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 перший (бакалаврський)

Веб-застосунок для підприємства з виготовлення пластикових пакувань і суміжного устаткування

(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІ-21-3
Олександра БУКРЄЄВА
(власне ім’я, прізвище)

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

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

Освітня програма
Комп’ютерна інженерія
(повна назва освітньої програми)

Керівник: ст. викл. Олена СЕВОСТЬЯНОВА
(посада, власне ім’я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім’я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

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

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

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав.
кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Букрєєвій Олександрі Сергіївні
(прізвище, ім’я, по батькові)

1. Тема
Веб-застосунок для підприємства з виготовлення пластикових пакувань і суміжного устаткування
затверджена наказом по університету“ 26 ” травня 2025 р.№ 424 СТ
2. Термін подання здобувачем роботи до 17 червня 2025 р.
3. Вхідні дані до роботи
1) мова програмування – JS;
2) середовище управління базами даних – PostgreSQL;
3) Операційна система Windows

4. Перелік питань, що потрібно опрацювати у роботі
1) аналіз проблеми та огляд існуючих рішень;
2) вибір технології розробки (мови програмування) та інструментальних засобів (середовище розробки, программи для хостингу, бібліотеки тощо);
3) розробка взаємозв’язків між таблицями даних, налагодження зв’язку між клієнтською і серверної частинами за стосунку для збору даних;
4) детальний опис етапів створення веб-застосунку

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, ілюстраці

Слайд-презентація – 12 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	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих	06.05-08.05.2025	
	рішень		
2	Вибір технології розробки	09.05 – 11.05.2025	
3	Розробка та відлагодження ПЗ	12.05 – 21.05.2025	
4	Оформлення матеріалів роботи	22.05.2025	
5	Подання кваліфікаційної роботи, захист	23.05 – 24.05.2025	
6	Подання роботи на рецензування	25.06.2025	

Дата видачі завдання "05" травня 2025р.

Здобувач _____ (підпис)

Керівник роботи _____ (підпис) ст. викл. Олена СЕВОСТЬЯНОВА (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 53 с., 16 рис., 12 джерел, 1 дод.

ВЕБ-ЗАСТОСУНОК, ВЕБ-РОЗРОБКА, REACT, FRONTEND, BACKEND, РЕЛЯЦІЙНА БАЗА ДАНИХ.

Метою кваліфікаційної роботи є розробка веб-застосунку для одного з харківських підприємств.

У ході виконання кваліфікаційної роботи було розглянуто приклади веб-застосунків підприємств з виготовлення пакування; обговорено побажання щодо веб-застосунку із Замовником; описано етапи розробки веб-застосунку; розроблено клієнтську частину застосунку (наверстано макет застосунку із базовими стилями, розбито його на React-компоненти, розроблено логіку взаємозв'язку модулів і налаштовано навігацію по посиланнях у щапці веб-застосунку; було розроблено компонент форми для збору даних щодо замовлення); було розроблено серверну частину застосунку - створено базу даних для зберігання зібраних даних у структурованому вигляді, під'єднано її до форми і перевірено їх взаємозв'язок.

ABSTRACT

Bachelor’s thesis: 53 pages, 16 figures, 12 sources. 1 appendice

WEB APPLICATION, WEB DEVELOPMENT, REACT, FRONTEND, BACKEND, RELATIONAL DATABASE.

The major goal of this thesis was to develop a web application for one of the Kharkiv enterprises.

While working on the thesis, examples of web applications of packaging manufacturing enterprises were considered; wishes for the web application were discussed with the Customer; the stages of web application development were thoroughly described; the client (Frontend) part of the application was developed (the application layout with basic styles was created, it was divided into React components, the logic of module interconnection was developed and navigation by links in the web application panel was configured; a form component was developed for collecting order data); the server (Backend) part of the application was developed - a database was created to store the collected data in a structured form, it was connected to the form and their relationship was checked.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

Н.п. – наприклад

ПЗ – програмне забеспечення

ПК – персональний комп’ютер

СУБД (СКБД) – система управління (керування) базами даних

ACID - Atomicity, Consistency, Isolation, Durability (атомарність, узгодженість, ізольованість, довговічність) - набір властивостей, що гарантують надійну роботу транзакцій бази даних

CMS – Content Management System, система керування вмістом; програмне забезпечення для організації веб-застосунків чи інших інформаційних ресурсів в Інтернеті чи окремих комп'ютерних мережах

CSS - Cascading Style Sheets (Каскадні Таблиці Стилів) – мова стилізації у веб-програмуванні

HTML – Hypertext Markup Language (Мова Розмітки Гіпертексту)

JS – JavaScript

MPA - Multiple Page Application (багатосторінковий додаток)

MVCC - MultiVersion Concurrency Control (Керування паралельним доступом за допомогою багатоверсійності) - метод керування паралельним доступом, який зазвичай використовується СКБД та в мовах програмування, щоб реалізувати технологію транзакційної пам’яті

MVP – Minimal Viable Product (мінімальний робочий продукт)

PHP - Hypertext Preprocessor (препроцесор гіпертексту)

SEO - Search Engine Optimization - пошукова оптимізація

SPA - Single Page Application (односторінковий додаток)

UI – User Interface

UX – User Experience

ВСТУП

У сучасному світі складно уявити бізнес, який не мав би власної веб-сторінки, веб- або мобільного застосунку тощо. Там, де малі бізнеси можуть обмежитись веденням сторінок у соціальних мережах для переконання у цінності свого продукту чи послуги, виробництва і великі підприємства потребують індивідуального підходу – саме таким підходом є веб-застосунки. На даний момент існує багато сервісів, за допомогою яких людина, використовуючи шаблони, може створити веб-сторінку самостійно - без кодування або інших специфічних знань. Серед них особливо виділяється WordPress – система управління контентом (CMS), яка, за даними W3Techs [1], використовується на 43,1% усіх сайтів у світі (станом на травень 2025). Ця платформа надає шаблони і плагіни, що дозволяє запускати блоги, інтернет-магазини та корпоративні сайти за лічені години. Разом із іншими конструкторами, як-от Wix або Squarespace, WordPress робить веб-розробку доступною для малого бізнесу та приватних осіб. Однак ці рішення не покривають ще близько 29% ринку – існує велика кількість веб-застосунків, які розроблені без використання сторонніх сервісів через специфічні потреби замовника; також в мережі досі є достатня кількість застарілих проектів, які потребують підтримки або перероблення. Саме тут набуває значення розуміння основ програмування та можливість створювати і підтримувати веб-застосунки без допомоги сторонніх сервісів. Написання власного коду дозволяє розробникам повністю контролювати функціонал, адаптуючи продукт під унікальні потреби бізнесу, уникати вразливостей, властивих популярним CMS, і мати можливість маштабувати проект - адже архітектура, побудована з урахуванням майбутнього розширення, забезпечить стабільність веб-застосунку зі зростанням кількості користувачів. Метою роботи розробка веб-застосунку із базою даних для демонстрації і продажу продукції одного з харківських підприємств.

1 ПОСТАНОВКА ЗАВДАНЬ ПРОЕКТУ

1.1 Наведенння бажань замовника

Наразі, клієнти знаходять інформацію щодо підприємства за допомогою сторонніх сервісів інтернет-реклами, н.п. порталу ua-region.com, і залишають замовлення по телефону або за допомогою електронної пошти чи факсу. Менеджером підприємства було поставлено кілька умов щодо проекту власного веб-застосунку:

- Застосунок повинен містити в собі розділи з інформацією щодо підприємства і виготовлюваної ним продукції (пластикове пакування і устаткування для виготовлення пластикового пакування);
- Замовник повинен мати можливість зробити замовлення крізь застосунок;
- Інформація щодо замовлення повинна зберігатись за допомогою серверних потужностей Замовника, тобто із використанням приватного сховища даних, або бази даних на комп'ютері Замовника;
- Менеджер, який приймає замовлення, повинен мати впорядкований доступ до поточних замовлень, які були зроблені крізь застосунок, а також до історії замовлень користувачів;
- Розробка, тестування і хостинг проекту повинні бути виконані повністю з використанням безоплатних сервісів, адже замовник не передоставляє для виконання цього проекту жодного бюджету.

Замовник не ставив ніяких умов щодо зовнішнього вигляду проекту, адже в підприємства немає свого визначеного бренду. Для мінімального робочого продукту (MVP – Minimal Viable Product) було запропоновано приділити увагу у першу чергу працездатності проекту, а саме аспекту збору даних щодо замовлення від користувача. Зовнішній вигляд і додаткові функції проекту, такі як локалізація англійською мовою, оплата замовлень

онлайн і генерування звітів за даними замовлень планується опрацювати разом із замовником вже поза рамками кваліфікаційної роботи.

1.2 Огляд аналогів проекту

В якості аналогів було розглянуто 2 веб-застосунки інших підприємств, які займаються роздрібним і оптовим продажем власної продукції, а саме веб-застосунки компаній AVI PACK [2] і Алфаінтерпласт [3]. Слід зазначити, що розглядатиметься тільки клієнтська (Frontend) частина застосунків, тобто їх зовнішній вигляд, зміст і структура, адже доступу до інформації про структури баз даних цих веб-застосунків, або про способи зв`язку між клієнтською і серверною частинами цих застосунків, немає.

