МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Секція інформаційно-комунікаційних технологій

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Веб-додаток організації продажів на основі технології React JS»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Проценко О.Б.**

Студента гр.ІКм-91 **Волошин В.В.**

СУМИ 2020

Сумський Державний Університет
(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*

Спеціальність *Інформаційно-комунікаційні технології*

Затверджую:
зав.кафедрою _____
_____
"_____" _____ 20___р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ МАГІСТЕРСЬКУ РОБОТУ
СТУДЕНТОВІ

Волошину Володимиру Володимировичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Веб-додаток організації продажів на основі технології React JS*

затверджую наказом по інституту від "__" _____ 20 20р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____
3. Вхідні данні до проекту (роботи)
_____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Інформаційний огляд. 2) Вибір методу рішення. 3) Практична реалізація.
_____
_____
_____

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
_____
_____
_____
_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

ЗАПИСКА: 54 сторінки, 5 рисунків, 14 джерел літератури, 1 додаток.

ОБ'ЄКТ ДСЛІДЖЕННЯ – технологія створення крос-браузерного і крос-платформного веб-додатку для інтернет-торгівлі.

МЕТА РОБОТИ – проектування та розробка інтернет-магазину фільтрів для води через реалізацію веб-додатка за допомогою технології React JS.

МЕТОДИ ДОСЛІДЖЕННЯ – база даних Mongo DB, технологія React JS, клієнт-серверна технологія, програмна платформа Node.JS.

РЕЗУЛЬТАТ – було обрано оптимальні технології для розробки веб-додатку; архітектура та інтерфейс забезпечують необхідний функціонал, тестування додатку показало його готовність до запуску.

MONGO DB, NODE.JS, REACT JS, ВЕБ-ДОДАТОК
ФРЕЙМОРК, ІНТЕНЕТ-МАГАЗИН

ЗМІСТ

ВСТУП

Успішний бізнес у сучасних реаліях вимагає не лише продуктів хорошої якості, а ще й просування їх у мережі Інтернет усіма доступними способами. Для того щоб отримати максимальний прибуток з онлайн продажів необхідно дотримуватись основних правил:

• рекламний маркетинг для привернення уваги потенційних клієнтів;

• достовірність загальної інформації та характеристик про продукцію;

• підтримка відносин із існуючими клієнтами через різного роду оповіщення про новинки, розпродажі, тощо;

• якісні адміністративні відносини: операції виставлення рахунку, дзвінки клієнтам, тощо [1, 2].

Онлайн торгівля дозволяє покупцям швидко та зручно обрати та оплатити необхідний товар чи послугу у будь-який час, за рахунок чого продавці в рази збільшують товарооборот. Інтернет-магазини дають змогу придбати авторський товар, який відсутній у роздрібних магазинах або не представлений у певному регіоні країни. Як показує аналітика, ринок електронної комерції демонструє позитивну динаміку зростання з кожним роком, у тому числі у країнах що розвиваються, що є позитивним для розвитку економіки. Розповсюдження пояснюється кількістю реклами в мережі, забезпечення безпеки платіжних систем і особистих даних клієнтів, покращення якості сервісу і збільшення кількості служб доставки [3].

Актуальність роботи пояснюється її практичною цінністю використання у якості елемента ведення сучасного бізнеса.

Метою роботи є проектування та розробка веб-додатку для організації онлайн продажів для магазину фільтрів для вони за допомогою React JS.

1. ЛІТЕРАТУРНИЙ ОГЛЯД

1. Аналіз принципів побудови інтернет-магазинів

Торгівля у мережі Інтернет для України явище відносно нове і розвивається не дуже стрімкими темпами через проблеми на законодавчому рівні. Проте не дивлячись на всі існуючі перепони динаміка розвитку позитивна і з кожним роком набирає оберти [3]. Все частіше покупці або організації замовляють товари через різноманітні доступні інтернет-майданчики. Для забезпечення стабільної роботи інтернет-магазинів з боку продавців необхідно забезпечити безперервний доступ в мережі Інтернет та розробити такий інтерфейс веб-додатку, який би забезпечив інтерес одразу після переходу на ресурс. Розумно продумана та спроектована архітектура і реалізований UX/UI інтерфейс є ефективним інструментом торгівлі. Головний маркетинговий хід – це дати бажане покупцю за декілька кліків мишкою і закликати користувача виконати необхідні дії. Саме тому головна сторінка формує загальну думку і від того на скільки все продумано буде залежати успіх продажів на цьому ресурсі. Розглянемо найбільш поширені інтерент-магазини.

«AliExpress» – це глобальний віртуальний торговий майданчик, яка дає можливість придбати товари, вироблені в КНР. Продукція в інтернет-магазині продається в роздріб та дрібний опт для фізичних осіб. Веб-додаток отримав свою популярність завдяки тому низьким цінам та широкому асортименту товарів, а ще при цьому не треба платити митний податок. Всі товари на цьому майданчику розбиті по категоріям, що у значній мірі прискорює пошук необхідного. За офіційними даними щодня виконується більш як 50 мільйонів покупок. Офіційно магазин був запущений в 2010 році.

Інтерфейс магазину відповідає типовому представленню сучасних інтернет-магазинів: кошик користувача, пошук за товарами та категоріями, особистий кабінет покупця з можливістю відновлення пароля, вибір способів

оплати. Варто відзначити певні особливості реалізації інтерфейсу та безпеки користувачів:

1) захист платежів покупців. Сайт магазину надає захист покупки та оплати користувачів – продавець отримує гроші за відправлений товар лише після того, як користувач підтверджує його отримання;

2) захищена оплата за фактом замовлення товару виключно віртуальними коштами, що дозволяє захистити і продавців, і покупців від зловмисників;

3) рейтинг продавців та їх товарів у системі. Крім того, веб-сайт пропонує спеціально створену систему відгуків покупців про продавців, які покупець залишає після отримання товару [4].

«Amazon» – інтернет-майданчик, головний офіс якого знаходиться у Сполучених Штатах, проте за останні роки території збільшуються, за рахунок виходу на територію Європи та Азії. Продавцями можуть бути будь-хто, необхідно тільки заплатити певні кошти за можливість виставити свої товари.

Інтерфейс аналогічно має типову структуру, у якій товари розбиваються по категоріям, пошук значною мірою може полегшити пошук і фільтрування. Через те що існують ряд обмежень на доставку в Україну, певні товари замовити або складно, або просто неможливо. Але не дивлячись на це даний сервіс набуває популярності через можливість отримати ексклюзивні товари за помірними цінами.

В Україні одним із найпопулярніших інтернет-майданчиків є «Rozetka». Аналогічно до світових лідерів даний маркетплейс представляє можливість продавати продукцію з будь-яких категорій, та у будь-якій кількості. Продавці заключають відповідні договори, які забезпечують безпеку покупців.

Перевагою є можливість обрати різні типи доставки, зокрема безкоштовно товари можна забрати у пунктах видачі магазину та різні способи оплати. Негативними моментами можна назвати відсутність україномовного інтерфейсу, точніше не повна його працездатність (не всі категорії описані українською мовою, не всі характеристики) та складність врегулювання

питання неякісного обслуговування продавцями, які просто орендують місце у маркетплейсі.

Провівши аналіз існуючих сучасних інтернет-майданчиків та магазинів (не тільки тих що були наведені) можна зробити висновок що обов’язковими функціональними частинами є:

1. каталог продукції – багаторівнева структура об’єктів, яка представлена у вигляді дерева. Верхнім рівнем є категорії товарів, які складаються з посилань на відповідні види товарів;

2. пошукова система, яка дозволяє швидко і точно знайти необхідний товар чи послугу, особливо коли є велике розгалуження категорій та підкатегорій. Можна стверджувати що це обов’язковий елемент динамічного каталогу;

3. кошик – масив даних, який зберігає повну інформацію про замовлення. Зберігання може бути тимчасовим, тобто відбувається повне видалення замовлення через визначений проміжок часу або після завершення сеансу на вкладці, так і постійним, тобто інформація не видаляється;

4. реєстраційна форма користувачів – дозволяє вводити персональні дані користувачів для виконання замовлення. Така форма особливо актуальна для постійних користувачів. Вона дозволить у майбутньому ідентифікувати користувача у системі, що у достатній мірі може прискорити час оформлення замовлення;

5. форма відправлення замовлення, яка передає дані для виконання менеджерам магазину, а з боку клієнтів дозволить самостійно переглядати статус виконання.

1.2. Інструменти для розробки веб-додатків

Інтернет-магазин може бути реалізований як з боку сервера, так і з боку клієнта, все залежить від поставлених задач від замовника або необхідної функціональності. У першому випадку потрібно використовувати сценарії

сервера на основі ASP, Perl, PHP та інших. З іншого боку можна використовувати JavaScript, Java.

Популярним рішенням для оптимізації роботи використовувати системи управління контентом або CMS – це система управління інформацією, яка використовується для заповнення, управління та підтримки вмісту веб-програми. Основною метою є збір та управління інформацією з різних джерел. Ця система дозволяє різним співробітникам взаємодіяти з базою даних, роблячи процес пошуку і повторного використання інформації комфортним.

