НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри комп'ютерних наук**
_____ **Ігор БОЛБОТ**	_____ **Белла ГОЛУБ**
(підпис)	(підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: Програмне забезпечення для веборієнтованої системи з продажу прикрас

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми
к.т.н., доцент _____ **Ганна ВАЙГАНГ**
(підпис)

Керівник бакалаврської кваліфікаційної роботи
к.ф.-м.н., доцент _____ **Віктор КИРИЧЕНКО**
(підпис)

Виконала _____ **Альбіна ЛИТОВКА**
(підпис)

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Белла ГОЛУБ
(підпис)

«___» _____ 2025 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ
Литовці Альбіні Сергіївні

_____
(прізвище, ім'я, по батькові)

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи «Програмне забезпечення для веборієнтованої системи з продажу прикрас»

затверджена наказом від «19» грудня 2025 р. № 3204 «С

Термін подання завершеної роботи на кафедру «20» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): фреймворк Django, мова програмування Python, система управління базами даних PostgreSQL, мови HTML, CSS, JavaScript, сервіс онлайн-оплати WayForPay, засоби контейнеризації Docker.

Перелік питань, що підлягають дослідженню:
Аналіз предметної області, Моделювання предметної області, Проєктування інформаційного забезпечення, Розробка програмного забезпечення, Впровадження та експлуатація системи

Дата видачі завдання «19» грудня 2025 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Віктор КИРИЧЕНКО**

Завдання взяла до виконання _____ (підпис) **Альбіна ЛИТОВКА**

ЗМІСТ

ВСТУП 5
1 ВЕБОРІЄНТОВАНА СИСТЕМА ПРОДАЖУ ПРИКРАС ЯК ОБ'ЄКТ ДОСЛІДЖЕННЯ 8
1.1 Аналіз онлайн-магазину прикрас як предметної області 8
1.2 Моделювання предметної області 9
1.2.1 Загальні відомості 9
1.2.2 Діаграма прецедентів 11
1.2.3 Діаграма послідовності 13
1.2.4 Діаграма діяльності 14
1.3 Огляд існуючих аналогічних систем 16
1.4 Постановка завдання 19
2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ 22
2.1 Логічна модель даних у вигляді ER-діаграми 22
2.2 Вибір системи управління базами даних 26
2.3 Розробка структури інформаційної бази 27
2.4 Схема та фізична структура бази даних 29
3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 32
3.1 Абстракція предметної області 32
3.2 Моделювання структури та взаємодії об'єктів 34
3.3 Організаційна структура програмного забезпечення 37
3.4 Компонентна структура системи 42
3.5 Вибір інструментарію для створення прикладного програмного забезпечення 45
3.6 Алгоритмізація та програмування програмних модулів 48
4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ 52
4.1 Тестування системи 52
4.2 Вимоги до апаратного та програмного забезпечення 56
4.3 Склад інсталяційного пакету 59
4.4 Аналіз отриманих результатів 61
ВИСНОВКИ 64
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 66
Додаток А 68
Додаток Б 69

Додаток В 71
Додаток Г 73
Додаток Д **75**

ВСТУП

У сучасному швидкозмінному середовищі електронної комерції все більшої актуальності набуває створення спеціалізованих веборієнтованих систем, які забезпечують не лише продаж товарів, а й персоналізовану взаємодію з користувачем. Одним із перспективних напрямів є розробка інтернет-магазинів прикрас, що поєднують функції електронного каталогу, обробки замовлень, онлайн-оплати, адміністрування товарів та персонального підбору продукції за визначеними характеристиками. Для бренду Astrobracelet це завдання є особливо важливим, оскільки асортимент пов’язаний не лише з естетичними властивостями виробів, а й з такими параметрами, як камінь, знак зодіаку, колір та символічний запит користувача.

Актуальність теми полягає в необхідності створення зручної, функціональної та сучасної вебсистеми, яка дає змогу автоматизувати процес представлення асортименту, пошуку товарів, оформлення замовлень, приймання платежів та адміністрування магазину. Додатковою перевагою такої системи є можливість реалізації персоналізованого підбору прикрас на основі індивідуальних потреб користувача, що підвищує якість обслуговування, покращує користувацький досвід і сприяє зростанню продажів.

Метою бакалаврської кваліфікаційної роботи є розробка програмного забезпечення для вебсистеми продажу прикрас Astrobracelet, яка забезпечує повний цикл роботи інтернет-магазину: перегляд каталогу, фільтрацію товарів, підбір прикрас за характеристиками, додавання товарів до кошика, оформлення замовлення, обробку платежів та адміністрування даних.

Для досягнення поставленої мети необхідно виконати такі завдання:

- провести аналіз предметної області онлайн-продажу прикрас;
- дослідити існуючі програмні рішення та визначити їх переваги і недоліки;

- сформулювати функціональні та нефункціональні вимоги до системи;
- спроєктувати структуру бази даних та зв’язки між основними сутностями;
- обрати інструменти та технології реалізації;
- розробити серверну частину системи на базі Django;
- реалізувати клієнтську частину з адаптивним інтерфейсом користувача;
- реалізувати модулі каталогу, кошика, оформлення замовлення та оплати;
- створити адміністративну панель для керування товарами, замовленнями та платежами;
- виконати тестування програмного забезпечення та оцінити результати функціонування системи.

Об’єктом дослідження є процес функціонування вебсистеми продажу прикрас.

Предметом дослідження є методи, моделі та програмні засоби розробки клієнт-серверного вебзастосунку для електронної комерції з використанням персоналізованих механізмів підбору товарів.

Методи та технології. Під час виконання роботи використовувалися методи аналізу предметної області, методи проєктування баз даних, принципи клієнт-серверної архітектури, засоби об’єктно-орієнтованого програмування, а також сучасні технології веброзробки. Програмне забезпечення реалізовано з використанням мови програмування Python, вебфреймворку Django, системи керування базами даних PostgreSQL, HTML, CSS, JavaScript, а також інструментів для адміністрування й тестування вебзастосунків.

Практичне значення одержаних результатів полягає в тому, що розроблена система може бути використана як основа для реального функціонування інтернет-магазину прикрас. Вона забезпечує структуроване зберігання даних про товари, категорії, камені, кольори, знаки зодіаку,

замовлення та платежі, а також дає змогу організувати зручну взаємодію між клієнтом і продавцем. Реалізований механізм фільтрації та персоналізованого підбору підвищує цінність системи для кінцевого користувача та вирізняє її серед типових інтернет-магазинів.

Апробація. Проблематика, актуальність та ціль даного проекту були викладені у тезах «Програмне забезпечення для веборієнтованої системи з продажу прикрас», які були у VII всеукраїнській науково-практичній конференції студентів і аспіратів «Теоретичні та приладні аспекти розробки комп'ютерних систем 2026».

Структура записки. Структурно кваліфікаційна робота складається зі вступу, основних розділів, висновків, списку використаних джерел та додатків. У першому розділі розглядається предметна область, аналізуються існуючі рішення та формулюється постановка задачі. У другому розділі описується інформаційне забезпечення системи та структура бази даних. Третій розділ присвячено розробці прикладного програмного забезпечення, архітектурі системи та реалізації основних модулів. У четвертому розділі наведено рекомендації щодо впровадження, експлуатації та тестування системи. У висновках подано узагальнення результатів виконаної роботи.

Ця структура забезпечує чітке уявлення про кожен етап розробки, реалізації та впровадження системи продажу прикрас та результативності продажів, дозволяючи ефективно організувати процес виконання проекту.

1 ВЕБОРІЄНТОВАНА СИСТЕМА ПРОДАЖУ ПРИКРАС ЯК ОБ'ЄКТ ДОСЛІДЖЕННЯ

1.1 Аналіз онлайн-магазину прикрас як предметної області

Сучасна електронна комерція охоплює значну кількість напрямів, серед яких окреме місце займає онлайн-продаж прикрас. На відміну від багатьох інших товарів, прикраси обираються не лише за ціною чи практичністю, а й за емоційним сприйняттям, символічним значенням, кольоровою гамою, матеріалом та особистими асоціаціями покупця. Саме тому предметна область продажу прикрас має власну специфіку та потребує більш гнучкого підходу до організації вебсистеми.

Для бренду Astrobracelet прикраси є не просто елементами оздоблення, а виробами, які поєднують естетику, індивідуальність та особистий сенс. У центрі уваги знаходяться браслети та інші прикраси, пов'язані з натуральними каменями, які можуть асоціюватися зі знаком зодіаку, внутрішнім станом людини, певним запитом або настроєм. Такий підхід формує особливий формат взаємодії між користувачем і товаром: покупець шукає не лише привабливу річ, а прикрасу, що найбільше відповідає його відчуттям і потребам.

Предметна область онлайн-магазину прикрас включає кілька основних складових. Насамперед це самі товари, представлені у вигляді прикрас певного типу. Окрім цього, важливу роль відіграють додаткові характеристики товарів, зокрема камені, кольори, знаки зодіаку, категорії та символічні запити користувачів. Окремою складовою є процес взаємодії користувача з магазином, який охоплює перегляд каталогу, вибір товару, фільтрацію, оформлення замовлення, оплату та отримання інформації про результат покупки. Також важливим елементом предметної області є адміністрування асортименту, замовлень і платежів.

Особливість цієї предметної області полягає в тому, що один і той самий камінь може підходити кільком знакам зодіаку, а один товар може одночасно відповідати кільком параметрам відбору. Наприклад, прикраса може належати до певної категорії, мати конкретний колір, містити один або кілька каменів і водночас підходити для кількох знаків зодіаку та кількох запитів користувача. Саме тому система повинна підтримувати складні зв’язки між сутностями та дозволяти гнучко організовувати пошук.

Ще однією важливою особливістю цієї предметної області є значення візуальної складової. Для покупця прикрас велике значення мають фотографії, композиція сторінки, стиль подачі товарів, кольори інтерфейсу та загальна атмосфера сайту. На відміну від технічних або побутових товарів, прикраси часто купуються під впливом емоційного враження, тому вебсистема має не лише надавати інформацію, а й формувати довіру до бренду, створювати відчуття цілісності та підкреслювати індивідуальність товару.

Таким чином, предметна область вебсистеми продажу прикрас Astrobracelet охоплює не лише класичні процеси електронної комерції, а й механізми персоналізованого підбору товарів, що робить систему більш складною, але водночас більш цінною для кінцевого користувача.

1.2 Моделювання предметної області

1.2.1 Загальні відомості

Моделювання предметної області є одним із ключових етапів розробки інформаційної системи, оскільки саме на цьому етапі формується узагальнене уявлення про структуру майбутньої системи, її основні сутності, ролі користувачів та логіку взаємодії між окремими компонентами. Метою моделювання є формалізація процесів, які відбуваються в межах предметної області, а також їх подальше подання у вигляді зрозумілих графічних моделей.

Для вебсистеми продажу прикрас Astrobracelet моделювання предметної області є особливо важливим, оскільки система поєднує стандартні функції

інтернет-магазину з персоналізованим підбором товарів за знаком зодіаку, каменем, кольором та символічним запитом користувача. Такий підхід потребує не лише опису класичних процесів електронної комерції, а й деталізації тих механізмів, які забезпечують більш індивідуальний досвід взаємодії покупця з магазином.

У процесі моделювання для опису поведінки системи доцільно використовувати UML-діаграми, які дають можливість подати функціональні можливості системи у наочному вигляді. У межах даної роботи для опису предметної області застосовано діаграму прецедентів, діаграму послідовності та діаграму діяльності. Кожна з них відображає окремий аспект роботи вебсистеми. Діаграма прецедентів показує, які користувачі взаємодіють із системою та які функції для них доступні. Діаграма послідовності дозволяє відстежити порядок обміну повідомленнями між основними учасниками під час виконання конкретного сценарію. Діаграма діяльності дає можливість відобразити логіку проходження процесу від початку до завершення з урахуванням можливих варіантів розвитку подій.

Окрім поведінкових моделей, моделювання предметної області дозволяє виділити основні сутності системи та встановити зв’язки між ними. Основною сутністю системи є товар. Саме товар виступає центральним об’єктом каталогу та пов’язує між собою більшість інших сутностей. Для кожного товару зберігаються базові характеристики, зокрема назва, опис, ціна, наявність, короткий опис, зображення та належність до категорії. Разом із цим товар пов’язується з каменями, кольорами, знаками зодіаку та запитами користувача.

Наступною важливою сутністю є категорія. Вона використовується для групування товарів за типом прикраси або товарної групи. Одна категорія може містити багато товарів, тоді як кожен товар належить до однієї основної категорії. Окрему групу сутностей становлять характеристики

персоналізованого підбору: камінь, знак зодіаку, колір та запит користувача. Саме вони забезпечують можливість більш гнучкої фільтрації та добору товарів відповідно до потреб користувача.

Для реалізації процесу купівлі в системі також виділяються сутності замовлення, позиції замовлення та платіж. Замовлення містить інформацію про покупця, дату створення, статус, суму та параметри доставки. Позиція замовлення зберігає конкретний товар і його кількість у межах певного замовлення. Платіж фіксує інформацію про оплату замовлення, її статус та технічні параметри транзакції. Окремою сутністю системи є користувач, який у межах поточної реалізації може виконувати роль покупця або адміністратора.

Таким чином, моделювання предметної області дозволяє не лише краще зрозуміти структуру системи, а й створити основу для подальшого проєктування бази даних, архітектури програмного забезпечення та реалізації окремих функціональних модулів.

1.2.2 Діаграма прецедентів

Діаграма прецедентів використовується для відображення взаємодії між акторами системи та її основними функціональними можливостями. Вона дозволяє визначити, які ролі присутні в системі, які дії вони можуть виконувати, а також якими є межі функціональності розроблюваного програмного забезпечення. На рис. 1.1 наведено діаграму прецедентів вебсистеми продажу прикрас Astrobracelet.

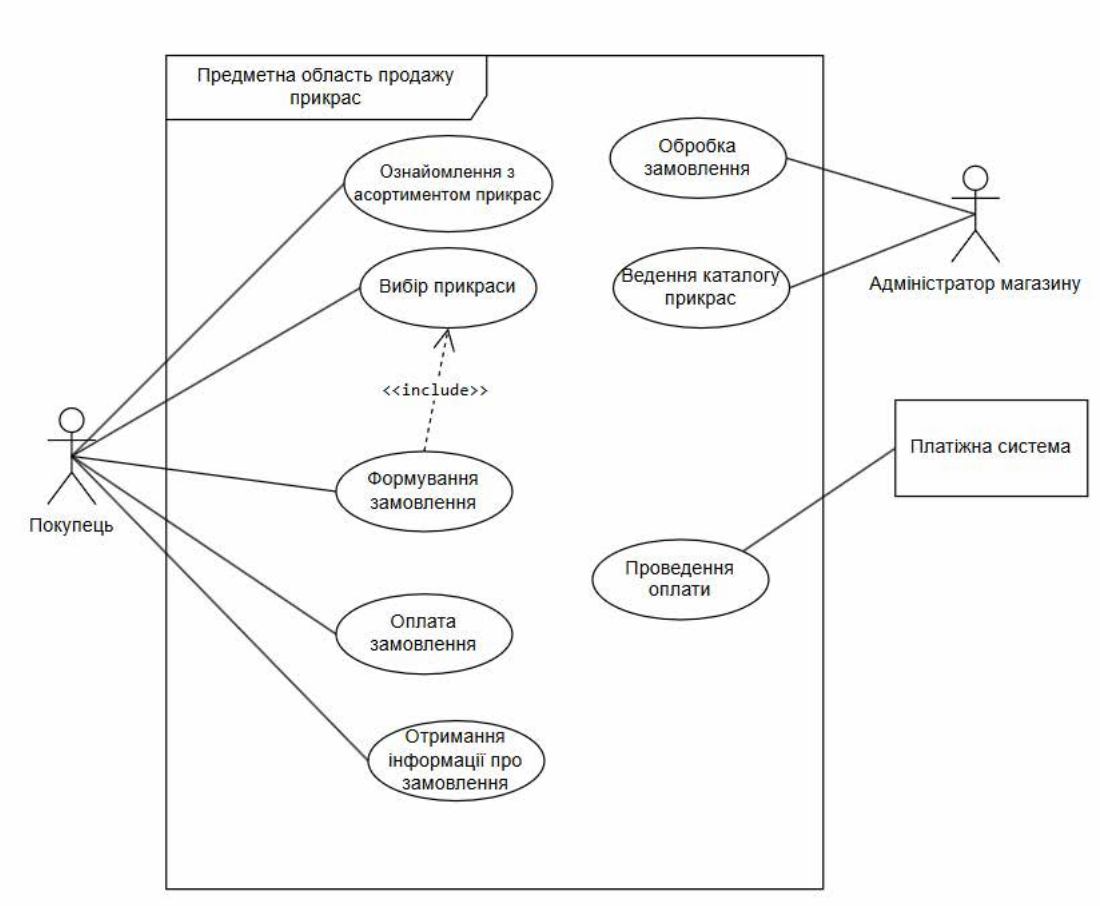

Рис. 1.1 – Діаграма прецедентів вебсистеми продажу прикрас Astrobracelet

Основним актором системи є покупець. Саме він взаємодіє з публічною частиною вебзастосунку та виконує найбільшу кількість дій. Покупець має можливість ознайомитися з асортиментом прикрас, вибрати потрібний товар, сформувати замовлення, перейти до оплати та отримати інформацію про статус замовлення. Таким чином, діаграма відображає типовий сценарій використання системи з точки зору кінцевого користувача.

