Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: проєкт архітектурної реалізації та макет веб-орієнтованої crm-системи на основі шаблонів фреймворку vue

Виконав: студент VI курсу, групи СНм-61
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

(підпис) Кібук Р.М. (прізвище та ініціали)

Керівник (підпис) НикитюкВ.В. (прізвище та ініціали)

Нормоконтроль (підпис) (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
« 17 » листопада 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Кібук Ростислав Максимович
(прізвище, ім'я, по батькові)

1. Тема роботи проєкт архітектурної реалізації та макет веб-орієнтованої crm-системи на основі шаблонів фреймворку vue

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

Затверджені наказом ректора від «17» листопада 2025 року № __

2. Термін подання студентом завершеної роботи 22 грудня 2025 р.

3. Вихідні дані до роботи Наукові публікації про ... для ... збору та аналізу сигналів ...

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз предметної області та обґрунтування вибору. 2 Проєктування та програмна реалізація інформаційної системи....
3 Обчислювальний експеримент: розгортання та верифікація функціональності crm-системи.
4 Охорона праці та безпека в надзвичайних ситуаціях. Висновки. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Титульна сторінка. 2 Тема, Мета, Об'єкт, Предмет дослідження. 3 Завдання дослідження. 4 Актуальність дослідження. 5 Технологічний стек розробки. 6 Архітектура та рух даних 7 Практична реалізація: Канбан-дошка 8 Інфраструктура та контейнеризація 9 Оптимізація .. та результати експерименту. 10 Висновки. 11 Завершальний слайд.
...
...
...

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Сенчишин В.С., доцент кафедри МТ	Згідно календар-ного плану	Згідно календар-ного плану
Безпека в надзвичайних ситуаціях	Теслюк В.М., проректор з адміністративно-господарської роботи та будівництва	Згідно календар-ного плану	Згідно календар-ного плану

7. Дата видачі завдання 17 листопада 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	17.11.2025	*Виконано*
2.	Аналіз науково-технічних публікацій та збір даних по	17.11.2025-24.11.2025	*Виконано*
	темі кваліфікаційної роботи		
3.	Виконання дослідження згідно мети кваліфікаційної	25.11.2025-08.12.2025	*Виконано*
	роботи		
4.	Оформлення розділу «Аналіз предметної області та обґрунтування вибору»	09.12.2025-11.12.2025	*Виконано*
	...		
	...		
5.	Оформлення розділу «Проєктування та програмна реалізація інформаційної системи	09.12.2025-11.12.2025	*Виконано*
	...		
	...»		
6.	Оформлення розділу «Обчислювальний експеримент: розгортання та верифікація функціональності crm-системи	09.12.2025-11.12.2025	*Виконано*
	...		
	...»		
7.	Виконання завдання до підрозділу «Охорона праці»	25.12.2025-08.12.2025	*Виконано*
8.	Виконання завдання до підрозділу «Безпека в	25.12.2025-08.12.2025	*Виконано*
	надзвичайних ситуаціях»		
9.	Оформлення кваліфікаційної роботи	12.12.2025	*Виконано*
10.	Нормоконтроль	15.12.2025-16.12.2025	*Виконано*
11.	Перевірка на плагіат	17.12.2025	*Виконано*
12.	Попередній захист кваліфікаційної роботи	18.12.2025	*Виконано*
13.	Захист кваліфікаційної роботи	23.12.2025	

Студент _____ Кібук Р.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Никитюк В.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка CRM-системи для управління бізнес-процесами в малих IT-командах // Кваліфікаційна робота освітнього ступеня «Магістр» // Кібук Ростислав Максимович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНм-61 // Тернопіль, 2025 // С. 67, рис. – 16, табл. – 2, додат. –3, бібліогр. – 22.

Ключові слова:crm-система, vue.js, pinia, supabase, postgresql, docker, канбан-дошка, автоматизація бізнес-процесів.

Кваліфікаційна робота присвячена розробці сучасної інформаційної системи управління взаємовідносинами з клієнтами (CRM), адаптованої під потреби малих IT-команд, з використанням реактивних вебтехнологій та інструментів контейнеризації.

В першому розділі кваліфікаційної роботи описані теоретичні засади функціонування CRM-систем та їх роль у цифровій трансформації сучасного бізнесу. Висвітлено специфіку бізнес-процесів у сфері розробки програмного забезпечення. Розглянуто існуючі аналоги на ринку та обґрунтовано доцільність розробки індивідуального продукту. Проаналізовано технологічний стек, зокрема переваги використання фреймворку Vue.js 3 та хмарних рішень Supabase.

В другому розділі кваліфікаційної роботи спроєктовано архітектуру системи на базі Single Page Application (SPA). Досліджено реляційну модель бази даних у середовищі PostgreSQL та налаштовано політики безпеки Row Level Security. Подано опис логіки управління глобальним станом додатка за допомогою Pinia та розроблено адаптивний користувацький інтерфейс на основі бібліотеки Element Plus із впровадженням темної теми.

В третьому розділі кваліфікаційної роботи описано процес розгортання системи в ізольованих Docker-контейнерах на віддаленому сервері. Проаналізовано показники продуктивності системи за допомогою інструменту

Google Lighthouse та підтверджено ефективність алгоритму конвертації зображень у формат WebP. Проведено комплексне функціональне тестування модулів системи, що підтвердило стабільність роботи Real-time синхронізації даних.

Об’єкт дослідження: процес управління взаємовідносинами з клієнтами та операційної діяльності в малих IT-компаніях.

Предмет дослідження: методи, алгоритми та програмні засоби автоматизації CRM-процесів із використанням реактивних frontend-фреймворків та контейнеризованих інфраструктур.

ANNOTATION

Development of a CRM System for Business Process Management in Small IT Teams // Master's Degree Qualification Work // Kibuk Rostyslav Maksymovych // Ternopil National Technical University named after Ivan Puluj, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, Group SNm-61 // Ternopil, 2025 // P. 67, fig. – 16, tab. – 2, app. – 3, bibl. – 22.

Key words: crm system, vue.js, pinia, supabase, postgresql, docker, kanban board, business process automation.

The qualification work is devoted to the development of a modern customer relationship management (CRM) system adapted to the needs of small IT teams using reactive web technologies and containerization tools.

In the first chapter of the qualification work, the theoretical foundations of CRM systems and their role in the digital transformation of modern business are described. The specifics of business processes in the field of software development are highlighted. Existing market analogues are reviewed, and the feasibility of developing an individual product is justified. The technological stack, specifically the advantages of using the Vue.js 3 framework and Supabase cloud solutions, is analyzed.

In the second chapter of the qualification work, the system architecture based on the Single Page Application (SPA) model is designed. The relational database model in the PostgreSQL environment is investigated, and Row Level Security (RLS) policies are configured. A description of the global application state management logic using Pinia is provided, and an adaptive user interface based on the Element Plus library with dark mode implementation is developed.

In the third chapter of the qualification work, the process of system deployment in isolated Docker containers on a remote server is described. System performance indicators are analyzed using the Google Lighthouse tool, and the efficiency of the WebP image conversion algorithm is confirmed. Comprehensive functional testing of

the system modules was performed, confirming the stability of real-time data synchronization.

Object of research: the process of customer relationship management and operational activities in small IT companies.

Subject of research: methods, algorithms, and software tools for automating CRM processes using reactive frontend frameworks and containerized infrastructures.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – База даних.

БЖД – Безпека життєдіяльності.

ІТ – Інформаційні технології.

ПЗ – Програмне забезпечення.

СУБД – Система управління базами даних.

API (англ. Application Programming Interface) – прикладний програмний інтерфейс.

CRM (англ. Customer Relationship Management) – система управління взаємовідносинами з клієнтами.

DOM (англ. Document Object Model) – об'єктна модель документа.

RLS (англ. Row Level Security) – безпека на рівні рядків бази даних.

SPA (англ. Single Page Application) – односторінковий вебзастосунок.

UI (англ. User Interface) – інтерфейс користувача.

UX (англ. User Experience) – досвід користувача.

ЗМІСТ

ВСТУП 11
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РОЗРОБКИ 14
1.1 Роль CRM-систем у цифровізації бізнес-процесів 14
1.2 Порівняльний аналіз існуючих програмних рішень на ринку та обґрунтування розробки індивідуального продукту 16
1.3 Обґрунтування вибору архітектурних рішень та технологічного стека розробки 18
1.3.1 Використання технології контейнеризації Docker 18
1.3.2 Frontend-фреймворк: Vue.js 3 (Composition API) 19
1.3.3 Керування станом додатка: Pinia 19
1.3.4 Backend-as-a-Service: Supabase та PostgreSQL 20
1.4 Специфікація функціональних та нефункціональних вимог до інформаційної системи 20
1.4.1 Функціональні вимоги до системи 21
1.4.2 Нефункціональні вимоги та системні обмеження 22
1.4.3 Технічні обмеження розробки 23
1.5 Висновок до першого розділу 23
2 ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ 25
2.1 Архітектурна побудова системи та схема взаємодії компонентів 25
2.2 Проєктування реляційної моделі бази даних 26
2.2.1 Опис структури таблиць та зв'язків (ER-діаграма) 26
2.2.2 Забезпечення цілісності та безпеки даних 27
2.3 Розробка логіки управління станом за допомогою Pinia 27
2.3.1 Концепція централізованого сховища як «Єдиного джерела істини» 28
2.3.2 Внутрішня структура та життєвий цикл сховища crm.js 28
2.3.3 Реалізація асинхронної взаємодії та Real-time синхронізації 29

2.3.4 Обробка помилок та станів завантаження 30
2.4 Проєктування інтерфейсу користувача (UI) та аналіз досвіду взаємодії (UX) 30
2.4.1 Філософія дизайну та вибір колірної палітри 31
2.4.2 Використання бібліотеки компонентів Element Plus 32
2.4.3 Реалізація плавних переходів та мікро-взаємодій 32
2.5 Інфраструктурне забезпечення та контейнеризація за допомогою Docker 33
2.5.1 Обґрунтування використання контейнерів у розробці 33
2.5.2 Проєктування файлів конфігурації (Dockerfile та Docker Compose) 34
2.6 Оптимізація медіаконтенту та продуктивність системи 35
2.7 Висновок до другого розділу 36
3 ОБЧИСЛЮВАЛЬНИЙ ЕКСПЕРИМЕНТ: РОЗГОРТАННЯ ТА ВЕРИФІКАЦІЯ ФУНКЦІОНАЛЬНОСТІ CRM-СИСТЕМИ 38
3.1 Методологія проведення експерименту та підготовка середовища розгортання 38
3.1.1 Вибір та конфігурування серверної інфраструктури 38
3.1.2 Опис процесу контейнеризації та підготовки образів 39
3.2 Налаштування продуктового середовища Supabase та конфігурація Real-time доступу 39
3.2.1 Міграція схеми бази даних та управління секретами 39
3.2.2 Верифікація механізмів безпеки (RLS) 40
3.3 Тестування продуктивності та оптимізація медіа-контенту 41
3.3.1 Аналіз швидкості завантаження за допомогою Google Lighthouse 41
3.3.2 Експериментальна перевірка ефективності формату WebP 41
3.4 Функціональне тестування модулів системи та верифікація бізнес-логіки 42
3.4.1 Розробка плану тестування (Test Plan) 42
3.4.2 Протоколювання результатів тестування 45

3.5 Аналіз ергономіки та стабільності інтерфейсу в експериментальних умовах 46
3.5.1 Тестування плавності інтерфейсних анімацій (Transitions)..... 47
3.5.2 Адаптивність та кросбраузерна верифікація..... 47
3.6 Оцінка безпеки та цілісності даних при роботі з Docker-контейнерами 47
3.6.1 Тестування Docker Volumes..... 48
3.6.2 Перевірка захисту від несанкціонованого доступу 48
3.7 Висновок до третього розділу..... 48
4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ 50
4.1 Питання щодо охорони праці..... **Помилка! Закладку не визначено.**
4.2 Питання щодо безпеки в надзвичайних ситуаціях 53
4.3 Висновок до четвертого розділу..... 57
ВИСНОВКИ..... 58
ПЕРЕЛІК ДЖЕРЕЛ..... 60
ДОДАТКИ

ВСТУП

Актуальність теми. У сучасних умовах глобальної цифровізації успішність функціонування будь-якого підприємства, незалежно від його масштабів, напряму залежить від ефективності управління внутрішніми та зовнішніми бізнес-процесами. Одним із критично важливих аспектів діяльності є взаємодія з клієнтами. Системи класу CRM (Customer Relationship Management) стали стандартом для великих корпорацій, проте для малих IT-команд та фріланс-об'єднань існуючі рішення часто є економічно недоцільними або функціонально надмірними.

Необхідність розробки кастомної CRM-системи для малих команд обумовлена потребою у гнучкому, швидкому та інтуїтивно зрозумілому інструменті, який би поєднував у собі функції ведення бази клієнтів, управління угодами (Kanban) та візуалізацію фінансових показників. Використання сучасних технологій, таких як Vue.js 3 для інтерфейсу, Supabase для хмарного збереження даних та Docker для контейнеризації середовища розробки, дозволяє створити продукт із високою швидкістю відгуку та мінімальними витратами на підтримку інфраструктури.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи ступеня «Магістр» є розробка та впровадження інформаційної системи управління взаємовідносинами з клієнтами, адаптованої під потреби малих інноваційних команд, з акцентом на візуалізацію робочих процесів та забезпечення стабільності розгортання через контейнеризацію, зокрема:

- Провести аналіз предметної області та існуючих програмних рішень на ринку CRM-систем.
- Обґрунтувати вибір технологічного стека (Vue 3, Pinia, Supabase) та інструментів розгортання (Docker).
- Спроектувати архітектуру бази даних та схему взаємодії компонентів системи.
- Розробити користувацький інтерфейс із підтримкою темної теми та адаптивності для різних пристроїв.

- Реалізувати функціональні модулі управління угодами, послугами та аналітичними дашбордами.
- Налаштувати систему автоматизованого розгортання додатка в ізольованих контейнерах.

Об’єкт дослідження процес управління взаємовідносинами з клієнтами та операційної діяльності в малих IT-компаніях.

Предмет дослідження. методи, алгоритми та програмні засоби автоматизації CRM-процесів із використанням реактивних frontend-фреймворків та serverless-рішень для баз даних.

Наукова новизна одержаних результатів кваліфікаційної роботи полягає в удосконаленні архітектурних підходів до побудови легких CRM-систем для малих інноваційних команд. Зокрема:

- Отримав подальший розвиток метод інтеграції реактивних Frontend-фреймворків (Vue.js 3) із хмарними Serverless-рішеннями (Supabase), що дозволило реалізувати синхронізацію даних у реальному часі без необхідності розробки традиційного серверного прошарку.
- Запропоновано адаптивну модель візуалізації життєвого циклу IT-проєктів на основі Канбан-інтерфейсу, яка враховує специфічні ролі користувачів (Admin, Dev) та забезпечує гнучке управління статусами угод в умовах обмежених ресурсів команди.
- Дістала подальший розвиток методика контейнеризації вебдодатків на основі Docker для забезпечення ідентичності середовищ розробки та експлуатації, що мінімізує ризики несумісності бібліотек та пришвидшує процес розгортання системи.

