МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ
Кафедра Комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ Аліна САВЧЕНКО
«_____»_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ДИПЛОМНА РОБОТА, ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ

"МАГІСТРА"

ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ "ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ"

Тема: «Вебзастосунок для автоматизації роботи магазину з використанням мікросервісної архітектури»

Виконав: студент групи УС-211М Бухаров Ілля Олегович

Керівник: професор Воронін Альберт Миколайович

Нормоконтролер Ігор РАЙЧЕВ

Київ – 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *Комп'ютерних наук та технологій*

Кафедра *Комп'ютерних інформаційних технологій*

Галузь знань, спеціальність, освітньо-професійна програма: 12 "Інформаційні технології",122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології "

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____Аліна САВЧЕНКО

"___"_____ 2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи студента

Бухаров Ілля Олегович

(прізвище, ім'я, по батькові)

1. **Тема роботи:** «Вебзастосунок для автоматизації роботи магазину з використанням мікросервісної архітектури», затверджена наказом ректора від "29" вересня 2023р. за № 1976/ст.

2. **Термін виконання роботи :** з 02 жовтня 2023р. по 31 грудня 2023р.

3. **Вихідні дані до роботи:** теоретичні та практичні відомості та основи створення сучасного вебзастосунку для автоматизації роботи магазину з використанням мікросервісної архітектури.

4. **Зміст пояснювальної записки:** вступ, аналіз існуючих веб-додатків, постановка задачі до веб-додатку, опис обраних компонентів для реалізації веб-додатку, розробка системи для автоматизації роботи магазину.

5. **Перелік обов'язкового ілюстративного матеріалу:** слайди, презентація.

6. **Календарний план-графік:**

№ п/п	Завдання	Термін виконання	Підпис керівника
1	Аналіз і опрацювання літератури	02.10.2023 – 10.10.2023	
2	Провести консультацію з науковим керівником щодо розділів дипломної роботи	10.10.2023 – 15.10.2023	
3	Підготовка та написання розділу 1	15.10.2023 – 18.10.2023	
4	Підготовка та написання розділу 2	18.10.2023 – 24.10.2023	
5	Підготовка та написання розділу 3	24.10.2023 – 06.11.2023	
6	Оформлення пояснювальної записки	06.11.2023 – 19.11.2023	
7	Оформлення графічної частини роботи	19.11.2023 – 27.11.2023	
8	Подати дипломну роботу керівнику	11.12.2023	
9	Підготовка до захисту дипломної роботи	12.12.2023 – 20.12.2023	

7. **Дата видачі завдання:** 02.10.2023р.

Керівник дипломної роботи _____ Альберт ВОРОНІН
(підпис керівника)

Завдання прийняв до виконання _____ Ілля БУХАРОВ
(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Вебзастосунок для автоматизації роботи магазину з використанням мікросервісної архітектури» містить 92 сторінки, 29 рисунків, 1 додаток. Список бібліографічних посилань складається з 30 найменувань.

Ключові слова: WEB, ВЕБ-ДОДАТОК, JAVA SCRIPT, FRONT-AND, КЛІЄНТ-СЕРВЕР, ПОТІК ДАНИХ, РОЗРОБКА.

Об’єкт дослідження: є система управління роботою магазину.

Предмет досліджена: веб-додаток для автоматизації процесу управління роботи магазину на основі мікросервісної архітектури.

Мета дипломної роботи: розробка веб-додатку для автоматизації роботи магазину, що базується на мікросервісній архітектурі, з метою підвищення ефективності бізнес-процесів та оптимізації взаємодії із клієнтами.

Метод дослідження: використовуються методи аналізу, проектування, програмування та тестування веб-додатків.

Область застосування: веб-додатки. Під час написання дипломної роботи досліджувалися існуючі веб-додатків для автоматизації роботи магазину, програмні засоби для розробки веб-додатків.

Результати роботи: розроблений веб-додаток для автоматизації роботи магазину на базі мікросервісної архітектури відіграє важливу роль у практичному аспекті. Він не лише значно підвищує ефективність роботи магазину через автоматизацію рутинних завдань, але і сприяє збільшенню оборотів завдяки швидшому обслуговуванню клієнтів та ефективному управлінню відносинами з ними. Крім того, інтегрований модуль управління складом дозволяє магазину у режимі реального часу стежити за наявністю товарів, їхньою оборотністю та запасами.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ПЗ	Програмне забезпечення
API	Application Programming Interface
URL	Uniform Resource Locator
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
HTTP	Hyper Text Transfer Protocol
CDN	Content Delivery Network

ВСТУП

В сучасний період інформаційних технологій, велике значення набуває оптимізація та автоматизація бізнес-процесів. Особливо це стосується роздрібного бізнесу, який прагне до максимальної ефективності та оперативності роботи. Один із ключових напрямків оптимізації – це розробка веб-додатків, які спрямовані на автоматизацію роботи магазинів.

Ринок веб-додатків постійно розширюється, але існуючі системи не завжди відповідають індивідуальним потребам конкретного бізнесу. Зокрема, многі магазини потребують гнучких систем, які можуть бути легко адаптовані під їх конкретні потреби. Звертаючи увагу на динаміку розвитку техніки та технологій, можна відзначити зростаючий попит на розробку веб-додатків, які базуються на мікросервісній архітектурі.

Метою цієї роботи є розробка веб-додатку для автоматизації роботи магазину, що базується на мікросервісній архітектурі, з метою підвищення ефективності бізнес-процесів та оптимізації взаємодії із клієнтами..

Завданнями дослідження є:

1. Аналіз існуючих веб-додатків для автоматизації роботи магазину. Це необхідно для вивчення сильних і слабких сторін поточних рішень і визначення ключових вимог до нового веб-додатку.
2. Дослідження особливостей мікросервісної архітектури. Для розуміння її переваг, можливостей та потенційних складнощів при впровадженні.
3. Проектування структури бази даних. Оскільки коректна структура даних є ключем до ефективної роботи будь-якої системи.
4. Розробка архітектури та інтерфейсу веб-додатку. Створення логічної структури веб-додатку та зручного інтерфейсу для користувачів.
5. Реалізація основних модулів веб-додатку. Управління складом, CRM-система, модуль замовлень та модуль продажу товару.

6. Тестування розробленого веб-додатку. Проведення як функціонального, так і вантажного тестування для виявлення можливих недоліків і їх подальшого виправлення.

7. Оцінка ефективності впровадження системи. Порівняння продуктивності роботи магазину до та після впровадження веб-додатку.

8. Підготовка документації для користувачів та адміністраторів системи. Створення зрозумілих інструкцій для ефективного використання веб-додатку.

Методи дослідження. Використовуються методи аналізу, проектування, програмування та тестування веб-додатків.

Метод аналізу передбачає детальне вивчення, розбиття на складові частини та дослідження різних аспектів проблеми або предмета. У контексті дипломної роботи, метод аналізу було використано для вивчення існуючих веб-додатків, їхніх можливостей, переваг та недоліків, а також потреб ринку.

Метод проектування передбачає планування, розробку концепцій та структури рішення на основі зібраної інформації та встановлених вимог. Було використано для створення архітектури веб-додатку, проектування його інтерфейсу, а також структури бази даних.

Метод програмування було застосовано для реалізації розробленого веб-додатку, включаючи реалізацію різних модулів та функцій.

Метод тестування допоміг у перевірці його функціональності, стабільності та безпеки. Також дозволив оцінити ефективність розробленого рішення та його відповідність вимогам.

Розроблений веб-додаток для автоматизації роботи магазину на базі мікросервісної архітектури відіграє важливу роль у **практичному аспекті**. Він не лише значно підвищує ефективність роботи магазину через автоматизацію рутинних завдань, але і сприяє збільшенню оборотів завдяки швидшому обслуговуванню клієнтів та ефективному управлінню відносинами з ними. Крім того, інтегрований модуль управління складом дозволяє магазину у режимі реального часу стежити за наявністю товарів, їхньою оборотністю та запасами.

Що стосується **наукової новизни**, то розробка базується на декількох ключових інноваційних підходах. Перше – це використання мікросервісної архітектури в контексті автоматизації роботи магазину. Ця концепція дозволяє гнучко розширювати та модернізувати систему, інтегруючи нові сервіси без впливу на існуючу інфраструктуру. Крім того, великий акцент робиться на гнучкій інтеграції модулів та розробці адаптивного інтерфейсу користувача, який здатний автоматично налаштовуватися під потреби конкретного користувача. Останнє, але не менш важливе, нововведення – це оптимізовані алгоритми обробки замовлень, які спрямовані на покращення взаємодії з клієнтами. Всі ці елементи разом формують основу дипломної роботи, що представляє значущий науковий і практичний внесок у сферу електронної комерції.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ВЕБ-ДОДАТКУ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ МАГАЗИНУ

1.1 Аналіз існуючих веб-додатків для автоматизації роботи магазину

В епоху інформатизації та швидкого розвитку технологій, ключовим елементом ефективного управління організацією є автоматизація бізнес-процесів, зокрема у сфері управління людськими ресурсами. Інформаційний менеджмент персоналу, який включає збір, накопичення, обробку та аналіз даних про працівників, стає фундаментом для прийняття стратегічних кадрових рішень і вимагає розуміння всієї сукупності інформаційних процесів.

У сучасному світі існує велика кількість веб-додатків для автоматизації роботи магазинів. До найпопулярніших можна віднести такі системи, як Magento, Shopify, WooCommerce, Bitrix24 та інші.

Magento – це потужна платформа електронної комерції з відкритим вихідним кодом, яка надає широкі можливості для створення інтернет-магазинів будь-якого масштабу та складності [19].

Magento відрізняється гнучкими та розширеними функціями налаштування, що дозволяє точно адаптувати систему під конкретні потреби бізнесу. Платформа підтримує створення довільної кількості атрибутів для товарів, гнучке налаштування податків, доставки, оплати та багато іншого. Також Magento має відкритий API, що спрощує інтеграцію з іншими системами.

Важливою особливістю Magento є підтримка мультирівневої системи доступу. Це дозволяє створювати окремі облікові записи для різних відділів компанії та надавати співробітникам різні рівні доступу до функцій управління магазином.

Однією з найсильніших сторін Magento є потужна система управління товарним каталогом. Вона дозволяє зручно керувати величезними базами продуктів,

використовуючи різноманітні інструменти для імпорту/експорту, пошуку, фільтрації, порівняння та багато іншого.

Ще однією ключовою перевагою є можливості маркетингової аналітики в Magento. Система збирає детальну статистику про поведінку користувачів, ефективність рекламних кампаній, результати конверсії на сайті. Це дозволяє приймати обґрунтовані маркетингові рішення.

Рис. 1.1. Логотип Magento [19]

Основним недоліком Magento можна вважати порівняно високу складність налаштування та подальшої підтримки системи. Через широкі можливості конфігурації, іноді Magento використовують надмірно ускладненим чином, що призводить до проблем у роботі. Тому потрібні кваліфіковані фахівці для коректного впровадження та супроводу.

Shopify – це провідна платформа для створення інтернет-магазинів, яка позиціонує себе в першу чергу як сервіс для невеликих та середніх онлайн-бізнесів.

Головною перевагою Shopify є легкість та швидкість розгортання інтернет-магазину. Навіть без спеціальних технічних знань, магазин можна створити та запустити за кілька годин. Це досягається завдяки інтуїтивному веб-інтерфейсу та великій кількості готових шаблонів дизайну [20].

Рис. 1.2. Логотип Shopify [20]

Інтерфейс адміністративної панелі Shopify відрізняється високою гнучкістю та зручністю налаштування основних параметрів магазину. Є можливості зміни дизайну, управління товарами, замовленнями, доставкою та оплатою. При цьому функціонал трохи обмежений у порівнянні з Magento.

Серед інших вагомих переваг Shopify слід відзначити простоту інтеграції зі сторонніми сервісами через API. Наприклад, легко підключаються CRM-системи, складські та логістичні рішення. Також є мобільний додаток для керування магазином.

Основним недоліком можна вважати відсутність потужних аналітичних можливостей, як у Magento. Shopify має базову статистику для невеликих проектів. Для серйозного аналізу даних потрібно використовувати сторонні сервіси.

Отже, Shopify - це ідеальне рішення для стартапів та невеликого бізнесу, якому потрібен швидкий старт онлайн-продажів. При зростанні може знадобитися перехід на більш функціональну платформу.

WooCommerce - це одне з найпопулярніших рішень для створення інтернет-магазинів на базі платформи WordPress. Воно реалізоване у вигляді плагіну, який перетворює звичайний сайт WordPress на повноцінний інтернет-магазин.

Основною перевагою WooCommerce є легкість встановлення та подальшого налаштування. Для розгортання магазину не потрібні глибокі технічні знання та навички програмування. Інтуїтивний інтерфейс WordPress дозволяє швидко виконати базову конфігурацію [21].

Рис. 1.3. Логотип WooCommerce [21]

Також вагомим плюсом є доступ до величезної кількості безкоштовних і платних модулів та шаблонів оформлення для WooCommerce. Це надає гнучкості та можливості розширення функціональності магазину. Модулі існують для інтеграції зі складським обліком, системами аналітики, різними платіжними сервісами тощо.

Основним недоліком WooCommerce можна вважати падіння продуктивності сайту при зростанні навантаження та ускладненні конфігурації. Це пов’язано із самою платформою WordPress. Для великих проектів може знадобитися оптимізація бази даних, кешування, використання CDN та інших додаткових рішень.

Ще однією проблемою є необхідність постійно оновлювати як сам плагін WooCommerce, так і використовувані модулі. Це пов'язано із виправленням уразливостей та покращенням функціоналу, тому не можна ігнорувати.

Отже, WooCommerce - гарне бюджетне рішення для невеликих онлайн-магазинів, але при значному масштабуванні потрібно враховувати його технічні обмеження.

Рішення Bitrix24 поєднує в собі функції системи управління сайтом, CRM та засобів внутрішніх комунікацій для бізнесу. Модуль CRM в Bitrix24 містить потужний інструментарій для автоматизації та оптимізації бізнес-процесів.

Рис. 1.4. Логотип Bitrix24 [22]

Однією з основних переваг CRM Бітрікс24 є широкі аналітичні можливості. Система збирає та структурує детальну інформацію про клієнтів, маркетингові кампанії, продажі та іншу важливу статистику. Гнучкі інструменти для побудови звітів дозволяють аналізувати ефективність роботи з клієнтами та приймати обґрунтовані управлінські рішення.

Ще однією сильною стороною CRM Бітрікс24 є функціонал для комунікації з клієнтами. Це включає в себе чати, електронну пошту, телефонію, sms-розсилки в межах однієї системи. Такий підхід дозволяє оптимізувати взаємодію з клієнтами та підвищити лояльність [22].

Однак у системи є і певні недоліки. Зокрема, інтерфейс CRM модуля відрізняється досить високою складністю та перевантаженістю функціями. Для невеликих компаній це може бути надмірно і потребує витрат часу на вивчення та налаштування.

Ще одним мінусом можна вважати надмірну універсальність системи Бітрікс24. Вона намагається охопити величезну кількість задач бізнесу, що іноді призводить до громіздких та заплутаних рішень. Для невеликих магазинів це може бути неоптимально.

Отже, CRM-модуль Бітрікс24 - потужний, але досить складний у використанні. Він більше підходить для середнього та великого бізнесу, ніж для стартапів та малих компаній.

Таблиця 1.1.

Порівняльна таблиця описаних систем автоматизації для інтернет-магазинів

Система	Переваги	Недоліки
Magento	• Потужний функціонал • Гнучкість налаштувань • Маркетингова аналітика	• Складність впровадження та підтримки
Shopify	• Швидкість старту • Простота використання • Інтеграція з сервісами	• Обмежена аналітика • Недостатньо можливостей для масштабування
WooCommerce	• Легке впровадження • Велика кількість готових рішень • Низька вартість	• Проблеми масштабованості та продуктивності • Потреба в постійних оновленнях
Bitrix24 (CRM)	• Можливості аналітики • Інструменти для комунікації	• Складний інтерфейс • Надмірний функціонал для малих проектів

Порівнюючи представлені системи, можна побачити, що вони орієнтуються на різні потреби та типи бізнесу.

Magento з його величезним функціоналом та гнучкістю підходить для великих магазинів, які готові інвестувати ресурси в складне налаштування заради отримання потужної системи.

Shopify - це швидкий старт для невеликих проектів, яким потрібні прості та зрозумілі рішення. Проте можливості для масштабування тут обмежені.

WooCommerce завдяки легкості розгортання та величезній екосистемі додатків підійде startup-ам, що шукають бюджетний варіант. Однак на певному етапі росту можуть виникнути технічні складнощі.

Bitrix24 та його CRM-модуль – це універсальне рішення для середнього та великого бізнесу. Потужна функціональність та аналітика в поєднанні зі складним

інтерфейсом. Отже платформа автоматизації має відповідати поточному рівню та амбіціям компанії. В подальшому із зростанням може виникнути необхідність переходу на більш масштабоване рішення.

1.2 Особливості та функції мікросервісної архітектури

Мікросервісна архітектура, яка зазнала широкого поширення в області розробки програмного забезпечення, відрізняється низкою особливостей та функцій, які роблять її важливою для сучасних технологічних вимог. Основною характеристикою мікросервісної архітектури є її децентралізована природа, де система розділена на дрібні, незалежні сервіси, кожен з яких відповідає за виконання певних завдань і функцій. Ця модульність дозволяє розробникам працювати над окремими частинами системи без ризику зруйнувати інші складові, що сприяє ефективній розробці та впровадженню змін.

Рис. 1.5. Особливості та функції мікросервісної архітектури

Другою важливою характеристикою є масштабованість. Оскільки кожен мікросервіс може бути масштабований незалежно, це забезпечує велику гнучкість у

розподілі ресурсів та оптимізації продуктивності, особливо в умовах великих та динамічних обсягів даних. Вдосконалення одного сервісу не вимагає перегляду всієї системи, що знижує вартість та час, необхідний для впровадження нововведень.

Третьою особливістю є легкість в розгортанні. Незалежність мікросервісів дозволяє швидко та легко розгортати, оновлювати або виправляти помилки в окремих частинах системи, не торкаючись інших модулів. Це веде до скорочення часу впровадження нових функцій та підтримки системи в цілому.

Четвертою перевагою є відмовостійкість. В мікросервісній архітектурі відмова одного сервісу не веде до катастрофічного збою всієї системи. Решта сервісів продовжують працювати, що підвищує загальну надійність та доступність системи.

Крім того, мікросервіси підтримують технологічну агностичність, дозволяючи використовувати різні технології, мови програмування та бази даних для різних сервісів. Це забезпечує високу гнучкість у виборі оптимальних технічних рішень для кожної задачі.

Мікросервісна архітектура ідеально підходить для реалізації методологій безперервної інтеграції та безперервного розгортання (CI/CD). Це дозволяє командам розробників швидко впроваджувати зміни, вдосконалювати функціональність та забезпечувати безперервну оптимізацію продуктивності та безпеки системи.

Отже, мікросервісна архітектура пропонує значні переваги для розробки сучасних програмних систем. Її децентралізований підхід, який базується на розділенні системи на незалежні сервіси, підвищує ефективність розробки та зменшує ризики, пов'язані зі змінами в системі. Масштабованість, що надає можливість оптимізувати продуктивність та ресурси, є ключовою для великих та динамічних обсягів даних. Легкість в розгортанні та відмовостійкість сприяють надійності системи та її швидкому оновленню. Технологічна агностичність та підтримка CI/CD забезпечують гнучкість та сприяють безперервному вдосконаленню.

У цілому, мікросервісна архітектура відкриває шляхи для створення більш модульних, масштабованих та відмовостійких систем, що важливо у висококонкурентному і швидко змінюваному технологічному середовищі сучасного програмного забезпечення.

1.3 Обґрунтування доцільності розробки специфікованого веб-додатку

Розробка специфікованого веб-додатку є доцільною з кількох важливих причин. На сучасному ринку існує виражена потреба у рішеннях, що відповідають специфічним запитам цільової аудиторії, дозволяючи заповнити існуючі прогалини у наявних послугах чи продуктах. Крім того, швидкий розвиток технологій надає нові можливості для створення більш ефективних та функціональних веб-додатків, які можуть вирізнятися на ринку своєю унікальністю.

З бізнес-перспективи, веб-додаток може стати ключовою частиною стратегії компанії, спрямованої на розширення ринку, збільшення продажів, або підвищення ефективності внутрішніх процесів. Це не лише відкриває нові горизонти для розвитку бізнесу, але й сприяє підвищенню конкурентоздатності компанії.

З точки зору кінцевого користувача, важливим є надання значної цінності через удосконалення користувацького досвіду, доступності послуг та нових функціональних можливостей. Веб-додаток, який забезпечує інтуїтивно зрозумілий інтерфейс та високий рівень зручності користування, може значно підвищити задоволеність користувачів.

Масштабованість та гнучкість також є важливими аспектами веб-додатків, оскільки вони дозволяють швидко адаптуватися до змінних ринкових умов та потреб користувачів. Таким чином, розробка веб-додатку не лише відповідає актуальним потребам ринку та користувачів, але й забезпечує компанії можливість бути гнучкою та інноваційною в довгостроковій перспективі.

Обґрунтування доцільності розробки специфікованого веб-додатку для автоматизації роботи магазину в порівнянні з універсальними платформами:

1) Можливість максимально точного врахування усіх особливостей та нюансів бізнес-процесів конкретного магазину. Універсальні готові IT-рішення часто є надмірно узагальненими і не враховують певної специфіки, притаманної саме цьому закладу. Наприклад, асортимент товарів, логіка оформлення замовлень, взаємодія з постачальниками, організація складського господарства можуть суттєво відрізнятися навіть для магазинів одного профілю. Спеціалізований веб-додаток може врахувати всі ці нюанси під час проектування.

2) Можливість оптимізації функціоналу та інтерфейсу веб-додатку виключно під необхідні конкретному магазину задачі та сценарії використання. Універсальні системи автоматизації часто містять багато непотрібних для конкретного випадку модулів, опцій, параметрів налаштування. Це ускладнює їх освоєння персоналом магазину та подальше адміністрування системи. Спеціалізований же веб-додаток буде включати тільки необхідні саме для ефективної роботи цього закладу компоненти та функції.

3) Спрощення процесів налаштування та оновлення системи під мінливі вимоги бізнесу за рахунок більш гнучкої архітектури. Завдяки застосуванню сучасних гнучких технологій розробки та відносній простоті спеціалізованого рішення, його набагато легше модифікувати та розширювати під зміни у бізнес-процесах чи функціональних вимогах. Це особливо важливо з огляду на динамічність сучасного ринку.

4) Потенційна економія коштів на розробку та подальшу підтримку системи у порівнянні з вартістю впровадження складних універсальних платформ автоматизації. Завдяки оптимізації функціоналу під потреби конкретного магазину та відносній простоті архітектури, витрати на створення та обслуговування спеціалізованого веб-додатку зазвичай є меншими. Це особливо відчутно для невеликих проектів.

5) Можливість більш тісної взаємодії та співпраці з розробником веб-додатку для кращого розуміння його командою конкретних потреб бізнесу замовника. Це дозволяє оперативно реагувати на зміни у вимогах та адаптувати систему відповідним чином. При використанні універсальних платформ такі можливості часто є обмеженими.

Отже, для невеликих чи середніх бізнес-проектів, яким потрібне гнучке та максимально адаптоване ІТ-рішення під свою специфіку, розробка спеціалізованого веб-додатку є цілком виправданою та доцільною. Вона дозволяє отримати оптимальне за функціоналом, вартістю та зручністю використання програмне забезпечення для автоматизації саме цього бізнесу.