Іншим важливим актором є адміністратор магазину. На відміну від покупця, його функції пов’язані не з купівлею товарів, а з підтримкою працездатності системи та керуванням її вмістом. Адміністратор відповідає за ведення каталогу прикрас, контроль замовлень, актуальність товарних позицій та загальне адміністрування магазину.

Окремо в діаграмі представлено платіжну систему як зовнішній учасник, що забезпечує проведення онлайн-оплати. Її участь у системі обмежується обробкою платіжної транзакції та передачею результату оплати.

Діаграма прецедентів дає змогу побачити, що система охоплює як клієнтські дії, пов’язані з вибором і купівлею прикрас, так і адміністративні дії, необхідні для підтримки каталогу й обробки замовлень. Це підтверджує, що розроблювана вебсистема має повний цикл функціонування сучасного інтернет-магазину.

1.2.3 Діаграма послідовності

Для наочного представлення процесу взаємодії між основними учасниками системи використовується діаграма послідовності. Вона відображає порядок обміну повідомленнями між користувачем, внутрішніми компонентами системи та зовнішніми сервісами в межах конкретного сценарію. На рис. 1.2 наведено діаграму послідовності процесу оформлення та оплати замовлення.

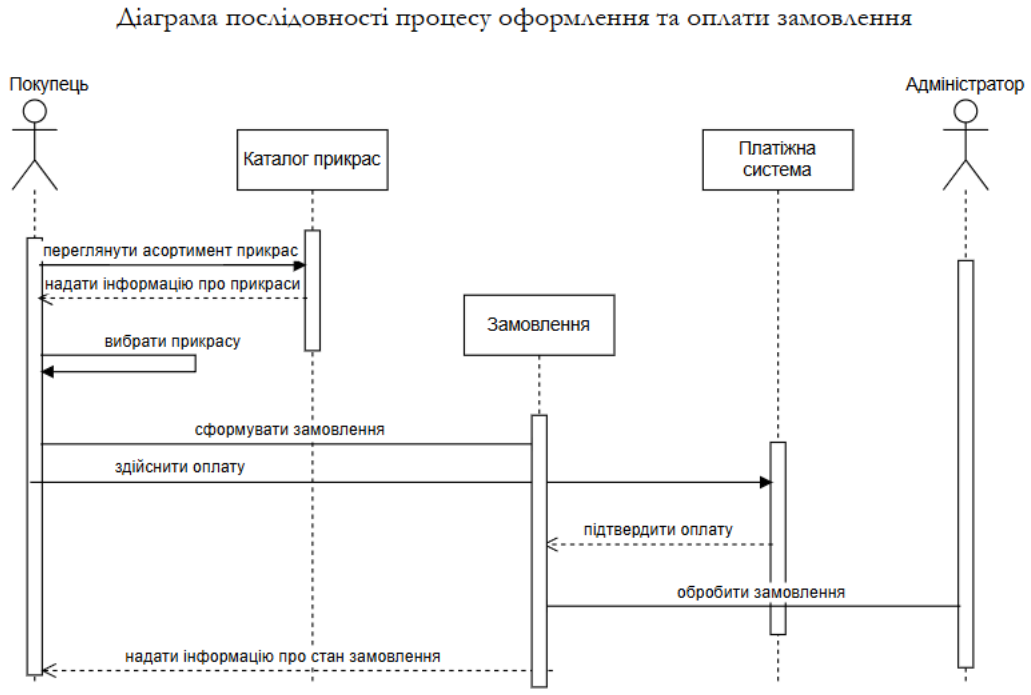

Рис. 1.2 – Діаграма послідовності процесу оформлення та оплати замовлення

Ця діаграма демонструє послідовність дій, які виконуються від моменту ознайомлення покупця з асортиментом до отримання інформації про результат оформлення замовлення.

У сценарії беруть участь такі основні об’єкти: покупець, каталог прикрас, замовлення, платіжна система та адміністратор. Процес починається з того, що покупець переглядає асортимент прикрас і отримує необхідну інформацію про доступні товари. Після цього користувач обирає потрібну прикрасу та переходить до формування замовлення.

На наступному етапі система створює замовлення та передає дані для виконання оплати. Після ініціювання платіжного процесу взаємодія переходить до платіжної системи, яка обробляє транзакцію та повертає результат її виконання. У разі успішної оплати система отримує підтвердження, після чого замовлення переходить до етапу подальшої обробки. На цьому етапі до процесу підключається адміністратор, який працює з новим замовленням у системі.

Завершальним елементом цього сценарію є надання покупцеві інформації про стан замовлення. Такий підхід дозволяє забезпечити логічно завершений цикл взаємодії, у якому користувач не лише формує замовлення та сплачує його, а й отримує зворотний зв’язок щодо результату виконаних дій.

Діаграма послідовності є важливою для розуміння того, як саме розподіляються ролі між компонентами системи, яким чином відбувається передача даних між ними та в якій послідовності виконується ключовий бізнес-процес.

1.2.4 Діаграма діяльності

Діаграма діяльності використовується для опису логіки виконання певного процесу у вигляді послідовності дій, умовних переходів та можливих варіантів завершення. На відміну від діаграми послідовності, яка акцентує

увагу на обміні повідомленнями між об’єктами, діаграма діяльності показує загальний потік виконання процесу від початку до завершення. На рис. 1.3 наведено діаграму діяльності, яка відображає процес вибору прикраси, оформлення замовлення та його подальшої обробки.

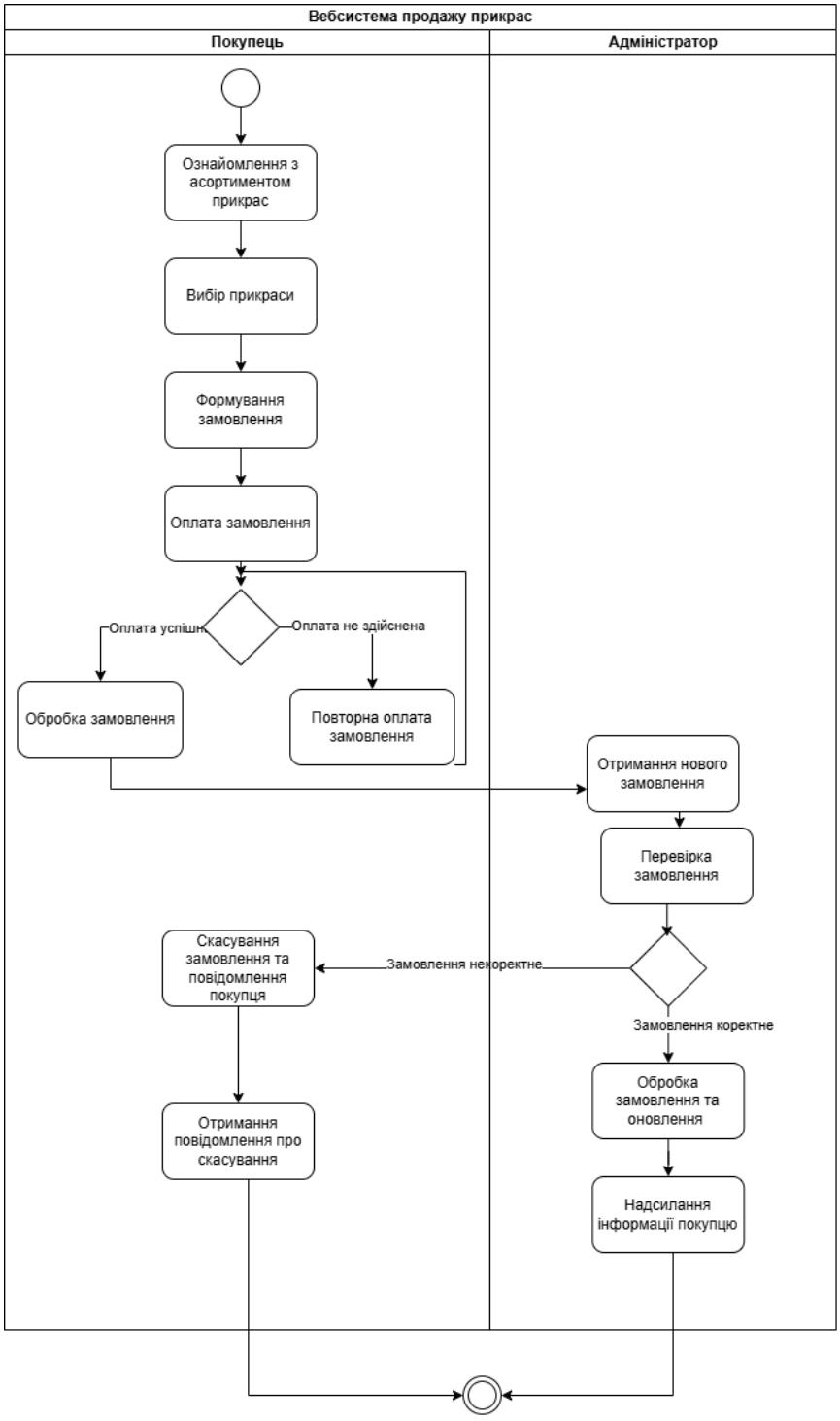

Рис. 1.3 – Діаграма діяльності процесу вибору прикраси, оформлення та обробки замовлення

Цей процес починається з ознайомлення покупця з асортиментом прикрас. Після цього користувач переходить до вибору прикраси та формування замовлення. Наступним кроком є оплата замовлення.

У межах цього процесу передбачено альтернативні варіанти розвитку подій. Якщо оплата проходить успішно, система переходить до етапу обробки замовлення. Якщо ж оплата не була здійснена, передбачено можливість повторної спроби оплати. Така логіка відповідає реальному сценарію роботи інтернет-магазину, де не всі користувачі завершують платіж з першої спроби.

Після успішного оформлення та підтвердження замовлення інформація про нього надходить до адміністратора. Адміністратор перевіряє замовлення та визначає, чи може воно бути коректно оброблене далі. Якщо замовлення відповідає вимогам системи, відбувається його подальша обробка та надсилання інформації покупцеві. Якщо ж виявлено проблему або некоректність, можливе скасування замовлення з відповідним повідомленням користувача.

Таким чином, діаграма діяльності відображає повний логічний цикл виконання одного з основних процесів системи. Вона дозволяє побачити як основний позитивний сценарій, так і альтернативні варіанти розвитку подій, що робить її корисним інструментом для аналізу поведінки системи на етапі проєктування.

Загалом моделювання предметної області дало змогу формалізувати основні ролі користувачів, структуру їхньої взаємодії із системою та логіку проходження ключових процесів. Це створює необхідну основу для подальшого переходу до проєктування інформаційного та програмного забезпечення системи.

1.3 Огляд існуючих аналогічних систем

Під час розробки вебсистеми продажу прикрас важливо враховувати вже існуючі підходи до організації онлайн-торгівлі, структури каталогу,

оформлення сторінок товарів, логіки замовлення та взаємодії з користувачем. Аналіз аналогічних систем дає змогу не лише ознайомитися з актуальними технологічними та функціональними рішеннями в цій сфері, а й виявити їхні сильні та слабкі сторони. Це, у свою чергу, дозволяє обґрунтовано визначити, які підходи доцільно використати під час створення власної вебсистеми.

Серед існуючих рішень у сфері онлайн-продажу прикрас можна умовно виділити кілька основних типів. До першого типу належать великі маркетплейси та універсальні платформи електронної комерції, де прикраси є лише однією з багатьох категорій товарів. Такі системи зазвичай мають добре реалізовану базову функціональність: зручний каталог, кошик, пошук, фільтрацію, оформлення замовлення та інтеграцію з платіжними сервісами. Їхньою перевагою є технічна стабільність, зрозуміла структура та відпрацьований сценарій купівлі. Проте для нішевих брендів прикрас подібні рішення часто виявляються занадто загальними, оскільки не враховують емоційну складову вибору виробу та не підтримують глибоку персоналізацію.

До другого типу належать спеціалізовані інтернет-магазини прикрас, які працюють із власним асортиментом та мають виражену візуальну айдентику. У таких магазинах, як правило, більше уваги приділяється дизайну, презентації товару, фотографіям, кольоровій гамі, історії бренду та формуванню емоційного зв’язку з покупцем. Перевагою таких рішень є більш цілісне користувацьке враження, краща відповідність стилю бренду та привабливіша подача товарів. Водночас недоліком часто є обмежена функціональність фільтрації та не завжди достатньо зручна логіка пошуку товарів.

До третього типу можна віднести бренди, що активно будують продажі через соціальні мережі, зокрема Instagram, і поступово переносять свою аудиторію у вебпростір. Для таких рішень характерні емоційна подача, акцент на фотографіях, візуальна атмосфера, жива презентація виробів та тісний

зв’язок із контентом бренду. Їхньою перевагою є сильний візуальний образ і близькість до цільової аудиторії. Однак у багатьох випадках подібні рішення мають обмежену функціональність: користувачеві не завжди зручно шукати товари, застосовувати фільтри, оформлювати замовлення або швидко знаходити прикраси за конкретними параметрами.

Аналіз подібних систем показує, що більшість інтернет-магазинів прикрас добре реалізують базові сценарії електронної комерції, але не завжди забезпечують персоналізований підбір. У багатьох випадках користувач може фільтрувати асортимент лише за стандартними параметрами, наприклад за ціною, типом виробу, новизною або наявністю. Для бренду Astrobracelet цього недостатньо, оскільки користувач часто орієнтується не тільки на зовнішній вигляд прикраси, а й на її сенс, камінь, знак зодіаку або внутрішній запит.

Ще одним недоліком багатьох існуючих рішень є або перевантаженість інтерфейсу, або, навпаки, надмірна спрощеність функціоналу. У першому випадку користувач отримує велику кількість елементів, що ускладнює сприйняття й знижує зручність навігації. У другому випадку сайт виглядає естетично, але не надає достатньо інструментів для зручного пошуку та оформлення покупки.

На основі проведеного огляду можна зробити висновок, що ефективна вебсистема продажу прикрас повинна поєднувати кілька ключових характеристик. Насамперед вона має забезпечувати зручну структуру каталогу та зрозумілу навігацію. Крім того, система повинна підтримувати гнучкий механізм фільтрації за характеристиками, які дійсно важливі для покупця в цій ніші. Важливе значення має і візуальна складова: стильний дизайн, якісні фото, продумане розташування елементів і гармонійна кольорова подача. Не менш важливо, щоб користувач міг легко пройти весь шлях від знайомства з товаром до оформлення замовлення та оплати.

У розроблюваній системі Astrobracelet було враховано зазначені особливості. На відміну від типових інтернет-магазинів, вона орієнтована не лише на демонстрацію товарів, а й на реалізацію персоналізованого підбору прикрас. Для цього передбачено фільтрацію за знаком зодіаку, каменем, кольором, типом прикраси та символічним запитом користувача. Крім того, система має адаптивний інтерфейс, адміністративну панель для керування асортиментом, модуль оформлення замовлення та інтеграцію з платіжною системою. Таким чином, розроблюване рішення поєднує технічну функціональність сучасного інтернет-магазину з емоційною та стилістичною подачею, яка відповідає концепції бренду Astrobracelet.

1.4 Постановка завдання

З функціональної точки зору система, що розробляється, призначена для організації повного циклу онлайн-продажу прикрас бренду Astrobracelet. Вона повинна забезпечувати зручний перегляд асортименту, персоналізований підбір товарів, оформлення замовлення, проведення онлайн-оплати та подальше адміністрування каталогу, замовлень і платежів. Особливістю розроблюваної системи є те, що вона орієнтована не лише на класичний продаж прикрас через інтернет, а й на індивідуальний підбір виробів за знаком зодіаку, каменем, кольором та символічним запитом користувача.

Основними користувачами системи є покупець та адміністратор магазину. Покупець взаємодіє з публічною частиною вебсистеми, переглядає каталог, підбирає прикраси за характеристиками, додає товари до кошика, оформлює замовлення та виконує оплату. Адміністратор магазину працює з адміністративною частиною системи та відповідає за наповнення каталогу, підтримку актуальності даних, контроль замовлень і платежів.

Для покупця система повинна забезпечувати можливість перегляду головної сторінки магазину, каталогу прикрас і детальної сторінки товару, виконання пошуку товарів, використання фільтрації за знаком зодіаку,

каменем, кольором, категорією або запитом, додавання товарів до кошика, зміни кількості позицій, оформлення замовлення, вибору параметрів доставки, переходу до оплати та отримання інформації про результат виконаного платежу.

Для адміністратора система повинна забезпечувати додавання нових товарів, редагування та видалення інформації про них, керування категоріями, каменями, кольорами, знаками зодіаку та запитами користувачів, перегляд замовлень, зміну їх статусів, перегляд інформації про платежі та загальне керування вмістом каталогу через адміністративну панель.

Система має бути зручною та інтуїтивно зрозумілою для користувача, забезпечуючи простий і логічний шлях від вибору прикраси до оформлення замовлення. Особливу увагу необхідно приділити навігації, структурі каталогу, якості відображення інформації про товари та зручності використання фільтрів. Поведінка системи повинна бути передбачуваною та послідовною. Перехід між сторінками каталогу, товару, кошика, оформлення замовлення та оплати має відбуватися так, як очікує користувач. Інтерфейс повинен містити зрозумілі елементи навігації, кнопки основних дій, засоби пошуку та фільтрації, а також доступну форму оформлення замовлення.

