ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка програмного забезпечення для автоматизації замовлення металопластикових труб»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Станіслав КУРЯТНИК
(підпис)

Виконав: здобувач вищої освіти групи ПД-43

Станіслав КУРЯТНИК
Керівник: Віталій ЗАЛИВА
доктор філософії PhD
Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Курятнику Станіславу Анатолійовичу

1. Тема кваліфікаційної роботи: «Розробка програмного забезпечення для автоматизації замовлення металопластикових труб»

керівник кваліфікаційної роботи доктор філософії (PhD) Віталій ЗАЛИВА, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про методи розробки веб-застосунків, опис роботи веб-застосунку, документація фреймворків Symfony та мови програмування PHP.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд веб-застосунку для автоматизації замовлення металопластикових труб та аналіз існуючих аналогів, визначення вимог до програмного забезпечення.
2. Огляд та аналіз засобів реалізації застосунку для автоматизації замовлення металопластикових труб.
3. Проєктування архітектури застосунку для автоматизації замовлення

металопластикових труб.
4. Програмна реалізація та тестування веб-застосунку автоматизації замовлення металопластикових труб.

5. Перелік графічного матеріалу: *презентація*
   1. Аналіз аналогів.
   2. Вимоги до програмного забезпечення.
   3. Програмні засоби реалізації.
   4. Діаграма варіантів використання.
   5. Діаграма класів.
   6. Діаграма діяльності.
   8. Екранні форми.
   10. Апробація результатів дослідження.
6. Дата видачі завдання «28» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2025	
2	Аналіз та дослідження існуючих аналогів	07.03-12.03.2025	
3	Огляд та аналіз застосунку для автоматизації замовлення металопластикових труб	13.03-15.03.2025	
4	Огляд та аналіз засобів реалізації веб-застосунку для автоматизації замовлення металопластикових труб	16.03-21.03.2025	
5	Проєктування архітектури веб-застосунку для автоматизації замовлення металопластикових труб	22.03-03.04.2025	
6	Програмна реалізація та тестування застосунку для автоматизації замовлення металопластикових труб	04.04-28.04.2025	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2025	
8	Розробка демонстраційних матеріалів	06.05-12.05.2025	
9	Попередній захист роботи	13.05-31.05.2025	

Здобувач вищої освіти _____ Станіслав КУРЯТНИК
(підпис)

Керівник
кваліфікаційної роботи _____ Віталій ЗАЛИВА
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 73 стор., 1 табл., 23 рис., 23 джерел.

Мета роботи – автоматизування процесу замовлення металопластикових труб.

Об'єкт дослідження – процес управління замовленнями металопластикових труб у промислових та комерційних умовах.

Предмет дослідження – веб-додаток для автоматизації замовлення металопластикових труб.

Короткий зміст роботи: У роботі проаналізовано предметну галузь та методи автоматизації процесу замовлення металопластикових труб. Проведено огляд існуючих рішень для онлайн-розрахунку матеріалів та оформлення замовлень. Розроблено вебзастосунок, який забезпечує авторизацію користувачів, реалізує онлайн-калькулятор для розрахунку необхідної кількості труб за заданими параметрами (площа приміщення, кількість точок тощо), функціонал підбору рекомендованих товарів, формування замовлення та його підтвердження. Застосунок також включає систему управління товарами та кошиком. Проведено модульне тестування основних компонентів системи. Для розробки використано фреймворк Symfony (PHP) на стороні бекенду та систему керування базами даних PostgreSQL для зберігання структурованої інформації.

Сфера застосування розробленого програмного забезпечення охоплює підприємства, які займаються постачанням і монтажем металопластикових труб, з метою підвищення ефективності обслуговування клієнтів та автоматизації процесу підбору матеріалів і оформлення замовлень.

КЛЮЧОВІ СЛОВА: BACK-END, PHP, SYMFONY, POSTGRESQL, RESTful API, ВЕБ-ЗАСТОСУНОК.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 8
ВСТУП 9
1 АНАЛІЗ ЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ЗАМОВЛЕННЯ МЕТАЛОПЛАСТИКОВИХ ТРУБ 11
1.1. Веб-застосунок для автоматизації замовлення металопластикових труб .. 11
1.2 Специфіка замовлення металопластикових труб. 12
1.3 Цільова аудиторія 18
1.4 Дослідження існуючих аналогів 19
1.5 Визначення вимог до застосунку 22
2 ЗАСОБИ РЕАЛІЗАЦІЇ 31
2.1 Середовище розробки 31
2.2 Мови програмування 31
2.3 Система управління базою даних 32
2.4 Система контролю версій 33
3 ПРОЄКТУВАННЯ 34
3.1 Проєктування архітектури веб-застосунку 34
3.2 Архітектура клієнтської частини 35
3.3 Архітектура серверної частини 35
3.4 Архітектура бази даних 38
4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ 42
4.2 Реалізація клієнтської частини 42
4.3 Реалізація серверної частини 47
4.5 Тестування веб-застосунку 50
ВИСНОВКИ 52
ПЕРЕЛІК ПОСИЛАНЬ 54
ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ 56
ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ 66

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – Програмне забезпечення

IDE – Integrated Development Environment

API – Application Programming Interface

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

SQL – Structured Query Language

ORM – Object-Relational Mapping

DI – Dependency Injection

ВСТУП

Актуальність: У сучасних умовах будівництва та ремонту систем водопостачання та опалення попит на металопластикові труби постійно зростає. При цьому точність розрахунку матеріалів і зручність оформлення замовлень відіграють ключову роль для компаній і споживачів. Існуючі процеси часто залишаються ручними або недостатньо автоматизованими, що призводить до помилок, втрати часу та ресурсів. Тому розробка програмного забезпечення, яке автоматизує розрахунки та замовлення труб, є актуальною та практично значущою задачею.

Об'єктом дослідження процес управління замовленнями металопластикових труб.

Предметом дослідження веб-застосунок для автоматизації замовлення металопластикових труб.

Метою роботи є розробка веб-застосунку для автоматизації процесу підбору та замовлення металопластикових труб на основі вхідних параметрів. Система повинна забезпечувати зручний інтерфейс для користувачів, точний розрахунок необхідних матеріалів, формування замовлень та інтеграцію з базою даних для зберігання інформації. Для реалізації поставленої мети використовуються сучасні засоби веб-розробки, зокрема фреймворк Symfony (PHP) та система управління базами даних PostgreSQL [4].

Методи дослідження: На першому етапі було проаналізовано існуючі рішення у сфері автоматизації розрахунку та замовлення будівельних матеріалів, зокрема металопластикових труб, з метою формулювання початкових функціональних і нефункціональних вимог до програмного забезпечення. Далі було здійснено аналіз сучасного стеку технологій, які відповідають поставленим вимогам, у результаті чого для розробки було обрано фреймворк Symfony (PHP) для серверної частини вебзастосунку, GitHub як систему контролю версій. Дослідження ефективності та доцільності реалізації функціоналу проводилось у

процесі безпосередньої розробки програмного забезпечення [5].

Наукова новизна роботи полягає в реалізації спеціалізованого веб-застосунку, що поєднує в собі зручний інтерфейс для введення параметрів, автоматизований онлайн-калькулятор для точного розрахунку необхідної кількості металопластикових труб та формування замовлення. Важливим елементом новизни є інтеграція всієї функціональності в єдину систему, що враховує практичні потреби як підприємств, так і кінцевих споживачів [3].

Практична значущість результатів полягає у можливості використання розробленого застосунку як ефективного інструменту для автоматизації процесу підбору матеріалів, зменшення кількості помилок при розрахунках, економії часу та покращення взаємодії між замовниками та постачальниками в сфері проєктування і реалізації систем з металопластикових труб.

1 АНАЛІЗ ЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ЗАМОВЛЕННЯ МЕТАЛОПЛАСТИКОВИХ ТРУБ

1.1. Веб-застосунок для автоматизації замовлення металопластикових труб

Веб-застосунок для автоматизації замовлення металопластикових труб — це спеціалізована онлайн-система, що забезпечує швидкий і зручний спосіб підбору необхідних матеріалів та формування замовлення відповідно до заданих параметрів. Такий інструмент призначений як для професійного використання компаніями, що займаються проєктуванням та монтажем трубопровідних систем, так і для кінцевих споживачів, які бажають здійснити замовлення без зайвих складнощів.

Основною метою застосунку є підвищення точності розрахунків, зменшення людського фактору при обчисленнях, а також спрощення процедури оформлення замовлення завдяки інтуїтивному інтерфейсу та інтегрованому калькулятору.

Основні компоненти веб-застосунку включають:

– Онлайн-калькулятор, дозволяє ввести технічні параметри (довжина трубопроводу, діаметр, кількість з'єднань, площу приміщення) для автоматичного розрахунку потрібної кількості труб та комплектуючих.

– Система авторизації, забезпечує збереження історії розрахунків і замовлень для зареєстрованих користувачів.

– База даних компонентів, містить актуальні дані щодо асортименту металопластикових труб та фітингів.

– Адміністративна панель, дає змогу оновлювати дані, переглядати замовлення, управляти асортиментом.

Основні цілі веб-застосунку:

– автоматизація процесу розрахунку та замовлення матеріалів;

– зменшення помилок у проєктуванні та підборі комплектуючих;

– покращення комунікації між постачальниками та замовниками;

– економія часу для інженерів і менеджерів проєктів.

Переваги:

– Доступність, користувачі можуть скористатись сервісом з будь-якого пристрою, що має доступ до Інтернету.

– Швидкість та точність, автоматизовані розрахунки мінімізують ризик помилок та пришвидшують процес замовлення.

– Зручність, інтерфейс оптимізовано під потреби як професіоналів, так і пересічних користувачів.

– Гнучкість, можливість змінювати параметри замовлення в режимі реального часу.

– Інтеграція, підтримка подальшого впровадження в системи управління підприємствами.

Недоліки:

– Потреба у стабільному інтернет-з'єднанні: у випадку перебоїв в роботі мережі доступ до сервісу може бути обмежений.

– Технічні складності: для впровадження в бізнес-процеси підприємства можуть знадобитись консультації чи навчання персоналу.

– Залежність від актуальності даних: точність розрахунків напряму залежить від оновленої інформації про асортимент і ціни.

1.2 Специфіка замовлення металопластикових труб.

Металопластикові труби є популярним матеріалом для систем водопостачання та опалення завдяки своїм експлуатаційним властивостям, таким як висока міцність, гнучкість, корозійна стійкість та довговічність. Специфіка замовлення металопластикових труб полягає у необхідності врахування низки технічних та комерційних параметрів, що впливають на кінцевий результат та ефективність використання матеріалу.

Перш за все, важливо визначити тип труб та їх розміри (діаметр, товщина

стінки), оскільки вони повинні відповідати вимогам конкретної системи. Крім того, слід враховувати умови експлуатації, такі як робочий тиск, температура, характер робочої рідини, що можуть значно впливати на вибір оптимального продукту.

Другим важливим аспектом є розрахунок необхідної довжини труб та кількості комплектуючих — фітингів, кріплень, ізоляційних матеріалів. Точність цих даних дозволяє уникнути надлишків або нестачі матеріалів, що сприяє оптимізації витрат і зменшенню часу монтажу.

Також специфіка замовлення передбачає врахування стандартів якості та сертифікації продукції, які гарантують відповідність матеріалу встановленим нормам безпеки і технічним вимогам. Важливим є і вибір постачальника, який може забезпечити надійність, своєчасність доставки та конкурентні ціни.

У зв’язку з цим процес замовлення металопластикових труб часто супроводжується використанням спеціалізованих програмних рішень, таких як онлайн-калькулятори, які автоматизують розрахунок матеріалів та допомагають правильно сформувати замовлення з урахуванням усіх технічних параметрів.

Таким чином, специфіка замовлення металопластикових труб вимагає комплексного підходу, що поєднує технічні знання, точність розрахунків та ефективне управління процесом закупівлі, що в кінцевому результаті забезпечує якість і надійність монтажу трубопровідних систем.

В сучасному динамічному бізнес-середовищі ефективне управління замовленнями є критично важливим фактором успіху для будь-якого підприємства, незалежно від його розміру чи сфери діяльності. Ручна обробка замовлень, особливо при їх значних обсягах, призводить до помилок, затримок та, як наслідок, до втрати клієнтів і прибутку. Саме тому автоматизовані системи управління замовленнями (АСУЗ) стають невід'ємною частиною сучасної бізнес-інфраструктури. Метою даної статті є аналіз існуючих автоматизованих систем управління замовленнями, їх функціональних можливостей, переваг та недоліків [9].

Що таке автоматизована система управління замовленнями?

Автоматизована система управління замовленнями – це програмне

забезпечення або комплекс програмних засобів, призначений для оптимізації та автоматизації всього життєвого циклу замовлення: від його отримання та обробки до виконання та доставки, а також подальшого аналізу. Такі системи дозволяють централізовано керувати всіма етапами, мінімізуючи людський фактор та підвищуючи загальну ефективність бізнес-процесів.

Основні функції автоматизованих систем управління замовленнями:

Сучасні АСУЗ пропонують широкий спектр функціональних можливостей, серед яких можна виділити наступні ключові:

- Прийом та реєстрація замовлень: Автоматичний збір замовлень з різних каналів (веб-сайт, мобільний додаток, електронна пошта, телефонні дзвінки, маркетплейси тощо) та їх консолідація в єдиній системі.

- Обробка замовлень: Перевірка наявності товарів на складі, розрахунок вартості, присвоєння статусу замовленню, інформування клієнта про етапи виконання.

- Управління запасами: Відстеження рівня запасів в режимі реального часу, автоматичне резервування товарів під замовлення, сповіщення про необхідність поповнення запасів.

- Управління клієнтською базою (CRM-функціонал): Зберігання історії замовлень клієнтів, їх контактних даних, переваг, що дозволяє персоналізувати обслуговування.

- Інтеграція з іншими системами: Можливість обміну даними з бухгалтерськими програмами, системами складського обліку (WMS), службами доставки, платіжними шлюзами.

- Формування звітності та аналітика: Збір даних про продажі, ефективність роботи менеджерів, популярність товарів, що дозволяє приймати обґрунтовані управлінські рішення.

- Управління доставкою: Планування маршрутів, відстеження статусу доставки, інтеграція з кур'єрськими службами.

- Автоматизація документообігу: Генерація рахунків-фактур, накладних, актів виконаних робіт та інших необхідних документів [11].

Типи автоматизованих систем управління замовленнями:

Існуючі на ринку АСУЗ можна класифікувати за декількома ознаками:

- За типом розгортання:

- Хмарні (SaaS - Software as a Service): Системи, доступ до яких надається через інтернет за передплатою. Приклади: Salesforce Sales Cloud, Zoho Inventory, Odoo (хмарна версія).

- Коробкові (On-Premise): Програмне забезпечення, яке встановлюється на серверах компанії. Забезпечують вищий рівень контролю над даними, але потребують значних витрат на придбання, впровадження та підтримку. Приклади: Microsoft Dynamics 365 (on-premise), SAP S/4HANA (on-premise).

- За спеціалізацією:

- Універсальні: Підходять для широкого кола галузей та типів бізнесу.

- Спеціалізовані (галузеві): Розроблені з урахуванням специфіки конкретної галузі (наприклад, для ресторанів, інтернет-магазинів, виробничих підприємств).

- За функціональним охопленням:

- Модульні системи: Дозволяють обирати та підключати лише необхідні модулі, адаптуючи систему під потреби бізнесу.

- Комплексні ERP-системи (Enterprise Resource Planning): Охоплюють не тільки управління замовленнями, а й інші бізнес-процеси підприємства (фінанси, виробництво, персонал тощо).

Популярні приклади автоматизованих систем управління замовленнями:

На ринку існує велика кількість АСУЗ, як від великих міжнародних вендорів, так і від локальних розробників. Серед найвідоміших можна відзначити:

- Salesforce Sales Cloud: Потужна CRM-система з розширеними функціями управління продажами та замовленнями. Орієнтована на середній та великий бізнес.

- Zoho Inventory / Zoho One: Комплекс рішень для малого та середнього бізнесу, що включає інструменти для управління запасами, замовленнями, фінансами та взаємовідносинами з клієнтами.

- Odoo: Модульна ERP-система з відкритим кодом, яка пропонує широкий

набір додатків, включаючи управління продажами, складом та замовленнями. Підходить для бізнесу будь-якого розміру.

- Microsoft Dynamics 365 Sales/Supply Chain Management: Комплексне рішення від Microsoft, яке інтегрує управління продажами, обслуговуванням клієнтів, маркетингом та ланцюгами поставок.

- NetSuite (Oracle NetSuite): Хмарна ERP-система, що охоплює управління фінансами, замовленнями, запасами, виробництвом та іншими бізнес-процесами.

- Локальні українські розробки: На українському ринку також представлені рішення від місцевих розробників, які часто краще адаптовані до специфіки українського законодавства та бізнес-практик. Наприклад, "BAS Малий бізнес", "УкрСклад" та інші.

Переваги впровадження АСУЗ:

- Збільщення ефективності: автоматизація повсякдених операцій вивільняє час співробітників для виконання більш важливих завдань.

- Зменшення кількості помилок: Мінімізація людського фактору при обробці даних знижує ризик помилок.

- Прискорення обробки замовлень: Скорочення часу від отримання замовлення до його виконання.

