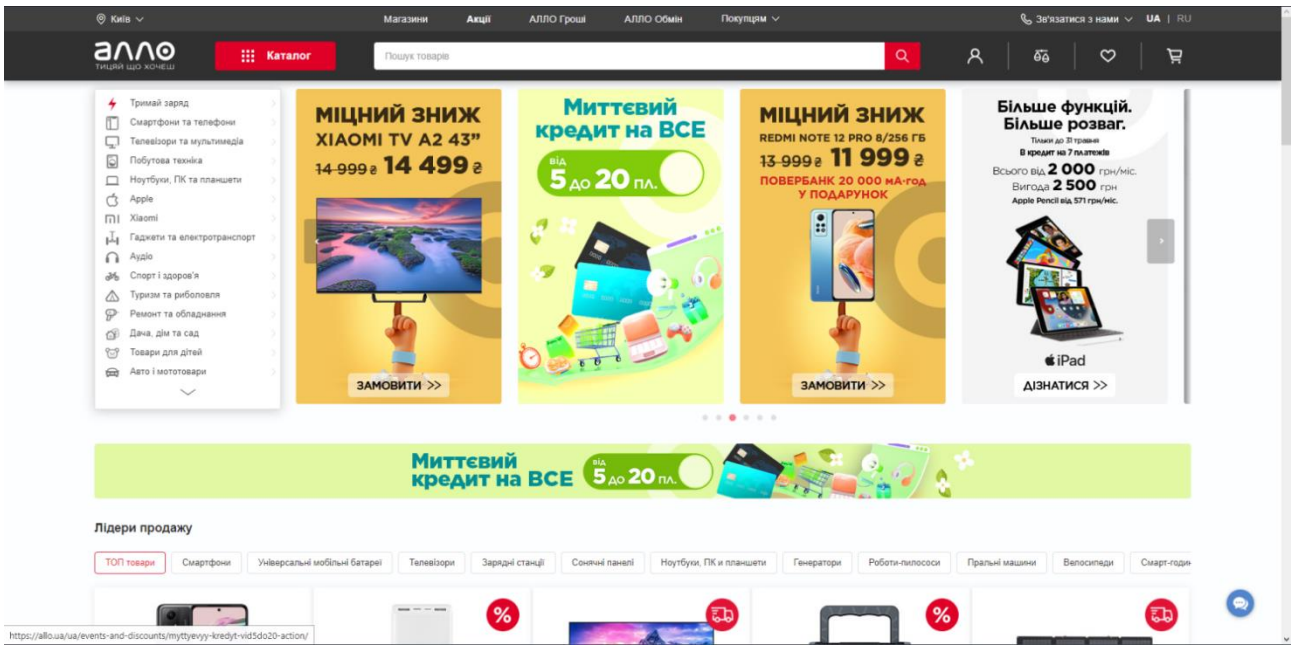

Рис.*5 Інтерфейс Allo*

Переглянувши та дослідивши програмні продукти-аналоги можемо побачити які переваги має Інтернет-магазин, створений за допомогою Telegram Bot API:

Зручність і швидкість доступу:

Завдяки використанню Telegram Bot API користувачі зможуть отримати доступ до Інтернет-магазину прямо з месенджера Telegram, що робить процес шопінгу зручним та швидким.

Немає потреби завантажувати та встановлювати окремий мобільний додаток, оскільки Telegram вже доступний на багатьох платформах.

Простота використання:

Telegram Bot API дозволяє розробити інтуїтивно зрозумілий та простий у використанні інтерфейс для користувачів.

Можливість надсилати команди та взаємодіяти з ботом через текстові повідомлення дозволяє швидко знайти та придбати потрібні товари.

Персоналізація та рекомендації:

За допомогою Telegram Bot API можна реалізувати функцію персоналізованих рекомендацій товарів на основі попередніх покупок, інтересів користувача тощо.

Бот може надавати інформацію про акції, знижки та новинки, що може бути корисним для користувачів.

Миттєвий зворотний зв'язок:

Завдяки можливості відправляти повідомлення боту у режимі реального часу, користувачі можуть отримувати оперативні відповіді на свої запитання та отримувати підтримку безпосередньо в месенджері Telegram.

Безпека та конфіденційність:

Telegram Bot API використовує шифрування для захисту комунікації між користувачами та ботом, забезпечуючи високий рівень безпеки під час здійснення покупок.

Налагоджені інтеграції та функціональність:

Telegram Bot API має можливість інтеграції з різними сервісами та платіжними системами, що дозволяє забезпечити розширену функціональність магазину.

За допомогою бота можна реалізувати можливість оформлення замовлень, оплату товарів та доставлення безпосередньо в Telegram.

Загалом, Інтернет-магазин, реалізований за допомогою Telegra Bot API, має потенціал забезпечити зручну, швидку та персоналізовану покупку для користувачів, пропонуючи зручний інтерфейс, миттєвий зворотний зв'язок та широкі можливості інтеграції.

1.3 Постановка завдання

Метою кваліфікаційної роботи є розробка Інтернет-магазину засобами Telegram Bot API. Основною задачею полягає у створенні ефективного та зручного застосунку для торгівлі товарами та послугами через популярний месенджер Telegram.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Огляд і аналіз наявних рішень: розгляд та порівняння присутній Інтернет-магазинів, реалізованих з використанням Telegram Bot API. Дослідження функціональності, переваг та недоліків кожного рішення.
2. Вивчення Telegram Bot API: огляд доступних можливостей API, вивчення документації, ознайомлення з основними принципами роботи та можливостями платформи.
3. Визначення вимог до Інтернет-магазину: аналіз потреб користувачів, визначення функціональних та нефункціональних вимог до системи. Розробка інтерфейсу та визначення основних функцій, які повинен виконувати магазин.
4. Проєктування архітектури системи: розробка структури бази даних, вибір технологій для розробки з використанням Telegram Bot API, проєктування архітектури програмного забезпечення.
5. Розробка функціональності: реалізація основних функцій Інтернет-магазину з використанням Telegram Bot API, таких як перегляд товарів, додавання їх у кошик, оформлення замовлення, оплата товару, відстеження статусу замовлення та інші.
6. Документування та написання звіту: описання процесу розробки, використання технологій та інструментів, опис функціональності системи, результати тестування та валідації. Написання висновків та рекомендацій щодо подальшого вдосконалення системи.

Результатом роботи є Інтернет-магазин, реалізований засобами Telegram Bot API, а також напрацьовані рекомендації для використання месенджера Telegram у сфері електронної комерції.

2 ДОСЛІДЖЕННЯ ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Аналіз сучасних технологій для створення клієнтської частини веб застосунків

Створення зручного та оптимального користувацького інтерфейсу це важлива задача яка ставиться перед кожним розробником будь-якого веб-застосунка. Для створення користувацького інтерфейсу сьогодні актуальні такі технології:

1. HTML/CSS: HTML використовується для створення структури сторінок, а CSS дозволяє задати їх зовнішній вигляд та стилізацію. Ці технології є основними для розробки клієнтської частини веб-застосунку та забезпечення зручного користувацького досвіду.
2. JavaScript та його фреймворки: JavaScript є мовою програмування, яка використовується для створення динамічних елементів на веб-сторінках. У разі розробки клієнтської частини Інтернет-магазину можна використовувати JavaScript для реалізації функцій, таких як додавання товарів у кошик, фільтрація, пошук інформації та інші. Крім того, фреймворки, такі як React, Angular або Vue.js, можуть значно спростити розробку клієнтської частини, надаючи зручні інструменти для роботи зі станом застосунку, роутингом, управлінням компонентами та іншими задачами.
3. AJAX і RESTful API: AJAX (асинхронний JavaScript і XML) використовується для асинхронного обміну даними між сервером та клієнтом без перезавантаження сторінки. Це особливо важливо при взаємодії з Telegram Bot API, оскільки ми можемо отримувати асинхронні оновлення від бота. RESTful API надає стандартизований спосіб взаємодії з сервером за допомогою HTTP-запитів. Використання AJAX і RESTful API дозволить ефективно комунікувати з серверною

частиною Інтернет-магазину та отримувати та оновлювати дані без перезавантаження сторінки.

4. Responsive Design: Оскільки користувачі використовують різні пристрої, важливо забезпечити, щоб веб-застосунок був добре відображений на різних пристроях і розмірах екранів. Використання технік responsive design, таких як медіа-запити та адаптивна сітка, дозволить створити інтерфейс, який адаптується до різних пристроїв та забезпечує комфортне користування.
5. UI/UX дизайн: Для створення привабливого та зручного інтерфейсу важливо враховувати принципи UI/UX дизайну. Це містить в собі використання ефективної навігації, зрозумілих інтерфейсних елементів, чітке розміщення контенту та інші аспекти, які забезпечують зручне користування та задоволення потреб користувачів.

Важливим питанням перед створенням веб-застосунка є вибір фреймворку для розробки клієнтської частини. В наш час їх існує багато, але основні серед них це React, Vue та Angular. Для того, що краще зрозуміти який фреймворк краще підходить для розробки Інтернет-магазину засобами Telegram Bot API детально розглянемо кожен з них.

ReactJS є одним з найпопулярніших JavaScript-фреймворків для розробки інтерфейсу користувача (UI). Він був розроблений компанією Facebook з метою створення швидкого, ефективного та простого використання інструменту для розробки вебдодатків [9].

Основні особливості ReactJS:

1. Віртуальний DOM: ReactJS використовує віртуальний DOM, що дозволяє ефективно оновлювати лише потрібні елементи веб-сторінки, зменшуючи навантаження на браузер та покращуючи продуктивність додатка.
2. Компонентна архітектура: ReactJS побудований на основі компонентної архітектури, що дозволяє розбивати інтерфейс на

невеликі, повторно використовувані компоненти. Це полегшує розробку, тестування та підтримку додатків.

3. JSX: ReactJS використовує JSX (розширений синтаксис JavaScript), який дозволяє писати HTML-подібний код безпосередньо в JavaScript-файлах. Це спрощує створення інтерфейсу та полегшує розробку.
4. Односторінкові додатки: ReactJS добре підходить для розробки односторінкових додатків, оскільки дозволяє швидко змінювати вміст сторінок без перезавантаження сторінки.

Переваги ReactJS:

1. Ефективність: Використання віртуального DOM та розподілення обов'язків між клієнтом і сервером забезпечує швидку реакцію додатка і поліпшує продуктивність.
2. Компонентна архітектура: Модульна компонентна архітектура ReactJS сприяє повторному використанню коду, зручному управлінню станом та підтримці додатків.
3. Велике товариство: ReactJS має велику спільноту розробників, що підтримують інструменти, бібліотеки та документацію. Це дозволяє швидко знайти рішення на випадки, коли виникають питання або проблеми.
4. Розширюваність: ReactJS дозволяє легко інтегрувати сторонні бібліотеки та інструменти для розширення можливостей фреймворку.

Недоліки ReactJS:

1. Підвищене використання пам'яті: Використання віртуального DOM може призводити до підвищеного використання пам'яті, особливо для великих додатків. Однак, зазвичай це не є серйозною проблемою, якщо використовуються ефективні підходи до оптимізації.
2. Брак стандартних рішень: ReactJS не має вбудованих рішень для багатьох завдань, таких як маршрутизація, керування станом або форми. Це вимагає використання сторонніх бібліотек або написання власного коду.

3. Ініціалізаційний час: Ініціалізація ReactJS може займати деякий час, оскільки вона вимагає завантаження додаткових файлів та виконання певних операцій.

Усупереч своїм недолікам, ReactJS залишається одним з найпопулярніших фреймворків завдяки своїм перевагам у швидкості, ефективності та модульності. Він широко використовується в індустрії веб-розробки та має велику екосистему інструментів та бібліотек, що допомагають розробникам ефективно створювати складні та масштабовані додатки.

Vue.js є прогресивним JavaScript-фреймворком для розробки інтерфейсу користувача (UI). Він був створений з метою полегшення розробки вебдодатків та створення ефективних інтерфейсів [10].

Основні особливості Vue.js:

1. Прогресивність: Vue.js побудований на основі прогресивного підходу, що дозволяє поступово впроваджувати його в теперішні проєкти. Ви можете використовувати Vue.js як бібліотеку для розробки конкретних частин сторінки або використовувати його як повноцінний фреймворк для створення цілих додатків.
2. Компонентна архітектура: Vue.js має компонентну архітектуру, що дозволяє розбивати інтерфейс на повторно використовувані компоненти. Це спрощує розробку, тестування та підтримку додатків.
3. Двостороннє зв'язування даних: Vue.js надає можливість зв'язувати дані між компонентами та шаблонами за допомогою двостороннього зв'язування. Це означає, що зміни в одному місці автоматично відображаються в інших місцях, що полегшує роботу з даними та їх синхронізацію.
4. Директиви: Vue.js має багатий набір вбудованих директив, які дозволяють легко маніпулювати DOM, виконувати анімацію, реагувати на події та інше.

Переваги Vue.js:

1. Легкість вивчення: Vue.js має простий і легкий синтаксис, який легко зрозуміти навіть для початківців. Це дозволяє швидко почати роботу з фреймворком і розробляти додатки без значного порога входу.
2. Флексібельність: Vue.js дозволяє розробникам вибирати, як використовувати фреймворк у їх проєкті. Ви можете використовувати лише необхідні функції та бібліотеки, не перенавантажуючи додаток непотрібними залежностями.
3. Екосистема: У Vue.js є широка екосистема, що складається з додаткових бібліотек, розширень та інструментів, що полегшують розробку та підтримку додатків.
4. Висока продуктивність: Завдяки використанню Virtual DOM і оптимізованих алгоритмів оновлення, Vue.js забезпечує високу продуктивність та швидкодію додатків.

Недоліки Vue.js:

1. Менша популярність: В порівнянні з ReactJS або Angular, Vue.js має меншу кількість розробників та меншу активність спільноти. Це може призвести до меншої кількості документації та ресурсів для розв'язання проблем.
2. Vue.js може мати обмежені можливості у деяких аспектах, таких як робота зі станом, роутинг або масштабування додатків. Однак, воно активно розвивається і відповідає на багато з цих обмежень.
3. Відсутність готових рішень: У порівнянні з Angular, Vue.js може мати меншу кількість готових рішень або шаблонів для певних завдань. Це може змусити розробника писати власний код або шукати сторонні бібліотеки.

В цілому, Vue.js є потужним і гнучким фреймворком, який забезпечує легкість використання та продуктивність. Він підходить для широкого спектра проєктів, від малих односторінкових додатків до складних корпоративних застосунків.

Angular є повноцінним фреймворком розробки вебдодатків, створеним компанією Google. Він використовується для побудови високопродуктивних та масштабованих клієнтських додатків [11].

