Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

РОЗРОБКА ТА ДОСЛІДЖЕННЯ ПЛАТФОРМИ ДЛЯ АВТОМАТИЗАЦІЇ БІЗНЕС-ПРОЦЕСІВ З ВИКОРИСТАННЯМ ВЕБТЕХНОЛОГІЙ

DEVELOPMENT AND RESEARCH OF A PLATFORM FOR AUTOMATING BUSINESS PROCESSES USING WEB TECHNOLOGIES

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав:
здобувач вищої освіти групи ІПЗм-21
Сергієнко Олександр Олександрович

Керівник:
к.т.н., доцент
Суринович Олена Миколаївна

Кваліфікаційну роботу
допущено до захисту
«___» _____ 20__ р.
Гарант освітньої програми:
к.т.н., доцент Суринович О. М.

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *магістр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«___» _____ 202__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА
ДРУГОГО (МАГІСТЕРСЬКОГО) РІВНЯ ВИЩОЇ ОСВІТИ

Сергієнку Олександру Олександровичу

1. Тема кваліфікаційної роботи: Розробка та дослідження платформи для автоматизації бізнес-процесів з використанням вебтехнологій
Керівник роботи: Суринович Олена Миколаївна, доцент, к.т.н.
_____
затверджені наказом закладу вищої освіти від «29» березня 2025 року № 190/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи: 04 грудня 2025 р.
3. Вихідні дані до роботи теоретичні дані, що використовуються у предметній області кваліфікаційної роботи, практичні результати з дослідження, що отримано під час підготовки та проходження практики.
4. Зміст розрахунково-пояснювальної записки: аналіз предметної області та постановка задачі, аналіз інструментів розробки та реалізації програмного забезпечення, розробка програмного застосунку, інструкція для кінцевих користувачів, тестування розробки.
5. Перелік графічного матеріалу 42 рисунка, 1 лістинг коду

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблеми за темою роботи та постановка завдань дослідження	*Суринович О. М.*		
Теоретичне дослідження та практична реалізація	*Суринович О. М.*		
Експериментальне дослідження системи	*Суринович О. М.*		
Нормоконтроль	*Повстяна Ю. С.*		
Гарант ОП	*Андрущак І. Є.*		
Показник запозичень тексту	___%		
Академічна доброчесність	*Суринович О. М.*		

7. Дата видачі завдання *«2» квітня 2025 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи магістра	Строк виконання етапів роботи	Примітка
1	Провести огляд літературних джерел по темі кваліфікаційної роботи	02.05.2025	
2	Провести аналіз загальної проблеми і вибір напрямків дослідження	24.09.2025	
3	Розробити функціональну модель та архітектуру системи	01.11.2025	
4	Описати засоби розробки об’єкта проектування	19.11.2025	
5	Практична реалізація об’єкта проектування	26.11.2025	
6	Розробка тестів та проведення тестування розробленого об’єкта	05.11.2025	
7	Аналіз результатів та внесення змін	15.11.2025	
8	Здача чистового варіанту кваліфікаційної роботи на кафедру	04.12.2025	

Здобувач вищої освіти _____ Сергієнко О. О.

Керівник кваліфікаційної роботи _____ Суринович О. М.

АНОТАЦІЯ

Сергієнко О. О. Розробка та дослідження платформи для автоматизації бізнес-процесів з використанням вебтехнологій. Рукопис.

Кваліфікаційна робота магістра ОП «Інженерія програмного забезпечення» спеціальності 121 «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота магістра складається з вступу, 3 розділів, висновків, списку використаних джерел.

У роботі досліджено аналіз предметної області та постановка задачі, аналіз інструментів розробки та реалізації програмного забезпечення, розробка програмного застосунку, інструкція для кінцевих користувачів, тестування розробки.

Для створення програми застосували JavaScript разом із Bootstrap та Python використовуючи фреймворк Flask та Flask Restful. Для роботи з базою даних обрано ORM-рішення SQLAlchemy, яке забезпечує прозору підтримку більшості реляційних СУБД.

Ключові слова: управління відносинами з клієнтами, CRM, Python, JavaScript, програмне забезпечення, вебтехнології, Bootstrap, Flask, Pytest, FullCalendar, SQLAlchemy, ORM, реляційні бази даних, чиста архітектура.

ABSTRACT

Serhiienko O. O. Development and Research of a Platform for Automating Business Processes Using Web Technologies. Manuscript.

Qualifying Magister's Thesis of the EP «Software Engineering», specialty 121 «Software Engineering». Lutsk National Technical University. Lutsk, 2025.

The qualifying magister's thesis consists of an introduction, 5 sections, conclusions and proposals, a list of references, and a conference certificate.

The paper investigates the analysis of the subject area and problem statement, analysis of development tools and software implementation, software application development, instructions for end users, and development testing.

JavaScript with Bootstrap and Python using the Flask and Flask Restful frameworks were used to create the program. The SQLAlchemy ORM solution was chosen for working with the database, which provides transparent support for most relational DBMSs.

Keywords: Customer Relationship Management, CRM, Python, JavaScript, Software, Web Technologies, Bootstrap, Flask, Pytest, FullCalendar, SQLAlchemy, ORM, Relational Databases, Clean Architecture.

ЗМІСТ

ВСТУП

Актуальність теми. Розробка та дослідження платформи для автоматизації бізнес-процесів з використанням вебтехнологій зумовлена потребою у розробці адаптивної вебплатформи, яка дозволить автоматизувати типові бізнес-процеси (облік клієнтів, задач, продажів тощо) із можливістю подальшої персоналізації відповідно до конкретних потреб організації.

Оскільки кожна компанія має свої особливі вимоги та процеси, створення індивідуального програмного забезпечення стає все більш актуальним. Використання вебтехнологій дає можливість розробляти додатки, інтерфейс яких працює на будь-якому пристрої з вебпереглядачем.

Загалом, у сучасному світі зростає потреба у швидкій та гнучкій автоматизації бізнес-процесів, а програмне забезпечення, розроблене на сучасних мовах програмування та вебстеку, може допомогти зробити цей процес більш доступним й ефективним для широкого кола підприємств.

Мета роботи – розробка прототипу власної CRM-системи з чистою архітектурою, побудованої на сучасних вебтехнологіях, яка забезпечує базову функціональність для автоматизації бізнес-процесів, є легкою у розгортанні, а також розширюваною в перспективі.

Завдання дослідження:

- обґрунтувати актуальність проблеми та доцільність створення системи;
- сформулювати чіткі вимоги до розроблюваної системи;
- проаналізувати інструменти, необхідні для розробки системи;
- розробити систему відповідно до визначених вимог;
- здійснити автоматизоване тестування системи;
- здійснити перевірку захисту системи.

Об’єктом дослідження охоплює весь цикл створення програмного забезпечення – від формування завдання до його впровадження та експлуатації.

Він включає аналіз користувацьких вимог, розробку застосунку, тестування, впровадження та подальшу підтримку системи.

Предметом дослідження є конкретні аспекти такої розробки, як методи взаємодії з користувачем, використання вебтехнологій для створення інтуїтивного інтерфейсу та динамічної візуалізації даних, адаптація модульної системи під специфічні бізнес-процеси, аналіз ефективності та результативності такого програмного забезпечення в контексті цифрової автоматизації операцій у малих та середніх підприємствах.

Наукова новизна роботи полягає в ефективному використанні чистої архітектури при розробці системи обліку клієнтів, що забезпечує розподіл відповідальності, незалежність від зовнішніх компонентів, зручність тестування та масштабованість. Розроблена система дає змогу оптимізувати та автоматизувати ключові бізнес-процеси, зокрема управління клієнтською базою, продажі, маркетинг і клієнтський сервіс. Окрему увагу приділено створенню комплексної системи тестування на базі Pytest, що охоплює різні аспекти функціональності та гарантує високу якість і надійність розробленої платформи.

Практична цінність полягає у наданні малим та середнім підприємствам готового інструменту для швидкої автоматизації управління клієнтами, задачами та продажами, що легко розгорнути та налаштувати під власні процеси.

Апробація результатів дослідження. Сергієнко О. О. Суринович О. М. Дослідження та розробка платформи для автоматизації бізнес-процесів з використанням вебтехнологій. Science and Information Technologies in the Modern World: Collection of Scientific Papers with Proceedings of the 3rd International Scientific and Practical Conference. International Scientific Unity. October 1-3, 2025. Athens, Greece. P. 64-66. [1].

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМАТИКИ ЗА ТЕМОЮ РОБОТИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд і аналіз предметної області проблеми, результатів існуючих теоретичних та експериментальних досліджень

Існують багато програмних застосунків, що пропонують різні підходи та функціонал для автоматизації бізнес-процесів і дозволяють швидко організовувати управління клієнтською базою, планувати та контролювати задачі, вести продажі та отримувати аналітичні звіти. Серед найпопулярніших рішень, що вже успішно застосовуються у різних компаніях, варто виділити:

Salesforce – провідна хмарна CRM-платформа, яка надає потужний набір інструментів для управління продажами, маркетингом, сервісом та аналітикою (рис. 1.1). Ключовою особливістю є Salesforce Lightning Platform, що дозволяє розробникам створювати користувацькі додатки, автоматизовані процеси (Flow) та інтеграції через мову програмування Apex. Платформа підтримує масштабування під великі корпорації з можливістю розгортання приватних хмар (Salesforce Private Cloud) для підвищеної безпеки та контролю даних [2].

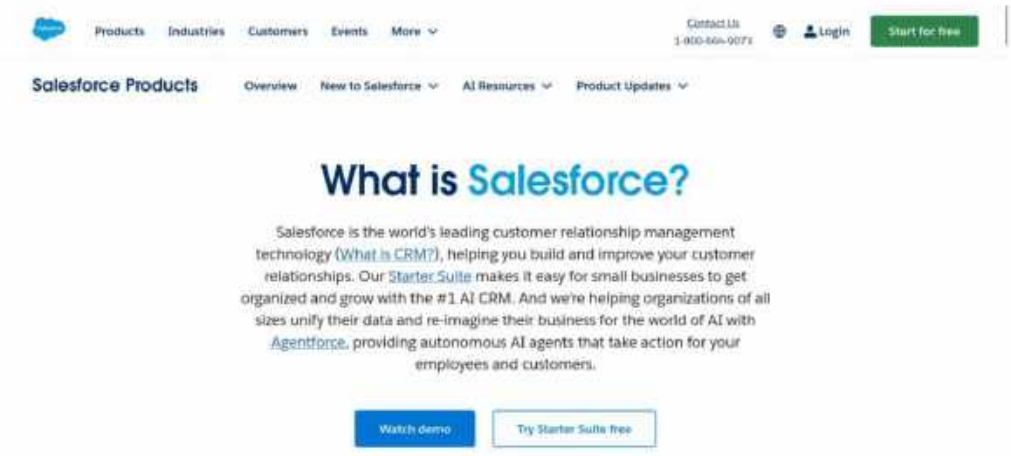

Рисунок 1.1 – Вебсторінка CRM-системи Salesforce [2]

Zoho CRM – це модульна система, орієнтовану на малий та середній бізнес з можливістю створювати власні скрипти автоматизації (рис. 1.2) [3].

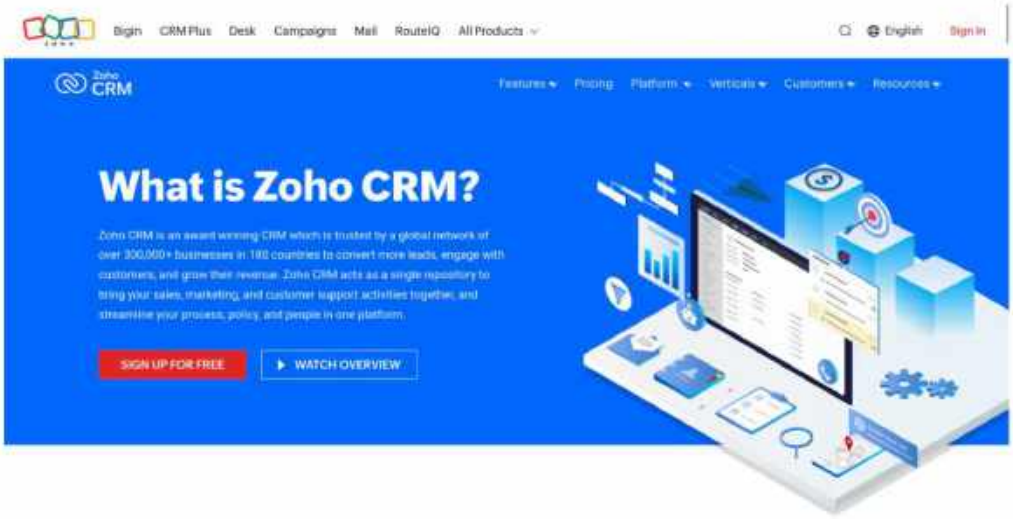

Рисунок 1.2 – Вебсторінка CRM-системи Zoho CRM [3]

HubSpot CRM – комплексну платформу, яка поєднує інструменти маркетингу, продажу та сервісу, пропонуючи гнучку аналітику, автоматичні розсилки та інтеграції з великою кількістю сторонніх сервісів [4].

Традиційний підхід до обліку клієнтів у багатьох компаніях базується на статичній таблиці даних, де кожен запис містить лише базову інформацію – ім'я, контактні дані та коротку нотатку. Така модель швидко втрачає актуальність, коли клієнт взаємодіє з організацією через різні канали (телефон, електронну пошту, соціальні мережі, чат-боти). Через відсутність уніфікованого сховища історії взаємодій, інформація розпорошується по різних системах, що ускладнює отримання повного уявлення про клієнта та прийняття обґрунтованих рішень [5].

Інший поширений метод – використання CRM-систем, орієнтованих на продажі, які зберігають історію дзвінків, листування та нотатки, проте часто не підтримують гнучку структуру даних. Користувачі змушені підлаштовуватися під жорстко задані правила, а додавання нових атрибутів (наприклад, інтереси,

сегментація за поведінковими ознаками) вимагає змін у базі даних та часто призводить до порушення сумісності з уже наявними звітами [5].

Третій підхід – сегментація клієнтів за допомогою аналітичних інструментів, які агрегують дані з різних джерел (ERP, маркетингові платформи, веб-аналітику) та формують профілі у вигляді окремих моделей. Хоча такий підхід дає глибоке розуміння клієнтської цінності, його реалізація вимагає складних процесів, регулярного оновлення даних і високих витрат на інфраструктуру. Крім того, без чіткого механізму синхронізації між основною базою виникає ризик розбіжностей у даних [5].

Усі розглянуті методи мають спільну проблему – відсутність єдиного, розширюваного механізму, який би дозволяв одночасно зберігати детальну історію взаємодій, підтримувати додаткові атрибути клієнтів й забезпечувати інтеграцію з аналітичними інструментами без значних технічних зусиль. Це створює потребу в системі, яка поєднує гнучкість відкритої схеми даних, модульність для додавання нових типів інформації та прості інтерфейси для синхронізації з зовнішніми сервісами.

1.2 Огляд і аналіз методів та засобів розробки платформи автоматизації бізнес-процесів для вирішення проблеми дослідження

Сучасні SaaS-рішення, такі як Salesforce, HubSpot, Zoho CRM, пропонують широкий функціонал, проте їх використання супроводжується низкою суттєвих обмежень.

По-перше, модель підписки створює постійні фінансові навантаження, що особливо важко для малих та середніх підприємств з обмеженим бюджетом.

По-друге, дані зберігаються у хмарі постачальника, що ускладнює контроль над конфіденційністю та відповідністю вимогам локального законодавства про захист інформації.

По-третє, персоналізація часто обмежена: додавання нових типів сутностей, бізнес-правил або інтеграцій вимагає використання власних

інструментів платформи (наприклад, мова програмування Apex у Salesforce) або придбання дорогих додаткових модулів, що підвищує вартість розробки [4].

Нарешті, у багатьох випадках користувачі стикаються з надмірною складністю інтерфейсу, що ускладнює швидке навчання персоналу та знижує ефективність впровадження. Ці проблеми створюють потребу в відкритій CRM-системі з модульною архітектурою, яка забезпечує контроль над даними.

