

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА**

**Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»**

Кафедра інформаційних систем в економіці

**ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА
«Інформаційні управляючі системи і технології»**

галузь знань	12 Інформаційні технології
Спеціальність	122 Комп'ютерні науки

Форма навчання: денна (очна)

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему: «Розроблення веб-платформи для управління процесами малого бізнесу»

здобувача **Бабійця Данила Ігоровича**

Науковий керівник: к.е.н., доцент Тішков Б. О.

(підпис)

**Робота допущена до захисту перед екзаменаційною
комісією з атестації здобувачів вищої освіти (ЕК)**

Завідувач кафедри: к.е.н., доцент Тішков Б. О.

(підпис)

Київ 2025

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА**

**Навчально-науковий інститут «Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці**

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА

«Інформаційні управляючі системи і технології»

галузь знань	12 Інформаційні технології
Спеціальність	122 Комп'ютерні науки

ПОГОДЖЕНО:

Керівник проєктної групи (гарант)
освітньо-професійної програми

_____ Тішков Б.О.

« ____ » _____ 202_ р.

ЗАТВЕРДЖУЮ:

Завідувач кафедри інформаційних систем
в економіці

_____ Тішков Б.О.

« ____ » _____ 202_ р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувача вищої освіти Бабійця Данила Ігоровича

денної форми навчання

на підготовку кваліфікаційної магістерської роботи

на тему: «Розроблення веб-платформи для управління процесами малого бізнесу»

Тему затверджено наказом ректора Університету від «07» березня 2025 р. №464-ст

**Кваліфікаційна магістерська робота виконується на матеріалах
відкритих публікацій та інтернет-джерел**

План кваліфікаційної магістерської роботи

Розділ I: Аналіз існуючих рішень для автоматизації малого бізнесу та рівень адаптованості до потреб малого підприємництва.

Розділ II: Проєктування та характеристика інформаційної системи веб-платформи для малого бізнесу.

Розділ III: Розроблення проєктних рішень, реалізація та тестування веб-платформи для управління малим бізнесом.

Об'єкт дослідження: Процеси управління та автоматизації діяльності малого бізнесу за допомогою сучасних веб-технологій.

Предмет дослідження: Веб-платформа для автоматизації бізнес-процесів: її функціональні, архітектурні та інтерфейсні рішення, спрямовані на підвищення ефективності управління.

Мета кваліфікаційної магістерської роботи полягає в розробленні розробленні інтегрованої веб-платформи для управління процесами малого бізнесу (ремонт, заправка, облік, продаж), яка забезпечить оптимізацію щоденних операцій, зменшення рутинного навантаження та покращення клієнтського сервісу.

Конкретні завдання, які здобувач повинен виконати для досягнення поставленої мети:

У розділі I провести аналіз сучасного стану та тенденцій розвитку веб-рішень для автоматизації діяльності малого бізнесу, зокрема інформаційних систем обліку замовлень, обробки клієнтських запитів, управління товарами та послугами, CRM-систем і систем обліку ресурсів. Виявити ключові проблеми, з якими стикаються представники малого бізнесу при використанні або впровадженні таких систем, зокрема в контексті обмежених ресурсів, потреби у простому інтерфейсі, адаптивності, підтримки мобільних пристроїв.

У розділі II розробити архітектуру та структурну модель веб-платформи для управління процесами малого бізнесу, яка б відповідала сучасним технічним вимогам, включала модульну побудову, масштабованість, захищений доступ, інтеграцію з іншими сервісами (наприклад, email, повідомлення, аналітика). Дослідити та обґрунтувати вибір технологій реалізації клієнтської та серверної частини, засобів для побудови інтерфейсу, бази даних і логіки взаємодії між компонентами системи.

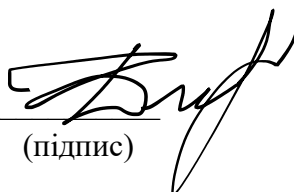
У розділі III розробити проєктні рішення щодо реалізації веб-платформи для управління малим бізнесом, зокрема в частині проєктування структури бази даних, розробки інтерфейсів для користувачів різних ролей, реалізації логіки оброблення замовлень, відображення інформації про клієнтів, статуси виконання робіт та взаємодію з архівними даними.

Завдання підготував
науковий керівник

(підпис)

О.П. Тішков Б. О.

Завдання одержав
здобувач


(підпис)

«10» березня 2025 р.

Д.І. Бабієць

«10» березня 2025 р.

ВІДГУК

на кваліфікаційну магістерську роботу студента
навчально-наукового інституту
"Інститут інформаційних технологій в економіці"
ОПП "Інформаційні управляючі системи і технології"
Бабійця Данила Ігоровича
виконаного на тему:

"Розроблення веб-платформи для управління процесами малого бізнесу"

Кваліфікаційна магістерська робота Бабійця Д.І. присвячена розробці веб-платформи для управління процесами малого бізнесу, яка дозволить автоматизувати ключові процеси підприємства, забезпечить інтеграцію обліку, аналітики та зв'язку з клієнтами в єдиній системі, з урахуванням простоти використання та можливості масштабування.

Для створення інформаційної системи здобувачем розв'язано такі задачі: дослідити теоретичні основи побудови інформаційно-керуючих систем у сфері малого бізнесу; сформулювати функціональні та технічні вимоги до платформи; розробити архітектуру системи та спроектувати її базу даних; обґрунтувати вибір технологій для реалізації веб-платформи; реалізувати основні модулі системи: облік замовлень, клієнтів, матеріалів, звітність, комунікація; провести тестування функціональності платформи відповідно до технічного завдання; сформулювати рекомендації щодо впровадження розробленого рішення в практику малого бізнесу.

В першому розділі кваліфікаційної магістерської роботи аналізуються задачі, що розв'язуються, описується сучасний стан розвитку програмних засобів предметної галузі, а також обґрунтовуються вибір підходу до створення інформаційної системи.

Другий розділ присвячений опису проектних рішень, структури та характеристики системи, виконанню постановки та розробленню алгоритму розв'язання задач, математичному опису, а також опису використовуваних методів та моделей при створенні системи.

В розділі проектування студентом виконано проектування баз моделей і баз даних на конкретному об'єкті дослідження. А також, наведено організаційне, програмне та технічне забезпечення системи. Висвітлені питання проектування бази даних та бази знань системи, розроблення користувацького інтерфейсу, а також забезпечувальної частини системи. Обґрунтований комплекс технічних засобів, а також програмне забезпечення, що використовується для створення інформаційної системи.

При виконанні наукової роботи студент Бабієць Д.І. показав високий рівень знань, уміння узагальнювати практичний досвід використання сучасних комп'ютерних технологій і самостійно розробляти стратегічні проектні рішення. Кваліфікаційна магістерська робота Бабієць Д.І. відповідає нормативним вимогам, виконана якісно і може бути допущена до захисту в екзаменаційній комісії.

Керівник кваліфікаційної
магістерської роботи
к.е.н., доц. кафедри ІСЕ



Б.О. Тішков

Реферат

Кваліфікаційна магістерська робота містить 128 сторінок, таблицю, 15 рисунків, список використаних джерел з 56 найменувань, додатки.

«Розроблення веб-платформи для управління процесами малого бізнесу»

Об'єктом дослідження виступають процеси організації та автоматизації бізнес-процесів у сфері малого бізнесу з використанням сучасних веб-технологій.

Предметом дослідження є веб-платформа для управління процесами малого бізнесу, а також моделі, методи, технології та архітектурні рішення, що забезпечують ефективну цифрову підтримку операційних і адміністративних завдань бізнесу.

Мета роботи полягає у розробленні функціональної веб-платформи, яка забезпечує автоматизацію ключових бізнес-процесів малого підприємства, зокрема облік замовлень, управління клієнтською базою, ведення складу, взаємодію з постачальниками та аналітику.

Відповідно до поставленої мети визначені такі завдання:

- Провести аналіз існуючих систем управління бізнес-процесами, виявити їхні ключові функції, переваги, недоліки та можливості адаптації до потреб малого бізнесу.
- Дослідити теоретичні засади організації бізнес-процесів та сучасні підходи до їх цифрової трансформації.
- Розробити архітектуру та структурну модель інформаційної системи веб-платформи, яка враховує потреби малого бізнесу.
- Обґрунтувати вибір методів, технологій і інструментів, що будуть використані під час розроблення платформи.
- Реалізувати проектні рішення, зокрема базу даних, бекенд, інтерфейсні компоненти та механізми безпеки.
- MVP веб-платформи, протестувати її функціональність на реальних сценаріях використання, оцінити ефективність і масштабованість системи.

Теоретична, методична та практична значущість отриманих результатів. У процесі дослідження було систематизовано та узагальнено сучасні підходи до цифровізації малого бізнесу, що забезпечують підвищення ефективності управлінських процесів. Практична цінність роботи полягає в розробленні універсальної веб-платформи, яка дозволяє підприємцям автоматизувати повсякденні операції, зменшити витрати часу та підвищити продуктивність.

Рік виконання кваліфікаційної магістерської роботи – 2025

Рік захисту роботи – 2025

Ключові слова: веб-платформа, малий бізнес, автоматизація бізнес-процесів, управління замовленнями, клієнтська база, цифрова трансформація, інформаційна система, CRM, Laravel.

Рецензія
на кваліфікаційну магістерську роботу
здобувача вищої освіти
Бабійця Данила Ігоровича

**Тема «Розроблення веб-платформи для управління процесами
малого бізнесу»**

Актуальність теми кваліфікаційної магістерської роботи і доцільність її розроблення: Розробка веб-платформи для управління малим бізнесом є актуальною через потребу в доступних та інтегрованих інструментах для оптимізації ключових процесів, підвищення ефективності та підтримки росту малих підприємств в умовах зростаючої конкуренції.

Якість проведеного дослідження: Якість проведеного дослідження є високою, та обґрунтовано вибір технологічного стеку PHP Laravel та Vue.js як оптимальних для поставлених завдань. Процес розробки веб-платформи описано детально, з поясненням прийнятих архітектурних рішень та особливостей реалізації основних модулів.

Позитивні риси кваліфікаційної магістерської роботи: Однією з позитивних рис є ефективне застосування сучасного фреймворку Laravel для розробки серверної частини, що дозволило створити добре структурований та легко підтримуваний код.
А також платформа забезпечує інтегроване управління основними бізнес-процесами та має зручний інтерфейс.

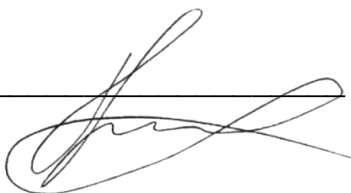
Зауваження: Потребує розширення функціоналу в частині розробки мобільного застосунку для зручнішого доступу з мобільних пристроїв.

Практична значимість висновків і рекомендацій: розроблена в рамках кваліфікаційної роботи веб-платформа має значну практичну цінність для малих підприємств, оскільки надає комплексний набір інструментів для автоматизації ключових бізнес-процесів.

Місце роботи та посада рецензента

Науковий ступінь, учене звання (за наявності) Керівник технічної команди (Техлід) ТОВ «BinarGear»

Підпис засвідчую: Гурський Б.В.



ЗМІСТ

ВСТУП	4
РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПІДХОДІВ ДО АВТОМАТИЗАЦІЇ БІЗНЕС-ПРОЦЕСІВ	7
1.1 Аналіз існуючих веб-платформ для автоматизації малого бізнесу	7
1.1.1 Дослідження типових функцій і можливостей сучасних інформаційних систем	7
1.1.2 Аналіз інформаційно-керуючих систем в галузі сервісного обслуговування та продажу	10
1.2 Обґрунтування вибору технологій і підходів до розроблення системи управління малим бізнесом	12
РОЗДІЛ 2. ПОСТАНОВКА ЗАВДАННЯ ТА ПРОЄКТУВАННЯ ВЕБ-ПЛАТФОРМИ	15
2.1 Структура і характеристика інформаційної системи у предметній області	15
2.2 Методи в інформаційних управляючих системах і технологіях	16
РОЗДІЛ 3. РОЗРОБЛЕННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-ПЛАТФОРМИ ДЛЯ УПРАВЛІННЯ ПРОЦЕСАМИ МАЛОГО БІЗНЕСУ	19
3.1 Проєктування структури бази даних	19
3.1.1 Інфологічна (логічна) модель бази даних	19
3.1.2 Обґрунтування вибору СКБД та проєктування фізичної моделі	21
3.1.3 Контрольний приклад та підрахунок прогнозованих обсягів	22
3.2 Реалізація логіки обліку та керування бізнес-процесами	24
3.3 Розробка програмного забезпечення платформи	31
3.3.1 Проєктування програмного забезпечення	31
3.3.2 Вибір архітектури програмного забезпечення	32
3.3.3 Опис реалізації основних модулів	34
3.3.4 Тестування програмного забезпечення	36
3.3.5 Керівництво користувача	39
3.4 Технічне забезпечення	41
3.4.1 Обґрунтування вибору середовища розгортання	41
3.4.2 Ресурсні вимоги до розгортання та підтримки	43
3.5 Розгортання платформи та перевірка її працездатно	45
ВИСНОВКИ	55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	57
ДОДАТКИ	60
Додаток А	60
Код створення бази даних Leads через міграції	60
Додаток Б	62
Код контролера для сторінки Leads на серверній частині	62
Додаток В	83

	3
Код форми для сторінки Leads на серверній частині	83
Додаток Г	91
Код сторінки Leads на клієнтській частині	91
Додаток Д	115
Код ендпоінтів для сторінки Leads на серверній частині	115
Додаток Е	117
Код для CI/CD GitLab - AWS пайплайну	117
Додаток Є	121
Код Docker Compose	121
КОПІЯ ТЕЗ ДОПОВІДІ	123
IV Всеукраїнська студентська наукова конференція «ФОРМУВАННЯ СУЧАСНОЇ НАУКИ: МЕТОДИКА ТА ПРАКТИКА»	123

ВСТУП

Актуальність теми магістерської роботи зумовлена потребою малого бізнесу в ефективній цифровій трансформації внутрішніх процесів. В умовах зростання конкуренції, швидкої зміни ринку та зростання витрат, підприємства малого бізнесу все частіше звертаються до автоматизованих рішень для обліку клієнтів, замовлень, матеріалів, контролю робочих процесів, аналітики та комунікації. Проте більшість наявних інструментів є або занадто складними для впровадження в малих компаніях, або не враховують специфіку певних ніш — зокрема, бізнесу, пов'язаного з ремонтом техніки, заправкою картриджів та продажем витратних матеріалів. Відсутність універсальних та водночас адаптивних платформ гальмує ефективне управління ресурсами та розвиток підприємств.

Аналіз останніх досліджень і публікацій. Останні дослідження в галузі цифровізації бізнесу та автоматизації управлінських процесів вказують на високу ефективність галузевих інформаційних систем. Праці таких організацій, як OECD, McKinsey, Gartner, а також наукові публікації у сфері бізнес-інформатики акцентують увагу на необхідності адаптивних платформ для МСП (малих та середніх підприємств). Утім, більшість досліджень спрямовані на великі корпоративні рішення, а специфіка малого бізнесу залишається недостатньо дослідженою, зокрема в контексті архітектури систем, що мають охоплювати повний цикл управління – від замовлення до звітності. Це створює потребу в розробленні прикладної веб-платформи, адаптованої до потреб малого бізнесу з урахуванням зручності, простоти та економічної ефективності.

Наукова проблема полягає у відсутності комплексного, але водночас простого у впровадженні рішення для цифрового управління бізнес-процесами малого підприємства в сфері надання послуг.

Суть проблеми полягає у фрагментованості існуючих рішень, які або надто складні в обслуговуванні, або не охоплюють усі потреби підприємства, через що власники змушені користуватись кількома несумісними інструментами, що ускладнює облік та прийняття управлінських рішень.

Об'єктом дослідження у магістерській кваліфікаційній роботі є цифрового управління бізнес-процесами у малому підприємстві.

Предметом дослідження є структура, архітектура, моделі та засоби реалізації веб-платформи.

Метою дослідження є розробка веб-платформи для управління процесами малого бізнесу, яка дозволить автоматизувати ключові процеси підприємства, забезпечить інтеграцію обліку, аналітики та зв'язку з клієнтами в єдиній системі, з урахуванням простоти використання та можливості масштабування.

Завдання були сформовані такі як:

- проаналізувати існуючі веб-платформи для управління бізнесом, виявити їх переваги та обмеження;
- дослідити теоретичні основи побудови інформаційно-керуючих систем у сфері малого бізнесу;
- сформулювати функціональні та технічні вимоги до платформи;
- розробити архітектуру системи та спроектувати її базу даних;
- обґрунтувати вибір технологій для реалізації веб-платформи;
- реалізувати основні модулі системи: облік замовлень, клієнтів, матеріалів, звітність, комунікація;
- провести тестування функціональності платформи відповідно до технічного завдання;
- сформулювати рекомендації щодо впровадження розробленого рішення в практику малого бізнесу.

Методи дослідження, використані у роботі:

- аналіз наукових публікацій, технічної документації та існуючих рішень;

- методи системного аналізу та функціонального моделювання бізнес-процесів;
- архітектурне проектування інформаційних систем;
- засоби об'єктно-орієнтованого програмування та веб-розробки;
- функціональне тестування;

Результати дослідження були отримані під час проходження виробничої практики на базі ФОП «Гурський Богдан Вадимович», де була вивчена структура бізнес-процесів, зібрано вимоги до системи, спроектовано і реалізовано веб-платформу, а також протестовано її функціональність у реальних умовах.

Практична цінність розробленого програмного продукту полягає у можливості його впровадження в діяльність малого підприємства для підвищення ефективності, зниження операційних витрат та покращення клієнтського сервісу. Платформа легко масштабована і може бути адаптована до потреб інших схожих бізнесів.

Інформаційна база дослідження включає наукові публікації у сфері ІТ та бізнес-інформатики, технічну документацію до веб-фреймворків та СКБД, аналітичні звіти консалтингових компаній, а також дані, зібрані безпосередньо під час практики.

Структура роботи відповідає меті та завданням дослідження. Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. У першому розділі проаналізовано сучасні інформаційні системи та підходи до автоматизації малого бізнесу. У другому – визначено функціональні вимоги, спроектовано архітектуру платформи та базу даних. У третьому – реалізовано веб-платформу, проведено тестування, а також надано рекомендації щодо впровадження та подальшого розвитку.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПІДХОДІВ ДО АВТОМАТИЗАЦІЇ БІЗНЕС-ПРОЦЕСІВ

1.1 Аналіз існуючих веб-платформ для автоматизації малого бізнесу

1.1.1 Дослідження типових функцій і можливостей сучасних інформаційних систем

В сучасних умовах розвитку інформаційних технологій автоматизація бізнес-процесів є важливою складовою підвищення ефективності управління малим бізнесом. Інформаційні системи (ІС) для автоматизації малого бізнесу представляють собою програмні рішення, які сприяють оптимізації діяльності підприємств за рахунок автоматичного виконання типових операцій, зменшення людського фактора і підвищення швидкості обробки даних.

Класифікація веб-платформ:

CRM-системи (Customer Relationship Management) - основним завданням таких систем є управління взаємовідносинами з клієнтами: збір і збереження контактів, ведення історії комунікацій, планування взаємодій, аналітика поведінки клієнтів. Популярні приклади: Bitrix24, AmoCRM, HubSpot. Ці платформи зазвичай включають можливість автоматизації маркетингу, створення комерційних пропозицій, нагадувань.

ERP-системи (Enterprise Resource Planning) - це комплексні рішення, які охоплюють управління ресурсами підприємства — склад, закупівлі, фінанси, виробництво та продажі. Для малого бізнесу доступні полегшені версії ERP-систем, наприклад, Odoo, 1С:Підприємство. Вони дозволяють інтегрувати ключові процеси в єдину систему.

Системи управління продажами та POS-системи призначені для автоматизації операцій безпосередньо в точках продажу, ведення обліку товарів і касових операцій. Приклади: Vend, Square. Зазвичай мають інтуїтивний інтерфейс та інтеграцію з платіжними терміналами.

Системи управління проектами і завданнями використовуються для організації роботи співробітників, планування і контролю виконання завдань. Приклади — Trello, Asana, Jira. Такі системи можна інтегрувати з CRM і ERP для кращої координації бізнес-процесів.

Спеціалізовані системи для сервісного обслуговування та ремонту

Платформи, орієнтовані на бізнеси з надання послуг (ремонт техніки, сервісне обслуговування), наприклад, RepairShopr, ServiceM8. Вони пропонують функції обліку замовлень, планування робіт, управління запасами запчастин.

Основні характеристики та переваги веб-платформ:

Доступність і мобільність: оскільки веб-платформи працюють через браузер, вони забезпечують доступ до системи з будь-якого пристрою та будь-якої точки з інтернетом.

Масштабованість: більшість сучасних рішень підтримують масштабування, що дозволяє розширювати функціонал або кількість користувачів у міру росту бізнесу.

Інтеграції: платформи пропонують інтеграції з популярними бухгалтерськими програмами, платіжними системами, маркетинговими інструментами, що значно підвищує їх ефективність.

Автоматизація рутинних процесів: Автоматичне створення звітів, нагадувань, формування документів і контроль за виконанням завдань зменшує навантаження на персонал.

Недоліки та виклики:

- Висока вартість деяких комерційних рішень, що може бути обмежуючим фактором для малих підприємств;
- Необхідність адаптації системи під специфіку конкретного бізнесу;
- Ризики, пов'язані з безпекою даних, особливо у випадку хмарних платформ;
- Складність освоєння платформи для співробітників без досвіду роботи з ІТ-системами;
- Для розроблення ефективної веб-платформи необхідно провести аналіз типових функцій і можливостей, які реалізують сучасні інформаційні системи. На основі дослідження різних платформ (наприклад, CRM-систем,

ERP-рішень, систем управління продажами і сервісом) можна виділити такі ключові функціональні напрямки:

- Управління клієнтською базою;
- Збір, збереження та обробка даних про клієнтів (контактна інформація, історія замовлень, комунікації). Ця функція дозволяє вести ефективну взаємодію з клієнтами, персоналізувати пропозиції та підвищувати рівень задоволеності;
- Управління товарами і послугами;
- Облік асортименту товарів, послуг, їх характеристик, цін і наявності на складі. Забезпечує актуальність інформації про ресурси підприємства, що є основою для точного планування та продажів;
- Обробка замовлень і продажів: реєстрація замовлень клієнтів, контроль статусів їх виконання, автоматичне формування документів, інтеграція з платіжними системами. Це допомагає упорядкувати процес продажів і підвищити швидкість обслуговування;
- Управління складом і запасами: відстеження руху товарів на складі, формування звітів про залишки, автоматичне поповнення запасів. Автоматизація цих процесів зменшує ймовірність помилок і недостач;
- Фінансовий облік і звітність: ведення обліку доходів і витрат, формування бухгалтерських звітів, контроль оплат. Забезпечує прозорість фінансових потоків і допомагає приймати обґрунтовані управлінські рішення;
- Планування та управління ресурсами: організація робочих графіків співробітників, планування ремонту або обслуговування обладнання, управління замовленнями на виконання послуг. Дозволяє оптимізувати використання ресурсів;
- Аналітика та бізнес-інтелект: збір і обробка статистичних даних, формування діаграм, прогнозів і рекомендацій для підвищення ефективності роботи бізнесу. Інструменти аналітики дають можливість контролювати ключові показники діяльності;

- Безпека і доступ: розмежування прав доступу користувачів до різних модулів і даних системи, захист інформації від несанкціонованого доступу; Важливий аспект для забезпечення цілісності і конфіденційності даних.