Веб-застосунок підприємства AVI PACK має стандартну для сучасних веб-застосунків верстку, яка починається із навігаційної секції, або header (“шапки”), і “секції-героя” або hero section, наведеної нижче на рисунку 1.1:

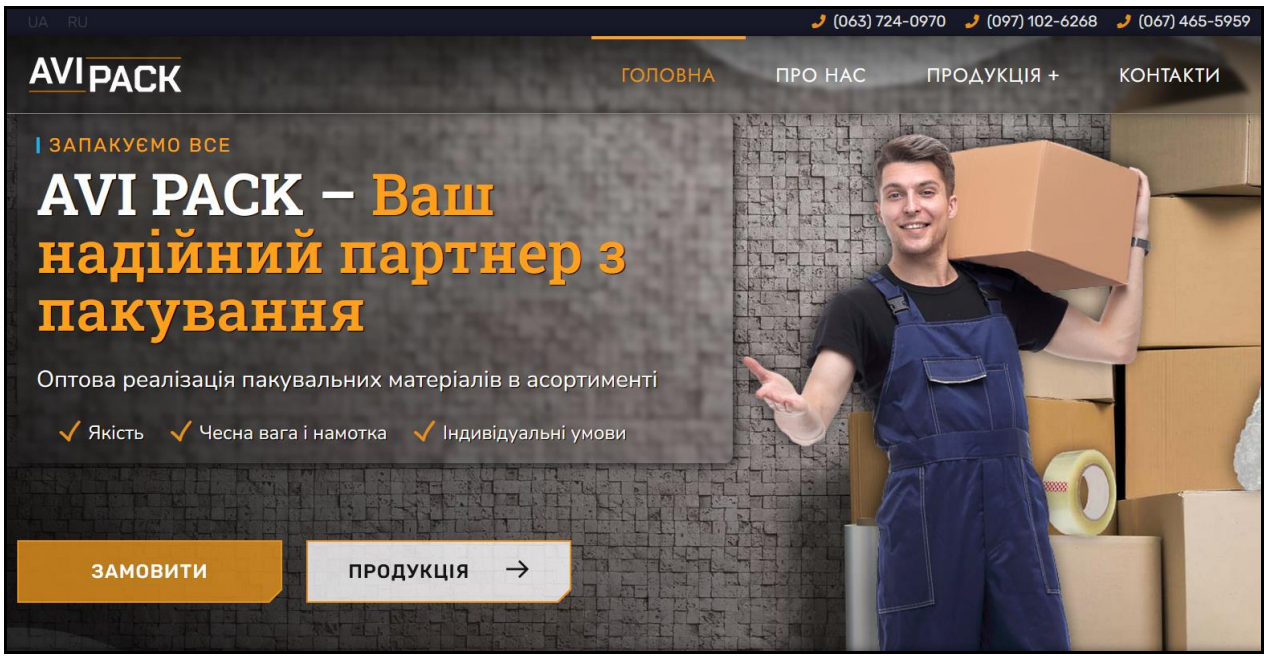

Рисунок 1.1 – початок головної сторінки веб-застосунку компанії AVI PACK

Мета навігаційної секції – це, по першу чергу, навігація. У навігаційній секції розміщується головне меню з посиланнями на основні розділи сайту

("Головна", "Про нас", "Продукція", "Контакти"). Це дозволяє користувачам швидко переміщатися між сторінками. Ця секція також містить логотип компанії, який дуюлює посилання на головну сторінку при натисканні.

Секція-герой виконує кілька ключових функцій, які роблять її важливою для ефективності сайту.

По-перше, ця секція привертає увагу відвідувача з перших секунд за допомогою великого заголовку і анімації (привітний пакувальних “виїзджає” з правої сторони екрану). Вона містить стислий заголовок, підзаголовок та призив до дії (кнопки “Замовити” і “Продукція”), що дозволяє швидко донести основну ідею сайту або зекономити час клієнтові, якщо він хоче одразу зробити замовлення. Це критично важливо, адже користувачі часто приймають рішення про подальшу взаємодію з сайтом за лічені секунди. Іншою важливою функцією секції-героя є допомога в індексації сайту. Використання ключових слів у заголовках та підзаголовках Hero-секції покращує позиції сайту в пошукових системах. Наприклад, заголовок "Оптова реалізація пакувальних матеріалів в асортименті" містить важливі для SEO фрази.

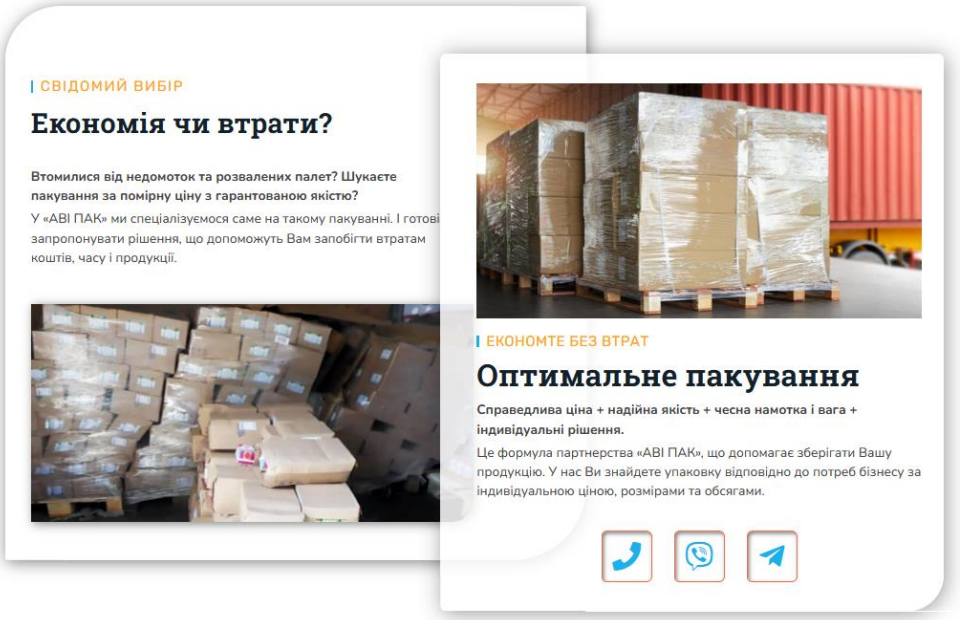

Рисунок 1.2 – Секція, яка слідує за секцією-героєм головної сторінки веб-застосунку компанії AVI PACK

За секцією-героєм слідує велика стилізована секція, ціль якої – переконати користувача у надійності і чесності продавця.

На мою думку, це не є вдалим рішенням, але подібний варіант стилізації текстової секції виглядає стримано і привабливо. У цій секції є заклик до зв`язку із продавцем, адже в ній є інтерактивні кнопки для зв`язку крізь меседжери, що також є перевагою, адже в замовників часто виникають питання під час ознайомлення із застосунком.

Наступна секція, у якій демонструються категорії доступних до замовлення виробів, представлена за допомогою slider (рисунок 1.3):

Рисунок 1.3 – Демонстрація доступних до замовлення виробів на головній сторінці веб-застосунку компанії AVI PACK

В секції такого типу є потенціал у разі, якщо веб-застосунок переглядається клієнтами з мобільного пристрою – слайдер дозволить клієнту легко “пролистувати” варіанти доступних товарів пальцем. Цей варіант верстки варто використовувати у разі, якщо застосунок верстається з урахуванням принципів адаптивності.

Варто зауважити, що slider в цьому випадку використовується лише для швидкої демонстрації, адже у веб-застосунку присутній набір сторінок для кожного виду продукції, що пропонується, кожну з яких можна відкрити

за посиланням у елементах slider, або за допомогою випадаючого списку при кліку на слово “Продукція” у навігаційній секції.

Наступна секція, наведена нижче на рисунку 1.4, надає стислу інформацію щодо доставки і замовлення виробів, які є у наявності, з повторним закликом до зв'язку (кнопки із логотипами месенджерів у правому верхньому кутку є інтерактивними і привертають до себе увагу за допомогою анімації – табло із кнопками рухається під час наведення на нього курсору).