Основні особливості Angular:

1. TypeScript: Angular базується на TypeScript – суперсеті JavaScript, який додає типізацію та розширені можливості до мови. Це дозволяє забезпечити статичний аналіз коду, зменшити кількість помилок і підвищити надійність застосунку.
2. Компонентна архітектура: Angular побудований на основі компонентної архітектури, де додаток розбивається на невеликі, повторно використовувані компоненти. Це дозволяє полегшити розробку, тестування та підтримку додатків.
3. Директиви: Angular має потужну систему директив, що дозволяють маніпулювати DOM, реагувати на події та виконувати різноманітні додаткові функції.
4. Залежність відірвана від шаблонів: Angular використовує окремі шаблони для відображення даних, що дозволяє розробникам зосередитися на логіці застосунку без прив'язки до розмітки.

Переваги Angular:

1. Повноцінний фреймворк: Angular надає широкий набір інструментів і функціональності для розробки вебдодатків. Він містить в собі готові рішення для маршрутизації, керування станом, валідації форм, автентифікації та багато іншого.
2. Строга архітектура: Angular пропонує чітку структуру та правила організації коду, що сприяє підтримці, швидкості розробки та спільній роботі в команді.
3. Висока продуктивність: Завдяки використанню віртуального DOM, підходів оптимізації та потужних механізмів кешування, Angular забезпечує високу продуктивність та швидкодію додатків.

4. Широка спільнота та підтримка: Angular має активну спільноту розробників, яка надає документацію, плагіни, розширення та інші ресурси для підтримки розробки.

Недоліки Angular:

1. Великий розмір: Фреймворк Angular має великий розмір, що може впливати на час завантаження застосунку. Це особливо помітно для менших проєктів, де розмір фреймворку може бути перевищувати розмір самого додатку.
2. Складність вивчення: Angular має високий поріг входу для нових розробників. Його широкий функціонал і складність деяких концепцій можуть потребувати більше часу для освоєння.
3. Більш жорстка структура: Angular накладає деякі обмеження та стандарти на розробку, що може бути несумісним з вже існуючими проєктами або вимагати додаткового часу для їх адаптації.

У підсумку, Angular є потужним фреймворком, який дозволяє розробникам створювати складні та масштабовані вебдодатки. Він має широкий функціонал і підтримується сильною спільнотою розробників. Однак, його великий розмір і висока складність можуть бути недоліками для деяких проєктів.

Оглянувши та дослідивши всі переваги й недоліки цих фреймворків та провівши порівняння, яке було викладено в таблиці 1, для створення клієнтської частини Інтернет-магазину було обрано ReactJS.

ReactJS було обрано через його компонентну архітектуру, яка полегшує розробку, тестування та підтримку Інтернет-магазину на Telegram Bot API. За допомогою компонентів, можна створювати повторно використовувані елементи інтерфейсу, що прискорює розробку та забезпечує легку масштабованість проєкту.

Таблиця *1*

Порівняння фреймворків React, Vue та Angular

	React	**Vue**	**Angular**
Крива навчання	Середня	Легка	Складна
Продуктивність	Відмінна	Добра	Добра
Екосистема	Велика	Середня	Велика
Спільнота	Велика та активна	Активна	Велика та активна
Гнучкість	Дуже гнучкий	Гнучкий	Менш гнучкий
Перевикористання компонентів	Високе	Високе	Високе
Віртуальний DOM	Так	Так	Ні
Розмір	Легкий	Легкий	Важкий
Підтримка мобільних пристроїв	Так	Так	Так
Інтеграція	Легка	Легка	Середня
Популярні компанії, що використовують	Facebook, Instagram, Airbnb	Xiaomi, Alibaba, Xiaomi, Baidu	Google, Microsoft, IBM

Крім того, ReactJS має широку спільноту розробників, що надає багато ресурсів та підтримки. Це містить в собі документацію, навчальні матеріали, розширення та інструменти, що допомагають забезпечити успішну розробку Інтернет-магазину. Все це робить ReactJS привабливим вибором для розробки клієнтської частини Інтернет-магазину на Telegram Bot API.

2.2 Аналіз NoSQL СУБД

У сучасному світі обробка великих обсягів даних стає все більш розповсюдженою. Традиційні реляційні бази даних можуть виявитись обмеженими в ефективності та масштабованості при обробці таких даних. NoSQL (Not only SQL) бази даних виникли як альтернатива реляційним системам для роботи з великими обсягами даних та немасивною структурою. Далі ми детально розглянемо NoSQL СУБД, їх особливості, типи та використання.

NoSQL бази даних володіють більшою гнучкістю, ніж традиційні реляційні бази даних. Дані, які потрібно зберегти, можуть бути структурованими лише частково або зовсім неструктурованими. Це означає, що дані можна зберігати у будь-якому форматі, і не потрібно наперед визначати їх структуру. NoSQL бази даних легко зберігають такі дані та дозволяють легко додавати нові "колонки" до вже наявних документів або колекцій. Тому NoSQL бази даних використовуються в проєктах, де структура даних невідома наперед або часто змінюється. Ці бази даних часто застосовуються в програмах з великим обсягом даних, де важлива масштабованість. Вони також підходять для вебдодатків, які працюють в реальному часі й вимагають швидкого доступу до даних. Крім того, NoSQL бази даних є ідеальним варіантом для програм, що працюють з хмарними технологіями [12].

NoSQL Системи Управління Базами Даних є сімейством баз даних, які не використовують традиційну модель таблиць реляційних СУБД. Вони зазвичай використовують інші моделі для організації та зберігання даних, такі як ключ-значення, документи, стовпчики або графи.

NoSQL СУБД поділяються на декілька типів, розглянемо їх.

1. Ключ-значення: Цей тип СУБД зберігає дані у вигляді пари ключ-значення, де ключ унікально ідентифікує запис, а значення може бути будь-яким типом даних. Приклади: Riak, Redis, Amazon DynamoDB.
2. Документи: Документ-орієнтовані СУБД зберігають дані у вигляді документів, таких як JSON або XML. Кожен документ може мати різну структуру. Приклади: MongoDB, Couchbase.
3. Стовпці: Ці СУБД організовують дані у вигляді стовпців, а не рядків. Вони підходять для аналітичних роботів та обробки великих обсягів даних. Приклади: Apache Cassandra, HBase.
4. Графи: Графові СУБД зберігають дані у вигляді вузлів та ребер графа. Вони підходять для роботи зі складними залежностями між даними. Приклади: Neo4j, Amazon Neptune.

До особливостей NoSQL СУБД можна віднести наступне:

1. Горизонтальне масштабування: NoSQL СУБД добре масштабуються горизонтально, тобто можуть працювати на розподілених системах з багатьма вузлами.
2. Гнучкість схеми: Більшість NoSQL СУБД не вимагають жорсткої схеми даних. Вони дозволяють зберігати дані різної структури в одній базі даних.
3. Висока продуктивність: NoSQL СУБД зазвичай надають високу швидкодію при операціях читання та запису даних.

Перевагами використання NoSQL СУБД є наступне

1. Масштабованість: NoSQL СУБД можуть ефективно масштабуватись на великі обсяги даних та високі навантаження.
2. Гнучкість: Вони дозволяють зберігати дані різної структури без необхідності строго дотримання схеми бази даних.
3. Висока доступність: Багато NoSQL СУБД підтримують реплікацію та резервне копіювання, що забезпечує високу доступність даних.

Але NoSQL СУБД також мають деякі обмеження:

1. Відсутність складних транзакцій: Деякі NoSQL СУБД не підтримують повноцінні транзакції ACID (Atomicity, Consistency, Isolation, Durability).
2. Менш розвинуті інструменти: У порівнянні з традиційними реляційними СУБД, інструментальна підтримка NoSQL СУБД може бути менш розвиненою.
3. Складні запити: Запити до NoSQL СУБД можуть бути складнішими, оскільки їм властива менш виразна мова запитів.

Як висновок аналізу NoSQL СУБД можемо зазначити, що NoSQL Системи Управління Базами Даних стали популярними в різних областях, де потрібно обробляти великі обсяги даних зі змінною структурою. Вони надають гнучкість, масштабованість та високу продуктивність, але також мають свої обмеження. Вибір NoSQL СУБД залежить від конкретних вимог проєкту та характеру даних, що будуть зберігатись та оброблятись.

Для розробки бази даних проєкта Інтернет-магазина за допомогою Telegram Bot API було вирішено використовувати Документ-орієнтовану СУБД MongoDB.

MongoDB – це документ-орієнтована NoSQL база даних, яка зберігає дані у вигляді гнучких документів у форматі JSON (JavaScript Object Notation) з можливістю вкладених полів. MongoDB створена з урахуванням вимог до масштабованості, продуктивності та простоти використання. Основні характеристики й особливості MongoDB включають [13]:

1. Гнучка схема: MongoDB не вимагає фіксованої схеми даних, що дозволяє легко зберігати різноманітні дані без заздалегідь визначеної структури. Кожен документ може мати свою власну схему, і це дозволяє зручно виконувати зміни в структурі даних без необхідності модифікувати всі наявні записи.
2. Горизонтальна масштабованість: MongoDB надає можливість горизонтального масштабування, що дозволяє розподілити

навантаження на кілька серверів і додавати нові вузли при збільшенні обсягу даних або навантаження.

3. Багатопоточність: MongoDB підтримує багатопотокові операції, що дозволяє одночасно виконувати багато запитів і оновлень. Це особливо корисно в вебдодатках, які мають велику кількість одночасних користувачів.
4. Широкий набір запитів: MongoDB надає потужний механізм запитів, який дозволяє виконувати різноманітні операції, включаючи пошук, сортування, агрегацію, групування та інші.
5. Підтримка індексів: MongoDB підтримує індексування для швидкого доступу до даних. Ви можете створювати індекси на будь-якому полі в документах, що дозволяє оптимізувати запити й покращити продуктивність.
6. Гнучкість реплікації: MongoDB надає можливість налаштування реплікації для забезпечення високої доступності та збереження даних у разі відмови сервера. Реплікація дозволяє створювати копії даних на декількох серверах і автоматично відновлювати роботу в разі відмови головного сервера.

Переваги MongoDB:

1. Гнучка схема даних дозволяє легко працювати зі змінюваними структурами даних.
2. Горизонтальна масштабованість забезпечує високу продуктивність і масштабованість при збільшенні обсягу даних.
3. Швидкість виконання запитів завдяки можливості індексування та багатопотоковій обробці.
4. Простота використання та широкий набір документації та ресурсів сприяють швидкому розвитку проєктів.
5. Підтримка реплікації та висока доступність даних.

Недоліки MongoDB:

1. Вищі вимоги до апаратного забезпечення порівняно з традиційними реляційними базами даних.
2. Відсутність транзакцій в певних версіях MongoDB (хоча у новіших версіях вони частково підтримуються).
3. Документ-орієнтований підхід може вимагати більше уваги до проєктування схеми даних та запитів.
4. Відсутність підтримки JOIN-операцій, що може ускладнити складні запити, які потребують зв'язку даних з різних колекцій.

Все враховуючи, MongoDB є потужною базою даних з великою кількістю функцій та гнучкістю, яка дозволяє ефективно працювати зі схемою даних, що змінюється, та виконувати вимоги високої продуктивності та масштабованості.

2.3 Огляд REST API та WebSocket

REST API та WebSocket – це дві ключові технології веброзробки, які використовуються для забезпечення комунікації між клієнтськими та серверними додатками. Ці технології мають велике значення для розробників програмного забезпечення, оскільки дозволяють забезпечити ефективну та гнучку взаємодію між різними компонентами системи. У цьому огляді ми розглянемо основні концепції, переваги та застосування REST API та WebSocket.

REST API (Representational State Transfer Application Programming Interface) є архітектурним стилем, який використовується для побудови вебслужби на основі протоколу HTTP. REST API використовує різні HTTP-методи (GET, POST, PUT, DELETE) для виконання операцій з ресурсами, які представлені в URL-адресі.

Основні концепції REST API включають:

1. Ресурси: Ресурси є ключовими елементами REST API й можуть бути представлені в URL-адресі. Кожен ресурс має унікальний ідентифікатор і може мати різні стани (наприклад, "user" або "product").
2. HTTP-методи: REST API використовує різні HTTP-методи для виконання операцій з ресурсами. Наприклад, GET для отримання даних, POST для створення нового ресурсу, PUT для оновлення ресурсу, DELETE для видалення ресурсу.
3. Представлення: Дані передаються між клієнтом та сервером у форматі, який може бути легко розуміти та обробити. Зазвичай використовуються формати, такі як JSON або XML.

Переваги REST API:

1. Стандартизація: REST API використовує HTTP-протокол, який є стандартом у світі веброзробки, тому його легко розуміти та використовувати.
2. Масштабованість: REST API можна легко масштабувати для обслуговування багатьох клієнтів зі збереженням стану сервера.
3. Кешування: REST API може використовувати механізми кешування HTTP для зменшення навантаження на сервер та покращення продуктивності.

WebSocket — це протокол комунікації, який забезпечує багаторазову двосторонню комунікацію між клієнтом та сервером. У порівнянні з традиційним HTTP, де клієнтський браузер може лише надсилати запити на сервер і отримувати відповіді, WebSocket дозволяє встановити постійне з'єднання між клієнтом і сервером, що дозволяє обмінюватися даними в реальному часі.

Основні концепції WebSocket:

1. Відкриття з'єднання: WebSocket встановлює постійне з'єднання між клієнтом і сервером, що дозволяє обмінюватись даними в реальному часі.

2. Двостороння комунікація: WebSocket дозволяє як клієнту, так і серверу надсилати повідомлення один одному без необхідності оновлення сторінки або використання запитів-відповідей.
3. Ідентифікатори сесій: Кожному WebSocket-з'єднанню надається унікальний ідентифікатор, який може бути використаний для відстеження сесій та обміну даними між клієнтом і сервером.

Переваги WebSocket:

1. Реальний час: WebSocket забезпечує миттєвий обмін даними між клієнтом і сервером без затримок, що дозволяє реалізувати функціонал в реальному часі, такий як чат, потокова передача даних тощо.
2. Зменшення навантаження: WebSocket використовує постійне з'єднання, що дозволяє уникнути зайвих запитів-відповідей і зменшити навантаження на сервер та мережу.
3. Ефективність: WebSocket має мінімальний накладний розхід у порівнянні з HTTP, оскільки використовує одне з'єднання для багаторазової комунікації.

2.4 Огляд Telegram Bot API

Telegram Bot API — це інтерфейс програмування застосунків (API), який надається Telegram для розробки ботів [14]

Боти в наш час є дуже розвинутим інструментом для покращення життя людей, і їх можливо застосовувати в різних сферах життя. На рисунку 6. наведено в яких сферах можливо застосовувати чат-боти.

Чат-бот став незамінним помічником у сфері електронної комерції, оскільки він дозволяє уникнути необхідності збільшувати кількість операторів для обробки кожного запиту, забезпечуючи початкову онлайн консультацію щодо типових запитів користувачів. На інформаційних порталах і онлайн довідниках такі програми допомагають знайти потрібну інформацію за кілька секунд. Чат-боти активно використовуються для

навчання іноземних мов, допомагають самостійно опановувати англійську та інші мови, навіть оцінюють рівень знань своїх учнів. Крім того, чат-боти можуть грати в ігри з користувачем та надавати інформацію про правила гри [15].