1.3 Постановка завдання на кваліфікаційну роботу магістра

У рамках дослідження необхідно визначити, як створити CRM-платформу, що одночасно забезпечує гнучкість розширення, контроль над даними, простоту використання та постановити наступні завдання:

- обґрунтувати актуальність проблеми та доцільність створення системи;
- сформулювати чіткі вимоги до розроблюваної системи;
- проаналізувати інструменти, необхідні для розробки системи;
- розробити систему відповідно до визначених вимог;
- здійснити автоматизоване тестування системи;
- здійснити перевірку захисту системи.

Висновки до розділу 1

На основі сформульованих задач дослідження можна стверджувати, що успішна автоматизація бізнес-процесів вимагає архітектурного підходу, який розділяє бізнес-логіку, інфраструктурні сервіси та інтерфейс користувача. Така структура дозволяє безболісно додавати нові типи правила автоматизації та зовнішні інтеграції, зберігаючи цілісність системи.

Крім того, необхідно впровадити гнучку схему даних для управління історією взаємодій та користувацькими атрибутами клієнтів, що створює основу для точного аналітичного моніторингу. Безпечна автентифікація, ролі

користувачів і можливість розгортання у власному середовищі гарантують контроль над даними та відповідність вимогам конфіденційності.

В цілому, ці висновки підкреслюють важливість розробки програмного забезпечення автоматизації бізнес-процесів, яке поєднує індивідуальний підхід, ефективні методи обліку клієнтів та масштабовану, чисту архітектуру, що сприяє успіху підприємства.

РОЗДІЛ 2
ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Обґрунтування вибору шляхів, технологій, алгоритмів і засобів вирішення поставленого завдання

Вибір шляхів, технологій, алгоритмів і засобів для платформи з автоматизації бізнес-процесів є ключовим етапом, оскільки від цього залежить якість, швидкість та ефективність розробки.

Python – потужна, високорівнева мова програмування з широким спектром застосувань. Python, розроблений Гвідо ван Россумом у 1991 році, відомий своєю простотою, об'єктноорієнтованим підходом та динамічною типізацією. Управління пам'яттю здійснюється автоматично завдяки «збирачу сміття» (Garbage Collector), який ефективно звільняє непотрібні ресурси [6].

На відміну від багатьох інших мов, Python не повертає всю звільнену пам'ять операційній системі одразу; замість цього працює спеціальний менеджер пам'яті, який оптимально розподіляє блоки для малих об'єктів [7].

Серед головних переваг Python – великий набір вбудованих бібліотек, таких як os, sys, datetime, що дозволяють працювати з файлами, системними ресурсами та датою з часом. Крім того, існує безліч сторонніх бібліотек і фреймворків [8].

Flask – це легковаговий фреймворк для розробки серверної частини вебзастосунків на Python. Створений у 2010 році Арміном Ронахером (Armin Ronacher) й випущений як відкрите програмне забезпечення.

Фреймворк спрощує швидке створення вебзастосунків і вебсервісів завдяки простому API, побудованому на WSGI. Його синтаксис інтуїтивний, що дозволяє реалізовувати проєкти з мінімальними зусиллями. Модульна архітектура полегшує розширення функціональності за допомогою різноманітних розширень та плагінів [9].

Flask надає розробникам значну свободу у виборі структури проєкту та інструментів, має компактний розмір та мінімалістичний набір вбудованих можливостей, що дозволяє зосередитися на головних аспектах розробки. Чудова документація швидко вводить у роботу з фреймворком і допомагає освоїти його деталі.

Flask-RESTful – це розширення до Flask, яке спрощує створення REST-орієнтованих API для вебзастосунків на Python. Воно організовує програму у вигляді ресурсів, кожен з яких відповідає за певний об'єкт чи концепцію, і дозволяє працювати з ними за допомогою HTTP-методів (GET, POST, PUT, DELETE тощо). Розширення автоматично конвертує Python-об'єкти в JSON і навпаки, що полегшує обмін даними на рівні API [10].

Flask-RESTful має зрозумілий й лаконічний синтаксис, що дозволяє швидко розробляти та підтримувати REST-API без зайвих складнощів, а також надає зручні інструменти для тестування, що підвищує якість і надійність застосунку.

У підсумку, Flask-RESTful – потужний і зручний інструмент для створення REST API на Python у поєднанні з Flask; його простота та багатий функціонал роблять його відмінним вибором для будь-яких проєктів, які потребують вебсервісів або інтерфейсів взаємодії між додатками та сервісами.

SQLAlchemy – це ORM-бібліотека, створена Майклом Байером у 2006 році, яка підтримує різноманітні СУБД (SQLite, PostgreSQL, MySQL, MariaDB тощо) та дозволяє працювати з різними типами баз даних без суттєвих змін у коді. Завдяки об'єктноорієнтованому підходу замість написання складних SQL-запитів розробники оперують об'єктами Python, що спрощує розробку та підтримку [11].

Бібліотека пропонує високорівневий API для створення, читання, оновлення та видалення записів, підтримує транзакції для групування змін, захищає від SQL-ін'єкцій і надає інструменти для побудови складних запитів (з'єднання, фільтрація, сортування, групування) [12].

Використання SQLAlchemy робить роботу з базою даних ефективнішою, зручнішою та безпечнішою; його потужність і гнучкість роблять його відмінним вибором для будь-якого проєкту на Python, що потребує взаємодії з базою даних.

Pytest – популярний фреймворк для тестування на Python, який ідеально пасує до стека з Flask та SQLAlchemy. Він простий, гнучкий й дозволяє легко писати тести для перевірки логіки та API [13].

Серед переваг – зручний синтаксис assert з детальними звітами про помилки, підтримка fixtures для підготовки даних та plug-in архітектура для розширень. Pytest швидко інтегрується з системами CI/CD, забезпечуючи високу якість коду без зайвих зусиль [13].

Таким чином, pytest оптимальний для автоматизації тестування в цьому проєкті.

JavaScript – одна з найпопулярніших мов програмування, розроблена Бренданом Айхом у 1995 році. Вона використовується як на клієнтській стороні (у веббраузерах).

Завдяки великій екосистемі бібліотек та фреймворків, а також асинхронній моделі виконання, JavaScript дозволяє створювати швидкі та ефективні застосунки. Вибір цієї мови дає можливість розробляти різноманітні додатки для різних платформ та пристроїв, забезпечуючи ефективність, масштабованість і швидкість розробки [14].

FullCalendar – потужна JavaScript-бібліотека для створення інтерактивних календарів у вебзастосунках. Вона підтримує різні режими відображення (місяць, тиждень, день), drag-and-drop подій та інтеграцію з backend API [15].

Серед переваг – responsive дизайн, простота налаштування, велика кількість плагінів і чудова документація. FullCalendar легко інтегрується з Flask-RESTful для завантаження даних з сервера, забезпечуючи зручний інтерфейс користувача для планування бізнес-процесів [15].

Таким чином, FullCalendar оптимальний для реалізації календарних функцій у цьому проєкті.

2.2 Практична реалізація об'єкта проектування

В рамках кваліфікаційної роботи магістра було здійснено практичну реалізацію системи, що передбачало створення вебзастосунку з використанням обраних технологій та інструментів. Основною метою розробки було створення зручної, функціональної та масштабованої платформи для управління клієнтською базою та автоматизації бізнес-процесів

Процес розробки системи включає побудову архітектури, створення бази даних та реалізацію функціоналу. Розглянемо ключові технічні рішення, використані інструменти та підходи, що дозволили створити ефективну та масштабовану платформу.

В першу чергу слід визначити основну архітектуру вебдодатка. Архітектура (рис. 2.1) демонструє послідовну схему взаємодій між клієнтом, логікою застосунку та системою зберігання даних. У верхній частині розташований вебклієнт, який надсилає запити та отримує результати у вигляді підготовлених представлень або ресурсів. Усередині програмного ядра ці два компоненти виконують роль посередників між зовнішнім світом та внутрішніми механізмами застосунку, оскільки вони лише приймають дані, перетворюють їх у зручний для користувача чи API формат та передають далі без власної бізнес-логіки [16].

Після надходження запиту робота переходить до рівня сценаріїв використання, де формується цілісна поведінка системи відповідно до того, що саме потрібно виконати. Саме тут описується змістовна логіка того, як система має реагувати на дію користувача чи зовнішній виклик. Далі ініціатива переходить до службового шару, який обробляє внутрішні правила, перевірки, перетворення та узгодження даних, виконуючи роль фундаменту бізнес-логіки нижчого рівня [17].

Найближче до даних розміщується репозиторій, що створює абстракцію над конкретним механізмом зберігання даних та дозволяє застосунку взаємодіяти з інформацією без прив'язки до технологічних деталей. Завершує

цю схему система керування базами даних, яка фізично зберігає інформацію та відповідає за її цілісність та доступність [17].

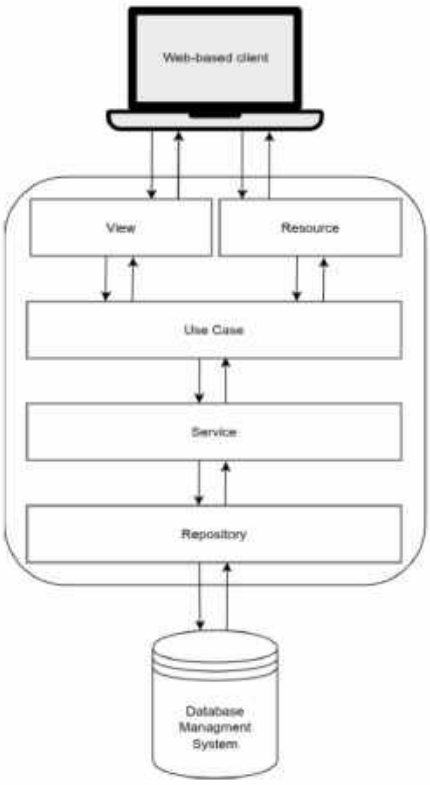

Рисунок 2.1 – Архітектура системи

Уся архітектура працює як чітко узгоджений потік: клієнт ініціює взаємодію, представлення передає запит сценарію, сценарій формує логіку дії, служба виконує обробку, репозиторій звертається до даних, а результат рухається тим самим шляхом у зворотному напрямку, доки не потрапляє назад до користувача. Такий підхід забезпечує зрозумілу структурність і дозволяє вдосконалювати чи змінювати будь-який компонент без руйнування всієї системи.

Вибір бази даних та її проєктування становлять ключові етапи створення додатка. Сьогодні доступна велика різноманітність баз даних – як реляційних систем керування даними, так і нереляційних. База даних впливає не лише на спосіб зберігання інформації, а й на механізми доступу до неї, тому її підбір є критичним аспектом розробки [18].

Загалом, проєктування бази даних – це процес розробки її схеми, яка забезпечує підтримку роботи всієї системи [19].

Діаграма бази даних у цій системі зображена на рисунку 2.2, де основні сутності – це користувачі, клієнти, події та послуги. Кожна таблиця відповідає за конкретний тип даних, а зв'язки між ними описують реальні взаємодії у бізнес-процесах.

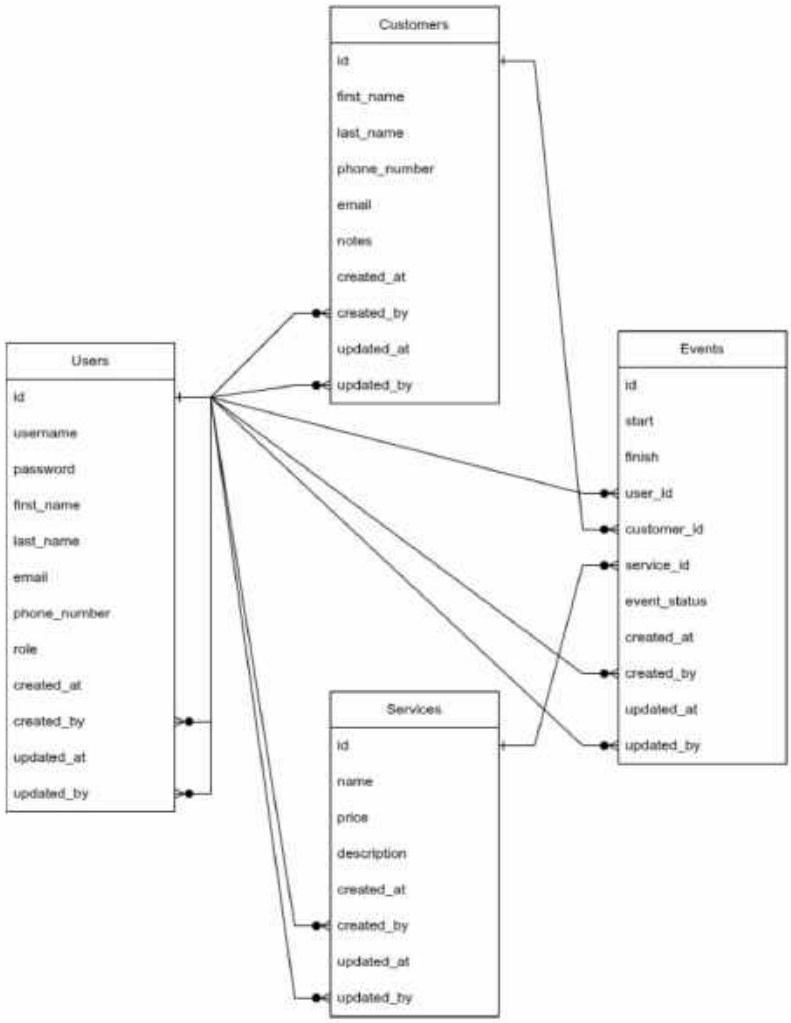

Рисунок 2.2 – Діаграма сутностей бази даних та їх відношення одних з одним

У центрі логіки перебуває таблиця користувачів, адже саме працівники системи створюють записи, змінюють їх та керують діяльністю. Кожен користувач має свою роль – адміністратор, менеджер або виконавець – що визначає рівень повноважень. Хоч ці ролі реалізовані технічно як перелік

констант у коді, на рівні бази даних вони виступають просто значенням поля, яке впливає на поведінку системи [19].

Клієнти описані окремою сутністю, оскільки вони є зовнішніми учасниками процесів. Для них зберігаються контактні дані та нотатки, які можуть бути корисними під час роботи менеджерів. Зв'язок між користувачами та клієнтами вказує, хто саме створив або відредагував відповідний запис, тому в кожній таблиці присутні поля «created_by» та «updated_by». Це робить дані прозорими й дозволяє завжди визначити відповідальну особу.

Послуги формують каталог того, що може запропонувати компанія. Їхня структура проста: назва, короткий опис й ціна. Так само як і в інших сутностях, зберігається інформація про створення та оновлення, що допомагає відстежувати зміни протягом роботи системи.

Найдинамічнішою частиною є події. Це окремі випадки взаємодії між клієнтом, працівником та послугою. Подія має дату початку, дату завершення, поточний стан та посилання на всі пов'язані сутності. Статус події інтерпретується в застосунку у зручному для людини вигляді: «активна», «скасована» або «виконана». У коді ці стани описані як простий перелік, але в базі даних вони зберігаються як текстові значення, що дозволяє застосунку коректно визначати їхній зміст.

Загалом структура бази даних побудована так, щоб у системі було легко орієнтуватися: користувачі відповідають за обслуговування клієнтів, клієнти замовляють послуги, а події фіксують конкретні приклади такої взаємодії. Водночас технічні деталі або механізми контролю змін, сховані за простими для розуміння полями, що робить базу даних не лише робочою, а й зрозумілою для будь-якого розробника.

Розробка графічного інтерфейсу користувача розпочалася з етапу прототипування в інструменті Penpot, який дозволив швидко створити макети. Penpot надав зручний візуальний редактор для моделювання ключових екранів: форми додавання клієнтів з полями для контактних даних та нотаток, календар подій, списки послуг [20].

На рисунку 2.3 зображено приклади прототипів.

Рисунок 2.3 – Penpot, інструмент для прототипування графічного інтерфейсу

Верстка інтерфейсу реалізовано за допомогою HTML, CSS та JavaScript з інтеграцією FullCalendar для календарних компонентів. Структура сторінок побудована на семантичних тегах для доступності [21].

На рисунку 2.4 зображено сторінку авторизації користувача, а на рисунку 2.5 – сторінку реєстрації. Обидві сторінки мають приємні для ока фонові зображення з пейзажем, що створює відчуття легкості та спокою. Форми розташовані у центрі екрана на білому тлі, що забезпечує чіткий контраст та концентрацію уваги користувача на полях для введення даних. Кнопки виконані у зеленому кольорі, який асоціюється з позитивними діями та підтвердженням. Використано мінімалістичний дизайн з акцентом на зручність та простоту використання. Шрифт обрано чітким та легко читабельним, що підвищує зручність взаємодії з інтерфейсом [22].