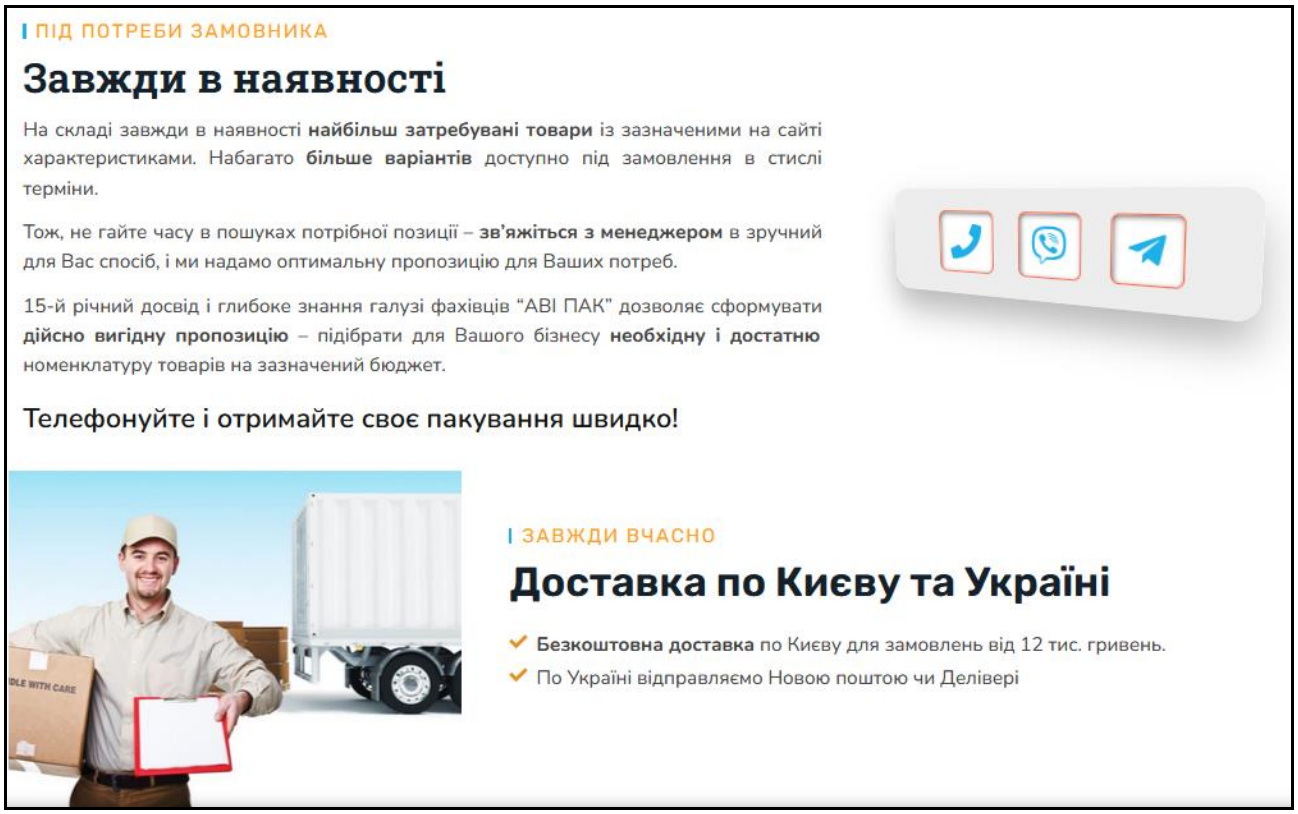

Рисунок 1.4 – Рекомендації щодо замовлення і стисла інформація щодо доставки на головній сторінці веб-застосунку компанії AVI PACK

Секція під назвою “Чому ми?”, наведена на рисунку 1.5, стисло розповідає про переваги співпраці із даним підприємством. Вона не є інтерактивною, але має приємний дизайн і робить веб-застосунок більш привабливим, що впливає на довіру клієнтів.

Покращити цю секцію можна, якщо додати до неї нову інфомацію або посилання на сторінку “Про нас”, де демонструються сертифікати компанії

та інші відомості щодо підприємства. Наразі до вищезгаданої сторінки можна перейти лише із навігаційної секції.

Нижніми секціями (рисунок 1.6) головної сторінки веб-застосунку є секція із повторним закликом до зв`язку, і футер (footer).

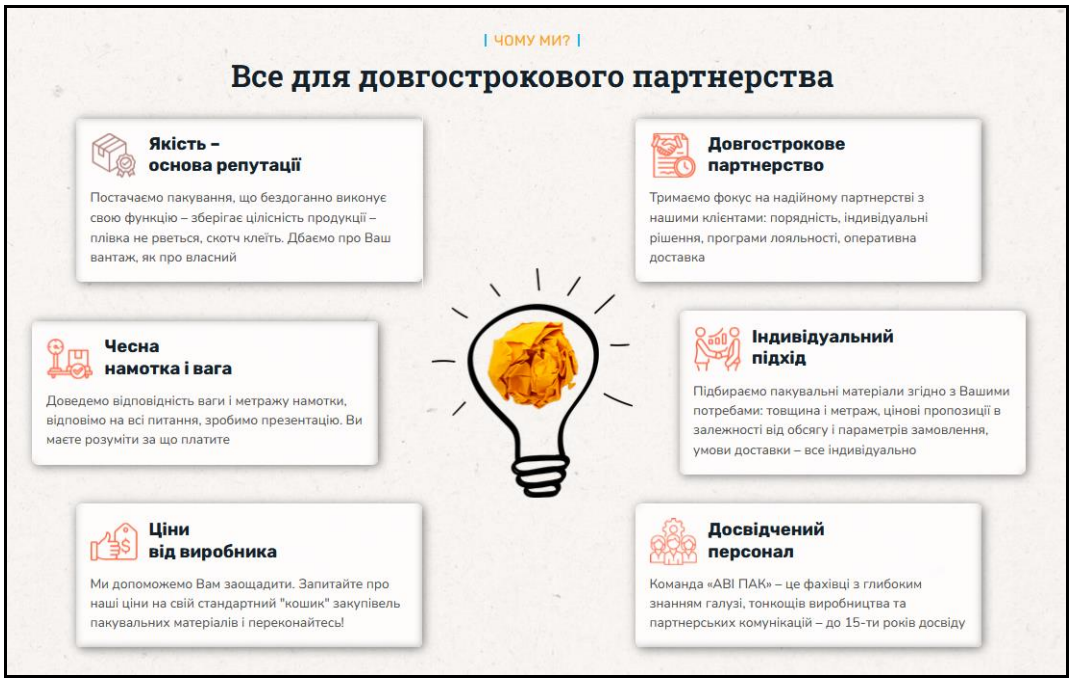

Рисунок 1.5 – Текстова секція “Чому ми?” на головній сторінці веб-застосунку компанії AVI PACK

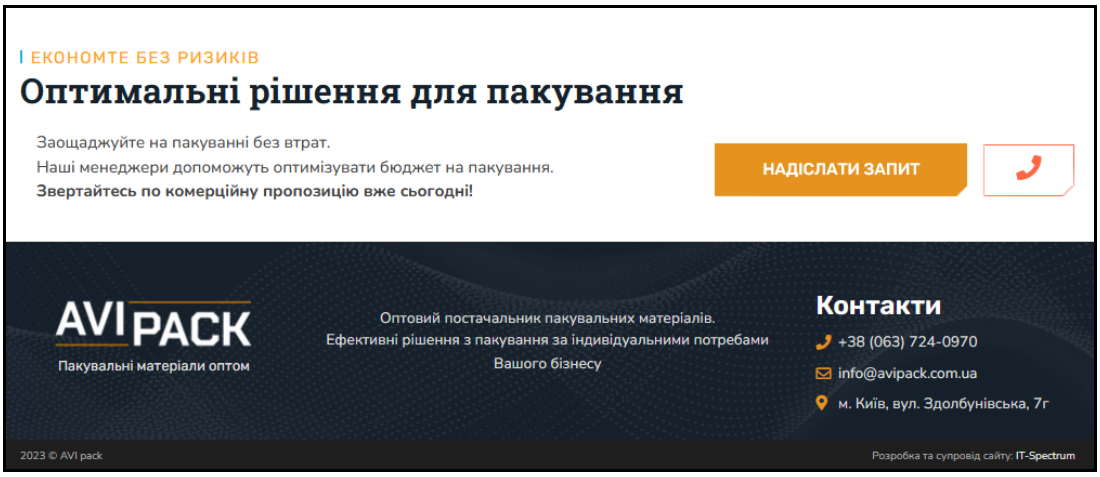

Рисунок 1.6 – Секція із додатковим закликлм до зв`язку і footer на головній сторінці веб-застосунку компанії AVI PACK

Футер - обов`язковий елемент кожної сторінки, який використовується

при індексації веб-сторінок, а також як додаткове місце для посилань, наведення контактних або юридичних даних компанії, тощо.

Сторінка “Продукція” (рисунок 1.7) демонструє продукцію, яку виробляє підприємство, у форматі блоків, де кожен блок є посиланням на окрему сторінку з описом продукту.

Рисунок 1.7 – Секція “Наша пропозиція” на сторінкці “Продукція” веб-застосунку компанії AVI PACK

Це є популярним і охайним способом представити продукцію, адже клієнт може швидко ознайомитись із ассортиментом без зайвої інформації, і обрати продукт, який потрібен саме йому. Під секцією “Наша пропозиція” сторінка “Продукція” дублює в собі навігаційну панель, а також секцію-герой і інші секції, наведені вище – це рішення може бути обумовлене змогою покращити індексацію веб-застосунку – за умови вірного підбору

заголовків секцій, у результатах пошуку в браузері з`являтиметься не тільки головна сторінка застосунку, а і всі інші сторінки

Сторінки з детальним описом продукції (рисунок 1.8) також дублюють в собі секції із головної сторінки, але мають іншу секцію-героя і містять багато додаткової інформації про конкретний вид продукції.