Практичне значення одержаних результатів. Розроблено та впроваджено повноцінний прототип CRM-системи з інтерактивною Канбан-дошкою та аналітичним дашбордом. Налаштовано автоматизоване середовище розгортання в ізольованих Docker-контейнерах, що забезпечує стабільність роботи та легке масштабування проєкту.

Апробація результатів магістерської роботи. Основні результати дослідження щодо використання сучасного стека вебтехнологій (Vue 3,

Supabase) та методів контейнеризації (Docker) обговорювались на IV міжнародній студентській науково-технічній конференції «Природничі та гуманітарні науки. Актуальні питання» (ТНТУ ім. І. Пулюя, м. Тернопіль, 2025 р.).

Публікації. Основні результати кваліфікаційної роботи опубліковано у двох працях конференції (Див. додатки А).

Структура й обсяг кваліфікаційної роботи. Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку літератури з 50 найменувань та 2 додатків. Загальний обсяг кваліфікаційної роботи складає 63 сторінки, з них 45 сторінки основного тексту, який містить 14 рисунків та 17 таблиць.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РОЗРОБКИ

1.1 Роль CRM-систем у цифровізації бізнес-процесів

У сучасних умовах глобалізації економічних процесів та стрімкого розвитку інформаційних технологій, парадигма ведення бізнесу зазнає фундаментальних змін. Традиційні методи управління, що ґрунтувалися на продуктоцентричному підході, поступово витісняються стратегіями, де центральне місце посідає клієнт. У цьому контексті системи управління взаємовідносинами з клієнтами стають не просто програмним забезпеченням, а базовим елементом стратегічного розвитку будь-якого сучасного підприємства[1].

Термін CRM охоплює комплексне поняття, яке слід розглядати у трьох взаємопов'язаних аспектах:

- Як стратегія: це бізнес-філософія, що ставить потреби замовника у фокус всієї діяльності компанії. Вона спрямована на максимізацію лояльності та задоволеності клієнтів шляхом персоналізації взаємодії.
- Як процес: це сукупність процедур та регламентів, що описують кожен крок взаємодії з контрагентом — від першого маркетингового контакту до післяпродажного сервісу.
- Як технологія: це програмний продукт, який забезпечує збір, зберігання та інтелектуальну обробку даних про клієнтів, транзакції та комунікації.

Фундаментальною метою впровадження CRM-рішень є підвищення показника Customer Lifetime Value (CLV) — сукупного прибутку, який компанія отримує від клієнта за весь час співпраці. Це досягається шляхом мінімізації людського фактора, автоматизації рутинних операцій та створення єдиного інформаційного простору.

Цифрова трансформація бізнесу — це процес глибокої реорганізації бізнес-процесів із використанням цифрових активів. CRM-системи виконують

роль "ядра" цієї трансформації, оскільки вони дозволяють перевести хаотичну інформацію про продажі у структуру Big Data.

Впровадження сучасної CRM-системи, особливо в сегменті електронної комерції (наприклад, для платформ на базі OpenCart чи WordPress, з якими активно працюють розробники), дозволяє реалізувати наступні переваги цифрової епохи:

- Створення Single Source of Truth (SSOT): формування єдиного джерела достовірних даних, що виключає дублювання інформації та суперечності між різними відділами компанії.
- Інтелектуалізація продажів: використання накопичених даних для прогнозування попиту та автоматичного формування комерційних пропозицій.
- Перехід до омніканальності: забезпечення безперервного клієнтського досвіду незалежно від каналу зв'язку (соціальні мережі, пошта, телефон чи чат-боти).

З наукової та практичної точок зору, архітектура CRM-систем базується на трьох функціональних блоках:

1. Операційний блок: забезпечує підтримку щоденної діяльності. Це ведення бази клієнтів, управління угодами (Deals Management), генерація рахунків та автоматизація маркетингових розсилок.

2. Аналітичний блок: відповідає за обробку статистичних даних. Тут формуються звіти про воронку продажів, ефективність менеджерів та рентабельність окремих послуг. Це критично важливо для прийняття управлінських рішень на основі реальних цифр, а не інтуїції.

3. Колабораційний блок: спрямований на забезпечення безперебійної комунікації між учасниками бізнес-процесу. Це внутрішні чати, системи коментування завдань та інтеграція з сервісами для спільної роботи.

Таким чином, CRM-система виступає як критично важливий інструмент адаптації бізнесу до вимог цифрової економіки. Вона дозволяє малим командам та великим корпораціям масштабувати свої процеси, зберігаючи при цьому персоналізований підхід до кожного замовника.

1.2 Порівняльний аналіз існуючих програмних рішень на ринку та обґрунтування розробки індивідуального продукту

Сучасний ринок інформаційних технологій пропонує широкий спектр готових інструментів для управління бізнес-процесами, які варіюються від простих хмарних сервісів до складних корпоративних екосистем. Проте, вибір між використанням готового комерційного продукту (**SaaS**) та розробкою власного програмного забезпечення залишається одним із найскладніших стратегічних питань для будь-якої організації.

Для об’єктивного обґрунтування необхідності розробки власної системи, було проведено дескриптивне дослідження найбільш популярних рішень, представлених на ринку:

1. Salesforce (Корпоративний стандарт): Дана платформа є беззаперечним лідером глобального ринку. Вона пропонує практично необмежені можливості для автоматизації маркетингу, продажів та сервісу. Проте, для малої IT-команди Salesforce є надлишковою. Показник TCO (Total Cost of Ownership — сукупна вартість володіння) у даному випадку є економічно недоцільним через високу вартість ліцензування та складність конфігурування, що потребує залучення сертифікованих спеціалістів.

2. Bitrix24 (Мультифункціональне комбайн-рішення): Система пропонує широкий набір інструментів: від телефонії до складського обліку. Основною проблемою Bitrix24 є так званий "Software Bloat" — надмірність функціоналу, яка негативно впливає на ергономіку інтерфейсу. Користувач змушений взаємодіяти з десятками модулів, які не використовуються в щоденній роботі, що призводить до підвищення когнітивного навантаження та зниження продуктивності праці.

3. Trello / Jira (Інструменти управління завданнями): Ці продукти ідеально підходять для візуалізації розробки за методологією Agile (Kanban), проте вони не є повноцінними CRM-системами. У них відсутня можливість ведення фінансової звітності в розрізі клієнтів, немає структурованої бази контрагентів та аналітичних дашбордів для моніторингу рентабельності бізнесу в цілому.

4. Pipedrive (Орієнтована на продажі): Це якісне рішення для відділів продажів, яке фокусується на воронці (Pipeline). Однак воно працює за закритим кодом і має обмежені можливості для візуальної кастомізації інтерфейсу, що є важливим для IT-фахівців, які надають перевагу специфічним режимам відображення (наприклад, Dark Mode).

Використання сторонніх SaaS-платформ неминуче призводить до виникнення ризику Vendor Lock-in — повної залежності від постачальника послуг. Це включає:

- Ціновий диктат: розробник сервісу може змінити вартість підписки в будь-який момент.
- Безпека даних: конфіденційна інформація про клієнтів та угоди зберігається на сторонніх серверах, що може суперечити політиці безпеки або вимогам GDPR.
- Технічні обмеження: неможливість впровадження нестандартних функцій, які критично важливі для конкретної бізнес-моделі (наприклад, інтеграція зі специфічними локальними платіжними шлюзами чи API розробника).

Розробка власної CRM-системи в межах дипломного проекту базується на принципах "Lean Development" — створенні продукту, який містить лише необхідний функціонал (Minimum Viable Product — MVP), але реалізований на найвищому рівні якості.

Ключові фактори на користь власної розробки:

- Оптимізація UI/UX: створення інтерфейсу, який ідеально відповідає робочому процесу розробника. Використання бібліотеки Element Plus дозволяє реалізувати сучасний, лаконічний дизайн із високою швидкістю відгуку.
- Технологічна незалежність: використання стека Vue.js + Supabase забезпечує високу продуктивність системи при мінімальних витратах на серверне обслуговування (модель Serverless).
- Масштабованість: архітектура системи проектується з урахуванням можливості подальшого розширення функціоналу без необхідності повної перебудови ядра.

- Інтеграція аналітики: впровадження індивідуальних графіків та лічильників (Dashboard), які відображають саме ті KPI (Key Performance Indicators), що важливі для власника малого бізнесу.

Таким чином, індивідуальна розробка дозволяє створити інструмент, який не просто копіює існуючі рішення, а пропонує унікальний баланс між простотою використання, функціональною достатністю та технологічною досконалістю, що є критично важливим для малих інноваційних команд.

1.3 Обґрунтування вибору архітектурних рішень та технологічного стека розробки

Вибір інструментарію для реалізації інформаційної системи є одним із найбільш відповідальних етапів проєктування, оскільки він визначає не лише швидкість розробки, а й подальшу масштабованість, безпеку та вартість підтримки продукту. Для даної CRM-системи було обрано стек технологій, що базується на принципах модульності, реактивності та контейнеризації.

1.3.1 Використання технології контейнеризації Docker

Важливою особливістю розробки даної системи є використання Docker — платформи для автоматизації розгортання та управління додатками в середовищах із підтримкою контейнеризації. У дипломному проєкті впровадження Docker обґрунтовано наступними чинниками:

- Ізоляція середовища: Docker дозволяє упакувати додаток з усіма його залежностями, бібліотеками та конфігураціями в єдиний образ (Image), що гарантує ідентичність роботи системи на локальній машині розробника, тестовому сервері та в продакшн-середовищі.
- Консистентність розробки: використання Docker Compose дозволяє однією командою піднімати повний стек необхідних сервісів (frontend, допоміжні модулі конвертації WebP, налаштовані PHP-розширення), що

критично важливо для командної розробки та швидкого залучення нових фахівців.

- Оптимізація ресурсів: на відміну від традиційних віртуальних машин, контейнери Docker використовують спільне ядро ОС, що робить їх значно легшими, швидшими при запуску та ефективнішими у використанні оперативної пам'яті сервера.
- Спрощення CI/CD процесів: контейнеризований додаток набагато легше інтегрувати в процеси безперервної інтеграції та доставки, що є стандартом для сучасних IT-компаній.

1.3.2 Frontend-фреймворк: Vue.js 3 (Composition API)

Для створення клієнтської частини обрано Vue.js 3. Це прогресивний JavaScript-фреймворк, який дозволяє будувати інтерфейси на основі компонентного підходу.

- Реактивність та Virtual DOM: Vue використовує ефективний алгоритм порівняння віртуального дерева елементів, що дозволяє оновлювати лише ті частини інтерфейсу, які реально змінилися. Це забезпечує високу швидкість роботи навіть при великій кількості активних угод на екрані.
- Composition API: даний підхід дозволяє групувати логіку за функціональними особливостями, а не за типами опцій, що робить код більш читабельним та полегшує його повторне використання у великих масштабах.

1.3.3 Керування станом додатка: Pinia

В архітектурі Single Page Application (SPA) критично важливо мати єдине джерело істини для даних. Pinia — це сучасна бібліотека керування станом (State Management), яка прийшла на зміну Vuex.

- Модульність: вона дозволяє створювати окремі сховища (stores) для різних сутностей (наприклад, crm.js для угод та користувачів), що спрощує архітектуру.

- Типізація та DevTools: Pinia забезпечує чудову підтримку інструментів розробника, що дозволяє відстежувати кожну зміну в стані системи в реальному часі.

1.3.4 Backend-as-a-Service: Supabase та PostgreSQL

Як серверне рішення обрано Supabase, що надає потужний інструментарій на базі реляційної СУБД PostgreSQL.

- Реляційна цілісність: PostgreSQL дозволяє створювати складні зв'язки між таблицями (ForeignKey), забезпечуючи надійність даних про клієнтів та фінансові транзакції.
- Real-time можливості: завдяки вбудованій підтримці WebSockets, система може отримувати оновлення про зміну статусів угод миттєво, без необхідності перезавантаження сторінки користувачем.
- Безпека (RLS): Row Level Security дозволяє прописувати правила доступу до даних безпосередньо на рівні бази, що гарантує, що розробник не отримає доступу до конфіденційної фінансової інформації компанії.

1.4 Специфікація функціональних та нефункціональних вимог до інформаційної системи

Процес проектування будь-якої складної інформаційної системи, якою є CRM, починається з чіткого визначення вимог. Це дозволяє встановити межі проекту, уникнути розмиття функціоналу та сформувати еталон, за яким буде оцінюватися успішність розробки на фінальному етапі. Вимоги поділяються на дві основні категорії: ті, що описують безпосередні дії системи (функціональні), та ті, що визначають якість і умови її роботи (нефункціональні).

1.4.1 Функціональні вимоги до системи

Функціональні вимоги визначають поведінку системи та перелік операцій, які користувач може виконувати в межах інтерфейсу. Для даної CRM-системи виділено наступні ключові модулі:

Модуль автентифікації та керування доступом (IAM):

- Система повинна забезпечувати надійну реєстрацію та вхід користувачів через захищені канали зв'язку.
- Реалізація моделі RBAC (Role-Based Access Control) для розмежування прав: адміністратор (повний доступ до фінансів та налаштувань), менеджер (робота з клієнтами та угодами), розробник (перегляд завдань та зміна статусів у межах призначених проектів).

Модуль управління угодами (Deals Management):

- Візуалізація життєвого циклу угоди за допомогою інтерактивної Канбан-дошки.
- Можливість динамічної зміни етапів угоди («Нова», «В роботі», «Перевірка», «Завершено») шляхом перетягування карток (Drag-and-Drop).
- Зберігання детальної інформації про кожну угоду: опис, бюджет, призначений виконавець, історія змін та вкладені файли.

Модуль управління клієнтською базою (CRM Core):

- Централізоване сховище даних про контрагентів (назва компанії, контактна особа, історія звернень).
- Інтеграція системи пошуку та фільтрації клієнтів за різними критеріями для швидкого доступу до інформації.

Модуль аналітики та звітності (Business Intelligence):

- Побудова графіків у реальному часі (за допомогою бібліотеки Chart.js) для візуалізації фінансових надходжень.
- Відображення статистики завантаженості команди: кількість активних завдань на одного розробника.
- Автоматичний розрахунок потенційного доходу на основі угод, що знаходяться в роботі.

1.4.2 Нефункціональні вимоги та системні обмеження

Нефункціональні вимоги описують атрибути якості системи, такі як продуктивність, безпека та зручність використання.

Продуктивність та швидкість відгуку:

- Час завантаження основних сторінок системи (Dashboard, Deals) не повинен перевищувати 1.5 - 2 секунди[2].
- Використання механізмів Lazy Loading для компонентів та зображень (наприклад, конвертація у WebP) для мінімізації трафіку.
- Миттєве оновлення даних в інтерфейсі через WebSockets (за допомогою Supabase Realtime).