Стиль вебсистеми має відповідати концепції бренду Astrobracelet. Інтерфейс повинен бути візуально легким, сучасним і гармонійним, із використанням м’якої кольорової палітри, естетичних фотографій та акцентів, що створюють атмосферу індивідуальності, ніжності та емоційної привабливості. Основним вмістом сторінок мають бути блоки з інформацією про товари, елементи персоналізованого підбору, кнопки дій, фільтри та форми взаємодії з користувачем.

Нефункціональні вимоги до системи визначають її якісні характеристики, що є важливими для ефективної реалізації та подальшої експлуатації. Продуктивність системи повинна забезпечувати швидке

завантаження сторінок і комфортну взаємодію користувача з вебзастосунком. Бажано, щоб основні операції, такі як відкриття каталогу, сторінки товару, кошика чи оформлення замовлення, виконувалися без відчутних затримок. Масштабованість системи повинна передбачати можливість її подальшого розвитку. Архітектура застосунку має дозволяти розширення функціональності, додавання нових категорій товарів, нових характеристик для підбору, розширення асортименту та інтеграцію з додатковими сервісами без критичної перебудови всієї системи.

Однією з ключових вимог є також зручність використання. Інтерфейс має бути адаптивним, зрозумілим для користувача та придатним для роботи як на персональних комп’ютерах, так і на мобільних пристроях. Користувач не повинен витрачати зайвий час на пошук основних функцій або розуміння логіки роботи системи. Сумісність системи повинна забезпечувати коректну роботу в сучасних веббраузерах і на різних операційних системах. Також важливою є надійна взаємодія з серверною частиною, базою даних та платіжним сервісом.

Отже, розроблювана вебсистема повинна забезпечити комплексний і сучасний підхід до організації продажу прикрас в онлайн-середовищі. Вона має поєднувати функціональність інтернет-магазину, гнучкі механізми персоналізованого підбору, зручність користування, привабливий дизайн і надійність обробки даних. Саме така система може стати ефективним інструментом для повноцінної роботи бренду Astrobracelet у цифровому середовищі.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

Етап проєктування інформаційного забезпечення є одним із визначальних у процесі створення програмної системи, оскільки саме на цьому етапі формується структура даних, встановлюються зв’язки між основними сутностями та визначається спосіб їх подальшого фізичного зберігання. Для вебсистеми продажу прикрас Astrobracelet це має особливе значення, оскільки система поєднує класичні механізми електронної комерції з додатковими засобами персоналізованого підбору товарів.

У межах цього розділу розглядаються логічна модель даних, обґрунтування вибору системи управління базами даних, розробка структури інформаційної бази та її фізичне представлення у середовищі PostgreSQL. Послідовне виконання цих етапів дозволяє побудувати основу для надійної та стабільної роботи всієї системи.

2.1 Логічна модель даних у вигляді ER-діаграми

Після формування концептуального уявлення про предметну область наступним етапом проєктування є побудова логічної моделі даних. Логічна модель є важливою частиною розробки інформаційної системи, оскільки саме вона визначає структуру даних, які будуть використовуватися в програмному забезпеченні, а також зв’язки між ними. Вона виступає проміжною ланкою між загальним описом предметної області та фізичною реалізацією бази даних.

Для побудови логічної моделі використовується модель «сутність-зв’язок» (Entity-Relationship Model, ER-модель). Вона дає можливість у наочній формі подати основні сутності системи, їх атрибути та відношення між ними. Такий підхід є зручним, оскільки дозволяє формалізувати структуру даних незалежно від конкретного програмного середовища та ще до етапу фізичної реалізації оцінити повноту й коректність майбутньої бази даних.

Основними компонентами ER-моделі є сутності, атрибути та зв’язки. Сутності відображають об’єкти або поняття, які мають значення для системи. Атрибути задають характеристики цих сутностей. Зв’язки відображають відношення між сутностями та формують загальну логіку роботи системи. На рис. 2.1 наведено ER-діаграму бази даних вебсистеми продажу прикрас Astrobracelet, яка відображає основні сутності системи та зв’язки між ними.

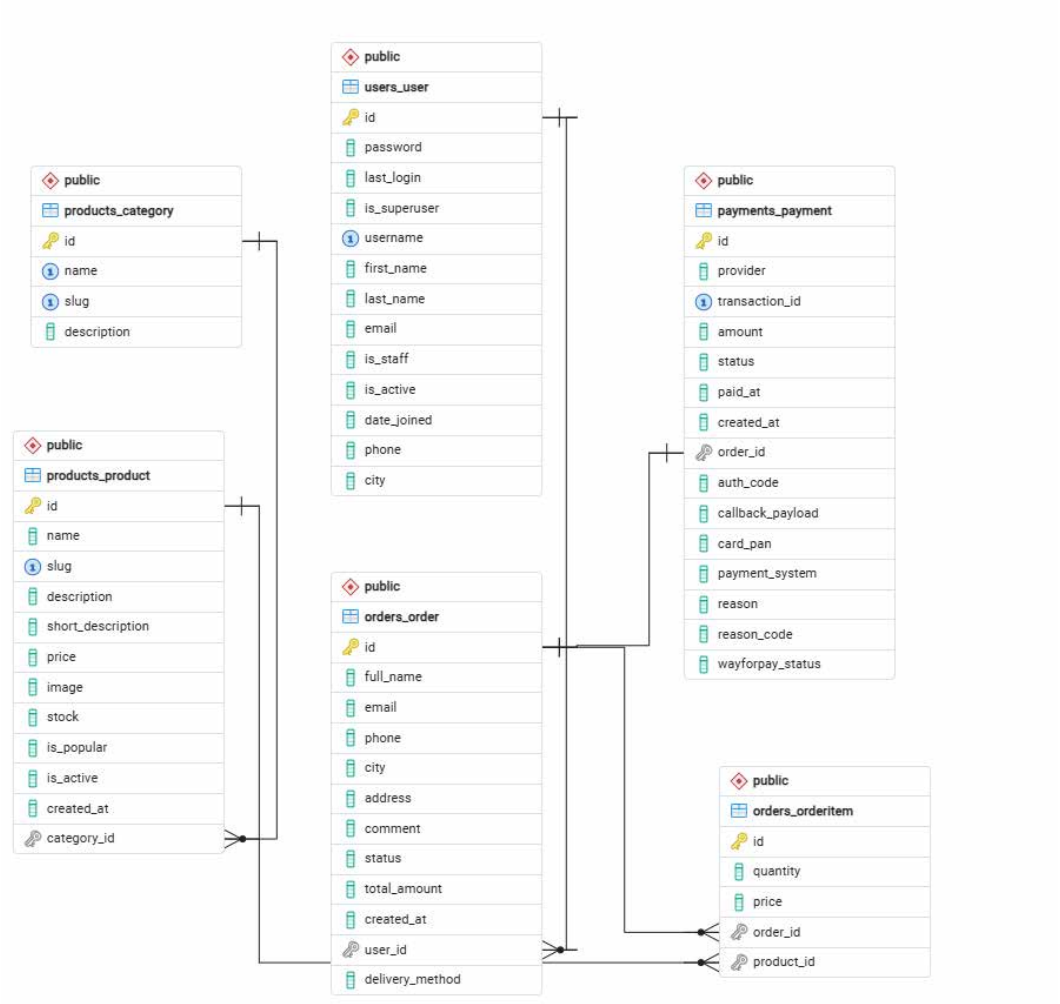

Рис. 2.1 – ER-діаграма бази даних вебсистеми продажу прикрас Astrobracelet

У логічній моделі вебсистеми Astrobracelet основними сутностями є користувач, категорія, товар, замовлення, позиція замовлення та платіж. Центральне місце в моделі посідає сутність товару, оскільки саме вона є основою каталогу прикрас і пов’язує між собою більшість інших елементів системи.

Сутність користувача використовується для збереження даних про осіб, які мають доступ до системи. У межах реалізації ця сутність необхідна насамперед для адміністративної частини, а також може використовуватися для підтримки розширених користувацьких сценаріїв. Для кожного користувача зберігаються основні ідентифікаційні дані, електронна пошта, службові параметри доступу та додаткові контактні відомості.

Сутність категорії призначена для логічного групування товарів. Вона дозволяє поділити асортимент за типом прикраси або товарної групи. Така структура спрощує навігацію по каталогу та робить пошук товарів зрозумілішим для користувача.

Сутність товару містить основну інформацію про прикрасу: її назву, короткий та повний опис, ціну, ознаку активності, кількість на складі, дату створення та належність до категорії. Саме товар є базовою одиницею каталогу та безпосередньо відображається в інтерфейсі сайту.

Сутність замовлення зберігає інформацію про оформлену покупку. До її складу входять дані про покупця, контактна інформація, адреса, статус, загальна сума, дата створення та інші атрибути, потрібні для супроводу замовлення. Замовлення є ключовою сутністю в процесі взаємодії між клієнтом і магазином.

Сутність позиції замовлення деталізує склад замовлення. Її використання є необхідним, оскільки одне замовлення може містити кілька різних товарів. У межах цієї сутності фіксуються кількість товару та його ціна на момент оформлення замовлення. Саме через позицію замовлення реалізується зв’язок між товарами та замовленнями.

Сутність платіж відповідає за збереження інформації про оплату замовлення. Вона містить суму, статус, ідентифікатор транзакції, назву платіжного провайдера та інші службові параметри, які потрібні для обробки й перевірки платіжних операцій.

Між сутностями в системі реалізовано низку зв’язків. Між категорією та товаром існує зв’язок типу «один до багатьох», оскільки одна категорія може містити багато товарів, але кожен товар належить лише одній категорії. Між користувачем і замовленням також реалізовано зв’язок «один до багатьох», тому що один користувач може мати кілька замовлень.

Між замовленням і позицією замовлення існує зв’язок «один до багатьох», оскільки одне замовлення може включати кілька позицій. Аналогічно між товаром і позицією замовлення також реалізовано зв’язок «один до багатьох», оскільки один товар може входити до багатьох різних замовлень. Між замовленням і платежем у межах логіки системи реалізовано зв’язок «один до одного», оскільки для одного замовлення формується окремий платіжний запис.

Крім основних зв’язків, у системі існують і розширені зв’язки, пов’язані з персоналізованим підбором прикрас. Один товар може бути пов’язаний з кількома каменями, кольорами, знаками зодіаку та користувацькими запитами, що дозволяє реалізувати більш гнучкий механізм фільтрації. Такі зв’язки мають важливе значення для особливостей бренду Astrobracelet, оскільки саме вони забезпечують індивідуалізацію вибору товарів.

Логічна модель даних була приведена до нормалізованого вигляду. Усі атрибути сутностей є атомарними, що відповідає першій нормальній формі. Неключові атрибути повністю залежать від первинних ключів, що відповідає другій нормальній формі. Також у моделі усунуто транзитивні залежності, що дозволяє говорити про її приведення до третьої нормальної форми. Це забезпечує зменшення дублювання даних, підвищує цілісність інформації та створює основу для стабільної роботи системи.

Таким чином, побудована ER-модель повністю відображає структуру даних вебсистеми продажу прикрас Astrobracelet та є основою для подальшої реалізації фізичної бази даних у PostgreSQL.

2.2 Вибір системи управління базами даних

Вибір системи управління базами даних є важливим етапом проєктування інформаційного забезпечення, оскільки саме від нього залежить надійність зберігання інформації, швидкість доступу до даних, підтримка зв’язків між таблицями та зручність подальшого супроводу системи. Для вебсистеми Astrobracelet було обрано PostgreSQL.

PostgreSQL є сучасною реляційною системою управління базами даних, яка широко використовується в розробці вебзастосунків. Її перевагою є підтримка складних зв’язків між таблицями, зовнішніх ключів, механізмів забезпечення цілісності даних, транзакцій та високий рівень стабільності. Для системи електронної комерції ці характеристики є особливо важливими, оскільки база даних повинна надійно зберігати товари, замовлення, платежі та допоміжні сутності без ризику втрати або некоректного пов’язування інформації.

Ще однією причиною вибору PostgreSQL є її сумісність із фреймворком Django, який використовується для розробки серверної частини системи. У Django передбачено зручний механізм ORM, який дозволяє працювати з таблицями бази даних через Python-моделі. Завдяки цьому структура інформаційної бази описується на рівні коду, а PostgreSQL виступає надійним середовищем фізичного зберігання даних.

Для даного проєкту важливою є також підтримка розширюваності. Оскільки система Astrobracelet містить не лише базові таблиці електронної комерції, а й додаткові сутності, пов’язані з персоналізованим підбором прикрас, обрана СКБД повинна була легко підтримувати розширення структури бази даних. PostgreSQL добре підходить для таких задач, оскільки дозволяє без суттєвих труднощів змінювати структуру таблиць, додавати нові зв’язки та працювати з більшими обсягами даних у разі подальшого розвитку проєкту.

Порівняно з легшими рішеннями, зокрема вбудованими базами даних, PostgreSQL є більш придатною для реального багатокористувацького вебзастосунку. Вона забезпечує кращу керованість даними, більшу надійність і кращі можливості для інтеграції з серверною інфраструктурою.

Отже, вибір PostgreSQL для реалізації інформаційної бази вебсистеми Astrobracelet є обґрунтованим з точки зору надійності, сумісності з технологічним стеком, підтримки реляційної моделі та можливості подальшого розширення системи.

2.3 Розробка структури інформаційної бази

Після вибору системи управління базами даних наступним етапом стала розробка структури інформаційної бази. На цьому етапі логічна модель даних була конкретизована у вигляді набору таблиць, полів і зв’язків, що відповідають реальним потребам вебсистеми.

Структура інформаційної бази системи Astrobracelet побудована таким чином, щоб забезпечити підтримку всіх основних функціональних сценаріїв: перегляду каталогу, збереження інформації про товари, оформлення замовлень, проведення оплати та адміністрування вмісту магазину.

Основними таблицями інформаційної бази є таблиця користувачів, таблиця категорій, таблиця товарів, таблиця замовлень, таблиця позицій замовлення та таблиця платежів. Кожна з них виконує окрему роль у загальній структурі системи.

Таблиця користувачів зберігає службові та контактні дані осіб, які мають доступ до системи. У поточній реалізації вона використовується насамперед для адміністративної частини, але також створює основу для розширення користувацького функціоналу в майбутньому.

Таблиця категорій призначена для логічного групування прикрас. Її структура є відносно простою, проте саме вона забезпечує впорядкованість каталогу й спрощує подальшу навігацію по товарах.

Таблиця товарів є центральним елементом усієї бази даних. У ній зберігається інформація про назву виробу, його опис, короткий опис, ціну, зображення, кількість доступних одиниць, дату створення, ознаку популярності та активності, а також посилання на категорію. На рівні бізнес-логіки саме ця таблиця використовується для формування каталогу, відображення карток товарів і реалізації пошуку.

Таблиця замовлень містить відомості про оформлені покупки. У ній зберігаються ім’я покупця, електронна пошта, номер телефону, місто, адреса, спосіб доставки, коментар до замовлення, його статус, загальна сума та дата створення. Це дозволяє системі зберігати повну інформацію про кожне оформлене замовлення та супроводжувати його на всіх етапах обробки.

Таблиця позицій замовлення виконує допоміжну, але дуже важливу роль. Вона пов’язує товари із замовленнями та дає змогу зберігати окремі елементи замовлення разом із їх кількістю та ціною. Такий підхід є необхідним, оскільки одне замовлення може містити кілька товарів, а один товар може входити до різних замовлень.

Таблиця платежів використовується для збереження результатів роботи платіжної системи. У ній фіксується сума платежу, його статус, ідентифікатор транзакції, технічна інформація, яка повертається платіжним провайдером, та зв’язок із відповідним замовленням. Це дозволяє контролювати оплату на рівні системи та забезпечує зв’язок між процесом купівлі і фінансовою частиною.

Окремо слід зазначити, що структура інформаційної бази в системі не обмежується лише цими таблицями. Для підтримки персоналізованого підбору прикрас використовуються також таблиці, що описують додаткові характеристики товарів, зокрема камені, кольори, знаки зодіаку та користувацькі запити. Саме ці елементи забезпечують більш гнучку логіку

відбору прикрас і дозволяють користувачеві знаходити товари не лише за формальними, а й за змістовими характеристиками.

Отже, розроблена структура інформаційної бази є достатньо повною для підтримки всіх основних процесів вебсистеми Astrobracelet і водночас залишається придатною до подальшого розширення.

2.4 Схема та фізична структура бази даних

Після розробки структури інформаційної бази було виконано її фізичну реалізацію у середовищі PostgreSQL. На цьому етапі всі основні сутності, визначені в логічній моделі, були представлені у вигляді реальних таблиць бази даних із відповідними типами полів, первинними ключами та зовнішніми зв’язками.

Фізична структура бази даних реалізована у схемі public і містить таблиці, які безпосередньо використовуються в предметній області проєкту. До них належать users_user, products_category, products_product, orders_order, orders_orderitem та payments_payment. Саме ці таблиці формують основу фізичного зберігання інформації в системі.

У додатку А подано схему фізичної структури бази даних вебсистеми Astrobracelet у середовищі PostgreSQL, сформовану на основі фактично реалізованих таблиць.

