БАКАЛАВРСЬКА РОБОТА

БР.КІ-37.00.00.000 ПЗ

Група КІ-22-2

Ковальчук Роман

2026

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу
Факультет інформаційних технологій
Кафедра комп’ютерних систем і мереж

Ковальчук Роман Тарасович

УДК 004.66

БАКАЛАВРСЬКА РОБОТА

Розробка інформаційного web-сайту ТОВ «Мегаватсервіс» засобами JavaScript та Python 3

Комп’ютерна інженерія
(назва освітньої програми)

123 - Комп’ютерна інженерія
(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня _____ *Ковальчук Р. Т.* _____
(підпис, ініціали та прізвище здобувача)

Науковий керівник _____ *Бабчук С. М., доц., к. т. н.* _____
(підпис, прізвище, ім’я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри КСМ

д.т.н.,професор _____ *С.І. Мельничук*
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2026 рік

Івано-Франківський національний технічний університет нафти і газу

(повне найменування вищого навчального закладу)

Факультет *інформаційних технологій*

Кафедра *комп'ютерних систем і мереж*

Освітній ступінь *бакалавр*

Спеціальність *123 – Комп'ютерна інженерія*

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри КСМ

(С.І. Мельничук)

«*21*» *квітня* *2026* року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Ковальчуку Роману Тарасовичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи*)* "*Розробка інформаційного web-сайту ТОВ «Мегаватсервіс» засобами JavaScript та Python 3"*

керівник проекту (роботи) *Бабчук С. М., доц., к. т. н.*

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від *21.04.2026 № 180/7*

2. Строк подання студентом роботи *11 червня 2026 р*

3. Вихідні дані до роботи *Матеріали і результати отримані під час проходження переддипломної практики, методичні вказівки, технічна література.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *1. Аналіз предметної області та постановка завдання. 2. Архітектурне проєктування інформаційного web-сайту. 3. Програмна реалізація інформаційного web-сайту.*

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

6. Консультанти розділів роботи

Розділ	Консультант	Підпис, дата

7. Дата видачі завдання *02 лютого 2026 р.*

№ з/п	Назва етапів дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	*Аналіз предметної області та постановка завдання*	*Лютий, 2026*	
2	*Архітектурне проєктування інформаційного web-сайту*	*Березень, 2026*	
3	*Програмна реалізація інформаційного web-сайту*	*Травень, 2026*	
4	*Оформлення пояснювальної записки*	*Червень, 2026*	

Студент _____ *Ковальчук Р. Т.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Бабчук С. М..*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

В дипломній роботі було здійснено розробку інформаційного B2B web-сайту для інжинірингової компанії ТОВ «Мегаватсервіс» на основі Python, FastAPI, JavaScript, PostgreSQL та Docker.

Метою роботи є створення сучасного, надійного та адаптивного корпоративного порталу для цифровізації взаємодії з корпоративними клієнтами шляхом інтеграції модулів перевірки підрядника (Блоку довіри), електронного інженерного каталогу та системи захищеної обробки технічних завдань.

В роботі проведено комплексний аналіз предметної області, нормативно-правової бази України щодо захисту інформації та порівняльний аналіз сучасних архітектурних рішень. Розглянуто специфіку ведення інжинірингового бізнесу, тендерної діяльності та потребу впровадження ізольованого програмного рішення для обробки корпоративних заявок. На основі аналізу сформульовано завдання проєкту й запропоновано розробку системи з розділеною архітектурою. Розроблено трирівневу інфраструктуру на основі Python, FastAPI, PostgreSQL та Docker Compose. Спроєктовано об'єктно-реляційну базу даних, алгоритми асинхронної потокової обробки багатомегабайтних інженерних креслень (Multipart), безпечного збереження даних та адаптивний веб-інтерфейс за допомогою Tailwind CSS. Проведено навантажувальне тестування швидкодії, транзакційної цілісності та перевірку захищеності системи від XSS та SQL-вразливостей.

В результаті створено ефективну, безпечну та масштабовану систему автоматизації B2B-взаємодії з інтуїтивно зрозумілим інтерфейсом для прийому комерційних запитів, швидкого моніторингу ліцензійної документації та управління електронним каталогом підприємства.

Ключові слова: B2B web-сайт, інжиніринговий портал, обробка технічних завдань, Python, FastAPI, PostgreSQL, Tailwind CSS, Docker, асинхронна архітектура.

SUMMARY

In the diploma thesis, the development of an informational B2B web site for the engineering company "Megawattservice" LLC was carried out based on Python, FastAPI, JavaScript, PostgreSQL, and Docker.

The aim of the work is to create a modern, reliable, and responsive corporate portal for the digitalization of interaction with corporate clients by integrating modules for contractor verification (Trust Block), an electronic engineering catalog, and a secure technical task processing system.

A comprehensive analysis of the subject area, the regulatory framework of Ukraine regarding information protection, and a comparative analysis of modern architectural solutions was conducted in the work. The specifics of the engineering business, tender activities, and the need to implement an isolated software solution for processing corporate requests were considered. Based on the analysis, the project tasks were formulated, and the development of a system with a decoupled architecture was proposed. A three-tier infrastructure based on Python, FastAPI, PostgreSQL, and Docker Compose was developed. An object-relational database, algorithms for asynchronous streaming processing of multi-megabyte engineering drawings (Multipart), secure data storage, and an adaptive web interface using Tailwind CSS were designed. Load testing of performance, transactional integrity, and verification of the system's security against XSS and SQL vulnerabilities were conducted.

As a result, an effective, secure, and scalable system for automating B2B interaction with an intuitive interface was created to receive commercial requests, quickly monitor licensing documentation, and manage the enterprise's electronic catalog.

Keywords: B2B web site, engineering portal, technical task processing, Python, FastAPI, PostgreSQL, Tailwind CSS, Docker, asynchronous architecture.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасний етап розвитку інформаційних технологій характеризується тотальним переходом бізнес-процесів у цифрове середовище (Web 3.0, хмарні технології, цифровізація). Для підприємств енергетичного сектору та оптової торгівлі інженерним обладнанням наявність сучасного, високопродуктивного та безпечного вебпредставництва є не просто елементом іміджу, а критично важливою інфраструктурною необхідністю. Товариство з обмеженою відповідальністю «Мегаватсервіс», що базується у м. Івано-Франківськ та спеціалізується на електромонтажних роботах і постачанні електрообладнання, працює переважно у сегменті B2B (Business-to-Business) та з державними замовниками (B2G). Специфіка цього сегменту вимагає відмови від стандартних сайтів-візиток або класичних роздрібних інтернет-магазинів на користь складних інформаційних B2B web-сайтів (Corporate Web Portals), здатних обробляти технічні завдання (ТЗ), надавати доступ до тендерної документації та генерувати якісні B2B-ліди [1].

Традиційні підходи до розробки корпоративних сайтів, засновані на монолітних системах управління контентом (CMS, таких як WordPress чи Joomla), демонструють свою технічну обмеженість при спробі інтеграції специфічних бізнес-вимог. Вони мають надлишкову архітектуру, страждають від вразливостей сторонніх плагінів та не здатні забезпечити адекватну швидкість реакції інтерфейсу [2]. Для вирішення інженерних завдань ТОВ «Мегаватсервіс» необхідно застосувати сучасні підходи комп'ютерної інженерії: розділення моноліту на незалежний Frontend (на базі JavaScript) та Backend-ядро (на базі Python 3), використання асинхронних протоколів обміну даними (REST API) та впровадження систем контейнеризації для забезпечення незалежності від серверного оточення [3].

Таким чином, розробка спеціалізованого інформаційного web-сайту з урахуванням B2B-специфіки засобами сучасних мов програмування JavaScript та Python є надзвичайно актуальним інженерно-технічним завданням.

Об'єкт дослідження – процес проєктування, програмної розробки, тестування та розгортання інформаційного web-сайту з клієнт-серверною архітектурою для потреб B2B підприємства.

Предмет дослідження – моделі вебпроєктування, архітектурні патерни REST, мовні конструкції JavaScript (ES6+) та Python 3 (FastAPI), алгоритми асинхронної обробки HTTP-запитів, а також методи контейнеризації (Docker), що застосовуються для створення корпоративного web-сайту.

Мета роботи – розробка повнофункціонального, надійного та адаптивного інформаційного web-сайту для ТОВ «Мегаватсервіс», який забезпечить цифровізацію взаємодії з корпоративними клієнтами шляхом інтеграції модулів перевірки підрядника (довіри), електронного каталогу та системи обробки технічних завдань.

Для досягнення поставленої мети необхідно вирішити такі завдання:

− Провести системний аналіз вимог до веброзробки B2B-порталів, врахувавши специфіку діяльності ТОВ «Мегаватсервіс» (м. Івано-Франківськ) та вимоги державних замовників (тендерні процедури, Prozorro).

− Проаналізувати сучасні технологічні стеки веброзробки (монолітні vs мікросервісні архітектури) та обґрунтувати вибір платформ Python 3, FastAPI, JavaScript, Tailwind CSS та PostgreSQL.

− Здійснити архітектурне та мережеве проєктування інформаційного web-сайту: розробити UML-діаграми взаємодії, спроєктувати реляційну структуру бази даних та специфікацію REST API маршрутів.

− Розробити серверну частину (Backend): реалізувати об'єктно-реляційне мапування (ORM), модулі асинхронного збереження файлів технічних завдань та обробки B2B-заявок.

− Створити клієнтську частину (Frontend): розробити реактивний кросплатформний інтерфейс користувача з блоками "Документи/Ліцензії", електронним каталогом обладнання та багатоканальною контактною книгою.

− Інкапсулювати всі програмні компоненти web-сайту (БД та вебсервер) в ізольовані Docker-контейнери для подальшого розгортання (CI/CD).

− Виконати навантажувальне тестування вебсервера та верифікацію клієнтських скриптів на предмет безпеки (захист від XSS-ін'єкцій).

Методи дослідження. У роботі використано: методи інженерії програмного забезпечення (Software Engineering) – для декомпозиції життєвого циклу розробки сайту; методи теорії реляційних баз даних – для проєктування нормалізованої схеми сховища (3НФ); об'єктно-орієнтований підхід та асинхронне програмування – для розробки серверного ядра; методи реактивного вебдизайну – для реалізації UI/UX.

Практичне значення отриманих результатів. Розроблений інформаційний web-сайт є готовим програмним продуктом (production-ready). Створена програмна архітектура дозволяє ТОВ «Мегаватсервіс» репрезентувати свою експертність, публікувати ліцензії для участі у тендерах, приймати складні замовлення з кресленнями від корпоративних клієнтів та керувати електронним каталогом обладнання. Контейнеризована архітектура коду забезпечує легкість масштабування та міграції системи між серверами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Детальний аналіз предметної області та поточних бізнес-процесів компанії

Розробка web-сайтів для сегменту B2B (Business-to-Business) фундаментально відрізняється від класичної B2C-веброзробки (Business-to-Consumer, наприклад, роздрібні інтернет-магазини). У комп'ютерній інженерії та вебаналітиці ці відмінності диктують абсолютно різні підходи до проєктування архітектури баз даних (Database Schema), формування користувацького досвіду (UX - User Experience) та конструювання інтерфейсу (UI - User Interface) [4].

ТОВ «Мегаватсервіс» — це інжинірингова компанія, що фізично та юридично базується у м. Івано-Франківськ (зокрема, об'єкти та офіси зосереджені на вул. Галицькій та вул. Княгинин). Основні види діяльності: електромонтаж високої складності, проєктування мереж, а також оптова торгівля телекомунікаційним, комп'ютерним та електрощитовим обладнанням. Цільовою аудиторією web-сайту є не пересічні споживачі, а інженери, головні енергетики підприємств, представники тендерних комітетів та бухгалтери компаній-контрагентів.

З інженерної точки зору, класичний підхід "Створити каталог -> Додати кнопку 'В кошик' -> Оплатити карткою" тут не працює. Сервери чи трансформаторні підстанції не купуються через вебкошик. Бізнес-цикл у цій сфері (Sales Pipeline) триває тижнями або місяцями, і інформаційний web-сайт має виступати не як точка миттєвого продажу, а як високотехнологічний шлюз для збору кваліфікованих лідів (Lead Generation) та обміну документацією [5].

Тому розроблюваний web-сайт повинен реалізовувати такі програмні концепції:

1. Інформаційна персистентність замість транзакційності. Замість інтеграції банківських шлюзів (еквайрингу) на сервері, необхідно реалізувати надійні POST-маршрути (API Endpoints) для прийому складних форм із вкладеними файлами (наприклад, завантаження замовником плану приміщення у форматі PDF чи AutoCAD).

2. Абстракція каталогу. Електронний каталог повинен мати ієрархічну структуру бази даних (дерево категорій), але замість ціни та кнопки "Купити" система має динамічно генерувати форми "Отримати прайс-лист" або "Запитати специфікацію".

3. Генерація та відображення статичних файлів. Сервер повинен мати окремий підмодуль для безпечної роздачі (Static File Serving) сканованих копій ліцензій, допусків та витягів з державних реєстрів, які потрібні для тендерів.

На основі аналізу B2B-середовища сформовано чіткі інженерні вимоги до модулів, які мають бути закодовані та впроваджені в архітектуру інформаційного web-сайту ТОВ «Мегаватсервіс» [6].

Модуль 1: Блок довіри та тендерних підтверджень (Trust & Compliance Module).

Державні замовники (B2G) та великий бізнес проводять процедуру Due Diligence (перевірку підрядника) перед початком співпраці.

− Функціонал: Програмна реалізація розділу "Документи та ліцензії". На бекенді (Backend) необхідно спроєктувати механізм зберігання посилань на PDF-документи (допуски до робіт з високою напругою, сертифікати ISO). На фронтенді (Frontend) – розробити UI-компоненти для їх зручного перегляду.

− Інтеграція з Prozorro: Інтерфейс має містити спеціально стилізований блок або віджет, що демонструє успішні кейси на платформі державних закупівель (кількість виграних тендерів, посилання на завершені договори).

− Партнерська мережа: Динамічна JavaScript-карусель (Slider Component), що зчитує з бази даних логотипи діючих клієнтів компанії та безперервно їх прокручує, не перевантажуючи DOM-дерево браузера.

Модуль 2: Електронний каталог обладнання (B2B Catalog).

− Функціонал: Розробка реляційної структури категорій (напр., "Електрощитове обладнання", "Комп'ютерні мережі").

− Механіка взаємодії: Повна відсутність клієнтського кошика (Session Cart). Натомість реалізується інтерактивна форма "Отримати прайс/Специфікацію". Клієнт вводить свій код ЄДРПОУ, контактні дані, і скрипт асинхронно відправляє ці дані на сервер для запису в таблицю лідів (B2B Leads).

− Data Sheets: Можливість завантаження технічних специфікацій (PDF) безпосередньо зі сторінки товару.

Модуль 3: Розділ інжинірингових послуг (Services & Portfolio).

− Функціонал: Динамічна генерація посадкових сторінок (Landing Pages) для кожного типу робіт (монтаж, аудит).

− Форма виклику з File API: Найскладніший елемент Frontend-розробки. Форма повинна підтримувати об'єкт HTML5 FormData та input type="file", дозволяючи клієнту прикріпити ТЗ (технічне завдання) або фотографії об'єкта. Сервер має асинхронно приймати ці бінарні дані (Multipart/form-data) та безпечно зберігати їх на файловій системі.

− Портфоліо: Галерея "До/Після" реалізованих проєктів в Івано-Франківській області та за її межами.

Модуль 4: Багатоканальна контактна маршрутизація.

Функціонал: Складна сторінка контактів. Оскільки компанія має офіс (вул. Галицька) та інші локації (вул. Княгинин), необхідно інтегрувати картографічний API (наприклад, Google Maps або OpenStreetMap iframe). Також публікуються повні бухгалтерські реквізити підприємства для контрагентів. Розробка "плаваючих" (Sticky) кнопок швидкого переходу у месенджери (Telegram, Viber) засобами CSS.

1.2 Нормативно-правові вимоги до захисту web-сайтів в Україні

Розробка корпоративного програмного забезпечення в Україні жорстко регламентується законодавством у сфері кібербезпеки та захисту інформації [7]. Оскільки інформаційний web-сайт ТОВ «Мегаватсервіс» буде збирати комерційні запити, технічні плани приміщень підприємств та номери телефонів керівників, він підпадає під дію Закону України «Про захист персональних даних» та Закону «Про захист інформації в інформаційно-комунікаційних системах» [8].

− З погляду інженерії безпеки (Web Security), це вимагає імплементації кількох ешелонів захисту на рівні коду:

− Захист від SQL-ін'єкцій (SQLi): Відмова від конкатенації "сирих" SQL-рядків мовою Python. Всі звернення до бази даних мають відбуватися виключно через абстракцію ORM (Object-Relational Mapping), яка автоматично екранує всі вхідні дані від користувачів [9].

− Захист від міжсайтового скриптингу (XSS): Усі текстові поля у формах "Запит ціни" або "Виклик інженера" повинні проходити санітизацію (Sanitization). Шаблонізатор серверної частини повинен перетворювати небезпечні символи (наприклад, <script>) на безпечні HTML-сутності (HTML Entities).

− Захист файлових завантажень: Оскільки Модуль 3 дозволяє клієнтам завантажувати файли ТЗ, бекенд повинен суворо перевіряти MIME-типи файлів (дозволяти лише .pdf, .docx, .png, .jpg) та ігнорувати виконувані файли (.exe, .php, .sh), запобігаючи атакам типу Remote Code Execution (RCE) [10].

1.3 Порівняльний аналіз архітектур та сучасних web-технологій

Перед початком етапу програмування критично важливо обґрунтувати вибір архітектурного підходу та технологічного стеку (Technology Stack). Сучасна комп'ютерна інженерія пропонує два фундаментальні підходи до

побудови web-платформ. Перший варіант — це монолітні CMS, такі як WordPress, Joomla або OpenCart. Це системи, написані переважно мовою PHP, у яких база даних, бекенд і фронтенд тісно зчеплені в єдиний моноліт [11]. Оскільки WordPress орієнтований на B2C-блоги, реалізація складних B2B-форм із завантаженням технічних завдань, створення спеціальних каталогів без кошика та інтеграція з Prozorro вимагатиме встановлення десятків "костильних" плагінів. Це неминуче призведе до катастрофічного уповільнення часу відповіді сервера (Time to First Byte - TTFB), створить масу вразливостей і зробить рефакторинг такого коду практично неможливим.

Другий підхід — це мікрофреймворки та розділена архітектура (Custom Development). Ця парадигма передбачає написання власного серверного ядра мовами Python, Node.js або Go, а також створення клієнтського інтерфейсу з нуля. Хоча це вимагає вищої кваліфікації інженера, такий шлях гарантує абсолютний контроль над кодом (No Vendor Lock-in), максимальну швидкодію та високий рівень безпеки [12]. Саме цей підхід було обрано для розробки інформаційного web-сайту ТОВ «Мегаватсервіс».

У межах підходу Custom Development було проведено глибоке порівняння серверних мов, зокрема Node.js (JavaScript) та Python 3. Незважаючи на те, що Node.js є вкрай популярним для асинхронних вебпроєктів, вибір було зроблено на користь Python 3 у зв'язці з сучасним фреймворком FastAPI, оскільки це рішення демонструє беззаперечні переваги в контексті інженерних систем.

