Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Єріс Анастасії Євгенівни*
(ПІБ)

академічної групи *121-20-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка інтернет-магазину з продажу жіночого одягу з використанням бібліотеки React*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Спірінцев В.В.*			
розділів:				
спеціальний	*доц. Спірінцев В.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Мартиненко А.А.*			

Дніпро
2024

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-20-1* *Єріс Анастасії Євгенівни*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка інтернет-магазину з продажу жіночого одягу з використанням бібліотеки React*

затверджена наказом ректора НТУ «ДП» від *23.05.2024 р.* № *469-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***14.06.2024 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***20.06.2024 р.***

Завдання видав _____ *доц. Спірінцев В.В.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Єріс А.Є.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2024 р.*

Термін подання кваліфікаційної роботи до ЕК: *24.06.2024 р.*

РЕФЕРАТ

Пояснювальна записка: 84 с., 45 рис., 3 дод., 30 джерел.

Об'єкт розробки: інтернет-магазину з продажу жіночого одягу.

Мета кваліфікаційної роботи: розробка інтернет-магазину з продажу жіночого одягу з використанням бібліотеки React.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточняється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформу для розробки, виконано проектування і розробку веб-орієнтованої інформаційної системи, описана робота системи, алгоритм і структура його функціонування, а також виклик та завантаження додатку, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню додатку та розраховано час на його створення.

Практичне значення полягає у створенні веб-орієнтованої інформаційної системи, що забезпечує автоматизацію процесів придбання та продажу товарів, дозволяє користувачам швидко та зручно знаходити, обирати та купувати товари, завдяки інтуїтивному інтерфейсу та вдосконаленій функціональності пошуку і фільтрації, забезпечує гнучке управління контентом.

Актуальність розробки інтернет-магазину не викликає сумніву та визначається зростаючою потребою на дистанційний спосіб придбання товарів, який дозволяє клієнтам економити зусилля та час, а життя в умовах всесвітньої пандемії та війни особливо підкреслили і довели необхідність розвитку електронної комерції в Україні.

Список ключових слів: ІНФОРМАЦІЙНА СИСТЕМА, ІНТЕРНЕТ-МАГАЗИН, БАЗА ДАНИХ, HTML, CSS, JAVASCRIPT, TYPESCRIPT, REACT, NODE, STRAPI, VITE.

ABSTRACT

Explanatory note: 84 pp., 45 fig., 3 extra, 30 sources.

The object of development: online store for the sale of women's clothing.

The purpose of the qualification work: development of an online store for the sale of women's clothing using the React library.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the scope of its application substantiates the relevance of the topic and clarifies the formulation of the problem.

In the first chapter, the subject area is analyzed, the relevance of the task and the purpose of development are determined, the statement of the problem is formulated, the requirements for software implementation, technologies and software are indicated.

In the second section, the available solutions are analyzed, a platform for development is selected, the design and development of a web-oriented information system is carried out, the operation of the system, the algorithm and structure of its functioning, as well as the call and loading of the application are described, the input and output data are determined, the composition of the parameters of the technical means.

In the economic section, the labor intensity of the developed information system is determined, the cost of work on creating an application is calculated and the time for its creation is calculated.

The practical significance lies in the creation of a web-based information system, which will ensure automation of processes for purchasing and selling goods, allows to find goods quickly and conveniently, to choose and to buy products, due to the intuitive interface and to improved search and to filtering functionality, provides flexible content management.

The relevance of the development of online store for the sale of women's clothing is beyond doubt and determined by the growing need for a remote method of purchasing goods, which allows customers to save effort and time, and life in the conditions of a global pandemic and war especially emphasized and proved the need for the development of e-commerce in Ukraine.

List of keywords: INFORMATION SYSTEM, ONLINE STORE, DATABASE, HTML, CSS, JAVASCRIPT, REACT, NODE, STRAPI, VITE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ - Програмне забезпечення

ІКТ Інформаційно-комунікаційні технології

ІС - Інформаційна система

ОС - Операційна система

БД - База даних

СУБД - Система управління базами даних

CMS - Система управління контентом

CSS - Cascading Style Sheets

SQL - Structured Query Language

HTML - Hyper Text Markup Language

ВСТУП

Внаслідок стрімкого розвитку технологій в галузі електроніки та кібернетики суттєво прискорилися процеси інформатизації всіх сфер економіки. Це призвело до формування інформаційного суспільства, де інформаційно-комунікаційні технології досягають нового рівня розвитку. Зокрема, ІКТ забезпечують вирішення складних економічних завдань як для окремих бізнес-структур, так і на загальнодержавному рівні.

Електронна комерція є одним із ключових засобів підтримки процесів інформатизації в економіці. Вона дозволяє ефективно здійснювати комерційні операції, швидко адаптуватися до змін на ринку товарів і послуг, розширювати сфери впливу бізнесу та зміцнювати їх конкурентні переваги.

Цей технологічний прогрес докорінно змінив споживчу поведінку, створюючи нові можливості для покупців по всьому світу. Сьогодні для них практично не існує кордонів чи обмежень. Незалежно від місцезнаходження, покупець може замовити товар з будь-якої країни світу, а кур’єр доставить його прямо до дверей. Відкритість кордонів, доступність міжнародних оплат та стрімкий розвиток логістичних компаній значно розширюють можливості як для бізнесу, дозволяючи виходити на нові ринки та збільшувати обсяги продажів, так і для споживачів, які отримують широкий вибір товарів і послуг.

Актуальність розробки інформаційної системи для автоматизації діяльності підприємства в сфері електронної комерції є незаперечною та визначається зростаючою потребою на дистанційний спосіб придбання товарів, який дозволяє клієнтам економити зусилля та час, а життя в умовах всесвітньої пандемії та війни особливо підкреслили і довели необхідність розвитку електронної комерції в Україні.

Об'єктом дослідження є інтернет-магазину з продажу жіночого одягу.

Мета кваліфікаційної роботи: розробка інтернет-магазину з продажу жіночого одягу з використанням бібліотеки React.

Для вирішення поставленої мети необхідно вирішити наступні задачі:

– проаналізувати особливості електронної комерції в контексті розвитку економіки України;

– визначити основні вимоги до програмного продукту, що розробляється, зокрема вимоги до функціональних характеристик, інформаційної безпеки, складу та параметрів технічних засобів, інформаційної та програмної сумісності, змісту інформаційного наповнення системи;

– визначити функціональне призначення програми;

– визначити архітектуру, проаналізувати та обрати технології та мови програмування;

– розробити структуру програми та алгоритми її функціонування;

– ґрунтуючись на результатах попередніх пунктів розробити веб-додаток відповідно до поставленої задачі;

– виконати розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки.

Практичне значення полягає у створенні веб-орієнтованої інформаційної системи, що забезпечує автоматизацію процесів придбання та продажу товарів, дозволяє користувачам швидко та зручно знаходити, обирати та купувати товари, завдяки інтуїтивному інтерфейсу та пошуку і фільтрації, забезпечує гнучке управління контентом.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Предметною галуззю застосування веб-орієнтованої інформаційної системи, що забезпечує автоматизацію процесів придбання та продажу товарів є електронна комерція. Електронна комерція – комерційна взаємодія суб'єктів бізнесу з приводу купівлі-продажу товарів та послуг (матеріальних та інформаційних) з використанням інформаційних мереж (Internet, мережа стільникового зв'язку, внутрішні локальні мережі фірм) [1]. Вона поєднує цифрові та фізичні бізнес-моделі, що дозволяє компаніям розширювати свої ринки, знижувати операційні витрати та підвищувати ефективність взаємодії з клієнтами. У сучасних умовах, коли цифрові технології стрімко розвиваються, компанії повинні швидко адаптуватися до змін і використовувати автоматизацію для оптимізації своїх бізнес-процесів.

Розробка веб-орієнтованих інформаційних систем для автоматизації діяльності в електронній комерції передбачає створення комплексних рішень, що включають в себе інтегровані платформи для управління продажами, замовленнями, оплатами, складами, а також для забезпечення підтримки клієнтів і аналітики. Основним компонентом такої системи є онлайн-магазин, який надає покупцям можливість переглядати каталог товарів, додавати їх до кошика, оформлювати замовлення та здійснювати оплату. Важливим аспектом є система управління замовленнями, що забезпечує обробку і контроль замовлень, включаючи підтвердження, пакування, та відправлення.

Успішне функціонування таких систем вимагає ефективної автоматизації бізнес-процесів. Це особливо важливо у сучасному динамічному ринку, де зростаючі вимоги клієнтів до швидкості обслуговування, широкого вибору товарів і персоналізованого підходу диктують необхідність використання сучасних програмних рішень. Ці системи дозволяють покращити ефективність

операцій, зменшити кількість ручної праці, знизити ймовірність помилок і прискорити процеси обробки замовлень, управління складом та інші функції.

Веб-орієнтовані інформаційні системи мають декілька ключових переваг. Їхня доступність через інтернет забезпечує користувачам і персоналу можливість працювати з будь-якого пристрою, що має доступ до мережі, забезпечуючи гнучкість і зручність. Інтерактивність таких систем дозволяє користувачам отримувати динамічний і персоналізований досвід. Масштабованість і адаптивність веб-систем дозволяє легко розширювати їх функціональність і адаптуватися до нових вимог ринку і бізнесу. Безпека даних забезпечується за допомогою сучасних протоколів захисту, таких як шифрування і автентифікація, що гарантує високий рівень захисту конфіденційної інформації. Інтеграція з третіми сторонами, включаючи платіжні шлюзи, сервіси доставки і маркетингові інструменти, розширює функціональні можливості веб-систем.

Для досягнення ефективності та функціональності веб-орієнтованих інформаційних систем використовуються сучасні технології та архітектурні підходи. Мікросервісна архітектура дозволяє створювати систему з незалежних компонентів, кожен з яких виконує певну функцію, що полегшує її підтримку та оновлення. Сучасні JavaScript-фреймворки, такі як React, Angular, Vue.js, використовуються для створення інтерактивних і чуйних інтерфейсів користувача. Для обробки бізнес-логіки і управління даними застосовуються надійні серверні платформи, такі як Node.js, Django, Ruby on Rails. Зберігання та доступ до інформації забезпечуються реляційними і нереляційними базами даних, такими як MySQL, PostgreSQL, MongoDB та SQLite.

Таким чином, розробка програмного забезпечення веб-орієнтованої інформаційної системи для автоматизації діяльності компанії в сфері електронної комерції є важливою для досягнення стратегічних цілей бізнесу, підвищення ефективності операцій і поліпшення взаємодії з клієнтами. Це дозволяє компаніям залишатися конкурентоспроможними та адаптуватися до мінливих умов ринку, забезпечуючи зростання і розвиток у цифровому середовищі.

Повномасштабне вторгнення росії призвело до тяжких економічних наслідків в Україні. Але, попри відключення електроенергії та ракетні обстріли українські інтернет-магазини створюються та успішно працюють. Крім того, статистика говорить про те, що навесні 2023 року кількість онлайн-замовлень збільшилася майже на чверть, ніж було до початку повномасштабної війни, і продовжила зростати до кінця року. Це ще раз демонструє те, що бізнес в Україні продовжує розвиватися і нові підприємства створюються, незважаючи на тяжкі умови [2].

Бізнес-ніша продажу жіночого одягу на українському ринку є висококонкурентною. В просторі електронної комерції активно функціонує багато інтернет-магазинів цієї спеціалізації.

Переглянимо та порівняємо структуру популярних, вже існуючих в мережі магазинів жіночого одягу (рис. 1.1-1.7).

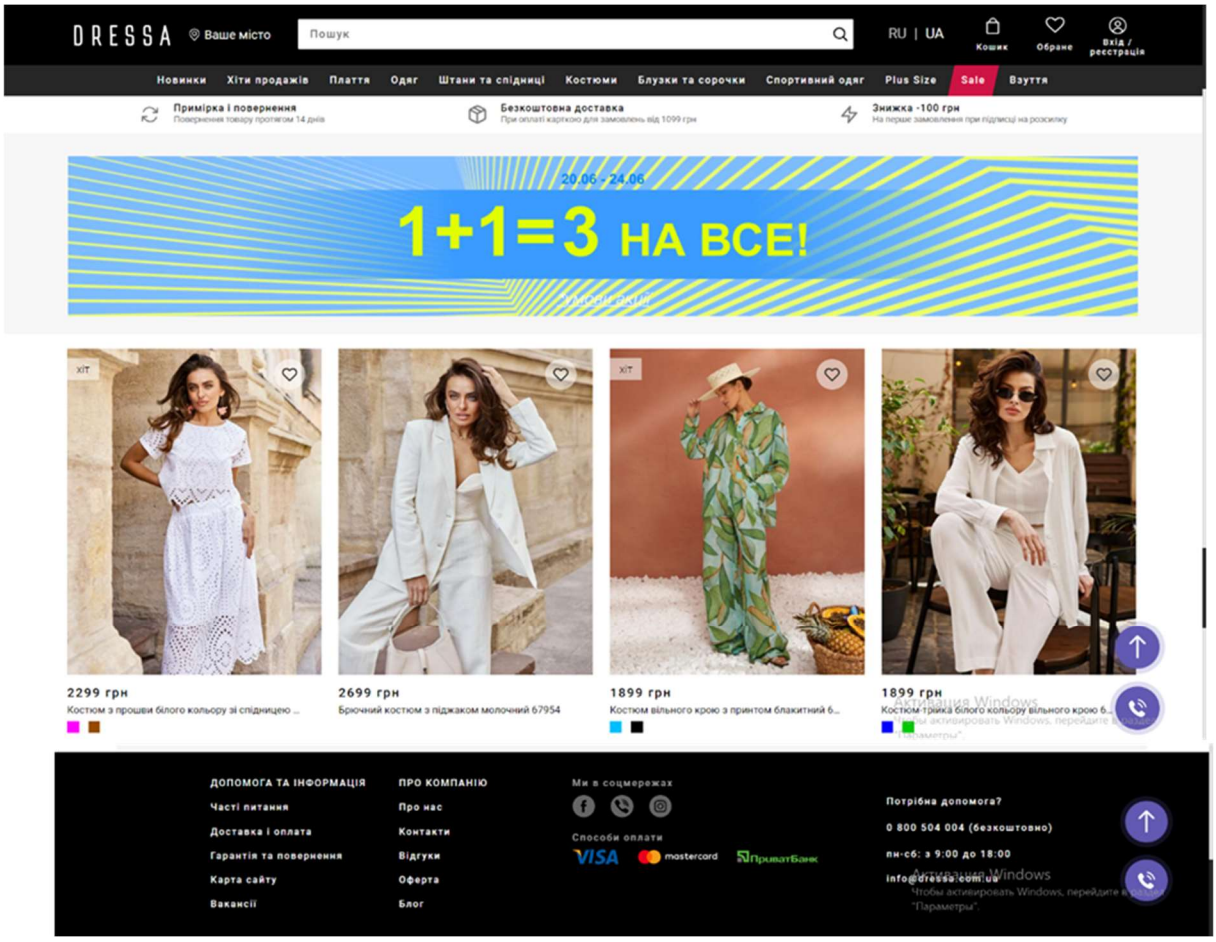

Рис. 1.1. Елементи головної сторінки інтернет-магазину «Dressa»

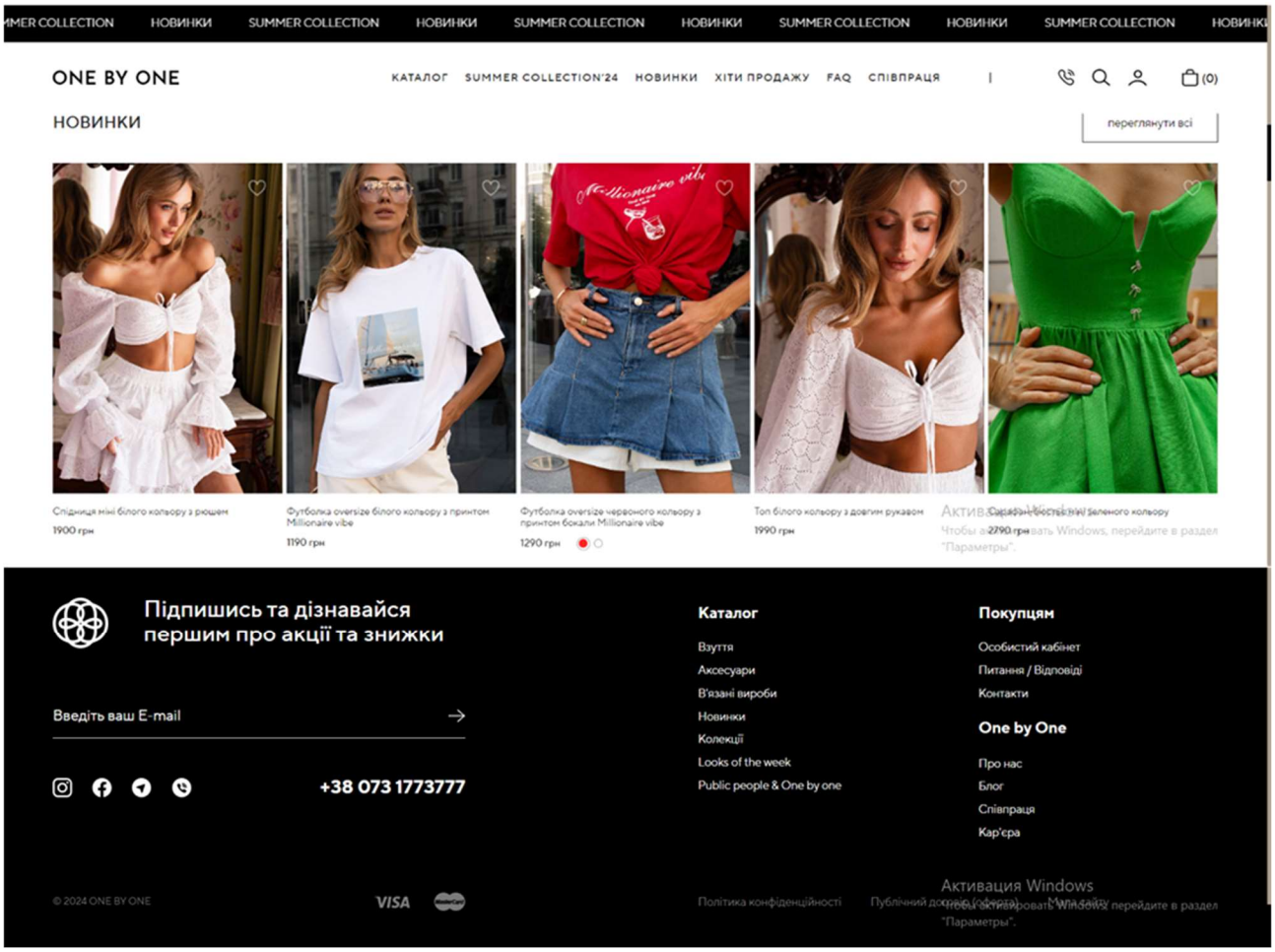

Рис. 1.2. Елементи головної сторінки інтернет-магазину «One by one»

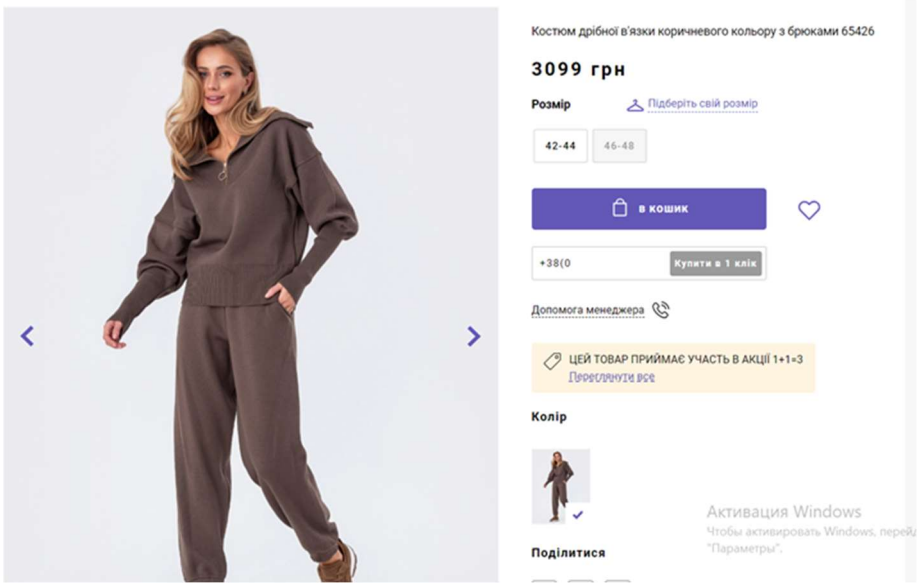

Рис. 1.3. Сторінка товару інтернет-магазину «Dressa»

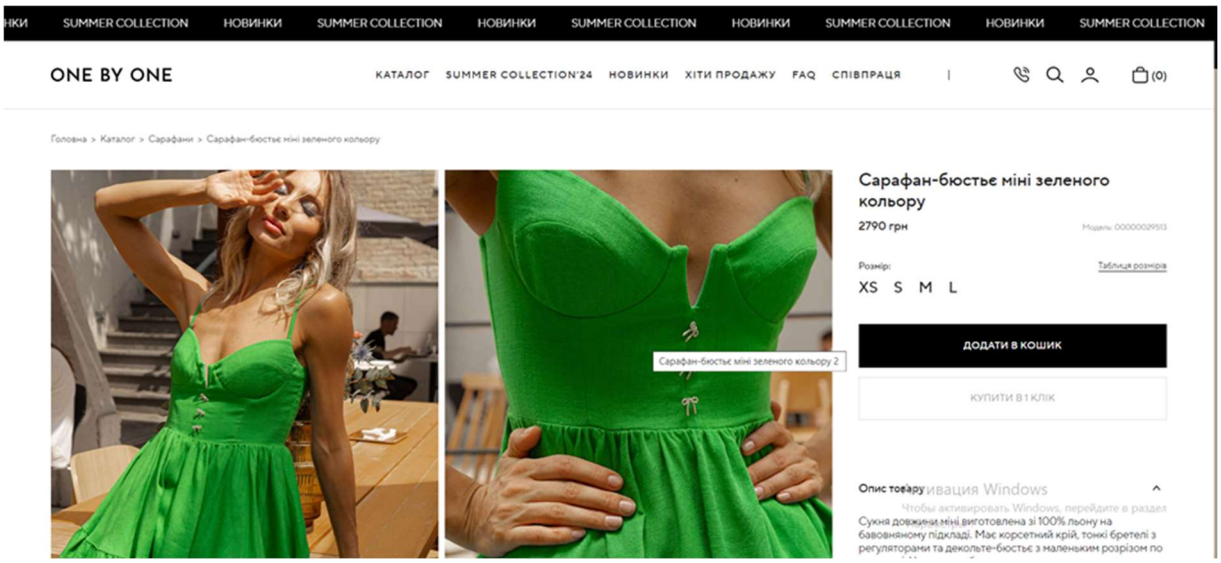

Рис. 1.4. Сторінка товару інтернет-магазину «One by one»

