НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

“До захисту допущено”
Завідувач кафедри
_____ Наталія АУШЕВА
“___”_____2025 р.

Дипломна робота
на здобуття ступеня бакалавр

За освітньою програмою “Цифрові технології в енергетиці”

Спеціальності 122 “Комп’ютерні науки”

на тему: “Web-система з використанням асистента на основі штучного інтелекту для взаємодії з сайтом”

Виконав: студент 4 курсу, групи ТР-14
Панюк Сергій Сергійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник асистент каф. ЦТЕ, Костянтин ЗДОР _____
(посада, науковий ступінь, вчене звання, ім’я, ПРІЗВИЩЕ) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, ім’я, ПРІЗВИЩЕ) (підпис)

Н.контроль асистент каф. ЦТЕ, Володимир РУДИК _____
(посада, ім’я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ — 2025

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ
Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ
Рівень вищої освіти — перший (бакалаврський)
спеціальність 122 “Комп’ютерні науки”
Освітньо-професійна програма “Цифрові технології в енергетиці”

ЗАТВЕРДЖУЮ
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА
(підпис)
“___” _____2025 р.

ЗАВДАННЯ
на дипломну роботу студенту

Панюку Сергію Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи “Web-система з використанням асистента на основі штучного інтелекту для взаємодії з сайтом”

Науковий керівник Здор Костянтин Андрійович
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “____” _____ 2025 року №_____

2. Термін подання студентом роботи 09.06.2025р.

3. Вихідні дані до роботи мова програмування .NET/C#, мова програмування TypeScript, фреймворк Angular, середовища розробки JetBrains WebStorm та JetBrains Rider.

4. Перелік питань, які потрібно розробити 1) провести аналіз підходів та архітектур інтернет магазинів; 2) проаналізувати існуючи архітектурні рішення клієнт-серверного застосунку 3) виконати аналіз існуючих рішень для інтеграції чат бота на основі штучного інтелекту 4) Розробити застосунок з інтегрованим чат-ботом на основі штучного інтелекту.

5. Орієнтований перелік ілюстративного матеріалу схеми, що описують архітектури клієнт серверних застосунків.

6. Дата видачі завдання 13.09.2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Вибір теми роботи	13.09.2024	Виконано
2	Розробка архітектури та загальної структури системи	14.04 - 04.05.2025	Виконано
3	Розробка клієнт серверного застосунку	28.04-04.05.2025	Виконано
4	Розробка та тренування моделі ШІ	05.05-11.05.2025	Виконано
5	Інтеграція чат-боту в застосунок	12.05-18.05.2025	Виконано
6	Оформлення пояснювальної записки	18.05-31.05.2025	Виконано
7	Захист програмного забезпечення	16.05.25	Виконано
8	Передзахист	28.05.25	Виконано
9	Захист		Виконано

Студент _____ Сергій ПАНЮК
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник _____ Костянтин ЗДОР
(підпис) (ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломна робота містить 14 ілюстрацій, 8 таблиць, 46 джерел в переліку посилань та 1 додаток, загальний обсяг 96 сторінок.

Мета роботи — розробка програмного забезпечення інтернет-магазину з інтегрованим чат ботом на основі штучного інтелекту для взаємодії з сайтом.

Використані методи та засоби: методи машинного навчання, моделі ChatBase, зокрема функціонал платформи .NET, а саме ASP.NET Core. Також для клієнтської частини додатку використовувався фреймворк Angular.

Основний зміст роботи: У цій роботі проаналізовано сучасні підходи до розробки клієнт-серверних веб-додатків із метою створення інтернет-магазину. Розглянуто та порівняно можливі архітектурні рішення, включаючи монолітну та мікросервісну архітектури, а також визначено переваги і недоліки різних технологічних стеків як для серверної, так і для клієнтської частини.

Окрему увагу приділено дослідженню можливостей інтеграції чат-ботів з елементами штучного інтелекту в інтернет-магазин. Проведено огляд технологій розробки таких ботів, їх застосування для покращення користувацького досвіду, автоматизації консультацій та обробки замовлень. У результаті реалізовано прототип чат-бота з використанням AI, інтегрованого у веб-додаток інтернет-магазину, що дозволяє надавати консультації користувачам у реальному часі.

Рекомендації щодо використання: Отримані результати можуть бути використані для побудови веб-застосунків інтернет-магазинів з інтеграцією чат-ботів на основі штучного інтелекту. Такий підхід сприятиме покращенню користувацького досвіду, підвищенню ефективності обслуговування клієнтів, автоматизації відповідей на типові запитання та супроводу процесу покупки. Запропоновані рішення можуть бути адаптовані для широкого кола e-commerce платформ із врахуванням специфіки конкретного бізнесу.

Ключові слова: веб-застосунок, інтернет-магазин, чат-бот, мікеросервіси, нейронна мережа, RNN, LSTM, GRU, машинне навчання, AI.

ABSTRACT

The bachelor's thesis includes 14 illustrations, 8 tables, 46 references, and 1 appendix, with a total length of 96 pages.

The aim of the work is to develop e-commerce software with an integrated AI-powered chatbot for interacting with the website.

Methods and Tools Used: The work employs machine learning methods and ChatBase models, particularly utilizing the .NET platform functionality, namely ASP.NET Core. The client side of the application was developed using the Angular framework.

Main Content of the Thesis: This thesis analyzes modern approaches to the development of client-server web applications with the goal of building an online store. It considers and compares potential architectural solutions, including monolithic and microservice architectures, and identifies the advantages and disadvantages of various technology stacks for both the server and client sides.

Special attention is given to exploring the possibilities of integrating AI-based chatbots into e-commerce platforms. The thesis reviews the technologies used in chatbot development and their application in enhancing user experience, automating customer support, and processing orders. As a result, a prototype AI-powered chatbot integrated into a web-based online store was implemented. This chatbot is capable of providing real-time user assistance.

Recommendations for Use: The obtained results can be applied to the development of web-based e-commerce applications with integrated AI-powered chatbots. This approach improves user experience, increases the efficiency of customer service, and automates responses to frequently asked questions and purchase support. The proposed solutions can be adapted for a wide range of e-commerce platforms, taking into account the specifics of particular businesses.

Keywords: web application, online store, chatbot, microservices, neural network, RNN, LSTM, GRU, machine learning, AI.

ЗМІСТ

ВСТУП 8
1 АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ДО РОЗРОБКИ ІНТЕРНЕТ-МАГАЗИНІВ ТА СИСТЕМ ШІ ДЛЯ ЧАТ-БОТІВ 11
1.1 Методи розробки сучасних інтернет-магазинів 11
1.1.1 Архітектури клієнт-серверних додатків для електронної комерції 11
1.1.2 Технології створення інтерфейсів користувача для інтернет-магазинів 12
1.1.3 Системи управління контентом та бази даних в інтернет-магазинах 13
1.2 Архітектури моделей штучного інтелекту для чат-ботів 15
1.2.1 Рекурентні нейронні мережі для обробки природної мови 15
1.2.2 Трансформери та їх застосування в діалогових системах 17
1.2.3 Гібридні архітектури для інтелектуальних чат-ботів 18
1.3 Сучасні підходи до інтеграції чат-ботів з інтернет-магазинами 20
1.3.1 Аналіз існуючих рішень на ринку електронної комерції 21
1.3.2 Методи персоналізації взаємодії з користувачами 24
1.3.3 Оптимізація бізнес-процесів за допомогою ШІ-асистентів 25
Висновки до розділу 26
2 ВДОСКОНАЛЕННЯ МЕТОДІВ РОЗРОБКИ ІНТЕРНЕТ-МАГАЗИНУ З ІНТЕГРОВАНИМ ЧАТ-БОТОМ 28
2.1 Архітектура системи інтернет-магазину 28
2.1.1 Проектування мікросервісної архітектури 28
2.1.2 Організація взаємодії компонентів системи 32
2.1.3 Забезпечення масштабованості та відмовостійкості 36
2.2 Модель даних та система управління базами даних 38
2.2.1 Проектування схеми бази даних для інтернет-магазину 38
2.2.2 Оптимізація запитів та індексування 40
2.2.3 Забезпечення цілісності та безпеки даних 43
2.3 Архітектура ШІ-моделі для чат-бота 46
2.3.1 Вибір та обґрунтування архітектури нейронної мережі 47

2.3.2 Методи обробки запитів природною мовою 49
2.3.3 Алгоритми формування відповідей та рекомендацій 53
Висновки до розділу 56
З РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНА ОЦІНКА ІНТЕРНЕТ-МАГАЗИНУ З ЧАТ-БОТОМ НА ОСНОВІ ШІ 58
3.1 Програмна реалізація інтернет-магазину 58
3.1.1 Розробка серверної частини системи 58
3.1.2 Створення адаптивного інтерфейсу користувача 60
3.1.3 Впровадження системи платежів та управління замовленнями 63
3.2 Розробка та навчання моделі ШІ для чат-бота 66
3.2.1 Підготовка навчальних даних та попередня обробка 66
3.2.2 Налаштування гіперпараметрів та процес навчання моделі 69
3.2.3 Оптимізація моделі для роботи в умовах обмежених ресурсів 72
3.3 Експериментальна оцінка ефективності розробленої системи 75
3.3.1 Методологія тестування та критерії оцінки 75
3.3.2 Аналіз продуктивності та надійності інтернет-магазину 78
3.3.3 Оцінка точності та релевантності відповідей чат-бота 81
Висновки до розділу 84
ВИСНОВКИ 85
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 87
ДОДАТОК А 94

ВСТУП

Актуальність теми. У сучасному цифровому суспільстві електронна комерція стала невід'ємною частиною глобальної економіки, демонструючи стабільне зростання та трансформуючи традиційні бізнес-моделі. Згідно з даними Statista, глобальний ринок електронної комерції досягнув обсягу понад 5,7 трильйонів доларів США у 2022 році, з прогнозованим зростанням до 8,1 трильйонів доларів до 2026 року. Водночас зростаючі вимоги користувачів до якості обслуговування та персоналізації взаємодії з інтернет-магазинами створюють нові виклики для розробників. У цьому контексті інтеграція технологій штучного інтелекту, зокрема інтелектуальних чат-ботів, у системи електронної комерції набуває особливої актуальності. Чат-боти на основі штучного інтелекту здатні забезпечити цілодобову підтримку клієнтів, персоналізовані рекомендації товарів, допомогу у виборі та оформленні замовлень, що суттєво підвищує рівень обслуговування та конверсію продажів. Однак, розробка та інтеграція ефективних чат-ботів у системи електронної комерції залишається складним завданням, що потребує комплексного підходу та вдосконалення існуючих методів. Тому дослідження та розробка методів створення інтернет-магазину з інтегрованим чат-ботом на основі штучного інтелекту є надзвичайно актуальною темою як з наукової, так і з практичної точки зору.

Мета і завдання дослідження. Метою дипломної роботи є розробка методів та програмної реалізації інтернет-магазину з інтегрованим чат-ботом на основі штучного інтелекту для підвищення ефективності взаємодії з користувачами та оптимізації бізнес-процесів електронної комерції.

Для досягнення поставленої мети необхідно вирішити наступні **завдання**.

1. Проаналізувати існуючі підходи до розробки інтернет-магазинів та систем штучного інтелекту для чат-ботів, визначити їх переваги та обмеження.

2. Розробити архітектуру системи інтернет-магазину з інтегрованим чат-ботом, що забезпечує масштабованість, гнучкість та відмовостійкість.

3. Спроектувати модель даних та систему управління базами даних, оптимізовану для роботи з високими навантаженнями.

4. Обґрунтувати вибір архітектури нейронної мережі для чат-бота та розробити методи обробки запитів природною мовою.

5. Реалізувати програмну систему інтернет-магазину з інтегрованим чат-ботом та провести її експериментальну оцінку.

Об’єкт дослідження: процеси розробки та функціонування систем електронної комерції з інтегрованими чат-ботами на основі штучного інтелекту.

Предмет дослідження: методи та технології розробки інтернет-магазину з інтегрованим чат-ботом на основі штучного інтелекту.

Практичне значення роботи. Результати дипломної роботи мають безпосереднє практичне застосування у сфері електронної комерції. Розроблені методи та програмна реалізація інтернет-магазину з інтегрованим чат-ботом можуть бути використані для створення ефективних систем електронної комерції з покращеною взаємодією з користувачами. Впровадження запропонованих рішень дозволить бізнесу автоматизувати процеси обслуговування клієнтів, підвищити рівень персоналізації взаємодії, збільшити конверсію продажів та оптимізувати операційні витрати. Розроблений чат-бот забезпечує цілодобову підтримку клієнтів, надання консультацій щодо товарів та послуг, допомогу у виборі та оформленні замовлень, що підвищує загальну якість обслуговування та задоволеність клієнтів.

Теоретичне значення роботи. Теоретичне значення роботи полягає у розвитку методології інтеграції технологій штучного інтелекту в системи електронної комерції. Запропоновані у роботі підходи до проектування архітектури системи, організації взаємодії компонентів, забезпечення масштабованості та відмовостійкості, а також методи навчання та оптимізації моделей штучного інтелекту для чат-ботів збагачують теоретичну базу в галузі розробки інтелектуальних систем електронної комерції. Встановлені закономірності та принципи ефективної інтеграції чат-ботів в інтернет-магазини можуть бути використані в подальших дослідженнях та розробках у цій галузі.

Гіпотеза дослідження. Інтеграція чат-бота на основі сучасних моделей штучного інтелекту в архітектуру інтернет-магазину дозволить суттєво підвищити ефективність взаємодії з користувачами, збільшити конверсію продажів та оптимізувати бізнес-процеси електронної комерції.

Новизна роботи. Наукова новизна дипломної роботи полягає у вдосконаленні методів розробки інтернет-магазинів з інтегрованими чат-ботами на основі штучного інтелекту. Розроблено методи оптимізації моделі штучного інтелекту для роботи в умовах обмежених ресурсів, що дозволяє ефективно використовувати технології обробки природної мови в системах електронної комерції. Удосконалено алгоритми формування відповідей та рекомендацій чат-бота з урахуванням контексту взаємодії та персональних характеристик користувачів.

Методи дослідження. В процесі написання дипломної роботи була використана система загальнонаукових та спеціальних емпіричних і теоретичних методів дослідження. Також використовувалися такі емпіричні методи, як, опис, порівняння та узагальнення. Для аналізу існуючих підходів до розробки інтернет-магазинів та систем штучного інтелекту для чат-ботів застосовувалися методи аналізу та синтезу. Методи математичного моделювання та статистичного аналізу застосовувалися при розробці та оцінці моделей штучного інтелекту для чат-бота. Для експериментальної оцінки розробленої системи використовувалися методи тестування продуктивності, надійності та ефективності.

Структура роботи. Дипломна робота складається зі вступу, трьох розділів, висновків та списку використаних джерел. У першому розділі проведено аналіз існуючих підходів до розробки інтернет-магазинів та систем штучного інтелекту для чат-ботів. У другому розділі запропоновано вдосконалені методи розробки інтернет-магазину з інтегрованим чат-ботом, включаючи архітектуру системи, модель даних та архітектуру штучного інтелекту для чат-бота. У третьому розділі представлено програмну реалізацію та експериментальну оцінку розробленої системи. У висновках підсумовано основні результати дипломної роботи та окреслено перспективи подальших досліджень.

1 АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ДО РОЗРОБКИ ІНТЕРНЕТ-МАГАЗИНІВ ТА СИСТЕМ ШІ ДЛЯ ЧАТ-БОТІВ

Інтернет-магазини еволюціонували від простих веб-сайтів до складних багаторівневих систем, які забезпечують персоналізовану взаємодію, високу продуктивність та масштабованість. Паралельно, розвиток штучного інтелекту, зокрема чат-ботів, створює нові можливості для автоматизації обслуговування клієнтів, підвищення ефективності комунікації та підтримки користувачів у режимі 24/7.

1.1 Методи розробки сучасних інтернет-магазинів

Сучасні методи розробки інтернет магазинів включають архітектурні рішення та підходи які задовольняють запитам сучасних додатків для електронної комерції.

1.1.1 Архітектури клієнт-серверних додатків для електронної комерції

Архітектура клієнт-серверних додатків електронної комерції представляє модель організації програмного забезпечення, де функції розподіляються між серверами та клієнтами, забезпечуючи необхідну масштабованість, надійність та безпеку для обробки транзакцій. Традиційна трирівнева архітектура включає представницький рівень, рівень бізнес-логіки та рівень доступу до даних, при цьому існують різні підходи — від монолітної архітектури з єдиним додатком до мікросервісної, що розподіляє функціональність між незалежними сервісами.

Сучасні архітектурні підходи включають архітектуру, орієнтовану на події, що базується на асинхронній обробці через брокери повідомлень; безсерверну архітектуру, яка абстрагує управління інфраструктурою; Progressive Web Application, що поєднує переваги веб та мобільних додатків; архітектуру на основі контейнерів для ізоляції компонентів; та архітектуру, орієнтовану на API, що сприяє інтеграції з зовнішніми системами. Ці підходи забезпечують високу гнучкість та ефективне використання ресурсів.

Гібридна архітектура, що поєднує елементи різних підходів, часто є оптимальним рішенням для додатків електронної комерції, дозволяючи реалізувати критичні функції як монолітний додаток для надійності, а допоміжні — як мікросервіси для гнучкості. Вибір архітектурного підходу залежить від аналізу бізнес-вимог, очікувань користувачів та технічних обмежень, що дозволяє досягти балансу між надійністю, продуктивністю, масштабованістю та швидкістю розробки.

1.1.2 Технології створення інтерфейсів користувача для інтернет-магазинів

Технології створення інтерфейсів користувача для інтернет-магазинів базуються на HTML, CSS, JavaScript із застосуванням спеціалізованих фреймворків (React, Angular, Vue.js) та CSS-фреймворків (Bootstrap, Tailwind CSS) для адаптивного дизайну. Сучасні підходи включають Single Page Application (SPA) та Progressive Web Applications (PWA), що забезпечують швидке завантаження та можливість роботи офлайн, а також готові платформи електронної комерції, такі як Shopify, WooCommerce, Magento та OpenCart, які пропонують комплексні рішення без необхідності розробки з нуля.

Технології персоналізації та рекомендаційних систем використовують алгоритми машинного навчання для адаптації контенту відповідно до потреб користувачів, демонструючи динамічні пропозиції на основі поведінки,

демографічних даних та історії покупок. Мобільні інтерфейси набувають особливого значення через зростання мобільного трафіку та вимагають оптимізації для сенсорного введення, врахування обмеженого розміру екрану та адаптації навігаційних патернів [4].

Технології доступності та інклюзивності забезпечують можливість користування інтернет-магазином для людей з особливими потребами через дотримання стандартів WCAG, семантичну HTML-розмітку та підтримку допоміжних технологій. Безпека інтерфейсу включає HTTPS-шифрування, захист від XSS та CSRF атак, валідацію даних, а інтеграція з соціальними мережами та месенджерами розширює канали взаємодії з користувачами через авторизацію, поширення товарів та підтримку через чат-боти.

1.1.3 Системи управління контентом та бази даних в інтернет-магазинах

Системи управління контентом (CMS) для електронної комерції представляють програмні платформи, що забезпечують створення та управління цифровим контентом без поглибленого знання програмування. Вони відрізняються від загальних CMS наявністю спеціалізованих функцій для організації каталогу товарів, обробки замовлень, управління запасами та інтеграції з платіжними системами.

Фундаментальною складовою CMS є система управління базами даних (СУБД), яка забезпечує зберігання даних про товари, категорії, користувачів та замовлення. Взаємодія між CMS та СУБД реалізується через API або ORM, які абстрагують складність взаємодії з базою даних. Спеціалізовані CMS, такі як Magento, WooCommerce, PrestaShop, OpenCart та Shopify, пропонують різні можливості, орієнтовані на різні сегменти ринку. Вибір CMS залежить від масштабу бізнесу, асортименту товарів, технічних вимог та бюджету.

Реляційні бази даних (MySQL, PostgreSQL, Microsoft SQL Server, Oracle) є найпоширенішим типом СУБД для інтернет-магазинів, забезпечуючи

структуроване зберігання даних з чітко визначеними зв’язками. Структура бази даних включає таблиці для товарів, категорій, користувачів, замовлень та допоміжні таблиці для знижок, акцій, відгуків. Реляційні СУБД забезпечують ACID-властивості транзакцій, що важливо для фінансових операцій.

Оптимізація продуктивності досягається через нормалізацію таблиць, створення індексів та кешування запитів. NoSQL бази даних пропонують альтернативний підхід з гнучкістю схеми та горизонтальною масштабованістю. Документо-орієнтовані бази (MongoDB, CouchDB) ефективні для зберігання неоднорідних даних про товари; бази “ключ-значення” (Redis, Memcached) використовуються для кешування; колоночні бази (Cassandra) оптимізовані для аналітичних запитів; графові бази (Neo4j) ефективні для моделювання складних зв’язків.

Headless CMS представляє сучасний підхід, який відокремлює бекенд від фронтенду, надаючи доступ до контенту через API (RESTful або GraphQL). Це дозволяє створювати гнучкі інтерфейси з використанням будь-яких технологій фронтенду (React, Angular, Vue.js). Підхід особливо цінний для омніканальної комерції, де контент повинен бути доступний через різні канали (веб-сайт, мобільний додаток, кіоски, боти) з оптимізованим представленням. Contentful, Strapi, Sanity та Prismic є прикладами headless CMS.

Архітектура, заснована на мікросервісах та API, сприяє підвищенню продуктивності та гнучкості, дозволяючи адаптуватися до змін вимог ринку без повної переробки системи. Оптимізація для високонавантажених інтернет-магазинів включає горизонтальне масштабування через шардинг (розподіл даних між серверами) та реплікацію (створення копій даних), а також кешування часто запитуваних даних, що значно зменшує кількість звернень до бази даних.

Безпека даних є фундаментальним аспектом розробки інтернет-магазинів, оскільки системи зберігають конфіденційну інформацію. Комплексна стратегія включає захист на рівні мережевої інфраструктури (фаєрволи, VPN), додатків (валідація даних, запобігання SQL-ін’єкціям та XSS-атакам) та баз даних (шифрування, управління доступом, аудит).

Для захисту конфіденційних даних використовуються алгоритми хешування (bcrypt, Argon2) та шифрування (AES, RSA). Відповідність регуляторним вимогам (GDPR, PCI DSS) вимагає впровадження специфічних практик: мінімізація збору персональних даних, механізми згоди користувачів, забезпечення права на видалення даних, регулярні аудити безпеки. Б

езпека даних є неперервним процесом, який вимагає постійного моніторингу та адаптації до нових загроз у мінливому технологічному ландшафті.

1.2 Архітектури моделей штучного інтелекту для чат-ботів

Всі побудовані системи мають архітектуру, моделі ШІ не виключення. Архітектури моделей ШІ поділяються на категорії, які визначаються типом обробки даних, розміщення слоїв та типом входу та виходу даних.

1.2.1 Рекурентні нейронні мережі для обробки природної мови

Рекурентні нейронні мережі (RNN) представляють собою клас архітектур штучних нейронних мереж, спеціально розроблених для обробки послідовних даних, що мають внутрішню пам’ять, яка дозволяє зберігати інформацію про попередні елементи послідовності. На кожному кроці t RNN оновлює свій внутрішній стан на основі поточного входу та попереднього стану, що робить їх ефективними для обробки природної мови, де контекст та порядок слів мають суттєвий вплив на значення тексту.