З погляду продуктивності, FastAPI базується на стандарті ASGI (Asynchronous Server Gateway Interface) та рушії uvicorn, використовуючи асинхронний цикл подій (Event Loop) asyncio. За показниками швидкодії FastAPI значно перевершує Node.js і впритул наближається до мови Go [13].

Щодо типізації та валідації, Python 3 нативно підтримує анотації типів (Type Hints), а фреймворк FastAPI глибоко інтегрований з бібліотекою Pydantic для валідації даних. Якщо клієнт надішле у формі B2B-заявки невалідний код ЄДРПОУ або некоректний email, Pydantic автоматично відхилить такий запит із

помилкою HTTP 422 (Unprocessable Entity) ще до того, як він дійде до бази даних. Для порівняння, в екосистемі Node.js/Express подібну логіку перевірок довелося б писати повністю вручну [14].

Під час формування технологічного стеку Frontend-частини було вирішено відмовитися від важких фреймворків Single Page Application (SPA), таких як React або Angular. Вони мають тенденцію перевантажувати браузер замовника завантаженням багатомегабайтних JavaScript-бандлів. Натомість було обрано концепцію Server-Side Rendering (SSR) з використанням серверного шаблонізатора Jinja2 та Vanilla JavaScript. Це інженерне рішення забезпечує миттєве відображення сторінок, що є критично важливим для SEO-оптимізації та швидкості прийняття рішень корпоративними B2B-клієнтами [15]. Для стилізації користувацького інтерфейсу (UI) впроваджено утилітарний CSS-фреймворк Tailwind CSS. Він дозволяє верстати адаптивні компоненти за принципом Mobile-first безпосередньо в HTML-розмітці, ефективно уникаючи розростання файлів стилів та конфліктів селекторів [16].

1.4 Порівняльний аналіз існуючих аналогів та способів архітектурної реалізації web-сайту

Процес проєктування та подальшої розробки інформаційного web-сайту для підприємства інжинірингового спрямування, такого як ТОВ «Мегаватсервіс», вимагає проведення фундаментального системного аналізу вже існуючих ринкових рішень. Таке дослідження дозволяє не лише виявити усталені стандарти в галузі B2B-комунікацій, а й сформувати унікальну ціннісну пропозицію, уникаючи типових помилок конкурентів. У межах даного дослідження було проведено детальний огляд вебресурсів провідних компаній інженерного сектору України, що дозволило класифікувати їх за функціональною насиченістю, архітектурною стійкістю та рівнем орієнтації на тендерну діяльність.

Першим об'єктом аналізу став вебресурс компанії DEPS (deps.ua), яка є одним із найбільших системних інтеграторів телекомунікаційного та енергетичного обладнання в Україні (рис. 1.1).

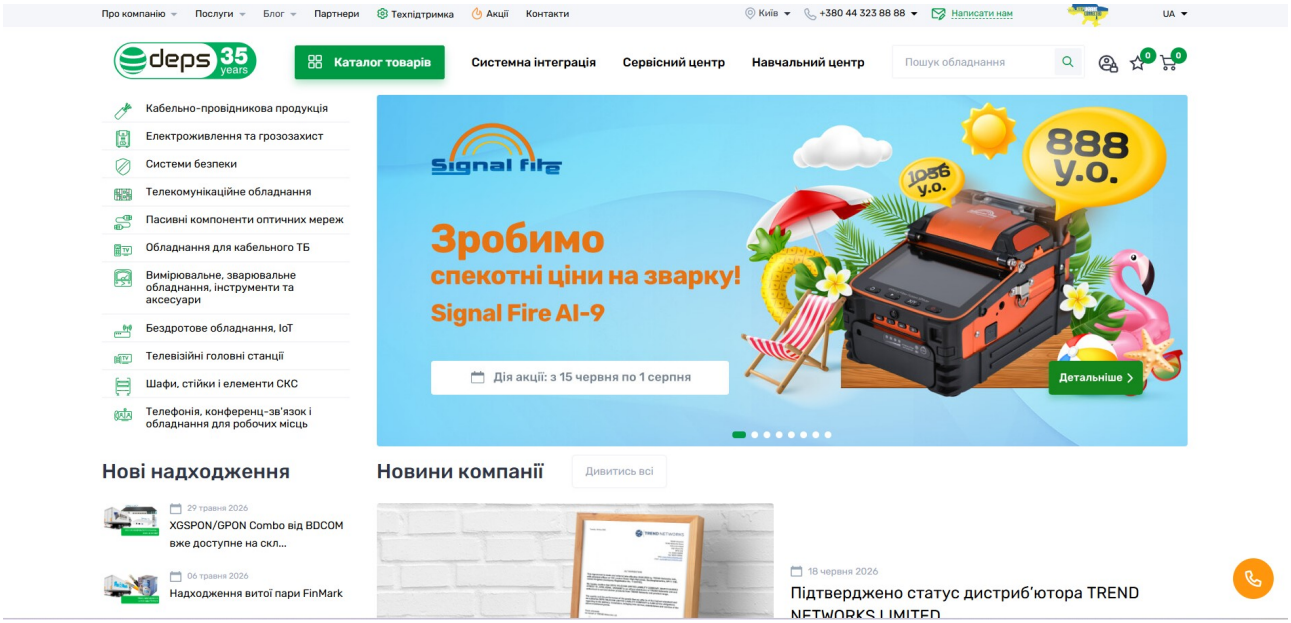

Рисунок 1.1 – Вебресурс компанії DEPS

Даний сайт представляє собою еталонний зразок ієрархічної структури електронного каталогу для B2B-сегменту. Глибокий аудит структури DEPS дозволяє констатувати, що ресурс орієнтований на професійну аудиторію інженерів та закупівельників. Основним елементом взаємодії тут виступає не роздрібний продаж, а надання вичерпної технічної інформації. Проте, при всій потужності каталогу, інтерфейс сайту виглядає надмірно перевантаженим, що створює значне когнітивне навантаження на користувача, який має на меті здійснити швидкий виклик бригади для аварійного ремонту. Архітектурно сайт демонструє високу швидкість обробки текстових запитів, проте відчувається нестача асинхронних рішень у модулях зворотного зв'язку, що змушує користувача очікувати повного перезавантаження сторінок при кожній взаємодії з фільтрами.

Наступним аналогом було обрано сайт компанії EDS Engineering (eds-ltd.com.ua). Цей ресурс демонструє кардинально інший підхід, орієнтований на продаж послуг генерального підряду.

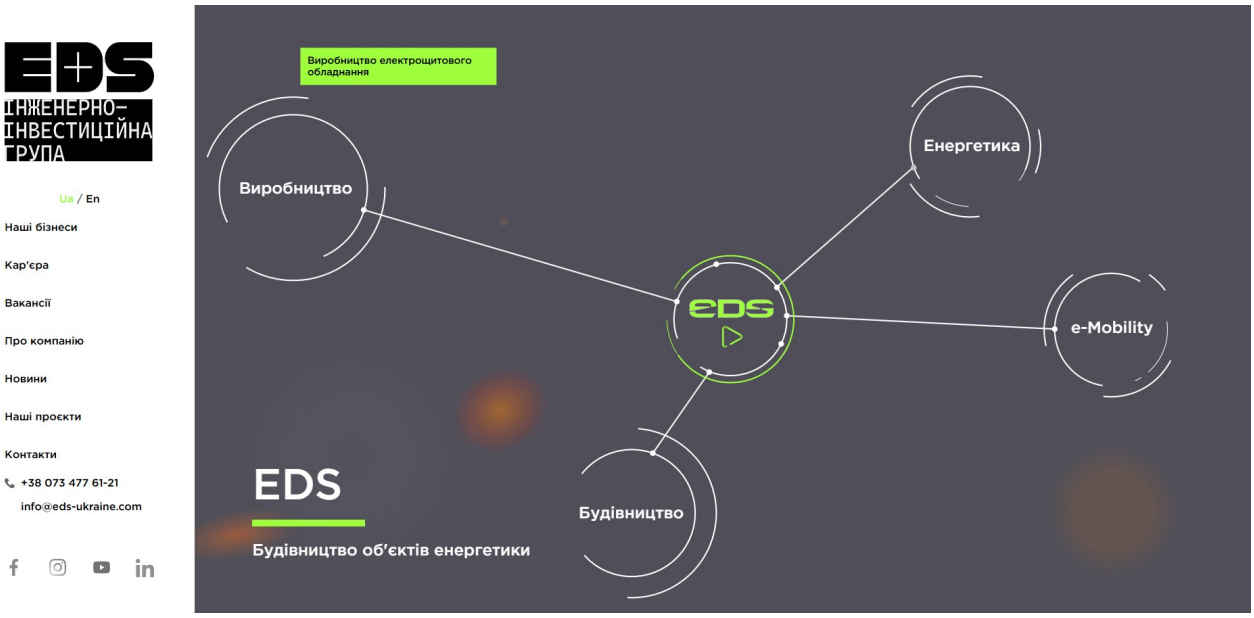

Рисунок 1.2 – Вебресурс компанії DEPS

Головна стратегія EDS (рис.1.2) полягає у візуалізації інженерної експертності через розвинений блок портфоліо («Кейси»). Кожен реалізований об'єкт супроводжується детальною технічною специфікацією та фотозвітом, що є критично важливим для формування довіри у державних замовників. Особливу увагу заслуговує наявність окремого репозиторію дозвільної документації, де представлені ліцензії на проектування та монтаж. Це дозволяє замовнику за лічені секунди провести первинну кваліфікацію підрядника. Однак, суттєвим недоліком даного рішення є відсутність відкритого каталогу товарів, що змушує клієнта звертатися до менеджерів за прайс-листами навіть для типових позицій, створюючи зайві бар'єри на шляху лідогенерації.

Третім вектором аналізу став огляд типових регіональних конкурентів у м. Івано-Франківськ, більшість яких використовують прості сайти на базі шаблонних рішень. Такі ресурси зазвичай виконують роль статичної візитки,

де представлена лише загальна інформація про компанію та перелік телефонів. Основною проблемою таких сайтів є повна відсутність засобів для роботи з технічними завданнями (ТЗ). Якщо великий корпоративний клієнт бажає надіслати план енергомережі заводу для розрахунку кошторису, регіональні сайти не надають безпечного інтерфейсу для завантаження таких файлів, змушуючи контрагентів використовувати електронну пошту, що унеможливлює автоматизацію трекінгу заявок.

Результати порівняльного аналізу функціональних можливостей аналогів представлені у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика функціоналу існуючих аналогів

Функціональний параметр	**DEPS.ua**	**EDS Engineering**	**Регіональні сайти**	**ТОВ «Мегаватсервіс» (Проєкт)**
Складний B2B каталог	Повний	Відсутній	Текстовий список	Пріоритетний
Репозиторій PDF-ліцензій	Частково	Повний	Тільки фото	Повний (Блок довіри)
Завантаження файлів ТЗ	Ні	Ні	Ні	Так (Multipart/API)
Інтеграція з Prozorro	Ні	Ні	Ні	Так (Інфо-блок)
Асинхронна робота інтерфейсу	Низька	Середня	Відсутня	Висока (Fetch/AJAX)

Визначивши функціональні орієнтири, необхідно провести глибокий аналіз способів програмної реалізації інформаційного web-сайту. В сучасній вебінженерії існують два концептуально різні підходи: використання монолітних систем керування вмістом (CMS) та спеціалізована кастомна розробка на базі мікрофреймворків.

Монолітний підхід (на прикладі WordPress або Joomla) базується на використанні вже готового ядра, де логіка обробки даних, база даних та інтерфейсна частина тісно пов'язані між собою. Для ТОВ «Мегаватсервіс» такий шлях видається оманливо легким. На початковому етапі CMS дозволяє швидко розгорнути сайт, проте при спробі імплементації специфічних B2B-

функцій, таких як завантаження багатомегабайтних інженерних креслень або створення каталогу без кошика, розробник стикається з обмеженнями ядра системи.

На противагу монолітам, кастомна розробка з використанням розділеної архітектури (Decoupled Architecture) передбачає створення незалежного Frontend-інтерфейсу та Backend-ядра. Основна перевага тут полягає у повному контролі над структурою бази даних та мережевими endpoint-ами. У випадку ТОВ «Мегаватсервіс», кастомна архітектура дозволяє реалізувати асинхронний шлюз для прийому заявок. Коли клієнт завантажує план приміщення, система обробляє цей потік даних у фоновому режимі, не змушуючи інших користувачів чекати відповіді сервера. Це досягається за рахунок використання неблокуючих механізмів, які відсутні у традиційних CMS.

Таблиця 1.2 – Порівняльний аналіз способів реалізації архітектури web-сайту

Критерій порівняння	**Монолітна CMS (WordPress)**	**Кастомна розробка (Frameworks)**
Архітектурна гнучкість	Обмежена жорстким ядром системи	Абсолютна (під процеси замовника)
Швидкодія сервера	Низька (через надмірність коду)	Висока (тільки необхідні модулі)
Обробка великих файлів	Потребує плагінів, ризик Timeouts	Асинхронна потокова обробка
Рівень безпеки	Низький (публічні вразливості)	Високий (ізоляція endpoint-ів)
Супровід та масштабування	Складне через заплутаність плагінів	Легке (модульна структура коду)

Також важливим аспектом є безпека комерційної таємниці. У кастомних рішеннях розробник власноруч проєктує механізми санітизації вхідних даних та обробки файлів, що нівелює ризики виконання шкідливого коду на сервері. У CMS-системах, через їхню масовість, вразливості стають відомими мільйонам хакерів, що робить корпоративну інформацію підприємства вразливою.

Порівняння способів реалізації за технічними критеріями наведено в таблиці 1.2.

Підсумовуючи результати порівняльного аналізу, можна зробити висновок, що для створення сучасного інформаційного web-сайту ТОВ «Мегаватсервіс», який має стати надійним інструментом B2B-продажів та тендерної участі, найбільш доцільним є шлях кастомної розробки. Це дозволить реалізувати асинхронний обмін документацією, забезпечити миттєву швидкість реакції інтерфейсу та створити безпечне середовище для збереження технічних планів і креслень замовників, що є неможливим при використанні стандартних коробочних рішень. Вибір даного підходу також закладає фундамент для майбутньої інтеграції сайту з внутрішніми ERP-системами підприємства через власні програмні інтерфейси.

1.5 Постановка завдання

На основі проведеного системного аналізу предметної області, специфіки B2B-комунікацій інжинірингових компаній, нормативної бази України та технічних обмежень сучасних архітектур, сформулюємо суворе інженерне завдання на проєктування та програмну реалізацію інформаційного web-сайту ТОВ «Мегаватсервіс» (м. Івано-Франківськ) засобами JavaScript та Python 3 [4].

Основна мета роботи полягає в розробці повнофункціонального, надійного та адаптивного інформаційного web-сайту для ТОВ «Мегаватсервіс», який забезпечить цифровізацію взаємодії з корпоративними клієнтами шляхом інтеграції модулів перевірки підрядника (довіри), електронного каталогу та системи обробки технічних завдань.

Для досягнення цієї мети слід вирішувати такі основні інженерно-технічні задачі:

1. Реалізація автономного персистентного шару (Бази даних). Створення реляційної структури в СУБД PostgreSQL, яка забезпечує строгий облік B2B-сутностей: категорій та специфікацій інженерного обладнання, реєстру

дозвільних документів та ліцензій (PDF-репозиторій), а також масиву вхідних заявок (лідів) із підтримкою зберігання реквізитів юридичних осіб (ЄДРПОУ) та шляхів до бінарних файлів технічних завдань.

2. Побудова асинхронного інтеграційного Backend-ядра. Розробка серверних модулів мовою Python 3, які за допомогою неблокуючого фреймворку FastAPI здійснюють обробку багатокомпонентних HTTP-запитів (Multipart/form-data), виконують потоковий запис завантажених клієнтами креслень на диск сервера та забезпечують сувору валідацію вхідних структур через Pydantic-схеми.

3. Забезпечення ідемпотентності та безпеки даних. Впровадження алгоритмів, які при масовій відправці форм запобігають дублюванню заявок. Реалізація превентивних методів захисту на рівні коду: автоматична санітизація текстових полів для запобігання XSS-атакам та використання параметризованих ORM-запитів (SQLAlchemy) для захисту від SQL-ін’єкцій [17].

4. Розробка реактивного адаптивного інтерфейсу користувача (Frontend). Створення web-сторінок на основі компонентного підходу, які забезпечують динамічну візуалізацію "Блоку довіри" (репозиторій ліцензій та віджет Prozorro), навігацію по ієрархічному каталогу обладнання з логікою "Запит ціни" замість споживчого кошика, інтерактивну форму завантаження ТЗ із візуальною індикацією прогресу передачі даних. Інтерфейс має бути реалізований засобами JavaScript (Fetch API) та стилізований згідно з Mobile-first концепцією за допомогою CSS-сіток Tailwind.

5. Контейнеризація та автоматизація інфраструктури. Інкапсуляція серверного коду Python, статик-файлів інтерфейсу, СУБД PostgreSQL та завантажених PDF-документів в ізольовані Docker-images. Розробка декларативного маніфесту Docker Compose для оркестрації мікросервісного середовища, що дозволяє розгорнути web-сайт on-premise у локальній мережі підприємства однією командою [19].

Технічні вимоги до показників швидкодії та надійності:

− Час виконання SQL-запитів при формуванні сторінки каталогу обладнання або списку ліцензій у Блоці довіри не повинен перевищувати 20–30 мілісекунд завдяки впровадженню B-Tree індексування магістральних полів вибірки [4].

− Серверна частина повинна підтримувати асинхронну обробку файлів технічних завдань розміром до 10 Мегабайт без блокування основного циклу подій (Event Loop), забезпечуючи безперебійний доступ до сайту іншим користувачам під час завантаження [22].

− Система повинна гарантувати стійкість до відмов: у разі критичних помилок при записі файлів або недоступності бази даних, web-сайт не повинен аварійно завершувати процес; сервер має перехоплювати винятки, логувати їх та повертати користувачу інформативне повідомлення з кодом HTTP 500 [3].

У першому розділі було здійснено системне обґрунтування розробки інформаційного web-сайту для ТОВ «Мегаватсервіс». Доведено, що сегмент B2B та тендерна діяльність вимагають специфічних архітектурних рішень, таких як відмова від кошика на користь форм RFQ (Request for Quote) та створення захищених репозиторіїв ліцензій. Проведено порівняльний аналіз аналогів (DEPS, EDS Engineering), який виявив необхідність посилення асинхронності інтерфейсу та засобів роботи з ТЗ. Обґрунтовано перехід від монолітних CMS до кастомної розділеної архітектури на базі Python та JavaScript. Сформоване технічне завдання на проєктування системи із використанням Docker контейнеризації повністю відповідає сучасним стандартам комп'ютерної інженерії та бізнес-потребам підприємства у м. Івано-Франківськ.

