Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Модель електронної торгівлі в сфері B2C
із застосуванням веб-технологій

(тема)

Виконав:

Студент II курсу, групи СПм-21-1

Вичевський В.В.
(прізвище, ініціали)

Спеціальність

123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доцент Голубничий.Д.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Вичевському Віктору Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Модель електронної торгівлі в сфері B2C із застосуванням веб-технологій

затверджена наказом по університету від " 07 " листопада 2022 р. № 1454 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 13 грудня 2022 р.

3. Вхідні дані до роботи

1) програмне забезпечення;
2) веб-сайт застосунку інтернет магазину;
3) перелік використаних програмних засобів: ОС Windows;
4) CRM-система.

4. Перелік питань, що потрібно опрацювати у роботі

1) аналіз предметної області;
2) вибір топології розробки;
3) теоретичні відомості про мову програмування Javascript;
4) теоретичні відомості про створення веб-сторінок для готельно-ресторанного комплексу, їх особливості та застосування
5) програмна реалізація веб-сторінки та налагодження CRM системи на сайті для зручності роботи з клієнтами.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)

Слайд-презентація – 16 слайдів.

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	18.09.22 - 21.09.22	
2	Аналіз та вибір методів рішення задачі	22.09.22 - 29.09.22	
3	Розробка алгоритмів та вибір засобів рішення	30.09.22 - 20.10.22	
4	Проектування та розробка програмного	21.10.22 - 27.11.22	
	забезпечення		
5	Тестування програмного забезпечення та	28.11.22 - 02.12.22	
	усунення проблем, які виникли у ході		
	проектування		
6	Оформлення матеріалів кваліфікаційної	03.12.22 - 12.12.22	
	роботи		
7	Подання кваліфікаційної роботи на	12.12.22	
	рецензування		

Дата видачі завдання 07 листопада 2022 р.

Студент _____
(підпис)

Керівник роботи _____ к.т.н., Голубничий Д.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 81 стор., 18 рис., 1 схема., 7 лістингів., 11 джерел.

ДОСТУП, ІНТЕРНЕТ, ТИП, РОЗРОБКА, ВЕБ-СТОРІНКА, ПРОЕКТУВАННЯ, ГОТЕЛЬ, МЕТОДИ, ФУНКЦІЇ, JAVASCRIPT.

Метою кваліфікаційної роботи є розробка моделі електронної торгівлі в сфері B2C з застосуванням веб-технологій, що включає в себе веб-сайт інтернет магазину, CRM-систему та систему адміністрування програмного модуля.

У ході виконання кваліфікаційної роботи було розглянуто сучасні технології і середовища програмування проаналізовано складові елементи та етапи створення веб-сторінки, досліджено різновиди CRM-систем, та процедури роботи з ними.

У результаті виконання кваліфікаційної розроблено веб-сторінку готельно-ресторанного комплексу на мові Javascript та мові розмітки гіпертексту (HTML, CSS). Створено систему адміністрування програмного модуля із застосуванням спеціальних засобів IT-менеджменту (CRM система Bitrix24).

ABSTRACT

Explanatory note of the qualification work: 81 pages, 18 figures, 1 diagram, 7 listings, 11 sources.

ACCESS, INTERNET, TYPE, DEVELOPMENT, WEB PAGE, DESIGN, HOTEL, METHODS, FUNCTIONS, JAVASCRIPT.

The object of the study is special software for the hotel and restaurant complex.

The purpose of this thesis is to create and implement software to improve the management of the hotel and restaurant business, namely the equipment of special latest technical IT tools and software product development.

During the qualification work, a workable system for managing the hotel and restaurant business in Javascript and hypertext markup language (HTML, CSS) was obtained, and special IT management tools (Bitrix24 CRM system) were implemented.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення

ПК – персональний комп’ютер

AJAX – (англ., Asynchronous Javascript and XML) підхід до побудови інтерактивних користувацьких інтерфейсів веб-застосунків, суть якого полягає в «фоновому» обміні даними браузера з веб-сервером.

CRM – (англ., Customer Relationship Management), сервіс, який записує та зберігає інформацію про клієнтів, заявках, замовленнях та угодами з ними.

CSS – каскадні таблиці стилів (англ., Cascading Style Sheets), спеціальна мова, створена для формування зовнішнього вигляду документу, створеного за допомогою мов розмітки (HTML або XHTML).

ECMA – (англ., European Computer Manufacturers Association) європейська організація по виготовленню комп’ютерів.

HTML – мова гіпертекстової розмітки (англ., HyperText Markup Language)

HTTP – протокол передачі гіпертексту (англ., HyperText Transfer Protocol), протокол прикладного рівня передачі даних, початково – в виді гіпертекстових документів в форматі HTML.

Redux, React – бібліотеки мови Javascript

SaaS –Програмне забезпечення як послуга

API – Програмний інтерфейс застосунку

UX – Досвід користувача

MPA – Багатосторінковий застосунок

SEO – Пошукова оптимізація

DOM – Об’єктна модель документу

ORM – Об’єктно-реляційне відображення

ВСТУП

Електронна комерція - це сфера бізнесу, без якої більшість сучасних людей не уявляють свого життя. Всі вже давно звикли купувати товари в Інтернеті, не виходячи з дому. Зручність та економія часу - одні з головних переваг інтернет-магазинів перед традиційними фізичними магазинами.

Оскільки електронна комерція нині перебуває на піку свого розвитку популярність, кількість конкурентів у цій галузі дуже велика. Продавці повинні забезпечити своєчасну, якісну та швидку доставку оновлювати інформацію про наявність товарів, налагоджувати процес обміну та повернення товару. Однак головне завдання продавців – забезпечити покупцям зручний, простий у використанні та інтуїтивно зрозумілий інтерфейс інтернет-магазину.

Замовлення розміщуються за допомогою спеціальної сторінки, зовнішній вигляд і зручність якої залежать від того, чи буде клієнт завершувати процес покупки. Тому ця сторінка має бути швидкою, зручною і простою у використанні, усі ці критерії часто не виконуються в інтернет-магазинах.

Тобто метою дипломної роботи є розробка системи, яка дозволяла б покупцю швидше і ефективніше оформлювати замовлення в інтернет магазинах. Дана система зможе підвищити рівень заробітку продавця, оскільки його клієнти з більшою імовірністю завершуватимуть процес замовлення.

У першому розділі даної роботи описано загальні відомості про електронну комерцію та інтернет-магазини, а також надана інформація про платформи, які дозволяють створювати інтернет-магазини.

Другий розділ містить у собі опис технологій та інструментів, які були використані при розробці системи.

У третьому розділі міститься інформація про структуру системи, організацію збереження даних в ній та архітектуру інтернет магазину в цілому.

Четвертий розділ містить в собі безпосередньо тестування програми, яким чином відбувається наповнення контентом, створення дизайну веб-сторінки та опис інтерфейсу користувача для застосунку та CRM-системи.

1 ПРОГРАМНІ СИСТЕМИ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

1.1 Електронна комерція та інтернет-магазини

За час пандемії Коронавірусу останніх років важливого значення та значного розвитку набула особлива сфера бізнесу — електронна комерція. Багато компаній змінили сферу своєї діяльності і досягли значного успіху саме за допомогою інтернет-магазинів. Електронна комерція — це процес продажу фізичних або віртуальних речей, послуг тощо за допомогою Інтернету. За допомогою електронної комерції користувачам в онлайн-режимі стали доступні будь-які книжки, квитки в кіно, музика, одяг, інвестування і навіть їжа. Електронна комерція містить у собі чотири підвиди, які описують процес взаємодії покупця і продавця. Перший з них — це “бізнес для бізнесу” (B2B), тобто і продавцем, і покупцем виступають два підприємства, які обмінюються послугами і товарами між собою. Наступний — “бізнес для споживача” (B2C), даний вид означає, що продаж товарів відбувається між підприємствами та їхніми клієнтами. Третім видом є “споживач для споживача” (C2C), саме за допомогою такого виду побудована популярна онлайн-платформа для розміщення оголошень OLX, на якій користувачі можуть продавати товари одне одному. Останній вид це “споживач для бізнесу” (C2B), який дозволяє приватним особам продавати власні товари для підприємств. Інтернет-продажі працюють абсолютно таким же чином, як і фізичні магазини. Клієнтам необхідно зайти у інтернет-магазин підприємства, обрати необхідні товари і здійснити онлайн-покупку. Проте це набагато зручніше, оскільки не потрібно нікуди їхати, стояти в довгих чергах і витрачати дорогоцінний час. А клієнтська база магазину перестає бути обмеженою одним регіоном або країною, оскільки люди з усієї планети можуть здійснювати покупки в інтернет-магазині. Інтернет-магазин — один з головних інструментів електронної комерції, у даному випадку назва терміну говорить сама за себе, інтернет-магазин — це магазин певних послуг або товарів, який розміщений у глобальній мережі. За

легким процесом покупки з точки зору користувача стоїть доволі складний і комплексний процес з точки зору продавця, проте він повністю себе виправдує, коли є чітко налагоджена послідовність дій і якісно розроблений інтернет-магазин. Першим етапом покупки є розміщення замовлення, даним етапом займається покупець взаємодіючи з інтернет-магазином. Сюди обов'язково входить процес вибору продуктів, заповнення всієї необхідної інформації для доставки та, за необхідності, онлайн-оплата замовлення. Після цього починається наступний етап — оброблення транзакції та зняття грошей з банківської картки клієнта, створюється замовлення, яке можуть обробити менеджери, або сам продавець. Фінальна дія — доставка замовлення, яка здійснюється за адресою, яку вказав клієнт на першому етапі, зазвичай доставкою інтернет-замовлень займаються сторонні компанії, які напряму взаємодіють з продавцем. Основною перевагою інтернет-магазинів для продавців є те, що вони майже повністю усувають необхідність підтримувати і використовувати фізичні магазини. Навіщо винаймати кожного місяця велике приміщення, якщо можна просто один раз створити інтернет-магазин, а потім тільки інколи займатись його адмініструванням? Електронний магазин може працювати цілодобово і йому не потрібні касири або консультанти, немає ніякого обмеження кількості полиць, тому можна розміщати стільки товарів, скільки того вимагатиме бізнес. Розширювати інтернет-магазин і його асортимент значно легше, ніж фізичний, оскільки не потрібно думати про кількість працівників, розміри магазину тощо, єдина проблема, яка може виникнути — це логістика і доставка товарів клієнтам, проте, якщо правильно побудувати процес, то і це не стане критичним фактором. Щодо підтримки клієнтів, то тут також виграють інтернет-магазини, оскільки онлайн-підтримка може цілодобово відповідати на усі електронні листи клієнтів. Рекламу і повідомлення про нову продукцію або акції можна також з легкістю автоматизувати, полегшивши процес маркетингу, відправка електронних листів на пошту клієнтів здійснюється моментально і в необмеженій кількості. Оскільки електронна-комерція має багато переваг як для продавця, так і для

клієнта, то у сучасному світі з’явилося багато вимог, яких мають дотримуватись сучасні інтернет-магазини, щоб мати високу конверсію. Поперше, це інтуїтивний, зручний та привабливий інтерфейс користувача, оскільки навряд хтось захоче купувати товари в магазині із сумнівним зовнішнім виглядом і складною структурою. Другий важливий критерій — це мобільна адаптація, велика частина усіх інтернет-замовлень робиться саме з мобільних пристроїв. йГнучка і зручна політика повернення коштів за товар — важлива вимога, на яку звертають увагу більшість клієнтів, оскільки доволі часто бувають ситуації, коли товар не підходить за характеристиками і необхідно здійснити повернення. Ну і основна причина, через яку великий відсоток користувачів не завершують процес оформлення замовлення — занадто складна і незрозуміла сторінка оформлення замовлень. Частково саме цю проблему буде вирішувати система, розробка і тестування якої будуть описані у третьому та четвертому розділах даної роботи.

1.2 Платформи для створення інтернет-магазинів

Розробка інтернет-магазинів справа не дуже легка і швидка, тому кожного року розробники намагаються вдосконалювати технології та інструменти, пришвидшуючи цей процес. Зазвичай окремий продавець або підприємство хочуть створити унікальний продукт із набором функцій, які будуть повністю забезпечувати користувача необхідної функціональністю. Розробка комплексного інтернет-магазину може займати від декількох місяців до декількох років, швидкість розробки залежить від команди, організації робочого процесу, технологій, що використовуються у процесі розробки та багатьох інших факторів. Проте що робити, якщо у продавця немає стільки часу? Що, якщо йому потрібно у максимально короткі терміни створити працюючий інтернет-магазин, який із самого початку його створення буде приносити дохід? У таких випадках на допомогу приходять спеціальні платформи, які дозволяють створювати і налаштовувати інтернет-магазини

всього у декілька кроків людям, які зовсім не тямлять у програмуванні.

1.2.1 Shopify

Shopify — платформа електронної комерції, за допомогою якої можна створювати власні інтернет-магазини або використовувати її у якості програми для оплати товарів у фізичних магазинах. Shopify працює як комплексний SaaS веб-застосунок і надає послуги за щомісячну підписку. Дана платформа є хмарним рішенням, тому продавцеві не потрібно хвилюватись про оновлення або підтримку серверів, йому достатньо створити магазин і підтримувати актуальність продуктів. За допомогою тем в Shopify можна налаштовувати зовнішній вигляд інтернет-магазину, що додає гнучкості та унікальності кожному веб-сайту, тем в спеціальному вбудованому магазині дуже багато, тому обрати можна під будь-які вимоги. Можна налаштовувати спливаючі вікна, керувати варіантами доставки, застосовувати необхідні податки, додавати і редагувати продукти та їхню наявність на складі та багато іншого.

Оплата і безпека даних користувачів на високому рівні, оскільки Shopify Payments підтримує 3-D Secure, за допомогою цієї вбудованої технології можна не налаштовувати сторонні платіжні шлюзи, оскільки вона підтримує всі основні способи оплати. Проте, якщо необхідно більше контролю над процесом оплати, то краще використовувати сторонні платіжні шлюзи, які можна легко інтегрувати в магазин. Shopify App Store дозволяє купувати і встановлювати розширення різних видів, які можуть модифікувати існуючий функціонал, або додавати новий.

Проблеми у Shopify є, проте порівняно із перевагами вони зовсім незначні і більшість підприємств не звертають на них уваги. Одна з таких — складнощі з ручним редагування коду інтернет-магазину, Shopify надає можливість редагувати код теми, проте недосвідченому користувачу важко розібратись з файлами і налаштуваннями, що там присутні. Shopify App Store хоч і має широкий вибір додаткових розширень, проте майже всі вони платні, особливо,

якщо це комплексний застосунок, який додає нову функціональність у магазин.

1.2.2 Wix

Якщо у вас є чудова ідея про продукт для продажу в Інтернеті, або якщо ваш поточний онлайн-магазин може використати сильнішу присутність, ви, найімовірніше, перебуваєте на ринку для нової платформи електронної комерції. На щастя для вас, існують якісні варіанти, даючи вам змогу попрацювати з кожним і вирішити, який із них підходить саме вам.