Інтеграція з зовнішніми сервісами: підключення до бухгалтерських програм, платіжних шлюзів, систем електронної пошти, SMS-оповіщень та інших інструментів, що підвищують функціональність системи.

Сучасні інформаційні системи часто мають хмарну архітектуру, що забезпечує доступність даних з будь-якого пристрою, гнучкість масштабування та зниження вартості підтримки інфраструктури. Водночас існують системи локального розгортання, які можуть бути більш безпечними для конфіденційних даних. Отже, в процесі розроблення веб-платформи для управління процесами малого бізнесу необхідно врахувати вищезазначені функціональні напрямки та адаптувати їх до специфіки сфери обслуговування, продажу і ремонту друкарської техніки, що є предметною областю даної роботи.

1.1.2 Аналіз інформаційно-керуючих систем в галузі сервісного обслуговування та продажу

Галузь сервісного обслуговування та продажу характеризується великою кількістю бізнес-процесів, що потребують чіткої організації та ефективного контролю. В таких умовах інформаційно-керуючі системи (ІКС) відіграють ключову роль у забезпеченні якості послуг, управлінні ресурсами та оптимізації операційної діяльності.

Основні завдання ІКС у сервісній сфері, в загальному у цій галузі призначені для автоматизації таких процесів, як прийом і обробка замовлень, планування виконання робіт, управління запасами, облік клієнтів та контроль якості обслуговування. Вони повинні забезпечувати:

- Швидкий доступ до інформації про клієнтів, їх замовлення, історію обслуговування та взаємодії;

- Планування і координацію робіт, зокрема розподіл завдань між співробітниками, визначення пріоритетів, контроль термінів виконання;
- Облік матеріалів, необхідних для ремонту або виконання послуг;
- Формування звітності для аналізу ефективності роботи, оцінки завантаженості та якості обслуговування;
- Інтеграцію з платіжними системами для автоматизації процесу оплати послуг;

Аналіз популярних систем, розглянемо кілька поширених ІКС, що застосовуються у сфері сервісного обслуговування та продажу:

RepairShopr - система, орієнтована на ремонт техніки. Включає функції CRM, управління замовленнями, облік запасів та фінансовий облік. Має інтуїтивний інтерфейс і дозволяє автоматизувати взаємодію з клієнтами через електронну пошту і SMS;

ServiceM8 - хмарна платформа для малого сервісного бізнесу, що дозволяє управляти заявками, графіками співробітників і інвентарем. Має мобільний додаток для зручної роботи на виїзді;

Zoho Desk - система підтримки клієнтів із CRM-функціоналом, призначена для поліпшення якості сервісу та оптимізації робочих процесів служби підтримки;

1C:Сервіс - локальна система з широким функціоналом для автоматизації сервісного обслуговування, включно з управлінням замовленнями, договорами, запасами і фінансами;

Основні переваги ІКС для сервісного бізнесу це, підвищення оперативності реагування на запити клієнтів і виконання замовлень; зменшення кількості помилок через автоматизацію обробки інформації; покращення контролю за ресурсами, що дозволяє знизити витрати; можливість аналітики та прогнозування на основі зібраних даних.

Визначені проблеми та виклики

- Необхідність налаштування і адаптації системи під специфіку конкретного бізнесу;
- Відсутність у деяких системах гнучкості у разі нестандартних операцій;

- Потреба у навчанні персоналу роботі з новим програмним забезпеченням;

- Забезпечення безпеки конфіденційної інформації;

В загальному, інформаційно-керуючі системи у галузі сервісного обслуговування та продажу надають широкий спектр інструментів для оптимізації бізнес-процесів, що робить їх незамінними для сучасних малих підприємств. Вибір конкретної системи повинен базуватися на глибокому аналізі потреб бізнесу, масштабах діяльності та можливостях інтеграції з іншими рішеннями.

1.2 Обґрунтування вибору технологій і підходів до розроблення системи управління малим бізнесом

Для розробки веб-платформи управління процесами малого бізнесу обрано сучасний технологічний стек та методологію, які відповідають вимогам продуктивності, безпеки, гнучкості та зручності підтримки.

Методологія розробки: використовується Agile — гнучкий підхід, що дозволяє швидко адаптуватися до змін у вимогах, оперативно тестувати окремі модулі та розвивати систему поетапно. Agile сприяє тісній співпраці розробників і замовника, що особливо важливо для малого бізнесу, де бізнес-процеси часто змінюються.

- Серверна частина (Backend);
- PHP — популярна мова серверної розробки, що забезпечує високу сумісність із різноманітними хостингами;
- Laravel Framework — сучасний PHP-фреймворк, що забезпечує швидку розробку, багатий набір інструментів (маршрутизація, ORM Eloquent, механізми безпеки), а також інтегровані засоби тестування;
- База даних: MySQL — надійна реляційна СКБД, оптимальна для зберігання структурованих даних малого та середнього бізнесу;

- Тестування: PHPUnit, Pest, Paratest — забезпечують покриття коду тестами та стабільність роботи платформи;

Важливі пакети:

- Laravel Sanctum — для безпечної автентифікації та управління сесіями;
- Laravel Debugbar — допомагає відслідковувати помилки та оптимізувати продуктивність;
- Laravel DomPDF — для генерації PDF-звітів;
- Monolog — для гнучкого логування подій;
- Guzzle — HTTP-клієнт для інтеграції з зовнішніми API;
- Faker — генерація тестових даних для розробки і тестування;

Клієнтська частина (Frontend):

- Vue.js — реактивний фреймворк для створення інтерактивних інтерфейсів, що забезпечує високу швидкість і гнучкість;
- Vite — інструмент збірки фронтенду, що дозволяє швидко компілювати і оновлювати код;
- TailwindCSS — утилітарний CSS-фреймворк, що пришвидшує розробку адаптивних і сучасних UI;

Додаткові Vue-компоненти:

- VeeValidate — для валідації форм користувача;
- Vue-flatpickr — календар і вибір дати/часу;
- Vue-cal — компонент календаря для планування;
- Vuedraggable — для drag-and-drop функціоналу;
- Chart.js — бібліотека для побудови інтерактивних діаграм та графіків, важлива для аналітики;

Інструменти розробки:

- NPM — менеджер пакетів для JavaScript, забезпечує легке керування бібліотеками фронтенду;
- Composer — менеджер пакетів PHP, що спрощує інтеграцію зовнішніх бібліотек і підтримку залежностей;

Шаблонізація:

- Blade — вбудований шаблонізатор Laravel, що дозволяє ефективно формувати HTML-сторінки;
- Інтеграція Vue.js у Blade дозволяє комбінувати серверний та клієнтський рендеринг для кращої продуктивності;
- Використання подій типу `view_render_event` для розширення функціональності та кастомізації відображення;
- Підтримка локалізації через директиву `@lang` для багатомовності платформи;

Обрані технології та підходи формують сучасну, масштабовану і безпечну платформу, адаптовану для потреб малого бізнесу. Вони забезпечують швидку розробку, легку підтримку, зручність використання і можливість подальшого розвитку функціоналу у відповідь на змінні бізнес-вимоги.

РОЗДІЛ 2. ПОСТАНОВКА ЗАВДАННЯ ТА ПРОЄКТУВАННЯ ВЕБ-ПЛАТФОРМИ

2.1 Структура і характеристика інформаційної системи у предметній області

Інформаційна система, розроблена для автоматизації процесів малого бізнесу у сфері сервісного обслуговування друкарської техніки, ремонту та продажу, є комплексним інструментом, що покликана забезпечити ефективне управління основними бізнес-процесами підприємства. Вона має на меті спростити і прискорити роботу з клієнтами, оптимізувати облік матеріалів та запчастин, а також покращити фінансовий контроль та звітність. За допомогою системи користувачі отримують можливість оперативно обробляти замовлення, відслідковувати статуси виконання ремонтних робіт, вести повний облік витратних матеріалів та формувати звіти, необхідні для аналізу діяльності підприємства.

Основу системи складає підсистема управління клієнтами, що дозволяє зберігати детальну інформацію про контактні дані, історію звернень, замовлень та послуг, що були надані кожному клієнту. Це сприяє підвищенню якості обслуговування, оскільки працівники можуть швидко отримувати всю необхідну інформацію для консультації або обробки повторних звернень. Важливим елементом є підсистема обробки замовлень, яка автоматизує прийом і реєстрацію заявок, контроль термінів та призначення завдань майстрам. Такий підхід дозволяє ефективно планувати робочий час та мінімізувати ризики пропуску або затримки виконання замовлень.

Для забезпечення безперервності бізнесу і контролю за наявністю необхідних матеріалів використовується підсистема обліку запасів, що відслідковує рух витратних матеріалів і запчастин на складі, оновлює залишки в режимі реального часу та формує заявки на поповнення запасів у постачальників.

Завдяки цьому підприємство завжди має під рукою необхідні ресурси для своєчасного виконання ремонтних робіт. Важливою складовою системи є також фінансовий модуль, який автоматизує облік доходів і витрат, формує рахунки клієнтам і дозволяє інтегруватися з платіжними системами для спрощення прийому оплат. Ця підсистема забезпечує прозорість фінансових потоків і допомагає керівництву контролювати фінансовий стан бізнесу.

Для підтримки прийняття управлінських рішень інформаційна система оснащена інструментами аналітики та звітності. Вона генерує детальні звіти про обсяги виконаних робіт, доходи, витрати, популярність послуг, а також ефективність роботи співробітників. Завдяки цьому керівництво отримує можливість планувати подальший розвиток бізнесу, коригувати стратегії продажів і покращувати якість обслуговування.

Архітектура системи побудована за принципом клієнт-серверної взаємодії, що забезпечує гнучкий доступ до функціоналу через веб-браузер з будь-яких пристроїв, підключених до інтернету. Такий підхід дозволяє співробітникам мати актуальну інформацію в режимі реального часу, незалежно від місцезнаходження, а також полегшує процес оновлення і підтримки програмного забезпечення. При розробці системи враховані специфічні особливості предметної області: необхідність точного контролю термінів виконання замовлень, особливості обліку запасів, пов'язаних із специфікою друкарської техніки, а також потреба у зручних засобах комунікації з клієнтами та між співробітниками.

Тому, розроблена інформаційна система є комплексним рішенням, що об'єднує в собі інструменти для управління клієнтською базою, замовленнями, складським обліком, фінансами та аналітикою, що значно підвищує ефективність і конкурентоспроможність малого бізнесу у сфері обслуговування друкарської техніки.

2.2 Методи в інформаційних управляючих системах і технологіях

При розробці веб-платформи для управління процесами малого бізнесу застосовується комплексний підхід, який поєднує сучасні методи моделювання

бізнес-процесів, організацію зберігання даних, автоматизацію управлінських операцій і створення інтуїтивно зрозумілого інтерфейсу користувача.

Для моделювання бізнес-процесів використовуються діаграми, зокрема BPMN (Business Process Model and Notation), які дозволяють детально описати послідовність операцій: від прийому заявки клієнтом до остаточного завершення замовлення. Цей підхід дає змогу виявити та оптимізувати «вузькі місця» процесу, усунути дублювання функцій та забезпечити автоматичний контроль виконання завдань. Наприклад, автоматизація переходу замовлення зі статусу «Очікує підтвердження» до «Виконується» та подальше оновлення статусу при отриманні оплати значно скорочують час обробки.

Зберігання інформації організовано на базі реляційної СУБД MySQL, оскільки вона забезпечує високу надійність, стабільність і гнучкість в роботі з великими обсягами структурованих даних. Розроблена логічна модель бази даних містить таблиці для клієнтів, замовлень, товарів, складських запасів, оплат та звітів. Кожна таблиця має унікальний первинний ключ, а також зв'язки між таблицями реалізовані за допомогою зовнішніх ключів, що забезпечує цілісність даних. Індеси на полях, які часто використовуються в умовах пошуку, суттєво покращують продуктивність системи, особливо при зростанні кількості користувачів і замовлень.

Автоматизація бізнес-процесів реалізована через створення серверних модулів на базі фреймворку Laravel, що дозволяє ефективно обробляти запити користувачів, керувати сесіями та забезпечувати безпеку даних. Наприклад, модуль обліку замовлень автоматично оновлює статуси, формує звіти для адміністрації і генерує фінансові документи у форматі PDF за допомогою пакету DomPDF. Автентифікація користувачів та контроль доступу реалізовані через пакет Laravel Sanctum, що дозволяє налаштовувати різні ролі (адміністратор, менеджер, клієнт) та права доступу до функцій системи.

Для інтеграції з платіжними сервісами використовується HTTP-клієнт Guzzle, який забезпечує безпечний обмін інформацією з платіжними шлюзами, такими як LiqPay або Stripe. Це дозволяє приймати онлайн-платежі, автоматично підтверджувати оплату і оновлювати статус замовлення без втручання оператора.

Клієнтська частина платформи побудована на фреймворку Vue.js, який забезпечує динамічний, швидкий і адаптивний інтерфейс. Застосування TailwindCSS дає змогу легко і гнучко стилізувати компоненти, забезпечуючи сучасний дизайн. Валідація форм реалізована за допомогою VeeValidate, що дозволяє перевіряти коректність введення даних прямо у браузері без додаткових запитів на сервер, що підвищує зручність для користувача. Використання бібліотеки Chart.js дозволяє візуалізувати ключові бізнес-метрики, наприклад, обсяг продажів, кількість замовлень, залишки товарів, у вигляді інтерактивних графіків і діаграм, що полегшує прийняття управлінських рішень.

Фронтенд реалізується за допомогою сучасних JavaScript-фреймворків, що дозволяють створити адаптивний, швидкий та зручний інтерфейс. Валідація форм користувача забезпечується спеціалізованими бібліотеками, які гарантують коректність введених даних без додаткових запитів до сервера. Візуалізація аналітичних даних здійснюється за допомогою інтерактивних діаграм, що дають змогу власнику бізнесу в режимі реального часу відстежувати ключові показники роботи.

Розробка здійснювалася із застосуванням методології Agile, що дозволило гнучко реагувати на зміни вимог і швидко тестувати нові функції. Для контролю якості використовувалися фреймворки PHPUnit, Pest та Paratest, що забезпечували покриття коду тестами, виявлення помилок на ранніх етапах і стабільність роботи системи.

РОЗДІЛ 3. РОЗРОБЛЕННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-ПЛАТФОРМИ ДЛЯ УПРАВЛІННЯ ПРОЦЕСАМИ МАЛОГО БІЗНЕСУ

3.1 Проектування структури бази даних

3.1.1 Інфологічна (логічна) модель бази даних

Проектування бази даних платформи відбувається з побудови інфологічної моделі, що відображає ключові складові предметної області та їхні зв'язки. В цьому пункті зазначено інфологічну (логічну) модель бази даних, яка є концептуальним відображенням предметної області та визначає основні сутності, їхні атрибути та взаємозв'язки. Метою цього етапу є створення чіткого розуміння даних, які будуть зберігатися та оброблятися системою, незалежно від конкретної системи управління базами даних (СУБД).

На основі аналізу вимог до веб-платформи та її функціональних можливостей, було виділено наступні ключові сутності:

1. Ліди (Leads): Представляють потенційних клієнтів або зацікавлені сторони. Основні атрибути включають ідентифікатор ліда, вартість, джерело, тип, відповідального продавця, очікувану дату закриття, статус, дату створення, опис, заголовок та пов'язану контактну інформацію;
2. Контакти (Contacts) / Особи (Persons): Містять інформацію про контактних осіб, включаючи електронні листи, номери телефонів, назву організації, ім'я, прізвище та посаду;
3. Цитати (Quotes): Відображають комерційні пропозиції, що включають тему, відповідального, термін дії, перелік товарів/послуг, суми (проміжну, знижку, податок, загальну) та статус;
4. Продукти (Products): Описують товари або послуги, що пропонуються, з такими атрибутами як артикул, назва, ціна та наявність;

5. Завдання (Tasks): Представляють дії, які необхідно виконати, з атрибутами: ідентифікатор, назва, опис, дати (створення, початок, завершення), статус, пріоритет, відповідальний, автор та пов'язана сутність;

6. Події календаря (Calendar Events): Відображають заплановані події, включаючи назву, опис, час початку та завершення, місце проведення, учасників, організатора, тип події та нагадування;

7. Документи (Documents): Містять інформацію про завантажені файли, такі як назва, тип, дати створення та редагування, автор, шлях до файлу та статус;

8. Користувачі (Users): Представляють користувачів системи з їхніми обліковими даними (логін, пароль), особистою інформацією (ім'я, прізвище, email) та ролями.

Для кожної з цих сутностей було визначено набір важливих атрибутів, що характеризують їхні властивості та необхідні для функціонування веб-платформи. Крім того, було визначено наступні ключові зв'язки між сутностями:

- Зв'язок один-до-багатьох між "Клієнтами" та "Контактами" (один клієнт може мати багато контактних осіб);
- Зв'язок багато-до-багатьох між "Клієнтами" та "Продуктами" через проміжну таблицю "Історія покупок";
- Зв'язок один-до-багатьох між "Цитатами" та "Продуктами" через сутність "Позиції цитати";
- Зв'язок багато-до-одного між "Цитатами" та "Клієнтами" (одна цитата належить одному клієнту);
- Зв'язок один-до-багатьох між "Лідами" та "Контактами" (один лід може мати кілька контактних осіб);
- Зв'язок багато-до-одного між "Завданнями" та "Користувачами" (відповідальний та автор);
- Зв'язок багато-до-одного між "Подіями календаря" та "Користувачами" (організатор). Можливий зв'язок багато-до-багатьох через проміжну таблицю "Учасники події";

- Зв'язок багато-до-одного між "Документами" та "Користувачами" (автор);
- Зв'язки багато-до-одного між "Завданнями", "Подіями календаря", "Документами" та іншими сутностями ("Ліди", "Клієнти", "Цитати") через поля для пов'язаної сутності та її типу.

Ця інфологічна модель є основою для подальшого проєктування логічної моделі бази даних MySQL, де будуть визначені конкретні таблиці, стовпці, типи даних, первинні та зовнішні ключі, а також обмеження цілісності для забезпечення ефективного зберігання та управління даними веб-платформи.

3.1.2 Обґрунтування вибору СКБД та проєктування фізичної моделі

Вибір MySQL як системи керування базами даних для веб-платформи управління процесами малого бізнесу є обґрунтованим з кількох ключових причин:

1. Поширеність та зрілість: MySQL є однією з найпопулярніших та найстаріших реляційних СКБД. Її широке використання протягом багатьох років означає наявність великої спільноти розробників, численної документації, готових рішень та інструментів. Це полегшує процес розробки, налагодження та підтримки платформи;
2. Відкритий вихідний код та доступність: MySQL має подвійну ліцензію (GPL та комерційна), що робить її безкоштовною для більшості випадків використання. Це знижує витрати на розробку та розгортання платформи, особливо на початкових етапах розвитку малого бізнесу;
3. Сумісність з PHP та Laravel: MySQL чудово інтегрується з мовою програмування PHP та фреймворком Laravel, який ви обрали для серверної частини. Laravel надає зручні інструменти (Eloquent ORM, Query Builder) для взаємодії з базами даних MySQL, спрощуючи розробку запитів та операцій з даними;

4. Продуктивність та масштабованість: MySQL забезпечує достатню продуктивність для веб-застосунків малого бізнесу. Існують механізми для масштабування, такі як реплікація та кластеризація, які можуть бути використані для обробки зростаючих обсягів даних та збільшення навантаження;

5. Надійність та безпека: MySQL є надійною та стабільною СКБД, яка підтримує різні механізми забезпечення безпеки даних, включаючи керування доступом, шифрування та захист від поширених веб-загроз;

6. Інструменти адміністрування: Існує велика кількість безкоштовних та комерційних інструментів для адміністрування баз даних MySQL (наприклад, phpMyAdmin, MySQL Workbench), що полегшує керування, моніторинг та оптимізацію роботи бази даних.

Враховуючи ці фактори, вибір MySQL є оптимальним рішенням для вашої веб-платформи, забезпечуючи баланс між вартістю, продуктивністю, надійністю та зручністю розробки.

3.1.3 Контрольний приклад та підрахунок прогнозованих обсягів

Для перевірки працездатності ключових бізнес-процесів та оцінки потенційних обсягів даних веб-платформи було сформовано тестове середовище з імітацією активності малого бізнесу. Це включало створення значної кількості тестових даних, що охоплюють основні сутності системи.

- 10 лідів: Різних джерел та статусів, з призначенням відповідальним менеджерам;
- 12 клієнтів: 3 різною контактною інформацією та історією взаємодії.
- 7 контактних осіб: Пов'язаних з лідами та клієнтами;
- 3 комерційних пропозицій (цитат): 3 різною кількістю позицій товарів/послуг. В середньому кожна цитата містила 5 позицій;
- 10 товарів/послуг: 3 детальним описом та цінами;

- 5 завдань: Різного пріоритету та статусу, призначених користувачам та пов'язаних з іншими сутностями (лідами, клієнтами);
- 3 подій календаря: Включаючи зустрічі, дзвінки та нагадування, з різною кількістю учасників;
- 10 документів: Різних типів (рахунки, договори, накладні), пов'язаних з відповідними сутностями;
- 10 користувачів системи: З різними ролями та правами доступу;

Після первинного заповнення тестовими даними, середній розмір бази даних MySQL склав близько 0.5 - 1 МБ (оціночно, залежно від кількості та розміру текстових полів).

Для прогнозування майбутніх обсягів даних було зроблено припущення про середній щомісячний приріст даних на рівні 5-15 %, враховуючи потенне зростання бізнесу та інтенсивність використання веб-платформи.

За таких темпів приросту, прогнозований обсяг бази даних через рік становитиме:

- За мінімального приросту (5% щомісяця): Приблизно $1 \text{ МБ} \times (1+0.05)^{12} \approx 1.8 \text{ МБ}$;
- За середнього приросту (10% щомісяця): Приблизно $1 \text{ МБ} \times (1+0.10)^{12} \approx 3.1 \text{ МБ}$;
- За максимального приросту (15% щомісяця): Приблизно $1 \text{ МБ} \times (1+0.15)^{12} \approx 5.4 \text{ МБ}$;

Враховуючи, що для розміщення бази даних планується використання серверного обладнання з доступним дисковим простором у кілька сотень гігабайт (наприклад, RAID масив), прогнозовані обсяги даних навіть при значному зростанні використання платформи залишаються надзвичайно малими порівняно з доступними ресурсами.

Результати тестування з початковими обсягами даних та прогнозовані обсяги свідчать про те, що обрана СКБД MySQL та спроектована структура бази даних є більш ніж достатніми для зберігання та управління даними веб-платформи на початковому етапі розвитку малого бізнесу. Прогнозований приріст даних не становитиме жодної загрози для доступного дискового простору

протягом тривалого періоду, забезпечуючи величезний запас ресурсу для майбутнього масштабування без необхідності швидкої модернізації обладнання. Проведене тестування також підтвердило коректність реалізації зв'язків між сутностями та базову продуктивність системи при роботі з типовими початковими обсягами даних малого бізнесу.

3.2 Реалізація логіки обліку та керування бізнес-процесами

Цей розділ описує підходи та механізми, використані для реалізації ключової логіки обліку та керування бізнес процесами в розробленій веб платформі. Особлива увага приділяється використанню фреймворку Laravel та його можливостей для взаємодії з базою даних MySQL та реалізації бізнес правил.