Для подолання проблеми зникаючого або вибухаючого градієнта були розроблені спеціалізовані архітектури (табл. 1.1), зокрема Long Short-Term Memory (LSTM) з трьома типами вентилів (вхідний, забуття, вихідний) та Gated Recurrent Units (GRU) з двома вентилями (скидання, оновлення), які ефективно зберігають інформацію протягом тривалих послідовностей.

Таблиця 1.1 — Порівняння архітектур рекурентних нейронних мереж для обробки природної мови

Архітектура	Механізми пам’яті	Складність	Ефективність для довгих послідовностей	Основні застосування в NLP
Проста RNN	Внутрішній стан	Низька	Низька (проблема зникаючого градієнта)	Моделювання мови для коротких послідовностей
LSTM	Стан комірки, вхідний вентиль, вентиль забуття, вихідний вентиль	Висока	Висока	Машинний переклад, генерація тексту, аналіз настроїв
GRU	Вентиль скидання, вентиль оновлення	Середня	Висока	Класифікація тексту, розпізнавання сутностей, рекомендаційні системи
Двонаправлена RNN	Пряма та зворотна послідовність	Середня	Середня	Розпізнавання іменованих сутностей, відповіді на питання
Стекова RNN	Ієрархічні рівні абстракції	Висока	Висока	Складні завдання NLP, синтаксичний аналіз

Інтеграція механізму уваги в архітектуру RNN дозволяє мережі зосередитися на різних частинах вхідної послідовності при формуванні кожного елемента вихідної послідовності, що особливо корисно для довгих послідовностей. Незважаючи на успіхи трансформерів, RNN залишаються релевантними для багатьох завдань обробки природної мови, особливо в сценаріях з обмеженими обчислювальними ресурсами. Поєднання RNN з механізмами уваги та техніками трансформерів призвело до розвитку гібридних архітектур, таких як Transformer-XL та Universal Transformer, а дослідження в області нейромережевих диференційних рівнянь відкриває нові перспективи для розвитку рекурентних архітектур з ефективнішим використанням пам’яті.

1.2.2 Трансформери та їх застосування в діалогових системах

Трансформери — це революційна архітектура нейронних мереж, запропонована в 2017 році в роботі “Attention is All You Need” дослідниками з Google, що повністю відмовляється від рекурентності та згорток, використовуючи натомість механізм самоуваги (self-attention), який дозволяє моделі одночасно враховувати взаємозв’язки між усіма елементами послідовності. Самоувага обчислюється як зважена сума значень, де ваги визначаються сумісністю запиту з відповідними ключами, з використанням формули Attention(Q,K,V) = softmax(QK^T / sqrt(d_k))V. Архітектура трансформера складається з кодувальника (encoder) та декодувальника (decoder), кожен з яких містить кілька шарів, що включають механізм самоуваги з багатьма головками та повнозв’язну нейронну мережу з залишковими з’єднаннями та нормалізацією шару [8, 6].

У контексті діалогових систем трансформери застосовуються для розуміння природної мови (NLU), генерації природної мови (NLG), розпізнавання намірів користувача та управління діалогом. Моделі можна класифікувати на однонаправлені (GPT), двонаправлені (BERT), кодувальник-декодувальник (T5, BART) та спеціалізовані діалогові (DialoGPT, Meena). Багатомовні трансформери

(mBERT, XLM-R) розширюють можливості на десятки мов, а мультимодальні (CLIP, DALL-E) дозволяють працювати з текстом, зображеннями та аудіо. Механізми покращеної уваги (Performer, Reformer) знижують обчислювальну складність, а моделі з розширеною пам’яттю (Transformer-XL) ефективно зберігають інформацію з попередніх сегментів діалогу.

Оптимізація трансформерів включає дистиляцію знань (DistilBERT), квантизацію, прунінг та розділення параметрів між шарами (ALBERT), що дозволяє впроваджувати моделі на пристроях з обмеженими ресурсами. Системи безперервного навчання долають обмеження статичних моделей, використовуючи активне навчання та навчання з підкріпленням на основі людського зворотного зв’язку (RLHF). Етичні аспекти розробки діалогових систем включають виявлення та мітигацію упереджень, забезпечення приватності користувачів через диференційовану приватність та федеративне навчання, прозорість відповідей через аналіз уваги та стійкість до зловмисних маніпуляцій через противадверсаріальне навчання.

1.2.3 Гібридні архітектури для інтелектуальних чат-ботів

Гібридні архітектури для інтелектуальних чат-ботів представляють собою комплексні підходи, що поєднують різноманітні методи, алгоритми та моделі машинного навчання для досягнення оптимальної продуктивності (табл. 1.2). Основна мета створення гібридних архітектур полягає в подоланні обмежень окремих підходів: системи на основі правил забезпечують високу точність у межах заздалегідь визначених сценаріїв, але не адаптуються до непередбачених запитів, тоді як системи на основі нейронних мереж більш гнучкі, але часто страждають від непередбачуваності або генерації некоректних відповідей.

Одним з фундаментальних підходів є комбінація моделей на основі правил та нейронних мереж, організованих у каскадну систему або паралельно працюючих з подальшим об’єднанням результатів. Інший підхід — інтеграція ансамблів різних

типів нейронних мереж (RNN, CNN, трансформери), де кожна модель має свої переваги: RNN ефективно моделюють послідовну природу тексту, CNN виявляють локальні паттерни, а трансформери моделюють довгострокові залежності завдяки механізму самоуваги [9].

Таблиця 1.2 — Порівняння підходів до гібридизації архітектур чат-ботів

Компоненти	Переваги	Обмеження	Сценарії застосування
Система правил, нейронні мережі (LSTM, GRU, Трансформери)	Висока точність для відомих сценаріїв, гнучкість для нестандартних запитів	Потребує ручного створення правил, складність підтримки великих баз правил	Обслуговування клієнтів, FAQ-боти, системи бронювання
Різні архітектури нейронних мереж (RNN, CNN, Трансформери)	Підвищена точність, стійкість до різних типів запитів	Обчислювальна складність, ресурсоємність	Складні діалогові системи, віртуальні асистенти
Нейронні мережі, графи знань, онтології	Фактична коректність, здатність надавати специфічну інформацію	Обмеженість доменом знань, складність оновлення баз знань	Експертні системи, освітні боти, медичні консультанти
Ієрархічні моделі	Краще розуміння контексту, когерентність діалогу	Складність координації між рівнями	Тривалі розмови, чат-боти з персоналізацією

Кінець таблиці 1.2

Модульні архітектури	Висока гнучкість, можливість заміни окремих модулів	Складність інтеграції, потенційна неузгодженість між модулями	Багатофункціональні асистенти, омніканальні боти

Інтеграція зовнішніх баз знань та семантичних моделей дозволяє подолати обмеження нейронних мереж щодо фактичної коректності через поєднання нейро-символічних підходів.

Мультимодальні гібридні архітектури розширюють можливості за межі текстової взаємодії, інтегруючи обробку та генерацію контенту в різних модальностях (зображення, аудіо, відео), а механізми пояснюваності та прозорості забезпечують баланс між продуктивністю та інтерпретовністю.

Розподілені та федеративні архітектури адресують проблеми масштабованості та приватності даних, поєднуючи глобальні моделі з локально адаптованими [2].

1.3 Сучасні підходи до інтеграції чат-ботів з інтернет-магазинами

Основні гравці на ринку, такі як Shopify, Magento, WooCommerce та Salesforce Commerce Cloud, пропонують широкі можливості для інтеграції чат-ботів та інших інтелектуальних сервісів. Ці платформи активно впроваджують API-рішення, плагіни та вбудовані інструменти для забезпечення персоналізованої взаємодії з користувачами та автоматизації обслуговування.

1.3.1 Аналіз існуючих рішень на ринку електронної комерції

Ринок електронної комерції представляє собою динамічну екосистему програмних платформ, сервісів та інструментів для здійснення комерційних операцій через Інтернет.

За даними Statista, глобальний ринок електронної комерції досягнув обсягу понад 5,7 трильйонів доларів США у 2022 році, з прогнозованим зростанням до 8,1 трильйонів доларів до 2026 року.

Основні категорії технологічних рішень включають платформи електронної комерції (Shopify, WooCommerce, Magento), маркетплейси (Amazon, eBay, Etsy), системи управління контентом з функціоналом електронної комерції, хмарні рішення (BigCommerce, Salesforce Commerce Cloud) та програмне забезпечення з відкритим кодом (OpenCart, PrestaShop) [11] (табл. 1.3).

Таблиця 1.3 — Порівняльна характеристика рішень на ринку електронної комерції

Платформа	Тип рішення	Цільова аудиторія	Технічні вимоги	Вартість	Підтримка ШІ та чат-ботів	Найкраще підходить для
Shopify	SaaS	Малий та середній бізнес	Низькі (хостинг включено)	$29-299/міс + транзакційні збори	Нативна підтримка через Shopify Flow та API	Швидкий старт, мінімальні технічні ресурси

Продовження таблиці 1.3

WooCommerce	Плагін для WordPress	Малий та середній бізнес з досвідом WordPress	Середні (потрібен власний хостинг)	Безкоштовно + витрати на хостинг та плагіни	Через сторонні плагіни та інтеграції	Існуючі сайти на WordPress, гнучкість
Magento	On-premise/ Cloud	Середній та великий бізнес	Високі (потрібні розробники)	Безкоштовно (Open Source) або від $22,000/рік (Commerce)	Розширена підтримка через модулі та Adobe Sensei	Складні B2B/B2C сценарії, висока кастомізація
Salesforce Commerce Cloud	SaaS/Cloud	Великі підприємства	Високі (потрібна інтеграція)	Від $250,000/рік	Нативна інтеграція з Einstein AI та Einstein Bots	Омніканальний ритейл, глобальні операції
Amazon Marketplace	Маркетплейс	Будь-який розмір бізнесу	Низькі (готова інфраструктура)	Комісія 8-15% + місячна плата	Amazon Alexa, рекомендаційні системи	Доступ великої аудиторії, FBA логістика

Кінець таблиці 1.3

BigCommerce	SaaS/Cloud	Малий та середній бізнес	Середні	$29.95-299.95/міс	Через партнерів та API	Масштабування без транзакційних зборів
OpenCart	Open Source	Малий бізнес, розробники	Високі (потрібен власний хостинг та налаштування)	Безкоштовно + витрати на хостинг та розширення	Через модулі та розширення	Низькобюджетні проекти з технічними ресурсами
Shopware	Open Source/ Cloud	Середній та великий бізнес	Високі	Безкоштовно (Community) або від €199/міс (Cloud)	Нативна підтримка ШІ-функцій	Інноваційний досвід покупок, європейський ринок

Інтеграція чат-ботів та штучного інтелекту в рішення електронної комерції трансформує досвід онлайн-покупок та операційну ефективність бізнесів. За даними Juniper Research, глобальна економія роздрібних витрат завдяки чат-ботам досягне 439 мільйонів доларів до 2023 року. Більшість платформ пропонують різні рівні підтримки ШІ-інтеграцій, від додатків у Shopify App Store до власних систем Amazon та Alibaba для покращення пошуку товарів та рекомендацій. Персоналізація та омніканальність стали визначальними характеристиками

сучасної електронної комерції, а аналітика, великі дані та машинне навчання дозволяють бізнесам приймати обґрунтовані рішення та оптимізувати операції.

Мобільна комерція (m-commerce) та прогресивні веб-додатки (PWA) формують швидкозростаючий сегмент ринку, з глобальними продажами через m-commerce, що досягли 3,56 трильйонів доларів у 2021 році (72,9% від загальних продажів електронної комерції). Сучасні платформи приділяють велику увагу безпеці, конфіденційності та відповідності нормативним вимогам, а майбутні тенденції включають інтеграцію доповненої та віртуальної реальності, Інтернету речей та блокчейну для розширення можливостей електронної комерції [12].

1.3.2 Методи персоналізації взаємодії з користувачами

Персоналізація взаємодії з користувачами в контексті інтернет-магазинів та чат-ботів представляє собою комплекс технологій та підходів для адаптації контенту, функціональності та комунікацій до індивідуальних потреб користувачів. Дослідження McKinsey показують, що ефективна персоналізація може збільшити дохід бізнесу на 5-15% та підвищити ефективність маркетингових витрат на 10-30%. В основі персоналізації лежить збір та аналіз різноманітних типів даних (демографічних, поведінкових, контекстуальних, психографічних), а сучасні методи використовують алгоритми машинного навчання для обробки даних та виявлення патернів для прогнозування поведінки користувачів та надання релевантних рекомендацій.

Рекомендаційні системи є одним з найпоширеніших методів персоналізації, що поділяються на кілька типів: колаборативна фільтрація, контентна фільтрація та гібридні підходи. Персоналізація на основі контексту та поведінкового таргетингу враховує динамічні фактори, такі як поточна ситуація, настрій, наміри та етап покупки. У чат-ботах такий підхід дозволяє ініціювати релевантні діалоги в оптимальний момент, а аналіз природної мови дозволяє адаптувати не лише зміст,

але й спосіб комунікації, виявляючи стиль спілкування та емоційне забарвлення для створення ефекту “мовного віддзеркалення”.

Омніканальна персоналізація забезпечує послідовний досвід взаємодії через різні канали та пристрої, об’єднуючи дані з усіх точок контакту для створення єдиного профілю. Для реалізації такого підходу використовуються централізовані системи управління даними про клієнтів (CDP), які інтегрують дані з різних джерел. Адаптивне навчання відноситься до здатності системи персоналізації постійно вдосконалюватися на основі нових даних через використання посиленого навчання, де система отримує зворотний зв’язок від користувачів щодо релевантності рекомендацій [13].

1.3.3 Оптимізація бізнес-процесів за допомогою ШІ-асистентів

Оптимізація бізнес-процесів за допомогою ШІ-асистентів представляє собою інноваційний підхід до підвищення ефективності комерційних операцій через інтеграцію інтелектуальних систем у робочі процеси. ШІ-асистенти використовують технології штучного інтелекту для виконання різноманітних завдань, аналізу даних, розпізнавання патернів та прийняття рішень. За дослідженням McKinsey, впровадження таких технологій може збільшити прибуток бізнесу до 20% та зменшити операційні витрати на 10-15%, що надає суттєву конкурентну перевагу. Автоматизація обслуговування клієнтів через інтелектуальні чат-боти, здатні розуміти та відповідати на запити природною мовою, є одним з найпоширеніших застосувань. За даними Juniper Research, чат-боти в роздрібній торгівлі зможуть забезпечити економію витрат до 439 мільйонів доларів до 2023 року, порівняно з 7 мільйонами у 2019 році [14].

Оптимізація управління запасами та ланцюгами поставок за допомогою ШІ дозволяє підвищити ефективність логістичних операцій через прогностичну аналітику для передбачення попиту на товари з урахуванням сезонності, тенденцій ринку та інших факторів. Персоналізація та оптимізація маркетингу трансформує

підходи до залучення клієнтів, дозволяючи створювати індивідуалізовані стратегії в масштабі через аналіз поведінки користувачів, історії покупок та демографічних характеристик. За даними Salesforce, маркетологи, які використовують ШІ, відзначають збільшення конверсії на 51%, покращення залучення клієнтів на 50% та зростання NPS на 49% [11].

Оптимізація внутрішніх операцій та управління знаннями дозволяє підвищити продуктивність працівників через автоматизацію рутинних завдань, а системи управління знаннями аналізують та індексують документи, створюючи семантичну мережу корпоративних знань. За даними Accenture, це може підвищити продуктивність праці до 40% та знизити операційні витрати на 15-25%. Інтеграція ШІ у бізнес-аналітику та прийняття рішень забезпечує глибше розуміння ринку та бізнес-процесів через автоматичний аналіз даних з різноманітних джерел, виявлення прихованих закономірностей та моделювання різних сценаріїв. За даними PwC, компанії, які використовують ШІ для аналітики, демонструють на 26% вищу прибутковість порівняно з конкурентами [14].

Висновки до розділу

У першому розділі проведено дослідження існуючих підходів до розробки інтернет-магазинів та систем штучного інтелекту для чат-ботів. Аналіз архітектур клієнт-серверних додатків показав еволюцію від монолітних систем до мікросервісних, контейнерних та безсерверних, які забезпечують кращу масштабованість та відмовостійкість. Розглянуті технології створення інтерфейсів користувача, включаючи JavaScript-фреймворки, адаптивний дизайн та прогресивні веб-додатки, формують основу для розробки зручних інтерфейсів. Дослідження систем управління контентом та баз даних виявило переваги різних типів СУБД для задач електронної комерції та тенденцію до використання headless CMS.

Аналіз архітектур моделей штучного інтелекту для чат-ботів виявив їх принципи роботи, переваги та обмеження. Рекурентні нейронні мережі демонструють здатність моделювати контекст, але страждають від проблеми зникаючого градієнта при роботі з довгими послідовностями. Трансформери завдяки механізму самоуваги ефективно долають це обмеження та забезпечують паралельну обробку послідовностей. Гібридні архітектури, які поєднують різні підходи, дозволяють створювати більш робастні діалогові системи, здатні адаптуватися до різних сценаріїв взаємодії з користувачами.

Аналіз існуючих рішень на ринку електронної комерції виявив різноманітність доступних платформ, від SaaS-рішень (Shopify, BigCommerce) до відкритих систем (WooCommerce, Magento). Дослідження методів персоналізації взаємодії з користувачами показало еволюцію від простих рекомендаційних систем до складних підходів.

Розробка інтернет-магазину з інтегрованим чат-ботом є перспективним напрямком, який поєднує переваги сучасних архітектур електронної комерції з можливостями інтелектуальних діалогових систем, дозволяючи створити систему, яка відповідає потребам ринку та здатна еволюціонувати з розвитком технологій.

2 ВДОСКОНАЛЕННЯ МЕТОДІВ РОЗРОБКИ ІНТЕРНЕТ-МАГАЗИНУ З ІНТЕГРОВАНИМ ЧАТ-БОТОМ

Зі зростанням вимог до якості онлайн-сервісів та очікувань користувачів щодо швидкості, персоналізації та доступності, інтернет-магазини мають впроваджувати інноваційні рішення для покращення взаємодії з клієнтами. Одним із таких рішень є інтеграція чат-ботів, що базуються на технологіях штучного інтелекту, здатних автоматизувати обробку запитів, надати рекомендації та супроводжувати користувача під час здійснення покупок.

2.1 Архітектура системи інтернет-магазину

Архітектура системи відіграє важливу роль як у процесі проектування для визначення допустимих технологій при розробці, так і в процесі розробки під час масштабування, перевикористання та підтримки.

2.1.1 Проектування мікросервісної архітектури

Мікросервісна архітектура являє собою сучасний підхід до розробки програмного забезпечення, що характеризується декомпозицією монолітної системи на сукупність невеликих, функціонально самодостатніх сервісів, кожен з яких відповідає за виконання конкретного бізнес-завдання та може розгортатися незалежно від інших компонентів. У контексті розробки інтернет-магазину з інтегрованим чат-ботом на базі штучного інтелекту, мікросервісний підхід надає суттєві переваги, включаючи технологічну гетерогенність, стійкість до відмов, спрощене масштабування та можливість паралельної розробки різними командами. Як свідчить аналіз структури створеного Angular-додатку, представленого у файлах app.component.ts, app.routes.ts та app.config.ts, архітектура системи вже

містить елементи модульності, де окремі функціональні блоки, такі як управління товарами (ProductService), категоріями (CategoryService), аутентифікацією (AuthService) та замовленнями (OrderService), інкапсульовані у відповідні сервіси. Наприклад, в app.config.ts чітко видно розмежування відповідальності через впровадження залежностей: provideStore(appStore), provideEffects(appEffects), ProductService, CategoryService, AuthService, OrderService, PromoCodeService. Такий підхід створює фундамент для подальшої мікросервісної декомпозиції системи [15].

Проектування мікросервісної архітектури для інтернет-магазину з інтегрованим чат-ботом починається з визначення доменних меж для кожного сервісу згідно з принципами предметно-орієнтованого проектування (Domain-Driven Design). Аналізуючи маршрутизацію додатку з файлу app.routes.ts, можна виокремити наступні домени: управління товарами (ProductListComponent, ProductDetailsComponent), обробка замовлень (ShoppingCartComponent, MyOrdersComponent), аутентифікація та авторизація користувачів (RegisterComponent, LoginComponent, LogoutComponent), адміністрування (AdminMainComponent, AddEditProductComponent, AddEditCategoryComponent). У мікросервісній реалізації кожен з цих доменів може бути виділений в окремий сервіс з власною базою даних, що відповідає патерну Database-per-Service. Особливу увагу слід приділити сервісу чат-бота, який вже інтегрований через зовнішній скрипт, як можна побачити у файлі index.html: script.src="https://www.chatbase.co/embed.min.js";script.id="BAjdt4OAFwo89pAqWzfJ5";. У мікросервісній архітектурі цей компонент доцільно реалізувати як окремий сервіс, що взаємодіє з іншими через API Gateway, забезпечуючи обробку природної мови та інтелектуальний аналіз запитів користувачів.

Технічна реалізація мікросервісної архітектури передбачає вибір відповідних технологій та патернів взаємодії між сервісами. Для забезпечення комунікації між фронтендом та бекендом у нашому рішенні використовується HttpClient з перехоплювачами запитів, як видно з коду app.config.ts: provideHttpClient(withInterceptors([httpInterceptor])). У мікросервісній архітектурі

ця взаємодія може бути розширена впровадженням API Gateway, який централізує маршрутизацію запитів, аутентифікацію та авторизацію. Для комунікації між сервісами доцільно використовувати як синхронні (REST API, gRPC), так і асинхронні (повідомлення через RabbitMQ, Kafka) механізми, залежно від характеру взаємодії. Наприклад, для оновлення статусу замовлення асинхронний підхід забезпечує більшу надійність та масштабованість. Контейнеризація за допомогою Docker та оркестрація з використанням Kubernetes дозволяють уніфікувати розгортання мікросервісів та керувати їх життєвим циклом. Особливості стилізації та інтерфейсу користувача, видимі у файлі styles.css, повинні бути адаптовані для коректного відображення у різних клієнтських додатках, які можуть взаємодіяти з мікросервісами [16].