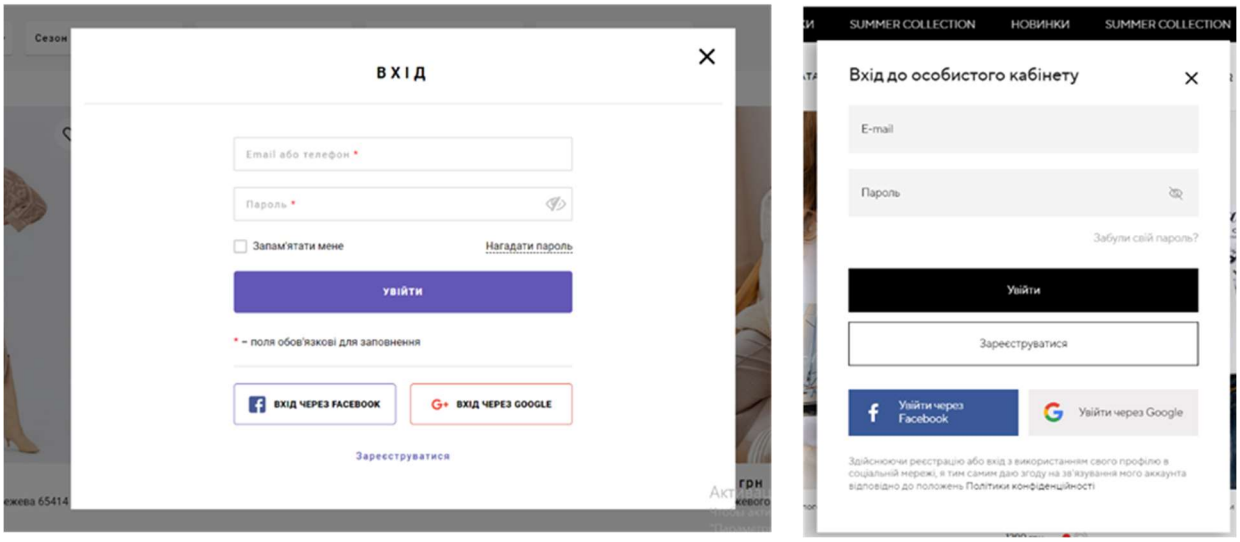

Рис. 1.5. Сторінки реєстрації користувача інтернет-магазинів «Dressa» та «One by one»

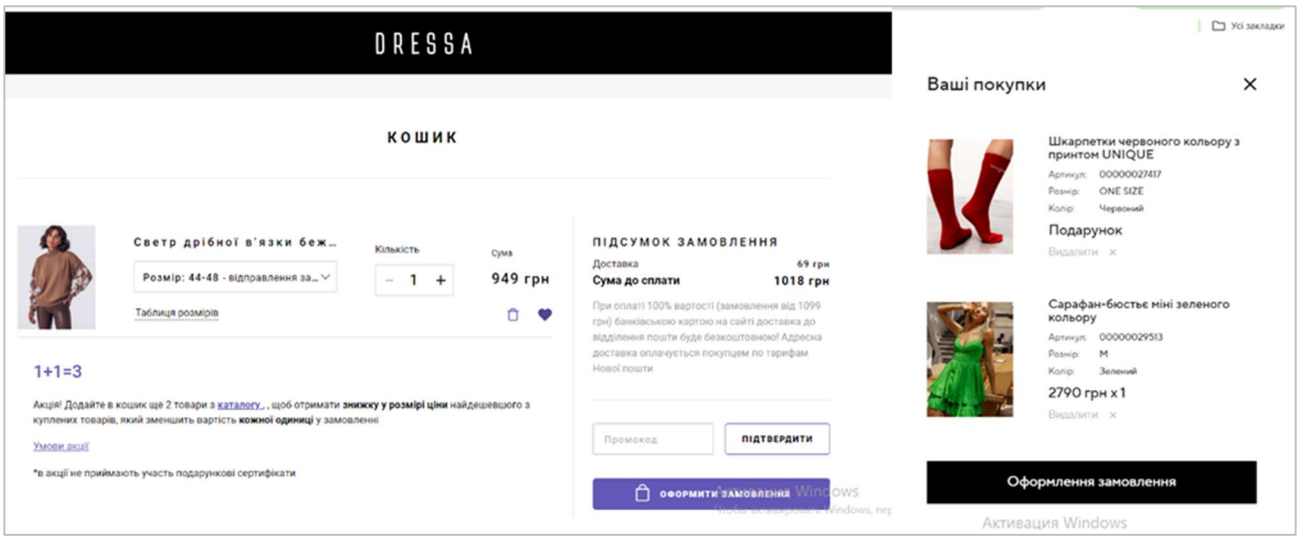

Рис. 1.6. «Кошик» інтернет-магазинів «Dressa» та «One by one»

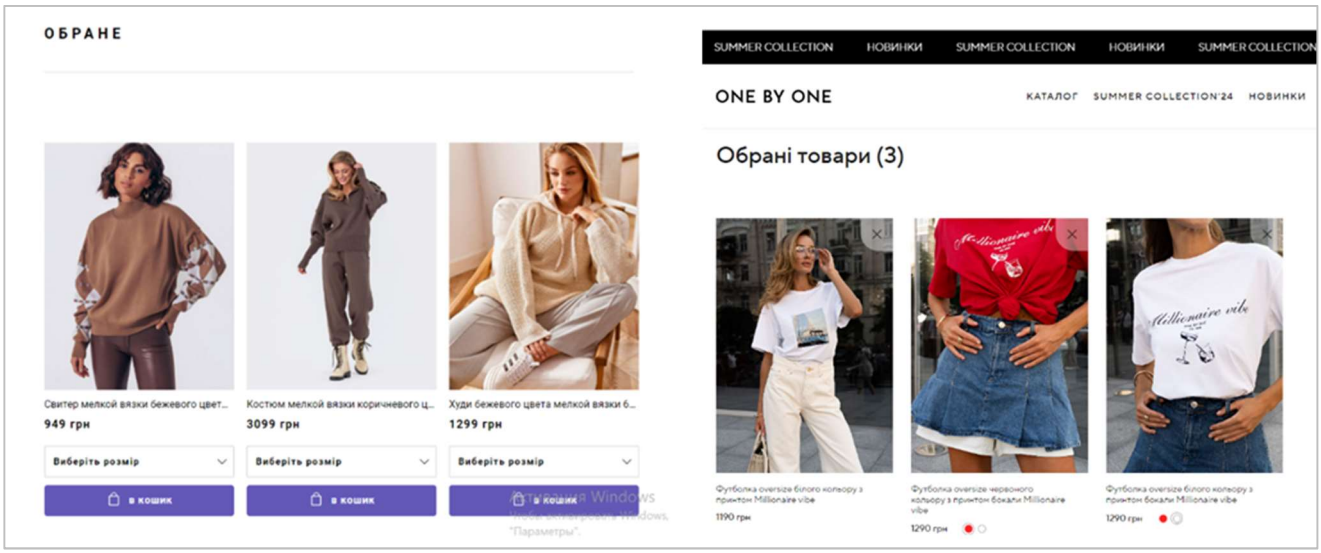

Рис. 1.7. Сторінка «Обране» інтернет-магазинів «Dressa» та «One by one»

Проаналізувавши структуру обраних магазинів, можна зазначити, що вони мають однакову структуру, основні частини якої це:

– головна сторінка;
– каталог товарів;
– окрема сторінка для кожного товару;
– вікно для реєстрації користувача;
– корзина та оформлення замовлення;
– сторінка з обраними товарами.

Головна сторінка містить такі елементи:

– Header, якій містить:

1) логотип;
2) пошуковий рядок;
3) посилання на кошик;
4) посилання на сторінку улюблених товарів;
5) посилання на авторизацію;

– каталог товарів з можливістю фільтрування та сортування;

– основна частина сторінка, яка наповнена зображеннями товарів з короткими відомостями про товар (назва, ціна);

– Footer, найнижчий елемент, в ньому міститься корисна інформація для відвідувачів магазину посилання на соціальні мережі, контактні дані, інформація про сайт і компанію та інше.

На сторінці товару міститься:

– заголовок;

– крупне фото товару;

– опис товару;

– наявні розміри;

– вартість;

– кнопка «Додати до кошика»;

– додавання до списку бажаного.

Така структура є звичною та комфортною для клієнта, тоді як запровадження кардинально нового інтерфейсу викликає у потенційного покупця розгубленість та почуття незручності. Тому при виконанні завдання кваліфікаційної роботи інтерфейс теж буде мати стандартний, інтуїтивно зрозумілий для клієнта інтерфейс.

Для створення позитивного клієнтського досвіду покупок навколо продукту, що продається пріоритетну роль в грає якісно розроблений сайт, якій вміщує в себе не тільки зручний та привабливий інтерфейс, вдалі візуальні рішення, а й має високу швидкість завантаження сторінки. Користувачі звикли

до швидкого серфінгу в мережі, тому негативно реагують навіть на невелику затримку. Якщо покупець відчуває себе незручно внаслідок очікування відгука сторінки, то з високою вірогідністю він вже не повернеться на цей сайт, навіть якщо придбав товар, що йому сподобався.

Довгий відгук сайту також негативно впливає на його місце в пошукових запитах. Якщо сайт буде уступати за швидкістю сайту-конкуренту (при приблизно однакових інших показниках), більш релевантним буде вважатися сайт конкурента.

Причиною низької швидкості можуть бути багато факторів, але найважливіші це: перевантаження сайту великими зображеннями, анімацією, недостатньо оптимізований код, нераціонально вибрані технології, що впливають на продуктивність.

Тому швидкість завантаження сайту потрібно закладати ще на етапі розробки. Для цього потрібно відшукати баланс між інформативністю сайту та його візуальним оформленням, а також використовувати сучасні технології та архітектурні підходи.

1.2. Призначення розробки та галузь застосування

Темою цієї кваліфікаційної роботи є «Розробка інтернет-магазину з продажу жіночого одягу з використанням бібліотеки React». Internet-магазин (електронний, віртуальний, e-shop) являє собою спеціалізований Web-сайт, який належить фірмі-товаровиробнику, торговій фірмі тощо та призначений для просування споживчих товарів на ринку, збільшення обсягів продажу, залучення нових покупців. При цьому розміщення споживацької інформації, замовлення товару і угода відбуваються там само, всередині мережі (на сайті інтернет-магазину). Товари, що пропонуються для продажу на цьому сайті знаходяться на складі торгівельного підприємства, що здійснює цю торгівлю.

Головною особливістю інтернет-магазину є повна автоматизація системи обробки замовлень, за рахунок чого можна працювати з кожним зареєстрованим клієнтом індивідуально.

Таким чином, сутність завдання кваліфікаційної роботи – створення спеціалізованого сайту, на якому буде розміщена інформація про товари з категорії «Жіночий одяг» з можливістю вибору та замовлення товару, повною автоматизацією системи обробки замовлень та який би підвищив ефективність роботи торгівельного підприємства, що здійснює цю торгівлю за рахунок діяльності в сфері електронної комерції.

1.3. Підстава для розробки

Підставами для розробки (виконанням кваліфікаційної роботи) є:

– освітня програма спеціальності 121 «Інженерія програмного забезпечення»;

– навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» № 469-с від 23.05.2024 р;

– завдання на кваліфікаційну роботу на тему «розробка інтернет-магазину з продажу жіночого одягу з використанням бібліотеки React».

1.4. Постановка завдання

Метою роботи цієї кваліфікаційної роботи є розробка програмного забезпечення інформаційної системи маркетингової діяльності підприємства для підвищення ефективності його роботи за рахунок автоматизації діяльності в сфері електронної комерції.

Призначення інформаційної системи – надання електронно-комерційних послуг з продажу жіночого одягу. Користувачами інформаційної системи буде переважно жіноча аудиторія віком віл 16 до 60 років.

Об'єктом, у ході управління яким можна використовувати систему, що розробляється, є торгівельне підприємство, що спеціалізується на продажу товарів з категорії «Жіночий одяг».

Тобто, завдання роботи – розробити інтернет-магазин для продажу жіночого одягу, який буде відповідати наступним вимогам:

– повинен мати сучасний, зручний та зрозумілий інтерфейс;

– мати каталог товарів;

– мати можливість фільтрування, сортування та пошуку товарів;

– можливість перегляду сторінки товару, на якій є збільшене зображення товару, опис моделі, можливість вибору розміру;

– мати корзину з переліком обраних для придбання товарів, з функцією вибору кількості товарів, розміру та виводом загальної суми покупки;

– можливість додавання товарів до списку обраних;

– можливість перегляду сторінки обраних товарів;

– мати високу швидкість завантаження контенту.

Інтернет-магазин буде розроблено з використанням бібліотеки React, що забезпечить високу продуктивність та інтерактивність користувацького інтерфейсу.

1.5. Вимоги до програми або програмного виробу

Вимоги до програмного забезпечення — набір вимог щодо властивостей, якості та функцій програмного забезпечення, що буде розроблено, або знаходиться у розробці [4].

Перш за все, програмний продукт повинен виконувати свою основну функцію, що включає обробку запитів клієнтів та формування відповідних результатів на основі цих запитів. Також він має забезпечувати роботу з базами даних клієнтів, співробітників і товарів. Ці функції повинні виконуватись із дотриманням вимог надійності, продуктивності, швидкості роботи та гнучкості архітектури.

Програмний продукт має бути орієнтований на користувача, задовольняючи його потреби, зокрема забезпечуючи максимальну зручність. Це передбачає легкість сприйняття, простоту у використанні, інтуїтивний інтерфейс і повну зрозумілість роботи. Усе це особливо важливо у сфері електронної комерції. Продукт також повинен бути економічно ефективним.

1.5.1. Вимоги до функціональних характеристик

До функціональних можливостей системи, що розроблюється, відносяться всі основні можливості та функції, які має підтримувати система, для забезпечення її стабільної роботи та відповідність очікуванням користувачів.

Ось основні функціональні вимоги, які повинні бути реалізовані:

– простий та інтуїтивно зрозумілий для користувача інтерфейс;

– дані повинні вводитися шляхом вводу інформації в поля інтерфейсу та відправлятися через запит користувача на сервер;

– ведення бази даних інтернет-магазину;

– інструменти керування магазином адміністратором;

– легкий пошук необхідних товарів;

– персональний кабінет користувачів;

– забезпечення великої швидкості роботи та малий час відгуку при виконанні операцій;

– сучасні технології для розробки.

1.5.2. Вимоги до інформаційної безпеки

Під інформаційною безпекою розуміють стан захищеності систем обробки і зберігання даних, при якому забезпечено конфіденційність, доступність і цілісність інформації, використання й розвиток в інтересах громадян або комплекс заходів, спрямованих на забезпечення захищеності інформації особи, суспільства і держави від несанкціонованого доступу, використання,

оприлюднення, руйнування, внесення змін, ознайомлення, перевірки запису чи знищення.

Для забезпечення надійного роботи та для запобігання некоректної роботи системи необхідно реалізувати:

– виведення повідомлень про помилки;

– можливість повторного введення даних;

– можливість безперервної роботи;

– платформна незалежність;

– забезпечення неушкодженого стану даних, що зберігаються в базі даних, у випадку відмови застосунку.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для стабільної роботи веб-орієнтованої інформаційної системи, повинні виконуватися зазначені нижче вимоги до технічних засобів.

Для клієнтської частини:

– процесор з тактовою частотою не менше 2.4 ГГц;

– оперативна пам'ять не менше 4 GB;

– 5 Гб вільного місця на жорсткому диску.

Для серверної частини:

– процесор з тактовою частотою не менш 2.4 ГГц;

– оперативна пам'ять не менше 4 GB;

– 10 Гб вільного місця на жорсткому диску.

Наведені технічні характеристики є рекомендованими. При використанні технічних засобів з характеристиками не нижче перелічених, програмний виріб буде працювати згідно з вимогами щодо швидкості обробки даних, безпеки та надійності.

1.5.4. Вимоги до інформаційної та програмної сумісності

Однією з ключових вимог при розробці веб-додатків є інформаційна та програмна сумісність з різними операційними системами та браузерами.

Для того, щоб охопити якомога ширшу аудиторію користувачів, інтернет-магазин, що розроблюється в цій кваліфікаційній роботі, має стабільно та ефективно працювати на різних операційних системах, зокрема таких як Windows, macOS, Linux, крім того, бути сумісним з різними версіями браузерів, такими як Google Chrom, Opera, Safari, Firefox, Microsoft Edge.

Кожен браузер може по-різному інтерпретувати код, тому після завершення розробки веб-додаток буде протестований на різних версіях браузерів.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Система, що розробляється в цій кваліфікаційній роботі буде мати такі функціональне призначення:

– ведення бази даних товарів – додавання, видалення редагування даних про товар;

– реєстрація та автентифікація користувачів – забезпечення можливості реєстрації нових користувачів та входу в систему за допомогою облікових записів;

– каталог товарів – додаток надаватиме зручний спосіб перегляду та пошуку товарів, представлених в магазині;

– можливість сортування та пошуку товарів;

– додавання товарів до кошика – користувачі матимуть можливість додавати товари до віртуального кошика для подальшого оформлення замовлення;

– додавання товарів в обране – користувачі матимуть можливість додавати товари в обране задля подальшого збереження товарів які сподобались;

– надання віддаленого доступу до застосунку через веб-браузер через пристрій користувача.

Експлуатаційне призначення:

– автоматизація процесів придбання та продажу товарів;

– контроль та систематизація замовлень;

– підвищення обсягу продажу товарів і, відповідно, збільшення прибутку;

– забезпечення можливості користувачам швидко та зручно знаходити, обирати та купувати товари.

2.2. Опис застосованих математичних методів

Особливості предметної області задачі, що розв’язується в цій роботі, не передбачають застосування математичних методів, тому при розробці інтернет-магазину математичні методи не використовувалися.

2.3. Опис використаної архітектури та шаблонів проектування

Архітектура веб-додатків описує взаємозв’язок між компонентами веб-додатку, їхню структуру та спосіб взаємодії. Вона визначає, як розподіляються функції та обов’язки між різними частинами додатку, щоб забезпечити його ефективну роботу та легкість розширення [6].

Один із найбільш розповсюджених підходів до архітектури веб-додатків – клієнт-серверна модель. Схема цієї моделі зображена на рис. 2.1. В цій моделі веб-додаток складається з двох компонентів: клієнта і сервера.

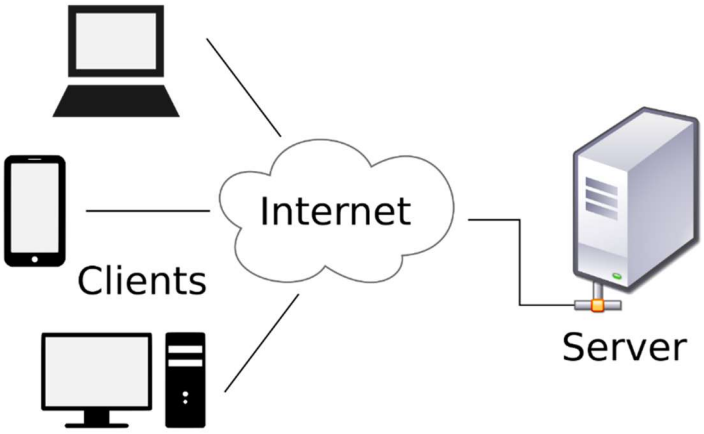

Рис. 2.1. Клієнт-серверна модель

Сервер реалізує такі функції:

– зберігання, доступ, резервне копіювання;
– обробка запиту клієнта;

– відправлення відповіді клієнту.

Функції клієнтської сторони:

– надання користувальницького інтерфейсу;

– створення запиту до сервера та його надсилання;

– отримання результатів запиту.

Принцип дії такої системи полягає в тому, що клієнт надсилає запит на сервер. Там запит оброблюється, готовий результат надсилається клієнтові.

Основні компоненти веб-додатків:

– клієнтська частина (front-end) – інтерфейс, який відображається користувачеві у веб-браузері;

– серверна частина (back-end) – сервер, що опрацьовує запити клієнта, зберігає та обробляє дані, виконує бізнес-логіку та повертає відповіді клієнту.

Веб-додаток, що розроблюється в цій кваліфікаційній роботі також працює за описаною вище архітектурою клієнт-сервер. Клієнтська частина розробляється за допомогою бібліотеки React.js, а серверна – фреймворку Strapi.

Strapi та React разом утворюють потужний тандем для створення динамічних, насичених контентом веб-додатків. Strapi пропонує надійний серверний механізм для керування вмістом, тоді як React відмінно підходить для створення інтерактивних інтерфейсів користувача.

Для побудови архітектури фронтенду використовується методологія FSD (Feature-Sliced Design). Вона надійна, відмінно масштабується, бізнес-орієнтована, дозволяє підтримувати лад в процесі додавання нових можливостей. Головна мета цієї методології – зробити проект більш зрозумілим та структурованим. На рис. 2.2 зображена схема методології FSD.

У FSD-архітектурі проект складається з рівнів, кожен рівень складається зі зрізів, а кожен зріз – з сегментів.