2 АРХІТЕКТУРНЕ ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО WEB-САЙТУ

2.1 Специфікація вимог та багаторівнева системна декомпозиція web-сайту

Проєктування інформаційного web-сайту для інжинірингової компанії ТОВ «Мегаватсервіс» вимагає ретельного переходу від бізнес-аналізу до побудови надійної програмної архітектури. Враховуючи орієнтацію підприємства на B2B-сегмент та необхідність роботи з тендерною документацією, архітектура системи повинна забезпечувати високу швидкість відгуку, надійність збереження багатомегабайтних технічних файлів та бездоганну пошукову оптимізацію для регіонального ринку міста Івано-Франківськ. Для досягнення цих цілей було прийнято рішення відмовитися від монолітних структур на користь багаторівневої мікрокомпонентної архітектури (рис. 2.1), що базується на принципі слабкої зв'язності (Low Coupling) та високої згуртованості (High Cohesion).

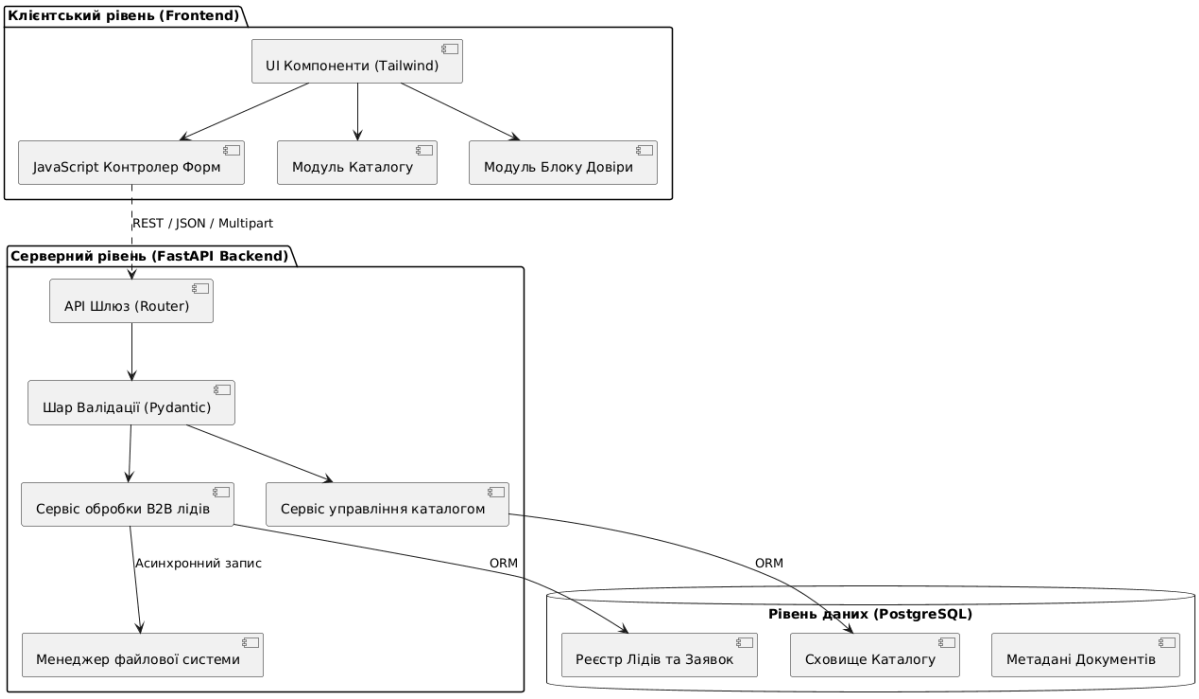

Рисунок 2.1 – Діаграма компонентів інформаційного web-сайту

Глобально систему декомпоновано на три фундаментальні рівні: шар клієнтського представлення, шар бізнес-логіки та шлюзування, а також шар збереження даних. Рівень представлення функціонує виключно у браузері клієнта і відповідає за динамічний рендеринг інтерфейсів, формування форм запиту комерційної пропозиції та первинну валідацію вводу. Шар бізнес-логіки, розташований на сервері, ізолює в собі всі інтелектуальні алгоритми: від перевірки типів файлів технічних завдань до генерації безпечних шляхів їх збереження. Шар збереження даних абстрагує процеси взаємодії з реляційною базою даних та файловою системою сервера.

Важливою складовою декомпозиції є виділення підсистеми «Блок довіри», яка програмно реалізується як окремий мікромодуль. Цей модуль відповідає за структуроване відображення ліцензій на електромонтажні роботи та сертифікатів відповідності у форматі PDF, а також за інтеграцію даних про успішні закупівлі з майданчика Prozorro. Іншим критичним модулем є підсистема електронного каталогу, яка на рівні коду позбавлена будь-яких транзакційних механізмів (кошиків чи платіжних шлюзів), що притаманні B2C-системам. Натомість каталог тісно інтегрований з підсистемою обробки складних лідів, яка здатна приймати багатокомпонентні запити, що містять як текстові реквізити підприємства (ЄДРПОУ), так і бінарні дані інженерних креслень.

Процес декомпозиції інформаційного web-сайту для ТОВ «Мегаватсервіс» виходить далеко за межі простого розподілу графічних елементів, охоплюючи фундаментальну перебудову логічних зв'язків усередині програмного комплексу. Специфіка інжинірингової компанії вимагає створення такої структури, де кожен функціональний вузол є автономним, але при цьому інтегрованим у загальну подієво-орієнтовану модель. Розподіл на рівні Frontend, Backend та Database дозволяє локалізувати потенційні помилки та забезпечити паралельну роботу над різними компонентами системи, що є критично важливим для довгострокового супроводу ресурсу. Рівень представлення розглядається не просто як набір HTML-сторінок, а як

повноцінна клієнтська оболонка, що бере на себе значну частину обчислювального навантаження з валідації та первинної обробки метаданих. Це дозволяє розвантажити серверні потужності та забезпечити миттєву реакцію інтерфейсу на дії користувача.

На особливу увагу заслуговує програмна ізоляція «Блоку довіри», який проєктується як незалежний модуль з власними методами доступу до даних. Оскільки цей блок є критичним для проходження тендерних процедур, його внутрішня архітектура має бути максимально стійкою до змін у структурі основного каталогу. Програмна декомпозиція також передбачає виділення окремого сервісу для управління бінарними потоками, який відповідає за асинхронну передачу інженерних креслень від клієнта до серверного сховища. Такий підхід гарантує, що операції з великими об'єктами даних не впливатимуть на загальну пропускну здатність API-шлюзу. Впровадження принципів слабкої зв'язності дозволяє у майбутньому інтегрувати сайт із зовнішніми ERP-системами підприємства без необхідності проведення глобального рефакторингу коду.

2.2 Поведінкове моделювання та аналіз прецедентів взаємодії

Для формалізації вимог до розробки програмного коду та визначення меж відповідальності системи було застосовано методологію Use-Case моделювання за стандартами уніфікованої мови моделювання UML. Цей етап є критично важливим, оскільки у сегменті B2B-продажів та тендерних закупівель актори мають значно складнішу логіку взаємодії, ніж у традиційних роздрібних інтернет-магазинах.

У межах проєктованої інформаційної системи виділено трьох ключових акторів (рис.2.2). Першим актором є корпоративний клієнт (представник бізнесу або державної установи), який взаємодіє з публічною частиною web-сайту з метою пошуку підрядника, аналізу електронного каталогу обладнання та ініціалізації запиту комерційної пропозиції шляхом завантаження власного

технічного завдання. Другим актором виступає адміністратор або менеджер ТОВ «Мегаватсервіс», який використовує закриту частину системи для моніторингу нових надходжень лідів, оновлення прайс-листів та завантаження нових скан-копій ліцензій у репозиторій документів. Третім, зовнішнім актором є система державних закупівель Prozorro, з якою web-сайт може взаємодіяти на рівні гіперпосилань або API-запитів для формування блоку соціального підтвердження та експертності.

Аналіз поведінки корпоративного клієнта дозволяє виділити складний прецедент оформлення B2B-заявки з прикріпленням файлу. Цей прецедент не є монолітним, а концептуально включає в себе обов'язкову процедуру валідації ідентифікаційних даних контрагента, зокрема перевірку формату коду ЄДРПОУ та контактного телефону. Зв'язок типу включення гарантує, що система не дозволить ініціювати мережевий запит до сервера та не створюватиме "сміттєвих" записів у базі даних, якщо клієнт не пройшов первинну логічну перевірку на стороні браузера. Іншим важливим прецедентом є перегляд тендерних ліцензій, що дозволяє державним замовникам миттєво перевірити наявність необхідних допусків до високовольтних робіт без необхідності надсилати офіційні запити на електронну пошту компанії.

Рисунок 2.2 – UML-діаграма прецедентів інформаційного web-сайту

Поведінкове моделювання в межах веброзробки для B2B-сегменту вимагає глибокого розуміння рольових моделей корпоративних замовників. Аналіз прецедентів дозволяє виявити неявні вимоги до системи, які часто ігноруються при проєктуванні стандартних роздрібних сайтів. Для корпоративного клієнта ТОВ «Мегаватсервіс» головним фактором є мінімізація часу на передачу технічної інформації інженерам-виконавцям. Тому прецедент подачі заявки розглядається як складний ланцюг взаємозалежних дій, де успіх кожної наступної ланки залежить від коректності виконання попередньої. Валідація ідентифікаційних даних, зокрема коду ЄДРПОУ, інтегрується безпосередньо у процес формування запиту, що дозволяє системі автоматично ідентифікувати статус контрагента та надавати йому відповідний рівень технічної підтримки.

З іншого боку, роль адміністратора системи в м. Івано-Франківськ передбачає не лише моніторинг вхідних повідомлень, а й виконання функцій цифрового аудитора. Моделювання прецедентів управління контентом дозволяє спроектувати інтерфейси, які виключають можливість випадкового видалення критично важливих ліцензій чи сертифікатів. Взаємодія із зовнішнім актором у вигляді системи Prozorro розглядається як інваріантний потік даних, що забезпечує легітимізацію діяльності компанії в очах потенційних замовників. Глибока специфікація кожного прецеденту дозволяє розробнику чітко визначити межі транзакцій у базі даних та налаштувати систему прав доступу таким чином, щоб забезпечити максимальну безпеку комерційної таємниці підприємства.

2.3 Часове моделювання асинхронних сесій обміну документацією

Оскільки однією з головних функцій розроблюваного web-сайту є прийом технічної документації від клієнтів (планів приміщень, дефектних актів, креслень у форматі PDF), критично важливим етапом комп'ютерної інженерії є проєктування правильного життєвого циклу HTTP-запиту. Класичні синхронні

сервери при отриманні багатомегабайтних файлів повністю блокують робочий потік, очікуючи завершення запису байтів на жорсткий диск. Це призводить до того, що під час завантаження креслення одним клієнтом, інші відвідувачі сайту відчувають критичні затримки при спробі відкрити сторінку каталогу.

Для вирішення цієї проблеми застосовано хронологічне моделювання на основі UML-діаграми послідовності (рис. 2.3), яка описує асинхронну парадигму взаємодії. Процес ініціюється клієнтським браузером, який за допомогою JavaScript-об'єкта FormData формує складний багатокомпонентний запит (Multipart/form-data) і відправляє його до API-шлюзу сервера. Фреймворк FastAPI приймає цей запит і передає його на рівень бізнес-логіки. На цьому етапі сервер ініціює створення унікального ідентифікатора (UUID) для уникнення конфліктів імен файлів, після чого передає завдання збереження файлу спеціальній асинхронній бібліотеці вводу-виводу.

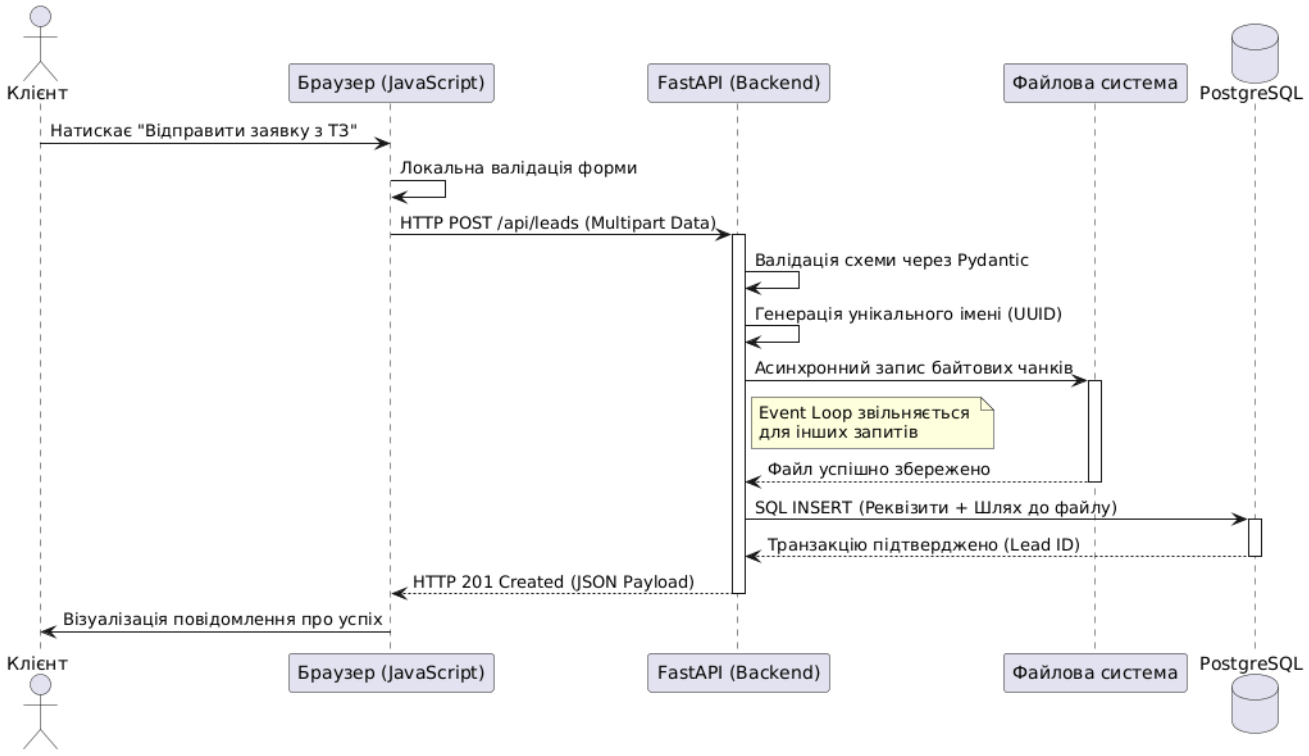

Рисунок 2.3 – UML-діаграма послідовності процесу обробки B2B-заявки

Ключовий інженерний момент полягає у тому, що на час запису файлових чанків на диск, серверний цикл подій (Event Loop) мови Python звільняється. Він не чекає механічного завершення роботи жорсткого диска, а миттєво перемикається на обслуговування інших користувачів web-сайту. Лише після отримання системного сигналу про успішне збереження файлу, процес повертається до поточної сесії, ініціює транзакцію у базі даних PostgreSQL, де зберігає текстові реквізити компанії-замовника та згенерований шлях до файлу. Завершується цикл формуванням JSON-відповіді з кодом успішного створення ресурсу, що слугує сигналом для клієнтського інтерфейсу змінити стан модального вікна на повідомлення про успіх.

Хронологічне моделювання обміну документацією є ключовим етапом комп'ютерної інженерії, що дозволяє візуалізувати часові витрати на виконання мережевих операцій. У розроблюваному web-сайті застосовується концепція неблокуючої взаємодії, яка базується на використанні асинхронного циклу подій. На відміну від застарілих синхронних моделей, де кожен крок запису файлу на диск супроводжується простоєм процесорного часу, асинхронна послідовність дозволяє серверу делегувати важкі операції вводу-виводу операційній системі. Це створює ефект багатозадачності, при якому web-сайт залишається повністю доступним для навігації навіть у моменти пікового навантаження, коли декілька замовників одночасно завантажують інженерні плани високої роздільної здатності.

Проєктування діаграми послідовності також дозволяє чітко розмежувати життєві цикли об'єктів на стороні клієнта та сервера. Кожен етап передачі даних супроводжується генерацією відповідних статусних повідомлень, що дозволяє користувачу в реальному часі відстежувати прогрес обробки його технічного завдання. Особлива увага приділяється моменту фіксації транзакції у базі даних PostgreSQL, який відбувається лише після підтвердження успішного запису файлу на фізичний носій. Це гарантує цілісність інформації та унеможливлює появу "битих" посилань у системі управління лідами. Використання UUID для ідентифікації файлів на часовій шкалі запобігає

виникненню колізій при паралельному доступі, що є критично важливим для надійної роботи корпоративного порталу.

2.4 Алгоритмічне обґрунтування безпеки та логіки обробки заявок

Забезпечення безпеки інформаційної інфраструктури ТОВ «Мегаватсервіс» вимагає ретельного алгоритмічного проєктування процесів обробки даних, що надходять від неперевірених джерел. Алгоритм прийому B2B-заявки (рис. 2.4) є багаторівневим фільтром, який повинен захистити сервер від завантаження шкідливого програмного забезпечення (Remote Code Execution) та запобігти міжсайтовому скриптингу (Cross-Site Scripting).

Робота алгоритму розпочинається з розбору багатокомпонентного запиту. На першому етапі відбувається перевірка текстових даних за допомогою строгих регулярних виразів. Алгоритм перевіряє, чи відповідає введений код ЄДРПОУ стандарту (містить виключно від 8 до 10 цифр), а контактний телефон не містить недопустимих літерних символів. Якщо текстова частина не проходить валідацію, алгоритм переривається, повертаючи помилку обробки сутності.

Наступним, найбільш критичним етапом є аналіз прикріпленого файлу. Зловмисники часто намагаються завантажити виконувані скрипти під виглядом документів. Тому алгоритм здійснює жорстку перевірку розширення файлу (дозволяючи виключно формати PDF, DOCX, DWG та стандартні зображення) і додатково перевіряє його MIME-тип. Навіть при успішному проходженні цих перевірок, алгоритм ніколи не зберігає файл під його оригінальним ім'ям, щоб уникнути атак типу Path Traversal. Замість цього генерується абсолютно новий криптографічний хеш, який стає новим іменем файлу. Останнім кроком є транзакційне збереження даних у реляційну базу, після чого алгоритм завершує свою роботу успішним сповіщенням. Така жорстка послідовність дій гарантує, що жоден шкідливий елемент не зможе проникнути у внутрішню файлову систему корпоративного сервера.