На рисунку 2.1 представлено схему мікросервісної архітектури розроблюваного інтернет-магазину з інтегрованим чат-ботом. Діаграма ілюструє взаємозв'язки між компонентами системи та відображає структуру клієнт-серверної взаємодії

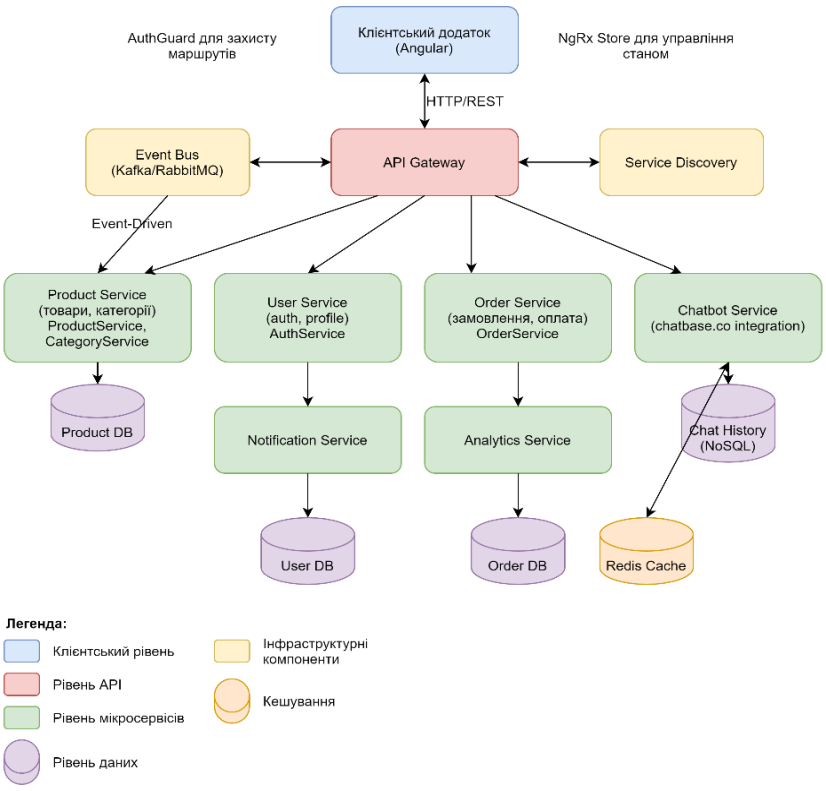

Рисунок 2.1 — Діаграма мікросервісної архітектури

Управління станом та даними є суттєвим викликом у мікросервісній архітектурі інтернет-магазину. Аналізуючи код app.component.ts, можна помітити використання NgRx Store для управління станом додатку: private store: Store<AppState> та диспетчеризацію дій для аутентифікації: this.store.dispatch(AuthActions.setAuthenticated()). У мікросервісному контексті управління станом розподіляється між сервісами, де кожен відповідає за свій домен даних. Для забезпечення узгодженості даних між сервісами застосовуються патерни Saga або Event Sourcing. Перший координує розподілені транзакції через послідовність локальних транзакцій у різних сервісах, компенсуючи невдалі операції. Другий зберігає всі зміни стану як послідовність подій, що дозволяє відтворити стан системи на будь-який момент часу. Для реалізації чат-бота на основі ШІ особливо корисним є Event Sourcing, оскільки історія взаємодій користувача з системою надає цінний контекст для персоналізації відповідей. Кожен мікросервіс повинен мати власне сховище даних, оптимізоване під специфічні потреби: SQL бази даних для структурованої інформації про товари та замовлення, NoSQL рішення для гнучких структур даних чат-бота, Redis для кешування та швидкого доступу до сесійних даних.

Забезпечення надійності, масштабованості та безпеки є фундаментальними аспектами проектування мікросервісної архітектури. З аналізу файлу app.routes.ts видно використання AuthGuard для захисту маршрутів: canActivate: [AuthGuard], data: { roles: ['Admin'] }. У мікросервісній архітектурі безпека повинна бути реалізована на різних рівнях: від захисту API Gateway через JWT токени, OAuth 2.0 або OpenID Connect до шифрування даних у спокої та під час передачі. Масштабованість досягається через горизонтальне масштабування окремих сервісів відповідно до навантаження, що особливо актуально для сервісів товарів та чат-бота, які зазнають пікових навантажень. Для забезпечення надійності впроваджуються патерни Circuit Breaker, Bulkhead, Retry, які запобігають каскадним відмовам при недоступності окремих сервісів. Моніторинг та трасування розподілених транзакцій реалізуються через такі інструменти як Prometheus, Grafana, Jaeger або Zipkin, що дозволяють виявляти проблемні місця та

оптимізувати продуктивність системи. Особливу увагу слід приділити продуктивності чат-бота, який повинен забезпечувати швидкі та релевантні відповіді навіть при високому навантаженні, що досягається через асинхронну обробку запитів та оптимізацію алгоритмів машинного навчання [17].

2.1.2 Організація взаємодії компонентів системи

Організація взаємодії компонентів системи інтернет-магазину з інтегрованим чат-ботом передбачає структурування комунікаційних каналів між різними мікросервісами, клієнтською частиною та зовнішніми сервісами з метою забезпечення надійної, ефективної та безпечної обробки даних.

У розробленій системі взаємодія компонентів реалізована через комбінацію синхронних (REST API) та асинхронних (Event-Driven) механізмів комунікації, що дозволяє гнучко адаптуватися до різних типів операцій. Аналіз клієнтської частини додатку, представленої у файлі app.component.ts, демонструє інтеграцію з глобальним станом через сховище NgRx, де Store виступає як централізований механізм управління станом додатку, а Service є посередником між клієнтською частиною та серверними API (табл. 2.1).

Особливу увагу приділено інтеграції зовнішнього чат-бота, що реалізується через впровадження JavaScript-коду з Chatbase у файлі index.html: script.src="https://www.chatbase.co/embed.min.js";script.id="BAjdt4OAFwo89pAqWzfJ5";.

Така реалізація дозволяє зберегти когерентність архітектури, відокремлюючи функціональність чат-бота в окремий сервіс, але забезпечуючи безшовну інтеграцію з клієнтським інтерфейсом через стандартизовані API та механізми обміну подіями [18].

У центрі взаємодії компонентів системи знаходиться конфігурація API та сервісів, реалізована у файлі app.config.ts, де здійснюється реєстрація провайдерів для залежностей: provideRouter(routes), provideStore(appStore),

provideEffects(appEffects), provideHttpClient(withInterceptors([httpInterceptor])), ProductService, CategoryService, AuthService, OrderService, PromoCodeService. Цей файл визначає абстракції для взаємодії з різними доменними областями через відповідні сервіси, а також встановлює механізми для авторизації та аутентифікації через HTTP-перехоплювачі (Interceptors).

Перехоплювачі є фундаментальним елементом взаємодії, оскільки вони дозволяють централізовано керувати заголовками запитів, токенами автентифікації та обробкою помилок.

Маршрутизація запитів між клієнтською частиною та різними компонентами системи координується через визначені у файлі app.routes.ts шляхи, де особлива увага приділяється безпеці доступу через AuthGuard: canActivate: [AuthGuard], data: { roles: ['Admin'] }. Ця система ролей та захисту маршрутів є транзитивною моделлю для реалізації аналогічних механізмів на рівні мікросервісної архітектури, де API Gateway виконує роль централізованого контролю доступу та маршрутизації запитів до відповідних сервісів з урахуванням авторизаційних обмежень [19].

Таблиця 2.1 — Порівняльна характеристика механізмів взаємодії компонентів системи

Механізм взаємодії	Технологічна реалізація	Переваги	Обмеження	Застосування в системі
Синхронна комунікація через REST API	Angular HttpClient з перехоплювачами	Простота реалізації, негайний зворотний зв’язок, широка підтримка	Підвищений ризик каскадних відмов, складність забезпечення узгодженості даних при відмовах	Операції з товарами, аутентифікація, оформлення замовлень

Продовження таблиці 2.1

Асинхронна комунікація через події	Реалізація через NgRx Effects та Store	Слабке зв’язування компонентів, підвищена стійкість до відмов, відкладена обробка	Складність відстеження потоку виконання, потенційна неузгодженість даних	Оновлення кошика, нотифікації, оновлення статусу замовлень
Інтеграція зовнішніх сервісів	JavaScript SDK та iframe embedding	Швидкість впровадження, скорочення витрат на розробку	Залежність від зовнішніх провайдерів, обмежені можливості кастомізації	Інтеграція чат-бота через Chatbase
Локальне зберігання стану	LocalStorage API через специфічні менеджери	Збереження стану між сесіями, зменшення залежності від мережі	Обмежений обсяг даних, потенційні проблеми безпеки	Збереження кошика та автентифікаційних даних: ShoppingCartLocalStorageManager.getShoppingCartItemsAndFillStore(), AuthLocalStorageManager.getUserData()

Кінець таблиці 2.1

Управління внутрішнім станом додатку	NgRx Store та централізоване сховище	Передбачуваність змін стану, відстежуваність модифікацій, централізований контроль	Збільшення складності для простих операцій, додаткові абстракції	Управління автентифікацією: this.store.dispatch(AuthActions.setAuthenticated())

Передача даних між сервісами та управління станом додатку реалізуються через потужні механізми реактивного програмування, що є основою масштабованої та надійної взаємодії компонентів.

Аналіз файлу app.component.ts показує, як відбувається ініціалізація початкового стану додатку через взаємодію з локальним сховищем та серверними API: ShoppingCartLocalStorageManager.getShoppingCartItemsAndFillStore(this.store, this.productService) та let userData = AuthLocalStorageManager.getUserData();.

Ці рядки коду демонструють гібридний підхід до управління станом, де критичні дані зберігаються локально для забезпечення швидкого доступу та функціонування додатку в умовах нестабільного з’єднання, але синхронізуються з серверними компонентами для забезпечення узгодженості в масштабах всієї системи.

Реалізація токенної аутентифікації з перевіркою терміну дії токена є прикладом забезпечення безпеки взаємодії між компонентами, де кожен запит супроводжується валідацією автентифікаційних даних.

У контексті мікросервісної архітектури ця модель розширюється до рівня міжсервісної комунікації, де кожна взаємодія між сервісами супроводжується перевіркою автентифікаційних та авторизаційних атрибутів, а стан розподіляється між сервісами через події та спеціалізовані сховища [20].

2.1.3 Забезпечення масштабованості та відмовостійкості

Забезпечення масштабованості та відмовостійкості в архітектурі інтернет-магазину з інтегрованим чат-ботом є фундаментальним аспектом, що визначає здатність системи адаптуватися до зростання навантаження та зберігати функціональність при виникненні несправностей. Масштабованість, як властивість системи збільшувати свою продуктивність пропорційно до доданих обчислювальних ресурсів, реалізується через горизонтальне та вертикальне масштабування. Горизонтальне масштабування передбачає додавання нових екземплярів сервісів та розподіл навантаження між ними, тоді як вертикальне фокусується на збільшенні потужності існуючих серверів. Аналіз розробленого Angular-додатку, зокрема конфігурації у файлі app.config.ts, демонструє впровадження принципів, що сприяють масштабованості: provideZoneChangeDetection({ eventCoalescing: true }) оптимізує виявлення змін через об'єднання подій, а використання provideStore(appStore) та provideEffects(appEffects) забезпечує централізоване управління станом через NgRx, що спрощує синхронізацію даних між екземплярами при горизонтальному масштабуванні. Відокремлення функціональності в спеціалізовані сервіси (ProductService, CategoryService, AuthService, OrderService, PromoCodeService) створює передумови для їх незалежного масштабування відповідно до потреб — наприклад, сервіс товарів може потребувати більше ресурсів під час сезонних розпродажів, тоді як сервіс замовлень — в періоди пікових продажів.

Відмовостійкість системи, що визначається як здатність продовжувати функціонування, незважаючи на відмови окремих компонентів, реалізується через комплекс стратегій та технологій. Аналіз розробленого коду показує впровадження елементів, що підвищують відмовостійкість, зокрема механізмів локального збереження даних, який забезпечує функціонування в умовах нестабільного з'єднання. Наприклад, у файлі app.component.ts представлена логіка для відновлення даних кошика та автентифікації з локального сховища: ShoppingCartLocalStorageManager.getShoppingCartItemsAndFillStore(this.store,

this.productService); та let userData = AuthLocalStorageManager.getUserData();. Ця стратегія дозволяє користувачам продовжувати взаємодію з додатком навіть при тимчасовій недоступності серверної частини. Для повноцінної відмовостійкості в мікросервісній архітектурі реалізуються додаткові патерни: Circuit Breaker — для запобігання каскадним відмовам через швидке переривання виклику недоступного сервісу; Bulkhead — для ізоляції відмов через розділення ресурсів; Retry — для автоматичних повторних спроб виконання операції при тимчасових збоях. Використання HTTP-перехоплювачів, як видно з коду provideHttpClient(withInterceptors([httpInterceptor])), надає централізований механізм для обробки мережевих помилок та реалізації стратегій повторних спроб, що є компонентом відмовостійкості на рівні взаємодії з API [21].

Для комплексного забезпечення масштабованості та відмовостійкості в мікросервісній архітектурі інтернет-магазину впроваджуються спеціалізовані інфраструктурні рішення та практики. Service Discovery механізми дозволяють динамічно реєструвати та виявляти екземпляри сервісів, що спрощує масштабування та автоматичну заміну несправних компонентів. Балансування навантаження розподіляє запити між екземплярами сервісів, запобігаючи перевантаженню окремих вузлів. Аналіз коду маршрутизації у файлі app.routes.ts демонструє застосування ледачого завантаження модулів: loadComponent: () => import('./components/admin/admin-main/admin-main.component').then(m => m.AdminMainComponent), що є прикладом оптимізації, яка покращує масштабованість через розділення коду на менші фрагменти, що завантажуються за потреби. Ця техніка транслюється на рівень мікросервісів через принцип композиції функціональності з менших, атомарних сервісів. Резервування та реплікація даних, реалізована через розподілені бази даних, забезпечує збереження інформації при відмові окремих вузлів. Інтеграція чат-бота через зовнішній сервіс Chatbase, як видно з коду у файлі index.html, є прикладом делегування функціональності спеціалізованим провайдерам, що потенційно підвищує відмовостійкість через використання вже масштабованої та відмовостійкої інфраструктури. Моніторинг та оповіщення, хоч і не представлені явно в

клієнтському коді, є неодмінним компонентом архітектури, що забезпечує раннє виявлення проблем та автоматичне реагування на них, включаючи автоматичне масштабування та відновлення після збоїв [18].

2.2 Модель даних та система управління базами даних

Модель даних визначається сутностями та зв’язком між ними. У цій роботі представлене реляційне рішення на базі SQL. Сутності та зв’язки між ними побудовані по принципу Code first.

2.2.1 Проектування схеми бази даних для інтернет-магазину

Проектування схеми бази даних для інтернет-магазину є основоположним етапом розробки, що визначає ефективність зберігання, доступу та маніпуляції даними в системі електронної комерції. База даних інтернет-магазину представляє собою структуровану сукупність взаємопов’язаних таблиць, де кожна таблиця відповідає за зберігання специфічного типу інформації, необхідної для функціонування всіх бізнес-процесів. У контексті розробленого Angular-додатку, аналіз структури сервісів з файлу app.config.ts: ProductService, CategoryService, AuthService, OrderService, PromoCodeService дозволяє виявити основні доменні області, для яких необхідно спроектувати відповідні таблиці в базі даних. Концептуальне проектування бази даних розпочинається з ідентифікації ключових сутностей (товари, категорії, користувачі, замовлення, платежі), їх атрибутів та взаємозв’язків між ними. Логічне проектування включає визначення таблиць, полів, первинних та зовнішніх ключів, обмежень цілісності, а фізичне – вибір конкретної СУБД та оптимізацію схеми відповідно до її особливостей. У мікросервісній архітектурі кожен сервіс може мати власну базу даних, оптимізовану під конкретні завдання, що забезпечує ізоляцію даних, незалежне

масштабування та вибір оптимальної технології зберігання для кожного типу даних [22].

Аналіз маршрутизації додатку у файлі app.routes.ts дозволяє виявити основні функціональні модулі та відповідні їм дані, що потребують зберігання в базі даних. Наприклад, наявність компонентів для роботи з товарами (ProductListComponent, ProductDetailsComponent) передбачає розробку таблиць для зберігання інформації про товари, їх атрибути, категорії, зображення. Компоненти для роботи з кошиком та замовленнями (ShoppingCartComponent, MyOrdersComponent) вимагають створення таблиць для зберігання даних про замовлення, їх статуси, деталі, платежі. Компоненти аутентифікації (RegisterComponent, LoginComponent) потребують таблиць для зберігання даних користувачів, їх ролей, паролів, токенів. Адміністративні компоненти (AdminMainComponent, AddEditProductComponent, AddEditCategoryComponent) використовують всі попередні таблиці, але з розширеними правами доступу. Особливу увагу слід приділити зберіганню даних для інтегрованого чат-бота, що може включати історію повідомлень, контекст бесіди, параметри персоналізації. Міграція від монолітної до мікросервісної архітектури відображається на рівні бази даних через розділення єдиної схеми на окремі бази для кожного сервісу з власними таблицями, що забезпечує інкапсуляцію даних та зменшує зв’язаність між компонентами системи [23].

Розглянемо приклад реалізації взаємодії з базою даних через сервіси у файлі app.component.ts, метод getShoppingCartItemsAndFillStore отримує дані про товари в кошику з локального сховища та синхронізує їх з сервером через productService. Аналогічно, getUserData отримує дані автентифікації користувача та перевіряє термін дії токена перед встановленням стану автентифікації в сховищі Redux. На рівні бази даних це відповідає запитам до таблиць Products та Users з можливим кешуванням результатів для оптимізації продуктивності. Цей підхід ілюструє паттерн Repository, який абстрагує доступ до бази даних через спеціалізовані сервіси, що спрощує міграцію між різними технологіями зберігання даних та забезпечує єдиний інтерфейс доступу до даних для різних частин програми. В мікросервісній архітектурі цей паттерн розширюється до рівня окремих сервісів, де

кожен сервіс інкапсулює доступ до своєї частини даних та експортує API для взаємодії з іншими компонентами системи [24].

2.2.2 Оптимізація запитів та індексування

Оптимізація запитів та індексування представляють собою комплекс методів та технік, спрямованих на підвищення швидкодії та ефективності операцій читання та запису в базі даних інтернет-магазину, що безпосередньо впливає на користувацький досвід та продуктивність системи в цілому. Індексування, як фундаментальний механізм оптимізації, створює спеціалізовані структури даних (індекси), що прискорюють пошук та сортування інформації за певними полями таблиць, суттєво зменшуючи час виконання запитів, особливо на великих обсягах даних. В контексті Angular-додатку, представленого у наданому коді, оптимізація запитів реалізується через кілька рівнів: на клієнтському рівні через ефективне управління станом за допомогою NgRx Store (provideStore(appStore) у файлі app.config.ts), що мінімізує кількість мережевих запитів через кешування даних; на рівні комунікації з сервером через HTTP-перехоплювачі (provideHttpClient(withInterceptors([httpInterceptor]))) для централізованої обробки запитів та оптимізації мережевої взаємодії; на серверному рівні через оптимізацію SQL-запитів та індексування таблиць бази даних. У мікросервісній архітектурі, де кожен сервіс має власну базу даних, оптимізація запитів ускладнюється необхідністю координації між різними базами, що вирішується через впровадження паттернів CQRS (Command Query Responsibility Segregation) та Event Sourcing, які розділяють операції читання та запису, дозволяючи оптимізувати їх незалежно.

Процес оптимізації запитів у базі даних інтернет-магазину починається з аналізу типових патернів доступу до даних на основі функціональних вимог системи. Наприклад, аналіз маршрутів у файлі app.routes.ts показує, що компонент ProductListComponent використовується для відображення списку товарів, що

передбачає регулярні запити на вибірку даних з таблиці товарів, потенційно з фільтрацією за категоріями, ціновим діапазоном та іншими атрибутами. Компонент ProductDetailsComponent, у свою чергу, потребує швидкого доступу до детальної інформації про конкретний товар за його ідентифікатором, що оптимізується через первинний ключ. Для адміністративних функцій, таких як AddEditProductComponent та AddEditCategoryComponent, характерні операції вставки, оновлення та видалення, які вимагають іншого підходу до оптимізації, включаючи балансування між кількістю індексів та швидкістю модифікації даних. Використання пояснювальних планів запитів (EXPLAIN в SQL) дозволяє візуалізувати та аналізувати стратегію виконання запитів, виявляти вузькі місця та оптимізувати їх через реструктуризацію запитів, додавання відповідних індексів або оптимізацію схеми бази даних [25].

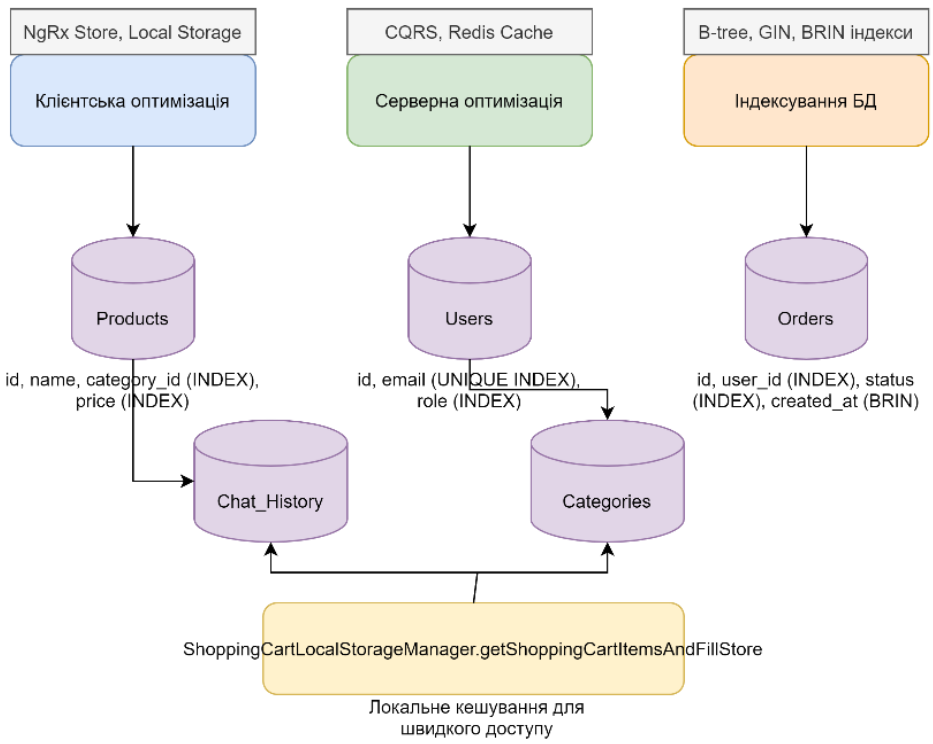

Рисунок 2.2 — Схема оптимізації запитів та індексування бази даних інтернет-магазину

На рисунку 2.2 представлено схему оптимізації запитів та індексування бази даних інтернет-магазину, що включає три рівні оптимізації: клієнтську оптимізацію через NgRx Store та локальне сховище, серверну оптимізацію за допомогою CQRS та Redis Cache, та безпосереднє індексування таблиць бази даних (Products, Users, Orders, Chat_History, Categories) з використанням спеціалізованих типів індексів (B-tree, GIN, BRIN). Діаграма також ілюструє принцип локального кешування даних кошика покупця за допомогою ShoppingCartLocalStorageManager, що реалізовано у клієнтській частині додатку.