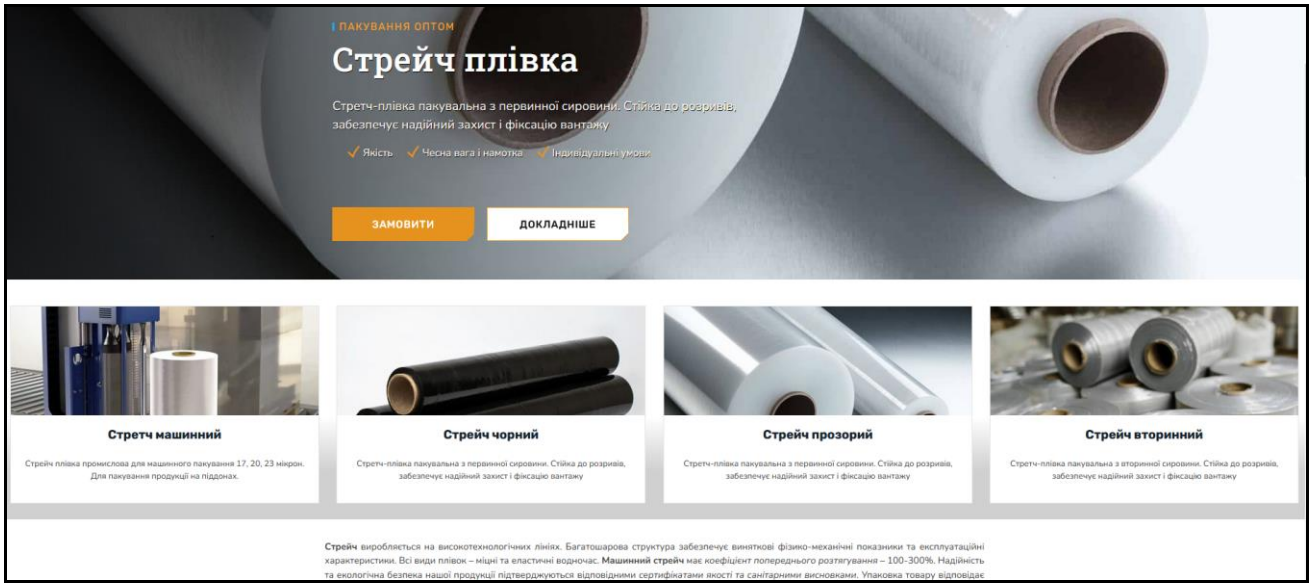

Рисунок 1.8 – Сторінка з детальним описом стрейч-плівки веб-застосунку компанії AVI PACK

Такі сторінки мають кілька переваг – по перше, вони ще більше покращують індексацію веб-застосунку. По-друге, вони дозволяють докладніше описати характеристики певного виду пропонованої продукції, що було б недоцільним на головній сторінці або на сторінці із багатьма видами продукції.

Рисунок 1.9 – Сторінка “Контакти” веб-застосунку компанії AVI PACK

Сторінка “Контакти” (рисунок 1.9) не копіює секції з головної сторінки, і має єдину секцію з формою і віджетом карти Google Maps з адресою підприємства, яка включена в розмітку за допомогою тегу iframe.

Веб-застосунок компанії AVI PACK має задовільні показники індексації, адже вибивався в результатах пошуку на першій сторінці в Chrome і Firefox. Застосунок також має охайний дизайн із приємною палітрою, використанням анімацій, і постійні, але ненав`язливі заклики до контакту з продавцем, що є ефективною тактикою для доведення клієнту до замовлення по телефону, або для консультації клієнта, якщо він не впевнений у своєму виборі чи потребує допомоги із великим асортиментом продукції..

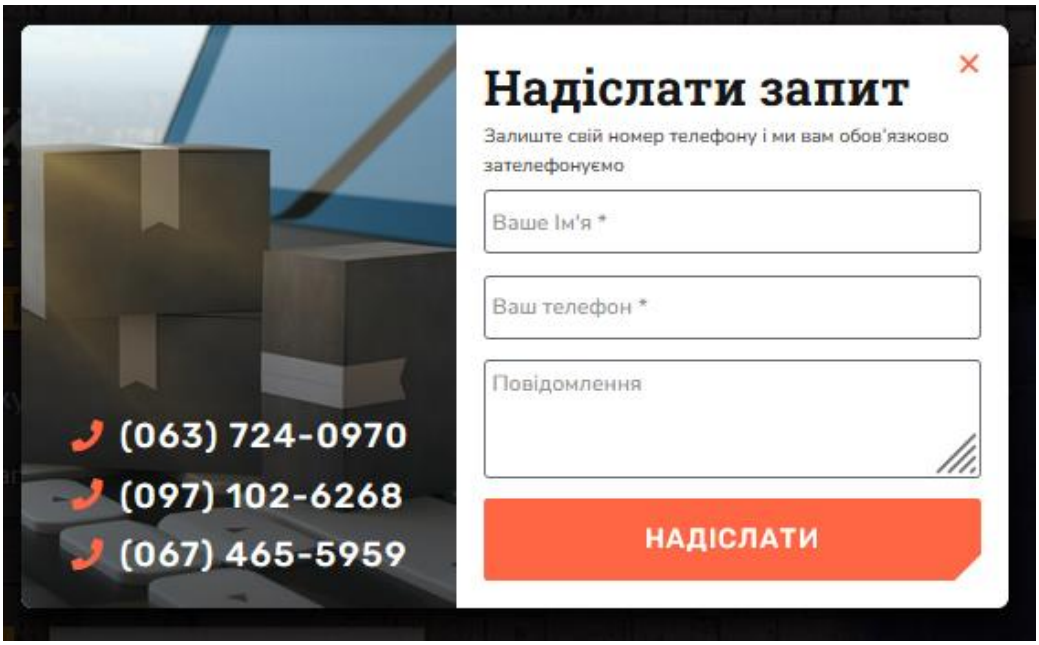

Рисунок 1.10 – Модальне вікно із формою для зв`язку на головній сторінці веб-застосунку AVI PACK

Недоліком сайту є його обмежений функціонал – неможливо замовити конкретну продукцію напряму крізь застосунок. Відправка повідомлення або зв`язок із продавцем крізь месенджери не підходять для замовника проекту, що розглядається під час цієї передатестаційної практики. Замовник має

потребу в базі даних, де зберігатимуться дані щодо клієнтів і замовлень.

Другим аналогом проекту був веб-застосунок компанії Алфаінтерпласт. Навігаційна секція, секція герой і інформаційні секції на головній сторінці є схлжими на представлені вище, але цей веб-застосунок має додатковий функціонал – можливість залишити повідомлення напряму крізь застосунок за допомогою форми, можливість зареєструвати свій особистий кабінет і зберігати дані про замовлення тощо. Всі функції доступні крізь навігаційну панель застосунку.

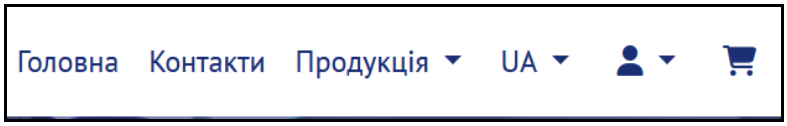

Рисунок 1.11 – Частина навігаційної панелі у веб-застосунку компанії Алфаінтерпласт

Під час оформлення замовлення клієнт заповнює кілька форм – форму “Покупець” (рисунок 1.12), “Спосіб доставки”, “Адреса доставки” тощо.

Рисунок 1.12 – Форма “Покупець” у розділі “Кошик” веб-застосунку

За допомогою цих форм користувач може зручно залишити замовлення крізь за стосунок, а продавець отримає структурований масив даних, який легко обробити.

Єдиним недоліком є неможливість оплати замовлення напряму крізь застосунок – замість цього користувачеві пропонується оплата за допомогою банківського переказу (рисунок 1.13).

Рисунок 1.13 – Інші форми розділу “Кошик” веб-застосунку компанії Алфаінтерпласт

Комбінування певних рішень щодо дизайну першого аналогу (окремі сторінки опису продукції, способи покращення індексації тощо) і функціоналу другого аналогу (замовлення крізь форми) дозволяють створити продукт, який відповідає вимогам замовника.

1.3 Постановка завдань проекту

Основним завданням проекту є створення веб-застосунку для демонстрації продукції підприємства і збору даних щодо замовлень.

Клієнтська частина MVP-версії веб-застосунку повинна містити у собі такі компоненти:

- Елементи, необхідні для покращення індексації веб-застосунку – шапку, секцію-героя, футер тощо;
- Головну сторінку із розділами, в які буде додано загальну інформацію щодо підприємства і виготовлення запропонованої продукції;
- Сторінку із розділами, в які буде додано опис видів пакування, яке виготовляє виробництво;
- Сторінку з контактами виробництва;
- Сторінку для подачі клієнтом заявки на замовлення із формами, в яких вказиватимуться конкретні дані щодо замовлення – розміри пакування, його вид, кількість одиниць товару, а також дані для доставки і контактні дані клієнта;

Вдосконалена версія проекту включатиме в себе особистий кабінет користувача з аутентифікацією, в якому клієнт зможе продивитись історію своїх замовлень, а також використовувати шаблони для повтору замовлень без повторного заповнення форм, другий під-застосунок для менеджера підприємства, в якому менеджер переглядатиме нові замовлення і коригуватиме стан поточних замовлень. Також під час розробки веб-застосунку потрібно врахувати додання оплати замовлення за допомогою онлайн-сервісів (e-commerce);

Єдиною умовою щодо бази даних є її структурованість, адже в майбутньому планується надати менеджерові можливість генерувати облікові звіти по замовленнях за визначений термін, а також надавати користувачеві доступ до його історії замовлень.

1.4 Етапи розробки веб-застосунку

На даний момент, будь-який веб-застосунок, окрім суто інформаційних ресурсів, складається з двох частин: зовнішньої частини, або Frontend, яка показується користувачеві в браузері і з якою користувач взаємодіє напряму, і внутрішньої частини, або Backend, яка зберігається на сервері і містить у собі всі дані, закладені в за стосунок. Frontend частина створюється за допомогою мови програмування Javascript і похідних від неї, а також HTML (Hypertext Markup Language, мова розмітки гіпертексту) і CSS (Cascading Style Sheets, каскадні таблиці стилів). Серверна частина описується за допомогою таких мов програмування, як Ruby, Python, PHP, Java тощо.

Основні етапи створення веб-застосунку із подобним описом, наведені нижче – деякі етапи були спрощені, а деякі відкинуті в рамках цієї роботи через стислі строки, але є обо'вязковими у разі виконання комерційного проекту з визначеним бюджетом.

1.4.1 Аналітика

На цьому етапі проводилося дослідження потреб компанії, збір інформації і аналіз рішень конкурентів Замовника – інформацію, яку було отримано із досліджень, наведено в розділі "Огляд аналогів проекту".