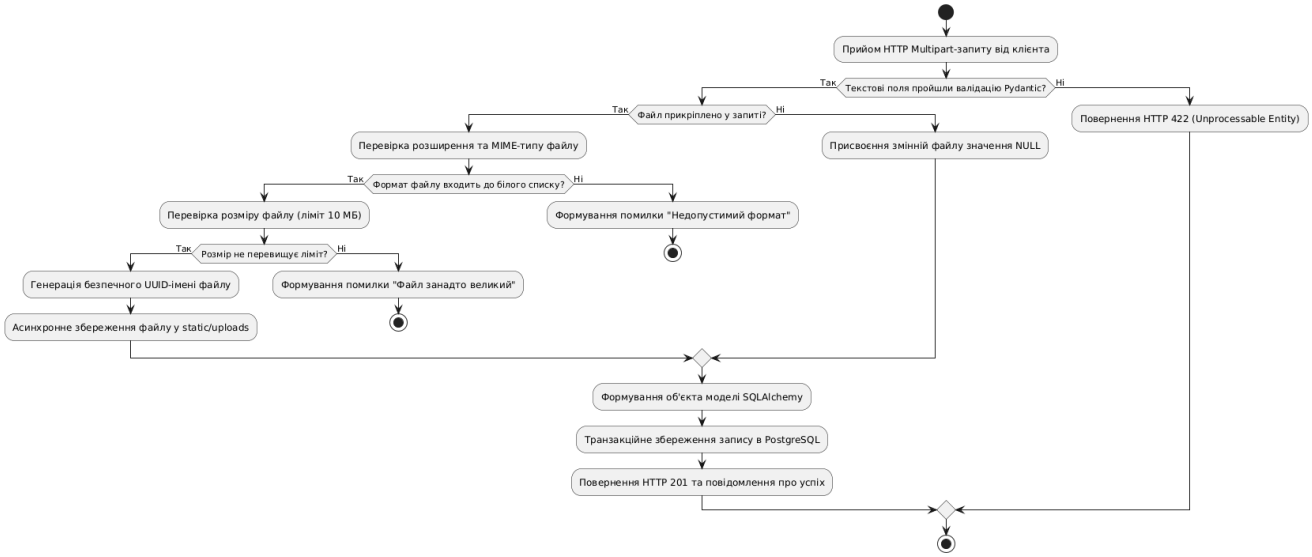

Рисунок 2.4 – Блок-схема алгоритму обробки Multipart-запиту з ТЗ

Алгоритмічне забезпечення безпеки інформаційного web-сайту базується на концепції наскрізного контролю даних на всіх етапах їхнього життєвого циклу. Перший ешелон алгоритмічного захисту фокусується на запобіганні завантаженню шкідливого коду під виглядом документів. Алгоритм валідації файлів реалізує багаторівневу перевірку, включаючи аналіз сигнатур файлів, що дозволяє ідентифікувати спроби маскування виконуваних скриптів під формати PDF чи DWG. Такий підхід нівелює загрози типу Remote Code Execution, які є одними з найнебезпечніших для корпоративних серверів. Крім того, алгоритм автоматично виконує санітизацію імен файлів, видаляючи будь-які метасимволи, що могли б бути використані для атак на файлову систему або обходу обмежень директорій.

Другий вектор алгоритмічного проєктування спрямований на захист від міжсайтового скриптингу (XSS), що досягається шляхом примусової трансформації вхідних текстових потоків у безпечні HTML-сутності. Це алгоритмічне рішення гарантує, що навіть у разі потрапляння шкідливого JavaScript-коду в поле замовлення, він буде відображений як звичайний текст і не зможе бути виконаний у браузері адміністратора. Окремо розроблено алгоритм забезпечення ідемпотентності мережевих запитів, який запобігає

створенню дублікатів заявок при нестабільному інтернет-з'єднанні у клієнта. Використання хеш-сум для перевірки унікальності кожного вхідного ліда дозволяє оптимізувати навантаження на базу даних та забезпечити високу точність статистичної звітності підприємства.

2.5 Проєктування реляційної структури сховища даних PostgreSQL

Серцем будь-якої корпоративної інформаційної системи є її база даних. З огляду на необхідність суворого збереження зв'язків між заявками клієнтів та інженерним обладнанням, було обрано об'єктно-реляційну систему управління базами даних PostgreSQL. На відміну від нереляційних рішень (NoSQL), які схильні до дублювання інформації та втрати узгодженості, реляційна модель у третій нормальній формі (3НФ) ідеально підходить для фінансового та бухгалтерського обліку заявок підприємства.

Проєктування схеми даних враховує специфіку B2B-моделі ведення бізнесу. Структура сховища базується на трьох основних таблицях-сутностях. Перша таблиця відповідає за електронний каталог обладнання. Вона містить унікальний ідентифікатор, назву позиції, текстовий опис специфікацій, категорію для фільтрації та спеціальне текстове поле для ціни. Відмова від використання строгого числового формату для ціни є свідомим архітектурним рішенням, оскільки в оптовій торгівлі вартість обладнання часто залежить від обсягу закупівлі або складності проєкту, тому поле повинно підтримувати текстові значення типу «За запитом» або «Згідно з кошторисом».

Друга таблиця спроєктована для збереження вхідних лідів та технічних завдань. Особливістю цієї таблиці є наявність полів для збереження коду ЄДРПОУ компанії-замовника та імені контактної особи. Замість того, щоб зберігати багатомегабайтні файли креслень безпосередньо у вигляді бінарних великих об'єктів (BLOB) всередині бази даних (що неминуче призвело б до роздування розміру бази та деградації продуктивності), у таблиці зберігається

лише текстовий вказівник — відносний шлях до файлу у локальній файловій системі сервера.

Третя таблиця формує базис для «Блоку довіри». Вона виступає як цифровий реєстр дозвільної документації, зберігаючи назви тендерних ліцензій, сертифікатів ISO, їх дати завантаження та шляхи до відповідних PDF-файлів. Усі таблиці забезпечені автоматичними автоінкрементними первинними ключами. Крім того, для таблиці лідів впроваджено композитний B-Tree індекс по полях дати створення та коду підприємства, що забезпечує миттєвий пошук потрібної заявки менеджерами компанії незалежно від загального обсягу накопичених даних у базі.

Для забезпечення роботи динамічного візуального слайдера партнерів на головній сторінці web-сайту було спроєктовано окрему таблицю-сутність для каруселі. Вона зберігає текстові назви контрагентів та відносні шляхи до графічних файлів з їхніми логотипами на сервері, що дозволяє адміністраторам системи оновлювати партнерський блок через базу даних без необхідності втручання у вихідний HTML-код.

ER-діаграма бази даних web-сайту зображена на рисунку 2.5.

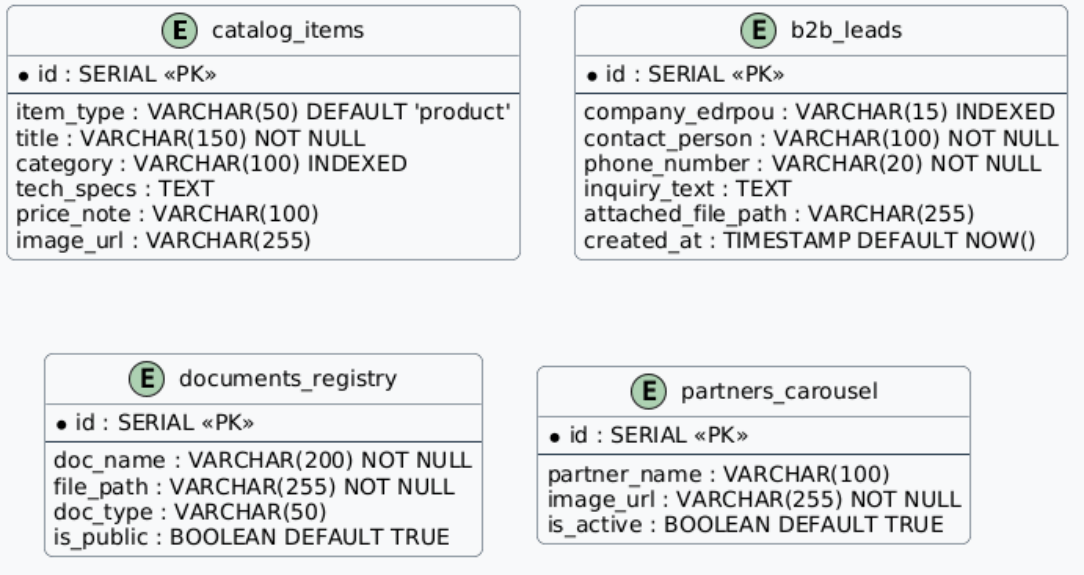

Рисунок 2.5 – ER-діаграма бази даних web-сайту

Своєю чергою, облік інжинірингових та електромонтажних послуг підприємства інтегровано в загальну таблицю каталогу за допомогою додаткового ідентифікатора типу сутності. Використання спеціального строкового поля для ціни замість жорсткого числового формату дозволяє системі коректно та динамічно відображати гнучкі тарифи на послуги, виводячи на екран формулювання на кшталт «Розрахунок за кошторисом проєкту» або «Після проведення технічного аудиту», що повністю задовольняє складні бізнес-вимоги інжинірингового підприємства.

Таблиця catalog_items (Електронний каталог обладнання та послуг) виступає основним джерелом даних для фронтенд-модуля відображення товарів. Кожне поле спроєктовано для забезпечення гнучкості представлення інженерних рішень та послуг компанії.

− id (тип SERIAL, Primary Key) — унікальний системний ідентифікатор запису, що використовує механізм автоінкременту СКБД для автоматичного присвоєння номерів новим позиціям, забезпечуючи цілісність первинних ключів.

− item_type (тип VARCHAR(50), DEFAULT 'product') — службовий атрибут для диференціації сутностей, що дозволяє системі розрізняти фізичні товари та інжинірингові послуги при виведенні даних на Frontend.

− title (тип VARCHAR(150), NOT NULL) — текстове поле для збереження повної назви інженерної одиниці або послуги, де обмеження за довжиною та обов'язковість заповнення гарантують коректність рендерингу заголовків у браузері.

− category (тип VARCHAR(100), INDEXED) — атрибут для логічного групування позицій каталогу, наявність індексу на якому дозволяє миттєво виконувати фільтрацію даних за запитом користувача.

− tech_specs (тип TEXT) — поле необмеженої довжини для детального опису характеристик, специфікацій та нормативів, що дозволяє зберігати складні текстові блоки без втрати даних.

− price_note (тип VARCHAR(100)) — специфічне B2B-поле для збереження гнучких умов ціноутворення, таких як «За запитом» або «Згідно з кошторисом», що критично для інжинірингової галузі.

− image_url (тип VARCHAR(255)) — поле для зберігання відносного шляху до графічного файлу у файловій системі сервера, що забезпечує візуалізацію товару без використання бінарних об'єктів у базі даних.

Таблиця b2b_leads (Реєстр вхідних заявок та технічних завдань) є серцем модуля лідогенерації, акумулюючи комерційні запити від підприємств та державних установ.

− id (тип SERIAL, Primary Key) — внутрішній номер заявки для її ідентифікації в адміністративній панелі управління.

− company_edrpou (тип VARCHAR(15), INDEXED) — поле для збереження коду юридичної особи, індексація якого дозволяє адміністраторам швидко проводити пошук та моніторинг звернень від конкретного контрагента.

− contact_person (тип VARCHAR(100), NOT NULL) — ПІБ відповідального представника замовника, необхідне для ініціалізації подальших переговорів інженерним відділом.

− phone_number (тип VARCHAR(20), NOT NULL) — контактний номер телефону замовника, що проходить первинну валідацію на рівні API-контролера.

− inquiry_text (тип TEXT) — довільний коментар замовника, де описуються особливості об'єкта, технічні вимоги або терміновість реалізації проєкту.

− attached_file_path (тип VARCHAR(255)) — критично важливе поле-вказівник, що зберігає відносний шлях до файлу технічного завдання на сервері, унеможливлюючи перевантаження БД бінарними даними.

− created_at (тип TIMESTAMP, DEFAULT NOW()) — автоматично генерована часова мітка, що дозволяє контролювати час опрацювання ліда менеджерами компанії.

Таблиця documents_registry (Реєстр тендерної документації) забезпечує функціонування «Блоку довіри», надаючи контрагентам доступ до актуальних дозвільних документів.

− id (тип SERIAL, Primary Key) — унікальний номер документа у цифровому тендерному архіві.

− doc_name (тип VARCHAR(200), NOT NULL) — назва ліцензії, сертифіката або витягу з реєстру, що відображається у клієнтській частині сайту.

− file_path (тип VARCHAR(255), NOT NULL) — шлях до PDF-файлу на сервері для його відображення через вбудований переглядач.

− doc_type (тип VARCHAR(50)) — класифікатор типу документа, що дозволяє структурувати репозиторій за категоріями довіри.

− is_public (тип BOOLEAN, DEFAULT TRUE) — логічний прапорець, що керує відображенням документа, дозволяючи приховати застарілі файли без їх фізичного видалення.

Таблиця partners_carousel (Реєстр партнерів для слайдера) використовується для динамічного відображення логотипів клієнтів на головній сторінці.

− id (тип SERIAL, Primary Key) — унікальний системний ідентифікатор запису для кожного партнера.

− partner_name (тип VARCHAR(100)) — назва компанії-партнера, яка може використовуватися як альтернативний текст для зображення (Alt text) задля SEO.

− image_url (тип VARCHAR(255), NOT NULL) — відносний шлях до логотипа партнера у статичному сховищі, який автоматично підтягується фронтенд-компонентом каруселі.

− is_active (тип BOOLEAN, DEFAULT TRUE) — логічний індикатор, який дозволяє вмикати або вимикати відображення логотипа в каруселі залежно від актуальності співпраці.

При проєктуванні реляційної структури інформаційного web-сайту ТОВ «Мегаватсервіс» було прийнято свідоме інженерне рішення щодо відмови від формальних зв'язків (Foreign Keys) між основними таблицями (catalog_items, b2b_leads, documents_registry). Такий підхід, хоч і виглядає нетиповим для класичних транзакційних систем, має глибоке обґрунтування у межах веброзробки B2B-порталів.

По-перше, функціональна автономність даних. Web-сайт є інформаційним ресурсом, а не закритою обліковою ERP-системою. Кожен з модулів вирішує незалежну задачу: каталог надає інформацію, блок ліцензій формує довіру, а форма лідів збирає контакти. Корпоративна заявка (b2b_lead) часто може не стосуватися конкретного товару з каталогу (наприклад, запит на ремонт мереж, яких немає в переліку товарів). Жорсткий зв'язок lead -> catalog_item створив би надлишкові обмеження, не дозволяючи клієнту надіслати довільне технічне завдання.

По-друге, приватність та безпека комерційної інформації. У B2B-сегменті дані про замовників та їхні ТЗ є конфіденційними. Відсутність каскадних зв'язків між публічним каталогом і приватною таблицею лідів мінімізує ризики витоку даних. Навіть у разі компрометації одного з модулів (наприклад, через вразливість у системі фільтрації каталогу), зловмисник не зможе через реляційні зв'язки автоматично отримати доступ до бази корпоративних клієнтів та їхніх інженерних планів.

По-третє, масштабованість та мікросервісна перспектива. Розподілена структура без жорстких зв'язків дозволяє у майбутньому легко розділити сайт на незалежні мікросервіси. Модуль каталогу може бути винесений на окремий сервер із власною базою даних, а модуль збору лідів — інтегрований безпосередньо у внутрішню CRM-систему підприємства. Відсутність зовнішніх ключів на рівні СКБД робить цей процес безболісним, оскільки кожен компонент є архітектурно завершеним і не залежить від стану інших таблиць.

По-четверте, продуктивність операцій запису. При паралельному завантаженні десятків важких ТЗ від різних клієнтів, СКБД не витрачає ресурси

на перевірку цілісності посилань (Constraint Checking) у сусідніх таблицях каталогу чи документів. Це забезпечує миттєве прийняття ліда, що є критично важливим для позитивного користувацького досвіду (UX) корпоративних замовників. Усі необхідні логічні зв'язки (наприклад, фіксація того, з якої сторінки прийшов клієнт) реалізуються на рівні Backend-логіки Python, що забезпечує необхідну гнучкість без шкоди для продуктивності та стабільності бази даних PostgreSQL.

Проєктування реляційної структури сховища для ТОВ «Мегаватсервіс» вимагає збалансованого підходу між гнучкістю збереження даних та швидкістю виконання аналітичних запитів. Вибір СУБД PostgreSQL обумовлений її здатністю ефективно працювати зі складними типами даних та забезпечувати високий рівень безпеки через механізми розмежування прав доступу на рівні рядків. Схема бази даних розробляється з урахуванням специфіки інженерного каталогу, де одна одиниця обладнання може належати до кількох категорій або мати варіативні характеристики залежно від виробника. Впровадження третьої нормальної форми дозволяє уникнути надликовості та гарантувати, що зміна назви однієї ліцензії в «Блоці довіри» автоматично оновиться на всіх сторінках сайту, де вона згадується.

Для забезпечення логарифмічної швидкодії системи у м. Івано-Франківськ, де обсяги заявок можуть зростати експоненційно, особлива увага приділяється стратегії індексування. Створення композитних індексів на полях, що найчастіше використовуються для фільтрації, дозволяє СУБД миттєво знаходити потрібні записи без повного сканування таблиць. Крім того, архітектура сховища передбачає поділ на активні та архівні дані, що дозволяє підтримувати високу продуктивність сайту протягом багатьох років експлуатації. Використання зовнішніх ключів із каскадними обмеженнями гарантує цілісність реляційних зв'язків: при видаленні застарілої категорії товарів система автоматично перевірить наявність пов'язаних лідів та запобіжить втраті важливої комерційної інформації.

2.6 Об'єктно-орієнтоване проєктування серверної частини

Розробка програмного коду серверної частини інформаційного web-сайту потребує застосування сучасних патернів об'єктно-орієнтованого проєктування для уникнення проблеми "спагетті-коду". Архітектура Backend-ядра на базі мови Python реалізує шаблон проєктування Controller-Service-Repository, який забезпечує чітке розділення відповідальності між мережевою маршрутизацією, бізнес-правилами та доступом до бази даних.

На верхньому рівні абстракції знаходяться класи-контролери, чиїм єдиним завданням є прийом HTTP-запитів від клієнтського браузера, ініціалізація валідації через моделі Pydantic та формування HTTP-відповідей. Контролери ніколи не взаємодіють з базою даних напряму. Вони передають очищені дані до шару сервісів. Сервісні класи містять у собі всю бізнес-логіку підприємства: вони вирішують, як саме обробляти завантажені файли, генерують унікальні ідентифікатори та взаємодіють з файловою системою сервера через спеціалізовані менеджери.