Shopify є явним лідером за популярністю, але Wix також має тенденцію підніматися в розмові, оскільки понад 68 мільйонів користувачів використовують Wix. Це не означає, що всі Wix користувачі продають товари в Інтернеті, але варто зазначити, що WIX справді забезпечує інструменти для електронної комерції, які конкурують із Shopify. Shopify - спеціалізований важковаговик у світі електронної комерції, покликаний надати підприємствам все, що їм потрібно для продажу в Інтернеті. Понад 600,000 XNUMX підприємств використовують Shopify глобально. З іншого боку, Wix все про простоту перетягування і легке створення інтернет-магазинів та сайтів для новачків. Якщо ви шукаєте конструктор інтернет-магазину, який допоможе вам швидко почати роботу, тоді Wix, ймовірно, буде найкращим варіантом для вас. З іншого боку, Shopify може впоратися з набагато більшими продажами, маючи доступ до всього: від інтеграції з соціальними мережами до dropshipping. Хоча обидва Shopify і Wix мають різні функції, які допоможуть вам почати роботу в прекрасному світі створення інтернет-магазину, Wix з більшою ймовірністю буде найкращим вибором, якщо ви новачок, тоді як Shopify допоможе вам розширити.

Wix варто обирати, якщо немає необхідності у встановленні якогось незвичайного функціоналу, бо всі функції відразу будуть реалізовані і не потребуватимуть додаткового встановлення. Shopify потрібен для більшого контролю магазину, оскільки він має величезний Shopify App Store, який надає

додаткові функції та можливості. Wix потрібно обирати виключно для онлайн бізнесу, в той час, як Shopify частково можна використовувати і з фізичними магазинами. В усьому іншому вибір залишається за продавцем, а щоб досягти успіхів в електронній комерції важливо не тільки правильно обрати платформу, а й правильно налаштувати магазин і процес оформлення замовлень.

1.3 Оплата замовлень та платіжні шлюзи

Однією з переваг інтернет-магазинів є можливість оплатити замовлення онлайн, що дозволяє при отриманні товару не використовувати фізичні гроші або банківську картку. Великий відсоток покупців активно використовують цю функцію, тому сучасні магазини обов’язково повинні використовувати платіжні шлюзи для проведення транзакцій.

Платіжний шлюз — технологія, яка передає дані онлайн-платежу від клієнта до продавця, а потім повідомляє або про відхилення операції, або про її успішність. Платіжний шлюз перевіряє і обробляє дані банківської карти, яка була введена, а також забезпечує доступ до необхідної суми коштів. Передача даних картки здійснюється у зашифрованому вигляді, тому безпека покупця гарантована.

Про платіжний шлюз можна думати як про посередника між продавцем та його клієнтом, який займається обробкою транзакцій. За допомогою платіжного шлюзу продавцям не потрібно напряму взаємодіяти з банківськими системами, а користувачі інтернет-магазинів можуть взаємодіяти із зручним вбудованим інтерфейсом платіжного шлюзу, якщо такий є.

Весь процес роботи платіжного шлюзу можна описати декількома етапами. Спочатку клієнт обирає необхідні продукти, заповнює всю необхідну інформацію для оформлення замовлення і переходить на сторінку оплати. В залежності від виду інтеграції інтернет-магазину з платіжним шлюзом, ця сторінка може бути відразу готовою, і необхідно лише перенаправити клієнта на неї, або це може бути безпосередньо сторінка магазину, якщо інтеграція

відбувається лише на сервері продавця. Після введення даних банківської карти вони безпечно передаються в платіжний шлюз на основі тієї інтеграції, яка є в магазині. Платіжний шлюз спочатку перевіряє дані картки на шахрайство, потім шифрує їх та відправляє до банку-еквайру. Після цього банк-еквайр надсилає інформацію до потрібного типу платіжної системи, наприклад Visa або Mastercard, де вони проходять ще один етап перевірки на шахрайство і відправляються до банка-емітента. В останньому відбувається схвалення або відхилення платежу і інформація про це надсилається спочатку до платіжної системи, а потім до банку-еквайру. Відповідно банк-еквайр передає інформацію про схвалення платежу до платіжного шлюзу і повідомлення про це можна відобразити покупцю. Після схвалення платежу банк-еквайр отримує потрібну суму платежу і зберігає на своїх рахунках, потім ця сума переходить безпосередньо до продавця.

1.3.1 Payline

Payline був у бізнесі платіжного шлюзу деякий час. Тепер вони пропонують чіткі збори та вигідуі модель ціноутворення плюс обмін. Якщо ви збираєтеся приймати в основному кредитні картки, це може бути найкращим рішенням для вас.

У цій моделі з вас списується плата за транзакції, пов'язані з кожною карткою, плюс комісія оператора. Це, як то кажуть, вам знадобиться окремий торговий рахунок для роботи з Payline, що робить процес встановлення складнішим і, можливо, не таким доброзичливим, якщо ви тільки починаєте створення нового інтернет-магазину.

Лінія виплат більше практичного платіжного шлюзу. Це означає, що вам потрібно налаштувати такі елементи, як регулярні платежі або інші нестандартні схеми платежів. Іншими словами, реалізація конкретного рішення може бути складнішою і, отже, більш придатною для усталених підприємств.

Щодо цін, через модель ціноутворення interchange-plus з вашої кредитної картки стягується вартість, яку ви обробляєте + 0.3% від суми транзакції (ви можете узгодити суму комісії до 0.2%, якщо у вас досить великий обсяг).

Є також окрема плата на місяць, щоб ваш обліковий запис було включено.

1.3.2 Authrorize.net

Authrorize.net — платіжний шлюз від Visa, на сьогоднішній день від надає невеликим підприємствам та бізнесам усі необхідні умови для обробки платежів в інтернет-магазинах. Основна орієнтація даного платіжного шлюзу йде на невеликі магазини, тому налаштування і використання Authrorize.net є дуже простим і зручним.

Одна з переваг — прозоре ціноутворення, якщо у продавця є обліковий запис, то йому не потрібно хвилюватися за комісії, або мінливу ціну за місяць, оскільки щомісячна плата є фіксованою. Підтримка користувачів на високому рівні і здійснюється цілодобово, тому у разі виникнення будь-яких проблем з оплатами, вони будуть швидко вирішені. Інтеграцію з даним платіжним шлюзом можна досить швидко і легко реалізувати, що є беззаперечною перевагою для розробників.

При виборі Authrorize.net варто враховувати, що у нього доволі обмежені міжнародні можливості, він підтримує валюту США та Канади, а у Європі всього 8 валют, що звісно не дуже добре для продавців, які хотіли б розширювати міжнародну базу клієнтів. Кожен з описаних платіжних шлюзів орієнтований на різні потреби бізнесу, проте і Payline, і Authrorize.net виконують поставлені завдання та роблять сучасні інтернет-магазини більш зручними та безпечними.

1.3.3 CRM система

Визначення CRM розшифровується як Customer Relationship Management, що означає "управління взаємовідносинами з клієнтами" і відноситься до всіх стратегій, методів, інструментів і технологій, які використовує бізнес для розвитку, утримання та залучення клієнтів.

Основна мета впровадження CRM-стратегії - створення єдиної екосистеми із залучення нових і розвитку наявних клієнтів.

Основна перевага CRM-системи в тому, що вона може бути корисною практично будь-якому організаційному підрозділу - від продажів і обслуговування клієнтів до рекрутингу, маркетингу та розвитку бізнесу. Зберігання всієї інформації про клієнтів в одному місці, реєстрація проблем з обслуговуванням, визначення можливостей продажів, управління маркетинговими кампаніями - це всього лише кілька можливостей, які надає CRM.

Оскільки CRM забезпечує швидкий доступ до даних, користувачам стає набагато простіше співпрацювати між собою - як наслідок, вирішуються питання внутрішньокомандної взаємодії та підвищується продуктивність.

1.4 Аналіз наявних систем оформлення замовлень

Більшість систем оформлення замовлень, які схожі на ту, що розробляється в даній роботі, в Bitrix24 або є приватними, або платними. Тому неможливо в повній мірі оцінити недоліки або переваги та проаналізувати аналогічні системи. Але оскільки система, яка буде розроблена, має застосовуватись для того, щоб покращити конверсію в інтернет-магазині порівняно зі стандартною сторінкою оформлення замовлень, то є сенс проаналізувати цю сторінку та зрозуміти, що потрібно покращити та які вимоги до розроблюваної системи. Перша і основна вимога до системи оформлення замовлень — вона має коректно і повністю реалізувати процес оформлення

замовлення для користувача, який має містити в собі етап заповнення інформації про клієнта, вибір варіанту доставки та етап оплати замовлення, також може містити інші пункти, це все можна налаштувати при створенні форми в самій CRM та імпорт цієї форми на сайт інтернет магазину. На рисунку 1.1 наведено приклад стандартної сторінки в Bitrix24, де зберігаються замовлення.

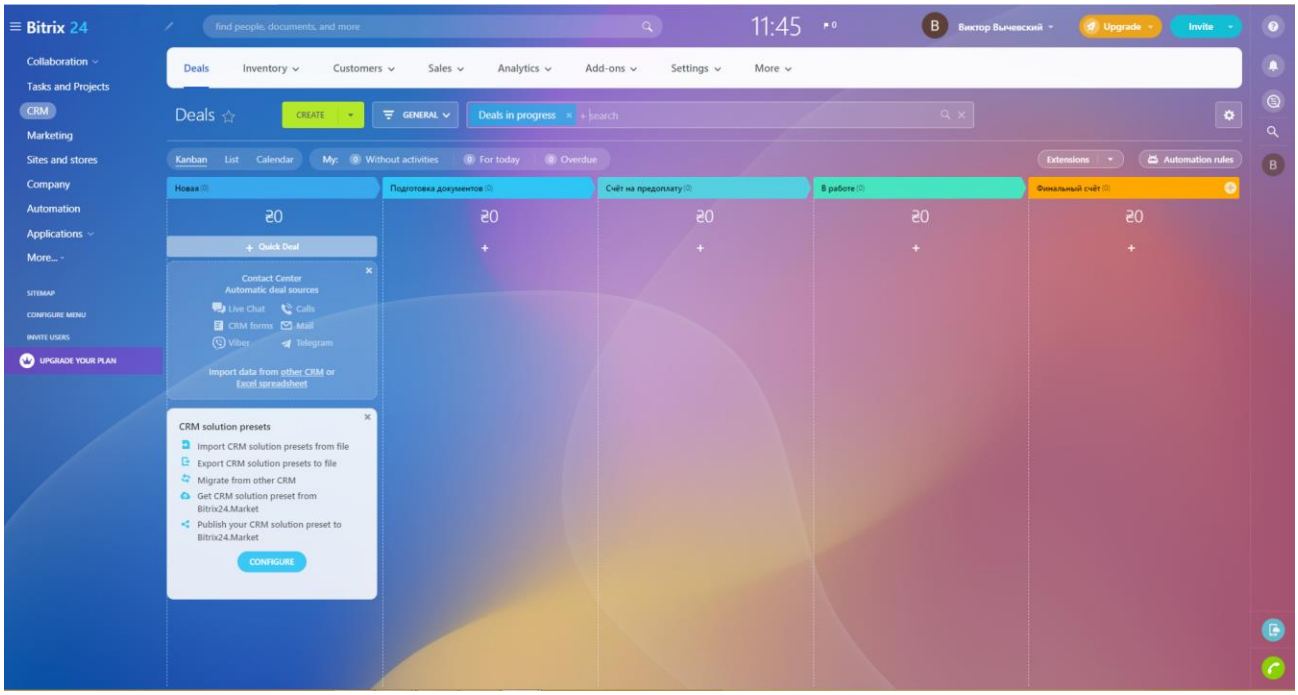

Рисунок 1.1 – Стандартна сторінка для зберігання замовлень в Bitrix24

Також в даній системі представлена воронка продажу, яка допомагає розділити етап продажу товарів на декілька частин, це допомагає адміністратору вести звіт по клієнтам магазину та розуміти, які з них є холодні, теплі та гарячі.

CRM система дає змогу створювати лендінги як всередині платформи, так і інтегрувати форми різного типу на інші сайти, не створені всередині самої системи. Таким чином CRM дає змогу створити форму для замовлення та інтегрувати її в інтернет магазин, приклад створення форми наведено на рисунку 1.2.

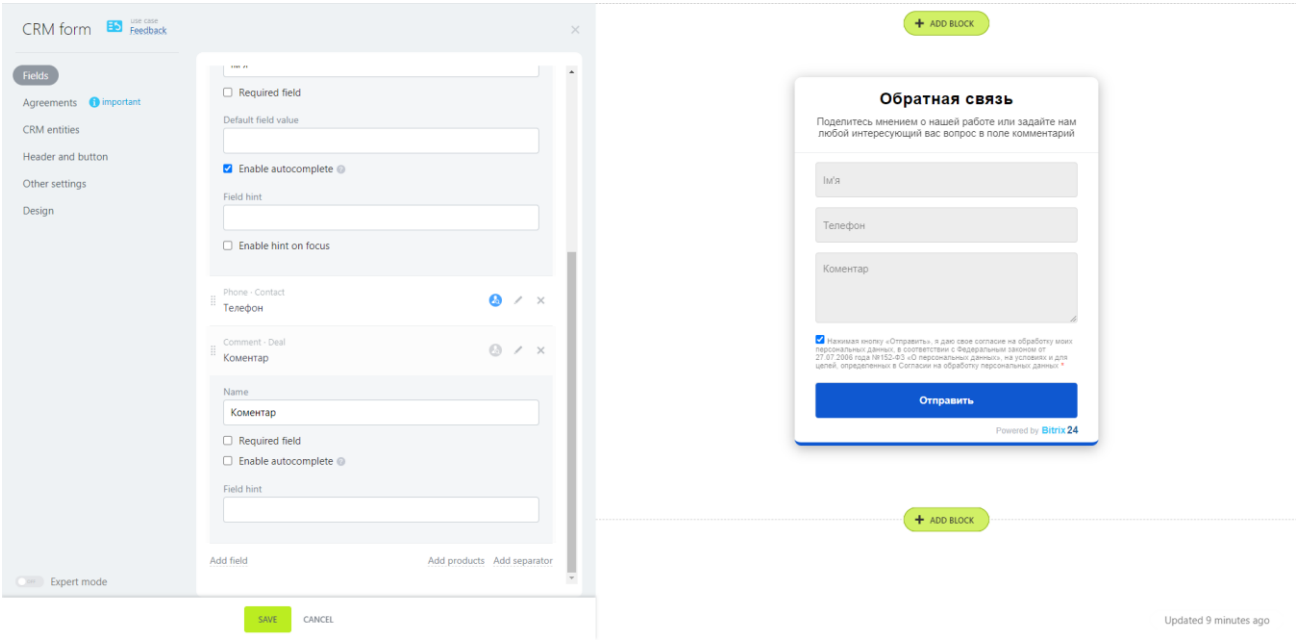

Рисунок 1.2 – Інструмент для створення форми для замовлення

Після створення форми, Bitrix24 дозволяє доступ до коду форми, який інтегрується в інтернет магазин. Після заповнення форми в магазині, в CRM приходить замовлення, в якому описана інформація про замовлений товар та інформація про клієнта, як показано на рисунку 1.3.