Проводилися обговорення із Замовником, протягом яких визначаються основні вимоги і побажання щодо проекту, його функціонал тощо. На даному етапі було важливо вирішити, які цілі має виконувати веб-додаток – для реалізації яких бізнес-завдань він потрібний. У випадку даного проекту, веб-застосунок в майбутньому стане інтернет-магазином, і матиме структуру, типову для інших застосунків e-commerce. Результатом обговорень на даний момент є вимоги, наведені в розділах 1.1 (Наведення бажань замовника).

Було також сформовано портрет потенційного користувача веб-застосунку – ним наймовірніше є ФОП або представник юридичної особи

(підприємства з виготовлення іншої продукції, яка потребує пластикового пакування), рідше фізична особа.

Такий користувач найімовірніше робитиме замовлення за допомогою комп’ютера, а не мобільного пристрою, тому верстка проекту була орієнтована в першу чергу під великий екран (Desktop-first).

1.4.2 Планування

На цьому етапі зазвичай створюється документація до проекту, визначаються технології, які будуть у ньому використані, продумуються стилі сторінок і конкретних елементів, в залежності від їх призначення, а також застосунку в цілому. На етапі планування збирається команда фахівців, які будуть залучені в роботі над різними частинами проекту – але враховуючи, що даний проект виконується лише однією людиною, і що в даного проекту немає бюджету або визначених строків виконання (маються на увазі строки співпраці із Замовником, а не строки виконання кваліфікаційної роботи), етап планування було значно спрощено.

Під час планування обиралися технології створення веб-застосунку, які описані у розділі 2, був створений список елементів, які є обов’язковими для MVP, розписані вимоги до MVP – ці вимоги наведено у розділі 1.3 (Постановка завдань проекту).

Також було прийнято рішення щодо архітектури проекту – було обрано архітектуру SPA (Single Page Application), або односторінкову архітектуру. Ця архітектура є більш сучасною за MPA (Multiple Page Application, або многосторінкову архітектуру) і має ряд переваг, таких як порівняно низке використання трафіку при більш швидкому завантажені клієнтської частини при навігації, і розвантаження серверної частини через зменшену кількість запитів до серверу. Це стало основною причиною вибору цієї технології, адже замовник планує хостинг веб-застосунку зі свого ПК.

В технології SPA є один значний недолік – зниження показників SEO

ри, що може призвести до проблем з вибитті веб-застосунку у результатах пошуку у браузері. Методи виправлення цієї проблеми не розлядаються в рамках цієї кваліфікаційної роботи.

1.4.3 Прототипування

На етапі прототипування робиться “нарис” майбутнього веб-застосунку. Етап проектування призначений для визначення структури всіх компонентів та елементів застосунку, його функціональності, виду інтерфейсу та особливостей навігації веб-застосунком. Прототипування може бути зроблене різними способами – від буквального нарису сторінок і елементів веб-застосунку, до використання спеціальних сервісів, таких як Protoio, Canva, Figma, тощо. Це дозволяє Замовникові побачити, як виглядатиме готовий продукт, отримати уявлення про роботу веб-застосунку і внести необхідні правки.

Проектування необхідне в першу чергу для перевірки працездатності ідеї, а також пошуку і виявленні проблемних місць проекту – винайдення проблем на етапі проектування значно економить час на майбутніх етапах, особливо на етапі розробки Frontend-частини застосунку. Проектування також використовується для більш точного прорахування вартості проекту.

Для проектування даного проекту не використовувалися сторонні сервіси, адже через стислі строки виконання ознайомлення із подібними сервісами зайняло б забагато часу. Замість цього, прототипування клієнтської частини проводилося на папері.

1.4.4 UX, UI-дизайн

UI-дизайн охоплює всі кольори, типографіку та зображення, які користувач бачить на екрані, а також елементи, які він використовує для навігації інтерфейсом, такі як кнопки, смуги прокручування та дії з

проведенням пальцем, тощо.

На відміну від дизайну користувацького інтерфейсу, який зосереджений виключно на розробці комп'ютерного інтерфейсу, UX-дизайн охоплює всі аспекти сприйняття користувачем досвіду взаємодії з продуктом або веб-сайтом, такі як його зручність використання, корисність, бажаність, сприйняття бренду та загальна продуктивність. UX-дизайн також є елементом клієнтського досвіду (CX) та охоплює всі аспекти дизайну та етапи проектування, пов'язані з клієнтським досвідом.

Створення UX (User Experience, досвіду користувача) і оформлення UI-дизайну (User Interface Design) у комерційних проектах є окремими етапом розробки, адже потребують окремих спеціалістів – дизайнерів, і специфічних вмінь роботи із сервісами графічної розробки – раніше для цього найчастіше використовувалися програми Adobe, наприклад Photoshop і Adobe Illustrator, але наразі найпопулярнішими інструментами для UI-дизайну є Figma і Canva.

Рисунок 1.14 – Популярні інструменти для -дизайну

У рамках кваліфікаційної роботи розробка UX полягала в аналізі і приблизному копіюванні елементів дизайну і логіки веб-застосунків конкурентів з навчальною метою; UI-дизайн не розлядається зовсім, якщо не враховувати використання CSS-модулів для більш сприйнятного інтерфейсу проекту, адже Замовник ще не визначив вимог щодо брендування проекту.

1.4.5 Frontend

Цей етап розробки відповідає за створення клієнтської частини за стосунку, втілення у проекті частини того функціоналу, що був намічений у на етапах планування і прототипування, а також інтерфейсу і зовнішнього вигляду, який розроблявся UX/UI-дизайнерами. Тобто, створюється те, з чим користувачі можуть взаємодіяти безпосередньо: інтерфейс, вікна для введення символів, кнопки, навігація і посилання, анімовані елементи тощо.

Для розробки на етапі Frontend застосовуються три основні мови:

1) HTML (Hypertext Makrup Language, мова розмітки гіпертексту) – стандартизована мова розмітки документів для перегляду вебсторінок у браузері. Браузери отримують HTML документ від сервера за протоколами HTTP/HTTPS або відкривають його з локального диска, далі інтерпретують код в інтерфейс, який відображатиметься на екрані монітора. Елементи HTML є будівельними блоками сторінок HTML. За допомогою конструкцій HTML, зображення та інші об'єкти, такі як інтерактивні форми, можуть бути вбудовані у веб-застосунок. HTML надає засоби для створення структурованих документів, позначаючи структурну семантику тексту, наприклад заголовки, абзаци, списки, посилання, цитати та інші елементи. Елементи HTML окреслені тегами, написаними з використанням кутових дужок. Теги на кшталт <img /> чи <input /> безпосередньо виводять вміст на сторінку. Інші теги, такі як <p>, оточують текст і надають інформацію про нього, а також можуть включати інші теги як піделементи. Браузери використовують теги HTML для інтерпретації вмісту сторінки.

2) CSS (Cascading Style Sheets, каскадні таблиці стилів) – це спеціальна мова стилізування сторінок, що використовується для опису їхнього зовнішнього вигляду. CSS використовується, щоб визначити кольори, шрифти, та інші аспекти вигляду сторінки. Одна з головних переваг CSS — можливість розділити зміст сторінки (або наповнення, зазвичай

HTML, XML або іншу мову розмітки) від вигляду документу.

3) JavaScript (JS) – динамічна, об'єктно-орієнтована мова програмування. Використовується для створення сценаріїв вебсторінок, що надає можливість взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-застосунку. Тобто, саме мова JavaScript забезпечує реакцію компонентів на дії користувача - саме ця технологія відповідає за спливаючі повідомлення в момент наведення на них курсору, за можливість перетягувати та натискати на компоненти, вводити текст, вимагати команди тощо.

Для розгортання повноцінної клієнтської частини достатньо знати ці три мови, але на даний момент на основі JS розроблено безліч бібліотек і фреймворків (каркасів веб-застосунків), які спрощують розробку за рахунок автоматизації, і позбавляють від необхідності написання рутинного коду. Більшість каркасів спрощують доступ до БД, також зменшують дублювання коду. Для даного проекту використовується фреймворк React, який дозволяє розбити інтерфейс веб-застосунку на компоненти, і також велика кількість бібліотек – особливу увагу приділено бібліотеці Formik, адже саме за допомогою її розгорнуто форму для збору даних від користувача. Детальніше про React і Formik, а також інші використані технології, які стосуються Frontend, розписано у розділі 2.

1.4.6 Backend

На етапі Backend-розробки вирішується, яким чином Frontend-частина за стосунку взаємодіє із серверною частиною – які дані з сервера повертаються на запити користувача, і яким чином дані від користувача обробляються серверною частиною. Під час Backend-розробки застосовують різні серверні мови (Java, Python, Golang, тощо.), завдяки яким вдається побудувати потрібний функціонал і забезпечити обмін даними між сервером

та користувачем.

У рамках данного проекту використано середовище Node.js. Воно дозволяє надсилати запити між користувачем і сервером вигляді HTML-файлів, JavaScript-файлів, JSON-файлів тощо.

Обов’язковим елементом Backend є база даних, яка виконує функцію збереження і управління інформацією. Вибір структури БД залежатиме від потреб проекту – в даному проекті використовується реляційна база даних PostgreSQL, адже присутній зв’язок між даними користувача і даними, що стосуються його замовлень. Саме на етапі Backend-розробки закривається питання безпеки – за допомогою шифрування і авторизації налаштовується захист інформації користувачів, за допомогою фільтрації та валідації полів введення забезпечується захист від атак на скриптинг, SQL-ін’єкцій тощо. Одним із найефективніших методів запобігання атакам, що стосуються SQL-ін’єкцій, і методом який використано у рамках кваліфікаційної роботи, є очищення введення даних перед тим, як вони будуть використані в SQL запитах. Це включає в себе видалення або екранування спеціальних символів, таких як одинарні лапки (‘) і подвійні лапки (“), які можуть бути використані для впровадження шкідливого SQL коду.