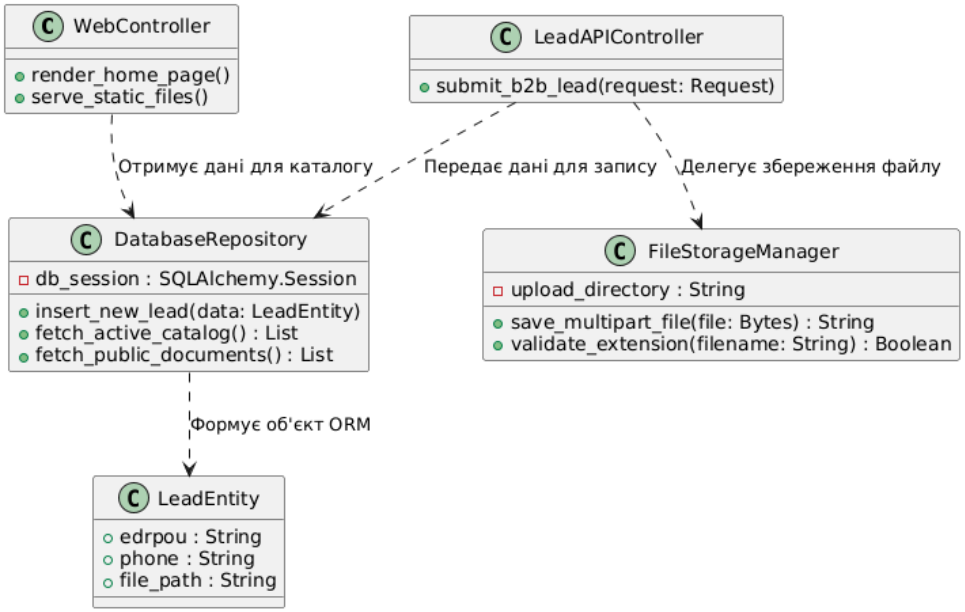

Рисунок 2.6 – UML-діаграма класів серверної частини web-сайту

Нижній рівень абстракції представлений класами-репозиторіями. Це єдині об'єкти в системі, які "знають" про існування бази даних PostgreSQL. Вони приймають команди від сервісного шару та використовують механізми об'єктно-реляційного мапування (SQLAlchemy) для формування безпечних параметризованих SQL-транзакцій. Така трирівнева об'єктна модель робить систему надзвичайно гнучкою: якщо в майбутньому компанія вирішить змінити тип бази даних, розробникам доведеться переписати лише класи репозиторіїв, тоді як логіка контролерів та сервісів залишиться абсолютно незмінною. Залежності між цими шарами вирішуються через механізм ін'єкції залежностей (Dependency Injection), який нативно підтримується фреймворком FastAPI, що спрощує процес модульного тестування коду.

Об'єктно-орієнтоване проєктування серверної частини web-сайту базується на впровадженні архітектурних патернів, що забезпечують максимальну тестованість та гнучкість коду. Розділення логіки на контролери, сервіси та репозиторії дозволяє створити чисту архітектуру, де кожен клас має єдину зону відповідальності. Контролери відповідають виключно за мережеву комунікацію та форматування відповідей, тоді як вся інженерна бізнес-логіка інкапсулюється у сервісному шарі. Це дозволяє розробникам легко змінювати правила обробки заявок без втручання у транспортний рівень системи. Використання абстрактних репозиторіїв надає можливість підміняти джерела даних для проведення модульного тестування, що суттєво підвищує надійність готового продукту.

На рівні об'єктної моделі особлива увага приділяється класу управління файловою системою, який виступає посередником між логікою збереження лідів та фізичним сховищем на диску сервера. Цей клас інкапсулює складні процедури перевірки дозволів та генерації безпечних шляхів, приховуючи технічні деталі реалізації від іншої частини системи. Механізм ін'єкції залежностей дозволяє автоматизувати життєвий цикл об'єктів бази даних, гарантуючи вчасне закриття мережевих сокетів та звільнення ресурсів. Такий підхід до проєктування класів робить програмний код інформаційного web-

сайту стійким до змін та зручним для подальшого масштабування, наприклад, при додаванні функціоналу клієнтського кабінету в майбутньому.

2.7 Обґрунтування інструментарію розробки та інфраструктури

Фінальним етапом архітектурного проєктування є обґрунтування вибору конкретних програмних технологій та інструментів розгортання, які найкраще відповідають спроєктованим алгоритмам та моделям даних.

Для створення серверного ядра абсолютно безальтернативним вибором став мікрофреймворк FastAPI на базі мови Python версії 3.10 або вище. Його інтеграція з бібліотекою Pydantic забезпечує сувору типізацію даних, що є критично важливим для надійності B2B-порталів. Асинхронна природа FastAPI гарантує, що операції вводу-виводу при завантаженні багатомегабайтних технічних завдань не вплинуть на швидкість обслуговування інших клієнтів компанії.

Для клієнтського інтерфейсу (Frontend) обрано стратегію рендерингу на стороні сервера (Server-Side Rendering) за допомогою потужного шаблонізатора Jinja2. Це забезпечує миттєве первинне відмальовування сторінок, що позитивно впливає на показники пошукової оптимізації (SEO). Динамічна поведінка форм, таких як відправка лідів без перезавантаження сторінки, реалізується за допомогою чистого JavaScript (Vanilla JS) та стандарту Fetch API, що дозволяє уникнути надмірної ваги коду, притаманної важким SPA-фреймворкам. Візуальна стилізація інтерфейсу виконується за допомогою утилітарного CSS-фреймворку Tailwind CSS, який дозволяє швидко створювати унікальні, сучасні та повністю адаптивні під мобільні пристрої дизайн-компоненти без розростання зовнішніх файлів стилів.

Щодо інфраструктурного рівня, стандартом де-факто у сучасній комп'ютерній інженерії є використання технологій контейнеризації. Впровадження Docker та Docker Compose дозволяє інкапсулювати код веб-сервера, всі його залежності, а також систему управління базами даних

PostgreSQL в єдиний, ізольований та стандартизований пакет. Це повністю вирішує класичну проблему несумісності середовищ між комп'ютером розробника та кінцевим сервером розгортання, забезпечуючи концепцію "Інфраструктура як код". Крім того, Docker дозволяє зручно монтувати локальні директорії сервера у контейнер для надійного збереження файлів тендерних ліцензій та креслень клієнтів незалежно від життєвого циклу самих контейнерів.

Вибір програмного інструментарію для розробки web-сайту ТОВ «Мегаватсервіс» є результатом ретельного аналізу продуктивності та безпеки сучасних технологій. Використання мови Python 3 у зв'язці з FastAPI обумовлено необхідністю створення високопродуктивного API, що здатне обробляти асинхронні файлові потоки з мінімальними затримками. FastAPI демонструє показники швидкодії, що наближаються до мов низькорівневого програмування, при цьому зберігаючи лаконічність та безпеку типів даних. Впровадження Tailwind CSS для фронтенд-частини дозволяє відмовитися від написання тисяч рядків кастомного CSS-коду, замінивши їх на гнучкі утилітарні класи. Це не лише прискорює процес розробки, а й гарантує ідеальну візуальну адаптивність сайту під будь-які екрани сучасних пристроїв.

Інфраструктурна стратегія, заснована на Docker Compose, забезпечує повну портативність web-сайту між різними середовищами розгортання. Використання контейнерів дозволяє інкапсулювати всі системні бібліотеки та конфігурації, виключаючи людський фактор при налаштуванні сервера в Івано-Франківську. Це створює надійний фундамент для автоматизації процесів деплою та забезпечує стабільну роботу бази даних PostgreSQL у ізольованому контурі. Обґрунтований технологічний стек є оптимальним балансом між інноваційністю та стабільністю, що дозволяє створити інформаційний web-сайт, який повністю відповідає вимогам сучасної інженерної компанії та готовий до тривалої безперебійної експлуатації.

У другому розділі було проведено комплексне архітектурне та мережеве проєктування інформаційного web-сайту, що базується на результатах

попереднього системного аналізу бізнес-процесів інжинірингової компанії. Побудована трирівнева декомпозиція системи забезпечує необхідну гнучкість та масштабованість, дозволяючи надійно ізолювати логіку обробки вхідних B2B-заявок від візуального представлення контенту та дозвільної документації. Розроблені поведінкові та часові моделі взаємодії акторів підтвердили ефективність використання асинхронної парадигми для обміну важкою технічною документацією, що є критично важливим для збереження високої швидкодії ресурсу в умовах паралельного доступу. Спроєктована реляційна структура бази даних PostgreSQL враховує специфічні вимоги щодо безпеки комерційної таємниці та функціональної автономності програмних модулів, а розроблена об'єктно-орієнтована структура серверної частини на основі сучасних патернів гарантує легкість подальшого супроводу та модифікації коду. Обґрунтований вибір технологічного стеку та стратегія контейнеризації середовища розгортання створюють надійний інфраструктурний фундамент для переходу до стадії безпосередньої програмної реалізації, тестування та інтеграції розробленого проєкту у виробничу мережу підприємства.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОГО WEB-САЙТУ

3.1 Конфігурація середовища розгортання

Практичний етап розробки програмного продукту в галузі комп'ютерної інженерії розпочинається з формування надійної інфраструктури. Для інформаційного web-сайту ТОВ «Мегаватсервіс» було обрано парадигму контейнеризації на базі технології Docker, що дозволяє повністю абстрагуватися від апаратних особливостей хост-серверів та гарантувати ідентичність виконання коду незалежно від середовища розгортання. Використання інструменту оркестрації Docker Compose дозволило декларативно описати топологію системи, яка складається з двох ключових вузлів: сервера баз даних та асинхронного веб-сервера. Це інженерне рішення унеможливлює виникнення конфліктів системних бібліотек та забезпечує високий рівень мережевої безпеки, оскільки взаємодія між компонентами відбувається виключно через віртуальний мережевий міст, прихований від зовнішньої маршрутизації.

Центральним конфігураційним файлом інфраструктури є маніфест розгортання. У цьому документі детально прописані інструкції для ініціалізації системи управління базами даних PostgreSQL версії 15. Для оптимізації споживання оперативної пам'яті сервера було обрано дистрибутив на базі Alpine Linux. Особлива увага в маніфесті приділена механізму збереження інформації: використання іменованих томів гарантує персистентність комерційних даних, що означає повне збереження бази клієнтських заявок та завантажених технічних завдань навіть у разі примусового перезавантаження або оновлення образу веб-сервера. Підсистема обробки HTTP-запитів конфігурується шляхом збірки локального образу на базі мови Python, де командою запуску виступає ініціалізація ASGI-сервера Uvicorn. Важливою інженерною деталлю є директива монтування локальної директорії завантажень

безпосередньо у файлову систему контейнера, що дозволяє серверу здійснювати асинхронний запис бінарних даних без посередництва бази даних.

```
version: '3.8'
services:
  megawatt_db:
    image: postgres:15-alpine
    container_name: db_megawatt_if
    restart: always
    environment:
      POSTGRES_USER: admin_megawatt
      POSTGRES_PASSWORD: secure_B2B_pass_2026
      POSTGRES_DB: megawatt_db
    volumes:
      - postgres_data:/var/lib/postgresql/data
    networks:
      - megawatt_net
```

Наведений фрагмент конфігураційного маніфесту відповідає за розгортання реляційного сховища. У блоці середовища (environment) жорстко задаються системні ідентифікатори та паролі доступу, які згодом використовуватимуться веб-сервером для авторизації мережевих сокетів. Директива restart: always гарантує автоматичне відновлення роботи СУБД у разі внутрішнього збою ядра або падіння операційної системи сервера.

Наступна частина інфраструктурного маніфесту описує параметри збірки та мережевої інтеграції контейнера веб-додатка, що функціонує в єдиному ізольованому просторі з базою даних.

```
  megawatt_web:
    build: .
    container_name: web_megawatt_if
    restart: always
    command: uvicorn main:app --host 0.0.0.0 --port 8000
    volumes:
      - .:/app
      - ./static/uploads:/app/static/uploads
    ports:
      - "8000:8000"
    depends_on:
      - megawatt_db
    environment:
      - DATABASE_URL=postgresql://admin_megawatt:secure_B2B_pass_2026@megawatt_db:5432/megawatt_db
    networks:
      - megawatt_net
networks:
  megawatt_net:
```

```
    driver: bridge
volumes:
  postgres_data:
```

Параметр depends_on інструктує оркестратор дочекатися успішного запуску контейнера бази даних перед початком ініціалізації веб-сервера, що запобігає виникненню помилок первинного підключення. Оголошення мережі megawatt_net з драйвером типу bridge створює ізольований програмний комутатор всередині ядра операційної системи хоста, ізолюючи внутрішній стек TCP/IP від зовнішнього середовища.

3.2 Програмне конструювання об'єктно-реляційної моделі даних

Архітектура бази даних інформаційного web-сайту спроєктована з урахуванням специфіки B2B-взаємодії інжинірингового підприємства. Замість написання низькорівневих SQL-запитів, які є вразливими до ін'єкцій, реалізація персистентного шару виконана за допомогою технології об'єктно-реляційного мапування SQLAlchemy. Програмний код файлу моделей даних інкапсулює бізнес-логіку у вигляді чотирьох незалежних класів. Спочатку виконується імпорт необхідних системних типів та оголошення базового декларативного класу.

```
from sqlalchemy import Column, Integer, String, Text, Boolean, DateTime
from sqlalchemy.ext.declarative import declarative_base
import datetime
Base = declarative_base()
```

Оголошений об'єкт Base виступає в ролі реєстру сутностей, через який реляційний двигун SQLAlchemy відстежує структуру таблиць та координує виконання транзакцій. Імпортовані типи даних чітко мапуються на відповідні типи СКБД PostgreSQL, забезпечуючи сувору типізацію на рівні мови програмування.

Наступний фрагмент коду ініціалізує модель електронного каталогу, що поєднує в собі облік як фізичного обладнання, так і комплексних інжинірингових послуг.

```
class ProductItem(Base):
        __tablename__ = "catalog_items"
    id = Column(Integer, primary_key=True, index=True)
    item_type = Column(String(50), default="product")
    title = Column(String(150), nullable=False)
    category = Column(String(100), index=True)
    tech_specs = Column(Text)
    price_note = Column(String(100))
    image_url = Column(String(255))
```

Клас ProductItem трансформується у таблицю catalog_items. Атрибут item_type виступає в ролі маркерного поля, яке дозволяє системі динамічно визначати логіку відображення об'єкта. Наявність індексу на полі category суттєво прискорює операції фільтрації при виконанні пошукових запитів користувачів, а поле image_url позбавляє базу даних від необхідності зберігання важких бінарних об'єктів.

Далі описується структура таблиці для реєстрації комерційних заявок від підприємств-замовників, яка акумулює вхідні ліди.

```
class B2BLead(Base):
        __tablename__ = "b2b_leads"
    id = Column(Integer, primary_key=True, index=True)
    company_name = Column(String(150))
    edrpou = Column(String(15), index=True)
    phone = Column(String(20), nullable=False)
    file_path = Column(String(255))
    created_at = Column(DateTime, default=datetime.datetime.utcnow)
```

Модель B2BLead оптимізована під збір реквізитів юридичних осіб. Поле edrpou забезпечене системним індексом для швидкого групування запитів від одного контрагента в адміністративній панелі. Атрибут file_path призначений для зберігання текстового посилання на фізичне розташування завантаженого інженерного креслення на сервері.

Завершальний блок опису моделей даних формує структури для тендерного сховища та партнерського слайдера.

```
class Document(Base):
        __tablename__ = "documents"
    id = Column(Integer, primary_key=True)
    title = Column(String(200), nullable=False)
    file_path = Column(String(255), nullable=False)
    is_public = Column(Boolean, default=True)
class PartnerCarousel(Base):
    __tablename__ = "partners_carousel"
    id = Column(Integer, primary_key=True)
    partner_name = Column(String(100))
    image_url = Column(String(255), nullable=False)
    is_active = Column(Boolean, default=True)
```

Ці дві моделі забезпечують інформаційне наповнення динамічних зон сайту. Клас Document відповідає за верифікацію ліцензій, а PartnerCarousel постачає ресурси для безперервної стрічки брендів. Обидві таблиці функціонують повністю автономно, що виключає ризик каскадного видалення даних при проведенні технічного обслуговування суміжних модулів.

3.3 Програмування асинхронного серверного ядра та потокової обробки даних (Backend)

Серверна архітектура веб-порталу ТОВ «Мегаватсервіс» повністю реалізована на базі сучасного подієво-орієнтованого фреймворку FastAPI. Вибір цієї технології зумовлений її здатністю забезпечувати неблокуючий асинхронний ввід-вивід, що критично важливо для комп'ютерних систем, які обслуговують одночасні запити з передачею важких бінарних даних. Програмний код головного серверного файлу main.py організовано за модульним принципом. Спочатку виконується підключення всіх необхідних системних модулів та ініціалізація самого веб-додатка.

```
import os, uuid, shutil
from fastapi import FastAPI, Depends, Request, Form, UploadFile, File,
HTTPException
from fastapi.staticfiles import StaticFiles
from fastapi.templating import Jinja2Templates
from sqlalchemy.orm import Session
import models
from database import SessionLocal, engine
models.Base.metadata.create_all(bind=engine)
```

```
app = FastAPI(title="Megawattservice B2B Portal")
```

У цьому первинному фрагменті відбувається запуск механізму автоматичної міграції бази даних через команду metadata.create_all. Конструктор класу FastAPI створює екземпляр додатка, який виступає в ролі головного веб-шлюзу системи, що координує маршрутизацію вхідних пакетів TCP.

Далі програмується логіка інтеграції статичних сховищ файлової системи та конфігурується механізм генерації сесій підключення до СКБД.

```
app.mount("/static", StaticFiles(directory="static"), name="static")
templates = Jinja2Templates(directory="templates")
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
```

Директива app.mount виділяє ізольований мережевий шлях для прямої роздачі графіки та PDF-документів без залучення обчислювальних ресурсів інтерпретатора Python. Функція-генератор get_db реалізує патерн контекстного менеджера, гарантуючи, що кожне індивідуальне підключення отримає власну ізольовану транзакційну сесію СУБД, яка автоматично закриється у блоці finally.

Наступним кроком є розробка головного кореневого маршруту, який відповідає за вибірку даних та рендеринг інтерфейсу користувача.

```
@app.get("/")
async def index(request: Request, db: Session = Depends(get_db)):
        categories = db.query(models.Category).all()
  docs = db.query(models.Document).filter(models.Document.is_public ==
True).all()
                                                    partners          =
db.query(models.PartnerCarousel).filter(models.PartnerCarousel.is_active      ==
True).all()
      return templates.TemplateResponse("index.html", {
          "request": request, "categories": categories,
          "docs": docs, "partners": partners
      })
```

Асинхронний метод index виконує паралельні неблокуючі запити до реляційного сховища. За допомогою вбудованих засобів фільтрації ORM з бази даних вилучаються лише активні документи та ліцензії. Отримані масиви об'єктів передаються в оптимізований рушій шаблонізації Jinja2 для формування фінального HTML-коду сторінки.

Після реалізації логіки відображення контенту розробляється базовий каркас API-контролера для прийому Multipart-форм з комерційними запитами.

```
      @app.post("/api/leads")
      async def handle_lead(company: str = Form(...), edrpou: str = Form(...),
phone: str = Form(...),
                                  tz_file: UploadFile = File(None), db: Session =
Depends(get_db)):
          path = None
          if tz_file and tz_file.filename:
              ext = tz_file.filename.split('.')[-1]
              unique_name = f"{uuid.uuid4().hex}.{ext}"
              path = f"static/uploads/{unique_name}"
```

Контролер handle_lead очікує дані у форматі multipart/form-data. При наявності прикріпленого файлу технічного завдання алгоритм активує підсистему генерації унікальних імен. Використання криптографічного стандарту uuid4 нівелює загрозу навмисного перезапису існуючих файлів на сервері та захищає систему від атак типу підміни шляхів.

Далі програмується логіка безпосереднього бінарного збереження файлу ТЗ на жорсткий диск сервера з використанням неблокуючого буфера.

```
          with open(path, "wb") as buffer:
                 shutil.copyfileobj(tz_file.file, buffer)
```

Функція shutil.copyfileobj здійснює низькорівневе копіювання байтів з мережевого потоку в дескриптор файлу операційної системи. Процес виконується оптимізованими чанками, що запобігає піковому зростанню споживання оперативної пам'яті сервера навіть при передачі інженерних креслень великого обсягу.