1.4 Вибір інструментів розробки

Вибір інструментів розробки для створення веб-додатку є критичним кроком, який впливає на продуктивність, доступність, та гнучкість кінцевого продукту. У нашому випадку вибір пав на наступний набір технологій: HTML, CSS, PHP, JavaScript, Bootstrap, та MySQL. Розглянемо кожен з них детальніше.

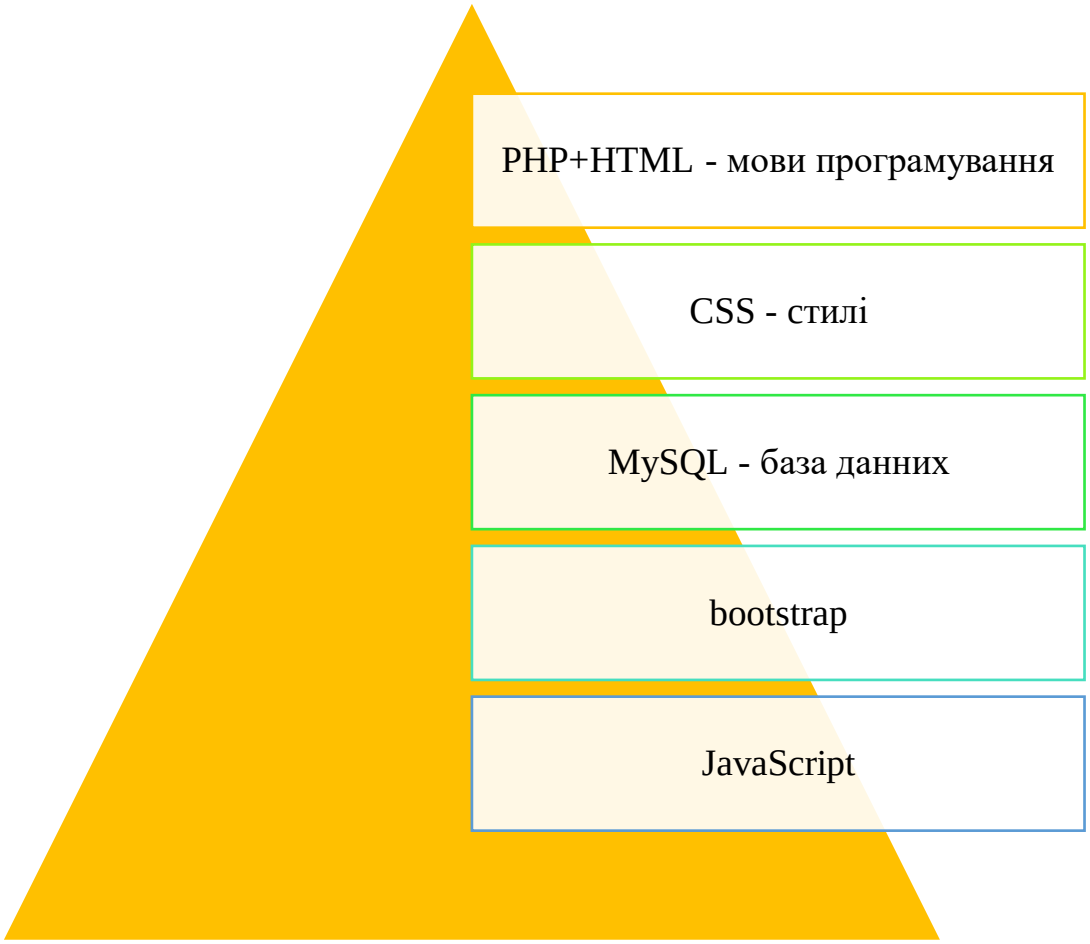

Рис. 1.6. Вибір інструментів розробки системи

HTML, або HyperText Markup Language, є фундаментальною мовою розмітки, що використовується для створення та проектування веб-сторінок. Ця мова служить каркасом для будь-якого веб-сайту, визначаючи структуру контенту на сторінці. HTML використовується для організації та представлення різноманітного контенту, включаючи текст, зображення, посилання, таблиці, списки та багато іншого.

У HTML, різні типи контенту та елементи структуруються за допомогою тегів. Теги - це спеціальні ключові слова, обрамлені кутовими дужками, які вказують браузеру, як відображати відповідний контент. Наприклад, тег **<p>** використовується

для визначення абзацу тексту, а тег **<img>** - для вставки зображення. HTML-теги можуть включати атрибути, які надають додаткову інформацію про елемент, наприклад, шлях до зображення у випадку тегу **<img>**.

Однією з ключових переваг HTML є його універсальність та широкий діапазон застосування. HTML-сторінки можуть відображатися у всіх популярних веб-браузерах, забезпечуючи доступність контенту для широкої аудиторії користувачів. Також HTML є основою для більш складних технологій, таких як CSS та JavaScript, які додають стилізацію та інтерактивність до веб-сторінок.

Використання HTML у розробці веб-додатку дозволяє розробникам ефективно організовувати контент, забезпечувати його чітку структуру та легкість навігації для користувачів. Це забезпечує зручний та інтуїтивно зрозумілий інтерфейс, що сприяє підвищенню користувацького досвіду.

CSS, або Cascading Style Sheets, є невід'ємною частиною веб-дизайну, яка забезпечує контроль над візуальним стилем веб-сторінок. У тандемі з HTML, який формує структуру веб-сторінки, CSS додає естетичне оформлення, роблячи веб-сторінку не тільки функціональною, але й візуально привабливою. CSS дозволяє веб-розробникам контролювати розміщення елементів на сторінці, налаштовувати шрифти, кольори, відступи, рамки, фони та багато іншого.

Основна перевага CSS полягає у тому, що вона дозволяє відокремити зміст веб-сторінки від її візуального представлення. Це означає, що можна легко змінювати дизайн, не зачіпаючи основний HTML-код. Наприклад, змінивши кілька рядків CSS, можна оновити вигляд усіх кнопок на сайті, не торкаючись HTML-структури кожної кнопки окремо.

CSS працює на основі "каскадного" принципу, де стилі можуть бути призначені для конкретних HTML-елементів, а також узагальнені для цілих класів чи ідентифікаторів. Це дозволяє створювати складні та згруповані стилі, що забезпечують консистентність та спрощують управління дизайном сайту. Крім того, CSS підтримує медіа-запити, які дозволяють адаптувати веб-сторінки під різні розміри екранів та пристроїв, забезпечуючи гнучкість відображення та відповідність стандартам відгукового веб-дизайну.

Використання CSS у розробці веб-додатку є ключовим для створення професійного, сучасного веб-інтерфейсу, що сприяє залученню та утриманню користувачів. Воно дозволяє створювати візуально приємні, чисті та організовані веб-сторінки, що є важливими елементами користувацького досвіду.

PHP (Hypertext Preprocessor) – це потужна серверна скриптова мова програмування, яка грає важливу роль у розробці веб-додатків. Основна перевага PHP полягає у її здатності створювати динамічний та інтерактивний контент на веб-сайтах. Вона відрізняється від HTML, яка використовується для створення статичних сторінок, оскільки PHP може виконувати скрипти на сервері, що генерують динамічний HTML, відправлений до браузера користувача.

PHP може взаємодіяти з різними базами даних, що робить її ідеальним вибором для розробки веб-додатків, які потребують зберігання, обробки та відображення даних. Це означає, що PHP може використовуватися для створення великого спектру веб-додатків, від простих блогів до складних комерційних платформ та систем управління контентом.

PHP підтримує широкий спектр функцій, включаючи обробку форм, управління сесіями, створення та обробку файлів, а також можливість виконувати запити та взаємодіяти з веб-сервером. Це дозволяє розробникам писати скрипти, які виконують складні завдання, такі як обробка користувацького вводу, виконання умовної логіки, інтеграція з API сторонніх сервісів та управління даними користувача.

Важливою перевагою PHP є його інтеграція з різними фреймворками та інструментами розробки, що дозволяє створювати структуровані, безпечні та ефективні веб-додатки. Використання PHP в поєднанні з HTML, CSS і JavaScript дозволяє створювати повнофункціональні веб-сайти та додатки, які задовольняють сучасні вимоги до веб-розробки.

Таким чином, вибір PHP як інструменту розробки для цього веб-додатку є виправданим завдяки її гнучкості, широкому функціоналу та великій підтримці спільноти розробників.

JavaScript є надзвичайно важливою мовою програмування у сфері веб-розробки, яка використовується для додавання інтерактивних елементів на веб-

сторінки. Ця мова дозволяє веб-розробникам створювати більш динамічний та інтерактивний користувацький досвід, перетворюючи статичні HTML-сторінки на багатофункціональні веб-додатки. Завдяки JavaScript, можливо реалізувати різноманітні функції, такі як реагування на кліки користувача, валідація форм, підвантаження контенту без потреби перезавантажувати сторінку, а також створення анімацій та візуальних ефектів.

JavaScript є ключовим компонентом в стеку технологій, відомому як MEAN (MongoDB, Express.js, AngularJS, Node.js), і він відіграє центральну роль у розробці як клієнтської, так і серверної частини додатків. На відміну від PHP, який виконується на сервері, JavaScript виконується безпосередньо в браузері користувача, що робить його ідеальним для створення швидких та відгукових веб-інтерфейсів.

Однією з найбільш значущих переваг JavaScript є його універсальність. Він може використовуватися разом з різними бібліотеками та фреймворками, такими як React, Angular, Vue.js, що дозволяє створювати складні односторінкові додатки (SPA), що забезпечують більш гладке та нативне-подібне користувацьке середовище. Це особливо корисно для розробки інтерфейсів електронної комерції, панелей керування, інтерактивних форм та інших елементів, що вимагають високого рівня взаємодії з користувачем.

JavaScript також підтримує асинхронне програмування за допомогою промісів та асинхронних функцій, що спрощує роботу з відкладеними або тривалими операціями, такими як запити до веб-API. Це дозволяє веб-додаткам завантажувати дані у фоновому режимі, покращуючи взаємодію з користувачем та загальну продуктивність.

Bootstrap - це надзвичайно популярний фреймворк, що базується на HTML, CSS і JavaScript, який значно спрощує процес розробки веб-інтерфейсів. Основною перевагою Bootstrap є його гнучкість та легкість у використанні, що робить його ідеальним вибором для розробників будь-якого рівня. Фреймворк надає різноманітні готові до використання стилі та компоненти, які допомагають швидко створювати адаптивні та мобільно-дружні веб-сторінки.

Bootstrap включає систему сітки, що дозволяє розробникам легко створювати макети сторінок, що є відповідними для різних розмірів екранів, від мобільних телефонів до великих моніторів. Це забезпечує гнучке та відповідне представлення контенту на різних пристроях. Крім того, Bootstrap містить велику кількість компонентів, таких як кнопки, навігаційні панелі, модальні вікна, вкладки та випадаючі меню, які можуть бути легко інтегровані та налаштовані відповідно до потреб проекту.

Ще однією важливою особливістю Bootstrap є його інтуїтивність та легкість у навчанні. Завдяки документації та численним прикладам, новачки можуть швидко освоїти основи та почати використовувати фреймворк у своїх проектах. Також, він співпрацює з різними JavaScript бібліотеками, що дозволяє додавати додаткову функціональність і створювати більш складні інтерфейси.

Використання Bootstrap у розробці веб-додатку може значно зменшити час та зусилля, потрібні для створення адаптивного дизайну, дозволяючи розробникам зосередитися на більш складних аспектах розробки, таких як логіка користувацького інтерфейсу та інтеграція з серверною частиною. У підсумку, Bootstrap є вибором, який дозволяє створювати високоякісні, професійні та адаптивні веб-сторінки з мінімальними зусиллями та у найкоротші терміни.

MySQL – це високопродуктивна система управління базами даних, що використовується для зберігання, управління та обробки великих обсягів даних у веб-додатках. Ця система забезпечує швидкий доступ до даних, підтримує великі обсяги інформації та пропонує гнучкі можливості для створення запитів, що робить її ідеальним вибором для веб-розробників, яким потрібна надійна та ефективна система для зберігання даних.

Однією з ключових переваг MySQL є її висока продуктивність і масштабованість. Це означає, що MySQL може ефективно обробляти як невеликі, так і великі бази даних, забезпечуючи високу швидкість обробки запитів та транзакцій. Така гнучкість робить MySQL ідеальним рішенням для широкого спектру веб-додатків, від невеликих блогів до великих комерційних сайтів.

Крім того, MySQL має багатий набір функцій, що включає підтримку складних запитів, транзакцій, реплікації, тригерів, в'юшок (views) та збережених процедур. Це дозволяє розробникам створювати гнучкі та ефективні рішення для обробки даних, а також гарантує безпеку та стабільність веб-додатку.

Ще одним важливим аспектом є те, що MySQL підтримує ряд програмувальних мов, включаючи PHP, Python, Java та інші, що розширює можливості її використання та інтеграції з різними веб-додатками. Така сумісність робить MySQL особливо зручною для розробки динамічних сайтів та додатків, які вимагають постійного обміну даними між сервером та клієнтом.

У підсумку, MySQL є міцною та надійною платформою для розробки веб-додатків, яка забезпечує високий рівень продуктивності, безпеки та масштабованості. Її універсальність та легкість у використанні роблять її однією з найпопулярніших систем управління базами даних, яку вибирають розробники по всьому світу.

Використання цих інструментів разом дозволяє створити надійний, гнучкий та масштабований веб-додаток, який може задовольнити потреби сучасного бізнесу та його користувачів.

Висновок до розділу 1

У першому розділі роботи було розглянуто теоретичні основи та передумови розробки веб-додатку для автоматизації роботи магазину.

Зокрема, в підрозділі 1.1 проведено аналіз існуючих на ринку рішень для автоматизації торгівлі. Визначено ключові переваги та недоліки популярних систем Magento, Shopify, WooCommerce, Bitrix24.

В підрозділі 1.3 на основі аналізу обґрунтовано доцільність розробки власного спеціалізованого веб-додатку, адаптованого під конкретні потреби і особливості функціонування магазину.

В підрозділі 1.4 проведено вибір технологій та інструментів розробки з урахуванням вимог щодо функціональності, надійності, безпеки та масштабованості майбутньої системи.

Отримані результати дозволили сформулювати вимоги до веб-додатку та створити необхідне підґрунтя для подальших етапів його проектування та розробки, що буде виконано в наступних розділах роботи.

РОЗДІЛ 2
РОЗРОБКА ВЕБ-ДОДАТКУ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ МАГАЗИНУ

2.1 Вимоги до системи та її функціональні можливості

Розробка будь-якого веб-додатку починається з формулювання чітких вимог, які визначають його мету, функціональність та параметри роботи. Для нашого веб-додатку для автоматизації роботи магазину були встановлені вимоги, що охоплюють широкий спектр аспектів: від функціональних і технічних до користувацьких та безпекових.

Серцевиною веб-додатку є його функціональні можливості, які були розроблені для вирішення специфічних потреб управління магазином. Основною ціллю є надання простого та ефективного інструменту для автоматизації повсякденних задач, що включає управління продуктовим каталогом, замовленнями, клієнтською базою та аналітикою продажів. Центральна частина системи забезпечує інтеграцію між різними модулями, забезпечуючи гладку координацію та обмін даними.

З точки зору технічних вимог, система повинна бути стійкою, швидкою та масштабованою. Враховуючи потреби сучасного ринку, веб-додаток був розроблений з використанням сучасних технологій та платформ, що забезпечують високу продуктивність та легкість у використанні. Важливим аспектом є також адаптивність інтерфейсу, що дозволяє користувачам легко взаємодіяти з системою з різних пристроїв, включаючи мобільні.

Користувацький інтерфейс має особливе значення, оскільки він є основним засобом взаємодії між користувачем і системою. Веб-додаток був розроблений із зосередженням на інтуїтивності, зручності використання та візуальній привабливості.

Велика увага приділялася логічній організації меню, панелей інструментів та інших елементів інтерфейсу, щоб забезпечити користувачам швидкий доступ до необхідних функцій.

У сучасному цифровому світі безпека даних є ключовим пріоритетом. Веб-додаток був розроблений з урахуванням передових методів захисту інформації та даних користувачів. Були впроваджені надійні механізми аутентифікації, шифрування даних та захисту від загроз, таких як SQL-ін'єкції, кросс-сайтовий скриптинг (XSS) та інші види кібератак.

Розробка веб-додатку для автоматизації роботи магазину заснована на детальному аналізі вимог та потреб користувачів. Комбінація функціональності, технічних можливостей, користувацького інтерфейсу та безпекових заходів забезпечує створення комплексного рішення, яке відповідає сучасним вимогам та трендам ринку. Цей підхід гарантує, що розроблений веб-додаток не тільки відповідає потребам магазину, але й надає стійку платформу для його розвитку та росту в майбутньому.

2.2 Проектування структури бази даних

У цьому розділі розглядається проектування структури бази даних для веб-додатку, який сприяє автоматизації роботи магазину. Проектування бази даних є одним з ключових аспектів розробки веб-додатків, оскільки від його ефективності залежить продуктивність системи, легкість доступу до даних та можливість їх адекватного управління.

Розроблені таблиці, такі як categories, products, clients, orders, users, workers та інші, формують основу бази даних. Кожна таблиця має унікальне ім'я та складається з рядків та стовпців, які визначають поля даних. Наприклад, таблиця products зберігає інформацію про товари, включаючи такі поля, як ID товару, SKU, назва, опис, ціна та інші.

Важливою частиною проектування є визначення зв'язків між таблицями. Ці зв'язки встановлюються через поле, спільне для двох таблиць, яке дозволяє зв'язати

дані між ними. Наприклад, зв'язок між таблицями products та categories здійснюється через поле CategoryID у таблиці products, яке відсилає до поля ID у таблиці categories.

Під час проектування бази даних велика увага приділяється нормалізації. Нормалізація допомагає уникнути дублювання даних і забезпечує їх цілісність. Це процес, який вимагає ретельного аналізу структури даних і забезпечує оптимізацію запитів до бази даних.

Безпека даних також є критично важливою. Вона забезпечується за допомогою механізмів шифрування, регулювання доступу до бази даних та інших заходів безпеки. Окрім того, важливо мати надійну систему резервного копіювання та відновлення, щоб запобігти втраті даних у разі збою системи чи інших нештатних ситуацій.

Для візуалізації структури бази даних та її зв'язків ефективно використовується ER-діаграма (Entity-Relationship Diagram). Вона демонструє таблиці, їх поля та відносини між ними, що допомагає краще розуміти структуру бази даних.

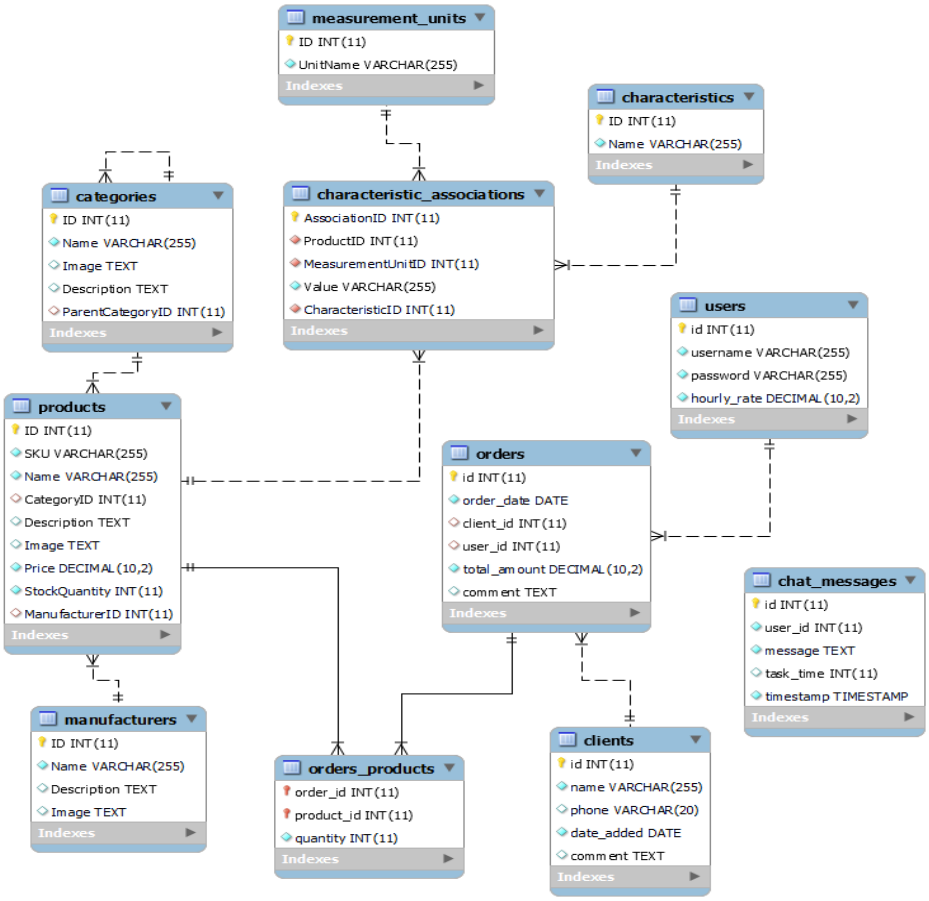

Рис. 2.1. ER діаграма бази даних

Завершуючи, проектування структури бази даних є важливим етапом розробки веб-додатку, який впливає на багато аспектів його функціонування. Від ефективного проектування залежать швидкість доступу до даних, продуктивність системи, її масштабованість та здатність адекватно реагувати на зміни в бізнес-процесах магазину.

При проектуванні структури бази даних для веб-додатку, який спрямований на автоматизацію роботи магазину, важливо звернути увагу на її оптимізацію та гнучкість для подальшого розвитку. Оптимізація бази даних полягає у вдосконаленні її структури, щоб забезпечити максимальну продуктивність та ефективність. Це включає аналіз та вдосконалення SQL-запитів, щоб зменшити навантаження на сервер та підвищити швидкість відповіді. Ключовим аспектом є підтримання балансу між нормалізацією та денормалізацією даних, що допомагає зберегти цілісність даних і в той же час підвищити швидкість читання.

Окрім того, важливо використовувати індексування для покращення швидкості пошуку даних, особливо у великих базах даних. Індексування повинно бути виконано розумно, щоб уникнути надмірного навантаження. Також корисним є використання кешування для зберігання часто використовуваних даних, що може значно зменшити навантаження на базу даних.

Регулярний моніторинг продуктивності бази даних і її налаштування для оптимальної роботи також є важливими. Це дозволяє вчасно виявляти та виправляти проблеми, а також підтримувати високу продуктивність системи.

Щодо можливостей подальшого розвитку, структура бази даних повинна бути гнучкою, щоб легко адаптуватися до зміни обсягу даних та зростання бізнесу. Важливо передбачити можливість масштабування, як горизонтального, так і вертикального. База даних повинна бути спроектована так, щоб вносити зміни у структуру даних, не порушуючи існуючі функціональності. Це забезпечує можливість легкої інтеграції з новими технологіями та розвитком бізнесу.

Таким чином, при проектуванні бази даних важливо не тільки забезпечити її ефективну початкову реалізацію, але й передбачити потенціал для її подальшого розвитку та оптимізації.

2.3 Розробка архітектури та інтерфейсу веб-додатку

У цьому підрозділі ми зосередимося на архітектурі та інтерфейсі системи управління магазином, ключових аспектах, які визначають її ефективність, гнучкість та зручність для користувачів.