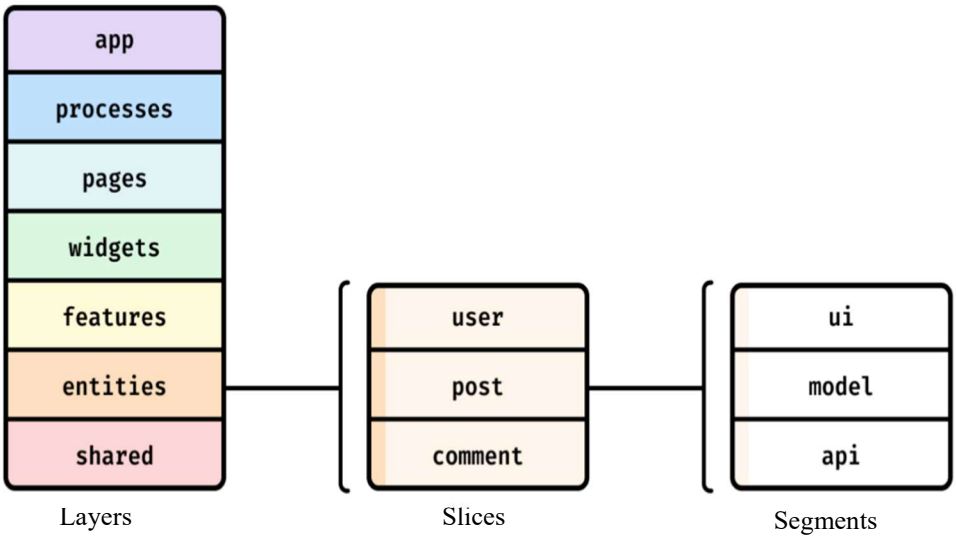

Рис. 2.2. Схема методології Feature-Sliced De*sign*

Layers – каталоги верхнього рівня (перший рівень ділення додатку). Шари стандартизуються, їх сім:

– app – рівень додатку, вміщує налаштування додатку, стилі, провайдери;

– processes – бізнес-процеси додатку, що керують сторінками;

– pages – рівень сторінок, композиційний рівень, що складається з об’єктів, фічів та віджетів;

– widgets рівень віджетів, вміщує компоненти, з яких створені об’єкти та фічі;

– features – частини функціональності додатку;

– entities – рівень об’єктів, вміщує бізнес-об’єкти, специфічні для проекту (user, product тощо);

– shared – вміщує різні ресурси для повторного використання, які не залежать від функціонування бізнес-логіки.

У рівнів діє сходинкова ієрархія, яка побудована так, щоб був зрозумілий однонаправлений потік даних.

Slices (зрізи) – підкаталоги в кожному із рівнів – відповідно до бізнес-домена. Правила, за якими код поділяється на слайси, залежить від конкретного проекту та його бізнес правил і не визначається методологією.

Segments допомагають розділити код всередині зрізу в залежності від технічного призначення коду:

– ui – вміщує логіку інтерфейсу користувача;

– model – бізнес-логіка модуля, зокрема операції збереження, дії, ефекти;

– lib – вміщує інфраструктурну логіку, зокрема утиліти та допоміжні функції;

– api – вміщує логіку запитів до API;

– config – модуль конфігурації додатку та його оточення.

2.4. Опис використаних технологій та мов програмування

Клієнтська частина проекту що розробляється в цій кваліфікаційні роботі, відповідно до теми роботи виконується за допомогою React, бібліотеки JavaScript з відкритим вихідним кодом, яка використовується для створення інтерфейсів користувача спеціально для односторінкових програм.

Обрана бібліотека суттєво полегшує розробку динамічних і швидких інтерфейсів користувача.

Головні переваги React.js:

– використання компонентного підходу. Додаток складається з компонентів, які повторно використовуються, код кожного з них розташований в окремих файлах, зміна стану компонента впливає тільки на його дочірні елементи, не змінюють інші частини інтерфейсу;

– забезпечення більш якісного контролю над даними завдяки однонаправленості потоку даних, це зменшує кількість помилок;

– декларативність – React своєчасно оновлює частини інтерфейсу після їх змінення;

– віртуальний DOM забезпечує ефективне оновлення та відтворення компонентів, підвищує швидкість роботи програми завдяки швидкої реакції на активність користувача, наприклад, оновлення кошика;

– модульність – можливість легко масштабувати програму.

Для збірки проекту, що розроблюється, використовується сучасний збірник модулів Vite, швидкий та легкий інструмент, який забезпечує більш швидку та економічну розробку сучасних веб-проектів. Vite має багато унікальних функцій, що відрізняють його від інших, а у поєднанні з бібліотекою React, що буде використовуватись для створення інтернет-магазину, майбутній проект отримає ще більше переваг:

– швидкий цикл розробки. Найбільша перевага Vite – його швидка гаряча заміна модулів HMR (Hot Module Replacement), у поєднанні з React це призводить до миттєвого відображення змін, що зроблені під час процесу розробки;

– повна підтримка jsx. Це означає, що можна писати та оновлювати компоненти React без додаткового налаштування, що прискорює процес розробки;

– оптимальний розмір збірки. Vite створює невеликі пакети, тому додатки на React швидко завантажуються, що сприяє кращій взаємодії з користувачем;

– інтуїтивно зрозуміле керування ресурсами. Vite обробляє ресурси як імпорт JavaScript, що дозволяє безпосередньо посилатися на зображення, шрифти та інші активні компоненти React. Це покращує продуктивність програми;

– спрощена зручна для React конфігурація. Vite надає готовий до використання шаблон для React, завдяки чому проект React можна швидко запустити, тому що витрачається більше часу на розробку програми та менше на налаштування.

При створенні додатку з використанням Vite в якості інструмента статичної типізації та розробки обирається TypeScript, який розширює можливості JavaScript та робить код більш надійним та зручним. Переваги використання TypeScript у React-проекті:

– більш безпечний код завдяки виявленню помилок на етапі компіляції;

– покращена зрозумілість, чітка структура та типізація роблять код більш читабельним та зрозумілим;

– підтримка TypeScript в редакторах та інструментах розробки покращує роботу з автодоповненням та навігацією по коду.

Використання React з Vite та TypeScript дозволяє створювати сучасний, типізований додаток, який можна легко підтримувати та розвивати.

При створення серверних програм на JavaScript потрібно використовувати платформу Node.js. В її основі лежить модульна система, яка дає змогу підтягувати окремі пакети в додаток. Для керування залежностями у Node.js використовується Node Package Manager (npm).

Серверна сторона програми реалізована за допомогою фреймворку Strapi, який дозволяє максимально швидко та с мінімальними ресурсами створити API для роботи з даними. Strapi - це фреймворк для керування контентом, який працює на Node.js. Це open source-проект, повністю безкоштовний. Система розгортується локально на власному сервері компанії, що забезпечує безпеку даних.

Головні особливості Strapi:

– відкритий вихідний код;

– широкі та гнучкі налаштування;

– локальне розміщення;

– система використовує JavaScript, що дозволяє працювати з однією мовою як в CMS, так і у фронтенді;

– налаштування доступу. В системі впроваджений контроль доступу з урахуванням різних рівнів та ролей користувачів (адміністраторів).

Для зберігання даних Strapi використовує базу даних SQLite.

SQLite — це компактна вбудована реляційна база даних з відкритим кодом. SQLite не є окремим сервером баз даних, як, наприклад, MySQL або PostgreSQL. Натомість це бібліотека, яку можна вбудувати безпосередньо в додаток. Це означає, що для роботи з SQLite не потрібно окремо встановлювати та налаштовувати сервер баз даних [24].

Авторизація користувачів, які можуть зайти до системи побудована на JSON Web Token, яка об’єднує в собі простоту реалізації та безпеку додатку.

JWT (JSON Web Token) – це відкритий стандарт для створення токенів доступу, заснований на форматі JSON. зазвичай він використовується для передачі даних для автентифікації користувачів в клієнт-серверних додатках. Токени створюються сервером, підписуються секретним ключем та передаються юзеру, який в подальшому використовує їх для підтвердження своєї особи.

2.5. Опис структури програми та алгоритмів її функціонування

Проект, що розроблюється у цій кваліфікаційної роботи складається з двох основних частин: клієнтської частини (знаходиться у папці FRONTEND) та серверної частина (знаходиться у папці BACKEND). Кожна з цих частин складається із папок та файлів, що мають різне функціональне значення (відповідають за виконання частин проекту, стилі з оформленням і адаптивним дизайном інтернет-магазину та конфігурацію).

Структури папок FRONTEND та BACKEND зображені на рис. 2.3.

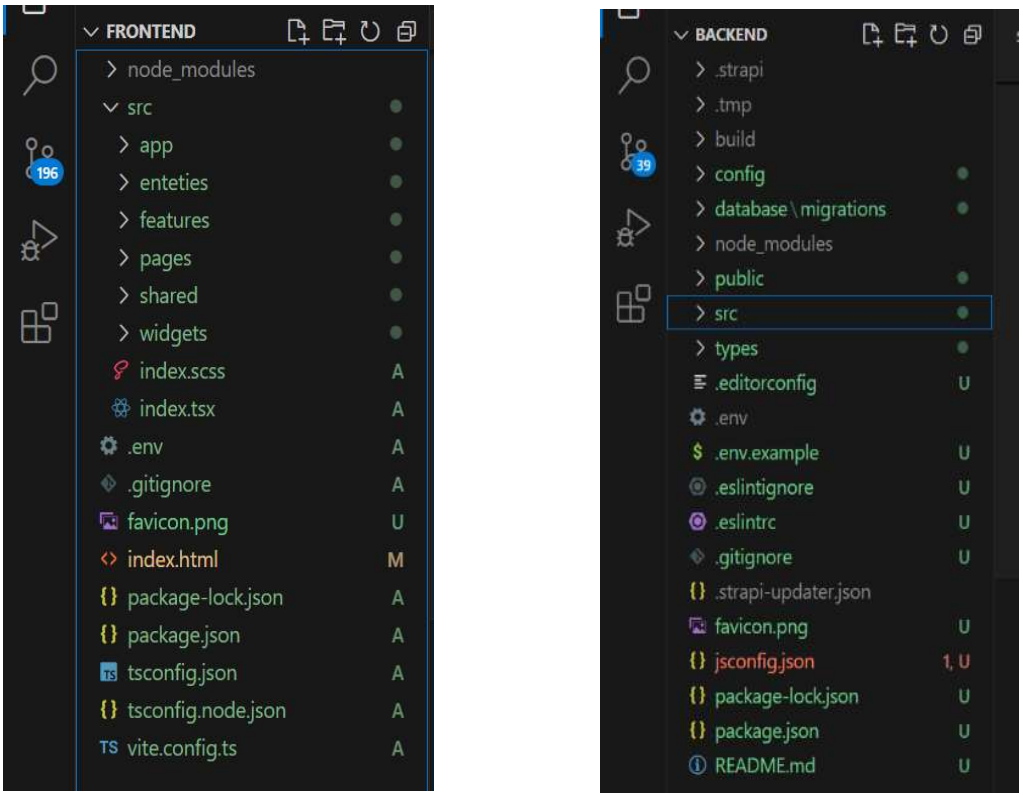

Рис. 2.3. Структури папок FRONTEND та BACKEND

Структура паки FRONTEND відповідає Feature-Sliced Design:

– vite.config.ts – файл конфігурації для Vite;

– tsconfig.node.json та tsconfig.json – вміщують налаштування TypeScript в проекті;

– package.json – вміщує необхідні залежності та скріпти для запуску, збірки та тестування додатку;

– package-lock.json – фіксує точні версії усіх встановлених пакетів та їх залежностей в проекті, це забезпечує стабільність збирання проекту;

– index.html є точкою входу для React-додатку, який розроблюється. В цьому файлі забезпечується базова структура сторінки, на якій монтується React-компонент;

– favicon.png – іконка веб-сайту, яка відображується у назві вкладки браузера.

Папка src вміщує головні елементи додатку:

– файл index.tsx – встановлює основну структуру додатку, забезпечує маршрутизацію, глобальний стан для запитів і повідомлення, а також рендеринг головного компонента програми. Він використовує сучасну модель рендерингу React і імпортує необхідні бібліотеки і стилі для коректної роботи програми;

– файл index.scss – підключає стилі;

– папка app – вміщує елементи для налаштування додатку, стилі, провайдери:

1) файл app.tsx є кореневим компонентом, викликає AppRouter для управління маршрутизацією програми;

2) папка config – вміщує дві папки react-query, в якій виконується експорт QueryClient для забезпечення управління станом даних в додатку, отриманням і керуванням даними через мережу (HTTP запити) та папку route в якій налаштовуються маршрути для відображення відповідних сторінок і макетів;

3) папка layouts вміщує файли, які забезпечує наявність однакового заголовка на різних сторінках, дозволяючи кожному компоненту

сторінки (Home, About тощо) рендеритися всередині макету, визначеного HeaderLayout;

4) папка providers вміщує файл QueryProvider.tsx, який використовується для надання react-query провайдера в додатку;

5) папка styles вміщує стилі;

6) папка types вміщує деклараційні файли для опису модулів;

– enteties вміщує файли, в яких забезпечується формування та зв’язок об’єктів проекту:

1) cart – забезпечує роботу кошика;

2) category – містить файли та папки, що забезпечують відображення та роботу з категоріями товарів;

3) product – містить файли та папки, що забезпечують відображення та роботу з товаром;

4) user – містить файли та папки, що забезпечують роботу з об’єктом «користувач»;

– features містить частини функціональності додатку:

1) auth містить хуки, що використовуються для реєстрації та входу користувача, роблячи їх доступними для імпорту в інших частинах програми.

2) cart містить хуки для роботи з кошиком покупок;

3) wishlist містить хук, що дозволяє керувати функціональністю додавання та видалення елементів зі списку бажань у додатку;

– pages:

1) auth містить файли та папки , які експортують компонент AuthPage, що дозволяє централізовано імпортувати і використовувати сторінку автентифікації (логін і реєстрація) у додатку;

2) cart містить файли та папки, що забезпечують доступ до компонента сторінки кошика, що використовується для перегляду та управління товарами в кошику покупок у додатку;

3) home містить файли та папки, що забезпечують роботу основний інтерфейс користувача та функціональність головної сторінки додатку;

4) product містить файли та папки, що забезпечують доступ до сторінки продукту, що використовується для відображення детальної інформації про окремий товар у додатку;

5) wishlist містить файли та папки, що забезпечують доступ до сторінки wishlist, що використовується для відображення детальної інформації про улюблені товари у додатку;

– shared вміщує різні ресурси для повторного використання, які не залежать від функціонування бізнес-логіки, наприклад, для здійснення HTTP-запитів до API сервера, сортування даних, створення кнопок з різними стилями та функціональністю тощо.

Структура і вміст папки BACKEND будується автоматично після установки та налаштування проекту в Strapi. Strapi за допомогою графічного інтерфейсу створює моделі даних і отримувати до них доступ можна через API (http-запити). Використовується база даних SQLite, яка вбудована в Strapi.

У меню Content-Type Builder сформовано три типи колекції: Category, Product, User.

Тип колекції Category визначає категорію товару. Він має два поля, ім’я категорії та зображення (рис. 2.4)., у вигляді посилання на піктограму, яка зберігається у Media Library.

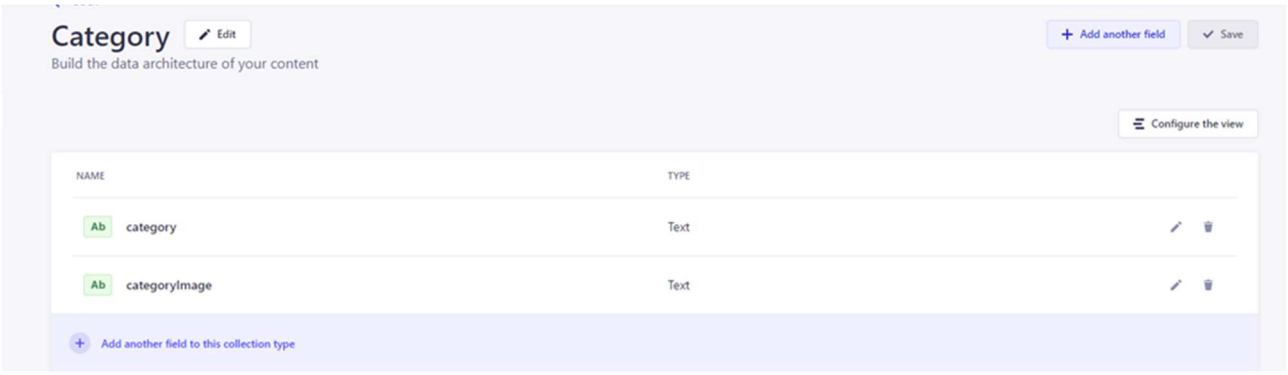

Рис. 2.4. Структура типу колекції Category

Тип колекції Product визначає товар, має шість полів (рис. 2.5):

– name – найменування продукту, тип даних Short text;

– description – опис продукту, тип даних Lond text;

– price – ціна продукту, тип даних integer;

– sizes – формату JSON (використовується для для передачі даних через мережу);

– category – найменування категорії продукту, тип даних Short text;

– image – зображення продукту, має формат Short text, тому що в полі міститься посилання на зображення.

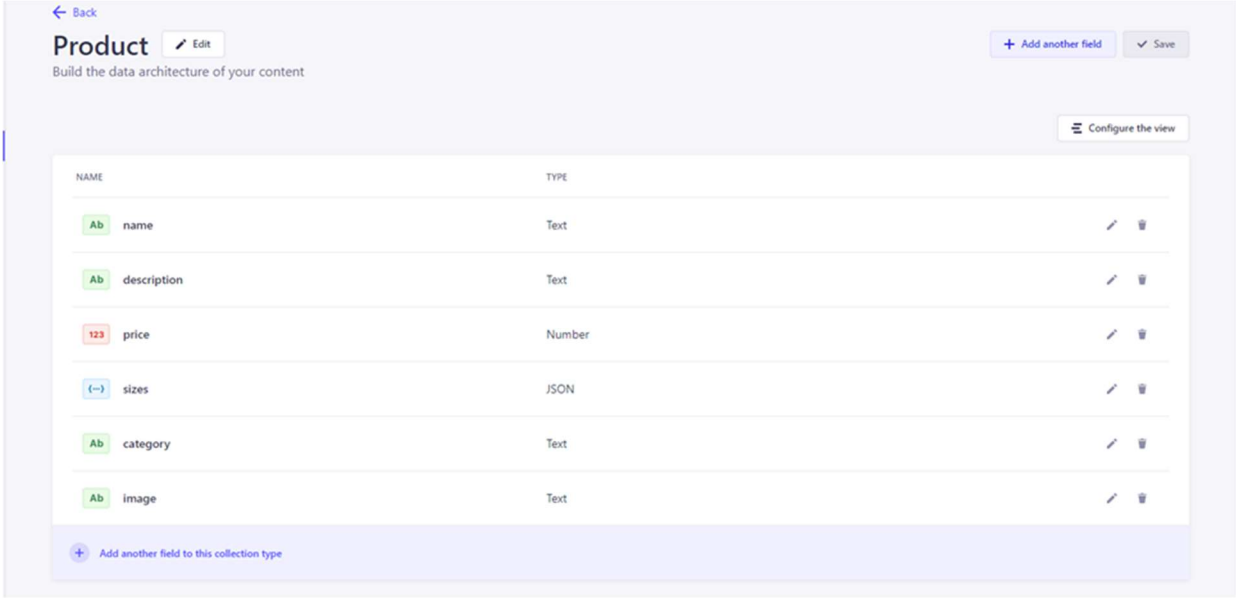

Рис. 2.5. Структура типу колекції Product

Між типами Category та Product встановлений зв'язок від «одного до багатьох».

Тип колекції User визначає користувача (рис. 2.6).

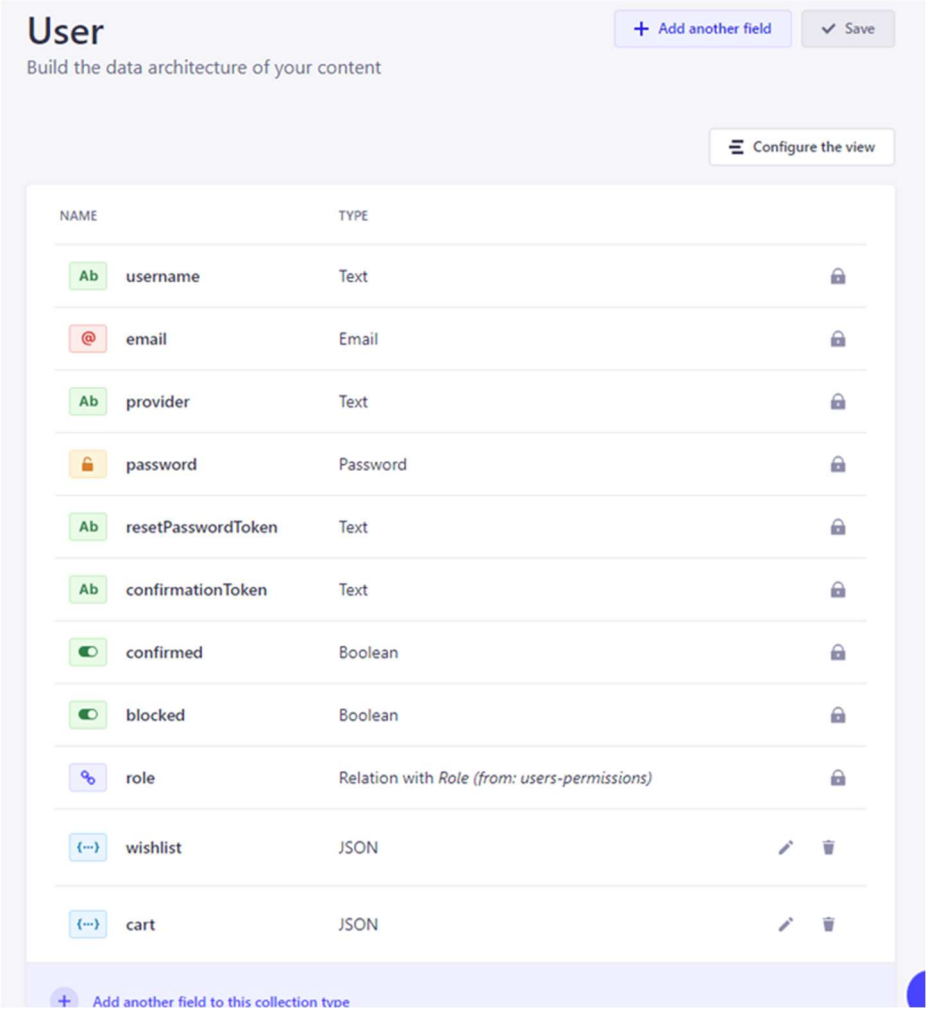

Рис. 2.6. Структура типу колекції User

Тип User - має такі атрибути:

– Username – ім’я користувача;

– email – електронна адреса користувача;

– provider – вказує метод автентифікації користувача;

– password – пароль користувача;

– resetPasswordToken – використовується для відновлення паролю, надсилається на електронну пошту користувача;

– confirmationToken – використовується для підтвердження електронної пошти, надсилається на електронну пошту користувача;

– confirmed – вказує, чи підтверджений обліковий запис користувача;

– blocked – вказує, чи заблокований обліковий запис користувача;

– role – визначає рівень доступу користувача;

– wishlist – вказує на вміст списку бажань користувача, формат даних JSON;

– cart - вказує на вміст кошику, формат даних JSON.