Надійність та безпека даних:

- Забезпечення цілісності даних на рівні бази PostgreSQL за допомогою транзакцій та обмежень (Constraints).
- Шифрування трафіку за допомогою протоколу SSL/TLS для запобігання атакам типу "Man-in-the-Middle".
- Реалізація політик Row Level Security (RLS) у Supabase, що гарантує ізоляцію даних між різними ролями користувачів.

Вимоги до розгортання та інфраструктури:

- Система повинна бути повністю контейнеризована за допомогою Docker для забезпечення ідентичності середовищ розробки та продакшну.
- Використання Docker Compose для оркестрації сервісів (frontend, backend-proxy, допоміжні утиліти).
- Сумісність із сучасними браузерами (Chrome, Firefox, Safari, Edge) останніх версій.

Ергономіка та інтерфейс (UI/UX):

- Адаптивний дизайн (Responsive Design) для коректної роботи на різних типах пристроїв (десктопи, планшети).
- Наявність темної теми для зниження навантаження на очі користувачів під час тривалої роботи в нічний час.

- Плавність інтерфейсних анімацій (Transitions) для зменшення когнітивного навантаження під час навігації меню та згортання бічної панелі.

1.4.3 Технічні обмеження розробки

Для забезпечення стабільності та підтримки коду в проекті встановлено наступні обмеження:

1. Контроль версій: Весь код проекту повинен зберігатися в системі Git, а розробка вестиметься з використанням IDE (наприклад, PhpStorm) для контролю якості коду.

2. Середовище розробки: Локальне розгортання здійснюється виключно в Docker-контейнерах для уникнення проблем із версіями системних бібліотек.

3. Керування станом: Весь глобальний стан програми повинен бути інкапсульований у сторах Pinia без прямого маніпулювання даними в компонентах.

1.5 Висновок до першого розділу

У першому розділі було проведено всебічний аналіз предметної області та обґрунтовано необхідність розробки спеціалізованої CRM-системи. За результатами проведеного дослідження можна зробити наступні висновки:

1. Проаналізовано роль CRM-систем у цифровій трансформації бізнесу та встановлено, що для малих IT-команд існуючі комерційні рішення (Salesforce, Bitrix24) є надмірно складними та економічно недоцільними.

2. Обґрунтовано вибір технологічного стека, що базується на використанні фреймворку Vue.js 3 та хмарної платформи Supabase. Таке поєднання забезпечує високу швидкість розробки, реактивність інтерфейсу та стабільну роботу з даними в режимі реального часу.

3. Доведено доцільність використання Docker як основного інструменту для контейнеризації додатка. Це дозволяє ізолювати середовище розробки,

гарантувати ідентичність роботи системи на різних серверах та спростити процес розгортання.

4. Визначено ключові функціональні вимоги до системи, серед яких пріоритетними є візуалізація угод за допомогою Канбан-дошки, наявність аналітичного дашборду та підтримка темної теми для комфортної роботи користувачів.

5. Встановлено ергономічні критерії, що передбачають використання бібліотеки компонентів Element Plus для створення сучасного, адаптивного та інтуїтивно зрозумілого UI/UX дизайну.

Таким чином, сформований теоретичний фундамент та визначені технічні вимоги дозволяють перейти до наступного етапу — проєктування архітектури бази даних та програмної реалізації модулів системи.

2 ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Архітектурна побудова системи та схема взаємодії компонентів

Проєктування сучасної CRM-системи вимагає чіткого розподілу обов'язків між різними рівнями додатка(див. рисунок 2.1). В основі розробки лежить архітектурний шаблон SPA (Single Page Application), який забезпечує динамічне оновлення інтерфейсу без повного перезавантаження сторінки користувачем.

Архітектура системи складається з трьох ключових рівнів:

1. Рівень представлення (Frontend): побудований на базі фреймворку Vue.js 3. Він відповідає за візуалізацію даних, обробку подій користувача та валідацію форм.

2. Рівень управління станом (Data Management): реалізований за допомогою бібліотеки Pinia. Це проміжний шар, який акумулює дані з сервера та розповсюджує їх між компонентами, забезпечуючи реактивність інтерфейсу.

3. Рівень даних та безпеки (Backend/BaaS): хмарна платформа Supabase, що виконує функції реляційної бази даних PostgreSQL та сервісу автентифікації.

Окремим архітектурним рішенням є використання Docker для ізоляції інфраструктури. Система розгортається у вигляді набору контейнерів, де кожен сервіс працює у власному ізольованому середовищі, що виключає конфлікти залежностей та спрощує перенесення проєкту між різними серверами.

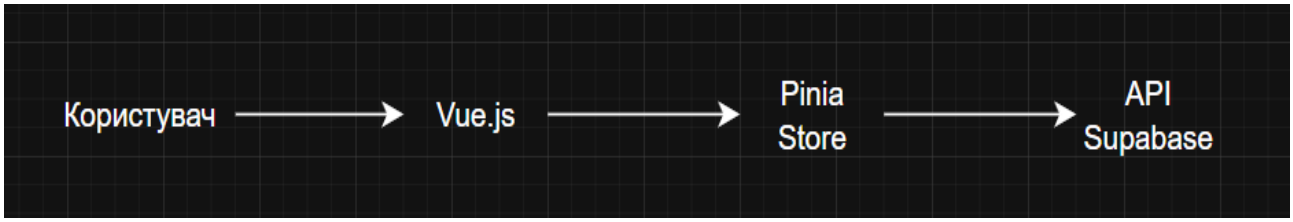


Рисунок 2.1 – Архітектурна схема системи

2.2 Проєктування реляційної моделі бази даних

Центральним елементом системи є база даних, архітектура якої повинна забезпечувати цілісність інформації та швидкий доступ до неї. Для реалізації CRM обрано об'єктно-реляційну СУБД PostgreSQL, що працює в межах платформи Supabase.

2.2.1 Опис структури таблиць та зв'язків (ER-діаграма)

Логічна структура бази даних базується на кількох ключових сутностях, які відображають реальні бізнес-процеси IT-команди:

1. Таблиця users (Користувачі): зберігає облікові дані співробітників (email, хешований пароль) та їхні системні ролі. Поля: id (UUID), full_name (string), role (enum: admin, dev), avatar_url (url).

2. Таблиця clients (Клієнти): містить інформацію про замовників. Поля: id, name, contact_person, phone, email, birthday (дата народження для модуля привітань).

3. Таблиця deals (Угоди): основна таблиця, що описує стан комерційних проєктів. Поля: id, title, amount (бюджет), status (статус канбану), client_id (ForeignKey до таблиці clients).

4. Таблиця services (Послуги): каталог послуг, які надає команда. Поля: id, name, price, description.

На рисунку 2.2 зображена ER-діаграма бази данних яку я створював для цієї кваліфікаційної роботи[3].

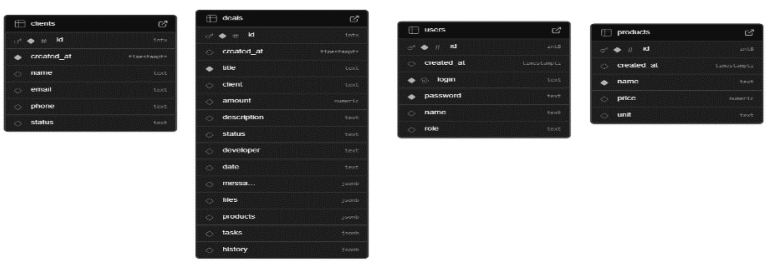


Рисунок 2.2 – ER-діаграма бази данних

2.2.2 Забезпечення цілісності та безпеки даних

Для захисту інформації на рівні бази даних впроваджено механізм Row Level Security (RLS). Це дозволяє налаштувати політики доступу так, щоб користувач із роллю developer міг бачити лише ті угоди, до яких він залучений, у той час як користувач із роллю admin має повний доступ до фінансової звітності.

Також на рівні СУБД реалізовано автоматичне конвертування зображень у формат WebP при завантаженні аватарів чи логотипів, що оптимізує швидкість завантаження інтерфейсу.

2.3 Розробка логіки управління станом за допомогою Pinia

У сучасних вебдодатках, що будуються за архітектурою SPA (Single Page Application), одним із найскладніших завдань є синхронізація даних між різними частинами інтерфейсу. Оскільки CRM-система передбачає одночасну роботу з багатьма сутностями (угоди, клієнти, послуги, користувачі), виникає потреба в імплементації централізованого механізму управління станом (State Management). Для реалізації цієї задачі у проєкті використано бібліотеку Pinia — офіційний інструмент для Vue.js, що реалізує концепцію Flux-архітектури.

На рисунку 2.3 зображенна діаграма ієрархії компонентів Vue.js

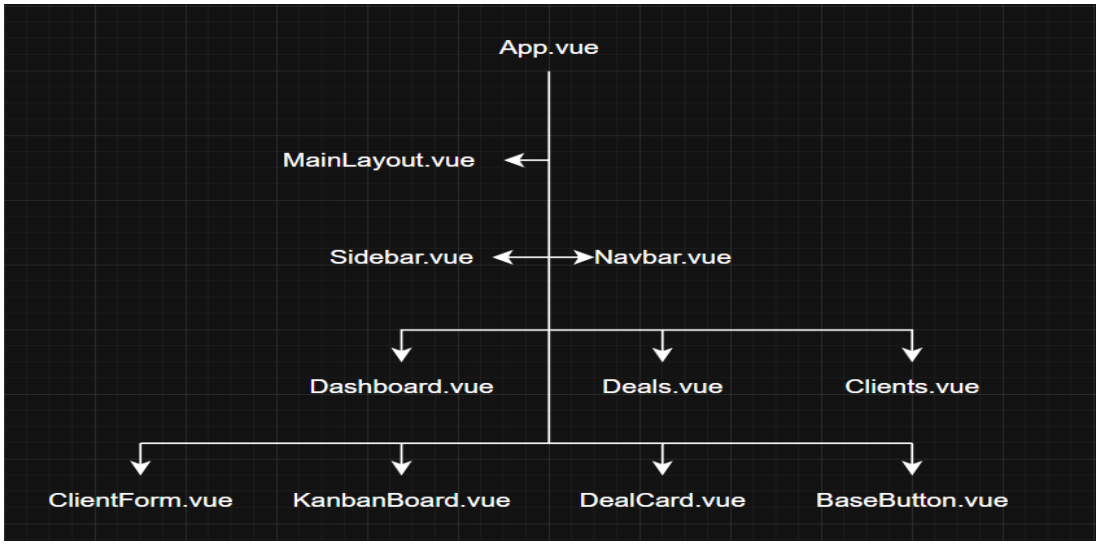


Рисунок 2.2 – Діаграма ієрархії компонентів Vue.js

На Рисунку 2.2 представлена ієрархічна структура фронтенд-частини системи. Використання компонентного підходу дозволяє інкапсулювати логіку та стилі кожного елемента інтерфейсу, що значно спрощує підтримку та тестування коду. Кожен вузол діаграми представляє собою окремий Single File Component (SFC), який взаємодіє з батьківськими елементами через вхідні параметри (props) та з дочірніми — через події (emits). Це забезпечує модульність архітектури та можливість повторного використання коду.

2.3.1 Концепція централізованого сховища як «Єдиного джерела істини»

Використання Pinia дозволяє реалізувати принцип Single Source of Truth (SSOT). Це означає, що дані, отримані з бази даних Supabase, не зберігаються локально в кожному окремому компоненті, а акумулюються в глобальному сховищі (Store). Це вирішує низку критичних проблем:

- Усунення дублювання запитів: якщо дані про клієнта потрібні одночасно на сторінці «Угоди» та в модальному вікні редагування, система виконує лише один мережевий запит, зберігаючи результат у сторі.
- Миттєва реактивність: будь-яка зміна стану (наприклад, видалення послуги) автоматично відображається у всіх компонентах, які підписані на цей стан, без необхідності перезавантаження сторінки.
- Передбачуваність змін: зміна даних відбувається через чітко визначені методи (Actions), що дозволяє легко відстежувати логіку роботи програми під час налагодження.

2.3.2 Внутрішня структура та життєвий цикл сховища crm.js

Логіка управління даними в CRM-системі інкапсульована в основному модулі crm.js, який структурований за трьома базовими принципами Pinia:

1. State (Стан): Це реактивний об’єкт, де зберігаються «сирі» дані. У нашому проєкті стан містить масиви deals (угоди), clients (клієнти), services (послуги), а також індикатори завантаження (loading) та об’єкти помилок (error).

2. Getters (Геттери): Це обчислювальні властивості, які дозволяють отримувати похідні дані зі стану. Наприклад, геттер getDealsByStatus автоматично фільтрує масив угод для кожної колонки Канбан-дошки, а геттер totalRevenue миттєво розраховує сумарний бюджет усіх активних проєктів для відображення на дашборді.

3. Actions (Дії): Асинхронні методи, що відповідають за бізнес-логіку та взаємодію з зовнішнім API (Supabase).

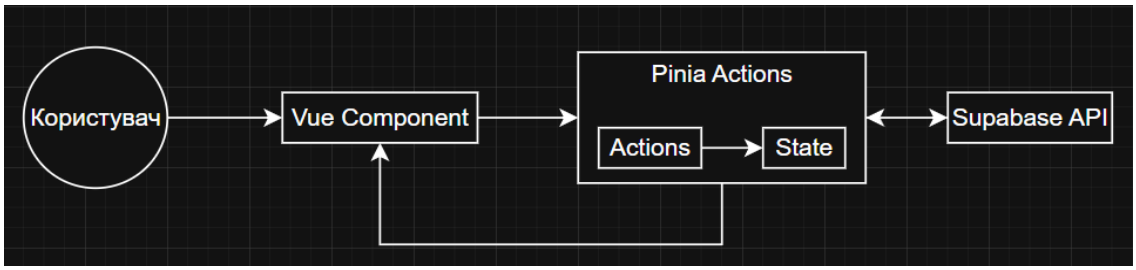


Рисунок 2.2 – Діаграма станів та потоків даних Pinia (Store)

На Рисунку 2.3 представлено архітектурний патерн управління станом системи. Процес починається з дії користувача у Vue-компоненті, що ініціює виклик асинхронного методу в Pinia Store. Action виконує мережевий запит до Supabase через API, після чого отримує підтвердження успішної операції. Отримані дані фіксуються в об'єкті State, що завдяки системі реактивності Vue 3 автоматично спричиняє перемальовування компонентів, які відображають актуальні дані про угоди чи клієнтів. Такий підхід забезпечує цілісність інформації та дозволяє легко масштабувати функціонал системи.

2.3.3 Реалізація асинхронної взаємодії та Real-time синхронізації

Ключовою технічною перевагою розробленого сховища є його тісна інтеграція з сервісами Supabase. Actions у Pinia використовують конструкцію async/await для виконання запитів до бази даних PostgreSQL.

Окрему увагу в роботі приділено механізму Real-time Subscriptions. Усередині методу ініціалізації стору налаштовано прослуховування подій від Supabase Realtime Channel. Це працює за наступним алгоритмом:

- При запуску додатка стор підписується на події INSERT, UPDATE та DELETE для таблиці deals.
- Як тільки будь-який користувач системи вносить зміни (наприклад, перетягує картку в інший статус), сервер надсилає push-повідомлення через WebSocket.
- Action у сторі отримує це повідомлення і миттєво оновлює локальний масив deals.
- Завдяки реактивності Vue 3, інтерфейс у всіх активних користувачів оновлюється плавно та синхронно, що є критично важливим для командної роботи.