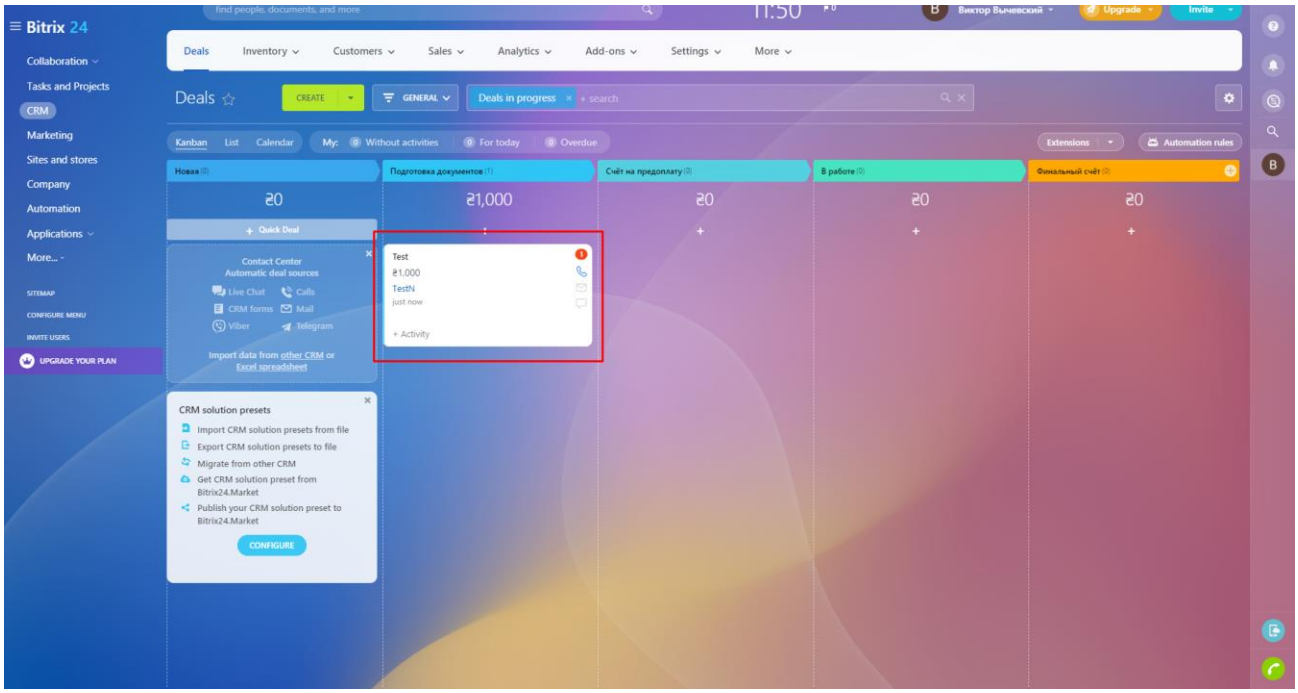

Рисунок 1.3 – Приклад роботи форми для замовлення

Можливість перегляду оформлених замовлень дуже зручна і важлива функція, яка обов’язково має бути збережена при створенні системи. В замовлені можна переглянути інформацію про клієнта, суму, а також інформацію про товар та клієнта. Дана система доволі проста і в цей же час дуже зручна, як для клієнтів, так і для адміністратора інтернет-магазину.

2 АНАЛІЗ АКТУАЛЬНИХ ЗАСОБІВ РОЗРОБКИ

2.1 Типи веб-застосунків

2.1.1 MPA

MPA — це веб-застосунок або веб-сайт, який має більше, ніж одну сторінку. Він працює традиційним чином, веб-сторінка повністю перезавантажується, коли користувач взаємодіє з нею.

Багатосторінкові веб-застосунки дуже часто мають велику і складну архітектуру. Зазвичай сторінки MPA містять в собі посилання на інші внутрішні сторінки, тому інтерфейс користувача також виходить доволі складний та багаторівневий.

Якщо говорити про переваги MPA, то тут однозначно варто виділити легке SEO-просування для таких веб-застосунків. Оскільки існують нові сторінки для кожного запиту і в них уже підготована вся інформація, яка буде відображатися користувачу, то спеціальні сканери з легкістю зможуть зчитати цю інформацію і видавати цей веб-застосунок для потрібних користувачів у пошукових системах. Другою перевагою є те, що MPA можна достатньо довго і успішно масштабувати, оскільки збільшуватиметься кількість сторінок, але складність коду майже не змінюватиметься. Оскільки багатосторінкові вебзастосунки одні з перших набули популярності серед розробників, то відповідно існує дуже багато готових рішень, фреймворків і бібліотек, які навіть у сучасному світі дозволяють створювати конкурентоспроможні проекти.

Головним недоліком MPA є швидкість їхньої роботи. Оскільки постійно необхідно завантажувати нові сторінки, то потрібно кожного разу чекати відповіді від сервера. Якщо сервер буде повільно працювати, то довгі завантаження будуть відразу помітні для користувача. Ще одним вагомим недоліком є час розробки таких веб-застосунків, оскільки необхідно, щоб

сервер тісно взаємодіяв із веб-сторінками, то розробникам потрібно частіше синхронізуватись і витрачати багато часу на обговорення тих чи інших рішень.

2.1.2 PWA

PWA — це особливий клас веб-застосунків, які дозволяють користувачеві взаємодіяти з сайтом таким же чином, як і з мобільним застосунком. Такий тип веб-застосунків можна розглядати, як застосунок зі своїми сповіщеннями, навігацією та іншими функціями, які доступні для мобільних застосунків. Ще однією особливістю PWA є те, що їх можна встановлювати собі на телефон і користуватися навіть без доступу до Інтернету.

PWA абсолютно нічим не відрізняються від звичайних мобільних застосунків, тому плавні анімації і красивий інтерфейс створюють позитивні враження для користувача. Ще однією перевагою таких веб-застосунків є те, що вони набагато дешевші у розробці, ніж мобільні аналоги, а також не потребують окремої розробки під різні операційні системи. Важливим для користувачів є і розмір застосунку, у PWA з цим все набагато краще, ніж у мобільних застосунків, вони мають менший розмір і не потребують постійних оновлень, користувачі завжди можуть бачити актуальну інформацію.

Важливою проблемою для користувачів є те, що PWA інтенсивніше використовують акумулятор телефону. Хоч і не потрібно розробляти різні застосунки для різних операційних систем, але, наприклад, в деяких мобільних операційних системах існують обмеження для таких веб-застосунків, тому не завжди вийде реалізувати увесь бажаний функціонал.

2.1.3 SPA

SPA — це тип веб-застосунку, який використовує одну веб-сторінку, як оболонку для всіх інших веб-сторінок. Такий веб-застосунок має динамічне оновлення інформації на сторінках, тому незалежно від того, як користувач

буде взаємодіяти із веб-застосунком, перезавантаження сторінки не буде відбуватись. На практиці це означає, що коли користувач буде переходити на інші сторінки, натискати кнопки тощо — всі елементи будуть динамічно оновлюватись без потреби перезавантаження сторінки.

Найочевиднішою перевагою SPA є швидкість роботи, оскільки не потрібно кожного разу завантажувати нову сторінку, зменшується час очікування для користувачів. Швидкість завантаження веб-застосунку є дуже важливим критерієм, оскільки у більшості випадків користувач не захоче очікувати декілька секунд, поки завантажиться нова інформація. Полегшується робота і для розробників, оскільки не потрібна тісна взаємодія серверу і вебзастосунку, що полегшує написання коду. Ще однією важливою особливістю SPA є можливість кешування даних, завдяки цьому такі веб-застосунки можуть працювати деякий час в автономному режимі.

Основним недоліком SPA є надзвичайно складне SEO-просування. Сканери можуть аналізувати вміст веб-сторінки, але в них починаються проблеми, коли відбувається динамічне завантаження інформації. Після першого завантаження сторінки, сканер не знає, що потім можуть бути завантажені інші дані, і тому уже на цьому етапі зупиняє аналіз веб-сторінки. Другим недоліком є недостатня захищеність SPA від XSS-атак, хакери можуть вставляти власні скрипти у код веб-застосунків, намагаючись викрасти або пошкодити інформацію. Проте цей недолік можна легко вирішити за допомогою захисту кінцевих точок на сервері, тому він не є на стільки критичним, як це могло здатись на перший погляд.

2.1.4 Вибір типу веб-застосунку

Проаналізувавши усі типи веб-застосунків, було вирішено, що в даній роботі найбільш актуально використати тип SPA. Оскільки висока швидкість взаємодії з користувачем і ефективність розробки це саме ті два критерії, які стали вирішальними при виборі. В інтернет-магазинах важливо, щоб

користувач довго не очікував на завантаження сторінки, оскільки це може призвести до погіршення конверсії в магазині. PWA не підходить для даної роботи, оскільки немає необхідності будувати застосунок, який користувачі могли б інсталювати на мобільний телефон. А MPA значно програє у швидкості та складності розробки порівняно з SPA.

2.2 Фреймворки та бібліотеки для розробки веб-застосунку

2.2.1 Розробка без використання фреймворків або бібліотек

Якщо говорити про розробку веб-застосунків, то відразу заходить мова про три основні технології, які використовуються розробниками — це HTML, CSS, JavaScript.

HTML — мова гіпертекстової розмітки, яка використовується для опису та розміщення елементів на веб-сторінці. Саме за допомогою цієї розмітки браузер розуміє, де потрібно розмістити відео, картинку або блок тексту. CSS — спеціальна мова, яка використовується для опису стилів, що будуть багаторазово використовуватись для HTML-розмітки веб-сторінки. Тобто всі кольори, шрифти і розміри блоків, які бачить користувач на веб-сторінці, чітко описані за допомогою CSS-правил, що застосовуються в потрібних місцях. JavaScript — це текстова мультипарадигменна мова програмування, яка використовується, щоб зробити веб-сторінки більш динамічними та інтерактивними для користувачів. JavaScript надає можливість відображати на веб-сторінках таймери, анімації, відтворювати медіа-контент та багато іншого, без чого користувачі вже не уявляють сучасні веб-сайти.

Проте, якщо говорити про розробку веб-застосунків, використовуючи лише ці три технології, виникає ряд проблем. По-перше, складна підтримка крос-браузерності, оскільки весь код для різних браузерів доведеться прописувати вручну без допомоги спеціальних інструментів та бібліотек, які можуть робити це автоматично. По-друге, із ростом веб-застосунку

виникатимуть проблеми із підтримкою і написанням бізнес-логіки, оскільки всі важливі частини коду знаходитимуться в багатьох різних файлах. Ще одним недоліком є те, що веб-застосунки, які написані тільки на цих трьох базових технологіях, зазвичай мають проблеми з архітектурою і ці проблеми тільки продовжують накопичуватись разом із ростом проекту. Ну і, мабуть, головна проблема такого підходу — це час, який витрачається на розробку. Цей час набагато більший у порівнянні з розробкою, коли використовуються фреймворки та бібліотеки, оскільки розробнику необхідно багаторазово писати одноманітний і повторюваний код, який займає час, але не додає нового функціоналу.

2.2.2 React

React — JavaScript-бібліотека, яка використовується для створення користувацьких інтерфейсів.

React використовує віртуальний DOM для зміни елементів на вебсторінці, замість справжнього DOM, що дає можливість робити зміни в якомусь елементі веб-застосунку без перемальовування всієї сторінки. За допомогою React розробники можуть розбивати веб-застосунок на компоненти, а потім перевикористовувати їх у потрібних місцях, що дуже покращує архітектуру застосунку. Варто відзначити також і гарну базову документацію та приязну спільноту розробників, які згуртувались навколо цієї бібліотеки. Для програмістів важлива не тільки розробка, а і можливість відлагоджування вебзастосунку, React пропонує React Developer Tools, які можна встановлювати як розширення у браузері, ці інструменти дозволяють з легкістю знаходити проблеми в коді та вирішувати їх.

Проблем у React не дуже багато, але все ж вони є. Оскільки при програмуванні на React необхідно використовувати спеціальний синтаксис JSX, замість звичайної HTML-розмітки, то це підвищує поріг входу для недосвідчених розробників. Документація у React є і вона достатня, щоб

повністю зануритись у бібліотеку і зрозуміти її суть, проте оновлення в цій бібліотеці виходять часто, а от документація оновлюється дуже рідко. Відповідно розробникам потрібно часто перенавчатися, але зазвичай ці знання беруться з інтернет-простору, а не з документації. Також присутні деякі проблеми з SEO-просуванням та індексацією для Google.

2.2.3 Angular

Angular — платформа розробки програмного забезпечення (фреймворк), що використовується для створення користувацьких інтерфейсів. По-перше, Angular вирішує майже всі основні проблеми архітектури у веб-застосунках. У даному фреймворку є чіткі правила та розділення, як частини коду мають взаємодіяти між собою, що значно полегшує роботу для розробника, оскільки не потрібно вигадувати власну архітектуру. Angular побудований з архітектурою Model-View-Controller і у фреймворку було синхронізовано модель і відображення. Коли змінюються дані в моделі — відбувається зміна відображення, відповідно інженерам не потрібно писати додатковий код для синхронізації цих двох частин веб-застосунку, що значно зменшує час розробки.

Проте, є основна проблема, яка часто відштовхує розробників від вибору Angular як інструмента для розробки проекту — це продуктивність. Angular абсолютно не підходить для невеликих SPA, оскільки у нього велика і важка екосистема, яка більше призначена для комплексних та складних проектів. Angular є універсальним інструментом, але це означає, що у ньому дуже багато способів розробки необхідного функціоналу, що в різних ситуаціях може тільки заважати розробникам. Чітка архітектура може бути одночасно як і плюсом, так і мінусом, оскільки можуть виникати завдання, які зовсім не підходять для такої архітектури, в такому випадку інженерам доводиться вигадувати обхідні шляхи, щоб подолати цю проблему.

2.2.4 Vue.js

Vue.js — це фреймворк, який охоплює більшість функцій, необхідних для розробки користувацького інтерфейсу.

Vue.js у багатьох характеристиках дуже схожий з React та Angular, проте в них є, все-таки, деякі відмінності. Як і React, Vue.js має компонентний підхід до написання веб-застосунків і високу продуктивність веб-застосунків, зокрема це досягається через невеликий розмір фреймворку. Як і Angular, Vue.js має свою чітко визначену архітектуру, але оскільки це фреймворк, то знову ж таки варіантів написання одного і того ж функціоналу дуже багато, у великих командах з великою кількістю розробників це може бути важливою проблемою. Простота коду та інтуїтивність у Vue.js також на високому рівні, присутня і чітко описана документація.