Поля provider, confirmed, blocked, resetPasswordToken та confirmationToken автоматично створюються і керуються системою.

На рис. 2.7 зображена структура реляційної бази даних, яка автоматично створена фреймворком Strapi.

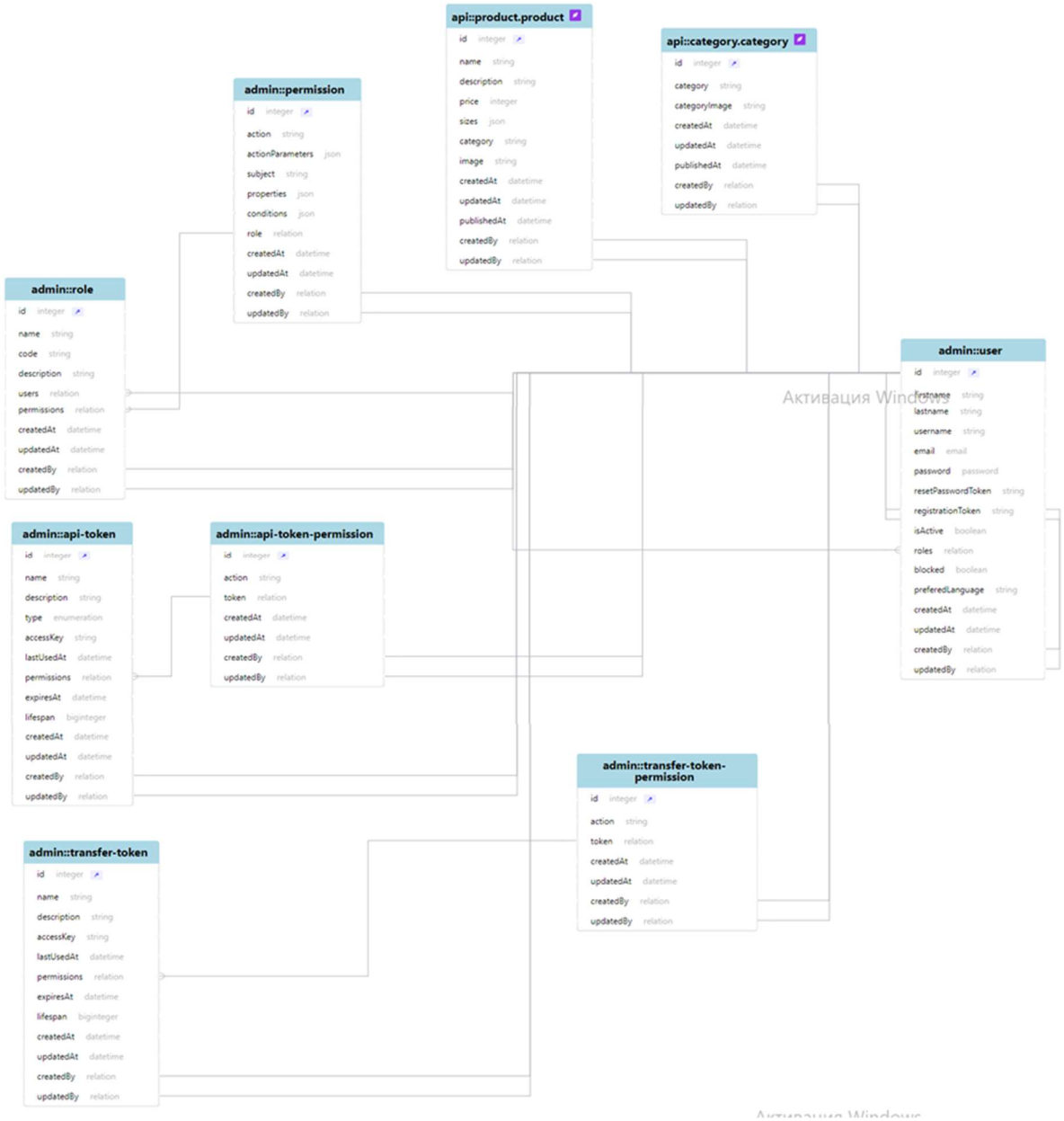

Рис. 2.7. Діаграма зв'язку сутностей

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Програма, що розроблюється буде отримувати вхідні дані шляхом завантаження інформації із бази даних системи та через передачу значень через глобальні змінні додатку інтернет-магазина.

Вхідні дані:

– особисті дані, а саме логін, пароль користувача для входу в особистий кабінет, та адміністратора до входу в адміністративну систему;

– інформація про обрані товари при додаванні у кошик та до сторінки обраних;

– інформація після вибору візуальних елементів управління застосунку.

Вихідні дані:

– товари, обрані користувачем;

– вартість кожного товару та сумарне значення вартості покупки;

– сторінки магазину, на яких можна здійснювати стандартні браузерні операції.

2.7. Опис розробленого програмного продукту

2.7.1. Використані технічні засоби

Для серверних технічних засобів рекомендована конфігурація, що забезпечує цілодобову роботу програми з резервуванням даних:

– процесор класу Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz;

– оперативна пам’ять 2 x DIMM DDR2-800 1024 Мб;

– жорсткі диски 3x 250 Гб SATA 2 16 Мб буфер, 7200 RPM;

– рідкокристалічний монітор з діагоналлю не менше 17 ";

– доступ до мережі Internet;

– клавіатура;

– маніпулятор "миша".

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником. Програмний продукт не є вимогливий до складу та параметрів технічних засобів та може завантажуватись на ПК, ноутбуках, планшетах чи смартфонах різних типів та конфігурацій під управлінням різних ОС.

Для завантаження сторінки необхідно лише веб-браузер.

2.7.2. Використані програмні засоби

Під час розробки даного застосунку були використані такі програмні засоби:

– Visual Studio Code (VS Code);

– браузер Google Chrome;

– операційна система: Windows 10.

Visual Studio Code − це простий редактор коду для кросплатформної розробки веб-додатків та хмарних програм.

Головні особливості Visual Studio Code:

– можливість розділити інтерфейс на декілька панелей коду;

– можливість відкрити декілька проектів в різних вікнах та паралельна робота в них;

– підтримка різних мов програмування, зокрема C++, C#, Java, Python, Go, PHP, JavaScript, TypeScript, HTML, CSS та інших;

– завдяки підсвічуванню синтаксису та виділенню синтаксичних конструкцій тексту полегшує читання тексту програм, та покращує візуальне сприйняття коду;

– завдяки системі IntelliSense, яка автоматично завершує написання коду при вводі початкових букв, забезпечує високу продуктивність;

– присутня можливість розробляти консольні додатки та графічний інтерфейс для користувача;

– можлива розробка для платформ ASP.NET і Node.js, без потреби в повній інтеграції середовища розробки;

– велика кількість бібліотек та шаблонів, фрагментів коду готових для використання, сніпетів з можливістю додавання своїх елементів;

– має вбудовані інструменти для роботи з Git, інструменти інтеграції з GitHub, засоби рефакторингу та навігації по коду, є вбудований відладчик і термінал.

2.7.3. Виклик та завантаження програми

Для запуску проекту у локальному хостингу треба виконати наступні дії:

– скопіювати на ПК, на якому буде працювати програма, архів clothes shop.zip та розпакувати його;

– папки FRONTEND та BACKEND відкрити за допомогою VS Code;

– перевірити версію NodeJS, набравши у терміналі node –v. Версія NodeJS не менше v.20.13.1;

– скачати node models, ввівши команду npm і дочекатися закінчення завантаження;

– запустити backend проекту, для цього у терміналі ввести npm run develop;

– запусти запропонований системою localhost;

– серверна частина проекту відкриється у браузері;

– перейти у VS Code до папки з frontend-кодом;

– скачати node models, ввівши команду npm i, дочекатися закінчення завантаження;

– запустити frontend проекту, для цього у терміналі ввести npm run dev;

– запусти запропонований системою localhost;

– веб-додаток інтернет-магазину відкриється у браузері.

2.7.4. Опис інтерфейсу користувача

2.7.4.1. Клієнтська частина

Після виклику та завантаженню програми, описаних у пункті 2.7.3 у браузері завантажується головна сторінка інтернет-магазину жіночого одягу «AY`s SHOP» (рис. 2.7).

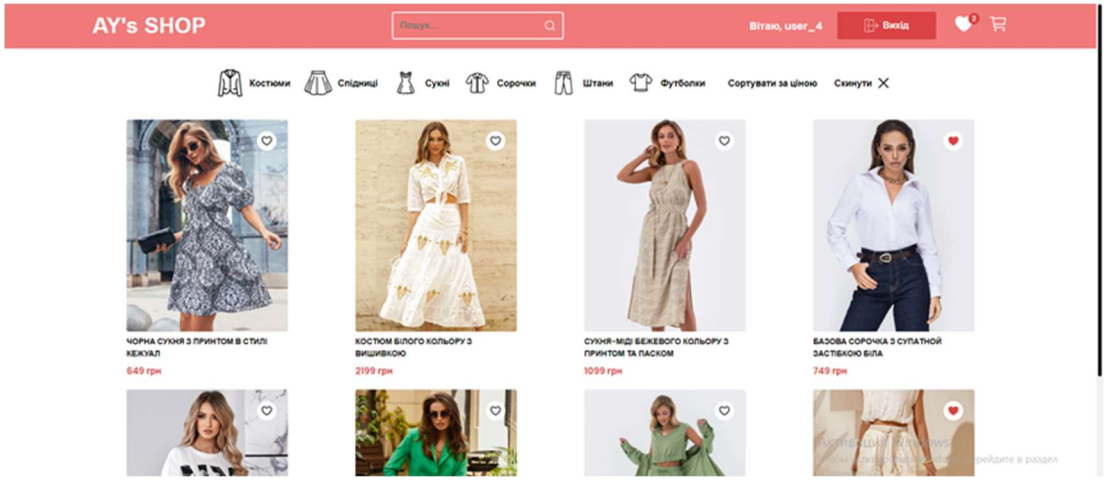

Рис. 2.7. Головна сторінка інтернет-магазину «AY`s SHOP»

Header головної сторінки незареєстрованого користувача містить логотип магазину, пошуковий рядок, посилання на авторизацію. Незареєстрований користувач має можливість переглядати асортимент магазину, відкривати сторінки товарів.

Щоб додавати товари до кошика, користувач повинен пройти авторизацію або ввійти в особистий кабінет, якщо реєстрація вже відбувалась. При спробі додати товар до кошика, з’являється повідомлення «Зареєструйтеся або увійдіть до акаунту» (рис. 2.8).

Рис. 2.8. Повідомлення про необхідність зареєструватись або увійти до акаунту

На рис. 2.9 зображено вікно для реєстрації користувача.

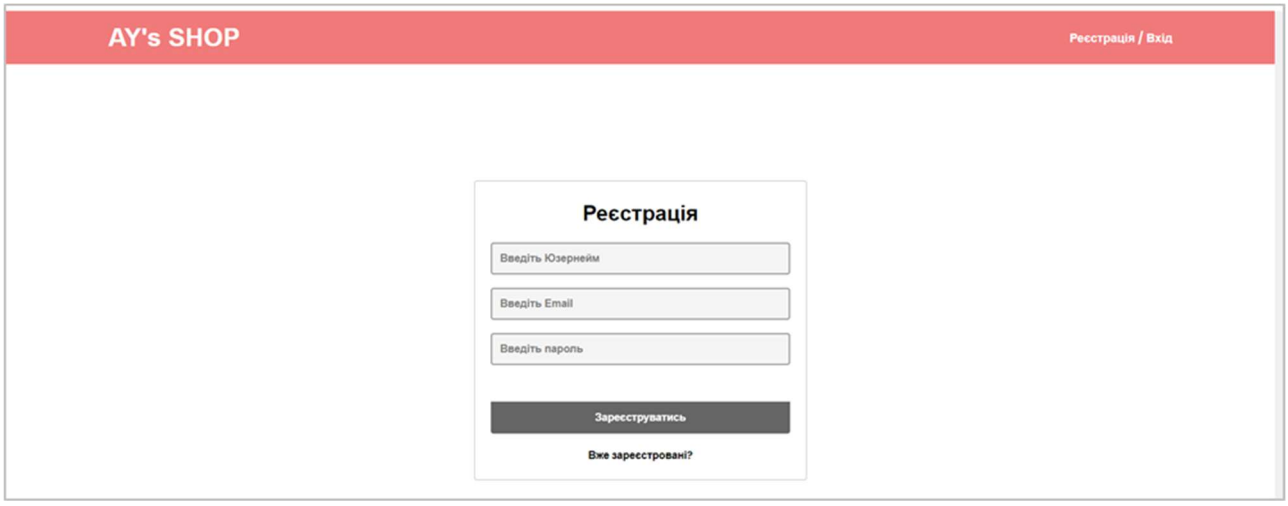

Рис. 2.9. Вікно реєстрації клієнта

Після введення даних і натискання кнопки «Зареєструватись» на екрані з’являється повідомлення «Реєстрація успішна» (рис. 2.10).

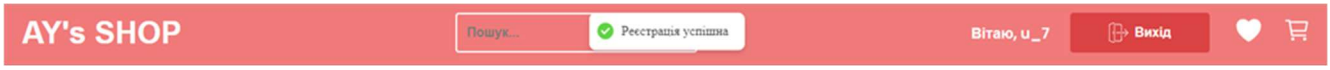

Рис. 2.10. Повідомлення про успішну реєстрацію

На рис. 2.11 зображено вікно входу користувача на особисту сторінку.

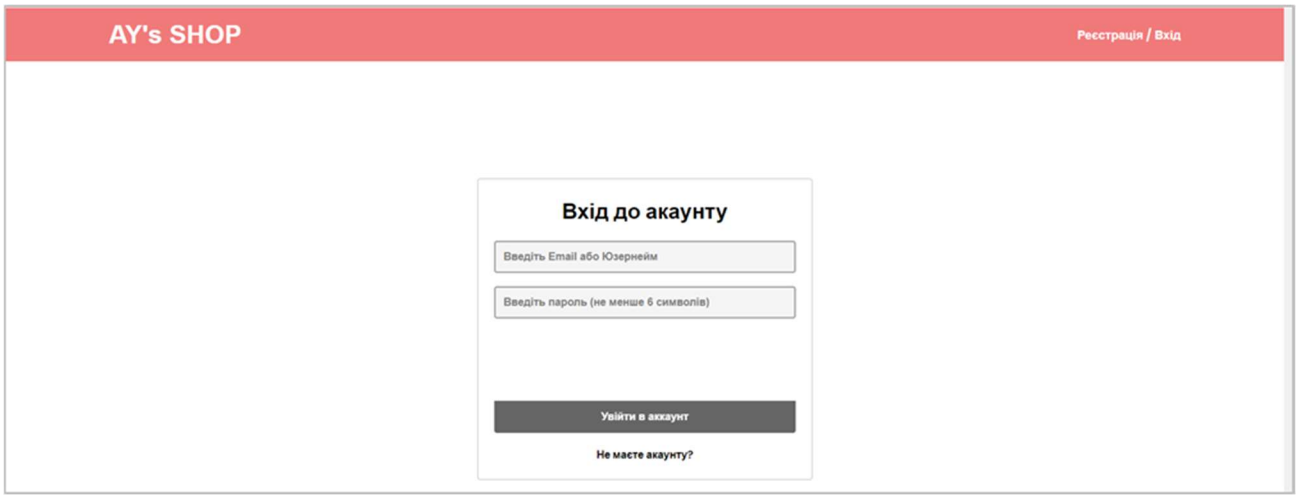

Рис. 2.11. Вікно входу до акаунту для зареєстрованих користувачів

Після введення даних і натискання кнопки «Увійти в акаунт» на екрані з’являється повідомлення «Вхід виконано успішно» (рис. 2.12).

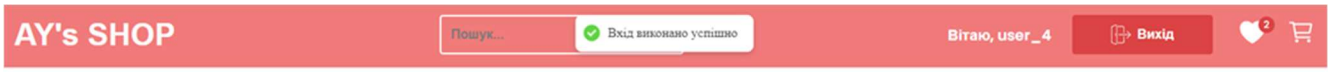

Рис. 2.12. Повідомлення про успішну реєстрацію

Після завершення реєстрації в хедері з’являється привітальне повідомлення, кнопка виходу з акаунту, посилання на кошик та на сторінку улюблених товарів у вигляді відповідних піктограм (рис. 2.13).

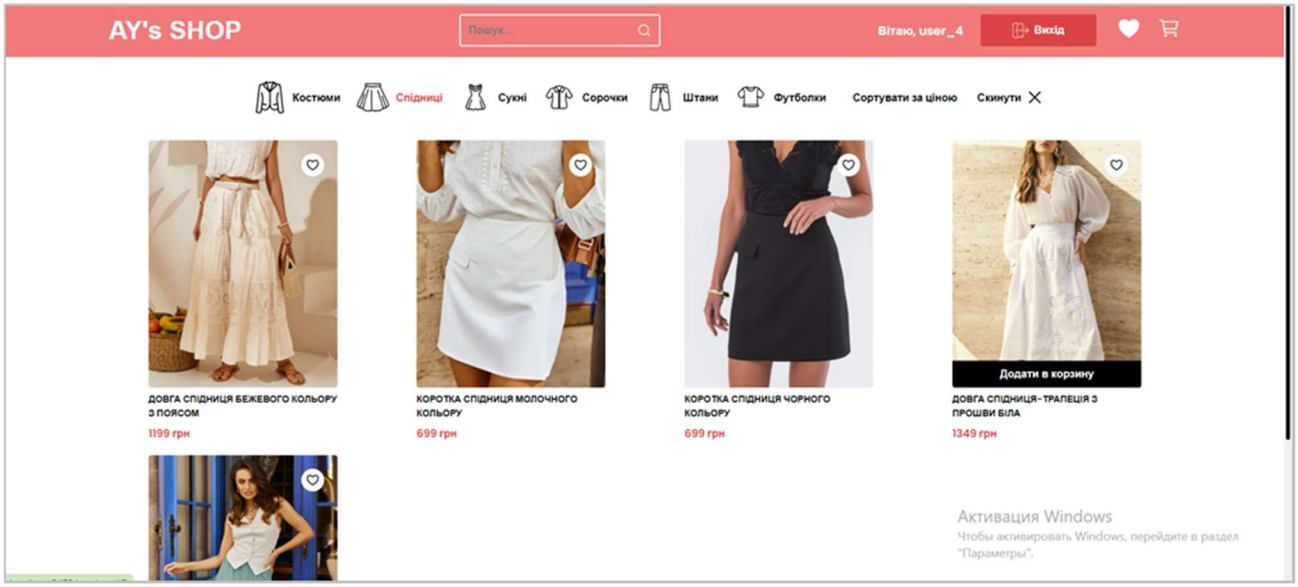

Рис. 2.13. Вигляд сторінки зареєстрованого користувача

Користувач може переглядати наявний асортимент магазину, застосовуючи скролінг та перехід між сторінками. Є можливість вибору категорії товару серед запропонованих у верхній частині сторінки, пошук за назвою, сортування за ціною, а також кнопка скиду результатів фільтрування.

Для того, що на екрані відображались товари заданої категорії, потрібно натиснути на відповідну піктограму. На рис. 2.14 зображені товари категорії «Сорочки».

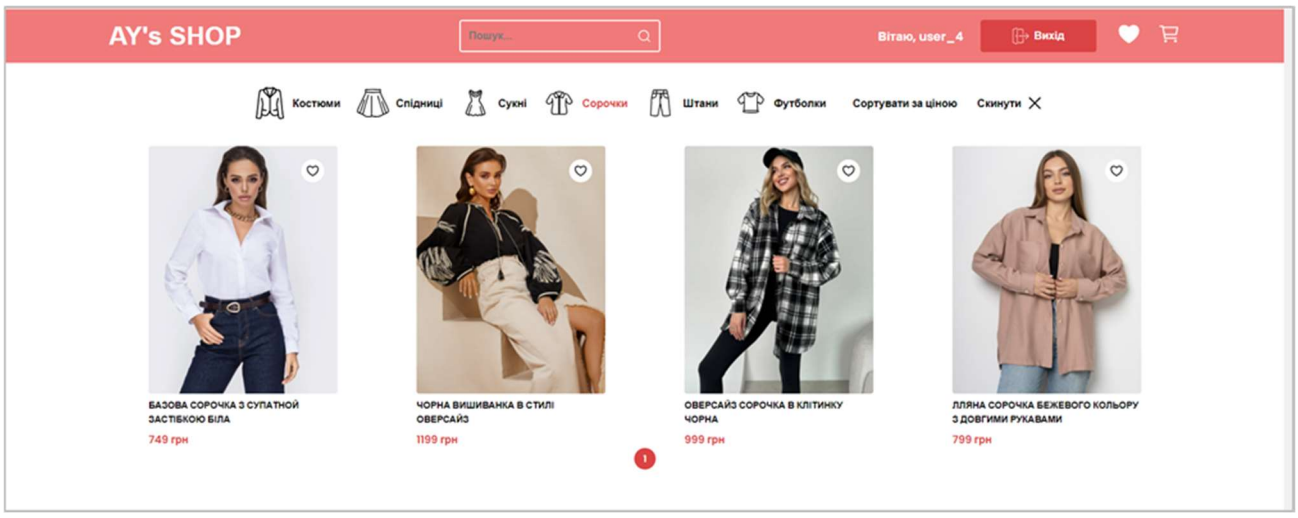

Рис. 2.14. Приклад вибору категорії товарів

Для пошуку товарів за назвою, потрібно у вікні «Пошук» почати вводити назву товару. Вже по перших декількох літерах система знаходить і відображає товари, що потрібні користувачеві. На рис. 2.15 показаний приклад пошуку товарів по запиту «Вишивка».