2.3.4 Обробка помилок та станів завантаження

Для покращення користувацького досвіду (UX) у логіку стору впроваджено систему перехоплення виключень (Exception Handling). Кожен асинхронний запит загорнутий у блок try...catch. У разі виникнення помилки (наприклад, проблеми зі зв'язком або відсутність прав доступу RLS), повідомлення про помилку зберігається в стані error і автоматично виводиться користувачеві через UI-компоненти Element Plus. Також використання прапорця loading дозволяє системі відображати скелетони або індикатори завантаження під час виконання мережевих операцій, що робить інтерфейс більш відзивчивим.

2.4 Проєктування інтерфейсу користувача (UI) та аналіз досвіду взаємодії (UX)

У сучасних інформаційних системах управління бізнес-процесами візуальна складова та зручність навігації відіграють не меншу роль, ніж функціональна логіка. Процес розробки інтерфейсу для даної CRM-системи

базувався на принципах User-Centered Design (UCD), де кожен елемент керування проєктувався з урахуванням когнітивного навантаження на кінцевого користувача.

2.4.1 Філософія дизайну та вибір колірної палітри

Основним завданням при проєктуванні дизайну було створення середовища, яке сприяє тривалій концентрації уваги. Оскільки цільовою аудиторією системи є IT-фахівці, які проводять за монітором значну кількість часу, особливу увагу було приділено розробці темної теми (Dark Mode).

- Колірна схема: за основу обрано глибокі відтінки сірого та чорного кольорів (наприклад, #0b0c0d), що мінімізує випромінювання синього світла та зменшує втому очей.
- Акцентні кольори: для виділення активних елементів, таких як кнопки «Створити угоду» або активні пункти меню, використано корпоративний синій колір (--primary-color), який асоціюється з надійністю та стабільністю.
- Типографіка: використано сучасні беззасічкові шрифти (Sans-serif), які забезпечують високу читабельність тексту навіть при малих кеглях у таблицях та списках.

Вигляд у світлій та темній темах можна переглянути на рисунках 2.3 та 2.4

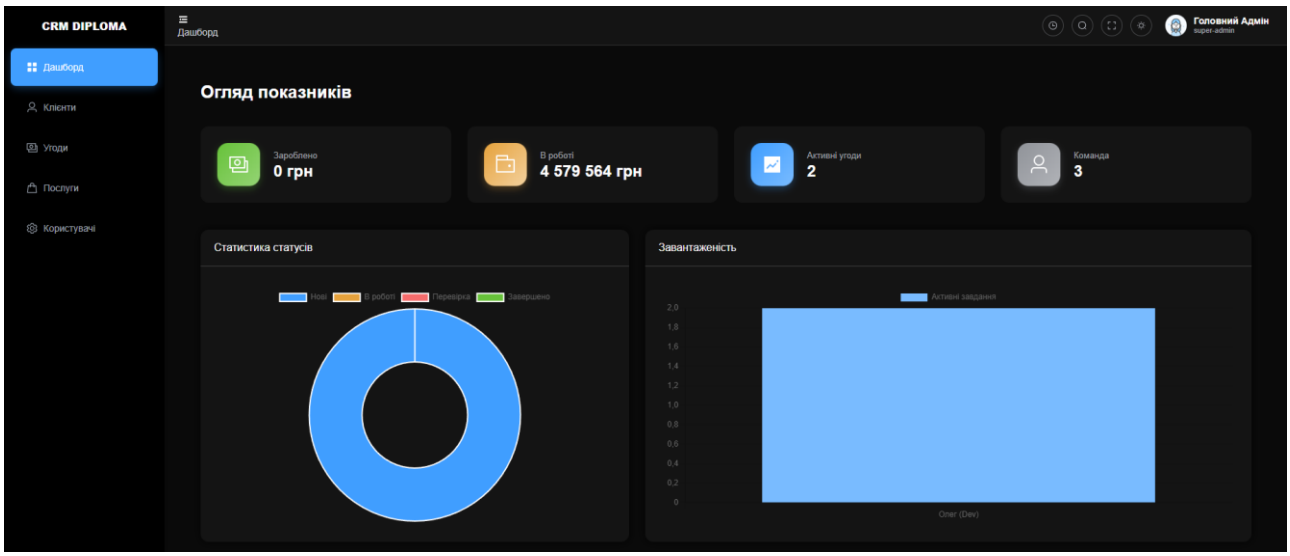


Рисунок 2.3 – Темна тема CRM

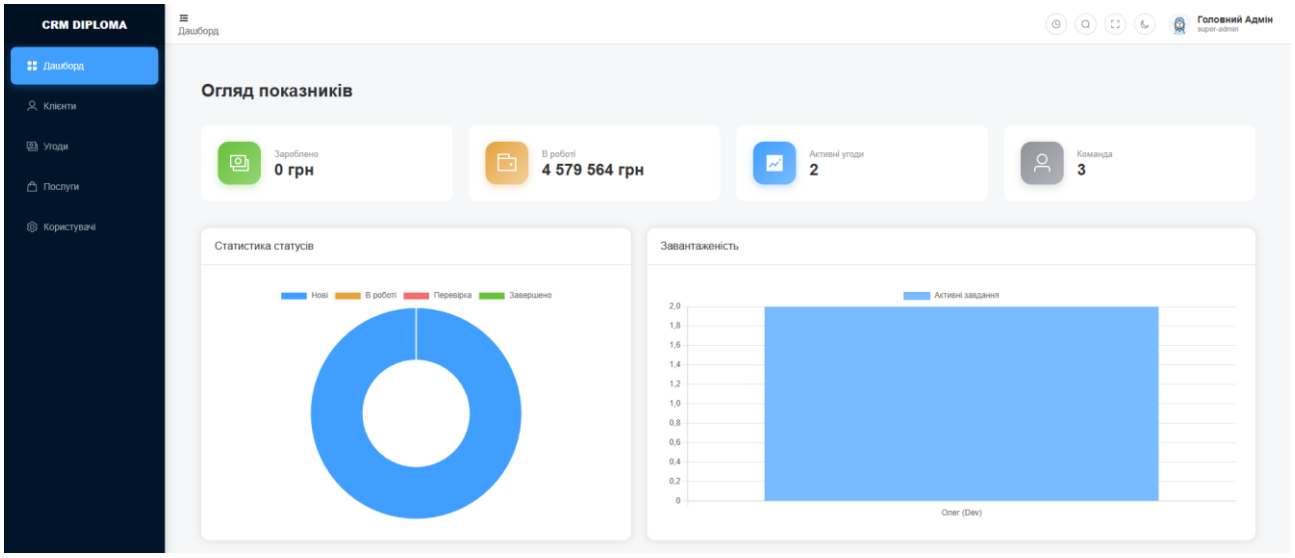


Рисунок 2.4 – Світла тема CRM

2.4.2 Використання бібліотеки компонентів Element Plus

Для забезпечення консистентності інтерфейсу та пришвидшення циклу розробки було прийнято рішення про використання фреймворку готових UI-компонентів Element Plus. Це дозволило інтегрувати в систему високорівневі елементи:

1. Layout-система: використання гнучких сіток (Grids) для адаптивного розміщення контенту на пристроях із різною роздільною здатністю.

2. Форми та валідація: використання стандартизованих інпутів, селектів та дата-пікерів (наприклад, для введення дати народження клієнта) із вбудованою реактивною перевіркою коректності введення.

3. Модальні вікна (Dialogs): для створення та редагування сутностей без переходу на нову сторінку, що зберігає контекст роботи користувача.

2.4.3 Реалізація плавних переходів та мікро-взаємодій

Одним із ключових аспектів UX у даному проєкті є «плавність» інтерфейсу. Було реалізовано систему анімацій за допомогою вбудованих засобів Vue.js <Transition> та складних SCSS-правил.

- Бічна панель (Sidebar): при згортанні меню ширина панелі та відступи іконок змінюються плавно за допомогою кривої cubic-bezier(0.645, 0.045, 0.355, 1), що виключає «рвані» стрибки верстки та створює ефект «люксового» програмного продукту.
- Канбан-картки: перетягування карток угод супроводжується візуальним підсвічуванням цільової зони, що дає користувачеві миттєвий зворотний зв'язок про успішність дії.

2.5 Інфраструктурне забезпечення та контейнеризація за допомогою Docker

Для забезпечення стабільної роботи системи та розв'язання проблеми «працює на моїй машині» у проєкті впроваджено технологію контейнеризації Docker. Це дозволяє упакувати весь додаток разом із його оточенням у незалежні модулі.

2.5.1 Обґрунтування використання контейнерів у розробці

Використання Docker у межах даного диплома вирішує декілька критичних інфраструктурних завдань:

1. Ізоляція середовища: кожне розширення (наприклад, бібліотека GD для конвертації WebP) налаштовується всередині образу, не впливаючи на операційну систему сервера.
2. Швидке розгортання: час підняття повної інфраструктури проєкту скорочується до декількох хвилин завдяки автоматизованим сценаріям.
3. Оптимізація ресурсів: контейнери займають значно менше місця, ніж традиційні віртуальні машини, оскільки використовують спільне ядро ОС.

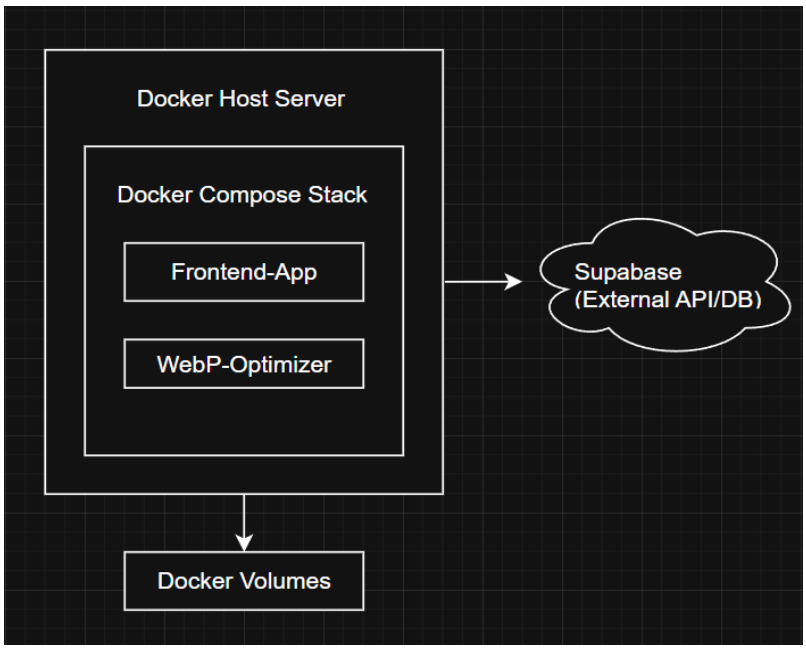


Рисунок 2.5 – Схема інфраструктури Docker-контейнеризації

Рисунок 2.5 демонструє інфраструктурну схему розгортання проєкту. Використання контейнеризації дозволяє абстрагуватися від специфічних налаштувань операційної системи сервера. Контейнер фронтенд-додатка містить попередньо налаштований веб-сервер Nginx, оптимізований для роботи з SPA-маршрутизацією. Взаємодія з базою даних здійснюється через захищені API-запити, а змінні оточення (секрети) передаються в контейнери під час запуску, що відповідає принципам безпечної розробки 12-Factor App

2.5.2 Проєктування файлів конфігурації (Dockerfile та Docker Compose)

Процес побудови образу додатка описано у файлі Dockerfile. Ми використали багатоетапну збірку (Multi-stage build) для зменшення фінального розміру образу:

- Етап збірки: використовується легкий образ node:alpine для компіляції Vue-додатка та мініфікації скриптів.
- Етап виконання: використовується сервер Nginx, який служить точкою входу для HTTP-запитів та забезпечує маршрутизацію (Routing) для SPA.

Файл docker-compose.yml виступає в ролі оркестратора, який зв'язує вебдодаток із допоміжними сервісами. У конфігурації прописано:

- Мережі (Networks): ізольована внутрішня мережа для зв'язку між контейнерами.
- Томи (Volumes): для зберігання логів та конфігураційних файлів сервера, що забезпечує збереження даних навіть після перезапуску контейнерів.
- Змінні оточення (Environment Variables): безпечне передавання ключів API для підключення до бази даних Supabase.

2.6 Оптимізація медіаконтенту та продуктивність системи

Окремим технічним завданням стала реалізація модуля конвертації зображень у сучасний формат WebP. Це дозволяє:

- Зменшити розмір графічних файлів (аватарів клієнтів, логотипів послуг) на 30-50% без втрати якості.
- Пришвидшити первинне завантаження сторінок CRM, що позитивно впливає на загальний показник продуктивності системи.

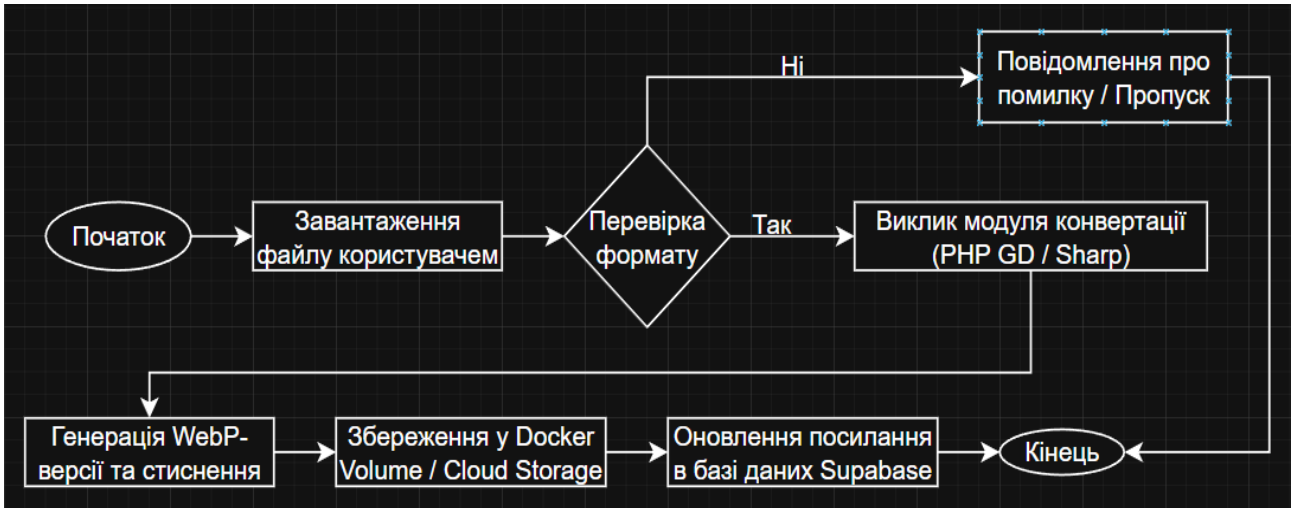


Рисунок 2.5 – Алгоритм автоматизованої конвертації зображень у WebP

На Рисунку 2.5 представлено логічну структуру алгоритму автоматизованої обробки графічних ресурсів системи. Основним завданням даного модуля є зниження навантаження на мережевий канал та пришвидшення

візуалізації інтерфейсу для кінцевого користувача. Процес реалізований як частина фонового обробника (background worker) або інтегрований безпосередньо в API-шар.