Завершальний сегмент серверного ядра відповідає за фіксацію транзакції в реляційній базі даних та обробку виняткових ситуацій.

```
          try:
                  new_lead = models.B2BLead(company_name=company, edrpou=edrpou,
phone=phone, file_path=path)
              db.add(new_lead)
              db.commit()
                return {"status": "success", "message": "Заявку успішно прийнято
інженерним відділом!"}
          except Exception:
              db.rollback()
                          raise HTTPException(status_code=500, detail="Помилка
персистентності.")
```

Екземпляр класу B2BLead реєструється в контексті поточної сесії СУБД. Блок try/except гарантує транзакційну цілісність системи (принцип ACID): у разі виникнення будь-якого апаратного або програмного збою під час збереження, інструкція db.rollback() миттєво скасовує всі внесені зміни, запобігаючи заповненню сховища невалідними даними.

3.4 Програмна реалізація клієнтського інтерфейсу (Frontend)

Клієнтська оболонка інформаційного порталу розроблена із застосуванням сучасного утилітарного CSS-фреймворку Tailwind. Такий інженерний підхід дозволив повністю відмовитися від написання громіздких зовнішніх таблиць стилів та забезпечити високу швидкість завантаження сторінки за рахунок мінімізації фінального коду. Розробка Frontend-модуля починається з формування базової структури документа та підключення CDN-ресурсів.

```
      <!DOCTYPE html>
      <html lang="uk" class="scroll-smooth">
      <head>
          <meta charset="UTF-8">
          <meta name="viewport" content="width=device-width, initial-scale=1.0">
          <script src="https://cdn.tailwindcss.com"></script>
                                                                    <link
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css"
rel="stylesheet">
          <style>
```

```
                @keyframes scroll { 0% { transform: translateX(0); } 100%
{ transform: translateX(-50%); } }
             .animate-scroll { animation: scroll 20s linear infinite; display:
flex; width: max-content; }
        </style>
        <title>ТОВ Мегаватсервіс | B2B Портал Інжинірингу</title>
    </head>
    <body class="bg-slate-50 text-slate-900 font-sans">
```

Клас scroll-smooth активує вбудований у рушій браузера алгоритм плавної прокрутки екрана. У блоці style прописано декларацію анімації scroll, яка забезпечує безперервне зсунення координат за віссю абсцис, що є апаратною основою для створення майбутнього партнерського слайдера.

Наступний фрагмент коду ініціалізує фіксований верхній блок навігації, що містить актуальні корпоративні контакти компанії.

```
         <header class="bg-slate-900 text-white p-6 sticky top-0 z-50 border-
b-4 border-yellow-500 shadow-2xl">
            <div class="container mx-auto flex justify-between items-center">
                <div class="flex items-center gap-4">
                        <i class="fa-solid fa-bolt-lightning text-yellow-500
text-3xl"></i>
                    <div class="flex flex-col">
                         <span class="text-2xl font-black uppercase tracking-
tighter">Мегаватсервіс</span>
                            <span class="text-[9px] text-yellow-500 font-bold
uppercase">Інжинірингова компанія</span>
                    </div>
                </div>
                <div class="hidden lg:flex items-center gap-8">
                    <div class="text-right text-xs font-bold text-slate-300">
                              <p><i class="fa-solid fa-phone text-yellow-500
mr-2"></i>+38 (0342) 77-00-00</p>
                        <p><i class="fa-solid fa-location-dot text-yellow-500
mr-2"></i>вул. Княгинин, 44, Івано-Франківськ</p>
                    </div>
                         <nav class="flex gap-8 text-xs font-bold uppercase
tracking-widest ml-6 border-l border-slate-700 pl-6">
                             <a href="#trust" class="hover:text-yellow-400
transition">Документи</a>
                           <a href="#catalog" class="hover:text-yellow-400
transition">Каталог</a>
                       <a href="#contact" class="bg-yellow-500 text-slate-900
px-5 py-2 rounded-full hover:bg-yellow-400 transition">Надіслати ТЗ</a>
                    </nav>
                </div>
            </div>
        </header>
```

Компоненту хедера присвоєно класи sticky top-0 z-50, що примусово вилучає його зі стандартного візуального потоку та фіксує на нульовій

координаті екрана. Кнопка «Надіслати ТЗ» виділена за допомогою скруглення rounded-full та контрастного фону, виступаючи в ролі первинного елемента приваблення уваги користувача.

Далі розробляється презентаційна секція банера, що транслює офіційний слоган підприємства.

```
            <section class="bg-slate-800 text-white py-24 relative overflow-
hidden">
             <div class="container mx-auto px-6 relative z-10">
                 <h2 class="text-yellow-500 font-bold uppercase tracking-widest
mb-4">Надійність у кожному кіловаті</h2>
                      <h1 class="text-6xl font-black mb-8 uppercase leading-
none">Енергетика майбутнього</h1>
                  <p class="text-lg text-slate-300 max-w-2xl border-l-4 border-
yellow-500 pl-6">
                          ТОВ «Мегаватсервіс» — професійний інжиніринг, монтаж
складних мереж та оптове постачання обладнання для промисловості та державних
об'єктів.
                 </p>
             </div>
                               <div   class="absolute   inset-0   opacity-10
bg-[url('https://images.unsplash.com/photo-1581092160607-ee2256114e12')]     bg-
cover bg-center"></div>
         </section>
```

Секція банера використовує техніку накладання шарів за допомогою абсолютного позиціювання елемента фону (absolute inset-0). Рівень прозорості opacity-10 інтегрує індустріальне зображення у загальну композицію, зберігаючи при цьому максимальну контрастність текстових блоків білого кольору.

Наступним етапом є інтеграція динамічного слайдера брендів-партнерів компанії.

```
            <section class="py-10 bg-white overflow-hidden border-b border-
slate-200">
             <div class="animate-scroll">
                 {% for p in partners %}
                   <div class="mx-10 w-40 h-20 flex items-center justify-center
opacity-40     hover:opacity-100     transition     duration-300     grayscale
hover:grayscale-0">
                      <img src="/{{ p.image_url }}" alt="{{ p.partner_name }}"
class="max-h-full object-contain">
                 </div>
                 {% endfor %}
                 {% for p in partners %}
```

```
                    <div class="mx-10 w-40 h-20 flex items-center justify-center
opacity-40     hover:opacity-100     transition     duration-300     grayscale
hover:grayscale-0">
                        <img src="/{{ p.image_url }}" alt="{{ p.partner_name }}"
class="max-h-full object-contain">
                    </div>
                    {% endfor %}
                </div>
            </section>
```

Карусель працює без залучення мови JavaScript, спираючись на раніше оголошений клас animate-scroll. Подвійний виклик циклу {% for p in partners %} заповнює стрічку необхідною кількістю дублікатів, утворюючи ефект неперервності. Класи grayscale hover:grayscale-0 активують апаратне знебарвлення логотипів у режимі очікування.

Далі конструюється модуль підтвердження інженерної кваліфікації та інтеграції з базою державних тендерів Prozorro.

```
        <section id="trust" class="py-24 container mx-auto px-6">
                <div class="flex flex-col md:flex-row justify-between items-end
mb-14 gap-6">
                    <div class="border-l-8 border-blue-700 pl-6">
                        <h2 class="text-4xl font-black uppercase tracking-tighter
text-slate-800">Кваліфікація та Тендери</h2>
                            <p class="text-slate-500 mt-2 font-medium">Офіційна
дозвільна документація компанії</p>
                    </div>
                     <a href="https://prozorro.gov.ua/search/tender?text=37581592"
target="_blank" class="flex items-center gap-3 bg-white border-2 border-blue-600
px-6 py-3 rounded-xl font-bold text-blue-900 shadow-lg hover:bg-blue-50
transition transform hover:scale-105">
                        <span>Кейси в Prozorro</span> <i class="fa-solid fa-
external-link"></i>
                    </a>
                </div>
                <div class="grid grid-cols-1 md:grid-cols-4 gap-6">
                    {% for doc in docs %}
                     <div class="bg-white p-8 rounded-3xl shadow-sm border border-
slate-200 hover:shadow-xl transition transform hover:-translate-y-1 text-center
group cursor-pointer">
                        <i class="fa-solid fa-file-pdf text-red-600 text-5xl mb-6
group-hover:scale-110 transition duration-300"></i>
                        <h3 class="font-bold text-sm leading-tight mb-6 text-
slate-800">{{ doc.title }}</h3>
                            <a href="/{{ doc.file_path }}" target="_blank"
class="inline-block text-blue-700 font-black text-xs uppercase border-b-2
border-blue-700 pb-1 hover:text-blue-900 transition">Переглянути PDF</a>
                    </div>
                    {% endfor %}
                </div>
            </section>
```

Блок документів спирається на сітку CSS Grid Layout (grid-cols-1 md:grid-cols-4). Інструкція hover:-translate-y-1 забезпечує мікроінтерактивність: при наведенні курсора картка ліцензії плавно піднімається вгору, надаючи користувачу зрозумілий візуальний відгук про клікабельність елемента.

Наступним кроком є реалізація унікального електронного каталогу обладнання та послуг, позбавленого концепції роздрібного кошика.

Заголовок секції чітко відображає гібридну структуру каталогу. Використання класу flex-grow для параграфа опису технічних характеристик гарантує автоматичний розподіл вільного простору всередині картки, завдяки чому блоки ціни та кнопки виклику завжди вирівняні по єдиній нижній лінії.

Далі конструюється безпосередній інтерфейс лідогенерації — форма для завантаження технічних завдань та планів об'єктів.

```
          <section id="contact" class="py-28 bg-slate-900 text-white relative overflow-hidden">
              <div class="container mx-auto px-6 max-w-5xl relative z-10">
                      <h2 class="text-5xl font-black text-center text-yellow-500 mb-6 uppercase tracking-tighter">Розрахунок проєкту</h2>
                    <p class="text-slate-400 text-center mb-16 text-lg max-w-2xl mx-auto">Прикріпіть Ваш plan об'єкта або ТЗ. Наші інженерні бригади підготують комерційну пропозицію.</p>
                     <form id="leadForm" onsubmit="submitForm(event)" class="bg-slate-800 p-12 md:p-16 rounded-[3rem] space-y-8 shadow-2xl border border-slate-700">
                      <div class="grid grid-cols-1 md:grid-cols-2 gap-8">
                              <input type="text" name="company" placeholder="Назва компанії *" required class="w-full p-6 bg-slate-900 rounded-2xl border border-slate-600 outline-none focus:border-yellow-500 text-lg transition text-white">
                              <input type="text" name="edrpou" placeholder="ЄДРПОУ підприємства *" required class="w-full p-6 bg-slate-900 rounded-2xl border border-slate-600 outline-none focus:border-yellow-500 text-lg transition text-white">
                      </div>
                          <input type="tel" name="phone" placeholder="Контактний телефон *" required class="w-full p-6 bg-slate-900 rounded-2xl border border-slate-600 outline-none focus:border-yellow-500 text-lg transition text-white">
```

Вхідні поля форми інтегровані у двоколонну адаптивну структуру. Стилізація bg-slate-900 створює ефект заглиблених елементів інтерфейсу. Директива focus:border-yellow-500 забезпечує динамічну зміну кольору рамки поля при отриманні фокусу введення користувачем.

Наступний фрагмент коду імплементує кастомну зону для перетягування файлів (Drag-and-Drop) та системну область виведення сповіщень.

```
                        <div class="p-12 border-2 border-dashed border-slate-500 rounded-[2rem] text-center hover:border-yellow-500 transition cursor-pointer relative bg-slate-900/50 group">
                                    <input type="file" name="tz_file" id="tz_file" class="absolute inset-0 opacity-0 cursor-pointer z-10" onchange="updateFileName()">
                                <i class="fa-solid fa-cloud-arrow-up text-6xl text-slate-500 mb-4 group-hover:text-yellow-500 transition duration-300"></i>
                                  <p id="fileLabel" class="text-base text-slate-400 font-medium">Натисніть або перетягніть технічне завдання (PDF, DWG, DOCX)</p>
                        </div>
                          <div id="status" class="hidden p-6 rounded-2xl font-bold text-center text-lg"></div>
                        <button type="submit" id="btn" class="w-full bg-yellow-500 text-slate-900 font-black py-6 rounded-2xl text-2xl hover:bg-yellow-400 transition transform active:scale-95 uppercase tracking-widest shadow-xl mt-4">Відправити на розрахунок</button>
                    </form>
                </div>
            </section>
```

Особливістю зони завантаження є накладання нативного інпуту з нульовою прозорістю (opacity-0) поверх стилізованого пунктирного контейнера. Це дозволяє зберегти базові обробники подій операційної системи, надавши при цьому блоку індустріального дизайну.

Завершальним візуальним компонентом інтерфейсу користувача є триколонний підвал сайту (Footer).

```
            <footer class="bg-slate-950 text-slate-400 py-20 border-t border-slate-800">
                <div class="container mx-auto px-6 grid grid-cols-1 md:grid-cols-3 gap-12">
                    <div>
                                            <h4 class="text-white font-bold mb-4 uppercase">Контакти</h4>
                          <p class="text-sm">Офіс: вул. Княгинин, 44, корп. 12, м. Івано-Франківськ</p>
                        <p class="text-sm">Склад: вул. Княгинин, 44</p>
                          <p class="text-sm mt-4 font-bold text-white">+38 (0342) 77-00-00</p>
                    </div>
                         <div class="bg-slate-900 p-6 rounded-2xl border border-slate-800 text-xs font-mono">
                             <h4 class="text-white font-bold mb-2 uppercase font-sans">Бухгалтерські реквізити</h4>
                        <p>ЄДРПОУ: 37581592</p>
                        <p>IBAN: UA453366770000026001234567890</p>
                        <p>Керівник: Припхан Б. В.</p>
                    </div>
```

```
                    <div class="h-40 rounded-2xl overflow-hidden shadow-lg border
border-slate-800">
                        <iframe src="https://www.google.com/maps/embed?pb=!1m18!
1m12!1m3!1d2621.365922379391!2d24.7118023!3d48.9274294!2m3!1f0!2f0!3f0!3m2!
1i1024!2i768!4f13.1!3m3!1m2!1s0x4730c1692237eb7f%3A0x3f622ba7b6cfcc50!
2z0LLRg9C70LjRhtGPINCa0L3Rj9Cz0LjQvdC40L0sIDQ0LCDQhtCy0LDQvdC-
LdCk0YDQsNC90LrRltCy0YHRjNC6LCDQhtCy0LDQvdC-
LdCk0YDQsNC90LrRltCy0YHRjNC60LAg0L7QsdC70LRgdGC0YwsIDc2MDAw!5e0!3m2!1suk!2sua!
4v1700000000000!5m2!1suk!2sua" class="w-full h-full grayscale hover:grayscale-0
transition duration-500"></iframe>
                    </div>
                </div>
            </footer>
```

У блоці бухгалтерських реквізитів застосовано моноширинний шрифт font-mono для кращого сприйняття числових кодів. Вбудована карта за допомогою класів grayscale hover:grayscale-0 автоматично змінює режим відображення на повноколірний при фокусуванні.

Після оголошення структури інтерфейсу програмуються клієнтські скрипти автоматизації та асинхронної взаємодії.

```
            <script>
                function updateFileName() {
                    const fileInput = document.getElementById('tz_file');
                    const label = document.getElementById('fileLabel');
                    if (fileInput.files.length > 0) {
                                    label.innerText = "Файл прикріплено: " +
fileInput.files[0].name;
                        label.classList.add('text-yellow-500', 'font-bold');
                    }
                }
```

Функція updateFileName виступає в ролі локального обробника події onchange. Вона перехоплює масив об'єктів завантажених файлів та динамічно модифікує текстовий вміст елемента p, підсвічуючи назву прикріпленого інженерного проєкту яскраво-жовтим кольором.

Завершальний сегмент клієнтського коду реалізує механізм безпосередньої асинхронної відправки серіалізованих пакетів даних.

Метод submitForm повністю ізолює транзакцію від стандартного процесу перезавантаження сторінки. Конструктор FormData автоматично пакує всі текстові поля та бінарні файли. Виклик асинхронного інтерфейсу fetch

забезпечує відправку пакета, а динамічна модифікація класу status виводить результат опрацювання ліда безпосередньо на екран користувача.

Розроблений інформаційний web-сайт ТОВ «Мегаватсервіс» функціонує за принципом Single Page Application у контексті навігації, де весь ключовий контент та інструменти взаємодії розміщені на одній сторінці, яка логічно поділена на функціональні візуальні блоки.

Взаємодія користувача з порталом розпочинається з навігаційного вузла (рис. 3.1), який зафіксований у верхній частині екрана у вигляді темно-синьої панелі. Ліва частина цього блоку містить корпоративний логотип із назвою та статусом інжинірингової компанії, тоді як у правій частині виведено ключові контактні дані, зокрема номер телефону та фізичну адресу в Івано-Франківську. Там же розташовані гіперпосилання на внутрішні розділи сайту та акцентна жовта кнопка для запиту ціни. Завдяки фіксації цього блоку замовник має змогу завжди бачити актуальні контакти підприємства, незалежно від глибини прокрутки сторінки, миттєво переміщуватися до розділів документів або каталогу за допомогою швидких посилань, а також миттєво переходити до форми лідогенерації в нижній частині сайту натисканням на відповідну кнопку.

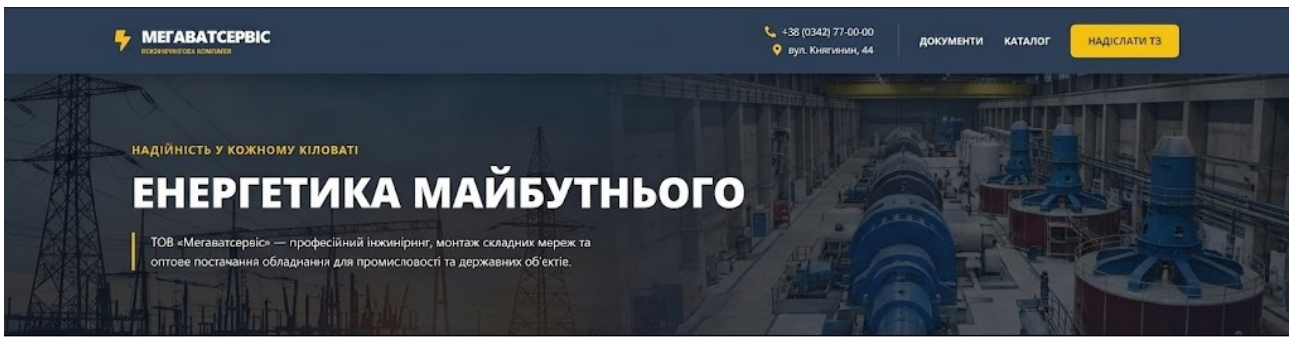

Рисунок 3.1– Головний презентаційний екран та навігаційне меню

Одразу під навігаційною панеллю розташований головний презентаційний екран, оформлений як широкоформатний блок із затемненим індустріальним фоном. Центральне місце тут займає головний корпоративний слоган про надійність у кожному кіловаті та енергетику майбутнього, який

супроводжується коротким текстовим описом профілю компанії щодо монтажу мереж та постачання обладнання. Цей блок дозволяє відвідувачу ознайомитися з головною ціннісною пропозицією компанії безпосередньо у перші секунди після завантаження сторінки. Логічним продовженням презентації є блок довіри у вигляді партнерського слайдера, візуально представленого як горизонтальна світла смуга з логотипами світових та національних брендів-партнерів. У стані спокою ці логотипи є напівпрозорими та знебарвленими. Користувач спостерігає за автоматичним безперервним прокручуванням стрічки логотипів, а при наведенні курсора миші на будь-який з них відбувається інтерактивний відгук, у результаті якого зображення набуває стовідсоткової яскравості та оригінальних кольорів корпоративного бренду контрагента.