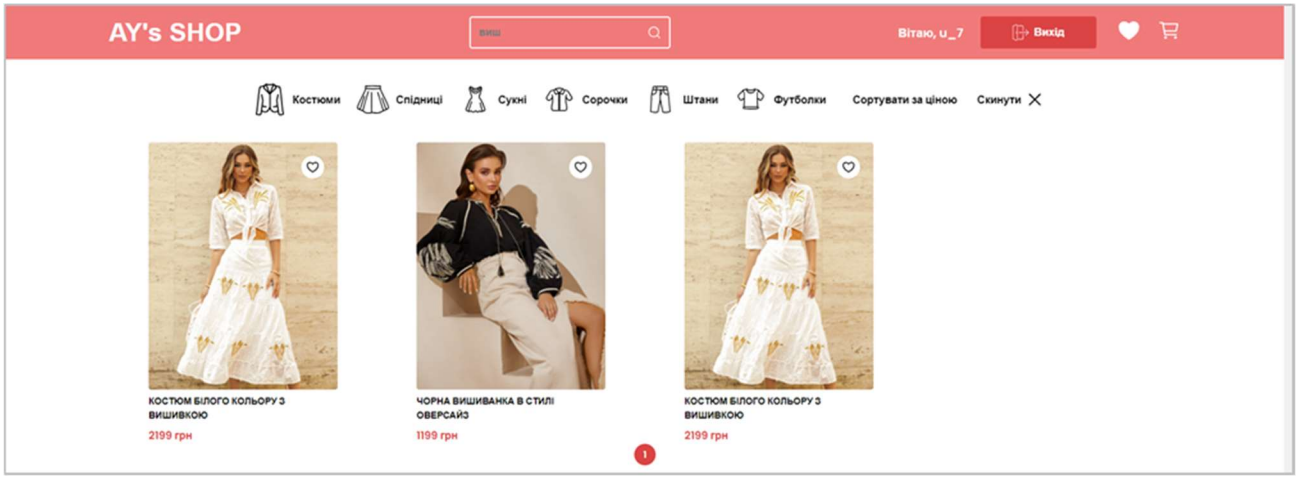

Рис. 2.15. Приклад використання вікна пошуку товарів за назвою

При некоректному вводі або відсутності товарів з заданою у пошуковому рядку назві, з’являється повідомлення, зображення на рис. 2.16.

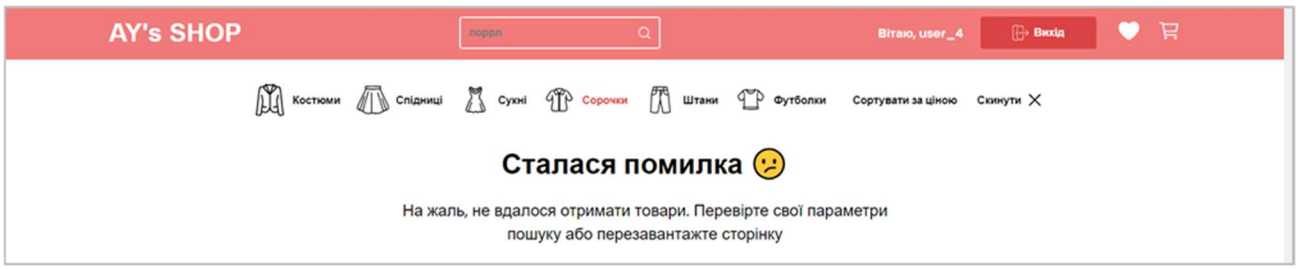

Рис. 2.16. Повідомлення про помилку пошуку

Для вибору та додавання товарів до сторінки улюблених товарів, потрібно натиснути на піктограму у вигляді сердечка у верхньому правому куті зображення товару. Сердечко змінює колір на червоне, а на екрані з'являється повідомлення «Список бажань успішно оновлено». На рис. 2.17 зображений приклад вибору товарів для додавання до списку обраних.

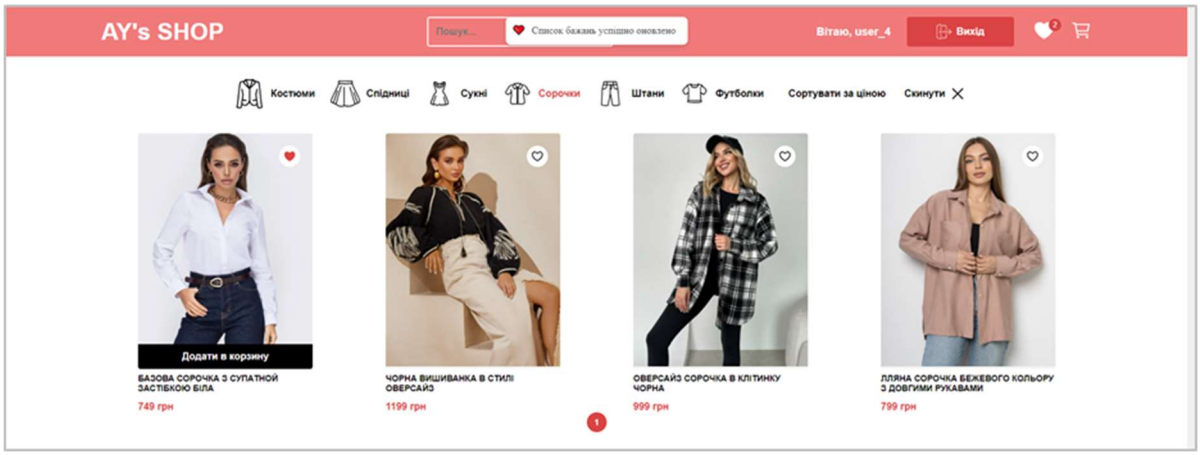

Рис. 2.17. Приклад додавання товарів до сторінки улюблених товарів

На рис. 2.18 зображений вміст сторінки обраних товарів.

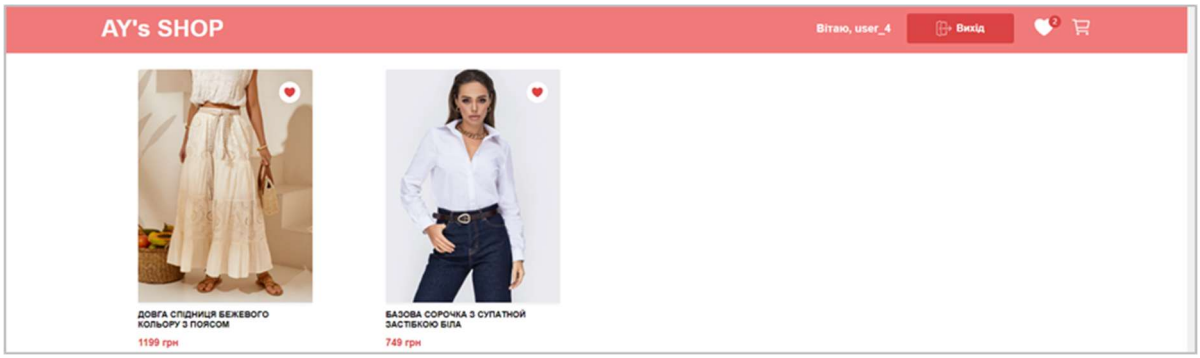

Рис. 2.18. Wish list

Для більш детального перегляду товару, ознайомлення з описом товару, вибору розміру потрібно перейти на сторінку товару. Для цього потрібно натиснути на зображення товару. На рис. 2.19 зображений вигляд сторінки товару.

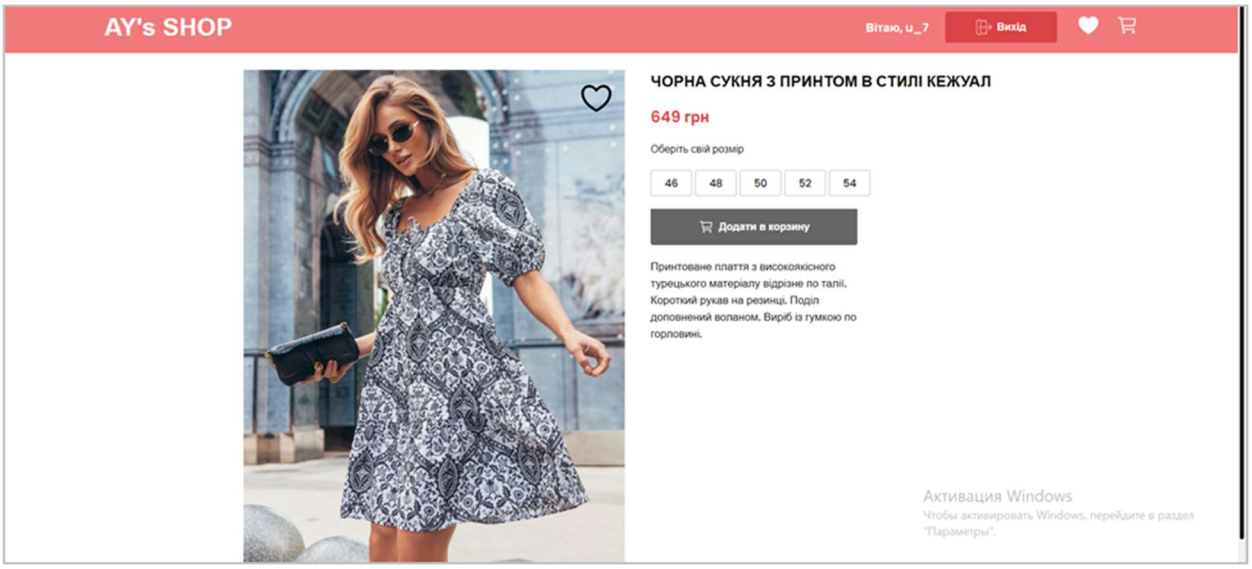

Рис. 2.19. Сторінка товару

Додавання товару в кошик відбувається після натискання кнопки «Додати в корзину», яка знаходиться на сторінці товару та в нижній частині зображення товару. Після натискання «Додати в корзину», з’являється повідомлення «Товар успішно додано до кошика» (рис. 2.20).

Рис. 2.20. Повідомлення про додавання товару до кошика

Приклад вмісту кошика зображений на рис. 2.21. Тут присутній перелік обраних товарів, невелике зображення, можливість вибору розміру, кількості. Підраховується сумарна вартість кожного виду товару та загальна сума до оплати. Є кнопка видалення товару та кнопка оформлення замовлення.

Рис. 2.21. Приклад наповнення кошика

Для видалення товару потрібно натиснути на зображення «Сміттєвий бак». Товар видаляється з переліку товарів корзини, а на екрані з'являється повідомлення «Товар успішно видалений з кошика». На рис. 2.22. наведений приклад видалення товару з кошика.

Рис. 2.22. Видалення товару з корзини та повідомлення про успішне видалення

Після натискання кнопки «Оформити замовлення», з'являється повідомлення «Замовлення відправлене в обробку» (рис. 2.23).

Рис. 2.23. Повідомлення про відправлення замовлення в обробку

2.7.4.2. Адміністрування системою

Як вже зазначалося в попередніх розділах адміністрування магазину відбувається за допомогою Strapi.

Для адміністрування системи, потрібно перейти до вкладки з відкритим додатком Strapi (сервер автоматично запущений на локальному хості після виконання дій, описаних у пункті 2.7.3). У вікні, що знаходиться на екрані ввести електронну адресу та пароль адміністратора (тестовий варіант даних для доступу: Email: testemail1@gmail.com, Password: Testemai1l@gmail.com) (рис. 2.24).

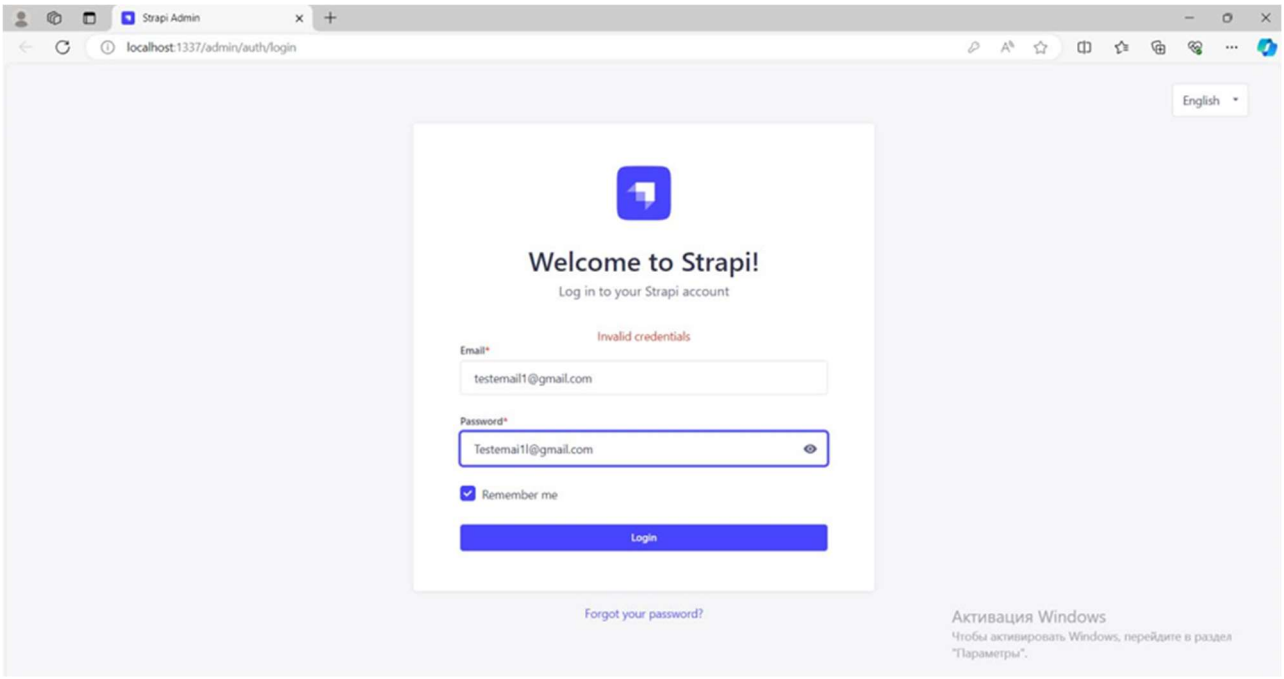

Рис. 2.24. Вхід до адмін-системи

Після входу в систему доступ до панелі адміністратора Strapi (рис. 2.25).

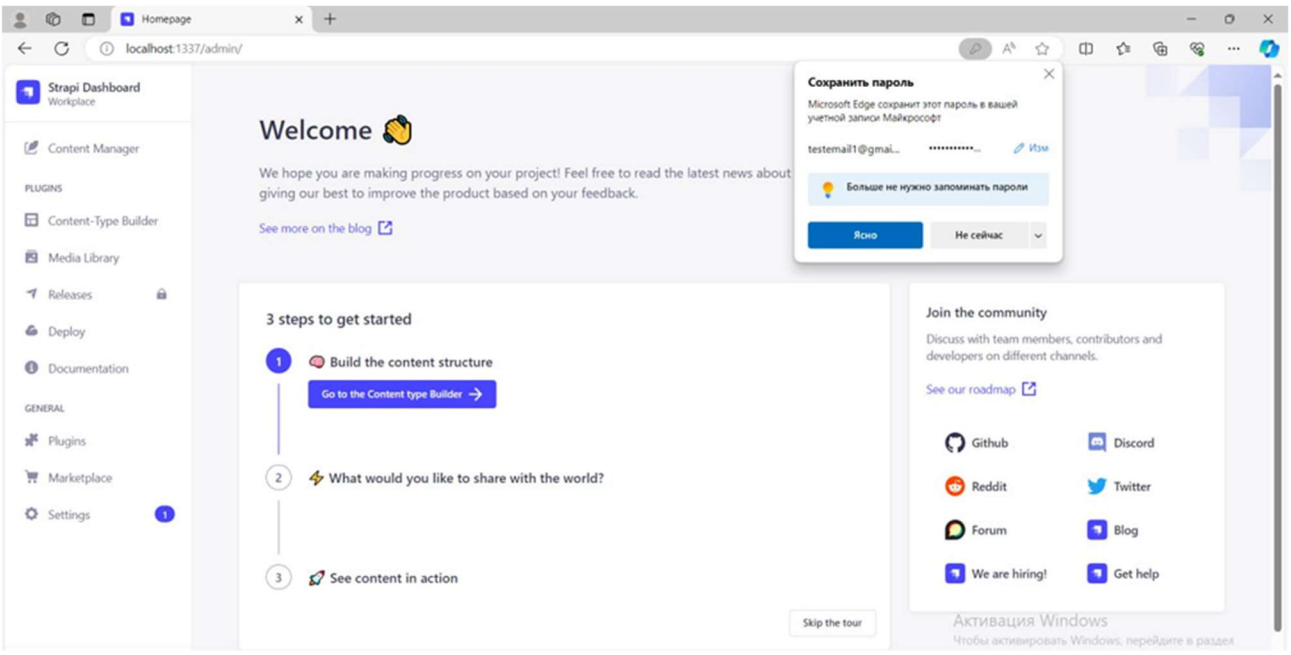

Рис. 2.25. Панель адміністратора Strapi

У лівому меню потрібно обрати пункт «Менеджер вмісту» (Content Manager) за допомогою якого буде здійснюватися управління контентом магазину. У підменю «Collection types» містяться типи контенту, які були створені за допомогою «Content-type Builder».

У проекті встановлені три типи даних: «Категорії» (Category), «Продукти» (Product), «Користувач» (User).

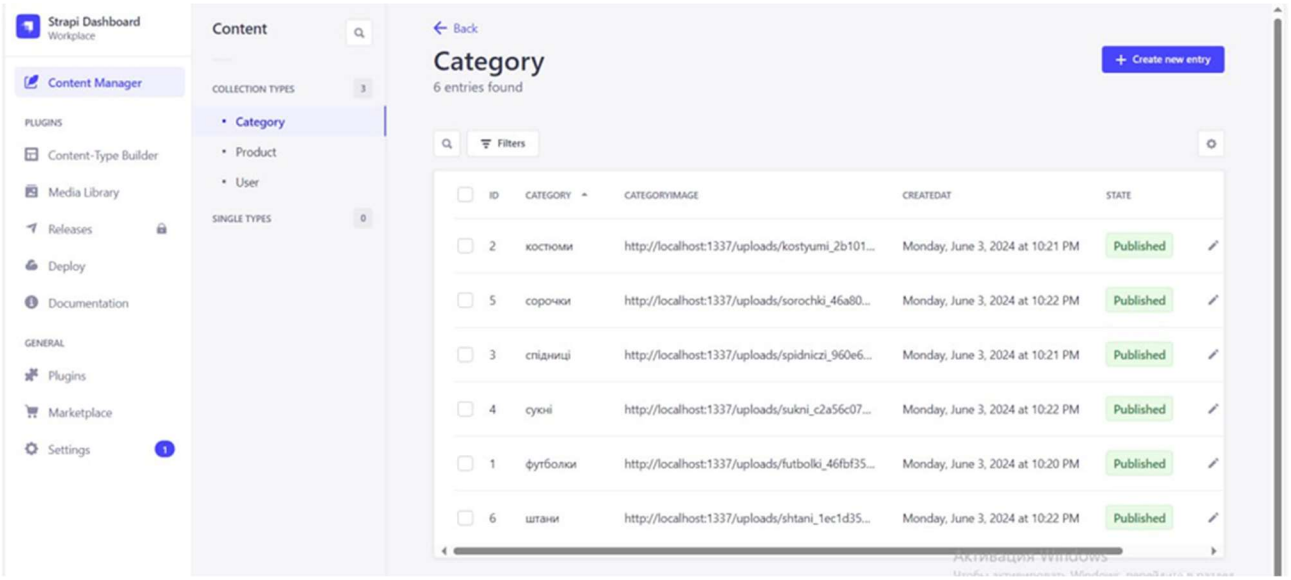

Рис. 2.26. Content Manager

Вміст першого у списку типу даних одразу завантажується в правій частині додатку. Це перелік категорій товарів (рис. 2.26). В цьому вікні можна виконувати такі дії:

– створювати новий запис. Для цього потрібно натиснути кнопку «Create new entry» в правому верхньому куті;

– виконувати пошук та встановлювати фільтри для пошуку конкретних записів. Для цього потрібно натиснути на відповідні іконки, розміщені над таблицею;

– налаштовувати поля, які відображаються в таблиці. Для цього потрібно натиснути на піктограму «шестерня», яка знаходиться над таблицею;

– редагувати запис. Наприклад, після вибору категорії «Костюми» відкривається вікно, в полях якого введені назва категорії та посилання на зображення, яка буде відображатися у клієнтський частині (рис. 2.27). Вміст цих полів можна змінити та натиснути кнопку «Save».