Одним з недоліків Vue.js є те, що це достатньо молода технологія і вона ще не набула широкої популярності серед розробників. Спільнота інженерів, які підтримують і використовують даний фреймворк дуже обмежена, відповідно під час написання коду, можуть виникнути проблеми з прийняттям тих, чи інших рішень, які ще не використовувались іншими розробниками. Велика кількість інструментів розробки, які підтримуються іншими бібліотеками та фреймворками досі недоступна для Vue.js через брак фінансування та підтримки зі сторони спільноти. Хоч в останні роки даний фреймворк набуває більшості популярності серед розробників, йому доведеться пройти ще довгий шлях до популярного і повноцінного рішення.

2.2.5 Вибір бібліотеки або фреймворку

Проаналізувавши усі варіанти, описані в попередніх підрозділах, було вирішено, що найбільш доцільно використати бібліотеку React. Написання сучасного веб-застосунку без використання фреймворку або бібліотеки було б дуже складним і вимагало б великої кількості часу та зусиль. Angular не варто

використовувати, бо проект не є на стільки комплексним і великим, тому він не потребує такої складної структури і внутрішнього середовища як в Angular. Vue.js хоч і доволі перспективний проект, але не на стільки популярний, як React, який ні в чому не поступається Vue.js, а популярність і підтримуваність вибраного рішення є також важливим критерієм. React дозволить створити високоефективний, крос-браузерний веб-застосунок із максимальним комфортом при розробці. Оскільки React не диктує ніяких правил архітектури, то можливо самостійно зробити вибір архітектури застосунку, опираючись на власний досвід, що теж є значною перевагою.

2.3 Менеджери станів для веб-застосунку

2.3.1 React Hooks

React Hooks — одна з найкращих функцій, які коли-небудь додавалися в бібліотеку React із самого початку її створення. Після введення хуків React Hooks функціональні компоненти змогли створювати і використовувати локальні стани для своїх потреб. По свої суті React Hooks це набір стандартних функцій у React, які дозволяють інженеру описувати бізнес-логіку всередині окремих компонентів веб-застосунку. За допомогою React Hooks можна написати застосунок від початку і до кінця, не використовуючи сторонні бібліотеки для менеджменту станів.

Основною перевагою React Hooks є те, що вони достатньо легкі у використанні для розробників, які уже знайомі з React, і не потребують встановлення додаткових бібліотек. З їхньою допомогою можна створювати перевикористовану і ізольовану логіку, яку легко тестувати і підтримувати.

Проте React Hooks мають і свої недоліки, наприклад, коли компоненти ізольовані і ніяк не взаємодіють між собою, то архітектура веб-застосунку, який написаний за допомогою такого інструменту ніяк не постраждає. Але чи часто бувають такі ситуації у реальних проектах? Звісно ні, зазвичай компоненти

мають тісно взаємодіяти між собою і логіка щільно переплітається, React Hooks в такому випадку не найкращий вибір, оскільки вони будуть створювати багато додаткових проблем та неочікуваних ефектів. Менеджмент станів з React Hooks орієнтований на невеликі веб-застосунки, компоненти яких дуже слабко зв’язані між собою. Для програмістів-новачків необхідна значна кількість тренувань, щоб навчитися правильно використовувати складніші функції, наприклад, такі як useEffect, useCallback та useMemo.

2.3.2 Redux

Redux — це передбачуваний контейнер станів для веб-застосунків, що написані за допомогою JavaScript. Redux можна використовувати з React, або з будь-якою іншою бібліотекою, він допомагає писати застосунки які послідовно працюють і можуть запускатися в різних середовищах. Redux має велику екосистему та багато інструментів для розробки, які полегшують написання та тестування коду.

Redux робить стани веб-застосунку передбачуваними і повністю відділяє їх від компоненти представлення, відповідно зменшується кількість проблем, які необхідно відслідковувати і виправляти інженеру. Redux добре справляється з глобальними станами, які застосовуються для усього веб-застосунку, він дозволяє легко керувати ними та змінювати їх. Відлагоджування в Redux теж не буде проблемою, оскільки є спеціальний інструмент Redux Devtools, який дозволяє відслідковувати всі зміни у станах прямо у браузері.

Redux не потрібно обирати для розробки веб-застосунку у декількох випадках. Перший з них це, коли застосунок не має дуже складної структури і не є комплексним (можна сказати, що за сферою використання Redux дуже схожий на фреймворк Angular, який уже описувався раніше). У невеликих SPA Redux зазвичай тільки погіршує читабельність коду, оскільки програмісту необхідно увесь час писати повторюваний код для менеджменту станів, який тільки засмічує нескладну логіку веб-застосунку. Другий — коли застосунок не

потребує глобального стану, наприклад, коли існують незалежні модулі застосунку, яким не потрібно взаємодіяти між собою. При використанні Redux виникають проблеми, коли необхідно використовувати не один глобальний стан на увесь застосунок, а декілька локальних станів, дана бібліотека не орієнтована на таке використання і не надає можливості створення локальних станів.

2.3.3 Effector

Effector — невеликий ефективний менеджер станів, який може поєднуватись з будь-якою бібліотекою відображення. Дана технологія є молодшою порівняно з Redux, але швидкими темпами набирає популярність серед розробників. Не потребує встановлення ніяких додаткових бібліотек для підтримки TypeScript.

Основними перевагами Effector є його універсальність і велика кількість функцій для роботи зі станами, що полегшує розробку, оскільки не потрібно писати довгі ланцюжки послідовностей, які будуть виконуватись одна за одною, достатньо використати стандартну функцію з даної бібліотеки. Effector має потужну екосистему, у якій присутні інструменти для відлагоджування, логування та взаємодії зі стандартними API браузера.

Значною перевагою Effector, яку необхідно окремо виділити, є можливість створювати окремі локальні стани, які повністю відділені від компоненти представлення. Тобто Effector об’єднав переваги Redux і React Hooks, у ньому є можливість повністю розділяти бізнес-логіку і представлення, а також не має потреби зберігати всі дані в одному глобальному стані, що значно полегшує розбиття веб-застосунку на компоненти.

Вагомий недолік Effector — поріг входу для розробника, оскільки наявна велика кількість функцій для роботи зі станами, тому інженеру необхідний деякий час для тренування та вивчення цих функцій.

2.3.4 Вибір бібліотеки або фреймворку

Виходячи з вище описаного, було зроблено вибір використати менеджер станів Redux, хоче Effector і поєднавує у собі переваги Redux і React Hooks, і його використання буде більш доречним, але працювавши з менеджером Redux і добре зрозумівши його певні переваги, було прийнято рішення використати саме його.

2.4 Препроцесори та бібліотеки для написання стилів веб-застосунку

2.4.1 CSS-модулі

CSS-модулі — це CSS файли, у якому усі назви файлів і анімацій за замовчуванням мають локальну область. Унікальність імен досягається за допомогою використання хешу на основі файлу, шляху до цього файлу та імені стилю.

Генерування унікальних імен для стилів дуже спрощує написання коду, оскільки розробнику не потрібно думати над унікальним ім’ям для стилю. Ця проблема також вирішується і за допомогою використання таких методологій як BEM, які диктують правила для написання стилів, але не кожен розробник захоче дотримуватись цих правил. Стилі знаходяться окремо від JSX, що покращує читабельність розмітки.

В усьому іншому це той же CSS-код з усіма його недоліками. Немає можливості використовувати вкладені стилі, крос-браузерність необхідно підтримувати вручну, немає можливості використовувати змінні та недоступні інші зручні функції препроцесорів.

Варто також зазначити, що глобальні стилі потрібно об’являти конструкціями, які відрізняються від звичайного CSS, це може бути неочікуваним для недосвідченого програміста. Ще одним вагомим недоліком є

необхідність прописувати текстову властивість className для елементів в JSX, можна легко сплутати імена класів, оскільки написання властивості відбувається вручну, в такому випадку застосуються хибні стилі.

2.4.2 SCSS/SASS

SCSS/SASS — препроцесорні скриптові мови, які компілюються або інтерпритуються в CSS. Єдина різниця між цими двома препроцесорами — це їхній синтаксис, у SCSS він більше схожий до синтаксису звичайного CSS.

Препроцесори надають можливість розбивати стилі на окремі модулі і повторно перевикористовувати їх. Стилі добре організовані і мають кращу структуру завдяки використанню вкладених стилів та можливості створення змінних. Найбільшою перевагою препроцесорів для розробника є зменшення часу написання стилів та зменшення дублювання коду завдяки вбудованим зручним функціям.

Щодо недоліків препроцесорів, то можна виділити час, який необхідний на компіляцію SCSS/SASS в CSS, це може значно впливати на процес розробки, оскільки розробнику необхідно майже миттєво бачити зміни на веб-сторінці. Препроцесори можуть генерувати дуже великі CSS-файли, завантаження яких займає більше часу, ніж завантаження оптимізованих розробником файлів. Відлагоджування веб-застосунку при використанні препроцесорів також може значно постраждати, оскільки додається величезна кількість нового синтаксису і функцій, використання яких для розробника без досвіду може бути проблематичним. Проблема використання className, як і в CSS-модулях нікуди не зникла.

2.4.3 Emotion

Emotion — це бібліотека, призначена для написання CSS-стилів за допомогою JavaScript. Emotion дозволяє розробнику використовувати CSSin-JS

підхід до написання стилів, такий підхід дозволяє уникнути проблеми глобальних стилів при розробці завдяки прив’язці стилів до потрібного компоненту прямо в JavaScript коді.

Emotion надає для використання вбудований SCSS, тому відповідно і всі переваги даного препроцесора також присутні. Основною перевагою Emotion є можливість виносити стилі в JavaScript об’єкти, а потім використовувати ці об’єкти передаючи в CSS властивість JSX-елемента. Не потрібно більше ніяких ручних введень className, аналізатор коду сам підкаже розробнику, яку властивість об’єкта зі стилями можна використати для стилізації того чи іншого елемента. Разом з Emotion стилізація стає більш гнучкою, можна оголошувати як глобальні стилі, так і в межах одного компонента, застосовувати стилі в залежності від умов, перевикористовувати анімації, застосовувати декілька наборів стилів для одного компонента та багато іншого.

Проблема компіляції нікуди не поділась, оскільки всеодно потрібно перетворювати Emotion код у звичайний CSS. Ще однією проблемою є те, що стилі стають більш схожими на JavaScript-код і це може бути неочікувано для розробників, які тільки починають працювати з Emotion, а до цього використовували лише звичайну стилізацію.

2.4.4 Emotion і styled-components

CSS надає стилів веб-сторінкам протягом багатьох років, і все більше і більше функцій постійно додаються до CSS, і ви не можете знайти жодної веб-сторінки без CSS в даний час.

Існують різні способи додавання css для реагування додатків, включаючи глобальні стилі та вбудовані стилі, які не рекомендуються. одним з найкращих способів додавання стилів до ваших компонентів є CSS в JS, який дозволить вам використовувати змінні JAVASCRIPT в CSS, ізолювати стилі компонентів і зробити стилі більш динамічними і придатними для повторного використання. У той час як деякі інші SPA-фреймворки мають вбудовані в компоненти

рішення css (наприклад, улюблений Vue js), просто не реагують на них, тому, хоча ми можемо використовувати такі рішення, як створення файлу css для кожного компонента або використання модулів css (що роблять багато розробників), існують сторонні бібліотеки, які можуть дозволити нам додавати CSS в JS або tsx-файли компонентів, що є більш ефективним і зручним.

Дві з найпопулярніших бібліотек CSS в JS - це Emotion та styled-components, тому давайте порівняємо їх і з'ясуємо, яку з них краще використовувати.

Стилізовані компоненти дозволяють створювати стилізовані багаторазові React компоненти і найголовніше рестайлінг існуючих компонентів, що дуже корисно, коли ви хочете налаштувати імпортовані компоненти або обернути і налаштувати компоненти з UI бібліотек.

styled components підтримує вкладені стилі, медіа запити, теми і ключові кадри, що робить його потужною бібліотекою для стилізації.

Emotion - це одна з найпопулярніших CSS в JS бібліотек. Це фреймворк-агностичний спосіб, який є більш легким і не вимагає зовнішніх плагінів або додаткових налаштувань, при цьому для div буде згенеровано ім'я класу з переданими стилями. Emotion також має модуль react, який додає новий проп "CSS", який є більш ефективним, ніж фреймворк-агностичний метод, і підтримує вкладені селектори та медіа запити.

Emotion react також підтримує SSR і тематизацію, він передбачуваний і легко тестується. Ще однією цікавою особливістю Emotion є те, що він має "стилізований API", який має схожий синтаксис зі стилізованими компонентами. Порівняння обидві бібліотеки корисні і популярні, обидві підтримують необхідні функції для стилізації.

2.4.5 Вибір препроцесора або бібліотеки для написання стилів

Проаналізувавши підходи до стилізації веб-застосунків було прийнято рішення використати styled-components для даної роботи. Оскільки CSS-модулі

не мають ніяких переваг над CSS, окрім унікальності імен, що вирішує проблему глобальних стилів, то їх варто використовувати лише разом із препроцесорами. Препроцесори додають багато нових функцій та можливостей для розробника, полегшуючи процес стилізації веб-застосунку, проте всеодно виникають деякі проблеми у зручності застосування цих стилів (введення властивості className для елементів).

Бібліотека Emotion змогла поєднати основні плюси двох попередніх технологій і додати компонентний підхід у написання стилів, що покращує архітектуру веб-застосунку і дозволяє інженеру швидко і ефективно розробляти його, але порівнюючи Emotion з styled-components, було прийнято рішення використати саме styled-components, хоча і Emotion має певні переваги і це новіший засіб, але більш звичнішим, зрозумілішим та затребуваним на даний момент є саме styled-components.

2.5 Мови програмування і фреймворки для розробки API

2.5.1 Golang і Gin

Golang — мова програмування, яка була розроблена Google і має відкритий вихідний код. Це достатньо швидка компільована мова, яка має статичну типізацію і дозволяє забезпечувати гнучку і модульну структуру програм. В останні роки Golang набирає популярності серед розробників через простий синтаксис, високу продуктивність (яка наближається до продуктивності мови програмування C) і швидкість компілювання без використання додаткових віртуальних машин тощо.

Golang достатньо гнучка мова, тому її можна використовувати у багатьох сферах програмування, наприклад, у розробці серверної частини вебзастосунків, для машинного навчання, розробці медіа-платформ та у багатьох інших сферах. Через достатньо простий синтаксис і легкість вивчення, розробники можуть доволі швидко опанувати дану мову і використовувати її на

реальних проектах. Вбудоване тестове середовище дозволяє будувати процес ефективного тестування програми, а інструменти для статичного аналізу та автоматичної документації коду допомагають зменшити кількість проблем та помилок у програмі.

Golang є гарним вибором при розробці SPA, проте, якщо виникає необхідність використати бібліотеку для написання інтерфейсу користувача лише за допомогою даної мови, розробників чекає розчарування, оскільки такої бібліотеки в Golang просто немає. Gin — це високопродуктивний HTTP-фреймворк, який написаний на Golang. Gin дозволяє розробнику створювати серверну частину веб-застосунків за допомогою вбудованих функцій для реалізації маршрутизації, кінцевих точок, створення контексту, валідації JSON тощо.