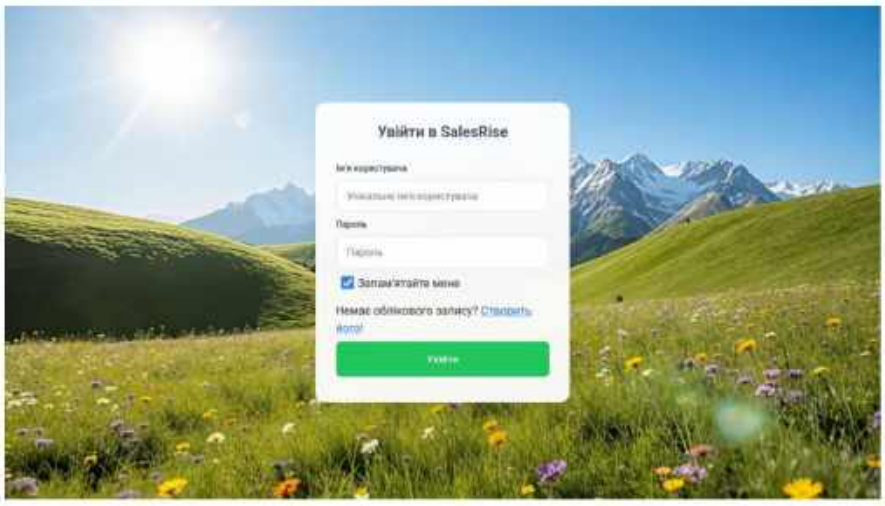

Рисунок 2.4 – Сторінка авторизації користувача

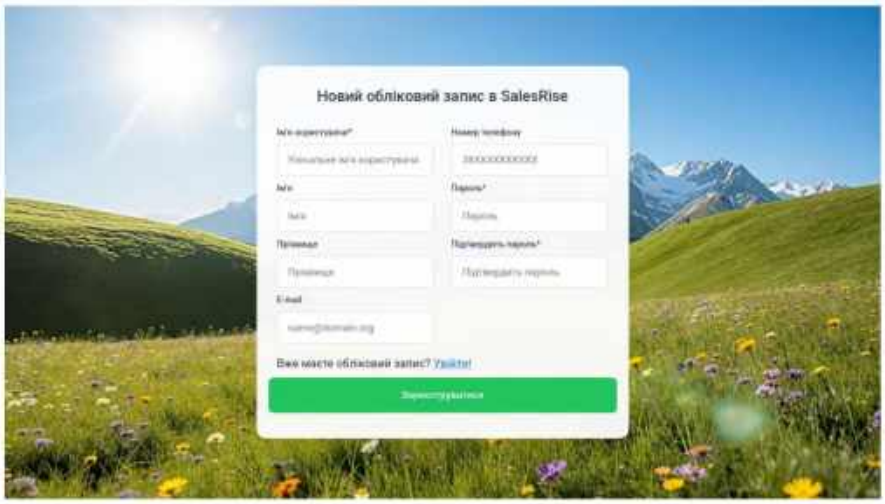

Рисунок 2.5 – Сторінка реєстрації користувача

На рисунках 2.6 та 2.7 зображена домашня сторінка з закритим та відкритим боковим меню відповідно. Дизайн сторінки виконаний в м'яких тонах з використанням фонового зображення, що створює відчуття затишку та професіоналізму. Бокове меню забезпечує зручну навігацію по розділах системи, а його відкриття/закриття дозволяє користувачеві налаштувати інтерфейс під свої потреби [23]. Верхня панель містить посилання на профіль користувача та вихід з системи. Вітальний текст звертається до користувача на ім'я, що створює відчуття персоналізації.

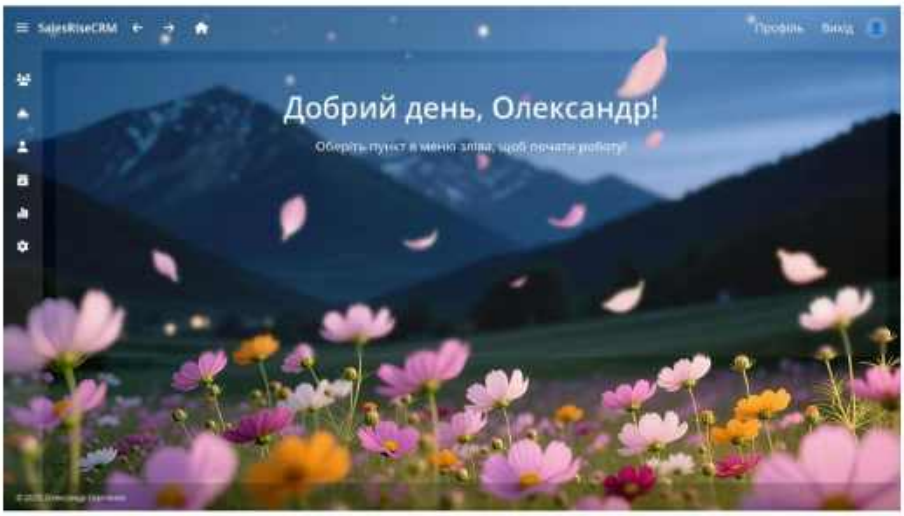

Рисунок 2.6 – Домашня сторінка з закритим боковим меню

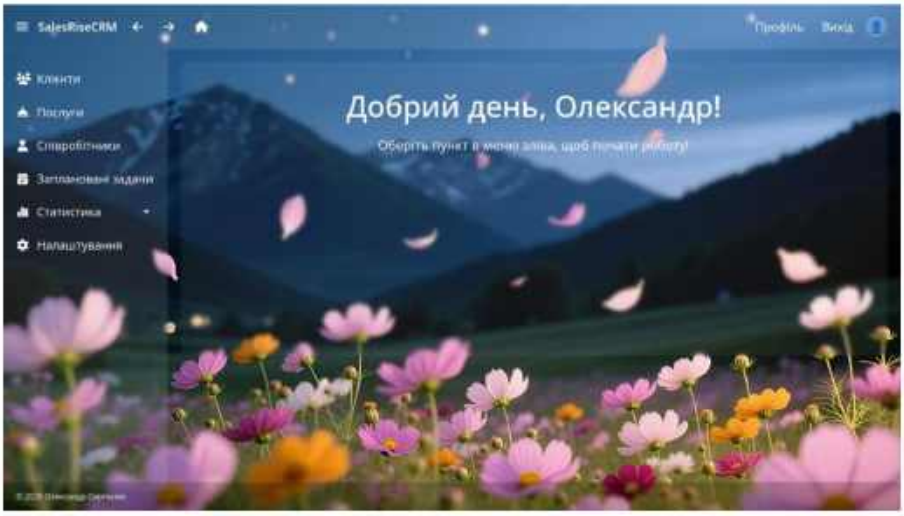

Рисунок 2.7 – Домашня сторінка з відкритим боковим меню

На рисунку 2.8 зображено сторінку списку користувачів системи. Дизайн списку виконано в стилі таблиці з чіткими заголовками стовпців. Кожен рядок таблиці містить інформацію про користувача, а також кнопки для редагування та видалення запису. Пагінація дозволяє зручно переміщатися між сторінками списку [23].

На рисунку 2.9 представлено сторінку створення нового користувача. Форма містить необхідні поля для введення даних, а кнопка «Створити користувача» дозволяє зберегти новий запис.

Рисунок 2.10 демонструє сторінку редагування користувача. Форма заповнена даними поточного користувача, що дозволяє легко внести зміни та зберегти їх натисканням кнопки «Оновити користувача».

Всі сторінки виконані в єдиному стилі з використанням фонового зображення та акценту на зручності та простоті використання [23].

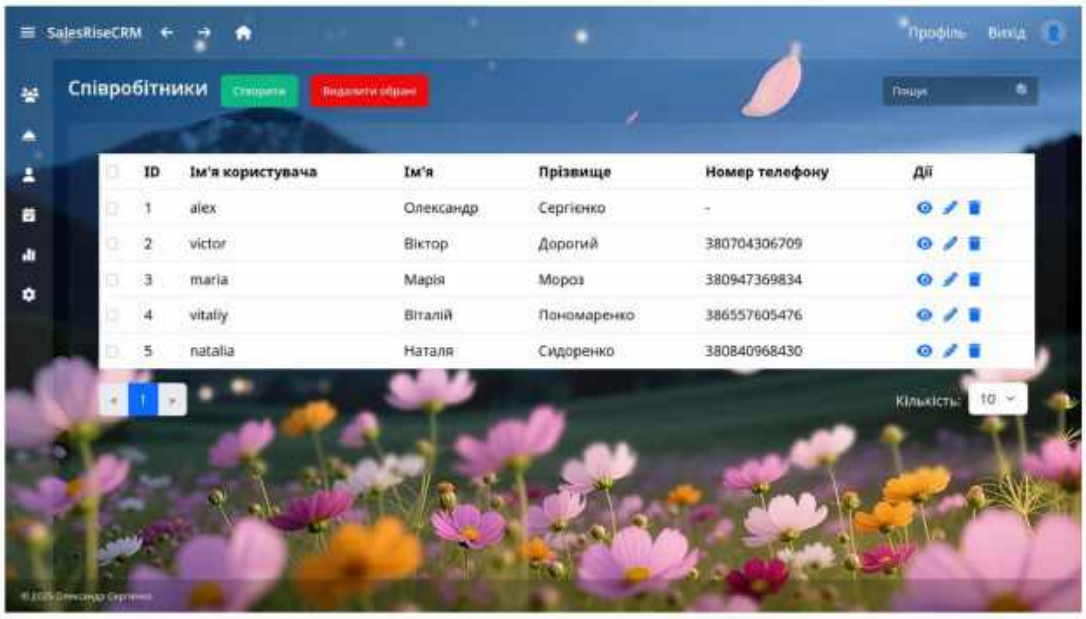

Рисунок 2.8 – Сторінка показу користувачів системи

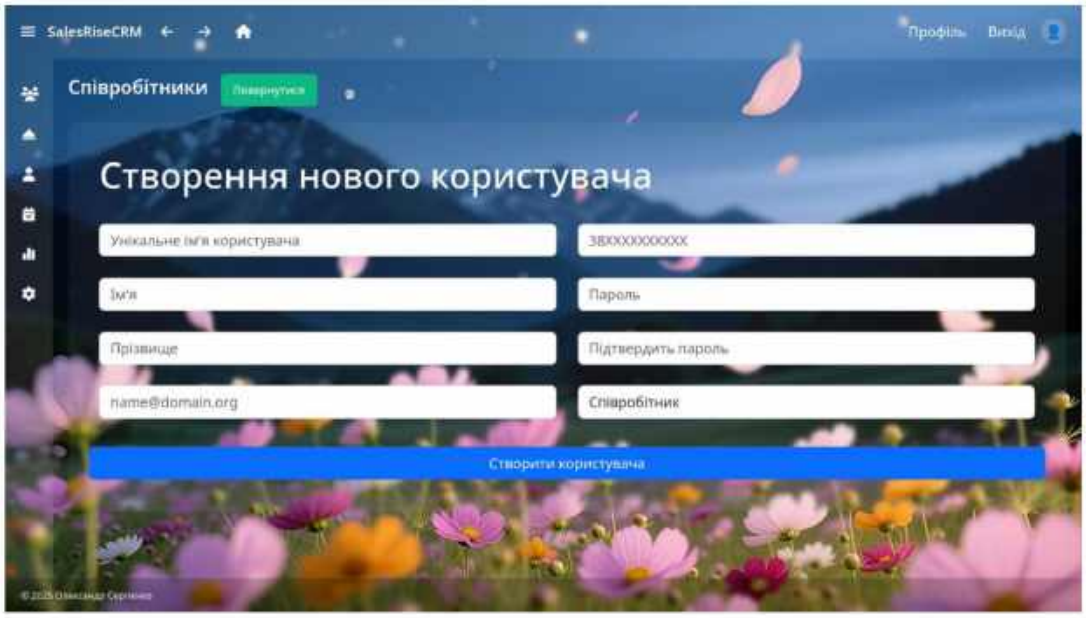

Рисунок 2.9 – Сторінка створення нового користувача

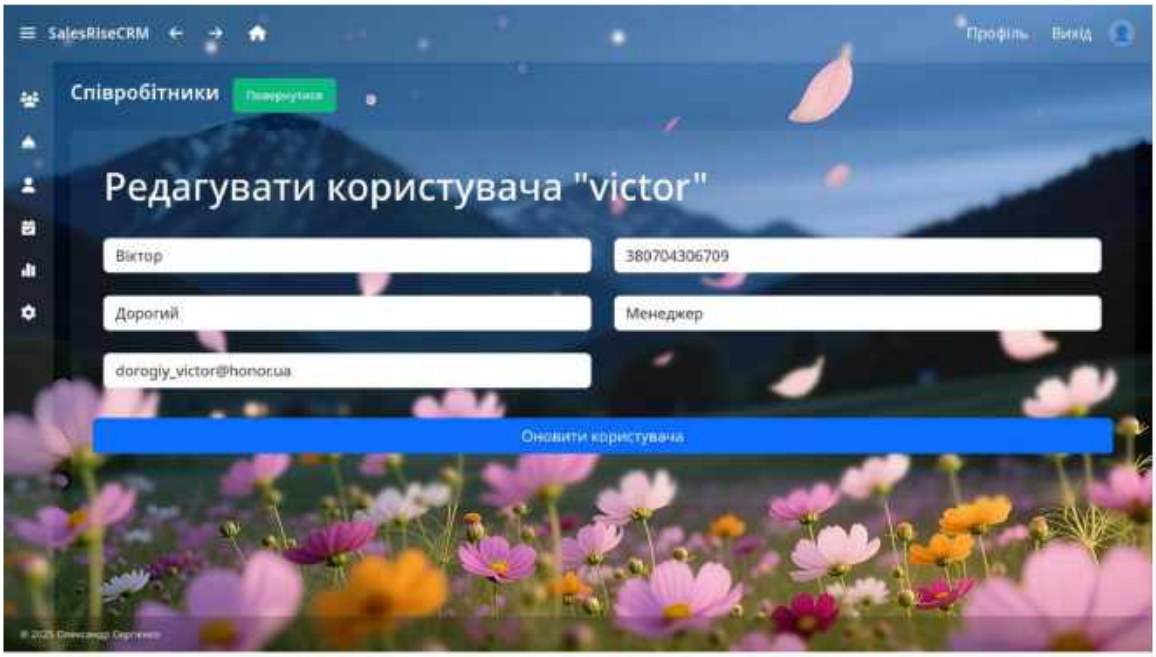

Рисунок 2.10 – Сторінка редагування користувача

На рисунку 2.11 зображено сторінку списку клієнтів. Список представлений у вигляді таблиці, яка дозволяє зручно переглядати та керувати інформацією про кожного клієнта. Кожен клієнт має кнопки для швидкого доступу до редагування та видалення, що спрощує процес управління даними. Пагінація полегшує навігацію по великому списку клієнтів, дозволяючи користувачам легко знаходити потрібну інформацію.

На рисунку 2.12 представлена сторінка створення нового клієнта. Форма створення клієнта містить мінімальний, але достатній набір полів для швидкого додавання запису, що робить процес максимально простим та ефективним.

Рисунок 2.13 демонструє сторінку редагування клієнта. Тут користувач може змінити необхідну інформацію про клієнта та зберегти зміни.

Як і інші сторінки, ці виконані в єдиному стилі з використанням фонового зображення, що створює приємну візуальну атмосферу та надає цілісності інтерфейсу.