1.4.7 Тестування

Тестування коду може проводитись локально, або на хостингу. Існують різні види тестування – регресивне (спрямоване на виявлення дефектів у вже протестованих елементах веб-застосунку), інтеграційним (тестується передача даних або взаємодія компонентів), навантажувальним (використовується здебільшого для тестування серверної частини додатку), може бути виконаним автоматично або вручну, стосуватися продуктивності, баз даних, сумісності з браузерами та гаджетами, інтерфейсу, веб-безпеки тощо. У рамках проекту під час розробки здебільшого використовується інтеграційне і регресивне тестування вручну.

2 ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

2.1 Технології Frontend – клієнтської частини за стосунку

2.1.1 React

Клієнтська частина застосунку розроблена з використанням бібліотеки React [4] - це бібліотека, яка дозволяє розробникам ефективно створювати високопродуктивні та легко підтримувані веб-додатки, використовуючи компонентний підхід та інші сучасні концепції.

React побудований на концепції реактивності. Під час створення додатку розробник не взаємодіє безпосередньо з DOM-деревом та не оновлює інтерфейс самостійно. Його завдання полягає в тому, щоб описати інтерфейс за допомогою компонентів (шаблону) та керувати зміною даних. React, при зміні даних, автоматично оновлює інтерфейс за заданим шаблоном.

React - мультиплатформний інструмент, з яким можна рендерити розмітку на сервері (Next.js), розробляти нативні (React Native) або десктопні (Electron) застосунки. При цьому синтаксис залишається однаковим, відрізняються лише використовувані інструменти. Це дозволяє створити динамічний та адаптивний інтерфейс, і у майбутньому легко адаптувати проект під десктопне застосування, якщо цього забажає Замовник.

Інтерфейс користувача складається з невеликих елементів, таких як кнопки, текст і зображення. React дозволяє об'єднувати їх у багаторазово використовувані компоненти - основні будівельні блоки React-застосунків, за допомогою яких інтерфейс розділяється на незалежні частини. Будь-який інтерфейс (дизайн) можна розбити на компоненти.

React-застосунок можна уявити як дерево компонентів. На верхньому рівні стоїть кореневий компонент, у якому вкладена довільна кількість інших

компонентів. Кожен компонент повертає розмітку залежно від стану інших компонентів, переданого props (скорочено від properties – об’єкт властивостей) і внутрішньої логіки, тим самим вказуючи, який HTML потрібно відрендерити в DOM.

React забезпечує компонентний підхід, де кожен елемент (наприклад, форма замовлення, елемент з інформацією про вид пакування) є незалежним модулем з власним станом і логікою. В подальшому, це значно спростить рендер елементів, які можуть динамічно змінюватись, наприклад відображення переліку видів пакування, доступних для замовлення, або ж виведення історії замовлень для клієнта чи менеджера, коригування менеджером даних замовлень чи видалення замовлень.

2.1.2 React Router

Навігація сайту складається з багатьох елементів. Головними елементами у навігації веб-застосунку є наступні header (або шапка веб-застосунку), в якому зазвичай знаходиться навігаційна панель по основних посиланнях сайту, і footer (нижня частина веб-застосунку, із навігацією, яка дублює шапку, а також із зовнішніми посиланнями). У застосунках інтернет-магазинів також є багато посилань на описи товарів - і з масштабуванням сайту і додаванням ассортименту і нового функціоналу, кількість посилань навігації зростає відповідно. Для правильної роботи посилань всередині веб-застосунку, а також для збереження історії навігації, використовується маршрутизація.

У React немає вбудованого модуля маршрутизації, тому для навігації між сторінками використовується React Router - бібліотека маршрутизації для React. Так само, як React надає набір примітивів для створення інтерфейсу користувача та роботи зі станом, React Router надає набір компонентів та хуків для створення маршрутизації, управління історією навігації користувача та відображення різних компонетів в залежності від

поточного значення URL в адресному рядку браузера.

Компонент <BrowserRouter> - основний компонент управління маршрутизацією, який приховує в собі всю логіку взаємодії із історією браузера. Створює маршуртизатор та об'єкт історії навігації, щоб синхронізувати інтерфейс із URL-адресою. Використовуючи React контекст передає інформацію про поточний стан історії навігації всім нащадкам. Все, що необхідно зробити, це обернути компонентом <BrowserRouter> всі програми.

Ця бібліотека дозволяє створювати динамічні маршрути, забезпечуючи безперебійний користувацький досвід шляхом зіставлення різних URL-адрес з компонентами. Вона забезпечує навігацію в односторінковому застосунку (SPA) без оновлення всієї сторінки, що дозволяє переходити по посиланнях у шапці без перезавантаження застосунку. [5]

2.1.3 Formik

Основним елементом проекту є форма для збору інформації щодо замовлення від користувача. Існує два види форм – неконтрольовані форми, тобто такі форми, стан і значення полів яких не відслідковується до відправки форми, і контрольовані форми – форми, інформацію щодо стану яких можна отримати під час заповнення форми до її відправки.

Враховуючи, що поля форми замовлення пов’язані між собою (н.п. від вибору виду пакування залежить набір додаткових властивостей, і потрібно відключати непотрібні поля, щоб користувач випадково їх не заповнив), а також що потрібно проводити валідацію (перевірку правильності формату і наявності введених даних; валідація на фронтенді означає передусім захист від неправильно введених значень перед їх відправленням на сервер) полів під час заповнення її користувачем, стає очевидною потреба проекту у контрольованій формі.

Контрольовані форми в React можна налаштувати самостійно, за

допомогою навішування стану на кожен елемент форми окремо – але кращою практикою вважається використання бібліотек для створення форм для автоматизації цієї роботи. Найпопулярнішою бібліотекою для цього на даний момент є Formik.

Для розгортання форми для збору інформації щодо замовлення використано бібліотеку Formik – невелику групу компонентів та хуків (гачків) React для створення форм у React та React Native. Formik дбає про рутинні аспекти розробки, такі як відстеження значень полів, валідацію та обробку даних. Це дозволяє витрачати менше часу на підключення стану та обробників змін, тобто написання коду форми. Для внутрішньої логіки Formik використовує лише звичайний стан та властивості React. Залишаючись в межах основного фреймворку React, Formik значно полегшує розгортання та налаштування форм без ускладнення коду.

Ця бібліотека допомагає із слідкуванням за станом полів форми за допомогою пропсу initialValues, фактично створюючи контрольовану форму без потреби у створенні великої кількості станів вручну. Для цього властивості initialValues потрібно передати початкові значення полів форми – зазвичай, це пусті значення. У даному проекті всі початкові значення полів було зібрано у масив даних, і передано до initialValues у якості змінної.

Валідація полів в Formik відбувається автоматично, за допомогою передачі схеми валідації пропсу validationSchema.

2.1.4 Yup

Yup — це конструктор схем для розбору та перевірки значень під час виконання, тобто при валідації. Використовується для того, щоб визначити схему валадіції, перетворити значення елементу на відповідність, тощо. Схеми Yup надзвичайно виразні та дозволяють моделювати складні, взаємозалежні перевірки або перетворення значень.

Функції Yup.string(), Yup.min(), Yup.max(), Yup.required() і інші - це

валідатори, які дозволяють додати певний критерій валідації. Кожен валідатор може приймати від нуля до двох параметрів. Першим параметром є критерій валідації, наприклад, довжина рядка чи значення числа. Другим варіантом є рядок, який буде використаний як помилка у разі валідації.

В данному проекті за допомогою Yup створено схему валідації, яку потім передано до пропсу validationSchema у Formik.

2.1.5 Axios

Для взаємодії з сервером використано бібліотеку Axios - HTTP-клієнт, який працює на основі асинхронних запитів (Promise) для node.js та браузера. Він ізоморфний, тобто може працювати у браузері та node.js з однаковою кодовою базою. На стороні сервера він використовує власний http-модуль node.js, тоді як на стороні клієнта (браузера) він використовує XMLHttpRequests. Бібліотека Axios забезпечує стабільні HTTP-запити з підтримкою промісів. [6]

2.1.6 Модулі CSS і Vite

CSS-модулі не є офіційною специфікацією і не реалізовані в браузерах. Це процес, що запускається під час збірки проєкту (наприклад, за допомогою Vite), який замінює імена класів на унікальні. Це дозволяє використовувати однакові імена класів у різних CSS-файлах без конфліктів. Цей підхід вирішує проблему глобальної області видимості в CSS.

Vite — локальний сервер розробки, який може використовуватися для React та інших фреймворків. Він підтримує TypeScript та JSX. Vite відстежує файли під час їх редагування, і після збереження файлу веб-браузер перезавантажує код, що редагується, за допомогою процесу під назвою Hot Module Replacement (HMR), який працює шляхом простого перезавантаження конкретного файлу, що змінюється, за допомогою модулів

ES6 (ESM) замість перекомпіляції всієї програми. До того ж, Vite надає вбудовану підтримку рендерингу на стороні сервера (SSR).

Vite за замовчуванням підтримує CSS-модулі - все, що необхідно зробити це створювати файли стилів з розширенням .module.css, наприклад OrderForm.module.css. Всередині модуля CSS можна використовувати будь-який валідний CSS.