CMS містять досить різноманітну інформацію: текстові документи, аудіо, графічні та відео файли, каталоги з даними і дозволяють зручно зберігати, керувати, обробляти, переглядати і публікувати такого роду дані.

Розробка сайту, який включає в себе готову систему управління контентом, має ряд переваг:

1) готовий інструмент вирішення завдань;

2) клієнт може оновити контент на сайті самостійно;

3) універсальність сучасних систем управління [5].

Крім готових рішень, багато розробників пропонують власні системи розробки. Вони підходять для невеликих сайтів, коли функціональність адміністративної панелі досить проста і в майбутньому не потребує вдосконалення. Ситуація набагато складніша з великими системами управління, особливо тими, які не пройшли функціональне і програмне тестування компонентів. Такі системи важко підтримувати і вони є уразливими для хакерських атак.

Іншим прикладом підходу до розробки є використання фреймворків – програмної платформи, яка визначає структуру інформаційної системи та включає в себе набір бібліотек для розробки різних компонентів програмного забезпечення. Він містить базові програмні модулі, які реалізують конкретні компоненти програми. Фреймворк пропонує розробнику вбудовані компоненти для написання функціональних форм, роботи з базою даних та опису логіки. Використовуючи ці програмні модулі, розробник економить час,

щоб супроводжувати проект і вносити зміни і досягнення найбільшої продуктивності. Таким чином, можна виділити наступні переваги фреймворку:

1) реалізовані проекти легко масштабуються і модернізуються;

2) додатки працюють швидше і підвищують рівень взаємодії користувача з продуктом;

3) безпечні;

4) стандартизована структура додатків дозволяє розробнику швидко зібрати «каркас» сайту за допомогою готових компонентів фреймворку, таких як авторизація, пошук тощо.

Однак є і недоліки:

1) фреймворки містять лише базову бізнес-логіку, тому більшість компонентів потрібно впроваджувати та налаштувати під потреби проекту;

2) для використання фреймворку потрібні знання його бізнес-процесів і структури, а також вміння правильно їх застосовувати [6].

1.3 Постановка задачі

Виходячи з того що інтерент-магазин має розширити географію продажів, зацікавити потенційних клієнтів і зберегти вже існуючих необхідно реалізувати наступні задачі:

1. виконати огляд інструментів для розробки та обрати оптимальні;

2. розробити інтуїтивно зрозумілий інтерфейс веб-додатка;

3. виконати проектування архітектури веб-додатка реалізації товарів;

4. розробити алгоритм роботи;

5. виконати тестування веб-додатку та зробити відповідні висновки по роботі.

У разі успішної реалізації буде можливим слідкувати за існуючими позиціями товарів, формувати замовлення.

2. МЕТОДИКА ВИРІШЕННЯ ПОСТАВЛЕНИХ ЗАДАЧ

Перш ніж проектувати веб-додаток необхідно добре продумати архітектуру сайту, для того щоб користувач під час пошуку необхідних товарів або послуг витратив мінімум зусиль і мав позитивний досвід від роботи з ресурсом. Саме тому архітектура сайту орієнтована в першу чергу на користувачів і направлена на просування в мережі Інтернет. Сучасна структура інтерне-магазинів має наступну структуру (рис.2.1):

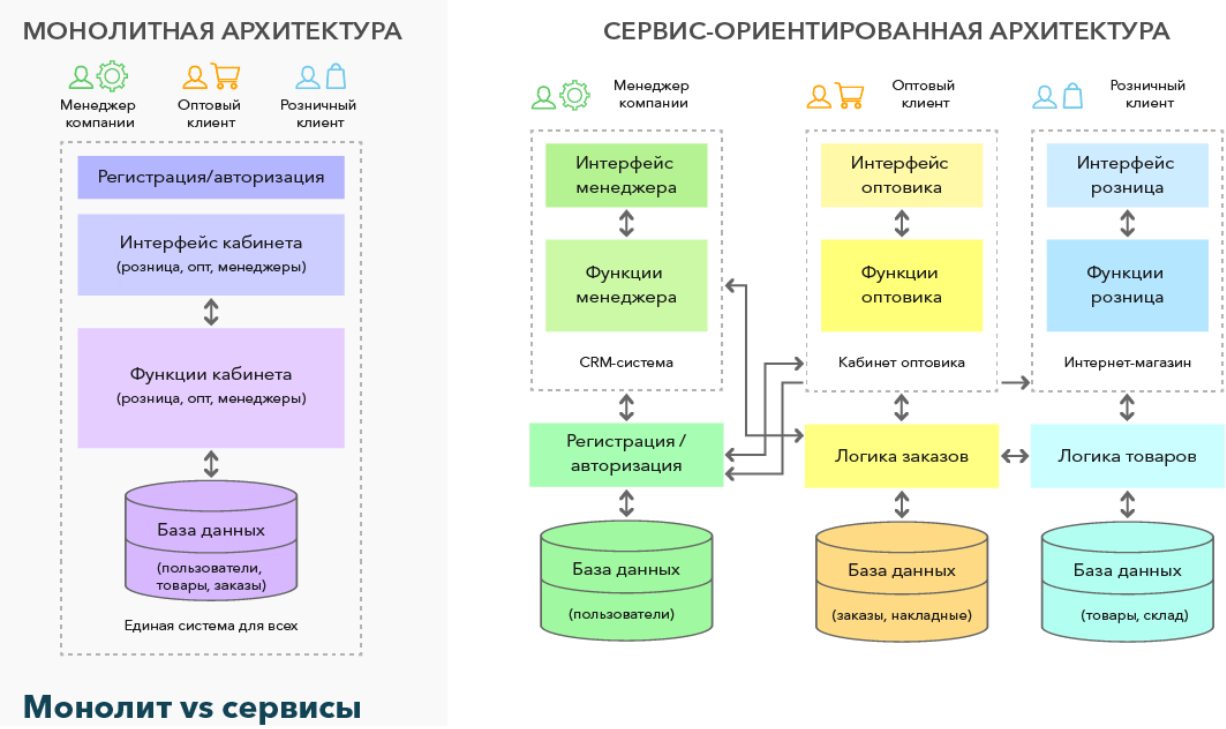

Рисунок 2.1 – Архітектура інтерент-магазину [7]

Загальний алгоритм роботи інтернет-магазину можна пояснити наступними кроками (рис.2.2):

1) користувач переходить на головну сторінку веб-додатку, де пропонується вибрати категорію певного товару. Якщо користувач не впевнений, до якої саме категорії належить необхідний продукт, то можна вибрати вкладку «Вся продукція» для ознайомлення з асортиментом всього магазину;

2) знайти необхідний товар покупець може продивившись вкладку з усіма найменуваннями або ж оптимізувати пошук за рахунок фільтрів та сортування по категорії, виробнику або знайти назву в пошуку;

3) після вибору товару користувачеві запропонують ознайомитися зі способами оплати і доставки та запропонують оформити заявку-замовлення.

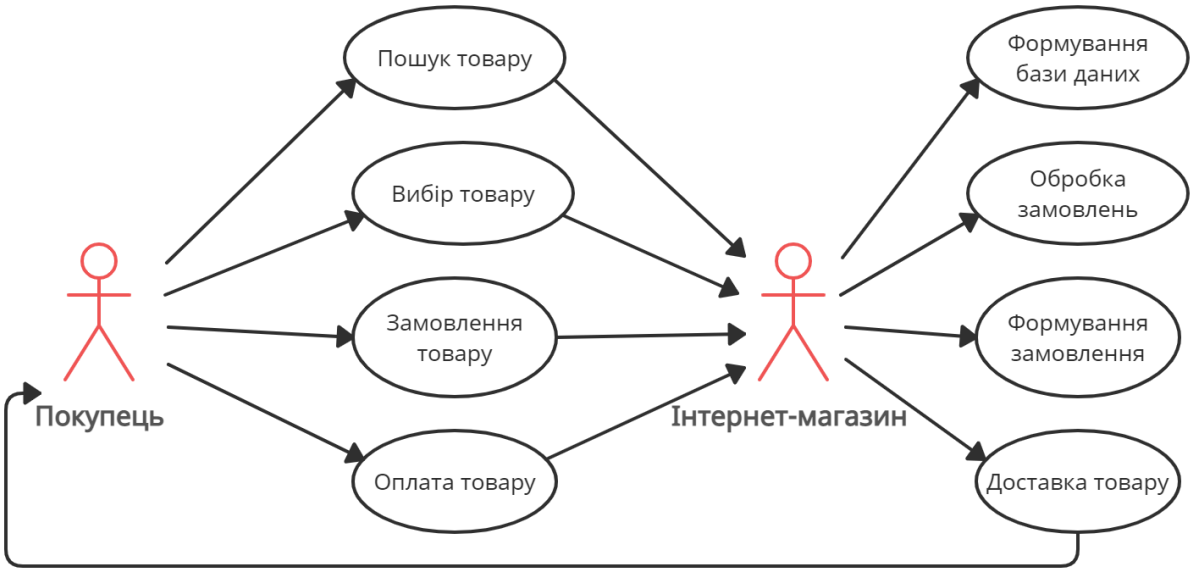

Рисунок 2.2 – Загальна діаграма взаємодії покупця та інтернет-магазину

2.1 Проектування бази даних

Усі програмні системи потребують зберігання інформації, яку потрібно обробити у певний момент часу звернення до певного блоку. Для цього використовуються структуровані SQL-запити обробки інформації реляційних базах даних. Такий тип БД використовують при роботі з крупними проектами, завдяки якісним і продуктивним рішенням для зберігання і управління даними. Однак реляційні бази даних не можуть бути застосовані до багатьох сучасних рішень розробки додатків, що призводить до пошуку альтернативних систем зберігання та управління даними. Фреймворк React JS працює виключно з документо-орієнтованими моделями баз даних, а саме Mongo DB.