Логіка обліку та керування лідами реалізована через CRUD операції (створення читання оновлення видалення) над сутністю "Лід" (таблиця leads). Laravel Eloquent ORM використовується для зручної взаємодії з базою даних. При створенні ліда дані введені у веб форму валідуються за допомогою Laravel Validation та зберігаються в таблиці leads через модель Lead. Одночасно може створюватися пов'язаний контакт у таблиці contacts через встановлене Eloquent відношення.

Для перегляду лідів застосовуються Eloquent запити з можливістю фільтрації за статусом джерелом відповідальним менеджером та іншими параметрами. Реалізована пагінація для ефективної обробки великої кількості записів. Оновлення інформації про лід (статус відповідальний вартість очікувана дата закриття) здійснюється через відповідні веб форми де дані перед оновленням проходять валідацію та зберігаються в базі за допомогою методу save моделі Lead. При зміні статусу ліда на "Виграно" реалізується логіка створення нового клієнта на основі даних ліда (запис у таблиці clients) та за потреби пов'язання існуючих

контактів. Для відстеження історії змін лідів застосовується механізм Laravel Eloquent Events або окрема таблиця аудиту.

Логіка обліку клієнтів та контактних осіб реалізована аналогічно до лідів з використанням моделей Client та Contact і відповідних таблиць. Особливу увагу приділено встановленню та використанню Eloquent відношень між клієнтами та їхніми контактними особами (один до багатьох). Реалізовано пошук клієнтів та контактів за різними критеріями а також можливість додавання редагування та видалення контактних осіб пов'язаних з конкретним клієнтом.

Керування цитатами включає створення цитат через форму яка дозволяє вибрати клієнта додавати товари чи послуги з таблиці products із зазначенням кількості та ціни. Дані зберігаються в таблицях quotes та quote items через відповідні Eloquent відношення. Перегляд цитат реалізований з можливістю фільтрації за статусом клієнтом датою створення. Редагування цитат передбачає зміну складу товарів або послуг кількості цін а також застосування знижок і податків. При редагуванні оновлюються записи в таблицях quotes та quote items. Для генерації PDF документів комерційних пропозицій використовується бібліотека Laravel DomPDF.

Для реалізації функціональності планування та керування завданнями і подіями календаря застосовуються моделі Task та Calendar Event і відповідні таблиці. Форми створення завдань і подій дозволяють вказувати назву опис терміни виконання відповідальних учасників (через зв'язок із користувачами та контактами) пріоритет. Відображення календаря реалізовано на клієнтській стороні за допомогою бібліотеки Vue cal із підвантаженням даних через API Laravel. Користувачам надається можливість змінювати статуси завдань (нове в процесі завершено).

Логіка обліку документів передбачає завантаження перегляд та керування документами пов'язаними з різними сутностями системи — лідами клієнтами цитатами. Завантаження файлів здійснюється через веб інтерфейс із збереженням

інформації про документ (назва тип шлях до файлу автор) у таблиці documents та фізичного файлу на сервері з використанням файлової системи Laravel. Реалізовано механізм зв'язування документів з іншими сутностями через поля related entity id та related entity type.

Для керування користувачами та їхніми правами доступу використовується модель User та таблиця users з полем role. Реалізовано реєстрацію і авторизацію користувачів з використанням Laravel Sanctum для API автентифікації. Забезпечено механізм розмежування прав доступу на основі ролей користувачів (адміністратор менеджер з продажу тощо). Для захисту маршрутів і дій застосовується Middleware Laravel залежно від ролі користувача.

Описано основні аспекти реалізації логіки обліку та керування бізнес процесами веб платформи з використанням фреймворку Laravel і структури бази даних MySQL. Подальші розділи можуть деталізувати розробку користувацького інтерфейсу та API для взаємодії з цією логікою.

Для обліку клієнтських замовлень реалізовано сутність "Замовлення" (orders) із атрибутами: ідентифікатор замовлення номер замовлення дата створення клієнт (зовнішній ключ) статус замовлення (нове в обробці виконано скасовано) загальна сума дата виконання відповідальний менеджер (зовнішній ключ) тощо. Існує зв'язок один до багатьох із клієнтами і позиціями замовлення. Сутність "Позиція замовлення" (order items) має атрибути: ідентифікатор позиції ідентифікатор замовлення товар або послугу (зовнішній ключ) кількість ціна за одиницю сума позиції. Зв'язок — багато до одного з замовленнями та продуктами.

Створення замовлення здійснюється через веб форму з вибором клієнта і додаванням товарів чи послуг із вказанням кількості. Дані зберігаються в таблицях orders та order items через Eloquent відношення. Для перегляду замовлень доступний список із фільтрацією за статусом клієнтом датою. Редагування замовлень дозволяє змінювати склад замовлення кількість товарів ціни та статус. Зміна статусу замовлення реалізує перехід між станами із можливістю сповіщень (наприклад через електронну пошту).

Для обліку витратних матеріалів передбачено сутність "Витратний матеріал" (consumables) з атрибутами: ідентифікатор назва одиниця виміру поточний залишок мінімальний залишок ціна за одиницю. Сутність "Рух витратних матеріалів" (consumable transactions) включає ідентифікатор ідентифікатор матеріалу тип операції (надходження списання) кількість дата операції опис операції пов'язане замовлення або ремонт.

Додавання витратних матеріалів здійснюється через веб форму із вказанням початкового залишку та мінімального порогу. Облік надходження фіксує надходження нових партій матеріалів збільшуючи залишок. Облік списання реєструє використання матеріалів при виконанні замовлень або ремонтів із зменшенням залишку та можливістю зазначення причин списання. Реалізовано контроль залишків з механізмом сповіщень при досягненні мінімального порогу.

Для обліку ремонтних процесів розроблено сутність "Ремонт" (repairs) із атрибутами: ідентифікатор номер ремонту дата надходження клієнт опис проблеми статус ремонту (прийнято в роботі виконано очікує видачі видано) відповідальний майстер дата завершення (очікувана і фактична) вартість ремонту. Сутність "Етап ремонту" (repair stages) містить ідентифікатор ідентифікатор ремонту назву етапу опис дату початку та завершення відповідального статус етапу. Сутність "Використані матеріали в ремонті" (repair consumables) включає ідентифікатор ідентифікатор ремонту ідентифікатор витратного матеріалу кількість.

Реєстрація ремонту виконується через веб форму із заповненням деталей. Для кожного ремонту можна додавати етапи з описом відповідальними та датами. Використання матеріалів при ремонті відображається в таблиці repair consumables що дає змогу відслідковувати витрати. Зміна статусів ремонту і етапів керується логікою бізнес процесів. В системі передбачено інтеграцію з замовленнями для зв'язку ремонту з клієнтськими замовленнями.

3.3 Розробка програмного забезпечення платформи

У цьому розділі детально описано повний цикл створення цієї системи. Тут представлено весь шлях від проєктування бізнес-логіки до розробки інструкцій для кінцевих користувачів. Процес розробки ґрунтувався на принципах модульності, що забезпечує гнучкість та легкість підтримки, а також тестованості для гарантії якості та стабільності функціонування платформи.

3.3.1 Проєктування програмного забезпечення

Платформа була побудована з використанням гнучкої методології Agile, що дозволило забезпечити адаптивність до змін вимог та швидке тестування функціональності. Основою серверної логіки платформи є мова програмування PHP з використанням фреймворку Laravel, що забезпечує високу швидкість розробки, безпеку та підтримку сучасних архітектурних рішень. Для зберігання даних використовується реляційна база даних MySQL, яка є надійною, масштабованою і добре інтегрується з Laravel.

Для стилізації інтерфейсу застосовувався TailwindCSS, який дозволив швидко створювати адаптивний та візуально привабливий дизайн без значних зусиль на написання власного CSS.

Для розширення функціональності клієнтської частини було використано наступні додаткові компоненти та бібліотеки [Vue.js](https://vuejs.org/):

- VeeValidate: для реалізації валідації форм;
- Vue-flatpickr: для забезпечення зручної роботи з календарями та вибору дат;
- Vue-cal: для побудови функціонального календарного планування;

- Vuedraggable: для реалізації можливості перетягування елементів інтерфейсу (drag-n-drop);
- Chart.js: для створення інтерактивних графіків та діаграм, що використовуються для візуалізації даних;
- В процесі розробки використовувалися наступні інструменти для управління залежностями;
- Composer: як менеджер PHP-пакетів для керування бібліотеками та залежностями серверної частини;
- NPM (Node Package Manager): як менеджер JavaScript-пакетів для керування залежностями фронтенду;
- Для генерації сторінок використовувався Blade — шаблонний рушій фреймворку Laravel, який також забезпечує інтеграцію з компонентами Vue.js. Взаємодія між frontend та backend частинами реалізована за допомогою подій (events) через механізм `view_render_event`. Підтримка багатомовності інтерфейсу була закладена на етапі розробки завдяки використанню Blade-директиви `@lang` для реалізації локалізації в майбутньому;

3.3.2 Вибір архітектури програмного забезпечення

Зважаючи на вимоги до гнучкості, масштабованості та зручності розробки, було обрано багаторівневу архітектуру (multi-tier architecture), яка поділяє застосунок на логічно незалежні шари. Це сприяє кращій організації коду, полегшує підтримку та масштабування окремих компонентів.

Основними шарами обраної архітектури є рівень представлення (Presentation Tier): Відповідає за відображення інформації користувачеві та обробку його взаємодії з платформою. У даному проєкті цей рівень реалізовано за допомогою Vue.js, сучасного JavaScript-фреймворку, що забезпечує реактивний та динамічний інтерфейс. Рівень бізнес-логіки (Application Tier): Містить основну бізнес-логіку застосунку, обробку даних та координацію дій між рівнем

представлення та рівнем даних. Цей рівень реалізовано на стороні сервера з використанням мови програмування PHP та фреймворку Laravel. Laravel надає потужні інструменти для розробки бізнес-логіки, маршрутизації, валідації даних та забезпечення безпеки.

Рівень даних (Data Tier): Відповідає за зберігання та отримання даних. У проєкті використовується реляційна база даних MySQL, обрана за її надійність, масштабованість та хорошу інтеграцію з Laravel.

Дана багаторівнева архітектура має наступні переваги:

- Розділення відповідальності (Separation of Concerns): Кожен рівень відповідає за свою чітко визначену функціональність, що полегшує розробку, тестування та підтримку коду;
- Повторне використання коду (Code Reusability): Бізнес-логіка, реалізована на відповідному рівні, може бути використана різними клієнтськими інтерфейсами (наприклад, веб-інтерфейсом та можливим API для мобільних застосунків у майбутньому);
- Масштабованість (Scalability): Окремі рівні можуть бути масштабовані незалежно один від одного залежно від навантаження. Наприклад, якщо зростає кількість користувачів, можна збільшити кількість серверів, що обробляють бізнес-логіку;
- Гнучкість (Flexibility): За потреби технології, що використовуються на кожному рівні, можуть бути замінені або оновлені без значного впливу на інші рівні;
- Крім багаторівневої архітектури, у проєкті також застосовувалися принципи RESTful API для взаємодії між клієнтською та серверною частинами. RESTful API забезпечує стандартизований та stateless спосіб обміну даними через протокол HTTP, що спрощує інтеграцію та робить систему більш гнучкою;
- Використання фреймворку Laravel сприяло застосуванню патерну Model-View-Controller (MVC) на рівні бізнес-логіки, що додатково структурує код та покращує його організацію;

Дана архітектура програмного забезпечення є оптимальною для розробки веб-платформи управління процесами малого бізнесу, забезпечуючи необхідну гнучкість, масштабованість, підтримуваність та надійність системи.

3.3.3 Опис реалізації основних модулів

Модуль управління лідами

Призначення: Модуль призначений для обліку та керування потенційними клієнтами (лідами) на різних етапах воронки продажів. Він дозволяє фіксувати нові ліди, відстежувати їхній прогрес, призначати відповідальних менеджерів та конвертувати їх у клієнтів.

Реалізація: для зберігання інформації про ліди використовується модель Lead та відповідна таблиця leads у базі даних MySQL. CRUD-операції (створення, читання, оновлення, видалення) реалізовані за допомогою контролерів Laravel (LeadController) та Eloquent ORM. Валідація вхідних даних забезпечується за допомогою Laravel Validation. Інтерфейс користувача для роботи з лідами (список, форма створення/редагування, детальна інформація) розроблено з використанням Vue.js компонентів. Зв'язок з контактними особами реалізовано через Eloquent-відношення hasMany між моделями Lead та Contact. Логіка зміни статусу ліда (наприклад, конвертація) включає оновлення поля status в таблиці leads та, за потреби, створення нового запису в таблиці clients на основі даних ліда. Для відображення джерел лідів використовується модель LeadSource та таблиця lead_sources зі зв'язком belongsTo з моделлю Lead.

Модуль управління клієнтами та контактами

Призначення: Модуль забезпечує централізований облік інформації про існуючих клієнтів та їхні контактні особи.

Реалізація: для зберігання даних про клієнтів використовуються моделі Client та таблиця clients. Для зберігання даних про контактних осіб використовується модель Contact та таблиця contacts.

Реалізовано Eloquent-відношення hasMany між моделлю Client та Contact.

CRUD-операції для клієнтів та контактів реалізовані за допомогою відповідних контролерів (ClientController, ContactController).

Інтерфейс користувача для роботи з клієнтами та контактами розроблено з використанням Vue.js компонентів, що забезпечують зручний перегляд, додавання, редагування та видалення інформації. Реалізовано пошук та фільтрацію клієнтів за різними критеріями.

Модуль управління комерційними пропозиціями (цитатами)

Призначення: Модуль дозволяє створювати, редагувати та відстежувати комерційні пропозиції для клієнтів.

Реалізація: для зберігання основної інформації про цитати використовується модель Quote та таблиця quotes. Для зберігання інформації про товари/послуги в цитаті використовується модель QuoteItem та таблиця quote_items зі зв'язками belongsTo з моделями Quote та Product. CRUD-операції для цитат та позицій цитат реалізовані за допомогою контролера QuoteController. Інтерфейс користувача для створення та редагування цитат включає можливість вибору клієнта, додавання товарів/послуг, вказання кількості та ціни. Генерація PDF-документів цитат реалізована за допомогою бібліотеки Laravel DomPDF, з використанням Blade-шаблонів для формування вмісту PDF на основі даних з бази даних.

Модуль планування та керування завданнями та подіями календаря

Призначення: Модуль надає інструменти для планування завдань, зустрічей, дзвінків та інших подій, пов'язаних з бізнес-процесами.

Реалізація: для зберігання інформації про завдання використовується модель Task та таблиця tasks. Для зберігання інформації про події календаря використовується модель CalendarEvent та таблиця calendar_events. CRUD-операції для завдань та подій реалізовані за допомогою контролерів TaskController та CalendarEventController. Інтерфейс календаря реалізовано з використанням Vue.js компоненти Vue-cal, що відображає події, отримані з backend через API Laravel. Зв'язки між завданнями/подіями та іншими сутностями (лідами, клієнтами, користувачами) реалізовані через зовнішні ключі.

Модуль обліку документів

Призначення: Модуль забезпечує завантаження, зберігання та керування документами, пов'язаними з різними аспектами бізнесу.

Реалізація: для зберігання інформації про документи використовується модель Document та таблиця documents. Функціональність завантаження файлів реалізована за допомогою можливостей Laravel Filesystem. Інформація про завантажений файл (назва, шлях, розмір) зберігається в базі даних. CRUD-операції для документів реалізовані за допомогою контролера DocumentController. Реалізовано зв'язки між документами та іншими сутностями через поля related_entity_id та related_entity_type.

Модуль ведення звітності та аналітики

Призначення: Модуль надає користувачам інструменти для отримання звітів та аналітичних даних про ключові бізнес-процеси.

Реалізація: логіка формування звітів реалізована в окремих класах (наприклад, LeadReportService, OrderReportService), які використовують Eloquent для вибірки та агрегації даних з бази даних. Маршрути та контролери (ReportController) відповідають за генерацію та відображення звітів на основі запитів користувача. Для візуалізації даних на інформаційних панелях використовуються Vue.js компоненти та бібліотека Chart.js, які отримують агреговані дані з backend через API. Реалізовано можливість фільтрації звітів за параметрами (період, статус, відповідальний тощо). Для експорту звітних даних у формати CSV або Excel можуть використовуватись бібліотеки Laravel Excel.

3.3.4 Тестування програмного забезпечення

Стратегія тестування базувалася на принципах раннього та частого тестування в рамках методології Agile. Це включало проведення тестування на кожному етапі розробки, починаючи з окремих компонентів і закінчуючи

інтеграцією всієї системи. Основні типи тестування, що застосовувалися, включають:

- Юніт-тестування: Тестування окремих функцій, класів та методів для перевірки їхньої коректної роботи;
- Інтеграційне тестування: Тестування взаємодії між різними модулями та компонентами системи;
- Функціональне тестування: Тестування основних бізнес-процесів та функціональних можливостей платформи з точки зору кінцевого користувача;
- UI-тестування: Тестування користувацького інтерфейсу на відповідність макетам, зручність використання та коректне відображення даних;
- Тестування продуктивності: Оцінка швидкодії та стабільності системи при різних рівнях навантаження;
- Тестування безпеки: Перевірка системи на наявність вразливостей та захист від несанкціонованого доступу;

Для проведення тестування використовувалися наступні інструменти:

- PHPUnit та Pest: Фреймворки для написання та запуску юніт- та функціональних тестів серверної частини (PHP). Pest використовувався завдяки своєму більш виразному синтаксису, що полегшує написання та читання тестів;
- Paratest: Інструмент для паралельного виконання тестів PHPUnit та Pest, що значно скоротило час виконання тестового набору в процесі CI/CD;
- Laravel Dusk: Фреймворк для проведення браузерних інтеграційних тестів, що дозволило імітувати дії користувача та перевіряти взаємодію між frontend та backend частинами;
- Jest та Vue Test Utils: Фреймворки для написання та запуску юніт-тестів компонентів Vue.js на клієнтській стороні;
- Postman та Insomnia: Інструменти для тестування API кінцевих точок (endpoints) серверної частини;
- LoadRunner або Apache JMeter : Для проведення тестів продуктивності та навантаження (якщо виникала необхідність оцінки поведінки системи при великій кількості одночасних користувачів);

- OWASP ZAP або подібні інструменти: Для проведення автоматизованого сканування на вразливості безпеки;

Процес тестування був інтегрований в цикл розробки. Після розробки кожної нової функціональності проводилося:

- Юніт-тестування: юніт-тести для перевірки окремих частин коду;
- Інтеграційне тестування: Перевірялася взаємодія нових компонентів з існуючими;
- UI-тестування: Перевірялася коректність відображення та зручність використання інтерфейсу;
- Автоматизовані тести (юніт-, інтеграційні та браузерні) запускалися автоматично при кожному коміті коду в систему контролю версій (наприклад, Git) за допомогою інструментів CI/CD (Continuous Integration/Continuous Delivery), таких як GitLab CI або GitHub Actions. Це дозволяло швидко виявляти регресії та забезпечувати стабільність кодової бази;
- Результати тестування документувалися, а виявлені помилки (баги) реєструвалися в системі відстеження помилок (наприклад, Jira або Redmine) з присвоєнням пріоритету та відповідального розробника для їхнього виправлення. Після виправлення помилок проводилося повторне тестування (регресійне тестування) для перевірки, що помилка усунена і не спричинила нових проблем;
- Проведене багаторівневе тестування дозволило забезпечити високий рівень якості розробленої веб-платформи. Більшість критичних помилок були виявлені та усунені на ранніх етапах розробки. Результати тестів продуктивності показали, що система здатна витримувати очікуване навантаження для малого бізнесу. Тестування безпеки виявило та допомогло усунути потенційні вразливості;

Загалом, застосована стратегія тестування та використання відповідних інструментів дозволили створити стабільну, надійну та зручну у використанні веб-платформу для управління процесами малого бізнесу.

3.3.5 Керівництво користувача

Інструкція з отримання доступу до веб-платформи містить інформацію про URL-адресу, вимоги до браузера та необхідність підключення до Інтернету. Авторизація передбачає детальний опис процесу входу в систему з використанням логіна та пароля, а також інформацію про відновлення пароля у разі його втрати. Інтерфейс користувача охоплює огляд основних елементів інтерфейсу, навігаційного меню, інформаційних панелей та їх призначення, а також опис розташування ключових функціональних блоків. Налаштування профілю включають інструкції з редагування особистої інформації користувача, зміни пароля та налаштування сповіщень, якщо така функція доступна. Для створення нового ліда надається опис полів форми, їх призначення, таких як джерело, статус, відповідальний тощо.

Перегляд списку лідів включає опис фільтрів і можливостей сортування, а редагування ліда — інструкції зі зміни інформації про лід та додавання нотаток. Процес конвертації ліда описує перетворення ліда на клієнта. Управління клієнтами та контактами передбачає перегляд списку клієнтів з описом відображуваної інформації та можливостей пошуку, створення нового клієнта з описом полів форми та редагування інформації, а також управління контактними особами — додавання, редагування та видалення контактів, пов'язаних із клієнтом. В частині управління комерційними пропозиціями (цитатами) наведено інструкції зі створення нової цитати, включно з вибором клієнта, додаванням товарів чи послуг, вказанням кількості, ціни, застосуванням знижок та податків, а також переглядом списку цитат і їх статусів. Редагування цитати включає генерацію PDF-документа для друку.

Планування та керування завданнями передбачає створення нового завдання з вказанням назви, опису, терміну виконання, відповідального та пріоритету, перегляд списку завдань за статусом, відповідальним чи терміном, редагування

завдання та зміну його статусу. Використання календаря охоплює перегляд за різними періодами (день, тиждень, місяць), створення нових подій (зустрічі, дзвінки, нагадування), редагування і видалення подій, а також запрошення учасників на події, якщо ця функція доступна. В розділі обліку документів описані процеси завантаження нових документів, перегляду списку завантажених, пошуку за назвою чи іншими критеріями, а також зв'язування документів з іншими сутностями, такими як ліди чи клієнти.

Ведення звітності передбачає перегляд доступних звітів за лідами, клієнтами, цитатами, замовленнями, витратними матеріалами та ремонтами, налаштування параметрів звіту, таких як період та фільтри, а також експорт звітів у формати CSV чи Excel. Перегляд інформаційних панелей (Dashboard) містить опис основних віджетів і їх призначення, інтерпретацію даних та можливість налаштування відображення, якщо така функція реалізована.

Облік клієнтських замовлень, якщо модуль є в системі, включає створення нового замовлення, перегляд списку замовлень за статусами, редагування та зміну статусу замовлення. Облік витратних матеріалів передбачає перегляд списку матеріалів і їх залишків, фіксацію надходження та списання, а також перегляд історії руху матеріалів. Описуються також додаткові функції платформи, наприклад, сповіщення та пошук по всій системі, даються поради та рекомендації щодо ефективного використання платформи для різних ролей користувачів, наведено поширені запитання та відповіді (FAQ) і контактну інформацію служби підтримки.

Окремо описані реалізовані функції для забезпечення доступності, такі як навігація за допомогою клавіатури, підтримка скрінрідерів, кольорові контрасти, а також інструкції з використання спеціальних можливостей операційної системи і браузера для покращення досвіду користування платформою.

