Прикарпатський національний університет імені Василя Стефаника

Факультет математики та інформатики

Кафедра комп’ютерних наук та інформаційних систем

ВИПУСКНА ДИПЛОМНА РОБОТА

Тема: «Корпоративна онлайн-система менеджменту завдань та колаборації»

Виконав: студент IV курсу гр. ІСТ-41

ОР бакалавр

спеціальності 126 “Інформаційні системи та технології”

Вітер Захара Андрійовича

Науковий керівник: к.т.н., доц. Превисокова Наталія Володимирівна

Рецензент: ст.викл Ізмайлов Артем Вікторович

м. Івано-Франківськ – 2025

Прикарпатський національний університет імені Василя Стефаника

Інститут, факультет _____
Кафедра _____
Освітньо-кваліфікаційний рівень_____
Напрям підготовки (Спеціальність)_____
_____
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри _____
(підпис)
_____
(прізвище, ініціали)
“____” _____20___ р.

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

_____
(прізвище, ім’я, по батькові)

1. Тема роботи_____
_____,

керівник роботи_____
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом університету від “___”_____20__ р. №_____.
2. Строк подання студентом роботи_____
3. Вихідні дані до роботи_____
_____
_____
_____
_____

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

5. Перелік графічного матеріалу
_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка

Студент _____ _____

(підпис) (прізвище та ініціали)

Керівник роботи _____ _____

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота складається з вступу, трьох розділів, висновку та рекомендацій, списку використаної літератури з 27-ма джерелами. Загальний обсяг роботи становить 55 сторінок, на яких представлено 5 рисунків, 1 таблицю.

Робота присвячена аналізу предметної області, та розробці корпоративнох онлайн-системи менеджменту завдань та колаборації, котра вирішує основні проблеми наявних SaaS рішень.

У першому розділі здійснено аналіз очікуваних функцій корпоративних онлайн-систем менеджменту завдань та колаборації. У другому розділі аналізуються існуючі продукти-аналоги, та визначаються проблематики, котрі такі системи мають. У третьому розділі розглядається розробка, та переваги розробленої системи.

Ключові слова: система менеджменту та колаборації, відкрите джерело, інтеграція, вебхук, Laravel, Vue, Система Колаборації.

ABSTRACT

The thesis consists of an introduction, three chapters, a conclusion and recommendations, a list of used literature with 27 sources. The total volume of the work is 55 pages, which present 5 figures, 1 table.

The work is devoted to the analysis of the subject area and the development of a corporate online task management and collaboration system that solves the main problems of existing SaaS solutions.

The first section analyzes the expected functions of corporate online task management and collaboration systems. The second section analyzes existing similar products and identifies the problems that such systems have. The third section considers the development and advantages of the developed system.

Keywords: management and collaboration system, open source, integration, webhook, Laravel, Vue, Collaboration Software.

ЗМІСТ

ВСТУП....4
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ КОРПОРАТИВНИХ ОНЛАЙН СИСТЕМ УПРАВЛІННЯ ТА СПІВПРАЦІ....6
1.1 Суть та призначення корпоративних систем менеджменту....6
1.2 Еволюція та роль онлайн платформ в бізнес-середовищі....8
1.3 Класифікація засобів: від особистих таск-трекерів до комплексних корпоративних платформ....10
1.4 Вимоги до сучасних систем менеджменту в умовах цифровізації....13
РОЗДІЛ 2. ОГЛЯД РИНКУ ТА АНАЛІЗ ІСНУЮЧИХ СИСТЕМ....16
2.1 Порівняльна характеристика популярних платформ....16
2.2 Проблематика впровадження готових рішень у середні та великі компанії....19
2.3 Аналіз типових недоліків SaaS-підходу....22
2.4 Переваги open-source підходу в контексті безпеки та гнучкості....26
2.5 Обґрунтування вибору технологічного стеку VILT....30
РОЗДІЛ 3. РОЗРОБКА КОРПОРАТИВНОЇ СИСТЕМИ КОЛАБОРАЦІЇ ТА МЕНЕДЖМЕНТУ 35
3.1 Архітектура та модульність створеного продукту....35
3.2 Процес розробки та реалізації....38
3.3 Функціональність системи: основні модулі та логіка роботи....41
3.4 Перспективи масштабування та інтеграції системи у внутрішню інфраструктуру компаній 45
3.5 Цінність рішення для корпоративного використання....47
3.6 Порівняльні переваги над SaaS-рішеннями....48
ВИСНОВОК....50
Перспективи подальшого розвитку....51
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ....53