Була обрана сучасна веб-орієнтована архітектура з використанням мікросервісного підходу. Це забезпечує високу гнучкість, масштабованість та можливості подальшої модернізації системи.

Мікросервісна архітектура передбачає розбиття системи на невеликі слабко пов'язані сервіси, кожен з яких виконує свою бізнес-функцію. У нашому проекті основними мікросервісами є:

- сервіс управління товарами;
- сервіс управління замовленнями;
- сервіс управління клієнтами;
- сервіс білінгу та фінансових операцій;
- сервіс бізнес-аналітики та звітності.

Така архітектура дозволяє легко масштабувати та оновлювати окремі компоненти без впливу на інші.

Для реалізації були використані сучасні хмарні технології на базі контейнеризації та оркестрування контейнерів. Це забезпечує високу надійність, доступність та ефективне використання ресурсів.

Що стосується інтерфейсу користувача, був розроблений адаптивний веб-дизайн з використанням фреймворку Bootstrap. Це дозволяє забезпечити зручне відображення на різних пристроях - від настільних ПК до мобільних телефонів.

Також був реалізований зрозумілий навігаційний механізм та ретельно спроектовані шаблони сторінок для полегшення використання основних функцій системи користувачами.

Загалом, запропоновані рішення з архітектури та інтерфейсу забезпечать сучасну, гнучку та масштабовану платформу для автоматизації роботи магазину з високою ефективністю та зручністю експлуатації.

2.4 Реалізація основних модулів веб-додатку (управління складом, CRM-система, модуль замовлень, модуль продажу товару)

У цьому розділі описується процес реалізації ключових модулів веб-додатку, що сприяють автоматизації роботи магазину. Модулі, такі як управління складом, CRM-система, модуль замовлень, та модуль продажу товару, є вирішальними для ефективного функціонування веб-додатку.

Почнемо з модулю управління складом. Цей модуль має велике значення для забезпечення точності і актуальності інформації про товари. Використовуючи PHP та SQL, реалізовано сторінку products.php, яка є основою для управління інформацією про товари.

Код сторінки products.php починається з ініціалізації сесії та перевірки авторизації користувача. Якщо користувач не авторизований, відбувається перенаправлення на сторінку авторизації. Важливою частиною є функція getProducts(), яка використовується для отримання даних про товари з бази даних. Використання SQL-запитів з JOIN-операціями дозволяє зібрати всю необхідну інформацію про товари, включаючи категорію та виробника.

Користувач має можливість переглядати список товарів, де відображаються основні характеристики, такі як ID, назва, категорія, виробник, ціна та кількість на складі. Додатково реалізовано функціонал для додавання, редагування, перегляду деталей та видалення товарів. Це забезпечує повний контроль над асортиментом товарів та їх характеристиками.

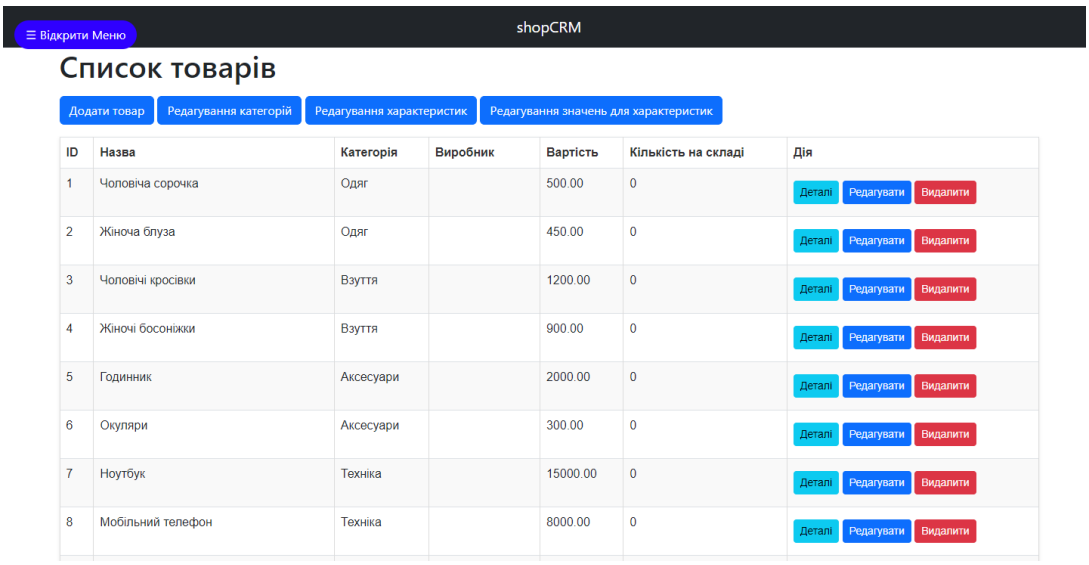

Рис. 2.2. Інтерфейс модуля управління складом

CRM-система, Модуль замовлень та Модуль продажу товару. Наступні модулі, такі як CRM-система, модуль замовлень та модуль продажу товару, є важливими для забезпечення ефективного управління взаємовідносинами з клієнтами, обробки замовлень та продажу товарів. Ці модулі дозволяють оптимізувати процеси продажу та обслуговування клієнтів, забезпечуючи високий рівень задоволеності клієнтів та ефективне управління запасами.

CRM-система інтегрована з базою даних клієнтів, що дозволяє збирати, аналізувати та використовувати дані для покращення взаємодії з клієнтами. Модуль замовлень дозволяє обробляти замовлення клієнтів, відстежувати їх статус та управляти доставкою. Модуль продажу товару включає функції для реалізації товарів, встановлення цін та акцій.

Продовжуючи підрозділ 2.4, ми переходимо до детального опису реалізації функціональності створення нового продукту в рамках модуля управління складом. Сторінка create_product.php відіграє ключову роль у цьому процесі, дозволяючи адміністраторам магазину ефективно додавати нові товари до асортименту.

Сторінка create_product.php починається з ініціалізації сесії та перевірки на авторизацію користувача, що є важливим аспектом забезпечення безпеки веб-додатку. Якщо користувач не авторизований, відбувається перенаправлення на сторінку авторизації. Далі використовуються функції getCategories, getCharacteristics,

getMeasurementUnits та getManufacturers для завантаження списків категорій, характеристик, одиниць вимірювання та виробників, які будуть використовуватися при створенні нового продукту.

Функціональність форми на цій сторінці дозволяє користувачу ввести інформацію про новий продукт, включаючи назву, артикул, категорію, виробника, опис, ціну, залишок на складі, а також завантажити зображення продукту. Особливу увагу варто звернути на роботу з характеристиками продукту. За допомогою динамічного JavaScript-скрипту користувач має можливість додавати та видаляти рядки характеристик, обираючи відповідні значення з передвизначених списків.

Після заповнення форми і натискання на кнопку "Створити продукт" дані передаються на сервер. Якщо було завантажено зображення продукту, воно зберігається у відповідному каталозі. Використовуючи транзакції SQL, реалізовано збереження інформації про продукт та його характеристики в базі даних. У випадку успішного збереження, відбувається перенаправлення користувача на сторінку зі списком товарів.

Цей процес відображає глибину інтеграції між різними компонентами системи, демонструючи важливість кожного елемента в загальній архітектурі веб-додатку. Він також підкреслює здатність системи адаптуватися до різноманітних потреб користувачів, від простого додавання нового продукту до управління складними наборами характеристик продуктів.

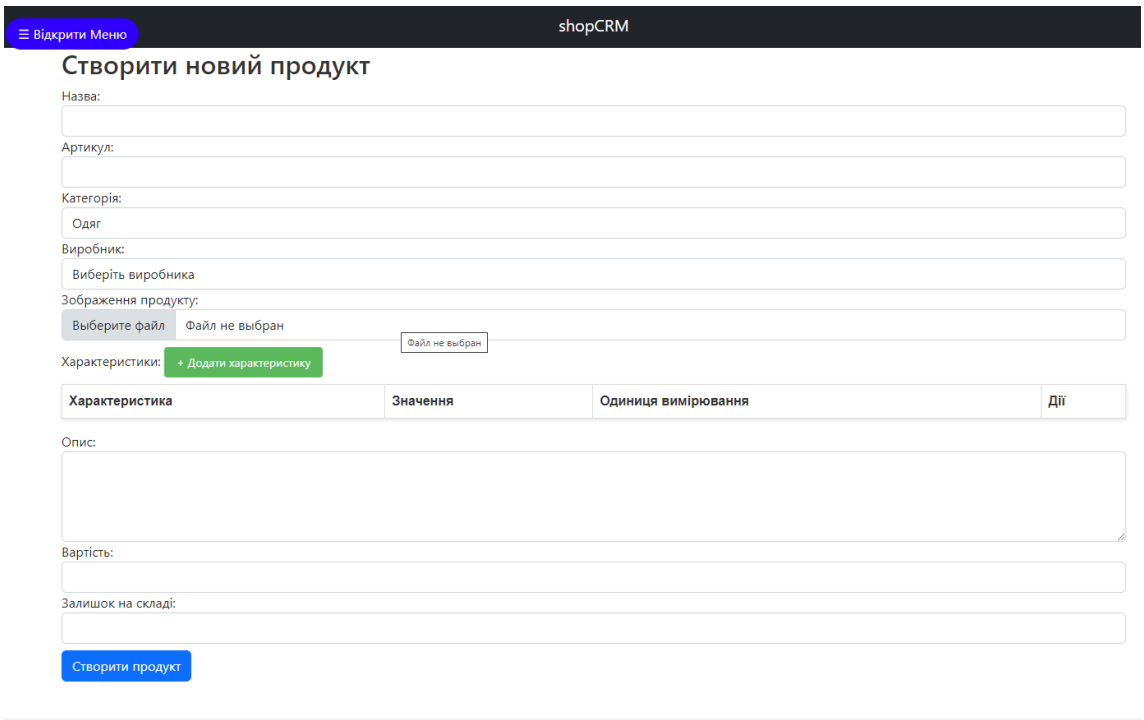

Рис. 2.3. Інтерфейс сторінки create product.php

Продовжуючи розділ 2.4 нашої дипломної роботи, ми фокусуємося на іншій важливій частині веб-додатку – реалізації функціоналу редагування категорій, представленому на сторінці edit categories.php. Ця сторінка є ключовою для управління категоріями товарів у магазині, дозволяючи адміністратору легко додавати, редагувати та видаляти категорії.

На сторінці edit categories.php реалізовано ієрархічний підхід до відображення категорій. Вона починається з ініціалізації сесії та перевірки авторизації користувача. Після цього відбувається завантаження основних категорій за допомогою функції getMainCategories, яка запитує дані з бази даних. Функції displayMainCategories та displaySubCategories відповідають за генерацію HTML-коду для відображення категорій та їх підкатегорій у вигляді вкладеного списку. Кожна категорія має кнопки для редагування та видалення.

Однією з ключових особливостей цієї сторінки є використання модальних вікон для додавання, редагування та видалення категорій. Це дозволяє користувачам виконувати ці дії без перезавантаження сторінки, забезпечуючи більш гладкий та інтуїтивно зрозумілий користувацький досвід. Коли користувач натискає на кнопку редагування чи видалення, відповідні дані категорії передаються в модальне вікно

через JavaScript. Це демонструє ефективне використання сучасних веб-технологій для створення динамічного та взаємодіючого інтерфейсу.

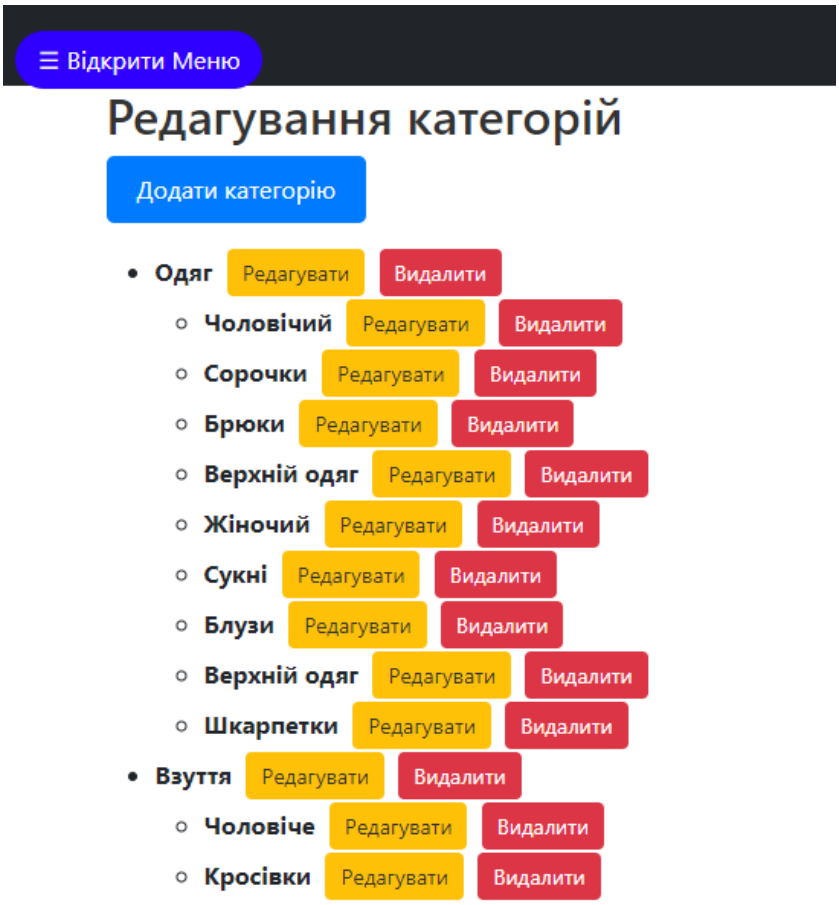

Рис. 2.4. Інтерфейс сторінки edit_categories.php

Завершуючи опис сторінки edit_categories.php, слід підкреслити, що ефективне управління категоріями є невід'ємною частиною успішного управління магазином. Ця сторінка демонструє гнучкість та масштабованість веб-додатку, дозволяючи легко адаптуватися до змін у асортименті товарів та потребах користувачів. Використання ієрархічної структури категорій, а також інтуїтивно зрозумілих інструментів для їх управління, значно спрощує процес навігації та пошуку товарів, як для адміністраторів, так і для кінцевих користувачів.

Далі розглядаємо сторінку edit_characteristics.php, яка відіграє ключову роль у управлінні характеристиками продуктів у веб-додатку. Ця сторінка дозволяє адміністраторам додавати, редагувати та видаляти характеристики продуктів, що є важливим для точного та ефективного каталогізування товарів.

Сторінка починається з ініціалізації сесії та перевірки на авторизацію користувача. Потім відбувається завантаження усіх характеристик за допомогою функції getAllCharacteristics. Важливою частиною сторінки є реалізація форм та модальних вікон для додавання, редагування та видалення характеристик.

Коли користувач натискає кнопку для створення нової характеристики, відкривається модальне вікно, де можна ввести назву характеристики. Аналогічно, для редагування існуючої характеристики використовується інше модальне вікно, куди передаються поточні дані характеристики. У випадку видалення, відображається підтвердження у модальному вікні. Ці дії відображаються у вигляді кнопок поруч з кожною характеристикою у табл. 2.5.

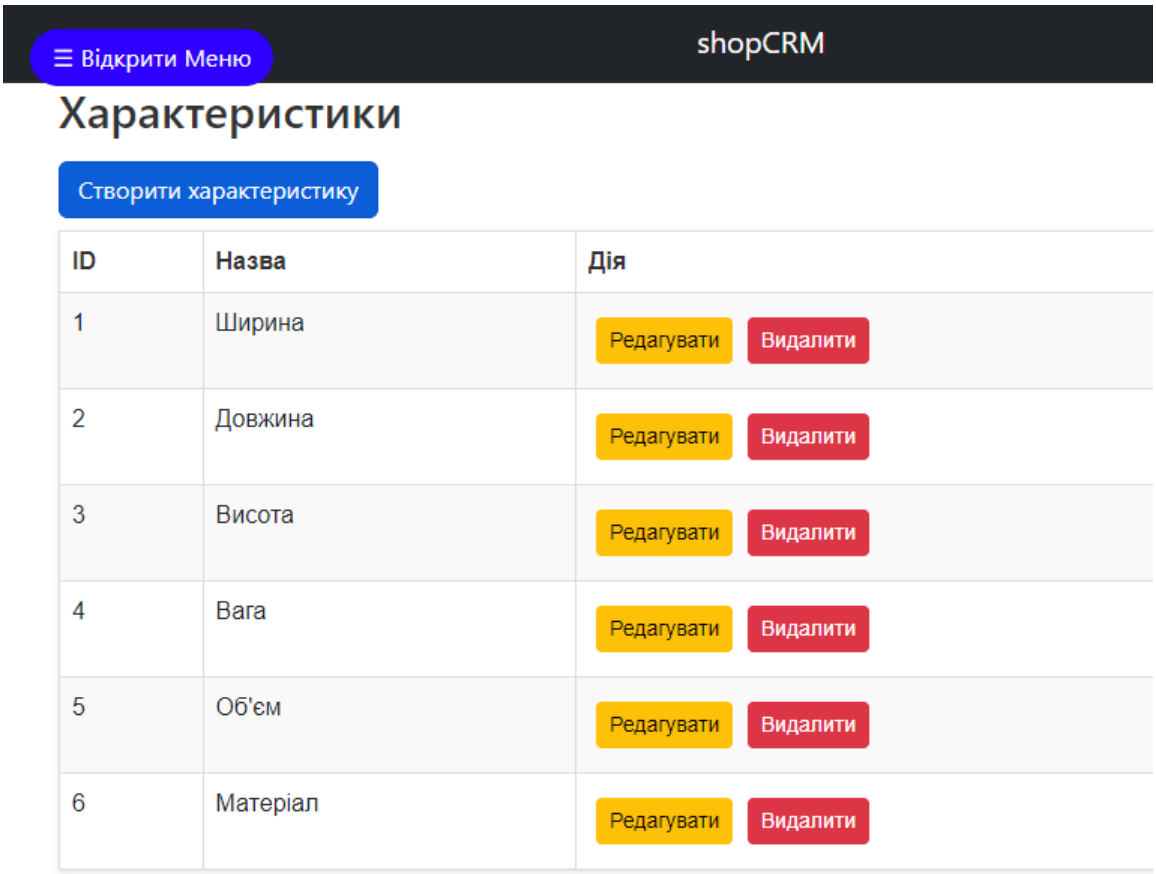

Рис. 2.5. Інтерфейс сторінки edit_characteristics.php

Завдяки JavaScript, ці модальні вікна легко інтегруються з таблицею характеристик. Коли користувач вибирає дію редагування чи видалення,

ідентифікатор та інша інформація про характеристику автоматично передаються до модального вікна, забезпечуючи зручність та ефективність взаємодії.

Ця сторінка демонструє важливість гнучкості в управлінні даними веб-додатку. Можливість швидкого та простого управління характеристиками продуктів є ключовою для забезпечення актуальності та точності інформації в каталозі магазину. Інтуїтивний інтерфейс з модальними вікнами та вбудованими формами робить цей процес не тільки ефективним, але й приємним для користувачів.

Завершуючи розгляд цього модуля, слід відзначити, що реалізація функцій управління характеристиками є важливою частиною забезпечення гнучкості та функціональності веб-додатку. Це дозволяє магазину швидко реагувати на зміни в асортименті товарів та потреби ринку, що є ключовим фактором успіху в сучасному роздрібному бізнесі.

Продовжуючи аналіз модулів веб-додатку, ми переходимо до розгляду функціоналу сторінки edit_units.php, яка займається управлінням одиницями вимірювання. Ця функціональність важлива для точного визначення характеристик продуктів, наприклад, ваги, об'єму, розмірів тощо, і є невід'ємною частиною ефективного управління інформацією про товари.

Структура сторінки edit_units.php слідує зрозумілому та логічному взаємодії. Починаючи з ініціалізації сесії та перевірки на авторизацію, сторінка надає інтерфейс для управління одиницями вимірювання. Ключовою частиною сторінки є таблиця, що відображає всі доступні одиниці вимірювання з можливістю їх редагування або видалення.

Для кожної одиниці вимірювання передбачені кнопки, які активують модальні вікна для внесення змін або видалення. Це забезпечує зручний та ефективний спосіб управління, дозволяючи адміністраторам легко оновлювати інформацію без необхідності перезавантаження сторінки.

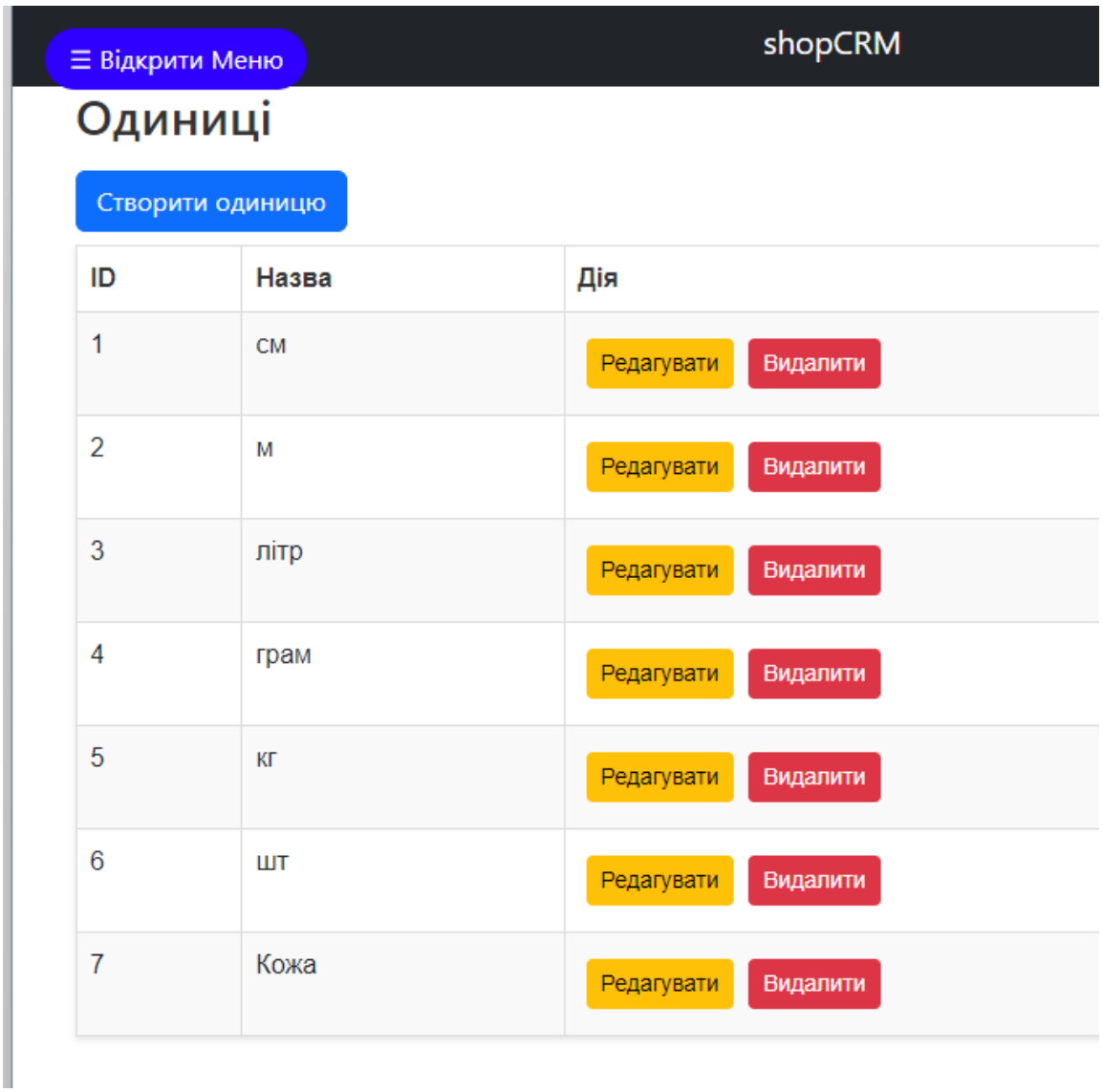

Рис. 2.6. Інтерфейс сторінки edit_units.php

Коли користувач натискає на кнопку для додавання нової одиниці вимірювання, відкривається відповідне модальне вікно, де можна ввести назву нової одиниці. Аналогічно, для редагування існуючої одиниці використовується інше модальне вікно, куди передаються поточні дані. Видалення одиниці також відбувається через модальне вікно, що запитує підтвердження дії.