3.4 Технічне забезпечення

3.4.1 Обґрунтування вибору середовища розгортання

Вибір середовища розгортання є критично важливим для забезпечення надійності, продуктивності та безпеки веб-платформи. Для розгортання розробленої системи було прийнято рішення використовувати віртуальний приватний сервер (VPS). Це рішення було обґрунтовано наступними факторами:

- **Контроль та гнучкість:** VPS надає значно більший рівень контролю над операційною системою, встановленим програмним забезпеченням та конфігурацією сервера порівняно зі звичайним shared-хостингом. Це дозволяє оптимально налаштувати середовище під потреби веб-платформи, зокрема встановити необхідні версії PHP, MySQL, налаштувати параметри безпеки та керувати ресурсами;
- **Ресурси, що гарантуються:** На відміну від shared-хостингу, де ресурси (CPU, RAM, дисковий простір) можуть бути розділені між багатьма користувачами, VPS зазвичай надає гарантований обсяг ресурсів. Це забезпечує більш стабільну та передбачувану продуктивність веб-платформи, особливо при зростанні навантаження;
- **Масштабованість:** VPS є більш масштабованим рішенням порівняно зі shared-хостингом. У міру зростання бізнесу та збільшення кількості користувачів та даних, ресурси VPS можуть бути відносно легко збільшені (апгрейд тарифного плану) без необхідності перенесення всієї системи на новий фізичний сервер на початкових етапах;
- **Безпека:** VPS забезпечує кращу ізоляцію від інших користувачів на сервері, що зменшує ризики, пов'язані з проблемами безпеки на сусідніх віртуальних машинах. Крім того, адміністратор VPS має повний контроль над налаштуваннями безпеки сервера (файрвол, правила доступу тощо);

- Вартість: VPS є компромісним рішенням між дешевим shared-хостингом з обмеженими можливостями та дорогим виділеним сервером з надмірними ресурсами на початкових етапах розвитку малого бізнесу. Він забезпечує достатню потужність та гнучкість за прийнятною ціною;

Вибір конкретного провайдера VPS здійснювався на основі наступних критеріїв:

- Репутація та надійність: Провайдер повинен мати позитивні відгуки та гарантувати високий uptime (час безперебійної роботи) серверів;

- Розташування серверів: Вибиралося розташування серверів, максимально наближене до цільової аудиторії для мінімізації затримки (latency). Технічна підтримка: Наявність кваліфікованої та оперативної технічної підтримки є важливим фактором для вирішення можливих проблем;

- Вартість та тарифні плани: Обиралися тарифні плани, що відповідають прогнозованим потребам у ресурсах на початковому етапі з можливістю подальшого масштабування;

- Додаткові послуги: Наявність таких послуг, як автоматичне резервне копіювання (backup), моніторинг сервера та інструменти керування VPS (наприклад, cPanel або Plesk), також враховувалася. Враховуючи ці фактори, було обрано надійного провайдера VPS з серверами в Європі, що забезпечує необхідний рівень контролю, продуктивності, масштабованості та безпеки для успішного розгортання та функціонування веб-платформи управління процесами малого бізнесу;

3.4.2 Ресурсні вимоги до розгортання та підтримки

Для успішного розгортання та подальшої підтримки веб-платформи управління процесами малого бізнесу необхідно враховувати кілька ресурсних вимог. Що стосується апаратних ресурсів на стороні сервера, на початковому етапі достатньо 2-4 віртуальних ядер процесора (vCores), проте зі збільшенням навантаження, наприклад, зростанням кількості користувачів або інтенсивності

використання, може знадобитися збільшення кількості ядер. Мінімально необхідний обсяг оперативної пам'яті становить 4-8 ГБ RAM для забезпечення плавної роботи операційної системи, веб-сервера (Apache або Nginx) та PHP-FPM, при цьому рекомендовано мати запас RAM для обробки пікових навантажень. Для дискового простору спочатку достатньо 50-100 ГБ SSD для розміщення файлів веб-платформи, бази даних, логів та резервних копій, при цьому потреба в дисковому просторі зростатиме з обсягом завантажених документів і даних у базі. Обсяг необхідного мережевого трафіку залежить від інтенсивності використання платформи, розміру файлів та кількості користувачів, тому рекомендується обирати тарифний план з достатнім або необмеженим трафіком.

Щодо програмних ресурсів на стороні сервера, рекомендується використовувати стабільну версію Linux-дистрибутиву, наприклад, Ubuntu, Debian або CentOS, як надійну і безпечну основу серверного середовища. Основним веб-сервером може бути Apache (з модулем `mod_php` або PHP-FPM) або Nginx (з PHP-FPM) для обробки HTTP-запитів. Важливо використовувати актуальну стабільну версію PHP, сумісну з фреймворком Laravel, з необхідними PHP-розширеннями, такими як PDO, MySQLi, GD, cURL, OpenSSL, Mbstring, XML. Для керування базою даних потрібна сумісна з Laravel версія MySQL. Composer використовується як менеджер залежностей PHP для встановлення та оновлення пакетів Laravel та інших бібліотек. Для фронтенд-частини необхідні менеджер залежностей JavaScript NPM для роботи з Vue.js, TailwindCSS та іншими пакетами, а також інструмент збірки Vite або аналогічний для оптимізації frontend-ресурсів. Система контролю версій Git забезпечує керування кодовою базою та розгортання оновлень. Інструменти моніторингу та логування, такі як Prometheus, Grafana або ELK Stack, потрібні для відстеження стану сервера, продуктивності платформи та запису подій. Для безпеки даних важливим є налаштування автоматичного резервного копіювання бази даних і файлів платформи.

Щодо людських ресурсів для підтримки платформи, необхідний адміністратор сервера або DevOps інженер, який відповідатиме за початкове налаштування сервера, забезпечення безпеки, моніторинг продуктивності, керування резервними копіями та розгортання оновлень. Розробник, що працює з бекендом і фронтендом, відповідатиме за виправлення помилок, розробку нових функціональних можливостей і внесення змін у кодову базу. Технічна підтримка надаватиме допомогу кінцевим користувачам у вирішенні проблем, пов'язаних з використанням платформи.

3.5 Розгортання платформи та перевірка її працездатно

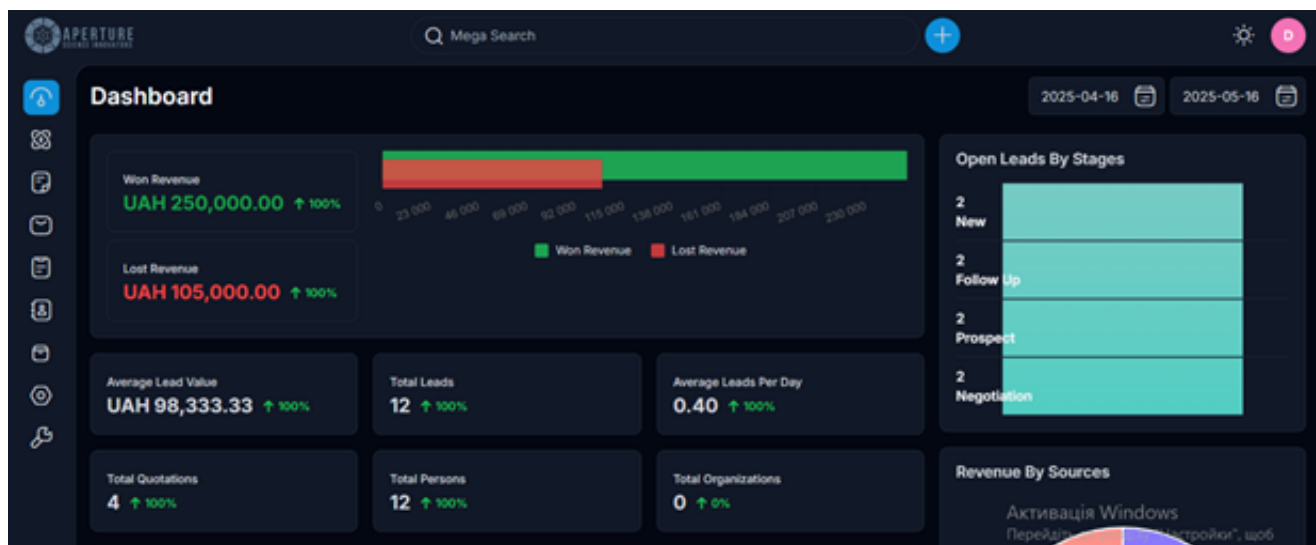


Рисунок - 3.10 – Головна сторінка

Джерело: розроблено автором самостійно

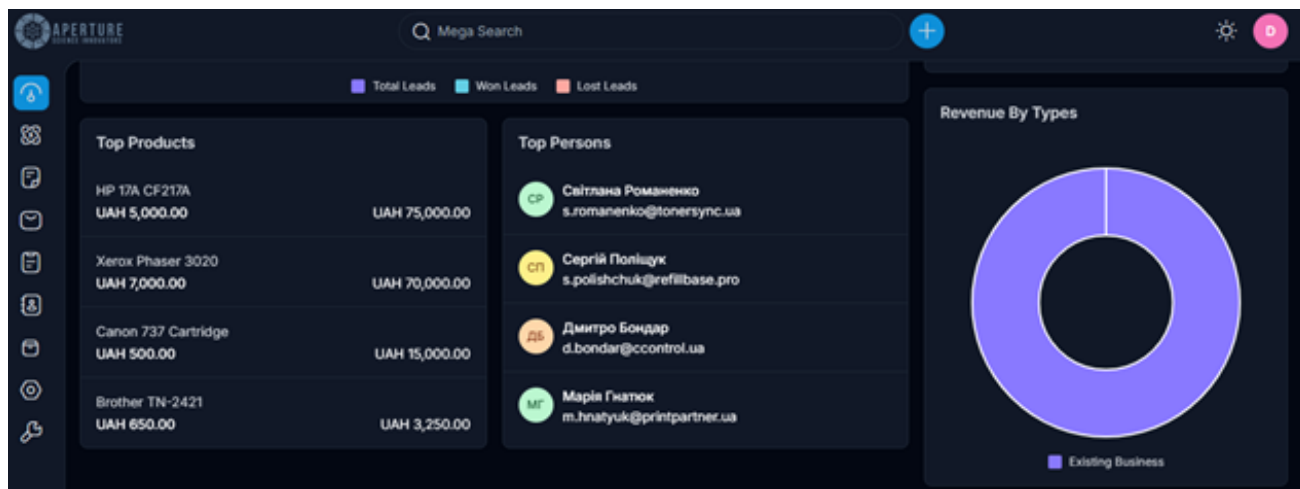


Рисунок - 3.11–Головна сторінка

Джерело: розроблено автором самостійно

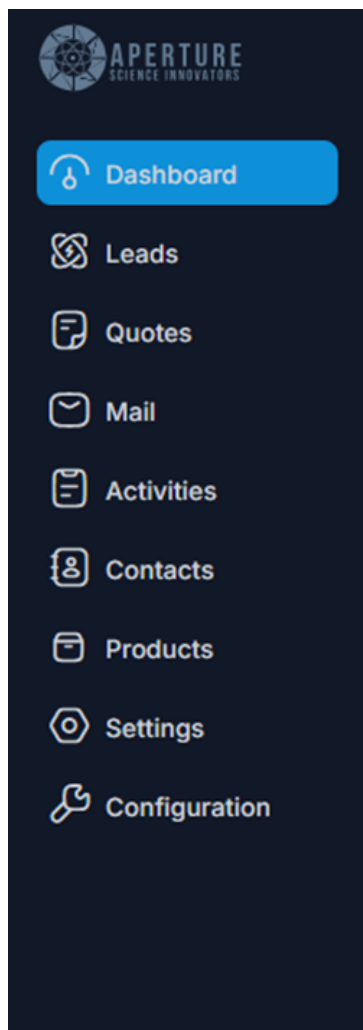


Рисунок - 3.12 – Бокова панель

Джерело: розроблено автором самостійно

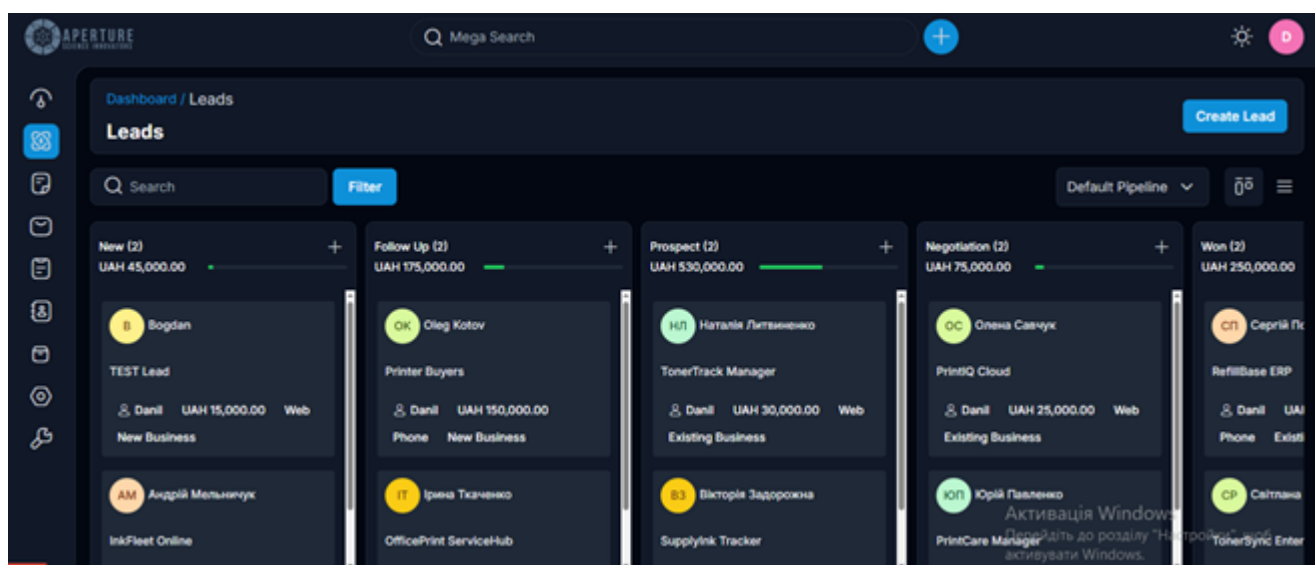


Рисунок - 3.13 – Сторінка "Leads"

Джерело: розроблено автором самостійно

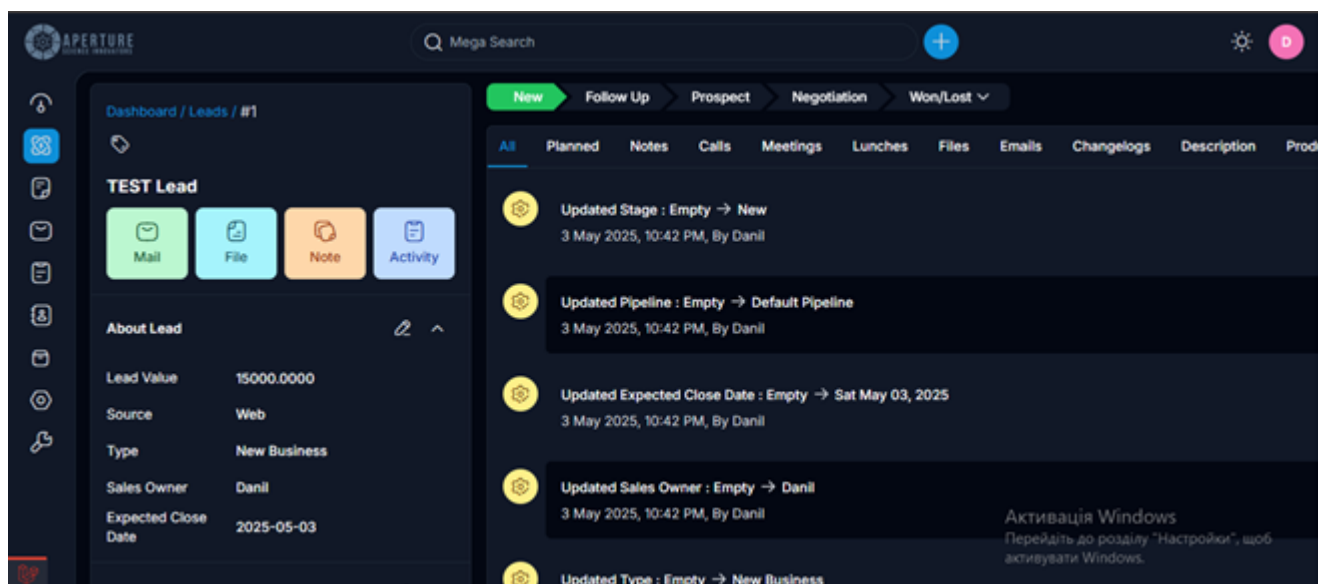


Рисунок - 3.14 – Сторінка зберігання інформації

Джерело: розроблено автором самостійно

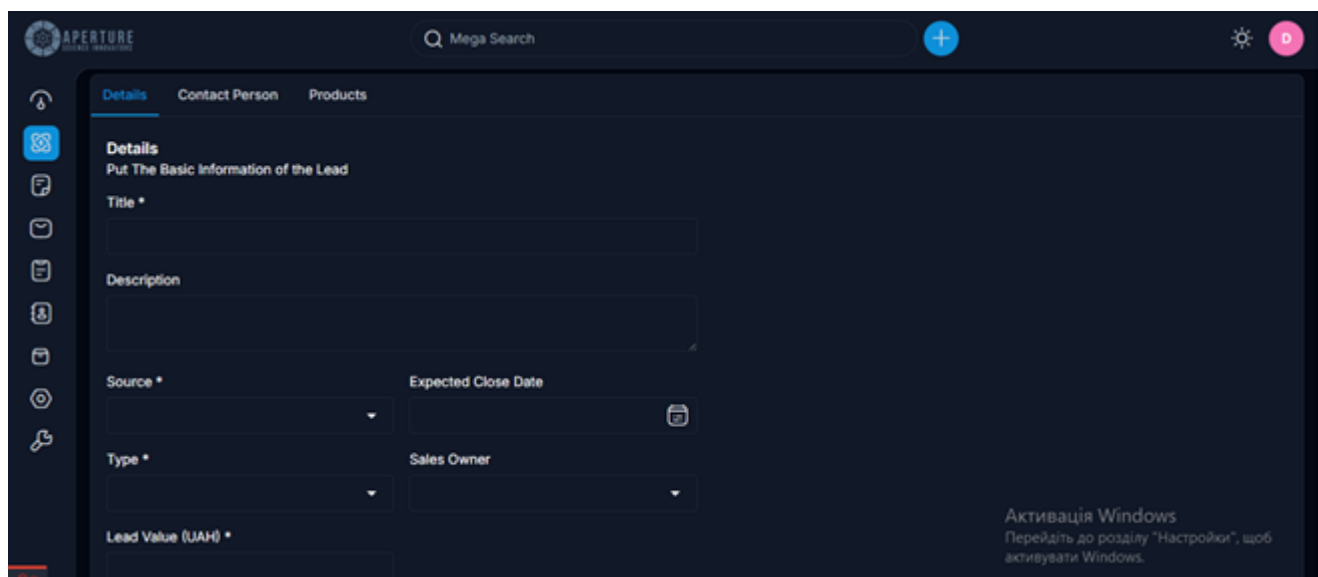


Рисунок - 3.15 – Сторінка інформації про клієнта

Джерело: розроблено автором самостійно

APERTURE

Mega Search

Click to Add

Email *

Work

+ Add More

Contact Number

Work

+ Add More

Organization

Click to add

Products

Information About the Products

Product Name	Quantity	Price	Amount	Action
Активация Windows				Перейдіть до розділу "Настройки", щоб активувати Windows.

+ Add More

Рисунок - 3.16 – Сторінка інформації про клієнта

Джерело: розроблено автором самостійно

APERTURE

Mega Search

Work

+ Add More

Contact Number

Work

+ Add More

Organization

Click to add

Products

Information About the Products

Can

Canon 737 Cartridge

Canon PIXMA G2411

+ Add More

Активация Windows

Перейдіть до розділу "Настройки", щоб активувати Windows.

Рисунок - 3.17 – Сторінка інформації про клієнта

Джерело: розроблено автором самостійно

	Subject	Sales Person	Expired Quote	Person	Subtotal	Discount	Tax	Adjustment	Grand Total	Expired At	Created At	Actions
☐	PrintPartner CRM	Danil	2025-08-21 00:00:00	Марія Гнатюк	UAH 10,500.00	UAH 0.00	UAH 0.00	UAH 0.00	UAH 10,500.00	21 Aug 2025	10 May 2025 09:56PM	
☐	PrintCare Manager	Danil	2025-07-10 00:00:00	Юрій Павленко	UAH 150,00 0.00	UAH 0.00	UAH 0.00	UAH 0.00	UAH 150,00 0.00	10 Jul 2025	10 May 2025 09:53PM	
☐	Supplylink Tracker	Danil	2025-06-07 00:00:00	Вікторія Задорожна	UAH 375,00 0.00	UAH 0.00	UAH 0.00	UAH 0.00	UAH 375,00 0.00	07 Jun 2025	10 May 2025 09:50PM	
☐	Oleg Kotov	Danil	2025-05-11 00:00:00	Oleg Kotov	UAH 300,00 0.00	UAH 0.00	UAH 0.00	UAH 0.00	UAH 300,00 0.00	11 May 2025	10 May 2025 07:37PM	

Рисунок - 3.18 – Сторінка комерційної пропозиції

Джерело: розроблено автором самостійно

Edit Quote

Quote Information | Address Information | Quote Items

Quote Information
Put the basic information of the quote.

Subject *
PrintPartner CRM

Description
CRM-система для бізнесу з ремонту принтерів і доставки картриджів

Sales Owner *
Danil

Expired At *
2025-08-21

Person *
Марія Гнатюк

Link to lead
PrintPartner CRM

Активация Windows
Перейдіть до розділу "Настройки", щоб активувати Windows.

Рисунок - 3.19 – Сторінка редагування пропозицій

Джерело: розроблено автором самостійно

Головна сторінка системи надає зведену інформацію про ключові показники бізнесу. Відображаються графіки, діаграми та числові значення. "Won Deals" показує загальну суму доходу від успішно закритих угод (функція - моніторинг ефективності продажів). "Lost Deals" відображає суму потенційного доходу, втраченого через невдалі угоди (функція - аналіз причин втрат). "Avg. Lead Value"

показує середній дохід, принесений одним лідом (функція - оцінка цінності потенційних клієнтів). "Total Leads" відображає загальну кількість потенційних клієнтів (функція - моніторинг обсягу роботи). "Leads by Stages" графічно представляє розподіл лідів за етапами воронки продажів (функція - візуалізація прогресу лідів). "Revenue by Sources" показує, з яких джерел надходить дохід (функція - аналіз ефективності каналів). Призначення панелі - надання керівництву та відділу продажів швидкого огляду для прийняття рішень.

Розширений аналітичний розділ деталізує інформацію про продукти, персонал та структуру доходів. "Top Products" показує найбільш продавані продукти (функція - визначення пріоритетних продуктів). "Top Persons" відображає рейтинг співробітників за ефективністю (функція - мотивація персоналу). "Revenue by Types" показує розподіл доходу за категоріями (функція - розуміння структури доходів). Призначення розділу - глибший аналіз даних для оптимізації.

Бокова панель містить основні розділи системи: "Dashboard" (перехід на головну), "Leads" (управління лідами), "Quotes" (управління пропозиціями), "Mail" (інтеграція з поштою), "Activities" (планування завдань), "Contacts" (управління контактами), "Products" (облік товарів), "Settings" (конфігурація), "Configuration" (розширені налаштування).