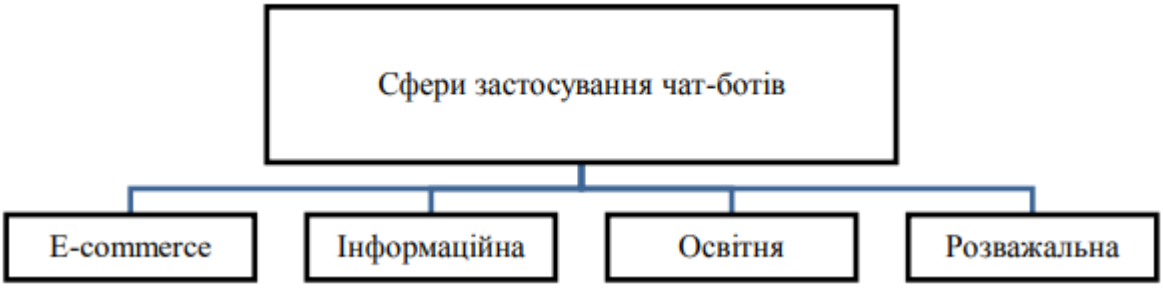

Рис.*6 Сфери застосування чат-ботів*

Нижче наведено детальний опис основних команд Telegram Bot API:

1. getUpdates: Ця команда дозволяє отримувати оновлення з серверів Telegram. Ви можете викликати цю команду, щоб отримувати оновлення. Приклад використання команди наведено в Лістингу 1.

Лістинг *1. Приклад команди getUpdates*

```
 bot.onText(/\/start/, (msg) => {
  const chatId = msg.chat.id;
  bot.getUpdates({offset: -1}).then((updates) => {
    // Обробка отриманих оновлень
    updates.forEach((update) => {
      // Додаткова логіка обробки оновлень
    });
  });
});
```

2. sendMessage: Використовується для надсилання повідомлень користувачам. Ви можете вказати отримувача (чат або користувача), текст

повідомлення та додаткові параметри, такі як розмітка тексту, клавіатура, прикріплення і т. д. Приклад використання наведено в Лістинг 2.

Лістинг *2. Приклад команди sendMessage*

```
bot.onText(/\/sayhello/, (msg) => {
  const chatId = msg.chat.id;
  bot.sendMessage(chatId, 'Привіт, я бот Telegram!');
});
```

3. forwardMessage: Команда передачі повідомлення дозволяє переслати повідомлення з одного чату в інший. Принципи адресації чатів у Telegram базуються на використанні унікальних ідентифікаторів чатів. Кожен чат, будь то приватний чат з користувачем або груповий чат, має свій унікальний ідентифікатор, який використовується для адресації та взаємодії з ним. Тому в команді треба вказати унікальні ідентифікатори та ідентифікатор повідомлення. Приклад використання наведено в Лістингу 3.

Лістинг *3. Приклад використання команди forwardMessage*

```
bot.onText(/\/forward/, (msg) => {
  const chatId = msg.chat.id;
  const messageId = msg.message_id;
  const targetChatId = 'TARGET_CHAT_ID';

  bot.forwardMessage(targetChatId, chatId, messageId);
});
```

4. sendPhoto, sendVideo, sendDocument, sendAudio, sendVoice, sendAnimation, sendSticker: Ці команди дозволяють надсилати різні типи медіафайлів, такі як фотографії, відео, документи, аудіо, голосові повідомлення, анімацію та наліпки. Ви можете передавати файл з локального сховища вказавши його розташування у сховищі, або за URL. Приклад використання наведено в Лістингу 4.

Лістинг *4. Приклад використання команди sendPhoto*

```
bot.onText(/\/sendphoto/, (msg) => {
  const chatId = msg.chat.id;
  const photoPath = 'path/to/photo.jpg';

  bot.sendPhoto(chatId, fs.readFileSync(photoPath));
});
```

5. sendLocation: Використовується для надсилання географічного становища користувачеві. Ви можете вказати координати широти та довготи, назву місця, а також додаткові параметри, такі як період дії місцеперебування. Приклад використання команди наведено в Лістингу 5.

Лістинг *6. Приклад використання команди sendLocation*

```
bot.onText(/\/sendlocation/, (msg) => {
  const chatId = msg.chat.id;
  const latitude = 40.7128; // Широта
  const longitude = -74.0060; // Довгота

  bot.sendLocation(chatId, latitude, longitude);
});
```

6. sendContact: Команда надсилання контакту дозволяє відправити контактну інформацію користувачеві. Вам потрібно вказати номер телефону та ім'я контакту. Приклад використання наведено в Лістингу 7.

Лістинг *7. Приклад використання команди sendContact*

```
bot.onText(/\/sendcontact/, (msg) => {
  const chatId = msg.chat.id;
  const phoneNumber = '+1234567890';
  const firstName = 'John';

  bot.sendContact(chatId, phoneNumber, firstName);
});
```

7. sendChatAction: Використовується для відображення користувачеві статусу дії бота в чаті. Наприклад, "набирає повідомлення", "завантажує фото" і т.д. Приклад використання наведено в Лістингу 8.

Лістинг *8. Приклад використання команди sendChatAction*

```
bot.onText(/\/typing/, (msg) => {
  const chatId = msg.chat.id;

  bot.sendChatAction(chatId, 'typing');
});
```

8. getUserProfilePhotos: Ця команда дозволяє отримувати фотографії профілю користувача. Вам потрібно вказати ідентифікатор користувача. Приклад використання наведено в Лістингу 9.

Лістинг *9. Приклад використання команди getUserProfilePhotos*

```
bot.getUserProfilePhotos(chatId, userId)
  .then((photos) => {
    // Обробка фотографій профілю
    const totalPhotos = photos.total_count;
    const firstPhoto = photos.photos[0][0].file_id;

    console.log(`Кількість фотографій профілю: ${totalPhotos}`);
    console.log(`ID першої фотографії: ${firstPhoto}`);
  })
  .catch((error) => {
    console.log('Помилка отримання фотографій профілю:', error);
  });
```

9. getFile: Команда дозволяє отримувати інформацію про файл на серверах Telegram. Ви можете використовувати цю команду, щоб отримати шлях до файлу за його ідентифікатором. Ідентифікатор файлу (File ID) – це

унікальний ідентифікатор, який призначений кожному файлу, який був надісланий або отриманий через Telegram. Кожен файл, який передається через Telegram, має свій власний унікальний ідентифікатор. Ідентифікатор файлу зазвичай представляється як рядок символів, наприклад, **AgACAgQAAxkDAAIBQF-_1H...**. Цей ідентифікатор може бути використаний для отримання додаткової інформації про файл або для отримання самого файлу через команду **getFile**. Приклад використання наведено в Лістинг 10.

Лістинг *10. Приклад використання команди getFile*

```
const fileId = 'file_id'; // Ідентифікатор файлу

bot.getFile(fileId)
  .then((fileInfo) => {
    // Обробка інформації про файл
    const fileUrl = `https://api.telegram.org/file/bot${TOKEN}/${fileInfo.file_path}`;
    console.log('URL файлу:', fileUrl);
  })
  .catch((error) => {
    console.log('Помилка отримання інформації про файл:', error);
  });
```

10. kickChatMember, unbanChatMember: Ці команди використовуються для керування користувачами в групових чатах. Ви можете виключати користувачів з чату (kickChatMember) та видаляти обмеження (unbanChatMember). Приклад використання наведено в Лістингу 11.

Лістинг *11. Приклад використання команди kickChatMember*

```
bot.kickChatMember(chatId, userId)
  .then(() => {
```

```
  console.log('Користувач успішно виключений з чату');
 })
 .catch((error) => {
  console.log('Помилка виключення користувача з чату:', error);
 });
```

Це лише деякі з основних команд Telegram Bot API. Існує багато інших команд і можливостей, таких як налаштування клавіатури, створення опитувань, робота з груповими чатами та багато іншого. Детальну документацію з описом всіх команд та їх параметрів можна знайти на офіційному сайті Telegram Bot API [14].

Взаємодія з користувачем в Telegram здійснюється через ботів за допомогою різних способів. Ось декілька детальних способів організації взаємодії з користувачем:

1. Надсилання повідомлень: Найпростіший спосіб взаємодії – надіслати повідомлення користувачеві. Бот може надсилати текстові повідомлення, а також повідомлення з розміткою HTML або Markdown, наліпки, медіафайли тощо. Це дозволяє ботам надсилати інформацію, запитувати відповіді та повідомляти користувачів про оновлення.

2. Клавіатура: Боти можуть використовувати клавіатуру для надання варіантів взаємодії користувача. Telegram підтримує два типи клавіатур: клавіатуру звичайних кнопок та клавіатуру зі вбудованими кнопками. Клавіатура звичайних кнопок відображається під полем вводу повідомлення і дозволяє користувачу обрати одну з передбачених опцій. Клавіатура з вбудованими кнопками може бути відображена непрямо на повідомленні, і кнопки реагують на натискання користувачем.

3. Опитування: Боти можуть створювати опитування для користувачів. Ви можете створювати питання зі списком варіантів відповідей, і користувачі можуть вибирати одну або декілька відповідей. Бот може аналізувати результати опитування та приймати рішення на основі них.

4. Взаємодія з медіа: Боти можуть працювати з різними типами медіа, такими як фотографії, відео, аудіо, голосові повідомлення та документи. Ви можете створювати ботів, які приймають фотографії від користувачів, обробляють їх, генерують меми, розпізнають об'єкти на зображеннях або створюють відео з фотографій.

5. Реакція на команди: Боти можуть реагувати на команди, які вводить користувач. Команди починаються з символу / і можуть використовуватись для виклику певних дій або отримання інформації. Наприклад, /start – команда, яка активує бота та починає взаємодію з ним.

6. Реакція на події: Боти можуть реагувати на різні події, такі як натискання кнопки, надсилання місцеперебування, зміна статусу чату тощо. Ви можете програмувати бота так, щоб він реагував на ці події й виконував певні дії.

Це лише кілька способів організації взаємодії з користувачем в Telegram. За допомогою Telegram Bot API розробники можуть створювати різноманітних ботів з різними функціями та можливостями.

3 РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ ЗА ДОПОМОГОЮ TELEGRAM BOT API

3.1 Опис предметної області

Для автоматизації роботи Інтернет-магазинів та полегшення процесу продажу товарів та послуг, з часом стає все більш популярне використання чат-ботів. Telegram Bot API надає зручний та потужний інтерфейс для розробки та інтеграції таких ботів в наявних магазинах.

Програмне забезпечення, розроблене на основі Telegram Bot API, дозволяє автоматизувати процеси замовлення товарів, спілкування з клієнтами, надання необхідної інформації та багато іншого. Завдяки інтуїтивному інтерфейсу та можливості взаємодії з клієнтами в реальному часі, Інтернет-магазини, які використовують Telegram Bot API, надають зручну та швидку обробку замовлень, що забезпечує підвищення ефективності та задоволення клієнтів.

За допомогою Telegram Bot API можна створити інтерактивне меню для вибору товарів, відправлення замовлень, оформлення платежів та надання зворотного зв'язку. Клієнти можуть переглядати каталог товарів, додавати їх до кошика, редагувати замовлення та отримувати сповіщення про стан доставлення.

Для забезпечення зручності та персоналізації обслуговування, Telegram Bot API дозволяє зберігати дані про клієнтів, їх замовлення та історію покупок. Це дозволяє магазинам забезпечувати індивідуальний підхід до кожного клієнта та рекомендації товарів на основі їх попередніх покупок.

Окрім того, інтеграція з Telegram Bot API дозволяє магазинам надавати клієнтам швидку підтримку через чат-бот. Клієнти можуть ставити питання, отримувати консультацію щодо товарів та послуг, а також розв'язувати проблеми через зручний та ефективний канал зв'язку.

Також, розробка Інтернет-магазину з використанням Telegram Bot API забезпечує можливість надсилати клієнтам повідомлення про підтвердження замовлення, статус доставлення та оновлення щодо їх покупок. Це покращує комунікацію з клієнтами та надає їм можливість бути в курсі всіх подробиць щодо їх замовлень.

Більшість платіжних систем можуть бути легко інтегровані з Telegram Bot API, що дозволяє клієнтам безпечно та зручно здійснювати оплату за покупки. Власники бізнесу також можуть використовувати аналітичні інструменти, щоб отримати дані про продажі, популярність товарів та поведінку клієнтів. Це дозволяє зробити обґрунтовані рішення щодо асортименту, маркетингових акцій та стратегії розвитку бізнесу.

Загалом, розробка Інтернет-магазину засобами Telegram Bot API дозволяє поліпшити якість обслуговування клієнтів, автоматизувати процеси продажу та збільшити ефективність бізнесу. Використання чат-ботів у поєднанні з Telegram Bot API стає все більш популярним рішенням для сучасних Інтернет-магазинів, які прагнуть надати своїм клієнтам найкращий сервіс та зручність покупок.

3.2 Архітектура системи

Система Інтернет-магазину, заснована на Telegram Bot API, втілює в собі сучасний та ефективний підхід до автоматизації торгівлі онлайн. Інтеграція клієнтської частини, реалізованої з використанням ReactJS, та серверної частини, що базується на NodeJS разом з MongoDB, пропонує розширені можливості для забезпечення високої продуктивності, стабільності та задоволеності користувачів.

Клієнтська частина розроблена з використанням ReactJS – потужного фреймворку для розробки користувацького інтерфейсу, який забезпечує високу швидкість роботи та відзначається його модульністю та гнучкістю. Ця складова системи надає користувачам зручний спосіб переглядати товари,

здійснювати покупки та виконувати операції в режимі реального часу. Реактивний підхід, що використовується у ReactJS, дозволяє плавно оновлювати вміст сторінки без перезавантаження, надаючи позитивний досвід користувачам.

Серверна частина системи, що базується на Telegram Bot API та NodeJS, гарантує надійну та ефективну обробку запитів користувачів. Telegram Bot API забезпечує простоту та ефективність у взаємодії з месенджером Telegram, дозволяючи отримувати та відправляти повідомлення, обробляти замовлення та забезпечувати зворотний зв'язок з клієнтами. NodeJS, як потужна платформа розробки серверних додатків, забезпечує високу швидкість обробки запитів та підтримку багатопотоковості, що дозволяє ефективно масштабувати систему при необхідності.

Окрім того, серверна частина використовує MongoDB для зберігання та управління даними. MongoDB, як NoSQL база даних, забезпечує гнучкість та розширюваність у роботі зі структурованими та неструктурованими даними. Це дозволяє зберігати інформацію про товари, замовлення, користувачів та інші важливі аспекти функціонування Інтернет-магазину у зручному та ефективному форматі.