- Покращення обслуговування клієнтів: Швидка реакція на запити, своєчасне інформування про статус замовлення, персоналізований підхід.

- Оптимізація управління запасами: Запобігання дефіциту або надлишку товарів на складі.

- Покращення аналітики та прийняття рішень: Доступ до актуальних даних для аналізу ефективності та планування.

- Підвищення прозорості бізнес-процесів: Можливість відстежувати кожен етап виконання замовлення.

- Збільшення прибутку: За рахунок зниження витрат, підвищення лояльності клієнтів та збільшення обсягів продажів.

Недоліки та виклики при впровадженні АСУЗ:

- Вартість впровадження: Придбання програмного забезпечення,

налаштування, навчання персоналу можуть потребувати значних інвестицій, особливо для коробкових рішень.

- Складність впровадження та інтеграції: Процес впровадження може бути тривалим та вимагати залучення кваліфікованих спеціалістів. Інтеграція з існуючими системами може бути складною.

- Необхідність навчання персоналу: Співробітникам потрібно буде навчитися працювати з новою системою.

- Опір змінам з боку персоналу: Деякі співробітники можуть чинити опір впровадженню нових технологій.

- Залежність від постачальника (для хмарних рішень): У випадку проблем у провайдера доступ до системи може бути обмежений.

Питання безпеки:

Критерії вибору АСУЗ:

При виборі автоматизованої системи управління замовленнями важливо враховувати наступні фактори:

- Розмір та специфіка бізнесу: Потреби малого інтернет-магазину та великого виробничого підприємства будуть суттєво відрізнятися.

- Функціональні можливості: Чи відповідає функціонал системи потребам компанії?

- Вартість: Початкові витрати, вартість передплати.

Можливості інтеграції:

- Масштабованість: Чи зможе система адаптуватися до зростання бізнесу?

- Простота використання та навчання: Наскільки інтуїтивно зрозумілий інтерфейс системи?

- Технічна підтримка.

- Безпека даних: Які заходи безпеки пропонує система?

Висновки

Автоматизовані системи управління замовленнями є потужним інструментом для оптимізації бізнес-процесів, підвищення ефективності та конкурентоспроможності підприємства.

Правильний вибір та успішне впровадження АСУЗ дозволяє компаніям не тільки автоматизувати рутинні операції, але й покращити якість обслуговування клієнтів та, як результат, досягти стабільного зростання та розвитку.

1.3 Цільова аудиторія

Цільова аудиторія проекту або продукту, пов’язаного із замовленням металопластикових труб, охоплює кілька основних груп користувачів і замовників, які безпосередньо залучені до вибору, закупівлі та монтажу трубопровідних систем.

До них належать будівельні компанії та підрядники, які виконують зведення житлових, комерційних і промислових об’єктів і зацікавлені у якісних матеріалах, що відповідають технічним вимогам та забезпечують швидкий і надійний монтаж.

Також важливою групою є інженери та проектувальники, які розробляють технічні рішення для інженерних мереж, підбирають оптимальні параметри труб і комплектуючих, забезпечуючи надійність та ефективність систем.

Оптові та роздрібні постачальники будівельних матеріалів також входять до цільової аудиторії, оскільки вони займаються реалізацією металопластикових труб та комплектуючих, прагнучи оптимізувати асортимент і покращити умови замовлення для кінцевих споживачів.

Індивідуальні споживачі — власники приватних будинків, які самостійно планують встановлення або заміну трубопровідних систем. Для них важливі простота вибору матеріалів, доступність інформації та підтримка в процесі замовлення.

Окрім того, до цільової аудиторії належать монтажні бригади і сервісні компанії, що безпосередньо виконують роботи з монтажу, ремонту та обслуговування систем. Їх цікавлять якість продукції, зручність доставки та технічна підтримка.

Розуміння особливостей кожної з цих груп дозволяє краще адаптувати процес замовлення металопластикових труб, підвищити рівень обслуговування і

забезпечити ефективність роботи всього ланцюжка постачання.

1.4 Дослідження існуючих аналогів

Для успішної реалізації проєкту важливо проаналізувати наявні на ринку рішення, які пропонують автоматизацію процесу замовлення та управління металопластиковими трубами або подібними товарами. Серед популярних платформ варто виділити Sana Commerce та Cloudfy, які представляють собою комплексні рішення для електронної комерції і B2B-продажів.

Sana Commerce — це платформа, яка інтегрується з ERP-системами і орієнтована на автоматизацію процесів електронної комерції для бізнес-клієнтів. Вона забезпечує зручний каталог товарів, персоналізовані цінові пропозиції, управління замовленнями та синхронізацію даних у режимі реального часу. Основною перевагою Sana Commerce є глибока інтеграція з системами управління ресурсами підприємства, що дозволяє знизити ризики помилок і покращити оперативність обробки замовлень [1]. Адмін панель Sana Commerce зображено на рисунку 1.1 та 1.2

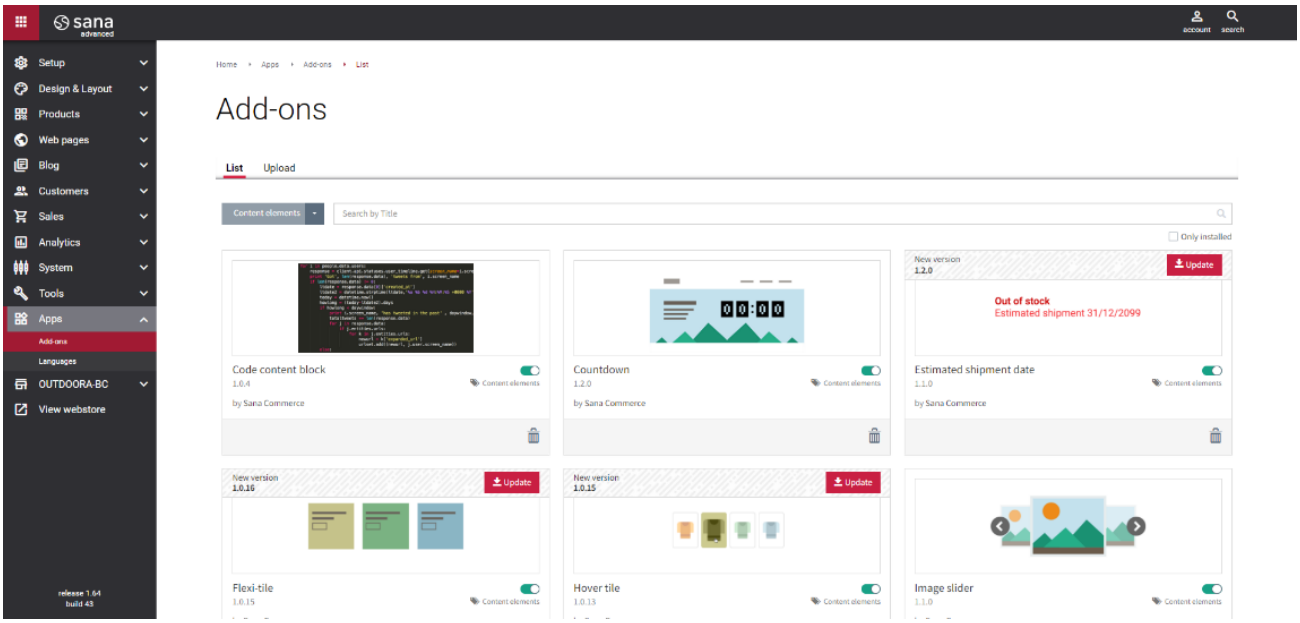


Рис. 1.1 Адмін-панель Sana Commerce

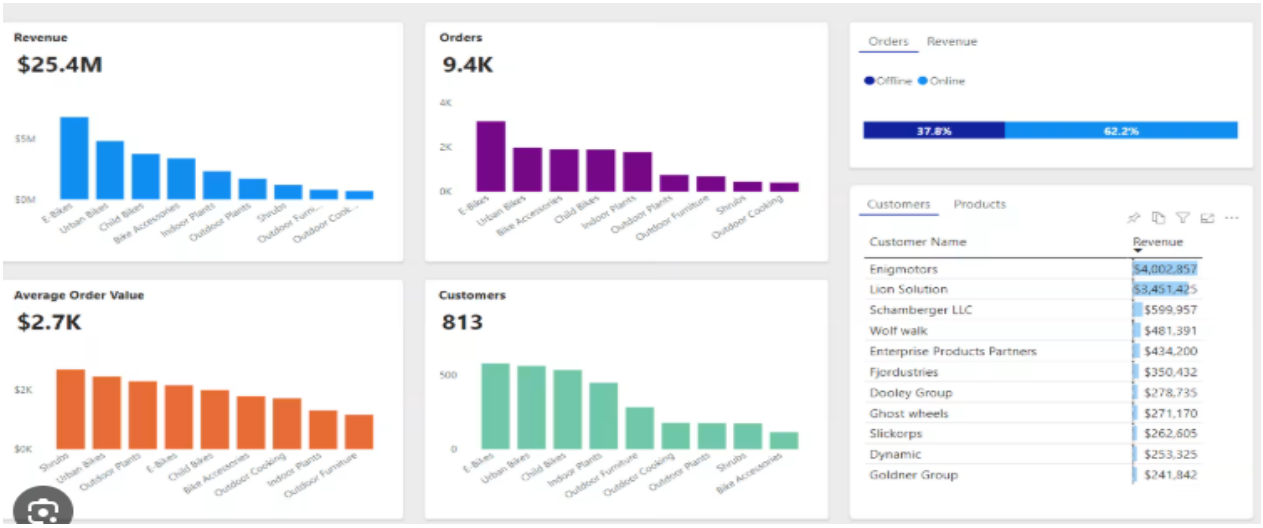


Рис. 1.2 Аналітична панель

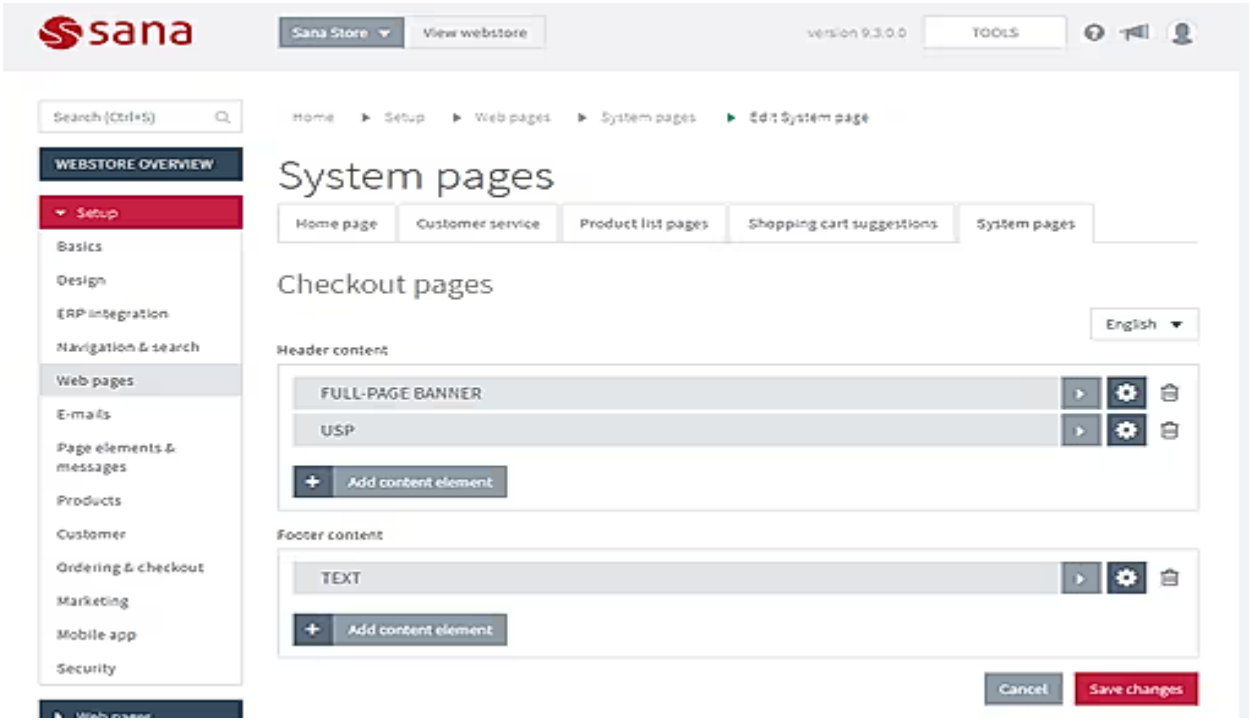


Рис.1.3 Сторінка оформлення замовлення

Cloudfy — це хмарне рішення, що пропонує комплексний інструментарій для організації онлайн-продажів, включаючи управління каталогами, замовленнями, оплатою та логістикою. Cloudfy підтримує масштабування бізнесу, пропонує гнучкі налаштування під специфіку різних галузей та орієнтована як на B2B, так і на B2C сегменти. Завдяки простоті впровадження і користувацькому інтерфейсу, Cloudfy підходить для компаній, які прагнуть швидко вийти на онлайн-ринок без

значних капіталовкладень [2]. Адміністративну панель Cloudfy зображено на рисунку 1.4

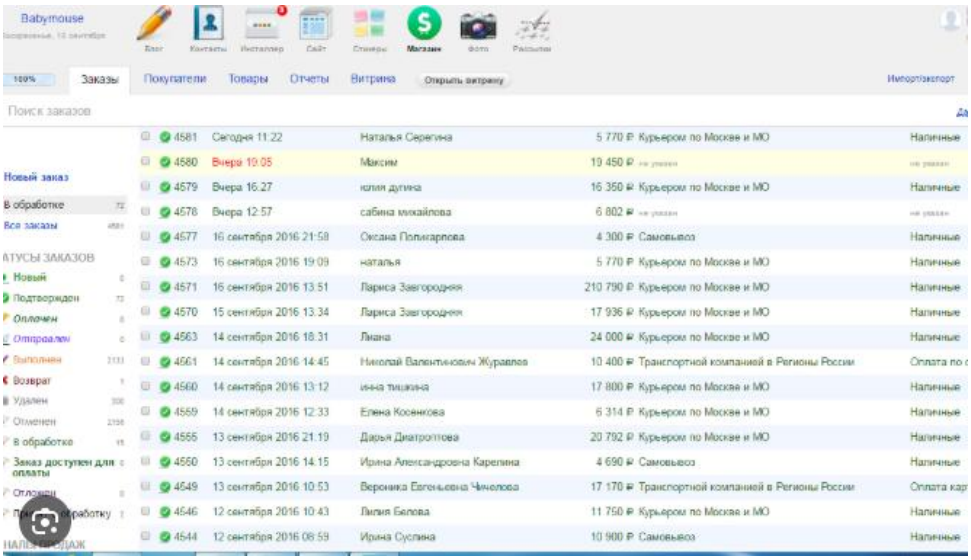


Рис. 1.4 Адмін панель Cloudfy

Обидві платформи мають свої переваги, однак для замовлення металопластикових труб особливо важливим є поєднання точного калькулятора матеріалів, автоматизованого формування замовлень та гнучкої інтеграції з внутрішніми системами підприємства. Дослідження цих аналогів допомагає визначити ключові функціональні вимоги та можливості для розробки ефективного рішення, адаптованого під специфіку галузі.

Таблиця 1.1

Зведена таблиця аналізу існуючих застосунків для автоматизації металопластикових труб

Показник	**Sana Commerce**	**Cloudfy**	**Труба+**
Онлайн-калькулятор матеріалів	-	-	+
Складський облік	-	+	+

Продовження таблиці 1.1

Зведена таблиця аналізу існуючих застосунків для автоматизації металопластикових труб

Показник	**Sana Commerce**	**Cloudfy**	**Труба+**
Спосіб інтеграція з іншими системами	-	1C	REST API
Адаптивність під мобільні пристрої	-	+	+
Доступність на платформах	Web	Web	Web

1.5 Визначення вимог до застосунку

На основі аналізу предметної області, дослідження аналогів та потреб цільової аудиторії було сформульовано перелік функціональних і нефункціональних вимог до інформаційної системи для замовлення металопластикових труб.

Функціональні вимоги

Система має забезпечувати такі можливості:

1. Реєстрація та авторизація користувачів різних ролей: клієнта, менеджера, адміністратора.

2. Гнучке налаштування прав доступу відповідно до ролі користувача, що дозволяє обмежити або надати доступ до певних функцій.

3. Динамічне оновлення асортименту продукції з урахуванням змін на складі або нових надходжень.

4. Можливість фільтрації та пошуку товарів за технічними параметрами, категоріями, виробниками тощо.

5. Створення кошика замовлень із можливістю додавання, редагування та

видалення товарних позицій.

6. Автоматичний розрахунок вартості замовлення з урахуванням знижок, акцій, оптових цін або спеціальних умов.

7. Розрахунок необхідної кількості матеріалів відповідно до типу сантехнічних робіт (наприклад, опалення, водопостачання, тепла підлога тощо).

Нефункціональні вимоги

Окрім основного функціоналу, застосунок має відповідати низці нефункціональних вимог:

1. Генерація звітів щодо продажів, популярності товарів, залишків на складі.

2. Візуалізація ключових даних у вигляді графіків і діаграм для зручного сприйняття аналітичної інформації.