Дана система керування дозволяє зберігати та керувати інформацією за допомогою безсхемної організації неструктурованих або слабоструктурованих даних. Документо-орієнтований спосіб організації даних не призначений для зв’язування різних документів і колекцій через ключові поля, оскільки не має власних засобів для створення таких зв’язків. Таким чином Mongo DB використовується для зберігання даних у документах колекцій у форматі JSON, які не мають чіткої структури та призначені для зберігання будь-якого типу даних. Такий формат та організація зберігання інформації гарантує швидку обробку, запис та вивід інформації, що чудово підходить для роботи з нереляційними даними, при цьому забезпечуючи високу продуктивність і необмежене горизонтальне масштабування колекцій (рис.2.3) [8-9]:

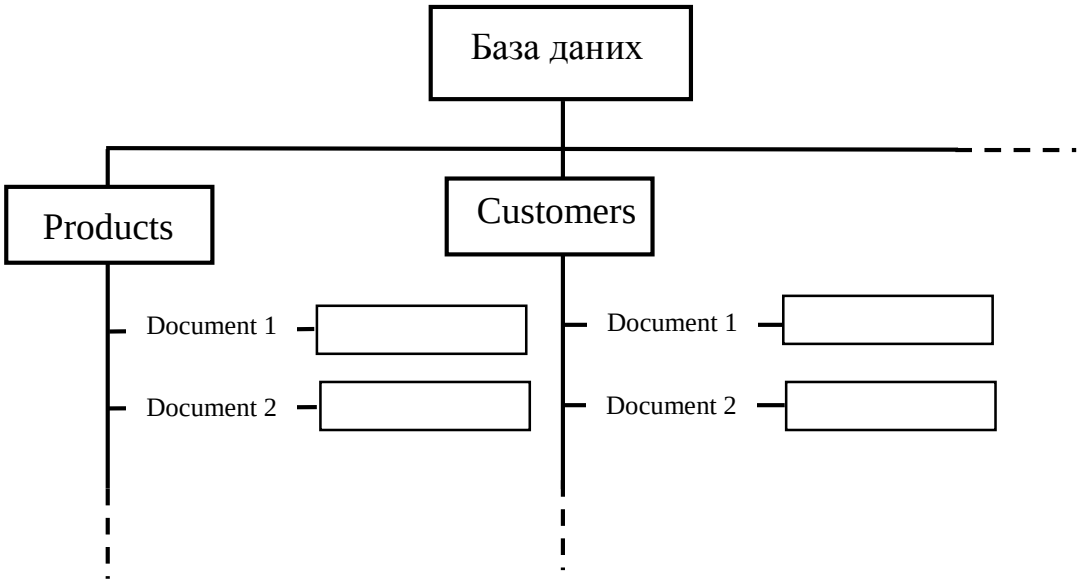

Рисунок 2.3 – Приклад бази даних Mongo DB

Створення реляційних баз починається зі структури таблиць, встановлення зв’язків, їх заповнення і формування SQL-запитів. Для Mongo DB спочатку створюється нова база даних з ім’ям, потім створюється колекція при додавати до неї даних, при цьому створюється документ з описом доданого об’єкта у форматі binary JSON. Дані у документах представляють собою структуру зв’язків ключ-значення у форматі BSON.

Структура Mongo DB не вимагає заздалегідь строго типізованої схеми, яка описує поля та типи даних, що може бути корисно, коли відбувається часта зміна наборів ресурсів даних. Основні особливості Mongo DB:

1) висока продуктивність;

2) висока швидкість зчитування та запису даних;

3) легка масштабованість і надійність зберігання даних [10].

Для реалізації веб-додатку необхідно розробити документо-орієнтовану базу даних з використанням результатів дослідження типових інтернет-магазинів та їх баз даних.

2.2 Фреймворк React JS

React JS – це JavaScript-бібліотека з відкритим вихідним кодом для розробки користувацьких інтерфейсів односторінкових та мобільних додатків. Фреймворк забезпечує високу швидкість взаємодії, масштабність і простоту у роботі. Особливості фреймворка:

1) використовує однонаправлену передачу даних від батьківських компонентів до дочірніх;

2) використовує віртуальний DOM, що дозволяє вирахувати різницю між попереднім і поточним станом інтерфейсу для оптимального оновлення браузера;

3) використовує спеціальне розширення синтаксису JavaScript XML (JSX), який дозволяє застосовувати схожий на код HTML для опису структури інтерфейсу;

4) реалізує методи життєвого циклу, що дозволяє запустити код на різних стадіях «життя» додатку.

Основні переваги React JS:

1) зручне та легке створення модулів та компонентів за допомогою API, яке підтримує композицію;

2) велика кількість готових розширень з відкритим кодом.

Мінуси фреймворку:

1) потрібне налаштування JSX, а повні рішення вимагають підключення сторонніх бібліотек;

2) наявність оптимізованих опцій затрудняє діяльність розробника-початківця [11].

2.3 Серверна частина веб-додатку

При проектування серверної частини архітектури необхідно враховувати взаємодію програмного забезпечення додатка з послугами провайдерів, на яких буде встановлений готовий продукт. Тобто необхідно знайти найкращий варіант хостингу або сервера для коректної роботи веб-додатка. Оптимальним рішенням буде скористатися віртуальним сервером для розміщення готового інтернет-магазину.

VPS – це орендований віртуальний виділений сервер, який відповідає фізичному серверу з точки зору управління операційною системою та містить власні IP-адреси, таблицю маршрутизації, кореневий доступ і порти. Перевагою серверів такого типу є можливість створювати та змінювати власні версії системних бібліотек, видаляти та додавати будь-які файли, а також встановлювати, налаштовувати та змінювати необхідні програми.

Деякі провайдери також надають доступ до апаратних систем віртуалізації з можливістю налаштування ядра операційної системи та драйверів пристроїв. Віртуальний виділений сервер емулює роботу окремого фізичного сервера, забезпечуючи управління і незалежний контроль для власника. Кожен сервер, як правило, має свої власні ресурси, конфігурацію, процеси та адміністрації на операційних системах Unix та Linux.

Оренда віртуального сервера є найпопулярнішим типом хостингу, який забезпечує оптимальний баланс між ціною та можливістю для більшості власників веб-додатків. Ціна може змінюватися у залежності від пакета необхідних послуг і підтримки постачальника. Сервери без підтримки

надаються за нижчими цінами і вимагають базових навичок адміністрування операційної системи та веб-додатків. Цей тип хостингу більше підходить для професіоналів і розробників з досвідом.

Порівняння VPS з віртуальним хостингом:

1) кожен віртуальний виділений сервер має власну копію системи з доступом до кореневого каталогу, що дозволяє компілювати, встановлювати власне програмне забезпечення з подальшою зміненою конфігурації;

2) гарантований мінімум ресурсів пам’яті;

3) можливість архівувати віртуальний сервер усієї системи або виконати резервне копіювання;

4) використання технології iSCSI дозволяє забезпечити високу швидкодію.

Для зручного налаштування, а потім зберігання та редагування веб-додатку було вирішено використовувати VPS-сервер, виходячи з усіх його переваг і простоти використання для поставленого завдання.

Таким чином можна стверджувати що програмна частина розглядається як взаємозв’язок операційної та серверної частин. Операційна частина – це безпосередньо середовище розробки веб-додатків, серверна – це розміщення проекту на хостингу, адміністративна – це інструменти управління контентом.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-ДОДАТКУ

3.1. Реалізація клієнтської частини за допомогою React JS

Структура веб-додатку була розроблена виходячи з поставлених задач і необхідного функціоналу. Навігація складається з двох необхідних груп: каталог і додаткові розділи.

Головна сторінка інтерфейсу складається з каталогу доступних товарів (Рис. 3.1), інформаційне меню розташоване у шапці додатка. У ньому представлена вся необхідна для покупців інформація:

1) інформація про магазин;
2) контакти;
3) оплата і доставка;
4) корисна інформація.