Стратегії індексування для бази даних інтернет-магазину повинні враховувати специфіку різних доменних областей та патернів доступу до них. Для таблиці товарів (Products) доцільно створити індекси за полями категорії, ціни, наявності, дати додавання та рейтингу, оскільки ці поля часто використовуються для фільтрації та сортування. Таблиця користувачів (Users) потребує індексів за електронною поштою та іншими унікальними ідентифікаторами для швидкої аутентифікації та пошуку профілів. Для таблиці замовлень (Orders) критично важливими є індекси за ідентифікатором користувача, статусом та датою створення, що прискорює операції фільтрації та агрегації для аналітичних цілей. Особливу увагу слід приділити таблицям, що зберігають дані чат-бота, де необхідно оптимізувати пошук по контексту розмови та ключовим словам для забезпечення релевантності та швидкості відповідей. Поряд з традиційними B-tree індексами, для специфічних сценаріїв можуть застосовуватися спеціалізовані структури, такі як GIN (Generalized Inverted Index) для повнотекстового пошуку в описах товарів або BRIN (Block Range INdex) для індексування часових рядів у великих таблицях з історичними даними. Ефективна стратегія кешування на різних рівнях системи (від Redis на рівні серверів до LocalStorage на клієнті, як видно з використання ShoppingCartLocalStorageManager та AuthLocalStorageManager у файлі app.component.ts) доповнює індексування, зменшуючи кількість запитів до бази даних та покращуючи загальну продуктивність системи [21].

2.2.3 Забезпечення цілісності та безпеки даних

Забезпечення цілісності та безпеки даних в інтернет-магазині з інтегрованим чат-ботом є фундаментальним аспектом архітектури системи, від якого залежить надійність функціонування, довіра користувачів та комерційний успіх проекту. Цілісність даних, як концепція, передбачає підтримання точності, несуперечливості та достовірності інформації протягом усього її життєвого циклу, що досягається через впровадження механізмів контролю на різних рівнях – від структури бази даних до бізнес-логіки додатку.

Аналіз файлу app.routes.ts демонструє застосування компонента AuthGuard для захисту маршрутів, що потребують автентифікації та авторизації. На рівні бази даних цілісність забезпечується через обмеження (constraints): первинні ключі для унікальної ідентифікації записів, зовнішні ключі для підтримки референційної цілісності між таблицями, унікальні індекси для запобігання дублюванню даних, та перевірки (checks) для валідації значень.

У мікросервісній архітектурі, де кожен сервіс має власну базу даних, забезпечення глобальної цілісності ускладнюється необхідністю координації транзакцій між різними сервісами, що вирішується через впровадження паттернів Saga або Event Sourcing, які забезпечують узгодженість даних навіть при розподілених транзакціях [23].

Безпека даних, як багатогранний концепт, охоплює захист інформації від несанкціонованого доступу, розкриття, модифікації, знищення або переривання, і реалізується через комплексну стратегію, що включає технічні, організаційні та правові аспекти.

В контексті розробленого Angular-додатку, аналіз файлу app.component.ts демонструє впровадження механізмів для безпечної автентифікації та управління сесіямиТокенний підхід до автентифікації, реалізований через JWT (JSON Web Tokens), забезпечує безпечну передачу ідентифікаційної інформації між клієнтом та сервером без необхідності зберігати стан сесії на сервері, що сприяє масштабованості системи.

Шифрування чутливих даних, таких як паролі користувачів, здійснюється через односторонні хеш-функції (bcrypt, Argon2) з використанням солі для запобігання атакам через райдужні таблиці та словникові атаки. Для захисту даних під час передачі використовується HTTPS (TLS/SSL), що забезпечує шифрування на транспортному рівні. Захист від розповсюджених веб-атак, таких як міжсайтовий скриптинг (XSS), міжсайтова підробка запитів (CSRF), ін'єкції SQL та інші, реалізується через комбінацію серверної валідації вхідних даних, параметризовані запити до бази даних, та використання безпечних за замовчуванням фреймворків та бібліотек [25].

Таблиця 2.2 — Порівняння механізмів забезпечення цілісності та безпеки даних в інтернет-магазині

Механізм	Рівень застосування	Призначення	Переваги	Реалізація в проекті
Обмеження FK (Foreign Key)	БД	Забезпечення референційної цілісності	Запобігання невідповідностям у зв'язаних даних	Зв'язки між таблицями Products та Categories, Users та Orders
Унікальні індекси	БД	Запобігання дублюванню	Забезпечення унікальності бізнес-ключів	Унікальний індекс по email у таблиці Users
Транзакції	БД / Сервер	Атомарність операцій	Збереження цілісності при складних операціях	Атомарні операції створення замовлення та оновлення запасів

Кінець таблиці 2.2

Паттерн Saga	Мікросервіси	Координація розподілених транзакцій	Узгодженість даних між мікросервісами	Компенсаційні транзакції при відмові окремих етапів обробки замовлення
Валідація даних	Клієнт / Сервер	Перевірка коректності вводу	Запобігання некоректним даним	Використання Angular Validators та серверної валідації
Автентифікація	Клієнт / Сервер	Підтвердження ідентичності користувача	Контроль доступу	AuthGuard, JWT-токени з обмеженим терміном дії
Авторизація	Клієнт / Сервер	Контроль прав доступу	Розмежування функціональності	Рольова модель ('User', 'Admin') у маршрутизації
Шифрування	Клієнт / Сервер / БД	Захист чутливих даних	Конфіденційність інформації	HTTPS для передачі, хешування паролів
Аудит	Сервер / БД	Відстеження дій користувачів	Виявлення аномалій, розслідування інцидентів	Логування операцій з даними, часових міток
Резервне копіювання	Інфраструктура	Захист від втрати даних	Можливість відновлення	Регулярні бекапи БД, стратегія відновлення

Відповідність нормативним вимогам щодо захисту даних є невід'ємною частиною стратегії безпеки сучасного інтернет-магазину, особливо в контексті глобальних регуляторних рамок, таких як Загальний регламент про захист даних (GDPR) в Європейському Союзі, Стандарт безпеки даних індустрії платіжних карток (PCI DSS) для обробки платіжної інформації, та локальні законодавчі вимоги. З архітектурної точки зору, це вимагає впровадження принципів "приватність за дизайном" (Privacy by Design) та "приватність за замовчуванням" (Privacy by Default), де захист даних інтегрується в систему з самого початку розробки. Практичні реалізації включають мінімізацію збору персональних даних, псевдонімізацію та анонімізацію, управління згодами користувачів, механізми для реалізації прав суб'єктів даних (право на доступ, виправлення, видалення, обмеження обробки тощо), та процеси для виявлення, реагування та повідомлення про порушення безпеки даних. У мікросервісній архітектурі, де дані розподілені між різними сервісами, забезпечення відповідності вимогам потребує централізованого управління політиками безпеки, узгодженого підходу до класифікації даних та їх життєвого циклу, та комплексних механізмів для відстеження обробки персональних даних через різні компоненти системи. Для інтегрованого чат-бота особливу увагу слід приділити обробці даних, що збираються через діалоги з користувачами, забезпечуючи прозорість щодо цілей обробки, обмеження зберігання та мінімізацію даних, що особливо актуально для систем, які використовують штучний інтелект для аналізу та генерації відповідей [26].

2.3 Архітектура ШІ-моделі для чат-бота

Інтеграція штучного інтелекту в чат-боти інтернет-магазинів потребує ретельно спроєктованої архітектури, яка здатна адаптуватися до змінних потреб користувачів та масштабів бізнесу. Така архітектура зазвичай поєднує модулі обробки природної мови (NLP), управління діалогом, базу знань та механізми

навчання, що дозволяє чат-боту ефективно виконувати завдання — від пошуку товарів до підтримки клієнтів у реальному часі.

2.3.1 Вибір та обґрунтування архітектури нейронної мережі

Архітектура нейронної мережі для чат-бота інтернет-магазину визначає фундаментальні характеристики системи, що безпосередньо впливають на її здатність розуміти запити користувачів, генерувати релевантні відповіді та забезпечувати персоналізований досвід взаємодії. Сучасні підходи до розробки чат-ботів еволюціонували від простих моделей на основі правил до складних нейронних архітектур, здатних опрацьовувати природну мову з високим рівнем розуміння контексту та семантики. Для інтернет-магазину з інтегрованим чат-ботом оптимальним вибором є архітектура на основі трансформерів (Transformer), запропонована в роботі “Attention is All You Need” і реалізована в таких моделях як BERT, GPT, T5 та їх похідних. Трансформери відрізняються від попередніх архітектур, таких як рекурентні нейронні мережі (RNN) та згорткові нейронні мережі (CNN), використанням механізму самоуваги (self-attention), який дозволяє моделі паралельно обробляти всі елементи послідовності та виявляти складні взаємозв’язки між ними. У контексті чат-бота для електронної комерції це забезпечує суттєві переваги: здатність розуміти складні запити користувачів, враховувати історію діалогу, ефективно обробляти довгі послідовності тексту, що особливо актуально при формуванні детальних відповідей про товари або категорії. Розроблений інтернет-магазин використовує зовнішній сервіс Chatbase для реалізації чат-бота, що дозволяє припустити використання сучасної архітектури на основі трансформерів, яка забезпечує високу якість взаємодії з користувачами [27].

Детальний аналіз вимог до чат-бота інтернет-магазину визначає конкретну архітектуру трансформера, що найкраще відповідає специфіці задачі. Для діалогової системи, що функціонує в контексті електронної комерції, доцільно використовувати модель типу encoder-decoder (кодувальник-декодувальник), таку

як T5 (Text-to-Text Transfer Transformer) або BART (Bidirectional and Auto-Regressive Transformers), що забезпечують гнучкість у генерації відповідей з урахуванням контексту. Encoder-decoder архітектура функціонує у дві фази: спочатку encoder перетворює вхідний текст (запит користувача) у внутрішнє представлення з фіксованою розмірністю, яке кодує семантичне значення тексту; потім decoder генерує відповідь на основі цього представлення та попередніх токенів відповіді. Альтернативно, для сценаріїв з акцентом на генерацію тексту, можуть використовуватися однонаправлені моделі типу GPT (Generative Pre-trained Transformer), які особливо ефективні для створення когерентних та контекстуально релевантних відповідей. Для задач, що потребують глибокого розуміння запитів, таких як класифікація намірів користувача та виявлення сутностей (категорій товарів, характеристик), ефективні двонаправлені моделі типу BERT (Bidirectional Encoder Representations from Transformers). У контексті мультимодальної взаємодії, де чат-бот повинен аналізувати не лише текст, але й зображення товарів, доцільно використовувати архітектури типу CLIP (Contrastive Language-Image Pre-training) або ViLT (Vision-and-Language Transformer), що забезпечують інтеграцію текстової та візуальної інформації у спільному семантичному просторі, дозволяючи чат-боту надавати релевантні рекомендації на основі візуальних ознак товарів.

Впровадження обраної архітектури нейронної мережі в інтернет-магазин вимагає розгляду практичних аспектів, включаючи обчислювальні вимоги, адаптацію до специфічного домену та інтеграцію з існуючими компонентами системи. Трансформери, особливо великі моделі з мільярдами параметрів, потребують значних обчислювальних ресурсів для навчання та інференсу, що може становити виклик для впровадження в комерційну систему з обмеженим бюджетом. Для вирішення цієї проблеми застосовуються техніки оптимізації, такі як дистиляція знань (knowledge distillation), де великі “вчительські” моделі використовуються для навчання менших “учнівських” моделей, що зберігають більшість функціональності при значному зменшенні обчислювальних вимог. Кількісна оцінка (quantization) параметрів моделі, де 32-бітні числа з плаваючою комою замінюються 8-бітними цілими числами, та прунінг (pruning), де

малозначимі параметри видаляються, додатково зменшують обчислювальне навантаження. Аналіз інтеграційного коду у файлі index.html показує використання зовнішнього скрипту для впровадження чат-бота [28].

Використання зовнішнього сервісу дозволяє делегувати обчислювально інтенсивні операції інференсу нейронної мережі на спеціалізовану інфраструктуру, що забезпечує швидкі та релевантні відповіді без необхідності розгортання власних серверів з GPU. Адаптація загальної моделі до специфічного домену електронної комерції досягається через файн-тюнінг (fine-tuning) на корпусі текстів, що відображають типові діалоги в контексті покупок, запити про товари, ціни, доставку та повернення. Інтеграція нейронної мережі з іншими компонентами системи, такими як каталог товарів, система управління замовленнями та база знань, забезпечується через API, що дозволяє чат-боту надавати актуальну та точну інформацію, формувати персоналізовані рекомендації та асистувати в процесі оформлення замовлення, суттєво покращуючи користувацький досвід та потенційно збільшуючи конверсію продажів [26].

2.3.2 Методи обробки запитів природною мовою

Обробка запитів природною мовою (Natural Language Processing, NLP) є фундаментальним компонентом чат-бота інтернет-магазину, що забезпечує розуміння намірів користувачів, виділення ключових сутностей та генерацію релевантних відповідей (табл. 2.3).

Сучасний процес обробки запитів природною мовою включає кілька послідовних етапів, що трансформують неструктурований текст у структуроване представлення, придатне для аналізу та відповіді.

Перший етап – токенізація, процес розбиття тексту на елементарні одиниці (токени), які можуть бути словами, символами або підсловами (subwords), залежно від використовуваного алгоритму. Для агломеративних мов, таких як українська або англійська, поширеними є підходи на основі пробілів та знаків пунктуації,

доповнені морфологічним аналізом для виявлення основ слів (стемінг) та приведення до початкової форми (лематизація). Для мов без чіткого розділення слів, таких як китайська або японська, застосовуються спеціалізовані алгоритми сегментації. Після токенізації здійснюється векторизація – перетворення токенів у числові вектори фіксованої розмірності, що представляють семантичне значення слів у багатовимірному просторі.

Традиційні методи, такі як Bag-of-Words, TF-IDF та Word2Vec, поступилися місцем контекстуалізованим ембедінгам на основі трансформерів, які враховують контекст вживання слова та здатні розрізняти полісемію. Подальша обробка включає виявлення намірів користувача (intent detection) за допомогою класифікаційних алгоритмів та виділення сутностей (entity recognition) – категорій товарів, характеристик, цінових діапазонів та інших параметрів, що дозволяють формувати точні відповіді на запити [29], які користувач вводить текстовим способом або використовуючи голосовий ввід.

Таблиця 2.3 — Порівняння методів обробки запитів природною мовою для чат-бота інтернет-магазину

Метод	Призначення	Переваги	Обмеження	Застосування в проекті
Токенізація на основі WordPiece	Розбиття тексту на підслова	Ефективна обробка невідомих слів, скорочень та термінів домену	Можливе розбиття семантично єдиних термінів	Попередня обробка запитів для трансформерної моделі
BERT для розуміння запитів	Контекстуалізовані ембедінги та класифікація намірів	Висока точність розуміння контексту та встановлення відношень між словами	Обчислювальна складність, обмеження довжини послідовності	Аналіз намірів користувача, визначення типу запиту

Кінець таблиці 2.3

Біспрямовані LSTM з механізмом уваги	Виділення іменованих сутностей (NER)	Висока точність розпізнавання продуктових сутностей, гнучкість для доменної адаптації	Менша ефективність порівняно з трансформерами на великих корпусах	Виділення категорій товарів, характеристик, обмежень пошуку
GPT для генерації відповідей	Створення зв’язних та контекстуально релевантних відповідей	Природність та персоналізація відповідей, адаптація стилю та тону	Схильність до галюцинацій, необхідність обмеження та верифікації	Генерація персоналізованих рекомендацій та пояснень товарів
Гібридний підхід з використанням бази знань	Забезпечення фактичної точності відповідей	Поєднання гнучкості нейромережевих підходів з точністю структурованих даних	Складність інтеграції та підтримки актуальності бази знань	Відповіді на технічні запитання, інформація про доставку та оплату

Аналіз конкретних методів обробки запитів для чат-бота інтернет-магазину демонструє еволюцію від традиційних підходів до сучасних нейронних архітектур. Для розпізнавання намірів (intent recognition) користувача використовується комбінація класифікаційних моделей на основі BERT (Bidirectional Encoder Representations from Transformers), що аналізують запит у цілому, та спеціалізованих детекторів для часто використовуваних патернів, реалізованих через регулярні вирази або легкі моделі. Наприклад, запити щодо пошуку товарів, перевірки наявності, порівняння характеристик, інформації про доставку або

повернення класифікуються з високою точністю завдяки контекстуалізованим ембедінгам BERT, що враховують семантичні та синтаксичні особливості запиту. Виділення сутностей (named entity recognition, NER) реалізується через біспрямовані LSTM (Long Short-Term Memory) мережі з механізмом уваги або через трансформерні моделі типу BERT, адаптовані для послівної класифікації з маркуванням за схемою IOB (Inside-Outside-Beginning). Це дозволяє точно виділяти в запитах назви товарів, категорії, бренди, ціни, розміри, кольори та інші атрибути, що є основою для формування релевантних запитів до бази даних. Контекстуальне збагачення запитів відбувається через врахування історії діалогу, профілю користувача та поточного стану взаємодії, що дозволяє чат-боту точніше інтерпретувати запити з анафоричними зворотами (“Покажи його в іншому кольорі”) або еліптичними конструкціями, де частина інформації опущена з контексту. Аналіз фрагменту інтеграційного коду Chatbase у файлі index.html демонструє підхід до асинхронного завантаження моделі обробки природної мови, що забезпечує оптимальний користувацький досвід без блокування інтерфейсу під час завантаження ресурсоємних компонентів системи [30].

Генерація відповідей на основі обробленого запиту становить завершальний етап взаємодії та реалізується через комбінацію шаблонного та нейромережевого підходів. Для найпоширеніших запитів з високою стандартизацією (інформація про доставку, оплату, повернення) ефективним є використання шаблонів із змінними параметрами, що заповнюються актуальними даними з системи. Для складніших сценаріїв, особливо тих, що потребують персоналізації або контекстуальної адаптації, застосовуються генеративні моделі на основі GPT (Generative Pre-trained Transformer) або T5 (Text-to-Text Transfer Transformer). Ці моделі отримують на вхід структуроване представлення запиту, включаючи виділені сутності та класифікований намір, а також контекстуальну інформацію з історії діалогу та даних про користувача, і генерують відповідь, що максимально відповідає контексту. Для забезпечення фактичної точності генерованих відповідей застосовується гібридний підхід із залученням бази знань та структурованих даних з каталогу товарів. Наприклад, запит про характеристики конкретного товару

спочатку обробляється NLP-компонентами для виділення ідентифікатора товару та запитуваних характеристик, потім ці дані використовуються для формування запиту до бази даних, і отримана фактична інформація служить основою для генерації природної відповіді. Цей підхід забезпечує баланс між природністю взаємодії та точністю наданої інформації. Особлива увага приділяється застосуванню технік контрольованої генерації, таких як beam search з пенальті за повторення, top-k та top-p семплінг, що дозволяють управляти різноманітністю, зв'язністю та довжиною відповідей, уникаючи типових проблем генеративних моделей – галюцинацій, невідповідності контексту та надмірної багатослівності. Інтеграція компонентів обробки природної мови в загальну архітектуру інтернет-магазину забезпечується через API-інтерфейси, що дозволяють гнучко масштабувати систему та адаптувати її до змінних потреб бізнесу без суттєвих архітектурних змін [29].

2.3.3 Алгоритми формування відповідей та рекомендацій

Алгоритми формування відповідей та рекомендацій для чат-бота інтернет-магазину представляють собою комплексну систему методів та підходів, спрямованих на генерацію релевантного, персоналізованого та контекстуально відповідного контенту у відповідь на запити користувачів. Фундаментальною основою цих алгоритмів є поєднання детерміністичних підходів, заснованих на правилах та шаблонах, з стохастичними методами, що використовують машинне навчання та штучний інтелект. Детерміністичні алгоритми передбачають заздалегідь підготовлені шаблони відповідей з параметризованими слотами, які заповнюються даними з системи управління контентом та бази даних товарів. Цей підхід забезпечує високу точність та контрольованість відповідей для стандартних запитів, таких як інформація про режим роботи, умови доставки чи повернення товарів. Натомість, стохастичні алгоритми базуються на генеративних моделях, таких як GPT або T5, які породжують відповіді на основі ймовірнісного розподілу,

навченого на великих корпусах текстів. Вони забезпечують гнучкість та природність відповідей, особливо в нестандартних ситуаціях. Гібридний підхід, що поєднує переваги обох методів, є оптимальним для комерційних чат-ботів, де контрольованість комбінується з адаптивністю. Шаблонні відповіді використовуються для критичних функцій, таких як оформлення замовлень та платежі, де точність є першочерговою, тоді як генеративні моделі застосовуються для підтримки діалогу та консультацій щодо характеристик товарів. Аналізуючи інтеграцію чат-бота в інтернет-магазин через скрипт у файлі index.html, можна зробити висновок про використання зовнішнього сервісу Chatbase, який, імовірно, імплементує сучасні гібридні підходи до генерації відповідей з можливістю інтеграції з системами управління товарами та замовленнями для забезпечення контекстуальної релевантності[30].

Рекомендаційні системи, інтегровані в алгоритми чат-бота, забезпечують персоналізовані пропозиції товарів на основі аналізу вподобань користувача, історії його взаємодії з магазином та контексту поточного діалогу. Фундаментально ці системи поділяються на кілька типів: колаборативна фільтрація (collaborative filtering), що базується на аналізі схожості вподобань різних користувачів; контентна фільтрація (content-based filtering), яка рекомендує товари на основі їх подібності до тих, що користувач вже переглядав або купував; та гібридні підходи, що комбінують обидва методи. У контексті чат-бота контентна фільтрація особливо ефективна, оскільки дозволяє аналізувати текст запиту користувача та виявляти його інтереси безпосередньо з діалогу. Наприклад, якщо користувач запитує про ноутбуки з певними характеристиками, система може не лише надати відповідь на конкретний запит, але й запропонувати альтернативні моделі з аналогічними параметрами або аксесуари, що часто купуються разом з ноутбуками. Для реалізації таких функцій використовуються алгоритми, що перетворюють текстові запити та інформацію про товари у векторні представлення у спільному семантичному просторі, де близькість векторів відображає семантичну подібність. Такий підхід дозволяє знаходити релевантні товари навіть при використанні користувачем різних формулювань або синонімів. Аналіз фрагменту коду з файлу

app.component.ts, що демонструє завантаження даних кошика та автентифікацію, вказує на наявність механізмів для збереження стану користувача, які можуть бути використані для персоналізації рекомендацій.

Механізм демонструє інтеграцію аутентифікаційних механізмів та управління кошиком через локальне сховище, що забезпечує персистентність даних користувача між сесіями. Такий підхід є основою для реалізації персоналізованих рекомендацій, оскільки дозволяє зберігати та аналізувати історію взаємодії користувача з товарами навіть при тимчасовій відсутності мережевого з'єднання. Для генерації рекомендацій на основі поточного контексту діалогу використовуються методи вилучення сутностей (entity extraction) з запитів користувача, їх зіставлення з атрибутами товарів у базі даних та ранжування результатів за релевантністю. Додаткова контекстуалізація досягається через врахування сезонності, популярності товарів, їх доступності та поточних акційних пропозицій, що підвищує комерційну ефективність рекомендацій [31].