Розділ "Leads" візуалізує ліди на стадіях процесу продажів у форматі канбан-дошки. Стадії (New, Follow Up, Prospect, Negotiation, Won) представляють етапи (функція - візуалізація воронки). Картки лідів відображають окремих клієнтів (функція - перегляд/редагування даних). Переміщення карток змінює стадію ліда (функція - оновлення статусу). Кнопка "Create Lead" додає нового ліда. Пошук та фільтри знаходять та сортують ліди. Призначення розділу, це організація процесу продажів.

Сторінка містить повну інформацію про обраний лід. Основні дані ліда, історія змін (функція - відстеження взаємодії), нотатки, файли, активності. Призначення сторінки це, централізоване зберігання інформації.

Форма створює нового ліда. Поля введення для назви, опису, джерела, типу, вартості, дати закриття, відповідального. Кнопка "Save" додає ліда. Призначення форми - збір інформації про лідів.

Розділи вводять контактну інформацію та інформацію про продукти ліда. Контактна інформація (email, телефон, організація) зберігає дані для зв'язку. Інформація про продукти вибирає продукти, що цікавлять ліда. Призначення розділів - розширення інформації про ліда.

Інтерфейс вибирає продукт при додаванні інформації про потреби ліда. Функції - пошук та вибір продуктів. Призначення - зручний спосіб зв'язати ліда з продуктами. Розділ "Комерційні пропозиції" містить таблицю зі списком пропозицій. Список пропозицій відображає інформацію про кожен пропозицію (функція - перегляд/управління). Кнопка "Create Quote" створює нову пропозицію. Пошук та фільтри знаходять пропозиції. Призначення розділу - управління процесом створення та відстеження пропозицій. Форма редагує інформацію про комерційну пропозицію. Поля введення для теми, опису, відповідального, дати закінчення, клієнта, ліда. Таблиця продуктів додає/редагує позиції. Призначення форми це налаштування пропозиції.

The screenshot displays the 'Delivery Page' (Сторінка доставки) in the Aperture CRM system. The page is divided into two main sections: 'Shipping Address' and 'Quote Items'.

Shipping Address: This section contains a form for entering the shipping address. The address is 'Amager Boulevard 70, 2300 København, Denmark'. There is a 'Select Country' dropdown menu with 'København' selected. Below the address, there are input fields for 'København' and '2300'.

Quote Items: This section contains a table for managing quote items. The table has the following columns: Product Name, Quantity, Price (UAH), Amount (UAH), Discount (UAH), Tax (UAH), and Total (UAH). The table contains one item: 'Brothe...' with a quantity of 30, a price of UAH 650.00, an amount of UAH 19,500.00, a discount of UAH 0.00, and a total of UAH 19,500.00.

The page also features a sidebar with navigation icons, a search bar labeled 'Mega Search', and a settings icon in the top right corner.

Product Name	Quantity	Price (UAH)	Amount (UAH)	Discount (UAH)	Tax (UAH)	Total (UAH)
Brothe... X	30	UAH 650.00	UAH 19,500.00	UAH 0.00	UAH 0.00	UAH 19,500.00

Рисунок - 3.20 – Сторінка доставки

Джерело: розроблено автором самостійно

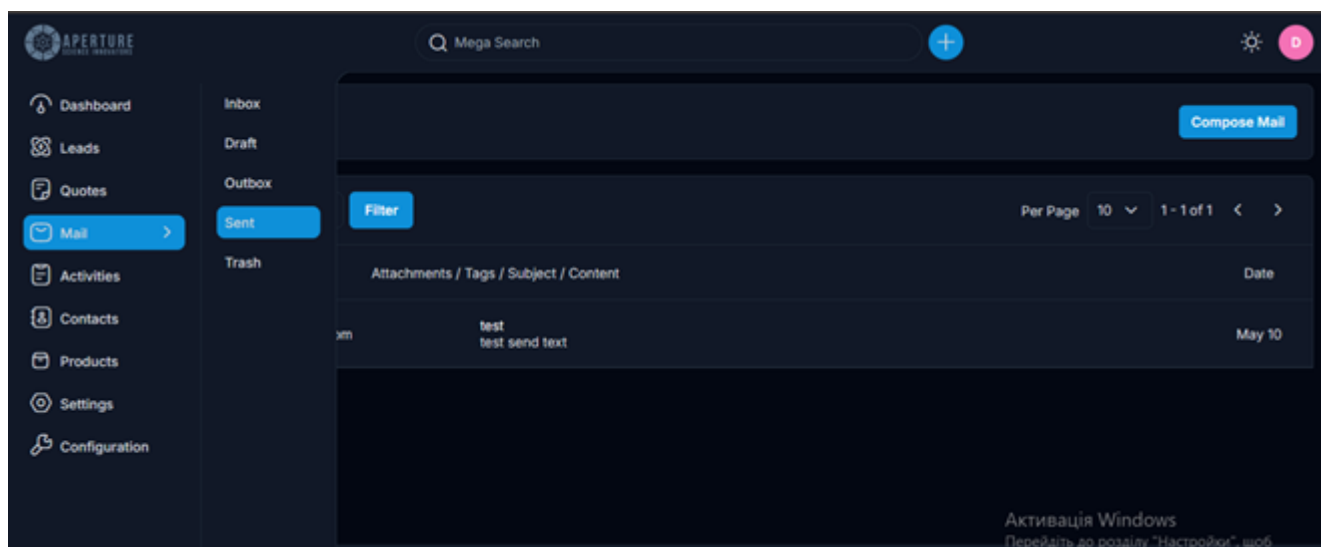


Рисунок - 3.21 – Сторінка пошти

Джерело: розроблено автором самостійно

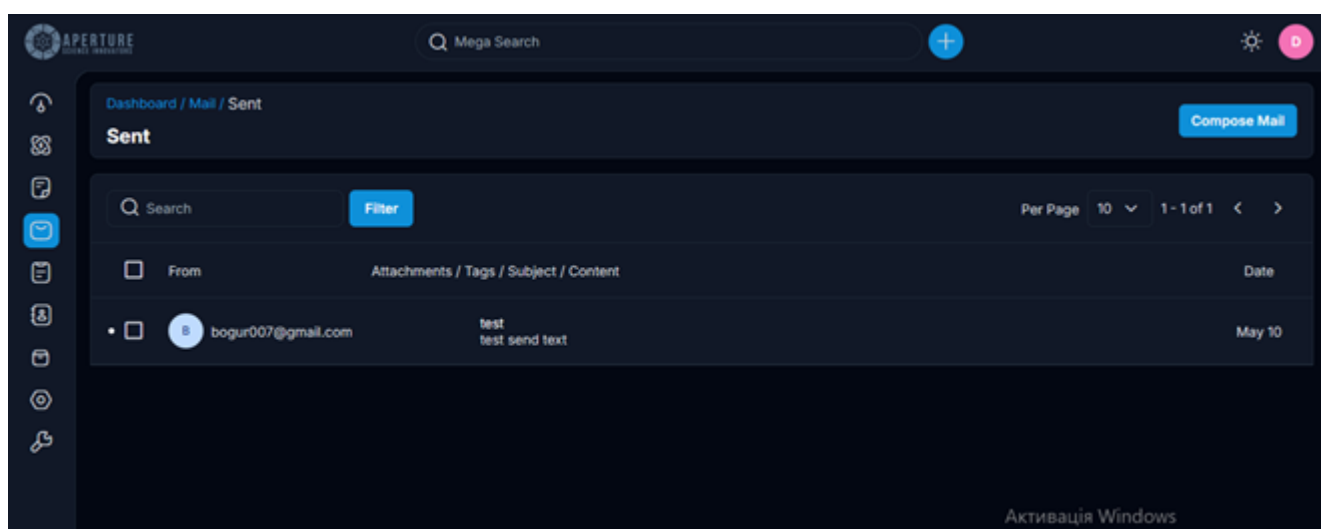


Рисунок - 3.22 – Сторінка управління електронною поштою

Джерело: розроблено автором самостійно

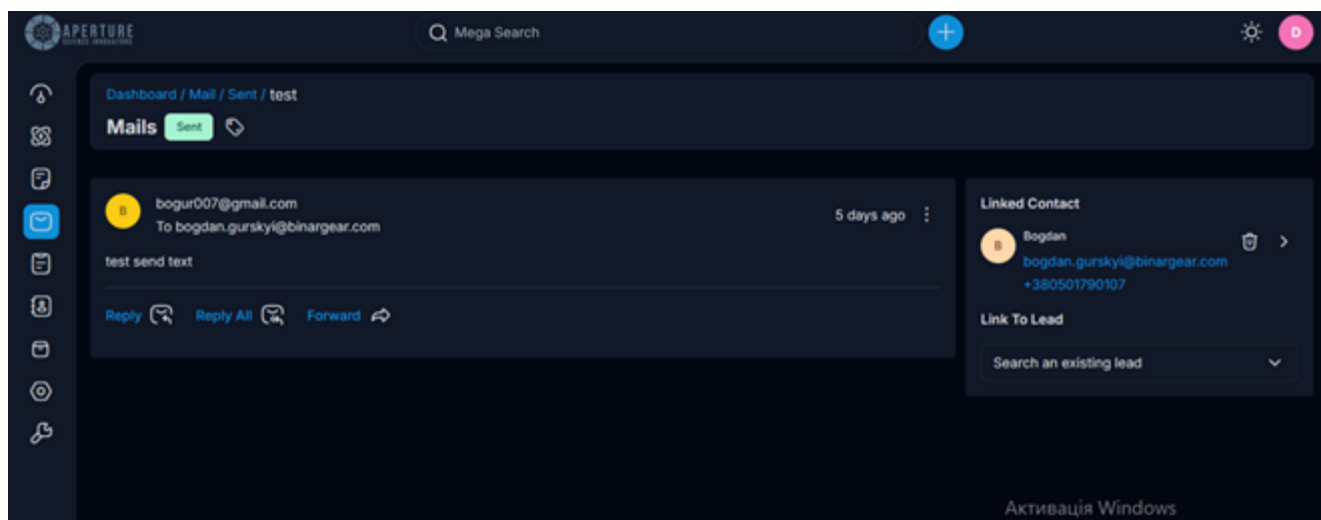


Рисунок - 3.23 – Сторінка управління електронною поштою

Джерело: розроблено автором самостійно

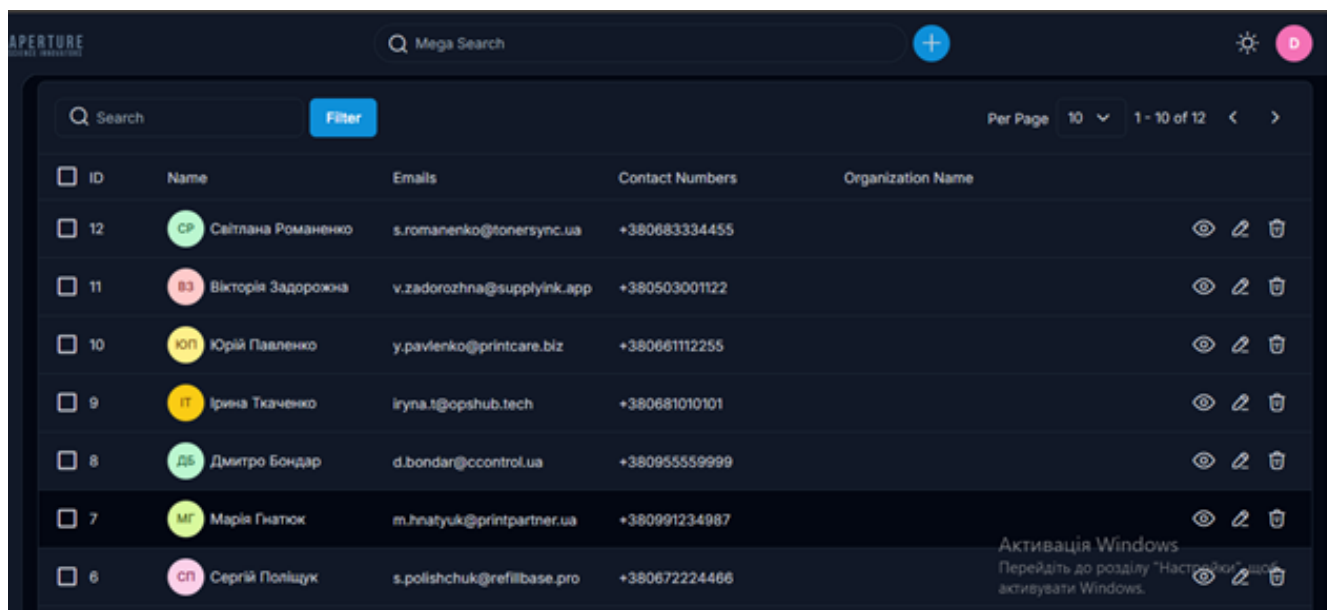


Рисунок - 3.24 – Сторінка інформації про контактних осіб.

Джерело: розроблено автором самостійно

На рисунку 3.20 зображено нову комерційну пропозицію. Містить поля для введення адреси доставки, вибору країни, додавання товарів/послуг та перегляду підсумкової таблиці. "Shipping Address" вводить адресу доставки. "Select Country" вибирає країну доставки. "Quote Items" додає товари або послуги. Таблиця з позиціями відображає інформацію про кожну позицію. "Add Product Request for this quote" додає новий товар/послугу. Призначення сторінки це створення комерційної пропозиції.

Розділ "Пошта" відображає вхідні повідомлення. Містить панель навігації, список листів, фільтри та елементи керування. Панель навігації переходить між модулями системи. Підрозділи "Пошти" організовують електронні листи. Кнопка "Compose Mail" створює новий лист. Фільтри сортують листи. Список листів відображає інформацію про кожне повідомлення. Пагінація перегортає сторінки зі списком листів. Призначення розділу - управління електронною поштою.

Розділ "Пошта" відображає надіслані повідомлення. Повторює елементи панелі навігації та загальні елементи розділу "Пошта". Відмінність: відображає відправлені листи. Призначення розділу - перегляд надісланих листів.

Сторінка відображає детальний вміст вибраного електронного листа. Містить інформацію про відправника, одержувача, тему, текст листа, панель дій та інформацію про пов'язаний контакт. Інформація про лист переглядає вміст листа. Панель дій відповідає на лист або пересилає його. "Linked Contact" надає інформацію про контактну особу. "Link To Lead" зв'язує лист з потенційним клієнтом. Призначення сторінки це, перегляд листа та виконання дій.

Розділ "Контакти" відображає список контактних осіб у таблиці. Містить інструменти для пошуку, фільтрації та управління контактами. Таблиця контактів відображає інформацію про кожну контактну особу. Пошук та фільтри знаходять та сортують контакти. Пагінація перегортає сторінки зі списком контактів. Кнопки дій управляють записами контактів. Призначення розділу це, управління інформацією про контактні особи.

ВИСНОВКИ

У рамках даної кваліфікаційної магістерської роботи було здійснено повний цикл розробки веб-платформи для управління процесами малого бізнесу. Метою роботи було створення комплексної, ефективної та зручної у використанні системи, яка б дозволила малому підприємству оптимізувати ключові бізнес-процеси, покращити організацію роботи та підвищити загальну ефективність.

На початковому етапі було проведено детальний аналіз предметної області та визначено основні потреби малого бізнесу в контексті управління. На основі цього було спроектовано інфологічну та логічну моделі бази даних, що стали фундаментом для зберігання та обробки інформації в системі. Було обґрунтовано вибір реляційної СКБД MySQL як надійного та масштабованого рішення.

У процесі проектування програмного забезпечення було обрано багаторівневу архітектуру, яка забезпечує чітке розділення відповідальності між клієнтською та серверною частинами. Для розробки серверної частини було використано потужний та гнучкий PHP-фреймворк Laravel, а для створення сучасного та реактивного користувацького інтерфейсу — JavaScript-фреймворк Vue.js. Застосування різноманітних додаткових бібліотек та інструментів дозволило розширити функціональність платформи та спростити процес розробки.

Основна частина роботи була присвячена реалізації ключових модулів платформи, включаючи управління лідами, клієнтами та контактами, комерційними пропозиціями, завданнями, календарем, документами, а також ведення звітності та аналітики. Залежно від потреб малого бізнесу, також було передбачено можливість інтеграції модулів обліку клієнтських замовлень, витратних матеріалів та ремонтних процесів.

Особливу увагу було приділено тестуванню розробленого програмного забезпечення на різних рівнях — від юніт-тестів окремих компонентів до інтеграційних та функціональних тестів основних бізнес-процесів. Це дозволило

виявити та усунути більшість потенційних помилок та забезпечити стабільність роботи системи.

Для забезпечення успішного впровадження та використання платформи було розроблено докладне керівництво користувача, яке містить інструкції з доступу, навігації та використання всіх основних модулів та функцій системи. Також було розглянуто питання технічного забезпечення, включаючи обґрунтування вибору середовища розгортання, оцінку ресурсних вимог та опис процесу розгортання та перевірки працездатності в умовах, наближених до реального бізнесу.

У результаті проведеної роботи було створено повноцінну веб-платформу для управління процесами малого бізнесу, яка об'єднує в собі необхідний набір інструментів для оптимізації ключових бізнес-процесів. Платформа є функціональною, протестованою та готовою до розгортання та використання в реальних умовах малого підприємства.

Розроблена система має потенціал значно підвищити ефективність роботи малого бізнесу за рахунок автоматизації рутинних завдань, покращення організації та комунікації, а також надання можливостей для аналізу даних та прийняття обґрунтованих управлінських рішень. Завдяки модульності архітектури, платформа може бути в подальшому розширена та адаптована до специфічних потреб різних малих підприємств.

Таким чином, поставлені в роботі завдання були успішно виконані, а розроблена веб-платформа є цінним інструментом для оптимізації управління в сфері малого бізнесу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Martin, R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. — Boston: Prentice Hall, 2017.
2. Freeman, E., Robson, E. Head First Design Patterns. — Sebastopol: O'Reilly Media, 2020.
3. Evans, E. Domain-Driven Design: Tackling Complexity in the Heart of Software. — Boston: Addison-Wesley, 2003.
4. Laravel Documentation. — URL: <https://laravel.com/docs>
5. Vue.js Documentation. — URL: <https://vuejs.org/guide/introduction.html>
6. TailwindCSS Documentation. — URL: <https://tailwindcss.com/docs>
7. Scrum Guide. — URL: <https://scrumguides.org/>
8. Beck, K. Test-Driven Development: By Example. — Boston: Addison-Wesley, 2002.
9. Kruchten, P. The Rational Unified Process: An Introduction. — Boston: Addison-Wesley, 2004.
10. Sommerville, I. Software Engineering. 10th ed. — Harlow: Pearson, 2015.
11. Cohn, M. Agile Estimating and Planning. — Upper Saddle River: Prentice Hall, 2005.
12. OpenAPI Specification. — URL: <https://swagger.io/specification/>
13. ISO/IEC 25010:2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE). — Geneva: ISO, 2011.
14. Gamma, E., Helm, R., Johnson, R., Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. — Boston: Addison-Wesley, 1994.
15. Knuth, D. E. The Art of Computer Programming. Vol. 1–3. — Boston: Addison-Wesley, 1997.
16. Pest PHP Testing Framework. — URL: <https://pestphp.com/>
17. PHPUnit Documentation. — URL: <https://phpunit.de/>
18. Paratest – Parallel testing for PHPUnit. — URL: <https://github.com/paratestphp/paratest>

- 19.Composer – Dependency Management for PHP. — URL:
<https://getcomposer.org/>
- 20.NPM – Node Package Manager. — npm, Inc., 2025. — URL:
<https://www.npmjs.com/>
- 21.NPM – Node Package Manager. — URL: <https://www.npmjs.com/>
- 22.Faker for PHP. — URL: <https://fakerphp.github.io/>
- 23.Vue Cal Documentation. — URL: <https://antoniandre.github.io/vue-cal/>
- 24.VeeValidate Documentation. — URL: <https://vee-validate.logaretm.com/v4/>
- 25.DomPDF for Laravel. — URL: <https://github.com/dompdf/dompdf>
- 26.Laravel Sanctum. — URL: <https://laravel.com/docs/sanctum>
- 27.Laravel Debugbar. — URL: <https://github.com/barryvdh/laravel-debugbar>
- 28.Monolog – Logging for PHP. — URL: <https://seldaek.github.io/monolog/>
- 29.Guzzle HTTP Client. — URL: <https://docs.guzzlephp.org/>
- 30.Chart.js Documentation. — URL: <https://www.chartjs.org/docs/>
- 31.Wieringa, R. Design Science Methodology for Information Systems and Software Engineering. — Springer, 2014.
- 32.Yourdon, E. Modern Structured Analysis. — Yourdon Press, 1989.
- 33.McConnell, S. Code Complete. — Microsoft Press, 2004.
- 34.UX Design: Nielsen Norman Group. — URL:
<https://www.nngroup.com/articles/definition-user-experience/>
- 35.Muntean, M., & Moldovan, D. Business Process Management Systems: Overview and Challenges.
- 36.OMG BPMN 2.0 Specification. — URL: <https://www.omg.org/spec/BPMN/2.0/>
- 37.ISO 9001:2015. Quality management systems – Requirements.
- 38.Enterprise Architecture TOGAF. — URL: <https://www.opengroup.org/togaf>
- 39.Odoo Documentation. — URL: <https://www.odoo.com/documentation>
- 40.Business Process Automation (BPA): IBM. — URL:
<https://www.ibm.com/topics/business-process-automation>
- 41.ISO/IEC/IEEE 29148:2018. Software Life Cycle Processes — Requirements Engineering.
- 42.Pahl, C., & Giesecke, S. An Ontology for Software Component Matching.