Всі складові системи інтегровані між собою з використанням HTTP протоколу та REST API, що забезпечує стандартизований та простий спосіб взаємодії між клієнтською та серверною частинами. Ця архітектура дозволяє забезпечити високу надійність, масштабованість та швидкість роботи системи Інтернет-магазину на основі Telegram Bot API.

Отже, система Інтернет-магазину, розроблена з використанням Telegram Bot API, ReactJS, NodeJS та MongoDB, пропонує зручну та функціональну платформу для ефективного торговельного досвіду, задовольняючи потреби сучасних користувачів та забезпечуючи надійну та стабільну роботу всієї системи.

На діаграмі розгортання (Див. Рис. 7) показано фізичне розташування застосунку та взаємодія користувача з системою.

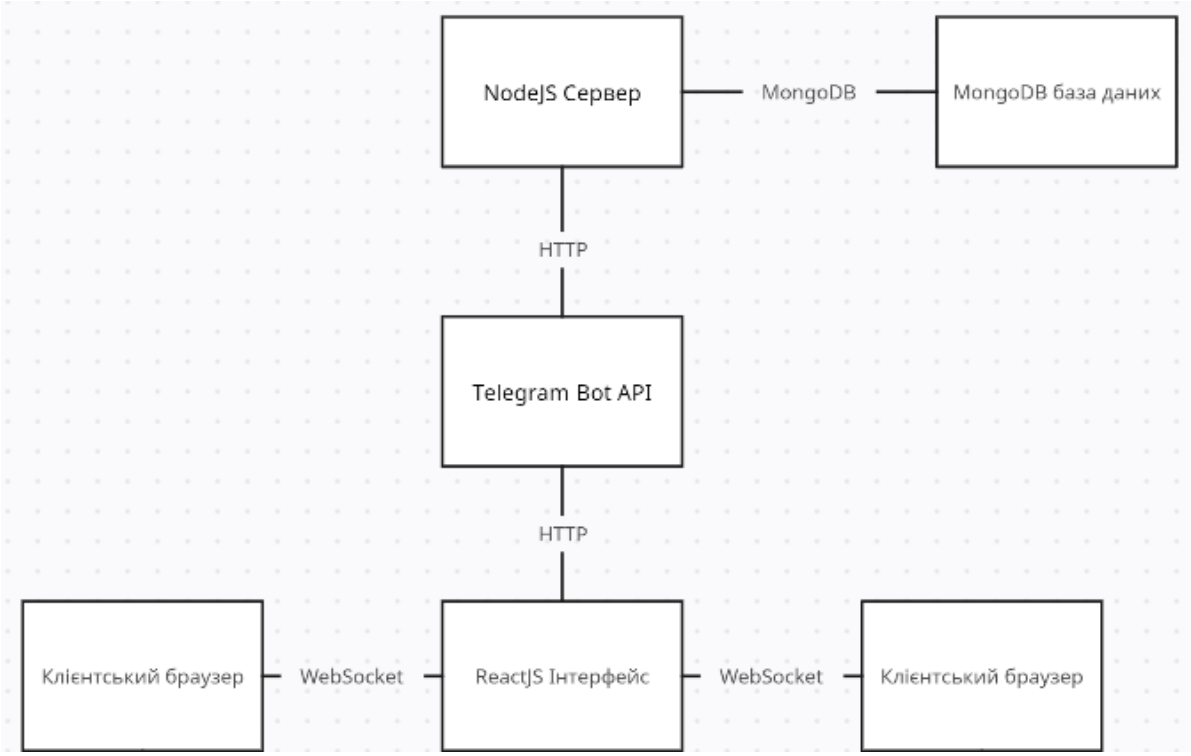

Рис.*7 діаграма розгортання застосунка*

Користувач може взаємодіяти з Telegram Bot API за допомогою HTTP запитів, відправляючи їх з браузера або іншого клієнтського застосунку. Протокол HTTP використовується для здійснення комунікації між клієнтом і сервером. Ви можете виконати різні типи HTTP запитів до API Telegram Bot, такі як POST, GET, PATCH, DELETE і т.д., в залежності від операції, яку ви хочете здійснити. Наприклад:

1) Для відправлення повідомлень, ви можете виконати POST запити на **https://api.telegram.org/bot<token>/sendMessage**, передаючи параметри повідомлення, такі як текст, отримувач і т.д.
2) Для отримання оновлень (нових повідомлень), ви можете виконати GET запити на **https://api.telegram.org/bot<token>/getUpdates**.

Після виконання запиту до API Telegram Bot, ви отримаєте відповідь у форматі JSON (Див. Рис. 8). Ви можете обробити цю відповідь в залежності

від своїх потреб, отримавши, наприклад, інформацію про відправлене повідомлення або результат виконання запиту.

Рис.*8 Приклад відповіді на команду sendMessage*

3.3 Функціональні вимоги системи

Нижче наведені вимоги для розробки та імплементації застосунку Інтернет-магазину з використанням Telegram Bot API, ReactJS, NodeJS та інтеграції з MongoDB.

1. Аутентифікація користувачів:
    - Система повинна надавати можливість користувачам аутентифікуватися для отримання доступу до функцій магазину, налаштувань вподобань, історії замовлень тощо.

- Користувачі повинні мати можливість створювати облікові записи з унікальними ідентифікаторами та паролями.

2. Перегляд товарів:
    - Користувачі повинні мати можливість переглядати доступні товари в магазині.
    - Товари повинні бути представлені з вказанням назви, опису, зображення, ціни та інших характеристик.
3. Додавання товарів у кошик:
    - Користувачі повинні мати можливість додавати товари (один або декілька) у свій кошик для подальшої покупки.
    - Система повинна підраховувати загальну суму замовлення.
    - Користувач повинен мати можливість видаляти товари з кошика.
    - Користувач повинен мати можливість змінювати кількість конкретного товару в кошику (збільшувати або зменшувати).
    - Система повинна забезпечувати збереження обраних товарів користувача між сеансами роботи.
4. Оформлення замовлення:
    - Користувачі повинні мати можливість оформити замовлення, вказавши платіжні дані картки у вікні оплати платіжного шлюзу Liqpay.
    - Система повинна забезпечувати підтвердження замовлення.
5. Керування замовленнями:
    - Адміністратори системи повинні мати можливість переглядати та керувати замовленнями користувачів.
    - Керування замовленнями повинно підтримувати його статуси: В обробці або сплачено.
6. Отримання сповіщень:
    - Користувачі повинні мати можливість отримувати сповіщення про статус замовлень, інформацію про нові надходження, у разі підписки на них та іншу важливу інформацію.

- Сповіщення повинні бути доставлені користувачам у режимі реального часу.

7. Забезпечення безпеки:
    - Система повинна забезпечувати безпеку введених користувачем даних, використовуючи шифрування та інші сучасні методи захисту.
    - Користувачі повинні мати обмежений доступ до чутливої інформації, такої як платіжні дані які він вводить у вікні платіжного шлюзу Liqpay (номер картки, строк дії та CVV).

Ці функціональні вимоги відображають ключові можливості застосунку Інтернет-магазину, які мають бути реалізовані для забезпечення задоволення потреб користувачів та ефективної роботи системи.

3.4 Проектування Інтернет-магазину

Одним з етапів проєктування Інтернет-магазину є створення діаграм використання (Use-Case) для визначення функціональності застосунку.

Для візуалізації функціональності та взаємодії між акторами (користувачами) і системою (Telegram ботом магазину) була створена Use-Case діаграма.

Елементи Use-Case діаграми представленої на рисунку 9 включають:

1. Актори:
    1. Користувач: основний актор, який взаємодіє з ботом магазину.
2. Використані сценарії:
    1. Реєстрація: користувач реєструється в системі, надаючи необхідну інформацію (наприклад, ім'я, електронну пошту, номер телефону тощо).
    2. Перегляд товарів магазину: користувач має можливість переглядати доступні товари в магазині.

3. Додавання товарів в кошик: користувач може додавати товари, які його зацікавили, до свого кошика.
4. Додавання замовлення у свій список замовлень: користувач може додавати обрані товари до свого списку замовлень.
5. Перегляд списку замовлень: користувач може переглядати список своїх замовлень, які він раніше додавав.
6. Оплата замовлення: користувач може обрати замовлення зі свого списку замовлень і здійснити його оплату.

3. Взаємодія між акторами та сценаріями:
    1. Користувач взаємодіє з ботом магазину, щоб зареєструватися, переглянути товари магазину, додати їх в кошик, додати замовлення до списку замовлень та здійснити оплату за замовлення.
    2. Бот магазину обробляє запити користувача, надає необхідну інформацію про товари магазину, дозволяє додавати товари в кошик, додавати замовлення до списку замовлень та здійснювати оплату замовлення.

Рис.*9 UseCase діаграма можливостей взаємодії користувача з магазином*

Для опису взаємодії користувача з Telegram ботом магазину були створені діаграми послідовностей (sequence diagram), (Див. Рис. 10 – 12). Вони показують послідовні кроки, які відбуваються під час реєстрації, входу в магазин, вибору товарів, оформлення замовлення та оплати.

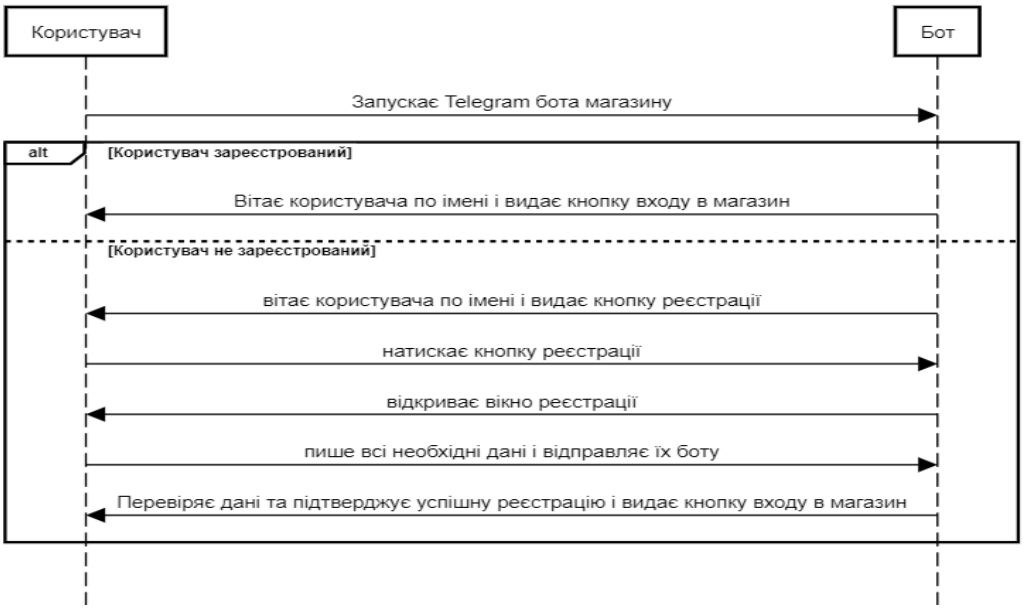

Рис.*10 Діаграма послідовності реєстрації*

Рис.*11 Діаграма послідовності перегляду та оформлення товарів*

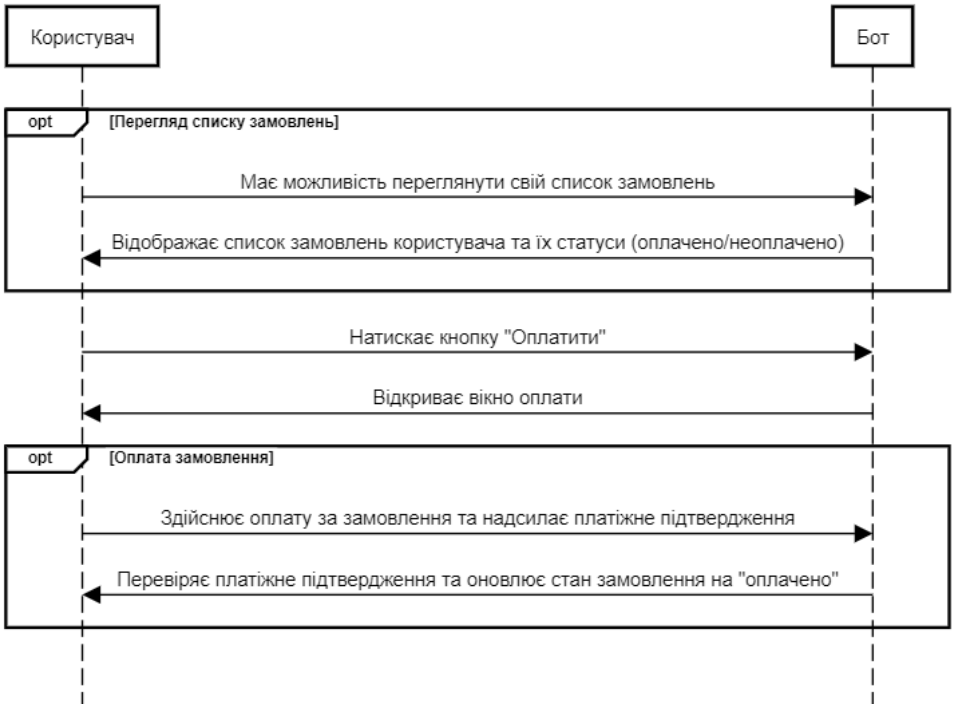

Рис.*12 Діаграма послідовності перегляду списку замовлень і оплата*

База даних MongoDB для цього застосунку містить дві колекції або таблиці – "Користувачі" і "Замовлення". Кожна з цих колекцій має визначені поля, що забезпечують збереження необхідної інформації.

1. Колекція "Користувачі":
    1. Поле "_id": унікальний ідентифікатор користувача.
    2. Поле "user_tg_id": унікальний ідентифікатор користувача в Telegram
    3. Поле "name": ім'я користувача.
    4. Поле "street": вулиця користувача.
    5. Поле "email": електронна пошта користувача.
    6. Поле "password": захешований пароль користувача.
    7. Поле "city": місто користувача.
    8. Поле "state": область користувача.
    9. Поле "zip": поштовий індекс користувача.
    10. Поле "orders": масив замовлень користувача.
2. Колекція "Замовлення":
    1. Поле "_id": унікальний ідентифікатор замовлення.
    2. Поле"user_id": посилання на користувача, який зробив замовлення.
    3. Поле "items": масив, що містить інформацію про товари, включаючи назву, ціну та кількість.
    4. Поле "total_price": загальна сума замовлення.
    5. Поле "status": статус замовлення (наприклад, "В обробці", "сплачено").
    6. Поле "date": дата створення замовлення.

Діаграма бази даних MongoDB, яку ми бачимо на рисунку 13, відображає основну структуру таблиць "Користувачі" і "Замовлення" та визначені поля для збереження інформації про користувачів та їх замовлення в системі Інтернет-магазину. З використанням цих таблиць, застосунок зможе ефективно керувати користувачами, їх особистою інформацією та замовленнями, забезпечуючи швидкий доступ до необхідних даних та зручну обробку замовлень.