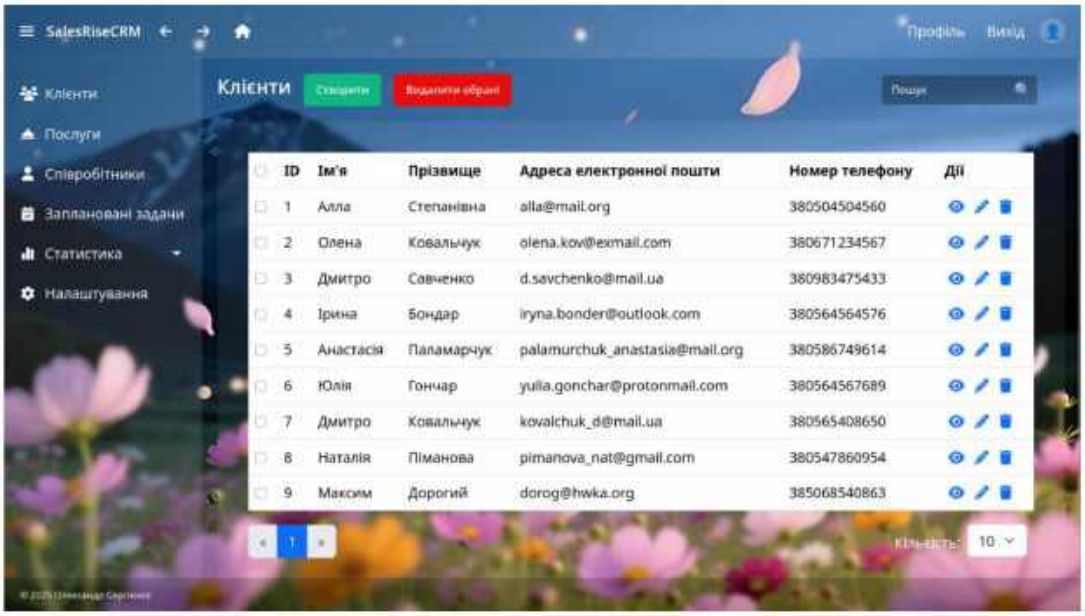

Рисунок 2.11 – Сторінка показу клієнтів

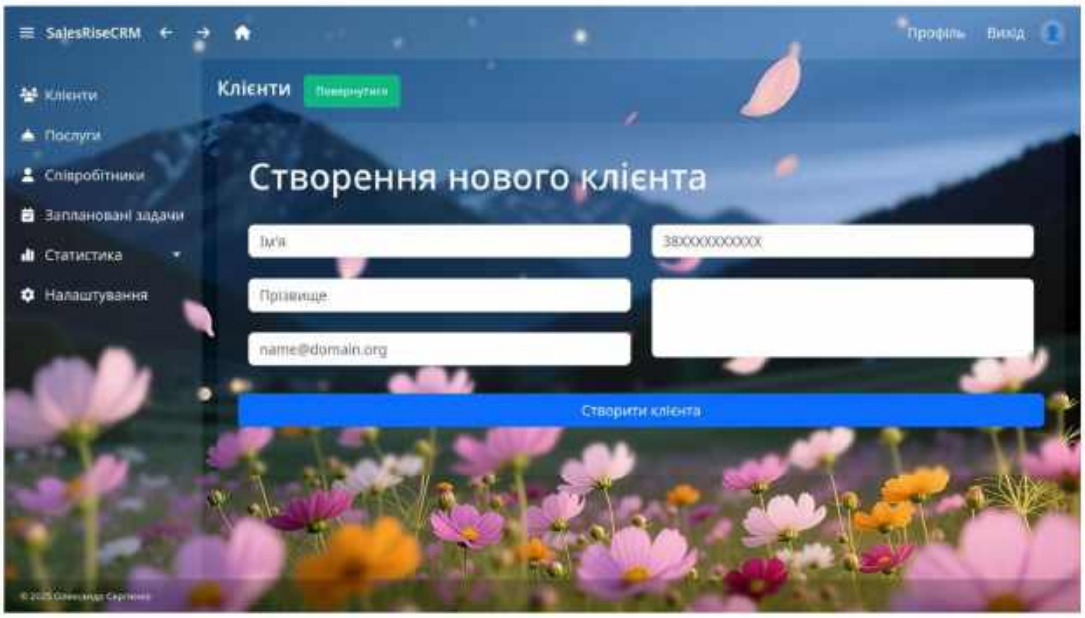

Рисунок 2.12 – Сторінка створення клієнта

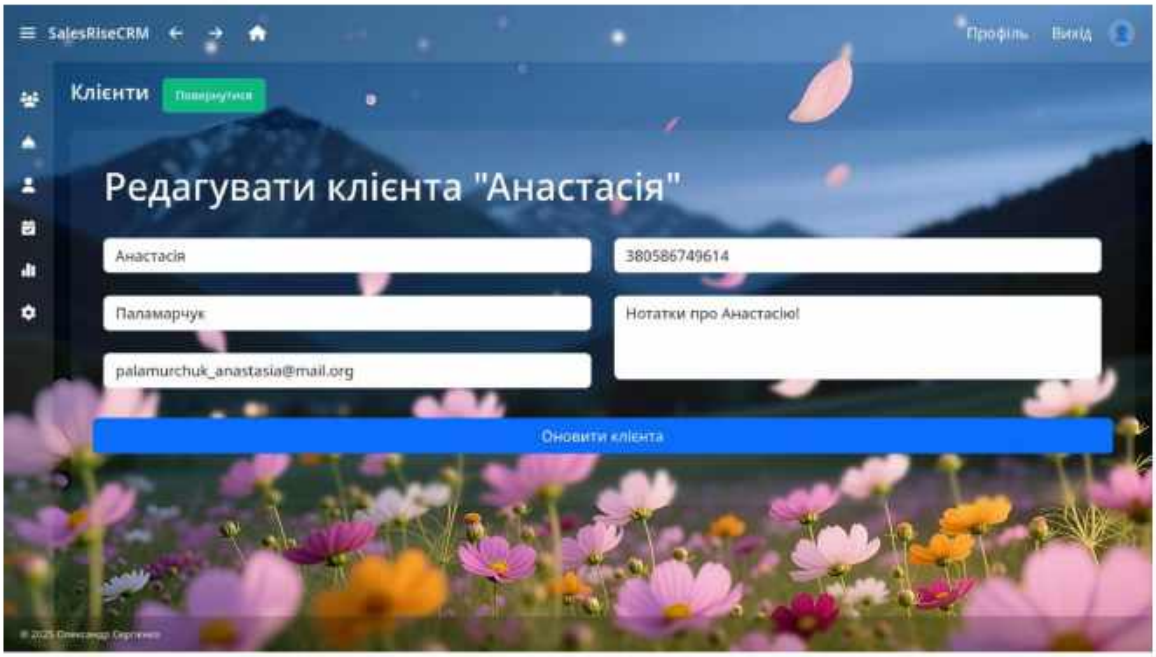

Рисунок 2.13 – Сторінка редагування клієнта

На рисунку 2.14 зображена сторінка списку послуг, представлених у вигляді таблиці. Таблиця має чітку структуру з інформацією про назву послуги, ціну, дати створення та оновлення. Зручні кнопки редагування та видалення дозволяють швидко керувати послугами. Наявна пагінація допомагає комфортно переміщатися між великим переліком послуг [23].

Рисунок 2.15 показує сторінку створення нової послуги. Форма створення нової послуги має прості поля для введення назви, ціни та опису. Лаконічний дизайн сприяє зосередженню на введенні інформації про нову послугу [23].

Рисунок 2.16 демонструє сторінку редагування послуги. Форма для редагування послуги зручно заповнена поточними даними, що робить процес внесення змін швидким та інтуїтивно зрозумілим. Можливість редагувати назву, ціну та опис послуги робить сторінку незамінною для підтримки актуальності інформації [23].

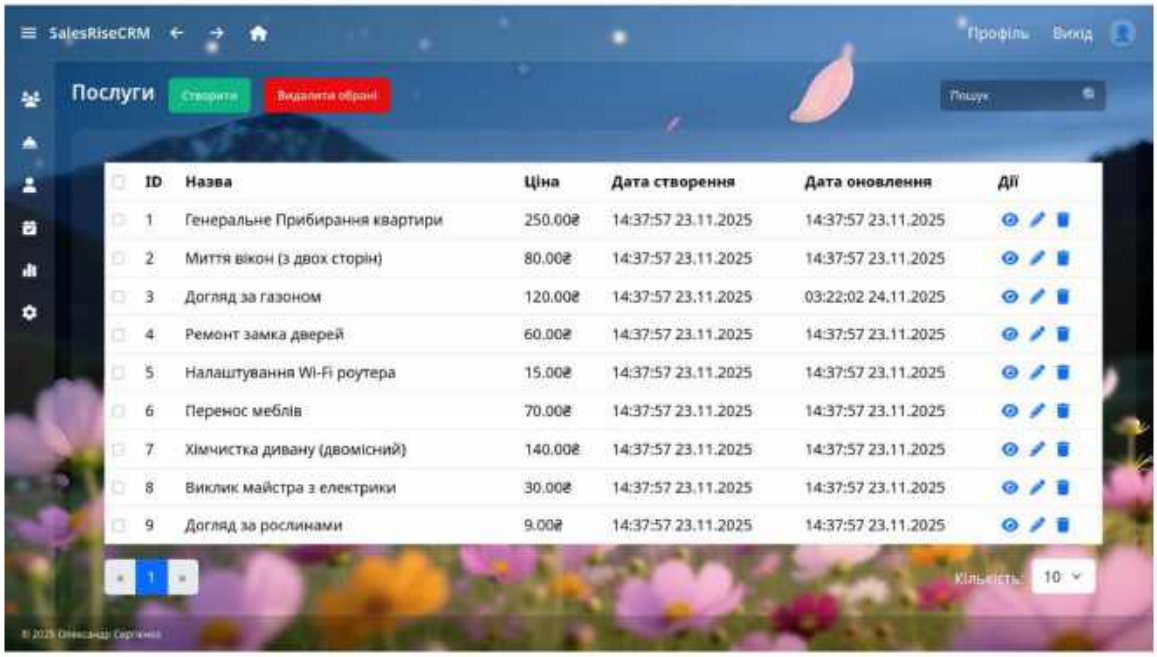

Рисунок 2.14 – Сторінка показу послуг

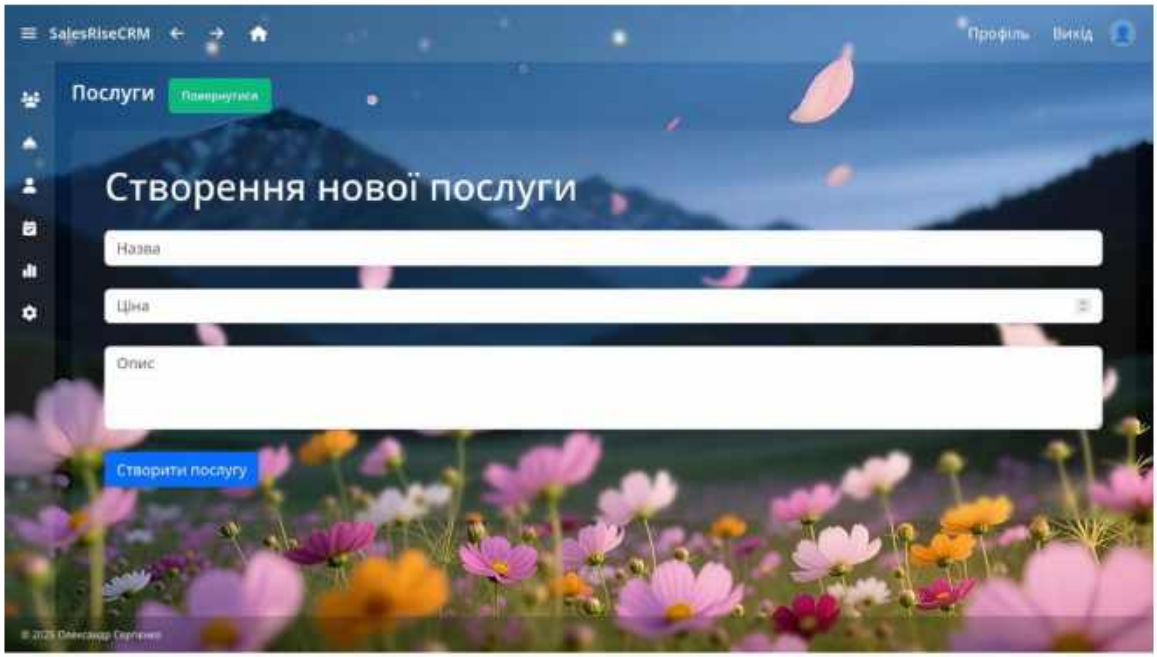

Рисунок 2.15 – Сторінка створення послуги

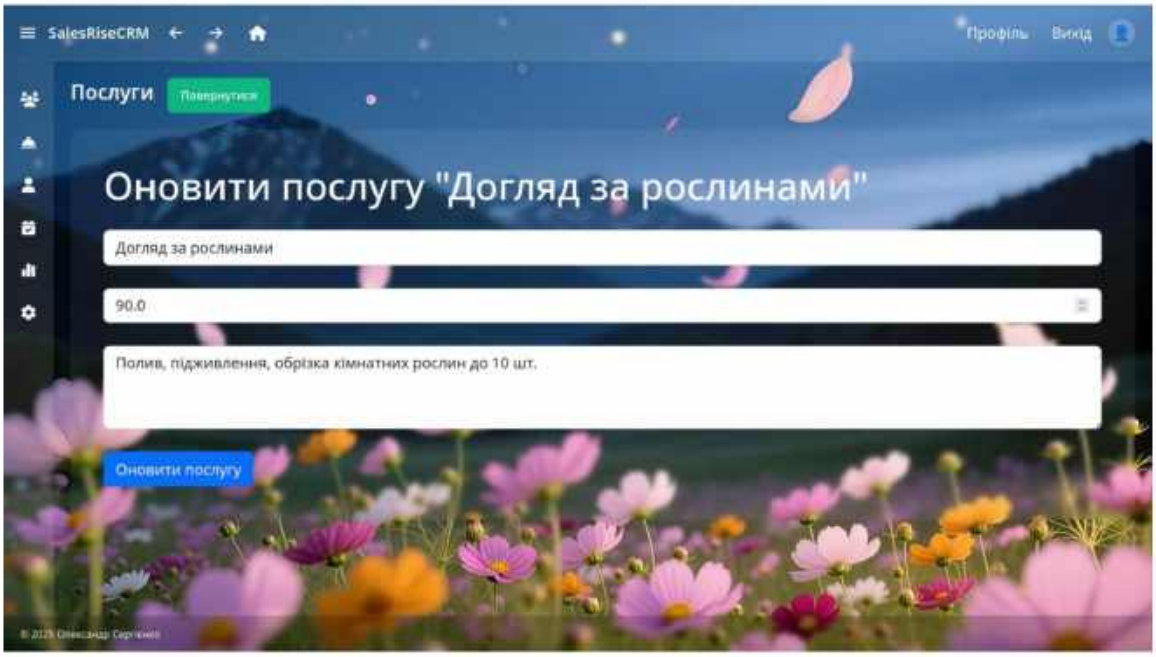

Рисунок 2.16 – Сторінка редагування послуги

На рисунку 2.17 зображена сторінка запланованих задач. Календарний інтерфейс на базі FullCalendar дозволяє переглядати задачі на тиждень, з можливістю переходу до окремого дня (рис. 2.18). Задачі відображаються у відповідних часових слотах, що робить планування наочним та зручним. Навігація між тижнями здійснюється за допомогою стрілок, що знаходяться у правому верхньому куті екрану.

Рисунок 2.18 демонструє перегляд задач за окремий день. Такий детальний перегляд дозволяє сконцентруватися на задачах, які необхідно виконати саме сьогодні. Усі елементи інтерфейсу гармонійно поєднані, створюючи зручний та інтуїтивно зрозумілий інструмент для управління задачами [22].