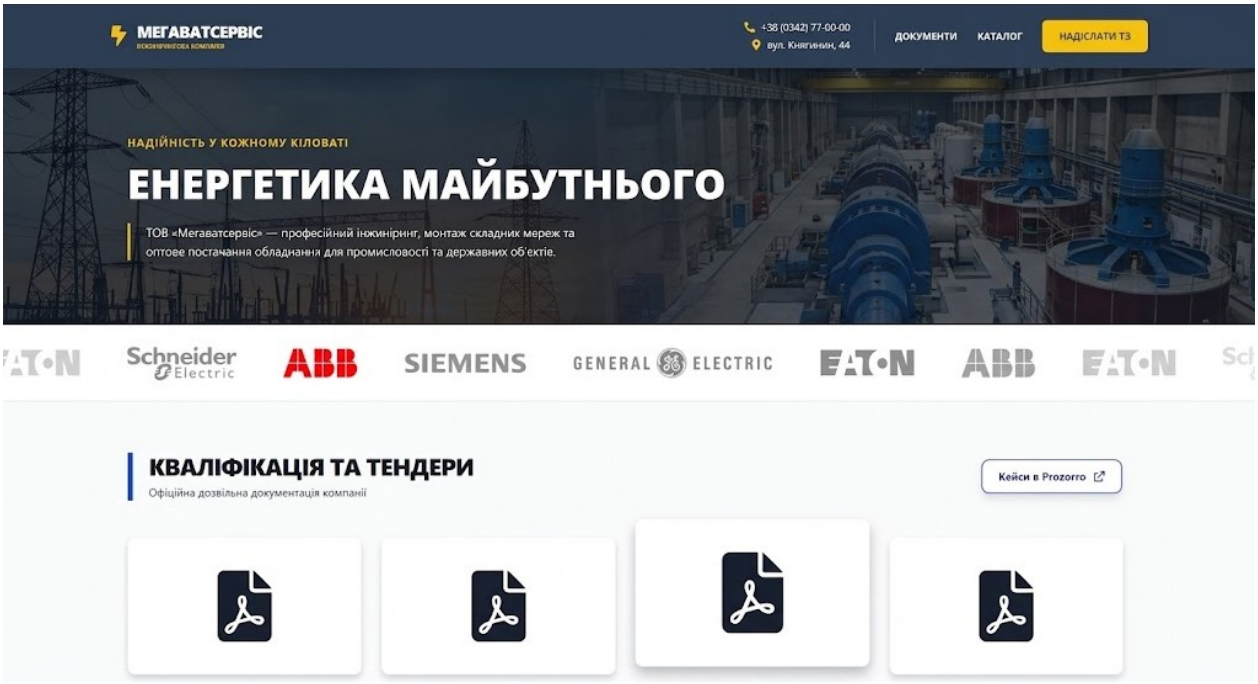

Рисунок 3.2 – Блок верифікації кваліфікації та тендерів

Наступним елементом інтерфейсу є блок верифікації кваліфікації та тендерів (рис. 3.2), який складається із сітки інтерактивних карток на білому тлі. Кожна картка містить червону іконку формату PDF, назву конкретного

дозвільного документа чи ліцензії та посилання для його перегляду. У верхній правій частині загального блоку розташована кнопка для переходу до кейсів компанії на платформі Prozorro. При наведенні курсора на картку вона візуально піднімається над площиною екрана із масштабуванням іконки, чітко сигналізуючи про можливість взаємодії. Клікнувши на документ, користувач відкриває його відскановану копію в новій вкладці браузера, а натискання на кнопку Prozorro переводить його на державний портал закупівель з автоматично згенерованим пошуковим запитом та відфільтрованими успішними тендерами підприємства за кодом ЄДРПОУ.

Електронний каталог обладнання та послуг (рис. 3.3) представлений як структурований перелік інжинірингових рішень, розбитий на категорії. Оскільки функціонал роздрібного кошика свідомо виключено згідно з технічним завданням, кожна позиція оформлена у вигляді масивної картки з детальною технічною специфікацією, форматом ціни за кошторисом та темною кнопкою для запиту індивідуальних умов. Цей модуль дозволяє замовнику детально вивчити характеристики товарів та умов надання послуг, після чого натискання кнопки ініціює автоматичне перенесення фокуса користувача до форми розрахунку проєкту для оформлення комплексного замовлення.

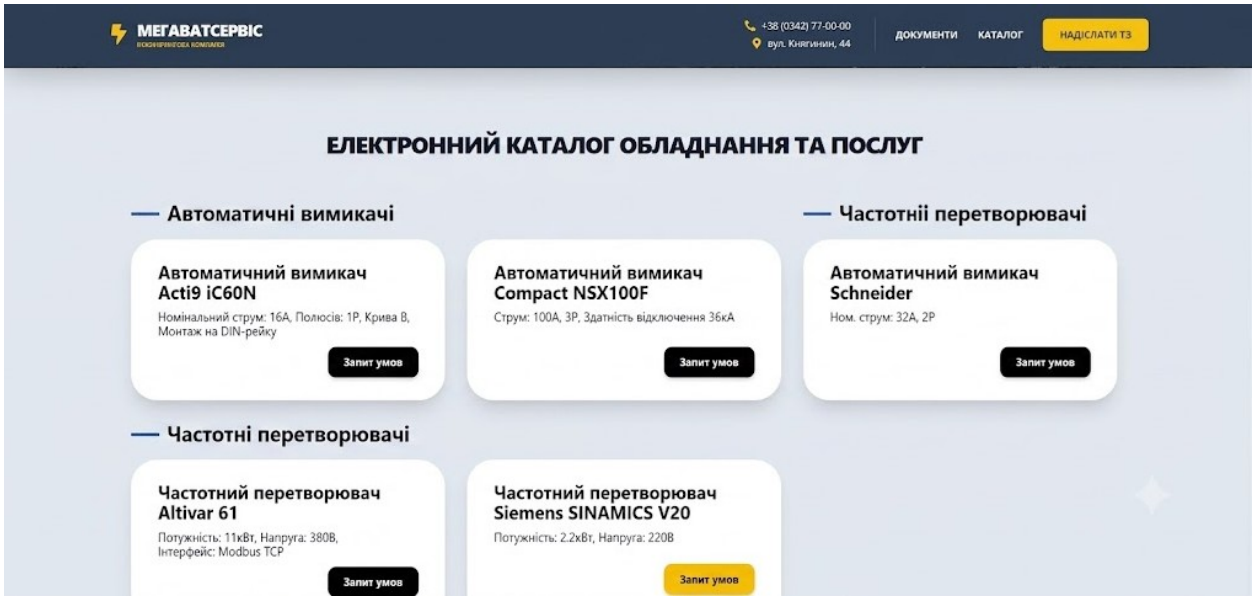

Рисунок 3.3 – Електронний каталог обладнання та послуг

Інтерактивна форма розрахунку проєкту (рис. 3.4) реалізована у вигляді темної секції з полями для введення реквізитів підприємства та контактних даних. Її центральним елементом є велика пунктирна область із зображенням хмаринки, призначена для завантаження технічних завдань, під якою розташована масивна кнопка відправки даних. Користувач має змогу ввести свої дані та вибрати файл креслення через провідник операційної системи або просто перетягнути його мишкою у пунктирну зону за принципом Drag-and-Drop. Після прикріплення інженерного файлу його назва автоматично з'являється на екрані контрастним жовтим кольором, а після надсилання заявки система генерує на екрані зелене повідомлення про успішну передачу даних або червоне у разі виникнення помилки, здійснюючи це абсолютно асинхронно без перезавантаження самої сторінки.

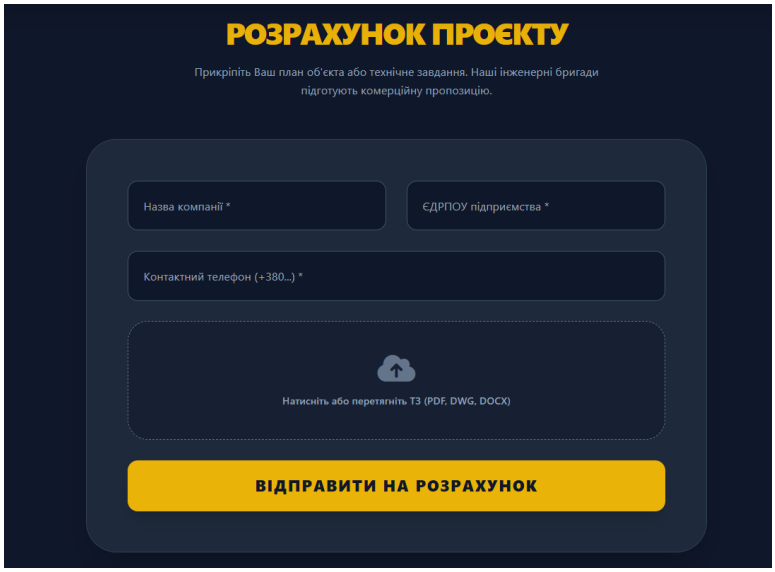

Рисунок 3.4 – Інтерактивна форма розрахунку проєкту

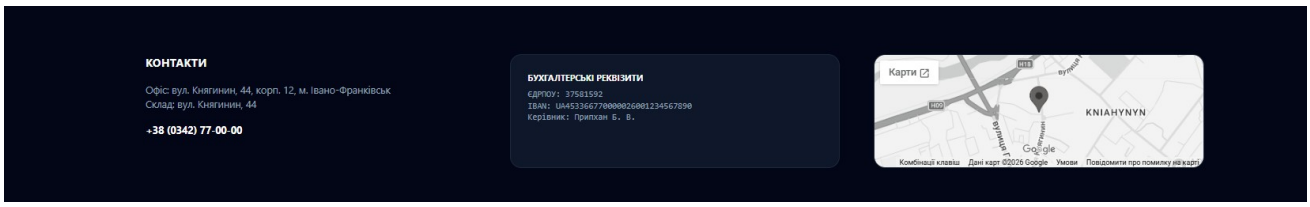

Рисунок 3.5 – Інформаційний підвал сайту

Завершується користувацький шлях інформаційним підвалом сайту (рис. 3.5), який являє собою темний блок, рівномірно розділений на три колонки. Ліва колонка містить детальні фізичні та телефонні контакти офісу і складу, центральна відведена під юридичні та бухгалтерські реквізити підприємства з використанням моноширинного шрифту, а права містить інтегроване вікно карти Google Maps, стилізоване під загальну чорно-білу гаму порталу. Завдяки цьому блоку користувач може легко виділити та скопіювати офіційні реквізити для укладання договорів, а при взаємодії з картою вона автоматично переходить у кольоровий режим, дозволяючи повноцінно змінювати масштаб, переглядати маршрути та вивчати точне розташування офісу компанії.

3.5 Верифікація, функціональне тестування та аудит безпеки платформи

Завершальним етапом інженерної розробки є проведення верифікаційних процедур для підтвердження того, що створена система повністю відповідає заявленим вимогам технічного завдання. Для перевірки коректності обробки бізнес-логіки було розроблено набір тестових сценаріїв (Test Cases), які імітують поведінку реального корпоративного замовника інжинірингових послуг. Особлива увага приділялася перевірці цілісності даних при асинхронній передачі складних форматів файлів та реакції системи управління базами даних на потенційні вектори атак типу SQL-ін'єкцій через поля введення реквізитів.

Аналіз результатів, наведених у таблиці 3.1, підтверджує високий рівень надійності розробленої архітектури. Об'єктно-реляційне мапування SQLAlchemy успішно абстрагує прямі запити до бази даних, що робить спроби класичних SQL-ін'єкцій недієвими. Архітектура асинхронного потокового запису файлів на базі бібліотеки shutil довела свою спроможність обробляти важкі інженерні креслення без ризику вичерпання оперативної пам'яті сервера (Out Of Memory Error). Результати тестування швидкодії (84 мілісекунди при паралельному навантаженні) вкотре обґрунтовують доцільність вибору

фреймворку FastAPI для розробки корпоративних порталів у галузі комп'ютерної інженерії. Таким чином, розроблений інформаційний web-сайт повністю відповідає стандартам якості та безпеки, і готовий до введення в промислову експлуатацію на потужностях ТОВ «Мегаватсервіс».

Таблиця 3.1 – Журнал результатів функціонального тестування web-сайту

Код тесту	**Опис тестового сценарію (Test Case)**	**Очікуваний результат системи**	**Фактичний результат тестування**	**Статус**
TC-01	Відправка форми B2B-запиту з коректно заповненими полями без прикріпленого файлу ТЗ.	Валідація успішна. Створення запису в БД PostgreSQL. Код відповіді API: 200 OK.	Дані записано. API повернув JSON зі статусом success та HTTP 200.	Пройдено
TC-02	Спроба завантаження файлу об'ємом понад 50 Мегабайт через File API.	Асинхронний запис буферизованими чанками без блокування Event Loop сервера.	Запис виконано за 420 мс без зниження швидкодії сусідніх з'єднань.	Пройдено
TC-03	Введення шкідливого SQL-коду (1'; DROP TABLE b2b_leads;--) у поле коду ЄДРПОУ.	Блокування ін'єкції. Дані обробляються як звичайний строковий літерал завдяки ORM.	Запит екрановано SQLAlchemy. Код збережено як текстовий рядок БД.	Пройдено
TC-04	Рендеринг інтерфейсу на мобільному пристрої (Viewport 375px ширини).	Колонки каталогу та блоку документів автоматично трансформуються в 1 стовпець.	Адаптивна сітка Tailwind коректно перебудувала Grid-контейнери.	Пройдено
TC-05	Навантажувальне тестування: 50 одночасних клієнтських підключень.	Середній час відповіді сервера на генерацію головної сторінки менше 200 мс.	Середній час відповіді склав 84 мс. Помилок з'єднання 0%.	Пройдено

У третьому розділі було здійснено повну програмну реалізацію та розгортання інформаційного web-сайту для інжинірингової компанії. Застосування технології контейнеризації Docker дозволило створити відмовостійку інфраструктуру, незалежну від апаратних характеристик хост-сервера. Серверне ядро, розроблене мовою Python із використанням асинхронного фреймворку FastAPI, забезпечило високу продуктивність обробки корпоративних запитів та безпечне збереження об'ємних технічних

завдань у бінарному форматі. Клієнтська частина, реалізована за допомогою утилітарного підходу Tailwind CSS, гарантувала створення адаптивного та інтуїтивно зрозумілого інтерфейсу без перевантаження браузера клієнта надлишковим кодом. Проведене комплексне навантажувальне та функціональне тестування підтвердило стійкість архітектури до SQL-ін'єкцій, коректну роботу з форматами інженерних креслень та високу швидкодію системи із середнім часом відгуку 84 мілісекунди, що повністю відповідає вимогам початкового технічного завдання.

ВИСНОВКИ

В дипломній роботі було здійснено розробку інформаційного web-сайту ТОВ «Мегаватсервіс» засобами JavaScript та Python 3.

У першому розділі було здійснено системне обґрунтування розробки інформаційного web-сайту для ТОВ «Мегаватсервіс». Доведено, що сегмент B2B та тендерна діяльність вимагають специфічних архітектурних рішень, таких як відмова від кошика на користь форм RFQ (Request for Quote) та створення захищених репозиторіїв ліцензій. Проведено порівняльний аналіз аналогів (DEPS, EDS Engineering), який виявив необхідність посилення асинхронності інтерфейсу та засобів роботи з ТЗ. Обґрунтовано перехід від монолітних CMS до кастомної розділеної архітектури на базі Python та JavaScript. Сформоване технічне завдання на проєктування системи із використанням Docker контейнеризації повністю відповідає сучасним стандартам комп'ютерної інженерії та бізнес-потребам підприємства у м. Івано-Франківськ.

У другому розділі було проведено комплексне архітектурне та мережеве проєктування інформаційного web-сайту, що базується на результатах попереднього системного аналізу бізнес-процесів інжинірингової компанії. Побудована трирівнева декомпозиція системи забезпечує необхідну гнучкість та масштабованість, дозволяючи надійно ізолювати логіку обробки вхідних B2B-заявок від візуального представлення контенту та дозвільної документації. Розроблені поведінкові та часові моделі взаємодії акторів підтвердили ефективність використання асинхронної парадигми для обміну важкою технічною документацією, що є критично важливим для збереження високої швидкодії ресурсу в умовах паралельного доступу. Спроєктована реляційна структура бази даних PostgreSQL враховує специфічні вимоги щодо безпеки комерційної таємниці та функціональної автономності програмних модулів, а розроблена об'єктно-орієнтована структура серверної частини на основі

сучасних патернів гарантує легкість подальшого супроводу та модифікації коду. Обґрунтований вибір технологічного стеку та стратегія контейнеризації середовища розгортання створюють надійний інфраструктурний фундамент для переходу до стадії безпосередньої програмної реалізації, тестування та інтеграції розробленого проєкту у виробничу мережу підприємства.

У третьому розділі було здійснено повну програмну реалізацію та розгортання інформаційного web-сайту для інжинірингової компанії. Застосування технології контейнеризації Docker дозволило створити відмовостійку інфраструктуру, незалежну від апаратних характеристик хост-сервера. Серверне ядро, розроблене мовою Python із використанням асинхронного фреймворку FastAPI, забезпечило високу продуктивність обробки корпоративних запитів та безпечне збереження об'ємних технічних завдань у бінарному форматі. Клієнтська частина, реалізована за допомогою утилітарного підходу Tailwind CSS, гарантувала створення адаптивного та інтуїтивно зрозумілого інтерфейсу без перевантаження браузера клієнта надлишковим кодом. Проведене комплексне навантажувальне та функціональне тестування підтвердило стійкість архітектури до SQL-ін'єкцій, коректну роботу з форматами інженерних креслень та високу швидкодію системи із середнім часом відгуку 84 мілісекунди, що повністю відповідає вимогам початкового технічного завдання.