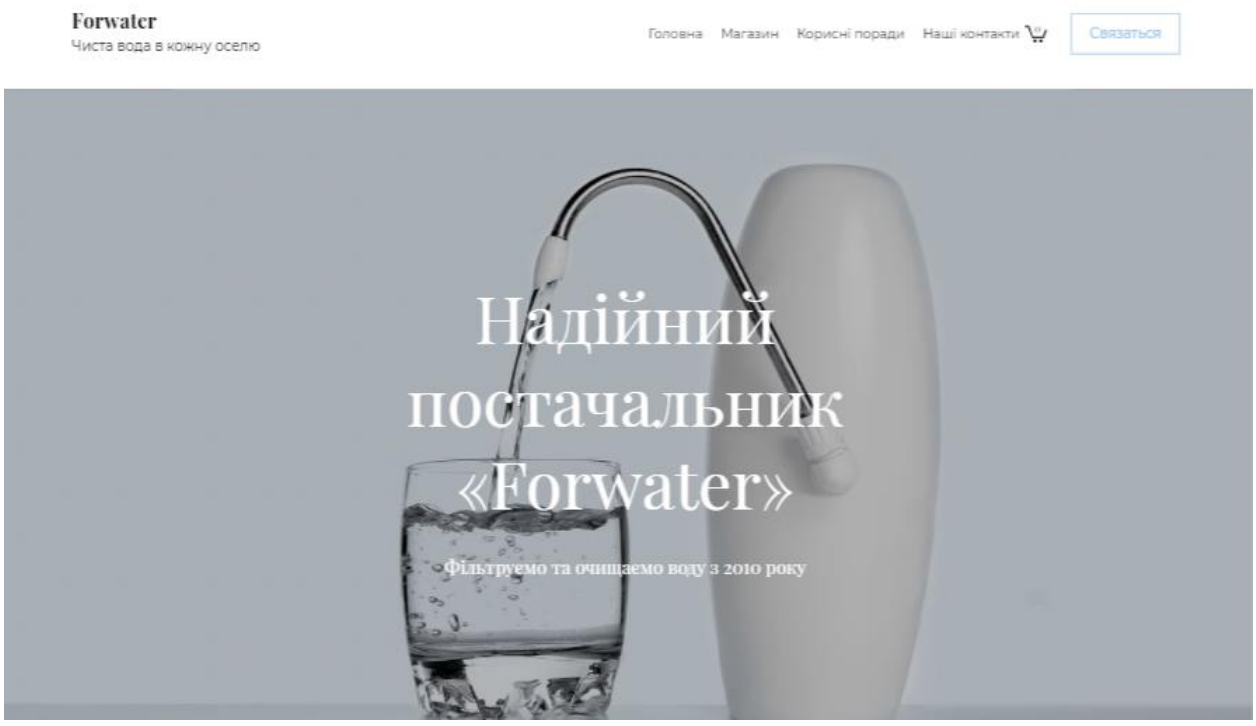

Рисунок 3.1 – Головна сторінка інтерфейсу

Під час реалізації інтерфейсу головної сторінки було додано слайдер з динамічним завантаженням даних для категорій товарів. Каталог продукції є

однією з найважливіших особливостей будь-якого інтернет-магазину. Саме за допомогою нього більшість користувачів виконує пошук потрібного продукту або послуги. Тому основним завданням було розробити зручний та структурований каталог для ознайомлення з асортиментом магазину.

Всі найменування у каталозі було розділено на логічні групи і таким чином сформувалось 5 категорій. Також було враховано динамічні групи товарів, які можуть з’являтися і зникати протягом року. До таких спеціальних груп відносяться товари у зв’язку з святковими акціями або сезонним розпродажем. Для зручного пошуку та перегляду продукції інтерфейс і дизайн для даної категорії блоку було уніфіковано.

Зовнішній вигляд кожного елемента в каталозі також уніфікований для зручності обробки даних користувачем. Як правило, на сторінці кожної з категорій відображається зображення товару, його назва і ціна. Якщо у каталозі більше 25 позицій, то у нижній частині списку реалізується функція переходу на іншу сторінку, а також реалізована функція контроля кількості одиниць товару на одній сторінці.

Окрім зазначених інструментів, для каталогу було реалізовано допоміжні навігаційні інструменти: фасетна навігація, рекламні матеріали, навігація по виробникам, корисні статті та рекомендації для вибору. Така структура сторінки каталогу збільшує шанси продати товар і зменшує кількість запитань до продавців.

Фасетна навігація дозволить користувачеві орієнтуватися по каталогу товарів, оскільки у різних категоріях буде різна кількість позицій товарів. Тому системи фільтрації товару дозволять швидко знайти потрібний не тільки за категоріями, але і за набором параметрів. Веб-додаток реалізує просту фільтрацію: за ціною, маркою, призначенням і виробником. Це загальні параметри, які застосовуються до всіх елементів. У інтерфейсі фільтрація розташована в лівій частині екрана (даний тип розташування є найбільш поширеним). Однорідні параметри візуально згруповані, а сам інтерфейс реалізується з динамічним завантаження даних, що дозволить не

перезавантажувати сторінку при виборі необхідних параметрів, тобто дані оновлюються автоматично.

Елемент «Пошук» в інтерфейсі інтернет-магазину візуально виділений і розташований на сайті шапки. Під час введення запиту автоматично з’являться підказки з бази даних товарів, для полегшення введення інформації.

Картка товару містить детальний опис товару, представляє характеристики, рекомендації з експлуатації, тощо. Усі картки уніфіковані, щоб клієнти могли зосередитися на інформації про конкретний продукт. У верхній частині сторінки розташована стандартна інформація, яка складається з контактів та пошуку товарів. Під меню знаходяться назви виробу, а нижче - вся інформація що відноситься до цього артикулу. Також на сторінці обов’язково є зображення товару, ціна (в тому числі зі знижкою), інформація про наявність, характеристики товару, кнопка «Купити», коротка інформація про оплату, доставку, гарантію.

3.2 Реалізація бази даних Mongo DB

Mongo DB – це безкоштовна база даних з відкритим вихідним. Для реалізації було встановлено поточну версію 4.4.2 під 64-розрядну ОС Windows 10, і встановлено відповідні драйвери для організації роботи між Mongo DB та React JS. Базу даних Shop створено для реалізації архітектури веб-додатку, яка складається з кількох основних колекцій (рис. 3.2):

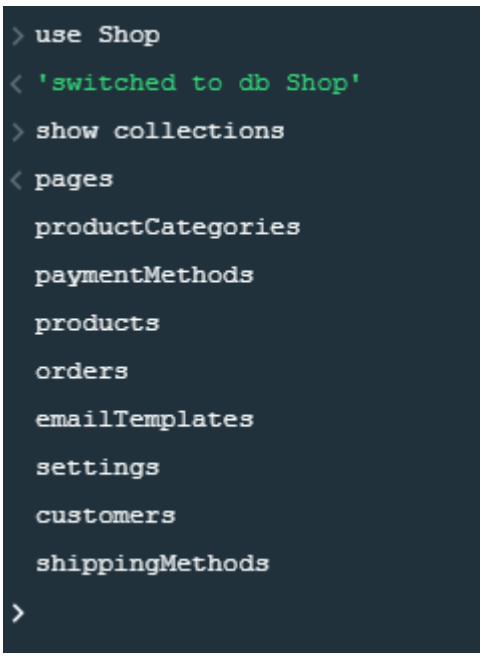

Рисунок 3.2 – Структура БД Shop

Колекція «Покупці» містить інформацію про користувача та ідентифікаційні ключі користувача. Додавання документів у колекції здійснюється за допомогою операції «Insert», результат виконання функцією «Find». Документ містить такі ключі:

1) ідентифікація за електронною поштою;

2) номер телефону;

3) логін;

4) адреса доставки, якщо вона доступна.

Колекція «Шаблони повідомлень» містить шаблони електронних листів, які при реєстрації в інтернет-магазині надходять за вказаною користувачем адресою. Колекція містить інформацію про замовлений товар, терміни та адресу доставки, спосіб оплати та суму замовлення. Автоматично генерується підпис листа з повними контактними даними продавця. Документ містить такі ключі:

1) ідентифікатор листа та ім’я шаблону;

2) зміст листа;

3) інформація про доставку та товари;

4) кількість замовлених одиниць і сума для оплати замовлення.

Колекція «Замовлення» містить інформацію про всі замовлення користувачів. Документ містить такі ключі:

1) номер замовлення;

2) електронна пошта користувача;

3) телефон користувача;

4) П.І.Пб.;

5) адреса доставки;

6) замовлені одиниці товару.

Колекція «Сторінки» містить всі системні сторінки, серед яких: Головна, Кошик, Контакти, Корисні поради, Відомості про компанії-виробники. Документ містить такі розділи:

1) ідентифікатор (id) сторінки та відносний шлях до неї;

2) мета-заголовок сторінки і доступність сторінки.

Приклад для «Кошик»:

```
{"_id" : ObjectId("5cf39afffb3f332214e736cw82p"), "slug" : "cart",
"meta_title" : "Кошик", "enabled" : true, "is_system" : true}
```

Колекція «Способи оплати» містять дані про оплату товарів і складається з таких розділів:

1) ідентифікатор об'єкта, яких створюється автоматично;

2) тип оплати (готівка, PayPal, Приватбанк).