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

ACL – Access Control List

СУБД – Система Управління Базами Даних

API – Application Programming Interface

VILT – Vue Inertia.js Laravel TailwindCSS

CSS – Cascading Style Sheets

MVC – Model View Controller

CRM – Customer Relationship Management

ERP – Enterprise Resource Planning

SSO – Single Sign On

ВСТУП

Упродовж останнього десятиліття корпоративні онлайн-системи управління проєктами та колаборації перетворилися з факультативного інструмента на практично незамінну складову сучасного бізнес-середовища [30, с. 2-3]. Потреба в оптимізації командної взаємодії, гнучкому управлінні завданнями та забезпеченні прозорості робочих процесів в умовах цифрової трансформації породила стрімке зростання таких платформ колаборації та менеджменту завдань, як Slack, Jira, Asana, Trello, ClickUp, Zoho Projects та інші [26]. Власне кажучи, мало яка сучасна компанія сьогодні може дозволити собі повністю проігнорувати такі рішення - незалежно від галузі чи розміру.

Проте разом із розширенням ринку, ускладнюються і самі інструменти. Підприємства все частіше стикаються з перенавантаженням інтерфейсів, високим порогом входу для нових працівників, а також - не менш критично - з обмеженнями безкоштовних або базових версій [31, с. 31]. Крім того, поширена практика зберігання даних у хмарних середовищах третіх сторін ставить під сумнів контроль над конфіденційною інформацією [45, с. 636-639].

На цьому тлі програмне забезпечення з відкритий код (open source) як альтернатива готовим SaaS-рішенням виглядає все привабливішим, адже можливість кастомізувати систему під власні потреби, локально розгортати її на внутрішніх серверах, інтегрувати із вже існуючими інструментами, самому контролювати місце збереження даних, тощо - усе це відкриває ширші горизонти для середнього і навіть малого бізнесу [1, с. 2492]. Але попри доступність коду, розробка власного продукту лишається складним інженерним завданням, що вимагає розуміння як технічної, так і бізнесової логіки.

Саме тому у цій дипломній роботі буде розглянуто питання розробки повноцінної корпоративної онлайн-системи менеджменту завдань та колаборації із початковим нахилом на подальший відкритий код, орієнтацію на використання в рамках окремого бізнесу - "одна компанія - одна система". Акцент буде зроблено на

простоті, безпеці, зручності масштабуванні та можливій адаптації до потреб конкретного користувача.

Мета роботи:

Розробити веб-систему корпоративного менеджменту та колаборації з відкритим кодом, адаптовану для автономного використання в межах одного підприємства.

Завдання дослідження:

- Провести аналіз сучасних систем менеджменту та колаборації.
- Обрати технології виконання, та обґрунтувати їх вибір.
- Створити веб-сервіс, котрий буде виконувати та дозволить здійснювати всі визначені вимоги та функції корпоративних онлайн-систем управління проєктами та завданнями.

Об’єкт дослідження:

Корпоративна онлайн-система менеджменту завдань та колаборації

Предмет дослідження:

Методи, інструменти та архітектури програмного забезпечення, які дозволяють реалізувати систему менеджменту з відкритим кодом.

Методи дослідження:

У дипломній роботі використано методи аналізу та синтезу інформаційних систем, порівняльну характеристику існуючих рішень, а також практичне моделювання та програмну реалізацію.

Структура роботи:

Робота складається з трьох розділів.