43. Yourdon, E. Decline and Fall of the American Programmer. — Prentice Hall, 1992.
44. Cockburn, A. Writing Effective Use Cases. — Addison-Wesley, 2001.
45. Business Model Canvas. Strategyzer. — URL:
<https://www.strategyzer.com/canvas/business-model-canvas>
46. RESTful API Design — Best Practices. — URL: <https://restfulapi.net/>
47. ISO/IEC 9126. Software Engineering — Product Quality.
48. W3C Web Accessibility Guidelines (WCAG). — URL:
<https://www.w3.org/WAI/standards-guidelines/wcag/>
49. BootstrapVue vs Tailwind — UI comparison. — URL: <https://bootstrap-vue.org/>
50. Laravel vs Symfony — PHP Frameworks Comparison. — URL:
<https://kinsta.com/blog/laravel-vs-symfony/>
51. Vue vs React: A Developer's Perspective. — URL:
<https://vuejs.org/v2/guide/comparison.html>
52. Open Source ERP Solutions for SMEs. — URL:
<https://opensource.com/resources/erp>
53. GitLab CI/CD Documentation. — URL: <https://docs.gitlab.com/ee/ci/>
54. DevOps and Agile: Microsoft Learn. — URL:
<https://learn.microsoft.com/en-us/devops/>
55. ISO/IEC 12207. Software Life Cycle Processes.
56. GitHub – Source Repositories. — URL: <https://github.com/>

ДОДАТКИ

Додаток А

Код створення бази даних Leads через міграції

return new class extends Migration

```
{  
    /**  
     * Run the migrations.  
     *  
     * @return void  
     */  
    public function up()  
    {  
        Schema::create('leads', function (Blueprint $table) {  
            $table->increments('id');  
            $table->string('title');  
            $table->text('description')->nullable();  
            $table->decimal('lead_value', 12, 4)->nullable();  
            $table->boolean('status')->nullable();  
            $table->text('lost_reason')->nullable();  
            $table->datetime('closed_at')->nullable();  
  
            $table->integer('user_id')->unsigned();  
            $table->foreign('user_id')->references('id')->on('users')->onDelete('cascade');  
  
            $table->integer('person_id')->unsigned();  
  
            $table->foreign('person_id')->references('id')->on('persons')->onDelete('cascade');  
  
            $table->integer('lead_source_id')->unsigned();
```

```
$table->foreign('lead_source_id')->references('id')->on('lead_sources')->onDelete('cascade');
```

```
    $table->integer('lead_type_id')->unsigned();
```

```
$table->foreign('lead_type_id')->references('id')->on('lead_types')->onDelete('cascade');
```

```
    $table->integer('lead_pipeline_id')->unsigned()->nullable();
```

```
$table->foreign('lead_pipeline_id')->references('id')->on('lead_pipelines')->onDelete('cascade');
```

```
    $table->integer('lead_stage_id')->unsigned();
```

```
$table->foreign('lead_stage_id')->references('id')->on('lead_stages')->onDelete('cascade');
```

```
    $table->timestamps();
```

```
});
```

```
}
```

```
/**
```

```
 * Reverse the migrations.
```

```
 *
```

```
 * @return void
```

```
 */
```

```
public function down()
```

```
{
```

```
    Schema::dropIfExists('leads');
```

```
}
```

```
};
```

Код контроллера для сторінки Leads на серверній частині

```
namespace DanCRM\Admin\Http\Controllers\Lead;

use Carbon\Carbon;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Resources\Json\AnonymousResourceCollection;
use Illuminate\Support\Facades\Event;
use Illuminate\View\View;
use Prettus\Repository\Criteria\RequestCriteria;
use DanCRM\Admin\DataGrids\Lead\LeadDataGrid;
use DanCRM\Admin\Http\Controllers\Controller;
use DanCRM\Admin\Http\Requests\LeadForm;
use DanCRM\Admin\Http\Requests\MassDestroyRequest;
use DanCRM\Admin\Http\Requests\MassUpdateRequest;
use DanCRM\Admin\Http\Resources\LeadResource;
use DanCRM\Admin\Http\Resources\StageResource;
use DanCRM\Attribute\Repositories\AttributeRepository;
use DanCRM>Contact\Repositories\PersonRepository;
use DanCRM\Lead\Repositories\LeadRepository;
use DanCRM\Lead\Repositories\PipelineRepository;
use DanCRM\Lead\Repositories\ProductRepository;
use DanCRM\Lead\Repositories\SourceRepository;
use DanCRM\Lead\Repositories\StageRepository;
use DanCRM\Lead\Repositories\TypeRepository;
use DanCRM\Tag\Repositories\TagRepository;
use DanCRM\User\Repositories\UserRepository;

class LeadController extends Controller
{
```

```
/**
 * Const variable for supported types.
 */
const SUPPORTED_TYPES = 'pdf,bmp,jpeg,jpg,png,webp';
```

```
/**
 * Create a new controller instance.
 *
 * @return void
 */
public function __construct(
    protected UserRepository $userRepository,
    protected AttributeRepository $attributeRepository,
    protected SourceRepository $sourceRepository,
    protected TypeRepository $typeRepository,
    protected PipelineRepository $pipelineRepository,
    protected StageRepository $stageRepository,
    protected LeadRepository $leadRepository,
    protected ProductRepository $productRepository,
    protected PersonRepository $personRepository
) {
    request()->request->add(['entity_type' => 'leads']);
}
```

```
/**
 * Display a listing of the resource.
 */
public function index(): View|JsonResponse
{
    if (request()->ajax()) {
        return datagrid(LeadDataGrid::class)->process();
    }
}
```

```

    }

    if (request('pipeline_id')) {
        $pipeline = $this->pipelineRepository->find(request('pipeline_id'));
    } else {
        $pipeline = $this->pipelineRepository->getDefaultPipeline();
    }

    return view('admin::leads.index', [
        'pipeline' => $pipeline,
        'columns' => $this->getKanbanColumns(),
    ]);
}

/**
 * Returns a listing of the resource.
 */
public function get(): JsonResponse
{
    if (request()->query('pipeline_id')) {
        $pipeline = $this->pipelineRepository->find(request()->query('pipeline_id'));
    } else {
        $pipeline = $this->pipelineRepository->getDefaultPipeline();
    }

    if ($stageId = request()->query('pipeline_stage_id')) {
        $stages = $pipeline->stages->where('id', request()->query('pipeline_stage_id'));
    } else {
        $stages = $pipeline->stages;
    }
}

```

```

foreach ($stages as $stage) {
    /**
     * We have to create a new instance of the lead repository every time, which is
     * why we're not using the injected one.
     */
    $query = app(LeadRepository::class)
        ->pushCriteria(app(RequestCriteria::class))
        ->where([
            'lead_pipeline_id' => $pipeline->id,
            'lead_pipeline_stage_id' => $stage->id,
        ]);

    if ($userIds = bouncer()->getAuthorizedUserIds()) {
        $query->whereIn('leads.user_id', $userIds);
    }

    $stage->lead_value = (clone $query)->sum('lead_value');

    $data[$stage->sort_order] = (new StageResource($stage))->jsonSerialize();

    $data[$stage->sort_order]['leads'] = [
        'data' => LeadResource::collection($paginator = $query->with([
            'tags',
            'type',
            'source',
            'user',
            'person',
            'person.organization',
            'pipeline',
            'pipeline.stages',
            'stage',

```

```

        'attribute_values',
    ]->paginate(10)),

    'meta' => [
        'current_page' => $paginator->currentPage(),
        'from'         => $paginator->firstItem(),
        'last_page'     => $paginator->lastPage(),
        'per_page'      => $paginator->perPage(),
        'to'            => $paginator->lastItem(),
        'total'         => $paginator->total(),
    ],
];
}

return response()->json($data);
}

/**
 * Show the form for creating a new resource.
 */
public function create(): View
{
    return view('admin::leads.create');
}

/**
 * Store a newly created resource in storage.
 */
public function store(LeadForm $request): RedirectResponse
{
    Event::dispatch('lead.create.before');

```

```

$data = $request->all();

$data['status'] = 1;

if (isset($data['lead_pipeline_stage_id'])) {
    $stage = $this->stageRepository->findOrFail($data['lead_pipeline_stage_id']);

    $data['lead_pipeline_id'] = $stage->lead_pipeline_id;
} else {
    $pipeline = $this->pipelineRepository->getDefaultPipeline();
    $stage = $pipeline->stages()->first();
    $data['lead_pipeline_id'] = $pipeline->id;
    $data['lead_pipeline_stage_id'] = $stage->id;
}

if (in_array($stage->code, ['won', 'lost'])) {
    $data['closed_at'] = Carbon::now();
}

$lead = $this->leadRepository->create($data);
Event::dispatch('lead.create.after', $lead);

session()->flash('success', trans('admin::app.leads.create-success'));

return redirect()->route('admin.leads.index', $data['lead_pipeline_id']);
}

/**
 * Show the form for editing the specified resource.
 */

```

```

public function edit(int $id): View
{
    $lead = $this->leadRepository->findOrFail($id);

    return view('admin::leads.edit', compact('lead'));
}

/**
 * Display a resource.
 */
public function view(int $id): View|RedirectResponse
{
    $lead = $this->leadRepository->findOrFail($id);
    $userIds = bouncer()->getAuthorizedUserIds();

    if ($userIds && ! in_array($lead->user_id, $userIds)) {
        return redirect()->route('admin.leads.index');
    }

    return view('admin::leads.view', compact('lead'));
}

/**
 * Update the specified resource in storage.
 */
public function update(LeadForm $request, int $id): RedirectResponse|JsonResponse
{
    Event::dispatch('lead.update.before', $id);

    $data = $request->all();

```

```

if (isset($data['lead_pipeline_stage_id'])) {
    $stage = $this->stageRepository->findOrFail($data['lead_pipeline_stage_id']);
    $data['lead_pipeline_id'] = $stage->lead_pipeline_id;
} else {
    $pipeline = $this->pipelineRepository->getDefaultPipeline();
    $stage = $pipeline->stages()->first();
    $data['lead_pipeline_id'] = $pipeline->id;
    $data['lead_pipeline_stage_id'] = $stage->id;
}

$lead = $this->leadRepository->update($data, $id);
Event::dispatch('lead.update.after', $lead);

if (request()->ajax()) {
    return response()->json([
        'message' => trans('admin::app.leads.update-success'),
    ]);
}

session()->flash('success', trans('admin::app.leads.update-success'));

if (request()->has('closed_at')) {
    return redirect()->back();
} else {
    return redirect()->route('admin.leads.index', $data['lead_pipeline_id']);
}
}

/**
 * Update the lead attributes.
 */

```

```

public function updateAttributes(int $id): JsonResponse
{
    $data = request()->all();

    $attributes = $this->attributeRepository->findWhere([
        'entity_type' => 'leads',
        ['code', 'NOTIN', ['title', 'description']],
    ]);

    Event::dispatch('lead.update.before', $id);

    $lead = $this->leadRepository->update($data, $id, $attributes);

    Event::dispatch('lead.update.after', $lead);

    return response()->json([
        'message' => trans('admin::app.leads.update-success'),
    ]);
}

```

```

/**

```

```

 * Update the lead stage.

```

```

 */

```

```

public function updateStage(int $id): JsonResponse
{
    $this->validate(request(), [
        'lead_pipeline_stage_id' => 'required',
    ]);

    $lead = $this->leadRepository->findOrFail($id);
    $stage = $lead->pipeline->stages()

```

```
->where('id', request()->input('lead_pipeline_stage_id'))
->firstOrFail();
```

```
Event::dispatch('lead.update.before', $id);
```

```
$lead = $this->leadRepository->update(
    [
        'entity_type'      => 'leads',
        'lead_pipeline_stage_id' => $stage->id,
    ],
    $id,
    ['lead_pipeline_stage_id']
);
```

```
Event::dispatch('lead.update.after', $lead);
```

```
return response()->json([
    'message' => trans('admin::app.leads.update-success'),
]);
}
```

```
/**
```

```
 * Search person results.
```

```
 */
```

```
public function search(): AnonymousResourceCollection
```

```
{
```

```
    if ($userIds = bouncer()->getAuthorizedUserIds()) {
```

```
        $results = $this->leadRepository
```

```
            ->pushCriteria(app(RequestCriteria::class))
```

```
            ->findWhereIn('user_id', $userIds);
```

```
    } else {
```

```

        $results = $this->leadRepository
            ->pushCriteria(app(RequestCriteria::class))
            ->all();
    }

    return LeadResource::collection($results);
}

/**
 * Remove the specified resource from storage.
 */
public function destroy(int $id): JsonResponse
{
    $this->leadRepository->findOrFail($id);

    try {
        Event::dispatch('lead.delete.before', $id);

        $this->leadRepository->delete($id);

        Event::dispatch('lead.delete.after', $id);

        return response()->json([
            'message' => trans('admin::app.leads.destroy-success'),
        ]);
    } catch (\Exception $exception) {
        return response()->json([
            'message' => trans('admin::app.leads.destroy-failed'),
        ], 400);
    }
}

```

```

/**
 * Mass update the specified resources.
 */

    public function massUpdate(MassUpdateRequest $massUpdateRequest):
JsonResponse
    {
        $leads = $this->leadRepository->findWhereIn('id',
$massUpdateRequest->input('indices'));

        try {
            foreach ($leads as $lead) {
                Event::dispatch('lead.update.before', $lead->id);

                $lead = $this->leadRepository->find($lead->id);

                $lead?->update(['lead_pipeline_stage_id' =>
$massUpdateRequest->input('value')]);

                Event::dispatch('lead.update.before', $lead->id);
            }

            return response()->json([
                'message' => trans('admin::app.leads.update-success'),
            ]);
        } catch (\Exception $th) {
            return response()->json([
                'message' => trans('admin::app.leads.update-failed'),
            ], 400);
        }
    }
}

```

```

/**
 * Mass delete the specified resources.
 */

    public function massDestroy(MassDestroyRequest $massDestroyRequest):
JsonResponse
    {
        $leads = $this->leadRepository->findWhereIn('id',
$massDestroyRequest->input('indices'));

        try {
            foreach ($leads as $lead) {
                Event::dispatch('lead.delete.before', $lead->id);

                $this->leadRepository->delete($lead->id);

                Event::dispatch('lead.delete.after', $lead->id);
            }

            return response()->json([
                'message' => trans('admin::app.leads.destroy-success'),
            ]);
        } catch (\Exception $exception) {
            return response()->json([
                'message' => trans('admin::app.leads.destroy-failed'),
            ]);
        }
    }

/**
 * Attach product to lead.

```

```
*/
```

```
public function addProduct(int $leadId): JsonResponse
```

```
{
```

```
    $product = $this->productRepository->updateOrCreate([
```

```
        [
```

```
            'lead_id' => $leadId,
```

```
            'product_id' => request()->input('product_id'),
```

```
        ],
```

```
        array_merge([
```

```
            request()->all(),
```

```
            [
```

```
                'lead_id' => $leadId,
```

```
                'amount' => request()->input('price') * request()->input('quantity'),
```

```
            ],
```

```
        ])
```

```
    );
```

```
    return response()->json([
```

```
        'data' => $product,
```

```
        'message' => trans('admin::app.leads.update-success'),
```

```
    ]);
```

```
}
```

```
/**
```

```
 * Remove product attached to lead.
```

```
*/
```

```
public function removeProduct(int $id): JsonResponse
```

```
{
```

```
    try {
```

```
        Event::dispatch('lead.product.delete.before', $id);
```

```

$this->productRepository->deleteWhere([
    'lead_id' => $id,
    'product_id' => request()->input('product_id'),
]);

Event::dispatch('lead.product.delete.after', $id);

return response()->json([
    'message' => trans('admin::app.leads.destroy-success'),
]);
} catch (\Exception $exception) {
    return response()->json([
        'message' => trans('admin::app.leads.destroy-failed'),
    ]);
}
}

/**
 * Kanban lookup.
 */
public function kanbanLookup()
{
    $params = $this->validate(request(), [
        'column' => ['required'],
        'search' => ['required', 'min:2'],
    ]);

    /**
     * Finding the first column from the collection.
     */

```

```
        $column = collect($this->getKanbanColumns()->where('index',
$params['column'])->firstOrFail());
```

```
/**
```

```
 * Fetching on the basis of column options.
```

```
*/
```

```
return app($column['filterable_options']['repository'])
        ->select([$column['filterable_options']['column']['label'].' as label',
$column['filterable_options']['column']['value'].' as value'])
        ->where($column['filterable_options']['column']['label'], 'LIKE',
'%'.$params['search'].'%')
        ->get()
        ->map
        ->only('label', 'value');
}
```

```
/**
```

```
 * Get columns for the kanban view.
```

```
*/
```

```
private function getKanbanColumns(): array
```

```
{
```

```
    return [
```

```
        [
```

```
            'index'          => 'id',
```

```
            'label'         => trans('admin::app.leads.index.kanban.columns.id'),
```

```
            'type'          => 'integer',
```

```
            'searchable'    => false,
```

```
            'search_field'  => 'in',
```

```
            'filterable'    => true,
```

```
            'filterable_type' => null,
```

```
            'filterable_options' => [],
```

```

        'allow_multiple_values' => true,
        'sortable'              => true,
        'visibility'            => true,
    ],
    [
        'index'                  => 'lead_value',
                                'label' =>
trans('admin::app.leads.index.kanban.columns.lead-value'),
        'type'                   => 'string',
        'searchable'             => false,
        'search_field'           => 'in',
        'filterable'             => true,
        'filterable_type'        => null,
        'filterable_options'     => [],
        'allow_multiple_values' => true,
        'sortable'               => true,
        'visibility'             => true,
    ],
    [
        'index'                  => 'user_id',
                                'label' =>
trans('admin::app.leads.index.kanban.columns.sales-person'),
        'type'                   => 'string',
        'searchable'             => false,
        'search_field'           => 'in',
        'filterable'             => true,
        'filterable_type'        => 'searchable_dropdown',
        'filterable_options'     => [
            'repository' => UserRepository::class,
            'column'     => [
                'label' => 'name',

```

```

        'value' => 'id',
    ],
],
'allow_multiple_values' => true,
'sortable'             => true,
'visibility'           => true,
],
[
    'index'             => 'person.id',
                        'label' =>
trans('admin::app.leads.index.kanban.columns.contact-person'),
    'type'              => 'string',
    'searchable'        => false,
    'search_field'      => 'in',
    'filterable'        => true,
    'filterable_options' => [],
    'allow_multiple_values' => true,
    'sortable'          => true,
    'visibility'        => true,
    'filterable_type'   => 'searchable_dropdown',
    'filterable_options' => [
        'repository' => PersonRepository::class,
        'column'     => [
            'label' => 'name',
            'value' => 'id',
        ],
    ],
],
[
    'index'             => 'lead_type_id',
    'label'             => trans('admin::app.leads.index.kanban.columns.lead-type'),

```

```

        'type'            => 'string',
        'searchable'      => false,
        'search_field'    => 'in',
        'filterable'      => true,
        'filterable_type' => 'dropdown',
        'filterable_options' => $this->typeRepository->all(['name as label', 'id as
value']->toArray(),
        'allow_multiple_values' => true,
        'sortable'        => true,
        'visibility'      => true,
    ],
    [
        'index'            => 'lead_source_id',
        'label'            => trans('admin::app.leads.index.kanban.columns.source'),
        'type'            => 'string',
        'searchable'      => false,
        'search_field'    => 'in',
        'filterable'      => true,
        'filterable_type' => 'dropdown',
        'filterable_options' => $this->sourceRepository->all(['name as label', 'id as
value']->toArray(),
        'allow_multiple_values' => true,
        'sortable'        => true,
        'visibility'      => true,
    ],
    [
        'index'            => 'tags.name',
        'label'            => trans('admin::app.leads.index.kanban.columns.tags'),
        'type'            => 'string',
        'searchable'      => false,
        'search_field'    => 'in',

```

```

        'filterable'          => true,
        'filterable_options' => [],
        'allow_multiple_values' => true,
        'sortable'           => true,
        'visibility'         => true,
        'filterable_type'    => 'searchable_dropdown',
        'filterable_options' => [
            'repository' => TagRepository::class,
            'column'     => [
                'label' => 'name',
                'value' => 'name',
            ],
        ],
    ],
];
}

```

```
/**
```

```
 * Create multiple leads.
```

```
*/
```

```
private function createLeads(array $rawLeads): array
```

```
{
```

```
    $leads = [];
```

```
    foreach ($rawLeads as $rawLead) {
```

```
        Event::dispatch('lead.create.before');
```

```
        foreach ($rawLead['person']['emails'] as $email) {
```

```
            $person = $this->personRepository
```

```
                ->whereJsonContains('emails', [['value' => $email['value']]])
```

```
                ->first();
```

```
        if ($person) {
            $rawLead['person']['id'] = $person->id;

            break;
        }
    }

    $pipeline = $this->pipelineRepository->getDefaultPipeline();

    $stage = $pipeline->stages()->first();

    $lead = $this->leadRepository->create(array_merge($rawLead, [
        'lead_pipeline_id'    => $pipeline->id,
        'lead_pipeline_stage_id' => $stage->id,
    ]));

    Event::dispatch('lead.create.after', $lead);

    $leads[] = $lead;
}

return $leads;
}
```

Код форми для сторінки Leads на серверній частині

```
namespace DanCRM\Admin\Http\Requests;

use Carbon\Carbon;

use Illuminate\Foundation\Http\FormRequest;

use DanCRM\Attribute\Repositories\AttributeRepository;

use DanCRM\Attribute\Repositories\AttributeValueRepository;

use DanCRM\Core\Contracts\Validations\Decimal;

class LeadForm extends FormRequest
{
    /**
     * @var array
     */

    protected $rules = [];

    /**
     * Create a new form request instance.
     *
     * @return void
     */
}
```

```
*/
```

```
public function __construct(
```

```
protected AttributeRepository $attributeRepository,
```

```
protected AttributeValueRepository $attributeValueRepository
```

```
) {}
```

```
/**
```

```
 * Determine if the product is authorized to make this request.
```

```
 *
```

```
 * @return bool
```

```
*/
```

```
public function authorize()
```

```
{
```

```
    return true;
```

```
}
```

```
/**
```

```
 * Get the validation rules that apply to the request.
```

```
 *
```

```
 * @return array
```

```
*/
```

```

public function rules()

{

    foreach (['leads', 'persons'] as $key => $entityType) {

        $attributes = $this->attributeRepository->scopeQuery(function ($query) use
($entityType) {

            $attributeCodes = $entityType == 'persons'

            ? array_keys(request('person') ?? [])

            : array_keys(request()->all());

            $query = $query->whereIn('code', $attributeCodes)

            ->where('entity_type', $entityType);

            if (request()->has('quick_add')) {

                $query = $query->where('quick_add', 1);

            }

            return $query;

        }->get());

        foreach ($attributes as $attribute) {

            if ($entityType == 'persons') {

```

```
$attribute->code = 'person.'.$attribute->code;

}
```

```
$validations = [];
```

```
if ($attribute->type == 'boolean') {

continue;

} elseif ($attribute->type == 'address') {

if (! $attribute->is_required) {

    continue;

}

}
```

```
$validations = [

    $attribute->code.'.address' => 'required',

    $attribute->code.'.country' => 'required',

    $attribute->code.'.state'    => 'required',

    $attribute->code.'.city'     => 'required',

    $attribute->code.'.postcode' => 'required',

];

} elseif ($attribute->type == 'email') {

$validations = [
```

```

        $attribute->code          => [$attribute->is_required ? 'required' :
'nullable'],

        $attribute->code.'.*.value' => [$attribute->is_required ? 'required' :
'nullable', 'email'],

        $attribute->code.'.*.label' => $attribute->is_required ? 'required' :
'nullable',

    ];

    } elseif ($attribute->type == 'phone') {

        $validations = [

            $attribute->code          => [$attribute->is_required ? 'required' :
'nullable'],

            $attribute->code.'.*.value' => [$attribute->is_required ? 'required' :
'nullable'],

            $attribute->code.'.*.label' => $attribute->is_required ? 'required' :
'nullable',

        ];

    } else {

        $validations[$attribute->code] = [$attribute->is_required ? 'required' :
'nullable'];

        if ($attribute->type == 'text' && $attribute->validation) {

            array_push($validations[$attribute->code],

                $attribute->validation == 'decimal'

```

```

        ? new Decimal

        : $attribute->validation

    );

}

if ($attribute->type == 'price') {

    array_push($validations[$attribute->code], new Decimal);

}

}

if ($attribute->is_unique) {

    array_push($validations[in_array($attribute->type, ['email', 'phone'])

        ? $attribute->code.'.*.value'

        : $attribute->code

    ], function ($field, $value, $fail) use ($attribute, $entityType) {

        if (! $this->attributeValueRepository->isValueUnique(

            $entityType == 'persons' ? request('person.id') : $this->id,

            $attribute->entity_type,

            $attribute,

            request($field)

        )

```

```

    ) {

        $fail('The value has already been taken.');
```

}

```

    });

}

$this->rules = array_merge($this->rules, $validations);

}

}

$this->rules['expected_close_date'] = [

    'date_format:Y-m-d',

    'after:'.Carbon::yesterday()->format('Y-m-d'),

];

return [

    ...$this->rules,

    'products' => 'array',

    'products.*.product_id' => 'sometimes|required|exists:products,id',

    'products.*.name' => 'required_with:products.*.product_id',

    'products.*.price' => 'required_with:products.*.product_id',