У фізичній структурі кожна таблиця містить набір полів, що відповідає її призначенню. Наприклад, таблиця products_category містить ідентифікатор категорії, її назву, символьний код і опис. Таблиця products_product включає назву товару, опис, короткий опис, ціну, шлях до зображення, кількість на складі, службові ознаки активності та популярності, дату створення та зовнішній ключ на категорію.

Таблиця orders_order містить поля для збереження даних покупця, контактної інформації, параметрів доставки, статусу замовлення, загальної

суми та дати створення. Таблиця orders_orderitem забезпечує фізичну реалізацію складу замовлення та пов’язує окремий товар із конкретним замовленням. Таблиця payments_payment містить поля, пов’язані з фінансовою транзакцією, зокрема ідентифікатор транзакції, статус, суму, технічні коди відповіді та дату оплати.

Фізична реалізація бази даних була здійснена за допомогою механізму міграцій Django. Це означає, що створення таблиць і зв’язків відбувалося не вручну через окремі SQL-скрипти, а на основі моделей, описаних у коді застосунку. Після визначення моделей система Django автоматично сформувала необхідні міграції, які потім були застосовані до PostgreSQL. Такий підхід забезпечив точну відповідність між програмною моделлю системи та реальною фізичною структурою бази даних.

Важливою перевагою такої організації є зручність супроводу та розвитку проєкту. У разі зміни структури таблиць або додавання нових сутностей ці зміни можуть бути послідовно внесені через нові міграції без потреби повністю перебудовувати базу даних. Це особливо важливо для системи, яка може надалі розширюватися, зокрема шляхом додавання нових механізмів персоналізації, нових способів доставки чи платіжних сервісів.

Таким чином, фізична структура бази даних вебсистеми Astrobracelet є реалізацією розробленої логічної моделі, забезпечує надійне збереження інформації та створює технічну основу для стабільної роботи всієї системи.

Підсумовуючи, можна зазначити, що етап проєктування інформаційного забезпечення дозволив сформувати повну структуру даних вебсистеми, визначити доцільну СКБД, реалізувати логічну та фізичну моделі бази даних і підготувати основу для подальшої програмної реалізації системи.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Абстракція предметної області

Абстракція предметної області є важливим етапом переходу від загального опису системи до її програмної реалізації. Якщо в першому та другому розділах увага зосереджувалася на аналізі предметної області, її моделюванні та проєктуванні інформаційного забезпечення, то на цьому етапі необхідно подати систему у вигляді більш узагальненої програмної моделі. Такий підхід дозволяє виділити основні об’єкти, ролі користувачів, суттєві властивості та взаємозв’язки, які надалі будуть реалізовані у вигляді класів, модулів і програмних компонентів.

Для вебсистеми продажу прикрас Astrobracelet абстракція предметної області має особливе значення, оскільки система поєднує класичну логіку інтернет-магазину з індивідуалізованим підбором прикрас. У межах такого підходу важливо виділити не лише стандартні сутності електронної комерції, а й ті елементи, що відображають специфіку бренду, а саме: знаки зодіаку, камені, кольори та символічні запити користувача.

На абстрактному рівні центральним об’єктом системи є прикраса. Саме прикраса виступає основною одиницею каталогу, навколо якої будується більшість функціональних сценаріїв. Вона має назву, опис, вартість, зображення, ознаку доступності, належність до категорії, а також набір характеристик, що використовуються для персоналізованого підбору. У межах абстракції прикраса розглядається не лише як товар, а як об’єкт, що поєднує матеріальні та символічні властивості.

Наступним важливим об’єктом є категорія. Вона використовується для узагальнення та групування прикрас за типом. Категорія дає змогу впорядкувати асортимент і сформувати зрозумілу структуру каталогу. На рівні

абстракції вона виконує роль засобу класифікації товарів і підтримує навігацію в системі.

Окрему групу об’єктів формують характеристики персоналізованого підбору. До них належать камінь, знак зодіаку, колір та запит користувача. Ці об’єкти є важливими саме для системи Astrobracelet, оскільки вони визначають не лише спосіб фільтрації товарів, а й логіку рекомендацій та емоційне наповнення вибору. Один і той самий товар може бути пов’язаний із кількома характеристиками, а одна характеристика, у свою чергу, може стосуватися багатьох товарів. Це робить модель більш гнучкою і наближеною до реальної поведінки користувача.

З точки зору взаємодії з системою важливими абстрактними об’єктами є покупець та адміністратор. Покупець розглядається як користувач, що здійснює пошук, перегляд, вибір і купівлю товарів. Його взаємодія із системою пов’язана з використанням каталогу, фільтрації, кошика, оформлення замовлення та оплати. Адміністратор, навпаки, виконує функції керування: він працює з товарами, категоріями, характеристиками, замовленнями та платежами. Таким чином, на рівні абстракції система вже поділяється на публічну частину для покупця та службову частину для адміністратора.

У межах процесу купівлі важливими об’єктами є замовлення, позиція замовлення та платіж. Замовлення виступає узагальненням покупки як завершеної дії користувача. Воно об’єднує інформацію про покупця, вибрані товари, параметри доставки та загальну суму. Позиція замовлення деталізує склад замовлення і дозволяє відобразити, які саме товари були придбані та в якій кількості. Платіж, у свою чергу, є окремим об’єктом, що відображає фінансовий аспект купівлі та містить відомості про статус і результат платіжної операції.

Таким чином, абстракція предметної області дозволяє подати вебсистему Astrobracelet як сукупність взаємопов’язаних об’єктів, кожен із

яких виконує власну роль у загальній логіці роботи системи. Такий підхід є необхідним для подальшого моделювання структури та взаємодії об’єктів, а також для переходу до програмної реалізації окремих модулів.

3.2 Моделювання структури та взаємодії об’єктів

Після виділення основних абстракцій предметної області наступним етапом розробки є моделювання структури та взаємодії об’єктів системи. Якщо на попередньому етапі увага зосереджувалася на тому, які саме сутності існують у межах вебсистеми продажу прикрас, то на цьому етапі важливо показати, яким чином ці об’єкти пов’язані між собою та як вони беруть участь у виконанні основних функціональних сценаріїв.

Моделювання структури об’єктів дозволяє подати систему у вигляді впорядкованого набору класів, кожен з яких відображає певну сутність предметної області та має власні атрибути і дії. Для системи Astrobracelet такий підхід є доцільним, оскільки він дає змогу простежити перехід від логічної моделі даних до програмної реалізації, а також показати, як саме сутності предметної області втілюються у структурі застосунку.

На рис. 3.1 наведено діаграму класів вебсистеми продажу прикрас Astrobracelet.

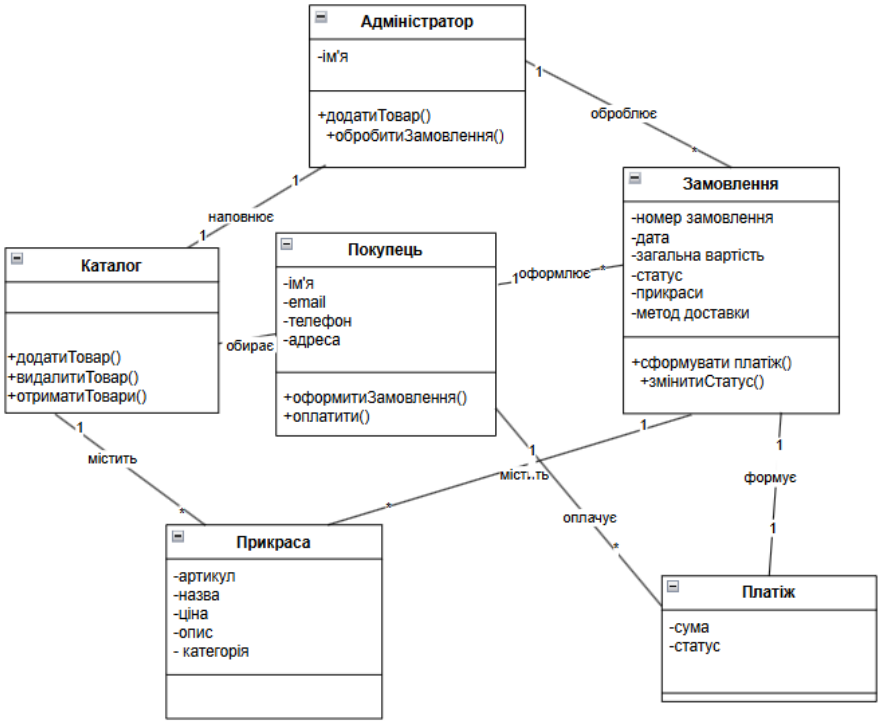

Рис. 3.1 – Діаграма класів вебсистеми продажу прикрас Astrobracelet

У межах цієї моделі центральне місце посідає клас прикраси, який відображає основний об’єкт каталогу. Саме він містить ключові характеристики товару, зокрема назву, опис, ціну та належність до категорії. Через цей клас реалізується зв’язок із каталогом, а також участь товару в процесі оформлення замовлення. Прикраса є тим об’єктом, з яким безпосередньо працює покупець під час взаємодії з публічною частиною системи.

З класом прикраси тісно пов’язаний клас категорії, що використовується для групування товарів за типом. Наявність такого зв’язку дозволяє впорядкувати каталог, побудувати логічну структуру розділів магазину та забезпечити зручну навігацію для користувача. У реальній роботі системи це означає, що кожна прикраса належить до певної категорії, а одна категорія може об’єднувати багато товарів.

Окреме місце в моделі посідають класи, пов’язані з користувачами системи. Клас покупця відображає сторону користувача, яка працює з каталогом, обирає товари, формує замовлення та виконує оплату. Клас адміністратора, навпаки, відповідає за службову частину системи та пов’язаний із діями з керування товарами, обробки замовлень і контролю платежів. Такий розподіл дозволяє на рівні моделі одразу показати різницю між публічним і адміністративним сценаріями використання системи.

Важливим елементом моделі є клас замовлення. Він виконує роль центрального об’єкта в процесі купівлі та пов’язує між собою покупця, товари й оплату. У межах цього класу зосереджується інформація про дату створення замовлення, його статус, загальну вартість та інші параметри, що характеризують покупку як завершену дію користувача. Саме через замовлення система переходить від перегляду товарів до фіксації реальної комерційної операції.

Із замовленням безпосередньо пов’язаний клас платежу. Він відображає фінансовий аспект роботи системи й використовується для збереження інформації про суму платежу, його статус та результат виконання транзакції. Наявність окремого класу платежу є важливою, оскільки це дозволяє розділити логіку оформлення замовлення і логіку оплати, що робить модель більш зрозумілою та зручною для подальшої реалізації.

Окрім структури самих класів, важливе значення має моделювання їхньої взаємодії. У системі Astrobracelet ця взаємодія найкраще простежується під час виконання основних користувацьких сценаріїв. Наприклад, під час оформлення замовлення покупець взаємодіє з каталогом, обирає прикрасу, після чого система створює об’єкт замовлення, додає до нього відповідні товари, формує суму та передає дані до підсистеми оплати. Після цього створюється або оновлюється об’єкт платежу, а результат платіжної операції впливає на подальший статус замовлення.

Інший важливий сценарій взаємодії об’єктів пов’язаний із адмініструванням. У цьому випадку адміністратор взаємодіє з об’єктами товарів, категорій, замовлень і платежів не як покупець, а як користувач, що керує вмістом системи. Він може додавати нові прикраси, змінювати їх характеристики, оновлювати статуси замовлень та контролювати результати оплат. Таким чином, одна й та сама структура об’єктів підтримує різні сценарії використання залежно від ролі користувача.

Слід також відзначити, що моделювання взаємодії об’єктів у даній системі не обмежується лише внутрішніми класами. Важливу роль відіграє також взаємодія із зовнішнім платіжним сервісом. Хоча платіжна система формально не належить до внутрішньої об’єктної моделі застосунку, вона безпосередньо впливає на стан об’єкта платежу та через нього — на стан об’єкта замовлення. Це означає, що модель взаємодії повинна враховувати не лише внутрішні, а й зовнішні зв’язки системи.

Отже, моделювання структури та взаємодії об’єктів дозволяє подати вебсистему Astrobracelet як цілісну програмну модель, у якій кожен об’єкт виконує власну роль, а зв’язки між ними забезпечують реалізацію основних бізнес-процесів. Такий підхід створює основу для подальшого опису організаційної структури програмного забезпечення та переходу до реалізації окремих модулів системи.

3.3 Організаційна структура програмного забезпечення

Організаційна структура програмного забезпечення відображає, яким чином побудовано систему на рівні її основних програмних частин, як ці частини розподілені за відповідальністю та яким чином вони взаємодіють між собою в процесі роботи. Для вебсистеми продажу прикрас Astrobracelet це особливо важливо, оскільки система поєднує кілька взаємопов’язаних напрямів: відображення каталогу, керування користувацьким сценарієм купівлі, обробку замовлень, оплату та адміністрування вмісту магазину.

На практичному рівні розроблюване програмне забезпечення організоване за модульним принципом. Такий підхід означає, що окремі частини системи винесені в самостійні програмні модулі або застосунки, кожен із яких відповідає за власну функціональну область. Це дозволяє зробити код більш впорядкованим, зрозумілим у супроводі та зручним для подальшого розвитку.

На рис. 3.2 наведено діаграму пакетів, яка відображає загальну організаційну структуру програмного забезпечення системи Astrobracelet.

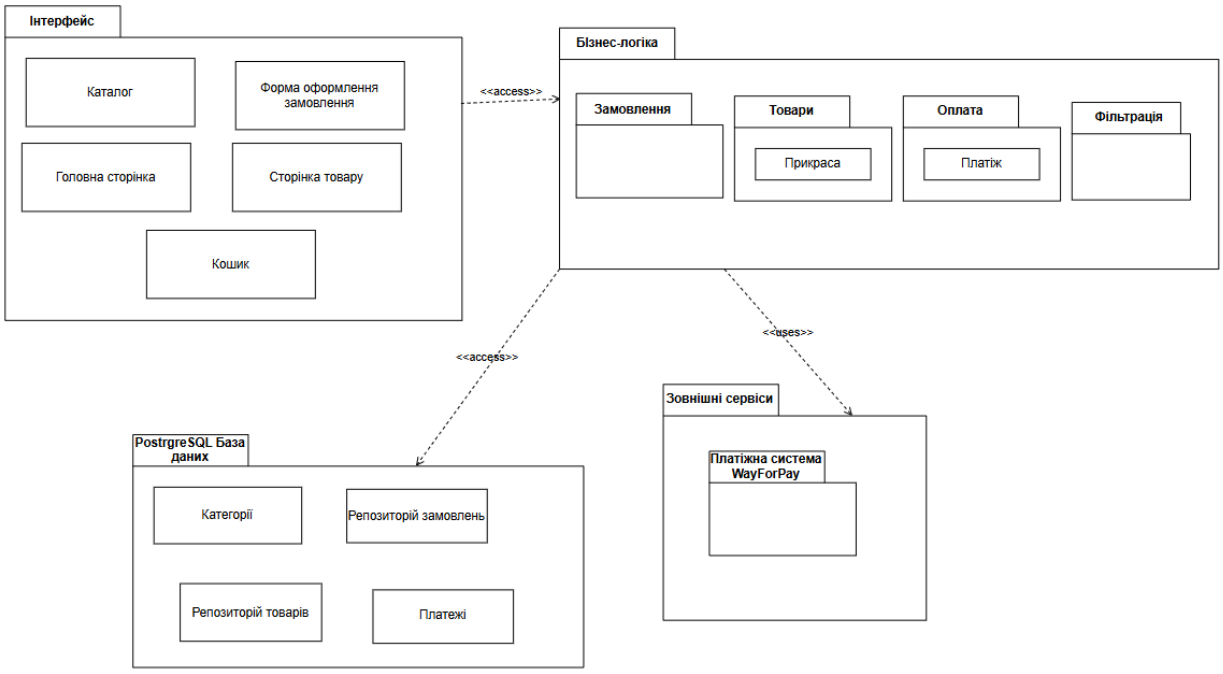

Рис. 3.2 – Діаграма пакетів вебсистеми продажу прикрас Astrobracelet

Узагальнено структуру системи можна поділити на чотири основні частини: інтерфейс, бізнес-логіку, базу даних та зовнішні сервіси. Пакет інтерфейсу охоплює ті елементи системи, з якими безпосередньо взаємодіє користувач. До нього належать головна сторінка, каталог, сторінка товару, кошик і форма оформлення замовлення. Саме ця частина відповідає за візуальне представлення даних, забезпечує навігацію та приймає дії користувача.

Інтерфейс головної сторінки забезпечує перше знайомство користувача з брендом та дозволяє перейти до основних розділів магазину. На рис. 3.3 наведено вигляд головної сторінки вебсистеми Astrobracelet.