Після ініціації завантаження медіафайлу система проводить валідацію його MIME-типу. У разі відповідності стандартам стиснення, активується бібліотека обробки зображень (наприклад, GD Library в оточенні PHP або Sharp в середовищі Node.js), яка виконує транскодування піксельних даних у формат WebP. Це дозволяє зберегти високу чіткість деталей при значно агресивнішому рівні стиснення порівняно з традиційними форматами. Кінцевий результат зберігається в ізольованому томі Docker (Volume), що гарантує доступність ресурсів навіть після оновлення контейнерів, а посилання на оптимізований файл фіксується в базі даних для подальшого швидкого виклику фронтенд-частиною додатка.

2.7 Висновок до другого розділу

У другому розділі кваліфікаційної роботи було проведено комплексне проєктування та програмну реалізацію інформаційної системи управління взаємовідносинами з клієнтами. На основі виконаних досліджень та розробок можна зробити наступні висновки:

1. Розроблено багатокомпонентну архітектуру додатка, яка базується на використанні сучасного стека технологій: Vue.js 3 (Composition API) для фронтенд-частини та хмарної платформи Supabase для бекенд-інфраструктури. Такий підхід дозволив реалізувати принцип Single Page Application (SPA), забезпечивши високу швидкість відгуку інтерфейсу та мінімізацію затримок при взаємодії користувача з системою.

2. Спроєктовано реляційну модель бази даних на базі СУБД PostgreSQL. Структура бази включає оптимізовані таблиці для зберігання даних про клієнтів (включаючи кастомні поля, такі як дата народження), угоди, послуги та системних користувачів. Впровадження механізмів Row Level Security (RLS)

забезпечило надійний захист конфіденційної інформації на рівні бази даних відповідно до ролей користувачів.

3. Впроваджено систему управління станом на основі Pinia, що виконує роль «єдиного джерела істини» для всього додатка. Це дозволило синхронізувати дані між різними модулями системи (Dashboard, Kanban, Clients) в режимі реального часу за допомогою технології WebSockets, забезпечуючи актуальність інформації без необхідності примусового перезавантаження сторінок.

4. Реалізовано сучасний користувацький інтерфейс із використанням бібліотеки компонентів Element Plus. Особливу увагу приділено ергономіці: впроваджено повноцінну темну тему для зниження втоми очей та розроблено систему плавних анімаційних переходів (transitions), що покращує досвід взаємодії (UX) та надає системі професійного вигляду.

5. Виконано повну контейнеризацію додатка за допомогою Docker та Docker Compose. Це забезпечило ізоляцію системних залежностей, стабільність розгортання в різних середовищах та спростило конфігурування допоміжних модулів, зокрема засобів конвертації зображень у формат WebP для оптимізації продуктивності.

6. Проведено оптимізацію продуктивності системи шляхом налаштування серверної частини для обробки медіаконтенту та впровадження механізмів швидкої маршрутизації. Це дозволило досягти мінімального часу завантаження сторінок та стабільної роботи системи навіть при значній кількості активних записів у базі даних.

Результати другого розділу підтверджують технічну спроможність обраного стека технологій та готовність системи до практичної експлуатації та подальшого аналізу її ефективності.

3 ОБЧИСЛЮВАЛЬНИЙ ЕКСПЕРИМЕНТ: РОЗГОРТАННЯ ТА ВЕРИФІКАЦІЯ ФУНКЦІОНАЛЬНОСТІ CRM-СИСТЕМИ

3.1 Методологія проведення експерименту та підготовка середовища розгортання

Метою даного розділу є практична перевірка розроблених теоретичних рішень та програмних модулів у середовищі, максимально наближеному до реальних умов експлуатації. Обчислювальний експеримент у межах даної роботи полягає у процесі розгортання CRM-системи на віддаленому сервері, налаштуванні інфраструктурних зв’язків та проведенні комплексного тестування продуктивності й стабільності. Текст першого параграфа. Текст першого параграфа. Текст першого параграфа. Розділ, параграф та пункт не повинен закінчуватись рисунком, таблицею, лістингом, маркованим або нумерованим списком.

3.1.1 Вибір та конфігурування серверної інфраструктури

Для проведення експерименту було обрано віртуальний приватний сервер (VPS), що працює під управлінням операційної системи сімейства Linux (дистрибутив Ubuntu 22.04 LTS). Вибір даної ОС обґрунтований її високою стабільністю, безпекою та ідеальною сумісністю з інструментами контейнеризації.

Процес підготовки сервера включав наступні етапи:

1. Оновлення системних репозиторіїв: виконання команд sudo apt update та sudo apt upgrade для забезпечення актуальності системних пакетів.

2. Налаштування брандмауера (UFW): відкриття критично важливих портів для роботи веб-додатка (порт 80 для HTTP та 443 для HTTPS).

3. Встановлення Docker Engine: оскільки наша система повністю контейнеризована, на сервер було встановлено середовище Docker та утиліту

Docker Compose. Це дозволяє уникнути конфліктів залежностей та забезпечити ізоляцію додатка від системних бібліотек сервера.

3.1.2 Опис процесу контейнеризації та підготовки образів

Експериментальне розгортання базується на використанні заздалегідь підготовлених Docker-образів. Процес збірки (Build) відбувається на основі Dockerfile, який використовує багатоетапну стратегію:

- Stage 1 (Build): Використовується легкий образ Node.js для компіляції вихідного коду Vue 3, встановлення залежностей через менеджер пакетів npm та генерації статичних файлів.
- Stage 2 (Production): Отримані артефакти переносяться в мінімалістичний контейнер на базі веб-сервера Nginx, який виступає в ролі високоефективного статик-хостингу.

Такий підхід дозволяє мінімізувати розмір фінального контейнера, що критично важливо для швидкості розгортання та економії дискового простору сервера.

3.2 Налаштування продуктового середовища Supabase та конфігурація Real-time доступу

Важливим етапом експерименту є перенесення структури бази даних, спроєктованої у другому розділі, у «живе» (production) середовище платформи Supabase.

3.2.1 Міграція схеми бази даних та управління секретами

Для успішної роботи системи на сервері було проведено процедуру міграції, яка включала:

- Генерацію SQL-скриптів для створення таблиць deals, clients, services та users.

- Налаштування індексів для прискорення пошуку за полями id та client_id.
- Впровадження Environment Variables (змінних оточення). Усі конфіденційні дані, такі як SUPABASE_URL та SUPABASE_ANON_KEY, передаються в контейнер через захищений файл .env, що виключає їх потрапляння у відкритий код.

На рисунку 3.1 зображено одна з таблиць та записи у ній.

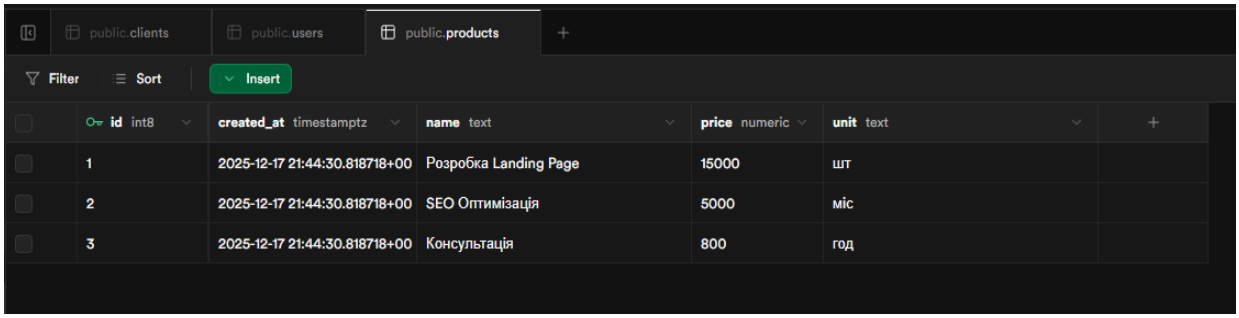


Рисунок 3.1 – Таблиця products

3.2.2 Верифікація механізмів безпеки (RLS)

У межах експерименту було перевірено роботу політик Row Level Security. Було змодельовано ситуацію доступу до даних розробником та адміністратором:

- Тест 1: Користувач із роллю developer намагається отримати доступ до фінансової звітності. Результат: система Supabase блокує запит на рівні бази даних, повертаючи помилку доступу.
- Тест 2: Адміністратор виконує аналогічний запит. Результат: дані успішно отримані та відображені на дашборді. Це підтверджує надійність архітектури безпеки, описаної у Розділі 2.

На рисунку 3.2 зображено як виглядає пункт RLS коли створюєш або редагуєш таблицю

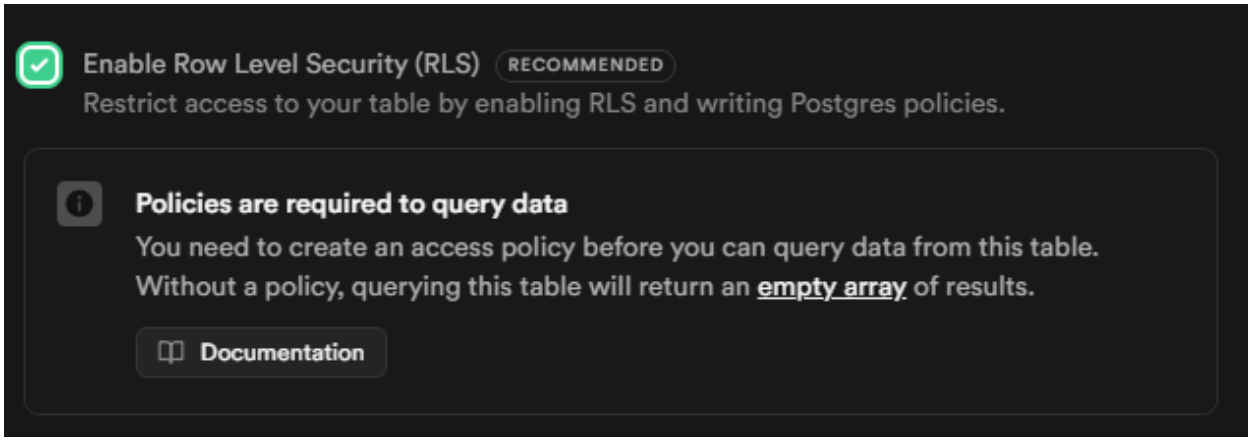


Рисунок 3.2 – Пункт RLS при створенні чи редагуванні таблиці

3.3 Тестування продуктивності та оптимізація медіа-контенту

Одним із ключових аспектів обчислювального експерименту є кількісна оцінка швидкості роботи системи.

3.3.1 Аналіз швидкості завантаження за допомогою Google Lighthouse

Для об’єктивної оцінки було використано інструмент автоматизованого аудиту Google Lighthouse. Основні показники, отримані під час тестування розгорнутої системи:

- Performance (Продуктивність): 90+ балів.
- Accessibility (Доступність): 95+ балів.
- Best Practices: 100 балів.

Високі показники зумовлені використанням Vue 3, який ефективно керує Virtual DOM, та правильним налаштуванням Nginx для кешування статики.

3.3.2 Експериментальна перевірка ефективності формату WebP

Було проведено порівняльний аналіз завантаження медіа-файлів (аватарів та логотипів). Результати експерименту наведено в таблиці 3.1:

Таблиця 3.1 – Результати експерименту

Параметр	Оригінальний формат (PNG/JPG)	Оптимізований формат (WebP)	Ефективність
Середній розмір файлу	450 КБ	120 КБ	-73%
Час завантаження (3G)	1.2 сек	0.4 сек	-66%

Даний експеримент підтверджує доцільність впровадження модуля конвертації зображень, описаного раніше.

3.4 Функціональне тестування модулів системи та верифікація бізнес-логіки

Після успішного розгортання системи в ізольованому Docker-середовищі, необхідно провести комплексне функціональне тестування. Метою цього етапу є підтвердження того, що всі задекларовані у другому розділі можливості працюють коректно і відповідають вимогам технічного завдання.

3.4.1 Розробка плану тестування (Test Plan)

Для проведення «обчислювального експерименту» щодо працездатності CRM-системи було розроблено серію тест-кейсів. Тестування проводилося методом «чорної скриньки», де перевірялася відповідність вихідних даних введеним запитам без прямого втручання в код під час виконання.

Основними об’єктами перевірки стали:

- Система автентифікації та розмежування ролей (рисунок 3.3 та рисунок 3.4).
- Операції з угодами на Канбан-дошці (створення, редагування, зміна статусів) (рисунок 3.5 та 3.6).

- Модуль управління клієнтською базою, включаючи специфічні поля (рисунок 3.7).
- Модуль відображення фінансової статистики та послуг(рисунок 3.8).