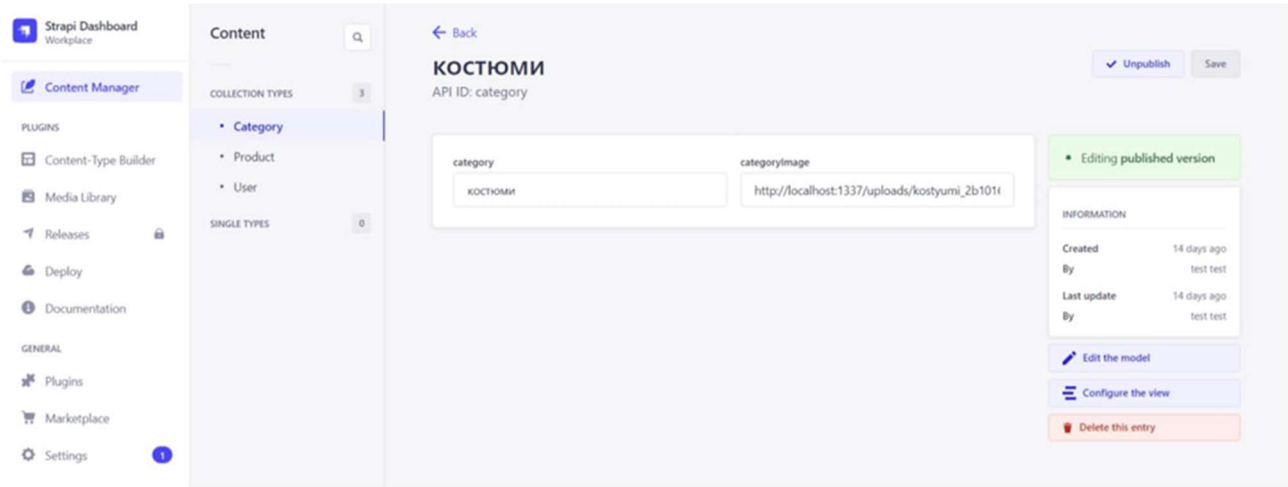

Рис. 2.27. Вікно редагування категорії «Костюми»

Зображення для категорій товару знаходяться у меню Media Library (рис. 2.28). Також тут можна розмістити інші потрібні медіаелементи.

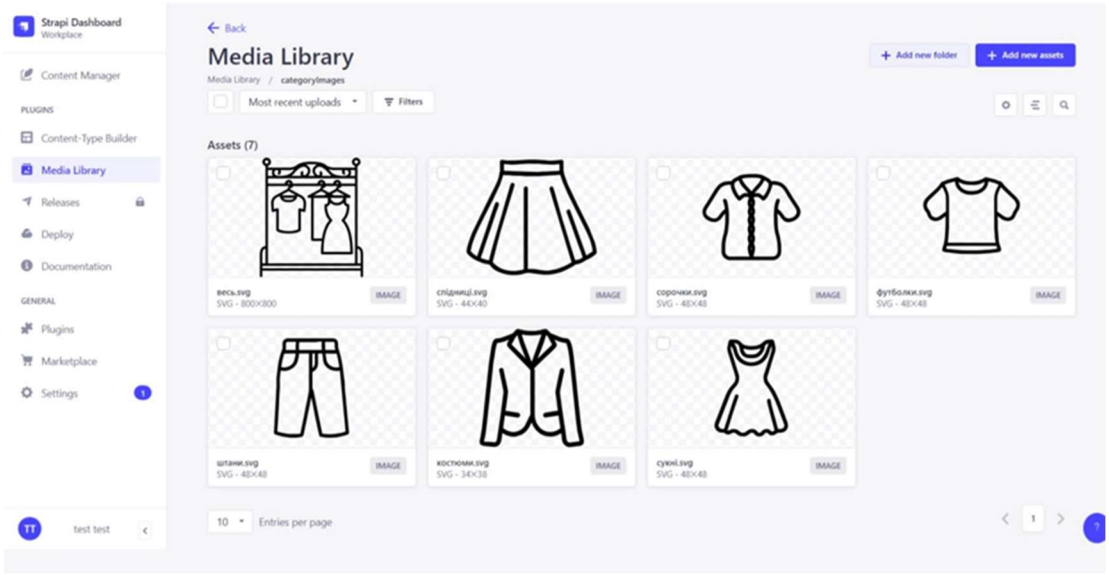

Рис. 2.28. Media Library

Тип Product вміщує перелік товарів, розміщених на сайті (рис. 2.29).

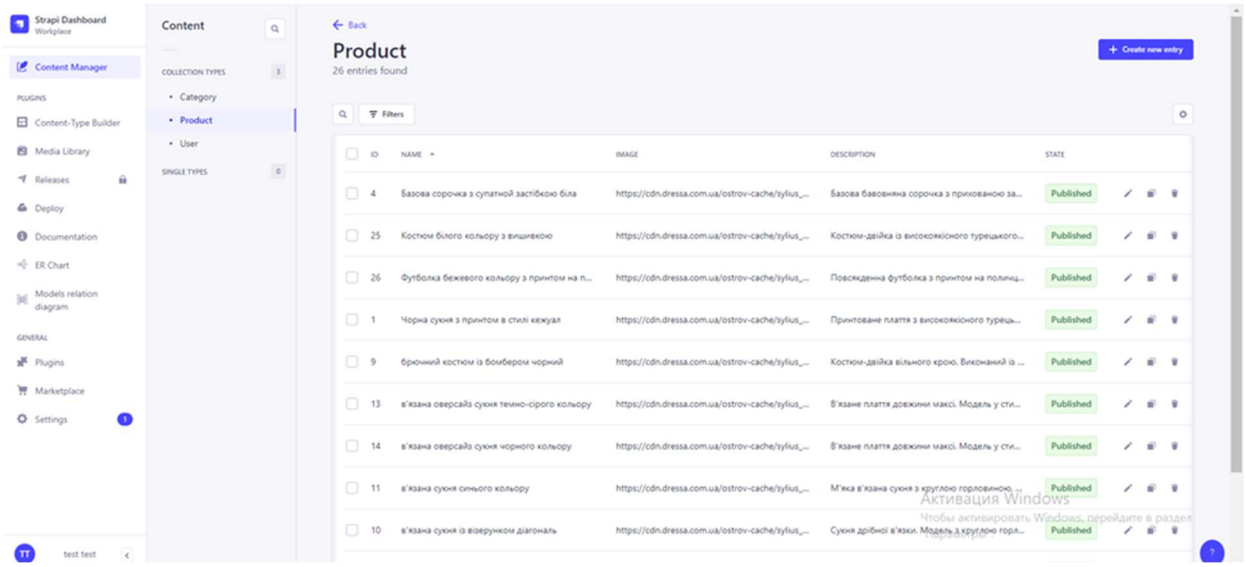

Рис. 2.29. Вміст категорії Product

Для перегляду та редагування інформації про товар потрібно натиснути на назву товару, відкриється нове вікно, в якому містяться такі поля: назва товару, посилання на зображення, опис, вартість, наявні розміри та вікно з назвою категорії, до якої належить товар. На рис. 2.30 наведений вигляд вікна для редагування.

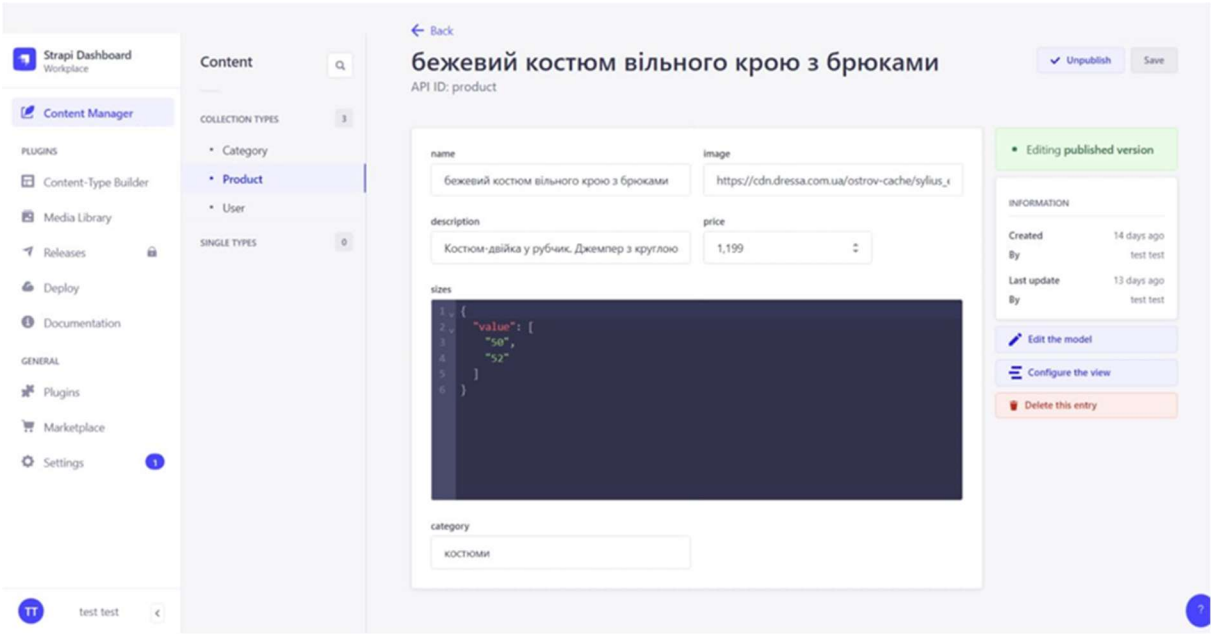

Рис. 2.30. Вигляд вікна для редагування товару

Тип User вміщує інформацію про користувачів, що зареєстровані в системі (рис. 2.31).

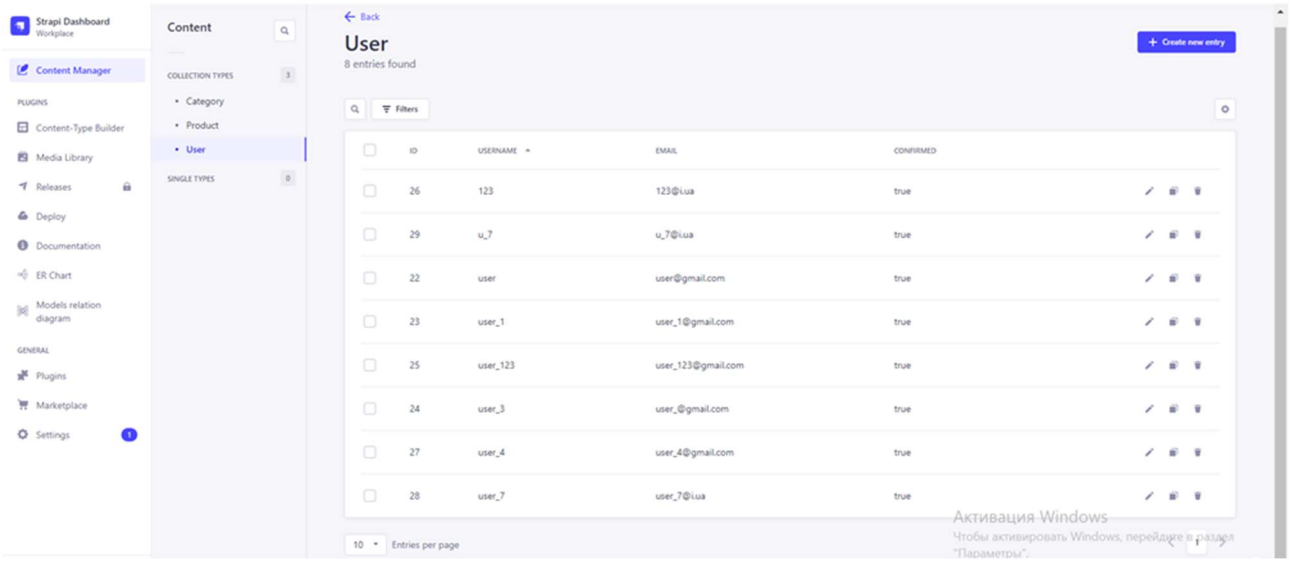

Рис. 2.31. Сторінка з даними про користувачів

Для перегляду розширеної інформації про користувача, потрібно натиснути на рядок, в якому вказане його ім’я. На рис. 2.26 зображено вікно з інформацією про користувача User_1.

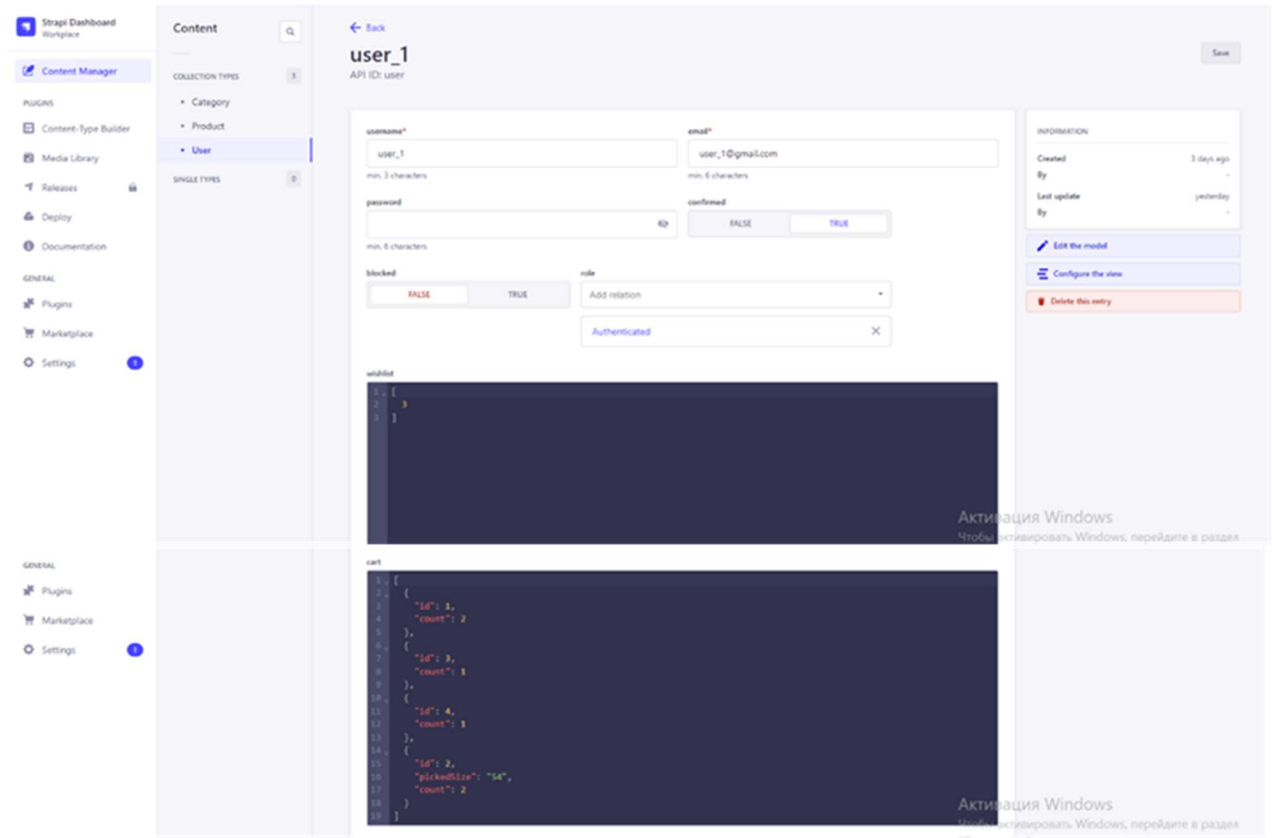

Рис. 2.32. Сторінка з інформацією про користувача

Тут вказані ім’я користувача, його електронна адреса, його роль, а також інформація про його активність в інтернет-магазині, а саме вміст списку бажань (перелік id обраних товарів) та кошику. (перелік id обраних товарів, їх розмір та кількість).

Кожний тип колекції можна редагувати, додавати нові поля, що будуть відображати нову інформацію. Для цього потрібно перейти до пункту меню Content-Type Builder (рис. 2.33).

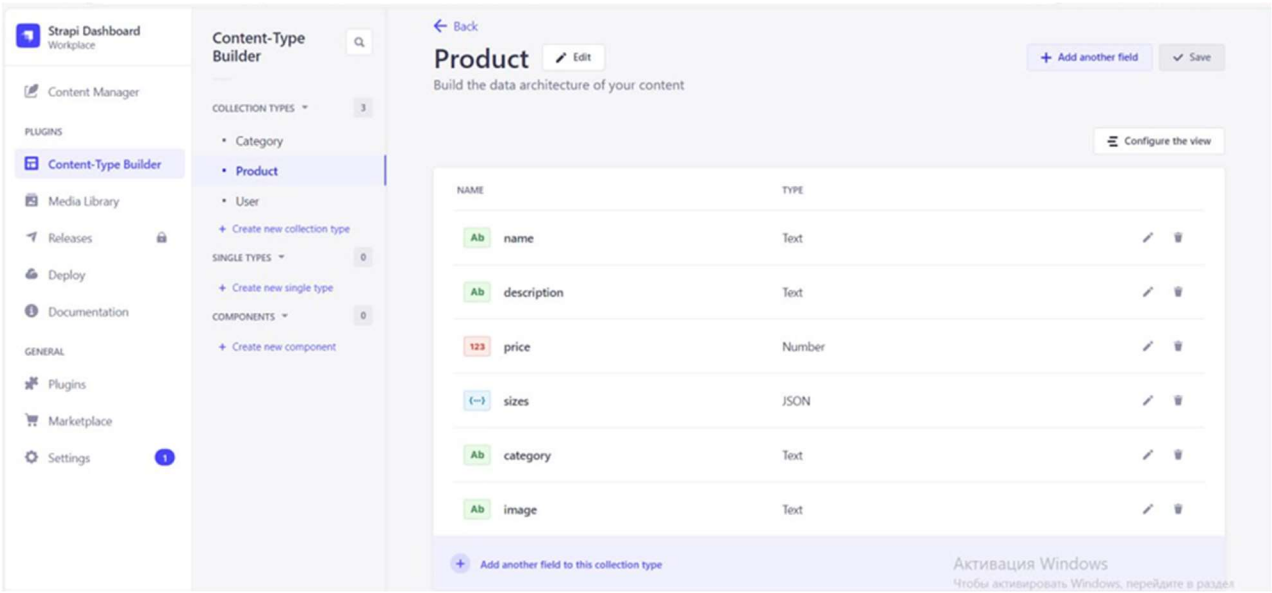

Рис. 2.33. Приклад редагування типу колекції

Для забезпечення безпеки та зручності управління у системі налаштовані ролі, з допомогою чого чітко визначено, що користувачі можуть і не можуть робити в адміністративній системі.

Для налаштування ролей адміністраторів потрібно в адмін-панелі Strapi перейти до «Users & Permissions», потім обрати пункт меню «Roles» (рис. 2.34).

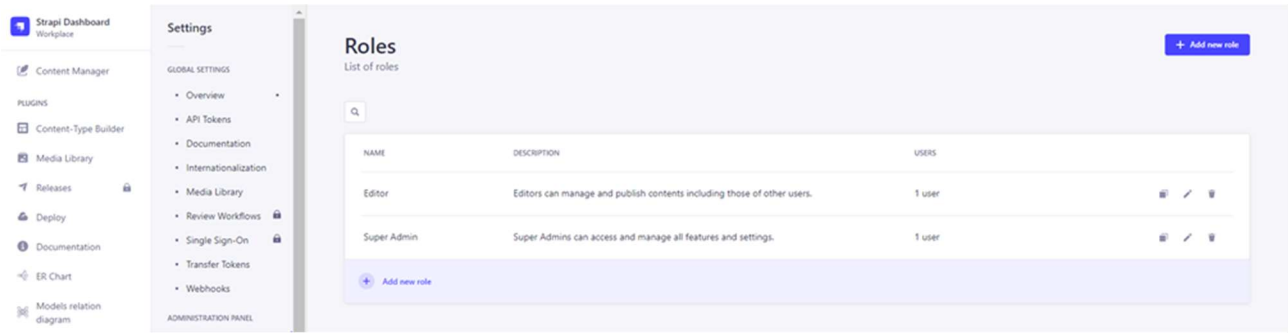

Рис. 2.34. Вікно для створення та редагування ролей

Для створення нової ролі потрібно натиснути кнопку «Add new role», заповнити запропоновані поля: назву ролі, її опис, налаштувати якій контент роль може переглядати, створювати, редагувати чи видаляти (рис. 2.35).

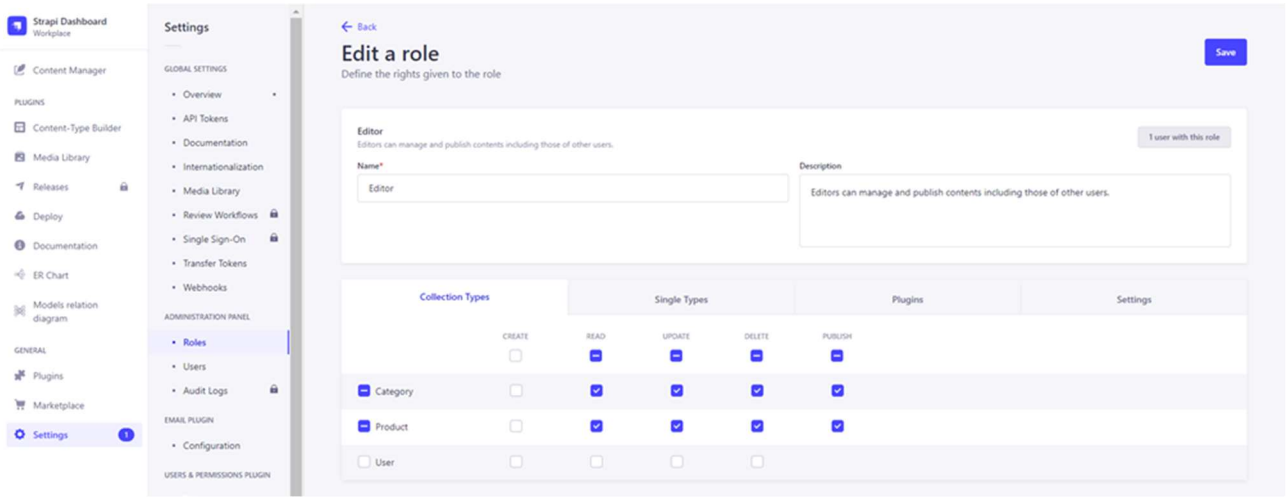

Рис. 2.35. Вікно для налаштування ролі

Для призначення адміністративної ролі для користувача потрібно перейти до пункту меню «Users» у адмін-панелі, вибрати користувача зі списку або створити нового, призначити роль користувачу, вибравши її зі списку, зберегти зміни (рис. 2.36).