Перший розкриває теоретичну базу корпоративних онлайн-систем менеджменту завдань та колаборації, а також визначає короткі завдання, котрі очікуються від такої системи.

У другому здійснюється детальний огляд та аналіз сучасних популярних рішень на ринку, а також визначенню проблематики більшості таких рішень, та формування технологічної основи для подальшої розробки.

Третій розділ присвячено архітектурі майбутньої системи, процесу розробки даної системи, а також потенційним перевагам над аналогами та порівнянням з попередньо проаналізованими аналогами.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ КОРПОРАТИВНИХ ОНЛАЙН СИСТЕМ УПРАВЛІННЯ ТА СПІВПРАЦІ

1.1 Суть та призначення корпоративних систем менеджменту

У сучасних умовах розвитку економіки - інформація набуває важливого статусу [30, с. 1]. Її ефективне опрацювання, систематизація, передача та захист можуть прямо впливати на результативність та ефективність діяльності підприємства. На цьому фоні зростає важливість технологічних рішень, які забезпечують не лише обмін даними між співробітниками, а й контроль за виконанням завдань, координацію дій і формування спільної стратегії. Власне саме таку роль - виконують корпоративні онлайн-системи менеджменту та колаборації.

Корпоративна система менеджменту - це комплексне програмне забезпечення, розроблене з метою автоматизації та системізації управління проєктами, контролю за виконанням завдань, організації внутрішньої комунікації та взаємодії між учасниками бізнес-процесів [3, с. 281]. Залежно від специфіки підприємства, такі системи можуть охоплювати різні сфери діяльності - від планування робочого часу та ресурсів до стратегічного управління портфелем проєктів. Однак спільною рисою всіх платформ є орієнтація на підвищення ефективності командної роботи шляхом зменшення рівня хаосу в інформаційних потоках та підвищення прозорості у прийнятті рішень та загальних направлень проєктів.

Вони дозволяють формалізувати те, що зазвичай відбувається у вигляді розмов, електронних листів або щотижневих нарад. Іншими словами платформи такого типу - це інструменти, що зводять елементи людської взаємодії у контрольовану систему, доступну аналітиці, автоматизації та масштабуванню [3, с. 297].

Особливо цінними такі системи мають для компаній, які працюють у форматі віддаленої або гібридної моделі. В умовах, коли працівники фізично не перебувають

в одному офісі, виникає потреба у спільному "цифровому просторі", який замінює дошку для нотаток, голосове обговорення завдань чи імпровізовані зустрічі в коридорі. Корпоративна система менеджменту в цьому сенсі виступає не просто інструментом, а, по суті, середовищем функціонування організації, хоча навіть у форматі роботи на місці такі системи показують ефективність через мінімізування повторень та репететивних питань [30, с. 30].

Не слід ототожнювати подібні системи з простою платформою списку завдань. Хоча базова функціональність часто й передбачає створення завдань, призначення виконавців, встановлення дедлайнів і моніторинг виконання, але корпоративний рівень передбачає глибшу інтеграцію в бізнес-процеси: побудову логіки підпроєктів, залежності між задачами, використання систем доступу та ролей, генерацію динамічних звітів, ефективну роздачу завдань, тощо. [5, с. 8]

З погляду класифікації, я пропоную виокремити кілька базових функціональних блоків, що є типовими для більшості корпоративних систем менеджменту:

- Планування - включає календарі, діаграми Ґанта, дорожні карти (з англ. Roadmaps – програми (в аналаговому сенсі) розвитку проєкту або фірми загалом) та інші інструменти, що дозволяють візуалізувати прогрес, дедлайни, напрямок, додаткову документацію, тощо.
- Комунікація - забезпечується внутрішніми чатами, системою коментарів, пуш-сповіщеннями, а іноді й інтеграцією з email чи зовнішніми месенджерами.
- Управління ресурсами - люди, техніка, бюджети, робочі години, відпустки, вихідні, особисті дані, тощо.
- Контроль доступу - розмежування прав і ролей з урахуванням корпоративної ієрархії.
- Інтеграції - взаємодія з іншими бізнес-системами: CRM, ERP, поштовими клієнтами, системами документообігу тощо.