Основними перевагами Gin є простота його використання, швидкість роботи і наявність усіх необхідних функцій для створення веб-застосунку. Продуктивність досягається за допомогою використання високопродуктивного і легкого маршрутизатора HttpRouter. Відстеження помилок програми у Gin також реалізоване на високому рівні, за допомогою проміжного програмного забезпечення можна записувати і логувати усі критичні помилки, а потім досліджувати і виправляти їх.

2.5.2 JavaScript, Node.js і Express

Про JavaScript уже йшла мова в минулих підрозділах у контексті розробки користувацьких інтерфейсів, проте дану мову програмування можна використовувати і для розробки серверної частини веб-застосунків. Саме із допомогою Node.js у розробників з’явилась можливість використовувати JavaScript не тільки у браузерах, а й у інших середовищах.

Node.js — це не фреймворк або бібліотека, насправді це середовище виконання, яке побудоване на Chrome V8 JavaScript. Найочевиднішою перевагою Node.js є те, що він дозволяє використовувати JavaScript і для

побудови користувацьких інтерфейсів, і для побудови API, що значно підвищує ефективність розробки. Розробник бачить схожий код в обох частинах вебзастосунку, йому не потрібно вивчати нову програмування і постійно перемикатися між іншими синтаксисами. Величезна екосистема, яка надає можливість легко інсталювати необхідні бібліотеки і пакетний менеджер npm пришвидшують розробку і роблять її більш гнучкою. В останні роки Node.js все частіше почали використовувати для побудови мікросервісної архітектури через універсальність і технологічність даного інструменту.

Express — простий і ефективний фреймворк для розробки API на JavaScript. Оскільки Express використовується у середовищі виконання Node.js, то відповідно усі додаткові бібліотеки з npm також доступні, що робить розробку гнучкою та універсальною. Ще однією перевагою є те, що Express є популярним інструментом і має велику спільноту розробників, які підтримують його використання, тому в мережі є багато прикладів коду на Express, які можна використовувати для навчання і вдосконалення навичок розробки. Основним недоліком даного фреймворку є проблеми при запитах на кінцеві точки API, якщо використовуються проміжні програмні забезпечення, що виконують певні функції перед обробкою запиту.

2.5.3 Python і Django

Python — це інтерпритована, об’єктно-орієнтована мова програмування, яка має динамічну семантику. Простий синтаксис і гарна читаємість коду дуже часто приваблюють програмістів. Оскільки етапу компіляції програм, написаних на Python, немає, то процес відлагоджування і розробки стає дуже швидким та ефективним. Python можна використовувати при розробці серверної частини веб-застосунків, для аналізу даних та машинного навчання, для автоматизації і написання скриптів.

Основними превагами Python є легкий синтаксис і швидке вивчення, великий обсяг різних бібліотек і фреймворків для різних застосувань, і

можливість запуску коду на різних платформах без необхідності його модифікації. До недоліків слід віднести обмеження у швидкості виконання коду через те, що мова є динамічно типізованою та інтерпритованою, та споживання величезного обсягу в пам’яті для забезпечення виконання усіх функцій.

Django — це високорівневий фреймворк для Python для розробки серверної частини веб-застосунків, він забезпечує чистий дизайн вебзастосунку та швидкість розробки.

До важливих переваг слід віднести можливість швидкого масштабування і продуктивність комплексних веб-застосунків, оскільки основною ідеєю Django є швидке створення проектів, що дуже приваблює розробників стартапів. Django досить гнучкий, щоб працювати і з розробкою машинного навчання, і для аналізу даних, і для розробки кінцевих точок API. Даний фреймворк є доволі популярним серед розробників, тому навколо нього згуртувалась дружня велика спільнота, а документація Django є добре структурованою і зрозумілою навіть для починаючих програмістів. Оскільки Django це високорівневий фреймворк, то він не зовсім підходить для менш складних проектів і зазвичай для таких веб-застосунків використовують інший фреймворк — Flask. Хоч Django має дуже детальну документацію і відкриту спільноту, розробка API неможлива без повного вивчення і розуміння структури даного фреймворку, це може сповільнити розробку на початкових етапах.

2.5.4 Вибір мови програмування і фреймворку

Обираючи мову програмування і фреймворк було вирішено в основному керуватися власним досвідом, оскільки неможливо вивчити нову мову програмування і почати використовувати невідомий фреймворк у короткі терміни. Тому було прийнято рішення використовувати мову JAVASCRIPT з фреймворком Expressйович, оскільки даний фреймворк забезпечує усі необхідні функції, які необхідні для якісної і ефективної розробки API. Django

не найкращий варіант для веб-застосунку, що розроблятиметься, оскільки він не буде на стільки комплексним і складним, відповідно усі ресурси і можливості Django не будуть повністю використані. Щодо Express, то у мене не було досвіду у використанні даного фреймворку, тому швидкість розробки порівняно з Gin була б значно нижчою.

3 РОЗРОБКА СИСТЕМИ

3.1 Структура системи

Система складається з двох основних частин, які певним чином взаємодіють між собою. Checkout — сторінка оформлення замовлень, яка заміняє стандартну сторінку оформлення замовлень в інтернет-магазині після інсталяції системи. Admin Dashboard — панель адміністратора, за допомогою якої можна керувати платіжними шлюзами та інтеграцією системи з інтернетмагазином, також є можливість переглядати замовлення, які створює покупець. API — серверна частина системи, з якою взаємодіє панель адміністратора і сторінка оформлення замовлень.

Оскільки система є доволі комплексною і розгортання трьох окремих застосунків є високовартісним, то було прийнято рішення використовувати безкоштовну версію Ngrok. Ngrok — утиліта, яка дозволяє створювати тунелі, таким чином відкриваючи локальні порти для глобальної мережі. У безкоштовної версії Ngrok є деякі недоліки, наприклад, необхідно кожні дві години перестворювати тунель, проте для тестування і розробки усіх частин застосунку такого інтервалу більш, ніж достатньо. Тобто для того, щоб після встановлення системи в інтернет-магазині відкривалася розроблена сторінка оформлення замовлень, необхідно ввести всього декілька команд і Checkout буде доступний в глобальній мережі, такі ж дії можна провести і з API.

API умовно можна розділити на частини, які необхідні для сторінки оформлення замовлень та панелі адміністратора. Наприклад, для панелі адміністратора необхідні кінцеві точки, які дозволяють авторизуватись адміністратору, взаємодіяти з магазинами, а також налаштовувати інтеграції та платіжні шлюзи цих магазинів. Для сторінки оформлення замовлень логіка взаємодії з магазинами та авторизацією не потрібна, проте необхідні кінцеві точки, які допомагають повністю проходити процес покупки та завершувати його.

3.2 Організація бази даних

Додатковим фактором, що впливає на продуктивність системи, є правильне використання баз даних. Усі сучасні проєкти для відображення динамічної інформації використовують СУБД (системи керування базами даних), а СУБД, своєю чергою, вимагає правильного налаштування своїх параметрів.

База даних являє собою незалежний клієнт-серверний додаток, який запускається і працює в операційній системі. Зовнішні додатки з'єднуються з базою даних через TCP/IP або внутрішні потоки, надсилають SQL-запити й отримують у відповідь від бази даних необхідні дані.

Можливі два типи з'єднання з базою даних для веб-додатка:

звичайне з'єднання;

постійне з'єднання (Persistent).

Звичайне з'єднання встановлюється щоразу під час виконання сторінки при першому зверненні до бази даних. Встановлене з'єднання звільняється (у більшості випадків і закривається) після завершення сторінки.

Постійне з'єднання (функції PHP зазвичай називаються *_pconnect) встановлюється один раз під час першого звернення до бази даних і під час повторних звернень, навіть з інших сторінок, використовуються вже відкриті з'єднання до бази даних.

З огляду на те, що відкриття з'єднання до бази даних - процес відносно дорогий за ресурсами та часом (відбувається встановлення TCP/IP-з'єднання, виділяються буфери пам'яті, проводиться перевірка та авторизація, налаштовуються перекодувальники та виконується ціла низка інших функцій), використання постійних з'єднань є переважним, якщо це не призводить до перевищення кількості одночасних з'єднань із базою даних. Доцільність застосування постійних з'єднань прямо пропорційна частоті звернень до бази даних, тобто відвідуваності сайту.

Встановлюючи з'єднання з базою даних, при вказівці параметру server

значення localhost або localhost:port, PHP здебільшого намагатиметься з'єднатися з локальним сокетом без використання TCP/IP. Якщо все ж таки потрібно використовувати TCP/IP, використовуємо адресу 127.0.0.1 замість localhost.

З'єднання без використання TCP/IP дає найвищі результати за продуктивністю, особливо під час передачі великих обсягів інформації з бази даних. Але якщо база даних розташована на іншій машині й уникнути з'єднання по TCP/IP не вдається, використання постійного з'єднання стає ще більш вигідним.

У більшості випадків база даних працює на тій самій машині, що й PHP-додаток і веб-сервер. Але цілком можлива ситуація, коли база даних встановлена на сусідній машині або навіть в іншого провайдера. "1С-Бітрікс: Керування сайтом" встановлює з'єднання з сервером бази даних через стандартні бібліотеки, які йдуть у постачанні мови програмування PHP.

В "1С-Бітрікс: керування сайтом" з версії 20.900.0 тип з'єднання з базою даних встановлюється у файлі /bitrix/.settings.php в ядрі D7 (рисунок 3.1).

```
'connections' => array (
        'value' => array (
            'default' => array (
                'className' => '\\Bitrix\\Main\\DB\\MysqlConnection',
                'host' => 'localhost:31006',
                'database' => 'admin_bus',
                'login' => 'admin_bus',
                'password' => 'admin_bus',
                'options' => 2,
                'handlersocket' => array (
                    'read' => 'handlersocket',
                ),
            ),
```

Рисунок 3.1 – Приклад вмісту файлу

Константа DBPersistent використовується щоб встановити тип з'єднання з базою даних.

Для вимкнення постійного з'єднання використовується define(“DBPersistent”, false);

Використання постійних з'єднань може зажадати деякого налаштування Apache і MySQL. Переконайтеся, що ви не перевищите максимальну кількість дозволених з'єднань. Читайте додаткову інформацію в документації PHP щодо використовуваної вами бази даних.

За замовчуванням дані про сесії користувачів зберігаються у файловій системі сервера.

Якщо веб-сервер єдиний, то такий спосіб зберігання сесій у файлах найзручніший. Основний його плюс - найвища продуктивність. Як показують різні навантажувальні тести, швидкість генерації сторінок сайту при зберіганні сесій у базі знижується на 3-5%.

Якщо веб-серверів кілька, то можливо, що один запит користувача (наприклад, безпосередньо авторизація) потрапить на один сервер, а наступний або будь-які інші запити - на інші сервери, де відвідувач ще не буде авторизований. Подібні ситуації доставлять цілу низку незручностей для відвідувачів сайту. Також, у разі зберігання сесій у файлах буде некоректно вестися статистика відвідувачів.

Користувацька сесія повинна бути "прозорою" для всіх серверів веб-кластера. Тому потрібно ввімкнути зберігання сесій у базі даних. Увімкнення механізму зберігання даних сесій користувачів у базі даних виконується за допомогою кнопки увімкнути зберігання даних сесій у БД модуля.

Для зниження навантаження на базу даних і забезпечення "прозорості сесії", можна сесії в базі не зберігати, а замість цього налаштувати і використовувати в nginx модуль ip_hash.

3.3 Архітектура веб-застосунку

При розробці API було вирішено використати Clean Architecture. В основі даної архітектури лежить принцип чіткого розділення застосунку на шари, принципи SOLID, а також принцип інверсії залежностей. Останній означає, що шари верхнього рівня не мають використовувати код шарів низького рівня.

Іншими словами — бізнес логіка не повинна залежати від бази даних, контролерів та сторонніх API. У розробленому API існують такі шари: транспортний, бази даних (або репозиторій), сервісний, сутності та сторонні API.

При розробці сторінки оформлення замовлення і панелі адміністратора було вирішено керуватися однаковими принципами і для обох веб-застосунків використати однаковий тип архітектури — Feature-Sliced. Дана архітектура встановлює певні обмеження на розділення коду, весь застосунок розбивається на певні модулі — шари, які містять в собі слайси, а ті в свою чергу містять в собі сегменти. У Feature-Sliced присутні декілька видів шарів, основні з них, які розташовані у порядку зниження рівня: app (шар, який відповідає тільки за ініціалізацію застосунку), pages (шар сторінок, які присутні у веб-застосунку), widgets (самостійні компоненти сторінок, які композують features та entities), features (шар певного функціоналу), entities (шар сутностей), shared (спільні компоненти, які можуть використовуватись усіма шарами). Слайси не визначаються архітектурою, оскільки вони залежать від бізнес-логіки застосунку, а сегменти мають традиційне розділення на ui (відображення), model (бізнес-логіка), lib (додаткові бібліотеки, які необхідні для слайсу). Основне правило Feature-Sliced виглядає так: модулі високого рівня повинні залежати тільки від модулів низького рівня. Наприклад, не повинно бути ситуацій, коли в код шару entities імпортується код шару features.

Розгляд реалізації архітектури буде відбуватися на прикладі панелі адміністратора. На рисунку 3.2 зображено високорівневу структуру директорій, за допомогою якої можна добре відслідкувати розбиття на шари.

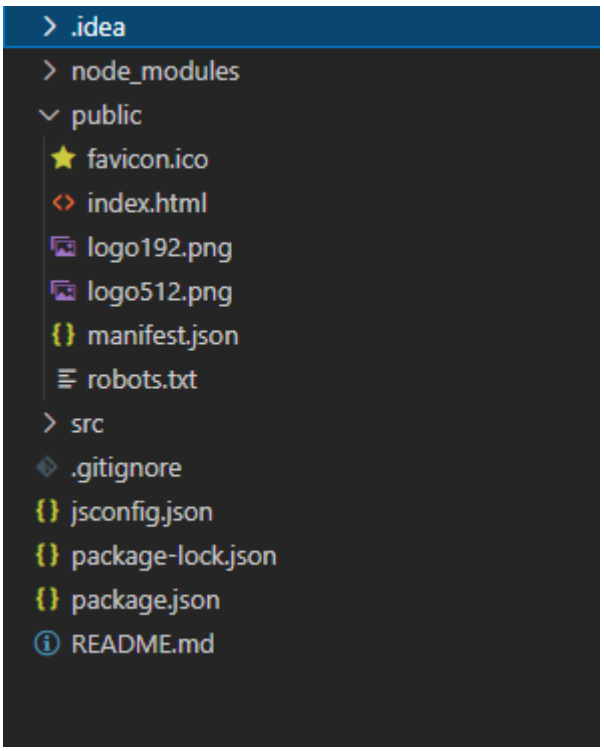

Рисунок 3.2 – Структура директорій та файлів API

На верхньому рівні присутні директорії .idea та react modules, які потрібні для запуску застосунку та для встановлення різних бібліотек для застосунку відповідно.

Директорія .public містить в собі файл index.html, який відповідає за генерацію самої веб-сторінки.