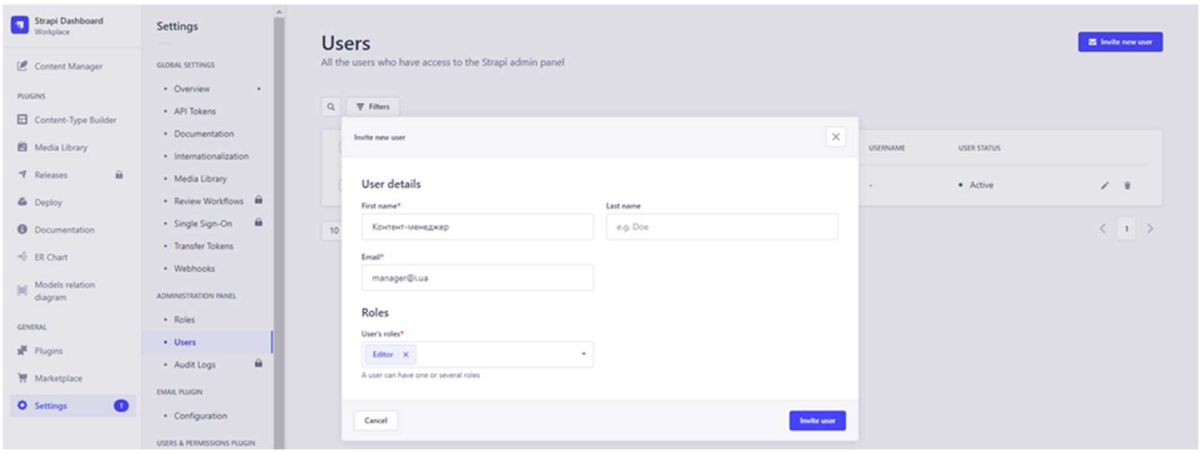

Рис. 2.36. Призначення ролі Editor контент-менеджеру

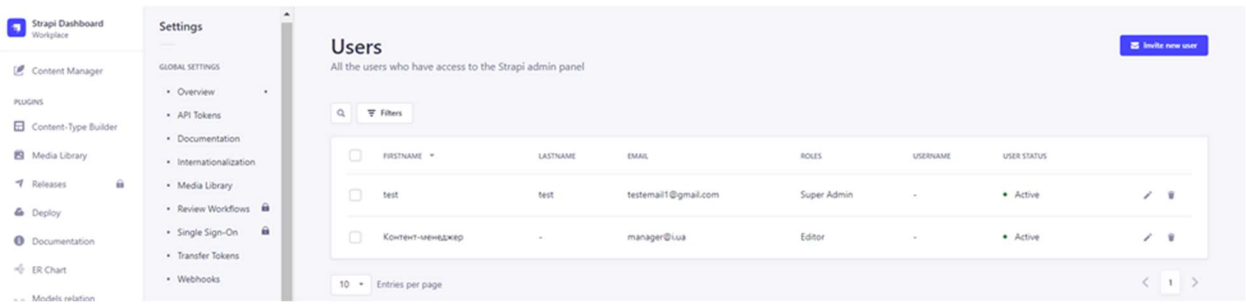

Рис. 2.37. Список адміністраторів

Для відвідувачів магазину призначені ролі Authenticated та Public (рис. 2.38).

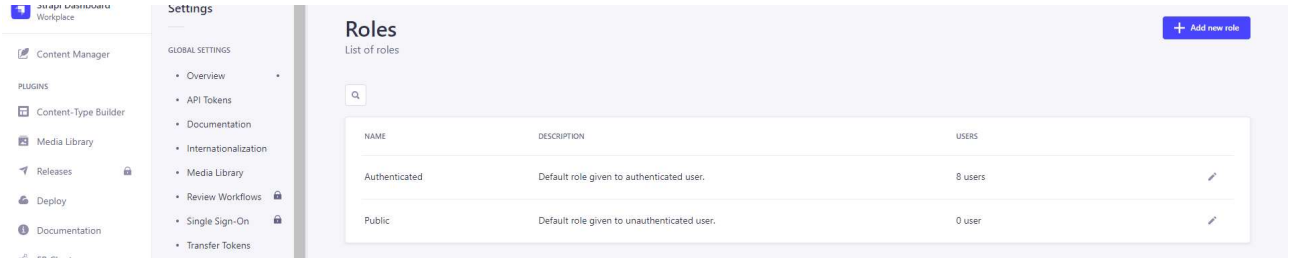

Рис. 2.38. Список ролей для відвідувачів магазину

Для кожного типу налаштовані відповідні дозволи. Роль Public для всіх незареєстрованих користувачів. Вони мають доступ тільки на перегляд товарів та категорій. Роль Authenticated для зареєстрованих користувачів, які увійшли до системи. Такі користувачі можуть переглядати і змінювати свої власні дані профілю, додавати товари до кошика та сторінки обраних товарів, оформлювати замовлення.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1 Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

– q – передбачуване число операторів – 750;

– C – коефіцієнт складності програми – 1,5;

– p – коефіцієнт корекції програми в ході її розробки – 0,07;

– B – коефіцієнт збільшення витрат праці в наслідок недостатнього опису задачі – 1,2;

– k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 0,8 (стаж менше 2 років);

– $C_{Пр}$ – середня погодинна заробітна плата React.js developer в Україні – 603 грн/год (відповідно до інформації, наведеній у [27], станом на 29 травня 2024 року);

– B_k – кількість розробників – 1;

– F_p – місячний фонд робочого часу (при 40 годинному робочому тижні F_p = 176 годин);

– вартість машино-години ЕОМ (складається з витрат на оплату електроенергії за тарифом 4,32 грн./кВт та амортизацію ноутбука, який використовується в роботі) – 300,3 грн/год.

Нормування праці в процесі створення програмного забезпечення істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки програмного забезпечення може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_в + t_a + t_n + t_н + t_д \quad (3.1)$$

де $t_о$ – витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

$t_в$ – витрати праці на дослідження алгоритму рішення задачі;

$t_а$ – витрати праці на розробку блок-схеми алгоритму;

$t_п$ – витрати праці на програмування по готовій блок-схемі;

$t_н$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \tag{3.2}$$

де q – передбачуване число операторів;

C – коефіцієнт складності програми;

p – коефіцієнт корекції програми в ході її розробки.

Звідси за формулою (3.2) умовне число операторів в програмі:

$$Q = 750 \cdot 1{,}5 \cdot (1 + 0{,}07) = 1203{,}75 \text{ людино-годин},$$

Витрати праці на вивчення опису задачі $t_в$ визначається з урахуванням уточнення опис

$$t_в = \frac{Q \cdot B}{(75..85) \cdot k} \tag{3.3}$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

$$t_в = (1203{,}75 \cdot 1{,}2) / (80 \cdot 0{,}8) = 22{,}57 \text{ людино-годин}.$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25)\cdot k} \tag{3.4}$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.6), отримаємо:

$$t_{\text{а}} = 1203{,}75 / (22 \cdot 0{,}8) = 68{,}39 \text{ людино-годин},$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k} \tag{3.5}$$

За формулою (3.5) обчислюємо витрати праці на програмування по готовій блок-схемі:

$$t_n = 1203{,}75 / (25 \cdot 0{,}8) = 60{,}19 \text{ людино-годин}.$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{\text{н}} = \frac{Q}{(4\ldots5)\cdot k}. \tag{3.6}$$

Витрати праці на налагодження програми на ЕОМ за умови автономного налагодження одного завдання за формулою (3.6):

$$t_н = 1203{,}75 / (4 \cdot 0{,}8) = 376{,}17 \text{ людино-годин},$$

– за умови комплексного налагодження завдання:

$$t_{\text{н}}^{k} = 1{,}5 \cdot t_{\text{н}} \tag{3.7}$$

Витрати праці на налагодження програми на ЕОМ за умови комплексного налагодження завдання за формулою (3.7):

$$t^{k}_{н} = 1{,}5 \cdot 376{,}17 = 564{,}26 \text{ людино-годин.}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_{д} = t_{др} + t_{до}\,. \tag{3.8}$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20) \cdot k} \tag{3.9}$$

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 \cdot t_{др}\,. \tag{3.10}$$

За формулами (3.9), (3.10) та (3.8) обчислюємо витрати праці на документацію:

$$t_{др} = 1203{,}75 / (17 \cdot 0{,}8) = 88{,}51 \text{ людино-годин,}$$

$$t_{до} = 0{,}75 \cdot 88{,}51 = 66{,}38 \text{ людино-годин,}$$

$$t_{д} = 88{,}51 + 66{,}38 = 154{,}89 \text{ людино-годин.}$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t = 50 + 22{,}57 + 68{,}39 + 60{,}19 + 564{,}26 + 154{,}89 = 920{,}3 \text{ людино-годин.}$$

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПЗ} = З_{ЗП} + З_{МВ} \tag{3.11}$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР} \tag{3.12}$$

де: t – загальна трудомісткість, людино-годин;

$С_{Пр}$ – середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 603 грн / год, за формулою (3.12) отримуємо:

$$З_{ЗП} = 920{,}3 \cdot 603 = 554\,943{,}8 \text{ грн.},$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot С_{мч} \tag{3.13}$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год;

$С_{мч}$ – вартість машино-години ЕОМ, грн/год.

Підставивши в формулу (3.13) відповідні значення, обчислюємо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 564{,}26 \cdot 300{,}34 = 169\,471{,}62 \text{ грн.}$$

Звідси, за формулою (3.11), витрати на створення програмного продукту:

$$К_{ПЗ} = 554\,943{,}8 + 169\,471{,}62 = 724\,415{,}42 \text{ грн.}$$

Очікуваний період створення програмного застосунку:

$$T = \frac{t}{B_k \cdot F_p}, \tag{3.14}$$

де B_k – число виконавців;

F_p – місячний фонд робочого часу.

Звідси, за формулою (3.14), витрати на створення програмного продукту:

$$T = 920{,}3 / (1 \cdot 176) \approx 5{,}23 \text{ місяців}$$

Висновки: Вартість розробки інтернет-магазину становить 724 415,42 грн. Час розробки очікується приблизно 5,23 місяців при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці. Цей термін включає час, необхідний для дослідження, розробки алгоритму, дизайну і документування. Загальна кількістю людино-годин, яку буде витрачено на розробку – 920,3.

ВИСНОВКИ

Мета кваліфікаційної роботи – розробка інтернет-магазину з продажу жіночого одягу з використанням бібліотеки React.

Інтернет-магазин представляє собою інформаційну систему призначену для:

– об’єднання елементів прямого маркетингу та традиційної торгівлі;

– надання зручного інструменту управління візуальним контентом та адміністративної частини проекту;

– забезпечення клієнтам інтернет-магазину максимальної зручності роботи із системою, простоти і комфортності доступу до каталогу товарів за рахунок оптимальних параметрів візуалізації його вмісту;

– підвищення ефективності роботи компанії, за рахунок автоматизації управління бізнес-процесами.

Для управління роботою інформаційної системи і її контентом передбачено існування певного механізму, що дає доступ до функціоналу, недоступному звичайним користувачам. З цією метою було розроблену адміністраторську систему. За її допомогою адміністратор отримує контроль над системою у повній мірі: управління товарами; управління розділами; управління замовленнями; статистика; моніторинг стану системи. Під словом «управління» слід розуміти перегляд, редагування, додавання та видалення інформації.

Актуальність розробки інтернет-магазину не викликає сумніву та визначається зростаючою потребою на дистанційний спосіб придбання товарів, який дозволяє клієнтам економити зусилля та час, а життя в умовах всесвітньої пандемії та війни особливо підкреслили і довели необхідність розвитку електронної комерції в Україні.

Розроблене програмне забезпечення інформаційної системи призначене для застосування в будь-якій компанії з подібним родом діяльності і схожими функціональними вимогами.

Створена система дозволить оптимізувати та спростити дії по веденню операцій онлайн-продажів; скоротити час на оформлення продажу; підвищити ефективність діяльності компанії за рахунок автоматизації його діяльності в сфері електронної комерції та можливості моніторингу стану системи і аналізу наявних даних (за рахунок наявності адміністративної частини проекту).

Проект реалізований з використанням сучасних технологій та архітектурних підходів з використанням бібліотеки React, фреймворку Strapi, що дозволило створити ефективну та сучасну систему, яка може задовільнити високі вимоги користувачів у сфері електронної комерції. Розробка проводилася із використанням редактор коду Visual Studio Code.

В «Економічному розділі» визначено трудомісткість розробки програмного забезпечення (920,3 людино-годин), підраховані витрати на створення програмного забезпечення (724 415,42 грн) і очікуваний період розробки (5,23 місяці).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шалева О.І. Електронна комерція [Електронний ресурс] // URL: https://westudents.com.ua/knigi/209-elektronna-komertsya-shaleva-o.html (дата звернення: 27.05.2024).

2. Статистика розвитку e-commerce у найбільших регіонах світу. // URL: https://magazine.ukr-china.com/statystyka-rozvytku-e-commerce-u-najbilshyh-regionah-svitu/. (дата звернення: 18.05.2024).

3. Тардаскіна Т.М. Електронна комерція: Навчальний посібник / Т.М.Тардаскіна, Є.М.Стрельчук, Ю.В.Терешко – Одеса: ОНАЗ ім. О.С. Попова, 2011. – 244 с.

4. Вимоги до програмного забезпечення [Електронний ресурс] // URL: Посилання. (дата звернення: 18.05.2024).

5. Загрози інформаційної безпеки [Електронний ресурс] // URL: https://uk.wikipedia.org/wiki/Загрози_інформаційної_безпеки (дата звернення: 27.05.2024).

6. Архітектура веб додатків [Електронний ресурс] // URL: https://medium.com/@IvanZmerzlyi/ (дата звернення: 18.05.2024).

7. Статичні та динамічні web-сайти [Електронний ресурс] // URL: https://armedsoft.com/ua/blog/statychni-ta-dynamichni-web-sayty (дата звернення: 27.05.2024).

8. Якою має бути головна сторінка інтернет-магазину [Електронний ресурс] // URL: https://redchameleon.com.ua/ua/articles/kakoy-dolzhna-byt-glavnaya-stranitsa-internet-magazina/ (дата звернення:18.05.2024).

9. Інтернет-магазин «Dressa» [Електронний ресурс] // URL: https://dressa.com.ua/ (дата звернення:18.05.2024).

10. Інтернет-магазин «One by One» [Електронний ресурс] // URL: https://onebyone.ua/ (дата звернення:18.05.2024).

11. Security of JSON Web Tokens (JWT) [Електронний ресурс] // URL: https://cyberpolygon.com/materials/security-of-json-web-tokens-jwt/ (дата звернення:18.05.2024).

12. Vite Next Generation Frontend Tooling [Електронний ресурс] // URL: https://vitejs.dev/ (дата звернення:18.05.2024).

13. React JavaScript-бібліотека для створення користувацьких інтерфейсів [Електронний ресурс] // URL: https://uk.legacy.reactjs.org/ (дата звернення: 18.05.2024).

14. Run JavaScript Everywhere [Електронний ресурс] // URL: https://nodejs.org/en (дата звернення: 18.05.2024).

15. Manage any content. Anywhere. [Електронний ресурс] // URL: https://strapi.io/ (дата звернення: 18.05.2024).

16. Create React App. / URL: https://create-react-app.dev/ (дата звернення: 13.05.2024).

17. Express.js (фреймворк для розробки серверної частини проекту) / URL: https://expressjs.com/ (дата звернення: 14.05.2024).

18. React Router (маршрутизація в React). / URL: https://reactrouter.com/ (дата звернення: 14.05.2024).

19. Axios (робота з HTTP-запитами). / URL: https://axios-http.com/ (дата звернення: 14.05.2024).

20. Material-UI (компоненти UI на основі Material Design). / URL: https://mui.com/ (дата звернення: 14.05.2024).

21. Що таке Node JS простими словами [Електронний ресурс] // URL: https://dan-it.com.ua/uk/blog/chto-jeto-takoe-node-js-prostymi-slovami/ (дата звернення: 18.05.2024).

22. «Express in Action: Написання, створення та тестування програм Node.js», Еван М. Хан, Manning Publications, Shelter Island, 2016 , 34 с

23. Що таке npm і навіщо він потрібен [Електронний ресурс] // URL: https://robotdreams.cc/uk/blog/271-chto-takoe-npm-i-zachem-on-nuzhen (дата звернення: 18.05.2024).

24. Що таке SQLite? [Електронний ресурс] // URL: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-sqlite/ (дата звернення:18.05.2024).

25. Visual Studio Code [Електронний ресурс] // URL: https://uk.wikipedia.org/wiki/Visual_Studio_Code (дата звернення: 15.06.2024).

26. Стандарт розробки React для початківців [Електронний ресурс] // URL: https://dou.ua/forums/topic/44659/ (дата звернення: 15.06.2024).

27. React. js developer: середня зарплата в Україні [Електронний ресурс] // URL: https://ua.jooble.org/salary/react.-js-developer#hourly. (дата звернення: 18.05.2024).

28. React + Vite: why use? [Електронний ресурс] // URL: https://dev.to/doccaio/react-vite-why-use-cg2 (дата звернення: 18.05.2024).

29. Create a Blog with React and Strapi [Електронний ресурс] // URL: https://medium.com/@adeyinkakazeemolufemioluoje/create-a-blog-with-react-and-strapi-cc3d8f0f01e1 (дата звернення: 18.05.2024).

30. Blog Using React & Strapi [Електронний ресурс] // URL: https://www.linkedin.com/pulse/blog-using-react-strapi-martin-pedraza (дата звернення: 18.05.2024).

ДОДАТОК А

КОД ПРОГРАМИ

FRONTEND

Лістинг файлу index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" href="favicon.png" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>AYs_shop</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/index.tsx"></script>
  </body>
</html>
```

Лістинг файлу index. tsx

```
// react
import React from "react";
import ReactDOM from "react-dom/client";
import { BrowserRouter } from "react-router-dom";
// providers
import { QueryProvider } from "@/app/providers/react-query/QueryProvider";
import { Toaster } from "react-hot-toast";
// app
import App from "./app/App.tsx";
// styles
import "./index.scss";

ReactDOM.createRoot(document.getElementById("root")!).render(
  <BrowserRouter>
    <QueryProvider>
      <App />
      <Toaster />
    </QueryProvider>
  </BrowserRouter>
);
```

Лістинг файлу App. tsx

```
// config
import { AppRouter } from "./config/route";
// styles
import styles from "./App.module.scss";
const App = () => {
  return (
    <div className={styles.App}>
      <AppRouter />
    </div>
  );
};
export default App;
```

Лістинг файлу App.module. tsx

```
.App {
  min-height: 100vh;
  margin: 0 auto;
  font-family: "Poppins", sans-serif;
  font-weight: 500;
}
```

Лістинг файлу AppRouter. tsx

```
// react
import { FC } from "react";
import { Route, Routes } from "react-router-dom";
// guards
import { LoggenInGuard } from "./routeGuards/LoggenInGuard";
import { NotLoggenInGuard } from "./routeGuards/NotLoggenInGuard";
// layouts
import { HeaderLayout } from "../../layouts/header";
// routes
import {
  getAuthRoute,
  getCartRoute,
  getHomeRoute,
  getProductRoute,
  getWishlistRoute,
} from "@/shared/libs/constants/routes";
// pages
import { HomePage } from "@/pages/home";
import { ProductPage } from "@/pages/product";
import { AuthPage } from "@/pages/auth";
import { WishListPage } from "@/pages/wishlist";
import { CartPage } from "@/pages/cart";

interface AppRouterProps {}

export const AppRouter: FC<AppRouterProps> = () => {
  return (
    <Routes>
      <Route element={<HeaderLayout />}>
        <Route path={getHomeRoute()} element={<HomePage />} />
        <Route path={getProductRoute(":productId")} element={<ProductPage />} />
        <Route element={<LoggenInGuard />}>
          <Route path={getAuthRoute()} element={<AuthPage />} />
        </Route>
        <Route element={<NotLoggenInGuard />}>
          <Route path={getWishlistRoute()} element={<WishListPage />} />
          <Route path={getCartRoute()} element={<CartPage />} />
        </Route>
      </Route>
    </Routes>
  );
};
```

Лістинг файлу QueryProvider. tsx

```
// react
import { FC, ReactNode } from "react";
// react-query
import { QueryClientProvider } from "@tanstack/react-query";
// config
import { queryClient } from "../../config/react-query/queryClient";
```

```
interface QueryProviderProps {
  children: ReactNode;
}

export const QueryProvider: FC<QueryProviderProps> = ({ children }) => {
  return (
    <QueryClientProvider client={queryClient}>{children}</QueryClientProvider>
  );
};
```

Лістинг файлу QueryProvider. tsx

```
// react
import { FC, ReactNode } from "react";
// react-query
import { QueryClientProvider } from "@tanstack/react-query";
// config
import { queryClient } from "../../config/react-query/queryClient";

interface QueryProviderProps {
  children: ReactNode;
}

export const QueryProvider: FC<QueryProviderProps> = ({ children }) => {
  return (
    <QueryClientProvider client={queryClient}>{children}</QueryClientProvider>
  );
};
```

Лістинг файлу strapiInstanse.ts

```
// libs
import axios from "axios";
import toast from "react-hot-toast";
// enteties
import { userActions } from "@/enteties/user";
// constants
import { ACCESS_JWT_TOKEN } from "../libs/constants/jwtToken";

export const strapiInstanse = axios.create({
  baseURL: "http://localhost:1337/api",
});

strapiInstanse.interceptors.request.use(
  (config) => {
    const token = localStorage.getItem(ACCESS_JWT_TOKEN);
    if (token) {
      config.headers.Authorization = `Bearer ${token}`;
    }
    return config;
  },
  (error) => {
    return Promise.reject(error);
  }
);

strapiInstanse.interceptors.response.use(
  (response) => {
    return response;
  },
  (error) => {
    if (error.response && error.response.status === 400) {
```