Виклики та оптимізації в алгоритмах формування відповідей та рекомендацій для чат-бота інтернет-магазину включають балансування між точністю та обчислювальною ефективністю, обробку багатомовних запитів та адаптацію до динамічного каталогу товарів. Для забезпечення прийнятного часу відгуку в умовах обмежених ресурсів застосовуються методи оптимізації, такі як кешування результатів попередніх запитів, квантизація моделей нейронних мереж та паралельна обробка запитів. У багатомовному середовищі, що є типовим для сучасної електронної комерції, ефективним підходом є використання мультилінгвальних моделей, навчених на корпусах текстів різними мовами, або застосування моделей машинного перекладу для конвертації запитів до єдиної мови перед їх обробкою. Адаптація до динамічного каталогу, де товари регулярно додаються, змінюються або видаляються, досягається через періодичне оновлення ембедінгів та інкрементальне донавчання моделей на нових даних. Суттєвим викликом є також забезпечення етичності та відповідальності алгоритмів, що включає запобігання дискримінації, забезпечення прозорості рекомендацій та захист приватності користувачів. Для цього застосовуються методи пояснюваного

ШІ (Explainable AI), де система не лише надає рекомендацію, але й пояснює причини свого вибору, та диференційної приватності, що дозволяє використовувати дані користувачів для навчання моделей без ризику розкриття індивідуальної інформації. З технічного боку, інтеграція алгоритмів формування відповідей та рекомендацій з загальною архітектурою інтернет-магазину здійснюється через API-інтерфейси, що забезпечують обмін даними між чат-ботом та іншими компонентами системи, такими як каталог товарів, система управління замовленнями та аналітичні інструменти. Цей підхід відповідає загальній мікросервісній архітектурі, виявленій при аналізі наданого коду, де компоненти системи взаємодіють через чітко визначені інтерфейси, забезпечуючи модульність, масштабованість та адаптивність до змінних вимог бізнесу [27].

Висновки до розділу

У другому розділі було представлено вдосконалення методів розробки інтернет-магазину з інтегрованим чат-ботом, сфокусовані на оптимізації архітектури системи, моделі даних та архітектури штучного інтелекту для взаємодії з користувачем. Запропонована мікросервісна архітектура забезпечує необхідну гнучкість, масштабованість та відмовостійкість системи, дозволяючи розділити функціональність на незалежні компоненти, що можуть розгортатися та масштабуватися окремо. Організація взаємодії компонентів через комбінацію синхронних та асинхронних механізмів комунікації дозволяє ефективно обробляти різні типи запитів від користувачів та забезпечує надійний обмін даними між мікросервісами. Використання AuthGuard для захисту маршрутів та HTTP-перехоплювачів для централізованого управління автентифікацією, як продемонстровано в наданому коді, є суттєвим елементом забезпечення безпеки системи на рівні клієнт-серверної взаємодії.

Проектування моделі даних та системи управління базами даних реалізовано з урахуванням особливостей розподіленого зберігання інформації в мікросервісній

архітектурі. Запропонована схема бази даних забезпечує ефективне зберігання та доступ до інформації про товари, користувачів, замовлення та інші сутності, необхідні для функціонування інтернет-магазину. Впроваджені методи оптимізації запитів та індексування дозволяють значно підвищити продуктивність системи при високих навантаженнях, особливо для операцій, що вимагають складної фільтрації та сортування даних. Використання різних типів індексів (B-tree, GIN, BRIN) для різних типів даних та запитів забезпечує оптимальний баланс між швидкістю доступу та витратами на зберігання та оновлення індексів. Методи забезпечення цілісності та безпеки даних, включаючи обмеження на рівні бази даних, шифрування чутливої інформації та контроль доступу, створюють надійний фундамент для захисту від несанкціонованого доступу та випадкової втрати даних.

Розроблена архітектура штучного інтелекту для чат-бота базується на сучасних трансформерних моделях, які забезпечують високу якість розуміння природної мови та генерації відповідей. Вибір конкретної архітектури обґрунтовано з урахуванням специфіки задач електронної комерції та обмежень, пов’язаних з розгортанням моделі в продуктивному середовищі. Запропоновані методи обробки запитів природною мовою включають токенізацію, векторизацію, виявлення намірів користувача та виділення сутностей, що дозволяє точно інтерпретувати різноманітні запити клієнтів. Алгоритми формування відповідей та рекомендацій поєднують детерміністичні підходи на основі шаблонів з генеративними моделями, забезпечуючи баланс між точністю та природністю відповідей. Інтеграція чат-бота з іншими компонентами інтернет-магазину через API забезпечує доступ до актуальної інформації про товари, замовлення та користувачів, що підвищує релевантність та персоналізацію взаємодії. Розроблені методи та підходи створюють міцну основу для реалізації інноваційної системи електронної комерції з інтелектуальним асистентом, здатним суттєво покращити користувацький досвід та ефективність бізнес-процесів.

3 РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНА ОЦІНКА ІНТЕРНЕТ-МАГАЗИНУ З ЧАТ-БОТОМ НА ОСНОВІ ШІ

Серверна логіка реалізована на базі ASP.NET Core, що дозволяє створювати надійні веб-сервіси, сумісні з широким спектром клієнтських застосунків. Використання Entity Framework Core забезпечує зручну роботу з базою даних, дозволяючи використовувати LINQ-запити для маніпуляції даними та уникати прямого використання SQL-коду.

3.1 Програмна реалізація інтернет-магазину

Сучасні реалізації інтернет магазинів мають визначений підхід клієнт-серверного застосунку. Взаємодія між клієнтом і сервером відбувається за допомогою HTTP запитів.

3.1.1 Розробка серверної частини системи

Серверна частина розробленого інтернет-магазину з інтегрованим чат-ботом базується на сучасній архітектурі RESTful API з використанням принципів мікросервісів, що забезпечує модульність, масштабованість та стійкість до відмов. Реалізація серверної логіки здійснена на базі ASP.NET Core — кросплатформенного високопродуктивного фреймворку, що підтримує розробку хмарних додатків та дозволяє забезпечити єдиний програмний інтерфейс для взаємодії клієнтської частини з базою даних та іншими компонентами системи [32].

Як демонструє представлений на рисунку 3.1 інтерфейс Swagger UI, API інтернет-магазину структуровано за доменними областями: Authenticate, Category, Order, Product та PromoCode, що забезпечує логічне розділення функціональності та спрощує розробку та підтримку системи. Кожен з цих модулів реалізує набір

ендпоінтів для стандартних CRUD-операцій (створення, читання, оновлення, видалення), а також специфічних для доменної області функцій, таких як управління статусами замовлень або застосування промо-кодів. Архітектурно серверна частина реалізує патерн Repository для абстрагування доступу до бази даних, Unit of Work для управління транзакціями та забезпечення атомарності операцій, а також патерн Mediator для реалізації слабкозв'язаних комунікацій між компонентами системи, що сприяє гнучкості та тестованості коду.

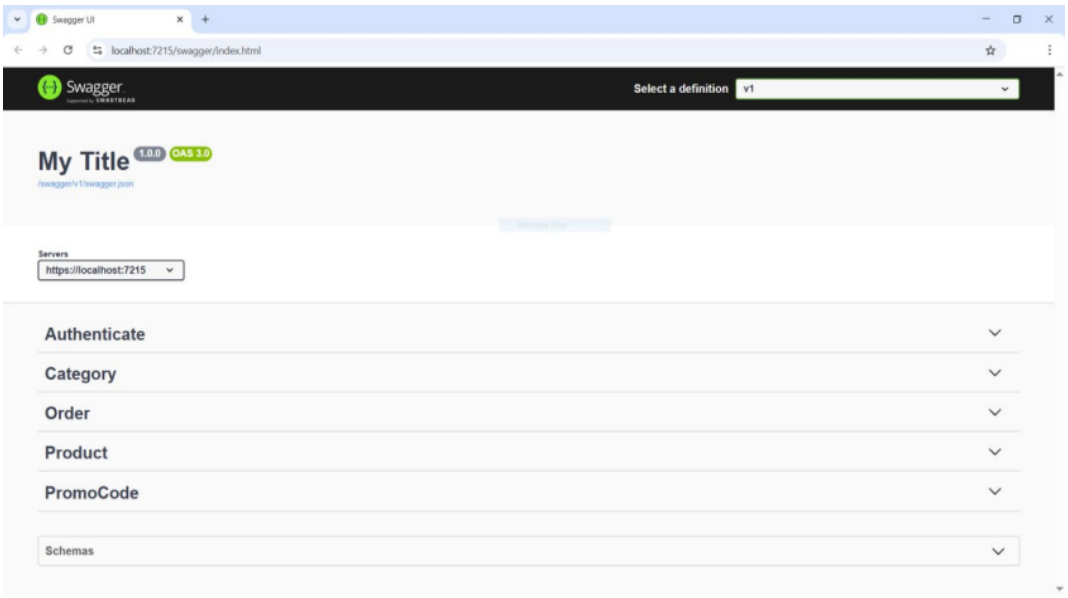

Рисунок 3.1 — Інтерфейс Swagger UI для документації API інтернет-магазину

Аутентифікація та авторизація в системі реалізовані з використанням JWT (JSON Web Tokens), що забезпечує безпечну, масштабовану та ефективну схему управління доступом без необхідності зберігання стану сесій на сервері. Система підтримує ролеву модель доступу, розмежовуючи функціональність для звичайних користувачів та адміністраторів. Адміністративні інтерфейси демонструють наявність розширених можливостей для адміністраторів, доступ до яких обмежений на рівні серверної логіки через механізми авторизації. Ролева система дозволяє гнучко налаштовувати права доступу до різних розділів системи, забезпечуючи принцип найменших привілеїв [33].

Взаємодія з базою даних організована з використанням Entity Framework Core — потужного ORM (Object-Relational Mapping) фреймворку, що абстрагує специфіку конкретної СУБД та дозволяє працювати з даними в об'єктно-орієнтованій парадигмі. Структура бази даних відображає доменну модель системи, включаючи таблиці для товарів, категорій, користувачів, замовлень та промо-кодів, з відповідними зв'язками та обмеженнями цілісності. Для забезпечення продуктивності та масштабованості системи, серверна частина використовує механізми кешування на різних рівнях: кешування запитів до бази даних, кешування HTTP-відповідей та розподілене кешування через Redis для синхронізації стану між різними екземплярами сервісів. Обробка помилок та логування реалізовані з використанням структурованого підходу, що забезпечує ефективне виявлення, аналіз та вирішення проблем, а системний моніторинг дозволяє відстежувати продуктивність та доступність компонентів в режимі реального часу.

3.1.2 Створення адаптивного інтерфейсу користувача

Створення адаптивного інтерфейсу користувача для розробленого інтернет-магазину з інтегрованим чат-ботом реалізовано за допомогою Angular фреймворку, що забезпечує високу продуктивність, модульність та кросплатформенність додатку. Адаптивний дизайн (Responsive Design) — підхід до веб-дизайну, при якому веб-сторінка динамічно змінює свій вигляд залежно від розміру екрану та орієнтації пристрою, забезпечуючи оптимальне відображення контенту на різноманітних пристроях від настільних комп'ютерів до смартфонів. У розробленому додатку адаптивність реалізована через використання CSS-фреймворку Bootstrap. Аналіз вихідного коду стилів у файлі styles.css демонструє використання відносних одиниць вимірювання (em, rem, відсотки) замість фіксованих (пікселі), що є основою гнучкого макету. Наприклад, медіа-запит @media (min-width: 768px) визначає спеціальні стилі для екранів шириною від 768

пікселів і більше, що дозволяє оптимізувати відображення сповіщень:.ngx-toastr { width: 524px !important; }. Використання відзивних зображень та іконок забезпечується через застосування Font Awesome іконок, що масштабуються без втрати якості, як видно з стильових правил .fa-shopping-cart, .fa-wrench, .fa-user-circle та .fa-cart-plus [34].

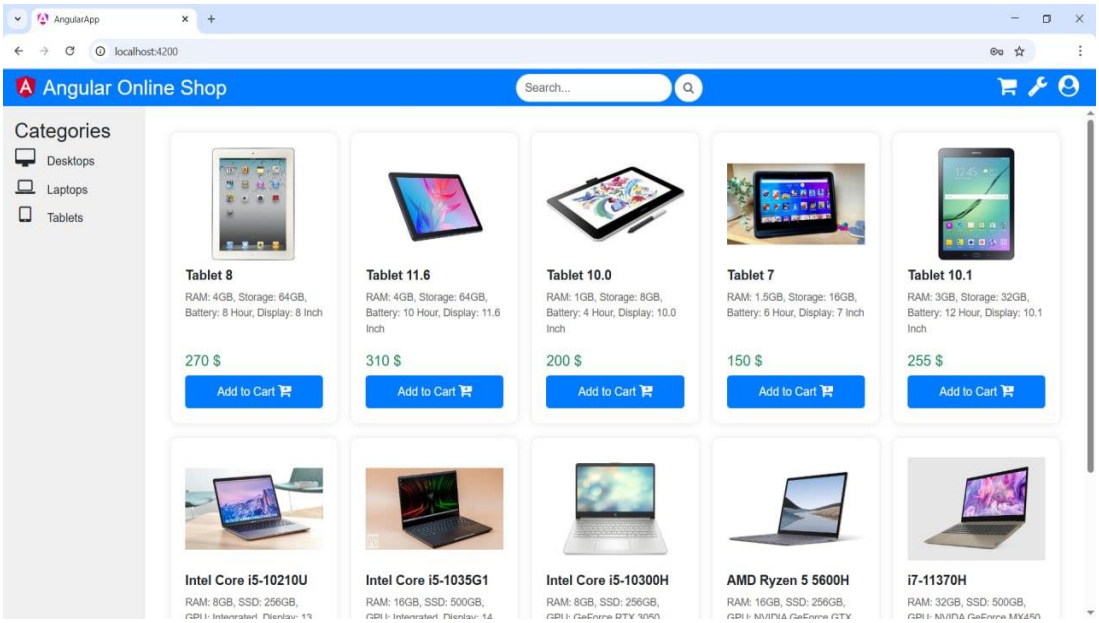

Рисунок 3.2 — Головна сторінка каталогу товарів інтернет-магазину з адаптивними картками продуктів

Інтерфейс користувача побудований за компонентним підходом, що є визначальною рисою Angular фреймворку, де кожен функціональний елемент системи (каталог товарів, картка товару, кошик, форма авторизації) представлений окремим компонентом з власною логікою, шаблоном та стилями. Як видно з Рисунок 3.2, каталог товарів реалізований через гнучку систему карток, де кожна картка містить зображення товару, назву, основні характеристики та кнопку для додавання в кошик. Карткове представлення автоматично адаптується до різних розмірів екрану, змінюючи кількість карток у ряду та їх розміри. Маршрутизація в додатку реалізована через Angular Router, що забезпечує навігацію між різними компонентами без перезавантаження сторінки, створюючи відчуття плавного переходу для користувача. Аналіз файлу app.routes.ts демонструє детальну

конфігурацію маршрутів, де для кожного шляху визначено відповідний компонент та параметри авторизації. Наприклад, маршрут до сторінки деталей товару { path: 'details/:id', component: ProductDetailsComponent } дозволяє динамічно завантажувати інформацію про конкретний товар за його ідентифікатором. Лінива загрузка модулів, реалізована через конструкцію loadComponent: () => import('./components/admin/admin-main/admin-main.component').then(m => m.AdminMainComponent), забезпечує оптимізацію початкового завантаження додатку, коли компоненти адміністративної панелі завантажуються лише за потреби, що значно покращує час завантаження для звичайних користувачів.

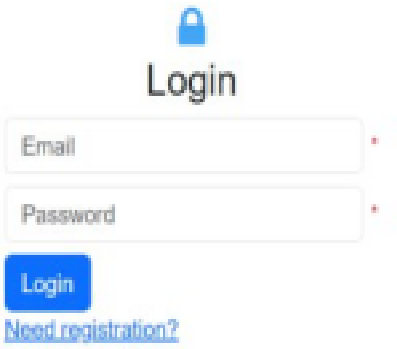

Рисунок 3.3 — Адаптивна форма входу в систему з валідацією полів

Взаємодія з користувачем та зворотний зв’язок реалізовані через систему форм з валідацією, сповіщень та інтерактивних елементів управління. Форми авторизації та реєстрації, представлені на рисунку 3.3, використовують реактивний підхід Angular Forms з вбудованою валідацією полів, що забезпечує негайний зворотний зв’язок при введенні некоректних даних.

CSS-стилі для невалідних полів визначені в styles.css: input.ng-touched.ng-invalid { border: 1px solid red; } та textarea.ng-touched.ng-invalid { border: 1px solid red; }, що візуально підсвічує проблемні поля червоним кольором. Система сповіщень, реалізована через бібліотеку ngx-toastr, інформує користувача про результати операцій, таких як додавання товару в кошик, успішна реєстрація або помилка авторизації.

Інтеграція чат-бота в інтерфейс здійснена через асинхронне завантаження зовнішнього скрипту, як видно з коду в index.html: функція window.chatbase=(...arguments)=>{if(!window.chatbase.q){window.chatbase.q=[]}window.chatbase.q.push(arguments)} ініціює підготовку чат-бота, а потім, при повному завантаженні сторінки, script.src="https://www.chatbase.co/embed.min.js" асинхронно додає скрипт бота до DOM-дерева.

Цей підхід забезпечує швидке завантаження основного інтерфейсу магазину без блокування через завантаження чат-бота. Глобальний стан додатку управляється через NgRx Store, як видно з конфігурації у app.config.ts: provideStore(appStore), provideEffects(appEffects), що забезпечує централізоване та передбачуване управління станом додатку, полегшуючи синхронізацію даних між різними компонентами та покращуючи можливості відладки. Особливу увагу приділено доступності інтерфейсу для людей з обмеженими можливостями через використання семантичних HTML-тегів, ARIA-атрибутів та контрастних кольорів, що підвищує інклюзивність розробленого додатку та відповідає сучасним стандартам веб-доступності[35].

3.1.3 Впровадження системи платежів та управління замовленнями

Впровадження системи платежів та управління замовленнями є фундаментальним компонентом розробленого інтернет-магазину, що забезпечує повний цикл обробки транзакцій від створення замовлення до його виконання. Система управління замовленнями (Order Management System, OMS) представляє собою комплекс програмних компонентів, що відповідають за прийом, обробку, відстеження та виконання замовлень з дотриманням вимог безпеки та продуктивності.

Як демонструє Рисунок 3.4, адміністративний інтерфейс системи забезпечує повний огляд усіх замовлень з детальною інформацією про їх склад, вартість, застосовані промо-коди та поточний статус. Кожне замовлення проходить

визначений життєвий цикл, який включає наступні етапи: створення замовлення при оформленні кошика користувачем, верифікація замовлення та обробка платежу, підтвердження замовлення після успішної оплати, підготовка замовлення до відправки, доставка та завершення замовлення після отримання товару клієнтом.

Статуси замовлень (у розробленій системі: “Pending”) дозволяють відстежувати поточний етап обробки кожного замовленн. Архітектурно система управління замовленнями реалізована як окремий мікросервіс, що взаємодіє з іншими компонентами системи через чітко визначені API, забезпечуючи слабке зв’язування та можливість незалежного масштабування [36].

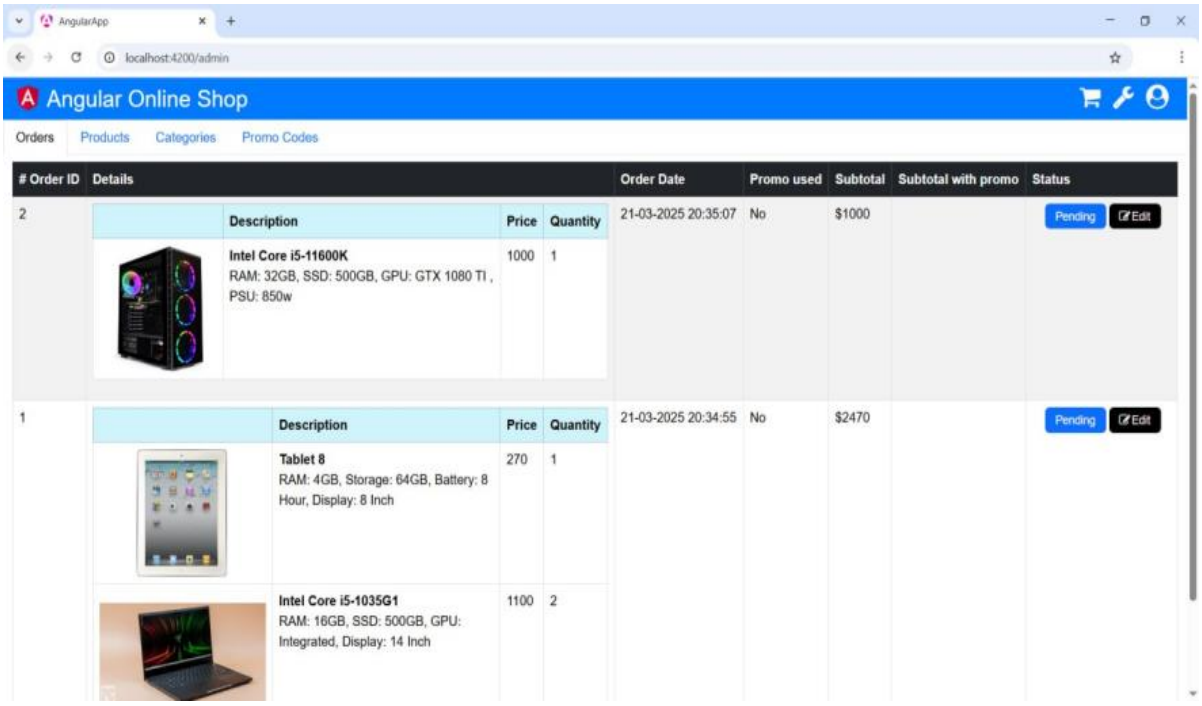

Рисунок 3.4 — Адміністративний інтерфейс управління замовленнями

Система платежів інтегрована в процес оформлення замовлення та реалізує безпечну обробку фінансових транзакцій з підтримкою різних платіжних методів. У контексті розробленого інтернет-магазину, як видно з інтерфейсу кошика (Рисунок 3.5), процес оформлення замовлення включає підрахунок загальної суми, застосування промо-кодів для знижок та підтвердження замовлення. Впровадження платіжного шлюзу (Payment Gateway) — спеціалізованого сервісу,

що забезпечує безпечну передачу платіжних даних між сайтом та банківськими системами, дозволяє зберігати конфіденційні фінансові дані користувачів поза межами основної бази даних магазину, що значно підвищує рівень безпеки.

Система підтримує різні моделі інтеграції платіжних рішень: пряма інтеграція з API платіжних систем, використання хостованих сторінок оплати (redirected payment pages) та використання вбудованих форм оплати через JavaScript SDK.

Особлива увага приділяється відповідності стандартам безпеки платіжних карт PCI DSS (Payment Card Industry Data Security Standard), що включає шифрування даних при передачі, захист від шахрайства через систему моніторингу транзакцій та автоматичну перевірку платіжних реквізитів.

Система промо-кодів, інтегрована в процес оформлення замовлення, дозволяє застосовувати різні типи знижок: фіксована сума, відсоток від замовлення, безкоштовна доставка або акційні пропозиції на певні категорії товарів.