Хоч на перший погляд може здатись, що такі системи тільки полегшують керівництво та менеджменту, та насправді вони також значно впливають і на

культуру праці в організації та ефективність працівників [28]. Середовище, де всі бачать свої задачі, дедлайни, зони відповідальності та мають прозорий зворотний зв'язок, стимулює і індивідуальну відповідальність і командну ефективність. Це, у свою чергу, призводить до зростання лояльності працівників, підвищення якості виконання завдань та покращення загальної керованості компанії.

Отже, корпоративна онлайн-система менеджменту не є лише технічним рішенням. Вона - відображення управлінської філософії підприємства, його підходу до організації праці, комунікації та стратегічного планування. І саме тому її грамотний вибір або розробка "під себе" може мати суттєві наслідки для ефективності бізнесу в цілому.

1.2 Еволюція та роль онлайн платформ в бізнес-середовищі

Ще кілька десятиліть тому ідея "спільної роботи над проєктами" обмежувалась фізичними зустрічами, стенограмами на папері та телефонними перемовинами. І хоч такі методи мали свої переваги - передусім у гнучкості та «людяності» взаємодії - вони абсолютно не витримували викликів часу, контролю чи швидкого відновлення інформації. Потреба в цифровій трансформації управління проєктами була неминучою. [32]

Першими передвісниками сучасних систем колаборації стали прості таск-менеджери кінця 90-х - початку 2000-х років. Це були автономні програми (часто локального встановлення), основна функція яких зводилась до фіксації списку справ. Microsoft Outlook, Lotus Notes чи більш вузькоспеціалізовані інструменти на зразок Basecamp були одними з перших кроків у напрямку структурованого планування та делегування завдань у цифровій формі. Проте їх функціональність лишалась обмеженою і часто не покривала потреби в командній роботі, а організації не сприймали їх важливою частиною робочого процесу. [32]

Справжній прорив стався із приходом епохи хмарних технологій. Поява інструментів, які дозволяли працювати над одним і тим самим проєктом у реальному часі - незалежно від географії учасників, пристроїв чи операційних

систем - змінила принцип управління командною взаємодією [32]. Онлайн платформи (Trello, Asana, Jira тощо) стали не просто «блокнотами з дедлайнами», а повноцінними середовищами для ефективної комунікації, прийняття рішень і контролю над їх реалізацією.

Водночас ця еволюція збіглась з кількома важливими тенденціями в бізнес-середовищі:

- Глобалізація команд. Усе частіше компанії формують проектні групи з учасників, які перебувають у різних містах, країнах чи навіть часових поясах [33]. Без інструментів цифрової координації така взаємодія була б ускладненою.
- Гнучкі методології управління (Agile, Scrum, Kanban). Їх широке впровадження у сфері IT, а згодом і в інших галузях, вимагало специфічних інструментів для візуалізації “беклогу”, “спринтів”, завдань, пріоритетів. Онлайн системи стали природним доповненням до цих підходів.
- Якщо раніше IT-інфраструктура була привілеєм великих корпорацій, то зараз - завдяки SaaS та Open Source моделям - навіть невелика команда може дозволити собі ефективний таск-менеджер із хмарним, або локальним, зберіганням без потреби в цілих IT відділах, або наймі зовнішніх IT-вендорів. [34 с.29-30]
- Потреба у прозорості та звітності. Бізнес дедалі частіше вимагає не просто результату, а й обґрунтування: чому саме такий шлях був обраний, скільки часу витрачено, хто виконував яку частину роботи. Онлайн платформи дозволяють фіксувати ці процеси автоматично, що значно знижує витрати на аудит.