Лістинг 3.1 – Вміст файлу index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
```

```
          <div id="root"></div>
          <script>
            (function(w,d,u){
              var
s=d.createElement('script');s.async=true;s.src=u+'?'+(Date.now()/60000|0);
              var
h=d.getElementsByTagName('script')[0];h.parentNode.insertBefore(s,h);
            })(window,document,'https://cdn-
ru.bitrix24.ru/b23340092/crm/site_button/loader_2_vngyd6.js');
          </script>
        </body>
      </html>
```

Також дана директорія містить в собі ікони(logo.png) та шрифти для веб-застосунку(robots.txt).

В директорії Src, в папці index.js відбувається ініціалізація React проекту, сам react підключається до кореневого елементу Root і саме в ньому відбувається рендеринг самого веб-застосунку.

Лістинг 3.2 – Вміст файлу index.js

```
import React from "react";
import ReactDOM from "react-dom";
import "styles/index.css";
import App from "./App";
import reportWebVitals from "./reportWebVitals";

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById("root")
);
reportWebVitals();
```

Файл App.js вміщує в собі основні елементи дім-дерева, такі як Header, Main, Footer, а також відповідає за мобільну навігацію.

Лістинг 3.3 – Елементи дім-дерево та мобільна навігація в файлі App.js

```
import styles from "styles/App.module.scss";
import { BrowserRouter as Router, Switch, Route } from "react-router-dom";
import clsx from "clsx";

// PAGES
import Home from "pages/Home";
import Detail from "pages/Detail";
```

```
import Category from "pages/Category";

// COMPONENTS
import Header from "components/Header";
import BasketSidebar from "components/BasketSidebar";
import Footer from "components/Footer";
import MobileBottomNav from "components/MobileBottomNav";

// HOOKS
import useMobileDetect from "hooks/useMobileDetect";

// CONTEXT
import BasketContextProvider from "context/BasketContext";

const App = () => {
  const device = useMobileDetect();

  return (
    <Router>
      <BasketContextProvider>
        <div      className={clsx(device.type      ===      "mobile"      &&
styles.paddingForMobile, styles.container)}>
          <Header />
          <main className={styles.main}>
            <Switch>
              <Route path="/" exact>
                <Home />
              </Route>
              <Route path="/product/:slug">
                <Detail />
              </Route>
              <Route path="/category/:slug">
                <Category />
              </Route>
            </Switch>
          </main>
          <Footer />
        </div>
        <BasketSidebar />
        {device.type === "mobile" && <MobileBottomNav />}
      </BasketContextProvider>
    </Router>
  );
};

export default App;
```

Header.js в свою чергу містить в собі посилання на головну сторінку магазину, на контактний номер телефону, посилання на навігацію по категоріям та корзину, що наведено на лістингу 3.4.

Лістинг 3.4 – Вміст файлу Header.js

```
import styles from "styles/Header.module.scss";
import { Link } from "react-router-dom";
import GetIcon from "components/GetIcon";
```

```
import clsx from "clsx";
import CategoryItem from "./CategoryItem";
import useMakeRequest from "hooks/useMakeRequest";
import { BasketContext } from "context/BasketContext";
import { useContext } from "react";
import facebook from "../images/icons8-facebook-96.png";
import instagram from "../images/icons8-instagram-120.png";

const Header = () => {
  const                                result                                =
useMakeRequest("https://fakestoreapi.com/products/categories");
  const { basketItems, setBasketIsOpen } = useContext(BasketContext);

  return (
    <header className={styles.header}>
      <div className={styles.logo}>
        <Link to="/">
          <h2>SmartShop Online</h2>
        </Link>
      </div>
      <div style={{display: 'flex', flexDirection: 'row'}}>
    <div style={{display: 'flex', flexDirection: 'row'}}>
        <a                  href="https://www.facebook.com/SmartShopWatch/"
className={styles.a}>
          <img style={{width: 24, height:24}} src={facebook} alt="" />
        </a>
        <a                  href="https://www.instagram.com/smart.shop.ua/"
className={styles.a}>
          <img style={{width: 30, height:30}} src={instagram} alt="" />
        </a>
    </div>
    <a href="tel:+380-99-527-67-87" className={styles.a}>
      Tel: +380-99-527-67-87
    </a>
  </div>
        <div className={styles.navContainer}>

          <nav className={styles.nav}>
            <ul>
              <li>

                <ul         className={styles.subMenu}>{result.data         ?
result.data.map((cat,  index)  =>  <CategoryItem  data={cat}  key={index}  />)  :
<div>{result.error}</div>}</ul>
              </li>
              <li>
                <Link    to="/"    onClick={(e)    =>    e.preventDefault()}
className={styles.a}>
                  Categories
                </Link>
                <ul         className={styles.subMenu}>{result.data         ?
result.data.map((cat,  index)  =>  <CategoryItem  data={cat}  key={index}  />)  :
<div>{result.error}</div>}</ul>
              </li>
              <li>
                <Link
                  to="/"
                  className={clsx(styles.basketBtn, styles.a)}
                  onClick={(e) => {
                    e.preventDefault();
                    setBasketIsOpen((oldState) => !oldState);
                  }}
                >
                  <GetIcon icon="BsCart4" size={25} color="#ffffff" />
```

```
                    {basketItems.length       >       0       &&       <span
className={styles.basketLength}> {basketItems.length} </span>}
                  </Link>
                </li>
              </ul>
            </nav>
          </div>
        </header>
      );
    };

    export default Header;
```

В main знаходиться навігація на домашню сторінку, а також на сторінку деталі про товар та категорії.

Footer, в свою чергу, містить в собі нижні елементи сторінки, тобто подвал веб-застосунку.

Лістинг 3.5 – Вміст файлу Footer.js

```
    import styles from "styles/Footer.module.scss";
    import GetIcon from "components/GetIcon";

    const Footer = () => {
      return (
        <footer className={styles.footer}>
          <p>
            <GetIcon  icon="BsFillHeartFill"  size={22}  color="#da0037"  />
Vichevskiy
          </p>
        </footer>
      );
    };
    export default Footer;
```

Директорія hooks має два файли useMakeRequest.js(що відповідає за запити на API) та useMobileDetect.js(допомагає визначити чи на мобільному пристрої відкритий веб-застосунок і якщо так, то відбувається орієнтація застосунку під цей пристрій).

Лістинг 3.6 – Запити на API

```
    import { useState, useEffect } from "react";

    const useMakeRequest = (endpoint) => {
      const [result, setResult] = useState({
```

```
    data: null,
    error: null,
  });

  useEffect(() => {
    const asyncFunc = async () => {
      try {
        const response = await fetch(endpoint);
        const json = await response.json();
        setResult((old) => ({ ...old, data: json }));
      } catch (error) {
        setResult((old) => ({ ...old, error: new Error(error).message }));
      }
    };

    asyncFunc();
  }, [endpoint]);

  return result;
};

export default useMakeRequest;
```

Лістинг 3.7 – Визначення мобільного пристрою та орієнтація під нього

```
import { useEffect, useState } from "react";

const useMobileDetect = () => {
  const [device, setDevice] = useState({
    type: "desktop",
  });

  useEffect(() => {
    window.addEventListener("resize", () => {
      if (window.innerWidth <= 768) {
        setDevice((old) => ({ ...old, type: "mobile" }));
      } else {
        setDevice((old) => ({ ...old, type: "desktop" }));
      }
    });

    window.addEventListener("load", () => {
      if (window.innerWidth <= 768) {
        setDevice((old) => ({ ...old, type: "mobile" }));
      } else {
        setDevice((old) => ({ ...old, type: "desktop" }));
      }
    });
  }, []);

  return device;
};