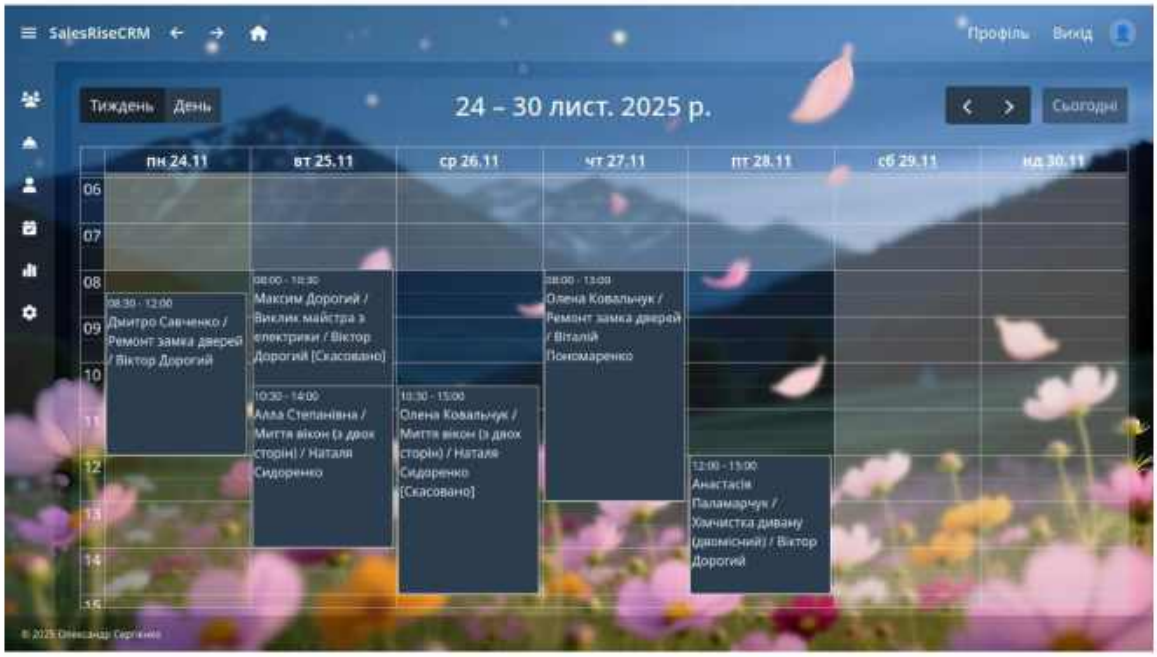

Рисунок 2.17 – Сторінка запланованих задач

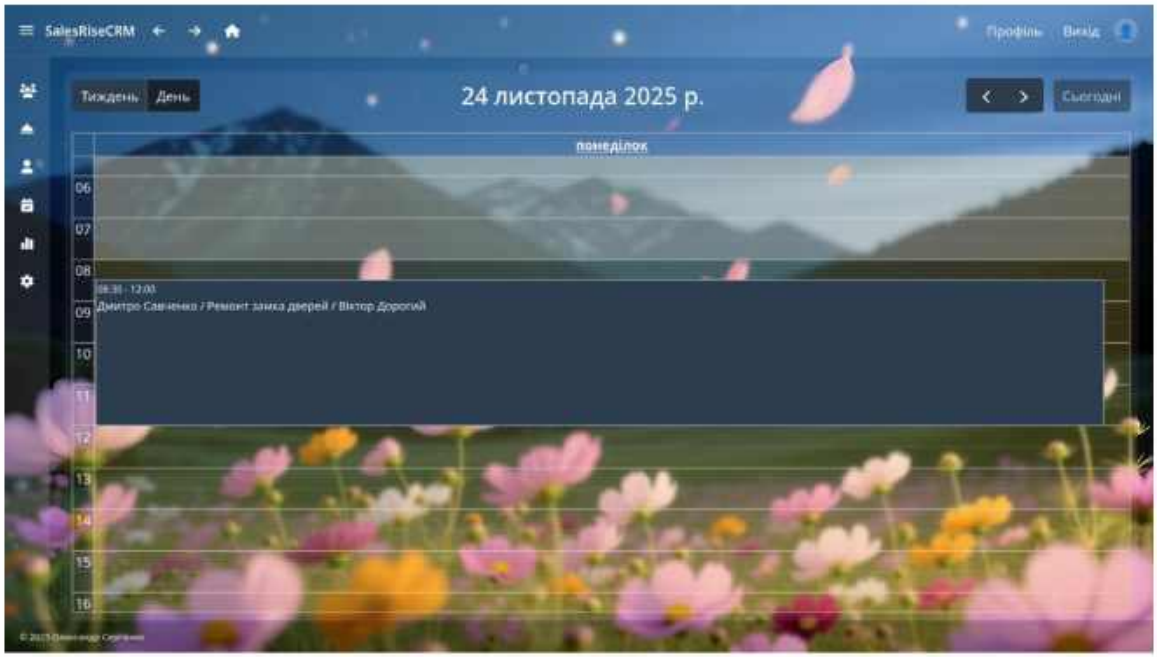

Рисунок 2.18 – Сторінка запланованих задач за днем

На рисунку 2.19 показано модальне вікно, яке з’являється при виділенні пустого місця в календарі. Воно містить поля для вибору клієнта, послуги та співробітника, а також налаштування дати та часу події. Випадаючи списки

спрощують процес вибору, а кнопки «Зберегти» та «Скасувати» дозволяють зберегти подію або скасувати створення.

На рисунку 2.20 представлено модальне вікно, яке з'являється при натисканні на вже заплановану подію. Воно містить ті ж поля, що й вікно створення, але вже заповнені інформацією про обрану подію. Крім того, додано кнопку «Видалити», яка дозволяє видалити подію з календаря.

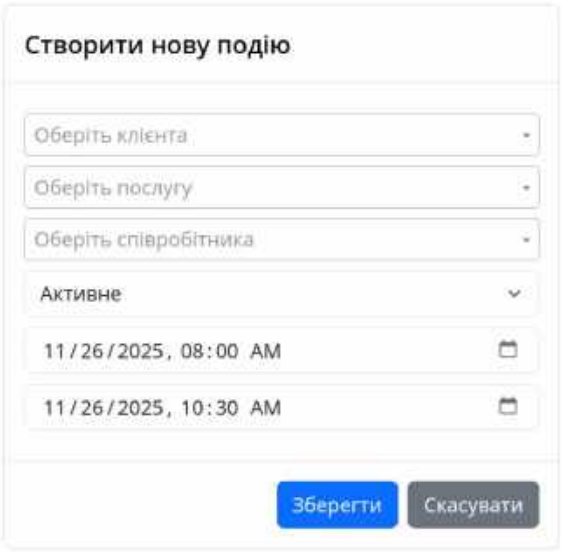

Рисунок 2.19 – Модальне вікно для створення події

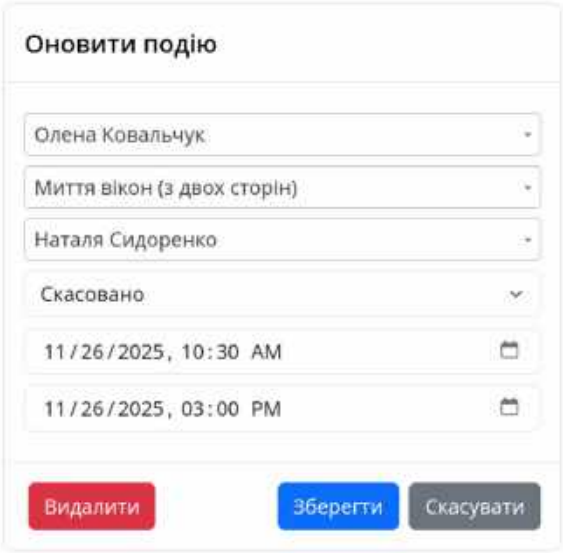

Рисунок 2.20 – Модальне вікно для редагування події

На рисунку 2.21 зображено сторінку загальної статистики, яка містить ключові показники діяльності системи. У верхній частині сторінки розташовані фільтри для вибору співробітника та послуги, а також діапазон дат для аналізу. Основну частину сторінки займають графіки та діаграми, що відображають статистику наданих послуг та їх загальний статус. Також в нижній частині сторінки розміщена таблиця з детальною інформацією про кожну подію.

На рисунку 2.22 представлена сторінка статистики клієнтів. У верхній частині розміщені фільтри для вибору співробітника, послуги та клієнта, а також фільтр за статусом. Основну частину сторінки займає таблиця з інформацією про клієнтів, послуги, співробітників, статус та дати замовлень. Сторінки статистики виконані в єдиному стилі з використанням чітких графіків та таблиць, що дозволяє легко аналізувати дані та приймати обґрунтовані рішення.

Рисунок 2.21 – Сторінка загальної статистики

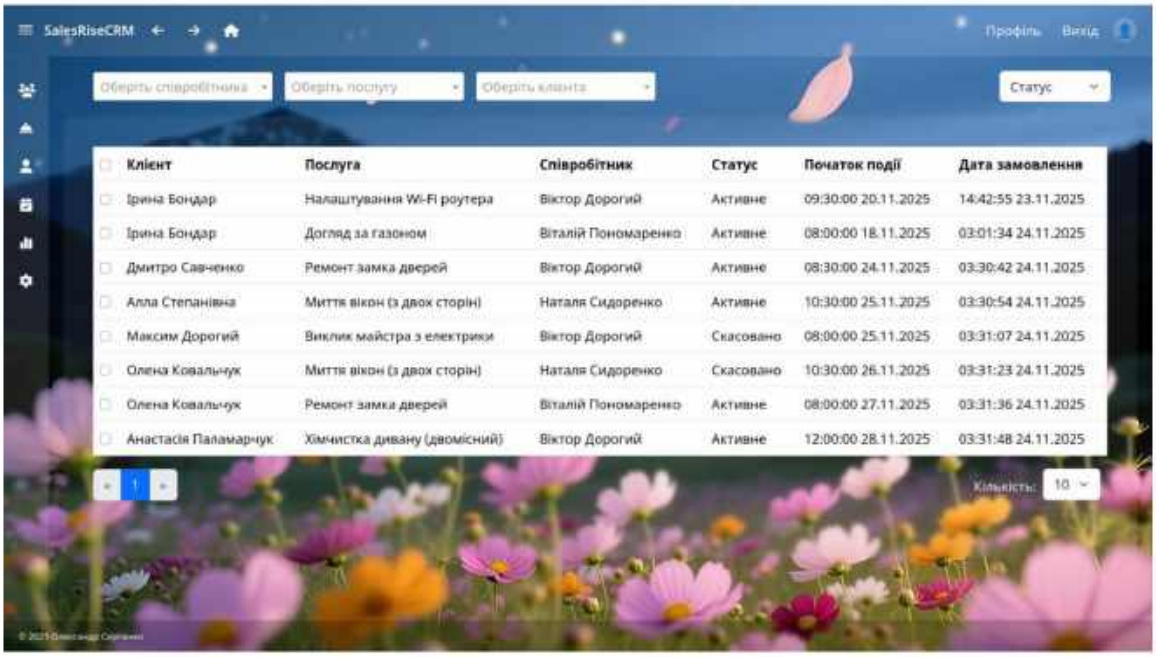

Рисунок 2.22 – Сторінка статистики клієнтів

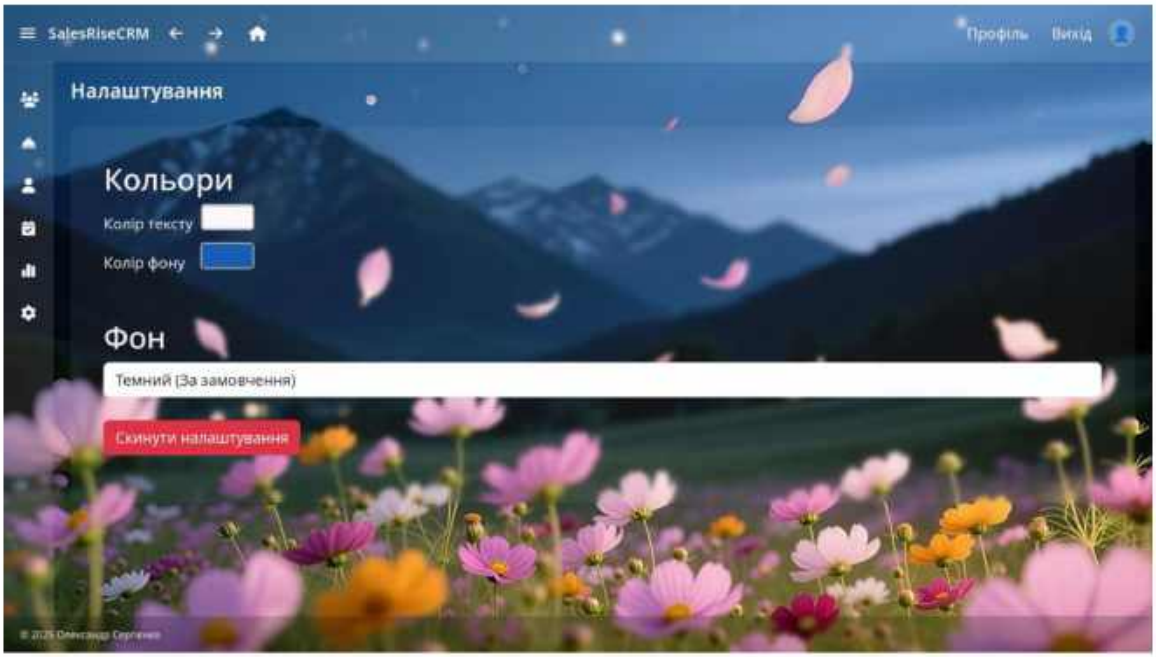

Рисунок 2.23 – Сторінка налаштування інтерфейсу та розширень

У верхній частині сторінки розташовані налаштування кольорів тексту та фону інтерфейсу, що дозволяє користувачам налаштувати зовнішній вигляд системи під свої вподобання. Трохи нижче розташований вибір фонового зображення. Також є кнопка для скидання налаштувань до початкових.

Інтерфейс виконаний в мінімалістичному стилі з акцентом на зручність та простоту налаштування [22].

Інструкція допоможе користувачам швидко освоїти основні функції та почати ефективно використовувати платформу. Розглянемо процес реєстрації, авторизації, додавання та редагування записів, планування завдань та відстеження статистики.

Для початку роботи з системою необхідно зареєструватися. Для цього потрібно перейти на сторінку реєстрації (рис. 2.24) та заповнити поля: ім'я користувача, номер телефону, ім'я, прізвище, email, пароль та підтвердження пароля. Після заповнення потрібно натиснути кнопку «Зареєструватися».

Якщо користувач вже зареєстрований, то він зможе увійти у систему через сторінку авторизації (рис. 2.25). Потрібно ввести ім'я користувача та пароль, що був вказаний при реєстрації, та натиснути кнопку «Увійти».

Після успішної авторизації або реєстрації користувач потрапить на головну сторінку системи (рис. 2.26), де він зможе почати роботу з усіма її функціями.

Рисунок 2.24 – Реєстрація користувача

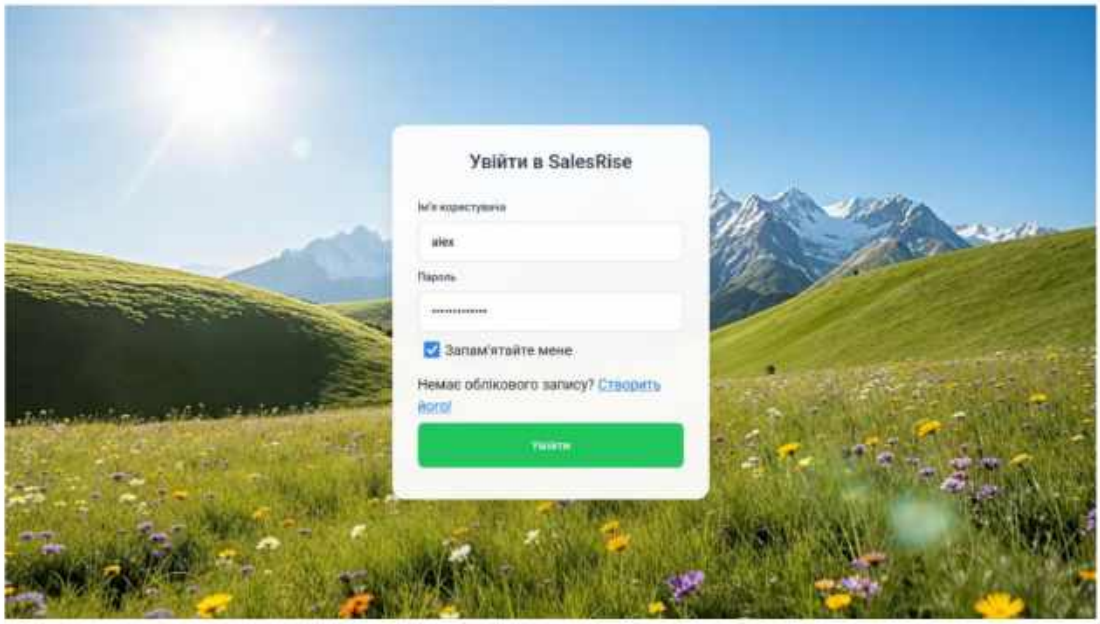

Рисунок 2.25 – Авторизація користувача

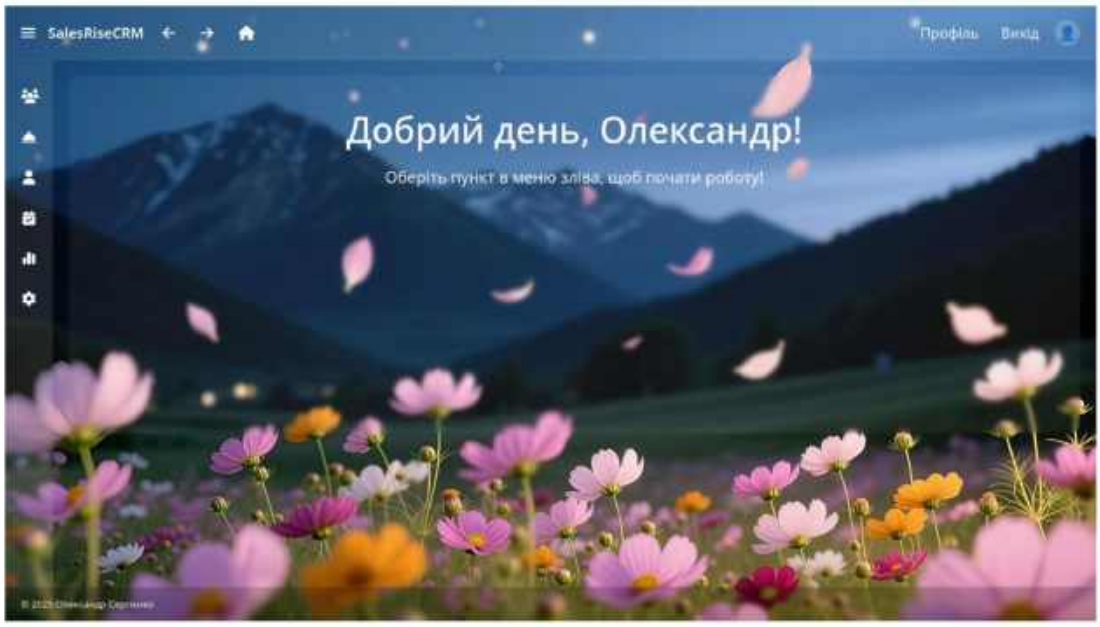

Рисунок 2.26 – Головна сторінка, що свідчить про успішну авторизацію

Процес реєстрації та авторизації максимально спрощений та інтуїтивно зрозумілий, що дозволяє користувачам швидко почати роботу з системою.