Важливо зазначити, що роль таких систем у сучасному бізнес-середовищі вже давно вийшла за межі "внутрішньої організації". Усе більше компаній включають клієнтів або зовнішніх підрядників до своїх платформ - задля прозорості, довіри, кращої якості сервісу. Наприклад, у проєктній документації замовник може залишати коментарі, узгоджувати завдання, бачити хід реалізації. У підсумку - скорочується кількість непорозумінь, зменшується потреба в постійних

погодженнях і пришвидшується цикл прийняття рішень. [4, с.11]

Втім, еволюція не завершена. Вже зараз спостерігається тенденція до посилення таких аспектів, як:

- Інтелектуалізація (використання AI для прогнозування ризиків, пріоритизації завдань),
- Інтеграція з бізнес-аналітикою (BI-додатки, автоматичні дашборди на основі внутрішніх KPI).

[35]

У цьому контексті все більш актуальними стають open-source платформи, які дозволяють компаніям не просто користуватися готовим продуктом, а будувати його на власних умовах: контролювати архітектуру, змінювати логіку, інтегрувати із вже існуючими системами, не боячись “vendor lock-in” (з англ. “Замикання постачальника” – явище, коли організація чи особа стає залежною від постачальника послуг, та його зміна обійдеться дорожче, аніж продовження користування незадовільним продуктом). [8, с. 4-5, 18-19]

Таким чином, корпоративні онлайн-системи не лише відображають сучасну організацію праці, а й формують її. Вони не просто підлаштовуються під бізнес - вони самі стають інструментом його перетворення.

1.3 Класифікація засобів: від особистих таск-трекерів до комплексних корпоративних платформ

У процесі становлення та поширення систем управління завданнями та колаборації створились певні типи таких рішень, які відображає не лише функціональні відмінності, а й суттєву різницю в цілях використання. Я пропоную поділити такі інструменти на три основні категорії за масштабом впровадження, рівнем спеціалізації та цільовою аудиторією: персональні, командні та корпоративні.

Персональні системи управління завданнями

До першої категорії належать рішення, орієнтовані на індивідуальних

користувачів. Найчастіше вони призначені для організації особистого часу, ведення нотаток, нагадувань та контроль за простими повторюваними завданнями. Прикладами таких систем є Todoist, Any.do, Google Tasks, Microsoft To Do.

Спільною рисою для цього типу є:

- Однокористувацький режим або обмежена командна взаємодія.
- Мінімалізм інтерфейсу та простота структури.
- Відсутність ієрархічної моделі даних (проєкт → підпроєкти → задачі).

Здебільшого такі системи не потребують складного налаштування, або налаштування взагалі й орієнтовані на швидкий старт. Їх перевага - у доступності та низькому порозі входу. Недоліком - відсутність можливості масштабування на рівень групової роботи. [17]

Командні системи

Другий тип охоплює продукти, які дозволяють координувати роботу невеликих груп, часто - у межах одного відділу чи проєктної команди. Вони забезпечують базовий рівень співпраці, дозволяючи створювати завдання, призначати відповідальних, вести коментарі, фіксувати строки виконання та статуси. Серед найпоширеніших представників цього класу - Trello, Asana, ClickUp.

Особливостями командних систем є:

- Організація даних у вигляді дощок, списків, спринтів або таблиць.
- Можливість спільного редагування та комунікації всередині завдань.
- Підтримка історії змін та простих інструментів звітності.

Такі системи часто є оптимальним вибором для стартапів, невеликих агентств, фріланс-команд. Вони достатньо гнучкі, мають інтеграції з іншими популярними інструментами (Google Workspace, Slack, GitHub), проте вразливі до перевантаження у разі масштабування. В певному сенсі, це компроміс між простотою й функціональністю. [11, 12]

Корпоративні платформи

Окрему категорію складають повноцінні корпоративні системи, призначені для середніх і великих підприємств. Вони відзначаються складною структурою, розвиненою логікою управління ролями, підтримкою великої кількості користувачів та модульною архітектурою. Відомими представниками є Jira, Microsoft Project, Zoho Projects, Wrike, Oracle Primavera.