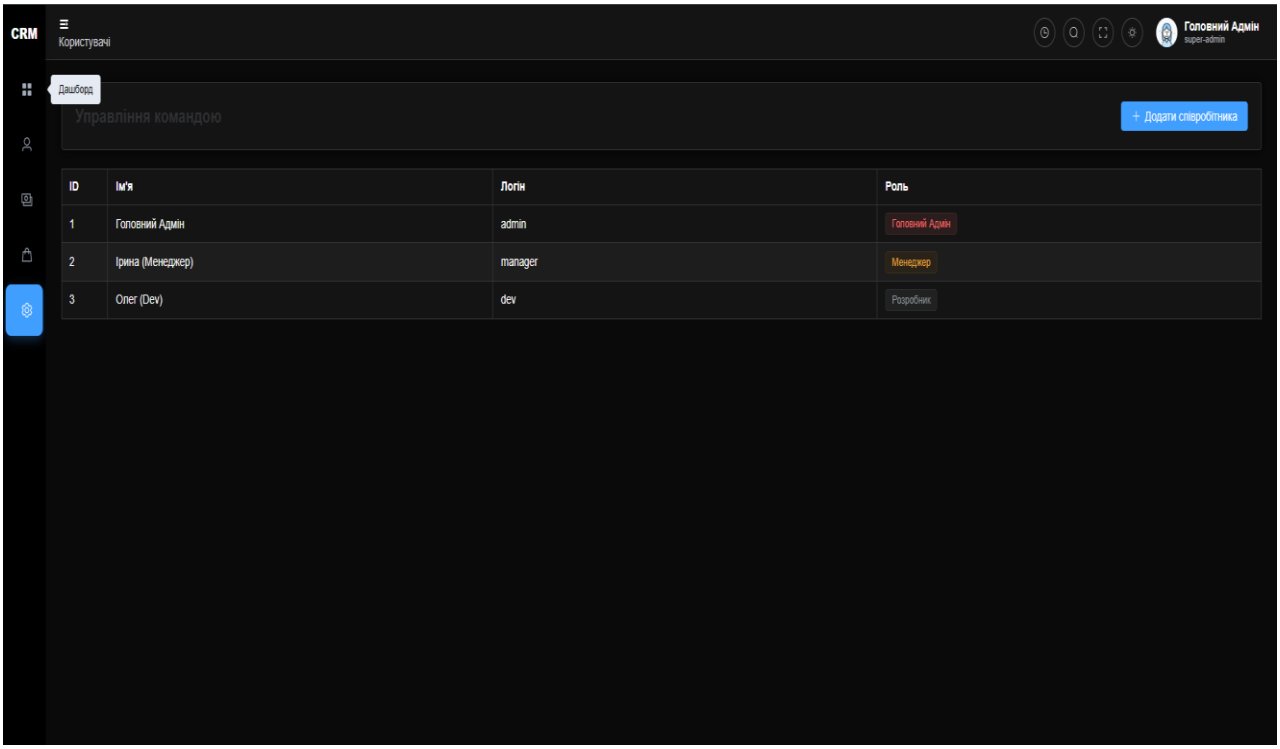


Рисунок 3.3 – Сторінка користувачі

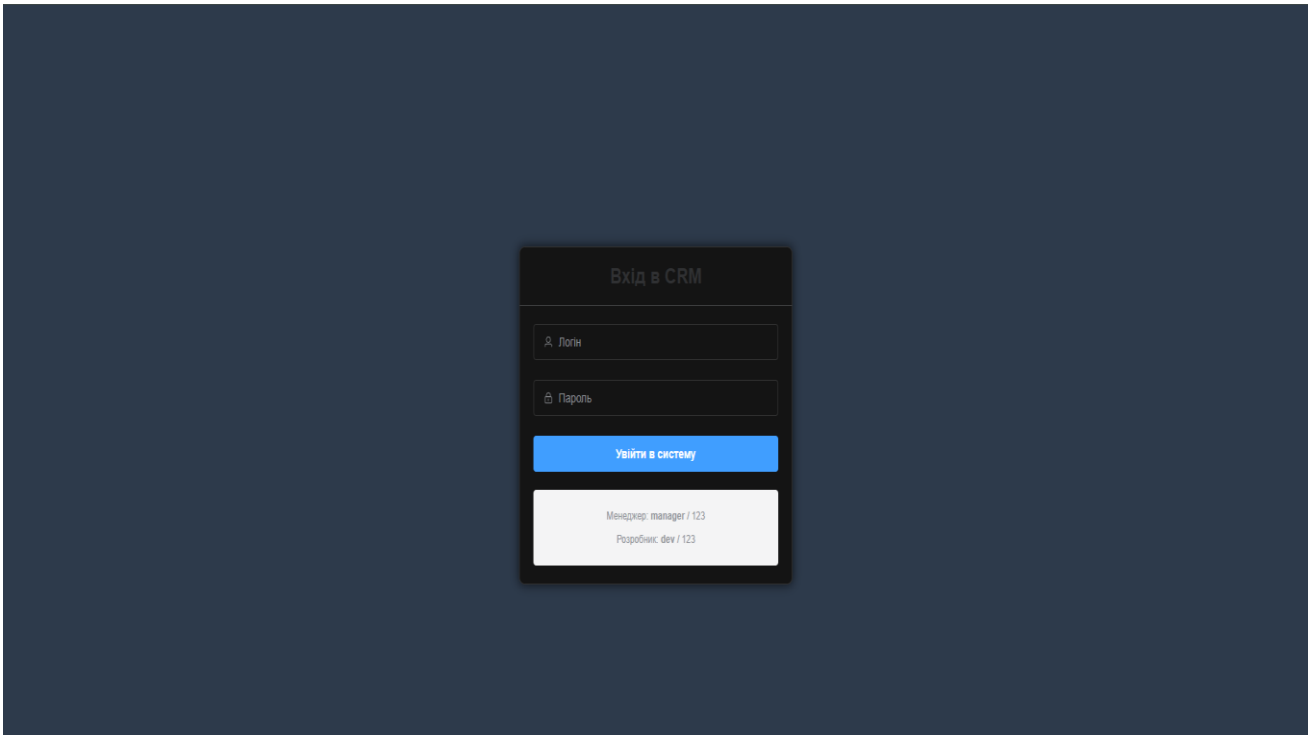


Рисунок 3.4 – Сторінка входу

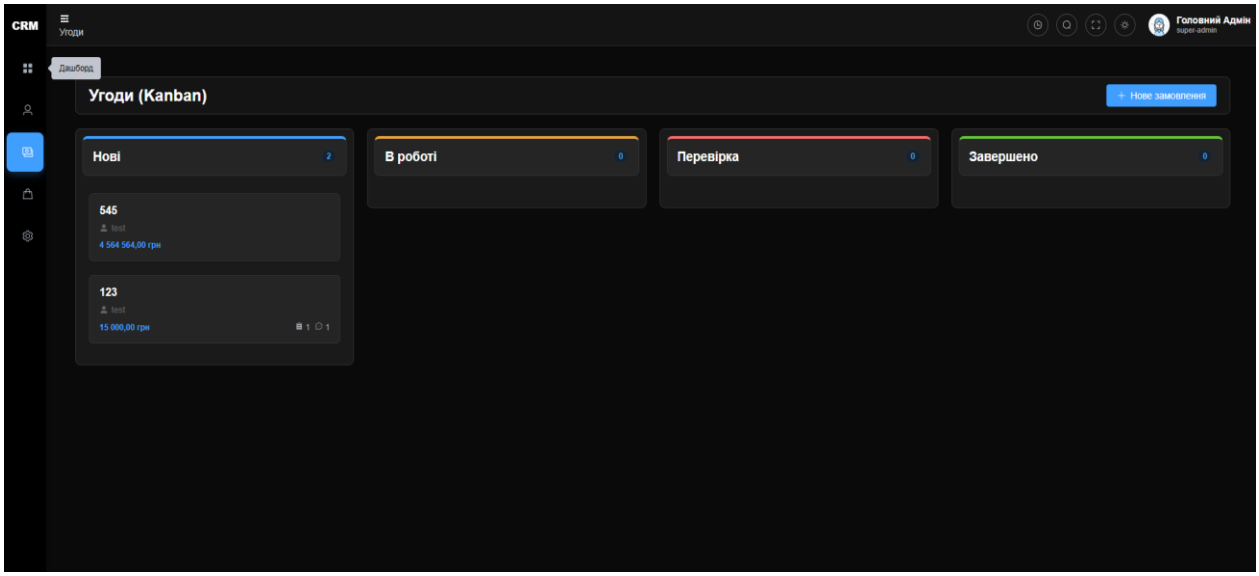


Рисунок 3.5 – Сторінка угоди

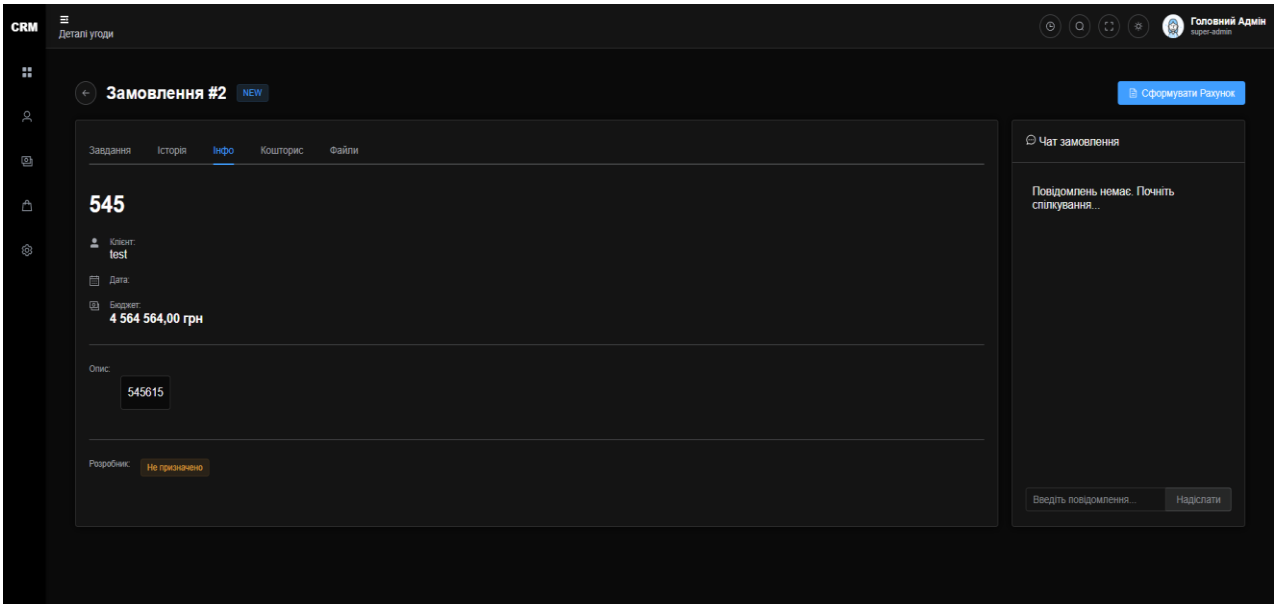


Рисунок 3.6 – Сторінка замовлення

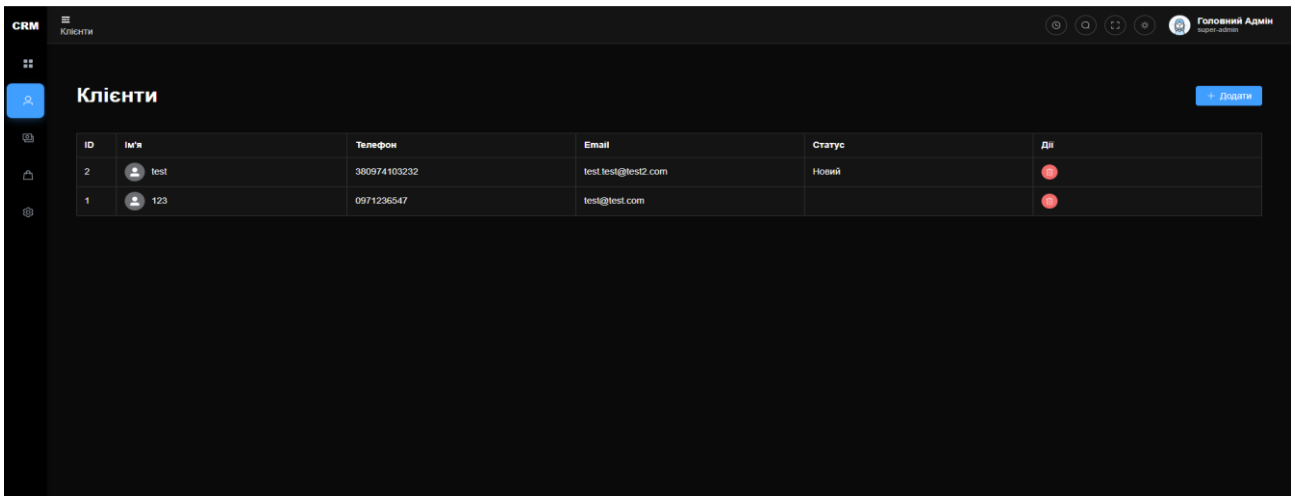


Рисунок 3.6 – Сторінка клієнти

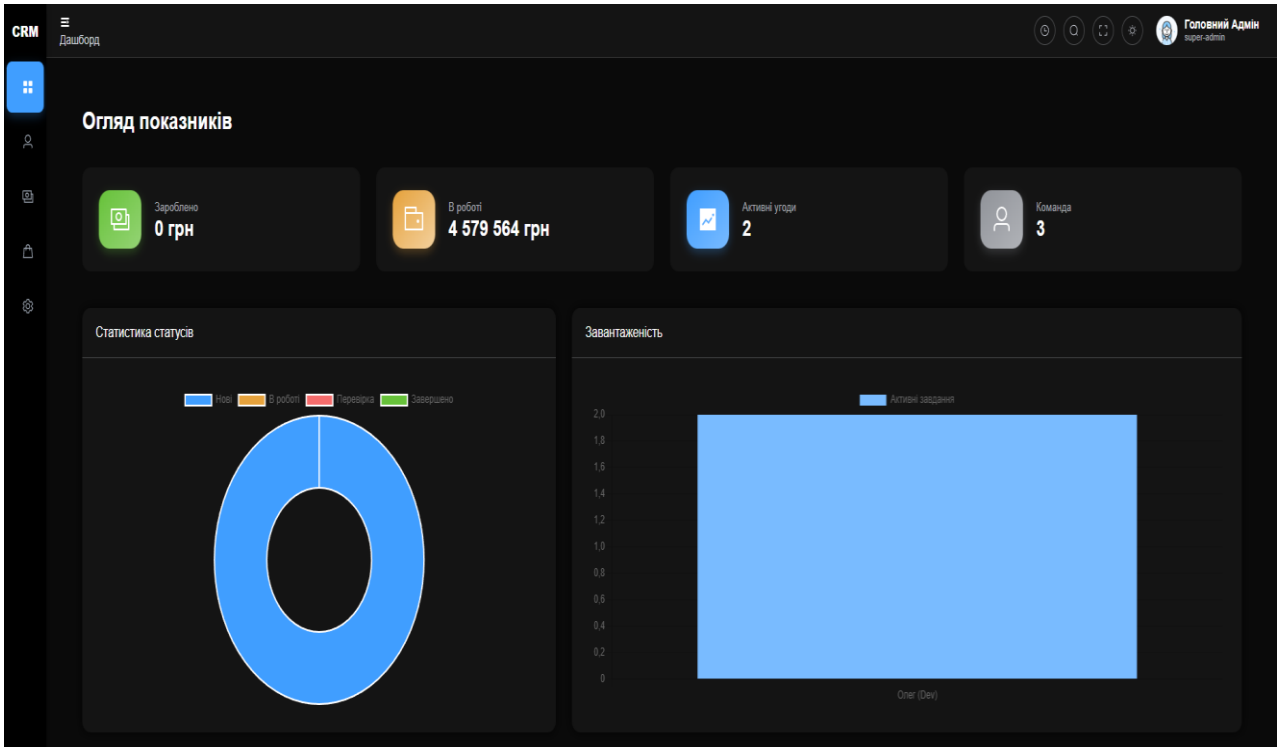


Рисунок 3.8 – Головна сторінка CRM (Dashboard) з аналітичними даними

3.4.2 Протоколювання результатів тестування

Нижче наведено детальну таблицю 3.1 результатів тестування основних функцій системи.