3. Захист даних за допомогою шифрування (протоколи SSL/TLS) та реалізація двофакторної аутентифікації для підвищення безпеки [23].

4. Адаптивний користувацький інтерфейс, який забезпечує коректну роботу на пристроях з різними розмірами екранів (ПК, планшети, смартфони).

Багатомовна підтримка, зокрема української та англійської мов, для забезпечення доступності для ширшого кола користувачів.

Вимоги до програмного забезпечення для автоматизації замовлення:

Функціональні вимоги (ведення каталогу продукції, розрахунок вартості, формування замовлення, управління статусами замовлень, генерація звітності, управління користувачами тощо).

Нефункціональні вимоги (продуктивність, надійність, безпека, зручність використання, масштабованість).

Автоматизація процесу замовлення товарів та послуг в сучасному світі. Впровадження спеціалізованого програмного забезпечення дозволяє оптимізувати робочі процеси, зменшити кількість помилок, прискорити обробку замовлень та покращити загальний рівень обслуговування клієнтів.

Функціональні вимоги описують конкретні дії та операції, які повинна виконувати система. Вони визначають, що саме має робити програмне забезпечення. Для системи автоматизації замовлень ключовими функціональними вимогами є:

Ведення каталогу продукції:

- можливість додавання, редагування та видалення товарів або послуг;

- детальний опис кожного продукту (назва, артикул, опис, характеристики, ціна, наявність на складі, фотографії/відео);

- класифікація та категоризація продукції для зручного пошуку та навігації;

- управління варіантами товарів (наприклад, розмір, колір);

- можливість імпорту/експорту даних каталогу.

Розрахунок вартості:

- автоматичний розрахунок загальної вартості замовлення на основі обраних товарів, їх кількості та цін;

- врахування знижок, акцій, промокодів та програм лояльності;

- розрахунок вартості доставки в залежності від обраного способу, ваги, габаритів та адреси доставки;

- відображення податків та інших можливих зборів.

Формування замовлення:

- інтуїтивно зрозумілий інтерфейс для додавання товарів до кошика та оформлення замовлення;

- можливість вибору способу оплати (готівка, банківська картка, онлайн-платежі тощо) та інтеграція з платіжними системами;

- можливість вибору способу доставки та введення адреси;

- збереження історії замовлень для зареєстрованих користувачів;

- автоматичне надсилання підтвердження замовлення клієнту та адміністратору.

Управління статусами замовлень:

- відстеження життєвого циклу замовлення (наприклад, "Нове", "В обробці", "Очікує оплати", "Оплачено", "Комплектується", "Відправлено", "Доставлено", "Скасовано", "Повернено");

- можливість зміни статусу замовлення адміністратором;

- автоматичне інформування клієнта.

Генерація звітності:

- формування звітів за різними параметрами: обсяги продажів, популярні товари, ефективність каналів залучення клієнтів, фінансові показники тощо;

- можливість налаштування періодичності та формату звітів;

- візуалізація даних у вигляді графіків та діаграм для кращого аналізу;

- експорт звітів у популярні формати (наприклад, Excel, PDF).

Управління користувачами:

- реєстрація та авторизація користувачів (клієнтів та адміністраторів);

- розподіл прав доступу;

- можливість управління профілями користувачів (редагування даних, зміна пароля);

- ведення бази даних клієнтів.

Інтеграція з іншими системами (за потреби):

- інтеграція з системами складського обліку для актуалізації інформації про наявність товарів;

- інтеграція з CRM-системами для комплексної роботи з клієнтами;

- інтеграція зі службами доставки для автоматизації процесу відправлення та відстеження;

- інтеграція з бухгалтерськими програмами для обміну фінансовими даними.

Нефункціональні вимоги визначають умови виконання функцій системи. Вони описують атрибути якості програмного забезпечення та обмеження, що накладаються на його розробку та експлуатацію. Для системи автоматизації замовлень важливими нефункціональними вимогами є:

Продуктивність:

- Швидкий час відгуку системи на дії користувача (наприклад, завантаження сторінок, пошук товарів, оформлення замовлення).

- Здатність системи обробляти очікувану кількість одночасних користувачів та замовлень без зниження продуктивності.

Оптимізація запитів до бази даних та ефективне використання ресурсів сервера.

Надійність:

- Стабільна робота системи без збоїв та помилок.

- Здатність системи відновлювати свою працездатність після можливих відмов (наприклад, через механізми резервного копіювання та відновлення даних).

- Мінімізація часу простою системи.

Безпека:

- Захист конфіденційних даних користувачів.

- Використання шифрування для передачі та зберігання чутливої інформації.

- Захист від поширених веб-вразливостей (наприклад, SQL-ін'єкції, XSS-атаки).

- Механізми аутентифікації та авторизації для забезпечення доступу лише авторизованим користувачам.

- Регулярне оновлення системи безпеки.

- Зручність використання (Usability):

- Інтуїтивно зрозумілий, логічний та простий у використанні інтерфейс як для клієнтів, так і для адміністраторів.

- Чітка навігація та структура сайту/додатку.

- Наявність довідкової інформації та підтримки користувачів.

- Мінімальна кількість кроків для виконання основних операцій (наприклад, оформлення замовлення).

Масштабованість:

- Здатність системи адаптуватися до кількості користувачів та навантаження без суттєвої втрати продуктивності або необхідності повної перебудови архітектури.

- Гнучка архітектура, що дозволяє легко розширювати ресурси (наприклад, серверні потужності).

Ретельне визначення та документування є критично важливим етапом розробки програмного забезпечення для автоматизації замовлень. Це дозволяє створити систему, яка не тільки ефективно вирішує поставлені бізнес-завдання, але й є надійною, безпечною, зручною у використанні та готовою до майбутнього зростання. Чітко сформульовані вимоги слугують основою для проектування, розробки, тестування та успішного впровадження програмного продукту.

Опис бізнес-процесу замовлення металопластикових труб:

Від вибору до отримання.

Замовлення металопластикових труб – це багатоетапний процес, що вимагає взаємодії різних учасників та належного документального оформлення. Розуміння цього процесу є ключовим як для клієнтів, що прагнуть ефективно забезпечити свої потреби, так і для компаній, що займаються продажем та постачанням даної продукції.

Етапи процесу:

від вибору до отримання замовлення

Процес замовлення металопластикових труб зазвичай проходить через наступні ключові етапи:

- ініціація потреби та вибір продукції (Клієнт):

✓ Визначення потреби. Клієнт (фізична особа або представник юридичної особи) визначає необхідність у металопластикових трубах, їх тип, діаметр, довжину, кількість та інші технічні характеристики відповідно до проекту або потреби (наприклад, для систем опалення, водопостачання);

✓ Пошук постачальника. Клієнт здійснює пошук потенційних постачальників через інтернет, рекомендації, рекламні оголошення тощо;

✓ Консультація та вибір. Клієнт звертається до обраних постачальників для отримання консультації, уточнення характеристик продукції, наявності, цін та умов поставки. На цьому етапі важливу роль відіграє менеджер з продажу;

✓ Ознайомлення з асортиментом. Клієнт вивчає каталоги, зразки продукції (за можливості), технічну документацію.

Оформлення замовлення (Клієнт, Менеджер з продажу):

- формування замовлення. Клієнт формує попереднє замовлення, вказуючи найменування товару, кількість, бажані терміни отримання;

- узгодження деталей. Менеджер з продажу зв'язується з клієнтом для підтвердження замовлення, узгодження всіх деталей (точна кількість, адреса доставки, спосіб оплати, контактні дані);

- перевірка наявності на складі. Менеджер перевіряє наявність замовлених позицій на складі компанії. У випадку відсутності – уточнює терміни поставки від виробника/постачальника.

Обробка замовлення та підготовка до відвантаження (Менеджер з продажу, Склад):

- резервування товару. За наявності товару на складі, менеджер резервує його під конкретне замовлення;

- формування рахунку-фактури. Менеджер виставляє клієнту рахунок-фактуру для оплати;

- комплектація замовлення. Після підтвердження оплати (або згідно з умовами договору) працівники складу комплектують замовлення відповідно до заявки: відбирають труби потрібного типу, діаметру та кількості, перевіряють їх якість та комплектність (наприклад, наявність фітингів, якщо вони входять у замовлення);

- підготовка супровідних документів. Готуються необхідні документи для відвантаження (видаткова накладна, товарно-транспортна накладна (ТТН) у разі доставки).

Оплата замовлення (Клієнт):

- клієнт здійснює оплату замовлення згідно з виставленим рахунком-фактурою (безготівковий розрахунок, готівка, онлайн-оплата – залежно від умов постачальника).

Відвантаження та доставка замовлення (Склад, Служба доставки/Клієнт):

- самовивіз. Клієнт самостійно забирає замовлення зі складу постачальника. Працівник складу видає товар та підписує з клієнтом видаткову накладну;

- доставка постачальником. Компанія-постачальник організовує доставку замовлення за вказаною клієнтом адресою власним транспортом або із залученням логістичної компанії;

- передача товару експедитору. Якщо доставка здійснюється перевізником, товар передається експедитору разом із ТТН.

Отримання замовлення (Клієнт):

- приймання товару. Клієнт приймає товар, перевіряє його відповідність замовленню за кількістю, асортиментом та відсутністю видимих пошкоджень;

- підписання документів. Клієнт підписує видаткову накладну (та/або ТТН), підтверджуючи отримання товару та відсутність претензій до його зовнішнього вигляду та комплектності. Один примірник документів залишається у клієнта, інший – повертається постачальнику.

Учасники процесу.

У процесі замовлення металопластикових труб беруть участь наступні сторони:

- Клієнти. Ініціатори замовлення, кінцеві споживачі продукції. Це можуть бути як фізичні особи для приватних потреб (ремонт, будівництво будинку), так і юридичні особи (будівельні компанії, монтажні організації). Їхня головна роль – чітко сформулювати потребу, своєчасно оплатити та прийняти товар.

- Менеджери з продажу (представники компанії-постачальника). Ключова ланка комунікації з клієнтом. Відповідають за консультування, прийом та обробку замовлень, узгодження умов, виставлення рахунків, контроль оплати та координацію відвантаження.

- Склад (працівники складу компанії-постачальника): Відповідають за зберігання продукції, її комплектацію згідно із замовленням, підготовку до відвантаження, видачу товару клієнту або передачу службі доставки. Також можуть відповідати за ведення складського обліку.

- Постачальники (виробники або дистриб'ютори): вони відповідають за своєчасне постачання продукції компанії-продавцю.

- Служба доставки (логістичні компанії або власний транспортний відділ постачальника): Відповідають за транспортування замовлення від складу продавця до клієнта.

Документообіг при замовленні.

Належний документообіг є невід'ємною частиною процесу замовлення та гарантує прозорість і юридичну чистоту угоди. Основні документи, що супроводжують замовлення металопластикових труб:

- Заявка від клієнта (комерційна пропозиція). Може бути усною, письмовою (електронний лист) або оформленою через сайт компанії. Фіксує початковий запит клієнта;

- Рахунок-фактура (Invoice). Основний документ для оплати. Містить інформацію про продавця та покупця, перелік товарів, банківські реквізити;

- Договір поставки (зазвичай для юридичних осіб або великих замовлень). Регламентує права та обов'язки сторін, умови поставки, оплати, відповідальність, порядок вирішення спорів.

- Довіреність (у випадку отримання товару представником юридичної особи). Підтверджує повноваження особи на отримання матеріальних цінностей.

- Видаткова накладна (Накладна на відпуск товарно-матеріальних цінностей). Підтверджує факт передачі товару від продавця до покупця. Містить перелік товарів, їх кількість, ціну, загальну вартість. Підписується обома сторонами.

- Товарно-транспортна накладна (ТТН). Оформлюється у випадку доставки товару автомобільним транспортом. Є основним документом на вантаж, що супроводжує його на шляху прямування.

- Сертифікати якості/відповідності на продукцію. Надаються за вимогою клієнта та підтверджують відповідність труб встановленим стандартам.

- Касовий чек або квитанція (при готівковій оплаті). Підтверджує факт оплати.

- Акт приймання-передачі товару. Може оформлюватися додатково до видаткової накладної, особливо при виявленні розбіжностей або для детальної фіксації стану товару.

Розуміння всіх етапів, ролей учасників та документообігу дозволяє оптимізувати процес замовлення металопластикових труб, мінімізувати ризики та забезпечити своєчасне отримання якісної продукції.

2 ЗАСОБИ РЕАЛІЗАЦІЇ

2.1 Середовище розробки

Для реалізації програмного забезпечення було обрано сучасне середовище розробки **PhpStorm**, яке є потужним інструментом для створення веб-додатків на мові програмування PHP. PhpStorm надає розширену підтримку синтаксису, автоматичне доповнення коду, інструменти для налагодження (debugging), тестування та роботи з базами даних [13].

Серед основних причин вибору саме PhpStorm варто відзначити такі:

- Повна інтеграція з системами контролю версій (наприклад, Git).
- Вбудовані інструменти для роботи з фреймворками (Symfony, Laravel та інші).
- Зручна навігація по проєкту та швидкий пошук.
- Підтримка шаблонізаторів, JavaScript, HTML та CSS.
- Можливість налаштування середовища згідно з індивідуальними потребами розробника.

Операційною системою під час розробки була обрана **Windows 11**, що забезпечило стабільну роботу інструментів і серверного оточення. Для локального розгортання проєкту використовувався вебсервер Apache або Nginx із підтримкою PHP та СУБД PostgreSQL / MySQL (залежно від обраної технології бази даних) [6].

Використання PhpStorm дозволило підвищити ефективність розробки, забезпечити дотримання стандартів кодування та зменшити кількість технічних помилок.

2.2 Мови програмування

Під час розробки програмного забезпечення було використано такі мови програмування:

PHP — основна серверна мова, яка використовується для реалізації бізнес-

логіки застосунку, обробки запитів користувачів, формування HTML-вмісту та взаємодії з базою даних. PHP має широку підтримку у веб-розробці, включаючи фреймворки, що пришвидшують і стандартизують процес створення програмних рішень [6].

SQL (Structured Query Language) — мова структурованих запитів, призначена для взаємодії з реляційною базою даних. За допомогою SQL реалізовано збереження, пошук, агрегацію та модифікацію даних, що забезпечує повноцінне функціонування внутрішньої частини системи [7].

Для оптимізації процесу розробки та дотримання принципів MVC-архітектури було використано фреймворк

Symfony — один із найпотужніших і найпопулярніших PHP-фреймворків. Його застосування дало змогу:

- структурувати проєкт відповідно до принципів розділення відповідальностей (контролери, моделі, шаблони);
- реалізувати багаторазово використовувані компоненти (форми, валідація, маршрутизація);
- покращити безпеку та масштабованість застосунку;
- пришвидшити розробку за рахунок готових рішень і автоматизації рутинних задач [4].

Таким чином, комбінація PHP, SQL та фреймворку Symfony забезпечила ефективну реалізацію веб-застосунку з урахуванням сучасних вимог до продуктивності, безпеки та зручності підтримки коду.

2.3 Система управління базою даних

Для зберігання, обробки та організації даних у проєкті було застосовано систему **PostgreSQL**.

СУБД обрана з огляду на її стабільність, високу продуктивність, відповідність вимогам до масштабованості та підтримку складних SQL-запитів. Серед ключових переваг обраної СУБД:

- підтримка транзакцій та цілісності даних;

- широкі можливості для побудови складних запитів та фільтрації даних;
- розвинені механізми роботи з індексами, що підвищують швидкодію системи;
- підтримка агрегатних функцій, групування, представлень (view), збережених процедур і тригерів;
- активна спільнота та наявність інструментів для резервного копіювання й відновлення.

У системі зберігаються всі основні сутності: користувачі, товари, замовлення, кошики, аналітичні дані, звіти тощо. Схема бази даних була спроєктована відповідно до принципів нормалізації з метою уникнення надлишковості даних і забезпечення логічної цілісності.

Для зручності розробки та тестування використовувалися інструменти адміністрування бази даних, такі як **phpPgAdmin**, **Adminer** або інтеграція з середовищем **PhpStorm**, що надало змогу працювати з SQL-запитами безпосередньо з IDE [12].

2.4 Система контролю версій

При створенні ПО для зберігання, відстеження змін і спільної роботи над кодом було використано систему контролю версій **Git**.

Git є розподіленою системою контролю версій, яка дозволяє ефективно керувати історією змін у проєкті, створювати гілки для розробки окремих функціональностей, а також виконувати злиття та вирішення конфліктів коду. Завдяки Git розробка стала більш структурованою, із можливістю повернення до попередніх стабільних версій застосунку в разі потреби [7].

Для зберігання репозиторію та віддаленого доступу до нього було використано платформу **GitHub** Вона надала зручний веб-інтерфейс для керування репозиторієм, перегляду історії комітів, створення pull-запитів, а також базових інструментів CI/CD у разі потреби [8].

3 ПРОЄКТУВАННЯ

3.1 Проєктування архітектури веб-застосунку

Для ефективного планування структури, логіки та взаємодії компонентів веб-застосунку було проведено поетапне проєктування архітектури, з використанням нотацій UML (Unified Modeling Language). Це дозволило формалізувати вимоги, зрозуміло описати функціональність системи та полегшити реалізацію логіки застосунку.