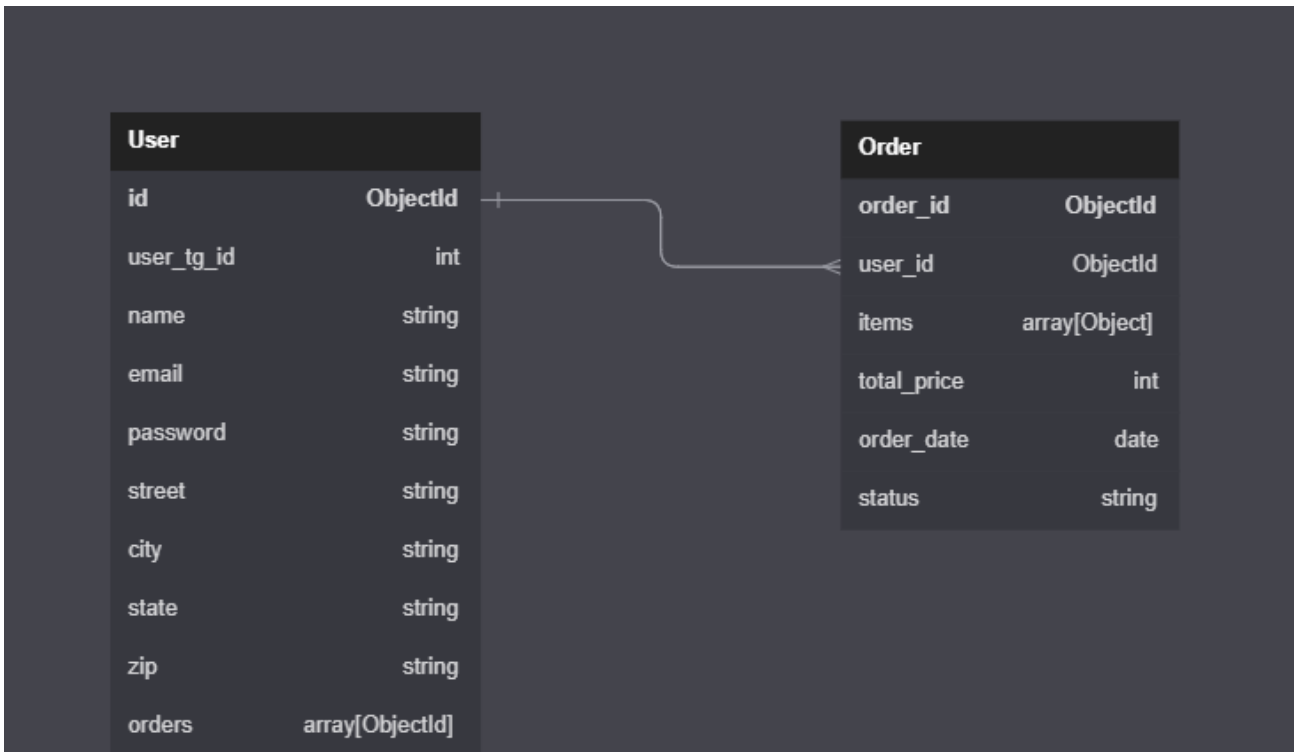

Рис.*13 Діаграма таблиць БД*

3.5 Програмна реалізація застосунка

Розглянемо структуру клієнтської частини застосунку. Клієнтська частина була написана на ReactJS і відповідно до цього має наступну файлову структуру (Див. Рис. 14 – 16).

На рисунку 14 представлена файлова структура компонентів які можливо буде використовувати повторно при створенні клієнтської частини. Було створено наступні компоненти: компонент кошика, компонент товарів кошика, компонент товару, компонент хедеру та компонент модульного вікна.

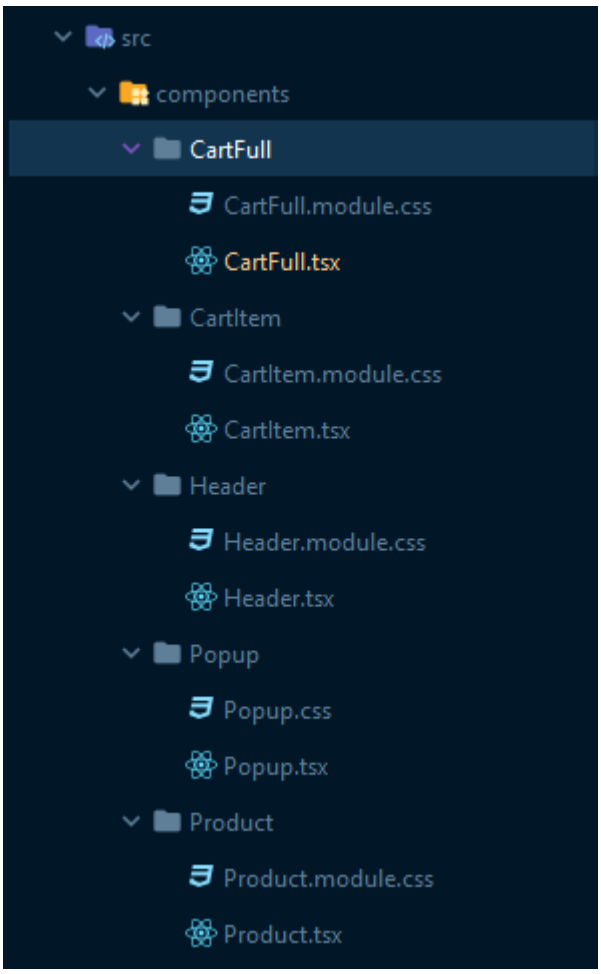

Рис.*14 Файлова структура компонентів React застосунка*

На рисунку 15 представлена файлова структура компонентів готових сторінок застосунку які використовуються для роутингу у застосунку. А саме такі компоненти як: сторінка кошика, сторінка всіх товарів, сторінка товарів по категоріях, сторінка для відображення помилки 404 та сторінка реєстрації користувача.

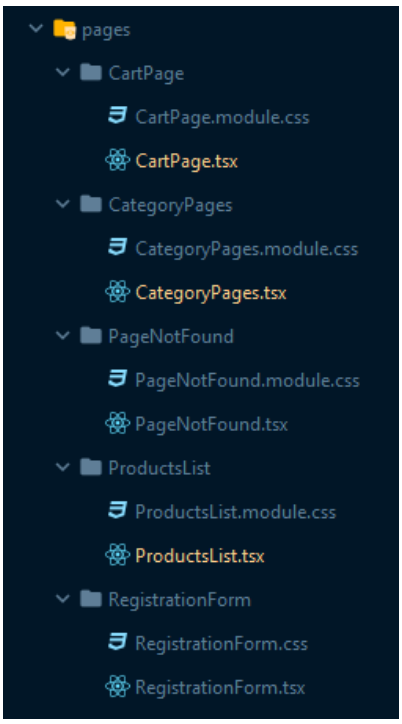

Рис.*15 файлова структура компонентів сторінок React застосунка*

На рисунку 16 представлена структура файлів для керування станом в застосунку. А саме для керування станом модального вікна та елементів кошика

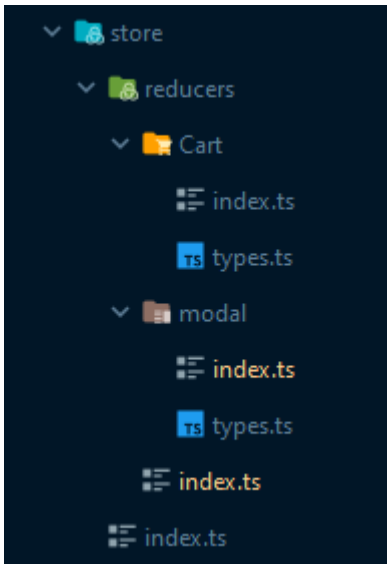

Рис.*16 структура файлів для керування станом*

Система керування станом використовується для ефективного управління даними та станом застосунку. У даному проєкті система керування станом була реалізована за допомогою бібліотеки Redux.

Redux використовує концепцію однонаправленого потоку даних, де стан застосунку зберігається в одному централізованому місці, відомому як "store". Це дозволяє зручно керувати станом застосунку та робити зміни у стані за допомогою дій (actions) і редукторів (reducers).

Як приклад розглянемо store для керування станом кошика.

Лістинг *12. лістинг коду визначення типів для станів модального вікна*

```
interface ArrTypes {
    quantity: number;
    img: string,
    name: string,
    price: number,
    vendorCode: string,
    count: number
}
export interface BasketState {
    itemArr: ArrTypes [],
}
export enum BasketActionEnum {
    ADD_ITEM = "ADD_ITEM",
    REMOVE_ITEM = "REMOVE_ITEM",
}
export interface AddItemAction {
    type: BasketActionEnum.ADD_ITEM;
    payload: any
}
export interface RemoveItemAction {
    type: BasketActionEnum.REMOVE_ITEM;
    payload: {
        vendorCode: string;
    };
}

export type BasketActions = AddItemAction |
RemoveItemAction
```

Лістинг *13. лістинг коду для керування станом кошика*

```
import {BasketActionEnum, BasketActions, BasketState} from
"./types";
```

```
const initialState:BasketState = {
    itemArr:  [],
}

export default function modalReducer(state = initialState,
action: BasketActions): BasketState {
    switch (action.type) {
        case BasketActionEnum.ADD_ITEM:
            return {...state, itemArr:  [...state.itemArr,
action.payload]}
        case BasketActionEnum.REMOVE_ITEM:
            // @ts - ignore
            return {...state, itemArr:
state.itemArr.filter(item => item.vendorCode !==
action.payload)}
        default:
            return state;
    }
}
```

У лістингу кодів 12-13 зазначений код для керування станом кошика, а саме для керування станом додаванням і видалення товарів з кошика. У лістингу 3.1 також визначені типи для полів товарів. В Лістингу 13 наведений код обробки різних дій (**BasketActionEnum**) за допомогою оператора **switch**. Залежно від типу дії, виконується відповідна логіка:

1. У разі дії **BasketActionEnum.ADD_ITEM**, створюється новий стан, розповсюджуючи всі поля поточного стану (**...state**) і додавши новий елемент до **itemArr** за допомогою оператора розгортання (**...state.itemArr, action.payload**). **action.payload** представляє доданий елемент.
2. У разі дії **BasketActionEnum.REMOVE_ITEM**, створюється новий стан, розповсюджуючи всі поля поточного стану (**...state**) і фільтруючи **itemArr** за умовою, що **vendorCode** елемента не збігатися з **action.payload**. Це видаляє елемент з **itemArr** за його **vendorCode**.
3. Якщо дія не збігається з жодним з вищезазначених випадків, повертається поточний стан без змін.

ReactJS дозволяє створювати повторно використовувані компоненти, що робить процес розробки застосунку зручним. Розглянемо вигляд компоненти на прикладі компоненти картки товару представленій на лістингу 14.

Лістинг *14. React компонента картки товару*

```
import React from 'react';
import styles from './Product.module.css';
import { useDispatch } from "react – redux";
type RequestType = {
    price: string | null;
    name: string;
    vendorCode: string;
    pictures: string;
    isActive: boolean;
};
const Product = ({ price, name, vendorCode, pictures,
isActive }: RequestType) => {
    const dispatch = useDispatch();
    const itemArr: (string | boolean | null) [] =
[pictures, price, name, vendorCode, true];
    return (
        <div className={styles.body}>
            <img className={styles.img} src={pictures}
alt="" />
            <h3 className={styles.name}>{name}</h3>
            <p className={styles.Code}>код:
{vendorCode}</p>
            <p className={styles.price}>{price} грн</p>
            <button onClick={() => {dispatch({type:
"SET_MODAL", payload: itemArr})}} className={styles.btn}>
                Придбати
            </button>
        </div>
    );
};
export { Product };
```

Компонента **Product** є реактивним компонентом, реалізованим за допомогою бібліотеки React. Ця компонента відображає картку з інформацією про товар, таку як назва, ціна, код та зображення товару. Вона приймає вхідні параметри **price**, **name**, **vendorCode**, **pictures** та **isActive** типу **RequestType**.

У компоненті використовується зворотній виклик **useDispatch** з бібліотеки **react-redux**, який дозволяє отримати функцію **dispatch** для виклику дій Redux. Далі, створюється масив **itemArr**, який містить значення **pictures**, **price**, **name**, **vendorCode** та **true**.

У методі **render** компонента відображає розмітку, що складається з блоку **div** з класом **body**, який містить зображення товару (**img**), назву (**h3**), код (**p**) та ціну (**p**). Кнопка "Придбати" (**button**) викликає функцію **dispatch** з дією "SET_MODAL" та параметром **itemArr** при натисканні.

Стилі компоненти задаються за допомогою модульних CSS класів. Кожен елемент має відповідний стиль з файлу **Product.module.css**, який імпортується у компоненту.

Узагальнюючи, компонента **Product** є простою карткою, яка відображає інформацію про товар та має кнопку для виклику дії Redux. Вона дозволяє взаємодіяти зі станом застосунку та здійснювати відповідні дії при натисканні на кнопку.

Далі розглянемо файлову структуру серверної частини (бота) написаному на NodeJS та використовуючи БД MongoDB для зберігання даних.

На рисунку 17 показана файлова структура серверної частини (бота), вона складається з таких директорій як: CommandServices, models та services. В директорії CommandServices знаходяться файли для обробки Telegram команд користувача. В директорії models знаходяться файли для визначення моделей бази даних користувачів та замовлень. В директорії services знаходяться файли для обробки бізнес-логіки та додавання в БД користувачів та замовлень.

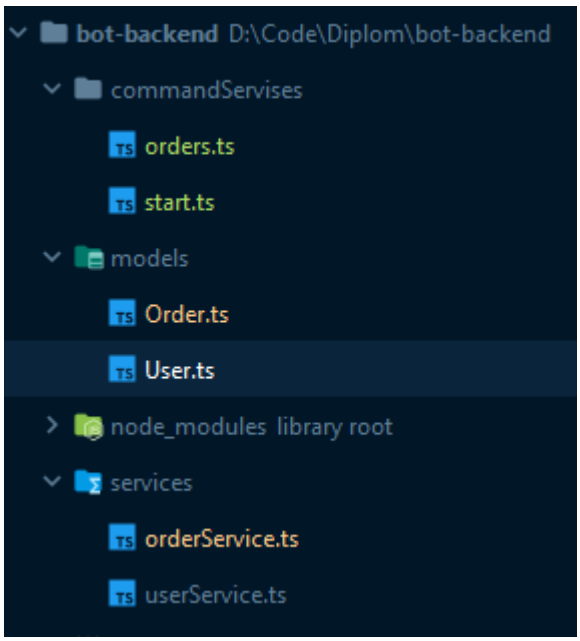

Рис.*17 файлова структура серверної частини Інтернет-магазину*

Розглянемо файл User в директорії models в лістингу 15

Лістинг *15 визначення моделі користувача для БД*

```
const {Schema, model} = require('mongoose');
const userSchema = new Schema({
    user_tg_id: {
        type: Number,
        required: true,
        unique: true
    },
    name: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true,
        unique: true
    },
    password: {
        type: String,
        required: true
    },
    street: {
        type: String,
        required: true
    },
    city: {
        type: String,
```

```
        required: true
    },
    state: {
        type: String,
        required: true
    },
    zip: {
        type: String,
        required: true
    },
    orders:  [{ type: Schema.Types.ObjectId, ref: 'Order'
}]
});
module.exports = model('User', userSchema);
```

За допомогою **require('mongoose')** підключається бібліотека Mongoose, яка надає можливість взаємодіяти з MongoDB. Об'єкт **Schema** використовується для визначення структури даних для колекції "User". Всі поля колекції мають певний тип та обов'язковість заповнення.