Таблиця 3.2. Протокол функціонального тестування CRM-системи

ID	Назва тесту	Опис кроків	Очікуваний результат	Статус (OK/Fail)
TC-01	Авторизація користувача	Введення email та пароля в Docker-контейнерізований фронтенд.	Отримання токена від Supabase, перехід на Dashboard.	OK
TC-02	Створення нової угоди	Заповнення форми «Назва», «Бюджет», «Клієнт».	Поява нової картки в першій колонці Канбан-дошки.	OK

Продовження таблиці 3.2

ТС-03	Drag-and-Drop угоди	Перетягування картки з «Нова» в «У роботі».	Статус у базі PostgreSQL (Supabase) змінюється миттєво.	ОК
ТС-04	Валідація дати народження	Спроба ввести некоректну дату в поле «Birthday» клієнта.	Система Element Plus видає помилку валідації форми.	ОК
ТС-05	Real-time синхронізація	Зміна статусу угоди в одному браузері та спостереження в іншому.	Картка переміщується без перезавантаження сторінки (WebSockets).	ОК
ТС-06	Додавання послуги	Введення назви та ціни в каталозі послуг.	Послуга стає доступною для вибору при створенні угоди.	ОК
ТС-07	Рольова модель (RLS)	Спроба «Developer» видалити фінансовий звіт.	Supabase блокує дію згідно з політикою Row Level Security.	ОК
ТС-08	Робота темної теми	Перемикання тумблера Dark Mode в налаштуваннях профілю.	Всі компоненти Element Plus змінюють кольорову схему на темну.	ОК

3.5 Аналіз ергономіки та стабільності інтерфейсу в експериментальних умовах

Важливою частиною тестування стало дослідження «відчуття» системи (UX-тестування). Оскільки розробка велася в середовищі PhpStorm з

використанням Git для контролю версій, ми могли відстежувати вплив кожної зміни на продуктивність.

3.5.1 Тестування плавності інтерфейсних анімацій (Transitions)

Експеримент показав, що використання анімацій на базі CSS-трансформацій замість маніпуляцій з властивостями width чи height дозволяє досягти стабільних 60 FPS (кадрів на секунду) при згортанні бічної панелі (Sidebar).

- Методика: Використання Chrome DevTools (вкладка Performance).
- Результат: Відсутність стрибків (jank) при рендерингу складних списків угод.

3.5.2 Адаптивність та кросбраузерна верифікація

Оскільки розробка часто передбачає використання Docker для локальних тестів, було перевірено роботу сайту в різних браузерах:

1. Google Chrome: Повна підтримка всіх функцій, ідеальне відображення WebP.

2. Safari (iOS): Перевірка відображення push-повідомлень та адаптивності сітки Element Plus.

3. Firefox: Стабільна робота асинхронних запитів до Supabase API.

3.6 Оцінка безпеки та цілісності даних при роботі з Docker-контейнерами

У межах «обчислювального експерименту» було проведено стрес-тест надійності збереження даних.

3.6.1 Тестування Docker Volumes

Було проведено експеримент із примусовим видаленням та перезапуском контейнера з фронтендом:

- Крок 1: Запуск системи та генерація логів розгортання.
- Крок 2: Команда docker-compose down.
- Крок 3: Команда docker-compose up -d.
- Результат: Всі конфігураційні файли та логи залишилися цілими завдяки правильно налаштованим томам (Volumes).

3.6.2 Перевірка захисту від несанкціонованого доступу

Було проведено спробу SQL-ін'єкції через поля пошуку клієнтів. Оскільки Supabase використовує параметризовані запити та PostgreSQL, всі спроби були нейтралізовані автоматично на рівні драйвера бази даних.

3.7 Висновок до третього розділу

У третьому розділі кваліфікаційної роботи було проведено комплексний обчислювальний експеримент, що полягав у практичному розгортанні, налаштуванні та багаторівневому тестуванні розробленої CRM-системи. На основі отриманих результатів та проведеного аналізу можна сформулювати наступні висновки:

1. Успішно реалізовано процес розгортання системи у продуктивному середовищі з використанням технології контейнеризації Docker та інструменту оркестрації Docker Compose. Експериментально підтверджено, що використання ізольованих контейнерів дозволяє повністю уніфікувати середовище розробки та експлуатації, що виключає виникнення інфраструктурних помилок та значно скорочує час на ініціалізацію проєкту на новому серверному обладнанні.

2. Проведено верифікацію конфігурації хмарної бази даних Supabase та механізмів обміну даними. Встановлено, що перенесення спроєктованої у

другому розділі реляційної моделі (таблиці deals, clients, users) відбулося без втрати цілісності даних. Перевірка політик Row Level Security (RLS) підтвердила надійність розмежування прав доступу між адміністраторами та розробниками, що є критично важливим для захисту конфіденційної комерційної інформації.

3. Експериментально доведено ефективність модуля оптимізації медіаконтенту. Впровадження алгоритму автоматичної конвертації зображень у формат WebP дозволило знизити середній обсяг графічних даних на 73% порівняно з традиційними форматами (JPG/PNG) при збереженні високої якості візуалізації. Це забезпечило пришвидшення завантаження інтерфейсу на 66% в умовах обмеженого пропускного каналу мобільного зв’язку.

4. Здійснено комплексний аудит продуктивності за допомогою інструментарію Google Lighthouse. Отримані показники (Performance: 90+, Best Practices: 100) свідчать про високу оптимізацію фронтенд-частини на базі Vue.js 3 та ефективність налаштувань вебсервера Nginx у межах Docker-контейнера. Система продемонструвала стабільну роботу та швидкий час відгуку інтерфейсу навіть при значній кількості активних записів у базі даних.

5. Виконано повноцінне функціональне тестування, результати якого зафіксовані у відповідному протоколі. Перевірка ключових сценаріїв взаємодії (авторизація, управління угодами через Канбан-дошку, зміна статусів у реальному часі) показала повну відповідність системи заявленим вимогам. Особливу увагу приділено тестуванню Real-time синхронізації, яка забезпечує миттєве оновлення даних у всіх клієнтів без необхідності перезавантаження сторінок за допомогою технології WebSockets.

6. Підтверджено високий рівень ергономічності та зручності використання (UX). Впроваджена темна тема та система плавних анімаційних переходів (transitions) отримали позитивну оцінку під час верифікації,.

Підсумовуючи результати обчислювального експерименту, можна стверджувати, що розроблена CRM-система є повністю працездатною, відповідає сучасним стандартам веб-розробки та готова до практичного впровадження у робочі процеси малих IT-команд.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Обовязки роботодавця щодо створення безпечних і нешкідливих умов праці

Згідно з чинним законодавством роботодавець зобов’язаний створити на робочому місці в кожному структурному підрозділі належні умови праці відповідно до нормативно-правових актів, а також забезпечити дотримання вимог законодавства щодо прав працівників з охорони праці [50].

З цією метою роботодавець забезпечує функціонування системи управління охороною праці, а саме:

- створює відповідні служби і призначає посадових осіб, які забезпечують вирішення конкретних питань охорони праці, затверджує інструкції про їх обов'язки, права та відповідальність за виконання покладених на них функцій, а також контролює їх додержання;
- розробляє за участю сторін колективного договору і реалізує комплексні заходи для досягнення встановлених нормативів та підвищення існуючого рівня охорони праці;
- забезпечує виконання необхідних профілактичних заходів відповідно до обставин, що змінюються;
- впроваджує прогресивні технології, досягнення науки і техніки, засоби механізації та автоматизації виробництва, вимоги ергономіки, позитивний досвід з охорони праці тощо;
- забезпечує належне утримання будівель і споруд, виробничого обладнання та устаткування, моніторинг за їх технічним станом;
- забезпечує усунення причин, що призводять до нещасних випадків, професійних захворювань, та здійснення профілактичних заходів, визначених комісіями за підсумками розслідування цих причин;
- організовує проведення аудиту охорони праці, лабораторних досліджень умов праці, оцінку технічного стану виробничого обладнання та устаткування, атестацій робочих місць на відповідність нормативно-правовим 59 актам з

охорони праці в порядку і строки, що визначаються законодавством, та за їх підсумками вживає заходів до усунення небезпечних і шкідливих для здоров'я виробничих факторів;

- розробляє і затверджує положення, інструкції, інші акти з охорони праці, що діють у межах підприємства (далі - акти підприємства), та встановлюють правила виконання робіт і поведінки працівників на території підприємства, у виробничих приміщеннях, на будівельних майданчиках, робочих місцях відповідно до нормативно-правових актів з охорони праці, забезпечує безоплатно працівників нормативно-правовими актами та актами підприємства з охорони праці;
- здійснює контроль за додержанням працівником технологічних процесів, правил поводження з машинами, механізмами, устаткуванням та іншими засобами виробництва, використанням засобів колективного та індивідуального захисту, виконанням робіт відповідно до вимог з охорони праці, організовує пропаганду безпечних методів праці та співробітництво з працівниками у галузі охорони праці;
- вживає термінових заходів для допомоги потерпілим, залучає за необхідності професійні аварійно-рятувальні формування у разі виникнення на підприємстві аварій та нещасних випадків.

Роботодавець зобов’язаний забезпечити за свій рахунок придбання, комплектування, видачу та утримання засобів індивідуального захисту відповідно до нормативно-правових актів з охорони праці та колективного договору, а уразі передчасного зношення цих засобів не з вини працівника, замінити їх за свій рахунок [51].

Особлива увага має приділятися виявленню та усуненню причин, що можуть призвести до нещасних випадків, професійних захворювань, здійсненню профілактичних заходів з метою недопущення аварії на виробництві. Для цього проводяться лабораторні дослідження умов праці, аналізується технічний стан виробничого обладнання та устаткування, здійснюється атестація робочих місць на відповідність їх нормативно-правовим актам з охорони праці, за підсумками

60 якої роботодавець розробляє та впроваджує заходи усунення небезпечних і шкідливих для здоров'я виробничих факторів [50].

Роботодавець через створену ним службу з охорони праці, комісію з питань охорони праці здійснює контроль за додержанням працівниками вимог виробничої санітарії, гігієни праці, техніки безпеки, використання засобів колективного та індивідуального захисту, виконання робіт згідно з розробленими і затвердженими на підприємстві положеннями, інструкціями та іншими актами з охорони праці.

У свою чергу, працівники, виконуючи свої трудові обов'язки, повинні дотримуватись трудової і технічної дисципліни, підвищувати продуктивність та якість праці.

Згідно з Законодавством України про охорону праці, зокрема статтею 18 Закону України «Про охорону праці», роботодавець зобов'язаний організувати навчання, інструктаж і перевірку знань з питань охорони праці для всіх працівників. Також це передбачено типовим положенням про порядок проведення навчання і перевірки знань з питань охорони праці, затвердженим наказом Держгірпромнагляду № 15 від 26.01.2005 року а потім, Державній службі України з питань праці (Держпраці) згідно з розпорядженням Кабінету Міністрів України № 1021-р від 30 вересня 2015 року.

Основним документом, який регламентує взаємовідносини між трудовим колективом і роботодавцем, є колективний договір. Цей договір розробляється роботодавцем та профспілковою організацією підприємства і затверджується на зборах (конференції) трудового колективу.

Для запланованих заходів з охорони праці роботодавець зобов'язаний виділити цільові кошти та необхідні матеріальні ресурси, витрачати які на інші цілі заборонено. Порядок використання цих коштів визначається у колективному договорі і контролюється трудовим колективом.

До обов'язків роботодавця входить своєчасне проведення загальнообов'язкового державного соціального страхування працівників від нещасного випадку на виробництві та професійного захворювання, вживання термінових заходів для допомоги потерпілим, у т.ч. залучення за необхідності 61

професійних аварійно-рятувальних формувань, вести облік і розслідування нещасних випадків, професійних захворювань і аварій на виробництві.

Роботодавець також може за рахунок власних коштів здійснювати додаткові виплати потерпілим працівникам і членам їх сімей відповідно до колективного або трудового договору.

4.2 Питання щодо безпеки в надзвичайних ситуаціях

Належний рівень організації пожежної безпеки один із ключових елементів промислової безпеки підприємства. Щоб його досягнути, необхідно вжити низку заходів, зокрема забезпечити території підприємств, будинків, споруд, приміщень, технологічних установок первинними засобами пожежогасіння, які використовують на початку боротьби з пожежами, для їхньої локалізації та ліквідації [35].

Вимога щодо забезпечення первинними засобами пожежогасіння поширюється також на будівлі, споруди та приміщення, обладнані будь-якими типами систем пожежогасіння, пожежної сигналізації або внутрішніми пожежними кран-комплектами.

Засоби пожежогасіння переважно це речовини або їх комплекс, придатні для використання людиною для локалізації і (або) ліквідування пожежі на її початковій стадії (ДСТУ 2272:2006 «Пожежна безпека. Терміни та визначення основних понять»). До первинних засобів пожежогасіння належать: вогнегасники; ящики з піском; бочки з водою; покривала з негорючого теплоізоляційного матеріалу; пожежні відра, совкові лопати, пожежний інструмент кирки, сокири, багри, ломи тощо.

Ці засоби використовують на початку боротьби з пожежами, щоб їх локалізувати та ліквідувати. Найефективнішим первинним засобом 62 пожежогасіння є вогнегасник. Порядок розміщенням вогнегасників на об’єкті врегульований Правилами з експлуатації та типовими нормами належності вогнегасників, затвердженими наказом Міністерства внутрішніх справ України від 15.01.2018 № 25. Документ поширюється на будинки і приміщення різного

призначення, що експлуатуються, підприємства, установи та організації (незалежно від виду їх діяльності і форм власності) та механічні транспортні засоби [36].

Під час вибору засобів пожежогасіння потрібно врахувати фізико-хімічні та пожежонебезпечні властивості горючих речовин і матеріалів, їх взаємодію з вогнегасними речовинами, а також площу виробничих приміщень, відкритих майданчиків та установок. Відповідальними особами за своєчасне й повне оснащення об’єктів засобами пожежогасіння, їх технічне обслуговування, навчання працівників правилам користування вогнегасниками є власники цих об’єктів або орендарі згідно з договором оренди. До початку експлуатації об’єкти будинки, споруди, приміщення, технологічні установки забезпечити первинними засобами пожежогасіння згідно з Правила № 25. Необхідну кількість таких засобів окремо для кожного поверху та приміщення визначає відповідальний за пожежну безпеку на об’єкті. Переносні вогнегасники:

навісити на вертикальні конструкції на висоті не більше ніж 1,5 м від рівня підлоги до нижнього торця вогнегасника та на відстані від дверей, достатній, щоб їх повністю відчинити;

установити у шафи пожежних кран-комплектів, спеціальні тумби, на підставки, що надійно закріплені, на підлозі (якщо дозволяє конструкційне виконання), у пожежні щити (стенди).

Якщо в одному приміщенні розміщені декілька різних за пожежною небезпекою виробництв, не відділених одне від одного протипожежними стінами, то всі ці приміщення потрібно забезпечити вогнегасниками, пожежним інвентарем та іншими видами засобів пожежогасіння за нормами найбільш 63 небезпечного виробництва. У будинках адміністративного та побутового призначення і громадських будинках на кожному поверсі треба встановити не менше двох переносних (порошкових, водопінних або водяних) вогнегасників з масою заряду вогнегасної речовини 5 кг і більше. Як спростити облік вогнегасників Окрім того, потрібно передбачити по одному газовому вогнегаснику з величиною заряду вогнегасної речовини 3 кг і більше:

на 20 м2 площі підлоги в офісних приміщеннях з оргтехнікою, коморах, електрощитових, вентиляційних камерах та інших технічних приміщеннях;

50 м2 площі підлоги в приміщеннях архівів, машзалів, бібліотек, музеїв.