Колекція «Категорія товарів» містить інформацію про категорії та підкатегорії товарів. Документ містить такі розділи:

1) ідентифікатор і назва категорії;

2) відносний шлях та адреса зображення;

3) інформація про батьківську категорію (за наявності) та доступність.

Приклад:

```
{"_id" : ObjectId("5cf39afbf33f332214e736cw986"), "name" : "Побутові фільтри",
"slug" : "products", "image" : "", "parent_id": null, "enabled" : true}
```

Колекція «Продукція» містить детальну інформацію та характеристики про кожен товар на сайті. Документ містить такі розділи:

1) ідентифікатор і назва продукту;

2) відносний шлях і ідентифікатор категорії, до якої належить товар;

3) ціна та доступна кількість продукції;

4) наявність, акції, характеристики продукту (назва бренду, колір, комплектація, тощо.).

Колекція «Налаштування» містить інформацію про глобальні змінні веб-додатка.

Колекція «Доставка» містить інформацію про можливі варіанти доставки замовлення (кур'єрська доставка виключно у межах м. Суми). Документ містить такі розділи:

1) ідентифікатор доставки;

2) тип доставки;

3) наявність на складі у даний момент;

4) умови доставки (область, місто, мінімальна/максимальна кількість товарів, мінімальна/максимальна вага товару в порядку).

3.3 Реалізація серверної частини

Розгортання документо-орієнтованої бази даних Mongo DB можливо, тільки на VPS-сервері. У якості хмарного серверу було обрано платформу VPS.ua.

У якості програмної платформи було обрано Node.JS, яка дає можливість JavaScript взаємодіяти з пристроями вводу-виводу через API, підключати зовнішні бібліотеки, написані різними мовами. Node.JS використовується здебільшого на сервері, діючи як веб-сервер, але можна використовувати і при розробці десктопних додатків. Найпопулярніша система управління базами даних для Node.JS на даний момент є Mongo DB.

Під час підключення та взаємодії з базою даних у Mongo DB можна виділити наступні кроки:

1) підключення до сервера;

2) отримання об’єкта бази даних на сервері;

3) отримання об’єкта колекції в базі даних;

4) взаємодія з колекціями (додавання, видалення, отримання, тощо).

Ключовим класом для роботи з Mongo DB є клас Mongo Client, через нього будуть йти всі взаємодії зі сховищем даних. Підключення до сервера відбувається за допомогою метода connect (). Спочатку створюється об’єкт клас Mongo Client. Для цього до його конструктора передаються два параметри. Першим параметром є адреса сервера. Як адреса протоколу встановлюється «mongodb://». Локальною адресою буде localhost після чого вказується номер порту 27017.

Другим параметром використовується об’єкт конфігурації, який має властивість useNewUrlParser: true яке вказує інфраструктурі Mongo DB потрібно використовувати новий парсер адреси сервера. Далі за допомогою методу connect виконується підключення до сервера. У якості параметра, метод приймає функцію зворотного виклику, яка запускається при встановленні підключення. Ця функція приймає два варіанти: err у результаті помилки в установці з'єднання і client – це посилання на клієнта, підключеного до сервера. При завершенні роботи з базою даних потрібно закрити з’єднання за допомогою метода client.close().

Отримавши об’єкт підключеного клієнта можна отримати доступ до бази даних на сервері. Для цього використовується метод client.db (). У якості параметра в метод передається назва бази даних, до якої необхідно під’єднатися.

Як уже було сказано раніше Mongo DB не має таблиць. Замість цього всі дані знаходяться у колекції. У випадку взаємодії з Node.JS для роботи потрібно буде отримати об’єкт колекції за допомогою методу db.collection(), з назвою колекції в аргументі методу. Щоб додати один документ-об’єкт user

застосовується метод insertOne(). Цей метод має функцію зворотного виклику, яка виконується після додавання. Запуск серверу Mongo DB завершує налаштування реалізації серверної частини.

3.4 Тестування працездатнотності веб-додатку

3.4.1 Функціональне тестування

На даному етапі роботи необхідно перевірити виконання функціональних вимог, відповідно до поставлених задач для розробки. При вході на сайт користувач повинен отримати всю необхідну інформацію про пропоновану продукцію. Крім того важливим моментом є, щоб інформація була актуальною на момент покупки, відображались можливі акції, промо-коди на товари, контактна інформацію про магазин, способи оплати і доставки товару. Необхідно перевірити функціонування елементів, які знаходяться на головній сторінці, а саме:

1) навігація по магазину (категорії товарів і спеціальні пропозиції);

2) операція пошуку;

3) автоматична робота слайдера і ручне перегортання на версіях для ПК і мобільних пристроях;

4) перехід на відповідні сторінки при натисканні на банери зі спеціальними пропозиціями.

Після того, як було переглянуто головну сторінку, виконується перехід до каталогу з товарами. Каталог при цьому виступає у якості навігатора для магазину, включає в себе категорії і підкатегорії. Основні етапи верифікації:

1) перехід до будь-якої з категорій та підкатегорій;

2) на мобільних пристроях каталог трансформується в зручне меню з випадаючим списком;

3) каталог складається з сітки активних товарів з назвами і відповідними цінами;

5) при натисканні на продукт відбувається перехід на сторінку з докладним описом та характеристиками;

6) товар можна додати у кошик.

Після того як було знайдено необхідний товар відбувається перехід на сторінку з детальним його описом. Представлена інформація має допомогти зробити остаточний вибір про придбання. На сторінці з докладним описом характеристик обов'язковими елементами є:

1) найменування товару;

2) артикул;

3) опис характеристик продукту;

4) зображення;

5) ціна;

6) наявність на складі.

Додавання до кошику обраного товару є важливим етапом онлайн-покупки, де користувач не тільки збирає інформацію про продукти, які їм подобаються, але і відкладає їх для подальшого вивчення і прийняття рішення про покупку. З точки зору функціональності, важливо спочатку перевірити, що:

1) товар додається в кошик і це очевидно користувачеві (є повідомлення і міні-кошик поповнюється обраною позицією);

2) покупець бачить повну вартість всієї покупки в міні-картці;

3) ціни правильні, тобто не змінюються при додаванні в кошик, при додаванні декількох пунктів достовірно сумується їх вартість з урахуванням знижок або промокодів, якщо вони були застосовані;

4) доступна можливість переглядати вміст кошика у будь-який час;

5) якщо обрана адресна доставка товару, то вартість доставки відображається в загальній сумі до сплати;

6) користувач може змінити кількість товарів - додати або видалити їх;

7) якщо товару немає на складі, його не можна додати в кошик;

Після вибору товару і додавання його у кошик відбувається фомування замовлення. При тестуванні процесу оформлення замовлення необхідно звернути увагу на:

1) для фізичних осіб та організацій деталі замовлення будуть різними. Ці нюанси необхідно продумати у реєстраційній формі;

2) можливість повернутися до попереднього кроку оформлення і змінити вже внесені дані;

3) додати можливість вилучити товар зі списку або, навпаки, збільшити їх кількість.

Для того щоб замовлення врешті решт було оформлено, необхідно щоб процес не займав багато часу і був інтуїтивно зрозумілим.

3.4.2 Крос-браузерне і крос-платформне тестування

З поширенням різних гаджетів світ став більш мобільним і все більше і більше людей хочуть мати можливість зробити замовлення за допомогою не тільки ПК, тому інтернет-магазин повинен бути крос-браузерним і крос-платформним – тобто ідеально виглядати в будь-якому браузері, при будь-якій роздільній здатності екрану і на будь-якому пристрої. Помилки макета не такі критичні, як помилки функціональності, але вони можуть призвести до того, що покупець відмовиться від покупки на сайті. Тому тестування макета інтернет-магазину має одну істотну особливість: слід враховувати пункти прийняття рішень. Важливо звернути увагу на наступні кроки:

1) перегляд каталогу. Навряд чи користувач захоче продовжити процес покупки, якщо продуктова сітка не сприймається: товар повинен бути вирівняний по відношенню один до одного та не повинен накладатися один на одного, кнопка додавання до кошику не повинна зникати та змінювати своє положення з прокруткою;

2) перегляд товару. У будь-якому браузері сторінка з детальним описом продукту повинна виглядати однаково, а зображення не повинні бути обрізані або розтягнуті на всю ширину екрану на мобільних пристроях;

3) оформлення замовлення. Занадто мала кнопка оплати або неактивні комірки-прапорці в одному з браузерів можуть бути значною перешкодою щоб зробити покупку.

Тестування розробленого веб-додатку показало, що розроблена структура зрозуміла для користувача і має необхідний функціонал. У готовому рішенні навігація ділиться на дві групи: каталог і допоміжні розділи. При розробці інтерфейсу, основна увага привертається до каталогу. Він дозволяє користувачеві переміщатися по групах продуктів і вибирати те, що їм потрібно. Інформаційне меню представлено у шапці сайту. У ньому є необхідні розділи для клієнтів. Зовнішній вигляд кожного елемента у каталозі уніфікований для зручності обробки даних користувача. Якщо в каталозі більше 25 елементів, реалізується перемикання сторінок.

