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 *Проскури Віктора Сергійовича*
(ПІБ)

академічної групи *121-18-2*
(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 фарбувального обладнання 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. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

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

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-18-2* *Проскура В.С.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка інтернет-магазину з продажу фарбувального обладнання з використанням бібліотеки React*

затверджена наказом ректора НТУ «ДП» від №

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

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

Завдання прийняв до виконання _____ *Проскура В.С.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 73 с., 16 рис., 2 табл., 3 дод., 24 джерел.

Об'єкт розробки: інтернет-магазин з продажу фарбувального обладнання.

Мета кваліфікаційної роботи: розробка інтернет-магазину з продажу фарбувального обладнання 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 уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування застосунку, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження сайту, описана робота програми.

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

Практичне значення полягає у розробці веб-орієнтованої додатку, що забезпечує відвідувачам сайту доступ до каталогу товарів та оформлення замовлення.

Актуальність інтернет-магазину визначається великим попитом на онлайн послуги, що оптимізують та спрощують обслуговування клієнтів, сприяють створенню позитивного іміджу компанії.

Список ключових слів: ІНТЕРНЕТ-МАГАЗИН, САЙТ, ВЕБ-ДОДАТОК, БАЗА ДАНИХ, БРАУЗЕР, REACT.

ABSTRACT

Explanatory note: 73 p., 16 fig., 2 table, 3 extra, 24 sources.

Object of development: online store for the sale of painting equipment.

The purpose of the diploma project: development of an online store for the sale of painting equipment 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, provides a justification for the relevance of the topic and clarifies the problem. In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes the existing solutions, selects the platform for development, performs design and development of the program, describes the algorithm and structure of the system, determines the input and output data, provides characteristics of the parameters of technical means, describes the call and application load, describes the program.

The economic section determines the complexity of the developed information subsystem, calculates the cost of work to create an application and calculates the time for its creation.

The practical significance is to development of a web application that provides site visitors with access to the catalog of products and makes orders.

The relevance of the online store is determined by the high demand for online services that optimize and simplify customer service, contribute to the creation of a positive image of the company.

Keywords: ONLINE STORE, WEBSITE, WEB APP, DATABASE, BROWSER, REACT.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

БД - база даних;

ІС - інформаційна система;

ПК - персональний комп’ютер;

API - Application Programming Interface;

HTML - HyperText Markup Language.

CSS - Cascading Style Sheets;

SASS - Syntactically Awesome StyleSheets;

DOM - Document Object Model;

IT - інформаційні технології;

FE - Front-End;

BE - Back-End;

JS - JavaScript;

ВСТУП

У наш час через стрімкий розвиток Інтернету в Україні та по всьому світу спостерігається зростання популярності онлайн торгівлі. Кожен, хто має доступ в Інтернет, може замовити будь-які товари та послуги з усього світу.

Електронна комерція прискорює більшості бізнес-процесів за рахунок їх проведення електронним чином. У цьому випадку інформація передається безпосередньо до одержувача, минаючи стадію створення паперової копії на кожному етапі. Всі компоненти електронної комерції перебувають у взаємозв’язку завдяки використанню телекомунікаційних технологій і глобальної мережі Інтернет як інструменту організації єдиного інформаційного простору електронного бізнесу.

Інтернет-магазин є одним із видів електронної комерції. Для створення успішного веб-додатку необхідно враховувати оптимізацію коду проекту, швидкість завантаження сторінки та адаптивність сайту під мобільні пристрої. Користувачів також приваблює зрозуміла та опрацьована візуальна частина, якісні зображення товарів та легкість оформлення замовлення. Сайти знаходяться у відкритому доступі для будь-якого користувача, зберігають багато корисної та важливої інформації, відсутність географічних, часових і мовних бар'єрів дозволяють легко просувати товари, тому вони широко користуються попитом на сьогоднішній день.

Метою кваліфікаційної роботи є розробка інтернет-магазину з продажу фарбувального обладнання з використанням бібліотеки React.

Для досягнення мети необхідно вирішити наступні задачі:

- поглиблено вивчити технологію створення веб-сайтів;
- висунути функціональні вимоги до програмного забезпечення;
- спроектувати й створити власний інформаційний продукт.

Результатом виконання даної роботи є створення веб-додатку, що сприяє розвитку продаж та бізнесу компанії з продажу фарбувального обладнання.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

На ранніх етапах популярності інтернету більшість людей не розуміли, навіщо потрібні сайти, бо здавалося, що знаходити інформацію в книжках та купувати товари з рук в руки зручніше та легше, ніж онлайн, адже необхідність передоплати без додаткових гарантій, велика кількість шахраїв та фальшивих сайтів породжували сумніви. Проте в нинішні часи, особливо під час пандемії, послуги, що надаються в інтернеті, зокрема електронна торгівля, сильно розвинулися, тож чітко визначають важливе місце інтернет сайтів в житті людей, їхнє значення для отримання інформації, розваг, бізнесу та іншого.

У інтернет-магазинів є багато своїх переваг [3], по-перше для покупця набагато зручніше протягом усього дня мати можливість отримати всю актуальну інформацію про асортимент, який зараз є в наявності, отримати про нього актуальну інформацію та дізнатися про кращі ціни, після обрати потрібний варіант з каталогу та розрахувати вартість доставки та оформити замовлення. По-друге як для продавця також більше позитивних моментів, адже не потрібно витрачати зайві кошти на оренду землі та її обслуговування, також реклама онлайн, порівняно з рекламою на вітринах вимагає менший бюджет. Крім цього без прив’язки до місця розташування охоплюється різноманітна зацікавлена аудиторія, цей трафік можна проаналізувати та дізнатися, якій категорії людей більш актуальний товар магазину.

Зазвичай сайти можна поділити на дві основні категорії: односторінкові та багатосторінкові. Перший вид сайтів часто називають Landing page, він використовується для простого відображення інформації, наприклад портфоліо, візитки, статті, вітрини, дані про захід чи продукт, тощо. Другий вид сайтів зазвичай являє собою інтернет-магазини, сервіси для роботи з клієнтами, державні сервіси чи послуги, банківські додатки - тобто усі, що

мають багато інформації, яку потрібно структурувати. Різниця роботи між такими веб-додатками полягає в тому, що односторінкові мають лише головну сторінку та завантажують усю інформацію одразу, при цьому, у більшості випадків, вони не мають окремої бази даних. Багатосторінкові, у свою чергу, довантажують дані відповідно до дій та запитів користувача, а саме працюють з базою даних, отримують, відправляють та обробляють запити.

Статичні односторінкові сайти створюються на мові розмітки документів HTML, при цьому для створення таких веб-сторінок не застосовують бази даних та Back-end, але для написання та покращення інтерфейсу користувача використовуються різні стилі та фреймворки з шаблонами компонентів.

При цьому такі сайти мають свої переваги та недоліки. Наприклад серед позитивних моментів можна виділити те, що сайт буде без проблем працювати на локальному сервері та будь-якому хостингу, не потрібно чекати завантаження даних, розробка можлива в будь якій програмі та саме редагування зовнішнього вигляду легке та не впливає на інші сторінки. Серед негативних аспектів можна зазначити, що такий спосіб створення сайтів застарілий та використовується рідко, адже в ньому складно внести зміни до структури і зовнішнього виду сайту, бо відсутнє розподілення на компоненти, тобто якщо ми змінюємо стиль елементу сайту в одному місці, то треба змінювати це на всіх інших сторінках.

Для удосконалення сайтів до “чистого” HTML додають різні бібліотеки, мови програмування та фреймворки, які дають можливість зробити сайт динамічним та допомагає удосконалити роботу з базами даних та комфортною навігацією між сторінками та частинами додатку. Тому динамічним [5] слід називати будь-який сайт, на якому є мінімум одна динамічна сторінка - це сторінка, що формується сервером з декількох частин, може отримуватися шляхом внесення або заміни даних, які зберігається на сервері, на заготовку сторінки. Тобто на відміну від статичної динамічна сторінка збирається з даних, що знаходяться на сервері, і тільки після цього показується

користувачу. Також окрім інформаційного наповнення, динамічно можуть створюватися елементи навігації по сайту, в цьому випадку можна просто відредагувати текст для нової сторінки у базі даних, чи внести зміни в адміністративній панелі сайту, якщо вона є.

При цьому динамічні сайти також мають свої недоліки, але вони вже не такі суттєві. Наприклад, хоч у наш час існують багато фреймворків та бібліотек для більш зручного процесу розробки, для побудови динамічних сторінок необхідно використовувати додаткові програмні засоби, через це може виникнути складність в зміні чи редагуванні великої структури сайту. В даному випадку все зав'язано на програмному забезпеченні, яке об'єднує елементи дизайну і даних в один повноцінний сайт.

При розробці динамічного сайту спочатку потрібно визначити варіант зберігання інформації, яка представляється на сайті. В цьому проекті використовується база даних з MockAPI - інструмент для простого створення API-інтерфейсів, який дозволяє генерувати дані користувача та виконувати з ними операції за допомогою інтерфейсу RESTful. В ній база даних представляє собою таблицю масиву формату JSON, в якому структуровані дані товарів магазину. Також для збереження дій користувача на веб-сторінці використовується LocalStorage - це об'єкти веб-сховища, що дозволяють зберігати пари ключ/значення у браузері. При цьому об'єкт не очищує значення, які ми туди запишемо, якщо ми перезавантажимо сторінку або навіть зовсім закриємо браузер. Якщо на сайті немає розподілу на ролі та користувачів, то це зручний спосіб для зберігати певних даних між сесіями користувача. При цьому сховище прив'язане до джерела домену, протоколу чи порта. Це означає, що різні протоколи або піддомени визначають різні об'єкти сховища і вони не можуть отримати доступ до даних один одного.

Після ініціалізації та створення бази даних розробляються сторінки на відкритій бібліотеці ReactJS та уся Front-end частина додатку, яка спілкується запитами з базою даних та локальним сервером. Такий варіант розробки

динамічних веб-сторінок спрощує розробку та завантаження сторінок, завдяки чому сервер і сторінка швидко та коректно відображають контент сайту.

1.2. Призначення розробки та галузь застосування

Як об'єкт розробки інтернет-магазину з продажу фарбувального обладнання “AirBox” з використанням бібліотеки React JS та API для отримання та відправки даних користувача на сервер та покращення роботи сайту, в якому користувач в зручному інтерфейсі за допомогою головної сторінки та інструменту пошуку може знайти фарбувальні пістолети (фарбопульт) та інше якісне обладнання для фарбувальних робіт.

Розроблені частини розраховані для:

- можливості редагування контенту магазину;
- коректного відображення контенту інтернет-магазину;
- простота і комфортність користувача в доступі до каталогу товарів за рахунок оптимальних параметрів візуалізації його вмісту;
- надання користувачам сайту комфортності доступу до каталогу товарів та оформлення замовлення;
- забезпечення відвідувачам сайту гарантії безпечної обробки інформації при замовленні товару.
- зручність роботи користувача з системою: сайт має зрозумілу структуру та категорії, швидкість і безпомилковість роботи веб-додатку, відсутність помилок, оптимізація, якісний та достовірний контент;
- функціональні модулі для взаємодії з користувачем: зручні способи перегляду товару, фільтри, сортування та меню пошуку за параметром чи назвою; актуальні ціни; історія замовлень користувачів; можливість оформити замовлення та зв’язатися з адміністрацією; додавання товару до кошику;
- сайт має бути адаптований під всі види платформ та девайсів.

1.3. Підстави для розробки

Відповідно до ОПП, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

- ОПП за спеціальністю 121 «Інженерія програмного забезпечення»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № _____від____.____.2022 р;
- завдання на дипломний проект на тему «Розробка інтернет-магазину з продажу фарбувального обладнання з використанням бібліотеки React».

1.4. Постановка завдання

Завданням кваліфікаційної роботи є розробка інтернет-магазину з продажу фарбувального обладнання з використанням бібліотеки React.

Для успішної взаємодії з клієнтами та підвищення ефективності продаж інтернет-магазин повинен виконувати вимоги щодо: високої швидкості роботи сторінок та переходу між ними, зручність інтерфейсу для користувача, просте управління змістом.

Програма повинна реалізувати наступні функції:

- сайт повинен мати зручний інтерфейс;
- отримувати базу даних з товарами для інтернет-магазину;
- формувати веб-сторінки на основі шаблонів з використанням інформації, яка отримуватиметься з бази даних;
- фільтрування, сортування та пошук товарів;
- контактування за адміністраторами даного магазину;
- мати корзину та сторінку оформлення замовлення;

- мати інформаційні сторінки та сторінку з останніми замовленнями;
- сайт має бути адаптований під всі види девайсів: смартфон, комп’ютер, планшет.

Для досягнення поставленої мети в роботі необхідно:

- розробити базу даних, з якої буде братися інформація про товар.
- проаналізувати галузь предметної області в поставленій задачі;
- визначення архітектуру файлової та логічної структури сторінок;
- розробка макет з дизайном інформаційної системи та веб-сторінки;
- розробити алгоритми та модулі для реалізації поставленого завдання;
- заповнення контентом ІС та БД;
- тестування й публікація інформаційної системи в мережі Інтернет.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Інтернет-магазину з продажу фарбувального обладнання, що є метою розробки в кваліфікаційній роботі, повинен мати наступні функціональні можливості:

- ведення бази даних інтернет-магазину;
- формування веб-сторінок на основі шаблону з використанням інформації, яка отримуватиметься з бази даних;
- забезпечує велику швидкість роботи та малий час відгуку при виконанні операцій;
- зв’язок з адміністраторами магазину та онлайн підтримка;
- обробка інформації від користувача та передача відповіді;
- дані повинні вводитися шляхом вводу інформації в поля інтерфейсу та відправлятися через запит користувача на сервер;
- сайт повинен мати сторінку з останніми товари, які користувачі замовили на сайті;
- сайт повинен мати зручний інтуїтивно-зрозумілий інтерфейс;

- сайт має можливість отримати віддалений доступ до сайту через будь який веб-браузер, девайс та платформу з коректним відображенням.

1.5.2. Вимоги до інформаційної безпеки

Вимоги для забезпечення надійного функціонування та для уникнення некоректної роботи програми необхідно реалізувати:

- контроль вхідних даних;
- обробку виняткових ситуацій;
- виведення повідомлень про помилки;
- можливість повторного введення даних;
- можливість безперервної роботи протягом не менше 120 годин (5 діб);
- платформна незалежність;
- забезпечення неушкодженого стану даних, що зберігаються в базі даних, у випадку відмови застосунку.

Веб-додаток має передавати інформацію користувача використовуючи спеціальні техніки, щоб зловмисники не могли отримати доступ до особистих даних користувача. Для зберігання інформації про товари з корзини, яка може бути підв’язана до браузера користувача використовується Local Storage браузера.

1.5.3. Вимоги до складу та параметрів технічних засобів

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 нормального функціонування програмного модулю, повинні виконуватися певні вимоги до технічних засобів.