До характерних рис корпоративних систем належать:

- Можливість налаштування прав доступу на рівні об'єкта або дії.
- Підтримка ієрархій задач, підзадач, етапів та фаз.
- Інтеграція з CRM, ERP, SCM-системами та сторонніми API.
- Розвинена аналітика, можливості прогнозування, контроль за ресурсами.
- Підтримка документації, журналювання дій користувачів, логування подій.

Водночас із високим рівнем потужності такі системи часто мають істотні вимоги до впровадження: вони потребують технічної підготовки персоналу, наявності внутрішніх адміністраторів, інтеграторів або навіть окремого підрозділу підтримки. Це робить їх менш доступними для малого бізнесу. [13, 16]

Проміжні або гібридні рішення

Сучасний ринок демонструє тенденцію до стирання меж між зазначеними категоріями. Багато платформ пропонують гнучкі тарифні плани, в яких базовий функціонал можна використовувати безкоштовно, а за додаткові можливості (розширене керування доступами, API, аналітика) - сплачувати. Це дозволяє почати з простого таск-трекера, а згодом трансформувати його у майже повноцінну систему управління проєктами.

Наприклад, ClickUp у базовій версії може функціонувати як командний інструмент, проте з часом - при відповідному налаштуванні - здатен обслуговувати великі організації з розподіленими відділами [14].

Висновки щодо класифікації

Розуміння вищезгаданої класифікації необхідне не лише для вибору готового

продукту, але й для побудови власної системи. Розробляючи open-source платформу для корпоративного використання, варто чітко усвідомлювати, в яку категорію має потрапити майбутній продукт. У випадку даної курсової роботи - йдеться саме про корпоративну онлайн-систему гібридного типу, яка одночасно підпаде і під корпоративну і під командну класифікацію.

1.4 Вимоги до сучасних систем менеджменту в умовах цифровізації

Сучасне цифрове середовище формує цілком нові вимоги до програмних засобів, призначених для організації роботи в середині підприємства. Від систем менеджменту вже недостатньо очікувати простої реєстрації завдань чи підтримки колективної роботи - вони мають бути адаптивними, безпечними, масштабованими й, водночас, інтуїтивно зрозумілими для користувача. З огляду на загальну тенденцію цифровізації бізнес-процесів - ці вимоги стають не лише бажаними, а фактично критичними для збереження конкурентоздатності. [30, с. 4, 10-11]

Першим бар’єром, який часто виникає при переході на нову систему, є складність її впровадження як та технічному, так і на “людяному” рівні [17, с. 319]. Підприємства, особливо малі й середні, не завжди мають змогу виділяти окрему команду на тривалу адаптацію інструменту. Саме тому сучасна система повинна забезпечувати:

- Швидке розгортання - як локально, так і в хмарному середовищі.
- Мінімальні технічні вимоги до інфраструктури.
- Зрозумілий та інтуїтивний інтерфейс, який не потребує тривалої підготовки або навчання персоналу.

Ці аспекти не лише пришвидшують старт роботи, а й знижують загальні витрати на впровадження.

Однією з головних проблем SaaS-продуктів - це жорстко задана логіка, яка часто не враховує специфіку конкретного бізнесу. У свою чергу, сучасні вимоги диктують потребу в гнучких, кастомізованих рішеннях, які можна адаптувати до внутрішніх процесів компанії. [20]

Ідеально, якщо система має відкритий API або, ще краще - є open-source рішенням, що дозволяє здійснювати глибші зміни на рівні логіки або інтерфейсу [8, с. 4-5, 18-19].

Варто зазначити, що у період, коли інформаційна безпека стає чи не головною загрозою для цифрового бізнесу, корпоративні системи, що потребують підвищеного захисту власних даних (такі як системи, котрі обробляють особисті дані медичних пацієнтів) повинні забезпечувати: локальне зберігання даних або, принаймні, можливість вибору між хмарною та локальною інсталяцією, повну закритість системи від зовнішніх запитів, задля мінімізації “знаходження прогалини” та контроль доступу до кожного елемента системи з урахуванням ролей, зон відповідальності та ієрархій. [37, с. 33-34, 18-19]

Також система повинна ефективно функціонувати як у невеликих командах, так і при зростанні до сотень користувачів. Це передбачає:

- Оптимізовану архітектуру бази даних, здатну витримувати великі обсяги транзакцій.
- Швидку обробку запитів без затримок навіть при високому навантаженні.
- Гнучку структуру модулів, яку можна розширювати новими функціями без потреби в переробці базових компонентів.