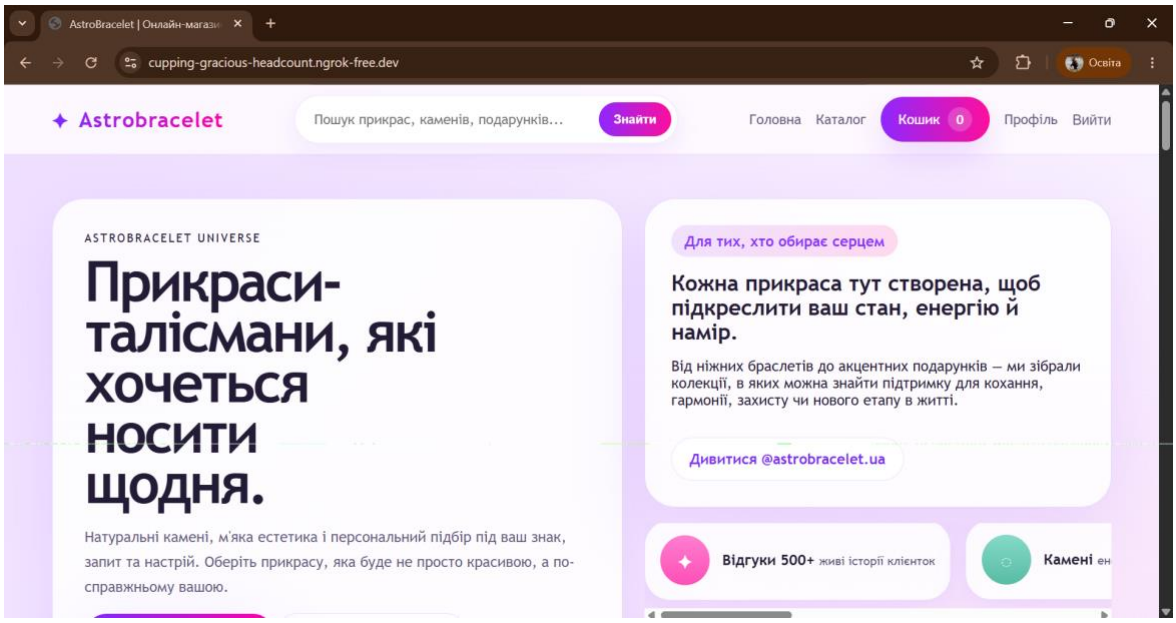

Рис. 3.3 – Головна сторінка вебсистеми Astrobracelet

Основним засобом взаємодії покупця з асортиментом є каталог прикрас. Саме через нього користувач переглядає товари, застосовує фільтри та переходить до детальнішого ознайомлення з окремими позиціями. На рис. 3.4 наведено сторінку каталогу прикрас.

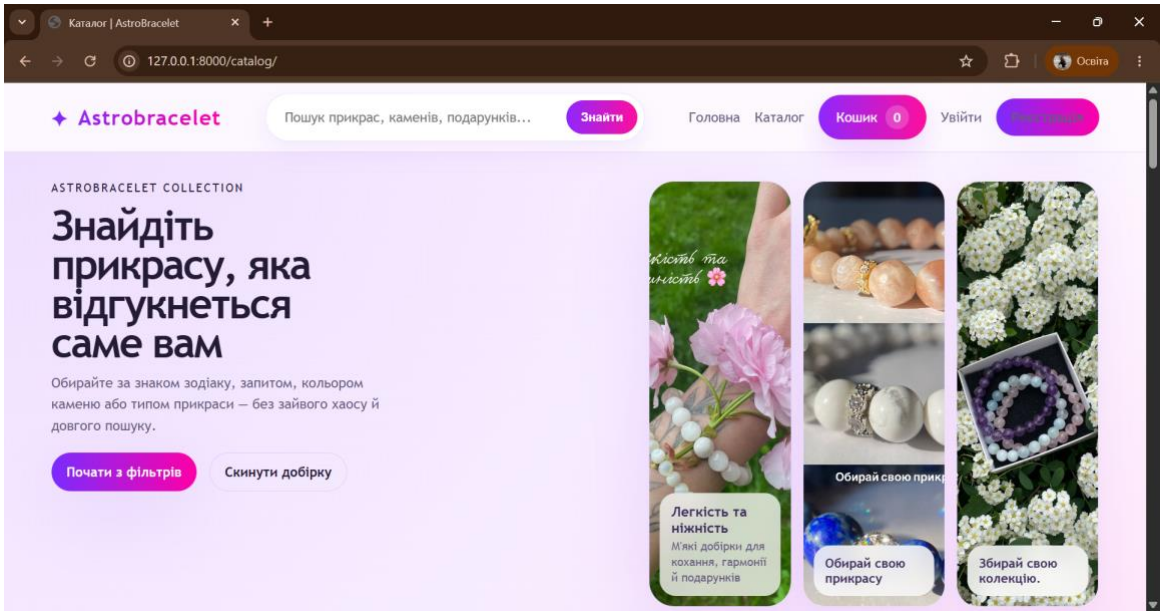

Рис. 3.4 – Каталог прикрас вебсистеми Astrobracelet

Після вибору товару користувач переходить на сторінку окремої прикраси, де отримує повну інформацію про неї, включаючи опис, вартість і додаткові характеристики. На рис. 3.5 наведено сторінку товару.

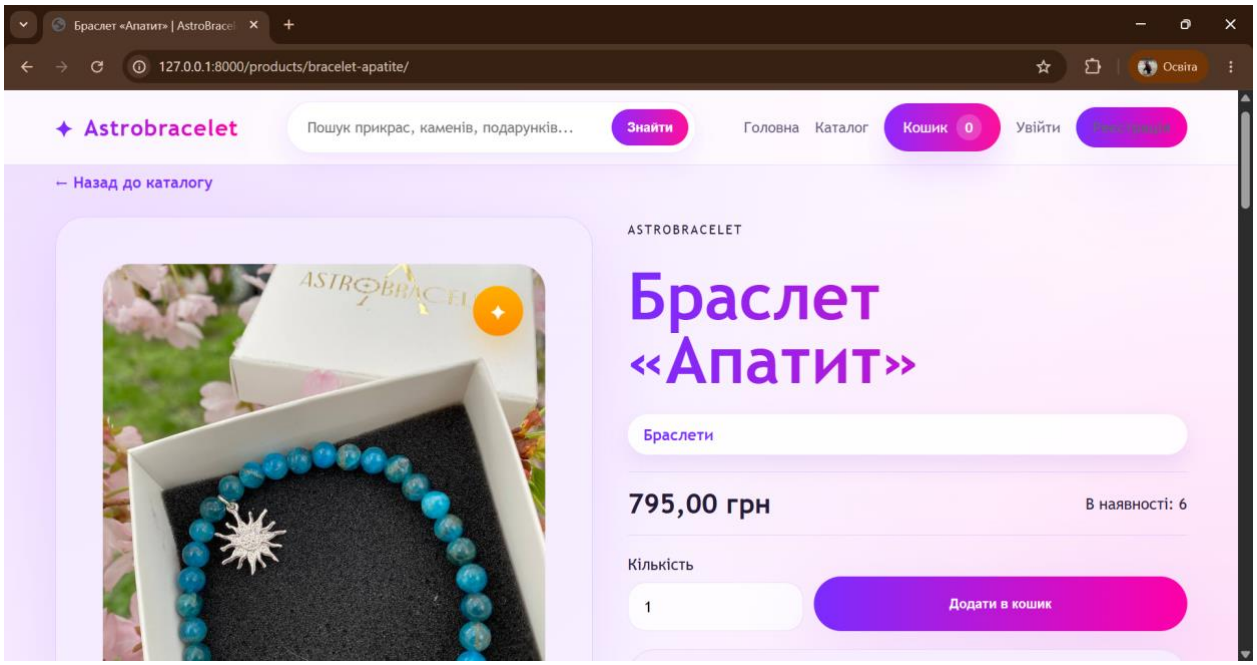

Рис. 3.5– Сторінка окремого товару вебсистеми Astrobracelet

Пакет бізнес-логіки містить модулі, які забезпечують основну поведінку системи. Тут зосереджено обробку каталогу, логіку замовлень, підготовку та перевірку даних, фільтрацію товарів, роботу з кошиком, підготовку параметрів оплати та супровід замовлення після його створення. Саме на цьому рівні система приймає рішення про те, як реагувати на дії користувача, які дані зберігати, які сторінки формувати та які стани об’єктів змінювати.

Пакет бази даних відповідає за фізичне збереження інформації. До нього належать таблиці, пов’язані з товарами, категоріями, замовленнями, позиціями замовлень, платежами, а також допоміжними сутностями, що підтримують персоналізований підбір прикрас. Окремим елементом організаційної структури є пакет зовнішніх сервісів, до якого належить платіжна система WayForPay.

На рівні програмної реалізації організаційна структура системи деталізується через окремі застосунки Django. У межах проєкту Astrobracelet

ключову роль відіграють застосунки products, orders, payments, users та api. Кожен із них відповідає за власний набір сутностей, представлень та логіки.

Застосунок products охоплює все, що пов’язано з каталогом товарів, категоріями та додатковими характеристиками прикрас. Саме тут реалізуються моделі товарів, логіка відображення каталогу, сторінки окремого товару та механізми фільтрації. Застосунок orders відповідає за роботу кошика, створення замовлення, збереження його позицій та організацію переходу до оплати. Застосунок payments реалізує логіку взаємодії з платіжною системою, фіксує результати платежів і відповідає за зміну платіжних статусів. Застосунок users забезпечує підтримку користувацької моделі та адміністративного доступу. Застосунок api формує програмний інтерфейс для обміну даними між окремими частинами системи або для можливого подальшого розширення функціоналу.

Такий модульний поділ має кілька важливих переваг. По-перше, він дозволяє розмежувати відповідальність між окремими частинами коду. По-друге, це спрощує супровід проєкту, оскільки зміни в одному функціональному блоці не обов’язково вимагають втручання в усю систему. По-третє, така структура полегшує масштабування, адже нові можливості можна додавати у вигляді нових модулів або через розширення вже наявних.

Отже, організаційна структура програмного забезпечення вебсистеми Astrobracelet побудована за модульним принципом і охоплює інтерфейс, бізнес-логіку, базу даних та зовнішні сервіси. Такий підхід забезпечує зрозумілу внутрішню будову системи, спрощує її супровід та створює основу для подальшого вдосконалення програмного забезпечення.

3.4 Компонентна структура системи

Компонентна структура системи дозволяє розглянути програмне забезпечення на більш технічному рівні та показати, з яких основних програмних блоків воно складається, через які інтерфейси відбувається

взаємодія між цими блоками та яким чином забезпечується зв’язок між клієнтською частиною, серверною логікою, базою даних і зовнішніми сервісами. Якщо організаційна структура програмного забезпечення дає загальне уявлення про поділ системи на логічні модулі, то компонентна структура деталізує, як саме ця система функціонує в програмному середовищі.

Для вебсистеми Astrobracelet компонентний підхід є особливо доречним, оскільки застосунок побудовано за принципами клієнт-серверної архітектури. Це означає, що окремо існує клієнтська сторона, через яку користувач взаємодіє із системою, і серверна сторона, яка відповідає за обробку запитів, бізнес-логіку, доступ до бази даних та інтеграцію із зовнішніми сервісами. Саме компонентна структура дозволяє наочно показати цей поділ та встановити зв’язки між його основними елементами.

На рис. 3.6 наведено діаграму компонентів вебсистеми продажу прикрас Astrobracelet.

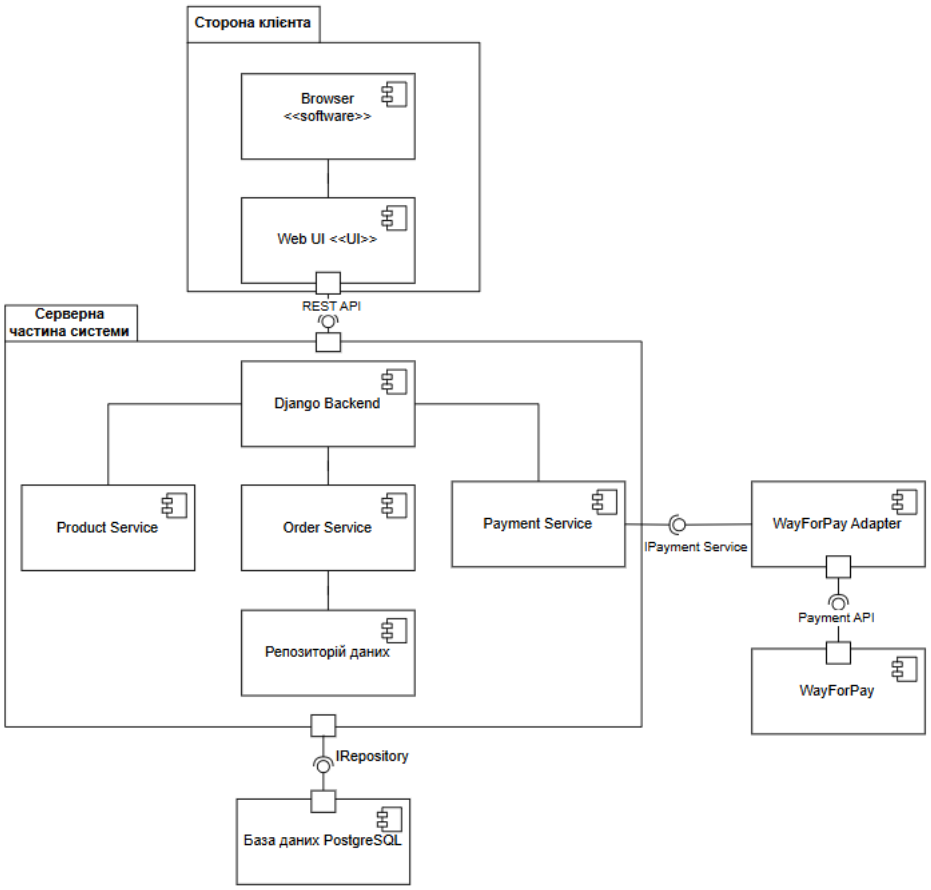

Рис. 3.6 – Діаграма компонентів вебсистеми продажу прикрас AstrobraceletPuc.

На клієнтському рівні система представлена браузером і вебінтерфейсом. Браузер виступає середовищем виконання, через яке користувач переглядає сторінки магазину, працює з каталогом, відкриває сторінку товару, переходить до кошика та оформлення замовлення. Вебінтерфейс, у свою чергу, забезпечує відображення всіх цих сторінок і реалізує механізм взаємодії користувача з функціями системи. Саме через нього надсилаються запити до серверної частини і повертаються результати обробки.

Зв’язок між клієнтською та серверною сторонами здійснюється через програмний інтерфейс взаємодії. У межах цієї системи він реалізований як сукупність HTTP-запитів і серверних маршрутів, а також, у разі потреби, API-запитів для окремих сценаріїв. Це дозволяє чітко розмежувати функції відображення інтерфейсу та функції обробки даних.

Центральним компонентом серверної частини є Django Backend. Саме він об’єднує основну бізнес-логіку системи й виступає ядром, через яке проходять запити користувача. У межах серверної частини доцільно виділити кілька функціональних сервісів або підкомпонентів. Компонент роботи з товарами відповідає за відображення каталогу, отримання списків товарів, обробку сторінок окремих прикрас та реалізацію механізму фільтрації. Компонент роботи із замовленнями забезпечує функціонування кошика, створення замовлення, збереження позицій замовлення та підготовку інформації для подальшої оплати. Компонент оплати відповідає за формування платіжних параметрів, збереження статусу транзакції, обробку результатів оплати та синхронізацію цих результатів із замовленням.

Окремим важливим елементом серверної частини є рівень доступу до даних. Він забезпечує взаємодію між бізнес-логікою та базою даних і дозволяє ізолювати механізми збереження інформації від прикладної логіки. У межах Django ця роль частково реалізується через ORM, яка дозволяє працювати з таблицями бази даних як із Python-об’єктами, не переходячи постійно на рівень ручних SQL-запитів. Це робить структуру системи більш зрозумілою та спрощує супровід коду.

На рівні збереження даних використовується база даних PostgreSQL. Вона виступає окремим компонентом, що забезпечує фізичне зберігання всієї інформації, яка використовується в системі. Саме тут зберігаються товари, категорії, користувачі, замовлення, позиції замовлення, платежі та інші сутності, необхідні для повноцінної роботи інтернет-магазину. Використання окремого серверного компонента для бази даних відповідає класичній клієнт-серверній архітектурі та підвищує керованість системи.

Особливе місце в компонентній структурі посідає взаємодія із зовнішнім платіжним сервісом WayForPay. Для цієї інтеграції в системі доцільно виділити окремий компонент або адаптер, який відповідає за обмін даними з

платіжною платформою. Завдяки такому підходу внутрішня логіка застосунку не залежить безпосередньо від технічних деталей зовнішнього сервісу. Це не лише спрощує структуру коду, а й створює можливість у майбутньому замінити або доповнити платіжний сервіс без суттєвої перебудови всієї системи.

Компонентна структура також показує, що окремі частини системи не існують ізольовано, а працюють у тісній взаємодії. Користувач через браузер надсилає запит до вебінтерфейсу, той передає його на сервер, серверна логіка опрацьовує дані, за потреби звертається до бази даних або платіжного сервісу, після чого формує відповідь і повертає її користувачу. Такий ланцюг взаємодії є основою функціонування всієї системи.

Отже, компонентна структура вебсистеми Astrobracelet відображає її як цілісне програмне рішення, у якому клієнтська частина, серверна логіка, база даних і зовнішній платіжний сервіс поєднані в єдиний механізм. Такий підхід дозволяє не лише краще зрозуміти архітектуру застосунку, а й обґрунтовує технічні рішення, прийняті під час його реалізації.