- процесор класу Intel Pentium з тактовою частотою не менш 2.4 ГГц;
- не менше 2 GB оперативної пам'яті;

- монітор з діагоналлю не менше 11";
- 10 Гб вільного місця на жорсткому диску;
- 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.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для нормального функціонування програми необхідно, щоб програмне забезпечення персонального комп'ютера, на якому буде функціонувати веб-орієнтована система, відповідало наступним вимогам:

- операційна система Windows (7+), Linux, MacOs;
- веб-браузер Microsoft Edge /Google Chrome / Firefox / Opera / Safari.

Сайт повинен бути реалізований на мові програмування JavaScript з використанням бібліотеки ReactJS.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

За завданням кваліфікаційної роботи було реалізовано розробку інтернет-магазину з продажу фарбувального обладнання з використанням бібліотеки React.

Призначення розробленої системи:

- систематизування та групування даних та товарів;
- організація даних в інтерфейс для комфорту клієнту;
- надання можливість користувачам легко обрати потрібні товари;
- маніпулювання з базою даних товарів інтернет-магазину;
- надати можливість обмінюватися даними та запитами між користувачем та адміністратором.

Розроблений додаток має наступні функції та можливості:

- формування веб-сторінки на основі шаблонів з використанням інформації, яка отримується і передається з бази даних;
- форма контакту з адміністраторами сайту;
- має можливість сортування, фільтрації та пошуку товарів;
- має корзину та сторінку оформлення замовлення, список останніх замовлень всіх користувачів;
- сайт адаптований під всі види девайсів користувачів.

Для досягнення поставленої задачі розроблене програмне забезпечення підтримує виконання таких операцій:

- надання віддаленого доступу до застосунку через веб-браузер через девайс користувача;
- зчитування вхідних даних з сервісу бази даних;
- формування замовлення та відправлення до бази даних.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної галузі розв'язуваної задачі не передбачають застосування математичних методів, при розробці інтернет-магазину з продажу фарбувального обладнання математичні методи не використовувалися.

2.3. Опис використаної архітектури та шаблонів проектування

Для роботи з сервером використовується Fetch API та JavaScript бібліотека Axios. Fetch являє собою нативний низькорівневий інтерфейс JS для виконання HTTP-запитів з використанням обіцянок через глобальний метод fetch(). В свою чергу Axios є JavaScript бібліотекою, яка використовується на основі обіцянок для виконання HTTP-запитів, вони відмінно підходить для роботи з сервером через архітектуру REST (Representational State Transfer - передача стану уявлення).

Архітектура REST [2] для моделювання роботи з сервером була обрана як найпопулярніша та найзручніша. Вона визначає, як компоненти розподіленої системи повинні взаємодіяти один з одним. У загальному випадку архітектура складається з клієнтів та серверів, взаємодія відбувається за допомогою запитів-відповідей. Клієнти ініціюють запити до серверів, а сервери обробляють запити та повертають відповідні відповіді. Запити та відповіді створюються на основі передачі уявлень ресурсів. Запити та відповіді найчастіше відправляються за протоколом HTTP (HyperText Transfer Protocol - "протокол передачі гіпертексту").

RESTful API [6] — це стандарт, реалізований за допомогою HTTP та принципів REST, який використовується при розробці API для програмного забезпечення, додатків та веб-сервісів для полегшення управління ресурсами. Націлюються як текст, посилання чи файли, включаючи стани ресурсів, які форматуються та передаються через HTTP. RESTful є одним із найбільш часто

використовуваних стилів дизайну API сьогодні, щоб дозволити різним додаткам спілкуватися один з одним.

За допомогою інструменту MockAPI, який використовується в додатку, можна легко та швидко створити API-інтерфейси та виконувати з ними операції за допомогою інтерфейсу RESTful. Після отримання даних ними можна маніпулювати та виводити на сторінці.

Найважливішою функцією RESTful є вказувати, як використовувати методи HTTP і як форматувати URL-адреси для веб-програм для керування ресурсами. Методи, які можна використовувати:

- GET: для отримання даних;
- POST: для запису даних;
- PUT: для оновлення даних;
- DELETE: видалення даних.

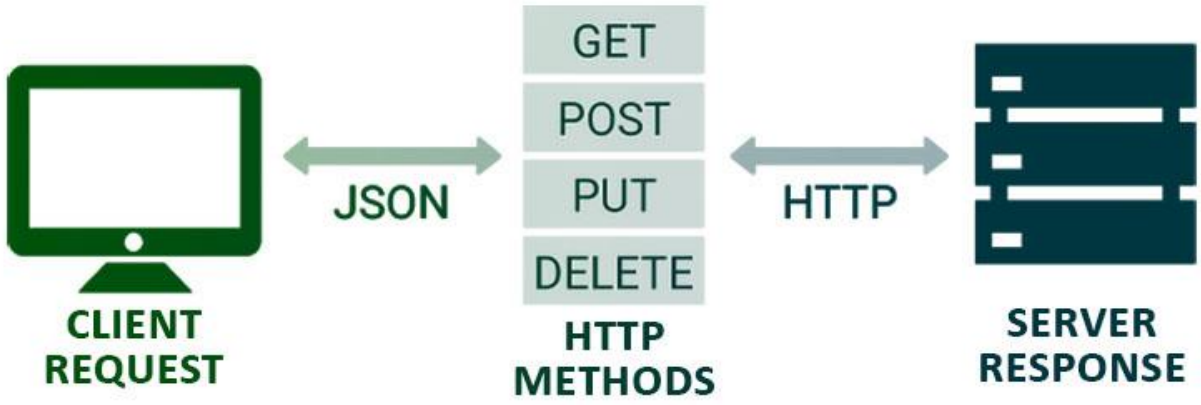

Рис. 2.1. Схема роботи архітектури REST

До важливих особливостей також відноситься використання віртуального DOM (Virtual DOM). Віртуальний DOM - об'єкт, в якому зберігається інформація про стан інтерфейсу. Використання віртуального DOM дозволяє бібліотеці ефективно оновлювати реальний DOM.

2.4. Опис використаних технологій та мов програмування

Застосунок має бути реалізовано на мові програмування JavaScript з використанням бібліотеки React JS, розмітки HTML та стилями SCSS.

Основою будь-якої веб-сторінки є мови розмітки, вони використовуються для структурування і відображення веб-сторінки та її контенту, тобто щоб повідомляти вашому браузеру те, як відображати веб-сторінки, які ви відвідуєте.

Існує досить багато мов розмітки, існує навіть можливість зробити свою власну, проте з низки чинників найпопулярнішою є мова HTML. HyperText Markup Language є стандартним для інтернет-документів, для роботи з текстом, початок та кінець кожного елемента позначається спеціальними тегами, укладеними у кутові дужки, браузер слідує цим правилам перетворення, які трохи відрізняються в залежності від системи, версії та типу оглядача. Якщо в коді припущено помилку, то на сторінці може з'явитися неправильне відображення.

На сьогоднішній день випущено п'ять версій мови, перша версія була розроблена між 1986 і 1991 роками, а остання версія 5.2 була випущена у 2017 році. Спочатку вона мала стати незалежною від будь-яких платформ та відображатися скрізь однаково. Але цього не сталося бо у користувачів зростали вимоги до мультимедіа. Як результат, код інтерпретується по-різному не лише на різних пристроях, а й у різних браузерах. У п'ятій редакції HTML став більш незалежним від інших мов. Раніше перевірка правильного заповнення форм була лише в частині JavaScript, а тепер частину завдань наприклад визначити тип даних в поля вводу, можна перекласти на HTML.

Також виділяються популярні мови розмітки XHTML та XML. Перша має більшу синтаксичну строгість, що є сильним плюсом та мінусом, адже правильно складеними сторінками простіше керувати і змінювати їх за допомогою програми, але створювати людині їх важче. XML розхвалюється як революційна технологія розмітки, вона дозволяє розробникам створювати

та задавати їх власну мову розмітки, створюючи власні теги, але через занадто велику кількість індивідуальних мов на базі XML його було обмежено, і більшість розробників користуються більш загальноприйнятими мовами.

Для зміни зовнішнього вигляду документа, застосовується мова ієрархічних правил, тобто таблиць стилів - Cascading Style Sheets (CSS). Він описує як елемент повинен відображатися на екрані. Створити сторінку і оформити її можна без використання таблиць, прописуючи візуальні властивості кожного елемента в його описі, але якщо сторінок дуже багато, застосовувати такий метод незручно, адже при зміні оформлення доводиться міняти безліч документів, до того ж це погіршує читаність верстки. Тому завдяки гнучкості та різноманітності можливостей каскадних таблиць. використання CSS вважається стандартом оформлення сайтів.

Для збільшення рівня абстракції CSS-коду та спрощення файлів каскадних таблиць стилів використовувався модуль SASS. Мова SASS має два синтаксиси, Sass відрізняється відсутністю фігурних дужок, у ньому вкладені елементи реалізовані з допомогою відступів та не потрібно ставити крапку з комою після кожної властивості, але більшість розробників не сприймали такий метод, адже багато в чому він був незвичним на погано читаний, тому був розроблений другий SCSS (Sassy CSS), який використовує фігурні дужки, як і сам CSS, він і був використаний під час розробки.

Також Scss додає до CSS константи та домішки по типу функцій. Це полегшує підтримку цілісності даних у великому наборі стилів. Константи дозволяють встановити значення та використовувати його всередині стилів, за допомогою домішок це саме можна зробити з блоком атрибутів стилю. Різницю між CSS та SCSS можна побачити на рис. 2.2. [10].

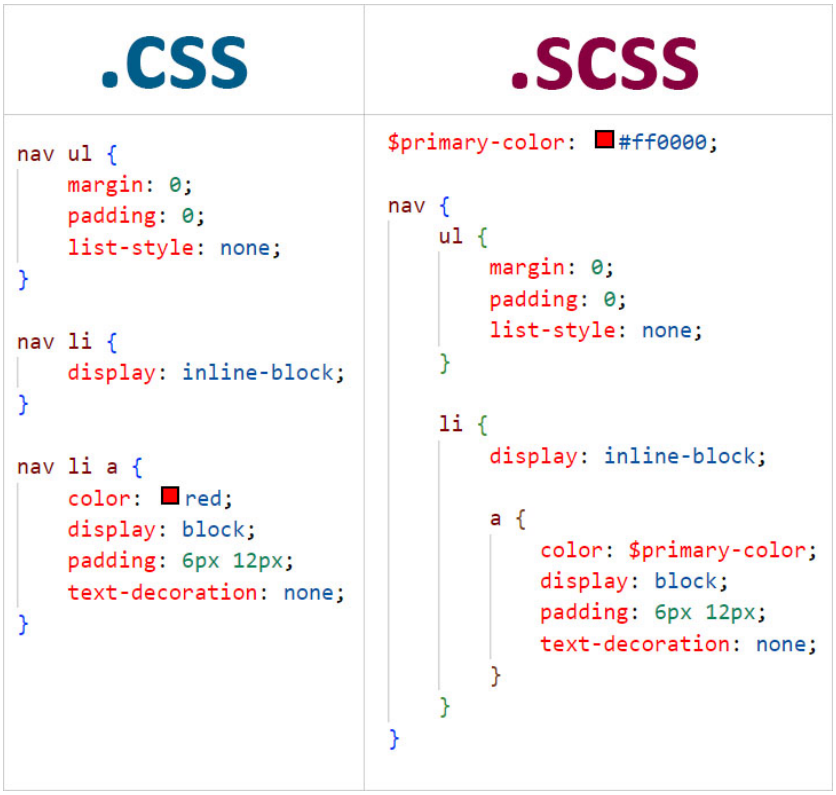

Рис. 2.2. Порівняння CSS та SCSS

Була застосована технологія CSS-модулів [7] — це задача, яка запускається на стадії складання проекту, у процесі виконання якої імена класів та селектори змінюються так, щоб утворилася свого роду локальна область видимості на зразок простору імен.

Ось так зазвичай прописуються класи в HTML та CSS:

```
<h1 className="title">Приклад заголовку</h1>
```

CSS-модулі використовують інший підхід. У процесі збирання компілятор проаналізує styles.scss, який ми імпортували, потім проаналізує JavaScript і зробить клас .title доступним через styles.title. Потім згенерує на їх основі нові HTML та CSS-файли, вже з новими класами.

```
import styles from "./styles.scss";
function Title() {
    return <h1 className={styles.title}>Приклад заголовку</h1>;
}
```

Після згенерований HTML може виглядати так:

```
<h1 class="_styles__title__qn0YPT">Приклад заголовку</h1>
```

Значення атрибута class та селектор .title замінені на нові. При цьому вихідний CSS до браузера взагалі не потрапляє.

Цей підхід був розроблений для вирішення проблеми глобальної області видимості в CSS. Крім цього CSS-модулі гарантують, що всі стилі одного компонента:

- Знаходяться в одному місці;
- Застосовуються тільки для одного компонента.

JavaScript є мовою програмування веб-мережі, при цьому вона являється високорівневою, динамічною та інтерпретованою, добре підходить для об'єктно-орієнтованого та функціонального стилів програмування. На більшій частині веб-сайтів використовують JavaScript, а всі сучасні браузери на сучасних девайсах включають інтерпретатори JavaScript, роблячи JS однією з найпоширеніших мовою програмування, та дивлячись на тенденції в останні роки, згідно опитуванню українських програмістів від DOU [4], кількість JS розробників тільки збільшується, адже мова програмування постійно розвивається.

Рис. 2.3. Опитування по використанню мов програмування

Назва JavaScript з’явилася не відразу, перші передумови з'явилися ще в 1992 році, коли було розпочато розробку скриптової мови Cmm. Пізніше він був перейменований в ScriptEase, з 1995 року мова називалася Mocha, потім

LiveScript та нарешті отримала своє сучасне ім'я. Тоді щоб залучити більше розробників для роботи з новою мовою, було вирішено використати його назву Java, хоч це і різні інструменти для розробки.

Остання на сьогодні версія мови ES6 вийшла у 2015 році, що було проривним в її розвитку, бо з'явилися нові стандарти та можливість роботи, наприклад з константами. Зазнав змін і сам код, мова стала дотримуватися принципу скорочення коду задля більшої функціональності. Хоч в першу чергу JavaScript широко використовується у веб-розробці, працюючи у поєднанні з HTML та CSS, мова також часто використовується в створенні мобільних додатків на React Native, в серверній Back-end розробці за допомогою Node.js та для створення комп’ютерних програмах.

Є багато факторів, чому JavaScript такий популярний, адже у нього багато переваг:

- підтримка всіма відомими браузерами;
- продуктивність та швидкість роботи;
- інтеграція з розміткою та стилями сторінки (HTML, CSS) та частиною серверу;
- велика кількість бібліотек та фреймворків;
- для складних завдань існують готові патерни та шаблони для вирішення, а прості виконуються швидко

Серед недоліків та обмежень можна виділити:

- відсутнє читання та завантаження файлів;
- довільна нестрога типізація даних, наприклад при порівнянні даних може виникнути помилка та повернеться невірне значення, про що програма не скаже та буде працювати так, наче нічого не відбулося, але до цього звикаєш, або можна використовувати TypeScript, але це вже інша історія;
- не краща система безпеки та відсутня підтримки віддаленого доступу.