```
      toast.error(error.response.data.error.message);
    }

    if (error.response && error.response.status === 403) {
      toast.error("Спочатку вам потрібно увійти в аккаунт");
    }

    if (error.response && error.response.status === 401) {
      toast.error("Ваша сесія закінчилася , ввійдіть ще раз");
      localStorage.removeItem(ACCESS_JWT_TOKEN);
      userActions.logout();
    }

    return Promise.reject(error);
  }
);
```

Лістинг файлу vite.config.ts

```
import { defineConfig } from "vite";
import react from "@vitejs/plugin-react";
import svgr from "vite-plugin-svgr";

export default defineConfig({
  plugins: [react() ,svgr()],
  resolve: {
    alias: [{ find: "@", replacement: "/src" }],
  },
});
```

Лістинг файлу AuthPage.tsx

```
// react
import { FC, useState } from "react";
// ui
import { SignInForm } from "../SignInForm/SignInForm";
import { SignUpForm } from "../SignUpForm/SignUpForm";
// styles
import styles from "./AuthPage.module.scss";

interface AuthPageProps {}

export const AuthPage: FC<AuthPageProps> = ({}) => {
  const [isRegisterForm, setRegisterForm] = useState(false);

  const onRegisteredToggle = () => {
    setRegisterForm(!isRegisterForm);
  };

  return (
    <div className={styles.AuthPage}>
      {isRegisterForm ? (
        <SignInForm onClick={onRegisteredToggle} />
      ) : (
        <SignUpForm onClick={onRegisteredToggle} />
      )}
    </div>
  );
};
```

Лістинг файлу SignInForm.tsx

```
// react
import { FC, useState } from "react";
import { useNavigate } from "react-router-dom";
// features
import { useLoginUser } from "@/features/auth";
// routes
import { getHomeRoute } from "@/shared/libs/constants/routes";
// ui
import { Input } from "@/shared/ui/Input";
import { Form } from "@/shared/ui/Form";
import { Button } from "@/shared/ui/Button";
// styles
import styles from "./SignInForm.module.scss";

interface SignInFormProps {
  onClick: () => void;
}

export const SignInForm: FC<SignInFormProps> = ({ onClick }) => {
  const navigate = useNavigate();

  const [userData, setUserData] = useState({
    identifier: {
      value: "",
      type: "email",
      placeholder: "Введіть Email або Юзернейм",
    },
    password: {
      value: "",
      type: "password",
      placeholder: "Введіть Пароль",
    },
  });

  const onSuccessCallback = () => {
    navigate(getHomeRoute());
  };

  const { mutate: loginUser } = useLoginUser(onSuccessCallback);

  const onChangeInput = (
    e: React.ChangeEvent<HTMLInputElement>,
    key: string
  ) => {
    setUserData({
      ...userData,
      [key]: {
        // @ts-expect-error
        ...userData[key],
        value: e.target.value,
      },
    });
  };

  const onLoginClick = () => {
    loginUser({
      identifier: userData.identifier.value,
      password: userData.password.value,
    });
  };
```

```
  const isButtonDisabled =
    !userData.identifier.value || userData.password.value.length < 6;

  return (
    <div className={styles.SignInForm}>
      <Form>
        <div className={styles.title}>Вхід в аккаунт</div>

        {Object.entries(userData).map(([key, value]) => (
          <Input
            key={key}
            borderStyle="all"
            inputWrapperHeight="m"
            placeholder={value.placeholder}
            type={value.type}
            value={value.value}
            onChange={(e) => onChangeInput(e, key)}
          />
        ))}

        <div className={styles.button}>
          <Button
            backgroundColor="black"
            disabled={isButtonDisabled}
            onClick={onLoginClick}
            size="medium"
          >
            Війти в аккаунт
          </Button>
        </div>

        <div className={styles.action} onClick={onClick}>
          Не маєте аккаунту?
        </div>
      </Form>
    </div>
  );
};
```

Лістинг файлу SignUpForm.tsx

```
// react
import { useNavigate } from "react-router-dom";
import { FC, useState } from "react";
// features
import { useRegisterUser } from "@/features/auth";
// constants
import { getHomeRoute } from "@/shared/libs/constants/routes";
// ui
import { Input } from "@/shared/ui/Input";
import { Form } from "@/shared/ui/Form";
import { Button } from "@/shared/ui/Button";
// styles
import styles from "./SignUpForm.module.scss";

interface SignUpFormProps {
  onClick: () => void;
}

export const SignUpForm: FC<SignUpFormProps> = ({ onClick }) => {
  const navigate = useNavigate();

  const [userData, setUserData] = useState({
```

```
  username: {
    value: "",
    type: "text",
    placeholder: "Введіть Юзернейм",
  },
  email: {
    value: "",
    type: "email",
    placeholder: "Введіть Email",
  },
  password: {
    value: "",
    type: "password",
    placeholder: "Введіть Пароль",
  },
});

const onSuccessCallback = () => {
  navigate(getHomeRoute());
};

const { mutate: registerUser } = useRegisterUser(onSuccessCallback);

const onChangeInput = (
  e: React.ChangeEvent<HTMLInputElement>,
  key: string
) => {
  setUserData({
    ...userData,
    [key]: {
      // @ts-expect-error
      ...userData[key],
      value: e.target.value,
    },
  });
};

const onRegisterClick = () => {
  registerUser({
    email: userData.email.value,
    password: userData.password.value,
    username: userData.username.value,
  });
};

const isButtonDisabled =
  !userData.email.value ||
  userData.password.value.length < 6 ||
  !userData.username.value;

return (
  <div className={styles.SignUpForm}>
    <Form>
      <div className={styles.title}>Реєстрація</div>

      {Object.entries(userData).map(([key, value]) => (
        <Input
          key={key}
          borderStyle="all"
          inputWrapperHeight="m"
          placeholder={value.placeholder}
          type={value.type}
          value={value.value}
```

```
          onChange={(e) => onChangeInput(e, key)}
        />
      ))}

      <div className={styles.button}>
        <Button
          backgroundColor="black"
          disabled={isButtonDisabled}
          onClick={onRegisterClick}
          size="medium"
        >
          Зареєструватись
        </Button>
      </div>

      <div className={styles.action} onClick={onClick}>
        Вже зареєстровані?
      </div>
    </Form>
  </div>
 );
};
```

Лістинг файлу CartPage.tsx

```
// react
import { FC } from "react";
// ui
import { CartProducts } from "../CartProducts/CartProducts";
// styles
import styles from "./CartPage.module.scss";

interface CartPageProps {}

export const CartPage: FC<CartPageProps> = ({}) => {
 return (
  <div className={styles.CartPage}>
   <CartProducts />
  </div>
 );
};
```

Лістинг файлу CartProducts.tsx

```
// react
import { FC } from "react";
// widgets
import { EmptyProductsError } from "@/widgets/emptyProductsError";
// enteties
import { CartItemList, ICart } from "@/enteties/cart";
import { getUserState, useUserStore } from "@/enteties/user";
import { useGetProductsByIds } from "@/enteties/product";
// ui
import { Loader } from "@/shared/ui/Loader";
import { CartSubTotal } from "../CartSubTotal/CartSubTotal";
import { CartOrderButton } from "../CartOrderButton/CartOrderButton";
// styles
import styles from "./CartProducts.module.scss";

interface CartProductsProps {}

export const CartProducts: FC<CartProductsProps> = ({}) => {
 const { user } = useUserStore(getUserState);
```

```
  const result = useGetProductsByIds({
    arrayObjectWithIds: user.cart,
  });

  // @ts-expect-error
  const cartProducts: ICart[] | [] = !result?.includes(undefined) ? result : [];

  if (!user.cart.length) {
    return (
      <div className={styles.container}>
        <EmptyProductsError
          title="Ваша корзина порожня"
          description="Перейдіть то каталогу та виберіть товари"
        />
      </div>
    );
  }

  if (user.cart.length && result?.includes(undefined)) {
    return <Loader additionalClassname={styles.container} />;
  }

  return (
    <div className={styles.CartProducts}>
      <CartItemList cartItems={cartProducts} />
      <div className={styles.bottomSide}>
        <CartSubTotal cartItems={cartProducts} />
        <CartOrderButton />
      </div>
    </div>
  );
};
```

Лістинг файлу CartSubTotal.tsx

```
// react
import { FC } from "react";
// enteties
import { ICart } from "@/enteties/cart";
// styles
import styles from "./CartSubTotal.module.scss";

interface CartSubTotalProps {
  cartItems: ICart[];
}

export const CartSubTotal: FC<CartSubTotalProps> = ({ cartItems }) => {
  const subtotal = cartItems?.reduce(
    (acc, product) => acc + product?.price * product?.count,
    0
  );

  return <div className={styles.CartSubTotal}>Сумма : {subtotal} грн </div>;
};
```

Лістинг файлу CartQrderButton.tsx

```
// react
import { FC } from "react";
// features
import { useCartOrder } from "@/features/cart";
// ui
```

```
import { Button } from "@/shared/ui/Button";
// styles
import styles from "./CartOrderButton.module.scss";

interface CartOrderButtonProps {}

export const CartOrderButton: FC<CartOrderButtonProps> = ({}) => {
  const { order } = useCartOrder();

  const onOrderClick = () => {
    order();
  };

  return (
    <div className={styles.CartOrderButton}>
      <Button backgroundColor="black" onClick={onOrderClick} size="large">
        Оформити Замовлення
      </Button>
    </div>
  );
};
```

Лістинг файлу HomePage.tsx

```
// ui
import { HomeCategoryList } from "../HomeCategoryList/HomeCategoryList";
import { HomeProducts } from "../HomeProducts/HomeProducts";
import { HomeResetFilters } from "../HomeResetFilters/HomeResetFilters";
import { HomeSortByPrice } from "../HomeSortByPrice/HomeSortByPrice";
// styles
import styles from "./HomePage.module.scss";

export const HomePage = ({}) => {
  return (
    <div className={styles.HomePage}>
      <div className={styles.content}>
        <div className={styles.filters}>
          <HomeCategoryList />
          <HomeSortByPrice />
          <HomeResetFilters />
        </div>
        <HomeProducts />
      </div>
    </div>
  );
};
```

Лістинг файлу HomeCategoryList.tsx

```
// react
import { FC } from "react";
// store
import { useHomeStore } from "../../model/store/homeStore";
// actions
import { homeActions } from "../../model/actions/homeActions";
// selectors
import { getPickedCategory } from "../../model/selectors/homeSelectors";
// ui
import { CategoryItemList } from "@/enteties/category";

interface HomeCategoryProps {}

export const HomeCategoryList: FC<HomeCategoryProps> = ({}) => {
```

```
  const pickedCategory = useHomeStore(getPickedCategory);

  const onPickCategoryClick = (category: string) => {
    homeActions.setPickedCategory(category);
    homeActions.setCurrentPage(1);
  };

  return (
    <>
      <CategoryItemList
        onPickCategoryClick={onPickCategoryClick}
        pickedCategory={pickedCategory}
      />
    </>
  );
};
```

Лістинг файлу HomeSortByPrice.tsx

```
// react
import { FC } from "react";
// store
import { useHomeStore } from "../../model/store/homeStore";
// actions
import { homeActions } from "../../model/actions/homeActions";
// selectors
import { getSortValue } from "../../model/selectors/homeSelectors";
// libs
import ascIcon from "../../libs/assets/png/asc.png";
import descIcon from "../../libs/assets/png/desc.png";
// styles
import styles from "./HomeSortByPrice.module.scss";

interface HomeSortByPriceProps {}

export const HomeSortByPrice: FC<HomeSortByPriceProps> = ({}) => {
  const sortValue = useHomeStore(getSortValue);

  const handleSortByPrice = () => {
    if (!sortValue) {
      homeActions.setSortValue("asc");
    }

    if (sortValue === "asc") {
      homeActions.setSortValue("desc");
    }

    if (sortValue === "desc") {
      homeActions.setSortValue("asc");
    }
  };

  return (
    <div className={styles.HomeSortByPrice} onClick={handleSortByPrice}>
      <div>Сортувати за ціною</div>
      {sortValue && (
        <img src={sortValue === "asc" ? ascIcon : descIcon} alt="" />
      )}
    </div>
  );
};
```

Лістинг файлу HomeResetFilters.tsx

```
// react
import { FC } from "react";
// assets
import DeleteIcon from "../../libs/assets/svg/deleteIcon.svg?react";
// actions
import { homeActions } from "../../model/actions/homeActions";
// styles
import styles from "./HomeResetFilters.module.scss";

interface HomeResetFiltersProps {}

export const HomeResetFilters: FC<HomeResetFiltersProps> = ({}) => {
  const handleResetFilters = () => {
    homeActions.resetFiltersAndSort();
  };

  return (
    <div className={styles.HomeResetFilters} onClick={handleResetFilters}>
      <div>Скинути</div>
      <DeleteIcon />
    </div>
  );
};
```

Лістинг файлу HomeProducts.tsx

```
// entetites
import { useGetProducts } from "@/enteties/product";
import { ProductList } from "@/enteties/product";
// widgets
import { EmptyProductsError } from "@/widgets/emptyProductsError";
// ui
import { HomePagination } from "../HomePagination/HomePagination";
// shared
import { Loader } from "@/shared/ui/Loader";
// store
import { useHomeStore } from "../../model/store/homeStore";
// selectors
import {
  getCurrentPage,
  getPickedCategory,
  getSearchQuery,
  getSortValue,
} from "../../model/selectors/homeSelectors";
// styles
import styles from "./HomeProducts.module.scss";

export const HomeProducts = ({}) => {
  const pickedCategory = useHomeStore(getPickedCategory);

  const searchQuery = useHomeStore(getSearchQuery);

  const sortValue = useHomeStore(getSortValue);

  const currentPage = useHomeStore(getCurrentPage);

  const { data, isError, isLoading, isFetching } = useGetProducts({
    category: pickedCategory || "",
    searchQuery: searchQuery || "",
    sortValue: sortValue,
    page: currentPage,
```

```
  });

  if (isLoading || isFetching) {
    return <Loader additionalClassname={styles.loaderContainer} />;
  }

  if ((data && !data?.data.length) || isError) {
    return (
      <EmptyProductsError
        title="Сталася помилка"
        description="На жаль, не вдалося отримати товари. Перевірте свої параметри пошуку або перезавантажте
сторінку"
      />
    );
  }

  return (
    <div>
      {data && (
        <>
          <ProductList products={data.data} />
          <HomePagination pagesCount={data.meta.pagination.pageCount} />
        </>
      )}
    </div>
  );
};
```

Лістинг файлу HomePagination.tsx

```
// react
import { FC } from "react";
// store
import { useHomeStore } from "../../model/store/homeStore";
// actions
import { homeActions } from "../../model/actions/homeActions";
// selectors
import { getCurrentPage } from "../../model/selectors/homeSelectors";
// widgets
import { PaginationItemList } from "@/widgets/pagination";

interface HomePaginationProps {
  pagesCount: number;
}

export const HomePagination: FC<HomePaginationProps> = ({ pagesCount }) => {
  const currentPage = useHomeStore(getCurrentPage);

  const onChangePageClick = (page: number) => {
    homeActions.setCurrentPage(page);
  };

  return (
    <PaginationItemList
      onClick={onChangePageClick}
      currentPage={currentPage}
      pagesCount={pagesCount}
    />
  );
};
```

BACKEND

Лістинг файлу server.js

```
module.exports = ({ env }) => ({
  host: env('HOST', '0.0.0.0'),
  port: env.int('PORT', 1337),
  app: {
    keys: env.array('APP_KEYS'),
  },
  webhooks: {
    populateRelations: env.bool('WEBHOOKS_POPULATE_RELATIONS', false),
  },
});
```

Лістинг файлу.env

```
HOST=0.0.0.0
PORT=1337
APP_KEYS=UxAVDiMJTPWyJapWxZ4MDw==,Z8UyOdhEOJJSC4d1xJvTsg==,n/71P49rSSJubxePboCWWQ==,l9
CG0mcAMrDLFp4luUiB6Q==
API_TOKEN_SALT=lrTDC3QZvoGhZxeCdqZ5mA==
ADMIN_JWT_SECRET=uVL3VQ5IJBAlP4v+0szrYQ==
TRANSFER_TOKEN_SALT=zlh5ZzrBUgaNO16gBU+aPQ==
# Database
DATABASE_CLIENT=sqlite
DATABASE_FILENAME=.tmp/data.db
JWT_SECRET=KCw1OqzBR8pWyI58rpn9vQ==
```

Лістинг файлу database.js

```
const path = require('path');

module.exports = ({ env }) => {
  const client = env('DATABASE_CLIENT', 'sqlite');

  const connections = {
    mysql: {
      connection: {
        connectionString: env('DATABASE_URL'),
        host: env('DATABASE_HOST', 'localhost'),
        port: env.int('DATABASE_PORT', 3306),
        database: env('DATABASE_NAME', 'strapi'),
        user: env('DATABASE_USERNAME', 'strapi'),
        password: env('DATABASE_PASSWORD', 'strapi'),
        ssl: env.bool('DATABASE_SSL', false) && {
          key: env('DATABASE_SSL_KEY', undefined),
          cert: env('DATABASE_SSL_CERT', undefined),
          ca: env('DATABASE_SSL_CA', undefined),
          capath: env('DATABASE_SSL_CAPATH', undefined),
          cipher: env('DATABASE_SSL_CIPHER', undefined),
          rejectUnauthorized: env.bool(
            'DATABASE_SSL_REJECT_UNAUTHORIZED',
            true
          ),
        },
      },
      pool: { min: env.int('DATABASE_POOL_MIN', 2), max: env.int('DATABASE_POOL_MAX', 10) },
    },
    mysql2: {
      connection: {
        host: env('DATABASE_HOST', 'localhost'),
        port: env.int('DATABASE_PORT', 3306),
```

```
      database: env('DATABASE_NAME', 'strapi'),
      user: env('DATABASE_USERNAME', 'strapi'),
      password: env('DATABASE_PASSWORD', 'strapi'),
      ssl: env.bool('DATABASE_SSL', false) && {
        key: env('DATABASE_SSL_KEY', undefined),
        cert: env('DATABASE_SSL_CERT', undefined),
        ca: env('DATABASE_SSL_CA', undefined),
        capath: env('DATABASE_SSL_CAPATH', undefined),
        cipher: env('DATABASE_SSL_CIPHER', undefined),
        rejectUnauthorized: env.bool(
          'DATABASE_SSL_REJECT_UNAUTHORIZED',
          true
        ),
      },
    },
    pool: { min: env.int('DATABASE_POOL_MIN', 2), max: env.int('DATABASE_POOL_MAX', 10) },
  },
  postgres: {
    connection: {
      connectionString: env('DATABASE_URL'),
      host: env('DATABASE_HOST', 'localhost'),
      port: env.int('DATABASE_PORT', 5432),
      database: env('DATABASE_NAME', 'strapi'),
      user: env('DATABASE_USERNAME', 'strapi'),
      password: env('DATABASE_PASSWORD', 'strapi'),
      ssl: env.bool('DATABASE_SSL', false) && {
        key: env('DATABASE_SSL_KEY', undefined),
        cert: env('DATABASE_SSL_CERT', undefined),
        ca: env('DATABASE_SSL_CA', undefined),
        capath: env('DATABASE_SSL_CAPATH', undefined),
        cipher: env('DATABASE_SSL_CIPHER', undefined),
        rejectUnauthorized: env.bool(
          'DATABASE_SSL_REJECT_UNAUTHORIZED',
          true
        ),
      },
      schema: env('DATABASE_SCHEMA', 'public'),
    },
    pool: { min: env.int('DATABASE_POOL_MIN', 2), max: env.int('DATABASE_POOL_MAX', 10) },
  },
  sqlite: {
    connection: {
      filename: path.join(
        __dirname,
        '..',
        env('DATABASE_FILENAME', '.tmp/data.db')
      ),
    },
    useNullAsDefault: true,
  },
};

return {
  connection: {
    client,
    ...connections[client],
    acquireConnectionTimeout: env.int('DATABASE_CONNECTION_TIMEOUT', 60000),
  },
};
};
```

```
HOST=0.0.0.0
PORT=1337
APP_KEYS=UxAVDiMJTPWyJapWxZ4MDw==,Z8UyOdhEOJJSC4d1xJvTsg==,n/71P49rSSJubxePboCWWQ==,l9
CG0mcAMrDLFp4luUiB6Q==
API_TOKEN_SALT=lrTDC3QZvoGhZxeCdqZ5mA==
ADMIN_JWT_SECRET=uVL3VQ5IJBAlP4v+0szrYQ==
TRANSFER_TOKEN_SALT=zlh5ZzrBUgaNO16gBU+aPQ==
# Database
DATABASE_CLIENT=sqlite
DATABASE_FILENAME=.tmp/data.db
JWT_SECRET=KCw1OqzBR8pWyI58rpn9vQ==
```

Лістинг файлу.env

```
HOST=0.0.0.0
PORT=1337
APP_KEYS=UxAVDiMJTPWyJapWxZ4MDw==,Z8UyOdhEOJJSC4d1xJvTsg==,n/71P49rSSJubxePboCWWQ==,l9
CG0mcAMrDLFp4luUiB6Q==
API_TOKEN_SALT=lrTDC3QZvoGhZxeCdqZ5mA==
ADMIN_JWT_SECRET=uVL3VQ5IJBAlP4v+0szrYQ==
TRANSFER_TOKEN_SALT=zlh5ZzrBUgaNO16gBU+aPQ==
# Database
DATABASE_CLIENT=sqlite
DATABASE_FILENAME=.tmp/data.db
JWT_SECRET=KCw1OqzBR8pWyI58rpn9vQ==
```

ДОДАТОК Б

ВІДГУК
керівника економічного розділу
на кваліфікаційну роботу бакалавра
на тему:
«Розробка інтернет-магазину з продажу жіночого одягу
з використанням бібліотеки React»
студентки групи 121-20-1 Єріс Анастасії Євгенівни

Керівник економічного розділу Л. В. Касьяненко
доцент каф. ПЕП та ПУ, к.е.н

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота_Єpic.docx	Пояснювальна записка до кваліфікаційній роботі. Документ Word.
Кваліфікаційна робота_Єpic.pdf	Пояснювальна записка до кваліфікаційній роботі в форматі PDF
Програма	
Program.zip	Архів. Містить коди програми
Презентація	
Презентація_Єpic.pptx	Презентація кваліфікаційної роботи