[9, с. 23-28]

Окремим пунктом виступає розділення системи на фронтенд і бекенд, що дозволяє окремо масштабувати інтерфейс та обробку логіки, залежно від навантаження.

І нарешті, незалежно від кількості функцій, система має бути зручною. Забезпечення позитивного користувацького досвіду (UX) - це вже не побажання, а стандарт [10, с. 2275-2276]:

- Інтуїтивний дизайн інтерфейсу.
- Доступність для користувачів з різними рівнями IT-компетентності.
- Адаптивність до мобільних пристроїв - смартфонів, планшетів.

- Підтримка локалізацій, включаючи українську мову.

Ці вимоги формуються не лише ринком, а й очікуваннями користувачів, які звикли до простоти у щоденних цифрових інструментах (наприклад, месенджерах, календарях, банківських додатках).

Вцілому, сучасні системи менеджменту та колаборації вже давно вийшли за межі функціоналу "створити задачу - виконати задачу". Вони стали платформами, навколо яких вибудовується цифрове життя підприємства та більшість його бізнес-процесів. Їх роль - не лише координувати дії, а й забезпечувати аналітичну основу для ухвалення рішень, підвищувати прозорість, забезпечувати безпеку, зменшувати ризики та, зрештою - підсилювати конкурентоздатність компанії. [2, с. 38; 30 с. 5-7]

Саме тому при розробці таких систем - зокрема, open-source продуктів - важливо орієнтуватися не лише на функціональні блоки, а й на архітектурні принципи, що дозволять цій системі бути не просто технічним інструментом, а надійним «скелетом» внутрішніх процесів компанії, а також легко піддаватись модифікаціям та покращенням як “мейнтейнерами” таких продуктів, так і напряму користувачами з достатніми IT-навичками.

РОЗДІЛ 2. ОГЛЯД РИНКУ ТА АНАЛІЗ ІСНУЮЧИХ СИСТЕМ

2.1 Порівняльна характеристика популярних платформ

На сучасному ринку існує значна кількість програмних рішень, орієнтованих на управління завданнями, планування проєктів та організацію командної взаємодії. Кожна система має свої унікальні риси, які визначають доцільність її використання в тому чи іншому бізнес-контексті.

Для об’єктивного аналізу - розглянемо найпопулярніші платформи в сфері корпоративного менеджменту та колаборації: Trello, Asana, Jira, ClickUp, Microsoft Planner (частина Teams) та Zoho Projects.

Trello - платформа, що базується на підході Kanban-дошки. Через простоту візуального інтерфейсу задач - вона здобула велику популярність серед невеликих команд, фрілансерів, тощо. Тут користувачі організовують завдання у вигляді карток, які можна перетягувати між колонками, що символізують етапи виконання (наприклад: «To Do», «In Progress», «Done»). [11]

- Переваги:
  - Інтуїтивно зрозумілий інтерфейс.
  - Швидкий старт без навчання.
  - Наявність базового безкоштовного тарифу.
  - Велика кількість шаблонів.
- Недоліки:
  - Відстутність відстеження часу.
  - Обмеженість кастомізації прав доступу.
  - Не придатний для складних проєктів з великою кількістю залежностей.

Оптимальний для персонального користування та невеликих команд. Не рекомендований для компаній із багаторівневою структурою управління, або складними бізнес-процесами. [11]

Asana - більш функціонально насичена платформа, яка дозволяє створювати як прості списки завдань, так і складні проєкти зі зв’язками між задачами, пріоритетами та дедлайнами. Вона активно використовується у сферах маркетингу, дизайну, IT. [12]

- Переваги:
  - Нативна інтеграція з Google Workspace, Slack, Outlook.
  - Система пріоритетів і тегів.
  - Розвинена аналітика.
- Недоліки:
  - Складність інтерфейсу для новачків.
  - Суттєва обмеження функціоналу у безкоштовній версії.
  - Вразливість до перевантаження інформацією при великій кількості завдань.