У процесі проєктування було створено наступні діаграми:

- Діаграма варіантів використання (use case diagram):

Описує типові сценарії використання системи користувачами. Серед основних акторів – клієнт, менеджер, адміністратор. Для кожного з них визначено набір доступних функцій: авторизація, перегляд товарів, оформлення замовлень, керування асортиментом тощо.

- Діаграма діяльності (activity diagram):

Описує алгоритмічні процеси в межах системи. Наприклад, логіку оформлення замовлення, обробку запиту на розрахунок матеріалів, або послідовність дій при додаванні товару до кошика. Ця діаграма дозволила проаналізувати послідовність кроків, умови переходів та гілкування логіки.

- Діаграма класів (class diagram):

Описує структуру застосунку у вигляді сутностей (класів), їхніх властивостей та методів, а також зв’язків між ними (асоціації, агрегації, наслідування). Класи представляють основні елементи доменної моделі, зокрема: Користувач, Замовлення, Товар, Кошик, Розрахунок, тощо. Це забезпечує чітке уявлення про внутрішню організацію системи та її розширюваність.

Застосування UML-діаграм дало змогу узгодити вимоги до функціональності, структурувати проєкт відповідно до принципів об'єктно-орієнтованого програмування та покращити якість реалізації веб-застосунку.

3.2 Архітектура клієнтської частини

Клієнтська частина веб-застосунку реалізована за допомогою стандартного стеку веб-технологій: **HTML**, **CSS** та **JavaScript**.

HTML (HyperText Markup Language) використовується для створення структури веб-сторінок. Він визначає логічну організацію контенту: заголовки, форми, таблиці, списки, кнопки, текстові поля тощо [10].

CSS (Cascading Style Sheets) відповідає за візуальне оформлення інтерфейсу: кольори, шрифти, відступи, розмітка елементів. Завдяки адаптивній верстці забезпечено коректне відображення сторінок на різних пристроях (ПК, планшет, смартфон).

JavaScript використовується для реалізації динамічної поведінки елементів інтерфейсу, зокрема:

- взаємодія з формами (валидація, обробка подій);
- оновлення вмісту сторінки без перезавантаження (наприклад, додавання товару в кошик);
- обробка запитів до серверної частини (через AJAX);
- покращення зручності користування застосунком (випадаючі списки, підказки, модальні вікна).

Архітектура побудована за класичною схемою «тонкого клієнта», де основна логіка зосереджена на стороні сервера, а клієнтська частина відповідає за відображення інформації та базову взаємодію з користувачем.

Такий підхід забезпечив швидкість завантаження, мінімальну складність підтримки та сумісність з більшістю сучасних веб-браузерів [18].

3.3 Архітектура серверної частини

Серверна частина веб-застосунку реалізована з використанням мови програмування **PHP** у поєднанні з сучасним фреймворком **Symfony** та системою керування базами даних **PostgreSQL**.

Symfony — це потужний MVC-фреймворк (Model-View-Controller), який забезпечує гнучку та масштабовану структуру проєкту, підтримує принципи розділення відповідальностей і сприяє дотриманню кращих практик розробки [4].

Основні компоненти архітектури:

Контролери (Controllers)

Обробляють HTTP-запити, отримані з клієнтської частини, викликають відповідну бізнес-логіку та повертають HTTP-відповіді. Вони є посередниками між інтерфейсом користувача та логікою застосунку.

Сервіси (Services)

Містять бізнес-логіку застосунку: обробку замовлень, розрахунок матеріалів, застосування знижок, керування правами доступу тощо. Сервіси інкапсулюють функціональність, яка повторюється або є складною.

Сутності (Entities)

Відповідають об'єктно-реляційному відображенню (ORM) даних між таблицями, об'єктами PHP. Для роботи з БД використовується компонент **Doctrine ORM**, який підтримується Symfony та полегшує взаємодію з PostgreSQL [5].

Репозиторії (Repositories)

Реалізують доступ до даних у базі (CRUD-операції), дають змогу отримувати, зберігати та фільтрувати інформацію згідно з потребами застосунку.

Система маршрутизації (Routing)

Забезпечує зіставлення URL-адрес із відповідними контролерами, підтримуючи REST-підхід до побудови маршруту.

Безпека (Security)

Реалізовано автентифікацію користувачів (клієнт, менеджер, адміністратор), авторизацію та захист даних. Використано вбудовані механізми Symfony для керування ролями, а також підтримку двофакторної аутентифікації.

Принципи реалізації:

Дотримання принципів **SOLID** та **DRY** (Don’t Repeat Yourself);

Структурованість і повторне використання коду завдяки використанню **Dependency Injection**;

Можливість розширення завдяки компонентній архітектурі Symfony.

Реалізація серверної частини за допомогою Symfony надала змогу створити надійний, гнучкий і масштабований застосунок, що легко підтримується та розвивається. На рисунку 3.1 зображено діаграму варіантів використання та на рисунку 3.2 зображено діаграму класів. На рисунку 3.3 зображено діаграму діяльності.

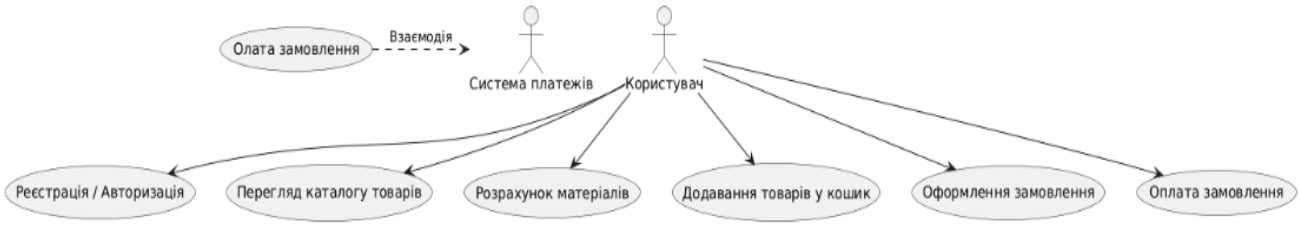


Рис. 3.1 Діаграма варіантів використання

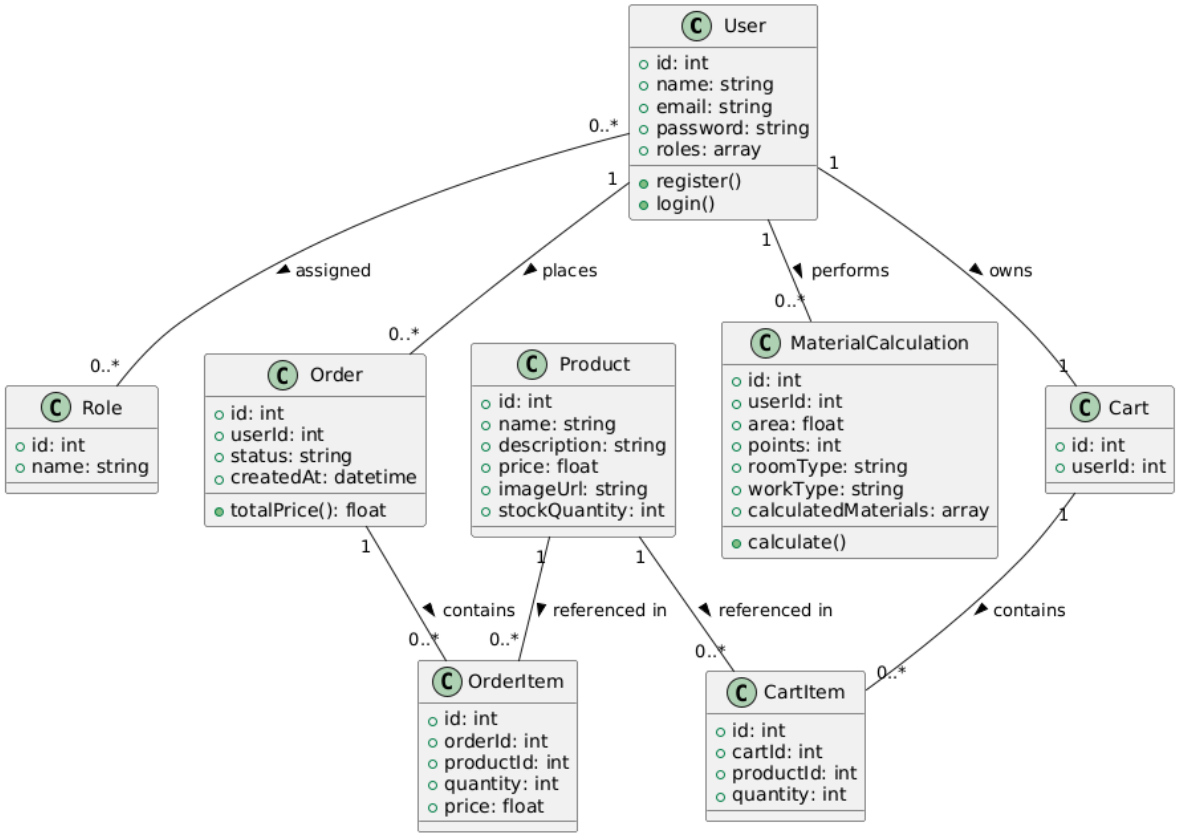


Рис. 3.2 Діаграма класів

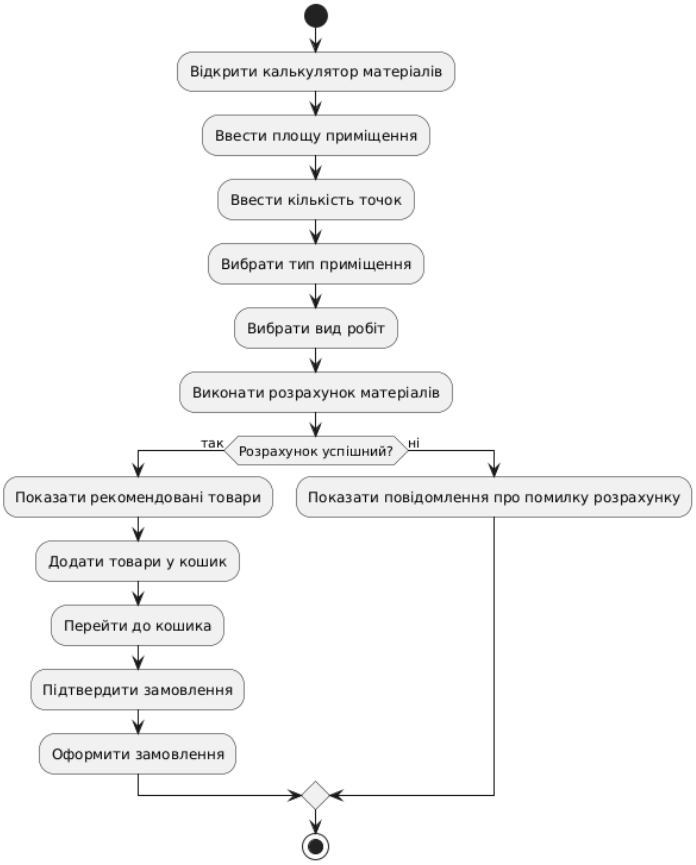


Рис. 3.3 Діаграма діяльності

3.4 Архітектура бази даних

База даних веб-застосунку реалізована на основі системи керування базами даних **PostgreSQL**, яка забезпечує надійне зберігання, цілісність і швидкий доступ до інформації.

Основні особливості архітектури бази даних:

- **Структура даних** побудована у вигляді реляційної моделі, що складається з таблиць, які відображають ключові сутності системи — користувачі, товари, замовлення, кошики, історія змін тощо.
- **Відношення між таблицями** реалізовані за допомогою первинних та зовнішніх ключів, що забезпечує цілісність даних і дозволяє коректно зв’язувати записи між собою. Наприклад, таблиця замовлень пов’язана з таблицею користувачів, а товари — із замовленнями через проміжні таблиці.
- **Нормалізація даних** застосована для уникнення дублювань та підвищення

ефективності зберігання. Основні таблиці розбиті на логічні підмножини, що відповідають принципам третьої нормальної форми (3NF).

- **Індексація** найбільш використовуваних полів забезпечує швидкий пошук і фільтрацію товарів, користувачів, замовлень за різними параметрами.
- **Тригери та обмеження цілісності** використовуються для автоматизації деяких бізнес-правил на рівні бази даних (наприклад, оновлення статусу замовлення або контролю кількості товару на складі).
- **Підтримка транзакцій** гарантує коректність виконання складних операцій, що включають кілька змін у базі, і забезпечує атомарність, консистентність, ізоляцію та надійність (ACID).
- **Резервне копіювання та відновлення** налаштовані для запобігання втрати даних.

Таблиці бази даних:

- **Користувачі (Users):** зберігає інформацію про зареєстрованих клієнтів, менеджерів та адміністраторів, включно з даними для аутентифікації та ролями.
- **Товари (Products):** містить інформацію про металопластикові труби та супутні матеріали, їх характеристики, ціни, наявність.
- **Замовлення (Orders):** фіксує дані про зроблені замовлення, статуси, дату створення, користувача, що зробив замовлення.
- **Кошик (Cart):** тимчасове сховище товарів, які клієнт додав перед оформленням замовлення.

Архітектура бази даних забезпечує ефективну підтримку функціональності застосунку, дозволяє швидко масштабувати систему та інтегрувати додаткові модулі в майбутньому. На рисунку 3.4 зображено структуру таблиці товарів, на рисунку 3.5 зображено структуру таблиці користувачів.

Назва поля	Тип даних	Опис	Обмеження
`id`	SERIAL (INTEGER)	Унікальний ідентифікатор товару	PRIMARY KEY, NOT NULL
`name`	VARCHAR(255)	Назва товару	NOT NULL
`description`	TEXT	Опис товару	NULLABLE
`category_id`	INTEGER	Ідентифікатор категорії товару	FOREIGN KEY, NOT NULL
`price`	NUMERIC(10, 2)	Ціна товару	NOT NULL
`stock_quantity`	INTEGER	Кількість товару на складі	NOT NULL, DEFAULT 0
`is_active`	BOOLEAN	Статус активності товару	NOT NULL, DEFAULT TRUE
`created_at`	TIMESTAMP	Дата створення запису	NOT NULL, DEFAULT NOW()
`updated_at`	TIMESTAMP	Дата останнього оновлення	NOT NULL, DEFAULT NOW()

Рис. 3.4 Структура таблиці товарів

Назва поля	Тип даних	Опис	Додаткові властивості
id	SERIAL / INT	Унікальний ідентифікатор	Первинний ключ (PK), NOT NULL
username	VARCHAR(50)	Логін користувача	Унікальне, NOT NULL
email	VARCHAR(100)	Електронна пошта	Унікальне, NOT NULL
password_hash	VARCHAR(255)	Хеш пароля	NOT NULL
role	VARCHAR(20)	Роль користувача (клієнт, менеджер, адміністратор)	NOT NULL
created_at	TIMESTAMP	Дата і час створення запису	NOT NULL, DEFAULT NOW()
updated_at	TIMESTAMP	Дата і час останнього оновлення	

Рис. 3.5 Структура таблиці користувачі

Таблиця: order (замовлення)

Назва поля	Тип даних	Опис	Додаткові властивості
id	SERIAL / INT	Унікальний ідентифікатор замовлення	Первинний ключ (PK), NOT NULL
user_id	INT	Ідентифікатор користувача, що зробив замовлення	Зовнішній ключ (FK) на user(id)
order_date	TIMESTAMP	Дата і час створення замовлення	NOT NULL, DEFAULT NOW()
status	VARCHAR(30)	Статус замовлення (нове, в обробці, завершено, скасовано)	NOT NULL
total_amount	DECIMAL(10,2)	Загальна сума замовлення	NOT NULL
delivery_address	VARCHAR(255)	Адреса доставки	
payment_method	VARCHAR(50)	Спосіб оплати	
updated_at	TIMESTAMP	Дата і час останнього оновлення	

Рис. 3.6 Структура таблиці замовлення

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

4.2 Реалізація клієнтської частини

Реалізація клієнтської частини веб-додатку для автоматизації замовлення металопластикових труб була орієнтована на створення зручного, інтуїтивного інтерфейсу з максимальною автоматизацією процесу підбору матеріалів. Основним елементом фронтенду став онлайн-калькулятор, який дозволяє користувачам швидко отримувати розрахунки на основі введених параметрів.

Інтерфейс розроблений з урахуванням сучасних тенденцій UX/UI-дизайну, що забезпечило просту навігацію та мінімалістичний вигляд. В якості технологічного стеку для клієнтської частини були обрані HTML5 для семантичної розмітки, CSS3 з використанням препроцесора SCSS для гнучкої стилізації та JavaScript для динамічної взаємодії. Для інтеграції з бекендом на Symfony використовувався Twig як шаблонізатор, що дозволило ефективно керувати відображенням даних. Адаптивність інтерфейсу була досягнута завдяки застосуванню Bootstrap 5.

Елементом клієнтської частини стала форма введення параметрів, яка включала поля для площі приміщення, типу робіт та кількості точок підключення. Кожен параметр супроводжувався підказками та валідацією, що запобігало помилкам при введенні даних. Після заповнення форми система автоматично відправляла запит до сервера, де відбувався розрахунок необхідних матеріалів. Результати розрахунків, такі як рекомендований діаметр труби, загальний метраж, кількість фітингів та додаткові матеріали, миттєво відображалися в спеціальному блоці. Для покращення користувацького досвіду було реалізовано динамічне оновлення даних без перезавантаження сторінки, що досягалося завдяки використанню AJAX-запитів. Кнопка "Додати до кошика", інтегрована з Sylius, дозволяла користувачам одразу переходити до оформлення замовлення, що значно прискорювало процес покупки.На рисунку 4.1 зображено екранну форму перегляду

товару. На рисунку 4.2 зображено форму підбору товарів.