Така організація інтерфейсу є виразом сучасних тенденцій розробки веб-додатків, де увага приділяється як функціональності, так і зручності користувача. Чітке розмежування дій, інтуїтивно зрозумілий інтерфейс і використання модальних вікон роблять процес управління одиницями вимірювання простим та зрозумілим.

Завершуючи розгляд цього компоненту, можна сказати, що управління одиницями вимірювання відіграє важливу роль у загальному управлінні асортиментом продукції. Ця функціональність забезпечує необхідну гнучкість та точність у категоризації товарів, що є важливим для забезпечення високої якості обслуговування клієнтів та ефективного управління запасами.

Продовжуючи глибокий аналіз модулів нашого веб-додатку, ми переходимо до розгляду модуля, присвяченого управлінню інформацією про виробників, який представлений сторінкою manufacturers.php. Цей модуль відіграє значущу роль у забезпеченні детальної інформації про виробників, що є необхідним для точної категоризації товарів та надання клієнтам повної інформації про продукти.

Сторінка manufacturers.php починається з ініціалізації сесії та перевірки авторизації користувача. Це гарантує, що доступ до інформації про виробників мають лише авторизовані користувачі, забезпечуючи безпеку даних. Після цього, за допомогою функції getManufacturers, завантажується список усіх виробників з бази даних.

Сторінка містить таблицю, яка відображає перелік усіх виробників разом з їх ID, назвою та описом. Для кожного виробника передбачені кнопки для перегляду деталей, редагування та видалення. Це надає адміністратору гнучкі можливості управління інформацією про виробників, дозволяючи оновлювати дані або видаляти їх із системи.

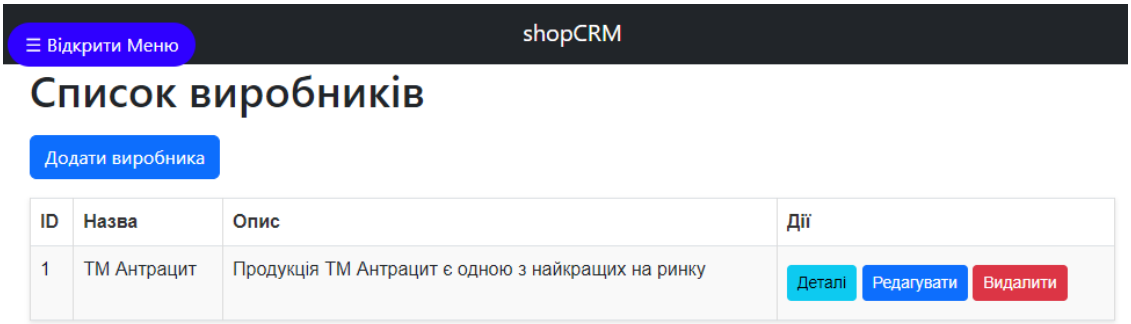

Рис. 2.7. Інтерфейс сторінки manufacturers.php

Особливо важливим є функціонал для видалення інформації про виробників, оскільки він впливає на точність і актуальність даних у каталозі товарів. Модуль включає підтвердження дій видалення, що допомагає запобігти випадковому видаленню важливої інформації.

Ця сторінка є прикладом ефективної інтеграції важливих бізнес-процесів у веб-додаток, що забезпечує адміністраторам магазину потужні інструменти для управління інформацією про виробників. Завдяки зручному інтерфейсу та чітко структурованій презентації даних, цей модуль вносить значний вклад у загальну ефективність управління продуктами та покращення обслуговування клієнтів.

Завершуючи розгляд модуля управління виробниками, важливо підкреслити, що такі інструменти не тільки спрощують роботу з базою даних продуктів, але й забезпечують точність та повноту інформації, доступної для клієнтів. Це сприяє підвищенню довіри клієнтів та їх задоволеності, а також підтримує репутацію магазину як надійного постачальника товарів.

Продовжуючи аналіз модулів нашого веб-додатку, ми переходимо до розгляду сторінки clients.php, яка відіграє важливу роль у управлінні інформацією про клієнтів. Цей модуль є фундаментальним для CRM-системи, оскільки надає можливості для ефективного управління клієнтською базою, включаючи додавання, редагування та видалення даних клієнтів.

На сторінці clients.php реалізовано функціональність, яка дозволяє адміністраторам переглядати список всіх клієнтів, зареєстрованих у системі. Кожен запис у таблиці включає ідентифікаційний номер клієнта, його ім'я, контактний телефон, дату додавання до системи та набір дій для керування цією інформацією.

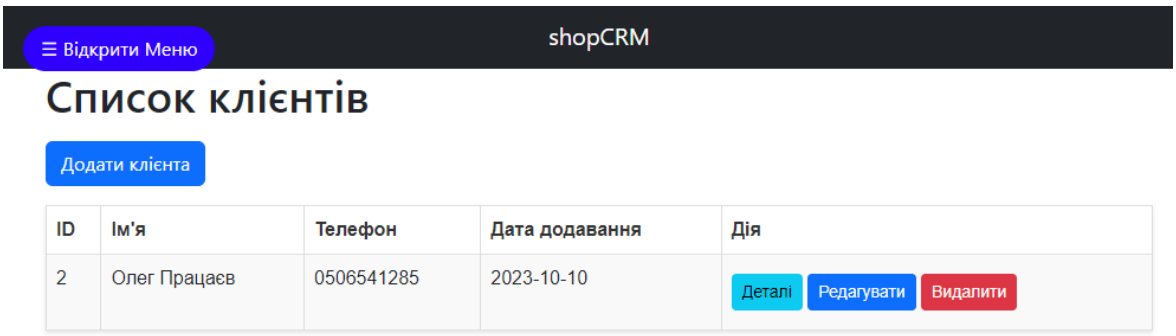

Рис. 2.8. Інтерфейс сторінки clients.php.

Користувачам надається можливість не тільки переглядати деталі про кожного клієнта, але й редагувати цю інформацію або навіть видаляти клієнтів із системи. Додавання нового клієнта також можливе за допомогою кнопки, яка веде до відповідної форми. Це забезпечує повний контроль над клієнтською базою, дозволяючи оперативно оновлювати інформацію згідно з поточними потребами бізнесу.

Функціонал сторінки показує важливість точного та актуального управління даними клієнтів у контексті CRM-системи. Це дозволяє не тільки підтримувати відносини з існуючими клієнтами, але й аналізувати клієнтську базу для розробки нових стратегій маркетингу та продажу.

Ця сторінка є чудовим прикладом того, як сучасні веб-технології можуть бути використані для створення ефективних і зручних інструментів управління. Інтуїтивно зрозумілий інтерфейс, логічна структура таблиці та простота використання роблять процес управління клієнтською базою швидким та ефективним.

Завершуючи розгляд модуля управління клієнтами, можна сказати, що він є ключовим елементом CRM-системи та відіграє важливу роль у загальній стратегії управління відносинами з клієнтами. Ефективне управління цією інформацією сприяє підвищенню задоволеності клієнтів, що є критично важливим для успіху в сучасному бізнес-середовищі.

Продовжуючи розділ 2.4, ми розглядаємо модуль замовлень, представлений сторінкою orders.php. Цей модуль має критичне значення для управління процесами

продажу, дозволяючи відстежувати всі замовлення, що були зроблені в магазині. Він є інтегральною частиною веб-додатку, спрямованого на автоматизацію та оптимізацію роботи магазину.

На сторінці orders.php представлена таблиця з переліком усіх замовлень. Для кожного замовлення відображаються такі дані, як ID замовлення, ім'я клієнта, сума замовлення та набір кнопок для управління (деталі, редагування, видалення). Такий підхід дозволяє адміністраторам швидко отримати необхідну інформацію та вжити відповідних дій щодо кожного замовлення.

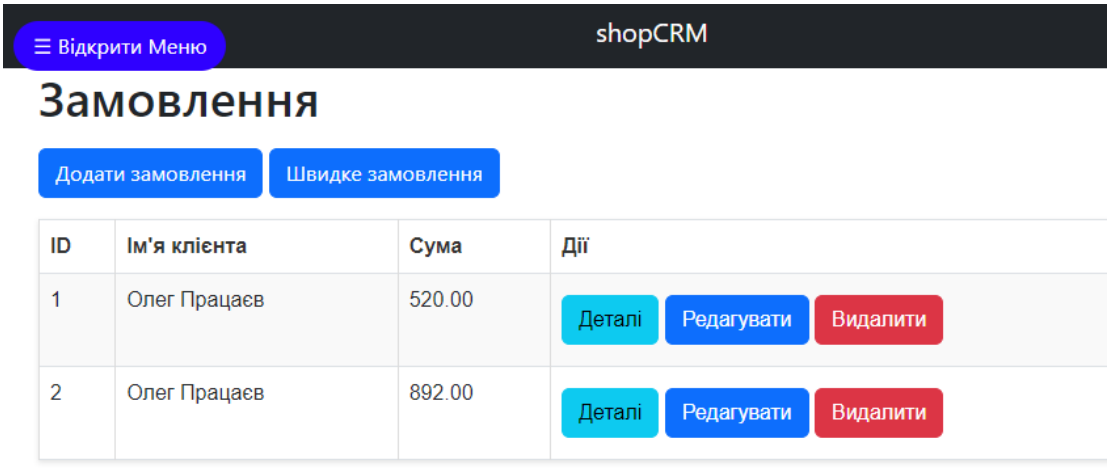

Рис. 2.9. Інтерфейс сторінки orders.php

Крім перегляду існуючих замовлень, адміністратори мають можливість додавати нові замовлення за допомогою кнопки "Додати замовлення". Це особливо важливо для випадків, коли замовлення приймається не через веб-інтерфейс, а, наприклад, за телефоном або в офлайн-магазині.

Кнопка "Швидке замовлення" слугує додатковим інструментом для спрощення процесу створення замовлень, дозволяючи швидко оформити замовлення з мінімальним введенням даних. Такий функціонал є особливо зручним для постійних клієнтів та швидких продажів.

Завершуючи аналіз модуля замовлень, слід підкреслити його значущість для загальної роботи веб-додатку. Цей модуль не тільки дозволяє ефективно управляти

замовленнями, але й відіграє важливу роль у підтримці клієнтських відносин, забезпечуючи високий рівень обслуговування. Він сприяє підвищенню задоволеності клієнтів та підтримує плавність бізнес-процесів, від замовлення товару до його доставки.

Продовжуючи розділ 2.4 нашого веб-додатку, ми переходимо до розгляду функціональності створення замовлення, представленої сторінкою create_order.php. Цей модуль є важливим елементом у процесі управління замовленнями, дозволяючи адміністраторам ефективно створювати нові замовлення, що є ключовим для підтримки потоку продажів та задоволення потреб клієнтів.

На сторінці create_order.php реалізована можливість вибору клієнта та додавання продуктів до замовлення. Це включає в себе динамічне створення рядків для вибору продуктів, кількості та автоматичне обчислення загальної суми замовлення. Такий підхід гарантує гнучкість та легкість у процесі створення замовлень.

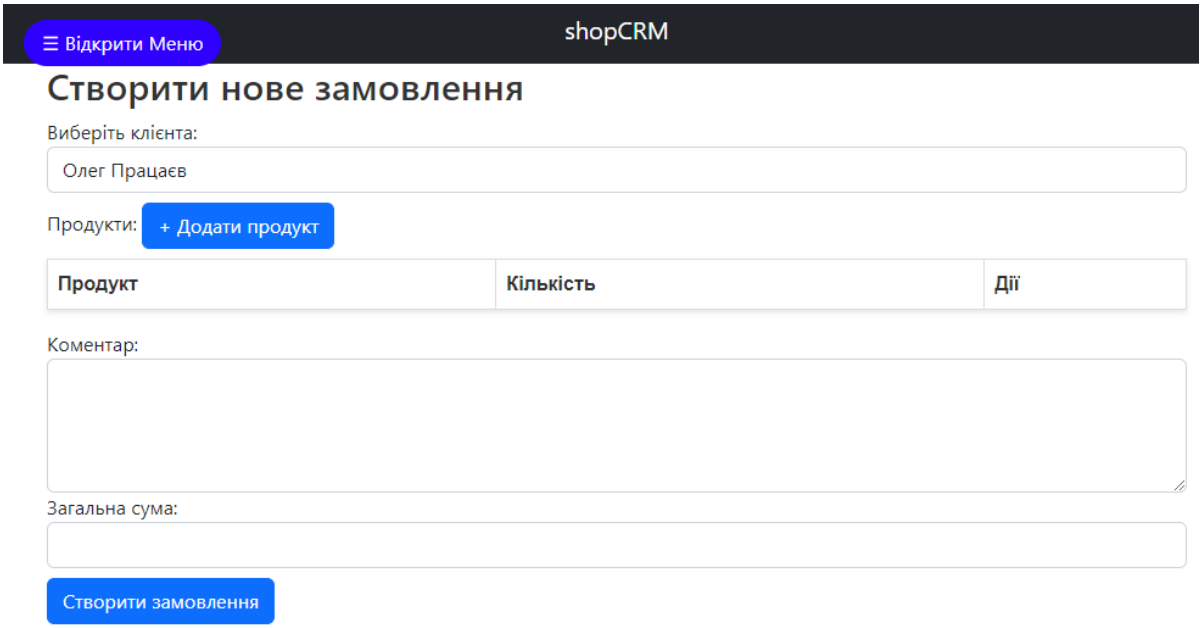

Рис. 2.10. Інтерфейс сторінки create_order.php

Ключовим елементом цієї сторінки є JavaScript-скрипт, який дозволяє динамічно додавати та видаляти рядки для продуктів у замовленні. Це надає користувачам можливість гнучко управляти складом замовлення, змінюючи вибір

продуктів та їхні кількості в реальному часі. Автоматичне обчислення загальної суми замовлення забезпечує точність розрахунків та підвищує ефективність процесу створення замовлення.

Завершення процесу створення замовлення відбувається натисканням кнопки «Створити замовлення», після чого дані передаються на сервер і обробляються.

2.5 Тестування розробленого веб-додатку

Тестування розробленого веб-додатку є ключовим етапом, який забезпечує його надійність, безпеку та коректність функціонування перед впровадженням у реальні умови експлуатації. В рамках цього процесу було проведено ряд тестів, що охоплюють різні аспекти додатку, включаючи функціональні, вантажні, безпекові та користувацькі тести. Метою тестування було ідентифікувати та виправити потенційні проблеми, що могли б вплинути на ефективність та надійність веб-додатку.

Тестування веб-додатку проводилося в декілька етапів. Починаючи з ручних тестів, які допомагали виявити помилки у функціональності та інтерфейсі користувача, ми переходили до автоматизованих тестів для перевірки вантажності та безпеки системи. Використання автоматизованих інструментів дозволило нам виконати глибокий аналіз коду, перевірити його на вразливості та ефективно симулювати різні сценарії використання додатку.

Таблиця 2.1

«Результати тестування» презентує зведену інформацію за кожним типом тестів

Тип тесту	Кількість тестів	Успішно пройдено	Неуспішно пройдено	Зауваження
Функціональне тестування	50	48	2	Виправлені помилки в модулі редагування продуктів
Вантажне тестування	10	10	0	Оптимальна робота під високим навантаженням
Безпекове тестування	20	18	2	Виявлено вразливості SQL-ін'єкцій, вже виправлено
Користувацьке тестування	30	27	3	Зауваження щодо інтуїтивності інтерфейсу виправлено

Загальні результати тестування показали високий рівень готовності веб-додатку до впровадження. Однак, виявлені проблеми вимагали виправлення перед фінальним запуском. Найважливішими аспектами, на які було звернуто увагу, стали вразливості безпеки та покращення користувацького інтерфейсу.

На основі результатів тестування були розроблені наступні рекомендації:

- провести додатковий рефакторинг коду для усунення виявлених помилок;
- збільшити покриття автоматизованими тестами для забезпечення більш ретельної перевірки функціональності;
- впровадити додаткові заходи безпеки для захисту від SQL-ін'єкцій та інших потенційних загроз;
- продовжити роботу над удосконаленням користувацького інтерфейсу, зосередившись на інтуїтивності та зручності використання.

Підсумовуючи, тестування виявило деякі проблеми, але в цілому підтвердило високу якість і стабільність веб-додатку. Запропоновані заходи спрямовані на підвищення надійності та забезпечення безпечної та ефективної роботи системи у довгостроковій перспективі.

Висновок до розділу 2

У розділі 2 нашої дипломної роботи, присвяченому розробці веб-додатку для автоматизації роботи магазину, ми провели всебічний аналіз та реалізацію ключових компонентів системи. Основна увага була зосереджена на вимогах до системи, її архітектурі, інтерфейсу, реалізації основних модулів, а також на тестуванні розробленого веб-додатку. В розділі 2.1 ми визначили фундаментальні вимоги до системи та її функціональні можливості, поклавши основу для подальшої розробки. Це включало управління продуктами, замовленнями, клієнтами, а також забезпечення високого рівня безпеки та легкості у використанні.

В розділі 2.2 ми зосередилися на проектуванні структури бази даних, що є критично важливим для забезпечення ефективності обробки даних та їх зберігання. Було розроблено оптимальну схему бази даних, яка дозволяє легко масштабувати та адаптувати систему до змінних вимог бізнесу.

У розділі 2.3 ми присвятили увагу розробці архітектури та інтерфейсу веб-додатку. Архітектура була спроектована з метою забезпечення високої продуктивності, надійності та легкості обслуговування. Інтерфейс створено інтуїтивно зрозумілим і зручним для користувача, що забезпечує ефективне взаємодіяння з системою.

Розділ 2.4 охопив реалізацію основних модулів веб-додатку, включаючи управління складом, CRM-систему, модуль замовлень та модуль продажу товару. Кожен модуль був ретельно спроектований та розроблений, щоб відповідати конкретним потребам бізнесу та забезпечити високий рівень функціональності.

У розділі 2.5 ми провели тестування розробленого веб-додатку, щоб забезпечити його надійність та відповідність встановленим вимогам. Тестування включало функціональні, вантажні, безпекові та користувацькі тести, які допомогли виявити та усунути потенційні проблеми. Загалом, цей розділ демонструє глибокий і всебічний підхід до розробки веб-додатку, з акцентом на його функціональності, надійності, безпеці та користувацькому досвіді. Результати, досягнуті на цьому етапі, становлять міцну основу для наступних етапів проекту та його успішного впровадження в експлуатацію.

РОЗДІЛ 3
ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ ВЕБ-ДОДАТКУ

3.1 Рекомендації щодо впровадження веб-додатку в роботу магазину

Успішне впровадження будь-якої інформаційної системи є комплексним завданням, яке вимагає ретельного планування та врахування багатьох аспектів. Це особливо актуально для веб-додатків, орієнтованих на автоматизацію конкретних бізнес-процесів, таких як управління роботою магазину.

Ефективна інтеграція подібних систем в операційну діяльність є запорукою швидкого отримання результату та окупності інвестицій в розробку. Тому дуже важливо з самого початку передбачити комплекс заходів, які дозволять якісно та безболісно впровадити веб-додаток в роботу магазину.

У цілому, розробка ЕСУП вимагає глибокого розуміння потреб бізнесу, знання сучасних технологій та вміння врахувати вимоги безпеки, користувацької зручності, масштабованості та інтегрованості. Врахування цих вимог є ключовим для створення ефективної, зручної та надійної системи, яка буде відповідати потребам сучасного управління персоналом і забезпечить компанію необхідними інструментами для досягнення її стратегічних цілей.

Проведення якісного та всебічного навчання співробітників є запорукою успішного впровадження будь-якої інформаційної системи, оскільки саме працівники будуть безпосередніми користувачами веб-додатку у своїй повсякденній роботі.

Рекомендується розробити детальну програму навчання з урахуванням різних ролей користувачів та специфіки їх взаємодії з системою. Наприклад, для адміністраторів системи доцільно передбачити більш глибоке вивчення технічних аспектів, можливостей конфігурації та налаштувань. Для менеджерів модулі CRM та управління продажами мають пріоритет.

Формат навчання може включати як групові семінари та тренінги, так і персональне наставництво для ключових співробітників. Останнє особливо ефективне при роботі з модулями, пов'язаними з основним функціоналом магазину.

Важливою складовою є розробка докладних інструкцій користувача з великою кількістю скріншотів та практичних кейсів. Така технічна документація сприятиме кращому розумінню системи.

Після початкового навчання рекомендується запровадити постійно діючу службу технічної підтримки (Help Desk), яка допомагатиме співробітникам оперативно вирішувати будь-які питання та проблеми.

Для закріплення знань ефективно проводити регулярні опитування, тестування користувачів, а також надавати доступ до бази знань та можливостей самонавчання.

Рис. 3.1. Рекомендації щодо впровадження розробленого веб-додатку для автоматизації роботи магазину

Підготовка та перевірка вхідних даних, які будуть завантажені у веб-додаток, є дуже важливим етапом, оскільки саме від якості та повноти цих даних залежатиме коректність подальшої роботи системи. Некоректні або неповні вхідні дані можуть спричинити багато проблем та ускладнити процес впровадження.

Особливу увагу варто приділити підготовці масивів довідникової інформації - класифікаторів товарів, характеристик продукції, довідників клієнтів, постачальників тощо. Ці дані мають бути максимально повними та верифікованими.

Також вкрай важливою є синхронізація фактичних залишків товарів на складах із каталожними даними. Необхідно звірити наявні запаси з інформацією, яка буде перенесена у веб-додаток для уникнення розбіжностей.

Крім того, потрібно не забути про історичні дані, такі як архів попередніх замовлень клієнтів, реєстр проведених операцій тощо. Ця інформація також має бути експортована у потрібному форматі.

Для зручності перенесення даних рекомендується підготувати шаблони export/import з описом структур файлів. Це полегшить процес завантаження початкових масивів і уникнення можливих помилок.

Для мінімізації ризиків та забезпечення безперервності бізнес-процесів, вкрай важливо реалізувати поступовий підхід до впровадження системи замість одночасного повноцінного запуску в експлуатацію.

На першому етапі рекомендується налаштувати тестове середовище, яке дозволить комплексно відпрацювати всі задіяні функції, сценарії використання, завантаження початкових даних тощо.

Після успішного тестування можна переходити до пілотного запуску в рамках одного підрозділу чи навіть на одному робочому місці, не припиняючи існуючих «ручних» процесів. Це надасть можливість паралельно опрацьовувати операції в обох режимах та порівнювати результати.