Щоб суттєво полегшити процес розробки та об'єднання певних компонентів під час створення програм створюються спеціальні бібліотеки та

фреймворки. Вони створені як основа, котра дозволяє додавати компоненти в залежності від потреб, формується програма будь-якого призначення досить швидко і без особливих труднощів, тим самим зменшує час і витрати на розробку проектів, адже можна сконцентруватися на логіці, а не на створенні блоків та компонентів дизайну.

У наш час існують сотні фреймворків, особливо популярні з них лише десятки. Серед JavaScript фреймворків та бібліотек особливо популярні:

- React JS;
- Vue JS;
- Angular JS.

Вони займають провідну позицію серед JavaScript фреймворків. Згідно ресурсу npm-trends (рис. 2.4) [8], який відображає кількість завантажень пакетів npm, які надходять з API, можемо побачити популярність трендів за останні п’ять років та зростання популярності фреймворків с роками, де React займає лідируючу позицію. Також в останні роки зростає популярність Vue JS, а Angular JS займає останню сходинку серед трендів, адже він самий важкий для освоєння, тому не приваблює нових користувачів.

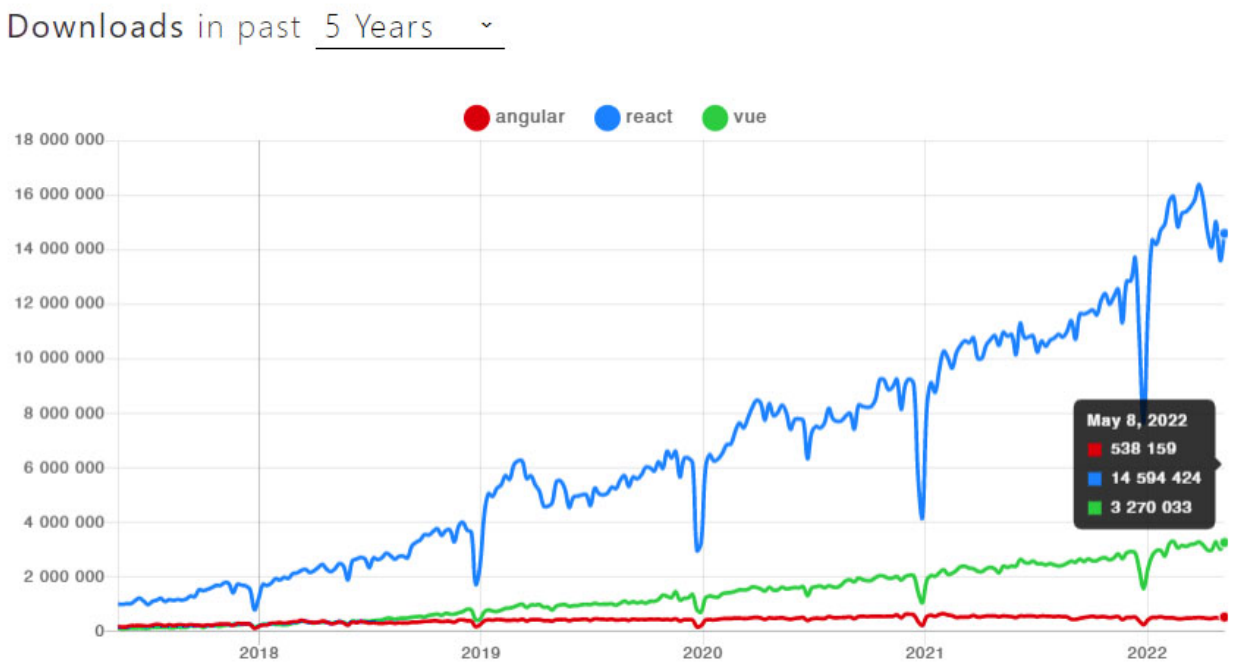

Рис. 2.4. Статистика завантажень бібліотек

React [9] являється дуже популярною бібліотекою, вона розроблена та підтримується компанією Facebook. Бібліотеку почали використовувати на сайті цієї соціальної мережі у 2011 році, та використовували для відображення практично всіх блоків сайту Facebook, а у 2013 році Facebook відкрив вихідний код React.

В React весь код знаходиться всередині функції App, вона та інші схожі функції називаються компонентами - це фрагмент інтерфейсу, який містить розмітку і пов'язану з нею логіку, якщо така існує. Хоч всі програми React будуються на компонентах, сам компонентний підхід з'явився задовго до React, але його поєднали з декларативністю. Об'єднуючи компоненти, розробник створює завершений інтерфейс веб-додатку.

Також за допомогою React розробники мають можливість створювати веб-додатки, які змінюють відображення без перезавантаження сторінки. Завдяки цьому додаток не завантажує одні данні при переході з однієї сторінки на іншу та швидко реагує на дії користувача, наприклад на заповнюючи форми, фільтруючи сторінку, на додавання товарів у кошик та інше. Крім цього з React використовують бібліотеки для управління станом і роутінгом, наприклад, Redux або React Router, які дозволяють автоматично робити повторний рендер компонентів при зміні значень реактивних змінних. Це дозволяє не оновлювати сторінку кожен раз при зміні інформації на ній, а змінювати елементи окремо один від одного, що також потребує проробки.

Однією важливою особливість React є використання JSX, який розширює синтаксис JavaScript, щоб зручно використовувати для опису інтерфейсу. Простими словами JSX дозволяє писати та додавати HTML у коді React, адже компоненти JavaScript коду практично повністю повторюють HTML розмітку, чим спрощує розробку. Наочне порівняння як визначити формат файлу наведено на рис. 2.5.

index.js

```
const Component = () => {
    return 'Это простая строчка'
}
```

index.jsx

```
import React from 'react';

const Component = () => {
    return <h1>А это уже JSX-разметка</h1>
}
```

Рис. 2.5. Порівняння JS та JSX

Як правило це більше потрібно для редактора, щоб він зміг коректно читати та підсвічувати код, який містить окрім звичайного JS, ще й JSX-розмітку. Звичайно можна не писати .jsx і програма буде збиратися та працювати нормально і навіть редактор може не скаржитися, але бажано дотримуватися правил для вірного синтаксису, та щоб розробник розумів, що у цьому файлі зберігається ще й JSX-розмітка. Адже більшість розробників цінує його за наочність при роботі з інтерфейсом в JavaScript коді. Крім цього, JSX допомагає React робити повідомлення про помилки та попередження зрозуміліше.

2.5. Опис структури програми та алгоритмів її функціонування

Структура проекту роботи складається з папок, що мають різне функціональне значення, та файлів, що відповідають за функціональність, стилями з оформленням і адаптивним дизайном інтернет-магазину та конфігурацію роботи проекту. Структура додатку наведена на рис. 2.6.

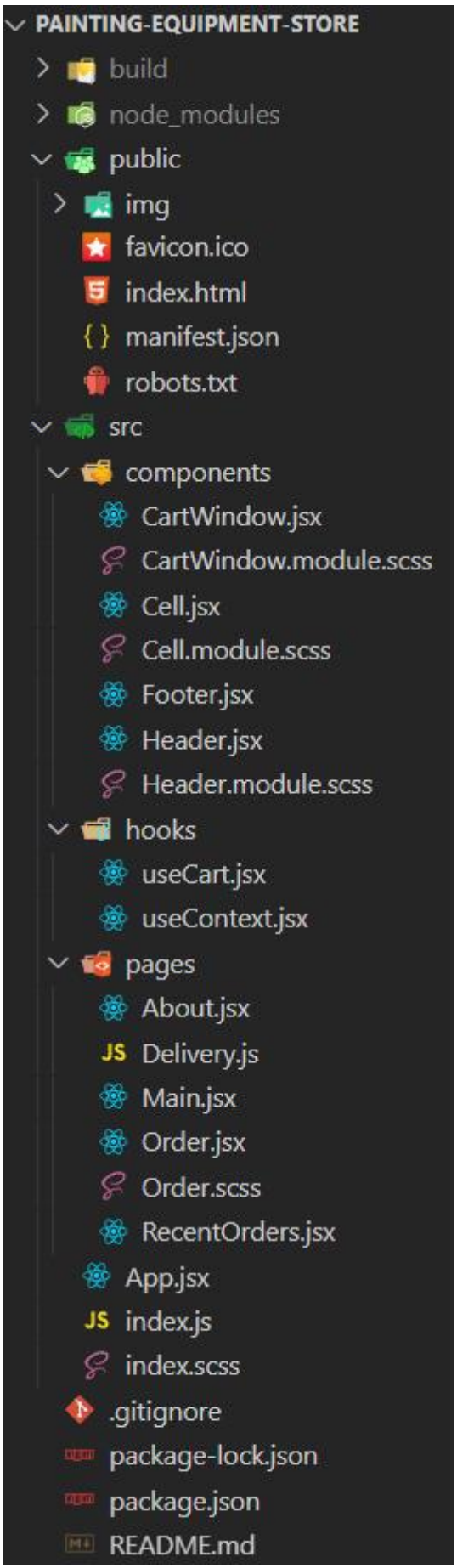

Рис. 2.6. Структура папок та файлів проекту

Короткий опис файлів структури:

- Папки build та node_modules мають службове призначення та не несуть сенсу для бізнес логіки проекту, в них знаходяться скомпільована програма та пакети для роботи React;
- Папка img (знаходиться в папці public) містить всі зображення, які використовуються на сайті;
- Файл favicon.ico – іконка що відображається у вікні браузеру;
- Файл index.html – головний файл додатку, у який React монтує усю логіку проекту;
- Файл manifest.json надає інформацію про програму;
- Файл robots.txt – індексний текстовий документ, дає пошуковим роботам рекомендації, які сторінки чи файли варто сканувати;
- Папка components в директорії src має всі окремі компоненти, що використовуються та вбудовуються на сайті;
- Файли CartWindow містять в собі інформацію про вікно корзини;
- Файли Cell містять в собі інформацію про вид відображення окремого товару;
- Файл Footer.jsx містить в собі інформацію про нижній блок сторінки;
- Файли Header містять в собі інформацію про шапку сторінки;
- Папка hooks має в собі всі хуки контексту, які передають інформацію між сторінками сайту, що використовуються в додатку;
- Папка pages має всі окремі сторінки, що використовуються та вбудовуються на сайті;
- Файл Delivery.js містить в собі інформацію про доставку та оплату товарів;
- Файл About.jsx містить в собі інформацію про контактні дані та вікно зворотного зв’язку в адміністраторами сайту;
- Файл Main.jsx містить в собі інформацію про головну сторінку сайту;
- Файли Order містять в собі інформацію про сторінку оформлення замовлення;

- Файл RecentOrders.jsx містить в собі інформацію про сторінку останніх замовлень всіма користувачами сайту;
- Файл App.jsx об’єднує всі React компоненти та сторінки;
- Файл index.scss – загальний файл стилів для усього додатку;
- Файл index.js – загальний функціональний файл, у якому відбувається монтування усього додатку;
- Файл .gitignore містить в собі налаштування системи контролю версій Git;
- Файли package-lock.json, package.json містить в собі інформацію про конфігурації усього проекту, які зберігають зв’язки з необхідними бібліотеками, що пакет NPM встановлює для коректної роботи веб-додатку.

Опис структури бази даних.

База даних запитується з серверу методом GET в форматі JSON. Кожен елемент має первинний ключ – атрибут чи група атрибутів, що єдиним чином ідентифікують кожен елемент. Крім цього є інші атрибути, які місять дані про інформаційне відображення властивостей об'єкта. Після запиту з БД отриманими даними можна маніпулювати та виводити на сторінку.

Таблиця 2.1. містить в собі інформацію про властивості, які мають товари на головній сторінці веб-додатку.

Таблиця 2.1.

Опис властивостей товарів

Товари (airbrush)		
№	Поле	Описання
1.	Id	Ідентифікатор товару
2.	title	Назва товару
3.	cost	Ціна товару
4.	imgUrl	Посилання на зображення товару

Таблиця 2.2. містить інформацію та властивості про користувача, який оформив замовлення та товари, які були в цей час у нього в корзині.

Таблиця 2.2.

Опис властивостей замовлень

Замовлення (orders)		
№	Поле	Описання
1.	id	Ідентифікатор замовлення
2.	orderItems	Товари додані до корзини
3.	userData	Вся інформація про користувача
4.	firstName	Ім’я
5.	lastName	Прізвище
6.	phoneNumber	Номер телефону
7.	email	Електронна пошта
8.	city	Місто
9.	novaposhta	Відділення нової пошти
10.	comment	Коментар до замовлення

Для запуску додатку на локальному сервері потрібно зібрати компоненти та бібліотеки за допомогою команди npm install, а після викликати команду npm start або npm run build для запуску проекту або створення зібраного каталогу проекту відповідно.

Після чого завдяки збирачеві модулів усі файли починають взаємодію між собою, під час його React починає процедуру створення Virtual DOM дерева для монтування компонентів на сторінку. Усі супутні бібліотеки також починають монтуватися у проект. Так, SCSS modules перетворюються у звичайні CSS стилі, але вже з згенерованими ключами до відповідних компонентів.

При завантаженні сторінки завдяки хуку useEffect або спеціальним функціям та запитами Axios додаток отримує, обробляє та зберігає дані з бази даних. На цьому етапі завантаження сторінки завершується, React реагує на дії, які користувач робить на веб-додатку та відображає відповідні дані або сторінки.

Розглянемо деякі ключові елементи функціювання веб-додатку.

Файл index.html у папці public має у собі елемент, у який React монтує усі компоненти. Також цей файл містить усі необхідні підключення та налаштування для сторінки, яка буде відображатися у браузері.

```
<!DOCTYPE html>
<html lang="en">
     <head>
       <meta charset="utf-8" />
       <meta name="viewport" content="width=device-width,
     initial-scale=1" />
       <meta name="theme-color" content="#000000" />
       <meta name="description"
         content="Thesis on the creation of a website with the
     sale of painting equipment, created using create-react-app"
     />
       <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
      //...font link
       <title>AirBox - покрасочное оборудование</title>
     </head>
     <body>
       <div id="root"></div>
     </body>
</html>
```

Файл index.js у папці src відтворює монтування бібліотекою React версії v.18 усіх компонентів у компонент div, що знаходиться у головному та єдиному html файлі проекту.

```
import React from 'react';
import { createRoot } from "react-dom/client"
```

```
import { BrowserRouter as Router } from 'react-router-dom';
import './index.scss';
import App from './App';
const root = createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <Router>
      <App />
    </Router>
  </React.StrictMode>,
);
```

У файлі App.jsx ми збираємо усі компоненти веб-додатку. Якраз цей єдиний файл ми імпортували до index.js. В них використовується та налаштовується бібліотеку React Router, що дозволяє зв'язувати сторінки сайту з відповідними адресами. Тобто в залежності від URL адреси, на якій знаходиться користувач, буде відображено відповідний інтерфейс. Також це дає змогу переключати сторінки сайту без перезавантаження, адже на сторінці буде змінюватися лише контент.