Наприклад, поле **user_tg_id** є числовим типом (**Number**), обов'язковим (**required: true**) та унікальним (**unique: true**). Аналогічно, поля **name**, **email**, **password**, **street**, **city**, **state** та **zip** також мають певні типи (**String**) та обов'язковість заповнення.

Крім того, присутнє поле **orders**, яке є масивом об'єктних ідентифікаторів (**Schema.Types.ObjectId**) і вказує на зв'язок з колекцією "Order".

У кінці файлу, за допомогою **module.exports**, експортується модель "User", створена за допомогою **model('User', userSchema)**. Це дозволяє використовувати цю модель в інших частинах програми для звернення до колекції "User" у базі даних MongoDB.

Загалом, наведений код визначає схему колекції "User" у базі даних та експортує модель, що дозволяє взаємодіяти з цією колекцією, забезпечуючи структуру даних та додаткові обмеження.

Далі розглянемо файл Orders в цій директорії на лістингу 16.

Лістинг *16 визначення моделі замовлень для БД*

```
const {Schema, model} = require('mongoose');

const orderSchema = new Schema({
    user_id: {
        type: Schema.Types.ObjectId,
        ref: 'User',
        required: true
    },
    items:  [{
        name: String,
        price: Number,
        quantity: {
            type: Number,
            default: 1
        },
        img: String,
    }],
    total_price: {
        type: Number,
        required: true
    },
    date: {
        type: Date,
        default: Date.now
    },
    status: {
        type: String,
        required: true,
        default: "В обробці"
    }
})

module.exports = model('Order', orderSchema);
```

За допомогою **require('mongoose')** підключається бібліотека Mongoose, яка надає можливість взаємодіяти з MongoDB. Об'єкт **Schema** використовується для визначення структури даних для колекції "Order".

У схемі присутні наступні поля:

- **user_id** є посиланням на об'єктний ідентифікатор користувача з колекції "User". Воно вказує на зв'язок між замовленням і конкретним користувачем. Це поле є обов'язковим (**required: true**).

- **items** є масивом об'єктів, що містять інформацію про товари у замовленні. Кожен об'єкт містить поля **name** (назва товару), **price** (ціна товару), **quantity** (кількість товару) та **img** (шлях до зображення товару). Поле **quantity** має значення за замовчуванням 1 (**default: 1**).
- **total_price** визначає загальну ціну замовлення. Це поле є обов'язковим (**required: true**).
- **date** визначає дату створення замовлення і має значення за замовчуванням – поточну дату та час (**default: Date.now**).
- **status** визначає статус замовлення. Це поле є обов'язковим (**required: true**) і має значення за замовчуванням "Прийнято".

У кінці файлу, за допомогою **module.exports**, експортується модель "Order", створена за допомогою **model('Order', orderSchema)**. Це дозволяє використовувати цю модель в інших частинах програми для звернення до колекції "Order" у базі даних MongoDB.

Загалом, наведений код визначає схему колекції "Order" у базі даних та експортує модель, що дозволяє взаємодіяти з цією колекцією, забезпечуючи структуру даних та додаткові обмеження.

Далі розглянемо код з файлу start в директорії CommandServices який наведено в лістингу 17.

Лістинг *17. код обробки команди start*

```
const start = async (bot, msg, user, webUrl) => {
    const user_tg_id = msg.chat.id;
    await bot.sendMessage(user_tg_id, `Добрий день,
${msg.chat.first_name}, це телеграм бот який дозволить вам
здійснювати покупки з магазину NorlandShop`);
    if (user) {
        await bot.sendMessage(user_tg_id, `З поверненням,
приємних покупок)`, {
            reply_markup: {
                keyboard:  [
                     [{text: "Увійти до магазину", web_app:
{url: webUrl}}],
                ]
            }
        });
```

```
    } else {
        await bot.sendMessage(user_tg_id, `Для продовження
будь ласка пройдіть реєстрацію`, {
            reply_markup: {
                keyboard:  [
                    [{text: "Зареєструватися", web_app:
{url: webUrl + "/registration"}}]
                ]
            }
        });
    }
}
module.exports = {start};
```

Функція **start** отримує наступні параметри: **bot** (об'єкт Telegram Bot API), **msg** (об'єкт повідомлення від користувача), **user** (об'єкт користувача) та **webUrl** (URL вебдодатку).

У першому рядку функція отримує **user_tg_id**, що представляє ідентифікатор телеграм-чату користувача, з повідомлення **msg.chat.id**.

Наступним кроком є надсилання привітального повідомлення користувачеві через бота за допомогою **bot.sendMessage**. Повідомлення містить привітання та пояснення про те, що це телеграм-бот, який дозволяє здійснювати покупки з магазину NorlandShop. Ім'я користувача отримується з **msg.chat.first_name** для більш особистого привітання.

Потім виконується перевірка наявності користувача (**if (user)**). Якщо користувач існує, тобто вже зареєстрований, то бот вітає його з поверненням і надсилає повідомлення з варіантом "Увійти до магазину". Варіанти кнопок відображаються за допомогою **reply_markup** і визначаються масивом об'єктів кнопок.

У разі відсутності користувача, бот надсилає повідомлення з пропозицією зареєструватися через вебдодаток. Повідомлення також містить варіанти кнопок з відповідними URL для реєстрації.

На останок, за допомогою **module.exports**, функція **start** експортується, щоб була доступна для використання в інших частинах програми.

У цілому, наведений код виконує початкову ініціалізацію та взаємодіє з користувачем через Telegram-бота, привітавши його, запропонувавши увійти або зареєструватися в магазині.

Також розглянемо функціонал з файлу orderService на лістингу 18.

Лістинг *18 Функціонал обробки замовлень*

```
const Order = require('../models/Order.ts');
const User = require('../models/User.ts');
async function addOrder(data, user_tg_id){
    for (let i = 0; i < data.items.length; i++) {
        console.log(data.items[i])
        if (data.items[i].quantity === null) {
            data.items[i].quantity = 1;
        }
    }
   const items = data.items;
   const total_price =  data.total_price;
   const date = data.date;
   const status = data.status;
   const user = await User.findOne({user_tg_id});
   if (!user) {
       throw new Error('Користувач не знайдений');
   } else {
       const newOrder = new Order({user_id: user._id,
items, total_price, date, status});
       const savedOrder = await newOrder.save();
       await User.findOneAndUpdate({ user_tg_id }, { $push:
{ orders: savedOrder._id } }, { new: true });
   }
}
async function getUserOrders(user_tg_id) {
    const user = await User.findOne({ user_tg_id
}).populate('orders');
    if (!user) {
        throw new Error('Користувач не знайдений');
    }
    return user.orders || [];
}
async function changeStatusOrder(user_tg_id, orderId,
newStatus) {
    const user = await User.findOne({ user_tg_id });
    if (!user) {
        throw new Error('Користвач не знайдений');
    }
    const order = await Order.findOne({ _id: orderId,
```

```
user_id: user._id });
    if (!order) {
        throw new Error('Замовлення не знайдене');
    }
    order.status = newStatus;
    await order.save();
}
async function isOrderPaid(orderId) {
    const order = await Order.findById(orderId);
    if (!order) {
        throw new Error('Замовлення не знайдене');
    }
    return order.status === 'Сплачено';
}
module.exports = {addOrder, getUserOrders,
changeStatusOrder, isOrderPaid};
```

1. Функція **addOrder** отримує дані про замовлення (**data**) та ідентифікатор користувача (**user_tg_id**). У цій функції перевіряється кількість товарів в замовленні, і якщо кількість не вказана (**null**), встановлюється значення за замовчуванням – 1. Далі зберігаються дані про товари, загальна вартість, дата та статус замовлення. Знаходиться користувач за **user_tg_id** і, якщо він існує, створюється нове замовлення (**newOrder**) з відповідними даними. Замовлення зберігається в базі даних, а його ідентифікатор додається до поля **orders** користувача. В результаті виконання функції повертається збережене замовлення.
2. Функція **getUserOrders** приймає **user_tg_id** і повертає всі замовлення, пов'язані з цим користувачем. Спочатку знаходиться користувач за **user_tg_id**, після чого повертаються всі замовлення (**orders**), пов'язані з цим користувачем. Якщо користувач не знайдений, генерується виключна ситуація.
3. Функція **changeStatusOrder** використовується для зміни статусу замовлення. Вона отримує **user_tg_id**, **orderId** та **newStatus**. Спочатку знаходиться користувач за **user_tg_id**, після чого знаходиться замовлення за **orderId** та перевіряється, чи належить воно до цього

користувача. Якщо замовлення знайдене, оновлюється його статус (**order.status**) на новий (**newStatus**), після чого замовлення зберігається в базі даних.

4. Функція **isOrderPaid** перевіряє, чи замовлення оплачено. Вона приймає ідентифікатор замовлення (**orderId**). Знаходиться замовлення за цим ідентифікатором, і якщо воно знайдене, перевіряється, чи його статус (**order.status**) дорівнює "Сплачено". Якщо так, повертається значення **true**, в іншому випадку – **false**.

Усі функції працюють з моделями **Order** і **User**, що імпортуються з відповідних файлів. Цей код демонструє реалізацію функцій для операцій замовлень та взаємодії з базою даних.

Застосунок використовує API магазину в якому зазначені всі товари й категорії, додаванням товарів до каталогу займається адміністратор магазину.

3.6 Розробка інтерфейсу Інтернет-магазину та Telegram бота

Сучасний зручний інтуїтивно зрозумілий інтерфейс це важлива складова будь-якого вебзастосунку. Привабливий та зручний інтерфейс безпосередньо впливає на ефективність застосунку та задоволення користувачів, а також на їхню загальну задоволеність від використання системи.

Професійний підхід до розробки інтерфейсу включає аналіз потреб та вимог користувачів, створення прототипів та їхнє вдосконалення. Окрім того, використовуються сучасні інструменти та технології, такі як ReactJS, CSS фреймворки та інші, щоб забезпечити швидку та ефективну розробку інтерфейсу.

Результатом розробки є інтерфейс, який дозволяє користувачам з легкістю здійснювати покупки та замовлення через Інтернет-магазин за допомогою Telegram бота. Забезпечуючи зручність, ефективність та

задоволення користувачів, розроблений інтерфейс сприяє підвищенню конкурентоспроможності Інтернет-магазину та залученню більшої кількості клієнтів.

Більш детально інтерфейс представлений на рисунках 18 – 27.