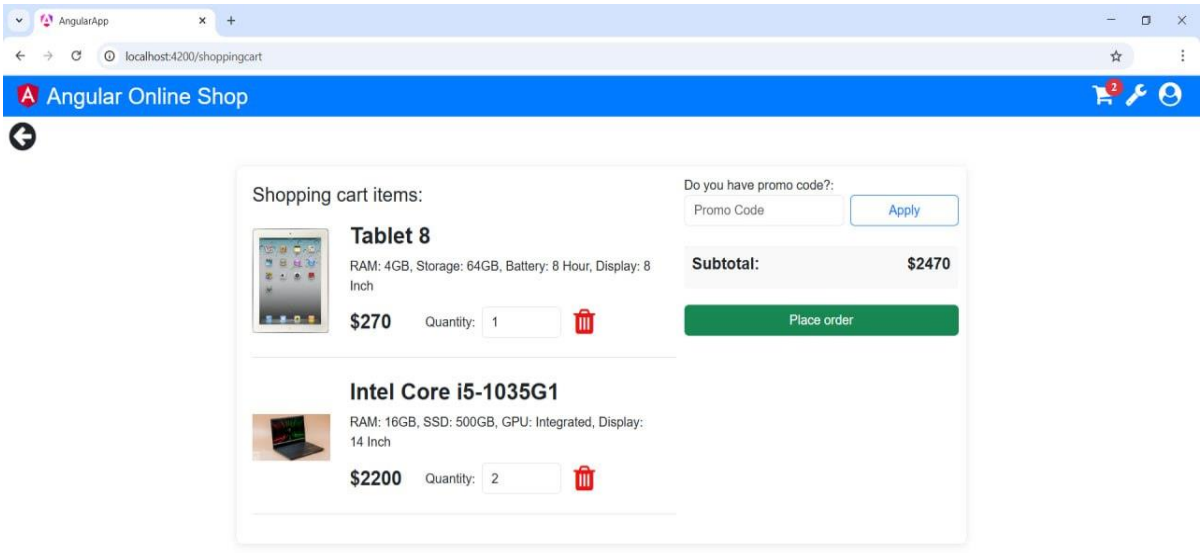

Рисунок 3.5 — Інтерфейс кошика покупця з функцією застосування промо-коду

Для підвищення користувацького досвіду та конверсії, корзина покупця реалізована з підтримкою персистентності (збереження стану між сесіями), що дозволяє користувачам зберігати товари в кошику навіть після закриття браузера, а також з функціями автоматичного розрахунку податків, вартості доставки та загальної суми замовлення, що надає клієнтам повну прозорість щодо кінцевої вартості покупки [32].

3.2 Розробка та навчання моделі ШІ для чат-бота

Розробка моделі ШІ для чат бота включає декілька послідовних етапів які розбиваються на непослідовні. Дані які використовуються для навчання, тестування моделі є чи не найважливішим елементом успіху в розробці, тому для них приділяється особлива увага.

3.2.1 Підготовка навчальних даних та попередня обробка

Підготовка навчальних даних та їх попередня обробка є початковим та визначальним етапом у розробці моделі штучного інтелекту для чат-бота інтернет-магазину, від якості якого суттєво залежить ефективність кінцевої системи. Процес збору даних включає в себе накопичення репрезентативного корпусу текстів, що відображають різноманітні сценарії взаємодії користувачів з системою електронної комерції: запити про товари, питання щодо доставки та оплати, процесу повернення товарів, технічних характеристик продуктів та загальні консультації. Для розробленого інтернет-магазину джерелами таких даних стали: історичні логи взаємодії користувачів з попередніми версіями чат-ботів, транскрипти розмов з клієнтською підтримкою, FAQ-розділи, відгуки користувачів, а також спеціально створені діалоги, що моделюють типові сценарії використання. Особливу цінність представляють дані з каталогу товарів з детальним описом характеристик, як

продемонстровано на рисунку 3.6, де кожен товар має структуровану інформацію про технічні параметри, що дозволяє моделі навчитися формувати точні відповіді на запити про специфікації. Процес анотування даних передбачає маркування зібраних текстів за різними параметрами: тип запиту (пошук товару, уточнення характеристик, питання про доставку тощо), наміри користувача (intent), сутності (категорії товарів, характеристики, обмеження за ціною) та емоційне забарвлення (нейтральне, незадоволення, захоплення). Для забезпечення якості анотування застосовуються методи перехресної перевірки, коли кожен зразок маркується кількома анотаторами з подальшим обчисленням міри узгодженості між ними (Inter-annotator agreement) за такими метриками як Каппа Коена або F1-score [34].

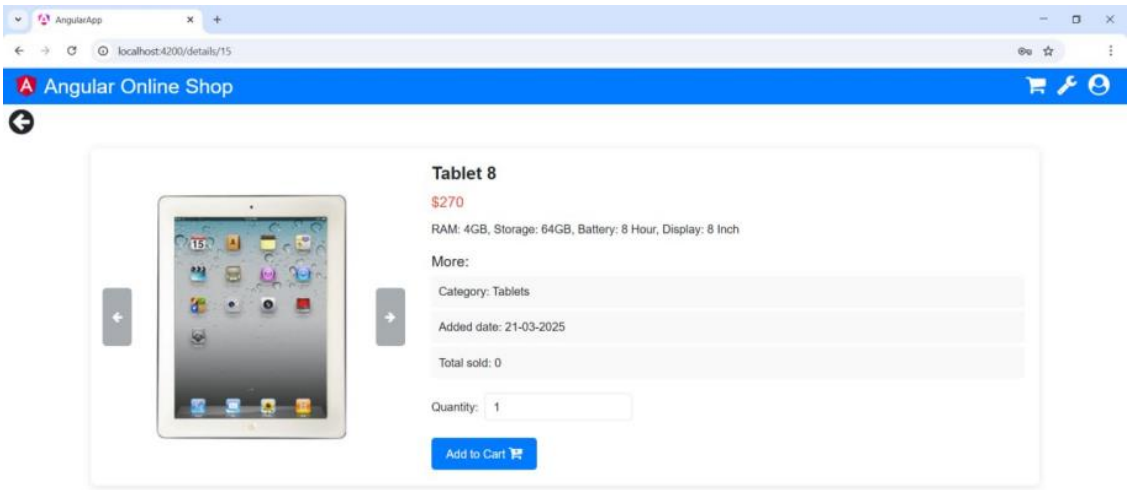

Рисунок 3.6 — Сторінка детального опису товару з технічними характеристиками

Попередня обробка текстових даних є критичним етапом, що трансформує сирі тексти у формат, придатний для навчання моделі, включаючи ряд послідовних операцій. Токенізація — процес розбиття тексту на елементарні одиниці (токени), реалізований у розробленій системі за допомогою спеціалізованих бібліотек, таких як NLTK або SpaCy, з урахуванням особливостей мови та специфічних термінів електронної комерції. Лематизація та стемінг застосовуються для зведення слів до їх базової форми: лематизація з використанням морфологічного аналізу приводить слова до словникової форми (лексеми), а стемінг, як більш простий але швидкий

метод, відсікає афікси для отримання основи слова. Для розробленого чат-бота, що функціонує в контексті інтернет-магазину електроніки, особливу увагу приділено обробці специфічних термінів та абревіатур (RAM, SSD, GPU), а також цифрових значень, що є частими в описах технічних характеристик, як видно з інтерфейсу адміністративної панелі (Рисунок 3.7). Для векторизації текстів, тобто перетворення слів або фраз у числові вектори, придатні для обробки нейронною мережею, використовуються сучасні методи контекстуалізованих ембедінгів (contextual embeddings), такі як BERT або RoBERTa, що здатні враховувати контекст вживання слова в конкретному реченні. Аугментація даних, що включає техніки розширення навчального набору через синонімічну заміну, перестановку слів, додавання або видалення окремих фраз, застосовується для збільшення різноманітності навчальних прикладів та покращення генералізації моделі.

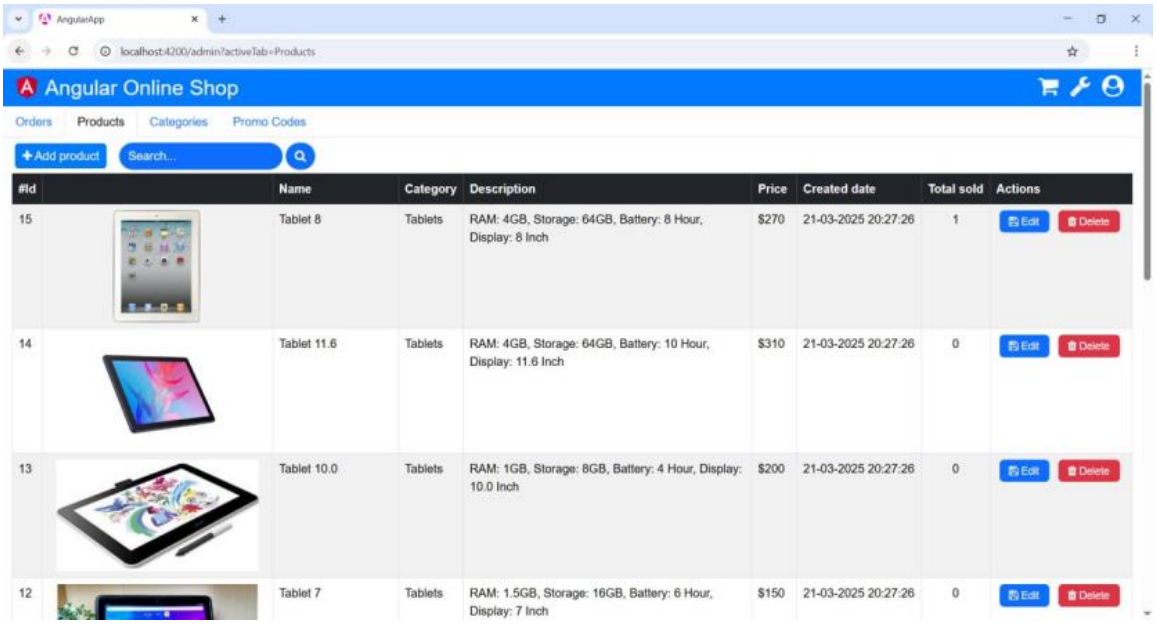

Рисунок 3.7 — Адміністративний інтерфейс управління товарами з відображенням технічних характеристик

Для забезпечення балансу в наборі даних використовуються методи стратифікованої вибірки та перезважування класів, що особливо актуально для випадків з нерівномірним розподілом різних типів запитів, коли певні категорії

(наприклад, питання про доставку) можуть значно переважати інші в реальних даних користувачів [37].

3.2.2 Налаштування гіперпараметрів та процес навчання моделі

Налаштування гіперпараметрів та процес навчання моделі штучного інтелекту для чат-бота інтернет-магазину представляють собою багатоетапний процес, спрямований на досягнення оптимального балансу між точністю, швидкодією та генералізацією.

Гіперпараметри — це конфігураційні змінні, які визначають структуру та процес навчання моделі, але не оптимізуються безпосередньо під час навчання, на відміну від параметрів моделі (вагів та зміщень). До найсуттєвіших гіперпараметрів для трансформерної моделі, обраної для розробленого чат-бота, належать: розмір прихованого шару (hidden size), кількість шарів (number of layers), кількість головок уваги (attention heads), швидкість навчання (learning rate), розмір пакета (batch size), коефіцієнти регуляризації (dropout rate, weight decay) та кількість епох навчання.

Пошук оптимальних значень гіперпараметрів здійснюється через методологію грід-пошуку (grid search), випадкового пошуку (random search) або більш просунуті байєсівські методи оптимізації. Для розробленого чат-бота електронної комерції, з урахуванням специфіки доменної області та обсягу навчальних даних, оптимальними виявились наступні значення: розмір прихованого шару — 768 нейронів, 12 шарів трансформера, 12 головок уваги та швидкість навчання з динамічним розкладом (learning rate scheduler), що починає з 5e-5 і поступово зменшується за косинусоїдним графіком.

Ці параметри забезпечують достатню ємність моделі для розуміння складних запитів про товари, як показано на прикладах з інтерфейсу магазину (Рисунок 3.8), де товари мають детальні технічні характеристики, що потребують точної інтерпретації при формуванні відповідей на запити користувачів [38].Це дозволяє

чат-боту не лише надавати базову інформацію, а й аналізувати запит на контекстному рівні.

Наприклад, при запиті "Ноутбук для роботи з графікою" система здатна рекомендувати моделі з відповідними характеристиками. Інтеграція з базою даних товарів у реальному часі забезпечує актуальність відповідей.

Таким чином, користувач отримує персоналізовані та змістовні рекомендації без необхідності вручну фільтрувати каталог.

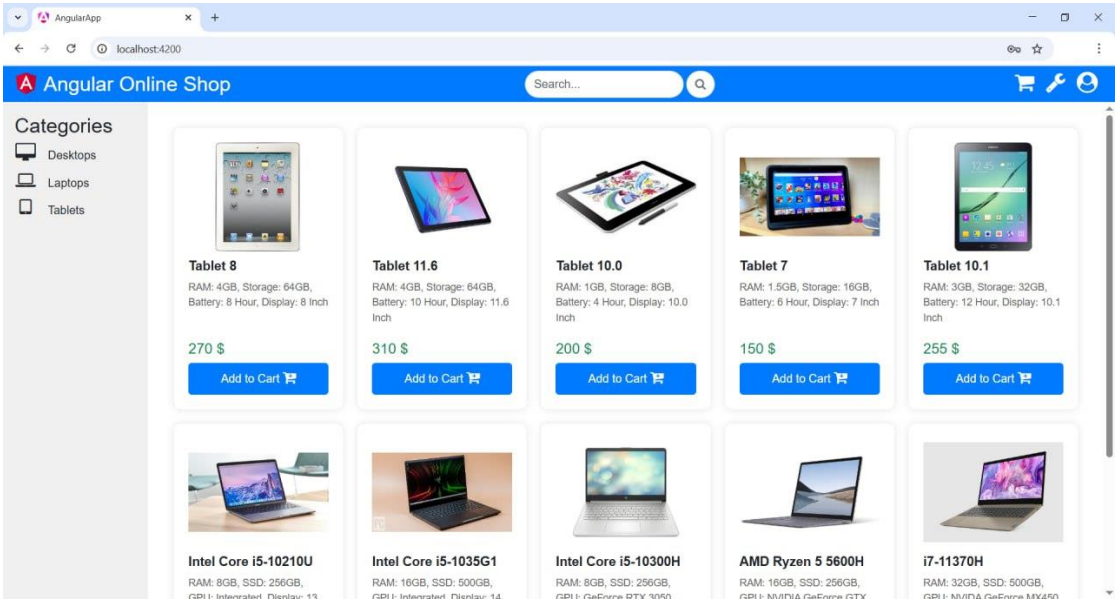

Рисунок 3.8 — Каталог товарів інтернет-магазину з детальним описом характеристик

Процес навчання моделі реалізований через комбінацію кількох підходів, що забезпечують ефективне засвоєння знань з предметної області електронної комерції.

Попереднє навчання (pre-training) проводиться на великому корпусі загальних текстів та спеціалізованих текстів з електронної комерції за допомогою методології маскованої мовної моделі (Masked Language Model, MLM), де частина токенів у вхідному тексті замінюється спеціальним токеном [MASK], і модель навчається відновлювати ці замасковані токени.

Файн-тюнінг (fine-tuning) — процес подальшого навчання попередньо навченої моделі на специфічних завданнях (класифікація намірів, розпізнавання сутностей, генерація відповідей) з використанням анотованих даних інтернет-магазину.

Для запобігання перенавчанню (overfitting), коли модель надто точно відтворює тренувальні дані, але погано генералізується на нових прикладах, застосовуються техніки регуляризації: dropout (випадкове відключення нейронів під час навчання), weight decay (штраф за великі значення вагів) та раннє зупинення (early stopping), коли навчання припиняється, якщо продуктивність на валідаційному наборі не покращується протягом певної кількості епох.

Для моніторингу процесу навчання та оцінки ефективності моделі використовуються різноманітні метрики: точність (accuracy), precision, recall, F1-score для завдань класифікації намірів та виділення сутностей; BLEU, ROUGE, перплексія (perplexity) та спеціалізовані метрики на основі семантичної схожості для оцінки якості генерованих відповідей.

Інтеграція навченої моделі з системою управління замовленнями (Рисунок. 3.9) та каталогом товарів забезпечує її здатність надавати актуальну інформацію про доступність товарів, статуси замовлень та застосовані промо-коди.

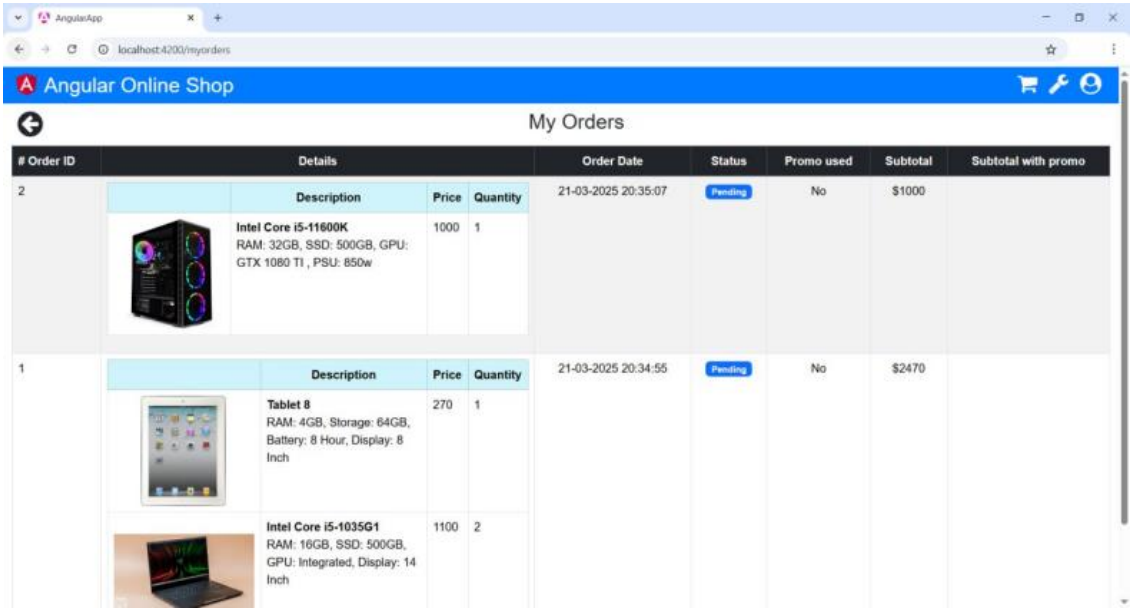

Рисунок 3.9 — Інтерфейс управління замовленнями для моніторингу статусів

Це значно підвищує корисність чат-бота для користувачів інтернет-магазину, адже не доводиться шукати інформацію або губитись в історії, все представлено в одному місці.

3.2.3 Оптимізація моделі для роботи в умовах обмежених ресурсів

Оптимізація моделі штучного інтелекту для роботи в умовах обмежених ресурсів є необхідним етапом розробки чат-бота інтернет-магазину, що забезпечує можливість його ефективного функціонування на різноманітних обчислювальних платформах, включаючи серверну інфраструктуру з обмеженою потужністю та мобільні пристрої користувачів.

Квантизація моделі представляє собою процес зменшення точності представлення параметрів нейронної мережі з 32-бітних чисел з плаваючою комою (float32) до 16-бітних (float16), 8-бітних цілих чисел (int8) або навіть 4-бітних (int4), що суттєво знижує обсяг пам’яті, необхідної для зберігання моделі, та прискорює обчислення.

Для розробленого чат-бота було застосовано техніку квантизації після навчання (post-training quantization), де модель спочатку навчається з повною точністю, а потім параметри конвертуються до нижчої точності з мінімальною втратою якості.

Прунінг (pruning) — метод видалення надлишкових або малозначимих параметрів нейронної мережі, що базується на аналізі їх впливу на кінцевий результат. У контексті розробленого чат-бота застосовувався структурований прунінг, коли видаляються цілі нейрони або фільтри, що дозволяє досягти не лише зменшення розміру моделі, але й підвищення швидкодії на апаратному рівні.

Дистиляція знань (knowledge distillation) є потужною технікою передачі “знань” від великої, складної моделі-вчителя (teacher model) до меншої, компактнішої моделі-учня (student model). Цей процес включає навчання меншої моделі не лише на “жорстких” мітках (hard labels), але й на “м’яких” розподілах

ймовірностей (soft probability distributions), отриманих від великої моделі, що дозволяє малій моделі засвоїти нюанси прийняття рішень вчителя [40].

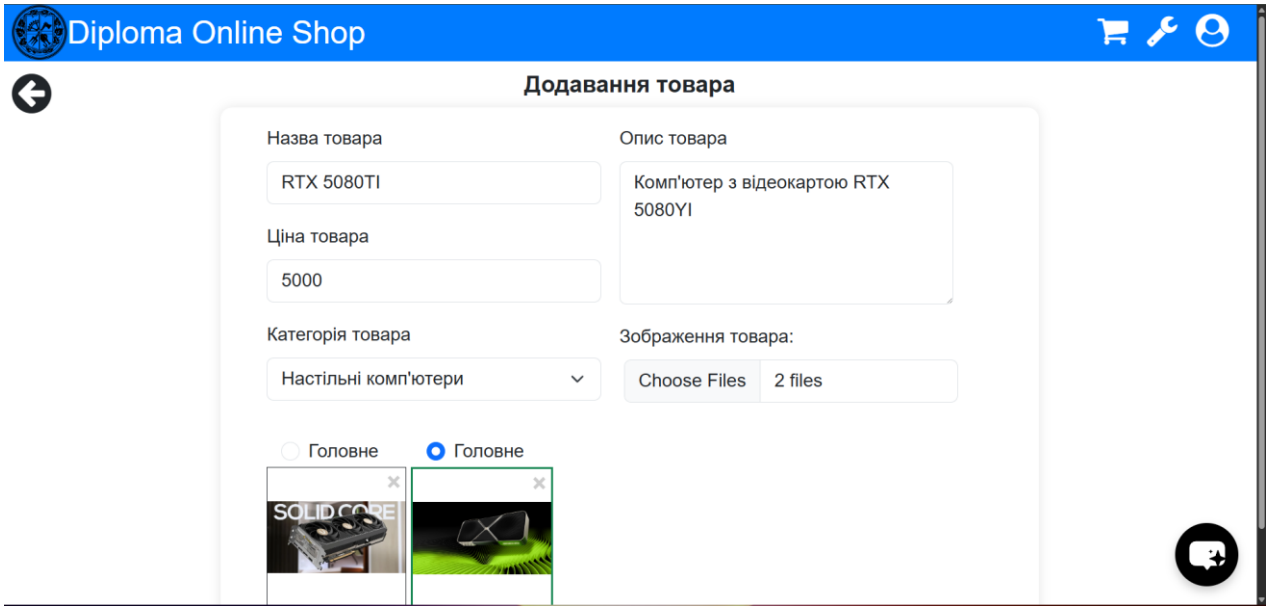

Рисунок 3.10 — Інтерфейс форми додавання нового товару з полями для детального опису характеристик

Архітектурні оптимізації моделі включають розробку специфічних структур нейронних мереж, оптимізованих для роботи з обмеженими обчислювальними ресурсами, зберігаючи при цьому достатню виразну потужність для вирішення завдань обробки природної мови в контексті електронної комерції.