Для реалізації архітектури веб-додатків була створена база даних за допомогою інструменту Mongo DB, у якому знаходиться кілька колекцій: покупці, шаблони повідомлень, замовлення, категорії продуктів, налаштування доставки замовлення.

Тестування веб додатку показало його повну працездатність і готовність до користування.

ВИСНОВКИ

У ході виконання випускової кваліфікаційної роботи магістра було виконано літературний огляд за тематикою розробки веб-додатків на основі чого було сформовано задачі для виконання.

У результаті виконання було розроблено веб-додаток, який дозволяє:

1) ефективно систематизувати замовлення;
2) зберігати і накопичувати клієнтську базу;
3) просувати продажі продукції у мережі.

Тестування додатку показало задовільний результат, таким чином можна вважати роботу над додатком закінченою.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. The Ultimate Guide To eCommerce Software [Електронний ресурс] // Zapier Inc. – 2016. – Режим доступу до ресурсу: https://zapier.com/learn/ecommerce/.

2. E-commerce Web Application: Why Your Business Needs One? [Електронний ресурс] // https://www.seasiainfotech.com/ – Режим доступу до ресурсу: https://www.seasiainfotech.com/blog/e-commerce-web-application-business-needs-one/.

3. Малюта І. А., Оголь А. Є. Аналіз сучасного стану та перспективи розвитку інтернет-торгівлі в Україні. *Ефективна економіка*. 2019. № 1. – Режим доступу до ресурсу: http://www.economy.nayka.com.ua/?op=1&z=6845

4. Aliexpress com - интернет магазин товаров из Китая, инструкция по покупке и доставке [Електронний ресурс] – Режим доступу до ресурсу: https://www.vxzone.com/internet-shops/catalog-eshops/hypermarkets/687-aliexpress.html.

5. Що таке CMS сайту [Електронний ресурс] – Режим доступу до ресурсу: https://hostiq.ua/wiki/ukr/cms-ukr/.

6. ТОП-5 PHP-фреймворков 2017 года [Електронний ресурс] – Режим доступу до ресурсу: https://techrocks.ru/2017/12/09/top-5-php-frameworks-list/.

7. Електронний ресурс – Режим доступу до ресурсу: https://evergreens.com.ua/ru/development-services/web-services-development.html

8. What Is MongoDB? [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/what-is-mongodb.

9. Relation Data Model [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/dbms/relational_data_model.htm.

10. Парфенов Ю. Э. Особенности разработки java-приложений для MongoDB / Ю. Э. Парфенов. // Системи обробки інформації. – 2017. – №2 (148). – С. 57–60.

11. Посібник: знайомство з React [Електронний ресурс] – Режим доступу до ресурсу: https://uk.reactjs.org/tutorial/tutorial.html.

12. Guides [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/en/docs/guides/.

13. Vice R. Mastering React / R. Vice, A. Horton., 2016. – 260 с. – (Packt Publishing).

14. Lets Understand The ReactJS Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.phptpoint.com/reactjs-tutorial/

ДОДАТОК А

Додаток А1. Лістинг коду головної сторінки

```
import React, { Fragment } from 'react';
import PropTypes from 'prop-types';
import { themeSettings } from '../lib/settings';
import MetaTags from '../components/metaTags';
import CustomProducts from '../components/products/custom';
const IndexContainer = props => {
 const {
 addCartItem,
 state: { pageDetails, settings }} = props;
 return (
 <Fragment>
 <MetaTags
 title={pageDetails.meta_title}
 description={pageDetails.meta_description}
 canonicalUrl={pageDetails.url}
 ogTitle={pageDetails.meta_title}
 ogDescription={pageDetails.meta_description}
 />
 <FrontPageSlider slide={listItems[0]}/>
 </Fragment>);};
IndexContainer.propTypes = {
 addCartItem: PropTypes.func.isRequired,
 state: PropTypes.shape({
 settings: PropTypes.shape({}),
 pageDetails: PropTypes.shape({})
 }).isRequired};
```

```
const listItems = [
{title: Фільтри зворотного осмосу', content: ['Зворотній', 'осмос'], additionalClass: 'item-slide reverse-osmosis ', categoryLink: '/ reverse-osmosis '},
{title: 'Глечики-фільтри', content: ['Фільтри', 'глечики'], additionalClass: 'item-slide filter-jugs ', categoryLink: '/ filter-jugs '},
{title: 'Змінні та комплектуючі елементи', content: ['Комплектуючі', 'складові'], additionalClass: 'item-slide filter-accessories ', categoryLink: '/ filter-accessories '},];
function SlideItem(props) {
 return (
 <div className={props.additionalClass}>
 <div className="item-description">
 <h2>{props.title}</h2>
 <h1>
 {props.content[0]}
 <br />
 {props.content[1]}
 </h1>
 <a href={props.link} ><span>Переглянути все</span></a>
 </div>
 <div className="item-image" />
 </div>);}
const Indicators = (props) => {
 const listIndicators = listItems.map((item, index) =>
 <li key={index} className={props.currentSlide === index ? 'active' :
"} onClick={() => props.changeSlide(index)}>
 <span>{listItems[index].content}</span>
 </li>);
 return (
 <ul className="indicators">
```

```
{listIndicators}
</ul>);
class FrontPageSlider extends React.Component {
constructor(props) {
super(props);
this.state = {
slideshow: props.slide,
slideIndex: 0};
this.currentIndex = 0;
this.pause = false;}
componentDidMount() {
var that = this;
this.timeout = setTimeout(function () {
that.goTo('auto')}, 6000);}
componentDidUpdate() {
var that = this;
if (this.pause === true) {
clearInterval(this.timeout);
this.timePause = setTimeout(function() {
clearInterval(this.timePause);
}, 8000);
this.pause = false;}
this.timeout = setTimeout(function () {
that.goTo('auto')
}, 6000);}
componentWillUnmount() {
clearInterval(this.timeout);}
goTo = (direction) => {
let index = 0;
switch(direction) {
```

```
case 'auto':
index = this.currentIndex + 1;
this.currentIndex = index >= listItems.length ? 0: index;
break;
case 'prev':
index = this.currentIndex - 1;
this.currentIndex = index < 0? listItems.length - 1: index;
this.pause = true;
break;
case 'next':
index = this.currentIndex + 1;
this.currentIndex = index >= listItems.length ? 0 : index;
this.pause = true;
break;
default:
this.currentIndex = direction;
this.pause = true;
break;}
this.setState({
slideIndex: this.currentIndex,
slideshow: listItems[this.currentIndex]
});};
render() {
return (
<div className="slideshow-simple">
<SlideItem
title={this.state.slideshow.title}
content={this.state.slideshow.content}
additionalClass={this.state.slideshow.additionalClass}
link={this.state.slideshow.categoryLink}
```

```
 />
 <Indicators
 changeSlide={this.goTo}
 currentSlide={this.state.slideIndex}
 content={this.state.slideshow.content}
 />
 </div>);}
export default IndexContainer;
```