Для додавання нового запису (наприклад, клієнта або послуги) потрібно перейти до відповідного розділу (наприклад, «Клієнти» або «Послуги») та

натиснути кнопку «Створити». Заповнити необхідні поля у формі та зберегти (рис. 2.27).

Для редагування запису потрібно знайти його у списку та натисніть кнопку «Редагувати». Внесіть необхідні зміни у формі та натисніть кнопку «Зберегти» (рис. 2.28).

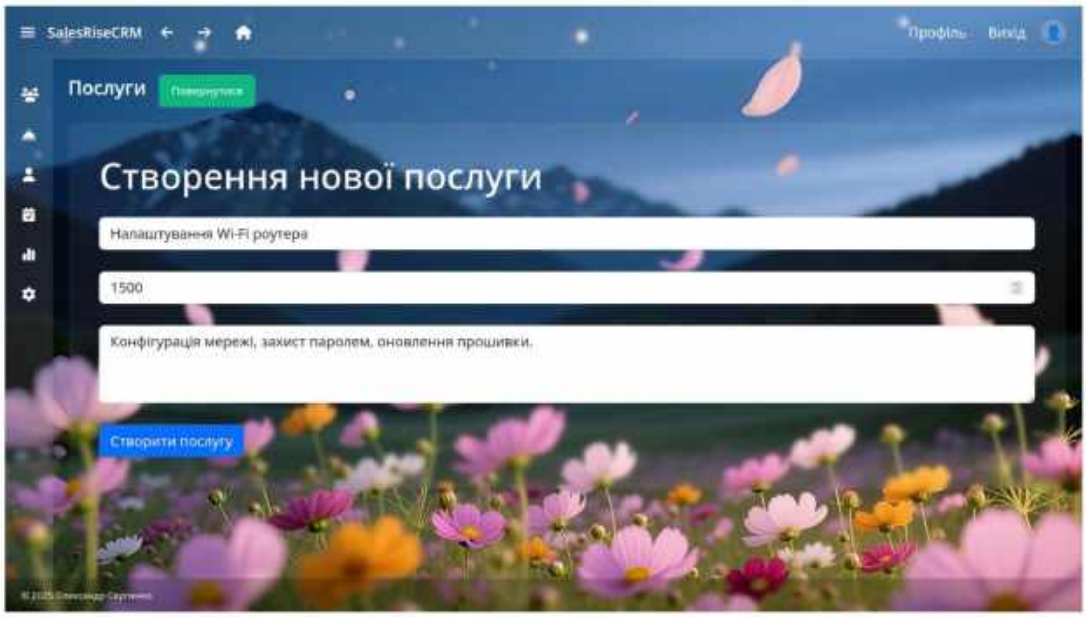

Рисунок 2.27 – Створення нової послуги

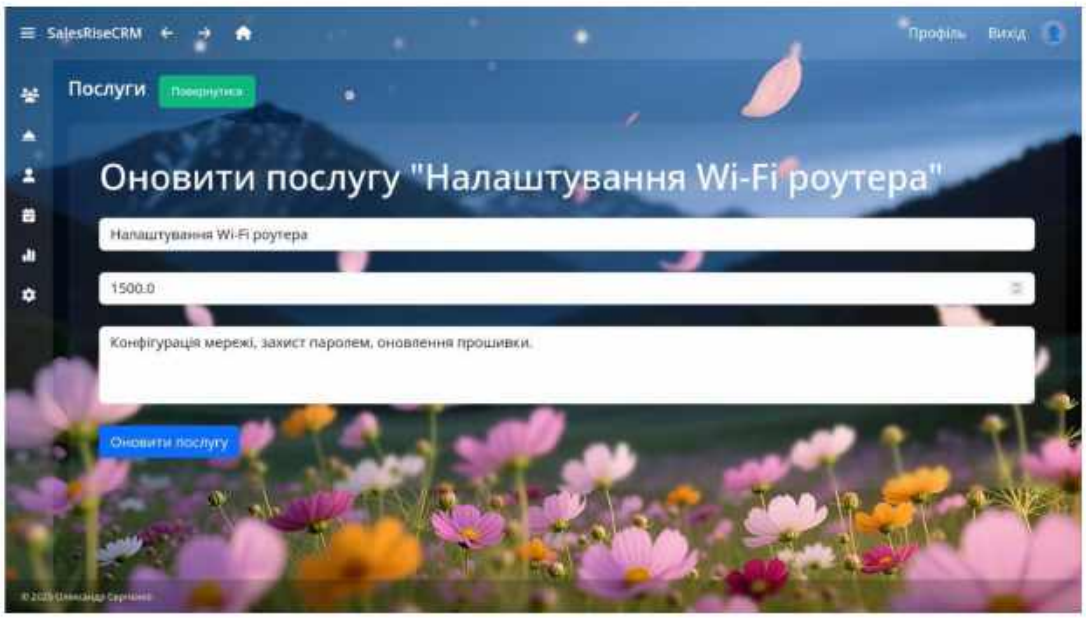

Рисунок 2.28 – Оновлення послуги

Інтерфейс додавання та редагування записів розроблений таким чином, щоб бути максимально зручним та інтуїтивно зрозумілим для користувачів. Усі необхідні поля для введення даних розташовані на видному місці, а кнопки збереження та скасування дозволяють швидко виконати необхідні дії.

Для планування завдань потрібно перейти до розділу «Заплановані задачі». У календарі (рис. 2.29) обрати потрібну дату та час, щоб створити нову подію. З'явиться модальне вікно (рис. 2.30), в якому потрібно вказати клієнта, послугу, відповідального співробітника, статус та час початку і закінчення задачі. Після заповнення всіх полів потрібно натиснути кнопку «Зберегти».

Щоб відредагувати задачу, потрібно обрати її в календарі. Відкриється модальне вікно редагування (рис. 2.31) з можливістю змінити будь-які параметри задачі. Після внесення змін потрібно натиснути кнопку «Зберегти» або, якщо потрібно видалити задачу, кнопку «Видалити».

Рисунок 2.29 – Календар на сторінці «Заплановані задачі»

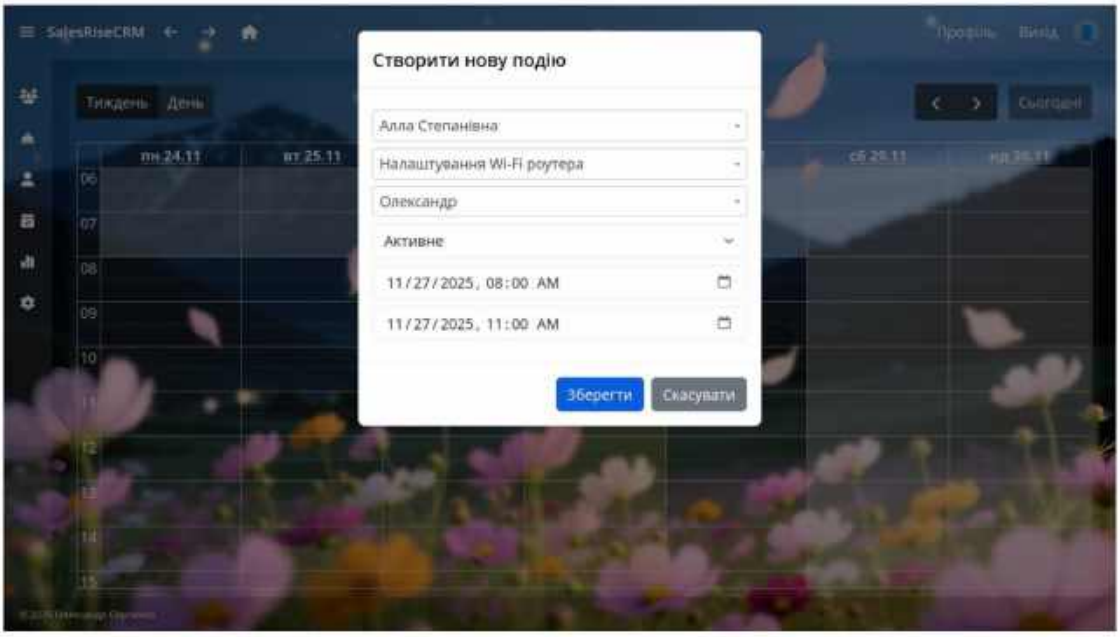

Рисунок 2.30 – Створення події

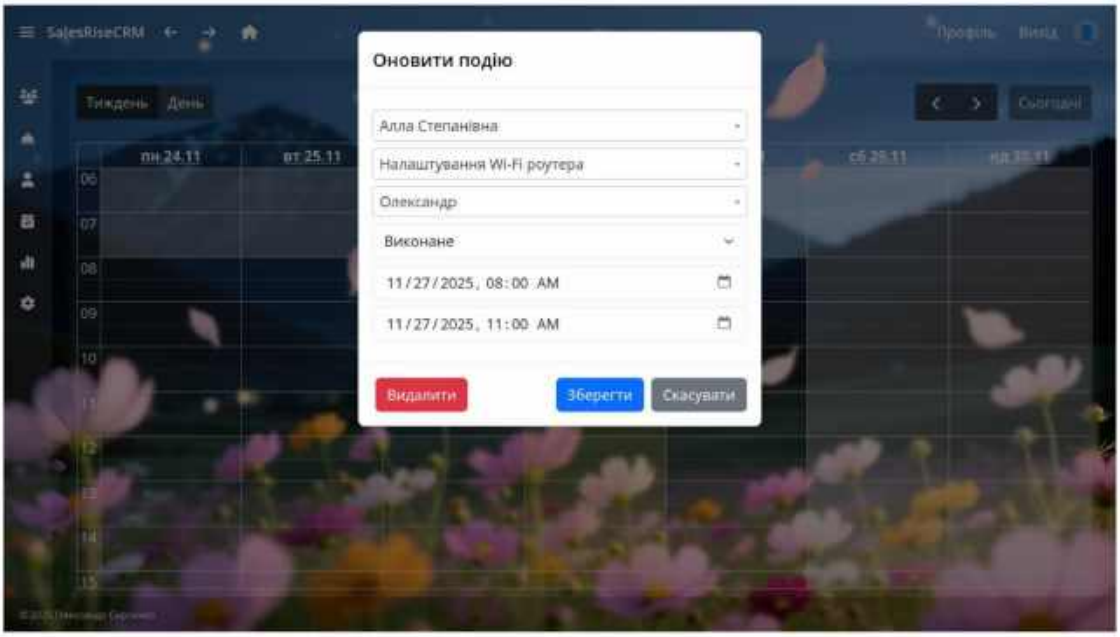

Рисунок 2.31 – Редагування події

Календар запланованих задач візуально відображає всі заплановані події, роблячи процес планування простим та ефективним. Можливість перегляду задач за днем та тижнем, а також редагування чинних подій, забезпечує гнучкість та зручність у використанні.

Для перегляду статистики потрібно перейти до розділу «Статистика». Тут можна знайти два підрозділи: «Загальна статистика» (рис. 2.32) та «Статистика клієнтів» (рис. 2.33).

На сторінці «Загальна статистика» можна побачити загальну картину по наданих послугах за певний період часу. Графіки та діаграми демонструють кількість наданих послуг, їх статус та загальний прибуток. Також, є можливість фільтрувати дані за співробітником та послугою.

На сторінці «Статистика клієнтів» розташована детальна інформація про клієнтів, послуги, співробітників та статусів замовлень. Є можливість фільтрувати дані за співробітником, послугою, клієнтом та статусом.

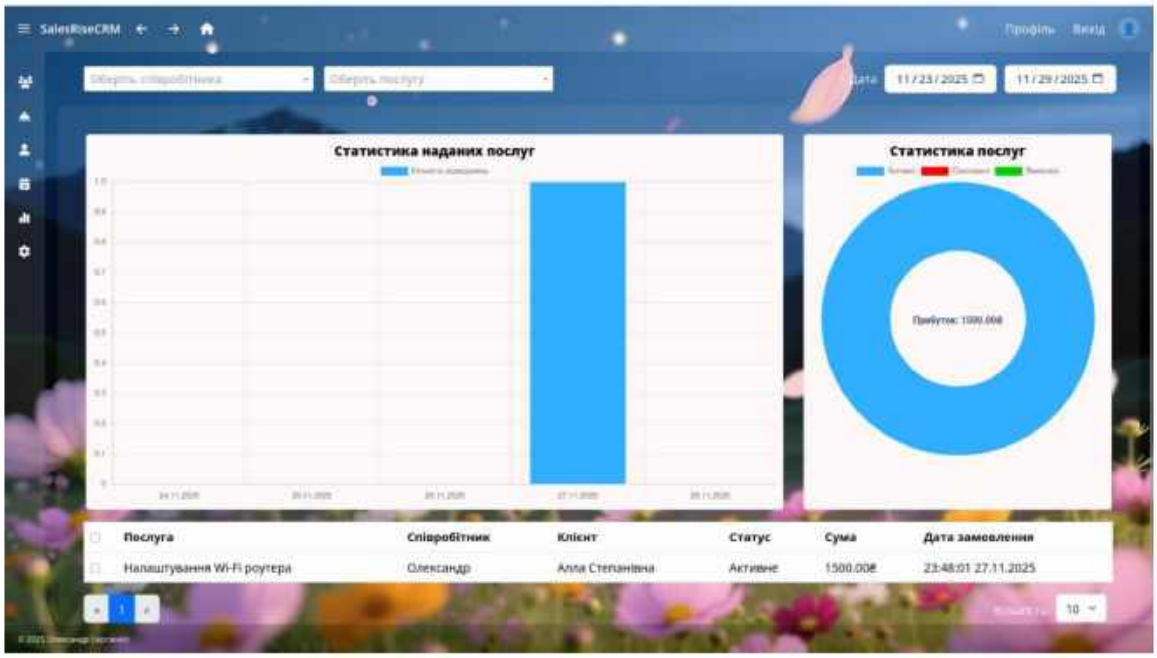

Рисунок 2.32 – Сторінка загальної статистики

Сторінки статистики надають цінну інформацію для аналізу ефективності роботи системи та прийняття обґрунтованих рішень. Зручні фільтри та наочні графіки дозволяють швидко знаходити та аналізувати потрібні дані.