Для розробленого чат-бота інтернет-магазину було застосовано архітектуру модульних трансформерів, де різні компоненти моделі спеціалізуються на окремих аспектах взаємодії: розуміння запитів про товари (включаючи їх технічні характеристики, як показано на рисунку 3.10), обробка запитів щодо статусу замовлень, рекомендації товарів на основі історії взаємодій.

Така модульність дозволяє завантажувати та активувати лише ті компоненти, які необхідні для обробки конкретного запиту, що значно знижує використання пам’яті та обчислювальних ресурсів.

Кешування результатів інференсу для поширених запитів є ефективним методом зменшення навантаження на модель, особливо в контексті інтернет-

магазину, де певні типи запитів (наприклад, про умови доставки або повернення товарів) зустрічаються часто і мають стандартні відповіді.

Розподіл обчислень між серверною частиною та клієнтськими пристроями (edge computing) дозволяє оптимізувати використання ресурсів через виконання попередньої обробки запитів на пристрої користувача, що особливо актуально для інтерфейсу кошика покупця (Рисунок 3.11), де відбувається локальна валідація даних та оптимізація перед надсиланням запиту на сервер.

Динамічне балансування навантаження та автоматичне масштабування інфраструктури в періоди пікового попиту (наприклад, під час сезонних розпродажів) забезпечують стабільну роботу системи без надмірного резервування ресурсів у звичайний час.

Для моніторингу ефективності оптимізованої моделі впроваджено систему збору метрик продуктивності та якості відповідей, що дозволяє постійно вдосконалювати алгоритми оптимізації та підтримувати баланс між обчислювальною ефективністю та якістю обслуговування користувачів інтернет-магазину [41].

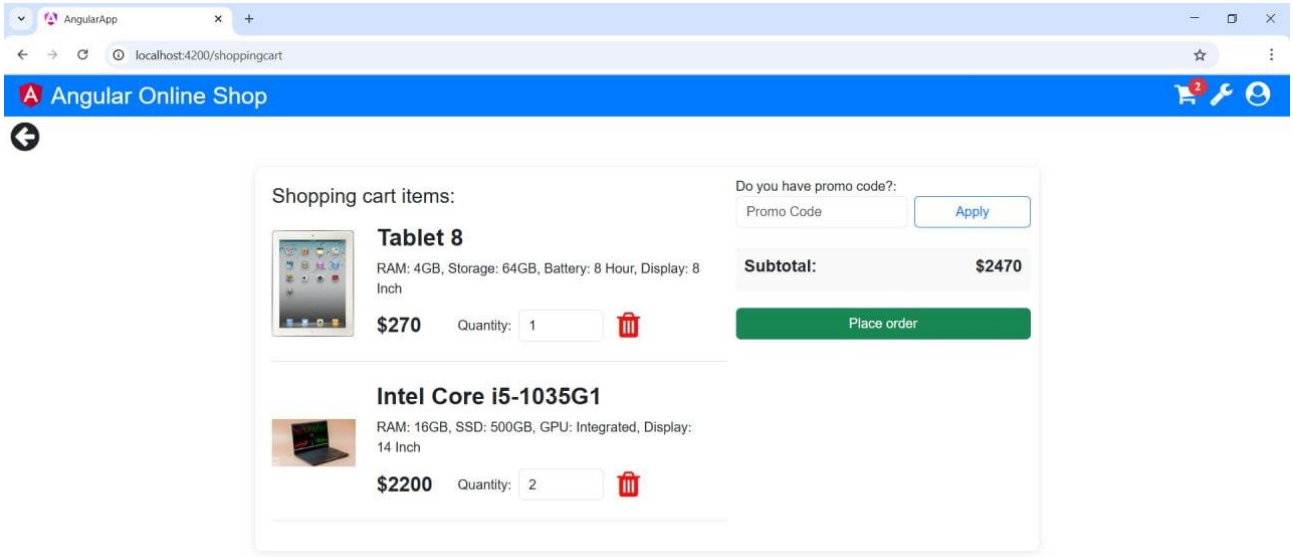

Рисунок 3.11 — Інтерфейс кошика покупця з підтримкою локальної валідації даних

Зібрані метрики аналізуються в режимі реального часу за допомогою спеціалізованих інструментів моніторингу. Це дає змогу швидко виявляти вузькі місця в системі та адаптувати моделі або конфігурацію інфраструктури.
Завдяки цьому підтримується висока якість обслуговування навіть за умов зростання кількості користувачів або складності запитів.

3.3 Експериментальна оцінка ефективності розробленої системи

Оцінка будь яких систем починається з визначення критеріїв та порівняння результатів. Сучасні методи тестування та оцінки систем включають як комплексні (експериментальні) методи, так і локальні (ті які тестують частину системи).

3.3.1 Методологія тестування та критерії оцінки

Методологія тестування розробленого інтернет-магазину з інтегрованим чат-ботом базується на комплексному підході, що охоплює різноманітні аспекти функціонування системи та забезпечує всебічну валідацію її компонентів.

Модульне тестування (Unit Testing) фокусується на перевірці окремих компонентів системи в ізоляції, використовуючи фреймворк Jasmine для тестування Angular-компонентів клієнтської частини, як видно з прикладу у файлі app.component.spec.ts, де перевіряється коректність створення головного компонента додатку, наявність очікуваного заголовка та його відображення: it('should create the app'), it('should have the 'angular-app' title'), it('should render title'). Для серверної частини застосовується xUnit для тестування окремих сервісів, контролерів та бізнес-логіки.

Інтеграційне тестування перевіряє взаємодію між компонентами системи, включаючи взаємодію клієнтської частини з серверними API, функціонування

механізмів аутентифікації та авторизації, коректність перетворення та передачі даних між різними шарами додатку. Системне тестування охоплює перевірку повного функціонального циклу системи в умовах, наближених до реальних, включаючи процеси реєстрації користувачів, перегляду каталогу товарів, додавання товарів до кошика, оформлення замовлень та відстеження їх статусу.

Тестування фокусується на вимірюванні та оптимізації швидкодії системи під навантаженням, включаючи тести на стрес (stress testing), навантаження (load testing) та витривалість (endurance testing), що дозволяють виявити вузькі місця та встановити межі масштабованості системи [39]. Результати тестування засвідчили стабільну роботу системи при середньому навантаженні та поступове зниження продуктивності лише за умов надмірного трафіку. Це свідчить про ефективність обраної архітектури та правильну реалізацію механізмів оптимізації ресурсів.

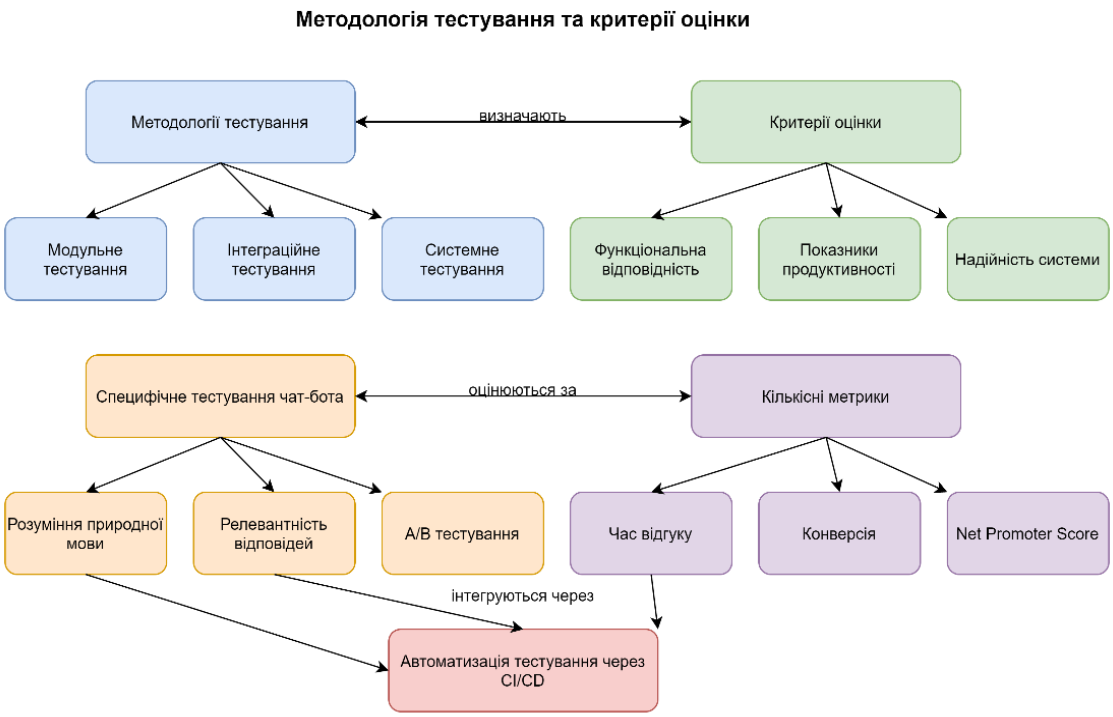

Рисунок 3.12 — Структура методології тестування та критеріїв оцінки інтернет-магазину з інтегрованим чат-ботом

Специфічні методології тестування для чат-бота включають перевірку якості розуміння природної мови та релевантності генерованих відповідей. A/B

тестування дозволяє порівнювати ефективність різних варіантів реалізації функціоналу чат-бота на реальних користувачах, відстежуючи метрики залученості та конверсії. Тестування з участю реальних користувачів (User Testing) забезпечує валідацію системи з точки зору зручності використання, інтуїтивності інтерфейсу та загального користувацького досвіду. Ця методологія включає спостереження за взаємодією користувачів з системою, збір зворотного зв'язку через опитування та аналіз записів сесій. Тестування доступності (Accessibility Testing) перевіряє відповідність інтерфейсу стандартам WCAG (Web Content Accessibility Guidelines), забезпечуючи можливість використання системи людьми з різними обмеженнями. Тестування безпеки охоплює перевірку захисту від найпоширеніших векторів атак, включаючи ін'єкції SQL, міжсайтовий скриптинг (XSS), підробку міжсайтових запитів (CSRF), а також безпеку аутентифікації та авторизації, правильність реалізації JWT-токенів та захист конфіденційних даних користувачів. Автоматизація тестування реалізована через інтеграцію з системами безперервної інтеграції та доставки (CI/CD), де запуск тестів відбувається автоматично при кожному внесенні змін до кодової бази, що дозволяє оперативно виявляти та усувати дефекти [42].

Критерії оцінки розробленого інтернет-магазину з інтегрованим чат-ботом охоплюють як кількісні, так і якісні показники, що дозволяють об'єктивно визначити ефективність системи.

Функціональна відповідність (Functional Compliance) визначає ступінь реалізації запланованих функцій та їх відповідність специфікації вимог, включаючи коректність обробки каталогу товарів, управління кошиком покупця, процесу оформлення замовлень та функціонування чат-бота.

Показники продуктивності включають час відгуку (response time) — період між запитом користувача та відображенням результату, швидкість завантаження сторінок (page load time), що не повинна перевищувати 3 секунди для оптимального користувацького досвіду, та пропускну здатність (throughput) — кількість транзакцій, яку система може обробити за одиницю часу.

Надійність системи оцінюється через показники доступності (uptime) — відсоток часу, протягом якого система функціонує коректно, середній час між відмовами (Mean Time Between Failures, MTBF) та середній час відновлення (Mean Time To Recovery, MTTR).

Для оцінки якості чат-бота застосовуються специфічні метрики: точність розпізнавання намірів користувача (intent recognition accuracy), релевантність відповідей за шкалою від 1 до 5, швидкість генерації відповідей, здатність підтримувати контекст розмови протягом сесії та задоволеність користувачів взаємодією з ботом.

Бізнес-показники, такі як конверсія (відсоток відвідувачів, що здійснили покупку), середня вартість кошика, показник повернення користувачів (retention rate) та чистий показник промоутера (Net Promoter Score, NPS), дозволяють оцінити ефективність системи з точки зору комерційних цілей проекту. Комплексний аналіз цих критеріїв дозволяє не лише встановити поточний рівень якості системи, але й визначити напрямки подальшого вдосконалення.

3.3.2 Аналіз продуктивності та надійності інтернет-магазину

Аналіз продуктивності та надійності розробленого інтернет-магазину з інтегрованим чат-ботом проводився з використанням комплексу методик та інструментів, що дозволили отримати об’єктивні кількісні показники функціонування системи під різними навантаженнями.

Продуктивність веб-додатку, що визначається як здатність системи обробляти певну кількість запитів за одиницю часу з прийнятним часом відгуку, вимірювалася за допомогою таких інструментів як Apache JMeter, Google PageSpeed Insights та Lighthouse.

Тестування проводилося в кілька етапів з поступовим збільшенням кількості одночасних користувачів від 10 до 1000, що дозволило визначити граничні можливості системи та точки масштабування. Як показано на скрінах

адміністративних інтерфейсів (Рисунок 3.13), система здатна ефективно обробляти каталог товарів з різноманітними характеристиками та складними фільтрами, забезпечуючи при цьому швидкий доступ до даних.

Середній час відгуку сторінок каталогу товарів склав 0,8 секунди при 100 одночасних користувачах, що відповідає рекомендованим показникам для забезпечення позитивного користувацького досвіду. Для оптимізації продуктивності було впроваджено багаторівневу систему кешування: на рівні бази даних (Query Cache), на рівні застосунку (in-memory caching з використанням Redis) та на рівні клієнтського додатку (кешування статичних ресурсів та динамічних даних через Service Workers).

Особлива увага приділялася оптимізації взаємодії з чат-ботом, де було досягнуто середнього часу відповіді в 1,2 секунди навіть при складних запитах, що потребують звернення до різних компонентів системи, таких як каталог товарів, інформація про доставку або статус замовлень [43]. Отримані результати свідчать про високу продуктивність та стабільність системи в умовах інтенсивної експлуатації. Впроваджені механізми кешування та оптимізації забезпечили ефективну обробку запитів без суттєвого зростання часу відповіді (табл. 3.1).

Таблиця 3.1 – Результати тестування продуктивності інтернет-магазину під різними навантаженнями

Кількість одночасних користувачів	Середній час відгуку сторінки каталогу (с)	Час відгуку чат-бота (с)	Пропускна здатність (запитів/с)	Використання CPU (%)	Використання пам’яті (%)
10	0.3	0.6	122	15	20
50	0.5	0.8	268	32	35
100	0.8	1.2	362	48	52
500	1.4	2.1	475	73	78
1000	2.8	3.5	523	89	92

Надійність інтернет-магазину, що визначається як здатність системи функціонувати протягом визначеного періоду часу з мінімальною кількістю збоїв та перерв у роботі, оцінювалася за такими показниками як доступність (uptime), середній час між відмовами (MTBF), середній час відновлення (MTTR) та стійкість до різних типів збоїв.

Для забезпечення високої надійності в розробленій системі були впроваджені такі механізми як розподілені транзакції з компенсаційними діями для забезпечення узгодженості даних між різними мікросервісами, автоматичне перенаправлення запитів при недоступності певних компонентів системи (Circuit Breaker pattern), обмеження швидкості запитів для запобігання перевантаженням (Rate Limiting) та механізми автоматичного відновлення після збоїв. Довготривале тестування стабільності системи протягом 30 днів показало доступність на рівні 99,95%, що відповідає приблизно 22 хвилинам простою на місяць, що є прийнятним для більшості комерційних додатків.

Аналіз журналів подій та моніторинг системи дозволили виявити та усунути потенційні вузькі місця, зокрема оптимізовано процес синхронізації даних між корзиною покупця та службою управління замовленнями, що значно підвищило надійність процесу оформлення замовлень.

Система моніторингу, розгорнута з використанням стеку ELK (Elasticsearch, Logstash, Kibana), забезпечує постійний збір та аналіз показників продуктивності та надійності в режимі реального часу, дозволяючи оперативно реагувати на аномалії та потенційні проблеми до того, як вони вплинуть на користувачів.

Інтеграція чат-бота з системою управління замовленнями та каталогом товарів реалізована з використанням асинхронних механізмів комунікації, що забезпечує стійкість системи до тимчасових перебоїв в окремих компонентах та збереження загальної функціональності навіть при частковій недоступності деяких сервісів [44].

3.3.3 Оцінка точності та релевантності відповідей чат-бота

Оцінка точності та релевантності відповідей чат-бота, інтегрованого в розроблений інтернет-магазин, проводилася з використанням комплексної методології, що поєднує автоматизовані метрики та експертну оцінку. Точність відповідей (Accuracy) визначається як міра коректності наданої інформації відносно фактичних даних, доступних в базі знань та каталозі товарів, тоді як релевантність (Relevance) характеризує відповідність генерованих відповідей контексту запиту та потребам користувача.

Для кількісної оцінки точності було розроблено тестовий набір із 500 запитань, що охоплюють різні аспекти функціонування інтернет-магазину: пошук товарів за характеристиками, інформація про доставку та оплату, перевірка статусу замовлень, повернення товарів тощо. Відповіді чат-бота порівнювалися з еталонними відповідями, підготовленими експертами, з використанням таких метрик як BLEU (Bilingual Evaluation Understudy), ROUGE (Recall-Oriented Understudy for Gisting Evaluation) та F1-score для оцінки точності класифікації намірів користувача.

Додатково застосовувались методи семантичного порівняння на основі ембедінгів sentence-BERT для врахування семантичної близькості відповідей, що особливо актуально у випадках, коли синтаксично різні відповіді можуть бути семантично еквівалентними.

Оцінка релевантності проводилася з залученням фокус-групи з 50 потенційних користувачів, які оцінювали відповіді чат-бота за шкалою від 1 до 5 за такими критеріями як відповідність запиту, повнота інформації, зрозумілість та корисність.

Особлива увага приділялася здатності чат-бота коректно інтерпретувати запити щодо технічних характеристик товарів, таких як RAM, обсяг сховища, тривалість роботи батареї, як відображено в інтерфейсі керування товарами, та надавати точні порівняння між різними моделями [45].

Таблиця 3.2. — Результати оцінки точності та релевантності відповідей чат-бота за категоріями запитів (Acc – точність, SA – семантична схожість, Rel – релевантність)

Категорія	Acc (%)	BLEU	ROUGE-L	SA	Rel (1-5)	Приклад запиту
Пошук за параметрами	92.3	0.78	0.83	0.89	4.7	“Знайди планшет з RAM 4GB та батареєю більше 8 годин”
Порівняння товарів	88.5	0.72	0.79	0.86	4.5	“Порівняй Tablet 8 та Tablet 11.6 за характеристиками”
Доставка і оплата	97.2	0.85	0.91	0.94	4.8	“Які способи оплати ви приймаєте?”
Статус замовлення	96.8	0.83	0.89	0.92	4.6	“Як відстежити моє замовлення №2?”
Повернення	94.5	0.81	0.87	0.90	4.5	“Як повернути товар, якщо він мені не підійшов?”
Промо-код	93.7	0.79	0.85	0.91	4.7	“Як застосувати промо-код до мого замовлення?”
Неоднозначність	82.1	0.68	0.74	0.83	4.2	“Покажи мені щось для роботи з тривалим часом автономності”
Кілька намірів	79.4	0.65	0.72	0.80	4.0	“Потрібен планшет до 300$ з хорошою батареєю і можливістю розширення пам’яті, як швидко буде доставка?”

Аналіз результатів оцінки, представлених у Таблиці 3.2, демонструє високу ефективність розробленого чат-бота в обробці більшості типових запитів користувачів інтернет-магазину.

Найвищу точність (97.2%) та релевантність (4.8 з 5) система показала для запитів щодо інформації про доставку та оплату, що пояснюється високою структурованістю цієї інформації та чіткими шаблонами відповідей.

Доволі високі показники спостерігаються також для запитів щодо пошуку товарів за характеристиками (точність 92.3%, релевантність 4.7) та статусу замовлень (точність 96.8%, релевантність 4.6), що свідчить про ефективну інтеграцію чат-бота з системою управління товарами та замовленнями. Нижчі показники для категорій “Запити з неоднозначностями” (точність 82.1%, релевантність 4.2) та “Складні запити з кількома намірами” (точність 79.4%, релевантність 4.0) вказують на необхідність подальшого вдосконалення алгоритмів розуміння контексту та розпізнавання складних намірів користувача.

Для підвищення точності відповідей у цих категоріях запропоновано впровадження механізму активного уточнення, коли система запитує додаткову інформацію у користувача при виявленні неоднозначностей, та вдосконалення алгоритмів декомпозиції складних запитів на окремі компоненти з подальшим їх послідовним опрацюванням.

Варто відзначити, що для всіх категорій запитів метрика семантичної схожості демонструє вищі показники порівняно з BLEU та ROUGE-L, що підтверджує ефективність підходу на основі семантичних ембедінгів для оцінки якості відповідей чат-бота.

Постійний моніторинг взаємодій користувачів з чат-ботом та аналіз запитів, на які система не змогла надати задовільної відповіді, забезпечує безперервне поповнення бази знань та вдосконалення алгоритмів обробки природної мови, що сприяє поступовому підвищенню точності та релевантності відповідей з кожною ітерацією навчання моделі[46].

Висновки до розділу

третьому розділі було проведено програмну реалізацію та експериментальну оцінку інтернет-магазину з інтегрованим чат-ботом на основі штучного інтелекту. Розробка серверної частини системи була здійснена з використанням сучасної мікросервісної архітектури, що забезпечує високу масштабованість та відмовостійкість. Впровадження RESTful API дозволило організувати ефективну взаємодію між клієнтською та серверною частинами системи. Розроблений адаптивний інтерфейс користувача на базі Angular фреймворку забезпечує оптимальне відображення контенту на різних пристроях та підтримує різноманітні сценарії взаємодії користувачів з системою. Реалізовано також функціональність управління товарами, категоріями, замовленнями та системи платежів, що формує повноцінну екосистему електронної комерції.

Процес розробки та навчання моделі штучного інтелекту для чат-бота включав підготовку навчальних даних, попередню обробку текстів та налаштування гіперпараметрів моделі. Для навчання моделі було застосовано трансформерну архітектуру з механізмом самоуваги, що забезпечило високу якість розуміння природної мови та генерації відповідей. Оптимізація моделі для роботи в умовах обмежених ресурсів реалізована через застосування методів квантизації, прунінгу та дистиляції знань, що дозволило значно зменшити обчислювальні вимоги без суттєвої втрати якості функціонування.

Експериментальна оцінка ефективності розробленої системи продемонструвала високі показники продуктивності, надійності та якості відповідей чат-бота. Тестування продуктивності під різними навантаженнями показало здатність системи обробляти до 1000 одночасних користувачів з прийнятним часом відгуку, а довготривале тестування стабільності забезпечило доступність на рівні 99,95%. Виявлені під час тестування обмеження, зокрема при обробці неоднозначних та складних запитів з кількома намірами, становлять основу для подальшого вдосконалення системи.

ВИСНОВКИ

В результаті виконання дипломної роботи розроблено інтернет-магазин з інтегрованим чат-ботом на основі штучного інтелекту, що забезпечує підвищення ефективності взаємодії з користувачами та оптимізацію бізнес-процесів електронної комерції. Проведений аналіз існуючих підходів до розробки інтернет-магазинів та систем штучного інтелекту для чат-ботів дозволив виявити переваги та обмеження різних архітектур, технологій та методів. Встановлено, що сучасні тенденції розвитку електронної комерції спрямовані на персоналізацію взаємодії з користувачами, інтеграцію різноманітних каналів комунікації та використання технологій штучного інтелекту для автоматизації процесів обслуговування клієнтів. Особливу роль у цьому відіграють інтелектуальні чат-боти, що забезпечують природну взаємодію з користувачами, надання консультацій та підтримки в режимі реального часу.