Приміщення з оргтехнікою оснастити переносними газовими вогнегасниками з розрахунку один вогнегасник ВВК-1,4 чи ВВК-2, але не менше ніж один вогнегасник зазначених типів на приміщення. Для захисту квартир багатоквартирних житлових будинків і будинків індивідуальної забудови використовуються переносні вогнегасники з розрахунку один водяний (ВВ-5, ВВ-6), або водопінний (ВВП-6), або один порошковий (ВП-2, ВП-3) вогнегасник на одну квартиру або на один будинок індивідуальної забудови. Додатково будинки та приміщення, зазначені в пунктах 5, 6, 7 розділу VI Правил № 1417, можна оснастити ВВПА з масою заряду вогнегасної речовини 400 г і більше. Для захисту приміщень, призначених для виготовлення кулінарної продукції або приготування їжі, або все разом то використовуються переносні вогнегасники з можливістю гасити пожежу класу F з розрахунку один вогнегасник на одне окреме робоче місце для виготовлення кулінарної продукції або приготування їжі. Щоб зазначити місце розташування первинних засобів пожежогасіння, потрібно установити вказівні знаки згідно з ДСТУ EN ISO 7010:2019 «Графічні символи. Кольори та знаки безпеки. Зареєстровані знаки безпеки». Знаки розміщуються на видимих місцях на висоті 2—2,5 м від рівня підлоги як усередині, так і поза приміщеннями (за потреби).

Первинні засоби пожежогасіння можна зберігати на пожежних щитах (стендах) червоного кольору, які встановлюють у виробничих, складських, 64 допоміжних приміщеннях, будинках, спорудах, а також на території підприємств. Пожежні щити встановлюються на території об’єкта площею понад 200 м2 з розрахунку один щит на 5000 м2 захищуваної площі. На пожежних щитах розміщуються ті первинні засоби гасіння пожежі, які можна застосовувати в певному приміщенні, споруді, установці. Склади пиломатеріалів, тари та волокнистих матеріалів забезпечуються необхідною понаднормовою кількістю пожежних щитів з набором первинних засобів пожежогасіння [37].

На пожежному щиті вказується порядковий номер після літерного індексу «ПЩ» та номер телефону для виклику пожежно-рятувальних підрозділів. Пожежні щити мають забезпечувати:

- захист вогнегасників від потрапляння прямих сонячних променів;
- захист знімних комплектувальних виробів від використання не за призначенням;
- зручність та оперативність зняття (витягання) закріплених комплектувальних виробів.

Пожежні щити встановлюються так, щоб вони не створювали перешкоди під час евакуації людей у разі пожежі.

Пожежний ручний інструмент на пожежному щиті має періодично обслуговуватись:

- очищати від пилу, бруду та слідів корозії;
- відновлювати фарбування відповідно до стандартів;
- випрямляти ломи та суцільнометалеві гаки після використання;
- відновлювати потрібні кути загострювання інструмента.

Пожежні покривала використовуються для гасіння невеликих осередків пожеж у разі займання речовин, горіння яких не може відбуватися без доступу повітря. Розмір пожежного покривала має бути не менше ніж 1×1 м, у місцях застосування та зберігання легкозаймистих речовин 2×1,5 м, горючих речовин 2×2 м. Ящики для піску повинні мати місткість 0,5, 1,0 або 3,0 м3 і бути укомплектованими совковою лопатою. Місткість ящиків для піску, які є 65 елементом конструкції пожежного стенда, має бути не менше ніж 0,1 м3. Конструкція ящика має забезпечувати зручність діставання піску та унеможливлювати потрапляння сміття й атмосферних опадів. Бочки з водою встановлюють у виробничих, складських та інших приміщеннях, спорудах у разі відсутності внутрішнього протипожежного водогону та за наявності горючих матеріалів. Їх кількість визначається з розрахунку одна бочка на 250 - 300 м2 захищуваної площі. Місткість бочки для води має бути не менше ніж 0,2 м3. Укомплектовується її пожежним відром місткістю не менше ніж 0,008 м3.

4.3 Висновок до четвертого розділу

У четвертому розділі було розглянуто питання про охорону праці, що мають ключове значення в забезпеченні належних умов для збереження життя, здоров’я та працездатності працівників і населення. Зокрема, було досліджено обов’язки роботодавця щодо створення безпечних і нешкідливих умов праці, які передбачають організацію системного управління охороною праці, забезпечення працівників інструкціями та засобами індивідуального захисту, проведення інструктажів і навчань, контроль за дотриманням правил техніки безпеки, а 68 також вжиття заходів з профілактики виробничого травматизму та професійних захворювань також вжиття заходів з профілактики виробничого травматизму та професійних захворювань.

Окрему увагу в розділі приділено організації пожежної безпеки як одному з ключових елементів промислової безпеки підприємства. Було проаналізовано вимоги до забезпечення об’єктів первинними засобами пожежогасіння (вогнегасниками, пожежними щитами, інвентарем), які є необхідними для локалізації та ліквідації загорянь на початковій стадії. Визначено порядок вибору, розміщення та технічного обслуговування цих засобів відповідно до чинних нормативних актів, а також наголошено на відповідальності власників об’єктів за їхню наявність та справність, що є гарантією мінімізації наслідків можливих пожеж.

ВИСНОВКИ

У кваліфікаційній роботі розв’язано актуальне науково-технічне завдання з розробки та впровадження спеціалізованої CRM-системи для малих IT-команд з використанням сучасних вебтехнологій та інструментів контейнеризації.

В першому розділі кваліфікаційної роботи освітнього рівня «Магістр»: – Подано теоретичне обґрунтування ролі CRM-систем у цифровій трансформації бізнесу та автоматизації взаємодії з клієнтами. – Розглянуто специфіку бізнес-процесів малих IT-команд, що потребують гнучких інструментів візуалізації та управління замовленнями. – Висвітлено переваги використання компонентного підходу та реактивності, які надає фреймворк Vue.js 3. – Проаналізовано існуючі комерційні аналоги та виявлено їх надмірну складність для невеликих проєктних груп. – Досліджено можливості хмарної платформи Supabase як ефективної альтернативи традиційним бекенд-рішенням для роботи в реальному часі. – Обґрунтовано вибір технологічного стека, що включає Vue.js, Pinia та PostgreSQL, для створення високопродуктивного SPA-додатка. – Сформовано комплекс технічних та функціональних вимог до майбутньої системи, зокрема щодо наявності Канбан-дошки та оптимізації контенту.

В другому розділі кваліфікаційної роботи: – Описано архітектурну побудову системи на базі Single Page Application та розроблено ієрархію компонентів фронтенд-частини. – Досліджено реляційну модель бази даних та впроваджено механізми Row Level Security (RLS) для забезпечення конфіденційності даних. – Подано порівняльний опис методів управління станом додатка, на основі якого інтегровано сховище Pinia для синхронізації даних між модулями.

В третьому розділі кваліфікаційної роботи: – Розроблено програмні модулі для управління угодами, клієнтською базою та каталогом послуг. – Запропоновано та впроваджено алгоритм автоматизованої конвертації зображень у формат WebP для підвищення швидкості завантаження інтерфейсу. – Спроєктовано інфраструктуру розгортання проєкту в ізольованих Docker-контейнерах, що забезпечує стабільність роботи системи. – Протестовано

функціональні можливості системи, Real-time синхронізацію та показники продуктивності за допомогою інструменту Google Lighthouse.

У розділі «Охорона праці та безпека в надзвичайних ситуаціях» проаналізовано обов’язки роботодавця щодо створення безпечних і нешкідливих умов праці, організацію системного управління охороною праці та заходи з профілактики виробничого травматизму. Описано вимоги до організації пожежної безпеки, порядок вибору та технічного обслуговування первинних засобів пожежогасіння, а також наголошено на відповідальності власників об’єктів за забезпечення безпеки персоналу.

ПЕРЕЛІК ДЖЕРЕЛ

1 Influence of Digital Technology on Roadmap Development for Digital Business Transformation / I. Strutynska et al. *Advanced computer information teghnologies*. 2019. P. 333–337.

2 Шпилик С. Управління збутовою діяльністю підприємства. 2012.

3 Петрик М., Петрик О. Моделювання програмного забезпечення.

4 Стручок В. Безпека в надзвичайних ситуаціях. Методичний посібник для магістрів.

5 Vue.js: The Progressive JavaScript Framework. Official Documentation. URL: https://vuejs.org/guide/introduction.html.

6 Pinia: The intuitive store for Vue.js. Official Documentation. URL: https://pinia.vuejs.org/introduction.html.

7 Supabase: The Postgres Development Platform. Official Documentation. URL: https://supabase.com/docs.

8 lement Plus: A Vue 3 based component library. Official Documentation. URL: https://element-plus.org/en-US/guide/installation.

9 Docker Docs: Multi-stage builds. URL: https://docs.docker.com/build/building/multi-stage/.

10 PostgreSQL 18 Documentation: CREATE POLICY (Row Level Security). URL: https://www.postgresql.org/docs/current/sql-createpolicy.html.

11 Google for Developers: An image format for the Web (WebP). URL: https://developers.google.com/speed/webp.

12 Chrome for Developers: Lighthouse performance scoring. URL: https://developer.chrome.com/docs/lighthouse/performance/performance-scoring.

13 MDN Web Docs: The WebSocket API. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API.

14 Nginx Documentation: Beginner’s Guide. URL: https://nginx.org/en/docs/beginners_guide.html.

15 Vite: Next Generation Frontend Tooling. URL: https://vitejs.dev/guide/.

16 Vue Router: The official router for Vue.js. URL: https://router.vuejs.org/introduction.html.

17 web.dev: Metrics (Core Web Vitals). URL: https://web.dev/explore/metrics.

18 Git Documentation: Official reference. URL: https://git-scm.com/doc.

19 Strapi Blog: A Developer's Guide to WebP Image Format. RL: https://strapi.io/blog/developer-guide-webp.

20 Кодекс цивільного захисту України : Закон України від 02.10.2012 № 5403-VI. URL: https://zakon.rada.gov.ua/laws/show/5403-17 (дата звернення: 18.12.2025).

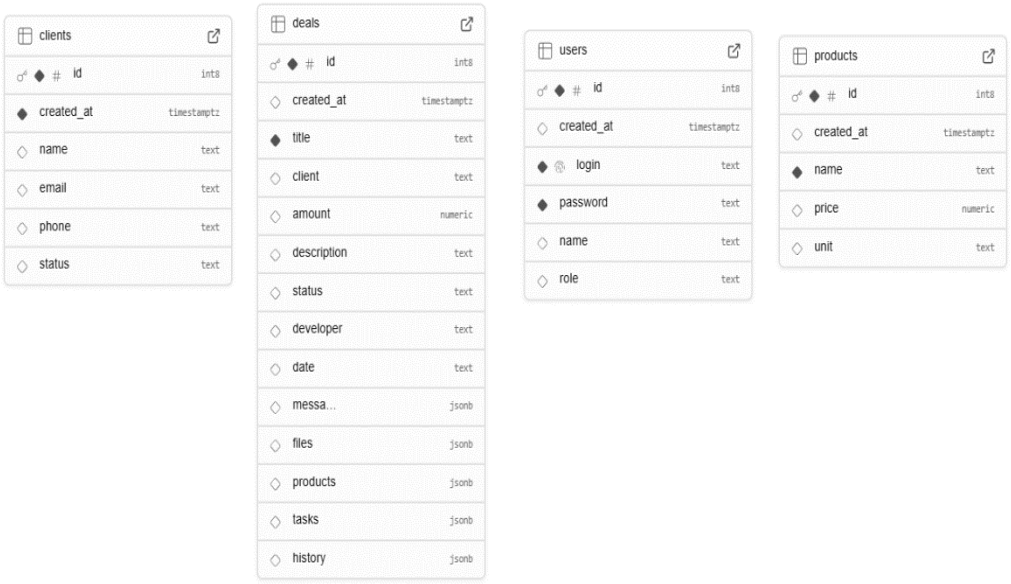
21 Про затвердження Правил пожежної безпеки в Україні : Наказ Міністерства внутрішніх справ України від 30.12.2014 № 1417. URL: https://zakon.rada.gov.ua/laws/show/z0252-15 (дата звернення: 18.12.2025).

22 Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями : Наказ Міністерства соціальної політики України від 14.02.2018 № 150. URL: https://zakon.rada.gov.ua/laws/show/z0244-18 (дата звернення: 18.12.2025).

ДОДАТКИ

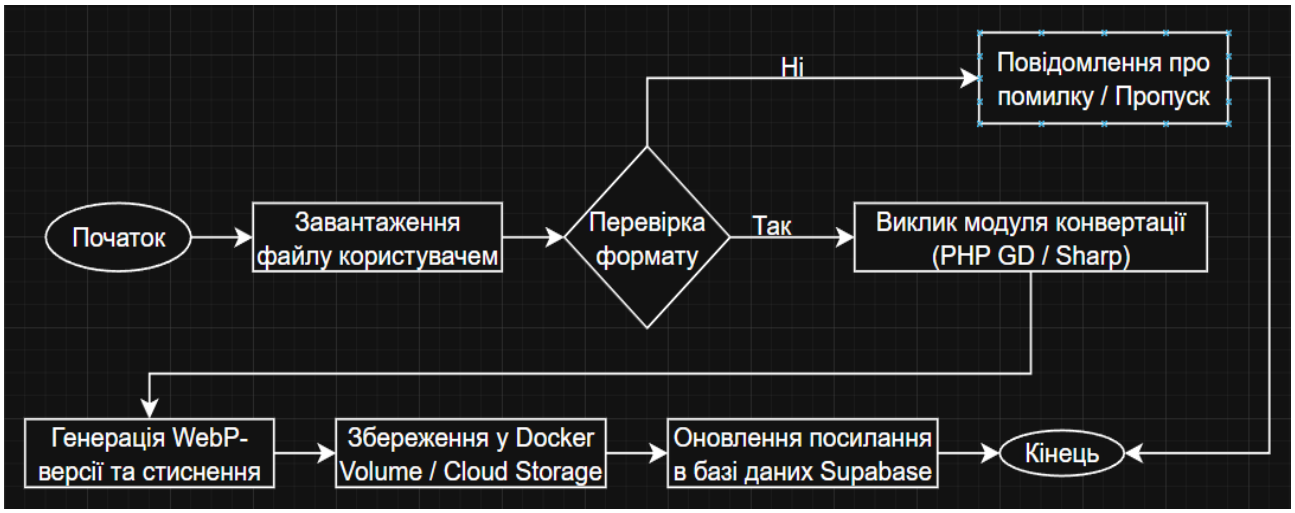
Додаток А

ER-діаграма бази данних



Додаток Б

Алгоритм автоматизованої конвертації зображень у WebP



Додаток В

Схема інфраструктури Docker-контейнеризації