3.5 Вибір інструментарію для створення прикладного програмного забезпечення

Вибір інструментарію для створення прикладного програмного забезпечення є одним із ключових етапів розробки, оскільки саме від нього залежить зручність реалізації системи, якість програмного коду, можливість подальшого супроводу та масштабування проєкту. Для вебсистеми продажу прикрас Astrobracelet було обрано такий набір технологічних засобів, який дозволяє реалізувати повноцінний клієнт-серверний застосунок із сучасним інтерфейсом, надійною базою даних та підтримкою онлайн-оплати.

Основною мовою програмування серверної частини стала Python. Вибір цієї мови зумовлений її читабельністю, логічною структурою, широкою популярністю у сфері веброзробки та великою кількістю бібліотек, що

дозволяють швидко реалізовувати прикладні задачі. Python добре підходить для освітніх і практичних проєктів, оскільки дозволяє зосередитися на логіці системи, не перевантажуючи код зайвою технічною складністю.

Як основний серверний фреймворк було використано Django. Його вибір є обґрунтованим з кількох причин. По-перше, Django надає готову структуру проєкту, яка дозволяє організовувати програмний код за модульним принципом. По-друге, фреймворк містить вбудовані засоби маршрутизації, роботи з формами, шаблонами, користувачами та адміністративною панеллю. По-третє, важливою перевагою Django є наявність ORM, що дозволяє працювати з базою даних через Python-моделі, а не лише через ручні SQL-запити. Для системи інтернет-магазину такий підхід значно спрощує розробку й супровід.

Для створення клієнтської частини системи використовуються HTML, CSS і JavaScript. HTML слугує основою для структурування сторінок і побудови інтерфейсу. CSS забезпечує оформлення, адаптивність і візуальну цілісність сайту. JavaScript використовується для покращення інтерактивності, реалізації динамічних сценаріїв і створення більш зручної взаємодії з окремими елементами інтерфейсу. Поєднання цих технологій дозволяє створити повноцінний вебінтерфейс, який коректно відображається на різних пристроях і підтримує сучасний рівень користувацького досвіду.

Для збереження інформації в системі використовується PostgreSQL. Ця система керування базами даних була обрана завдяки своїй надійності, підтримці реляційної моделі, роботі зі складними зв’язками між таблицями та високій сумісності з Django. PostgreSQL добре підходить для зберігання даних про товари, замовлення, платежі та інші сутності електронної комерції, а також забезпечує стійкість і цілісність інформації.

Для інтеграції онлайн-оплати використовується WayForPay. Вибір саме цього сервісу пояснюється його поширеністю, наявністю готового API та

зручністю підключення до вебзастосунку. Використання WayForPay дозволяє реалізувати перехід користувача до зовнішньої платіжної сторінки, отримання зворотної інформації про результат платежу та синхронізацію платіжного статусу із замовленням у межах системи.

У процесі розробки також використовуються допоміжні інструменти, які спрощують роботу з проєктом. Для написання й редагування коду доцільно використовувати інтегроване середовище розробки Visual Studio Code, яке забезпечує зручну роботу з Python, HTML, CSS, JavaScript і конфігураційними файлами проєкту. Для запуску системи в ізольованому середовищі та спрощення процесу розгортання використовується Docker, що дозволяє контейнеризувати застосунок і базу даних та забезпечити більш передбачуване середовище виконання.

Важливою перевагою обраного інструментарію є його узгодженість. Усі використані технології добре поєднуються між собою: Django ефективно працює з PostgreSQL, HTML і CSS природно інтегруються в шаблонну систему Django, а WayForPay підключається як зовнішній сервіс до модулю оплати. Завдяки цьому система розробляється в єдиному технічному середовищі без потреби в надмірно складних інтеграційних рішеннях.

Отже, вибір інструментарію для створення прикладного програмного забезпечення вебсистеми Astrobracelet є обґрунтованим як з точки зору технічної доцільності, так і з точки зору практичної реалізації. Обраний набір технологій дозволяє створити стабільну, розширювану та зручну у використанні систему, що відповідає вимогам сучасної веброзробки.

3.6 Алгоритмізація та програмування програмних модулів

Після визначення структури системи, її основних об’єктів, компонентів та інструментальних засобів наступним етапом стала безпосередня алгоритмізація та програмна реалізація основних модулів вебсистеми Astrobracelet. На цьому етапі логічні моделі та архітектурні рішення були

перетворені на реальний програмний код, який забезпечує роботу каталогу, фільтрації, кошика, оформлення замовлення, оплати та адміністративного керування.

Розробка виконувалася за модульним принципом. Це означає, що функціональність системи була розподілена між окремими застосунками, кожен із яких відповідає за власну частину бізнес-логіки. Такий підхід дозволяє зробити код більш зрозумілим, спрощує підтримку проєкту та створює основу для його подальшого розширення.

Одним із ключових програмних модулів системи є модуль роботи з товарами. Саме він забезпечує відображення каталогу, формування карток товарів, відкриття сторінки окремої прикраси та реалізацію пошуку і фільтрації. Алгоритм роботи цього модуля полягає в тому, що після звернення користувача до каталогу система формує вибірку товарів із бази даних, а потім застосовує до неї фільтри, передані з інтерфейсу. У межах Astrobracelet користувач може відбирати прикраси за категорією, знаком зодіаку, каменем, кольором і символічним запитом. Після обробки цих параметрів система повертає лише ті товари, які відповідають обраним умовам.

Окреме значення має модуль відображення сторінки товару. Після вибору прикраси система отримує детальну інформацію про конкретний товар, пов’язані з ним характеристики та додаткові рекомендації. У межах проєкту це особливо важливо, оскільки товар розглядається не лише як елемент каталогу, а як персоналізований об’єкт, пов’язаний із певними каменями, знаками зодіаку та запитами користувача. Саме тому алгоритм формування сторінки товару включає не лише отримання базової інформації, а й підбір супровідних характеристик.

Наступним важливим модулем є модуль кошика. Його задача полягає в організації тимчасового збереження вибраних товарів до моменту оформлення замовлення. Коли користувач додає товар у кошик, система перевіряє, чи існує

така позиція серед уже обраних, і залежно від цього або створює новий запис, або оновлює кількість товару. Аналогічно реалізовано зміну кількості позицій і видалення товарів із кошика. Логіка цього модуля забезпечує зручну взаємодію користувача з майбутнім замовленням ще до переходу до етапу оплати.

З метою наочного відображення логіки цього процесу блок-схему алгоритму оформлення замовлення подано в додатку Б.

Після завершення формування кошика система переходить до модуля оформлення замовлення. Алгоритм цього процесу починається з перевірки вмісту кошика та приймання даних, введених користувачем у формі. Далі система створює новий запис замовлення, у який заносить контактні дані покупця, адресу, спосіб доставки, коментар та інші необхідні параметри. Після цього для кожного товару з кошика створюється окрема позиція замовлення із зазначенням кількості та ціни на момент оформлення покупки. На завершальному етапі система обчислює загальну суму замовлення й підготовлює його до переходу в модуль оплати.

Окрему роль у системі виконує модуль оплати. Саме він забезпечує взаємодію із зовнішньою платіжною системою WayForPay. Алгоритм його роботи полягає в тому, що після створення замовлення формується або оновлюється запис про платіж, до якого заносяться сума, ідентифікатор транзакції, статус та інші службові параметри. Далі система готує набір даних для передачі у платіжний сервіс і перенаправляє користувача на зовнішню сторінку оплати. Після завершення платіжної операції система отримує результат транзакції та змінює статус платежу і пов’язаного з ним замовлення.

Блок-схему алгоритму проведення онлайн-оплати також наведено в додатку Б.

Особливістю реалізації платіжного модуля є те, що він враховує не лише успішний сценарій, а й альтернативні ситуації. Якщо оплата не була завершена

або пройшла з помилкою, система зберігає відповідний статус і дозволяє повторити спробу. Такий підхід є важливим з точки зору реальної експлуатації, оскільки в електронній комерції не всі платежі завершуються з першого разу, і система повинна коректно реагувати на такі випадки.

Окрім користувацьких сценаріїв, у системі реалізовано модулі адміністративної обробки. Через адміністративну частину можна додавати нові товари, змінювати інформацію про них, керувати категоріями, кольорами, каменями, знаками зодіаку та користувацькими запитами, переглядати замовлення і змінювати їх статуси. Програмно це реалізовано через адміністративну панель Django, яка була адаптована під потреби проєкту. Завдяки цьому адміністрування не потребує окремого складного інтерфейсу, але водночас забезпечує повний контроль над вмістом системи.

Важливим результатом алгоритмізації стало також розділення серверної логіки між окремими програмними модулями. У модулі products зосереджено логіку роботи з каталогом і прикрасами, у модулі orders — формування кошика та замовлення, у модулі payments — інтеграцію з WayForPay, у модулі users — підтримку користувацької моделі, а в модулі api — програмний інтерфейс для обміну даними. Такий розподіл підвищує читабельність коду та дозволяє пов’язати кожен фрагмент реалізації з конкретною функціональною частиною системи.

Таким чином, етап алгоритмізації та програмування дозволив реалізувати повний набір основних функціональних модулів вебсистеми Astrobracelet. У результаті було створено цілісне програмне рішення, яке підтримує повний цикл онлайн-продажу прикрас: від пошуку та персоналізованого підбору товарів до оформлення замовлення, проведення оплати та подальшого адміністрування системи.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

Після завершення етапів аналізу, проєктування та програмної реалізації системи важливим є визначення практичних умов її використання. Навіть функціонально повноцінний вебзастосунок не може вважатися завершеним без перевірки працездатності, оцінки технічних вимог, опису умов розгортання та аналізу отриманих результатів. Для вебсистеми продажу прикрас Astrobracelet ці аспекти мають особливе значення, оскільки система орієнтована не лише на демонстрацію технічної реалізації, а й на можливість подальшого практичного використання в умовах реального онлайн-продажу.

У межах цього розділу розглядаються результати тестування системи, вимоги до апаратного та програмного забезпечення, склад інсталяційного пакету, а також загальна оцінка отриманих результатів. Такий підхід дозволяє не лише показати завершеність розробки, а й обґрунтувати можливість її подальшого впровадження.

4.1 Тестування системи

Тестування є невід’ємною частиною процесу розробки програмного забезпечення, оскільки саме на цьому етапі перевіряється відповідність системи поставленим вимогам, виявляються можливі помилки та оцінюється стабільність роботи реалізованих функцій. Для вебсистеми продажу прикрас Astrobracelet тестування було необхідним для перевірки коректності роботи каталогу, механізму фільтрації, кошика, оформлення замовлення, модуля оплати та адміністративної частини.

У процесі перевірки системи використовувалися модульне, інтеграційне, системне та приймальне тестування. Модульне тестування дозволяло перевірити коректність роботи окремих елементів системи, зокрема логіки відображення товарів, створення замовлень та оновлення статусів

платежів. Інтеграційне тестування було спрямоване на перевірку взаємодії між окремими модулями, наприклад між каталогом, кошиком, замовленням і платіжним сервісом. Системне тестування дало змогу оцінити роботу вебзастосунку в цілому, а приймальне тестування дозволило визначити, наскільки система відповідає очікуванням кінцевого користувача та сформульованим вимогам.

Для основного етапу перевірки було обрано функціональне тестування методом «чорної скриньки». Такий підхід передбачає перевірку системи з точки зору користувача без заглиблення у внутрішню структуру програмного коду. Під час тестування аналізувалися вхідні дії користувача та результати, які система повертає у відповідь. Використання цього методу є доцільним для вебсистеми продажу прикрас, оскільки він дозволяє оцінити не лише технічну коректність виконання функцій, а й зручність практичного використання системи в реальному сценарії.

У межах тестування було перевірено кілька основних функціональних сценаріїв. Насамперед було протестовано перегляд каталогу та сторінки окремого товару. Перевірялося, чи коректно відображаються товари, їхні зображення, описи, ціни та додаткові характеристики. Також було протестовано механізм фільтрації за категорією, знаком зодіаку, каменем, кольором і символічним запитом. Результати перевірки показали, що система коректно формує вибірки товарів відповідно до заданих параметрів. На рис. 4.1 наведено приклад перевірки каталогу прикрас із використанням механізму фільтрації: обрано знак зодіаку «Овен».

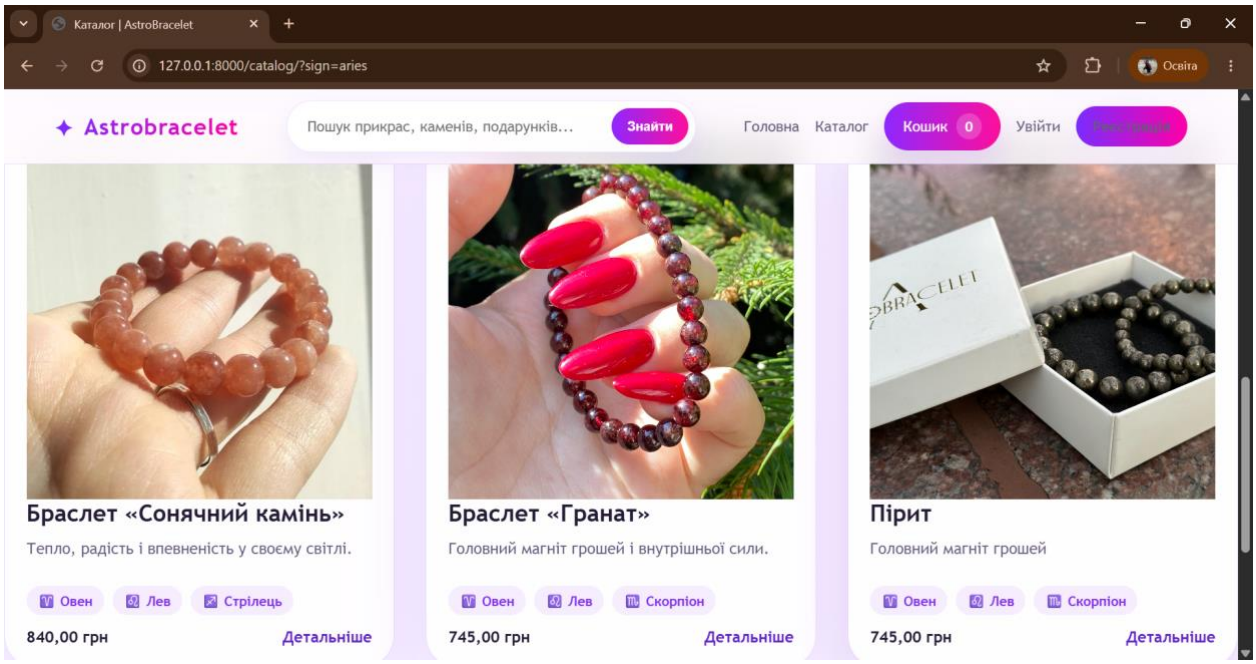

Рис. 4.1 – Перевірка роботи каталогу та фільтрації прикрас

Окрему увагу було приділено тестуванню кошика. У цьому сценарії перевірялася можливість додавання товару до кошика, зміни кількості позицій, видалення товарів і збереження актуальної суми замовлення. Було встановлено, що система коректно реагує на зміни вмісту кошика та правильно оновлює підсумкову вартість покупки.

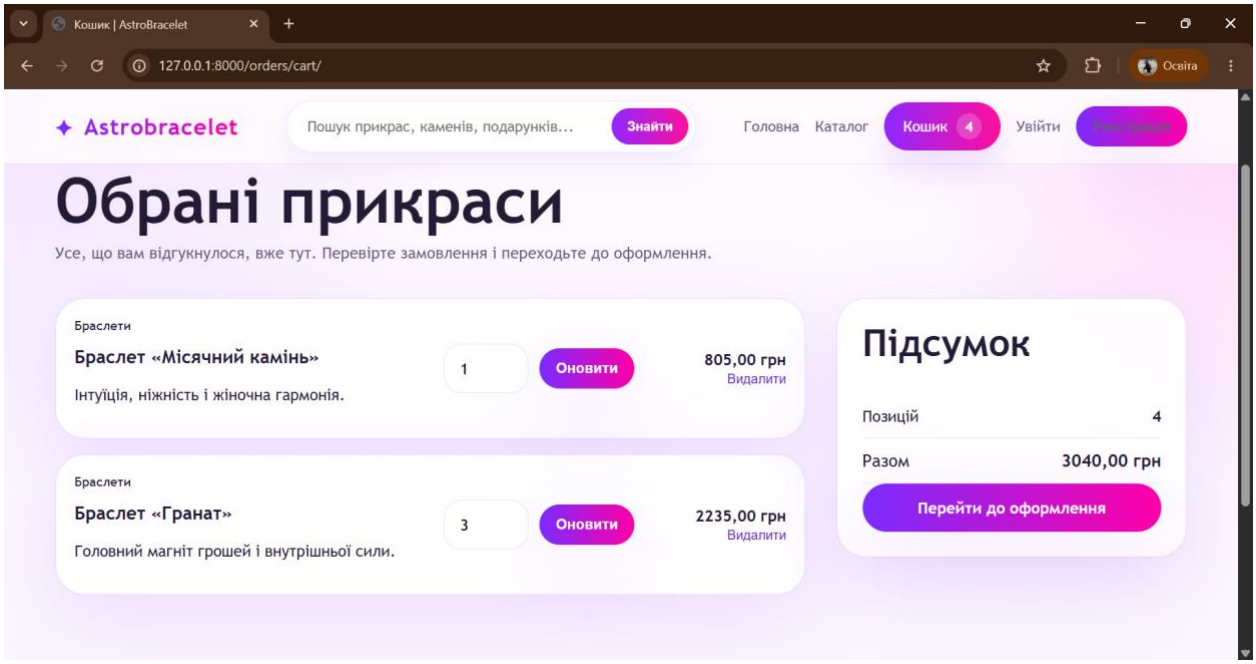

Рис. 4.2 – Перевірка роботи кошика

Наступним важливим етапом стало тестування оформлення замовлення. У межах цього сценарію перевірялося введення контактних даних користувача, вибір способу доставки, створення замовлення в базі даних та формування його позицій. Також оцінювалася правильність передавання загальної суми замовлення до платіжного модуля. За результатами перевірки встановлено, що замовлення формується коректно, а всі необхідні дані зберігаються у відповідних таблицях бази даних.