На основі пілоту слід скласти перелік доробок та оптимізацій, які після реалізації дозволять перейти до наступного рівня впровадження. Поступово охоплюючи все більшу частину підрозділів та функцій.

Тільки пересвідчившись в стабільності та коректності роботи системи, можна повністю ліквідувати старі «ручні» процеси і перевести магазин на повноцінне використання веб-додатку в усіх основних бізнес-операціях.

Такий еволюційний план впровадження дозволить уникнути раптових збоїв та зупинок бізнесу, забезпечуючи його безперебійне функціонування на всіх етапах автоматизації за допомогою веб-додатку.

Забезпечення оперативної технічної підтримки користувачів, особливо на початковому етапі впровадження веб-додатку, є невід'ємною запорукою успішності всього проекту автоматизації.

Саме в період первинного ознайомлення та освоєння системи у користувачів виникає найбільше питань та проблем, швидке вирішення яких запобігає розчаруванню в новому рішенні.

Рекомендується із самого старту запуску системи організувати цілодобову гарячу лінію підтримки для оперативного реагування на всі запити користувачів. Відповіді мають надаватися максимально швидко і з високим рівнем емпатії.

Доцільно сформувати базу знань з відповідями на найбільш поширені питання для прискорення обробки типових звернень.

Актуально на початковій стадії організувати «тіньову» підтримку, коли разом із користувачем працює фаховий технічний консультант, допомагаючи в реальному часі з будь-яких питань.

Особливу увагу приділити навчанню і стимулюванню внутрішніх IT-фахівців замовника, які з часом мають перебрати функції підтримки на себе. Своєчасна та якісна технічна підтримка в період впровадження сприяє швидкому зануренню користувачів у роботу з системою, попереджає розчарування та знижує опір змінам під час автоматизації.

Отже, впровадження веб-додатку для автоматизації магазину є комплексним завданням, успішне виконання якого потребує ретельного планування та врахування багатьох чинників.

Ключовими етапами, які забезпечать ефективну інтеграцію системи в операційну діяльність, є підготовка вхідних даних, проведення поетапного запуску, організація постійної технічної підтримки, налаштування інтеграції з іншими системами, а також комплексне навчання співробітників.

Саме врахування цих рекомендацій допоможе звести до мінімуму можливі проблеми і уникнути збоїв у роботі магазину при автоматизації ключових бізнес-процесів. Як наслідок, це дозволить максимізувати ефективність використання веб-додатку та швидко отримати очікувані результати від інвестицій в його розробку.

3.2 Інструкція користувача веб-додатку

Вступ до інструкції користувача веб-додатку розпочинається з опису процесу авторизації. Це перший крок, який зустрічає користувачів при роботі з нашим веб-додатком. Сторінка авторизації є важливою, оскільки вона забезпечує безпечний доступ до ресурсів системи. Користувачам необхідно ввести свої облікові дані, включаючи ім'я користувача та пароль, щоб увійти до системи.

Після завантаження сторінки авторизації, користувачі бачать форму входу, яка містить поля для введення імені користувача та пароля. У випадку, якщо користувач вже має активну сесію, система автоматично перенаправить його на головну сторінку. Це запобігає повторному входу в систему.

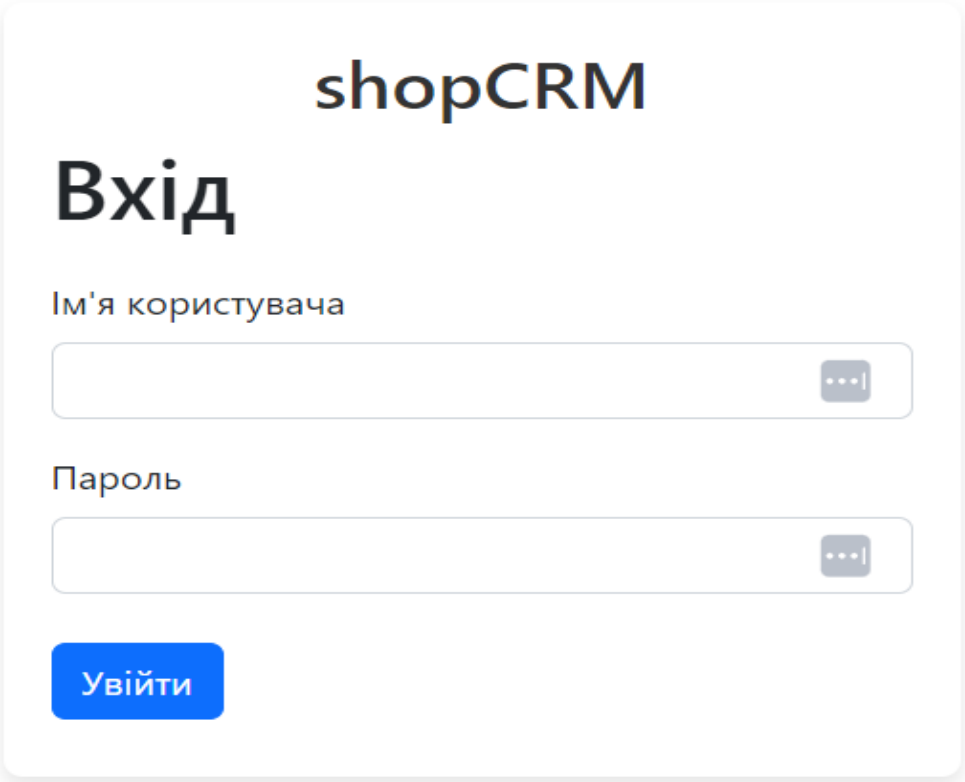

Рис. 3.2. Вигляд сторінки авторизації

У разі введення неправильних даних для входу, система виводить повідомлення про помилку, інформуючи користувача про неправильний логін або пароль. Це важливо для забезпечення безпеки доступу до веб-додатку. У разі успішної авторизації, користувач автоматично перенаправляється на головну сторінку системи.

Ця сторінка є входом до системи управління магазином, де користувачі можуть взаємодіяти з різними модулями веб-додатку. Важливість сторінки авторизації полягає не лише у забезпеченні доступу до функцій системи, але й у підтриманні безпеки користувацьких даних та інформації магазину.

У цілому, процес авторизації у веб-додатку розроблений таким чином, щоб забезпечити легкий та безпечний доступ до системи для користувачів. Зручний та інтуїтивно зрозумілий інтерфейс сторінки авторизації сприяє позитивному користувацькому досвіду та є важливою частиною загальної ефективності веб-додатку. структури бази даних для електронної системи управління персоналом (ЕСУП) є критичним етапом, який визначає ефективність та надійність усієї системи. Це вимагає ретельного планування та аналізу, щоб забезпечити, що база даних буде здатна зберігати, обробляти та надавати доступ до інформації про персонал швидко, безпечно та ефективно.

Після успішного входу в систему користувач потрапляє на головну сторінку веб-додатку, яка є центральним вузлом для доступу до всіх основних функцій та модулів управління магазином. Головна сторінка зроблена таким чином, щоб забезпечити користувачам зручний огляд ключових показників діяльності магазину, а також швидкий доступ до різноманітних функцій управління.

Панель навігації веб-додатку "shopCRM" грає ключову роль у забезпеченні легкого доступу до різних секцій та функцій системи. Вона розроблена таким чином, щоб користувачі могли легко переміщатися між різними частинами додатку, зберігаючи час та ефективність роботи.

Панель розгортається з лівого боку сторінки і містить посилання на основні розділи додатку. Це включає "Головну" сторінку, де користувачі можуть переглядати загальну інформацію та статистику, "Продукти" для управління асортиментом

товарів, "Виробників" для перегляду та управління інформацією про виробників, "Категорії" для організації товарів за категоріями, "Клієнти" для перегляду та управління даними клієнтів, та "Замовлення" для обробки та управління замовленнями.

Доступ до панелі навігації контролюється за допомогою сесійних перевірок, які гарантують, що тільки авторизовані користувачі можуть переглядати та взаємодіяти з панеллю. Це забезпечує додатковий рівень безпеки та зберігає інформацію від несанкціонованого доступу.

Панель має інтерактивну кнопку "Відкрити Меню", яка дозволяє користувачам відкривати та закривати панель навігації. Це забезпечує гнучкість, дозволяючи користувачам звільнити простір на екрані під час роботи в інших розділах додатку.

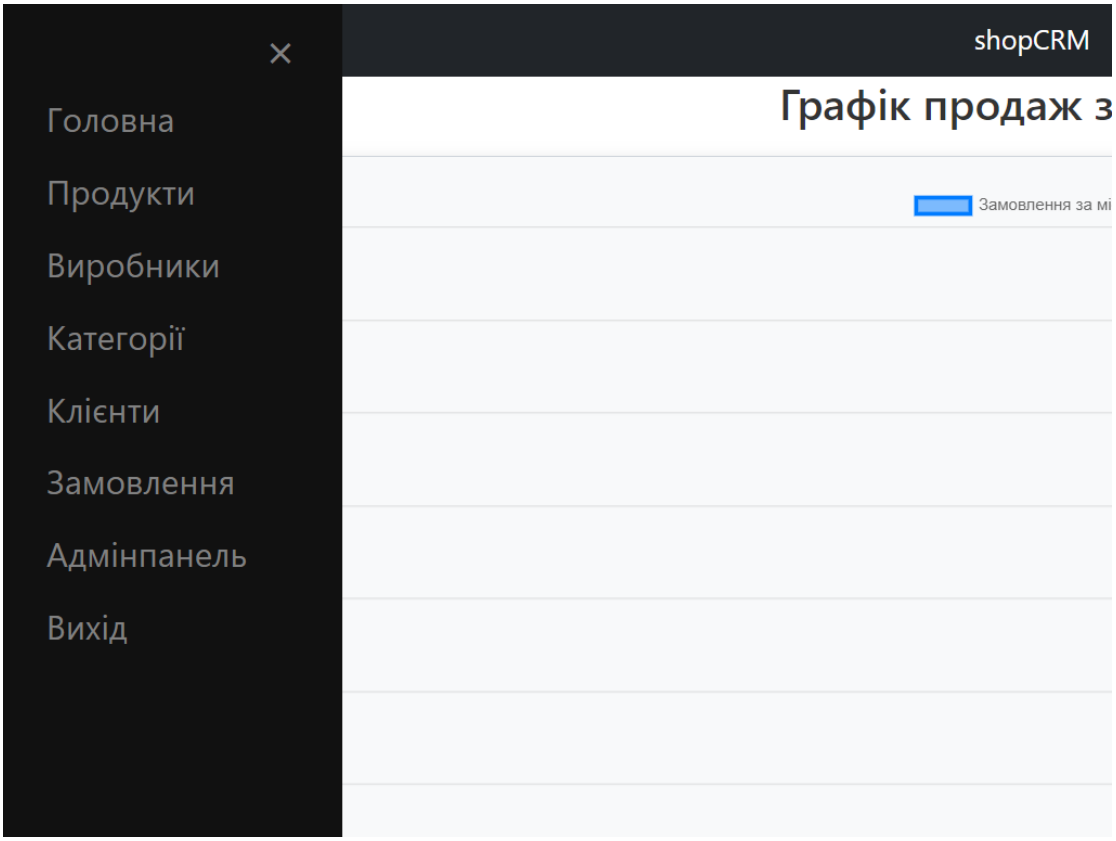

Рис. 3.3. Панель навігації shopCRM

Останнім елементом панелі навігації є посилання "Вихід", яке дозволяє користувачам безпечно вийти з системи. Це гарантує, що сесія користувача буде закрита після завершення роботи, підтримуючи загальну безпеку даних у системі.

Однією з важливих функцій на головній сторінці є графік продажів за місяць. Цей графік надає користувачам візуальне уявлення про динаміку продажів, дозволяючи легко відстежувати зміни та виявляти тенденції. Дані для графіка беруться безпосередньо з бази даних і відображаються у форматі лінійної діаграми, забезпечуючи інтуїтивно зрозумілу інформацію про продажі за різні місяці.

Графік продажів за місяць є інтерактивним. Користувачі можуть переглядати конкретні дані за кожен місяць, наводячи курсор на точки графіка. Це дозволяє отримати більш детальну інформацію про продажі в певний період. Для створення графіка використовується JavaScript-бібліотека Chart.js, яка забезпечує гнучкість та високу якість візуалізації даних.

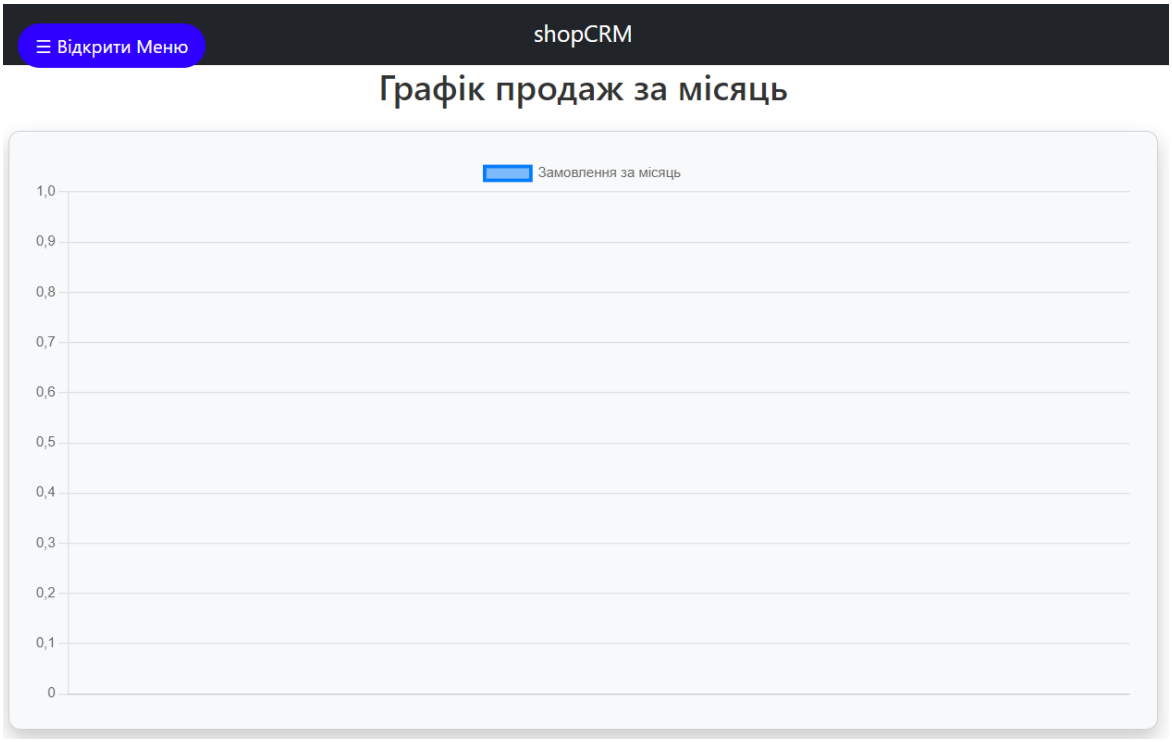

Рис. 3.4. Головна сторінка з графіком продажів

Крім графіка, головна сторінка також містить навігаційні елементи, що дозволяють користувачам легко переміщатися між різними розділами веб-додатку. Це може включати посилання на управління продуктами, замовленнями, клієнтською базою та інші важливі функції.

Головна сторінка веб-додатку виконує роль інформаційного та навігаційного центру, де користувачі можуть отримати швидкий огляд ключових аспектів бізнесу та легко доступити необхідні функції управління. Інтуїтивно зрозумілий інтерфейс, ефективна візуалізація даних та легка навігація роблять головну сторінку незамінною для ефективної роботи з веб-додатком.

Після переходу на сторінку управління товарами користувач відкриває один з ключових компонентів веб-додатку, який дозволяє ефективно управляти асортиментом продукції магазину. Ця сторінка демонструє здатність системи до організації, презентації та модифікації інформації про товари.

Сторінка управління товарами починається з візуального представлення списку товарів, що включає основні деталі кожного продукту, такі як ID, назва, категорія, виробник, вартість та кількість на складі. Ця інформація представлена у вигляді зручної таблиці, що дозволяє користувачам швидко оцінити та працювати з даними.

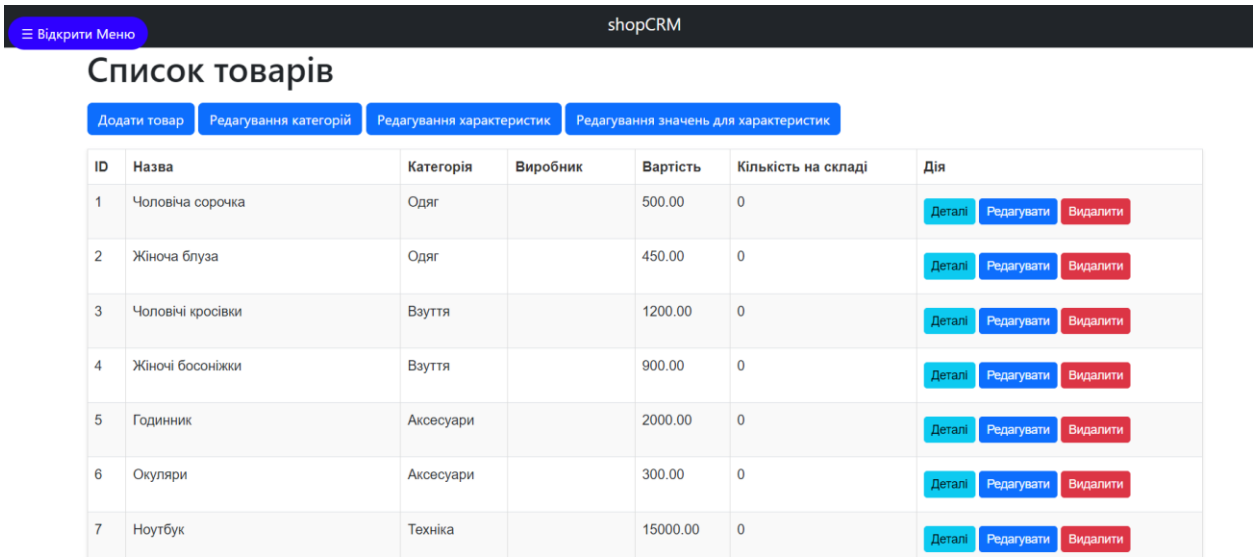

Рис. 3.5. Сторінка управління товарами

На цій сторінці користувачі мають можливість виконувати різноманітні дії з управління товарами. Найважливішими функціями є:

Додавання нового товару: За допомогою кнопки "Додати товар" користувачі можуть перейти до форми для створення нового товару, де вони можуть ввести всю необхідну інформацію.

Редагування товарів: Ця опція дозволяє користувачам оновлювати інформацію про існуючі товари. Натискання на кнопку "Редагувати" поряд з кожним товаром відкриває форму для внесення змін.

Видалення товарів: Якщо товар більше не потрібний у системі, він може бути видалений за допомогою кнопки "Видалити". Ця дія вимагає підтвердження, щоб запобігти випадковому видаленню.

Деталі товару: Для перегляду детальної інформації про товар можна скористатися кнопкою "Деталі".

Окрім основних функцій управління товарами, сторінка також містить посилання на редагування категорій, характеристик та одиниць вимірювання. Це забезпечує гнучке управління всіма аспектами товарів у магазині.

Сторінка управління товарами є важливою частиною веб-додатку, яка забезпечує ефективне та зручне управління продуктовим портфелем магазину. Інтуїтивно зрозумілий інтерфейс, зручні функції управління та візуальна презентація даних роблять цю сторінку незамінною для ефективної роботи з асортиментом товарів.

Розділ "Створення нового продукту" є одним із ключових аспектів управління товарним асортиментом магазину. На цій сторінці користувачі мають можливість додавати нові товари до каталогу, заповнюючи відповідну інформацію та деталі продукту. Процес створення продукту включає в себе кілька етапів, починаючи від введення базової інформації до встановлення характеристик та вартості.

На цій сторінці користувачі починають з введення основної інформації про продукт, включаючи його назву, артикул (SKU), опис, категорію, виробника та ціну. Всі ці поля є обов'язковими для заповнення, що гарантує повноту даних про кожен товар.

Користувачі мають можливість вибрати категорію та виробника продукту зі списків, які містять усі наявні в системі опції. Це дозволяє класифікувати товари та спрощує їх пошук та управління в майбутньому.

Далі користувачі можуть додати специфічні характеристики для товару, такі як колір, розмір чи будь-які інші важливі параметри. Це робиться шляхом додавання

нових рядків у таблицю характеристик, де можна вибрати відповідну характеристику, її значення та одиницю вимірювання. Також користувачі мають можливість завантажити зображення продукту, що значно підвищує його привабливість для потенційних покупців.

Після заповнення усієї необхідної інформації користувачі можуть переглянути підсумок та підтвердити створення нового продукту, натиснувши на кнопку "Створити продукт". Ця дія зберігає всю введену інформацію в базі даних та додає продукт до каталогу магазину.

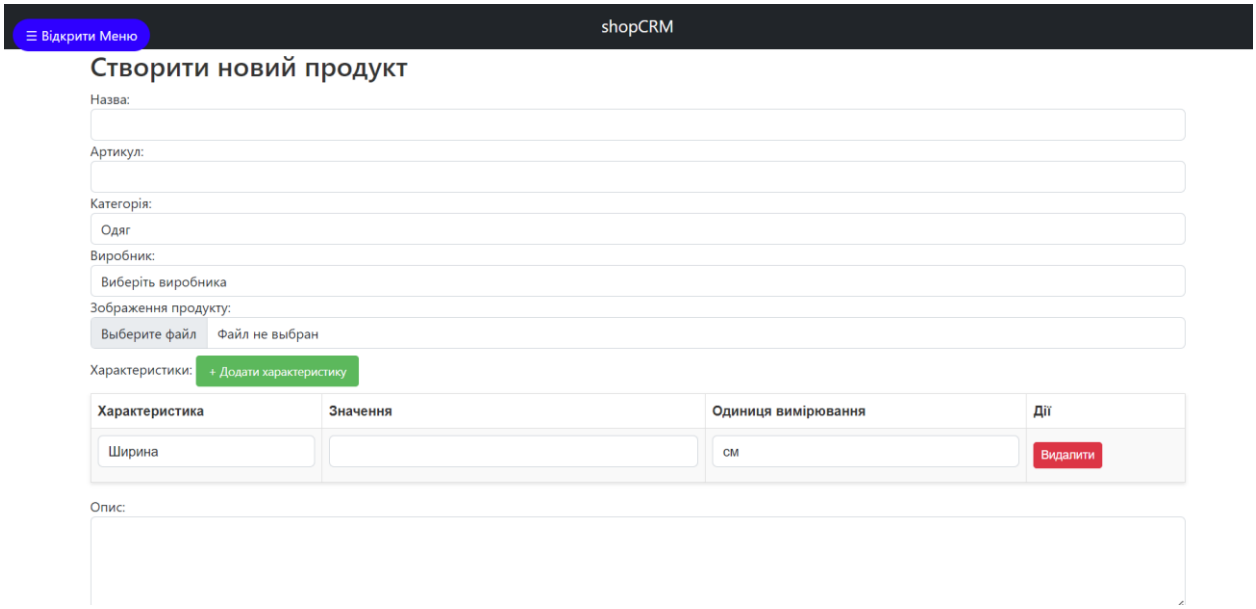

Рис. 3.6. Сторінка створення нового продукту

Створення нового продукту в веб-додатку є простим та інтуїтивно зрозумілим процесом. Завдяки чіткій структурі та легкості використання, ця сторінка дозволяє користувачам ефективно управляти асортиментом магазину, роблячи цей процес швидким та безпроблемним.