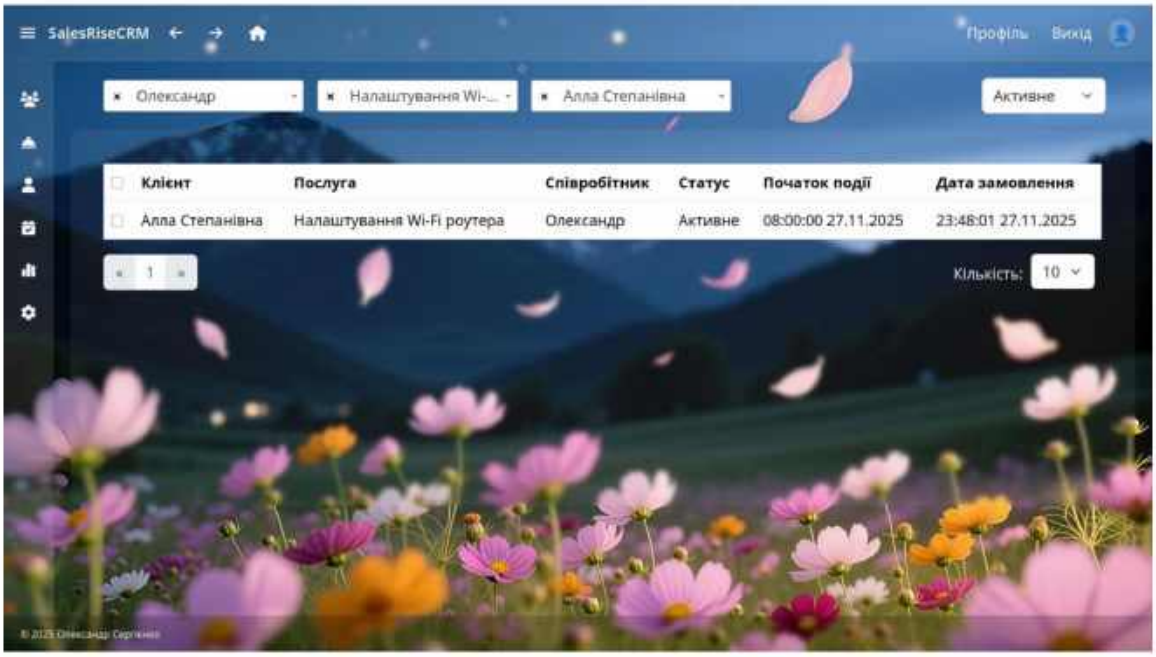

Рисунок 2.33 – Сторінка статистики клієнтів

Для налаштування інтерфейсу потрібно перейти до розділу «Налаштування», який знаходиться в боковому меню системи. Цей розділ дозволяє персоналізувати зовнішній вигляд системи, роблячи роботу з нею більш комфортною та ефективною (рис. 2.34).

У верхній частині сторінки розташовані налаштування кольорів. Користувач може змінити колір тексту, вибравши бажаний колір з палітри, щоб покращити читабельність інформації (рис. 2.35).

Далі ви можете обрати тему оформлення системи. Наразі доступна лише декілька фонових зображень, але в майбутньому можуть бути додані інші.

Якщо потрібно повернутися до початкових налаштувань – це можна зробити через кнопку «Скинути налаштування». Ця дія поверне всі налаштування інтерфейсу до значень за замовчуванням.

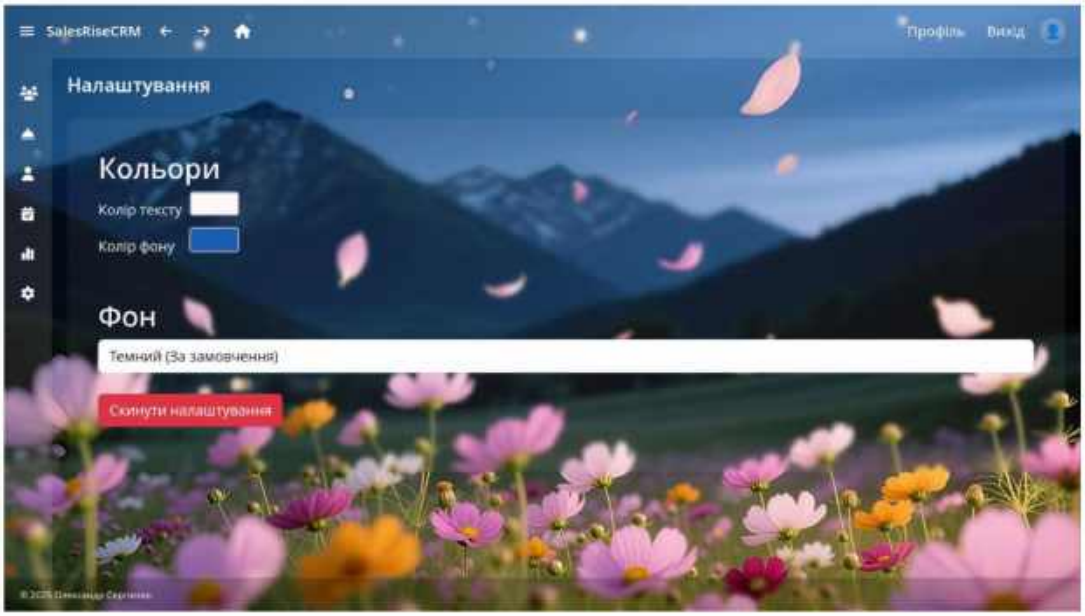

Рисунок 2.34 – Сторінка налаштування інтерфейсу

Рисунок 2.35 – Приклад зміни зображення фону

У лістингу 2.1 наведено клас мовою програмування JavaScript, призначений для динамічної зміни стилів вебсторінки. Цей клас дозволяє встановити фонове зображення, колір фону та тексту для всієї сторінки. Внутрішній приватний метод відповідає за безпосереднє встановлення

властивостей елементу сторінки, що впливає на глобальні стилі. Даний клас використовується для реалізації функцій персоналізації інтерфейсу.

Лістинг 2.1 – Клас «PageStyle» для керування стилями сторінки

```
class PageStyle {
    setBackground(imageUrl) {
        if (imageUrl !== "none") {
            imageUrl = `url(${imageUrl})`;
        }

        this.#setProperty("--background-image-url", imageUrl);
    }

    setBackgroundColor(color) {
        this.#setProperty("--background-color", color);
    }

    setTextColor(color) {
        this.#setProperty("--text-color", color);
    }

    #setProperty(property, value) {
        document.documentElement.style.setProperty(property, value);
    }
}

export default PageStyle;
```

Кінець лістингу 2.1

Персоналізація інтерфейсу системи дозволяє кожному користувачеві створити оптимальне робоче середовище, що сприяє підвищенню продуктивності та комфорту.

Висновки до розділу 2

Обрані мови, фреймворки та інструменти створюють збалансовану та оптимальну основу для розробки платформи.

Python підходить для серверної частини завдяки елегантному синтаксису й легкості масштабування; багаті бібліотеки й фреймворк Flask дозволяють швидко будувати вебзастосунки.

SQLAlchemy – потужний ORM-інструмент, який спрощує роботу з реляційними базами даних у Python, усуваючи потребу у складних SQL-запитах.

Pytest, як надзвичайно гнучкий фреймворк для тестування, гарантує, що код залишатиметься надійним в ході подальшого розвитку.

JavaScript забезпечує динамічність та ефективність інтерфейсу користувача, а FullCalendar надає зручний та візуально привабливий спосіб організації подій.

Усі ці технології разом створюють стабільну, легко підтримувану та масштабовану систему.

Детально описано розробку програмного застосунку, починаючи з архітектури та закінчуючи графічним інтерфейсом користувача. Архітектура застосунку чітко розділена на шари, що забезпечує модульність та гнучкість системи. Побудова бази даних відповідає основним бізнес-потребам, а структура інтерфейсу базується на принципах зручності та інтуїтивності.

Розробка графічного інтерфейсу користувача розпочалася з прототипування в додатку Penpot. Це дозволило швидко створити макети ключових екранів платформи. Усі сторінки виконані в єдиному стилі з використанням фонового зображення, що створює відчуття легкості та спокою.

Описані сторінки (авторизації, реєстрації, головна, користувачів, клієнтів, послуг, задач, статистики та налаштувань) демонструють повний цикл управління системою.

Наведено інструкції для кінцевих користувачів системи. Описано процес реєстрації та авторизації, додавання та редагування записів, планування завдань та відстеження статистики клієнтів та послуг.

Інтерфейс розроблений таким чином, щоб бути максимально зручним та інтуїтивно зрозумілим для користувачів. Усі ключові функції знаходяться у

швидкому доступі, а форми для введення даних та перегляду інформації спроєктовані з урахуванням ергономіки та потреб користувачів.

РОЗДІЛ 3
ЕКСПЕРЕМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РЕЗУЛЬТАТИВНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Методика проведення дослідження

Проведення дослідження результативності розробленого програмного забезпечення вимагало ретельного підходу до вибору та застосування методів тестування. Методика дослідження була побудована на принципах забезпечення правдивості, об'єктивності та повноти отриманих даних.

Першим етапом було формування тестового середовища, яке максимально відповідало реальним умовам експлуатації системи. Це включало налаштування серверної частини, бази даних та клієнтського оточення, ідентичного тому, яким користуватимуться кінцеві користувачі. Створення репрезентативної вибірки даних для тестування було наступним важливим кроком. Дані були підібрані таким чином, щоб відображати типові сценарії використання системи, включаючи як стандартні операції, так і граничні випадки [24].

Далі було застосовано методику модульного тестування з використанням фреймворку pytest. Особливість застосування pytest полягала в його здатності швидко та ефективно перевіряти окремі компоненти системи, ізолюючи їх від інших. Це дозволило точно визначити джерело можливих помилок та оцінити продуктивність кожного модуля зокрема. Тести були розроблені відповідно до варіантів використання системи, що забезпечило покриття всіх ключових функціональних можливостей [25].

Крім модульного тестування, було проведено тестування безпеки. Для цього використовувалися специфічні техніки перевірки захисту від XSS-атак та інших потенційних вразливостей. Метою цього етапу було підтвердження надійності системи в умовах можливих загроз [26].

Аналіз результатів тестування проводився шляхом зіставлення фактичних результатів роботи системи з очікуваними. Для цього використовувалися

детальні звіти, які генерував pytest, а також логування операцій системи. Особлива увага приділялася виявленню випадків, коли система працювала некоректно або нижче очікуваних показників продуктивності [13].

3.2 Обробка та аналіз отриманих результатів

Проведено процес тестування розробленої системи, включаючи методи тестування, використані інструменти та результати тестування. Розглянемо модульне тестування, тестування безпеки та оцінку загальної якості розробленої платформи.

Тестування – це перевірка розробленого продукту на відповідність стандартам якості. Воно включає практичне застосування, тестування програмного забезпечення (як ручне, так і автоматизоване), а також зіставлення фактичних результатів з очікуваними [24].

Цей процес дозволяє виявити проблеми в програмному продукті до його релізу, зокрема помилки та дефекти. Тестування тісно пов'язане з кінцевим продуктом [25].

Отже, тестування програмного додатка є важливим етапом розробки, який забезпечує виявлення та виправлення недоліків до випуску. Основні методи тестування включають:

Модульне тестування (Unit Testing) – це перевірка окремих модулів або компонентів програми.

Інтеграційне тестування (Integration Testing) – це перевірка взаємодії між інтегрованими модулями. Зазвичай проводиться після модульного тестування.

Системне тестування (System Testing) – це комплексна перевірка повного інтегрованого програмного продукту, що виконується тестувальниками в середовищі, максимально наближеному до реального.

Функціональне тестування (Functional Testing) – це перевірка функціональності програми на відповідність специфікаціям. Може

виконуватися як вручну, так і автоматизовано, на основі заздалегідь визначених вимог до системи.

Регресійне тестування (Regression Testing) – перевірка того, що нові зміни не порушили наявний функціонал. Зазвичай проводиться після внесення змін або виправлення помилок й часто автоматизується.

Альфа-тестування (Alpha Testing) – це внутрішнє тестування на ранніх етапах розробки для виявлення критичних помилок на початкових стадіях проєкту.

Бета-тестування (Beta Testing) – це тестування за участі кінцевих користувачів у реальних умовах, що дозволяє отримати реальні відгуки та знайти помилки, не виявлені під час альфа-тестування.

Тестування чорного ящика (Black Box Testing) – це перевірка функціональності програми без знання внутрішньої структури коду. Тестувальники зосереджуються на вхідних даних та очікуваних результатах.

Тестування білого ящика (White Box Testing) – це перевірка внутрішньої структури та логіки коду, що вимагає знання коду та внутрішньої логіки програми.

Тестування навантаженням (Load Testing) – це перевірка працездатності системи під різними навантаженнями. Виявляє реакцію системи на збільшення кількості користувачів або обсягу оброблюваних даних, а також оцінює стабільність додатка.

Тестування безпеки (Security Testing) – це перевірка безпеки програми, включаючи виявлення вразливостей, тестування аутентифікації та авторизації, захист від атак. Проводиться спеціалізованими командами або за допомогою автоматизованих інструментів.

Перелічені методи тестування забезпечують комплексний підхід до перевірки якості програмного забезпечення, гарантуючи його стабільність, продуктивність і безпеку перед випуском [24].

В розробці даної системи було обрано модульне тестування (Unit Testing) як основний метод забезпечення якості коду, оскільки модульне тестування

дозволяє перевіряти кожен окремий компонент системи (функцію, клас, модуль) ізольовано від інших. Це дозволяє швидко виявляти та виправляти помилки на ранніх етапах розробки, не чекаючи інтеграції повністю всіх компонентів [24].

Модульне тестування було реалізовано за допомогою фреймворку pytest.

Pytest – це потужний та гнучкий інструмент для автоматизованого тестування на Python. Він дозволяє писати короткі та зрозумілі тести, а також надає широкий спектр функцій для конфігурації, запуску та аналізу результатів тестування. Pytest підтримує різноманітні плагіни, що розширюють його можливості, наприклад, для тестування вебзастосунків, баз даних або API [13].

На рисунках 3.1 та 3.2 зображено результати запуску тестів, які охоплюють різні аспекти функціональності системи. Тести перевіряють створення, отримання, оновлення та видалення даних клієнтів, подій та послуг. Також перевіряється логіка користувачів, наприклад, чи існує користувач або отримання даних користувача за іменем.

Результати тестування показують, що всі 55 тестів пройшли успішно за 5.18 секунд, що свідчить про високу якість коду та надійність роботи розробленої системи. Рисунок 3.3 демонструє структуру тестів по файлах та варіантам використання системи, що дозволяє легко орієнтуватися в тестовому покритті.

Важливо відзначити, що модульні тести були організовані відповідно до варіантів використання системи (рис. 3.3). Тобто, кожен тест перевіряє конкретну дію, яку користувач може виконати в системі, наприклад, створення клієнта, редагування події або отримання списку послуг. Такий підхід дозволяє забезпечити повноцінне тестування функціональності системи з погляду користувача.