```

```

        'products.*.quantity' => 'required_with:products.*.product_id',

    ];

}

/**

 * Get the validation messages that apply to the request.

 */

public function messages(): array

{

    return [

        'products.*.product_id.exists'      =>
        trans('admin::app.leads.selected-product-not-exist'),

        'products.*.name.required_with'      =>
        trans('admin::app.leads.product-name-required'),

        'products.*.price.required_with'     =>
        trans('admin::app.leads.product-price-required'),

        'products.*.quantity.required_with'  =>
        trans('admin::app.leads.product-quantity-required'),

    ];

}

}

```

Код сторінки Leads на клієнтській частині

```
<x-admin::layouts>
  <x-slot:title>
    @lang('admin::app.leads.index.title')
  </x-slot>

  <!-- Header -->
  {!! view_render_event('admin.leads.index.header.before') !!}

  <div class="flex items-center justify-between rounded-lg border border-gray-200
bg-white px-4 py-2 text-sm dark:border-gray-800 dark:bg-gray-900
dark:text-gray-300">
    {!! view_render_event('admin.leads.index.header.left.before') !!}

    <div class="flex flex-col gap-2">
      <!-- Breadcrumb's -->
      <x-admin::breadcrumbs name="leads" />

      <div class="text-xl font-bold dark:text-white">
        @lang('admin::app.leads.index.title')
      </div>
    </div>

    {!! view_render_event('admin.leads.index.header.left.after') !!}

    {!! view_render_event('admin.leads.index.header.right.before') !!}

    <div class="flex items-center gap-x-2.5">
      <!-- Upload File for Lead Creation -->
      @if(core()->getConfigData('general.magic_ai.pdf_generation.enabled'))
```

```
@include('admin::leads.index.upload')
@endif
```

```
@if ((request()->view_type ?? "kanban") == "table")
    <!-- Export Modal -->
    <x-admin::datagrid.export :src="route('admin.leads.index')" />
@endif
```

```
<!-- Create button for Leads -->
<div class="flex items-center gap-x-2.5">
    @if (bouncer()->hasPermission('leads.create'))
        <a
            href="{{ route('admin.leads.create') }}"
            class="primary-button"
        >
            @lang('admin::app.leads.index.create-btn')
        </a>
    @endif
</div>
</div>
```

```
{!! view_render_event('admin.leads.index.header.right.after') !!}
</div>
```

```
{!! view_render_event('admin.leads.index.header.after') !!}
```

```
{!! view_render_event('admin.leads.index.content.before') !!}
```

```
<!-- Content -->

```

```

[&>*>*>*.toolbarRight]:max-md:gap-y-2 [amp;*>*>*.toolbarRight]:max-md:flex-wrap
mt-3.5 [amp;*>*>*.nth-child(1)]:max-lg:!flex-wrap">
    @if ((request()->view_type ?? "kanban") == "table")
        @include('admin::leads.index.table')
    @else
        @include('admin::leads.index.kanban')
    @endif
</div>

{!! view_render_event('admin.leads.index.content.after') !!}
</x-admin::layouts>

```

Kanban list:

```

{!! view_render_event('admin.leads.index.kanban.before') !!}

<!-- Kanban Vue Component -->
<v-leads-kanban ref="leadsKanban">
    <div class="flex flex-col gap-4">
        <!-- Shimmer -->
        <x-admin::shimmer.leads.index.kanban />
    </div>
</v-leads-kanban>

{!! view_render_event('admin.leads.index.kanban.after') !!}

@pushOnce('scripts')
    <script
        type="text/x-template"
        id="v-leads-kanban-template"
    >

```

```

<template v-if="isLoading">
  <div class="flex flex-col gap-4">
    <x-admin::shimmer.leads.index.kanban />
  </div>
</template>

<template v-else>
  <div class="flex flex-col gap-4">
    @include('admin::leads.index.kanban.toolbar')

    {!! view_render_event('admin.leads.index.kanban.content.before') !!}

    <div class="flex gap-2.5 overflow-x-auto">
      <!-- Stage Cards -->
      <div
        class="flex min-w-[275px] max-w-[275px] flex-col gap-1 rounded-lg
border border-gray-200 bg-white dark:border-gray-800 dark:bg-gray-900"
        v-for="(stage, index) in stageLeads"
      >
        {!!
view_render_event('admin.leads.index.kanban.content.stage.header.before') !!}

        <!-- Stage Header -->
        <div class="flex flex-col px-2 py-3">
          <!-- Stage Title and Action -->
          <div class="flex items-center justify-between">
            <span class="text-xs font-medium dark:text-white">
              @{{ stage.name }} (@{{ stage.leads.meta.total }})
            </span>

            @if (bouncer()->hasPermission('leads.create'))

```

```

<a
    :href="{{ route('admin.leads.create') }}" + '?stage_id=' +
stage.id"
    class="icon-add cursor-pointer rounded p-1 text-lg
text-gray-600 transition-all hover:bg-gray-200 hover:text-gray-800 dark:text-gray-600
dark:hover:bg-gray-800 dark:hover:text-white"
    target="_blank"
>
</a>
@endif
</div>

```

```

<!-- Stage Total Leads and Amount -->
<div class="flex items-center justify-between gap-2">
    <span class="text-xs font-medium dark:text-white">
        @{{ $admin.formatPrice(stage.lead_value) }}
    </span>

    <!-- Progress Bar -->
    <div class="h-1 w-36 overflow-hidden rounded-full bg-gray-200
dark:bg-gray-800">
        <div
            class="h-1 bg-green-500"
            :style="{ width: (stage.lead_value / totalStagesAmount) * 100
+ '%' }"
        ></div>
    </div>
</div>
</div>

```

```

                                                                    {!!
view_render_event('admin.leads.index.kanban.content.stage.header.after') !!}

                                                                    {!!
view_render_event('admin.leads.index.kanban.content.stage.body.before') !!}

<!-- Draggable Stage Lead Cards -->
<draggable
    class="flex h-[calc(100vh-317px)] flex-col gap-2 overflow-y-auto
p-2"
    :class="{ 'justify-center': stage.leads.data.length === 0 }"
    ghost-class="draggable-ghost"
    handle=".lead-item"
    v-bind="{ animation: 200 }"
    :list="stage.leads.data"
    item-key="id"
    group="leads"
    @scroll="handleScroll(stage, $event)"
    @change="updateStage(stage, $event)"
>
<template #header>
    <div
        class="flex flex-col items-center justify-center"
        v-if="! stage.leads.data.length"
    >
        

```

```

<div class="flex flex-col items-center gap-4">
  <div class="flex flex-col items-center gap-2">
    <p class="!text-base font-semibold dark:text-white">
      @lang('admin::app.leads.index.kanban.empty-list')
    </p>

    <p class="!text-sm text-gray-400 dark:text-gray-400">

@lang('admin::app.leads.index.kanban.empty-list-description')
    </p>
  </div>

  @if (bouncer()->hasPermission('leads.create'))
    <a
      :href="{{ route('admin.leads.create') }}" + '?stage_id=' +
stage.id"
      class="secondary-button"
    >
      @lang('admin::app.leads.index.kanban.create-lead-btn')
    </a>
  @endif
</div>
</div>
</template>

<!-- Lead Card -->
<template #item="{ element, index }">

{!!
view_render_event('admin.leads.index.kanban.content.stage.body.card.before') !!}

<a

```

```

class="lead-item flex cursor-pointer flex-col gap-5 rounded-md
border border-gray-100 bg-gray-50 p-2 dark:border-gray-400 dark:bg-gray-400"
:href="{{ route('admin.leads.view', 'replaceId')
}}'.replace('replaceId', element.id)"
>
{!!
view_render_event('admin.leads.index.kanban.content.stage.body.card.header.before')
!!}

```

```

<!-- Header -->
<div class="flex items-start justify-between">
  <div class="flex items-center gap-1">
    <x-admin::avatar ::name="element.person.name" />

    <div class="flex flex-col gap-0.5">
      <span class="text-xs font-medium">
        @{{ element.person.name }}
      </span>

      <span class="text-[10px] leading-normal">
        @{{ element.person.organization?.name }}
      </span>
    </div>
  </div>

  <div
    class="group relative"
    v-if="element.rotten_days > 0"
  >
    <span class="icon-rotten cursor-default text-xl
text-rose-600"></span>

```

```
<div class="absolute -top-1 right-7 hidden w-max flex-col
items-center group-hover:flex">
```

```
<span class="whitespace-no-wrap relative rounded-md
bg-black px-4 py-2 text-xs leading-none text-white shadow-lg">
```

```
@{{
"@lang('admin::app.leads.index.kanban.rotten-days', ['days'
'replaceDays'])".replace('replaceDays', element.rotten_days) }}
</span>
```

```
<div class="absolute -right-1 top-2 h-3 w-3 rotate-45
bg-black"></div>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
{!!
view_render_event('admin.leads.index.kanban.content.stage.body.card.header.after') !!}
```

```
{!!
view_render_event('admin.leads.index.kanban.content.stage.body.card.title.before') !!}
```

```
<!-- Lead Title -->
```

```
<p class="text-xs font-medium">
```

```
@{{ element.title }}
```

```
</p>
```

```
{!!
view_render_event('admin.leads.index.kanban.content.stage.body.card.title.after') !!}
```

```
<div class="flex flex-wrap gap-1">
```

```

<div
    class="flex items-center gap-1 rounded-xl bg-gray-200 px-2
py-1 text-xs font-medium dark:bg-gray-800 dark:text-white"
    v-if="element.user"
>
    <span class="icon-settings-user text-sm"></span>

    @{{ element.user.name }}
</div>

```

```

    <div class="rounded-xl bg-gray-200 px-2 py-1 text-xs
font-medium dark:bg-gray-800 dark:text-white">
        @{{ element.formatted_lead_value }}
    </div>

```

```

    <div class="rounded-xl bg-gray-200 px-2 py-1 text-xs
font-medium dark:bg-gray-800 dark:text-white">
        @{{ element.source.name }}
    </div>

```

```

    <div class="rounded-xl bg-gray-200 px-2 py-1 text-xs
font-medium dark:bg-gray-800 dark:text-white">
        @{{ element.type.name }}
    </div>

```

```

<!-- Tags -->
<template v-for="tag in element.tags">
    {!!
view_render_event('admin.leads.index.kanban.content.stage.body.card.tag.before') !!}

```

```

<div

```

```

class="rounded-xl bg-gray-200 px-2 py-1 text-xs
font-medium dark:bg-gray-800"
      :style="{
        backgroundColor: tag.color,
        color: tag.textColor[tag.color]
      }"
    >
      @{{ tag.name }}
    </div>

```

```

                                                                    {!!
view_render_event('admin.leads.index.kanban.content.stage.body.card.tag.after') !!}
      </template>
    </div>
  </a>

```

```

                                                                    {!!
view_render_event('admin.leads.index.kanban.content.stage.body.card.after') !!}
      </template>
    </draggable>

```

```

                                                                    {!!
view_render_event('admin.leads.index.kanban.content.stage.body.after') !!}
      </div>
    </div>

    {!! view_render_event('admin.leads.index.kanban.content.after') !!}
  </div>
</template>
</script>

```

```
<script type="module">
  app.component('v-leads-kanban', {
    template: '#v-leads-kanban-template',

    data() {
      return {
        available: {
          columns: @json($columns),
        },

        applied: {
          filters: {
            columns: [],
          }
        },

        stages: @json($pipeline->stages->toArray()),

        stageLeads: {},

        isLoading: true,

        tagTextColor: {
          '#FEE2E2': '#DC2626',
          '#FFEDD5': '#EA580C',
          '#FEF3C7': '#D97706',
          '#FEF9C3': '#CA8A04',
          '#ECFCCB': '#65A30D',
          '#DCFCE7': '#16A34A',
        },
      };
    }
  });
</script>
```

```
},
```

```
computed: {  
  totalStagesAmount() {  
    let totalAmount = 0;  
  
    for (let [key, stage] of Object.entries(this.stageLeads)) {  
      totalAmount += parseFloat(stage.lead_value);  
    }  
  
    return totalAmount;  
  }  
},
```

```
mounted () {  
  this.boot();  
},
```

```
methods: {  
  /**  
    * Initialization: This function checks for any previously saved filters in local  
storage and applies them as needed.  
    *  
    * @returns {void}  
    */  
  boot() {  
    let kanbans = this.getKanbans();  
  
    if (kanbans?.length) {  
      const currentKanban = kanbans.find(({ src }) => src === this.src);
```

```

    if (currentKanban) {
        this.applied.filters = currentKanban.applied.filters;

        this.get()
            .then(response => {
                for (let [sortOrder, data] of Object.entries(response.data)) {
                    this.stageLeads[sortOrder] = data;
                }
            });

        return;
    }
}

this.get()
    .then(response => {
        for (let [sortOrder, data] of Object.entries(response.data)) {
            this.stageLeads[sortOrder] = data;
        }
    });
},

/**
 * Fetches the leads based on the applied filters.
 *
 * @param {object} requestedParams - The requested parameters.
 * @returns {Promise} The promise object representing the request.
 */
get(requestedParams = {}) {
    let params = {
        search: "",

```

```

searchFields: ",
pipeline_id: "{{ request('pipeline_id') }}",
limit: 10,
};

```

```

this.applied.filters.columns.forEach((column) => {
  if (column.index === 'all') {
    if (! column.value.length) {
      return;
    }

```

```

    params['search'] += `title:${column.value.join(',')}`;
    params['searchFields'] += `title:like`;

```

```

    return;
  }

```

```

/**

```

* If the column is a searchable dropdown, then we need to append the column value

* with the column label. Otherwise, we can directly append the column value.

```

*/

```

```

    params['search'] += column.filterable_type === 'searchable_dropdown'
      ? `${column.index}:${column.value.map(option =>
option.value).join(',')}`
      : `${column.index}:${column.value.join(',')}`;

```

```

    params['searchFields'] += `${column.index}:${column.search_field}`;
  });

```

```

return this.$axios
  .get("{ { route('admin.leads.get') } }", {
    params: {
      ...params,

      ...requestedParams,
    }
  })
  .then(response => {
    this.isLoading = false;

    this.updateKanbans();

    return response;
  })
  .catch(error => {
    console.log(error)
  });
},

/**
 * Filters the leads based on the applied filters.
 *
 * @param {object} filters - The filters to be applied.
 * @returns {void}
 */
filter(filters) {
  this.applied.filters.columns = [
    ...(this.applied.filters.columns.filter((column) => column.index ===
'all')),
    ...filters.columns,

```

```
];
```

```
this.get()
```

```
.then(response => {
```

```
  for (let [sortOrder, data] of Object.entries(response.data)) {
```

```
    this.stageLeads[sortOrder] = data;
```

```
  }
```

```
});
```

```
},
```

```
/**
```

```
 * Searches the leads based on the applied filters.
```

```
 *
```

```
 * @param {object} filters - The filters to be applied.
```

```
 * @returns {void}
```

```
 */
```

```
search(filters) {
```

```
  this.applied.filters.columns = [
```

```
    ...(this.applied.filters.columns.filter((column) => column.index !==
```

```
'all')),
```

```
    ...filters.columns,
```

```
  ];
```

```
this.get()
```

```
.then(response => {
```

```
  for (let [sortOrder, data] of Object.entries(response.data)) {
```

```
    this.stageLeads[sortOrder] = data;
```

```
  }
```

```
});
```

```
},
```

```

/**
 * Appends the leads to the stage.
 *
 * @param {object} params - The parameters to be appended.
 * @returns {void}
 */
append(params) {
  this.get(params)
    .then(response => {
      for (let [sortOrder, data] of Object.entries(response.data)) {
        if (! this.stageLeads[sortOrder]) {
          this.stageLeads[sortOrder] = data;
        } else {
          this.stageLeads[sortOrder].leads.data =
this.stageLeads[sortOrder].leads.data.concat(data.leads.data);

          this.stageLeads[sortOrder].leads.meta = data.leads.meta;
        }
      }
    });
},

/**
 * Updates the stage with the latest lead data.
 *
 * @param {object} stage - The stage object.
 * @param {object} event - The event object.
 * @returns {void}
 */
updateStage(stage, event) {
  if (event.moved) {

```

```

        return;
    }

    if (event.removed) {
        stage.lead_value = parseFloat(stage.lead_value) -
parseFloat(event.removed.element.lead_value);

        this.stageLeads[stage.sort_order].leads.meta.total =
this.stageLeads[stage.sort_order].leads.meta.total - 1;

        return;
    }

    stage.lead_value = parseFloat(stage.lead_value) +
parseFloat(event.added.element.lead_value);

    this.stageLeads[stage.sort_order].leads.meta.total =
this.stageLeads[stage.sort_order].leads.meta.total + 1;

    this.$axios
        .put('{{ route('admin.leads.stage.update', 'replace') }}'.replace('replace',
event.added.element.id), {
            'lead_pipeline_stage_id': stage.id
        })
        .then(response => {
            this.$emitter.emit('add-flash', { type: 'success', message:
response.data.message });
        })
        .catch(error => {
            this.$emitter.emit('add-flash', { type: 'error', message:
error.response.data.message });
        });

```

```

        });
    },

    /**
     * Handles the scroll event on the stage leads.
     *
     * @param {object} stage - The stage object.
     * @param {object} event - The scroll event.
     * @returns {void}
     */
    handleScroll(stage, event) {
        const bottom = event.target.scrollHeight - event.target.scrollTop ===
event.target.clientHeight;

        if (! bottom) {
            return;
        }

        if (this.stageLeads[stage.sort_order].leads.meta.current_page ==
this.stageLeads[stage.sort_order].leads.meta.last_page) {
            return;
        }

        this.append({
            pipeline_stage_id: stage.id,
            pipeline_id: stage.lead_pipeline_id,
            page: this.stageLeads[stage.sort_order].leads.meta.current_page + 1,
            limit: 10,
        });
    },

```

```
//=====
=====

// Support for previous applied values in kanban's. All code is based on local
storage.
```

```
//=====
=====
```

```
/**
 * Updates the kanban's stored in local storage with the latest data.
 *
 * @returns {void}
 */
updateKanbans() {
  let kanbans = this.getKanbans();

  if (kanbans?.length) {
    const currentKanban = kanbans.find(({ src }) => src === this.src);

    if (currentKanban) {
      kanbans = kanbans.map(kanban => {
        if (kanban.src === this.src) {
          return {
            ...kanban,
            requestCount: ++kanban.requestCount,
            available: this.available,
            applied: this.applied,
          };
        }
      });
    }
  }
}
```

```

        return kanban;

    });

    } else {

        kanbans.push(this.getKanbanInitialProperties());

    }

    } else {

        kanbans = [this.getKanbanInitialProperties()];

    }

    this.setKanbans(kanbans);

},

/**
 * Returns the initial properties for a kanban.
 *
 * @returns {object} Initial properties for a kanban.
 */
getKanbanInitialProperties() {
    return {
        src: this.src,
        requestCount: 0,
        available: this.available,
        applied: this.applied,
    };
},

/**
 * Returns the storage key for kanban's in local storage.
 *
 * @returns {string} Storage key for kanban's.
 */

```

```

getKanbansStorageKey() {
    return 'kanbans';
},

/**
 * Retrieves the kanban's stored in local storage.
 *
 * @returns {Array} Kanban's stored in local storage.
 */
getKanbans() {
    let kanbans = localStorage.getItem(
        this.getKanbansStorageKey()
    );

    return JSON.parse(kanbans) ?? [];
},

/**
 * Sets the kanban's in local storage.
 *
 * @param {Array} kanbans - Kanban's to be stored in local storage.
 * @returns {void}
 */
setKanbans(kanbans) {
    localStorage.setItem(
        this.getKanbansStorageKey(),
        JSON.stringify(kanbans)
    );
},
}

});

```

</script>

@endPushOnce

Код ендпоінтів для сторінки Leads на серверній частині

```

Route::controller(LeadController::class)->prefix('leads')->group(function () {
    Route::get('', 'index')->name('admin.leads.index');
    Route::get('create', 'create')->name('admin.leads.create');
    Route::post('create', 'store')->name('admin.leads.store');

    Route::get('view/{id}', 'view')->name('admin.leads.view');
    Route::get('edit/{id}', 'edit')->name('admin.leads.edit');
    Route::put('edit/{id}', 'update')->name('admin.leads.update');

    Route::put('attributes/edit/{id}',
'updateAttributes')->name('admin.leads.attributes.update');
    Route::put('stage/edit/{id}', 'updateStage')->name('admin.leads.stage.update');
    Route::get('search', 'search')->name('admin.leads.search');
    Route::delete('{id}', 'destroy')->name('admin.leads.delete');

    Route::post('mass-update', 'massUpdate')->name('admin.leads.mass_update');
    Route::post('mass-destroy', 'massDestroy')->name('admin.leads.mass_delete');

    Route::get('get/{pipeline_id?}', 'get')->name('admin.leads.get');
    Route::delete('product/{lead_id}',
'removeProduct')->name('admin.leads.product.remove');
    Route::put('product/{lead_id}', 'addProduct')->name('admin.leads.product.add');
    Route::get('kanban/look-up', [LeadController::class,
'kanbanLookup'])->name('admin.leads.kanban.look_up');

    Route::controller(ActivityController::class)->prefix('{id}/activities')->group(function
() {
        Route::get('', 'index')->name('admin.leads.activities.index');
    });

```

```
Route::controller(TagController::class)->prefix('{id}/tags')->group(function () {  
    Route::post("", 'attach')->name('admin.leads.tags.attach');  
    Route::delete("", 'detach')->name('admin.leads.tags.detach');  
});
```

```
Route::controller(EmailController::class)->prefix('{id}/emails')->group(function () {  
    Route::post("", 'store')->name('admin.leads.emails.store');  
    Route::delete("", 'detach')->name('admin.leads.emails.detach');  
});
```

```
Route::controller(QuoteController::class)->prefix('{id}/quotes')->group(function () {  
    Route::delete('{quote_id?}', 'delete')->name('admin.leads.quotes.delete');  
});  
});
```

Код для CI/CD GitLab - AWS пайплайну

stages:

- prepare
- test
- build
- deploy
- post-deploy

variables:

PROJECT_DIR: /home/ubuntu/dancrm/prod/

SSH_USER_HOST: ubuntu@3.130.110.235

prepare:

stage: prepare

image: composer:2

script:

- composer install --no-interaction --prefer-dist

cache:

paths:

- vendor/

test:

stage: test

image: php:8.2

services:

- mysql:5.7

variables:

DB_HOST: mysql

DB_USERNAME: root

DB_PASSWORD: root

DB_DATABASE: testing

before_script:

- apt-get update && apt-get install -y unzip git zip
- docker-php-ext-install pdo pdo_mysql

script:

- php artisan migrate
- php artisan test

build:

stage: build

image: docker:latest

services:

- docker:dind

script:

- docker build -t dancrm:latest .

deploy:

stage: deploy

image: alpine:latest

only:

- main

before_script:

- apk --no-cache add openssh bash
- mkdir -p ~/.ssh
- echo "\$AWS_SSH_PRIVATE_KEY" | tr -d '\r' > ~/.ssh/id_rsa
- chmod 600 ~/.ssh/id_rsa
- echo "Host *\n\tStrictHostKeyChecking no\n" > ~/.ssh/config

- ssh-keyscan 3.130.110.235 >> ~/.ssh/known_hosts
- eval "\$(ssh-agent -s)"
- ssh-add ~/.ssh/id_rsa

script:

- |
- ssh \$SSH_USER_HOST << 'EOF'
- set -e
- cd "\$PROJECT_DIR"
- git fetch --all
- git checkout origin/main
- docker-compose exec -T prod composer install --no-interaction --prefer-dist
- docker-compose exec -T prod php artisan migrate --force
- EOF

post-deploy:

- stage: post-deploy
- image: alpine:latest
- only:
- main

before_script:

- apk --no-cache add openssh bash
- mkdir -p ~/.ssh
- echo "\$AWS_SSH_PRIVATE_KEY" | tr -d '\r' > ~/.ssh/id_rsa
- chmod 600 ~/.ssh/id_rsa
- echo "Host *\n\tStrictHostKeyChecking no\n" > ~/.ssh/config
- ssh-keyscan 3.130.110.235 >> ~/.ssh/known_hosts
- eval "\$(ssh-agent -s)"
- ssh-add ~/.ssh/id_rsa

script:

- |

```
ssh $SSH_USER_HOST << 'EOF'
cd "$PROJECT_DIR"
docker-compose exec -T prod php artisan config:cache
docker-compose exec -T prod php artisan route:cache
docker-compose exec -T prod php artisan view:cache
docker-compose restart
sudo chown -R :www-data "$PROJECT_DIR/code/storage"
echo "✅ Post-deploy completed."
EOF
```

Код Docker Compose

services:

prod:

build: ./backend

volumes:

- ./backend:/var/www/html

depends_on:

mysql:

condition: service_healthy

nginx:

image: nginx:latest

ports:

- "80:80"

volumes:

- ./backend:/var/www/html

- ./nginx.conf:/etc/nginx/conf.d/default.conf

depends_on:

- prod

node:

image: node:20

working_dir: /var/www/html

volumes:

- ./backend:/var/www/html

command: sh -c "npm install && npm run build"

mysql:

image: mysql:8.0

environment:

MYSQL_DATABASE: dancrm

MYSQL_ROOT_PASSWORD: root

MYSQL_USER: user

MYSQL_PASSWORD: password

volumes:

- db_data:/var/lib/mysql

- ./backend/init.sql:/docker-entrypoint-initdb.d/init.sql

ports:

- "3306:3306"

healthcheck:

test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]

interval: 10s

timeout: 5s

retries: 5

phpmyadmin:

image: phpmyadmin:5

ports:

- "8080:80"

environment:

PMA_HOST: mysql

PMA_USER: root

PMA_PASSWORD: root

depends_on:

mysql:

condition: service_healthy

volumes:

db_data:

КОПІЯ ТЕЗ ДОПОВІДІ

IV Всеукраїнська студентська наукова конференція «ФОРМУВАННЯ СУЧАСНОЇ НАУКИ: МЕТОДИКА ТА ПРАКТИКА»

Напрямок конференції: Інформаційні системи та технології.