Сторінка "Редагування категорій" в веб-додатку є фундаментальною для управління класифікацією товарів. На цій сторінці користувачі мають можливість створювати, редагувати та видаляти категорії продуктів, що дозволяє ефективно організувати асортимент магазину.

Процес створення нової категорії починається з натискання на кнопку "Додати категорію", яка відкриває модальне вікно. У цьому вікні користувачам необхідно

ввести назву категорії, надати опис та, за необхідності, завантажити зображення для категорії. Також можна вказати батьківську категорію, що дозволяє створювати ієрархічну структуру категорій.

Для редагування категорії користувачі можуть використовувати кнопку "Редагувати", розташовану поряд з кожною категорією. Після натискання на цю кнопку відкривається модальне вікно, де можна змінити назву, опис, зображення та батьківську категорію. Ця функція дозволяє легко оновлювати інформацію про категорії без необхідності їх повного видалення та створення заново.

Користувачі також можуть видаляти категорії, використовуючи кнопку "Видалити". При спробі видалення система відображає модальне вікно підтвердження, щоб уникнути випадкового видалення важливих категорій.

Сторінка відображає категорії у формі ієрархічного дерева, де батьківські категорії відображаються зі своїми підкатегоріями. Це надає користувачам чітке уявлення про структуру категорій та їх взаємозв'язки.

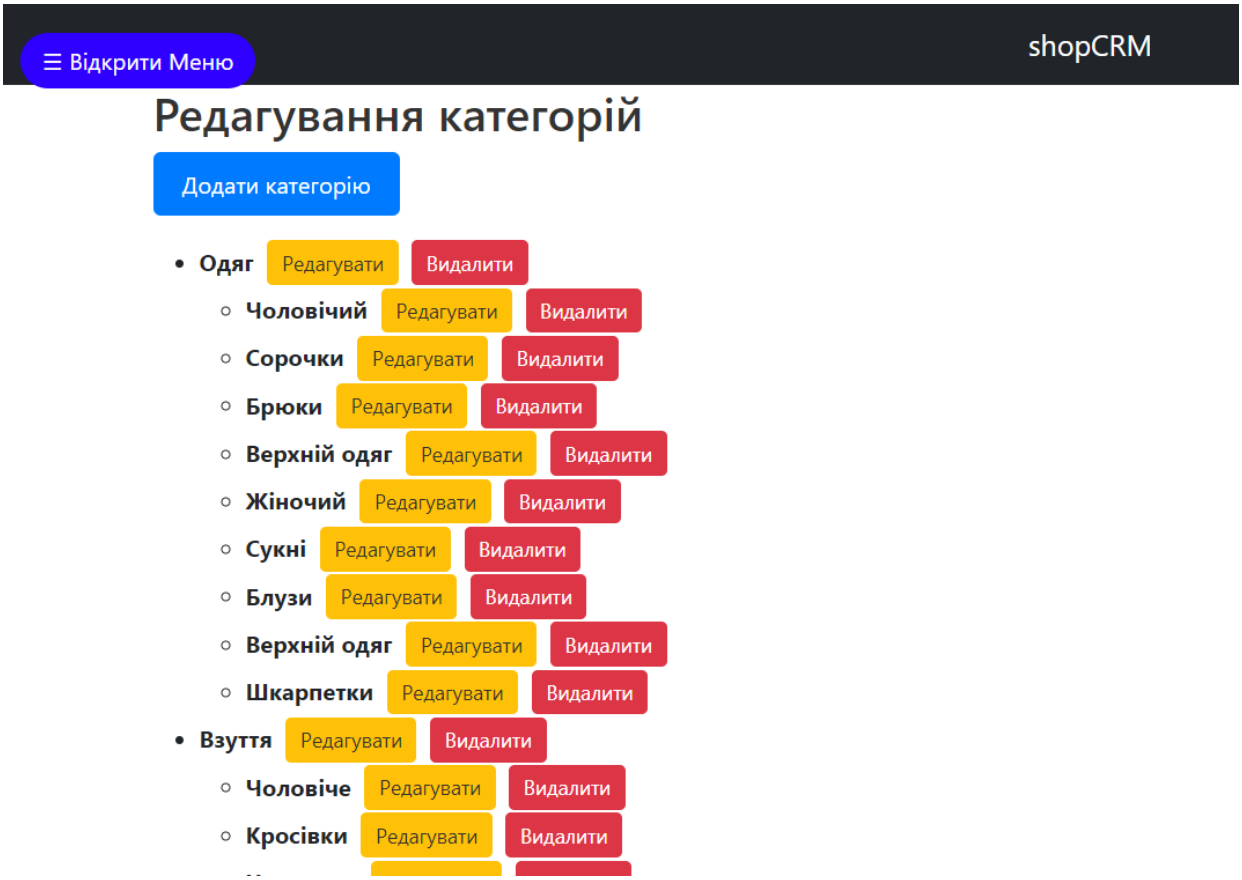

Рис. 3.7. Сторінка редагування категорій

Функціонал сторінки "Редагування категорій" забезпечує гнучке управління категоріями продуктів, що є ключовим для організації товарів у магазині. Інтуїтивно зрозумілий інтерфейс та легкість використання роблять цей процес ефективним та зручним для користувачів.

Сторінка "Редагування характеристик" відіграє ключову роль у детальному описі товарів магазину. Тут користувачі можуть додавати, редагувати, або видаляти різноманітні характеристики продуктів, такі як розмір, колір, матеріал та інші.

Для додавання нової характеристики користувачам потрібно натиснути кнопку "Створити характеристику". Ця дія відкриває модальне вікно, де можна ввести назву нової характеристики. Після заповнення поля і натискання кнопки "Зберегти", характеристика додається до списку.

Кожна характеристика в списку містить кнопки для редагування та видалення. При натисканні на кнопку "Редагувати" відкривається модальне вікно, де можна змінити назву характеристики. Це дозволяє користувачам швидко оновлювати інформацію без необхідності створення нової характеристики.

Для видалення характеристики користувачі використовують кнопку "Видалити". Після її натискання з'являється модальне вікно для підтвердження видалення, що допомагає уникнути випадкового видалення важливих даних.

Характеристики відображаються у вигляді таблиці, що робить легким орієнтування серед великої кількості інформації. Це спрощує пошук та управління характеристиками.

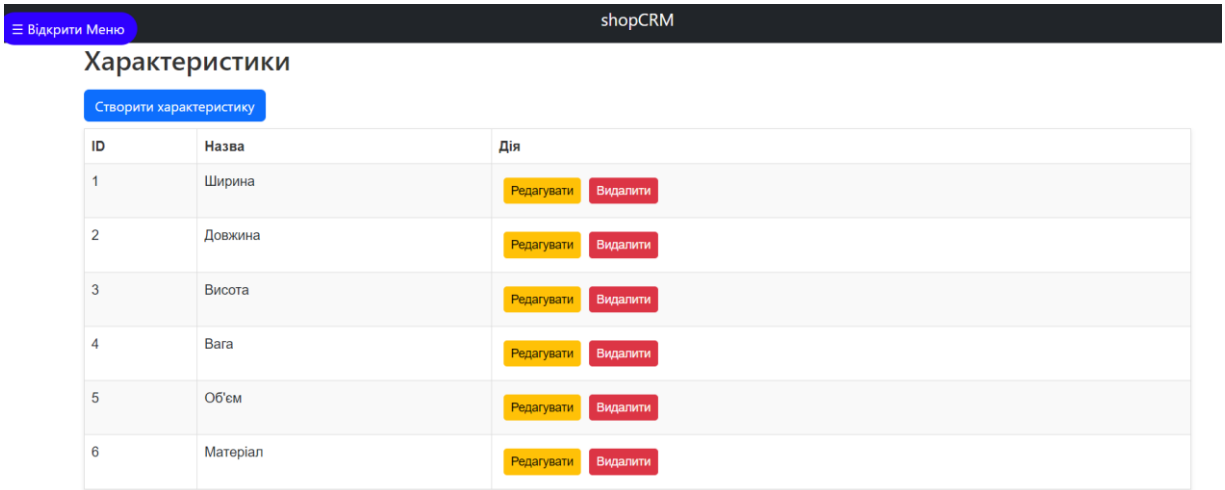

Рис. 3.8. Сторінка редагування характеристик

Функціональність сторінки "Редагування характеристик" забезпечує ефективне управління деталями продуктів, що є важливим для точного та ясного представлення товарів. Легкість використання та інтуїтивно зрозумілий інтерфейс роблять цей процес простим та зручним для користувачів.

Сторінка "Редагування одиниць вимірювання" є фундаментальною для управління та деталізації характеристик товарів. Тут користувачі можуть створювати нові одиниці вимірювання, редагувати або видаляти існуючі, що забезпечує точне та зрозуміле представлення інформації про товари.

Для додавання нової одиниці вимірювання, користувачам необхідно натиснути кнопку "Створити одиницю". Відкриється модальне вікно, де можна ввести назву нової одиниці. Після заповнення поля і натискання кнопки "Зберегти", одиниця додається до загального списку.

На сторінці редагування кожна одиниця вимірювання містить кнопки "Редагувати" та "Видалити". При виборі "Редагувати", користувачі можуть оновити назву одиниці у відкрившомуся модальному вікні. Це дозволяє легко модифікувати вже існуючі одиниці без необхідності їх повторного створення.

Кнопка "Видалити" використовується для видалення вибраних одиниць вимірювання. Відкриється модальне вікно для підтвердження видалення, щоб уникнути помилок та непотрібного видалення важливої інформації.

Список одиниць вимірювання представлений у формі таблиці, що полегшує навігацію та управління даними. Це дозволяє користувачам швидко знайти та виправити будь-яку інформацію.

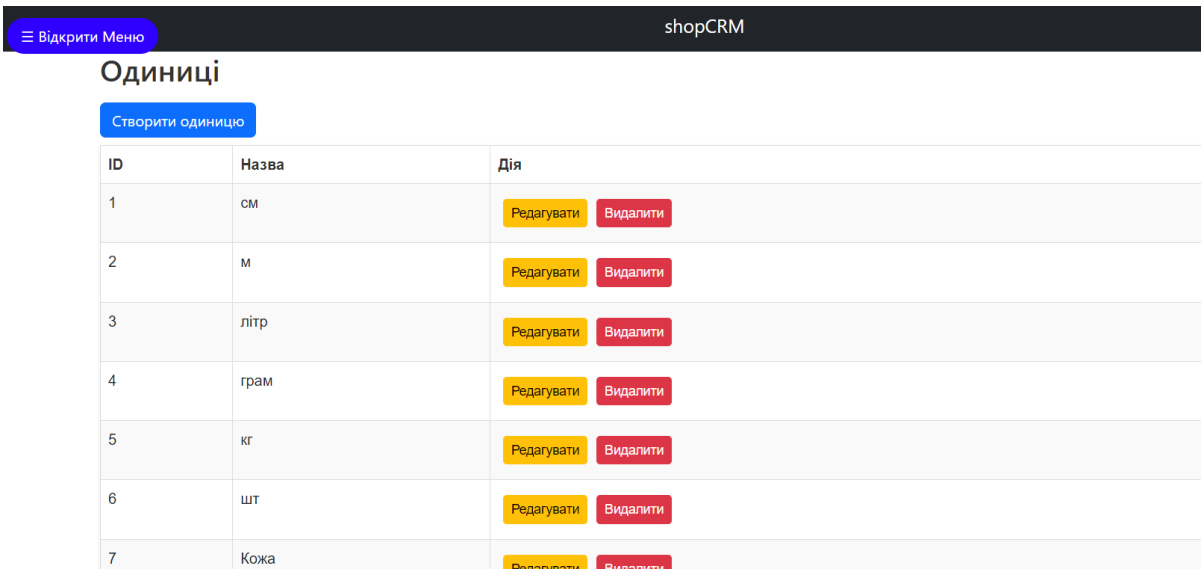

Рис. 3.9. Сторінка редагування одиниць вимірювання

Страница "Редагування одиниць вимірювання" є невід'ємною частиною системи, дозволяючи користувачам ефективно керувати деталями товарів. Завдяки інтуїтивно зрозумілому інтерфейсу та легкості використання, вона стає незамінною у процесі управління асортиментом.

Сторінка "Список виробників" надає користувачам можливість управління інформацією про виробників, що включає додавання, редагування, перегляд деталей та видалення записів.

Кнопка "Додати виробника" дозволяє користувачам створювати нові записи про виробників. Після натискання цієї кнопки відкривається форма для введення інформації, включаючи назву та опис виробника.

Кожен запис у списку виробників містить опції для перегляду детальної інформації ("Деталі") та редагування існуючої інформації ("Редагувати"). Це забезпечує швидкий доступ до повної інформації про виробника та можливість її оновлення.

Видалення запису про виробника можливе за допомогою кнопки "Видалити". Перед видаленням відбувається підтвердження дії, щоб уникнути помилкового видалення важливих даних.

Інформація про виробників представлена у вигляді таблиці для зручності перегляду та навігації. Кожен стовпець в таблиці надає ключову інформацію, включаючи ID, назву та опис виробника.

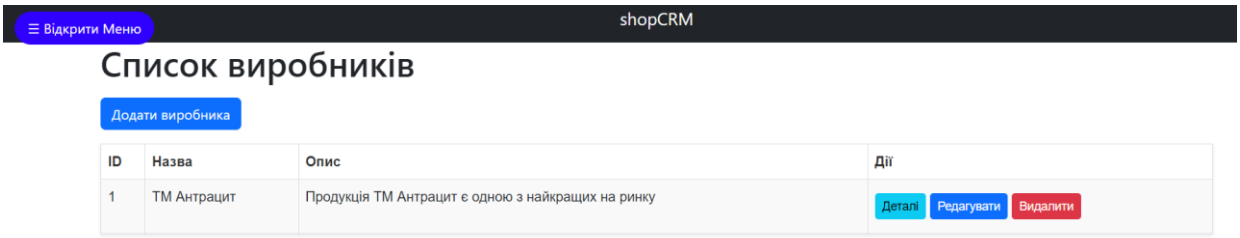

Рис. 3.10. Сторінка управління виробниками

Сторінка "Список виробників" є центральною для ефективного управління інформацією про виробників у системі. Її інтуїтивний інтерфейс та зрозуміла структура дозволяють користувачам легко вносити зміни та відслідковувати важливу інформацію.

Сторінка "Додати виробника" дозволяє користувачам вносити інформацію про нових виробників до системи. Цей процес включає заповнення деталей про виробника та, за необхідності, завантаження зображення.

Введення назви виробника: поле "Назва" призначене для введення назви нового виробника. Це обов'язкове поле і повинно бути заповнене.

Опис Виробника: У полі "Опис" можна ввести детальну інформацію про виробника. Це може включати будь-яку корисну інформацію, яка допоможе користувачам системи краще зрозуміти виробника.

Завантаження зображення: у разі наявності, можна завантажити зображення виробника. Це поле не є обов'язковим, але може бути корисним для візуальної ідентифікації виробника.

Відправлення даних: після заповнення всіх необхідних полів, натискання кнопки "Додати виробника" відправить інформацію до системи.

У разі завантаження зображення, файл автоматично розміщується у папці з назвою виробника на сервері. Це забезпечує організоване зберігання та легкий доступ до зображень.

Після успішного додавання виробника, користувач автоматично переадресовується на сторінку зі списком усіх виробників, де відображається новий запис.

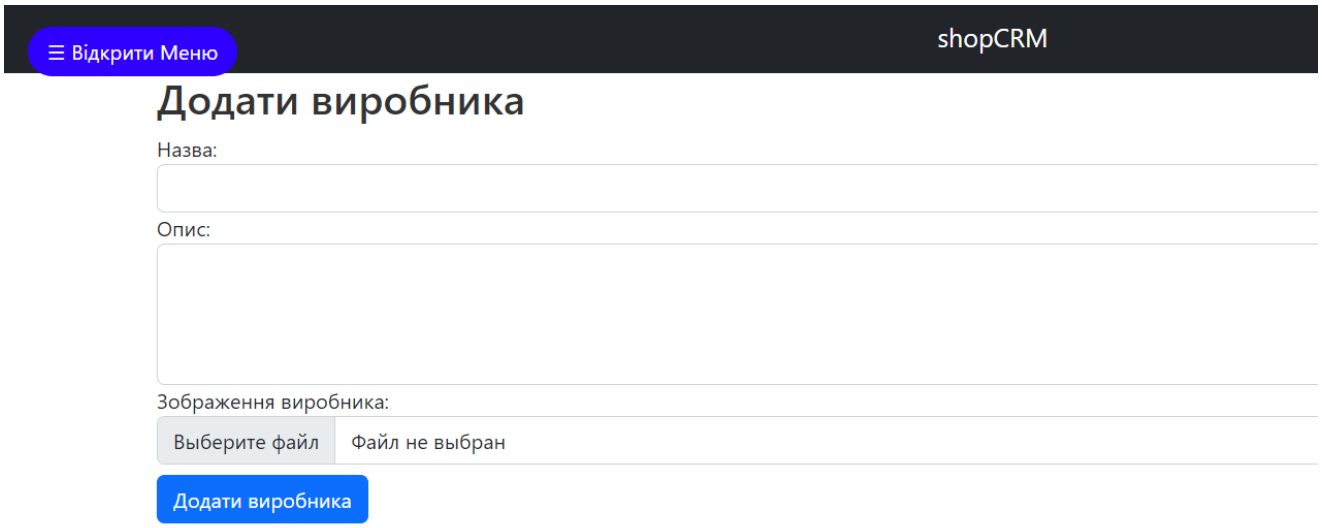

Рис. 3.11. Форма додавання виробника

Сторінка "Додати виробника" є простим та інтуїтивно зрозумілим інструментом для розширення бази даних виробників у веб-додатку. Її лаконічний інтерфейс забезпечує зручність введення даних, а також можливість візуального представлення виробників через завантаження зображень.

На сторінці "Список клієнтів" у веб-додатку користувачі мають можливість здійснювати глибокий аналіз та управління інформацією про клієнтів. З цієї сторінки, користувачі можуть переглядати усю існуючу клієнтську базу, яка відображається у вигляді таблиці з основними деталями, такими як ідентифікатор клієнта, ім'я, контактний телефон та дата додавання до системи. Ключовою особливістю цієї сторінки є її інтерактивність, оскільки вона дозволяє не тільки переглядати існуючу інформацію, але й виконувати різноманітні дії для кожного клієнта.

Для більш детального ознайомлення з інформацією про клієнта, можна скористатися кнопкою "Деталі", розташованою поруч з кожним записом. Це перенаправляє на сторінку з більш детальною інформацією, де можна переглянути повну історію взаємодій з клієнтом, включно з покупками та іншими важливими даними.

Кнопка "Редагувати" дає можливість оновити інформацію про клієнта. Після натискання на цю кнопку, користувачів перенаправляє на сторінку редагування, де можна внести зміни до основних даних клієнта, таких як ім'я, контактний номер телефону, чи іншу важливу інформацію.

У випадках, коли існує необхідність видалити клієнта з системи, наприклад, у разі помилково доданого або більше неактуального клієнта, можна скористатися кнопкою "Видалити". Ця дія зажадає підтвердження, щоб уникнути випадкового видалення важливої інформації.

Додавання нового клієнта відбувається через кнопку "Додати клієнта", розташовану у верхній частині сторінки. Натискання на цю кнопку перенаправить на форму додавання, де можна ввести усі необхідні дані для створення нового запису про клієнта.

Ця сторінка є важливою частиною веб-додатку, оскільки ефективне управління клієнтською базою є ключовим аспектом успішного ведення бізнесу. Вона дозволяє не тільки зберігати важливу інформацію, але й швидко реагувати на зміни, пов'язані з клієнтами.

На сторінці "Замовлення" веб-додатку користувачі мають доступ до комплексного переліку всіх замовлень. Структура сторінки спроектована таким чином, щоб забезпечити ефективне керування замовленнями, від їх створення до моніторингу та виконання.

Вгорі сторінки розташовані дві ключові кнопки: "Додати замовлення" та "Швидке замовлення". Перша кнопка веде до форми для створення нового замовлення, де користувач може ввести всі необхідні дані, включаючи вибір клієнта та продуктів. Опція "Швидке замовлення" призначена для створення замовлень у

скороченому режимі, що дозволяє швидко обробляти замовлення без необхідності введення детальної інформації.

Далі розташовується таблиця замовлень, яка надає користувачам загальний огляд всіх замовлень у системі. Кожен рядок таблиці містить інформацію про ID замовлення, ім'я клієнта, суму замовлення та набір дій, які можна виконати для кожного замовлення. Кнопка "Деталі" дозволяє користувачам переглянути повну інформацію про замовлення, включаючи список товарів, ціни та інші деталі. "Редагувати" дає можливість модифікувати існуючі замовлення, тоді як "Видалити" використовується для видалення замовлення з системи, з відповідним підтвердженням для запобігання випадковому видаленню.

Ця сторінка є критично важливою для оперативного управління бізнес-процесами, забезпечуючи користувачам повний контроль та гнучкість у роботі з замовленнями.

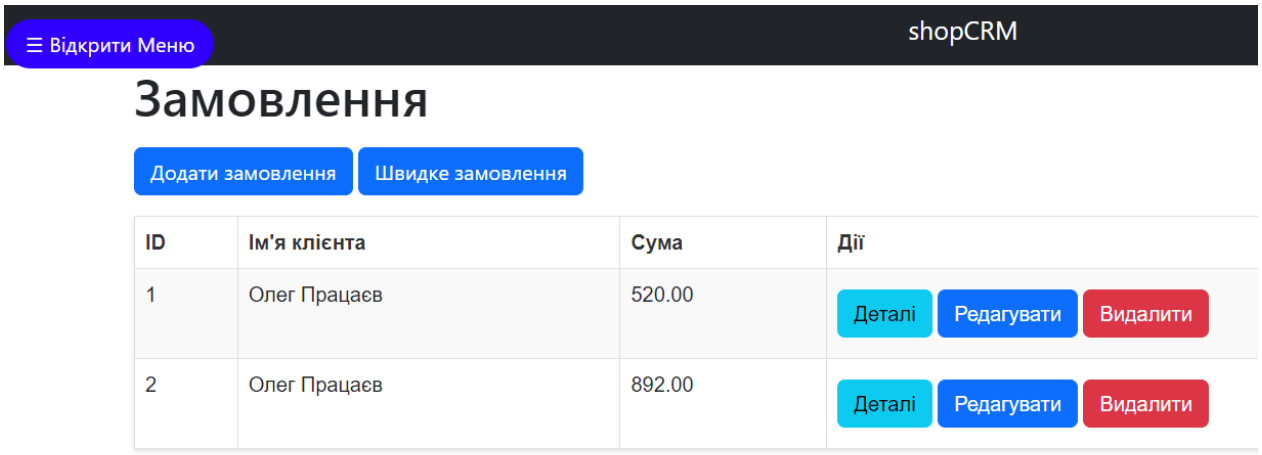

Рис. 3.12. Інтерфейс сторінки «Замовлення»

На сторінці "Створити нове замовлення" веб-додатку, користувачі здійснюють процес створення замовлень, що є ключовим елементом для управління продажами. Структура сторінки розроблена для забезпечення інтуїтивно зрозумілого та ефективного досвіду створення замовлень.