Додаток А.2. Код каталогу

```
import React, { Fragment } from 'react';
import PropTypes from 'prop-types';
import { themeSettings, text } from '../lib/settings';
import MetaTags from '../components/metaTags';
import ProductList from '../components/productList';
import ProductFilter from '../components/productFilter';
import Sort from '../components/sort';
import CategoryBreadcrumbs from '../components/categoryBreadcrumbs';
import AdditionalBreadcrumbs from '../components/additionalBreadcrumbs';
import * as helper from '../lib/helper';
const getFilterAttributesSummary = productFilter => {
 let attributesSummary = '';
 if (productFilter.attributes) {
 Object.keys(productFilter.attributes).forEach(attributeKey => {
 const attributeName = attributeKey.replace('attributes.', '');
 const attributeValue = productFilter.attributes[attributeKey];
 const attributeValueFormatted = Array.isArray(attributeValue)
 ? attributeValue.join(', ')
 : attributeValue;
```

```
 attributesSummary += `. ${attributeName}: ${attributeValueFormatted}`;
});} return attributesSummary;};
const getFilterPriceSummary = (productFilter, settings) => {
 let priceSummary = '';
 if (productFilter.priceFrom > 0 && productFilter.priceTo > 0) {
 const priceFrom = helper. formatCurrency(productFilter.priceFrom,
settings);
 const priceTo = helper.formatCurrency(productFilter.priceTo, settings);
 priceSummary = `. ${text.price}: ${priceFrom} - ${priceTo}`;} return
priceSummary;};
const CategoryHero = ({ categoryDetails, categories }) => (
 <Fragment>
 <div className="hero-body">
 {themeSettings.show_category_breadcrumbs && (
 <CategoryBreadcrumbs
 currentCategory={categoryDetails}
 categories={categories}/>)}
 <h1 className="category-title">{categoryDetails.name}</h1>
 <div
 className="category-description is-hidden-mobile content"
 dangerouslySetInnerHTML={{ __html: categoryDetails.description }}
/></div></Fragment>);
CategoryHero.propTypes = {
 categoryDetails: PropTypes.shape({}).isRequired,
 categories: PropTypes.arrayOf(PropTypes.shape({})).isRequired};
const CategoryContainer = props => {
 const {
 setSort,
 addCartItem,
 loadMoreProducts,
```

```
        getJSONLD,
        state,
        children,
        state: {
        products,
        categoryDetails,
        settings,
        productFilter,
        productsHasMore,
        categories,
        loadingProducts,
        loadingMoreProducts,
        currentPage}} = props;
        const filterAttributesSummary = getFilterAttributesSummary(productFilter);
        const filterPriceSummary = getFilterPriceSummary(productFilter, settings);
        const pageTitle =
        categoryDetails.meta_title && categoryDetails.meta_title.length > 0
        ? categoryDetails.meta_title
        : categoryDetails.name;
       const                                title                                =
`${pageTitle}${filterAttributesSummary}${filterPriceSummary}`;
        const jsonld = getJSONLD(state);
        const showFilter = themeSettings.show_product_filter;
        return (
        <Fragment>
        <MetaTags
        title={title}
        description={categoryDetails.meta_description}
        canonicalUrl={categoryDetails.url}
        imageUrl={categoryDetails.image}
```

```
ogType="product.group"
ogTitle={categoryDetails.name}
ogDescription={categoryDetails.meta_description}
jsonld={jsonld}/>
<div className="breadcrumb-wrapper">
<section className="container">
<CategoryHero categoryDetails={categoryDetails} categories={categories}
/>
<AdditionalBreadcrumbs {...props} />
</section>
</div>
<section className="section section-category">
<div className="container">
<div className="columns">
{showFilter === true && (
<div className="column is-one-quarter left-sidebar">
<ProductFilter {...props} />
</div>
<div className="column">
<div className="columns is-hidden-mobile">
<div className="column" />
<div className="column is-5">
<Sort
defaultSort={settings.default_product_sorting}
currentSort={productFilter.sort}
setSort={setSort}
/>
</div>
</div>
<ProductList
```

```
 products={products}
addCartItem={addCartItem}
settings={settings}
loadMoreProducts={loadMoreProducts}
hasMore={productsHasMore}
loadingProducts={loadingProducts}
loadingMoreProducts={loadingMoreProducts}/>
 </div>
 </div>
 </div>
 </section>
 </Fragment>);};
CategoryContainer.propTypes = {
 setSort: PropTypes.func.isRequired,
 addCartItem: PropTypes.func.isRequired,
 loadMoreProducts: PropTypes.func.isRequired,
 getJSONLD: PropTypes.func.isRequired,
state: PropTypes.shape({
 settings: PropTypes.shape({}),
 products: PropTypes.arrayOf(PropTypes.shape({})),
 productFilter: PropTypes.shape({}),
 productsHasMore: PropTypes.bool,
 categoryDetails: PropTypes.shape({}),
 categories: PropTypes.arrayOf(PropTypes.shape({})),
 loadingProducts: PropTypes.bool,
 loadingMoreProducts: PropTypes.bool
 }).isRequired};
export default CategoryContainer;
```

Додаток А.3 Фасетна навігація

```
import React from 'react';
import { NavLink } from 'react-router-dom';
import { themeSettings, text } from '../../lib/settings';
import Sort from '../sort';
import PriceSlider from './priceSlider';
import AttributeFilter from './attributeFilter';
export default class ProductFilter extends React.Component {
 constructor(props) {
 super(props);
 this.state = {
 sidebarIsActive: false};}
 sidebarToggle = () => {
 this.setState({
 sidebarIsActive: !this.state.sidebarIsActive});
 document.body.classList.toggle('noscroll');};
 sidebarClose = () => {
 this.setState({ sidebarIsActive: false });
 document.body.classList.remove('noscroll');};
 render() {
 const { sidebarIsActive } = this.state;
 const {
 categoryDetails,
 categories,
 settings,
 productFilter,
 productsMinPrice,
 productsMaxPrice,
 productsAttributes
 } = this.props.state;
```

```
return (
<div>
<div className="is-hidden-tablet">
<button className="button is-fullwidth" onClick={this.sidebarToggle}>
{text.filterProducts}
</button>
</div>
<div
className={sidebarIsActive ? 'modal is-active': 'is-hidden-mobile'}
style={{ zIndex: 101 }}>
<div
className={sidebarIsActive ? 'dark-overflow': ''}
onClick={this.sidebarClose}/>
<div className={sidebarIsActive ? 'modal-content': ''}>
<div className={sidebarIsActive ? 'box sidebar' : ''}>
<div className="is-hidden-tablet" style={{ marginBottom: 30 }}>
<Sort
defaultSort={settings.default_product_sorting}
currentSort={productFilter.sort}
setSort={this.props.setSort}/></div>
<AttributeFilter
attributes={productsAttributes}
setFilterAttribute={this.props.setFilterAttribute}
unsetFilterAttribute={this.props.unsetFilterAttribute}/>
<PriceSlider
minPrice={productsMinPrice}
maxPrice={productsMaxPrice}
minValue={productFilter.priceFrom}
maxValue={productFilter.priceTo}
setPriceFromAndTo={this.props.setPriceFromAndTo}
```

```
 settings={settings}/>
 <button
 className="button is-fullwidth is-dark is-hidden-tablet"
 onClick={this.sidebarClose}>
{text.close}
 </button>
```

Додаток А.4. Фасетна навігація

```
import React from 'react';
import { NavLink } from 'react-router-dom';
import { themeSettings, text } from '../../lib/settings';
export default class SearchBox extends React.Component {
 constructor(props) {
 super(props);
 this.state = {
 value: props.value,
 hasFocus: false};}
 handleChange = event => {
 this.setState({ value: event.target.value });};
 handleKeyPress = e => {
 if (e.keyCode === 13 || e.which === 13) {
 this.handleSearch();}};
 handleKeyDown = e => {
 if (e.keyCode === 27) {
 this.handleClear();}};
 handleSearch = () => {
 this.props.onSearch(this.state.value);};
 handleClear = () => {
 this.setState({ value: '' });
```

```
this.props.onSearch('');};
handleFocus = () => {
this.setState({ hasFocus: true });};
handleBlur = () => {
this.setState({ hasFocus: false });};
render() {
const { hasFocus } = this.state;
const placeholderText =
themeSettings.search_placeholder &&
themeSettings.search_placeholder.length > 0
? themeSettings.search_placeholder
: text.searchPlaceholder;
return (
<div
className={
'search-box ' + this.props.className + (hasFocus ? ' has-focus' : '')}>
<input
className="search-input"
type="text"
placeholder={placeholderText}
value={this.state.value}
onChange={this.handleChange}
onKeyPress={this.handleKeyPress}
onKeyDown={this.handleKeyDown}
onFocus={this.handleFocus}
onBlur={this.handleBlur}/>
<img
className="search-icon-search"
src="/assets/images/search.svg"
alt={text.search}
```

```
 title={text.search}
 onClick={this.handleSearch}/>
 {this.state.value &&
 this.state.value !== '' && (
 <img
 className="search-icon-clear"
 src="/assets/images/close.svg"
 onClick={this.handleClear}/>)}
 </div>);}}
```