Предметна область: ІТ – галузь.

Тема дослідження: Управління проєктами в ІТ сфері для підвищення ефективності систем.

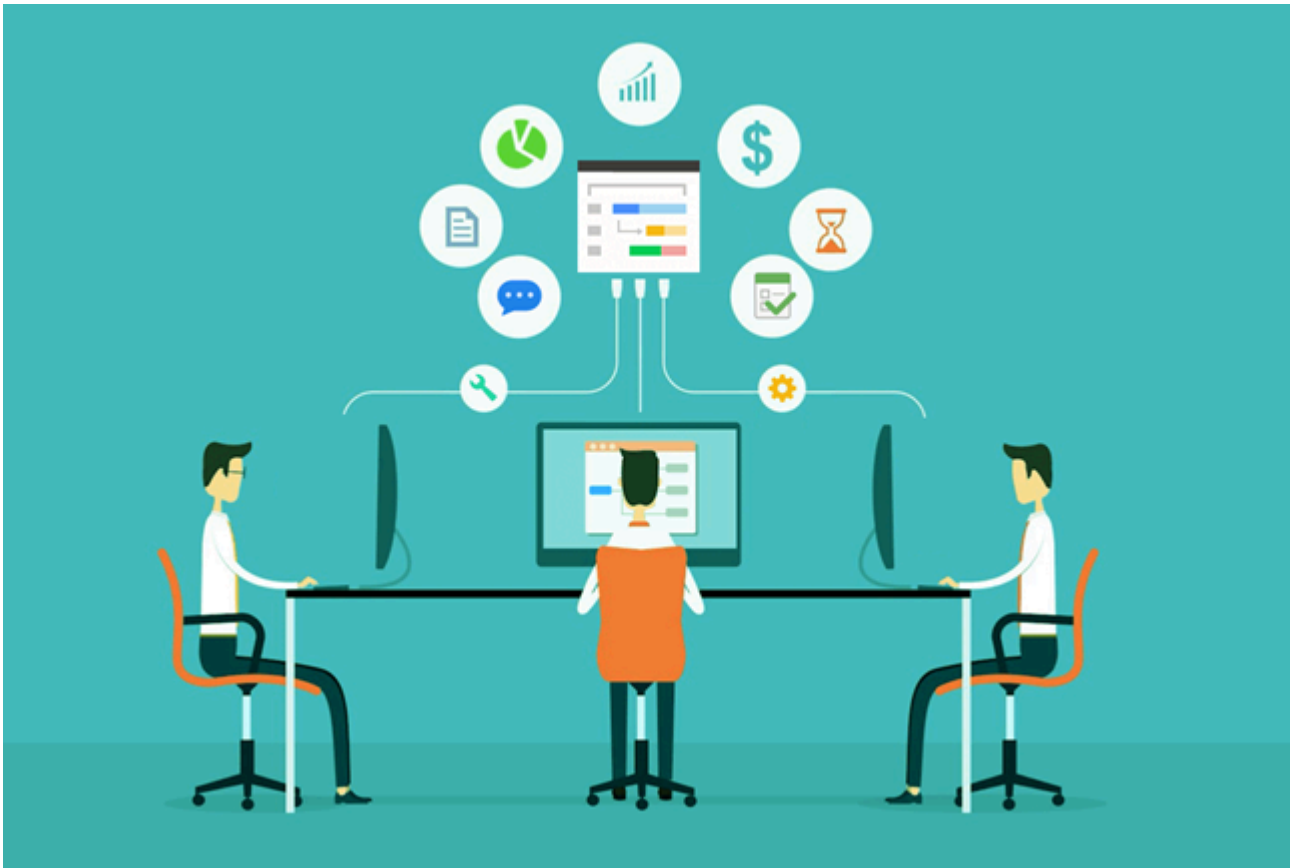
Бабієць Данило Ігорович

здобувач вищої освіти інституту інформаційних технологій в економіці
Київський національний економічний університет ім.Вадима Гетьмана, Україна

Науковий керівник: Устенко Станіслав Веніамінович

док. екон. наук, проф., професор кафедри інформаційних систем в економіці,
Київський національний економічний університет ім.Вадима Гетьмана, Україна

Управління проєктами в ІТ сфері для підвищення ефективності систем.



Дослідження присвячене вивченню сучасних стратегій та найкращих практик управління проєктами в ІТ-сфері з метою максимізації ефективності в реалізації проєктів. Мета дослідження полягає в розгляді оптимальних методів планування, виконання та контролю проєктів в інформаційно-технологічному середовищі.

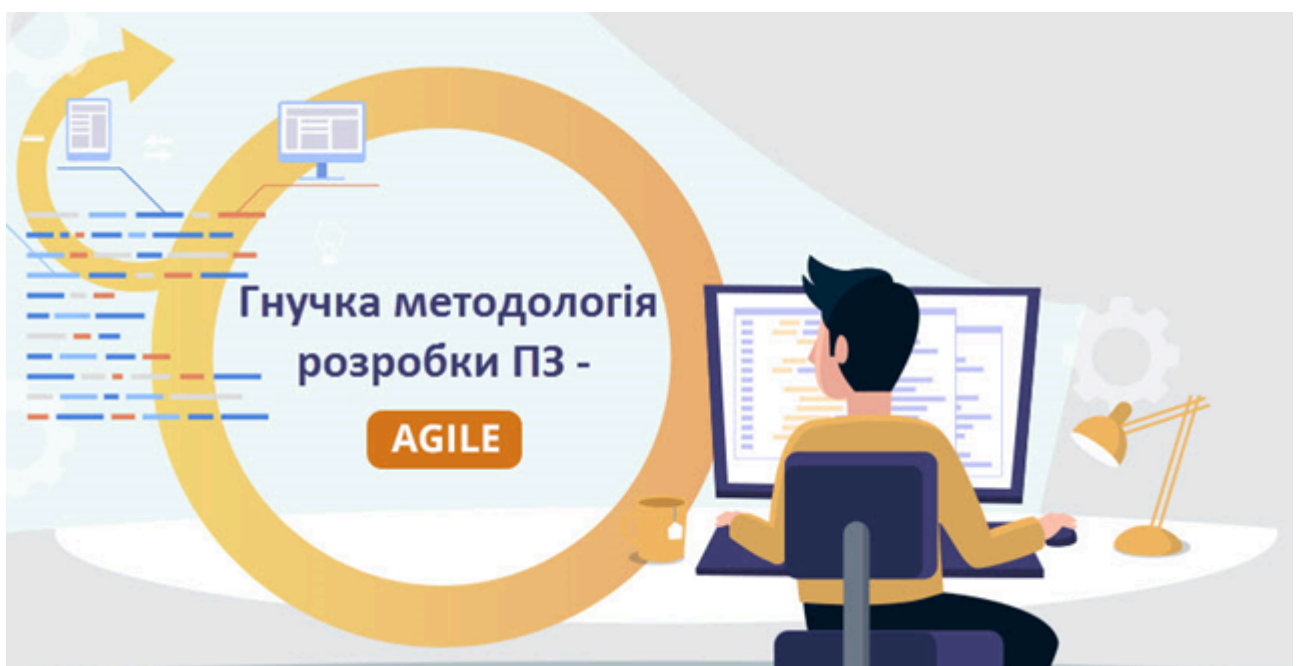
Дослідження включає в себе аналіз існуючих методологій управління проєктами, таких як Agile, Scrum, та Waterfall, з метою визначення їхнього застосування в ІТ-проєктах та виявлення оптимальних підходів до вибору стратегії управління в конкретних умовах.

Додатково, дослідження вивчає фактори, що впливають на успіх ІТ-проєктів, такі як комунікація, розподіл завдань, ефективне використання ресурсів та залучення зацікавлених сторін. Розглядаються питання підвищення адаптивності та

гнучкості проектних команд, а також використання інструментів для вдосконалення комунікації та забезпечення прозорості в управлінні.

Крім того, дослідження фокусується на впровадженні інноваційних технологій, таких як інтелектуальний аналіз даних, автоматизація завдань та використання інструментів для управління часом з метою підвищення продуктивності та зменшення ризиків у IT-проектах.

Agile



Agile — це підхід до розробки програмного забезпечення, який став популярним у сфері IT та розширюється на інші галузі. Основні принципи Agile визначені в "Маніфесті Агіль" (Agile Manifesto), який був вперше представлений у 2001 році. Основні аспекти Агіль включають гнучкість, взаємодію команд, ітеративність та здатність адаптації до змін. Розглянемо ключові елементи Агіль:

Ітеративність і інкрементальність:

Агіль розглядає розробку як ітеративний та інкрементальний процес. Проект розбивається на короткі ітерації, які називаються спринтами, зазвичай тривалістю від двох тижнів до місяця. Кожен спринт призводить до додавання нового функціоналу чи виправлення помилок.

Гнучкість та здатність до змін:

Агіль сприяє гнучкості у відповіді на зміни вимог. Клієнт (або представник користувачів) може вносити зміни вимог навіть пізніше в процесі розробки, і команда швидко адаптується до нових умов.

Взаємодія та комунікація: Агіль відділяє бар'єри між різними функціональними групами та підкреслює важливість активної комунікації між всіма учасниками проекту. Самоорганізація команди та спільна відповідальність підтримують ефективність.

Прозорість:

Агіль сприяє створенню прозорих процесів. Всі учасники проекту повинні мати доступ до інформації про стан проекту. Засоби для візуалізації, такі як дошки для справочної роботи (Kanban boards), часто використовуються для відстеження завдань.

Спрямованість на результат:

Важливий акцент на створенні функціонального та корисного продукту. Команда оцінює успішність не тільки за виконанням плану, але і за реальною користю для користувачів.

Співпраця із замовником:

Замовник (або представник користувачів) активно взаємодіє з командою протягом усього процесу. Це дозволяє забезпечити відповідність вимогам та миттєво реагувати на зміни в їхніх потребах.

Agile став ефективним підходом для розробки програмного забезпечення, особливо в умовах швидкої зміни бізнес-середовища. Його принципи та практики можуть бути успішно використані для підвищення ефективності управління проектами в ІТ-сфері.

SCRUM



Scrum є однією з найбільш популярних інкрементальних ітеративних методологій розробки програмного забезпечення, яка входить в сім'ю Агільних методів. Розглянемо ключові аспекти Scrum:

Ролі:

Product Owner: Відповідає за розробку і підтримку беклогу продукту, визначає пріоритети і вимоги.

Scrum Master: Забезпечує виконання правил Scrum, допомагає команді уникнути перешкод та вдосконалює процеси.

Розробники (Development Team): Самоорганізована команда, що виконує роботу та відповідає за результат.

Спринти:

Процес Scrum базується на коротких ітераціях, які називаються спринтами. Зазвичай, спринти тривають від 2 до 4 тижнів.

Беклог продукту (Product Backlog): Список всіх завдань і функціональностей, які мають бути виконані. Product Owner визначає пріоритети в беклозі.

Спринт-беклог (Sprint Backlog): Обраний з беклогу продукту набір завдань на конкретний спринт.

Спринт-планування (Sprint Planning): Команда визначає, які завдання вона взяне на себе протягом спринту, і розробляє план їх виконання.

Дейлі-стандап (Daily Scrum):

Коротке щоденне зустрічі, де кожен член команди вказує, що він виконав попереднього дня, що планує виконати на день та чи є які-небудь перешкоди.

- Спринт-огляд (Sprint Review): Завершальна зустріч спринту, де команда демонструє виконані роботи та отримує відгуки.

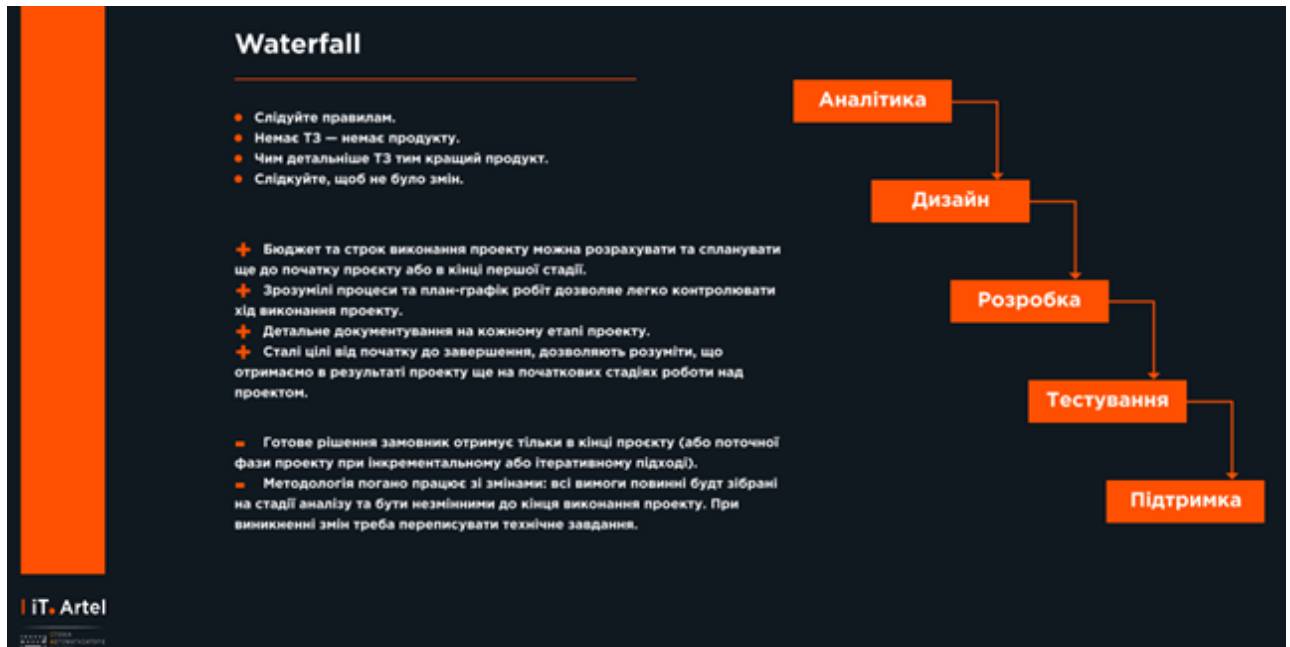
- Ретроспектива спринту (Sprint Retrospective): Зустріч для оцінки і вдосконалення процесів команди.

Переваги Scrum:

- Гнучкість та здатність до змін.
- Покращена комунікація між учасниками проекту.
- Висока прозорість завдань та їх прогресу.
- Швидке впровадження нових функцій.

Недоліки Scrum: Не всі проекти і команди підходять під Scrum. Може виникати складність управління проектами зі складною архітектурою або великим обсягом вимог. В цілому, Scrum виявляється дуже ефективним для проектів, які вимагають гнучкості, частих змін вимог і високої взаємодії в команді.

WATERFALL



Методологія **Waterfall**, також відома як каскадна методологія, є традиційним підходом до розробки програмного забезпечення, особливо великих проектів.

Розглянемо ключові аспекти Waterfall:

Лінійний процес:

Waterfall передбачає послідовне виконання етапів розробки, де кожен етап залежить від завершення попереднього. Етапи включають визначення вимог, проектування, реалізацію, тестування, впровадження та підтримку.

Фіксовані вимоги: Вимоги фіксуються на початковому етапі, і будь-які зміни можуть бути внесені тільки після завершення всього циклу проекту.

Мінімальна взаємодія з клієнтом:

Клієнт взаємодіє з командою переважно на початковому етапі для визначення вимог і на етапі впровадження.

Фазовість:

Проект поділяється на послідовні фази, і кожна фаза має конкретні цілі та виробляє певні вихідні результати.

Документація:

Waterfall вимагає обширної документації на кожному етапі розробки, такої як технічні специфікації, плани тестування і звіти.

Переваги Waterfall:

Простота управління та виконання.

Чітко визначені етапи та завдання.

Ідеально підходить для проектів із стабільними вимогами та невеликою необхідністю в змінах.

Недоліки Waterfall:

Нееластичність до змін вимог під час розробки.

Може бути складним управляти при великих та складних проектах.

Знижена взаємодія з клієнтом протягом процесу розробки.

Waterfall ідеально підходить для проектів із стабільними вимогами, де можливо чітко визначити фінальну мету та подробиці вимог заздалегідь. Однак, в умовах швидкозмінюваних вимог чи необхідності гнучкості, цей метод може виявитися обмеженим.

Висновок:

Аналіз вказує на те, що покращення адаптивності, гнучкості команди та ефективності управління проектами, можна досягти за допомогою сучасних методологій та інструментів. Загальний висновок полягає в тому, що підвищення адаптивності та гнучкості команд та використання ефективних інструментів для

комунікаціїта забезпечення прозорості в управлінні сприяє покращенню якості та швидкості виконання проектів.Команди, що активно впроваджують ці підходи, мають більше можливостей успішно вирішувати виклики та досягати поставлених цілей.

Список використаних джерел:

- 1) <https://it-artel.com.ua/pm-upravlinnja-proektami/>
- 2)<https://it-artel.ua/ru/blog/metodologiyi-upravlinnya-proektamy-vyvazhena-klasyka-w-aterfall-ta-gnuchkyj-agile/>
- 3) https://ela.kpi.ua/bitstream/123456789/34480/1/2019_Osnovy_upravlinnia.pdf
- 4)<http://dspace.wunu.edu.ua/retrieve/19638/%D0%9B%D0%B5%D0%BA%D1%86%D1%96%D1%97.pdf>



Звіт подібності

метадані

Назва організації

Kyiv National Economic University named after Vadym Hetman KNEU

Заголовок

Розроблення веб-платформи для управління процесами малого бізнесу

Автор

Науковий керівник / Експерт

Бабієць Данило ІгоровичТішков Б.О.

підрозділ

кафедра інформаційних систем в економіці

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1

25

Довжина фрази для коефіцієнта подібності 2



КП 2

9532

Кількість слів



КЦ

77982

Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		0
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)		9

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Копір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз

Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://ir.kneu.edu.ua/bitstreams/6ccdb206-8499-47f8-b15b-5d42b9b14ce2/download	28 0.29 %
2	https://ir.kneu.edu.ua/bitstreams/6ccdb206-8499-47f8-b15b-5d42b9b14ce2/download	25 0.26 %
3	Гладченко Ю.С._Па58-23 10/18/2024 Kyiv National Linguistic University (Кафедра теорії і практики перекладу з англійської мови)	17 0.18 %
4	https://ir.kneu.edu.ua/bitstreams/6ccdb206-8499-47f8-b15b-5d42b9b14ce2/download	12 0.13 %

5	Розроблення інформаційної управляючої системи для оптимізації діяльності цифрового інноваційного хабу 5/16/2025 Kyiv National Economic University named after Vadym Hetman KNEU (кафедра інформаційних систем в економіці)	11 0.12 %
6	http://ekhsuir.kspu.edu/bitstream/handle/123456789/18210/122_BelzaBI_fknfm_2023.pdf?sequence=1	10 0.10 %
7	https://jetiq.vntu.edu.ua/repository/getfile.php/8457.pdf	10 0.10 %
8	Інформаційна система для відділу іноземної літератури. Курсова робота 5/13/2025 Zhytomyr Agricultural Technical Professional College (Zhytomyr Agricultural Technical Professional College)	10 0.10 %
9	https://dspace.znu.edu.ua/jspui/bitstream/12345/19995/1/%D0%9A%D1%83%D1%88%D0%BD%D1%96%D1%80.docx	10 0.10 %
10	Розроблення інформаційної управляючої системи для оптимізації діяльності цифрового інноваційного хабу 5/16/2025 Kyiv National Economic University named after Vadym Hetman KNEU (кафедра інформаційних систем в економіці)	7 0.07 %

з бази даних RefBooks (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
------------------	-----------	--

з домашньої бази даних (0.25 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розроблення інформаційної управляючої системи для оптимізації діяльності цифрового інноваційного хабу 5/16/2025 Kyiv National Economic University named after Vadym Hetman KNEU (кафедра інформаційних систем в економіці)	24 (3) 0.25 %

з програми обміну базами даних (0.39 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Інформаційна система для відділу іноземної літератури. Курсова робота 5/13/2025 Zhytomyr Agricultural Technical Professional College (Zhytomyr Agricultural Technical Professional College)	20 (3) 0.21 %
2	Гладченко Ю.С._Па58-23 10/18/2024 Kyiv National Linguistic University (Кафедра теорії і практики перекладу з англійської мови)	17 (1) 0.18 %

з Інтернету (1.47 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://ir.kneu.edu.ua/bitstreams/6ccdb206-8499-47f8-b15b-5d42b9b14ce2/download	75 (5) 0.79 %
2	https://essuir.sumdu.edu.ua/bitstream/123456789/92908/1/Lysianskyi_bac_rob.pdf	29 (5) 0.30 %
3	http://ekhsuir.kspu.edu/bitstream/handle/123456789/18210/122_BelzaBI_fknfm_2023.pdf?sequence=1	16 (2) 0.17 %
4	https://jetiq.vntu.edu.ua/repository/getfile.php/8457.pdf	10 (1) 0.10 %