Важливим елементом тестування була перевірка модуля оплати. У цьому випадку перевірялася передача параметрів платежу до WayForPay, створення або оновлення платіжного запису в системі, зміна статусу після отримання результату операції, а також поведінка системи у випадку неуспішного або незавершеного платежу. Було встановлено, що система коректно взаємодіє з платіжним сервісом та належним чином синхронізує статуси замовлення і платежу.

На рис. 4.3 приклад перевірки сторінки оплати

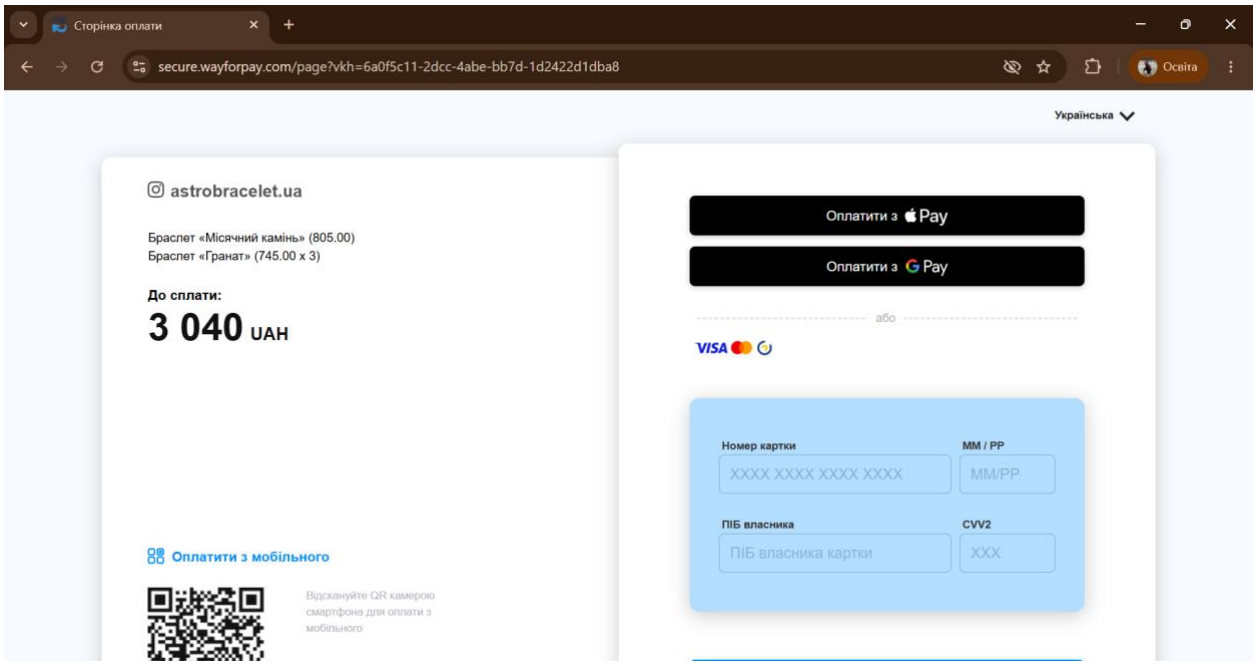

Рис. 4.3 – Перевірка модуля онлайн-оплати

Окремо було перевірено адміністративну частину системи. У межах цього тестування оцінювалася можливість додавання та редагування товарів,

зміни характеристик прикрас, перегляду замовлень, контролю платежів і загального керування вмістом каталогу. Результати перевірки показали, що адміністративна панель забезпечує коректне керування основними сутностями системи та може використовуватися для повсякденної роботи з магазином.

Отже, проведене тестування показало, що вебсистема Astrobracelet коректно реалізує основні функціональні можливості, передбачені постановкою завдання. Публічна частина забезпечує зручний перегляд каталогу, підбір прикрас, оформлення замовлення та оплату, а адміністративна частина надає всі необхідні засоби для керування товарами, замовленнями та платежами. Це дозволяє зробити висновок про працездатність системи та її готовність до подальшого використання й розвитку.

4.2 Вимоги до апаратного та програмного забезпечення

Вимоги до апаратного та програмного забезпечення визначають умови, за яких вебсистема Astrobracelet може коректно функціонувати як на стороні користувача, так і на стороні сервера. Дотримання цих вимог забезпечує стабільну роботу системи, належну продуктивність, коректне відображення інтерфейсу та надійну взаємодію з базою даних і зовнішніми сервісами.

Оскільки система побудована за клієнт-серверною архітектурою, вимоги до її функціонування доцільно розглядати окремо для клієнтської та серверної частин. На рис. 4.4 наведено діаграму розгортання вебсистеми Astrobracelet, яка відображає основні вузли системи, середовища виконання та зв’язки між ними.

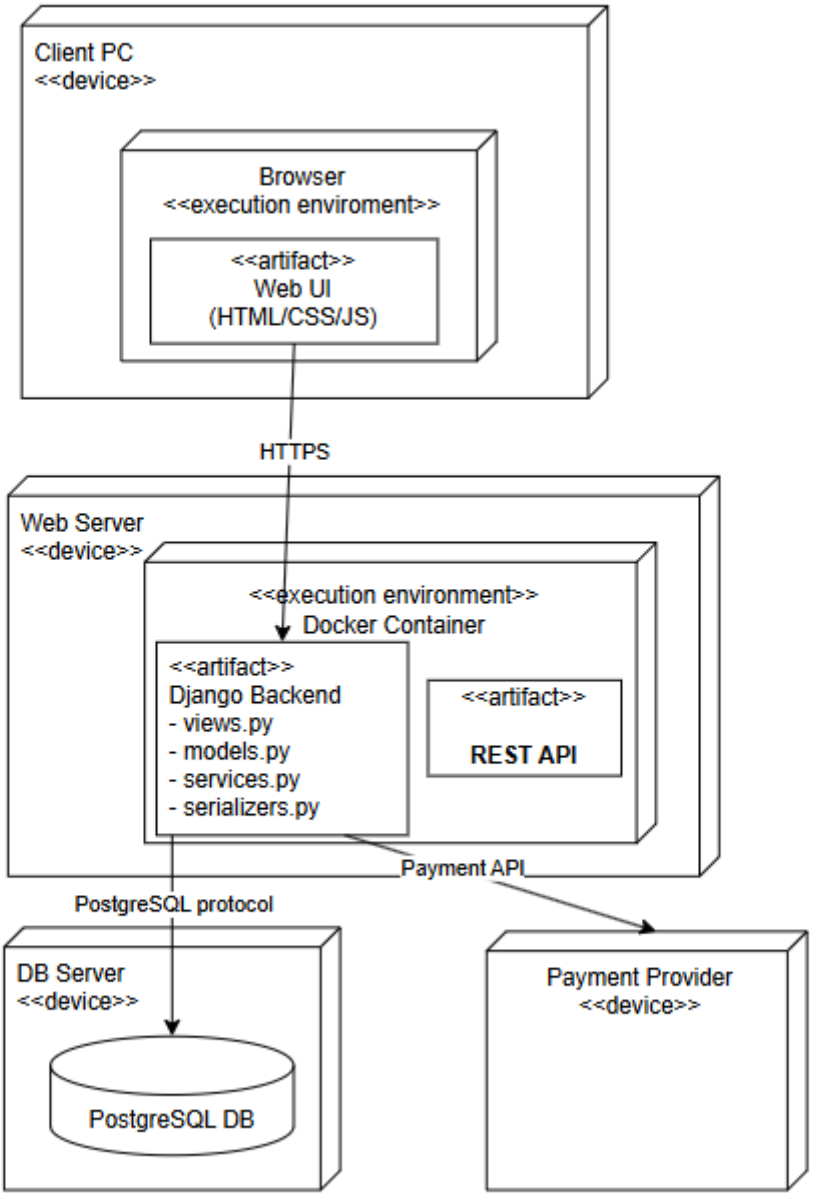

Рис. 4.4 – Діаграма розгортання вебсистеми Astrobracelet

Для клієнтської частини достатньо наявності персонального комп’ютера, ноутбука, планшета або сучасного смартфона з доступом до мережі Інтернет. Мінімальні апаратні вимоги для користувача є помірними: процесор середнього рівня, не менше 4 ГБ оперативної пам’яті, екран із роздільною здатністю не нижче 1280×720 та сучасний веббраузер. Для більш комфортної роботи доцільним є використання пристрою з 8 ГБ оперативної

пам’яті, стабільним швидкісним інтернет-з’єднанням і дисплеєм із вищою роздільною здатністю. Оскільки система є веборієнтованою, на клієнтській стороні не потрібне встановлення окремого спеціалізованого програмного забезпечення, окрім браузера.

Серверна частина висуває вищі вимоги, оскільки саме вона забезпечує обробку запитів, виконання бізнес-логіки, роботу з базою даних, збереження медіафайлів та взаємодію з платіжним сервісом. Для локального запуску або демонстрації системи достатньо комп’ютера з сучасним процесором, не менше 8 ГБ оперативної пам’яті та достатнім обсягом дискового простору для збереження проєкту, бази даних і статичних ресурсів. Для більш стабільної роботи в умовах реального використання бажано мати сервер або віртуальне середовище з 16 ГБ оперативної пам’яті, SSD-накопичувачем та стабільним мережевим підключенням.

Важливими є також мережеві вимоги. Для клієнтської частини необхідне стабільне підключення до Інтернету, достатнє для завантаження сторінок, зображень товарів і переходу до платіжного сервісу. Для серверної частини потрібен постійний доступ до мережі, оскільки система взаємодіє з базою даних, обслуговує HTTP-запити та приймає відповіді від зовнішньої платіжної системи. Особливо це важливо під час обробки callback-запитів після оплати.

З програмного боку клієнтська частина повинна бути сумісною з сучасними веббраузерами. Коректна робота системи забезпечується в останніх версіях Google Chrome, Mozilla Firefox, Microsoft Edge та Safari. Саме ці браузери підтримують сучасні стандарти HTML5, CSS3 і JavaScript, які використовуються в реалізації інтерфейсу.

Для серверної частини необхідна операційна система, що підтримує Python, PostgreSQL і Django. Найбільш придатними для розгортання є сучасні версії Linux, зокрема Ubuntu або Debian, однак локальний запуск також

можливий у середовищі Windows. До обов’язкових програмних складових належать Python, фреймворк Django, система управління базами даних PostgreSQL, бібліотеки проєкту, зазначені у файлі залежностей, а також засоби для роботи з платіжним сервісом WayForPay. Якщо система запускається в контейнеризованому середовищі, додатково необхідні Docker і Docker Compose.

Таким чином, вимоги до апаратного та програмного забезпечення для вебсистеми Astrobracelet є помірними й цілком реалістичними для навчального та прикладного вебпроєкту. Система може бути використана як у локальному середовищі для демонстрації та тестування, так і на серверній інфраструктурі для подальшого практичного застосування.

4.3 Склад інсталяційного пакету

Склад інсталяційного пакету визначає набір файлів і компонентів, необхідних для розгортання, налаштування та запуску системи. Для вебсистеми Astrobracelet інсталяційний пакет охоплює не лише вихідний код застосунку, а й конфігураційні файли, залежності, міграції бази даних, статичні ресурси та засоби контейнеризації.

Основу інсталяційного пакету становить програмний код Django-проєкту, який включає основні модулі системи: products, orders, payments, users та api. Саме в цих модулях реалізовано логіку роботи каталогу, кошика, оформлення замовлення, оплати, взаємодії з користувачами та програмного інтерфейсу обміну даними.

До складу пакету входять шаблони сторінок, статичні файли стилів і клієнтських сценаріїв, а також медіаресурси, необхідні для коректного відображення інтерфейсу. Окрему роль відіграють міграції Django, які забезпечують створення та оновлення структури бази даних відповідно до програмної моделі.

Важливими елементами інсталяційного пакету є конфігураційні файли середовища. У них задаються параметри доступу до бази даних, секретний ключ застосунку, службові налаштування, а також параметри інтеграції з платіжною системою WayForPay. Завдяки цьому система може бути адаптована до різних середовищ запуску без внесення змін у вихідний код.

Окреме місце в інсталяційному пакеті займають файли контейнеризації, зокрема Dockerfile, docker-compose.yml та допоміжні сценарії запуску. Їх використання дозволяє швидко розгорнути систему разом із базою даних PostgreSQL у стандартизованому середовищі, що спрощує тестування, демонстрацію та подальше перенесення проєкту на інший сервер.

До пакету також входить файл залежностей Python, який використовується для встановлення необхідних бібліотек і відтворення робочого програмного середовища. Це дає змогу уникнути проблем із несумісністю версій та забезпечує коректний запуск усіх модулів системи.

На рис. 4.5 подана узагальнена структура інсталяційного пакету або фрагмент структури проєкту, що демонструє його основні складові.

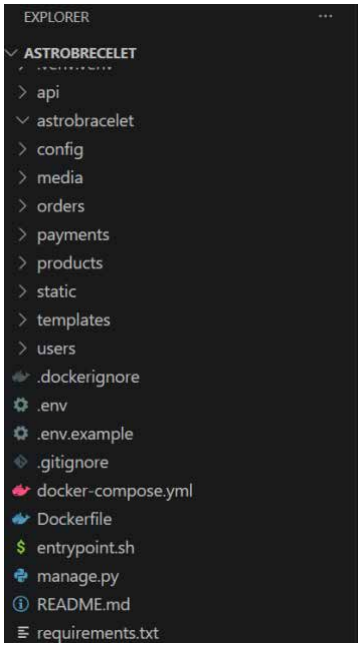

Рис. 4.5 – Основні складові інсталяційного пакету вебсистеми Astrobracelet

Отже, інсталяційний пакет вебсистеми Astrobracelet містить усі необхідні компоненти для повноцінного розгортання та запуску системи: вихідний код, конфігураційні файли, залежності, міграції, статичні ресурси та Docker-конфігурацію. Такий склад забезпечує зручність розгортання та робить систему придатною як для локального тестування, так і для подальшого використання у серверному середовищі.

4.4 Аналіз отриманих результатів

Завершальним етапом роботи є аналіз результатів, отриманих у процесі проєктування, програмної реалізації та тестування вебсистеми Astrobracelet. Такий аналіз дозволяє оцінити, наскільки розроблене програмне забезпечення відповідає поставленим цілям, чи вдалося реалізувати основні функціональні вимоги та якою мірою система придатна до подальшого практичного використання.

У результаті виконаної роботи було створено вебсистему продажу прикрас, яка підтримує повний цикл взаємодії користувача з інтернет-магазином. Система дозволяє переглядати каталог прикрас, застосовувати фільтрацію за змістовими характеристиками, переглядати сторінки окремих товарів, формувати кошик, оформлювати замовлення та переходити до онлайн-оплати. Це означає, що основна мета розробки була досягнута: створено функціональний вебзастосунок, який поєднує можливості електронної комерції з елементами персоналізованого підбору товарів.

Одним із найважливіших результатів є реалізація механізму персоналізованого підбору прикрас. На відміну від стандартного підходу, де товари подаються переважно як звичайні позиції каталогу, у системі Astrobracelet прикраса розглядається як об’єкт, що має індивідуальні смислові характеристики. Завдяки фільтрації за знаком зодіаку, каменем, кольором і символічним запитом користувач отримує можливість обирати товар не лише за зовнішнім виглядом або ціною, а й за особистісними критеріями. Це

підвищує цінність системи з точки зору користувацького досвіду та відрізняє її від типових інтернет-магазинів.

Важливим результатом є також реалізація адміністративної частини, яка забезпечує керування товарами, категоріями, характеристиками, замовленнями та платежами. Наявність такої частини робить систему придатною не лише як демонстраційний навчальний проєкт, а і як основу для подальшого практичного використання в межах бренду Astrobracelet. Адміністратор отримує можливість змінювати вміст магазину, оновлювати асортимент і контролювати процес обробки замовлень без необхідності втручання у програмний код.

З технічної точки зору позитивним результатом є використання сучасного й узгодженого набору технологій. Поєднання Django, PostgreSQL, HTML, CSS, JavaScript, WayForPay і Docker дозволило побудувати систему з чіткою клієнт-серверною архітектурою, структурованою базою даних та можливістю подальшого масштабування. Модульна побудова коду також є важливим результатом, оскільки вона спрощує підтримку системи та створює основу для розширення функціоналу в майбутньому.