Крім цього у файлі App.jsx ідентифікуються та зберігаються основні хуки стану та контексту, функції обробки інформації, які передаються звідти до інших сторінок або компонентів через хуки контексту та пропси. Також при завантаженні усього сайту ми робимо асинхронний запит Axios необхідну інформацію з бази даних, яку далі записуємо в хук airbrush для відображення на сторінці. А якщо користувач відвідує інтернет-магазин вперше, то створюється елемент в локальному сховищі, звідти завантажується інформація про товари в корзині та відображається на сторінці, це дозволяє користувачу перезавантажувати сторінку та зберігати зміни на сайті.

```
import React from 'react';
import axios from 'axios';
import { Routes, Route } from 'react-router-dom';
import StoreContext from './hooks/useContext';
import Header from './components/Header';
```

```
import Footer from './components/Footer';
import CartWindow from './components/CartWindow';
import Main from './pages/Main';
import Delivery from './pages/Delivery';
import About from './pages/About';
import Order from './pages/Order';
import RecentOrders from './pages/RecentOrders';
function App() {
  const [addCart, setAddCart] = React.useState([]);
  const [airbrush, setAirbrush] = React.useState([]);
  const [searchInput, setSearchInput] = React.useState('');
  const [openCart, setOpenCart] = React.useState(false);
  const [loadingReady, setLoadingReady] = React.useState(true);
  React.useEffect(() => {
    async function fetchData() {
      try {
        setLoadingReady(true);
        const itemsResponse = await
axios.get(`https://62792bd2d00bded55ae56077.mockapi.io/airbrush`
);
        setLoadingReady(false);
        setAirbrush(itemsResponse.data);
      } catch (error) {
        alert('Ошибка при запросе данных с сервера :(\nОбновите
страницу или попробуйте позже.');
        console.error(error);
      }
    }
    fetchData();
    if (!localStorage.getItem('Cart')) {
      localStorage.setItem('Cart', JSON.stringify([]));
    }
    setAddCart(JSON.parse(localStorage.getItem('Cart')));
  }, []);
  //...functions
```

```
  return (
    <>
      <StoreContext.Provider value={{ addCart, loadingReady,
isItemAdded }}>
        {openCart && < CartWindow onClickCart={() =>
setOpenCart(false)} onRemove={onRemoveCart} /> }
        <Header onClickCart={() => setOpenCart(true)}
onChangeSearch={onChangeSearch} searchInput={searchInput}
clearSearch={() => setSearchInput('')} />
        <div className='container'>
          <Routes>
            <Route exact path="/" element={<Main
              searchInput={searchInput}
              airbrush={airbrush}
              onAddToCart={onAddToCart}
              findItemMenu={findItemMenu}
              setAirbrush={setAirbrush}
            />} />
            <Route exact path='delivery' element={<Delivery />}
/>
            <Route exact path='about' element={<About />} />
            <Route exact path='order' element={<Order
setAddCart={setAddCart} />} />
            <Route exact path='recent-orders'
element={<RecentOrders />} />
            <Route path='*' element={<h1}}>404 not found</h1>}
/>
          </Routes>
        </div>
        <Footer />
      </StoreContext.Provider>
    </>
  );
}
export default App;
```

В файлі Main.jsx відображається вся інформація з головної сторінки, а саме елементи фільтрації та картки товарів. Дані до останніх передаються в Cell.jsx через пропси та useContext з App.jsx. За допомогою перебору елементів об’єкту з товарами на сайті відображаються всі товари з БД. Схожа система використовується й в інших міста, де потрібно відобразити список елементів. Наприклад в RecentOrders.jsx, CartWindow.jsx та Order.jsx.

```
import React from 'react';
import Cell from '../components/Cell';
import StoreContext from '../hooks/useContext';
function Main({ searchInput, airbrush, onAddToCart,
findItemMenu, setAirbrush }) {
    const { loadingReady } = React.useContext(StoreContext);
    const [filterOrder, setFilterOrder] = React.useState(true);
    //...sort function
    const itemsRender = () => {
        const airbrushFilter = airbrush.filter(item =>
item.title.toLowerCase().includes(searchInput.toLowerCase()))
        return (loadingReady ? [{ id: 1 }, { id: 2 }, { id: 3 },
{ id: 4 }, { id: 5 }, { id: 6 }, { id: 7 }, { id: 8 }, { id: 9
}, { id: 10 }]
            : airbrushFilter).map((cardObj) => (
                <Cell
                    key={cardObj.id}
                    {...cardObj}
                    handleClickAdd={(item) => onAddToCart(item)}
                />));
    };
    return (
        <>
          //...sort items
            <div className='disFlex'>
                {itemsRender()}
            </div>
        </>
```

```
    );
}
export default Main;
```

В файлі About.jsx відображається інформація про компанію та форма зворотнього зв’язку. За допомогою комплекту для розробки emailjs дані вписані у форми input відправляються на пошту адміністратора.

```
import React, { useRef } from "react";
import emailjs from '@emailjs/browser';
function About() {
    const form = useRef();
    const sendEmail = (e) => {
        e.preventDefault();
        emailjs.sendForm('service_hangal9', 'template_4lxgeci',
form.current, 'To8UwZDUHt8iKaYbP')
            .then((result) => {
                console.log(result.text);
                alert('Ваше сообщение отправлено!')
            }, (error) => {
                alert('Ошибка при отправке сообщения:(')
                console.log(error.text);
            });
    };
    return (
        <div className="about">
            ...
            <form ref={form} onSubmit={sendEmail}
className="contactUs" >
            <input name="name" placeholder="Имя" />
            <input name="email" type='email'
placeholder="Почтовый адрес (email)" />
            <textarea name="message" placeholder="Ваше
сообщение" />
                <button>Отправить сообщение</button>
            </form>
        </div>
```

```
    );
}
export default About;
```

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Програма отримує вхідні дані шляхом завантаження інформації із бази даних, локального сховища та через передачу значень через глобальні змінні додатку інтернет-магазина.

Вхідні дані:

- інформація про користувача та його замовлення при оформленні замовлення;
- натискання на візуальні елементи управління застосунку;
- інша інформація, що вводиться користувачем із клавіатури.

Вихідні дані:

- по заздалегідь створеному шаблону виводяться всі товари, які користувач обрати і замовити, під час вибору товару відбувається запис до локального сховища;
- web-сторінки інтернет-магазину, на яких він може здійснювати стандартні браузерні операції.

2.7. Опис розробленого програмного продукту

2.7.1. Використані технічні засоби

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. Програма створена в середовищі VS Code з використанням бібліотеки React.

Сайт розроблено на персональному комп’ютері:

- Операційна система Windows 11;
- процесор класу AMD Ryzen 5 3600 3.6(4.2)GHz, TSMC 7nm FinFET;
- материнська плата Asus Prime B450M-K (2 PCI-E x1, 1 PCI-E x16, 1 M.2, 2 DDR4 DIMM, Audio, Video, Gigabit LAN);
- два модулі пам’яті Kingston HyperX KHX3200C18D4/8GB DDR4-3200 DDR4 SDRAM ;
- Відеоадаптер NVIDIA GeForce GTX 1660 Ti (6 ГБ);
- SSD диск KINGSTON SA1000M8 240G, HDD диск WDC WD10EZEX-08WN4A0;
- рідкокристалічний IPS монітор з діагоналлю 24";
- з доступом до мережі Internet;
- маніпулятор "миша";
- клавіатура.

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.

2.7.2. Використані програмні засоби

Під час розробки даного застосунку були використані такі програмні засоби:

- Visual Studio Code (VS Code);
- Браузер Microsoft Edge;
- Операційна система: Windows 11.

Visual Studio Code − редактор коду, розроблений компанією Microsoft, який позиціонується як простий редактор коду для кросплатформної розробки веб-додатків та хмарних програм.

Можливості роботи Visual Studio Code [11]:

- можливість розділити інтерфейс на декілька панелей коду;

- можливість відкрити декілька проектів в різних вікнах та паралельна робота в них;
- редактор підтримує велику кількість мов програмування, таких як C++, C#, Java, Python, Go, PHP, JavaScript, TypeScript, HTML, CSS та інших;
- має підсвічування синтаксису, що полегшує читання тексту програм, та покращує візуальне сприйняття коду, завдяки виділенню синтаксичних конструкцій тексту, наприклад з використанням різних кольорів;
- система IntelliSense, яка автоматично завершує написання коду при вводі початкових букв, що забезпечують високу продуктивність;
- дозволяє розробляти консольні додатки та графічний інтерфейс для користувача, також має підтримку технології Windows Forms;
- 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;
- у редакторі є вбудовані інструменти для роботи з Git, інструменти інтеграції з GitHub, засоби рефакторингу та навігації по коду, є вбудований відладчик і термінал.

Переваги роботи Visual Studio Code:

- інтегрований термінал економить час, адже програмісту не доводиться постійно перемикатися між вікнами термінала та редактора. Також в ході автоматичного запуску проекту дуже легко помічати помилки;
- можна додати велику кількість плагінів, функцій та покращити інтерфейс завдяки розширенням. Наприклад можна додати нові іконки, теми, покращити автозаповнення, виділення коду, форматувати код з дотриманням всіх відступів, запустити локальний сервер проекту, додати підсвічування для всіх фігурних дужок, покращити роботу з Git, генерувати фейкові дані, автоматичне закриття тегів, можливість редагувати код паралельно на декількох девайсах та інше;

- простий та гнучкий інтерфейс з велико кількістю гарячих клавіш та налаштувань, який підтримує багато функцій та мов програмування;
- використання сніпетів та системи автозавершення IntelliSense підвищує продуктивність, адже вони автоматизують ручну працю прискорюючи роботу через те, що полегшують доступ до заздалегідь заготовлених фрагментів коду. В результаті програміст не витрачає часу на те, щоб писати код конструкцій, а продумує як їх правильно зробити. Також це дозволяє швидше освоїти фреймворки чи бібліотеки.

2.7.3. Виклик та завантаження програми

Для без серверної версії в локальному хостингу наступний виклик:

- завантажити код програми;
- встановити NodeJS, бажано версію не менше v.16.15.0;
- відкрити папку з кодом додатку в програмі VS Code;
- в терміналі додати всі пакети для роботи React, ввівши команду npm i;
- ввести команду npm start для запуска локального серверу, сторінка відкриється в браузері.

Для серверної версії достатньо перейти за посиланням в браузері та почати користуватися. Програма не вимоглива до виклику та завантаження і може працювати на ПК, ноутбуках, планшетах чи смартфонах різних типів та конфігурацій під управлінням різних ОС.

2.7.4. Опис інтерфейсу користувача

На початку роботи при розробці дизайну рекомендується створювати ескіз для того, щоб визначитися з кількістю колонок, елементів їх розташування. Також важливою частиною розробки є дизайн код сайту, щоб всі елементи виглядали гармонічно та в одному стилі, що для кінцевого користувача є зручним та приємним, адже якщо сторінки сайту добре

прооблені в деталях, розроблені гарні анімації у елементів, кнопок та елементів меню, що збільшує привабливість сайту та затримує клієнтів на довше. Крім цього обов'язково потрібно враховувати девайси, з яких відвідується сайт, адже вони мають різну діагональ, роздільну здатність екрану та кольоропередачу.

Веб додаток має наступні основні елементи: шапка (header), основний контейнер сайту та підвал (footer).

Розробка починається з головної сторінки сайту, на ній можна побачити структуру ресурсу, його наповненість, а також зорієнтувати користувача. Відразу видно каталог товарів, які користувач може обрати та замовити на сайті, в назві описані їх характеристики та ціна, користувач може додати за допомогою кнопки будь який товар до корзини, що відразу відображається на сторінці. За допомогою зручних фільтрів можна відсортувати товари за ключовими сховами та відобразити їх в спеціальному порядку: за замовчанням, за зростанням чи спаданням ціни.

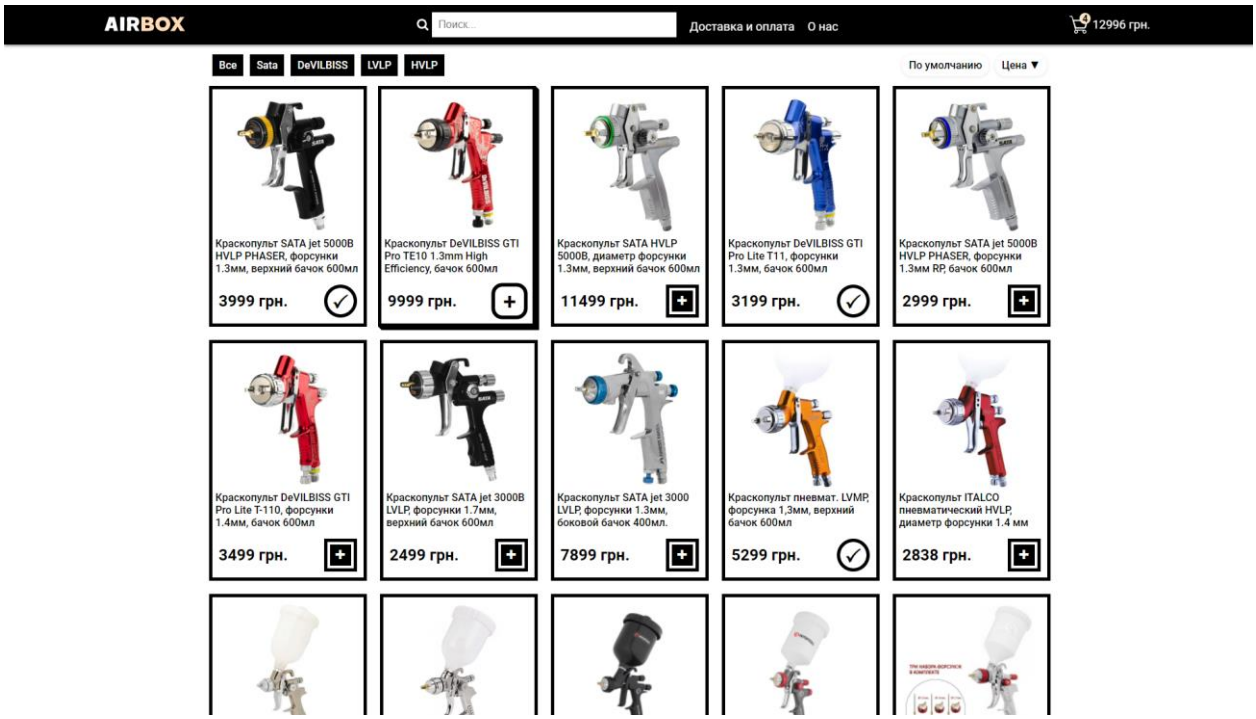

Рис. 2.7. Головна сторінка сайту

Важливою частиною сайту є верхнє меню (header), воно завжди прикріплене до верхньої частини сторінки, в ньому розміщені:

- логотип компанії, натиснувши на який можна перейти до головної сторінки;
- навігація, в якій розташоване меню пошуку, завдяки якому можна знайти потрібний товар за назвою, результат чого відразу відображається на сторінці;
- в навігації також розташовані посилання на дві сторінки сайту з інформацією про компанію, доставку та оплату;
- кошик, де відображається кількість доданих товарів та їх загальна ціна, натиснувши відкриється додаткове меню корзини.