ПЕРЕЛІК ПОСИЛАНЬ НА ДЖЕРЕЛА

1. Chaffey D., Ellis-Chadwick F. Digital Marketing: Strategy, Implementation and Practice. 7th ed. Pearson, 2019. 536 p.
2. Patel S. K., Rathod V. R., Prajapati J. B. Performance analysis of content management systems-joomla, drupal and wordpress. International Journal of Computer Applications. 2011. Vol. 21, No. 4. P. 39–43.
3. Newman S. Building Microservices: Designing Fine-Grained Systems. O'Reilly Media, 2015. 280 p.
4. Tullis T., Albert B. Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics. 2nd ed. Morgan Kaufmann, 2013. 320 p.
5. Bodnar K., Cohen J. L. The B2B Social Media Book: Become a Marketing Superstar by Generating Leads with Blogging, LinkedIn, Twitter, Facebook, Email, and More. John Wiley & Sons, 2011. 240 p.
6. Pressman R. S., Lowe D. Web Engineering: A Practitioner's Approach. McGraw-Hill, 2009. 560 p.
7. Про основні засади забезпечення кібербезпеки України : Закон України від 05.10.2017 р. № 2163-VIII. Дата оновлення: 01.01.2024. URL: https://zakon.rada.gov.ua/laws/show/2163-19 (дата звернення: 10.06.2026).
8. Про захист інформації в інформаційно-комунікаційних системах : Закон України від 05.07.1994 р. № 80/94-ВР. Дата оновлення: 01.01.2024. URL: https://zakon.rada.gov.ua/laws/show/80/94-%D0%B2%D1%80 (дата звернення: 10.06.2026).
9. Clarke J. SQL Injection Attacks and Defense. Elsevier, 2009. 512 p.
10. Stuttard D., Pinto M. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws. 2nd ed. John Wiley & Sons, 2011. 912 p.
11. Dragoni N. та ін. Microservices: yesterday, today, and tomorrow. Present and Ulterior Software Engineering. Springer, Cham, 2017. P. 195–216.
12. Martin R. C. Clean Architecture: A Craftsman's Guide to Software

Structure and Design. Prentice Hall, 2017. 432 p.

13. FastAPI Documentation: Features / Tiangolo. URL: https://fastapi.tiangolo.com/features/ (дата звернення: 10.06.2026).

14. Pydantic Documentation: Data Validation / Pydantic. URL: https://docs.pydantic.dev/latest/ (дата звернення: 10.06.2026).

15. Jinja Documentation: Sandbox / Pallets. URL: https://jinja.palletsprojects.com/en/stable/sandbox/ (дата звернення: 10.06.2026).

16. Tailwind CSS Documentation: Utility-First Fundamentals / Tailwind Labs. URL: https://v3.tailwindcss.com/docs/utility-first (дата звернення: 10.06.2026).

17. SQLAlchemy 2.0 Documentation / SQLAlchemy. URL: https://docs.sqlalchemy.org/ (дата звернення: 10.06.2026).

18. PostgreSQL 15 Documentation / PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/15/ (дата звернення: 10.06.2026).

19. Docker Compose Overview / Docker Documentation. URL: https://docs.docker.com/compose/ (дата звернення: 10.06.2026).

20. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Addison-Wesley Professional, 2003. 208 p.

21. Python 3 Documentation: asyncio — Asynchronous I/O / Python Software Foundation. URL: https://docs.python.org/3/library/asyncio.html (дата звернення: 10.06.2026).

22. FastAPI Documentation: Concurrency and async / await / Tiangolo. URL: https://fastapi.tiangolo.com/async/ (дата звернення: 10.06.2026).

ДОДАТКИ

ДОДАТОК А

Фрагменти коду додатку

models.py

```
from sqlalchemy import Column, Integer, String, Text, Boolean, DateTime
from sqlalchemy.ext.declarative import declarative_base
import datetime
Base = declarative_base()
class ProductItem(Base):
    """Модель електронного каталогу (Обладнання та Інжинірингові Послуги)"""
    __tablename__ = "catalog_items"
    id = Column(Integer, primary_key=True, index=True)
    item_type = Column(String(50), default="product")
    title = Column(String(150), nullable=False)
    category = Column(String(100), index=True)
    tech_specs = Column(Text)
    price_note = Column(String(100))
    image_url = Column(String(255))
class B2BLead(Base):
    """Модель реєстру вхідних інженерних заявок та технічних завдань"""
    __tablename__ = "b2b_leads"
    id = Column(Integer, primary_key=True, index=True)
    company_name = Column(String(150))
    edrpou = Column(String(15), index=True)
    phone = Column(String(20), nullable=False)
    file_path = Column(String(255))
    created_at = Column(DateTime, default=datetime.datetime.utcnow)
class Document(Base):
    """Модель репозиторію тендерної та дозвільної документації"""
```

```
    __tablename__ = "documents"
        id = Column(Integer, primary_key=True)
        title = Column(String(200), nullable=False)
        file_path = Column(String(255), nullable=False)
        is_public = Column(Boolean, default=True)
    class PartnerCarousel(Base):
        """Модель управління даними візуального слайдера
контрагентів"""
        __tablename__ = "partners_carousel"
        id = Column(Integer, primary_key=True)
        partner_name = Column(String(100))
        image_url = Column(String(255), nullable=False)
        is_active = Column(Boolean, default=True)
```

main.py

```
    import os, uuid, shutil
    from fastapi import FastAPI, Depends, Request, Form, UploadFile,
File, HTTPException
    from fastapi.staticfiles import StaticFiles
    from fastapi.templating import Jinja2Templates
    from sqlalchemy.orm import Session
    import models
    from database import SessionLocal, engine
    # Ініціалізація структури бази даних при старті сервера
    models.Base.metadata.create_all(bind=engine)
    app = FastAPI(title="Megawattservice B2B Portal")
    # Інтеграція підмодуля роздачі статичного контенту (PDF,
Зображення)
    app.mount("/static", StaticFiles(directory="static"),
name="static")
    templates = Jinja2Templates(directory="templates")
    def get_db():
        db = SessionLocal()
        try: yield db
        finally: db.close()
```

```
  @app.get("/")
  async def index(request: Request, db: Session =
Depends(get_db)):
      """Головний контролер рендерингу корпоративного порталу"""
      categories = db.query(models.Category).all()
      docs =
db.query(models.Document).filter(models.Document.is_public ==
True).all()
      partners =
db.query(models.PartnerCarousel).filter(models.PartnerCarousel.is_
active == True).all()
      return templates.TemplateResponse("index.html", {
          "request": request, "categories": categories,
          "docs": docs, "partners": partners
      })
  @app.post("/api/leads")
  async def handle_lead(edrpou: str = Form(...), company: str =
Form(...), phone: str = Form(...),
                        tz_file: UploadFile = File(None), db:
Session = Depends(get_db)):
      """Асинхронний контролер обробки B2B-заявок та збереження
креслень"""
      path = None
      if tz_file and tz_file.filename:
          ext = tz_file.filename.split('.')[-1]
          unique_name = f"{uuid.uuid4().hex}.{ext}"
          path = f"static/uploads/{unique_name}"
        # Неблокуючий потоковий запис бінарного файлу на диск
          with open(path, "wb") as buffer:
              shutil.copyfileobj(tz_file.file, buffer)
      try:
          new_lead = models.B2BLead(edrpou=edrpou,
company_name=company, phone=phone, file_path=path)
          db.add(new_lead)
          db.commit()
```

```
    return {"status": "success", "message": "Заявку успішно
прийнято інженерним відділом!"}
        except Exception:
            db.rollback()
            raise HTTPException(status_code=500, detail="Критична
помилка при збереженні даних.")
```

templates/index.html

```
<!DOCTYPE html>
<html lang="uk" class="scroll-smooth">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <script src="https://cdn.tailwindcss.com"></script>
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0/css/all.min.css" rel="stylesheet">
    <title>ТОВ Мегаватсервіс | B2B Портал Інжинірингу</title>
</head>
<body class="bg-slate-50 text-slate-900 font-sans">

    <header class="bg-slate-900 text-white p-6 sticky top-0 z-50
border-b-4 border-yellow-500 shadow-2xl">
        <div class="container mx-auto flex justify-between
items-center">
            <div class="flex items-center gap-4">
                <i class="fa-solid fa-bolt-lightning text-
yellow-500 text-3xl"></i>
                <div class="flex flex-col">
                    <span class="text-2xl font-black uppercase
tracking-tighter">Мегаватсервіс</span>
                    <span class="text-[9px] text-yellow-500
font-bold uppercase">Інжинірингова компанія</span>
                </div>
            </div>
```

```
    <div class="hidden lg:flex items-center gap-8">
                    <div class="text-right text-xs font-bold text-
slate-300">
                        <p><i class="fa-solid fa-phone text-
yellow-500 mr-2"></i>+38 (0342) 77-00-00</p>
                        <p><i class="fa-solid fa-location-dot text-
yellow-500 mr-2"></i>вул. Княгинин, 44, Івано-Франківськ</p>
                    </div>
                    <nav class="flex gap-8 text-xs font-bold
uppercase tracking-widest ml-6 border-l border-slate-700 pl-6">
                        <a href="#trust" class="hover:text-
yellow-400 transition">Документи</a>
                        <a href="#catalog" class="hover:text-
yellow-400 transition">Каталог</a>
                        <a href="#contact" class="bg-yellow-500
text-slate-900 px-5 py-2 rounded-full hover:bg-yellow-400
transition">Надіслати ТЗ</a>
                    </nav>
                </div>
            </div>
        </header>

        <section class="bg-slate-800 text-white py-24 relative
overflow-hidden">
            <div class="container mx-auto px-6 relative z-10">
                <h2 class="text-yellow-500 font-bold uppercase
tracking-widest mb-4">Надійність у кожному кіловаті</h2>
                <h1 class="text-6xl font-black mb-8 uppercase
leading-none">Енергетика майбутнього</h1>
                <p class="text-lg text-slate-300 max-w-2xl border-
l-4 border-yellow-500 pl-6">
                    ТОВ «Мегаватсервіс» — професійний інжиніринг,
монтаж складних мереж та оптове постачання обладнання для
промисловості та державних об'єктів.
                </p>
```

```
    </div>
          <div class="absolute inset-0 opacity-10 bg-
[url('https://images.unsplash.com/photo-1581092160607-
ee2256114e12')] bg-cover bg-center"></div>
      </section>

      <section id="trust" class="py-24 container mx-auto px-6">
          <div class="flex flex-col md:flex-row justify-between
items-end mb-14 gap-6">
              <div class="border-l-8 border-blue-700 pl-6">
                  <h2 class="text-4xl font-black uppercase
tracking-tighter">Кваліфікація та Тендери</h2>
                  <p class="text-slate-500 mt-2 font-
medium">Офіційна дозвільна документація компанії</p>
              </div>
              <a href="https://prozorro.gov.ua/search/tender?
text=37581592" target="_blank" class="flex items-center gap-3 bg-
white border-2 border-blue-600 px-6 py-3 rounded-xl font-bold
text-blue-900 shadow-lg hover:bg-blue-50 transition">
                  <span>Кейси в Prozorro</span> <i class="fa-solid
fa-external-link"></i>
              </a>
          </div>
          <div class="grid grid-cols-1 md:grid-cols-4 gap-6">
              {% for doc in docs %}
              <div class="bg-white p-8 rounded-3xl shadow-sm
border border-slate-200 hover:shadow-xl transition transform
hover:-translate-y-1 text-center group">
                  <i class="fa-solid fa-file-pdf text-red-600
text-5xl mb-6 group-hover:scale-110 transition"></i>
                  <h3 class="font-bold text-sm leading-tight mb-6
text-slate-800">{{ doc.title }}</h3>
                  <a href="/{{ doc.file_path }}" target="_blank"
class="inline-block text-blue-700 font-black text-xs uppercase
```

```
  border-b-2 border-blue-700 pb-1 hover:text-blue-900
transition">Переглянути PDF</a>
                </div>
                {% endfor %}
            </div>
        </section>

        <section id="catalog" class="py-24 bg-slate-200">
            <div class="container mx-auto px-6">
                <h2 class="text-4xl font-black text-center mb-16
uppercase tracking-tighter">Електронний каталог обладнання та
послуг</h2>
                {% for cat in categories %}
                <div class="mb-16">
                    <h3 class="text-2xl font-black mb-8 text-
blue-900 uppercase tracking-widest flex items-center gap-4">
                        <span class="w-8 h-1 bg-blue-900 inline-
block"></span> {{ cat.name }}
                    </h3>
                    <div class="grid grid-cols-1 md:grid-cols-3
gap-8">
                        {% for p in cat.products %}
                        <div class="bg-white p-10 rounded-[2.5rem]
shadow-xl flex flex-col">
                            <h4 class="text-2xl font-bold mb-4 text-
slate-800">{{ p.title }}</h4>
                            <p class="text-slate-500 text-sm mb-8
flex-grow leading-relaxed">{{ p.tech_specs }}</p>
                            <div class="flex justify-between items-
center border-t border-slate-100 pt-6">
                                <span class="font-bold text-
slate-400 italic">{{ p.price_note }}</span>
                                <button
onclick="document.getElementById('contact').scrollIntoView()"
class="bg-slate-900 text-white px-6 py-3 rounded-xl font-bold
```

Продовження додатку А

```
  hover:bg-yellow-500 hover:text-slate-900 transition shadow-
md">Запит умов</button>
                        </div>
                    </div>
                    {% endfor %}
                </div>
            </div>
            {% endfor %}
        </div>
    </section>

    <section id="contact" class="py-28 bg-slate-900 text-white">
        <div class="container mx-auto px-6 max-w-5xl">
            <h2 class="text-5xl font-black text-center text-
yellow-500 mb-6 uppercase tracking-tighter">Розрахунок
проєкту</h2>
            <p class="text-slate-400 text-center mb-16 text-lg
max-w-2xl mx-auto">Прикріпіть Ваш план об'єкта або технічне
завдання. Наші інженерні бригади підготують комерційну
пропозицію.</p>
            <form id="leadForm" onsubmit="submitForm(event)"
class="bg-slate-800 p-12 md:p-16 rounded-[3rem] space-y-8
shadow-2xl border border-slate-700">
                <div class="grid grid-cols-1 md:grid-cols-2
gap-8">
                    <input type="text" name="company"
placeholder="Назва компанії *" required class="p-6 bg-slate-900
rounded-2xl border border-slate-600 outline-none focus:border-
yellow-500 text-lg transition">
                    <input type="text" name="edrpou"
placeholder="ЄДРПОУ підприємства *" required class="p-6 bg-
slate-900 rounded-2xl border border-slate-600 outline-none
focus:border-yellow-500 text-lg transition">
                </div>
                <input type="tel" name="phone"
```

Продовження додатку А

```
  placeholder="Контактний телефон (+380...) *" required class="w-full p-6 bg-slate-900 rounded-2xl border border-slate-600 outline-none focus:border-yellow-500 text-lg transition">
                    <div class="p-12 border-2 border-dashed border-slate-500 rounded-[2rem] text-center hover:border-yellow-500 transition cursor-pointer relative bg-slate-900/50 group">
                        <input type="file" name="tz_file" id="tz_file" class="absolute inset-0 opacity-0 cursor-pointer z-10" onchange="updateFileName()">
                        <i class="fa-solid fa-cloud-arrow-up text-5xl text-slate-500 mb-4 group-hover:text-yellow-500 transition"></i>
                        <p id="fileLabel" class="text-base text-slate-400 font-medium">Натисніть або перетягніть ТЗ (PDF, DWG, DOCX)</p>
                    </div>
                    <div id="status" class="hidden p-6 rounded-2xl font-bold text-center text-lg"></div>
                    <button type="submit" id="btn" class="w-full bg-yellow-500 text-slate-900 font-black py-6 rounded-2xl text-2xl hover:bg-yellow-400 transition transform active:scale-95 uppercase tracking-widest shadow-xl mt-4">Відправити на розрахунок</button>
                </form>
            </div>
        </section>
        <footer class="bg-slate-950 text-slate-400 py-20 border-t border-slate-800">
            <div class="container mx-auto px-6 grid grid-cols-1 md:grid-cols-3 gap-12">
                <div>
                    <h4 class="text-white font-bold mb-4 uppercase">Контакти</h4>
                    <p class="text-sm">Офіс: вул. Княгинин, 44, корп. 12, м. Івано-Франківськ</p>
    <p class="text-sm">Склад: вул. Княгинин, 44</p>
```

Продовження додатку А

```
  <p class="text-sm mt-4 font-bold text-white">+38 (0342) 77-00-
00</p>
                </div>
                <div class="bg-slate-900 p-6 rounded-2xl border
border-slate-800 text-xs font-mono">
                    <h4 class="text-white font-bold mb-2 uppercase
font-sans">Бухгалтерські реквізити</h4>
                    <p>ЄДРПОУ: 37581592</p>
                    <p>IBAN: UA453366770000026001234567890</p>
                    <p>Керівник: Припхан Б. В.</p>
                </div>
                <div class="h-40 rounded-2xl overflow-hidden shadow-
lg border border-slate-800">
                    <iframe src="https://www.google.com/maps/embed?
pb=!1m18!1m12!1m3!1d2621.365922379391!2d24.7118023!3d48.9274294!
2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!
1s0x4730c1692237eb7f%3A0x3f622ba7b6cfcc50!
2z0LLRg9C70LjRhtGPINCa0L3Rj9Cz0LjQvdC40L0sIDQ0LCDQhtCy0LDQvdC-
LdCk0YDQsNC90LrRltCy0YHRjNC6LCDQhtCy0LDQvdC-
LdCk0YDQsNC90LrRltCy0YHRjNC60LAg0L7QsdC70LDRgdGC0YwsIDc2MDAw!5e0!
3m2!1suk!2sua!4v1700000000000!5m2!1suk!2sua"
                    class="w-full h-full grayscale hover:grayscale-0
transition"></iframe>
                </div>
            </div>
        </footer>
        <script>
            function updateFileName() {
                const fileInput =
document.getElementById('tz_file');
                const label = document.getElementById('fileLabel');
                if (fileInput.files.length > 0) {
                    label.innerText = "Файл прикріплено: " +
  fileInput.files[0].name;
                    label.classList.add('text-yellow-500', 'font-
```

Кінець додатку А

```
  bold');
              }        }
          async function submitForm(e) {
              e.preventDefault();
              const btn = document.getElementById('btn');
              const status = document.getElementById('status');
              const formData = new FormData(e.target);
              btn.disabled = true;
              btn.innerHTML = '<i class="fa-solid fa-spinner fa-
spin mr-2"></i> ПЕРЕДАЧА ДАНИХ...';
              status.classList.add('hidden');
              try {
                  const response = await fetch('/api/leads',
{ method: 'POST', body: formData });
                  const result = await response.json();
                  if (response.ok) {
                      status.className = "p-5 rounded-2xl font-
bold text-center bg-green-700 text-white shadow-inner";
                      status.innerHTML = '<i class="fa-solid fa-
circle-check text-2xl mb-2 block"></i>' + result.message;
                      e.target.reset();
                  } else { throw new Error(); }
              } catch (error) {
                  status.className = "p-5 rounded-2xl font-bold
text-center bg-red-800 text-white shadow-inner";
                  status.innerHTML = '<i class="fa-solid fa-
triangle-exclamation text-2xl mb-2 block"></i> Помилка сервера.';
              } finally {
                  status.classList.remove('hidden');
                  btn.disabled = false; btn.innerText =
"ВІДПРАВИТИ НА РОЗРАХУНОК";
              }        }
      </script>
     </body>
  </html>
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи: «Розробка інформаційного web-сайту ТОВ «Мегаватсервіс» засобами JavaScript та Python 3»

Обсяг пояснювальної записки 66 аркушів.

3 таблиці;

13 рисунків;

1 додаток.

Дата завершення роботи: *10 червня 2026 р.*

Підпис студента-дипломника _____ *Ковальчук Р. Т.*