На початку форми представлений випадаючий список для вибору клієнта. Це дозволяє швидко визначити, для кого призначене замовлення, зі списку наявних

клієнтів. Після вибору клієнта користувачі можуть додавати продукти до замовлення за допомогою кнопки "Додати продукт". Кожне додавання продукту створює новий рядок у таблиці, де можна вказати обрані продукти та їх кількість.

Для кожного продукту в таблиці, користувачі можуть вказати потрібну кількість, а також мають можливість видалити будь-який рядок за допомогою кнопки "Видалити". Це забезпечує гнучкість під час створення замовлень, дозволяючи легко модифікувати список продуктів перед фінальним оформленням.

Додаткове поле для коментарів надає можливість додати будь-які спеціальні інструкції або інформацію, пов'язану з замовленням. Автоматично обчислювана загальна сума замовлення, що відображається у відповідному полі, дає користувачам змогу відстежувати вартість замовлення в реальному часі, враховуючи обрані продукти та їх кількість.

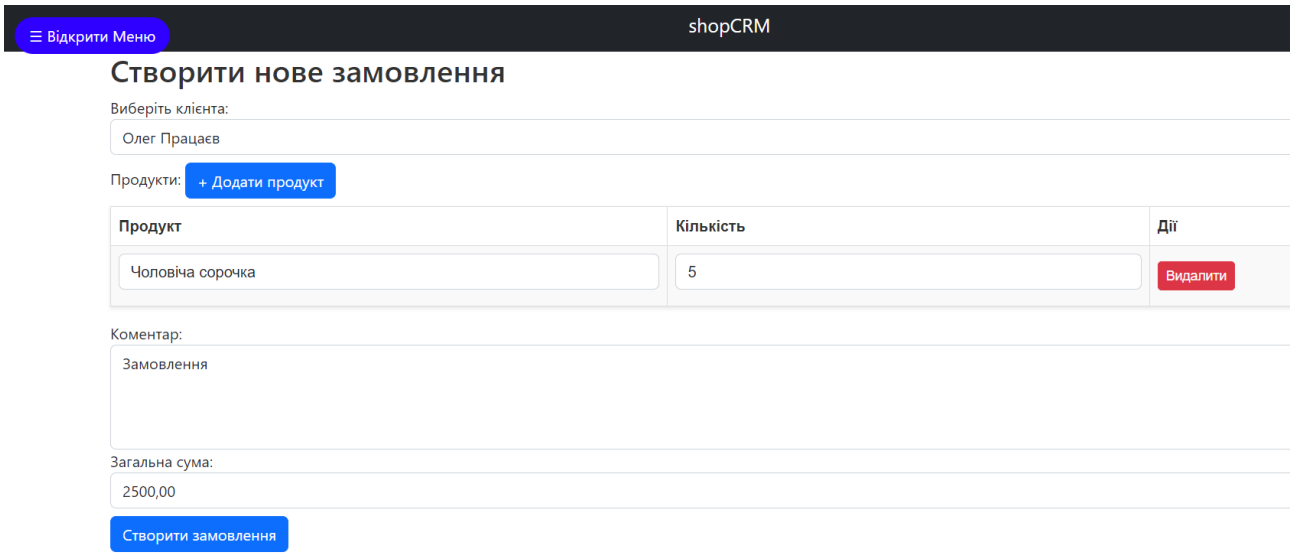

Рис. 3.13. Сторінка «Створити нове замовлення»

Після заповнення всіх необхідних даних та перевірки загальної суми замовлення, користувачі можуть натиснути кнопку "Створити замовлення" для завершення процесу. Ця кнопка відправляє дані на сервер, де вони обробляються, і замовлення реєструється в системі.

3.3 Переваги розробленого веб-додатку у порівнянні з існуючими рішеннями

Розроблений веб-додаток для автоматизації роботи магазину має ряд суттєвих переваг у порівнянні з існуючими на ринку рішеннями.

Порівнюючи функціональність та можливості створеного веб-додатку з існуючими на ринку аналогами, можна виділити низку суттєвих переваг:

1. Гнучкість та адаптованість під конкретні вимоги бізнесу. На відміну від універсальних платформ, даний додаток проектувався з нуля під особливості саме цього магазину та його бізнес-процеси. Це дозволяє точно задовольнити усі функціональні потреби без створення зайвої складності.

2. Оптимізація інтерфейсу та робочих процесів під звичний для персоналу спосіб роботи. Користувацькі сценарії, організація меню та елементів управління були розроблені на основі усталених бізнес-процесів магазину. Це робить використання системи максимально інтуїтивним та ефективним для працівників.

3. Легкість подальшої модернізації та розширення функціоналу завдяки гнучкій архітектурі та можливостям інтеграції. На відміну від громіздких універсальних систем, які важко модифікувати, даний додаток можна легко доповнювати новими модулями, сервісами та інтеграціями.

4. Масштабованість, надійність та економічна ефективність завдяки використанню хмарних технологій для розгортання системи. Це забезпечує гнучкі можливості розширення ресурсів при зростанні навантаження і мінімізує витрати на інфраструктуру.

5. Підвищення безпеки завдяки реалізації сучасних методів захисту інформації та ретельному тестуванню на проникнення. На відміну від типових конструкторів сайтів, де безпека часто недостатня.

Завдяки кастомному характеру розробки, веб-додаток максимально точно відповідає вимогам та особливостям роботи конкретного магазину. Це суттєво підвищує ефективність автоматизації та зручність використання у порівнянні з уніфікованими платформами.

Зробимо детальне порівняння розробленого веб-додатку з популярною платформою електронної комерції Magento:

1) Функціональні можливості:

- Magento має значно ширший спектр функцій, орієнтований на великі проекти електронної комерції;
- Розроблений додаток містить оптимізований набір функцій саме під потреби даного магазину.

2) Адміністративний інтерфейс:

- Інтерфейс Magento досить складний, перевантажений налаштуваннями;
- Інтерфейс нашого додатку простіший та інтуїтивніший за рахунок адаптації під конкретні задачі.

3) Гнучкість та можливості доробки:

- Magento відрізняється високою складністю архітектури та коду, що ускладнює модифікації;
- Архітектура розробленого веб-додатку набагато простіша, що спрощує доробки та додавання нового функціоналу.

Вартість впровадження:

- Вартість ліцензії Magento значно вища, також вимагає залучення дорогих фахівців;
- Впровадження розробленого рішення обійдеться дешевше через простоту та відсутність ліцензування.

4) Швидкодія:

- Продуктивність Magento значно залежить від конфігурації, часто вимагає оптимізації.
- Розроблений додаток оптимізований «під ключ» саме для цільового середовища магазину.

5) Масштабованість:

- Magento потребує складнішої інфраструктури для масштабування.
- Архітектура нашого рішення легше масштабується за рахунок використання хмарних сервісів.

6) Безпека:

- Magento часто містить вразливості, які потрібно відстежувати та виправляти;
- Безпека розробленого додатку перевірена тестуванням та використанням сучасних практик.

Отже, з точки зору індивідуальних вимог та можливостей даного бізнесу, розроблене кастомне рішення має переваги перед універсальною платформою Magento за багатьма ключовими параметрами.

На відміну від популярного сервісу для створення інтернет-магазинів Shopify, розроблений веб-додаток має низку суттєвих переваг. Зокрема, на противагу спрощеному підходу Shopify, даний додаток реалізує повноцінну систему управління товарним обліком, взаєминами з клієнтами та замовленнями, спеціально адаптовану під особливості роботи конкретного магазину. Це досягається завдяки гнучкій архітектурі та можливостям розширення функціоналу.

На відміну від обмежених аналітичних інструментів Shopify, даний додаток надає необхідну для ефективного управління статистику та звітність за продажами, запасами, попитом. Крім того, оптимізований користувацький інтерфейс значно підвищує зручність роботи персоналу магазину у порівнянні із стандартними шаблонами Shopify.

Ще однією важливою відмінністю є більша безпека завдяки використанню передових web-практик та ретельному тестуванню. У Shopify питанню інформаційної безпеки приділяється менше уваги.

Хоча Shopify позиціонується як простий сервіс для створення інтернет-магазинів, його можливості та гнучкість обмежені у порівнянні з кастомною розробкою. Враховуючи перспективу розвитку проекту, спеціалізований веб-додаток є більш оптимальним та масштабованим рішенням.

Порівнюючи розроблений веб-додаток з популярним рішенням для електронної комерції WooCommerce на базі WordPress, можна виділити наступні ключові відмінності:

1. На відміну від універсального WooCommerce, даний додаток реалізує саме необхідний набір функцій під конкретні потреби магазину. Це виключає зайві модулі, спрощує роботу персоналу та зменшує навантаження на систему.

2. Інтерфейс користувача розробленого рішення значно оптимізований та адаптований під звичні бізнес-процеси та особливості роботи працівників магазину. WooCommerce надає переважно типові універсальні шаблони.

3. Архітектура WooCommerce тісно інтегрована з WordPress, що накладає певні обмеження та технологічний борг. Натомість розроблений додаток побудований на сучасній мікросервісній архітектурі, яка набагато гнучкіша для модифікацій.

4. З точки зору продуктивності, WooCommerce схильний до зниження швидкодії під високим навантаження, на відміну від оптимізованого під цільове середовище кастомного додатку.

5. Розроблений веб-додаток пройшов комплексне навантажувальне та безпекове тестування перед впровадженням. WooCommerce як типове готове рішення приділяє цим аспектам менше уваги, що створює потенційні ризики.

Враховуючи особливості та вимоги конкретного бізнесу, спеціалізований веб-додаток має суттєві переваги перед уніфікованим WooCommerce за ключовими параметрами: функціоналом, інтерфейсом, гнучкістю, продуктивністю та безпекою. Це робить його більш ефективним та оптимальним рішенням для автоматизації роботи магазину.

Порівняльна таблиця наочно продемонструє суттєві переваги розробленого веб-додатку для автоматизації магазину у порівнянні з існуючими типовими рішеннями.

По-перше, на відміну від універсальних платформ, які частково відповідають вимогам проекту, даний додаток реалізує саме необхідний набір функцій, оптимізованих під конкретні задачі цього бізнесу.

По-друге, значна увага приділялася максимальній адаптації інтерфейсу та бізнес-логіки під звичний спосіб роботи співробітників магазину. Це суттєво підвищує ефективність та зручність у порівнянні зі складними чи типовими інтерфейсами готових рішень.

Гнучка мікросервісна архітектура забезпечує набагато вищу масштабованість та можливості модифікації функціоналу в майбутньому. На відміну від монолітних чи жорстко інтегрованих аналогів.

Таблиця 3.1

Порівняння розробленого веб-додатку та існуючих аналогічних рішень

Параметр	Наш додаток	Magento	Shopify	WooCommerce	Bitrix24
Відповідність вимогам проекту	Повна	Часткова	Часткова	Часткова	Часткова
Інтерфейс	Адаптований	Складний	Типовий	Типовий	Складний
Масштабованість	Висока	Середня	Низька	Низька	Середня
Можливості доробки	Високі	Низькі	Середні	Середні	Низькі
Вартість впровадження	Низька	Висока	Середня	Низька	Висока
Тестування безпеки та навантаження	Проведено	Частково	Ні	Ні	Частково

Комплексне тестування веб-додатку гарантує його високу надійність та захищеність, що не завжди властиве типовим «коробковим» рішенням.

Отже, з огляду на вимоги та потреби конкретного проекту, розроблений веб-додаток є оптимальним та ефективним рішенням в порівнянні з універсальними аналогами на ринку.

Висновок до розділу 3

У третьому розділі розглянуто практичні аспекти впровадження та використання розробленого веб-додатку для автоматизації роботи магазину.

У підрозділі 3.1 надано рекомендації щодо поетапного впровадження системи, підготовки персоналу, інтеграції з іншими інформаційними системами та забезпечення технічної підтримки користувачів.

Підрозділ 3.2 містить інструкцію для користувачів веб-додатку, яка полегшить його освоєння та ефективне використання.

У підрозділі 3.3 виконано порівняльний аналіз переваг розробленого додатку перед типовими ринковими рішеннями за ключовими параметрами.

Результати третього розділу становлять практичне підґрунтя для успішного впровадження та застосування веб-додатку, дозволяючи максимізувати ефект автоматизації для магазину.

Отже, поєднання теоретичної та практичної складових роботи формує міцний фундамент для створення і використання ефективного рішення з автоматизації конкретного бізнесу.

ВИСНОВКИ

У ході виконання дипломної роботи був розроблений веб-додаток для автоматизації роботи магазину, використовуючи сучасні технології та мікросервісну архітектуру. Цей процес включав декілька ключових етапів:

1. Огляд існуючих рішень для автоматизації торгівлі. У першому розділі дипломної роботи було проведено детальний аналіз існуючих рішень для автоматизації роботи магазинів. Були проаналізовані їхні переваги та недоліки, на основі чого було зроблено висновок про доцільність розробки специфікованого веб-додатку, який би відповідав конкретним потребам магазину.

2. Розробка веб-додатку. У другому розділі описано процес розробки веб-додатку, включаючи формулювання вимог до системи, проектування архітектури, бази даних та інтерфейсу, розробку основних функціональних модулів, а також тестування та верифікацію системи. Велика увага була приділена забезпеченню гнучкості та масштабованості системи.

3. Впровадження та порівняльний аналіз. У третьому розділі були наведені рекомендації щодо впровадження розробленого веб-додатку у практику роботи магазину. Також був проведений порівняльний аналіз переваг розробленого рішення у порівнянні з іншими універсальними платформами автоматизації.

Результатом дипломної роботи стало створення комплексного, спеціалізованого веб-додатку, який дозволяє автоматизувати основні бізнес-процеси магазину, підвищити його ефективність та конкурентоспроможність. Цей додаток був розроблений із зосередженням на специфічних потребах конкретного магазину, що робить його більш відповідним для цільового бізнесу, ніж універсальні рішення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Азарова А., Мороз О., Сторожа А. Моделювання системи підтримки прийняття рішень щодо покращення інноваційної діяльності підприємства. Вісник Хмельницького національного університету. 2012. № 6. Т. 1. С. 112–115.

2. Азарова А., Роїк О., Лобанкіна І. Впровадження та використання автоматизованих систем підвищення продуктивності праці на підприємстві за умов кризи. Економічний простір. 2010. № 42. С.125–132.

3. Байкарова О. О. Інформаційні технології – засіб оптимізації діяльності підприємств / О. О. Байкарова, Л. М. Тарасюк // Комп'ютерно-інтегровані технології : освіта, наука, виробництво. – 2013. – №11. – С. 177 – 182.

4. Бутенко Н.В. Впровадження концепції CRM на промисловому ринку. Економіка та держава. 2011. № 3. С. 40–42.

5. Вовчак І.С. Інформаційні системи та комп'ютерні технології в менеджменті : навч. посіб. / І. С. Вовчак. – Тернопіль : Карт-бланш, 2001. – 354 с.

6. Глівенко С. В. та ін. Інформаційні системи в менеджменті : навч. посіб. / С. В. Глівенко, С. В. Лапін, О. О. Павленко та ін. – Суми : ВТД «Університетська книга», 2005. – 407 с.

7. Ганущак-Єфіменко Л.М. CRM-система як ефективний інструмент розвитку готельного бізнесу в Україні. Вісник Київського національного університету технологій та дизайну. Серія : Економічні науки. 2017. № 4. С. 51–56.

8. Гнатенко І. А. Логіка впровадження інноваційних заходів на регіональний ринок праці в умовах його циклічного розвитку / І. А. Гнатенко, В. О. Рубежанська // Бізнес Інформ. – 2017. – № 8. – С. 110–115.

9. Гужва В. М. Інформаційні системи і технології на підприємствах : навч. посіб. / В. М. Гужва. – К. : КНЕУ, 2001. – 400 с.

10. Знахур В. С. Інформаційний менеджмент та маркетинг : конспект лекцій / С.В. Знахур. – Харків : Вид. ХНЕУ. – 2009. – 131 с.

11. Жаворонкова Г. В. Інформаційне підприємництво: інновації, консалтинг, маркетинг. – К.: Національний авіаційний університет, 2003. – 366 с.

12. Жежнич П.І. Технології інформаційного менеджменту : навч. посіб. / П.І. Жежнич. — Львів : Львівська політехніка, 2010. — 260 с.

13. Куденко Н. В. Менеджмент – управління інформацією / Навч. посібн. – К: КДТЕУ, 1999. – 313 с.

14. Крупський А. Ю., Феоктистова Л. А. Інформаційний менеджмент : навч. посіб. /А. Ю. Крупський, Л. А. Феоктистова. – М. : Видавничоторгова корпорація «Дашков і К °», 2008. – 80 с.

15. Матвієнко О. В., Цивін М. Н. Інформаційний менеджмент: опорний конспект лекцій у схемах і таблицях. Київ, 2007. 200 с.

16. Матвієнко О. В. Основи інформаційного менеджменту: Навчальний посібник. Київ, 2004. 128 с.

17. Марусей Т. Впровадження CRM-систем у маркетингову діяльність підприємства / Т. Марусей // Економіка та держава. – 2016. – № 6. – С. 87–89.

18. Майбутнє за технологіями: електронні програми управління фермою. URL: https://kurkul.com/blog/74-maybutnye-za-tehnologiyami-elektronni-programi-upravlinnya-fermoyu

19. Magento. URL: https://uk.wikipedia.org/wiki/Magento.

20. Shopify. URL: https://www.shopify.com/.

21. WooCommerce. URL: https://uk.wordpress.org/plugins/woocommerce/.

22. Bitrix24. URL: https://www.bitrix24.com/.

23. Лук'яненко Д. І. Розвиток інформаційного менеджменту як наукової категорії / Д. І. Лук'яненко // Науковий вісник Полтавського університету економіки і торгівлі. Сер.: Економічні науки. – 2013. – № 1. – С. 183–187.

24. Лисак В.М. Теоретичні аспекти автоматизації процесів збирання економічної інформації для управління підприємством // Вісник Хмельницького національного університету, - Хмельницький, 2008, № 5, Т. 2(119).

25. Принципи вибору програмного забезпечення для вирішення різних завдань управління персоналом. URL: https://studfile.net/preview/2398774/page:65/.

26. Результати дослідження ринку CRM в Україні. URL: https://www.bitrix24.ua (дата звернення 10.01.2019).

27. CRM-системи стали найбільшим сегментом ринку в 2017 році. URL: https://news.finance.ua (дата звернення 09.01.2019).

28. Сьомкіна Т.В., Литвинова О.В., Лобань О.О. Особливості моделей функціонування ІТ-компаній в Україні. Науковий вісник Ужгородського національного університету. Серія : Міжнародні економічні відносини та світове господарство. 2018. Вип. 19(3). С. 84–87.

29. Ушакова І.О. Соціальні мережі, як засіб впливу на взаємовідносини з клієнтами. Системи обробки інформації. 2012. Вип. 8. С. 54–58.

30. Фадєєва І.Г. Розвиток концептуальних засад автоматизованого аналітичного управління бізнес-процесами // Вісник Хмельницького національного університету, - Хмельницький, 2008, № 5, Т. 2(119).

ДОДАТКИ

Додаток А

Код модуля clients.php

```
<?php
session_start();
require_once 'php/functions.php';
if (!isLoggedIn()) {
   header('Location: login.php');
   exit;}
$clients = getClients();
include 'templates/header.php';?>
<div class="container">
   <h1>Список клієнтів</h1>
   <a href="create_client.php" class="btn btn-primary mb-3">Додати клієнта</a>
   <table class="table table-bordered">
      <thead>
         <tr>
            <th>ID</th>
            <th>Ім'я</th>
            <th>Телефон</th>
            <th>Дата додавання</th>
            <th>Дія</th>
         </tr>
      </thead>
      <tbody>
         <?php foreach($clients as $client): ?>
         <tr>
            <td><?= $client['id'] ?></td>
            <td><?= $client['name'] ?></td>
            <td><?= $client['phone'] ?></td>
            <td><?= $client['date_added'] ?></td>
            <td>
               <a href="details_client.php?id=<?= $client['id'] ?>" class="btn btn-info btn-sm">Деталі</a>
               <a href="edit_client.php?id=<?= $client['id'] ?>" class="btn btn-primary btn-sm">Редагувати</a>
               <a href="delete_client.php?id=<?= $client['id'] ?>" class="btn btn-danger btn-sm" onclick="return confirm('Ви впевнені, що хочете видалити цього клієнта?')">Видалити</a>
            </td>
         </tr>
         <?php endforeach; ?>
      </tbody>
   </table>
</div>
<?php include 'templates/footer.php'; ?>
```

Код модуля create_client.php

```
<?php
session_start();
require_once 'php/functions.php';
if (!isLoggedIn()) {
   header('Location: login.php');
   exit;}
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   $clientData = [
      'name' => $_POST['clientName'],
      'phone' => $_POST['clientPhone'],
      'comment' => $_POST['clientComment'] ];
   addClient($clientData);
   header('Location: clients.php');
   exit;}
include 'templates/header.php';?>
<div class="container">
   <h2>Додати нового клієнта</h2>
   <form action="create_client.php" method="post">
         <!-- Ім'я клієнта -->
      <div class="form-group">
         <label for="clientName">Ім'я:</label>
         <input type="text" class="form-control" id="clientName" name="clientName" required>
      </div>
      <!-- Телефон клієнта -->
      <div class="form-group">
         <label for="clientPhone">Телефон:</label>
         <input type="tel" class="form-control" id="clientPhone" name="clientPhone" required>
      </div>
      <!-- Коментар про клієнта -->
      <div class="form-group">
         <label for="clientComment">Коментар:</label>
         <textarea class="form-control" id="clientComment" name="clientComment" rows="3"></textarea>
      </div>
      <button type="submit" class="btn btn-primary">Додати клієнта</button>
   </form>
</div>
<?php include 'templates/footer.php'; ?>
```

Код модуля manufacturers.php

```
<?php
session_start();
require_once 'php/functions.php';
if (!isLoggedIn()) {
   header('Location: login.php');
   exit;}
$manufacturers = getManufacturers();
include 'templates/header.php';?>
```

```
<?php if (isset($_SESSION['error_message'])): ?>
    <div class="alert alert-danger">
        <?= $_SESSION['error_message']; ?>
    </div>
    <?php unset($_SESSION['error_message']); ?>
<?php endif; ?>
<?php if (isset($_SESSION['success_message'])): ?>
    <div class="alert alert-success">
        <?= $_SESSION['success_message']; ?>
    </div>
    <?php unset($_SESSION['success_message']); ?>
<?php endif; ?>
<div class="container">
    <h1>Список виробників</h1>
    <a href="create_manufacturer.php" class="btn btn-primary mb-3">Додати виробника</a>
    <table class="table table-bordered">
        <thead>
            <tr>
                <th>ID</th>
                <th>Назва</th>
                <th>Опис</th>
                <th>Дії</th>
            </tr>
        </thead>
        <tbody>
            <?php foreach($manufacturers as $manufacturer): ?>
            <tr>
                <td><?= $manufacturer['ID'] ?></td>
                <td><?= $manufacturer['Name'] ?></td>
                <td><?= $manufacturer['Description'] ?></td>
                <td>
                    <a href="details_manufacturer.php?id=<?= $manufacturer['ID'] ?>" class="btn btn-info btn-sm">Деталі</a>
                    <a href="edit_manufacturer.php?id=<?= $manufacturer['ID'] ?>" class="btn btn-primary btn-sm">Редагувати</a>
                    <a href="delete_manufacturer.php?id=<?= $manufacturer['ID'] ?>" class="btn btn-danger btn-sm" onclick="return confirm('Ви впевнені, що хочете видалити цього виробника?')">Видалити</a>
                </td>
            </tr>
            <?php endforeach; ?>
        </tbody>
    </table>
</div>
<?php include 'templates/footer.php'; ?>
```