Оцінюючи отримані результати, слід відзначити, що система успішно реалізує ті функції, які були визначені в постановці завдання. Працює перегляд каталогу, сторінки товарів, механізм фільтрації, кошик, оформлення замовлення, онлайн-оплата та адміністративне керування. Це свідчить про те, що вебсистема не є лише концептуальною моделлю, а представляє собою цілісне прикладне програмне рішення.

Разом із цим аналіз результатів дає змогу визначити напрями подальшого вдосконалення системи. У перспективі проєкт може бути розширений за рахунок створення особистого кабінету покупця, історії замовлень, системи відгуків, глибшої аналітики для адміністратора, додаткових способів доставки та інтеграції з іншими платіжними сервісами.

Можливим також є розвиток механізму рекомендацій, щоб персоналізований підбір прикрас став ще точнішим і зручнішим для користувача.

Таким чином, отримані результати підтверджують, що розроблена вебсистема Astrobracelet відповідає поставленим вимогам і може розглядатися як завершене прикладне рішення для онлайн-продажу прикрас. Вона поєднує функціональність сучасного інтернет-магазину, засоби персоналізованого підбору товарів, інтеграцію з платіжним сервісом та адміністративні інструменти керування, що робить її перспективною як для навчального використання, так і для подальшого практичного впровадження.

ВИСНОВКИ

У ході виконання бакалаврської кваліфікаційної роботи на тему «Програмне забезпечення для веборієнтованої системи продажу прикрас» було проведено аналіз предметної області онлайн-продажу прикрас та визначено її основні особливості. Установлено, що на відміну від багатьох інших напрямів електронної комерції, вибір прикрас значною мірою залежить не лише від вартості чи зовнішнього вигляду, а й від емоційного сприйняття, символічного змісту, кольору, каменю та індивідуального запиту користувача. Це зумовило необхідність створення вебсистеми, яка поєднує класичну функціональність інтернет-магазину з механізмами персоналізованого підбору товарів.

У процесі роботи було виконано моделювання предметної області, побудовано діаграми прецедентів, послідовності та діяльності, що дозволило формалізувати ролі користувачів, структуру взаємодії із системою та логіку проходження основних бізнес-процесів. На основі цього було спроєктовано інформаційне забезпечення системи, побудовано логічну модель даних у вигляді ER-діаграми та реалізовано фізичну структуру бази даних у середовищі PostgreSQL.

У результаті аналізу засобів реалізації було обрано технологічний стек, до якого увійшли Python, Django, PostgreSQL, HTML, CSS, JavaScript, Docker та платіжний сервіс WayForPay. Такий набір інструментів дозволив побудувати сучасну клієнт-серверну вебсистему з чіткою архітектурою, структурованою базою даних, адаптивним інтерфейсом і підтримкою онлайн-оплати.

У межах програмної реалізації було створено основні модулі системи, що забезпечують роботу каталогу, фільтрації, сторінки товару, кошика, оформлення замовлення, обробки платежів та адміністративного керування. Особливу увагу приділено механізму персоналізованого підбору прикрас за

знаком зодіаку, каменем, кольором і символічним запитом користувача, що є однією з ключових переваг розробленої системи.

Проведене тестування підтвердило працездатність системи та її відповідність основним функціональним і нефункціональним вимогам. Було перевірено роботу каталогу, механізму фільтрації, кошика, оформлення замовлення, оплати та адміністративної панелі. За результатами тестування встановлено, що система коректно виконує передбачені функції та може бути використана як основа для подальшого практичного впровадження.

Отже, поставленої мети бакалаврської кваліфікаційної роботи досягнуто. Розроблена вебсистема продажу прикрас Astrobracelet є завершеним прикладним програмним рішенням, яке поєднує функціональність сучасного інтернет-магазину, засоби персоналізованого підбору товарів, інтеграцію з платіжним сервісом та адміністративні інструменти керування. Отримані результати можуть бути використані як для подальшого розвитку реального онлайн-магазину, так і як основа для розширення функціональності системи в майбутньому.

.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Модель «сутність – зв’язок». Основні поняття моделі «сутність – зв’язок»: сутності, зв’язки, атрибути та їх класифікація. URL: https://wiki.cusu.edu.ua/index.php/Модель_«сутність_–_зв’язок»._Основні_поняття_моделі_«сутність_–_зв’язок»:_сутності,_зв’язки,_атрибути_та_їх_класифікація (дата звернення: 19.05.2026).
2. Діаграма прецедентів. URL: https://www.wikiwand.com/uk/Діаграма_прецедентів (дата звернення: 19.05.2026).
3. Діаграми сценаріїв. URL: https://web.posibnyky.vntu.edu.ua/fksa/4bevz_proektuvannya_programnyh_zasobiv_system_upravlinnya/53-9.htm (дата звернення: 19.05.2026).
4. Діаграма діяльності в UML: символ, компоненти та приклад. URL: https://www.guru99.com/uk/uml-activity-diagram.html (дата звернення: 19.05.2026).
5. Models | Django documentation. URL: https://docs.djangoproject.com/en/stable/topics/db/models/ (дата звернення: 19.05.2026).
6. Making queries | Django documentation. URL: https://docs.djangoproject.com/en/stable/topics/db/queries/ (дата звернення: 19.05.2026).
7. The Django admin site | Django documentation. URL: https://docs.djangoproject.com/en/stable/ref/contrib/admin/ (дата звернення: 19.05.2026).
8. Customizing authentication in Django | Django documentation. URL: https://docs.djangoproject.com/en/stable/topics/auth/customizing/ (дата звернення: 19.05.2026).

9. Templates | Django documentation. URL: https://docs.djangoproject.com/en/stable/topics/templates/ (дата звернення: 19.05.2026).
10. PostgreSQL: CREATE TABLE. URL: https://www.postgresql.org/docs/current/sql-createtable.html (дата звернення: 19.05.2026).
11. PostgreSQL: Constraints. URL: https://www.postgresql.org/docs/current/ddl-constraints.html (дата звернення: 19.05.2026).
12. WayForPay Documentation. URL: https://wiki.wayforpay.com/en (дата звернення: 19.05.2026).
13. HTML reference | MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/HTML/Reference (дата звернення: 19.05.2026).
14. CSS reference | MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/CSS/Reference (дата звернення: 19.05.2026).
15. JavaScript Guide | MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (дата звернення: 19.05.2026).
16. Національний університет біоресурсів і природокористування України. URL: https://elearn.nubip.edu.ua (дата звернення: 15.05.2024).

Додаток А

Схема фізичної структури бази даних:

Рис. А.1 – Схема фізичної структури бази даних вебсистеми Astrobracelet у PostgreSQL

Додаток Б

Блок-схеми алгоритмів оформлення замовлення та проведення онлайн-оплати:

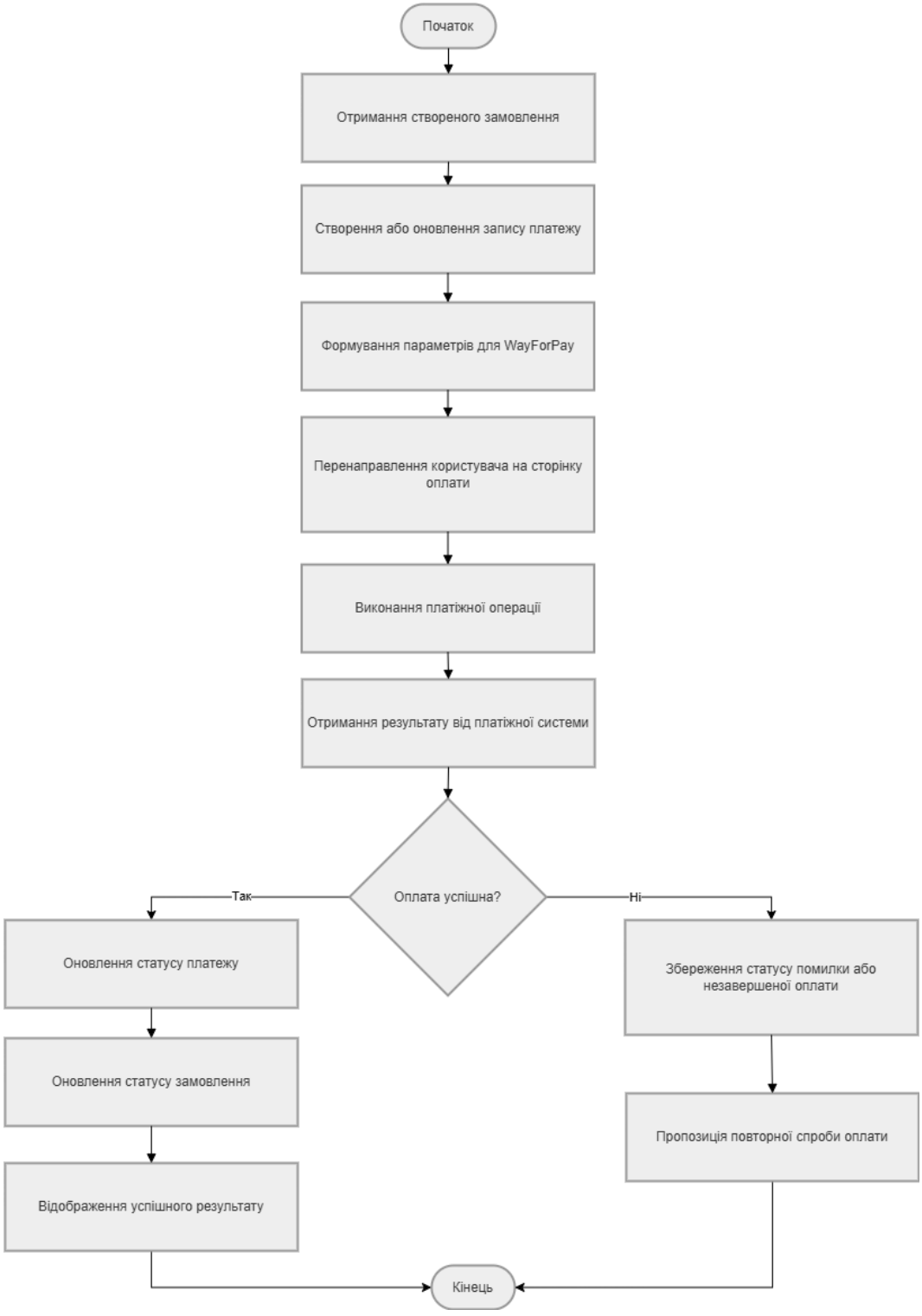

Рис. Б.1 – Блок-схема алгоритму оформлення замовлення

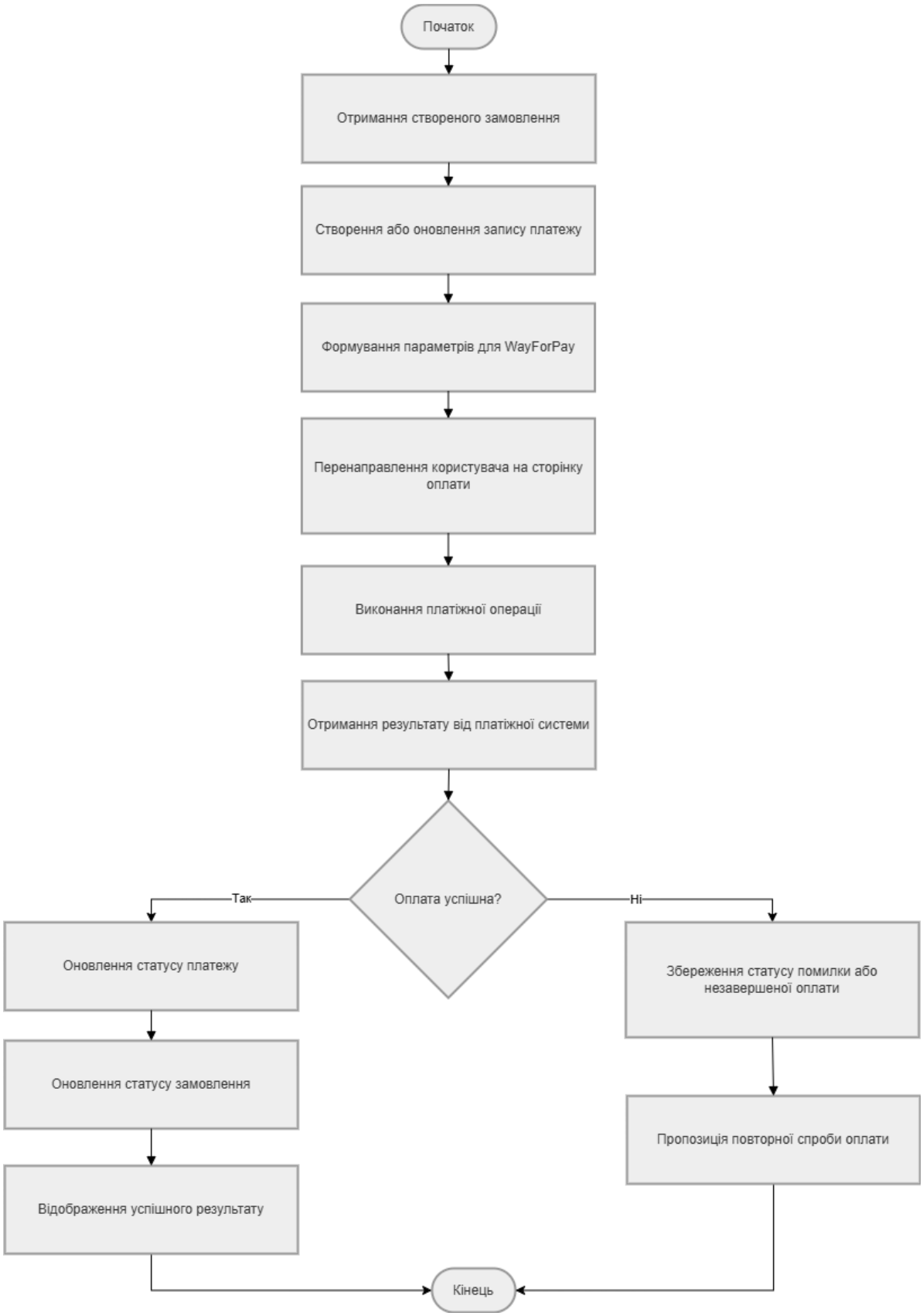

Рис. Б.2 – Блок-схема алгоритму проведення онлайн-оплати

Додаток В

Лістинг В.1 Фрагмент програмної реалізації моделі платежу:

```
from django.db import models
from orders.models import Order

class Payment(models.Model):
    STATUS_CHOICES = [
        ("pending", "Очікує"),
        ("paid", "Оплачено"),
        ("failed", "Помилка"),
    ]

    order = models.OneToOneField(Order, on_delete=models.CASCADE, related_name="payment")
    provider = models.CharField(max_length=120, default="WayForPay")
    transaction_id = models.CharField(max_length=120, unique=True)
    amount = models.DecimalField(max_digits=10, decimal_places=2)
    status = models.CharField(max_length=20, choices=STATUS_CHOICES, default="pending")
    wayforpay_status = models.CharField(max_length=40, blank=True)
    auth_code = models.CharField(max_length=40, blank=True)
    card_pan = models.CharField(max_length=32, blank=True)
    payment_system = models.CharField(max_length=40, blank=True)
    reason = models.CharField(max_length=255, blank=True)
    reason_code = models.CharField(max_length=20, blank=True)
    callback_payload = models.JSONField(default=dict, blank=True)
    paid_at = models.DateTimeField(null=True, blank=True)
```

```
created_at = models.DateTimeField(auto_now_add=True)

def __str__(self) -> str:
    return f"{self.provider} - {self.transaction_id}"
```

Додаток Г

Лістинг Г.1 Фрагмент програмної реалізації моделей замовлення

```
class Order(models.Model):
    STATUS_CHOICES = [
        ("new", "Нове"),
        ("paid", "Оплачено"),
        ("shipped", "Відправлено"),
        ("completed", "Завершено"),
    ]
    DELIVERY_METHOD_CHOICES = [
        ("nova_poshta_branch", "Нова пошта: відділення"),
        ("nova_poshta_locker", "Нова пошта: поштомат"),
    ]

    user = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
        related_name="orders",
        null=True,
        blank=True,
    )
    full_name = models.CharField(max_length=180)
    email = models.EmailField()
    phone = models.CharField(max_length=30)
    city = models.CharField(max_length=120)
    delivery_method = models.CharField(max_length=30,
choices=DELIVERY_METHOD_CHOICES, default="nova_poshta_branch")
```

```
        address = models.CharField(max_length=255)
        comment = models.TextField(blank=True)
        status = models.CharField(max_length=20, choices=STATUS_CHOICES,
default="new")
        total_amount = models.DecimalField(max_digits=10, decimal_places=2,
default=0)
        created_at = models.DateTimeField(auto_now_add=True)
```

Додаток Д

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Литовка Альбіна Сергіївна, здобувачка НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проєкт) на тему «Програмне забезпечення для веборієнтованої системи з продажу прикрас» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
    - [] інструменти генеративного ШІ не використовувалися.
    - [✓] інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL, _____інші (вказати назву)) були використані для:
        - [ ] перекладу іншомовних джерел;
        - [✓] стилістичного редагування та перевірки граматики;
        - [✓] допомоги у написанні/відлагодженні програмного коду;
        - [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до

скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«20» травня 2026 р. _____ **Альбіна ЛИТОВКА**
(підпис)