Також важливим є підвал сайту (footer), вона завжди знаходиться в нижній частини сайту, в ній розміщені:

- логотип компанії, натиснувши на який можна перейти до головної сторінки;
- навігація, в якій розташоване посилання на сторінку з останніми товарами, які користувачі замовили на сайті;
- Посилання на соціальні мережі та контакти з адміністраторами.

Коли користувач обрав товари для замовлення, відкривши меню кошику буде видно список всіх елементів доданих в кошик. За допомогою спеціальної кнопки їх можна звідти видалити. Для зручності користувача одразу відображається загальна ціна всіх товарів та розраховується приблизна ціна доставки. Натиснувши на кнопку “Оформить заказ” користувач перейде на сторінку для оформлення замовлення.

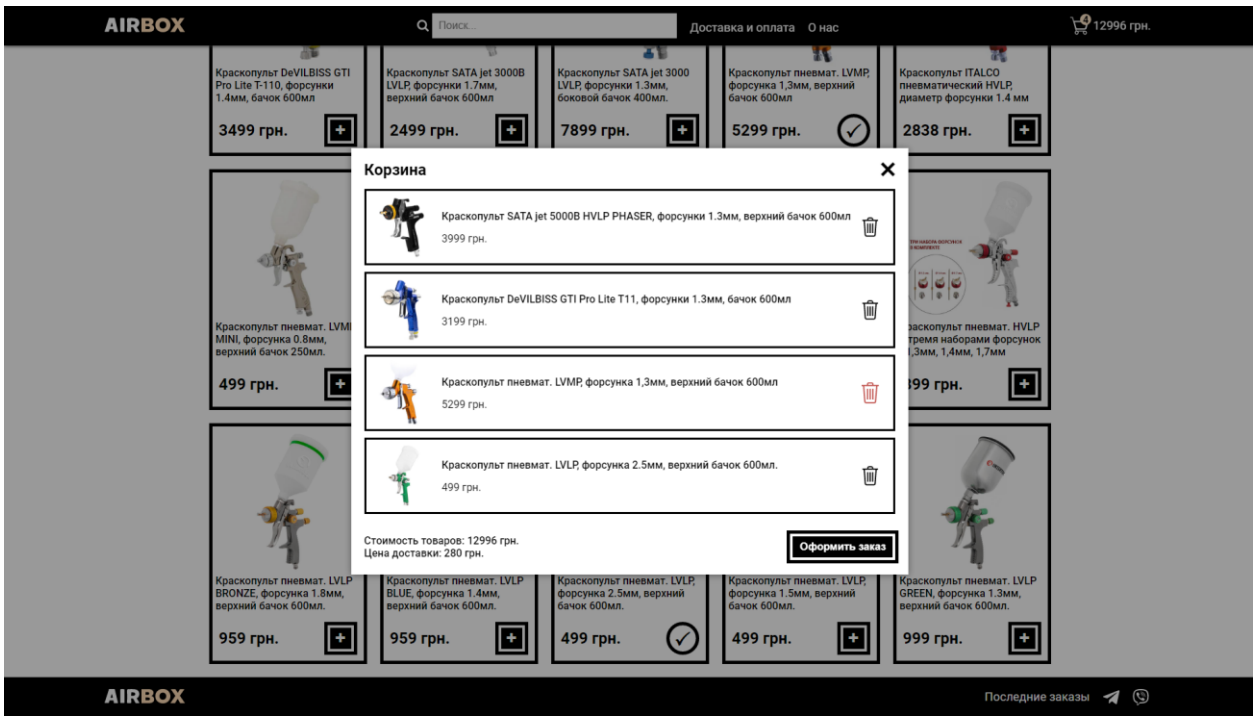

Рис. 2.8. Вікно кошика з доданими товарами

В додатковому вікні, якщо до кошика ще не додано жодного товару, то користувача буде про це повідомлено та відображено наступний контент, зображений на рис. 2.9.

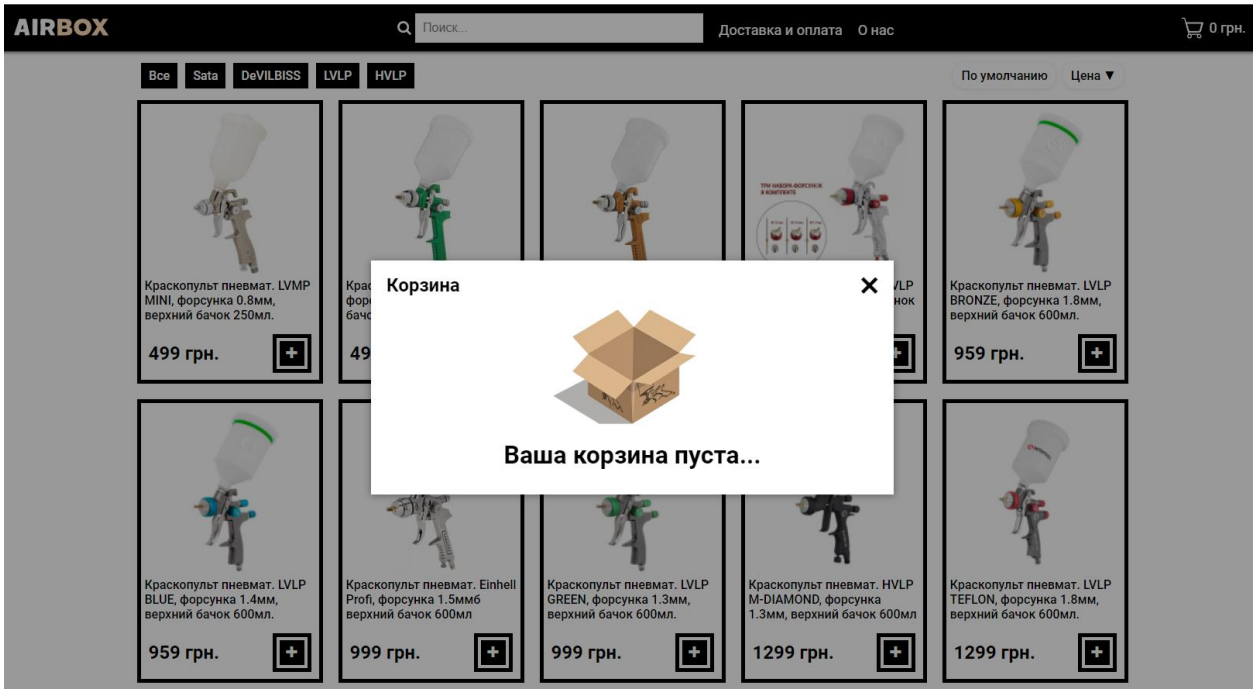

Рис. 2.9. Вікно пустого кошика

На сторінці оформлення замовлення знову відображаються всі товари, додані о корзини, їх спільна ціна, приблизна вартість доставки та їх підсумкова ціна. Користувачу потрібно ввести всі свої данні, куди потрібно відправили посилку, та коментар за бажанням. Натиснувши на кнопку “Подтвердить заказ” всі дані будуть відправлені до бази даних для менеджеру, всі поля вводу та кошик будуть очищені. Користувач буде повідомлений про успішне замовлення.

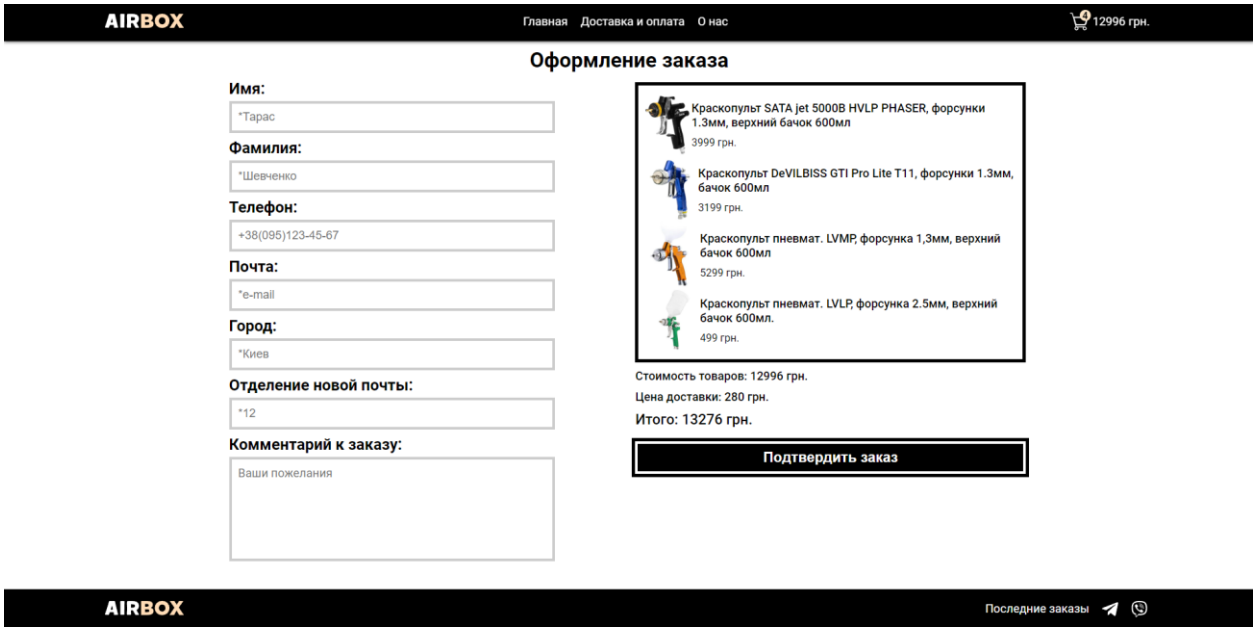

Рис. 2.10. Сторінка оформлення замовлення

З нижнього меню можна перейти до сторінки всіх товарів, оформлених користувачами. Якщо користувач вже оформив замовлення, то зверху на цій сторінці можна побачити своє замовлення.

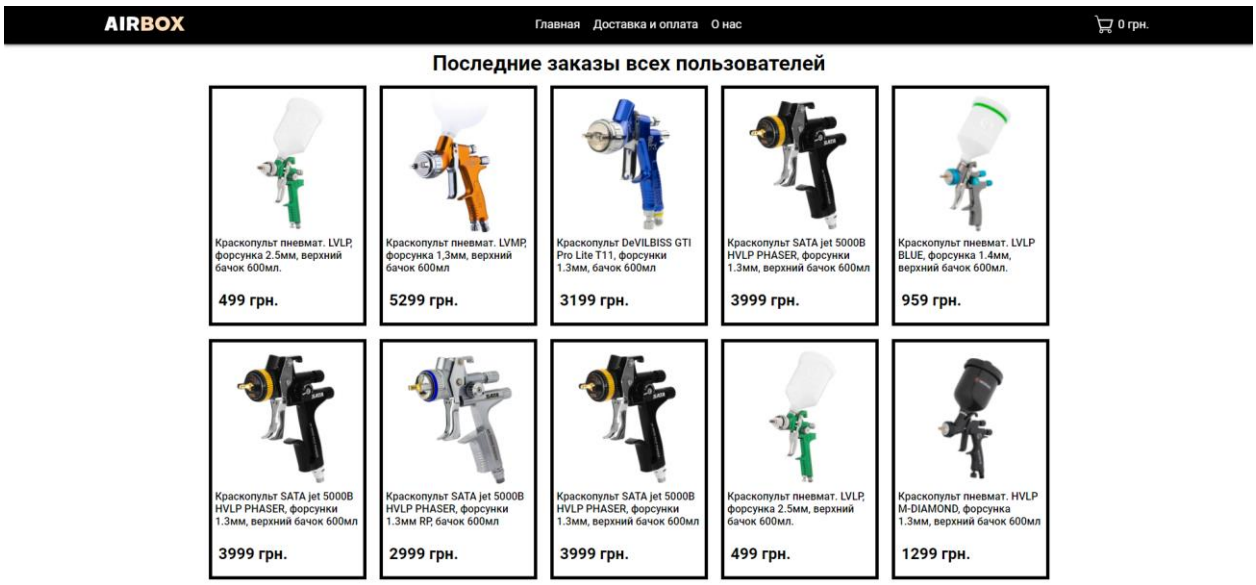

Рис. 2.11. Сторінка всіх товарів оформлених користувачами

При переході з верхнього навігаційного меню на сторінку “Доставка и оплата” буде виведена уся потрібна інформація про спосіб доставки та як буде проводитися оплата замовлених товарів та гарантії магазину.

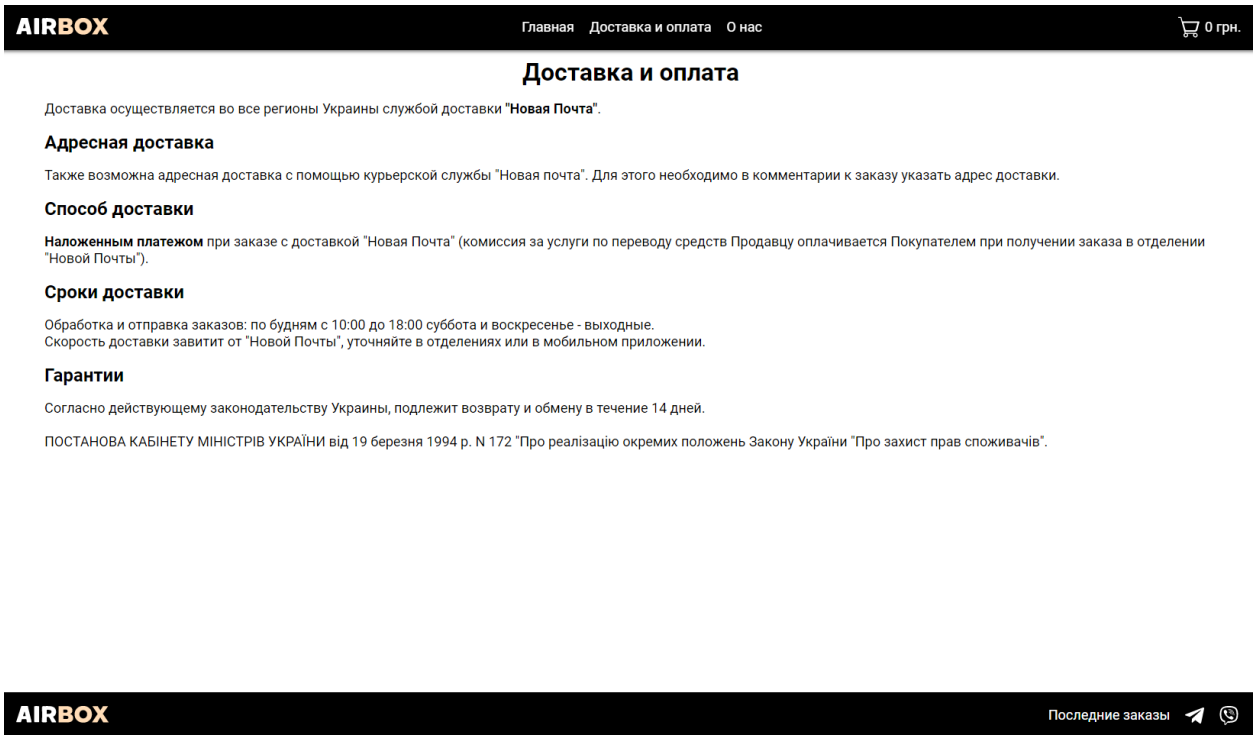

Рис. 2.12. Сторінка з інформацією про доставку і оплату

При переході з верхнього навігаційного меню на сторінку “О нас” буде виведена коротка інформація про компанію, контактний телефон, через який можна зв’язатися в продавцями, посилання на переписку в месенджері. Також на сторінці є форма зворотного зв'язку “Напишите нам”, зроблена для контакту з адміністратором, для з'ясування будь-яких питань та ситуацій, або для побажань або критики.