export default useMobileDetect;
```

Директорія images містить в собі усі зображення, які присутні на сайті, pages – усі сторінки сайту (категорії, деталі та домашня сторінка), styles – стилістику сайту (рисунок 3.9).

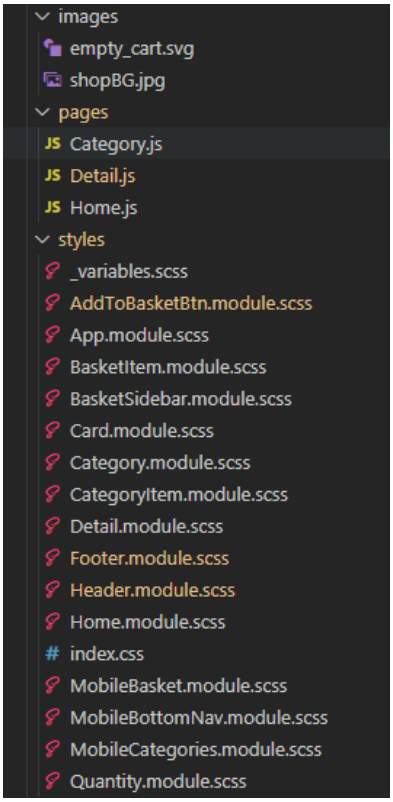

Рисунок 3.9 – Вміст директорій images, pages та styles

4 ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1 Тестування програмного продукту

Найважливішу роль у процесі створення сайту відіграє його тестування. Роботу виконує тестувальник, який перевіряє функціональність ресурсу згідно з набором критеріїв та виявляє помилки, які треба усунути.

Вибір домену та хостингу не було винесено як окремий етап. Він необхідний, але може проводитися в будь-який момент процесу створення ресурсу.

Створення веб-застосунку являється складною роботою, яка складається з багатьох етапів. В процесі створення веб-сторінки для онлайн магазину не були виконані всі етапи але тим не менш, можна побачити готову сторінку, яка має функціонал, тобто було виконано перші чотири етапи.

Тестування можна проводити різними способами, але потрібно не забувати про процес тестування та саму стратегію. Порядок дій залежить від цього. Сьогодні тестувальники веб-сайтів використовують такі типи, як:

- функціональне тестування;
- тестування простоти використання (зручність використання);
- тест продуктивності;
- тестування інтерфейсу користувача;
- тест безпеки.

Функціональне тестування.

Це один з важливих і незамінних видів тестів. Найважливішим правилом функціональних тестів є правильний розрахунок функцій. Наприклад, візьмемо Інтернет-магазин із не тільки знижками на товари, але і багатьма ситуаціями при покупці, n штук товару.

Всі ці варіанти слід розглянути. Зрештою, якщо функціональність проекту не працює в певному браузері, він ніде не буде працювати. Якщо ви отримуєте певну функціональність для веб-проектів, це переважно

перевіряється:

- пошук та придбання товарів, розміщення замовлення;
- навігація (юзабіліті);
- формати аутентифікації;
- товар, додавання, видалення, редагування тощо.

Тестування юзабіліті сайту.

Тестування юзабіліті – це вид тестування, який робить сайт корисним та практичним. Основна мета – показати користувачеві, що:

- сайт відкритий та корисний для інших;
- на сайті зручна навігація;
- справити хороше враження у користувача щодо сайту;
- розуміти, що може бути зайвим.

Основним завданням перевірки зручності використання сайту є його дизайн, де користувач хоче знайти та придбати, шукає необхідну інформацію, для чого йому ніщо не заважає.

Тестування навантаження сайтів.

Тестування продуктивності – це в основному стрес-тест. Тестування навантаження сайту в більшості випадків контролюється автоматично, тобто за допомогою спеціальних програм. Це дає вам можливість перевірити, наскільки добре це працюватиме за певного навантаження.

Метою цього тесту є кількість віртуальних користувачів, які вказують n запитів одночасно (навіть секунди). Як результат, чи може наш проект витримати, наприклад, 100 користувачів, які одночасно купують продукт або входять на сайт, відповідь показує, чи дійсно сайт може витримати таке навантаження.

Тестування інтерфейсу користувача.

Тестування інтерфейсу користувача – це тест графічного інтерфейсу користувача, який передбачає перевірку, чи відповідає сайт вимогам графічного інтерфейсу, чи виглядає він професійно, чи виконується в єдиному стилі.

У більшості випадків тестування інтерфейсу користувача виконується з

такими типами тестів (UI):

- тестування на відповідність стандартам графічного інтерфейсу;
- тестування з різною роздільною здатністю екрана;
- крос-браузерне тестування або сумісність з різними браузерами та версіями Інтернету;
- тестування локалізованих версій: точність перекладу (багатомовність, мультивалютність), перевірка довжини імен віджетів тощо;
- тестування графічного інтерфейсу користувача на цільових пристроях (смартфонах, контрольно-пропускних пунктах, планшетах);

Тестування сайту на наявність вразливостей.

4.2 Наповнення контентом та створення дизайну веб-сторінки

Передостанній етап створення веб-сайту – наповнення сторінок графічним та інформаційним наповненням. Тут є відео, фотографії, тексти та інша інформація, яку відвідувач може побачити або прочитати. Статті SEO написані на основі семантичного ядра, і якщо джерело призначене для просування за допомогою методу SEO, менеджер вмісту розміщує графічні елементи в логічній структурі. Сторінки проходять внутрішню оптимізацію. Усі етапи створення веб-сайту важливі для якісного функціонування ресурсу, але розробка дизайну є одним з основних етапів. Зрештою, дизайн - це те, що відвідувач сайту бачить насамперед, оцінює і вирішує залишитися на сторінці або закрити вкладку браузера. Технічне завдання, кнопки, банери та інші графічні елементи малюються відповідно до дизайнера. Важливо зазначити, що дизайнер малює шаблони для кількох основних шаблонів, використовуючи тенденції веб-дизайну, а не дизайн кожної сторінки. Дизайн завершується до тих пір, поки його не схвалить розробник, тобто удосконалить, на його думку, або, якщо замовник має, проект затвердить замовник.

4.3 Опис інтерфейсу користувача для застосунку та CRM-системи

Структура сайту інтернет магазину у значній мірі впливає на доступність ресурсу. Логічна та зручна структура дозволяє користувачам легко знаходити необхідні товари.

Зрештою, це впливає на конверсії, задоволеність відвідувачів і навіть на пошуковий трафік. На рисунку 4.1 зображено роботу алгоритму інтернет магазину на програмногому рівні та алгорим роботи з клієнтом (B2C).

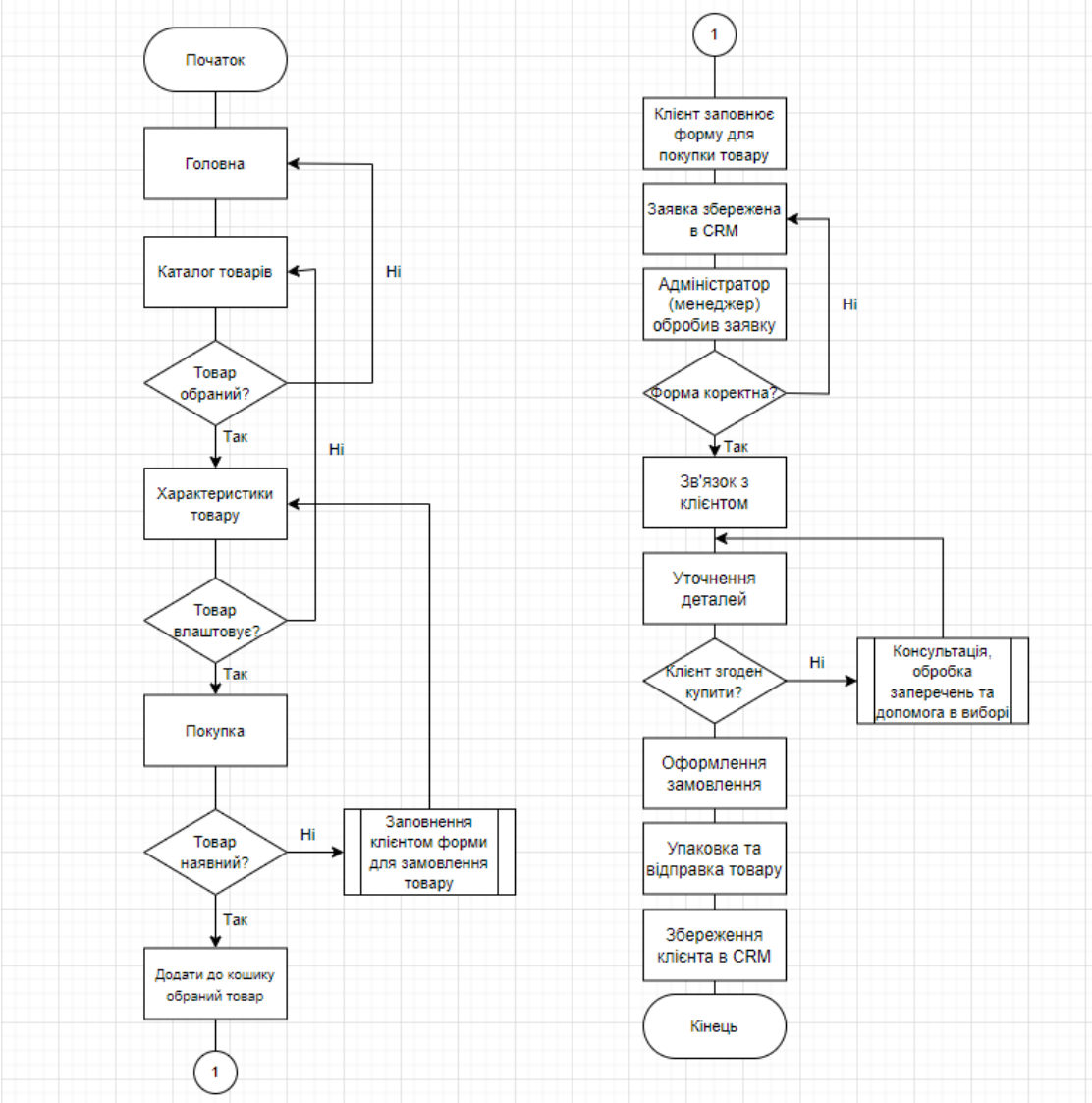

Рисунок 4.1 – Блок-схема роботи алгоритмів інтернет-магазину

На рисунку 4.2 показана головна сторінка веб-сайту для управління готельним комплексом. На верхній панелі значки соціальних мереж, при натисканні користувач був спрямований на сторінку соціальної мережі.

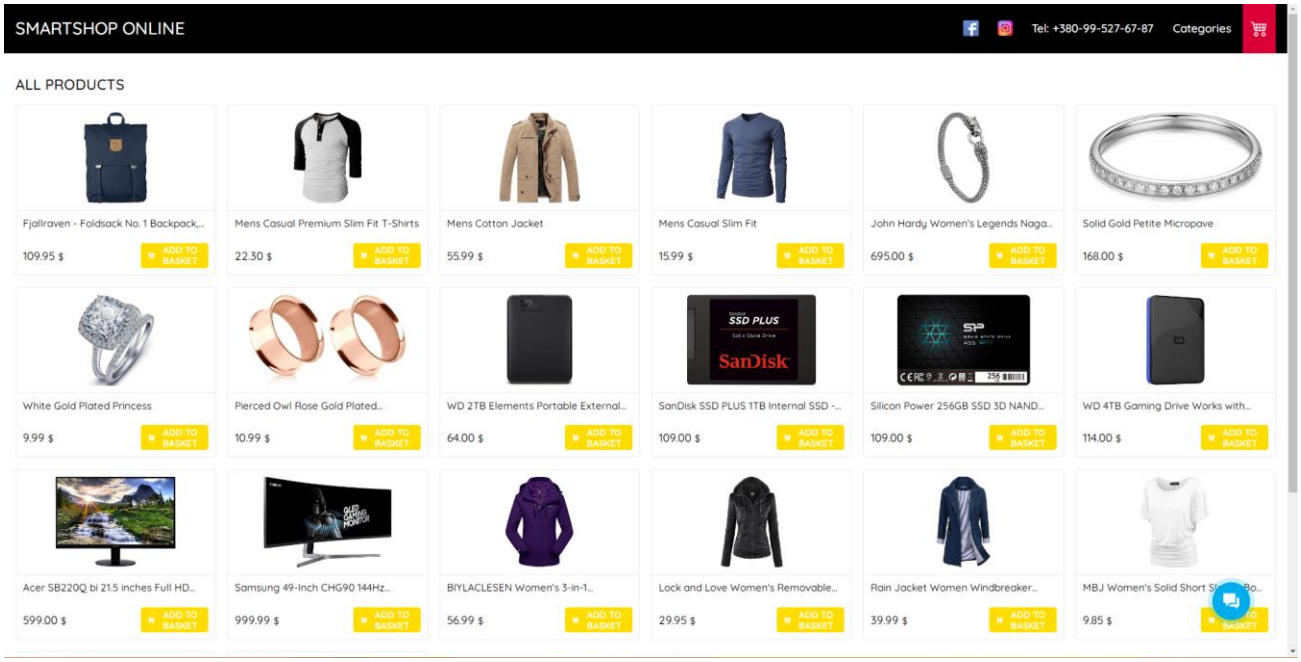

Рисунок 4.2 – Головна робоча область (сторінка) інтернет магазину

На рисунку 4.3 у верхній частині сторінки є посилання на номер телефону, посилання можна натиснути, при натисканні ви будете перенаправлені і ви можете зателефонувати на вказаний номер, а також тут є іконки з соціальними мережами, при натисненні на які вас буде перенаправлено в відповідну соціальну мережу.