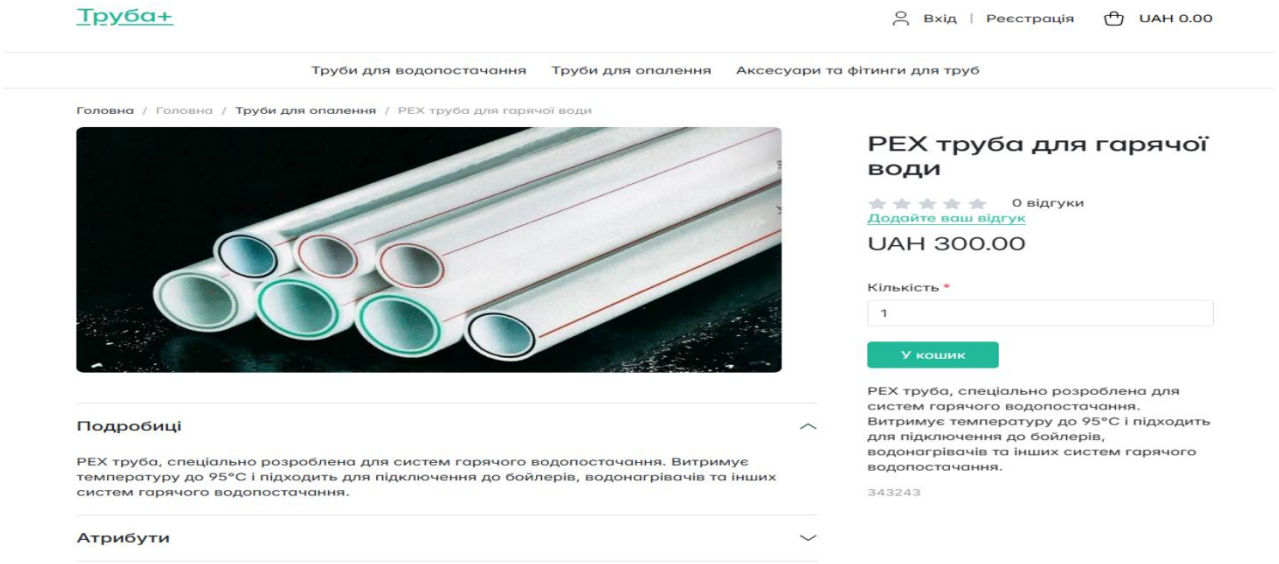


Рис. 4.1 Екранна форма перегляду товару

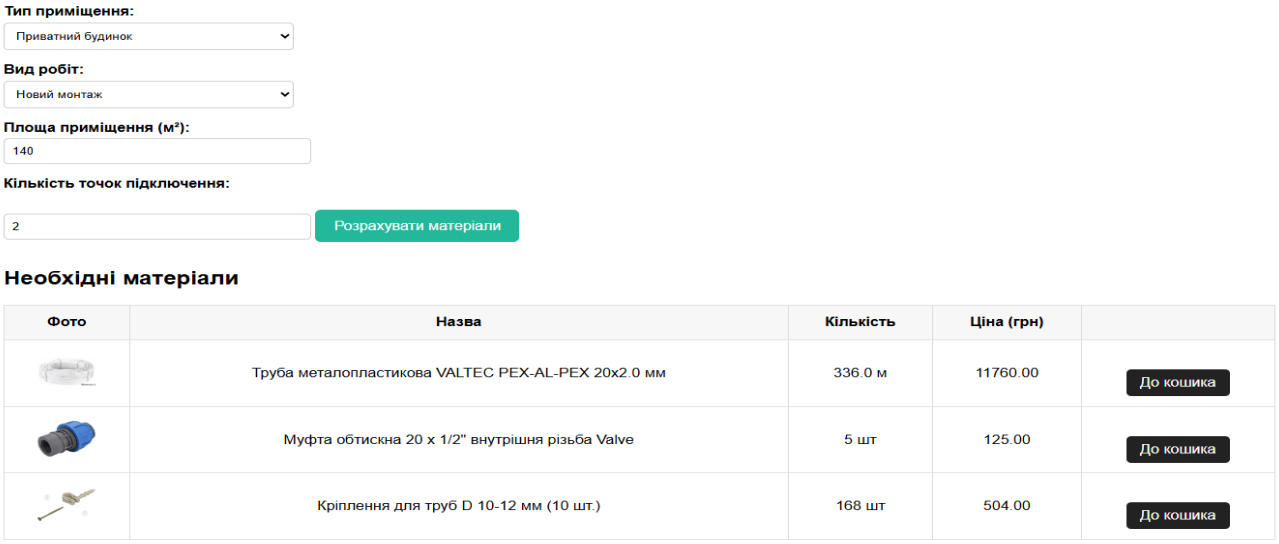


Рис. 4.2 Екранна форма підбору товарів

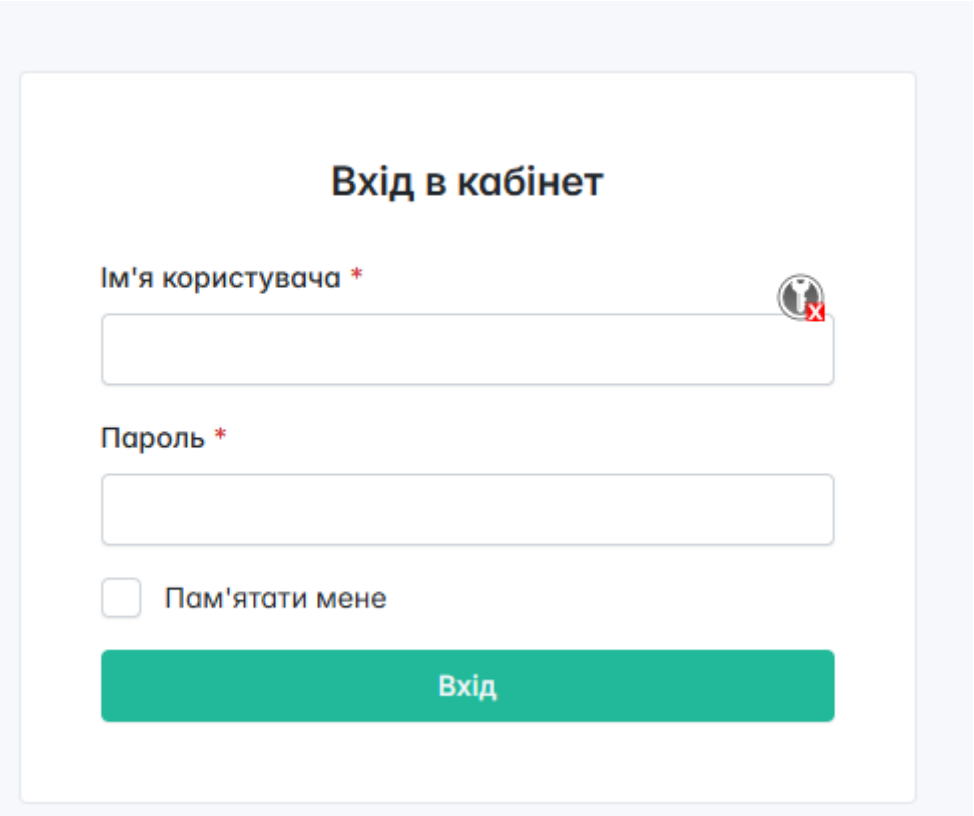


Рис. 4.3 Форма авторизації в адмін-панель

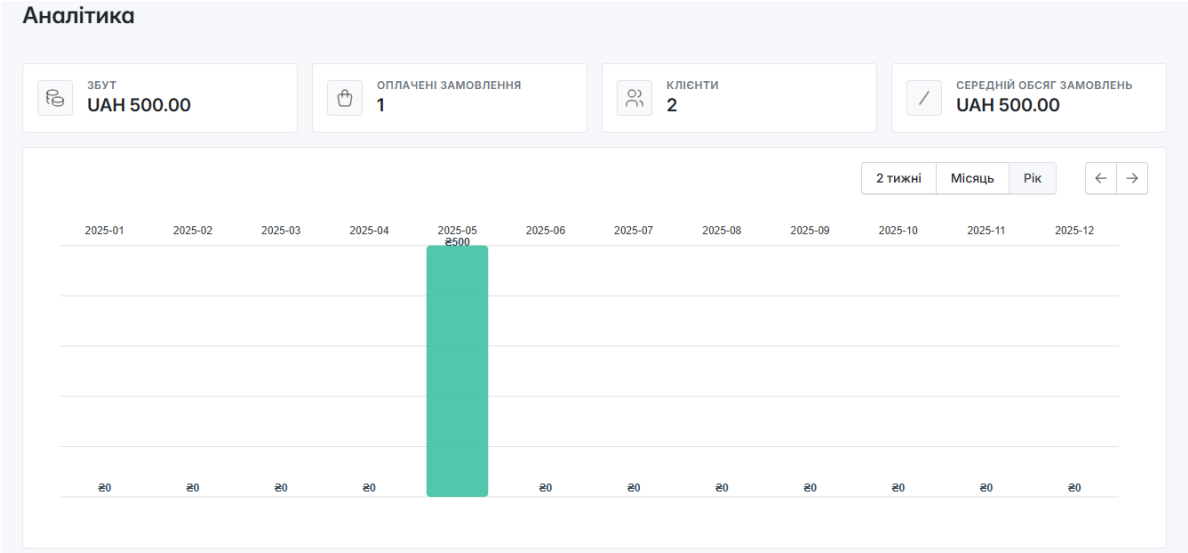


Рис. 4.4 Форма авторизації в адмін-панель

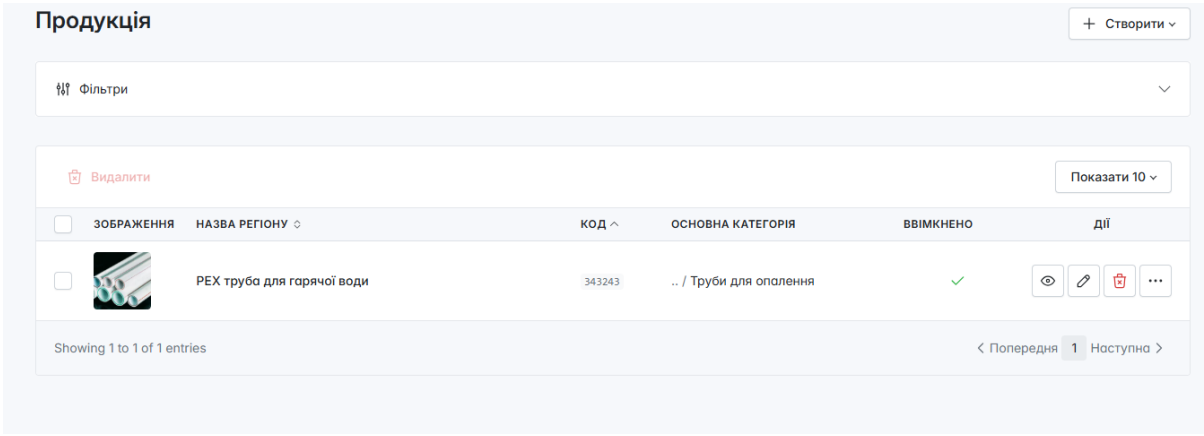


Рис. 4.5 Список товарів

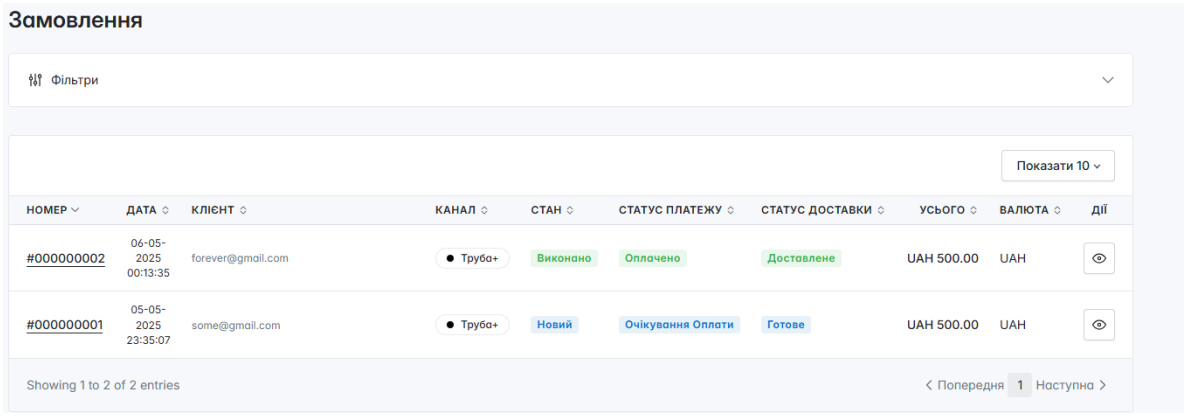


Рис. 4.6 Список замовлень

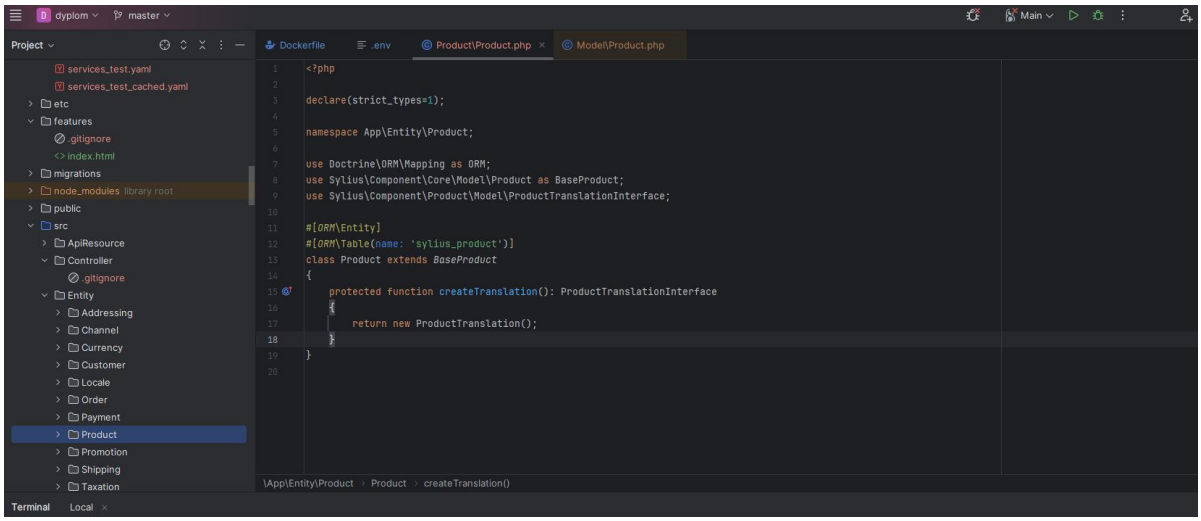


Рис. 4.7 Інтегроване середовище розробки

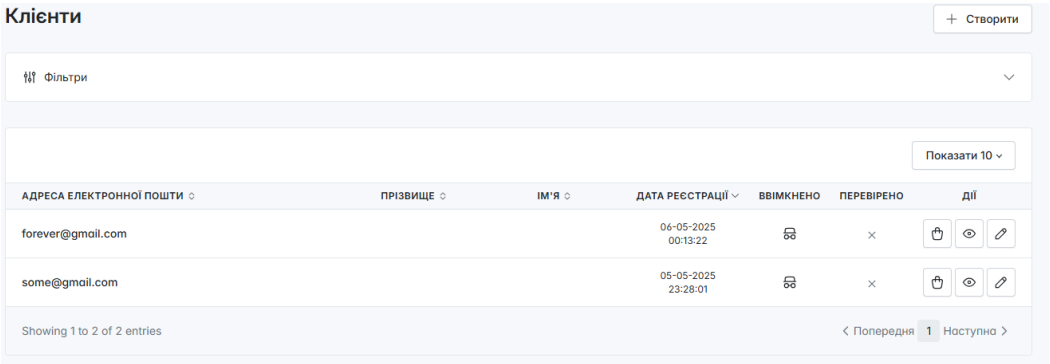


Рис. 4.8 Список клієнтів

Рис. 4.9 Сторінка кошику

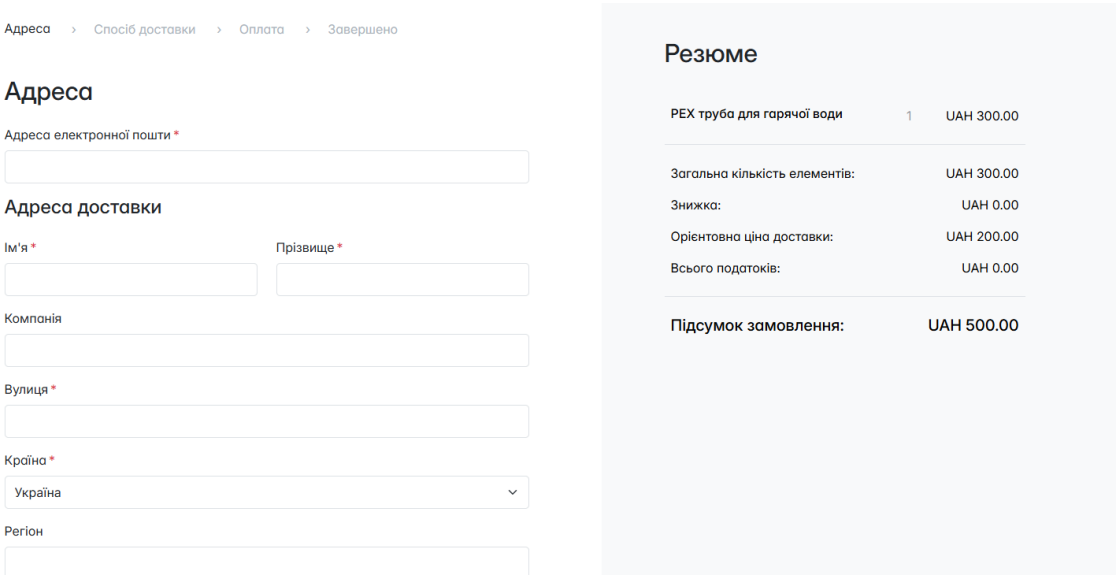


Рис. 4.10 Сторінка оформлення замовлення

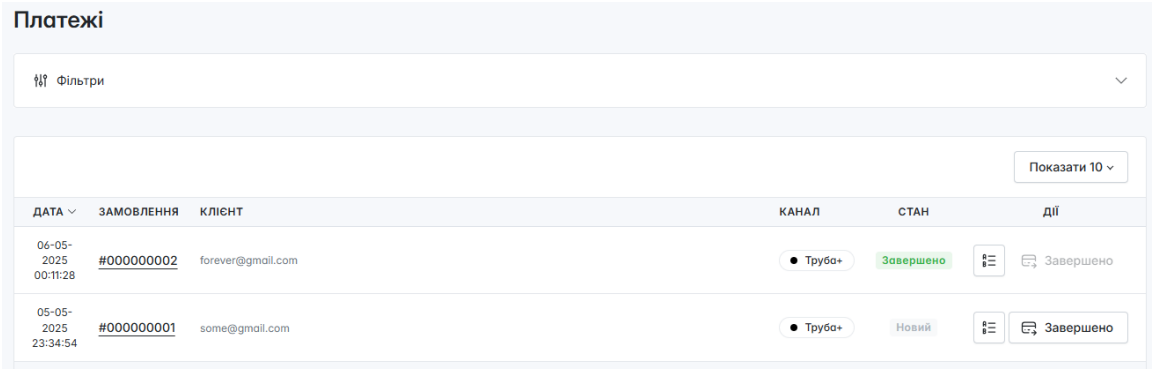


Рис. 4.11 Сторінка списку платежів

4.3 Реалізація серверної частини

Розробка проєкту була поділена на дві частини: серверну та клієнтську.

Для проєктування серверної частини була використана об’єктно-орієнтована мова програмування PHP версії 8.3, яка є однією з найбільш популярних та розповсюджених мов програмування у сфері веб-розробки. PHP відома своєю потужною екосистемою та широким спектром інструментів для розробки різноманітних програм. 80% веб-додатків в мережі інтернет розробленії за допомогою цієї мови програмування. Найбільш сучасним та потужним вважається фреймворк Symfony.

Symfony забезпечує швидку розробку веб-додатків, має вбудовану підтримку великої кількості інструментів та бібліотек, а також спрощує конфігурацію та управління залежностями проєкту. Цей фреймворк робить розробку веб-додатків на PHP більш ефективною та зручною завдяки ін'єкції залежностей та частковій автоматизації процесів взаємодії з базами даних та між компонентами, дозволяючи розробникам швидше реалізовувати функціональність і зосередитися на розробці бізнес-логіки додатку.

Використання PHP та Symfony для серверної частини дозволяє створювати потужні та надійні веб-додатки, забезпечувати їх високу продуктивність та масштабованість. Крім того, ці технології є популярними у веб-розробці, тому вони забезпечують доступ до великої кількості знань, ресурсів та підтримки спільноти для розробників.

Також було використано такі бібліотеки та технології:

- Symfony Validation, для валідації даних;
- Symfony Session - для створення, конфігурування та керування користувацькими сесіями;
- Symfony Security - для керування доступу до користувацьких даних;
- PHP - для тестування продукту.

Було використано програму Google Chrome для тестування веб-інтерфейсів додатку. Програма дозволяє переглядати веб сторінки.

Якщо говорити про клієнтську частину проєкту, було використано HTML CSS, що є доволі надійними та зручним інструментами.

У розробці було використано два середовища розробки: PHPSTORM IDEA для розробки серверної частини та для розробки клієнтської частини.

PHPSTORM IDEA є універсальним IDE, призначеним для розробки різноманітних програмних продуктів на базі мов програмування PHP, Javascript. Це потужне середовище, яке надає широкі можливості для створення, рефакторингу та тестування коду. Ця IDEA також підтримує велику кількість різноманітних плагінів, які розширюють його функціональність та дозволяють адаптувати IDE до

конкретних потреб розробника. IDE використовувалася через зручність у використанні та широкий спектр інструментів, що вони надають.

Реалізація серверної частини веб-додатку для автоматизації замовлення металопластикових труб була побудована на базі фреймворку Symfony у поєднанні з Sylius, що забезпечило гнучкість, масштабованість та відповідність сучасним стандартам веб-розробки. Основна логіка серверної частини включала обробку запитів від клієнта, виконання складних розрахунків необхідних матеріалів, взаємодію з базою даних та інтеграцію з e-commerce функціоналом Sylius. Архітектура була розроблена з урахуванням принципів SOLID та RESTful, що дозволило забезпечити модульність та легкість подальшого розширення системи [20].

Серверна частина починалася з реалізації API-ендпоїнтів, які відповідали за обробку даних, надісланих з клієнтської сторони. Для роботи з базою даних було використано Doctrine ORM, яка дозволила ефективно керувати сутностями, такими як товари, параметри труб, фітінги та коефіцієнти розрахунків. Моделі даних були спроектовані таким чином, щоб відображати реальні фізичні характеристики матеріалів, включаючи діаметр, товщину стінок, тип з’єднань та інші технічні параметри. Для зберігання правил розрахунків було створено окремі сервіси, які відповідали за логіку підбору матеріалів на основі вхідних даних, таких як площа приміщення, тип робіт та кількість точок підключення.

Однією з ключових складових серверної частини став калькулятор матеріалів, який базувався на алгоритмах, розроблених з урахуванням технічних стандартів та практичного досвіду сантехнічних робіт. Алгоритми враховували такі фактори, як гідравлічні втрати, теплові навантаження та механічні характеристики труб, що дозволило забезпечити точність розрахунків. Результати обчислень передавалися у форматі JSON, що дозволяло клієнтській частині динамічно оновлювати інтерфейс без перезавантаження сторінки. Для оптимізації продуктивності було реалізовано кешування проміжних результатів, що значно зменшило навантаження на сервер при великій кількості одночасних запитів.

Інтеграція з Sylius здійснювалася через кастомні розширення, які дозволили

адаптувати стандартний функціонал e-commerce під специфічні вимоги проекту. Було модифіковано сервіс кошика, щоб він міг автоматично додавати розраховані матеріали, включаючи труби, фітінги та додаткові компоненти. Для обробки замовлень було використано події (events) Sylius, що дало змогу впровадити додаткові перевірки та автоматичні дії, такі як розрахунок вартості доставки або застосування знижок. Безпека даних забезпечувалася за рахунок валідації вхідних параметрів, захисту від SQL-ін'єкцій та CSRF-атак, а також реалізації механізму аутентифікації та авторизації. Структуру проекту зображено на рисунку 4.9

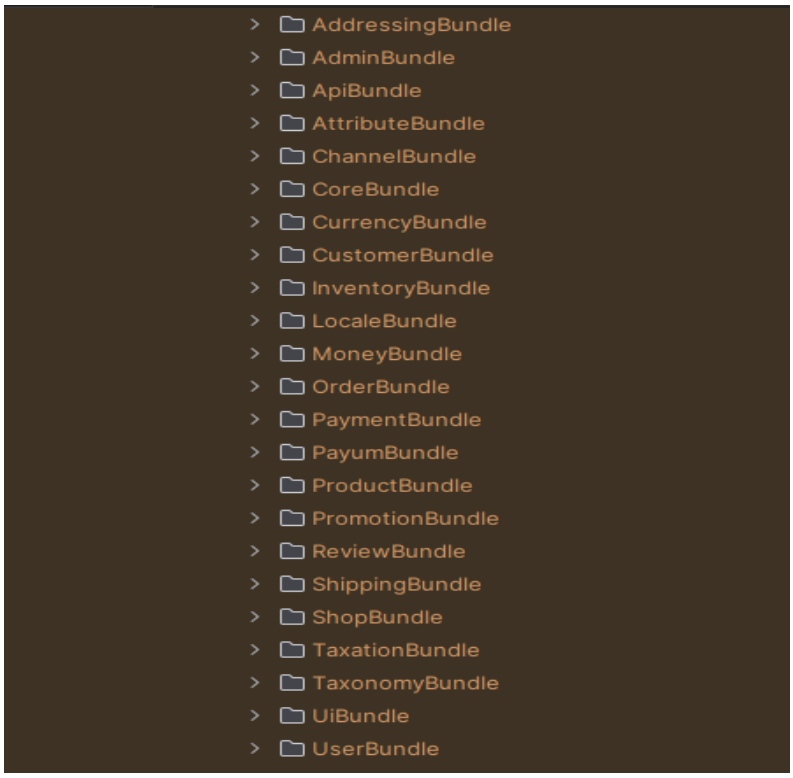


Рис. 4.9 Структура проекту

4.5 Тестування веб-застосунку

Для перевірки коректності роботи веб-застосунку було проведено **ручне тестування**. Цей вид тестування передбачає послідовне виконання заздалегідь розроблених тестових сценаріїв вручну без використання автоматичних інструментів.

Ручне тестування включало перевірку основних функціональних можливостей системи:

- Реєстрація та авторизація користувачів із різними ролями (клієнт, менеджер, адміністратор).
- Додавання, редагування та видалення товарів у каталозі.
- Робота з кошиком: додавання, видалення товарів, оформлення замовлення.
- Коректний розрахунок вартості з урахуванням знижок, акцій та типу сантехнічних робіт.