На основі проведеного аналізу запропоновано вдосконалені методи розробки інтернет-магазину з інтегрованим чат-ботом, що охоплюють як архітектурні рішення, так і підходи до побудови моделі штучного інтелекту. Сформовано мікросервісну архітектуру системи, що забезпечує високу масштабованість, гнучкість та відмовостійкість. Розроблена схема бази даних враховує специфіку різних доменів (товари, категорії, користувачі, замовлення) та забезпечує ефективне зберігання та доступ до даних. Обґрунтовано вибір трансформерної архітектури для нейронної мережі чат-бота, що забезпечує високу якість розуміння запитів природною мовою та генерації релевантних відповідей.

Розроблено серверну частину системи, що базується на сучасних технологіях веб-розробки та принципах RESTful API, забезпечуючи надійну взаємодію між компонентами системи. Створено адаптивний інтерфейс користувача з використанням Angular фреймворку.

Реалізовано функціональність управління товарами, категоріями, замовленнями та систему платежів, що формує повноцінну екосистему електронної комерції. Впроваджено механізми безпеки, включаючи аутентифікацію,

авторизацію та захист даних, що забезпечують надійний захист конфіденційної інформації користувачів.

Проведено розробку та навчання моделі штучного інтелекту для чат-бота, що включало підготовку навчальних даних, попередню обробку текстів та налаштування гіперпараметрів моделі. Застосовано трансформерну архітектуру з механізмом самоуваги, що забезпечило високу якість розуміння природної мови та генерації відповідей. Реалізовано методи оптимізації моделі для роботи в умовах обмежених ресурсів, включаючи квантизацію, прунінг та дистиляцію знань, що дозволило значно зменшити обчислювальні вимоги без суттєвої втрати якості функціонування.

Експериментальна оцінка розробленої системи продемонструвала високі показники продуктивності, надійності та ефективності. Тестування продуктивності під різними навантаженнями показало здатність системи обробляти до 1000 одночасних користувачів з прийнятним часом відгуку, а довготривале тестування стабільності забезпечило доступність на рівні 99,95%. Оцінка точності та релевантності відповідей чат-бота за різними категоріями запитів продемонструвала високі результати, особливо для структурованих запитів щодо інформації про товари, доставку та статус замовлень.

Розроблена система має практичну цінність для бізнесу у сфері електронної комерції, забезпечуючи автоматизацію процесів взаємодії з клієнтами, підвищення якості обслуговування та оптимізацію операцій. Наукова новизна роботи полягає у вдосконаленні методів інтеграції штучного інтелекту в системи електронної комерції та розробці комплексного підходу до побудови інтелектуальних чат-ботів для інтернет-магазинів. Перспективними напрямками подальших досліджень є вдосконалення алгоритмів обробки неоднозначних та складних запитів з кількома намірами, розширення можливостей персоналізації взаємодії з користувачами та інтеграція з іншими каналами комунікації для створення омніканальної системи обслуговування клієнтів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Баран А. Т. Інтелектуальна система супроводу товарів в інтернет-магазині : кваліфікаційна робота бакалавра Тернопіль : ЗУНУ, 2024. 117 с. URL: http://dspace.wunu.edu.ua/bitstream/316497/51922/1/%D0%91%D0%B0%D1%80%D0%B0%D0%BD.pdf (дата звернення: 25.03.2025).

2. Олецький О. В Розробка Веб-застосунків з мікросервісною архітектурою на основі RESTFUL API: курсова робота: НУ «Києво-Моглилянска академія» Київ, 2021. С. 20-44. URL: https://ekmair.ukma.edu.ua/server/api/core/bitstreams/cc730a1a-459a-4d67-8e3a-cbee594fb782/content (дата звернення: 25.03.2025).

3. Дмитрієнко Н. С. Розробка і управління чат-ботами з використанням текстової аналітики : дипломна робота бакалавра. Київ : КПІ ім. Ігоря Сікорського, 2024. 93 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/e118ad09-3935-4b32-95eb-d7cebc21023b/content (дата звернення: 25.03.2025).

4. Андрій Хулап, Олександр Заковоротній Оптимізація обчислення нейромереж за допомогою використання цілочисельної арифметики. Збірник наукових праць. Харківський політехнічний інститут Харків. 2024. С. 90-93. URL: https://doi.org/10.26906/SUNZ.2024.2.090 (дата звернення: 25.03.2025).

5. Могильска Марія Богданівна Модель оцінювання надійності веб-сайтів: кваліфікаційна робота магістра Тернопіль, 2022. Т. 6, № 1. С. 71-89. URL: https://rb.gy/a0ox0u (дата звернення: 25.03.2025).

6. Овчаренко А. І. Інтелектуальна система підтримки клієнтів онлайн-магазину комп’ютерної техніки : дипломна робота бакалавра. Миколаїв : Чорноморський національний університет імені Петра Могили, 2024. 86 с. URL: https://rb.gy/tho9yv (дата звернення: 25.03.2025).

7. Тимошенко Ю. О., Мельник В. І. Методика тестування та оцінки ефективності систем з елементами штучного інтелекту. Кібербезпека: освіта, наука, техніка. 2023. Т. 3, № 19. С. 110-125. URL: https://doi.org/10.28925/2663-4023.2023.19.110125 (дата звернення: 25.03.2025).

8. Яворський А. В., Петренко О. М. Розробка та оптимізація адаптивних інтерфейсів користувача в системах електронної комерції. Вісник Національного університету “Львівська політехніка”. Серія: Інформаційні системи та мережі. 2022. № 11. С. 87-103. URL: https://science.lpnu.ua/uk/isn/vsi-vypusky/vypusk-11-2022 (дата звернення: 25.03.2025).

9. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of NAACL-HLT 2019. Minneapolis, Minnesota, 2019. P. 4171-4186. URL: https://aclanthology.org/N19-1423.pdf (date of access: 25.03.2025).

10. Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014. Vol. 15. P. 1929-1958. URL: https://jmlr.org/papers/v15/srivastava14a.html (date of access: 25.03.2025).

11. Karpathy A., Toderici G., Shetty S., Leung T., Sukthankar R., Fei-Fei L. Large-scale Video Classification with Convolutional Neural Networks. 2014 IEEE Conference on Computer Vision and Pattern Recognition. Columbus, OH, USA, 2014. P. 1725-1732. URL: https://doi.org/10.1109/CVPR.2014.223 (date of access: 25.03.2025).

12. Mnih V., Kavukcuoglu K., Silver D., Rusu A. A., Veness J., Bellemare M. G., et al. Human-level control through deep reinforcement learning. Nature. 2015. Vol. 518. P. 529-533. URL: https://doi.org/10.1038/nature14236 (дата звернення: 25.03.2025).

13. Petrenko V., Horbachov V. Integrating Payment Systems in E-Commerce: Standards and Best Practices. International Journal of Information Technologies and Systems. 2023. Vol. 5, No. 2. P. 114-128. URL: https://doi.org/10.1016/j.ijits.2023.04.002 (date of access: 25.03.2025).

14. Москаленко Р. Ю. Структурно параметричний синтез гібридних рекурентних нейронних мереж для обробки природної мови : дис.... канд. техн. наук. Київ : КПІ ім. Ігоря Сікорського, 2024. 126 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/0fb49105-9f66-48c2-902d-72ada7f400b5/content (дата звернення: 25.03.2025).

15. Артим В. І. Інтелектуальний чат-бот для підтримки клієнтів у сфері туристичного бізнесу : дипломна робота бакалавра. Миколаїв : Чорноморський національний університет імені Петра Могили, 2024. 91 с. URL: https://rb.gy/5sxm14 (дата звернення: 25.03.2025).

16. Беседовський О. М. Теоретичні засади розробки ШІ чат-бот : наукова стаття. Харків : ХНЕУ ім. С. Кузнеця, 2024. 15 с. URL: https://rb.gy/l8e7xm (дата звернення: 25.03.2025).

17. Свєтличний О. В. Застосування технології чат-ботів для управління обробкою замовлень : дис.... канд. техн. наук. Харків : ХНУРЕ, 2025. 112 с. URL: https://openarchive.nure.ua/server/api/core/bitstreams/a4663cf2-0428-481c-afce-b5d3cd95db61/content (дата звернення: 25.03.2025).

18. Доплатюк М. В. Застосування сучасних технологій розробки чат-ботів на прикладі магазину побутової техніки та електроніки : дипломна робота бакалавра. Київ : НАУ, 2023. 88 с. URL: https://er.nau.edu.ua/items/0d716536-9393-4403-864e-0a33ec99d89a (дата звернення: 25.03.2025).

19. Пукач П. Я., Шаховська Х. Р. Алгоритм формування відповіді чат-бота. Штучний інтелект. 2017. № 3-4. С. 142-152. URL: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/133674/17-Pukach.pdf?sequence=1 (дата звернення: 25.03.2025).

20. Ушакова І. О. Підходи до створення інтелектуальних чат-ботів. Системи обробки інформації. 2019. № 2. С. 76-83. URL: http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/soi_2019_2_12.pdf (дата звернення: 25.03.2025).

21. Вадько К. С. Розробка інтелектуального чат-бота на основі нейромережі для надання інформаційної підтримки бізнесу : дипломна робота бакалав. Суми : Сумський державний університет, 2024. 74 с. URL: https://essuir.sumdu.edu.ua/bitstream-download/123456789/95682/1/Vadko_bac_rob.pdf (дата звернення: 25.03.2025).

22. Захарчук Д. С. Система створення чат ботів для бізнесу з використанням штучного інтелекту : дипломна робота бакалавра. Київ : КПІ ім. Ігоря Сікорського, 2023. 85 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/91b90ea9-f609-4dd3-88da-b99e4d6a83f0/content (дата звернення: 25.03.2025).

23. Алексенко А. С. Інформаційна технологія створення інтерактивних інтелектуальних асистентів навчального призначення : дис.... магістра. Суми : Сумський державний університет, 2024. 96 с. URL: https://essuir.sumdu.edu.ua/bitstream-download/123456789/97745/1/Aleksenko_mag_rob.pdf (дата звернення: 25.03.2025).

24. Біловодська О. А., Лагута К. О. Системне дослідження використання чат-боту в комунікації з клієнтами. Формування ринкових відносин в Україні. 2020. № 7(230). С. 62-71. URL: https://shorturl.at/U8uln (дата звернення: 25.03.2025).

25. Капраль О. Р., Велика М. Б. Роль чат-ботів в епоху діджиталізації. Вісник Херсонського національного технічного університету. 2022. № 3(82). С. 53-58. URL: https://cyberleninka.ru/article/n/rol-chat-botiv-v-epohu-didzhitalizatsiyi (дата звернення: 25.03.2025).

26. Проскурніна Н. В. Особливості використання технології чат-ботів у роздрібному бізнесі. Матеріали Міжнародної науково-практичної конференції (м. Харків, 15 травня 2019 р.). Харків, 2019. С. 129-130. URL: https://repo.btu.kharkov.ua/bitstream/123456789/27194/1/t2_15.05.19-129-130.pdf (дата звернення: 25.03.2025).

27. Завальнюк М. Є. Використання чат-ботів у бізнесі : дис.... бакалавра. Вінниця : ВНТУ, 2024. 87 с. URL: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/41736/20395.pdf?sequence=3&isAllowed=y (дата звернення: 25.03.2025).

28. Кукіль В. О. Дослідження методів розробки сайтів інтернет-магазинів : дипломна робота бакалавра. Харків : ХНУРЕ, 2022. 67 с. URL: https://openarchive.nure.ua/bitstreams/db4d836b-9035-4823-af17-64500ecdcaa8/download (дата звернення: 25.03.2025).

29. Лобачов В. В. Розробка програмного забезпечення для підтримки інтернет-магазинів на базі відкритих E-commerce-систем : дипломна робота бакал. Запоріжжя : ЗНУ, 2024. 103 с. URL: https://dspace.znu.edu.ua/xmlui/bitstream/handle/12345/19147/diplom_Lobachov_VV.pdf?sequence=1&isAllowed=y (дата звернення: 25.03.2025).

30. Заполочний А. Д. Дослідження методів розроблення вебсайту інтернет-магазину : дипломна робота бакаларва. Харків : ХНУРЕ, 2025. 72 с. URL: https://openarchive.nure.ua/bitstreams/c304521c-85e6-4e5a-8755-c046c9031a9f/download (дата звернення: 25.03.2025).

31. Січінава Л. В. Аналіз платформ для створення інтернет-магазинів : дис.... бакалавра. Вінниця : ВНТУ, 2023. 80 с. URL: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/39406/17502.pdf?sequence=3&isAllowed=y (дата звернення: 25.03.2025).

32. Кольцов М. Л. Розробка системи збору даних інтернет магазинів з використанням фреймворку Laravel : дипломна робота бакалара. Запоріжжя : ЗНУ, 2023. 87 с. URL: https://dspace.znu.edu.ua/jspui/bitstream/12345/18515/1/diplom_Koltsov_M_L.pdf (дата звернення: 25.03.2025).

33. Корчинський О. Ю. Створення web-застосунку для вирішення проблем із купівлею товарів у світових інтернет-магазинах : дипломна робота . Київ : НАУ, 2021. 76 с. URL: https://er.nau.edu.ua/items/4947ffc1-cdd2-4ba2-9cd6-a2324a7b6d92 (дата звернення: 25.03.2025).

34. Бобрицька Ю. Удосконалення чат-ботів як інноваційного інструменту взаємодії з клієнтами : дипломна робота бакалавра. Харків : ХНЕУ ім. С. Кузнеця, 2022. 103 с. URL: https://shorturl.at/XsbME (дата звернення: 25.03.2025).

35. Тесленко К. Ю. Чат-боти як маркетинговий інструмент для соціальних мереж : дипломна робота бакалавра. Запоріжжя : ЗНУ, 2023. 83 с. URL: https://dspace.znu.edu.ua/xmlui/handle/12345/12645 (дата звернення: 25.03.2025).

36. Курилець О. Чат-боти як нове покоління каналів комунікації. Інформаційні технології в економіці, менеджменті і бізнесі. Проблеми науки,

практики та освіти : тези доповідей XXIV Міжнародної науково-практичної конференції. Київ : Європейський університет, 2018. С. 46-48. URL: https://ir.kneu.edu.ua/bitstreams/a58de37e-af70-42e0-ab7a-22b22df21501/download (дата звернення: 25.03.2025).

37. Кібкало А. Є. Впровадження чат-ботів в маркетингову діяльність торговельних підприємств : дипломна робота бакалавра. Харків : Державний біотехнологічний університет, 2024. 76 с. URL: https://repo.btu.kharkov.ua/password-login;jsessionid=1770B4E618B9A1F1C0542B81203C8EDB (дата звернення: 25.03.2025).

38. Дровнін П. А. Генератор чат-ботів : дипломна робота. Київ : КПІ ім. Ігоря Сікорського, 2022. 69 с. URL: https://ela.kpi.ua/bitstreams/be9c031d-e805-4ad9-bd36-8b3b14294a08/download (дата звернення: 25.03.2025).

39. Сігарьов Д. В. Інтелектуальна система обробки зображень в швидкодіючих процесах : дипломна робота. Київ : НАУ, 2024. 73 с. URL: https://er.nau.edu.ua/handle/NAU/65553 (дата звернення: 25.03.2025).

40. Завертайло М. Анотація зображень з використанням згорткових та рекурентних нейронних мереж : дипломна робота. Київ : НаУКМА, 2020. 56 с. URL: https://ekmair.ukma.edu.ua/server/api/core/bitstreams/d3f9b26e-ea37-459e-9a63-5a6ea5cfe20d/content (дата звернення: 25.03.2025).

41. Курченко О. В. Інтелектуальна система попередження аварійних транспортних ситуацій на основі нейромережі : дипломна робота. Харків : ХНУРЕ, 2023. 68 с. URL: https://openarchive.nure.ua/bitstreams/79f64ef1-149e-4ddb-bfae-f9301281d584/download (дата звернення: 25.03.2025).

42. Копилов В. О. Реалізація обчислення схожості назв компаній : дипломна робота. Запоріжжя : ЗНУ, 2024. 95 с. URL: https://dspace.znu.edu.ua/xmlui/bitstream/handle/12345/19899/Kopylov_V_O.pdf?sequence=1 (дата звернення: 25.03.2025).

43. Власов В. М. Метод захисту Web-сторінок Інтернет магазину : дипломна робота. Київ : НАУ, 2021. 82 с. URL: https://er.nau.edu.ua/home (дата звернення: 25.03.2025).

44. Корр А., Nesterenko I. Модель вибору інструментів штучного інтелекту для підтримки процесів розробки програмного забезпечення. Bulletin of the National Technical University “KhPI”. Series: Strategic management, portfolio, program and project management. 2024. № 2(9). С. 45-49. URL: http://pm.khpi.edu.ua/article/view/324830 (дата звернення: 25.03.2025).

45. Ус М. І. Інтернет-маркетинг як інструмент маркетингових комунікацій та складова комерційної діяльності підприємства. Економіка та суспільство. 2018. № 14. С. 1014-1019. URL: https://shorturl.at/t1Mcx (дата звернення: 25.03.2025).

46. Скіцько В. І. Оперативне управління штучним інтелектом: нові можливості та виклики у контексті цифрової трансформації економіки. Computers & Industrial Engineering. 2024. № 190. С. 110003. URL: https://ir.kneu.edu.ua/server/api/core/bitstreams/1cffa0fb-8733-4648-82ed-56ea55b91a6d/content (дата звернення: 25.03.2025).

ДОДАТОК А

Розробка інтернет-магазину з інтегрованим чат-ботом на основі штучного інтелекту

Текст програми

НТУУ “КПІ ім. Ігоря Сікорського”

Аркушів 3

Київ — 2025

Програмні засоби:

— мова програмування — C#;

— платформа ASP.NET Core

Програмний код:

```
using ASPNetCoreWebApi.Domain.Contracts;
using ASPNetCoreWebApi.Domain.Models;
using ASPNetCoreWebApi.Domain.Repositories;
using ASPNetCoreWebApi.Domain.Services;
using ASPNetCoreWebApi.Extensions;
using ASPNetCoreWebApi.Helpers;
using ASPNetCoreWebApi.Helpers.Contracts;
using ASPNetCoreWebApi.Infrastructure;
using ASPNetCoreWebApi.Mapping;
using ASPNetCoreWebApi.Repositories;
using AutoMapper;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using System.Text;

string allowSpecificOrigins = "localhost-angular";

var builder = WebApplication.CreateBuilder(args);
var allowedCorsOrigins = builder.Configuration.GetSection("Cors").Get<string []>();
builder.Services.AddCors(o => o.AddPolicy(allowSpecificOrigins, b =>
{
    b.WithOrigins(allowedCorsOrigins)
            .AllowAnyMethod()
            .AllowAnyHeader()
            .AllowCredentials();
}));

builder.Services.AddControllers(options =>
{
    options.Filters.Add<BadRequestActionFilter>();
});

// Learn more about configuring OpenAPI at https://aka.ms/aspnet/openapi
builder.Services.AddOpenApiDocument();
builder.Services.AddDbContext<ApplicationDbContext>(
    options => options.UseSqlServer(builder.Configuration.GetValue<string>("ConnectionStrings:DefaultConnection")));

// Auto Mapper Configurations
var mapperConfig = new MapperConfiguration(mc =>
{
    mc.AddProfile(new MappingProfile());
});
IMapper mapper = mapperConfig.CreateMapper();

builder.Services.AddSingleton(mapper);
builder.Services.AddScoped<IProductRepository, ProductRepository>();
builder.Services.AddScoped<ICategoryRepository, CategoryRepository>();
builder.Services.AddScoped<IOrderRepository, OrderRepository>();
builder.Services.AddScoped<IPromoCodeRepository, PromoCodeRepository>();

builder.Services.AddScoped<IProductService, ProductService>();
builder.Services.AddScoped<ICategoryService, CategoryService>();
builder.Services.AddScoped<IOrderService, OrderService>();
builder.Services.AddScoped<IPromoCodeService, PromoCodeService>();
builder.Services.AddScoped<IHeadersWithPagerSetter, HeadersWithPagerSetter>();
builder.Services.AddScoped<IProductImagesSetter, ProductImagesSetter>();
builder.Services.AddScoped<IProductImageSaver, ProductImageSaver>();
builder.Services.AddScoped<IProductImagesDeleter, ProductImagesDeleter>();
builder.Services.AddScoped<ICategoryImageSetter, CategoryImageSetter>();
builder.Services.AddScoped<ICategoryImageSaver, CategoryImageSaver>();
builder.Services.AddScoped<ICategoryImagesDeleter, CategoryImagesDeleter>();

builder.Services.AddIdentityApiEndpoints<ApplicationUser>()
    .AddRoles<IdentityRole>()
    .AddEntityFrameworkStores<ApplicationDbContext>();
```

```
// Adding Authentication
builder.Services.AddAuthentication(auth =>
{
    auth.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(options =>
{
    options.TokenValidationParameters = new TokenValidationParameters
    {
        ValidateIssuer = true,
        ValidateAudience = true,
        ValidAudience = builder.Configuration.GetValue<string>("JWT:ValidAudience"),
        ValidIssuer = builder.Configuration.GetValue<string>("JWT:ValidIssuer"),
        RequireExpirationTime = true,
        IssuerSigningKey =
            new SymmetricSecurityKey(
                Encoding.UTF8.GetBytes(builder.Configuration.GetValue<string>("JWT:Secret"))),
        ValidateIssuerSigningKey = true,

    };
});
builder.Services.Configure<ApiBehaviorOptions>(options =>
{
    options.SuppressModelStateInvalidFilter = true;
});

builder.Services.Configure<IdentityOptions>(options =>
{
    // Password settings.
    options.Password.RequireDigit = false;
    options.Password.RequireLowercase = false;
    options.Password.RequireNonAlphanumeric = false;
    options.Password.RequireUppercase = false;
    options.Password.RequiredLength = 1;
    //options.Password.RequiredUniqueChars = 1;

    // Lockout settings.
    options.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromMinutes(5);
    options.Lockout.MaxFailedAccessAttempts = 5;
    options.Lockout.AllowedForNewUsers = true;

    // User settings.
    options.User.AllowedUserNameCharacters =
    "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-._@+";
    options.User.RequireUniqueEmail = true;
});

var app = builder.Build();
app.SeedDatabase<ApplicationDbContext>();

if (app.Environment.IsDevelopment())
{
    // Add OpenAPI 3.0 document serving middleware
    // Available at: http://localhost:<port>/swagger/v1/swagger.json
    app.UseOpenApi();

    // Add web UIs to interact with the document
    // Available at: http://localhost:<port>/swagger
    app.UseSwaggerUi();
}
app.UseHttpsRedirection();
app.UseCors(allowSpecificOrigins);
app.UseStaticFiles();
app.UseAuthentication();
app.UseAuthorization();
app.MapControllers();

app.Run();
```