2.2 Технології Backend – серверної частини за стосунку

2.2.1 Node.js

Для реалізації серверної частини веб-застосунку обрано Node.js – середовище виконання JavaScript на стороні сервера. Основною перевагою Node.js є подієво-орієнтована архітектура, що забезпечує високу продуктивність при обробці багатьох одночасних запитів. Це критично важливо для забезпечення стабільної роботи особистих кабінетів користувачів та обробки замовлень у режимі реального часу.

Node.js має велику екосистему бібліотек (наприклад, npm), що дозволяє швидко інтегрувати додаткові функції, такі як автентифікація, робота з PDF або взаємодія з базами даних [7]. Синтаксис JavaScript, знайомий розробникам фронтенду, спрощує створення клієнт-серверної логіки. Для обробки HTTP-запитів використано фреймворк Express, який надає простий API для маршрутизації, middleware та інтеграції з іншими технологіями. [8]

2.2.2 PostgreSQL

Основними типами даних, з якими взаємодіє веб-застосунок, є дані замовників, такі як пароль і логін, особисті дані (адреса відправки замовлення, номер телефону, тощо) та дані про замовлення (стандартизовані дані щодо розмірів пакування, кількості і типу замовленого товару тощо).

Структура таблиць і взаємозв`язки даних наведені нижче, на рисунку 2.3.1:

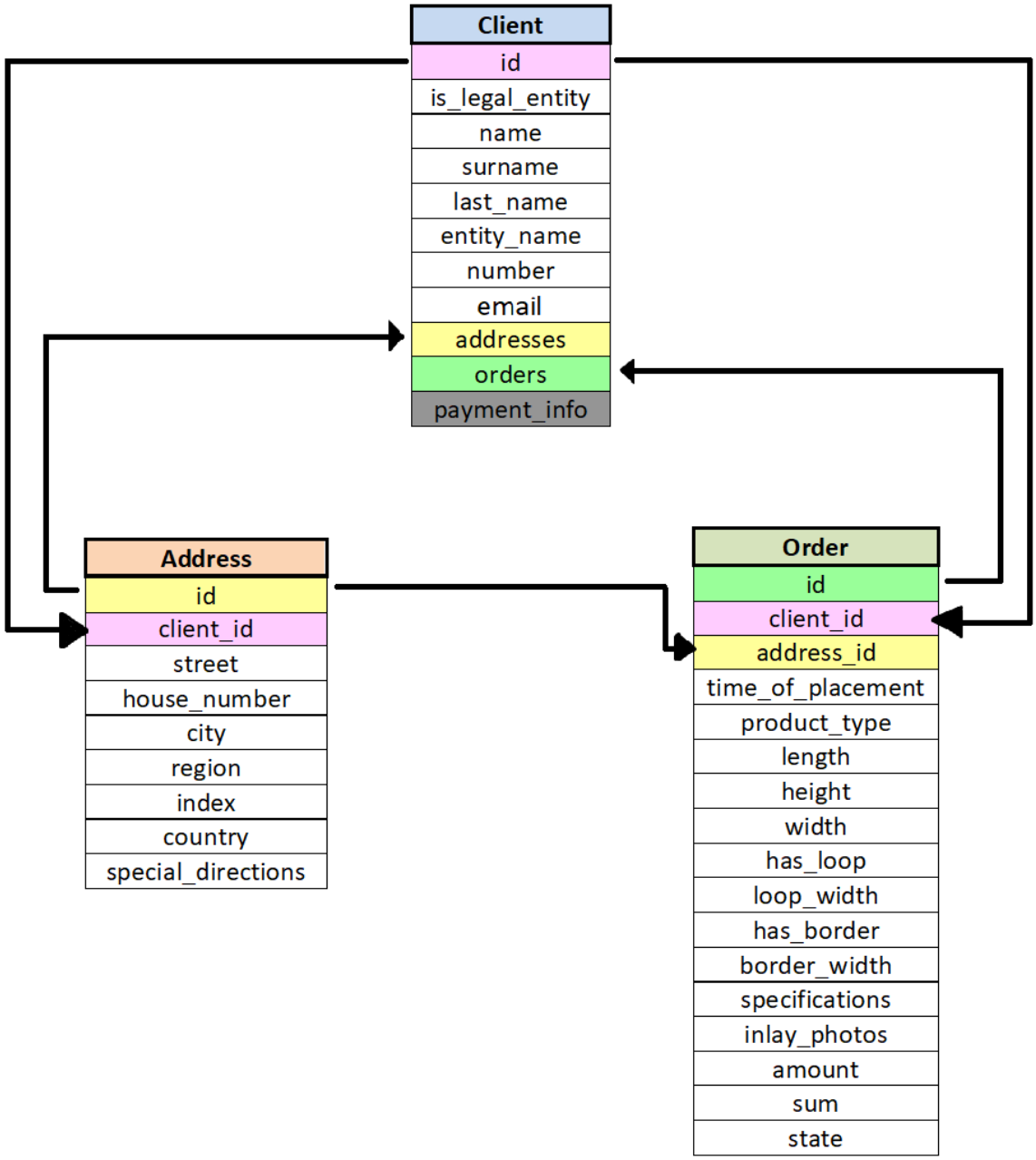

Рисунок 2.1 – зваємозязки між таблицями у базі даних проекту

Для зберігання даних користувачів та даних про замовлення обрано реляційну базу даних PostgreSQL, яка є альтернативою як комерційним СКБД (Oracle Database, Microsoft SQL Server, IBM DB2 тощо), так і СКБД з відкритим кодом (MySQL, Firebird, SQLite). [10]

Перевагою PostgreSQL є відкритий код, а також функції, які дозволяють виконувати деякий код безпосередньо сервером бази даних. PostgreSQL підтримує одночасну модифікацію БД декількома користувачами за допомогою механізму Multiversion Concurrency Control (MVCC). Завдяки цьому виконуються вимоги ACID, і практично відпадає потреба в блокуванні зчитування.

Серверна частина розгорнута на локальному пристрої з операційною системою Windows 10. Для стабільної роботи Node.js-сервер запущено у фоновому режимі через PowerShell-скрипт, що автоматизує процеси старту та перезавантаження. База даних PostgreSQL інтегрована за допомогою офіційного інсталятора для Windows. Взаємодія з БД, створення таблиць і їх налаштування проводилися крізь термінал SQL Shell (psql).

2.3 Інструменти для розробки і хостингу

Для розробки було використане середовище Microsoft Visual Studio Code - редактор початкового коду із великою кількістю розширень, таких як Emmet для автоматичного дозаповнення і виклику тегів HTML-розмітки, і Prettier форматування і покращення читабельності коду. Під час локального тестування використовувалося розширення React Developer Tools — набір інструментів для відстеження, аналізу та налагоджування React-застосунків. Вони допомагають зрозуміти, як працюють компоненти, і забезпечують інструменти для вирішення можливих проблем. Деплой для тестування під час розробки відбувався за допомогою сервісів Github – веб-сервісу для розробки програмного забезпечення, який базується на системі керування версіями Git, і Vercel – веб-сервісу для хмарного хостингу.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Розгортання проекту

Для розгортання проекту за допомогою Vite було виконано кілька кроків, а саме:

- Створено порожній репозиторій (проєкт) на GitHub;
- Клоновано цей репозиторій на свій комп'ютер;
- Запущено створення React-проекту за допомогою команди `npm create vite@latest` в терміналі і обрано потрібні налаштування;

Код проєкту за замовчуванням мініфікується і обфускується в продакшені, щоб оптимізувати завантаження додатка. Проте це може ускладнити перевірку коду час розробки або тестування на сторонньому хостунгу.

Source Map є інструментом, який дозволяє відображати транспільований або мініфікований код у вихідний код, зроблено це для комфортного перегляду в інструментах розробника браузера. Цей інструмент забеспечує можливість переглядати зручно читабельний код навіть після компіляції. Для спрощення пошуку помилок і зручності роботи над проектом під час розробки, опцію генерування Source Maps, наведену в лістингу 3.1, було додано у файл налаштувань Vite (vite.config.js), який розташований у кореневій папці проєкту.