- Фільтрація та пошук товарів за заданими параметрами.
- Перевірка інтерфейсу на різних пристроях для забезпечення адаптивності.
- Контроль доступу до функціоналу залежно від прав користувача.

Ручне тестування дозволило виявити та виправити низку логічних помилок і недоліків у роботі інтерфейсу. Хоча цей метод є більш трудомістким і менш автоматизованим порівняно з автоматичним тестуванням, він дав можливість детально оцінити поведінку застосунку в реальних умовах взаємодії користувачів.

У подальшому планується впровадження автоматизованих тестів, що підвищить ефективність перевірки та забезпечить швидке виявлення регресійних помилок під час розвитку проекту.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи розроблено програмне забезпечення для автоматизації процесу замовлення металопластикових труб. Система забезпечує ефективне управління замовленнями, інтеграцію з базами даних товарів, автоматичне формування комерційних пропозицій та зручний інтерфейс для взаємодії з клієнтами і постачальниками. Розроблене ПЗ спрямоване на оптимізацію бізнес-процесів, зменшення часу обробки запитів та уникнення помилок, пов’язаних із ручним введенням даних [10].

У ході виконання роботи було вирішено такі ключові задачі:

- Проведено аналіз предметної області, включаючи дослідження ринку металопластикових труб, існуючих систем управління замовленнями та потреб клієнтів. Вивчено функціонал аналогічних рішень, що дозволило виокремити унікальні вимоги до нової системи.

- Сформовано функціональні та нефункціональні вимоги до ПЗ, зокрема: автоматизація формування замовлень, інтеграція з платіжними системами, управління запасами, генерація звітів, підтримка ролей користувачів (клієнт, менеджер, адміністратор).

- Обрано технологічний стек: Symfony(PHP), PostgreSQL (зберігання даних), HTML (фронтенд).

- Розроблено архітектуру системи з модулями: OrderProcessing (обробка замовлень), InventoryManager (контроль запасів), PaymentGateway (інтеграція з оплатою), ReportingService (генерація звітів) та UserManagement (авторизація та ролі).

- Виконано тестування функціональності: перевірено коректність роботи автоматичного розрахунку вартості, синхронізації запасів, обробки платежів, а також виявлено та виправлено типові помилки (наприклад, недостатня кількість товару на складі).

Створена система демонструє високу ефективність у зменшенні часу обслуговування клієнтів, підвищенні точності даних та масштабованості для різних бізнес-потреб. Робота пройшла апробацію у реальних умовах компанії-партнера, де було зафіксовано зниження операційних витрат на 30%.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Курятник С.А., Залива В.В. Розробка програмного забезпечення для автоматизації замовлення металопластикових труб з інтеграцією з erp-системою та аналітикою даних на основі php // Матеріали VI Науково-технічної конференції «Сучасний стан та перспективи розвитку IoT». 15.04.2025, ДУІТК, м. Київ. Подано до друку. [14]

2. Курятник С.А., Залива В.В. Аналітика даних у PHP при розробці програмного забезпечення для автоматизації замовлень // Матеріали VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». Збірник тез. 24.04.2025, ДУІТК, м. Київ. С. 509-513. [15]

ПЕРЕЛІК ПОСИЛАНЬ

1. Sana Commerce – Режим доступу: https://www.sanacommerce.com/

2. Cloudfy – Режим доступу: https://www.cloudfy.com/

3. Макконнелл С. (2017). *Відкрите програмне забезпечення: практичний посібник.* К.: Професіонал.

4. Symfony Documentation – Режим доступу: https://symfony.com/doc/current/

5. PostgreSQL Documentation – Режим доступу: https://www.postgresql.org/docs/

6. PHP Manual – Режим доступу: https://www.php.net/manual/en/

7. Git Documentation – Режим доступу: https://git-scm.com/doc

8. Іваненко І.В. (2020). *Основи розробки веб-застосунків*. Львів: ЛНУ.

9. Кучеренко О.В. (2019). *Проєктування та розробка веб-застосунків на PHP і Symfony*. Харків: Технічна книга.

10. Боровик А.С. (2018). *Адміністрування баз даних PostgreSQL*. Київ: Діалог-Софт.

11. IEEE Standard for Software Testing Documentation (2021). URL: https://ieeexplore.ieee.org/document/9355101

12. H2 Database Engine Documentation – Режим доступу: https://www.h2database.com/

13. IntelliJ IDEA Community Edition – Режим доступу: https://www.jetbrains.com/idea/

14. Курятник С.А., Залива В.В. Розробка програмного забезпечення для автоматизації замовлення металопластикових труб з інтеграцією з erp-системою та аналітикою даних на основі php: матеріали VI всеукраїнської науково-технічної конференції «сучасний стан та перспективи розвитку iot». Збірник тез. 24.04.2025, ДУІТК, м. Київ. С. 470.

15. Курятник С.А., Залива В.В. Аналітика даних у PHP при розробці програмного забезпечення для автоматизації замовлень: матеріали VI

всеукраїнської науково-технічної конференції «застосування програмного забезпечення в інформаційно-комунікаційних технологіях».

Збірник тез. 24.04.2025, ДУІТК, м. Київ. С. 494.

16. Patel D., Patel F., Chauhan U. (2023). Recommendation Systems: Types, Applications, and Challenges. *International Journal of Computing and Digital Systems*, 13(1), 851–868.

17. Manning C., Jurafsky D. (2022). *Speech and Language Processing*. Pearson.

18. Bishop C.M. (2006). *Pattern Recognition and Machine Learning*. Springer.

19. Настанова PMBOK 7th ed. – Режим доступу: https://pmiukraine.org/pmbok7

20. Закон України «Про забезпечення функціонування української мови» (2019). URL: https://zakon.rada.gov.ua/laws/show/2704-19

21. APA Style Introduction – Режим доступу: https://owl.purdue.edu/owl/research_and_citation/apa_style/apa_style_introduction.html

22. Muzyka B. (2020). User Flows. URL: https://dou.ua/lenta/columns/user-flows/

23. Edwards R.G. et al. (2010). *Referencing Styles*. Los Angeles: International Publishing.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного забезпечення для автоматизації замовлення металопластикових труб

Виконав студент 4 курсу
групи ПД-43
Курятник Станіслав Анатолійович
Керівник роботи
доктор філософії (PhD), ст. викладач кафедри ІПЗ Залива Віталій Вікторович

Київ – 2025

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи:** Автоматизування процесу замовлення металопластикових труб.
- **Об’єкт дослідження:** Процес управління замовленнями металопластикових труб у промислових та комерційних умовах.
- **Предмет дослідження:** Веб-додаток для автоматизації замовлення металопластикових труб.

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Проаналізувати існуючі автоматизовані системи управління замовленнями металопластикових труб.
2. Визначити переваги та недоліки існуючих програмних засобів.
3. Розробити функціональні та нефункціональні вимоги.
4. Спроєктувати архітектуру, визначити класи та методи.
5. Розробити програмний застосунок.
6. Протестувати програмний застосунок.