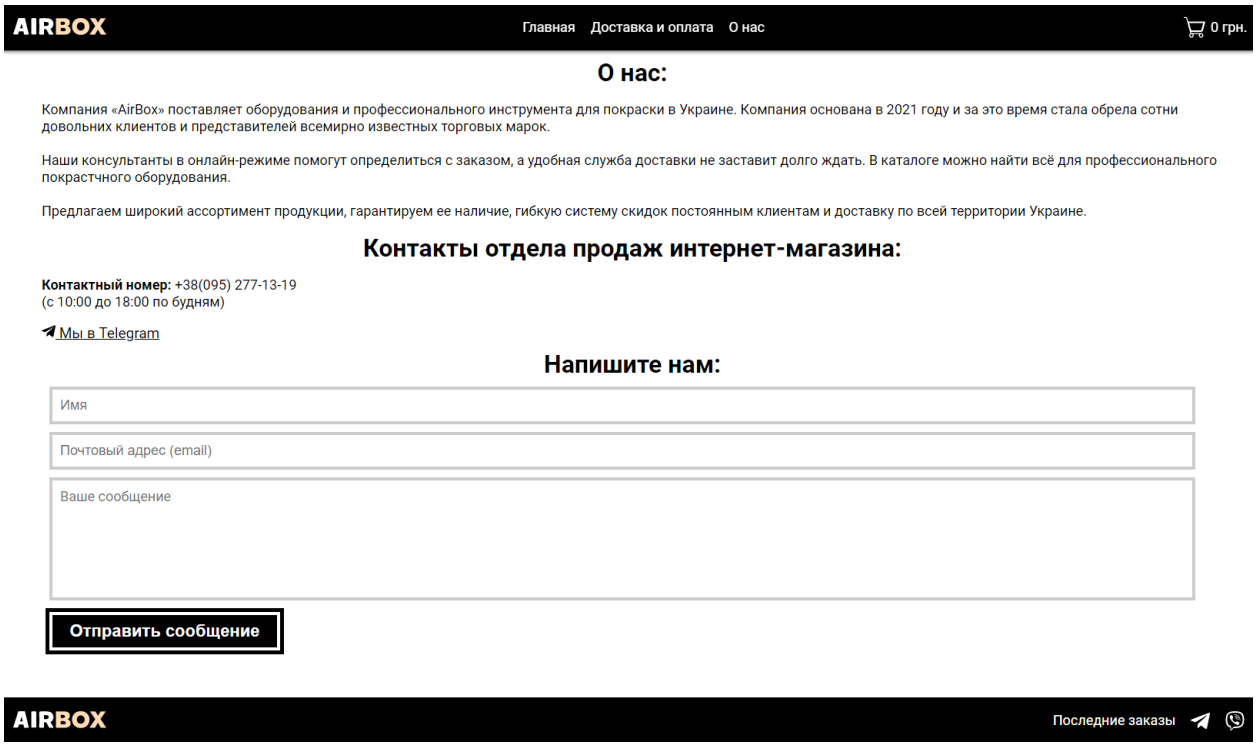

Рис. 2.13. Про компанію та форма зворотного зв’язку

Інформація з серверу може завантажитися не відразу, це може зайняти від долі секунди, що користувач цього навіть не помітить, але можуть були вийнятий, коли товари завантажуються з сервера декілька секунд, або зовсім відбудеться помилка запиту. Для цього за допомогою бібліотеки React Skeleton були розроблені анімовані градієнтом картки завантаження сторінки, щоб користувач розумів, що йде завантаження інформації з бази даних, а перед ним відображається не просто пуста сторінка.

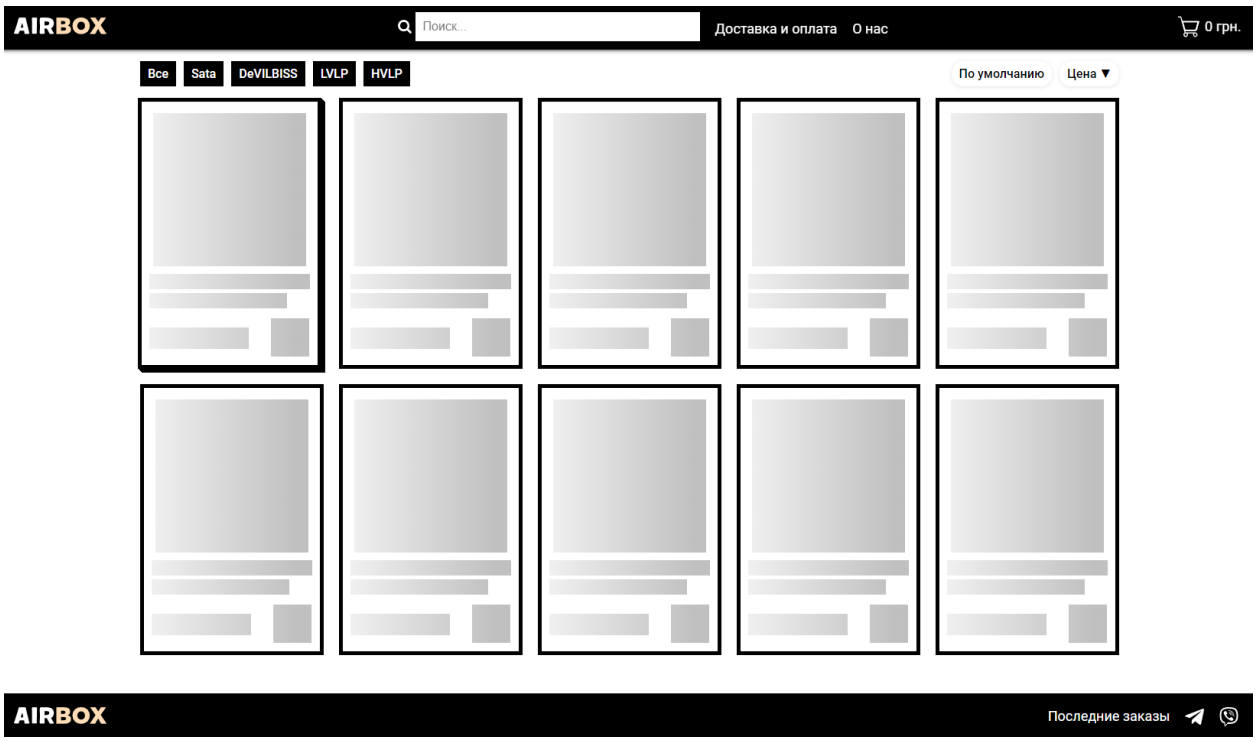

Рис. 2.14. Вигляд сторінки при завантаженні товарів

При вводі будь якого тексту в поле пошуку серед всіх товарів буде пошук збігів з введеним значенням. Результат миттєво буде відображено на сторінці, що дуже зручно для користувача та не вимагає від нього додаткових дій, переходу на інші сторінки та очікування результату.

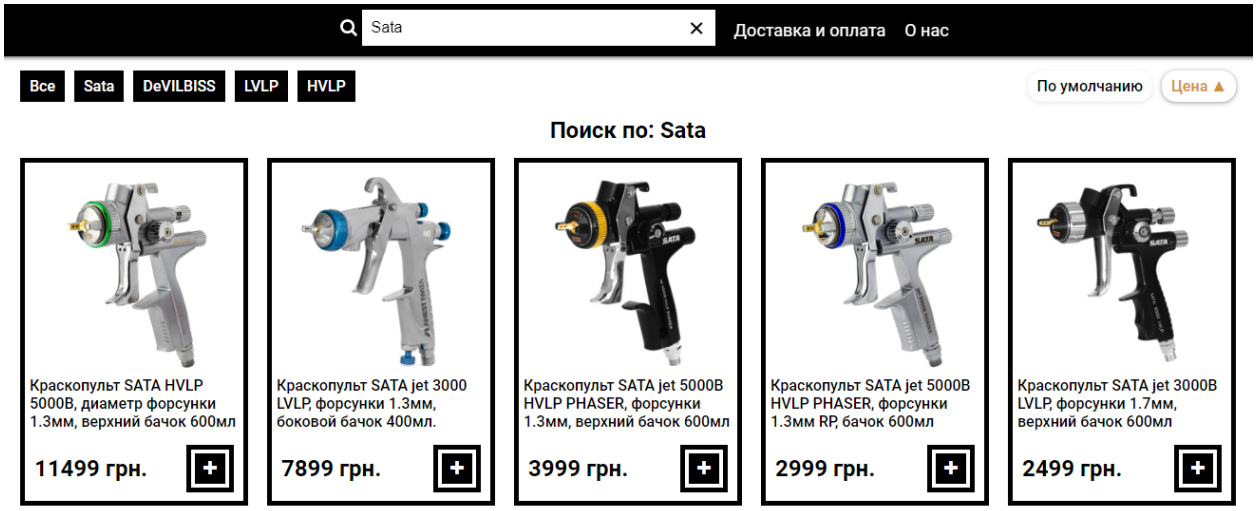

Рис. 2.15. Вигляд товарів при пошуку

Як вже зазначалося, важливу роль для сайту та користувача грає адаптивне відображення на різних девайсах, та платформах. Як приклад реалізації зручного адаптивного інтерфейсу показано головну сторінку на мобільному пристрохї з відсортованими товарами за зростанням ціни.

Рис. 2.16. Вигляд адаптивної сторінки на мобільному пристрої

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. Передбачуване число операторів програми – 1380;
2. Коефіцієнт складності програми – 1,3;
3. Коефіцієнт корекції програми в ході її розробки – 0,06;
4. Годинна заробітна плата програміста– 150 грн/год;

Середня годинна зарплатня Junior JavaScript Developer в Україні була вирахувати виходячи з даних «Української спільноти програмістів (DOU)»[1]. Станом на грудень 2021 року середня зарплатня Junior Front-End розробника 900$ у місяць. При курсі валют НБУ на початок червня 2022 року один американський долар дорівнює 29,25 грн, тому середня зарплата в гривнях дорівнює 26325 грн. При стандартному графіку (176 годин/місяць) зарплатня за годину буде становити близько 150 грн.

5. Коефіцієнт збільшення витрат праці в наслідок недостатнього опису задачі – 1,2;
6. Коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1;
7. Вартість машино-години ЕОМ – 10 грн/год.

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

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин}, \quad (3.1)$$

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

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

t_a – витрати праці на розробку блок-схеми алгоритму;

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

$t_{отл}$ – 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):

$$Q = q \cdot C \cdot (1 + p),$$

де q – передбачуване число операторів (1380);

C – коефіцієнт складності програми (1,3);

p – коефіцієнт корекції програми в ході її розробки (0,06).

Звідси умовне число операторів в програмі:

$$Q = 1380 \cdot 1{,}3 \cdot (1 + 0{,}06) = 1901{,}64$$

Витрати праці на вивчення опису задачі $t_и$ визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_и = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин,}$$

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

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

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1,2). З урахуванням коефіцієнта кваліфікації k = 1, отримуємо витрати праці на вивчення опису завдання:

$$t_u = (1901{,}64 \cdot 1{,}2) / (78 \cdot 1) = 29{,}25 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20...25)*k}, \text{ людино-годин,} \quad (3.2)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.2), отримаємо:

$$t_a = 1901{,}64 / (21 \cdot 1) = 90{,}55 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)*k}, \text{ людино-годин.}$$

$$t_n = 1901{,}64 / (25 \cdot 1) = 76{,}06 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4...5)*k}, \text{ людино-годин.}$$

$$t_{отл} = 1901{,}64 / (5 \cdot 1) = 380{,}3 \text{ людино-годин.}$$

– за умови комплексного налагодження завдання:

$$t_{отл}^{k} = 1{,}5 \cdot t_{отл} \text{ , людино-годин.}$$

$$t^{k}_{отл} = 1{,}5 \cdot 380{,}3 = 570{,}45 \text{ людино-годин.}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_{д} = t_{др} + t_{до} \text{ , людино-годин,}$$

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

$$t_{\text{др}} = \frac{Q}{(15 \ldots 20) * k}, \text{ людино-годин,}$$

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

$$t_{до} = 0{,}75 \cdot t_{др} \text{ , людино-годин.}$$

Підставляючи відповідні значення, отримаємо:

$$t_{др} = 1901{,}64 / (19 \cdot 1) = 100{,}08 \text{ людино-годин.}$$

$$t_{до} = 0{,}75 \cdot 100{,}08 = 75{,}06 \text{ людино-годин.}$$

$$t_{д} = 118{,}2 + 88{,}65 = 175{,}14 \text{ людино-годин.}$$

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

$t = 50 + 29,25 + 90,55 + 76,06 + 380,3 + 175,14 = 801,3$ 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:

$$К_{по} = З_{зп} + З_{мв}, \text{ грн.}$$

Заробітна плата виконавців визначається за формулою:

$$З_{зп} = t * С_{пр}, \text{ грн,}$$

де: t – загальна трудомісткість, людино-годин;

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

З урахуванням того, що середня годинна зарплата програміста становить 150 грн/год, отримуємо:

$$З_{зп} = 801,3 \cdot 150 = 120\ 195 \text{ грн.}$$

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

$$Змв = tотл * Смч \text{ , грн,} \quad (3.3)$$

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

$С_{мч}$ – вартість машино-години ЕОМ, грн/год (14 грн/год).

Підставивши в формулу (3.3) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 380{,}3 \cdot 10 = 3803 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 120\ 195 + 3803 = 123\ 998 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$t_u = \frac{t}{Bk * Fp}, \text{ міс.}$$

де B_k – число виконавців (дорівнює 1);

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

Звідси витрати на створення програмного продукту:

$$T = 801{,}3 / 1 \cdot 176 \approx 4{,}55 \text{ міс.}$$

Висновок: програмне забезпечення розроблено для роботи інтернет-магазину з продажу фарбувального обладнання. Вартість даного програмного забезпечення становить 123 998 грн. Очікуваний час розробки становить 801,3 годин, тобто 4,55 місяці при 40 годинному робочому тижні. Цей термін пов'язаний зі значним числом операторів, і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування по готовому алгоритму, налагодження програми і підготовку документації.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи був розроблений інтернет-магазин з продажу фарбувального обладнання з використанням бібліотеки React.