Рисунок 4.3 – Клікабельні іконки з соцмережами та клікабельний номер телефону

В меню, що знаходиться в шапці веб-застосунку під назвою “Categories”,

при наведенні на нього відкривається доступ до 4 категорій товарів, як показано на рисунку 4.4.

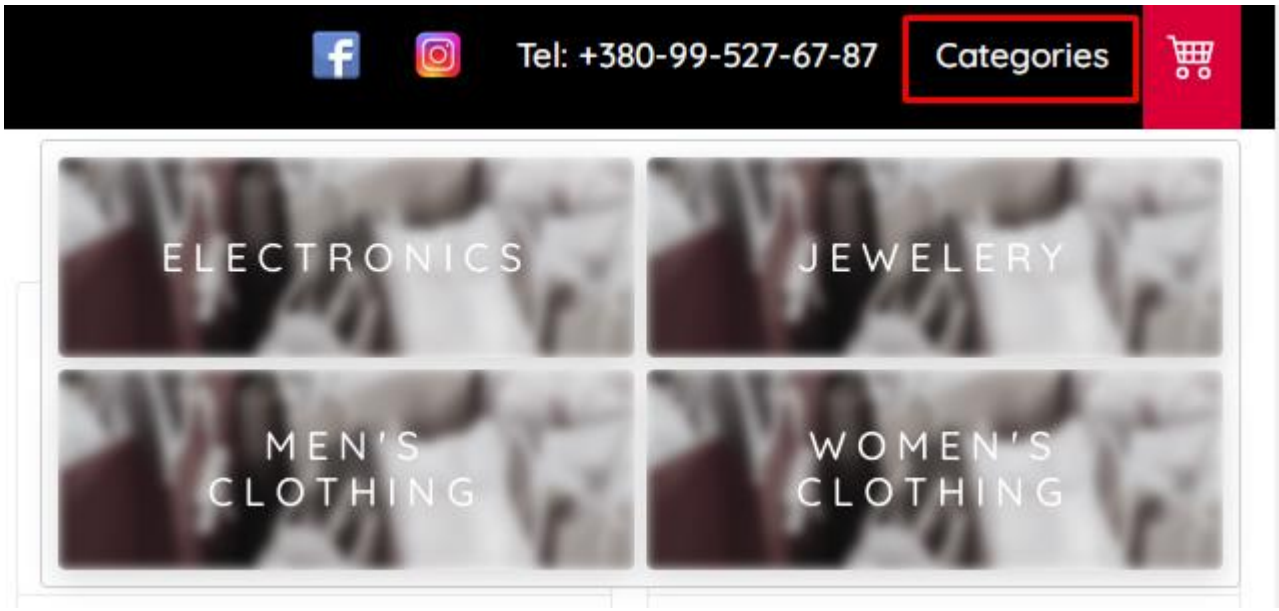

Рисунок 4.4 – Категорії товарів

При виборі будь-якої з категорій, вам будуть показані тільки ті товари, які входять до вибранної категорії. Наприклад, при виборі категорії “Electronics”, нам будуть доступні лише товари, які безпосередньо являються електронікою.

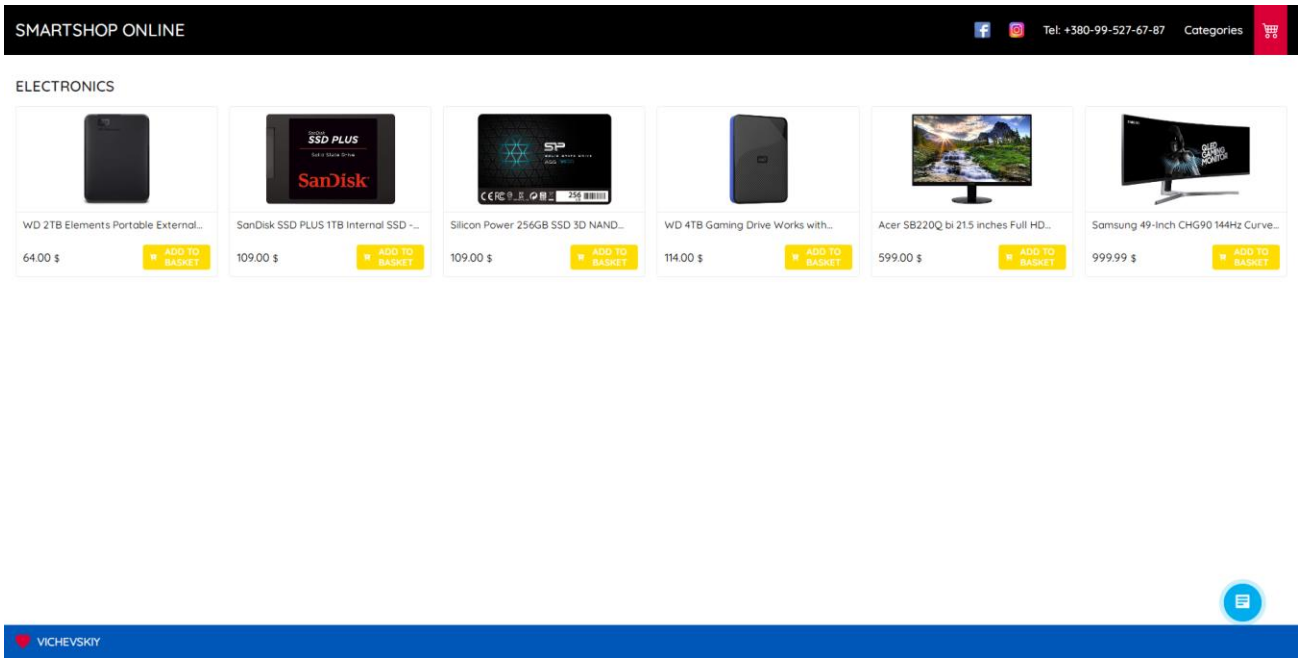

Рисунок 4.5 – Приклад вибору категорії

Вибравши будь-який товар зі списку товарів, можна подивитись більш детальну інформацію по цьому товару, як показано на рисунку 4.6.

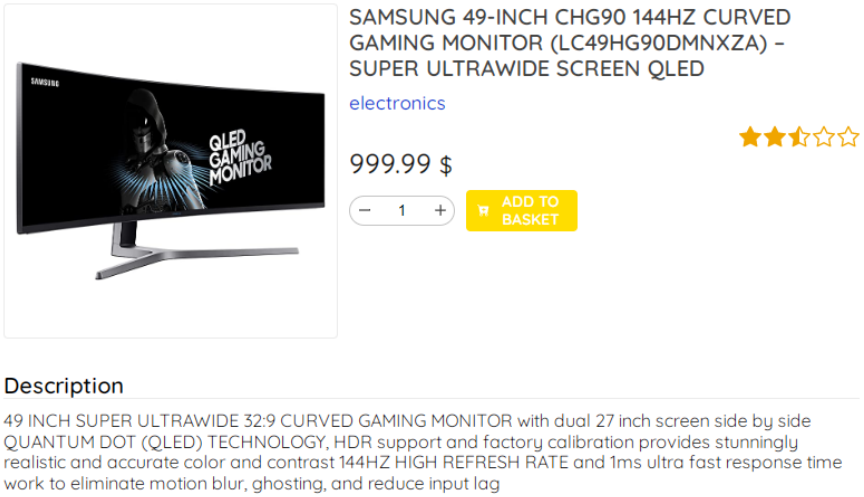

Рисунок 4.6 – Детальна інформація по товару

При виборі одного з товарі та натиснення на його іконку можна отримати його більш детальний опис, вибрати бажану кількість одиниць товару, побачити його рейтинг серед покупців та додати цей товар до корзини за допомогою кнопки за допомогою кнопки “add to basket”.

Є можливість обрати декілька товарі, обрати їх кількість та додати їх до кошика, де вони, в свою чергу, зберігаються як обрані та сумується їх вартість, як показано на рисунку 4.7.

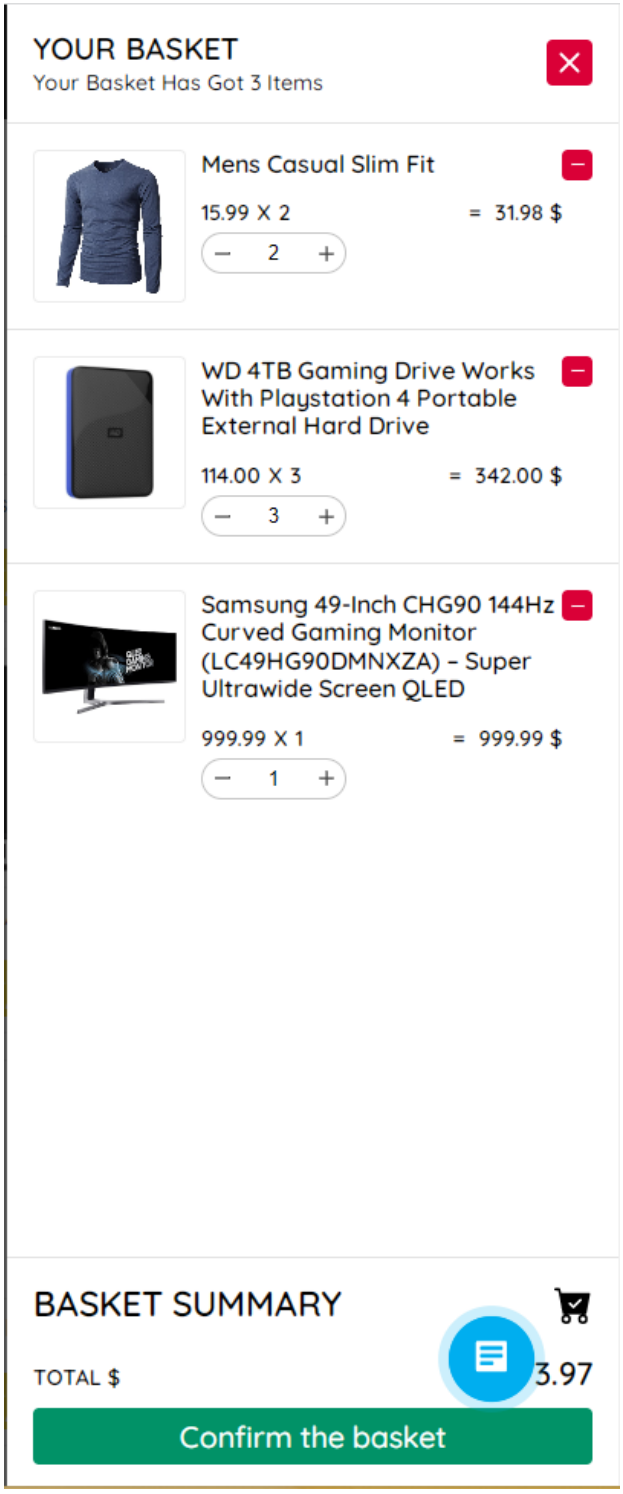

Рисунок 4.7 – Товари додані то кошика

Є також можливість видалити товари з кошика або ж зменшити чи збільшити кількість одиниць товарів, знизу можна виконати замовлення нажавши на кнопку “Confirm the basket”.

При натисканні на кнопку замовлення відкриваєтьсядодаткова форма, яку клієнт має заповнити, як показано на рисунку 4.8.

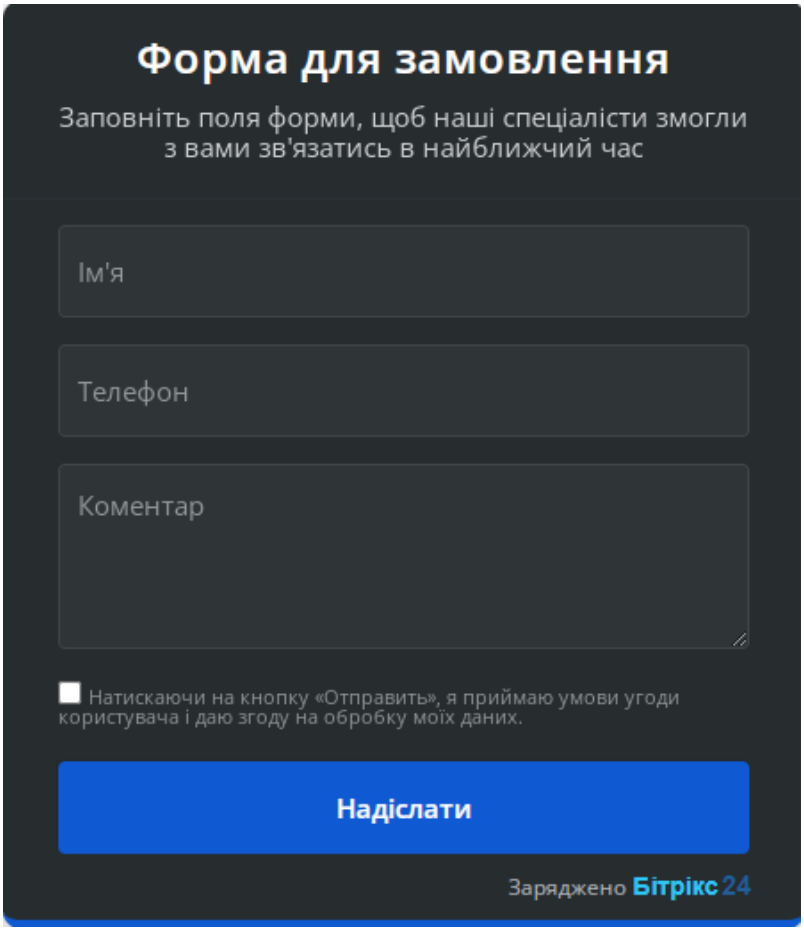

Рисунок 4.8 – Форма зворотнього зв’язку

Також на сайті є віджет, який розгортається, в ньому можна залишити заявку на когнсультацію або ж написати своє питання в lifechat.

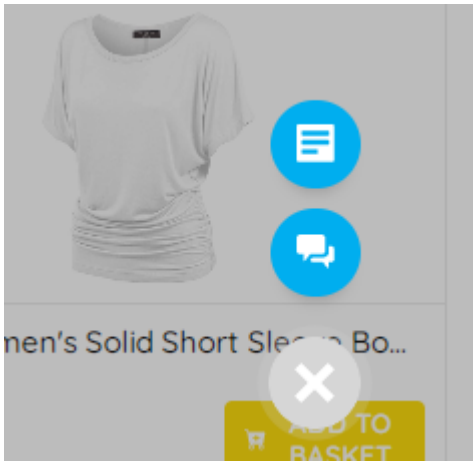

Рисунок 4.9 – Розгортання віджету

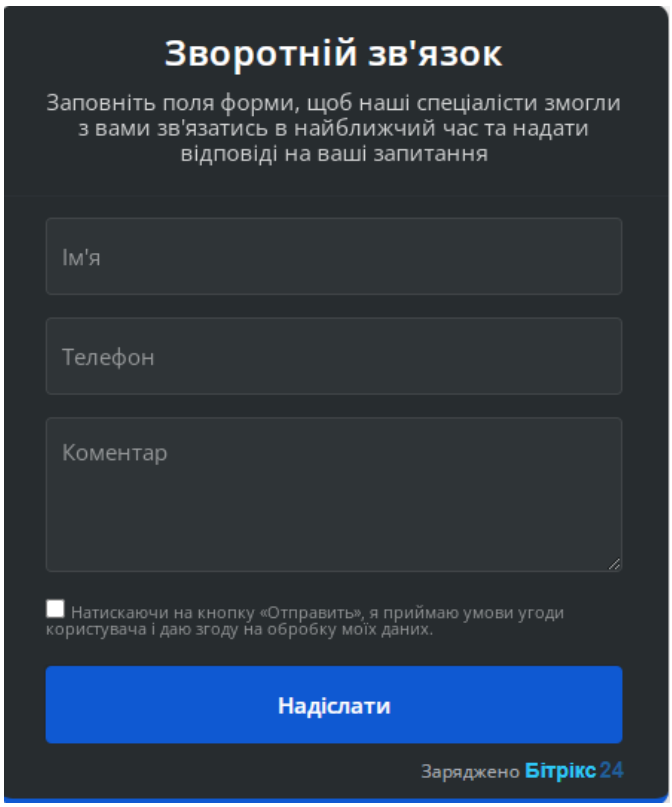

Рисунок 4.10 – Форма для консультації

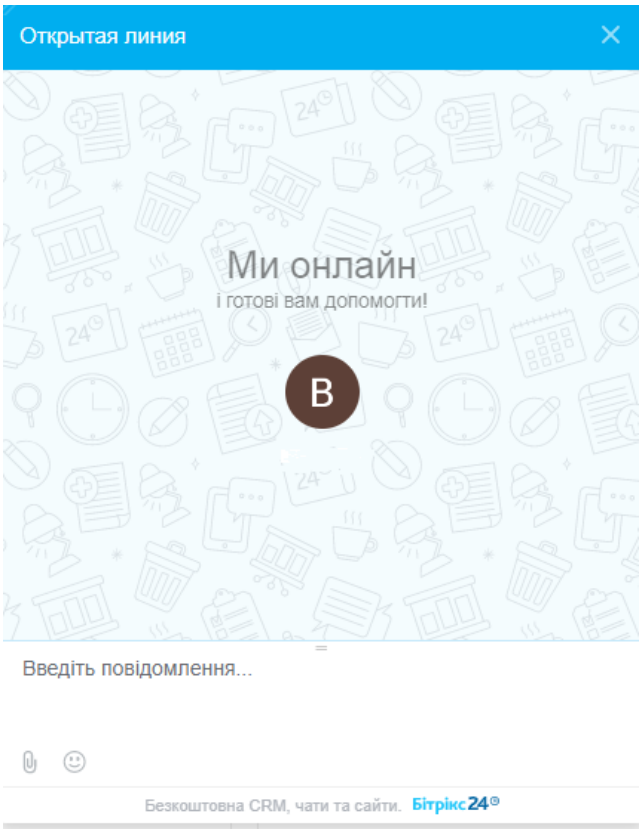

Рисунок 4.11 – Онлайн чат

Після заповнення будь-якої з форм або ж після написання повідомлення в чат запит передається на адміністративну панель в Bitrix24, де в подальшому, безпосередньо, адмыныстратор або ж менеджер обробляє дану заявку.

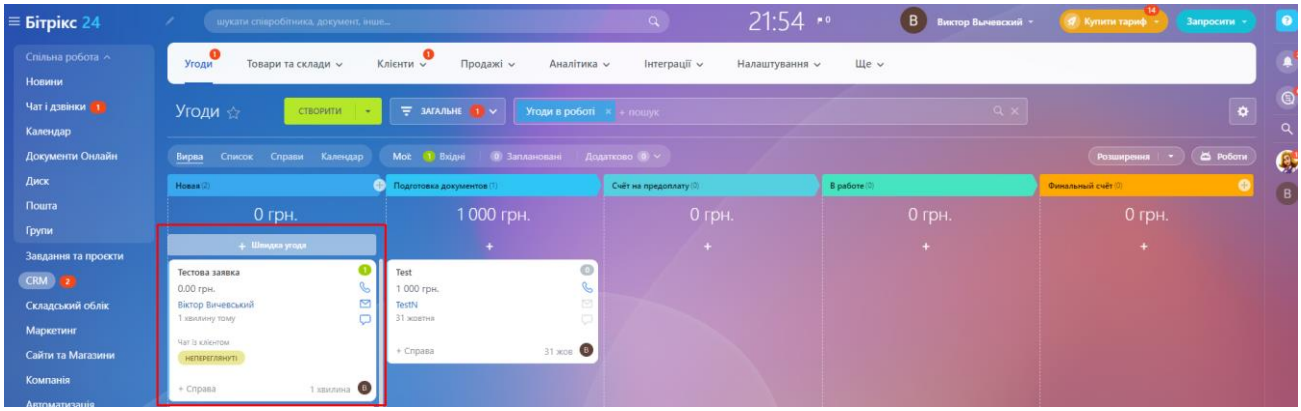

Рисунок 4.12 – Вигляд поданого запиту на адміністративній панелі Bitrix24

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було коротко описано переваги електронної комерції для продавців і їхніх клієнтів. Розглянуто основні етапи покупки, які проходить користувач, починаючи від процесу оформлення замовлення і закінчуючи доставкою товару.

Проаналізовано платформи — Shopify і Wix, які дозволяють створювати інтернет-магазини за дуже короткий період часу, що значно полегшує процес їхнього запуску і підтримки. Також був описаний процес обробки транзакції за допомогою платіжних шлюзів, проаналізовано і описано переваги і недоліки двох популярних платіжних шлюзів — Payline, Authorize.net та CRM систему (зокрема Bitrix24).

Практичним результатом роботи стала розробка програмного моделі електронної торгівлі в сфері B2C, а саме його веб-сторінки у вигляді інтернет магазину і системи адміністрування.

Було визначено основні вимоги до системи, що буде розроблятись, оскільки більшість схожим систем є платними, тестування не виявило проблем з програмним кодом або користувальницьким інтерфейсом веб-програми, отже можна стверджувати, що сайт є добре структурованим і працює задовільно з точки зору реалізації програмного забезпечення. Веб-програма проста у використанні та має інтуїтивно зрозумілий графічний користувальницький інтерфейс.

Для продавців і клієнтів важливим є досвід користування системою, тому і сторінка оформлення замовлення, і панель адміністратора повинні мати хороший UX. На сторінці оформлення замовлення повинна бути мобільна адаптація, оскільки значний відсоток замовлень у електронній комерції займає саме мобільна комерція.

Також варто збільшити зручність при введені інформації про доставку, наприклад пропонуючи можливі адреси під час введення інформації. Необхідно полегшити процес налаштування платіжних шлюзів, щоб їх зміна не займала

багато часу. Однією із загальних вимог є збереження процесу оформлення замовлення таким, який він є на стандартній сторінці оформлення замовлень.

Було проаналізовано та описано різні підходи до розробки веб-застосунків, архітектури, фреймворки та бібліотеки. Для даної роботи було вирішено використати тип SPA з використанням бібліотеки для створення користувацьких інтерфейсів React. У якості менеджера станів у вебзастосунку буде використано Redux, а для стилізації — styled-components.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Flanagan D. JavaScript: Definitive Guide / D. Flanagan. – Sebastopol: O'Reilly Media, 2021. – 982 p.

2. Wright R. Business-To-Business Marketing: A Step-By-Step Guide / R. Wright. – New Jersey: Prentice Hall, 2004. – 522 p.

3. Шиколенков Т. Ваш Інтернет-магазин від А до Я - актуальне керівництво / Т. Шиколенков. – Київ: АСТ, 2018. – 368 с.

4. Hogan B. HTML5 and CSS3: Level Up with Today's Web Technologies / B. Hogan. – Dallas, Texas: Raleigh, 2013. – 316 p.

5. Дронов В. React 17. Розробка веб-додатків на JavaScript / В. Дронов. – Books: Вінниця, 2018. – 360 с.

6. Morgan N. JavaScript for Kids: A Playful Introduction to programming / N.Morgan. – New-Jersy: John Wiley & Sons, 2016. – 288 p.

7. Фрімен Е. Вивчаємо програмування на JavaScript / Е.Фрімен, Е. Робсон. – New-York: O'Reilly, 2020. – 640 с.

8. Flanagan D. JavaScript: The Definitive Guide / D. Flanagan. – New-York: O'Reilly, 2020. – 320 с.

9. Колотилов Є. А. Продаж b2b. 101+ кейс / Є. А. Колотилов. – Київ: Ірис,2019. – 208 с.

10. Бритько В. Телеком Цілком. Системи продажу в B2C / В. Бритько. – Кировоград: Бізнес-психологія, 2018. – 192 с.

11. Мартін Р. Чистий код / Р. Мартін. – Харків: Вид. Фабула#Pro, 2019, - 416 с.