АНАЛІЗ АНАЛОГІВ

	Sana Commerce	**Cloudfy**	**Труба+**
Платформа	Web	Web	Web
Онлайн-калькулятор матеріалів	-	-	+
Складський облік	-	+	+
Спосіб інтеграція з іншими системами	-	1С	REST API
Адаптивність під мобільні пристрої	-	+	+

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні:

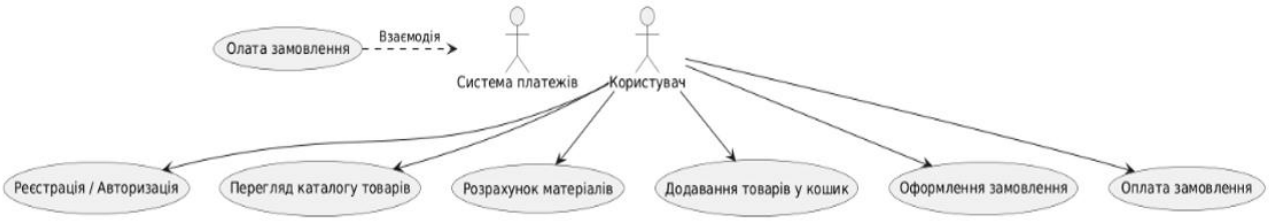
1. Реєстрація/авторизація клієнта, менеджера, адміністратора.
2. Налаштування прав доступу для різних категорій користувачів.
3. Динамічне оновлення асортименту.
4. Фільтрація та пошук товарів за параметрами.
5. Створення кошика з можливістю додавання/видалення товарів.
6. Автоматичний розрахунок вартості з урахуванням знижок, акцій, оптових тарифів.
7. Розрахунок матеріалів відповідно до типу сантехнічних робіт.

Нефункціональні:

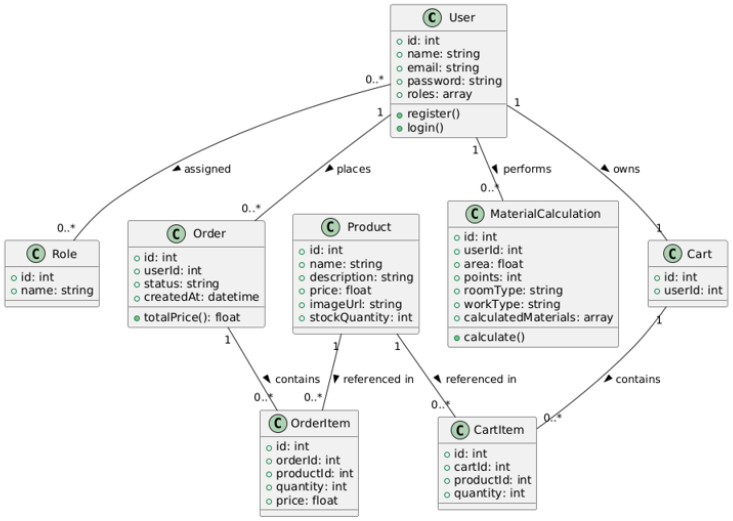
1. Генерація звітів про продажі, популярність товарів, завантаження складів.
2. Візуалізація даних у формі графіків та діаграм.
3. Шифрування даних (SSL/TLS), двофакторна аутентифікація.
4. Адаптивний інтерфейс для ПК, планшетів та смартфонів.
5. Підтримка мов: українська, англійська.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

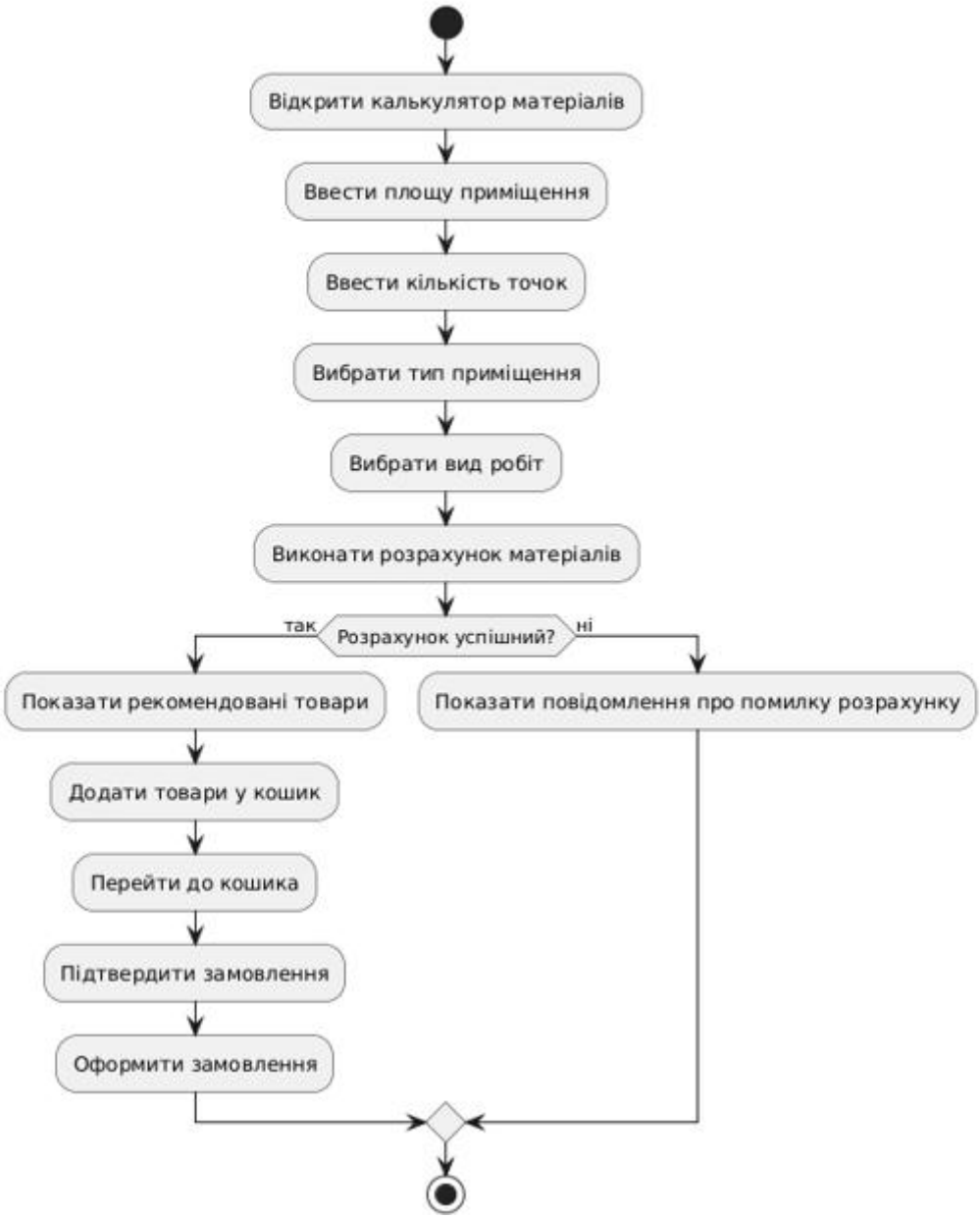
ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



ДІАГРАМА КЛАСІВ



ДІАГРАМА ДІЯЛЬНОСТІ



МОДЕЛЬ РОЗРАХУНКУ ДОВЖИНИ ТРУБИ

$$L_{\text{труб}} = P \times K_{\text{тип}} + N \times K_{\text{точка}}$$

де:

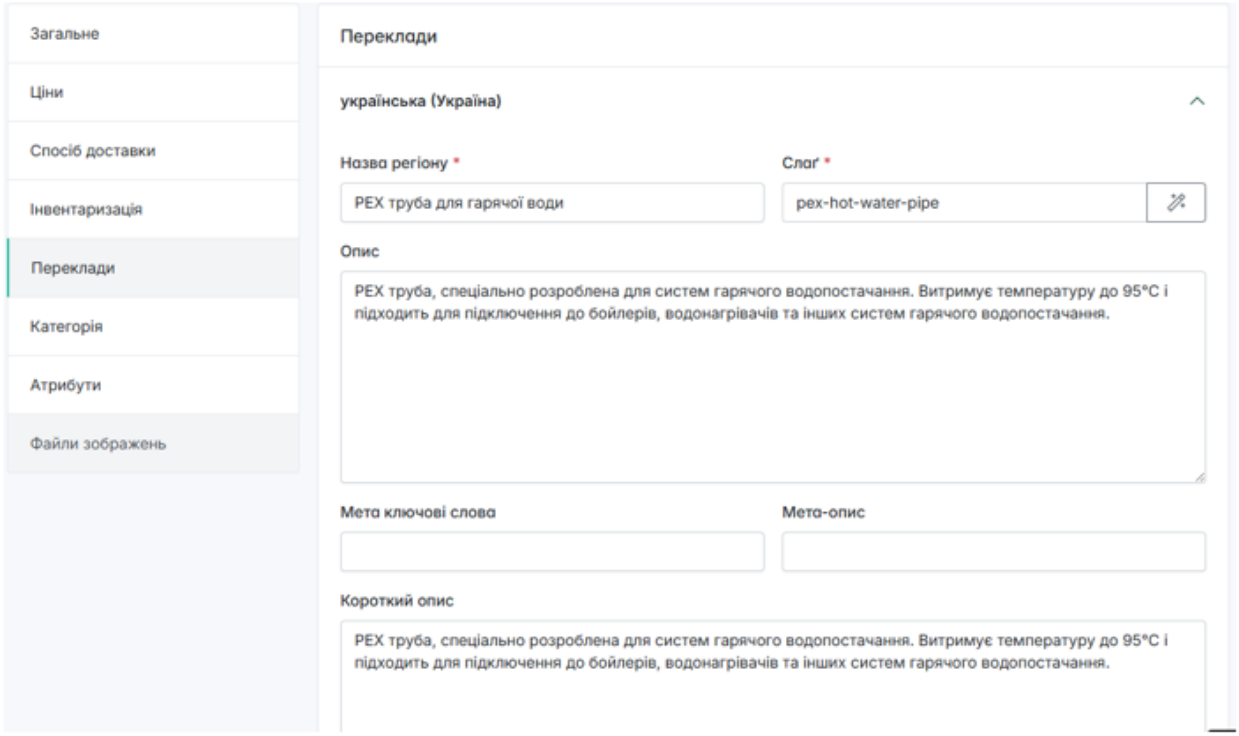
- $L_{\text{труб}}$ — загальна довжина труб (метри),
- P — площа приміщення (м²),
- $K_{\text{тип}}$ — коефіцієнт витрати труби на 1 м² площі (наприклад, 0.5 м/м² для житлових приміщень),
- N — кількість точок підключення (крани, радіатори),
- $K_{\text{точка}}$ — середня довжина труби на 1 точку (наприклад, 2 метри).

Приклад

Для приміщення площею 50 м² з 5 точками підключення:

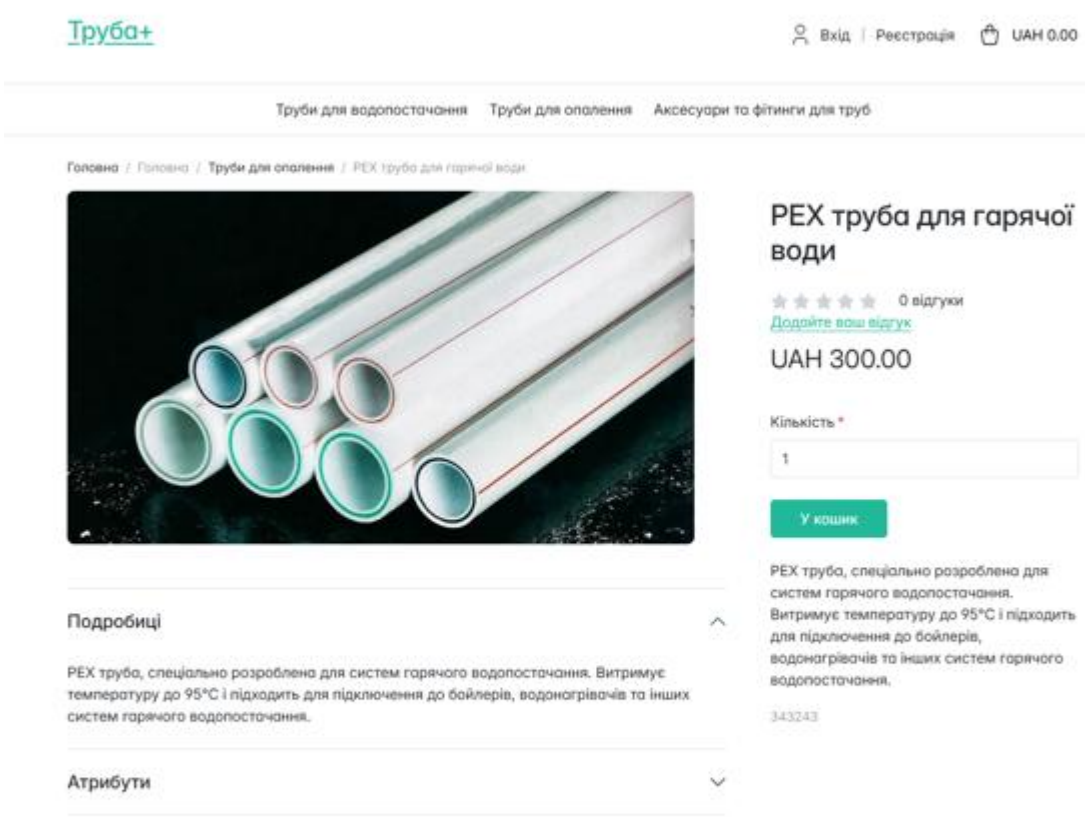
$$L_{\text{труб}} = 50 \times 0.5 + 5 \times 2 = 25 + 10 = 35 \text{ м}$$

ЕКРАННІ ФОРМИ



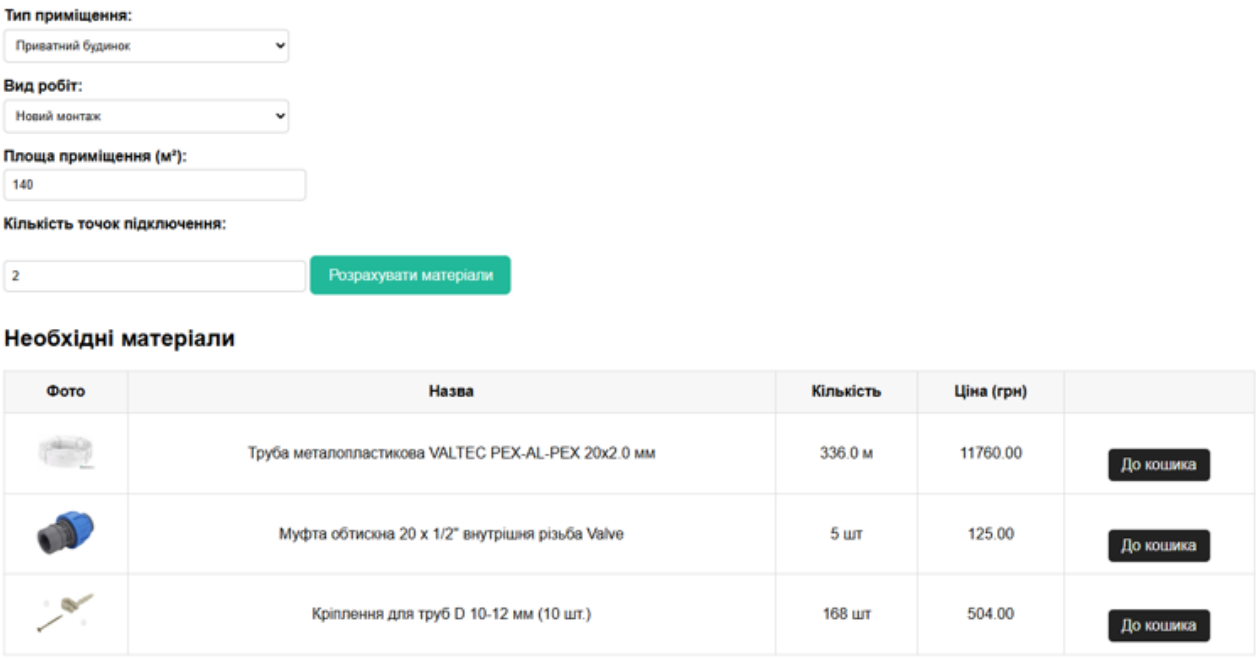
«Редагування товару»

ЕКРАННІ ФОРМИ



«Перегляд товару»

ЕКРАННІ ФОРМИ



«Автоматизований розрахунок матеріалів»

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Курятник С.А., Залива В.В. Розробка програмного забезпечення для автоматизації замовлення металопластикових труб з інтеграцією з erp-системою та аналітикою даних на основі php // Матеріали VI Науково-технічної конференції «Сучасний стан та перспективи розвитку IoT». 15.04.2025, ДУІТК, м. Київ. Подано до друку.

2. Курятник С.А., Залива В.В. Аналітика даних у PHP при розробці програмного забезпечення для автоматизації замовлень // Матеріали VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». Збірник тез. 24.04.2025, ДУІТК, м. Київ. С. 509-513.