Код модуля create_manufacturer.php

```
<?php
session_start();
require_once 'php/functions.php';
```

```
if (!isLoggedIn()) {
    header('Location: login.php');
    exit;}
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $manufacturerData = [
        'Name' => $_POST['manufacturerName'],
        'Description' => $_POST['manufacturerDescription'],
        'Image' => null ];
    // Check if an image is uploaded
    if (isset($_FILES['manufacturerImage']) && $_FILES['manufacturerImage']['error'] == 0) {
        $targetDir = "uploads/manufacturer/{$manufacturerData['Name']}/";
        if (!file_exists($targetDir)) {
            mkdir($targetDir, 0777, true);}
        $manufacturerData['Image'] = $targetDir . basename($_FILES["manufacturerImage"]["name"]);
        move_uploaded_file($_FILES["manufacturerImage"]["tmp_name"], $manufacturerData['Image']);}
    createManufacturer($manufacturerData);
    header('Location: manufacturers.php');
    exit;}
include 'templates/header.php';?>
<div class="container">
    <h2>Додати виробника</h2>
    <form action="create_manufacturer.php" method="post" enctype="multipart/form-data">
        <!-- Назва виробника -->
        <div class="form-group">
            <label for="manufacturerName">Назва:</label>
            <input type="text" class="form-control" id="manufacturerName" name="manufacturerName" required>
        </div>
        <!-- Опис виробника -->
        <div class="form-group">
            <label for="manufacturerDescription">Опис:</label>
            <textarea class="form-control" id="manufacturerDescription" name="manufacturerDescription" rows="4"></textarea>
        </div>
        <!-- Зображення виробника -->
        <div class="form-group">
            <label for="manufacturerImage">Зображення виробника:</label>
            <input type="file" class="form-control" id="manufacturerImage" name="manufacturerImage" accept="image/*">
        </div>
        <button type="submit" class="btn btn-primary">Додати виробника</button>
    </form>
</div>
<?php include 'templates/footer.php'; ?>
```

Функції з файлу functions.php

```
<?php require_once 'database.php';
if (session_status() == PHP_SESSION_NONE) {
    session_start();}
function isLoggedIn() {
    return isset($_SESSION['user_id']);}
```

```
function authenticateUser($username, $password) {
    global $db;
    $query = "SELECT * FROM users WHERE username = :username";
    $stmt = $db->prepare($query);
    $stmt->execute(['username' => $username]);
    $user = $stmt->fetch(PDO::FETCH_ASSOC);
    if ($user && password_verify($password, $user['password'])) {
        $_SESSION['user_id'] = $user['id'];
        $_SESSION['username'] = $user['username'];
        $_SESSION['user'] = $user;
        return true;}
    return false;}
function userExists($username) {
    global $db;
    $query = "SELECT * FROM users WHERE username = :username";
    $stmt = $db->prepare($query);
    $stmt->execute(['username' => $username]);
    $user = $stmt->fetch(PDO::FETCH_ASSOC);
    return $user ? true : false;}
function registerUser($username, $password) {
    global $db;
    $query = "INSERT INTO users (username, password) VALUES (:username, :password)";
    $stmt = $db->prepare($query);
    $result = $stmt->execute([
        'username' => $username,
        'password' => password_hash($password, PASSWORD_DEFAULT), ]);
   return $result;}
function logoutUser() {
    unset($_SESSION['user_id']);
     unset($_SESSION['username']);
     unset($_SESSION['user']);
     unset($_SESSION['author_id']);
     unset($_SESSION['author_name']);
     unset($_SESSION['author']);}
 function getUsers(){
    global $db;
    $query = "SELECT * FROM users";
    $stmt = $db->prepare($query);
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);}
function addUser($username, $password, $hourly_rate){
    global $db;
    $hashed_password = password_hash($password, PASSWORD_DEFAULT);
    $query = "INSERT INTO users (username, password, hourly_rate) VALUES (:username, :password, :hourly_rate)";
    $stmt = $db->prepare($query);
    $stmt->execute([':username' => $username, ':password' => $hashed_password, ':hourly_rate' => $hourly_rate]);}
function getUserById($id){
```

```
    global $db;
    $query = "SELECT * FROM users WHERE id = :id";
    $stmt = $db->prepare($query);
    $stmt->execute([':id' => $id]);
    return $stmt->fetch(PDO::FETCH_ASSOC);}
function updateUser($id, $new_id, $username, $password, $hourly_rate){
    global $db;
    $hashed_password = password_hash($password, PASSWORD_DEFAULT);
    $query = "UPDATE users SET id = :new_id, username = :username, password = :password, hourly_rate = :hourly_rate WHERE id = :id";
    $stmt = $db->prepare($query);
    $stmt->execute([':new_id' => $new_id, ':username' => $username, ':password' => $hashed_password, ':id' => $id, ':hourly_rate' =>
$hourly_rate]);}
function deleteUser($id){
    global $db;
    $query = "DELETE FROM users WHERE id = :id";
    $stmt = $db->prepare($query);
    $stmt->execute([':id' => $id]);}
/* Chat*/
function getChatMessages($userId = null) {
    global $db;
    $query = "SELECT chat_messages.*, users.username FROM chat_messages JOIN users ON chat_messages.user_id = users.id ORDER BY
chat_messages.timestamp";
    $stmt = $db->prepare($query);
    $stmt->execute();
    $messages = $stmt->fetchAll(PDO::FETCH_ASSOC);
    // присваиваем каждому сообщению имя пользователя, если оно существует
    foreach ($messages as &$msg) {
        if (!empty($msg['username'])) {
            $msg['user'] = [
                'id' => $msg['user_id'],
                'username' => $msg['username']];}
        unset($msg['username']);}
    if ($userId) { $messages = array_filter($messages, function ($msg) use ($userId) {
            return $msg['user']['id'] === $userId;});}
    return $messages;}
function addChatMessage($user_id, $message, $task_time) {
    global $db;
    $query = "INSERT INTO chat_messages (user_id, message, task_time) VALUES (:user_id, :message, :task_time)";
    $stmt = $db->prepare($query);
    $result = $stmt->execute([
        'user_id' => $user_id,
        'message' => $message,
        'task_time' => $task_time,]);
    return $result;}
function deleteChatMessage($id) {
    global $db;
    $query = "DELETE FROM chat_messages WHERE id = :id";
```

```
    $stmt = $db->prepare($query);
    $result = $stmt->execute(['id' => $id]);
    return $result;}
function updateChatMessage($id, $message, $task_time) {
    global $db;
    $query = "UPDATE chat_messages SET message = :message, task_time = :task_time WHERE id = :id";
    $stmt = $db->prepare($query);
    $result = $stmt->execute([
        'id' => $id,
        'message' => $message,
        'task_time' => $task_time, ]);
    return $result;}
//функції продуктів
function getProducts() {
    global $db;
    $query = "
        SELECT
            p.ID,
            p.Name,
            c.Name as CategoryName,
            m.Name as ManufacturerName,
            p.Price,
            p.StockQuantity
        FROM products p
        LEFT JOIN categories c ON p.CategoryID = c.ID
        LEFT JOIN manufacturers m ON p.ManufacturerID = m.ID ";
    $stmt = $db->prepare($query);
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);}
function getCategories() {
    global $db;
    $query = "SELECT ID, Name FROM categories";
    $stmt = $db->prepare($query);
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);}
function getCharacteristics() {
    global $db;
    $query = "SELECT ID, Name FROM characteristics";
    $stmt = $db->prepare($query);
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);}
function getMeasurementUnits() {
    global $db;
    $query = "SELECT ID, UnitName FROM measurement_units";
    $stmt = $db->prepare($query);
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);}
```

```
function createProductWithCharacteristics($productData) {
    global $db;
    try {
        // Розпочинаємо транзакцію
        $db->beginTransaction();
        // 1. Зберігання основної інформації про продукт
        $stmt = $db->prepare("INSERT INTO products (SKU, Name, CategoryID, Description, Image, Price, StockQuantity, ManufacturerID)
VALUES (?, ?, ?, ?, ?, ?, ?, ?)");
        $stmt->execute([
            $productData['SKU'],
            $productData['Name'],
            $productData['CategoryID'],
            $productData['Description'],
            $productData['Image'],
            $productData['Price'],
            $productData['StockQuantity'],
            $productData['ManufacturerID'] ]);
        $productID = $db->lastInsertId();
        // 2. Зберігання характеристик продукту
        $stmt = $db->prepare("INSERT INTO characteristic_associations (ProductID, CharacteristicID, MeasurementUnitID, Value) VALUES
(?, ?, ?, ?)");
        foreach ($productData['ProductCharacteristics'] as $char) {
            if (!empty($char['id']) && !empty($char['unit']) && !empty($char['value'])) {
                $stmt->execute([$productID, $char['id'], $char['unit'], $char['value']]);}}
        // Завершення транзакції
        $db->commit();
        return $productID;
    } catch (Exception $e) {
        // Відміна всіх операцій у разі помилки
        $db->rollBack();
        die("Failed to save product: " . $e->getMessage());}}
function getProductDetails($productID) {
    global $db;
    // Отримання основних даних про продукт
    $stmt = $db->prepare("
        SELECT products.*, categories.Name as CategoryName, manufacturers.Name as ManufacturerName
        FROM products
        LEFT JOIN categories ON products.CategoryID = categories.ID
        LEFT JOIN manufacturers ON products.ManufacturerID = manufacturers.ID
        WHERE products.ID = ?");
    $stmt->execute([$productID]);
    $product = $stmt->fetch(PDO::FETCH_ASSOC);
    // Отримання характеристик продукту
    $stmt = $db->prepare("
        SELECT characteristics.Name, characteristic_associations.Value, measurement_units.UnitName
        FROM characteristic_associations
        LEFT JOIN characteristics ON characteristic_associations.CharacteristicID = characteristics.ID
```

```
        LEFT JOIN measurement_units ON characteristic_associations.MeasurementUnitID = measurement_units.ID
        WHERE characteristic_associations.ProductID = ? ");
    $stmt->execute([$productID]);
    $characteristics = $stmt->fetchAll(PDO::FETCH_ASSOC);
    $product['Characteristics'] = $characteristics;
    return $product;}
function getProductDetailsEd($productID) {
    global $db;
    // Отримання основних даних про продукт
    $stmt = $db->prepare("
        SELECT products.*, categories.Name as CategoryName, manufacturers.Name as ManufacturerName
        FROM products
        LEFT JOIN categories ON products.CategoryID = categories.ID
        LEFT JOIN manufacturers ON products.ManufacturerID = manufacturers.ID
        WHERE products.ID = ? ");
    $stmt->execute([$productID]);
    $product = $stmt->fetch(PDO::FETCH_ASSOC);
    // Отримання характеристик продукту
    $stmt = $db->prepare("
        SELECT
            characteristics.ID as CharacteristicID,
            characteristics.Name,
            characteristic_associations.Value,
            measurement_units.ID as MeasurementUnitID,
            measurement_units.UnitName
        FROM characteristic_associations
        LEFT JOIN characteristics ON characteristic_associations.CharacteristicID = characteristics.ID
        LEFT JOIN measurement_units ON characteristic_associations.MeasurementUnitID = measurement_units.ID
        WHERE characteristic_associations.ProductID = ? ");
    $stmt->execute([$productID]);
    $characteristics = $stmt->fetchAll(PDO::FETCH_ASSOC);
    $product['characteristics'] = $characteristics;
    return $product;}
function getManufacturers() {
    global $db;
    $stmt = $db->prepare("SELECT * FROM manufacturers");
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);}
function updateProductWithCharacteristics($updatedData) {
    global $db;
    try {// Start a transaction
        $db->beginTransaction();
        // 1. Update the main product data
        $stmt = $db->prepare("UPDATE products SET SKU=?, Name=?, CategoryID=?, ManufacturerID=?, Description=?, Image=?, Price=?,
StockQuantity=? WHERE ID=?");
        $stmt->execute([
            $updatedData['SKU'],
```

```
            $updatedData['Name'],
            $updatedData['CategoryID'],
            $updatedData['ManufacturerID'], // Add ManufacturerID here
            $updatedData['Description'],
            $updatedData['Image'],
            $updatedData['Price'],
            $updatedData['StockQuantity'],
            $updatedData['ID'] ]);
        // 2. Update product characteristics (you can either update existing ones or delete all and insert the new ones, depending on the complexity and requirements)
        // Here I'll just delete and re-insert for simplicity
        $stmt = $db->prepare("DELETE FROM characteristic_associations WHERE ProductID = ?");
        $stmt->execute([$updatedData['ID']]);
        $stmt = $db->prepare("INSERT INTO characteristic_associations (ProductID, CharacteristicID, MeasurementUnitID, Value) VALUES (?, ?, ?, ?)");
        foreach ($updatedData['ProductCharacteristics'] as $char) {
            if (!empty($char['id']) && !empty($char['unit']) && !empty($char['value'])) {
                $stmt->execute([$updatedData['ID'], $char['id'], $char['unit'], $char['value']]);}}
        // Commit the transaction
        $db->commit();
    } catch (Exception $e) {
        // Rollback all operations in case of an error
        $db->rollBack();
        // Assuming you have defined a $logFile variable somewhere
        file_put_contents($logFile, date("Y-m-d H:i:s") . " - ERROR: " . $e->getMessage() . "\n", FILE_APPEND);
        die("Failed to update product: " . $e->getMessage());}}
function deleteProductAssociations($productID) {
    global $db;
    // Видаляємо характеристики асоційовані з продуктом
    $stmt = $db->prepare("DELETE FROM characteristic_associations WHERE ProductID = ?");
    return $stmt->execute([$productID]);}
function deleteProduct($productID) {
    global $db;
    // Видаляємо сам продукт
    $stmt = $db->prepare("DELETE FROM products WHERE ID = ?");
    return $stmt->execute([$productID]);}
//Робота з категоріями
function getMainCategories() {
    global $db;
    $stmt = $db->prepare("SELECT * FROM `categories` WHERE `ParentCategoryID` IS NULL");
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);}
function getSubCategories($parentCategoryID) {
    global $db;
    $stmt = $db->prepare("SELECT * FROM `categories` WHERE `ParentCategoryID` = ?");
    $stmt->execute([$parentCategoryID]);
    return $stmt->fetchAll(PDO::FETCH_ASSOC);}
```

```
function deleteCategory($categoryID) {
    global $db;
    $stmt = $db->prepare("DELETE FROM categories WHERE ID = ?");
    return $stmt->execute([$categoryID]);}
function addCategory($data) {
    global $db;
    $stmt = $db->prepare("INSERT INTO categories (Name, Image, Description, ParentCategoryID) VALUES (?, ?, ?, ?)");
    return $stmt->execute([$data['Name'], $data['Image'], $data['Description'], $data['ParentCategoryID']]);}
function updateCategory($data) {
    global $db;
    $stmt = $db->prepare("UPDATE categories SET Name = ?, Image = ?, Description = ?, ParentCategoryID = ? WHERE ID = ?");
    return $stmt->execute([$data['Name'], $data['Image'], $data['Description'], $data['ParentCategoryID'], $data['ID']]);}
//Функції виробників.
function createManufacturer($data) {
    global $db;
    $stmt = $db->prepare("INSERT INTO manufacturers (Name, Description, Image) VALUES (?, ?, ?)");
    $stmt->execute([$data['Name'], $data['Description'], $data['Image']]);
    return $db->lastInsertId();}
function getManufacturerDetails($manufacturerID) {
    global $db;
    $stmt = $db->prepare("SELECT * FROM manufacturers WHERE ID = ?");
    $stmt->execute([$manufacturerID]);
    return $stmt->fetch(PDO::FETCH_ASSOC);}
function updateManufacturer($updatedData) {
    global $db;
    $stmt = $db->prepare("UPDATE manufacturers SET Name=?, Description=?, Image=? WHERE ID=?");
    $stmt->execute([$updatedData['Name'], $updatedData['Description'], $updatedData['Image'], $updatedData['ID']]);}
function deleteManufacturer($manufacturerID) {
    global $db;
    // Перевірка, чи існують продукти, пов'язані з цим виробником
    $stmt = $db->prepare("SELECT COUNT(*) FROM products WHERE ManufacturerID = ?");
    $stmt->execute([$manufacturerID]);
    $count = $stmt->fetchColumn();
    if ($count > 0) {
        throw new Exception("Цей виробник використовується в {$count} продуктах. Видалення неможливе.");}
    $stmt = $db->prepare("DELETE FROM manufacturers WHERE ID = ?");
    $stmt->execute([$manufacturerID]);}
//Клієнти
function getClients() {
    global $db;
    $stmt = $db->prepare("SELECT * FROM clients");
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);}
function addClient($clientData) {
    global $db;
    $stmt = $db->prepare("INSERT INTO clients (name, phone, date_added, comment) VALUES (?, ?, NOW(), ?)");
    $stmt->execute([$clientData['name'], $clientData['phone'], $clientData['comment']]);}
```

```
function getClientDetails($clientID) {
    global $db;
    $stmt = $db->prepare("SELECT * FROM clients WHERE id = ?");
    $stmt->execute([$clientID]);
    return $stmt->fetch(PDO::FETCH_ASSOC);}
function updateClient($updatedData) {
    global $db;
    $stmt = $db->prepare("UPDATE clients SET name=?, phone=?, comment=? WHERE id=?");
    $stmt->execute([$updatedData['name'], $updatedData['phone'], $updatedData['comment'], $updatedData['id']]);}
function deleteClient($clientID) {
    global $db;
    try {
        $stmt = $db->prepare("DELETE FROM clients WHERE id = ?");
        $stmt->execute([$clientID]);
        return true;
    } catch (PDOException $e) {
        // In a real-world scenario, you might want to log this error and not display it directly.
        echo "Error: " . $e->getMessage();
        return false; }}
//Замовлення
function getOrders() {
    global $db;
    $stmt = $db->prepare("
        SELECT
            orders.id,
            clients.name as client_name,
            orders.total_amount
        FROM orders
        JOIN clients ON orders.client_id = clients.id ");
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);}
function getAllProducts() {
    global $db;
    $stmt = $db->query("SELECT * FROM products");
    return $stmt->fetchAll(PDO::FETCH_ASSOC);}
function createOrder($orderData) {
    global $db;
    try {$db->beginTransaction();
// 1. Зберігання основної інформації про замовлення
$stmt = $db->prepare("INSERT INTO orders (order_date, client_id, total_amount, comment) VALUES (CURDATE(), ?, 0, ?)");
$stmt->execute([$orderData['client_id'], $orderData['comment']]);
        $orderId = $db->lastInsertId();
        // 2. Зберігання продуктів замовлення
        $stmt = $db->prepare("INSERT INTO orders_products (order_id, product_id, quantity) VALUES (?, ?, ?)");
        $totalAmount = 0;
        foreach ($orderData['products'] as $product) {
            $stmt->execute([$orderId, $product['id'], $product['quantity']]);
```

```
            $productInfo = getProductById($product['id']);
            $totalAmount += $productInfo['Price'] * $product['quantity'];}
        // 3. Оновлення загальної суми замовлення
        $stmt = $db->prepare("UPDATE orders SET total_amount = ? WHERE id = ?");
        $stmt->execute([$totalAmount, $orderId]);
        $db->commit();
        return $orderId;
    } catch (Exception $e) {
        $db->rollBack();
        die("Failed to save order: " . $e->getMessage());}}
function getProductById($productId) {
    global $db;
    $stmt = $db->prepare("SELECT * FROM products WHERE ID = ?");
    $stmt->execute([$productId]);
    return $stmt->fetch(PDO::FETCH_ASSOC);}
function getOrderById($orderId) {
    global $db;
    $stmt = $db->prepare("SELECT * FROM orders WHERE id = ?");
    $stmt->execute([$orderId]);
    $order = $stmt->fetch(PDO::FETCH_ASSOC);
    if (!$order) {return null;}
    $stmt = $db->prepare("SELECT * FROM orders_products WHERE order_id = ?");
    $stmt->execute([$orderId]);
    $orderProducts = $stmt->fetchAll(PDO::FETCH_ASSOC);
    $order['products'] = $orderProducts;
    return $order;}
function updateOrder($orderId, $newOrderData) {
    global $db;
    try {$db->beginTransaction();
        // Оновити основну інформацію про замовлення
        $stmt = $db->prepare("UPDATE orders SET client_id = ?, comment = ? WHERE id = ?");
        $stmt->execute([$newOrderData['client_id'], $newOrderData['comment'], $orderId]);
        // Видалити старі продукти замовлення
        $stmt = $db->prepare("DELETE FROM orders_products WHERE order_id = ?");
        $stmt->execute([$orderId]);
        // Додати нові продукти замовлення
        $stmt = $db->prepare("INSERT INTO orders_products (order_id, product_id, quantity) VALUES (?, ?, ?)");
        foreach ($newOrderData['products'] as $product) {
            $stmt->execute([$orderId, $product['id'], $product['quantity']]);}
        $db->commit();
        return true;
    } catch (Exception $e) {
        $db->rollBack();
        die("Failed to update order: " . $e->getMessage());
        return false;}}
function deleteOrder($orderId) {
    global $db;
```

```
    try {$db->beginTransaction();
        // Видалити продукти замовлення
        $stmt = $db->prepare("DELETE FROM orders_products WHERE order_id = ?");
        $stmt->execute([$orderId]);
        // Видалити замовлення
        $stmt = $db->prepare("DELETE FROM orders WHERE id = ?");
        $stmt->execute([$orderId]);
        $db->commit();
        return true;
    } catch (Exception $e) {
        $db->rollBack();
        die("Failed to delete order: " . $e->getMessage());
        return false;}}
//Характеристики товарів
function getAllCharacteristics() {
    global $db;
    $stmt = $db->prepare("SELECT * FROM characteristics");
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);}
function addCharacteristic($name) {
    global $db;
    $stmt = $db->prepare("INSERT INTO characteristics (Name) VALUES (?)");
    $stmt->execute([$name]);
    return $db->lastInsertId();}
function updateCharacteristic($id, $name) {
    global $db;
    $stmt = $db->prepare("UPDATE characteristics SET Name = ? WHERE ID = ?");
    return $stmt->execute([$name, $id]);}
function deleteCharacteristic($id) {
    global $db;
    $stmt = $db->prepare("DELETE FROM characteristics WHERE ID = ?");
    return $stmt->execute([$id]);}
//Значення характеристик
function getAllUnits() {
    global $db;
    $stmt = $db->prepare("SELECT * FROM measurement_units");
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);}
function addUnit($name) {
    global $db;
    $stmt = $db->prepare("INSERT INTO measurement_units (UnitName) VALUES (?)");
    $stmt->execute([$name]);
    return $db->lastInsertId();}
function updateUnit($id, $name) {
    global $db;
    $stmt = $db->prepare("UPDATE measurement_units SET UnitName = ? WHERE ID = ?");
    return $stmt->execute([$name, $id]);}
```

```
function deleteUnit($id) {
    global $db;
    $stmt = $db->prepare("DELETE FROM measurement_units WHERE ID = ?");
    return $stmt->execute([$id]);}
//графік замовлень
function getMonthlyOrders() {
    global $db;
    $stmt = $db->prepare("SELECT DATE_FORMAT(order_date, '%Y-%m') as month, COUNT(*) as total_orders FROM orders GROUP BY
month ORDER BY month");
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);}
```