Практичне призначення даної системи полягає в забезпеченні відвідувачам сайту простого і комфортного доступу до каталогу товарів за рахунок оптимальних параметрів візуалізації його вмісту, що зробить процес покупки швидшим та зручнішим, і підвищить ефективність роботи інтернет-магазину.

Під час розробки даного проекту були виконані наступні задачі:

- досліджено та проаналізовано предметну галузь розв'язуваної задачі;
- створена база з товарами та даними замовлень зроблених на сайті;
- створено алгоритм для реалізації поставленого завдання;
- спроектовано і розроблено веб-сторінку додатку.

Розроблене програмне забезпечення дозволяє:

- формувати веб-сторінки на основі шаблонів з використанням контенту, отриманого з бази даних, з можливістю сортування, фільтрів і пошуку;
- контактування за адміністратором даного магазину;
- оформлення замовлень з даного магазину.

Програма реалізована на базі бібліотеки React з використанням мови програмування JavaScript, розмітки HTML та стилів SCSS.

В економічному розділі проведено розрахунок вартості роботи по створенню програми, яка становить близько 124 тис. грн. та розраховано час на його створення - 801,3 людино-годин, тобто 4,55 місяці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Зарплати українських розробників — зима 2022. URL: https://dou.ua/lenta/articles/salary-report-devs-winter-2022/
2. Знакомьтесь, архитектура REST. URL: https://html-templates.info/blog/znakomtes-arhitektura-REST
3. Переваги інтернет-магазину над звичайною торговою точкою. URL: https://webdevandseo.com.ua/online-store-benefits/
4. Рейтинг мов програмування 2022. URL: https://dou.ua/lenta/articles/language-rating-2022/
5. Статичні та динамічні web-сайти. URL: https://armedsoft.com/ua/blog/statychni-ta-dynamichni-web-sayty
6. Що таке Restful API. Основні компоненти Restful API. URL: https://www.semtek.com.vn/restful/
7. Что такое CSS-модули и зачем они нам? URL: https://frontender.info/css-modules-part-1-need/
8. NPM trends: Angular vs React vs Vue. URL: https://www.npmtrends.com/angular-vs-react-vs-vue
9. React - JavaScript-бібліотека для створення користувацьких інтерфейсів. URL: https://uk.reactjs.org/
10. Sass Basics. URL: https://sass-lang.com/guide
11. Visual Studio Code - Википедия. URL: https://ru.wikipedia.org/wiki/Visual_Studio_Code
12. Х. Брайан. HTML5 и CSS3. Веб-разработка по стандартам нового поколения / Хоган Брайан — СПб.: Питер, 2014. – 320с.
13. Э. Браун. Изучаем JavaScript. Руководство по созданию современных веб-сайтов / Этан Браун – Москва: Диалектика-Вильямс, 2020. – С. 35-47.
14. А. Бэнкс, Е. Порселло. React: современные шаблоны для разработки приложений. 2е изд. / Алекс Бэнкс, Ева Порселло – СПб.: Питер, 2021. – 320 с.

15. Васильев А.Н. JavaScript в примерах и задачах / Васильев А.Н. – Москва: Издательство “Э”, 2017. – 720 с.

16. Д. Дакетт. HTML и CSS. Разработка и дизайн веб-сайтов / Джон Дакетт – Москва: Эксмо, 2022. – 480 с.

17. Дронов В.А. HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов / В.А.Дронов – СПб.: БХВ-Петербург, 2011. – 416 с.

18. Д. Крокфорд. Как устроен JavaScript / Дуглас Крокфорд – СПб.: Питер, 2019. – 304 с.

19. Леонтьев А.А. Web-дизайн. Руководство пользователя / Леонтьев А.А. – Москва: Центр, 2000. – 126 с.

20. Т. Марк. React в действии / Тиленс Томас Марк — СПб.: Питер, 2018. – 368с.

21. Орлов Л.А. Як створити електронний магазин в Інтернет / Л. А. Орлов. – Москва: БУК-ПРЕС, 2016. – 384 с.

22. С. Пьюривал. Основы разработки веб-приложений / Сэмми Пьюривал – СПб.: Питер, 2015. – 272 с.

23. Д. Флэнаган. JavaScript. Полное руководство, 7-е изд. / Дэвид Флэнаган – СПб.: Диалектика, 2021. – 720 с.

24. Б. Хеник. HTML и CSS. Путь к совершенству. / Бен Хеник – СПб.: Питер, 2011. – 336 с.

ДОДАТОК А

КОД ПРОГРАМИ

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <meta name="theme-color" content="#000000" />
  <meta name="description"
    content="Thesis on the creation of a website with the sale of painting equipment, created using create-react-app"
/>
  <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500;700;900&display=swap"
rel="stylesheet">
  <title>AirBox - покрасочное оборудование</title>
</head>
      <body>
        <div id="root"></div>
      </body>
</html>
```

index.js

```
import React from 'react';
import { createRoot } from "react-dom/client"
import { BrowserRouter as Router } from 'react-router-dom';
import './index.scss';
import App from './App';

const root = createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <Router>
      <App />
    </Router>
  </React.StrictMode>,
);
```

App.jsx

```
import React from 'react';
import axios from 'axios';
import { Routes, Route } from 'react-router-dom';
import StoreContext from './hooks/useContext';
import Header from './components/Header';
import Footer from './components/Footer';
import CartWindow from './components/CartWindow';
import Main from './pages/Main';
import Delivery from './pages/Delivery';
import About from './pages/About';
import Order from './pages/Order';
import RecentOrders from './pages/RecentOrders';

let tempLS = [];
```

```
function App() {
  const [addCart, setAddCart] = React.useState([]); //список всіх товарів в корзині, також для додавання товару
в корзину
  const [airbrush, setAirbrush] = React.useState([]); //товари
  const [searchInput, setSearchInput] = React.useState(''); //текст в строці пошуку
  const [openCart, setOpenCart] = React.useState(false); //для открытия и закрытия корзины
  const [loadingReady, setLoadingReady] = React.useState(true); //підтвердження загрузки з БД
  tempLS = addCart;

  React.useEffect(() => {
    async function fetchData() {
      try {
        setLoadingReady(true);
        const itemsResponse = await axios.get(`https://62792bd2d00bded55ae56077.mockapi.io/airbrush`);
        setLoadingReady(false);
        setAirbrush(itemsResponse.data);
      } catch (error) {
        alert('Ошибка при запросе данных с сервера :(\nОбновите страницу или попробуйте позже.');
        console.error(error);
      }
    }
    fetchData();
    if (!localStorage.getItem('Cart')) { //якщо в localStorage не існує Корзини то створити її
      localStorage.setItem('Cart', JSON.stringify([]));
    }
    setAddCart(JSON.parse(localStorage.getItem('Cart'))); //задання корзини з localStorage для вівдображення на
сторінці
  }, []);

  const onAddToCart = (item) => { //додавання певного товару до корзини
    let keyTemp; //записати індекс обраного товару
    for (const key in addCart) {
      if (addCart[key].id === item.id) {
        keyTemp = key;
        break;
      }
    }
    if (addCart.filter(el => el.id === item.id).length > 0) {
      tempLS.splice(keyTemp, 1); //remove
    } else {
      tempLS.push(item); //add
    }
    localStorage.setItem('Cart', JSON.stringify(tempLS));
    setAddCart(JSON.parse(localStorage.getItem('Cart')));
  };
  const onRemoveCart = (Id) => { //додавання певного товару до корзини
    for (const key in tempLS) {
      if (Id === tempLS[key].id) {
        tempLS.splice(key, 1);
      }
    }
    localStorage.setItem('Cart', JSON.stringify(tempLS));
    setAddCart(JSON.parse(localStorage.getItem('Cart')));
  };
  const onChangeSearch = (event) => { //для вікна пошуку
    setSearchInput(event.target.value);
  }
  const findItemMenu = (text) => { //для пошуку по краскопультам
    setSearchInput(text);
  }
  const isItemAdded = (id) => {
    return !addCart.some(obj => obj.id === id)
```

```
  };
  return (
    <>
      <StoreContext.Provider value={{ addCart, loadingReady, isItemAdded }}>
        {openCart && < CartWindow onClickCart={() => setOpenCart(false)} onRemove={onRemoveCart} />
/*відкриття корзини, додавання в корзину*/}
        <Header onClickCart={() => setOpenCart(true)} onChangeSearch={onChangeSearch}
searchInput={searchInput} clearSearch={() => setSearchInput('')} />
        <div className='container'>
          <Routes>
            <Route exact path="/" element={<Main
              searchInput={searchInput}
              airbrush={airbrush}
              onAddToCart={onAddToCart}
              findItemMenu={findItemMenu}
              setAirbrush={setAirbrush}
            />} />
            <Route exact path='delivery' element={<Delivery />} />
            <Route exact path='about' element={<About />} />
            <Route exact path='order' element={<Order setAddCart={setAddCart} />} />
            <Route exact path='recent-orders' element={<RecentOrders />} />
            <Route path='*' element={<h1 style={{ margin: '50px 0 600px 0' }}>404 not found<br /><br />Страница
не найдена</h1>} />
          </Routes>
        </div>
        <Footer />
      </StoreContext.Provider>
    </>
  );
}
export default App;
```

CartWindow.jsx

```
import { Link } from 'react-router-dom';
import styles from './CartWindow.module.scss'
import { useCart } from '../hooks/useCart';

function CartWindow({ onRemove, onClickCart }) {
  const { addCart, cartCost } = useCart();
  return (
    <div className={styles.cartBG}>
      <div className={styles.cartWindow}>
        <div className={styles.cartTopBottom}>
          <h2>Корзина</h2>
          <p onClick={onClickCart}>✕</p> {/* закрити корзину */}
        </div>
        {addCart.length > 0 ?
          <>
            <div className={styles.items}>
              {addCart.map((obj) => ( //вивід даних товарів доданих в корзину
                <div key={obj.id} className={styles.cartItem}>
                  <div className="d-flex">
                    <div className={styles.imgWrap}>
                      <img height={100} src={obj.imgUrl} alt={obj.id} />
                    </div>
                    <div>
                      <p>{obj.title}</p>
                      <h6>{obj.cost} грн.</h6>
                    </div>
                  </div>
                  <img onClick={() => onRemove(obj.id)} className={styles.delete} src="img\basket.svg"
alt="delete" />
```

```
                    </div>
                  ))}
                </div>
                <div className={styles.cartTopBottom}>
                  <ul>
                    <li>Стоимость товаров: {cartCost} грн.</li>
                    <li>Цена доставки: {Math.round(cartCost / 100 * 2 + 20)} грн.</li>
                  </ul>
                  <Link to="/order" onClick={onClickCart}><button>Оформить заказ</button></Link>
                </div>
              </>
              : <>
                <img className='emptyCart' src="img\emptyCart.png" alt="emptyCart" />
                <h1 > Ваша корзина пуста...</h1>
              </>}
          </div>
        </div >
    );
}
export default CartWindow;
```

Cell.jsx

```
import React from 'react';
import ContentLoader from "react-content-loader"
import styles from './Cell.module.scss';
import StoreContext from '../hooks/useContext';

function Cell({ id, title, cost, imgUrl, handleClickAdd }) {
    const { loadingReady, isItemAdded } = React.useContext(StoreContext);
    // isItemAdded - зміна стилю та тексту кнопки при додаванні до корзини, змінення на сторінці та в корзині
    const addCartClick = () => {
        handleClickAdd({ id, title, cost, imgUrl });
    }
    return (
        <div className={styles.Cell}>
            {loadingReady ? //якщо товари з БД не завантажилась то відображати пусті картки
                <ContentLoader
                    speed={3}
                    width={240}
                    height={330}
                    viewBox="0 0 240 330"
                    backgroundColor="#C0C0C0"
                    foregroundColor="#F2F2F2"
                >
                    <rect x="10" y="10" rx="0" ry="0" width="200" height="200" />
                    <rect x="5" y="220" rx="0" ry="0" width="210" height="20" />
                    <rect x="5" y="245" rx="0" ry="0" width="180" height="20" />
                    <rect x="5" y="290" rx="0" ry="0" width="130" height="28" />
                    <rect x="164" y="278" rx="0" ry="0" width="50" height="50" />
                </ContentLoader>
                : <>
                    <div className={styles.imgWrap}>
                        <img height='220px' src={imgUrl} alt={id} />
                    </div>
                    <p>{title}</p>
                    <div className={styles.price}>
                        <p>{cost} грн.</p>
                        {handleClickAdd && <button className={isItemAdded(id) ? styles.addCart :
styles.addCartChecked}
                            onClick={addCartClick}>{isItemAdded(id) ? '+' : '✓'}</button>}
                    </div>
```

```
            </>}
        </div>
    );
}
export default Cell;
```

Footer.jsx

```
import React from 'react';
import { Link } from 'react-router-dom';

function Footer() {
    return (
        <>
            <footer>
                <div>
                    <Link to="/"><img height={24} src="img/logo.svg" alt="Коробка" /></Link>
                </div>
                <nav>
                    <ul>
                        <li><Link to="recent-orders">Последние заказы</Link></li>
                        <li><a href="https://t.me/Jabka" target="_blank" rel="noreferrer"><img src="img/Telegram.svg"
alt="telegram" /></a></li>
                        <li><a href="https://t.me/Jabka" target="_blank" rel="noreferrer"><img src="img/Viber.svg"
alt="viber" /></a></li>
                    </ul>
                </nav>
            </footer>
        </>
    );
}
export default Footer;
```

Header.jsx

```
import React from 'react';
import { Routes, Route, Link } from 'react-router-dom';
import styles from './Header.module.scss';
import { useCart } from '../hooks/useCart';

function Header({ onClickCart, onChangeSearch, searchInput, clearSearch }) {
    const { addCart, cartCost } = useCart();
    return (
        <>
            <header>
                <>
                    <Link to="/"><img height={24} src="img\logo.svg" alt="Коробка" /></Link>
                </>
                <nav>
                    <ul>
                        <Routes>
                            <Route exact path='delivery' element={<li><Link to="/">Главная</Link></li>} />
                            <Route exact path='order' element={<li><Link to="/">Главная</Link></li>} />
                            <Route path='*' element={<li><Link to="/">Главная</Link></li>} />
                            <Route exact path="/" element={ // увімкнути пошук тільки на головній сторінці
                                <li><div className='d-flex'>
                                    <div className={styles.search}>
                                        <img height={18} src="img\search.svg" alt="search" />
                                        <input onChange={onChangeSearch} placeholder='Поиск...' value={searchInput} />
                                        {searchInput && <p className={styles.searchClear} onClick={clearSearch}>✕</p>}
                                    </div>
                                </div></li>} />
                        </Routes>
```