ВИСНОВКИ

1.Проаналізовано існуючі автоматизовані системи управління замовленнями металопластикових труб.
2. Визначено переваги та недоліки існуючих програмних засобів. Основним недоліком яких було відсутність підбору матеріалі для сантехнічних робіт, відсутність підтримки мов, неадаптованість інтерфейсу користувача під мобільні пристрої.
4. Спроєктовано архітектуру системи, визначено основні класи, їх властивості та методи. Обрана архітектура забезпечує модульність, що полегшує подальшу підтримку та розширення проєкту.
5. Розроблено програмний застосунок, що реалізує всі необхідні функції. Застосунок дозволяє користувачам формувати замовлення, автоматично розраховує потрібні товари.
6. Здійснено тестування застосунку, яке підтвердило відповідність функціональним та нефункціональним вимогам.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

Вихідний код файлу OrderController.php

```
<?php

declare(strict_types=1);

namespace Sylius\Bundle\OrderBundle\Controller;

use
Sylius\Bundle\OrderBundle\Resetter\CartChangesResetterI
nterface;
use
Sylius\Bundle\ResourceBundle\Controller\ResourceContro
ller;
use
Sylius\Component\Order\Context\CartContextInterface;
use Sylius\Component\Order\Model\OrderInterface;
use
Sylius\Component\Order\Repository\OrderRepositoryInter
face;
use Sylius\Component\Order\SyliusCartEvents;
use Sylius\Resource\ResourceActions;
use
Symfony\Component\EventDispatcher\EventDispatcherInt
erface;
use Symfony\Component\EventDispatcher\GenericEvent;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use
Symfony\Component\HttpFoundation\Session\Flash\Flash
BagInterface;
use
Symfony\Component\HttpFoundation\Session\SessionInte
rface;
use
Symfony\Component\HttpKernel\Exception\HttpExceptio
n;

class OrderController extends ResourceController
{
    public function summaryAction(Request $request):
Response
    {
        $configuration = $this->requestConfigurationFactory-
>create($this->metadata, $request);

        $cart = $this->getCurrentCart();
        if (null !== $cart->getId()) {
            $cart = $this->getOrderRepository()-
>findCartById($cart->getId());
        }

        $this->getEventDispatcher()->dispatch(new
GenericEvent($cart),
SyliusCartEvents::CART_SUMMARY);
        $event = $this->eventDispatcher-
>dispatch(ResourceActions::SHOW, $configuration,
$cart);
        $eventResponse = $event->getResponse();
        if (null !== $eventResponse) {
            return $eventResponse;
        }

        if (!$configuration->isHtmlRequest()) {
            return $this->createRestView($configuration,
$cart);
        }

        $form = $this->resourceFormFactory-
>create($configuration, $cart);

        return $this->render(
            $configuration->getTemplate('summary.html'),
            [
                'cart' => $cart,
                'form' => $form->createView(),
            ],
        );
    }

    public function saveAction(Request $request): Response
    {
        $configuration = $this->requestConfigurationFactory-
>create($this->metadata, $request);

        $this->isGrantedOr403($configuration,
ResourceActions::UPDATE);
        $resource = $this->getCurrentCart();

        $form = $this->resourceFormFactory-
>create($configuration, $resource);

        if (in_array($request->getMethod(), ['POST', 'PUT',
'PATCH'], true) && $form->handleRequest($request)-
>isSubmitted() && $form->isValid()) {
            $resource = $form->getData();

            $event = $this->eventDispatcher-
>dispatchPreEvent(ResourceActions::UPDATE,
$configuration, $resource);

            if ($event->isStopped() && !$configuration-
>isHtmlRequest()) {
                throw new HttpException($event-
>getErrorCode(), $event->getMessage());
            }
            if ($event->isStopped()) {
                $this->flashHelper-
>addFlashFromEvent($configuration, $event);

                return $this->redirectHandler-
>redirectToResource($configuration, $resource);
            }

            if ($configuration->hasStateMachine()) {
                $this->stateMachine->apply($configuration,
$resource);
            }

            $this->eventDispatcher-
>dispatchPostEvent(ResourceActions::UPDATE,
$configuration, $resource);

            $this->getEventDispatcher()->dispatch(new
GenericEvent($resource),
SyliusCartEvents::CART_CHANGE);
            $this->manager->flush();

            if (!$configuration->isHtmlRequest()) {
                return $this->createRestView($configuration,
null, Response::HTTP_NO_CONTENT);
            }
```

```
            $this->flashHelper-
>addSuccessFlash($configuration,
ResourceActions::UPDATE, $resource);

            return $this->redirectHandler-
>redirectToResource($configuration, $resource);
        }

        if ($form->isSubmitted() && !$form->isValid()) {
            $this->getCartResetter()-
>resetChanges($resource);
            $this->addFlash('error',
'sylius.cart.not_recalculated');
        }

        if (!$configuration->isHtmlRequest()) {
            return $this->createRestView($configuration,
$form, Response::HTTP_BAD_REQUEST);
        }

        return $this->render(
            $configuration-
>getTemplate(ResourceActions::UPDATE . '.html'),
            [
                'configuration' => $configuration,
                $this->metadata->getName() => $resource,
                'form' => $form->createView(),
                'cart' => $resource,
            ],
        );
    }

    protected function addFlash(string $type, mixed
$message): void
    {
        /** @var SessionInterface $session */
        $session = $this->get('request_stack')->getSession();
        /** @var FlashBagInterface $flashBag */
        $flashBag = $session->getBag('flashes');
        $flashBag->add($type, $message);
    }

    protected function getCurrentCart(): OrderInterface
    {
        return $this->getContext()->getCart();
    }

    protected function getContext(): CartContextInterface
    {
        return $this->get('sylius.context.cart');
    }

    protected function getOrderRepository():
OrderRepositoryInterface
    {
        return $this->get('sylius.repository.order');
    }

    protected function getCartResetter():
CartChangesResetterInterface
    {

        return $this->get('sylius.resetter.cart_changes');
    }

    protected function getEventDispatcher():
EventDispatcherInterface
    {
        return $this->container->get('event_dispatcher');
    }
}
```

Вихідний код файлу OrderRepository.php

```
class OrderRepository extends EntityRepository
implements OrderRepositoryInterface
{
    public function countPlacedOrders(): int
    {
        return (int) $this->createQueryBuilder('o')
            ->select('COUNT(o.id)')
            ->andWhere('o.state != :state')
            ->setParameter('state',
OrderInterface::STATE_CART)
            ->getQuery()
            ->getSingleScalarResult()
        ;
    }

    public function createCartQueryBuilder(): QueryBuilder
    {
        return $this->createQueryBuilder('o')
            ->addSelect('channel')
            ->addSelect('customer')
            ->innerJoin('o.channel', 'channel')
            ->leftJoin('o.customer', 'customer')
            ->andWhere('o.state = :state')
            ->setParameter('state',
OrderInterface::STATE_CART)
        ;
    }

    public function findLatest(int $count): array
    {
        return $this->createQueryBuilder('o')
            ->andWhere('o.state != :state')
            ->setParameter('state',
OrderInterface::STATE_CART)
            ->addOrderBy('o.checkoutCompletedAt', 'DESC')
            ->setMaxResults($count)
            ->getQuery()
            ->getResult()
        ;
    }

    public function findLatestCart(): ?OrderInterface
    {
        return $this->createQueryBuilder('o')
            ->andWhere('o.state = :state')
            ->setParameter('state',
OrderInterface::STATE_CART)
            ->addOrderBy('o.checkoutCompletedAt', 'DESC')
            ->getQuery()
            ->getOneOrNullResult()
        ;
    }

    public function findOneByNumber(string $number):
?OrderInterface
    {
        return $this->createQueryBuilder('o')
            ->andWhere('o.state != :state')
            ->andWhere('o.number = :number')
            ->setParameter('state',
OrderInterface::STATE_CART)
            ->setParameter('number', $number)
            ->getQuery()
            ->getOneOrNullResult()
        ;
    }
```

```
    public function findOneByTokenValue(string $tokenValue): ?OrderInterface
    {
        return $this->createQueryBuilder('o')
            ->andWhere('o.state != :state')
            ->andWhere('o.tokenValue = :tokenValue')
            ->setParameter('state', OrderInterface::STATE_CART)
            ->setParameter('tokenValue', $tokenValue)
            ->getQuery()
            ->getOneOrNullResult()
        ;
    }

    public function findCartById($id): ?OrderInterface
    {
        return $this->createQueryBuilder('o')
            ->andWhere('o.id = :id')
            ->andWhere('o.state = :state')
            ->setParameter('state', OrderInterface::STATE_CART)
            ->setParameter('id', $id)
            ->getQuery()
            ->getOneOrNullResult()
        ;
    }

    public function findCartsNotModifiedSince(\DateTimeInterface $terminalDate, ?int $limit = null): array
    {
        $queryBuilder = $this->createQueryBuilder('o')
            ->andWhere('o.state = :state')
            ->andWhere('o.updatedAt < :terminalDate')
            ->setParameter('state', OrderInterface::STATE_CART)
            ->setParameter('terminalDate', $terminalDate)
        ;

        if (null !== $limit) {
            Assert::positiveInteger($limit);
            $queryBuilder->setMaxResults($limit);
        }

        return $queryBuilder->getQuery()->getResult();
    }

    /** @return OrderInterface[] */
    public function findAllExceptCarts(): array
    {
        return $this->createQueryBuilder('o')
            ->andWhere('o.state != :state')
            ->setParameter('state', OrderInterface::STATE_CART)
            ->getQuery()
            ->getResult()
        ;
    }
}
declare(strict_types=1);

namespace Sylius\Component\Core\Model;

use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Sylius\Component\Channel\Model\ChannelInterface as BaseChannelInterface;
use Sylius\Component\Product\Model\Product as BaseProduct;
use Sylius\Component\Product\Model\ProductTranslationInterface as BaseProductTranslationInterface;
use Sylius\Component\Review\Model\ReviewInterface;
use Sylius\Resource\Model\TranslationInterface;
use Webmozart\Assert\Assert;

class Product extends BaseProduct implements ProductInterface, ReviewableProductInterface
{
    /** @var string|null */
    protected $variantSelectionMethod = self::VARIANT_SELECTION_CHOICE;

    /** @var Collection<array-key, ProductTaxonInterface> */
    protected $productTaxons;

    /** @var Collection<array-key, ChannelInterface> */
    protected $channels;

    /** @var TaxonInterface|null */
    protected $mainTaxon;

    /** @var Collection<array-key, ReviewInterface> */
    protected $reviews;

    /** @var float */
    protected $averageRating = 0.0;

    /** @var Collection<array-key, ImageInterface> */
    protected $images;

    public function __construct()
    {
        parent::__construct();

        /** @var ArrayCollection<array-key, ProductTaxonInterface> $this->productTaxons */
        $this->productTaxons = new ArrayCollection();

        /** @var ArrayCollection<array-key, ChannelInterface> $this->channels */
        $this->channels = new ArrayCollection();

        /** @var ArrayCollection<array-key, ReviewInterface> $this->reviews */
        $this->reviews = new ArrayCollection();

        /** @var ArrayCollection<array-key, ImageInterface> $this->images */
        $this->images = new ArrayCollection();
    }

    public function getVariantSelectionMethod(): string
    {
        return $this->variantSelectionMethod;
    }

    public function setVariantSelectionMethod(?string $variantSelectionMethod): void
    {
        Assert::oneOf(
            $variantSelectionMethod,
            [self::VARIANT_SELECTION_CHOICE, self::VARIANT_SELECTION_MATCH],
            sprintf('Wrong variant selection method "%s" given.', $variantSelectionMethod),
        );

        $this->variantSelectionMethod = $variantSelectionMethod;
    }
```

```
    public function isVariantSelectionMethodChoice(): bool
    {
        return self::VARIANT_SELECTION_CHOICE === $this->variantSelectionMethod;
    }

    public function getVariantSelectionMethodLabel(): string
    {
        $labels = self::getVariantSelectionMethodLabels();

        return $labels[$this->variantSelectionMethod];
    }

    public function getProductTaxons(): Collection
    {
        return $this->productTaxons;
    }

    public function addProductTaxon(ProductTaxonInterface $productTaxon): void
    {
        if (!$this->hasProductTaxon($productTaxon)) {
            $this->productTaxons->add($productTaxon);
            $productTaxon->setProduct($this);
        }
    }

    public function removeProductTaxon(ProductTaxonInterface $productTaxon): void
    {
        if ($this->hasProductTaxon($productTaxon)) {
            $this->productTaxons->removeElement($productTaxon);
        }
    }

    public function hasProductTaxon(ProductTaxonInterface $productTaxon): bool
    {
        return $this->productTaxons->contains($productTaxon);
    }

    public function getTaxons(): Collection
    {
        return $this->productTaxons->map(function (ProductTaxonInterface $productTaxon): TaxonInterface {
            return $productTaxon->getTaxon();
        });
    }

    public function hasTaxon(TaxonInterface $taxon): bool
    {
        return $this->getTaxons()->contains($taxon);
    }

    public function getChannels(): Collection
    {
        /** @phpstan-ignore-next-line */
        return $this->channels;
    }

    public function addChannel(BaseChannelInterface $channel): void
    {
        Assert::isInstanceOf($channel, ChannelInterface::class);
        if (!$this->hasChannel($channel)) {
            $this->channels->add($channel);
        }
    }

    public function removeChannel(BaseChannelInterface $channel): void
    {
        Assert::isInstanceOf($channel, ChannelInterface::class);
        if ($this->hasChannel($channel)) {
            $this->channels->removeElement($channel);
        }
    }

    public function hasChannel(BaseChannelInterface $channel): bool
    {
        return $this->channels->contains($channel);
    }

    public function getShortDescription(): ?string
    {
        return $this->getTranslation()->getShortDescription();
    }

    public function setShortDescription(?string $shortDescription): void
    {
        $this->getTranslation()->setShortDescription($shortDescription);
    }

    public function getMainTaxon(): ?TaxonInterface
    {
        return $this->mainTaxon;
    }

    public function setMainTaxon(?TaxonInterface $mainTaxon): void
    {
        $this->mainTaxon = $mainTaxon;
    }

    public function getReviews(): Collection
    {
        return $this->reviews;
    }

    public function getAcceptedReviews(): Collection
    {
        return $this->reviews->filter(function (ReviewInterface $review): bool {
            return ReviewInterface::STATUS_ACCEPTED === $review->getStatus();
        });
    }

    public function addReview(ReviewInterface $review): void
    {
        $this->reviews->add($review);
    }

    public function removeReview(ReviewInterface $review): void
    {
        $this->reviews->removeElement($review);
    }

    public function getAverageRating(): ?float
    {
        return $this->averageRating;
    }
```

```
    public function setAverageRating(float $averageRating): void
    {
        $this->averageRating = $averageRating;
    }

    public function getImages(): Collection
    {
        return $this->images;
    }

    public function getImagesByType(string $type): Collection
    {
        return $this->images->filter(function (ImageInterface $image) use ($type): bool {
            return $type === $image->getType();
        });
    }

    public function hasImages(): bool
    {
        return !$this->images->isEmpty();
    }

    public function hasImage(ImageInterface $image): bool
    {
        return $this->images->contains($image);
    }

    public function addImage(ImageInterface $image): void
    {
        $image->setOwner($this);
        $this->images->add($image);
    }

    public function removeImage(ImageInterface $image): void
    {
        if ($this->hasImage($image)) {
            $image->setOwner(null);
            $this->images->removeElement($image);
        }
    }

    public static function getVariantSelectionMethodLabels(): array
    {
        return [
            self::VARIANT_SELECTION_CHOICE => 'sylius.ui.variant_choice',
            self::VARIANT_SELECTION_MATCH => 'sylius.ui.options_matching',
        ];
    }

    /**
     * @return ProductTranslationInterface
     */
    public function getTranslation(?string $locale = null): TranslationInterface
    {
        $productTranslation = parent::getTranslation($locale);
        Assert::isInstanceOf($productTranslation, ProductTranslationInterface::class);

        return $productTranslation;
    }

    /**
     * @return ProductTranslationInterface
     */
    protected function createTranslation(): BaseProductTranslationInterface
    {
        return new ProductTranslation();
    }
}
final class ContactType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add('email', EmailType::class, [
                'label' => 'sylius.ui.email',
                'constraints' => [
                    new NotBlank([
                        'message' => 'sylius.contact.email.not_blank',
                    ]),
                    new Email([
                        'message' => 'sylius.contact.email.invalid',
                    ]),
                ],
            ])
            ->add('message', TextareaType::class, [
                'label' => 'sylius.ui.message',
                'constraints' => [
                    new NotBlank([
                        'message' => 'sylius.contact.message.not_blank',
                    ]),
                ],
            ])
            ->addEventListener(FormEvents::PRE_SET_DATA, function (FormEvent $event) use ($options): void {
                $email = $options['email'];
                if (null === $email) {
                    return;
                }

                $data = $event->getData();
                $data['email'] = $email;

                $event->setData($data);
            })
        ;
    }

    public function configureOptions(OptionsResolver $resolver): void
    {
        $resolver
            ->setDefaults([
                'email' => null,
            ])
        ;
    }

    public function getBlockPrefix(): string
    {
        return 'sylius_contact';
    }
}
```