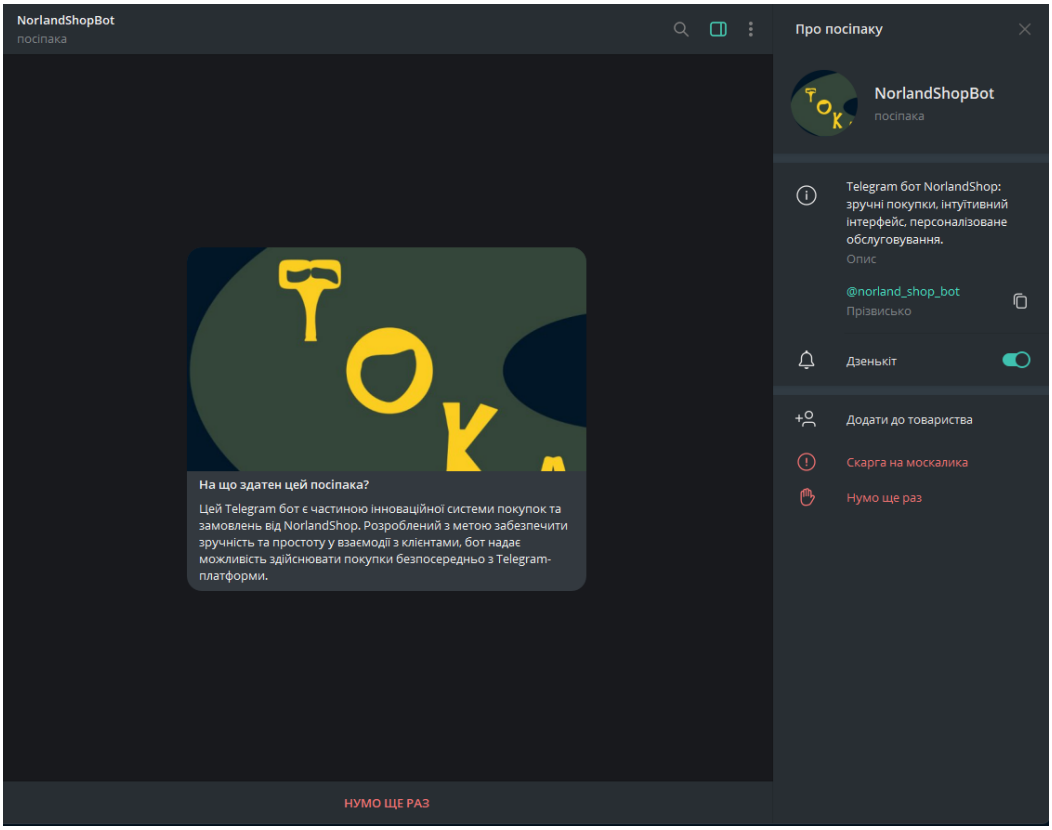

Рис.*18 Telegram бот до початку користування*

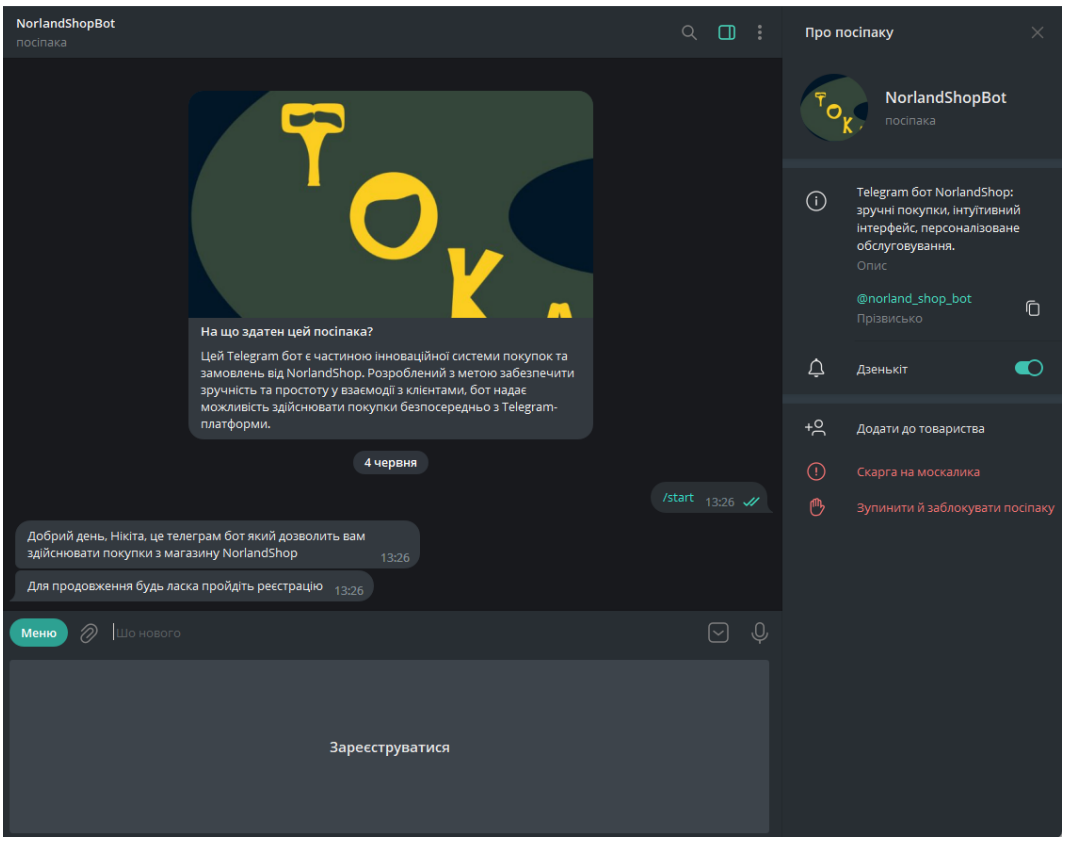

Рис.*19 Telegram бот до реєстрації користувача*

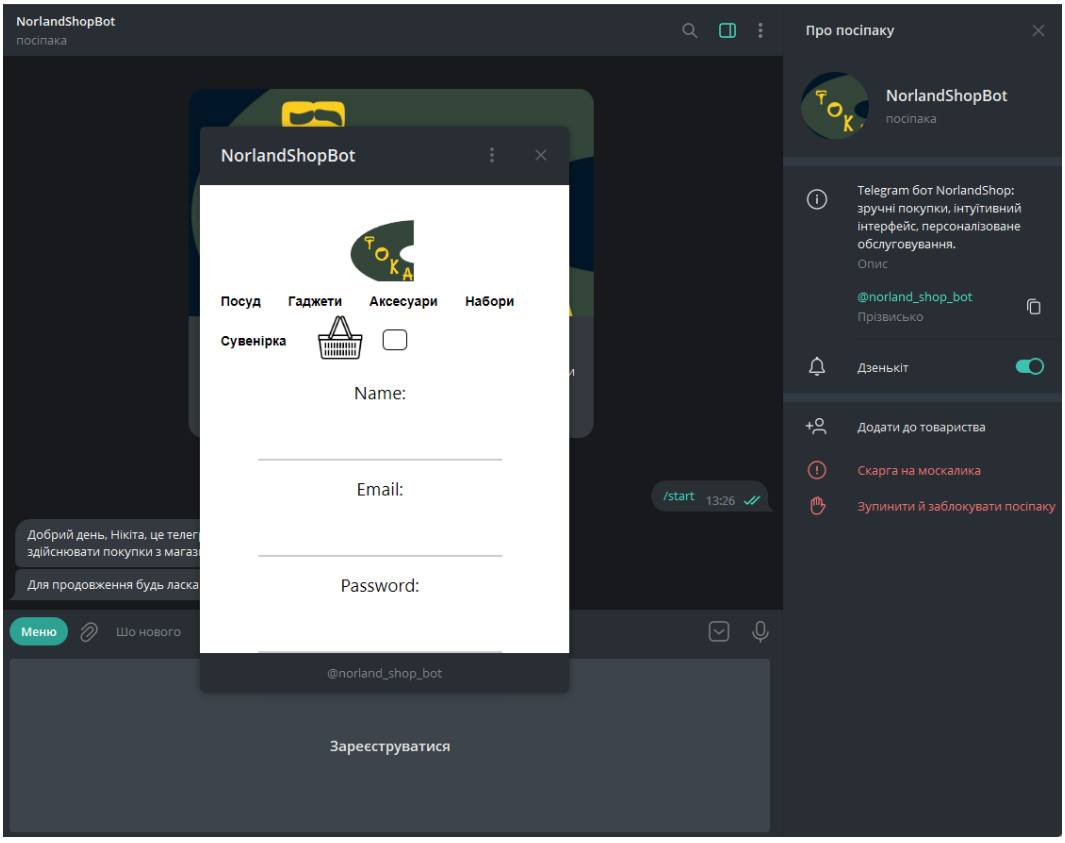

Рис.*20 Telegram бот вікно реєстрації користувача*

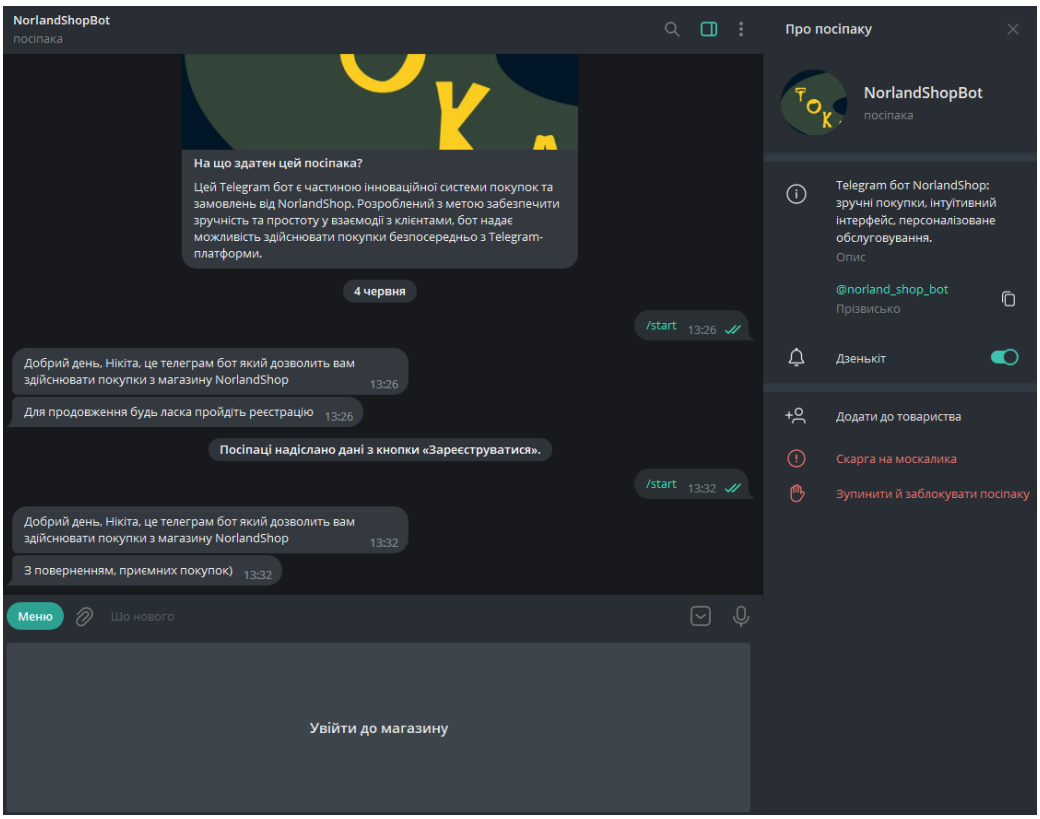

Рис.*21 Telegram бот для зареєстрованого користувача*

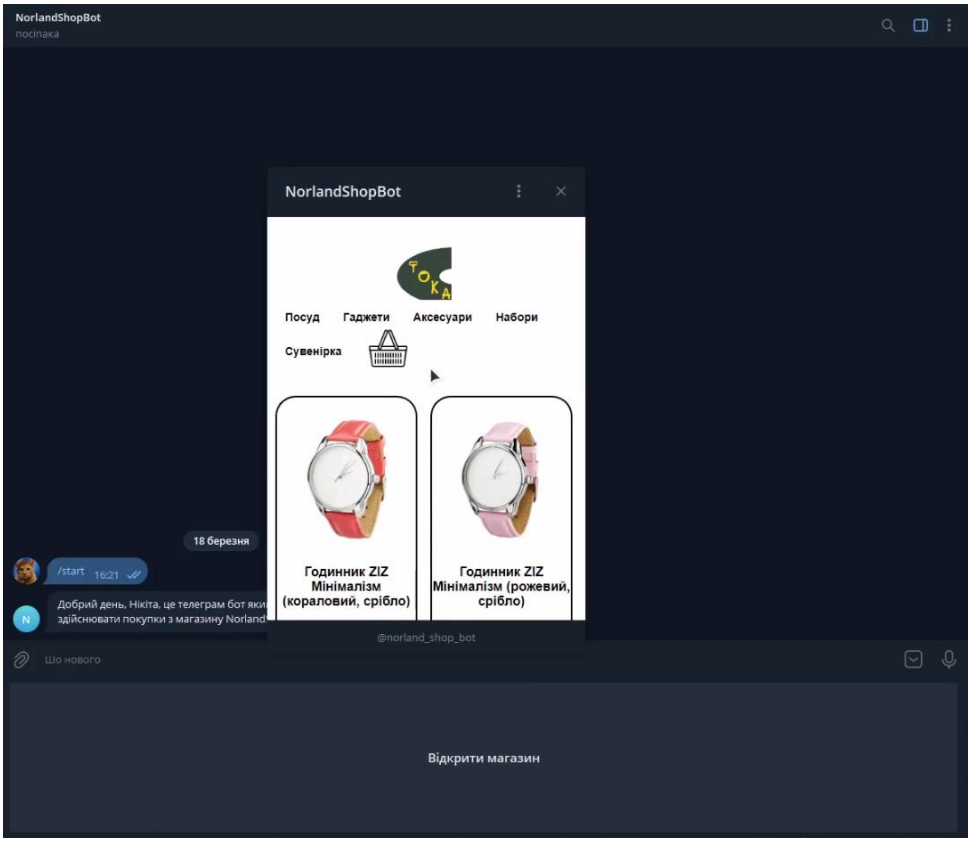

Рис.*22 Telegram бот вікно Інтернет-магазина*

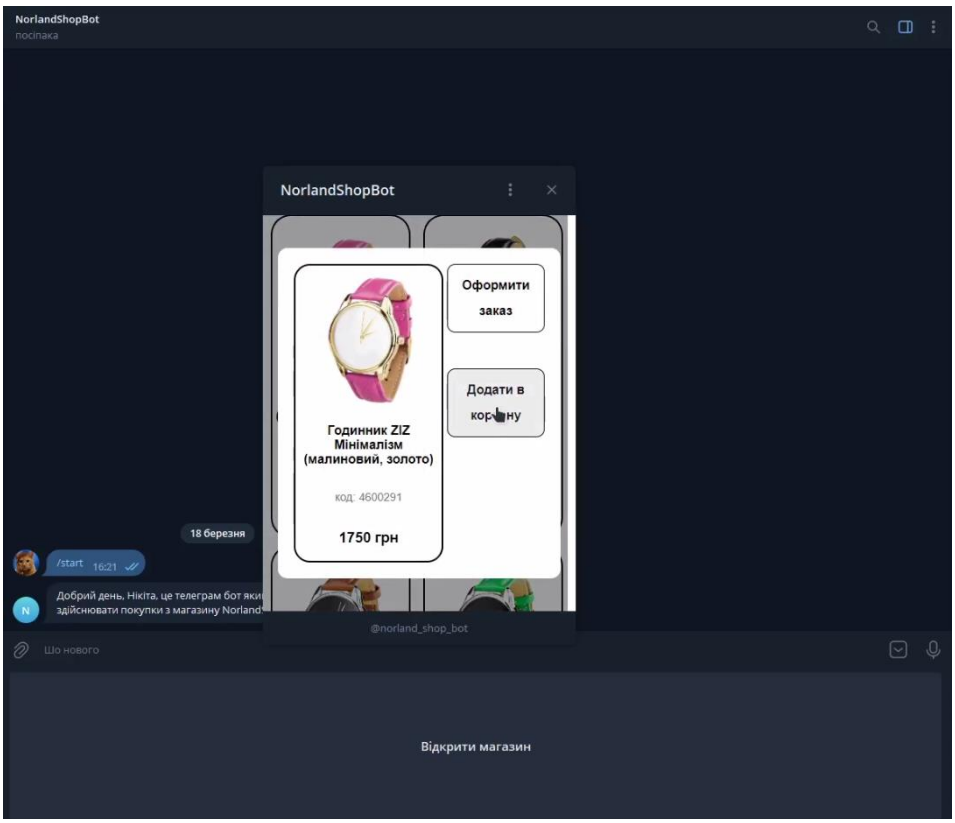

Рис.*23 Telegram бот модальне вікно*

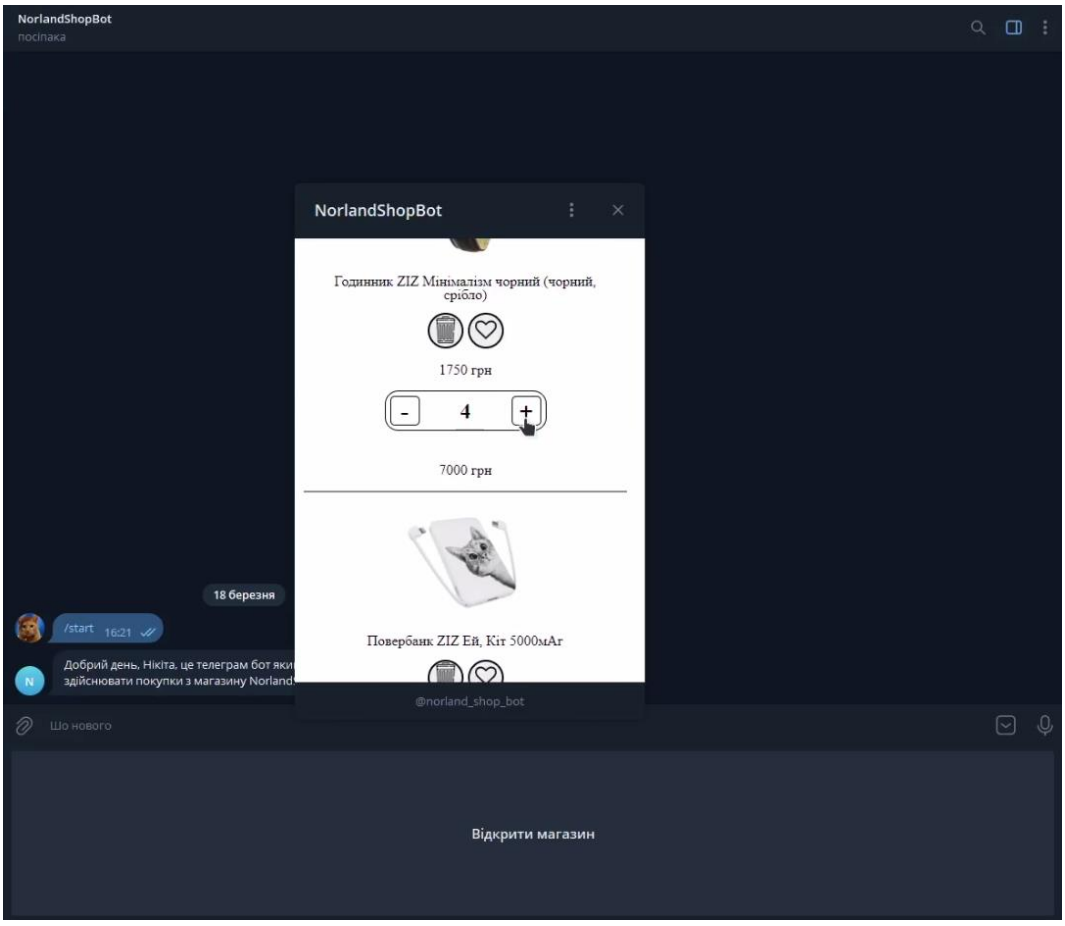

Рис.*24 Telegram бот вікно кошика магазина*

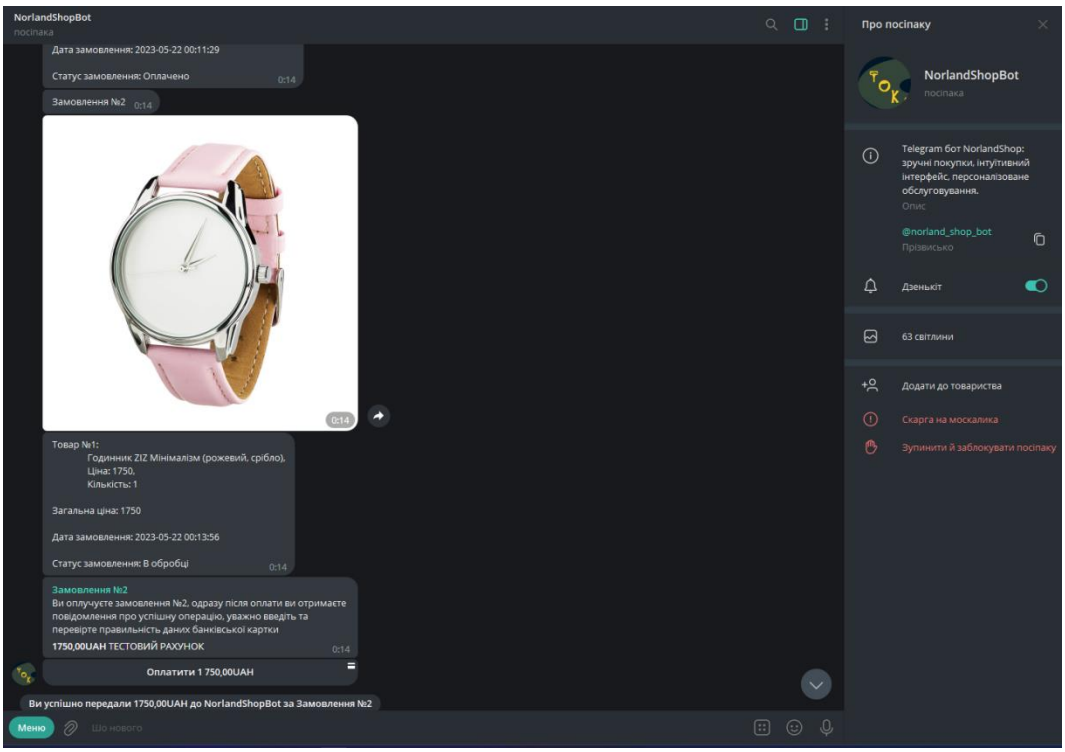

Рис.*25 Telegram бот список замовлень та кнопка оплати*

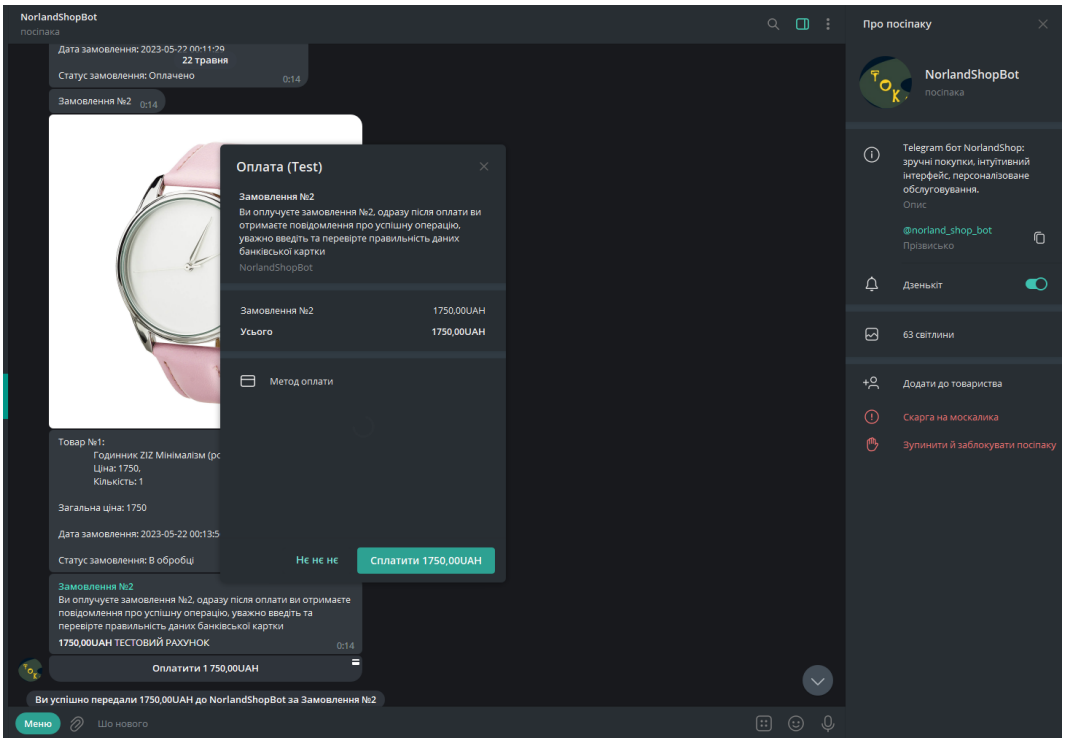

Рис.*26 Telegram бот вікно оплати замовлення*

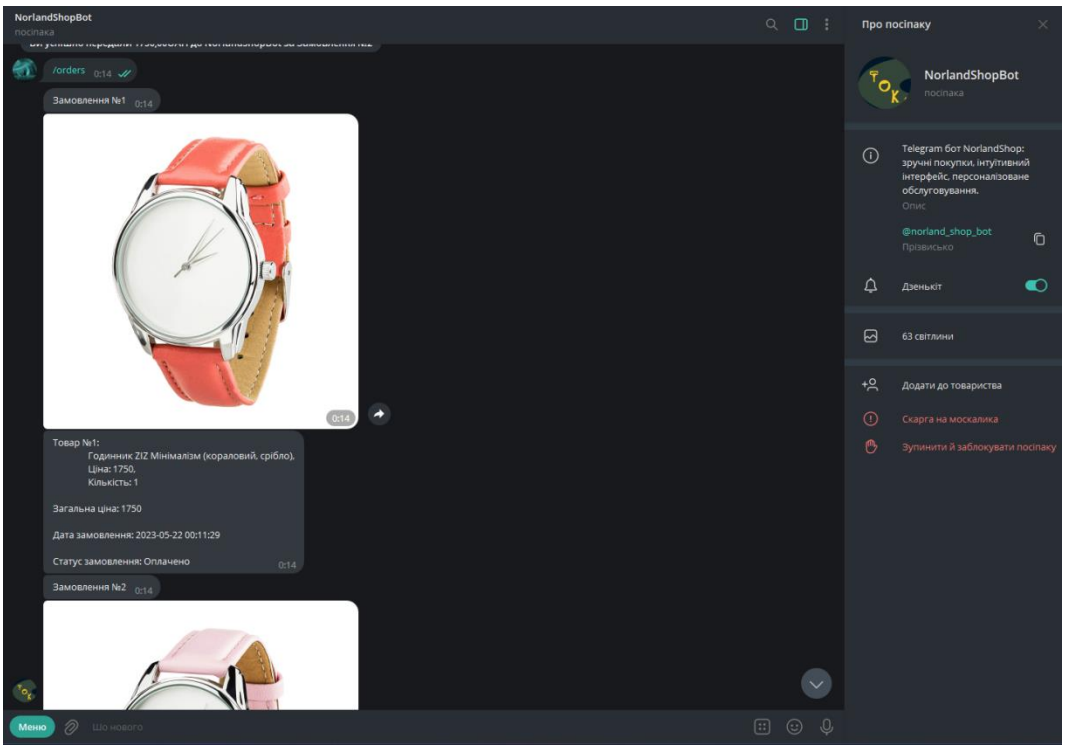

Рис.*27 Telegram бот список замовлень які вже оплачені*

ВИСНОВКИ

1. У процесі виконання кваліфікаційної роботи були детально вивчені джерела, присвячені дослідженням в області електронної комерції та розробці Інтернет-магазинів, а також проведено аналіз аналогічних продуктів.
2. Під час розробки було використано Telegram Bot API, який є потужним інструментом для створення ботів у месенджері Telegram. API надає можливість взаємодії з користувачами, обробки команд та надсилання повідомлень.
3. Під час проєктування архітектури була використана модульна структура, що дозволяє розділити функціональність системи на незалежні компоненти. Це сприяє зручності розробки, тестування та підтримки.
4. Реалізація функціональності включала розробку модулів для взаємодії з Telegram Bot API, обробки команд користувачів та збереження даних про товари та замовлення. Були використані сучасні методи програмування та дотримані кращі практики для забезпечення ефективності та безпеки системи.
5. В результаті роботи було успішно розроблено Інтернет-магазин засобами Telegram Bot API. Система надає можливість користувачам переглядати товари, додавати їх до кошика, оформляти замовлення та здійснювати оплату.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про електронну комерцію : Закон України від 03.09.2015 р. № 675-VIII : станом на 19 листоп. 2022 р. URL: https://zakon.rada.gov.ua/laws/show/675-19#Text (дата звернення: 11.06.2023).
2. Шевченко І. О. Електронна комерція як інструмент забезпечення розвитку цифрової торгівлі. *science, trends and development methods* : Abstracts of VIII International Scientific and Practical Conference, м. Tokyo, 19 – 21 груд. 2022 р. Tokyo, 2022. С. 279.
3. Економічна правда. За рік кількість українців у соцмережах зросла на 7 мільйонів – дослідження. Економічна правда. URL: https://www.epravda.com.ua/news/2021/03/17/672023/ (дата звернення: 28.05.2023).
4. 700 мільйонів користувачів та Telegram Premium. *Telegram*. URL: https://telegram.org/blog/700 – million – and – premium/uk (дата звернення: 28.05.2023).