```
                <li><Link to="delivery">Доставка и оплата</Link></li>
                <li><Link to="about">О нас</Link></li>
              </ul>
            </nav>
            <div className={styles.cart} onClick={onClickCart}>
              {Boolean(addCart.length) && <div><p>{addCart.length}</p></div>}
              <img src="img\cart.svg" alt="card" />
              <p>{cartCost} грн.</p>
            </div>
          </header>
        </>
    );
}
export default Header;
```

useContext.jsx

```
import React from 'react';

const StoreContext = React.createContext({});

export default StoreContext;
```

useCart.jsx

```
import React from 'react';
import StoreContext from './useContext';

export const useCart = () => {
    const { addCart } = React.useContext(StoreContext);
    const cartCost = addCart.reduce((sum, item) => item.cost + sum, 0);
    return { addCart, cartCost };
};
```

About.jsx

```
import React, { useRef } from "react";
import emailjs from '@emailjs/browser';

function About() {
    const form = useRef();
    const sendEmail = (e) => {
        e.preventDefault();
        emailjs.sendForm('service_hangal9', 'template_4lxgeci', form.current, 'To8UwZDUHt8iKaYbP')
            .then((result) => {
                console.log(result.text);
                alert('Ваше сообщение отправлено!')
            }, (error) => {
                alert('Ошибка при отправке сообщения:(')
                console.log(error.text);
            });
    };

    return (
        <div className="about">
            <h1>О нас:</h1>
            <p>Компания «AirBox» поставляет оборудования и профессионального инструмента для покраски в Украине. Компания основана в 2021 году и за это время стала обрела сотни довольних клиентов и представителей всемирно известных торговых марок.
                <br /><br />
                Наши консультанты в онлайн-режиме помогут определиться с заказом, а удобная служба доставки не заставит долго ждать. В каталоге можно найти всё для профессионального покрастчного оборудования.
                <br /><br />
```

```
            Предлагаем широкий ассортимент продукции, гарантируем ее наличие, гибкую систему скидок постоянным клиентам и доставку по всей территории Украине.</p>
          <p></p>
          <h1>Контакты отдела продаж интернет-магазина:</h1>
          <p><b>Контактный номер:</b> +38(095) 277-13-19   <br />(с 10:00 до 18:00 по будням)<br /></p>
          <a href="https://t.me/Jabka" target="_blank" rel="noreferrer"><img src="img/Telegram.svg" alt="telegram" /> Мы в Telegram</a> <br />

          <h1>Напишите нам:</h1>
          <form ref={form} onSubmit={sendEmail} className="contactUs" >
            <input name="name" placeholder="Имя" />
            <input name="email" type='email' placeholder="Почтовый адрес (email)" />
            <textarea name="message" placeholder="Ваше сообщение" />
            <button>Отправить сообщение</button>
          </form>
        </div>
    );
}
export default About;
```

Delivery.js

```
function Delivery() {
    return (
        <div className="delivery">
            <h1>Доставка и оплата</h1>
            <p>Доставка осуществляется во все регионы Украины службой доставки <b>"Новая Почта"</b>.</p>
            <h2>Адресная доставка</h2>
            <p>Также возможна адресная доставка с помощью курьерской службы "Новая почта". Для этого необходимо в комментарии к заказу указать адрес доставки.</p>
            <h2>Способ доставки</h2>
            <p><b>Наложенным платежом</b> при заказе с доставкой "Новая Почта" (комиссия за услуги по переводу средств Продавцу оплачивается Покупателем при получении заказа в отделении "Новой Почты").</p>
            <h2>Сроки доставки</h2>
            <p>Обработка и отправка заказов: по будням с 10:00 до 18:00 суббота и воскресенье - выходные.<br></br>
                Скорость доставки завитит от "Новой Почты", уточняйте в отделениях или в мобильном приложении. </p>
            <h2>Гарантии</h2>
            <p>Согласно действующему законодательству Украины, подлежит возврату и обмену в течение 14 дней.<br /><br />
                ПОСТАНОВА КАБІНЕТУ МІНІСТРІВ УКРАЇНИ від 19 березня 1994 р. N 172 "Про реалізацію окремих положень Закону України
                "Про захист прав споживачів".</p>
        </div>
    );
}
export default Delivery;
```

Main.jsx

```
import React from 'react';
import Cell from '../components/Cell';
import StoreContext from '../hooks/useContext';

function Main({ searchInput, airbrush, onAddToCart, findItemMenu, setAirbrush }) {
    const { loadingReady } = React.useContext(StoreContext);
    const [filterOrder, setFilterOrder] = React.useState(true);
    const itemsSort = (item, defSort) => {
        (defSort ? fetch(`https://62792bd2d00bded55ae56077.mockapi.io/airbrush`)
            : fetch(`https://62792bd2d00bded55ae56077.mockapi.io/airbrush?sortBy=${item}${filterOrder ? "&order=desc" : "&order=asc"}`))
```

```
        .then((res) => {
          return res.json()
        })
        .then((json) => {
          setAirbrush(json);
        });
  };
  const itemsRender = () => {
    const airbrushFilter = airbrush.filter(item => item.title.toLowerCase().includes(searchInput.toLowerCase()))
//фільтр для пошуку
    return (loadingReady ? [{ id: 1 }, { id: 2 }, { id: 3 }, { id: 4 }, { id: 5 }, { id: 6 }, { id: 7 }, { id: 8 }, { id: 9 }, {
id: 10 }]//Array(10).fill({})
        : airbrushFilter).map((cardObj) => (
          <Cell
            key={cardObj.id}
            {...cardObj}
            handleClickAdd={(item) => onAddToCart(item)} //для додання в корзину, передає об'єкт
          />));
  };

  return (
    <>
      <div className='Filters'>
        <ul className='sortItems'>
          <li onClick={() => findItemMenu("")}>Все</li>
          <li onClick={() => findItemMenu("Sata")}>Sata</li>
          <li onClick={() => findItemMenu("DeVILBISS")}>DeVILBISS</li>
          <li onClick={() => findItemMenu("LVLP")}>LVLP</li>
          <li onClick={() => findItemMenu("HVLP")}>HVLP</li>
        </ul>
        <ul className='filterItems'>
          <li onClick={() => itemsSort("id", true)}>По умолчанию</li>
          <li onClick={() => {
            itemsSort("cost")
            setFilterOrder(!filterOrder)
          }}>Цена {filterOrder ? "▼" : "▲"}</li>
        </ul>
      </div>
      {searchInput && <h2>Поиск по: {searchInput}</h2>}
      {Boolean(airbrush.filter(item => item.title.toLowerCase().includes(searchInput.toLowerCase()))).length
        || (loadingReady || <>
          <img className='emptyCart' src="img\emptyCart.png" alt="emptyCart" />
          <h1> Ничего не найдено...</h1>
        </>)}

      <div className='disFlex'>
        {itemsRender()}
      </div>
    </>
  );
}
export default Main;
```

Order.jsx

```
import React from 'react';
import axios from 'axios';
import './Order.scss';
import { useForm } from "react-hook-form";
import { useCart } from '../hooks/useCart';

function Order({ setAddCart }) {
  const { addCart, cartCost } = useCart();
```

```
  const onSendData = async (inputData) => {
    try {
      const orderData = await axios.post('https://62792bd2d00bded55ae56077.mockapi.io/orders', { orderItems:
addCart, userData: inputData });
      console.log(orderData.data)
      alert('Ваш заказ успешно оформлен!\n' + JSON.stringify(orderData.data));
    } catch (error) {
      alert('Не удалось создать заказ :(');
      console.error(error);
    }
  };
  const {
    register,
    formState: { errors },
    handleSubmit,
    reset
  } = useForm({ mode: "onBlur" });
  const onSubmit = (data) => {
    onSendData(data);
    setAddCart([]);
    localStorage.setItem('Cart', JSON.stringify([]));
    reset();
  }
  return (
    <>
      <h1>Оформление заказа</h1>
      <form className="orderForm" onSubmit={handleSubmit(onSubmit)}>
        <div className='inputForm'>

          <h2>Имя:</h2>
          <input placeholder="*Тарас" required {...register('firstName')} />

          <h2>Фамилия:</h2>
          <input placeholder="*Шевченко" required {...register('lastName')} />

          <h2>Телефон:</h2>
          <input placeholder="+38(095)123-45-67" required
            {...register('phoneNumber', { minLength: { value: 10, message: 'Минимум 10 цифр',
valueAsNumber: true } })} />
          <div style={{ color: 'red' }}><p>{errors?.phoneNumber?.message}</p></div><p></p>

          <h2>Почта:</h2>
          <input type="email" placeholder="*e-mail" required {...register('email')} />

          <h2>Город:</h2>
          <input placeholder="*Киев" required {...register('city')} />

          <h2>Отделение новой почты:</h2>
          <input type='number' placeholder="*12" required {...register('novaposhta')} />

          <h2>Комментарий к заказу:</h2>
          <textarea placeholder="Ваши пожелания" {...register('comment')} />

        </div>
        <div className='rightSide'>
          <div className='cardWrap'>
            {addCart.length > 0 ?
              <>
                {addCart.map((obj) => ( //вивід даних товарів доданих в корзину
                  <div key={obj.id} className="itemFlex">
                    <div className='imgWrap'>
                      <img height={100} width={100} src={obj.imgUrl} alt={obj.id} />
```

```
                        </div>
                        <div>
                            <p>{obj.title}</p>
                            <h6>{obj.cost} грн.</h6>
                        </div>
                    </div >
                ))}
            </>
            : <>
                <img className='emptyCart' src="img\emptyCart.png" alt="emptyCart" />
                <h1> Ваша корзина пуста...</h1>
            </>}
        </div>
        <p>Стоимость товаров: {cartCost} грн.</p>
        <p>Цена доставки: {Math.round(cartCost / 100 * 2 + 20)} грн.</p>
        <p><span>Итого: {cartCost + Math.round(cartCost / 100 * 2 + 20)} грн.</span></p>
        <button>Подтвердить заказ</button>
      </div>
    </form>
  </>
 );
}
export default Order;
```

RecentOrders.jsx

```
import React from 'react';
import axios from 'axios';
import Cell from '../components/Cell';

function RecentOrders() {
  const [ordersList, setOrdersList] = React.useState([]);
  const [loadingReady, setLoadingReady] = React.useState(true);
  React.useEffect(() => {
    (async () => {
      try {
        setLoadingReady(true);
        const { data } = await axios.get('https://62792bd2d00bded55ae56077.mockapi.io/orders');
        setLoadingReady(false);
        setOrdersList(data.reduce((prev, obj) => [...prev, ...obj.orderItems], []).reverse());
      } catch (error) {
        alert('Ошибка при запросе списка заказов :(');
        console.error(error);
      }
    })();
  }, []);

  return (
    <>
      <h1>Последние заказы всех пользователей</h1>
      <div className="disFlex">
        {(loadingReady ? [...Array(10)]
          : ordersList).map((obj, index) => (
            <Cell
              key={index}
              {...obj}
            />))}
      </div>
    </>
  );
}
export default RecentOrders;
```

index.scss

```
body {
  min-width: 680px;
  margin: 0;
  font-family: 'Roboto', sans-serif;
}
.container {
  padding: 60px 10% 10px 10%;
  overflow: hidden;
  min-height: calc(100vh - 130px);
}
@media screen and (max-width: 1124px) {
  .container {
    padding: 60px 10px 10px 10px;
  }
}
@media screen and (max-width: 970px) {
  .container {
    padding-top: 80px;
  }
}
.d-flex {
  display: flex;
  align-items: center;
}
.disFlex {
  display: flex;
  align-items: center;
  justify-content: center;
  flex-wrap: wrap;
}
h1,
h2,
h3 {
  text-align: center;
  margin: 10px 0 5px 0;
}
footer {
  background-color: black;
  height: 60px;
  display: flex;
  text-align: center;
  align-items: center;
  justify-content: space-between;
  padding: 0 8%;
  ul {
    align-items: center;
    display: flex;
    img {
      padding-top: 5px;
      height: 25px;
    }
  }
  a {
    color: white;
    font-size: large;
    text-decoration: none;
  }
  a:after,
  a:before {
    backface-visibility: hidden;
    border: 1px solid rgba(#fff, 0);
```

```
    bottom: 0px;
    content: " ";
    display: block;
    margin: 0 auto;
    position: relative;
    transition: all 280ms ease-in-out;
    width: 0;
  }
  a:hover:after,
  a:hover:before {
    backface-visibility: hidden;
    border-color: #fff;
    transition: width 350ms ease-in-out;
    width: 100%;
  }
  a:hover:before {
    border-color: black;
  }
}
@media screen and (max-width: 970px) {
  footer {
    padding: 0 10px;
  }
}
.emptyCart {
  height: 150px;
  display: block;
  margin-left: auto;
  margin-right: auto;
  margin-bottom: 10px;
}
.delivery,
.about {
  h2 {
    text-align: left;
  }
  p {
    font-size: 18px;
  }
  a {
    color: black;
    font-size: 19px;
    &:hover {
      text-decoration: none;
    }
  }
  img {
    height: 18px;
    filter: brightness(0%);
  }
}
.contactUs {
  display: flex;
  flex-direction: column;
  justify-content: center;
  textarea,
  input {
    margin: 5px 10px;
    width: 97%;
  }
  button {
    font-size: 22px;
```

```
    margin: 10px;
    height: 50px;
    width: 300px;
    box-shadow: inset 0 0 0 4px white;
    &:hover {
      transform: scale(1.01);
      color: black;
      box-shadow: inset 0 0 0 25px white;
    }
  }
}
.Filters {
  display: flex;
  justify-content: space-between;
  padding: 15px 0 5px 0;
  margin: 0 8%;
  li {
    font-weight: 500;
    padding: 7px 10px;
    margin: 0 5px;
    cursor: pointer;
    user-select: none;
  }
  .sortItems li {
    background: black;
    color: #fff;
    transition: all 300ms ease-in-out;
    &:hover {
      color: #FFE6C7;
      transform: scale(1.05);
      box-shadow: 0 1px 5px 0 rgba(0, 0, 0, 0.4);
    }
  }
  .filterItems li {
    border-radius: 15px;
    box-shadow: 0 1px 5px 0 rgba(0, 0, 0, 0.1);
    transition: all 300ms ease-in-out;
    &:hover {
      color: #d59036;
      box-shadow: 0 1px 5px 0 rgba(0, 0, 0, 0.4);
    }
  }
}
@media screen and (max-width: 1124px) {
  .Filters {
    margin: 0;
  }
}
```

Всі інші файли можна знайти в архіві з готовим проектом на електронному носії.

ДОДАТОК Б

ВІДГУК
керівника економічного розділу
на кваліфікаційну роботу бакалавра
на тему:
«Розробка інтернет-магазину з продажу фарбувального обладнання з використанням бібліотеки React»
студента групи 121-18-2 Проскури Віктора Сергійовича

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

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом Проскура.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом Проскура.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
diplom.zip	Архів. Містить коди програми.
Презентація	
Презентація Проскура.ppt	Презентація кваліфікаційної роботи.