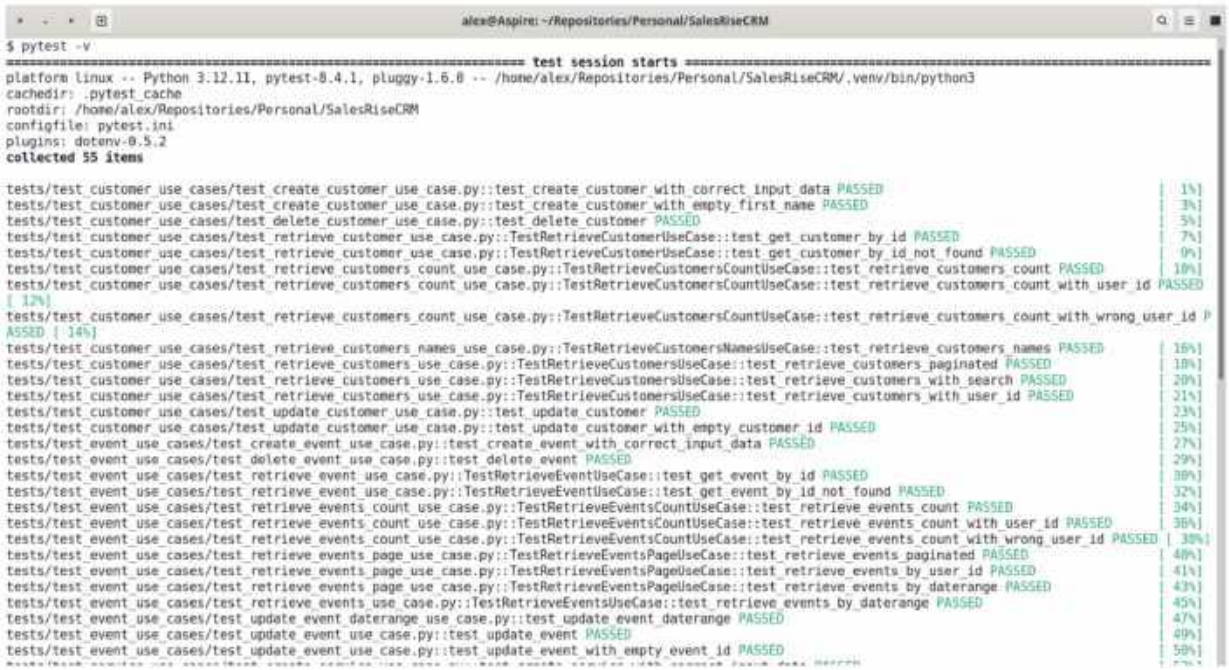

Рисунок 3.1 – Запуск тестування

Рисунок 3.2 – Результат тестування

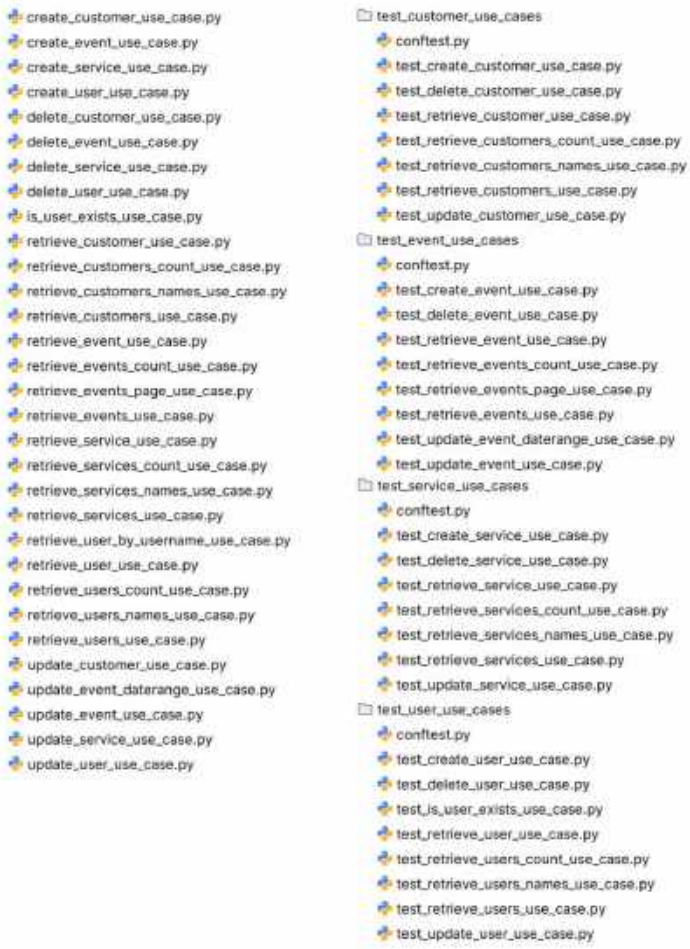

Рисунок 3.3 – Варіанти використання та їх відповідні модульні тести

Використання pytest дозволило забезпечити високе тестове покриття коду системи, що сприяло підвищенню її надійності та стабільності. Результати тестування показали, що всі модулі системи працюють коректно.

З метою забезпечення безпеки розробленої системи було проведено тестування на наявність вразливостей. Особливу увагу було приділено перевірці на XSS (Cross-Site Scripting) атаки, які дозволяють зловмисникам впроваджувати шкідливий код на сторінки сайту та отримувати доступ до даних користувачів [26].

На рисунку 3.4 видно, що при створенні нової послуги в поле опису було введено «XSS-payload» [26]. На рисунку 3.5 показано, як він відображається при редагуванні послуги.

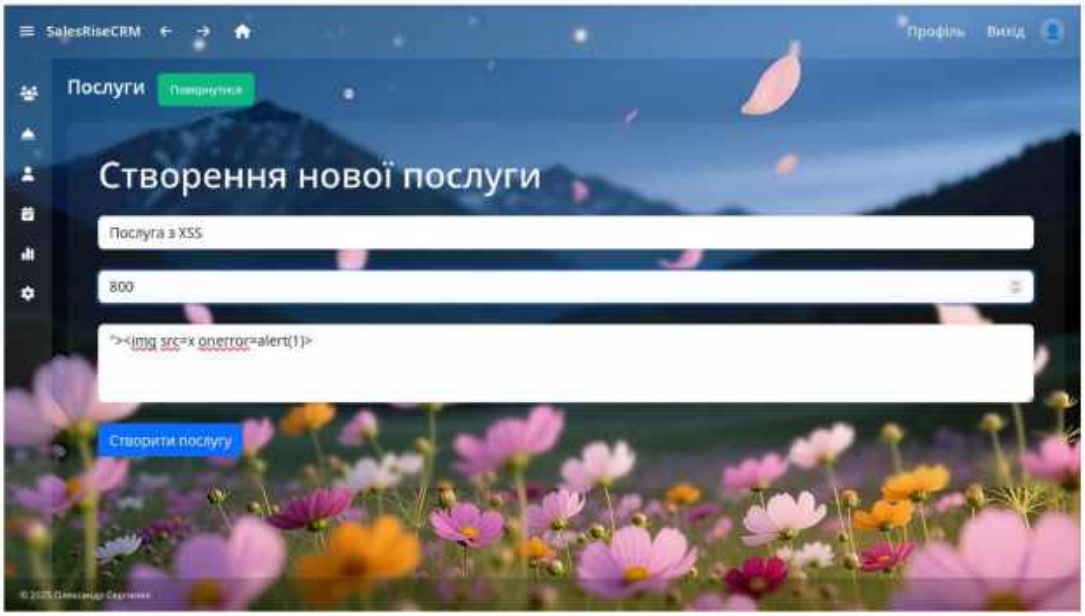

Рисунок 3.4 – Вписаний «XSS-payload» в поле опису

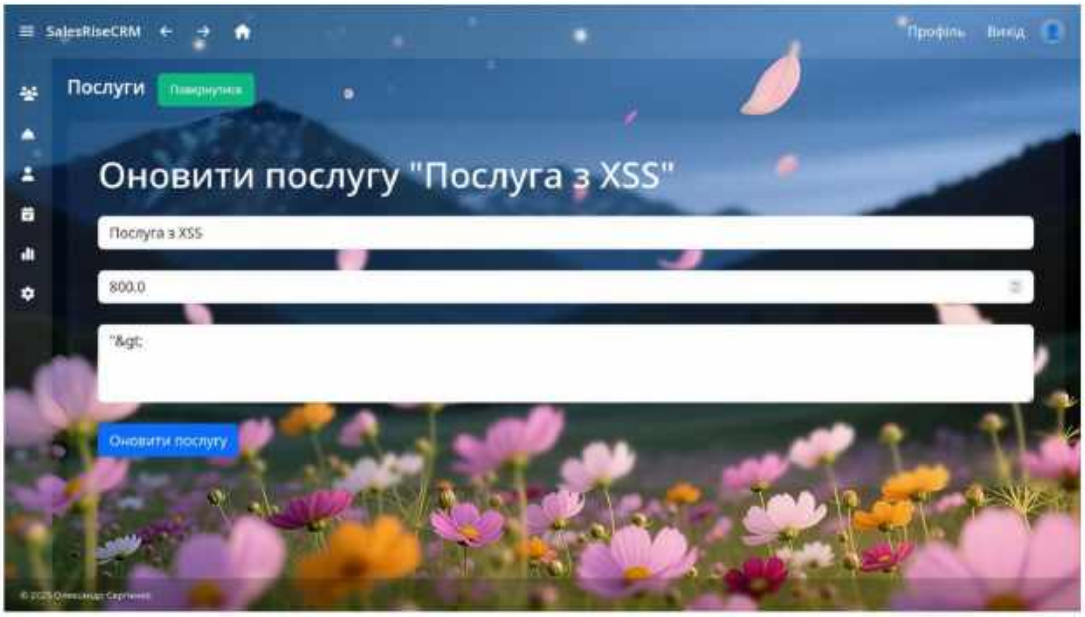

Рисунок 3.5 – Система успішно відфільтровує XSS атаку

Результати тестування показали, що система успішно відфільтровує XSS атаку, не дозволяючи шкідливому коду виконуватися на сторінках сайту, що свідчить про ефективність застосованих заходів захисту та високий рівень безпеки розробленої системи.

Висновки до розділу 3

Було проведено аналіз методів тестування та описано процес тестування розробленої системи. Було обрано модульне тестування (Unit Testing) як основний метод забезпечення якості коду, що дозволяє перевіряти кожен окремий компонент системи ізольовано від інших. Для автоматизації тестування використовувався фреймворк pytest, який дозволяє писати короткі та зрозумілі тести, а також надає широкий спектр функцій для конфігурації, запуску та аналізу результатів тестування.

Проведене тестування показало, що всі модулі системи працюють коректно. Крім того, було проведено тестування безпеки, яке підтвердило ефективність застосованих заходів захисту та високий рівень безпеки розробленої системи.

Загалом, результати тестування свідчать про високу якість розробленої системи та її готовність до використання.

ВИСНОВКИ

У процесі кваліфікаційної роботи магістра здійснено аналіз обраної теми, визначено об'єкт і предмет дослідження, сформульовано мету роботи.

Основним напрямком стало розробка платформи для автоматизації бізнес-процесів з чистою архітектурою, що дозволяє легко розширювати функціонал та забезпечує розділення бізнес-логіки, інфраструктурних сервісів і користувацького інтерфейсу.

На основі проведеного аналізу було сформульовано чіткі вимоги до розроблюваної системи, що враховують особливості бізнес-процесів та потреби користувачів. Вимоги охоплюють функціональні характеристики системи, вимоги до безпеки та зручності використання. Чітке визначення вимог дозволило зосередити зусилля на створенні ефективної та цілеспрямованої системи, що відповідає очікуванням користувачів.

Для розробки системи було проаналізовано та обрано оптимальний набір інструментів та технологій, що включає мови програмування Python та JavaScript, фреймворк Flask, ORM-систему SQLAlchemy, бібліотеку FullCalendar та фреймворк для тестування Pytest. Вибір даних інструментів дозволив створити сучасну, масштабовану та надійну систему з використанням передових вебтехнологій.

Відповідно до визначених вимог було розроблено повноцінну систему з чистим та інтуїтивно зрозумілим інтерфейсом. Система забезпечує управління клієнтською базою, послугами та задачами, а також надає інструменти для відстеження статистики та аналізу даних. Особлива увага приділена зручності використання та персоналізації налаштувань під потреби кожного користувача.

Для забезпечення якості та надійності розробленої системи було здійснено автоматизоване тестування з використанням фреймворку Pytest. Тести охоплюють різні аспекти функціональності системи, включаючи створення, редагування, видалення та перегляд даних, а також перевірку безпеки. Результати тестування підтвердили стабільність роботи системи.

З метою забезпечення безпеки системи було здійснено перевірку захисту від XSS-атак та інших вразливостей. Результати перевірки показали, що система успішно відфільтровує шкідливий код та забезпечує захист даних користувачів. Загалом, розроблена система відповідає заданим вимогам та є готовою до використання в реальних бізнес-умовах.

Виконання кваліфікаційної роботи магістра також покращило мої аналітичні навички та вміння систематично аналізувати й проєктувати програмне забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Science and Information Technologies in the Modern World: Collection of Scientific Papers with Proceedings of the 3rd International Scientific and Practical Conference. International Scientific Unity. October 1-3, 2025. Athens, Greece. P. 64-66. URL: https://isu-conference.com/en/archive/science-and-information-technologies-in-the-modern-world-01-10-25/ (дата звернення: 01.10.2025).

2. What is Salesforce? URL: https://www.salesforce.com/products/what-is-salesforce/ (дата звернення: 01.10.2025).

3. What is Zoho CRM? URL: https://www.zoho.com/crm/what-is-zoho-crm.html (дата звернення: 01.10.2025).

4. HubSpot CRM Review 2025: Pricing, Features, Pros & Cons. URL: https://crm.org/news/hubspot-crm-review (дата звернення: 01.10.2025).

5. What Is CRM (Customer Relationship Management)? URL: https://www.salesforce.com/crm/what-is-crm/ (дата звернення: 01.10.2025).

6. Беррі, П. Head First Python / пер. з англ. Г. Якубовська. Харків : ВД «Фабула», 2021. 624 с.

7. Бріґґс, Дж. Р. PYTHON. Вступ до програмування / пер. з англ. О. Гординчук. Львів : Видавництво Старого Лева, 2023. 400 с.

8. Learning Python, 6th Edition [Book]. URL: https://www.oreilly.com/library/view/learning-python-6th/9781098171292/ (дата звернення: 17.10.2025).

9. Fluent Python, 2nd Edition [Book]. URL: https://www.oreilly.com/library/view/fluent-python-2nd/9781492056348/ (дата звернення: 17.10.2025).

10. Restful Web API Patterns and Practices Cookbook. Connecting and Orchestrating Microservices and Distributed Data. 1st Edition. URL: https://kupichitay.com.ua/product/restful-web-api-patterns-and-practices-cookbook-connecting-and-orchestrating-microservices-and-distributed-data-mike-amundsen/ (дата звернення: 17.10.2025).

11. SQLAlchemy 2 In Practice: Learn to program relational databases in Python step by step. URL: https://dokumen.pub/sqlalchemy-2-in-practice-learn-to-program-relational-databases-in-python-step-by-step.html (дата звернення: 17.10.2025).

12. Bayer, M., Rhodes, B. Sqlalchemy: Database Access Using Python (Developer's Library). Addison-Wesley Professional, 1st ed., 2021. 504 p.

13. Python Testing with pytest [Book]. URL: https://www.oreilly.com/library/view/python-testing-with/9781680509427/ (дата звернення: 21.11.2025).

14. Flanagan, D. JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language. 7th ed. O'Reilly Media, 2020. 704 p.

15. FullCalendar - JavaScript Event Calendar. URL: https://fullcalendar.io/ (дата звернення: 22.11.2025).

16. Clean Architecture with Python [Book]. URL: https://www.oreilly.com/library/view/clean-architecture-with/9781836642893/ (дата звернення: 22.11.2025).

17. Python Architecture Patterns [Book]. URL: https://www.oreilly.com/library/view/python-architecture-patterns/9781801819992/ (дата звернення: 23.11.2025).

18. MySQL Crash Course [Book]. URL: https://www.oreilly.com/library/view/mysql-crash-course/9781098156824/ (дата звернення: 24.11.2025).

19. Hands-On MySQL Administration [Book]. URL: https://www.oreilly.com/library/view/hands-on-mysql-administration/9781098155889/ (дата звернення: 24.11.2025).

20. Penpot: The Design Tool for Design & Code Collaboration. URL: https://penpot.app/ (дата звернення: 25.11.2025).

21. Modern HTML & CSS From The Beginning 2.0 - Second Edition [Book]. URL: https://www.oreilly.com/videos/modern-html/9781835880562/ (дата звернення: 13.11.2025).

22. User Experience Design [Book]. URL: https://www.oreilly.com/library/view/user-experience-design/9781119829201/ (дата звернення: 26.11.2025).

23. Laws of UX, 2nd Edition [Book]. URL: https://www.oreilly.com/library/view/laws-of-ux/9781098146955/ (дата звернення: 26.11.2025).

24. What Is QA in Software Testing? URL: https://www.3pillarglobal.com/insights/what-is-qa-in-software-testing/ (дата звернення: 26.11.2025).

25. Моделювання та аналіз програмного забезпечення: методичні вказівки до практичних та лабораторних занять. / Укладач: Л.В. Глазунова - Одеса:ДУІТЗ, 2021., с. 92.

26. XSS Attacks [Book] URL: https://www.oreilly.com/library/view/xss-attacks/9780080553405/ (дата звернення: 26.11.2025).