5. Класифікація чат-ботів / О. Трофименко та ін. *«Системні технології» 2 (139) 2022 «System technologies»* : Регіон. міжвуз. зб. наук. пр. Дніпро, 2022. С. 147 – 159
6. О нас. *Rozetka*. URL: https://rozetka.com.ua/pages/about/ (дата звернення: 30.05.2023).
7. Про компанію. Foxtrot. URL: https://www.foxtrot.com.ua/uk/article/673 (дата звернення: 30.05.2023).
8. Хто ми. Allo. URL: https://work.allo.ua/about – us/ (дата звернення: 30.05.2023).
9. Stefanov S. React: up & running: building web applications. 2-ге вид. Sebastopol : O'Reilly Media, 2021. 230 с.
10. Hanchett E. Vue.js in action first edition. NY : Manning, 2018. 375 с.

11. Fain Y., Moiseev A. Angular Development with TypeScript. 2-ге вид. NY : Manning, 2018. 560 с.
12. Randles, C. (2023, 5 січня). SQL or NoSQL databases – Which one is best for storing data in your organisation? | Weld blog. Weld | Simplify your analytics and data engineering. https://weld.app/blog/sql – or – nosql – databaseswhich – one – is – best – for – storing – data – in – your – organisation
13. Banker K. MongoDB in action. NY : Manning Publications, 2011. 312 с.
14. Telegram bot API. *Telegram APIs*. URL: https://core.telegram.org/bots/api (date of access: 30.05.2023).
15. Чат-боти для бізнесу, сфери застосування, перспективи розвитку. | створення чат-ботів Gerabot. *Розробка та створення чат бота телеграм, viber – компанія Gerabot*. URL: https://gerabot.com/article/chatboti_dlya_biznesa_oblasti_primeneniya_perspektivi_razvitiya (дата звернення: 30.05.2023).
16. Третина населення Землі ніколи не користувалася Інтернетом: звіт ООН. 24 Канал. URL: https://24tv.ua/tech/tretina-naselennya-zemli-nikoli-ne-koristuvalasya-novini-tehnologiy_n1809451 (дата звернення: 06.06.2023).
17. За рік карантину кількість українських користувачів в соцмережах зросла на 7 млн і досягла 60% населення країни | GlobalLogic Ukraine. GlobalLogic Ukraine. URL: https://www.globallogic.com/ua/about/news/social-media-during-quarantine/ (дата звернення: 06.06.2023).
18. Фока М.К., Коломоєць Г.П. доцент, канд. фіз.-мат. наук – науковий керівник. Розробка Інтернет-магазину засобами Telegram Bot API. Збірник наукових праць студентів, аспірантів, докторантів і молодих вчених «Молода наука-2023» / Запорізький національний університет. – Запоріжжя : ЗНУ, 2023. – Т.5. – С. 125-126.

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Фока Микита Костянтинович , студент 4 курсу, форми навчання денної, Інженерного навчально-наукового інституту, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти ipz19bd-15@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «**Розробка Інтернет-магазина засобами Telegram Bot API**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

згоден/згодна на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 14.06.2023 Підпис _____ Фока Микита Костянтинович
(студент)

Дата 15.06.2023 Підпис _____ Безверхий Анатолій Ігорович
(науковий керівник)