Лістінг 3.1 – Вміст файлу налаштувань vite.config.js

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

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [react()],
  build: {
    sourcemap: true,
  }});
```

3.2 Архітектура проекту

Файлова структура проекту розділяє елементи веб-застосунку помодульно – файли компонентів із розширенням .jsx і модулі стилів до них із розширенням .module.css зберігаються в окремих папках у директорії src/components проекту, і імпортуються до місця збору, файлу App.jsx, за допомогою експорта за замовченням. Частину файлової структури наведено на рисунку 3.1.

Така файлова система не впливає на роботу застосунку,але вважається найкращою практикою, адже поліпшує читабельність коду. Щодо використання CSS-модулів, їх переваги описано вище, у підрозділі 2.1.6.

У папці assets зберігаються графічні файли, креслення і ілюстрації продукції тощо. Окрім папки components із компонентами проекту і assets, коренева папка src містить у собі власне, корінь проекту – файл main.jsx.

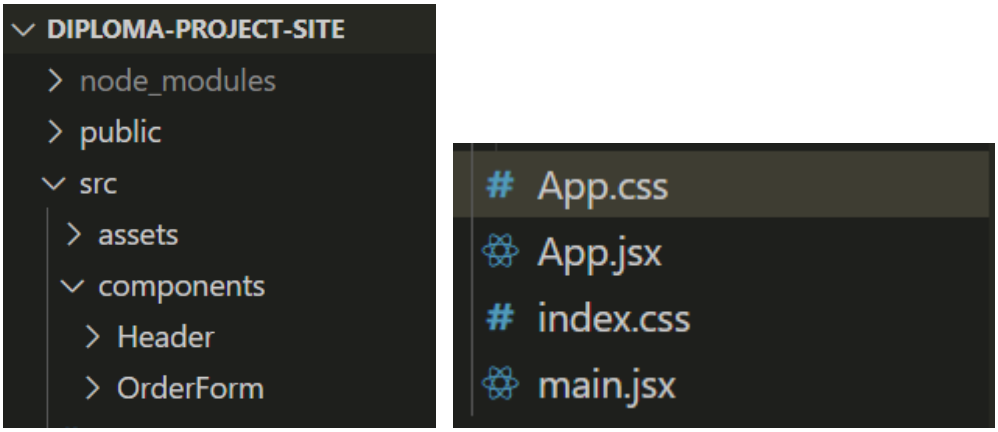

Рисунок 3.1 – Частини файлової структури проекту

З кореня проекту запускається веб-застосунок – структура веб-застосунку будується у файлі App.jsx, а потім імпортується для рендеру (“підмальовки”) у корні проекту, як наведено у лістингу 3.2.

Лістінг 3.2 – ініціалізація корня проекту, рендер проекту з файлу main.jsx

```
import { StrictMode } from 'react'
```

```
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App />
  </StrictMode>,
)
```

3.3 Створення бази даних

За допомогою терміналу було розгорнуто три таблиці в базі даних – таблиця замовлень, таблиця користувачів і таблиця адрес. В майбутньому, таке рішення дозволить користувачеві обирати збережені адреси і замовлення, уникаючи перезаповнення форму вручну у разі повтору замовлення. Таблиця замовлень і таблиця адрес пов’язані із таблицею за допомогою унікального ідентифікатора користувача, який зберігається у полі таблиць client_id.

3.4 Створення форми замовлення

Лістінг 3.3 – масив INITIAL_VALUES компоненту форми проекту

```
const INITIAL_VALUES = {
    client: "",
    name: "",
    number: "",
    email: "",
    fax: "",
    house_number: "",
    street: "",
    city: "",
    country: "",
    postal_index: "",
    delivery_specifics: "",
    shape: "",
    width: "",
    length: "",
    useful_width: "",
    useful_length: "",
    useful_height: "",
    film_thickness: "",
```

```
    logo_info: "",
    corners: "",
    euroloop: "",
    corex_color: "",
    quantity: "",
  };

...

<Formik initialValues={INITIAL_VALUES}...
```

Для створення індивідуальних ідентифікаторів елементів форми було використано встроєний хук React useId – хук для створення унікальних ідентифікаторів, які можна передавати атрибутам доступності. Індивідуальні ідентифікатори потрібні у випадку, якщо додаватимуться інші компоненти форм, щоб в однакових за назвою полів були різні відокремлюючи характеристики, і було зрозуміло, в якому з полів змінюється значення при введенні даних користувачем.

Лістінг 3.4 – призначення індивідуальних ідентифікаторов полям форми

```
  const clientId = useId();
  const nameId = useId();
  const numberId = useId();
  const emailId = useId();
  const faxId = useId();

  const houseId = useId();
  const streetId = useId();
  const cityId = useId();
  const countryId = useId();
  const postalId = useId();
  const deliveryId = useId();

  const shapeId = useId();
  const widthId = useId();
  const lengthId = useId();
  const usefulWidthId = useId();
  const usefulLengthId = useId();
  const usefulHeightId = useId();
  const filmThicknessId = useId();

  const logoInfoId = useId();
  const cornersId = useId();
  const euroloopId = useId();
  const colorId = useId();
```

```
  const quantityId = useId();
```

У компоненті форми розділи є доволі однотипними, тому є сенс навести код тільки двох розділів – розділу для збору інформації, яка стосується користувача(лістінг 3.3), і розділу для збору інформації щодо замовлення.

Активним елементом форми є компонент Field – встроєний компонент, який автоматично підключатиме вхідні дані до Formik. Він використовує атрибут name для зіставлення зі станом Formik – тобто, ключі, які перелічені у лістингу 3.2, автоматично зіставлятимуться з компонентами форми із співпадаючим ім’ям, і під час зміни у значенні елементу форми переписуватиметься відслідковуваний стан. <Field /> за замовчуванням використовуватиме HTML-елемент <input />, змінюючись за типами так само, як і звичайний HTML-елемент <input /> - за допомогою зміни властивості type. У випадку, якщо елемент форми повинен бути іншим React-компонентом, або іншим HTML-елементом, використовується властивість as.

Лістінг 3.5 – розділ компоненту форми для збору інформації, що стосується користувача

```
...
    <Formik initialValues={INITIAL_VALUES}
onSubmit={handleSubmit}>
      <Form className={s.form}>
        <div name="contactSection" className={s.formSection}>
          <span>Контакт:</span>
          <div>
            <label htmlFor="client">Замовник:</label>
            <Field type="text" name="client" id={clientId} />
          </div>

          <div>
            <label htmlFor="name">Контактна особа:</label>
            <Field type="text" name="name" id={nameId} />
          </div>

          <div>
            <label htmlFor="number">Телефон:</label>
            <Field type="text" name="number" id={numberId} />
          </div>
```

```
          <div>
            <label htmlFor="email">Електронна пошта:</label>
            <Field type="email" name="email" id={emailId} />
          </div>

          <div>
            <label htmlFor="fax">Факс:</label>
            <Field type="text" name="fax" id={faxId} />
          </div>
        </div>
...
```

Лістінг 3.6 – Частина компоненту форми для збору інформації, що стосується замовлення

```
        <div name="orderSection" className={s.formSection}>
          <div>
            <label htmlFor="shape">
              Розрахунок вартості блістерної упаковки без
загинання:
            </label>
            <Field type="radio" name="shape" value="uncurled"
id={shapeId} />
            <img
              src="src/assets/order_blister_no_curl.jpg"
              alt=""
              width="300"
              height="190"
            />
          </div>
          <div>
            <label htmlFor="shape">
              Розрахунок вартості блістерної упаковки із
загинанням (типу
              «пенал»):
            </label>
            <Field type="radio" name="shape" value="curled"
id={shapeId} />
            <img
              src="src/assets/order_blister_curl.jpg"
              alt=""
              width="300"
              height="183"
            />
          </div>
          <div>
            <label htmlFor="shape">
              Розрахунок вартості блістерної упаковки – корекс:
            </label>
            <Field type="radio" name="shape" value="corex"
id={shapeId} />
            <img src="src/assets/order_blister_correcs.jpg"
alt="" />
```

```
          </div>

          <div>
            <div>
              <label htmlFor="width">1. Ширина блістера (A),
мм</label>
              <Field type="number" name="width" id={widthId} />
            </div>
```

Поза межами форми, стан контролюється вручну, за допомогою встроєного хука React useState.

ВИСНОВКИ

Було створено MVP веб-застосунку для підприємства з виробництва пластикового пакування і устаткування, який відповідає вимогам Замовника. Застосунок забезпечує доступ до інформації про продукцію, дозволяє клієнтам оформлювати замовлення через форми, а менеджеру – зберігати інформацію щодо деталей замовлень у структурованому форматі.

Використання сучасного технологічного стеку (React, Node.js, PostgreSQL) дозволило забезпечити високу продуктивність, масштабованість та безпеку даних, що критично важливо для стабільної роботи системи.

Розгортання проекту із використанням безоплатного хмарного сервісу Vercel для публічного доступу під час розробки дозволило уникнути витрат на хмарні послуги.

Наразі застосунок має обмеження, такі як відсутність онлайн-оплати та локалізації, що вимагатиме подальшого вдосконалення. Для підвищення безпеки рекомендовано додати регулярне резервне копіювання через pg_dump та шифрування конфіденційних даних, а також більш детально проробити алгоритм валідації форми замовлення.

У майбутньому планується інтеграція з платіжними системами (наприклад, LiqPay), впровадження аналітики для генерації звітів у реальному часі та перехід до хостингу веб-застосунку із власних серверних потужностей.

Модульна архітектура та використання відкритих технологій відкривають широкі можливості для адаптації системи до нових бізнес-потреб, що робить її стратегічним інструментом для розвитку компанії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Usage statistics and market share of WordPress URL: https://w3techs.com/technologies/details/cm-wordpress

2. Пакувальні матеріали оптом ~ AVI PACK URL: https://www.avipack.com.ua/

3. Алфаінтерпласт - якісна та надійна пластикова тара URL: https://shop.alfainterplast.com.ua/ .

4. Quick Start – React URL: https://react.dev/learn .

5. React Router Home | React Router URL: https://reactrouter.com/home .

6. The Axios Instance | Axios Docs URL: https://axios-http.com/docs/instance .

7. Packages and modules | npm Docs URL: https://docs.npmjs.com/packages-and-modules .

8. Writing middleware for use in Express apps URL: https://expressjs.com/en/guide/writing-middleware.html .

9. The ngrok API | ngrok documentation URL: https://ngrok.com/docs/api/ .

10. PostgreSQL: Documentation: 17: PostgreSQL 17.5 Documentation URL: https://www.postgresql.org/docs/17/index.html .

11. Learn PostgreSQL Tutorial - Full Course for Beginners – YouTube URL: https://www.youtube.com/watch?v=qw--VYLpxG4 .

12. How To Create A Private Website For A Business: 2025 Guide URL: https://elementor.com/blog/how-to-create-a-private-website/ .