Додаток А.5. Код картки товару

```
import React, { Fragment } from 'react';
import { NavLink } from 'react-router-dom';
import * as helper from '../../lib/helper';
import { themeSettings, text } from '../../lib/settings';
import ViewedProducts from '../products/viewed';
import Breadcrumbs from './breadcrumbs';
import DiscountCountdown from './discountCountdown';
import AddToCartButton from './addToCartButton';
import Attributes from './attributes';
import Gallery from './gallery';
import Options from './options';
import Price from './price';
import Quantity from './quantity';
import RelatedProducts from './relatedProducts';
import Tags from './tags';
const Description = ({ description }) => (
 <div
 className="product-content"
```

```
dangerouslySetInnerHTML={{ __html: description }}/>);
export default class ProductDetails extends React.Component {
constructor(props) {
super(props);
this.state = {
selectedOptions: {},
selectedVariant: null,
isAllOptionsSelected: false,
quantity: 1};
this.onOptionChange = this.onOptionChange.bind(this);
this.findVariantBySelectedOptions =
this.findVariantBySelectedOptions.bind(
this);
this.addToCart = this.addToCart.bind(this);
this.checkSelectedOptions = this.checkSelectedOptions.bind(this);}
onOptionChange(optionId, valueId) {
let { selectedOptions } = this.state;
if (valueId === '') {
delete selectedOptions[optionId];} else {
selectedOptions[optionId] = valueId;
}
this.setState({ selectedOptions: selectedOptions });
this.findVariantBySelectedOptions();
this.checkSelectedOptions();}
findVariantBySelectedOptions() {
const { selectedOptions } = this.state;
const { product } = this.props;
for (const variant of product.variants) {
const variantMutchSelectedOptions = variant.options.every(
variantOption =>
```

```
selectedOptions[variantOption.option_id] === variantOption.value_id);
if (variantMutchSelectedOptions) {
this.setState({ selectedVariant: variant });
return;}}
this.setState({ selectedVariant: null });}
setQuantity = quantity => {
this.setState({ quantity: quantity });};
addToCart() {
const { product, addCartItem } = this.props;
const { selectedVariant, quantity } = this.state;
let item = {
product_id: product.id,
quantity: quantity};
if (selectedVariant) {
item.variant_id = selectedVariant.id;}
addCartItem(item);}
checkSelectedOptions() {
const { selectedOptions } = this.state;
const { product } = this.props;
const allOptionsSelected =
Object.keys(selectedOptions).length === product.options.length;
this.setState({ isAllOptionsSelected: allOptionsSelected });}
render() {
const { product, settings, categories } = this.props;
const { selectedVariant, isAllOptionsSelected } = this.state;
const maxQuantity =
product.stock_status === 'discontinued'
? 0
: product.stock_backorder
? themeSettings.maxCartItemQty
```

```
 : selectedVariant
 ? selectedVariant.stock_quantity
 : product.stock_quantity;
 if (product) {
 return (
<Fragment>
 <section className="section section-product">
 <div className="container">
 <div className="columns">
 <div className="column is-7">
 {themeSettings.show_product_breadcrumbs && (
 <Breadcrumbs product={product} categories={categories} )}
 <Gallery images={product.images} />
 </div>
 <div className="column is-5">
 <div className="content">
 <Tags tags={product.tags} />
 <h1 className="title is-4 product-name">{product.name}</h1>
 <Price
 product={product}
variant={selectedVariant}
isAllOptionsSelected={isAllOptionsSelected}
settings={settings}/>
 {themeSettings.show_discount_countdown &&
 product.on_sale === true && (
 <DiscountCountdown product={product} />)}
 <Options
 options={product.options}
onChange={this.onOptionChange}/>
 <Quantity
```

```
 maxQuantity={maxQuantity}
 onChange={this.setQuantity}/>
 <div className="button-addtocart">
 <AddToCartButton
 product={product}
variant={selectedVariant}
addCartItem={this.addToCart}
 isAllOptionsSelected={isAllOptionsSelected}/>
</div>
 </div>
 </div>
 </div>
 </div>
 </section>
 <section className="section section-product-description">
 <div className="container">
 <div className="content">
<div className="columns">
 <div className="column is-7">
 <Description description={product.description} />
 </div>
<div className="column is-5">
 <Attributes attributes={product.attributes} />
 </div>
 </div>
 </div>
 </div>
 </section>
 <RelatedProducts
 settings={settings}
```

```
addCartItem={this.addToCart}
ids={product.related_product_ids}
limit={10}/>
{themeSettings.show_viewed_products && (
<ViewedProducts
settings={settings}
addCartItem={this.addToCart}
product={product}
limit={themeSettings.limit_viewed_products || 4}/>)}
</Fragment>);} else {return null;}}}
```

Додаток А.6. Лістинг коду «Шаблон повідомлень» з бази даних

```
const addOrderConfirmationEmailTemplates = async db => {
 const emailTemplatesCount = await db
 .collection('emailTemplates')
 .countDocuments({ name: 'order_confirmation' });
 const emailTemplatesNotExists = emailTemplatesCount === 0;
 if (emailTemplatesNotExists) {
 await db.collection('emailTemplates').insertOne({
 name: 'order_confirmation',
 subject: 'Підтвердити замовлення',
 body: `<div>
 <div><b>Номер замовлення</b>: {{number}}</div>
 <div><b>Метод доставки</b>: {{shipping_method}}</div>
 <div><b>Спосіб оплати</b>: {{payment_method}}</div>
 <div style="width: 100%; margin-top: 20px;">
 Адресна доставка<br /><br />
 <b>Прізвище Ім’я По-Батькові</b>: {{shipping_address.full_name}}<br
/>
```

```
<b>Адреса 1</b>: {{shipping_address.address1}}<br />
<b>Адреса 2</b>: {{shipping_address.address2}}<br />
<b>Місто</b>: {{shipping_address.city}}<br />
<b>Номер телефону</b>: {{shipping_address.phone}}
</div>
<table style="width: 100%; margin-top: 20px;">
<tr>
<td style="width: 40%; padding: 10px 0px; border-top: 1px solid
#ccc; border-bottom: 1px solid #ccc; text-align: left;">Товар</td>
<td style="width: 25%; padding: 10px 0px; border-top: 1px solid
#ccc; border-bottom: 1px solid #ccc; text-align: right;">Ціна</td>
<td style="width: 10%; padding: 10px 0px; border-top: 1px solid
#ccc; border-bottom: 1px solid #ccc; text-align: right;">Кількість</td>
<td style="width: 25%; padding: 10px 0px; border-top: 1px solid
#ccc; border-bottom: 1px solid #ccc; text-align: right;">Загальна</td>
</tr>
{{#each items}}
<tr>
<td style="padding: 10px 0px; border-bottom: 1px solid #ccc;
text-align: left;">{{name}}<br />{{variant_name}}</td>
<td style="padding: 10px 0px; border-bottom: 1px solid #ccc;
text-align: right;">$ {{price}}</td>
<td style="padding: 10px 0px; border-bottom: 1px solid #ccc;
text-align: right;">{{quantity}}</td>
<td style="padding: 10px 0px; border-bottom: 1px solid #ccc; textalign:
right;">$ {{price_total}}</td>
</tr>
{{/each}}
</table>
<table style="width: 100%; margin: 20px 0;">
```

```
 <tr>
 <td style="width: 80%; padding: 10px 0px; text-align:
right;"><b>Вартість доставкиb></td>
 <td style="width: 20%; padding: 10px 0px; text-align: right;">$
{{shipping_total}}</td>
 </tr>
 <tr>
 <td style="width: 80%; padding: 10px 0px; text-align:
right;"><b>Загальна вартість</b></td>
 <td style="width: 20%; padding: 10px 0px; text-align: right;">$
{{grand_total}}</td>
 </tr>
 </table>
 </div>`});
 winston.info('- Шаблон додано у базу даних');}};
```

Додаток А.7. Підключення до бази даних

```
import winston from 'winston';
import url from 'url';
import { MongoClient } from 'mongodb';
import logger from './lib/logger';
import settings from './lib/settings';
const mongodbConnection = settings.mongodbServerUrl;
const mongoPathName = url.parse(mongodbConnection).pathname;
const dbName =
mongoPathName.substring(mongoPathName.lastIndexOf('/') + 1);
const CONNECT_OPTIONS = {
 useNewUrlParser: true};
(async () => {
```

```
let client = null;
let db = null;
try {
client = await MongoClient.connect(
mongodbConnection,
CONNECT_OPTIONS);
db = client.db(dbName);
winston.info(`Successfully connected to ${mongodbConnection}`);
} catch (e) {
winston.error(`MongoDB connection was failed. ${e.message}`);
return;}
const userEmail = process.argv.length > 2 ? process.argv[2] : null;
const domain = process.argv.length > 3 ? process.argv[3] : null;
await db.createCollection('customers');
await db.createCollection('orders');
await addAllPages(db);
await addAllProducts(db);
await addOrderConfirmationEmailTemplates(db);
await addForgotPasswordEmailTemplates_en(db);
await addForgotPasswordEmailTemplates_ru(db);
await addRegisterDoiEmailTemplates_en(db);
await addRegisterDoiEmailTemplates_ru(db);
await addShippingMethods(db);
await addPaymentMethods(db);
await createAllIndexes(db);
await addUser(db, userEmail);
await addSettings(db, {domain});
client.close();})();
```

Додаток А.8. Налаштування підключення до бази даних

```
const dbHost = process.env.DB_HOST || '127.0.0.1';
const dbPort = process.env.DB_PORT || 27017;
const dbName = process.env.DB_NAME || 'shop';
const dbUser = process.env.DB_USER || '';
const dbPass = process.env.DB_PASS || '';
const dbCred = dbUser.length > 0 || dbPass.length > 0? `${dbUser}:${dbPass}@`
: '';
const dbUrl = process.env.DB_URL ||
`mongodb://${dbCred}${dbHost}:${dbPort}/${dbName}`;
module.exports = {
 storeBaseUrl: process.env.STORE_BASE_URL || 'http://forwater.ua',
 adminLoginPath: process.env.ADMIN_LOGIN_PATH || '/login',
 mongodbServerUrl: dbUrl,
 smtpServer: {
 host: process.env.SMTP_HOST || '',
 port: process.env.SMTP_PORT || 587,
 secure: process.env.SMTP_SECURE || false,
 user: process.env.SMTP_USER || '',
 pass: process.env.SMTP_PASS || '',
 fromName: process.env.SMTP_FROM_NAME || '',
 fromAddress: process.env.SMTP_FROM_ADDRESS || ''},
 categoriesUploadUrl: '/images/categories',
 productsUploadUrl: '/images/products',
 themeAssetsUploadUrl: '/assets/images',
 language: process.env.LANGUAGE || 'ru',};
```