Добре підходить для середніх компаній і команд з потребою в гнучкому управлінні, але вимагає часу на адаптацію. [12]

Jira - один із найпотужніших інструментів для управління проєктами, що спеціалізується на IT-сфері, особливо на розробці програмного забезпечення. Його структура створена з урахуванням методологій Agile і Scrum, із підтримкою беклогів, спринтів, релізів тощо. [13]

- Переваги:
  - Гнучка система кастомізації.
  - Повна підтримка Agile/DevOps-процесів.
  - Автоматизація робочих процесів.
  - Розширений контроль доступу.
- Недоліки:
  - Надмірна складність для нетехнічних команд.
  - Значні ресурси на впровадження.
  - Потреба в адміністраторові системи.

Надійний вибір для великих технологічних компаній, але явно надлишковий для простих бізнес-процесів. [13]

ClickUp - відносно нова, але вже дуже популярна система, яка позиціонує себе як «єдине робоче місце» для всіх типів бізнес-процесів. Вона підтримує створення задач, документів, чату, тайм-трекінгу, звітів тощо - все в межах однієї платформи. [14]

- Переваги:
  - Широка функціональність «із коробки».
  - Гнучка настройка інтерфейсу.
  - Доступна API-інтеграція.
  - Підтримка кастомних статусів, полів, автоматизації.
- Недоліки:
  - Перевантаження функціями - складно зорієнтуватися з першого разу.
  - Висока потреба в налаштуваннях для зручності.

Ідеальне рішення для команд, які готові інвестувати час у глибоку персоналізацію інструменту. [14]

Microsoft Planner (у складі Microsoft Teams) - частина екосистеми Microsoft 365, що нативно інтегрується з Teams, Outlook, OneDrive. Основна перевага - у seamless-інтеграції для користувачів, які вже працюють у продуктах Microsoft. [15]

- Переваги:
  - Легке впровадження в існуючу IT-інфраструктуру Microsoft.
  - Централізоване зберігання даних.
- Недоліки:
  - Обмежена кастомізація.
  - Невеликий набір функцій.
  - Необхідність наявності підписки на Microsoft 365.

Найбільш доцільний варіант для компаній, що вже користуються Microsoft-середовищем. Як окремий продукт – слабкий. [15]

Zoho Projects

Цей інструмент вирізняється глибокою інтеграцією з екосистемою Zoho, що включає CRM, пошту, документи, звітність. Zoho Projects підходить для компаній,

які шукають комплексне рішення на основі єдиної платформи.

- Переваги:
  - Підтримка діаграм Ганта.
  - Розширена звітність.
  - Часткова нативна інтеграція з екосистемою Zoho.
  - Вбудована, хоч і слабка система ролей та дозволів.
- Недоліки:
  - Частина функцій доступна лише за підпискою.
  - Менш гнучкий поза межами Zoho-екосистеми.

Сильний варіант для користувачів Zoho, але в ізоляції від інших сервісів - менш універсальний. [16]

Порівняльний аналіз показує, що жодна з існуючих платформ не є універсальною - кожна відповідає окремим бізнес-сценаріям. Саме через цю обмеженість і виникає можливість розробити систему з відкритим вихідним кодом, яка дозволяють бізнесу самому собі формувати інструментарій, виходячи з реальних та власних, унікальних, потреб, а не підлаштовуватись під жорстку логіку сторонніх продуктів. У цьому контексті open-source рішення виступають не лише альтернативою, а потенційно - основою нової хвилі гнучкого корпоративного ПЗ. [8]

2.2 Проблематика впровадження готових рішень у середні та великі компанії

Попри великий набір функціональності та різноманітність готових систем управління завданнями й колаборації, їх впровадження в середовищі середніх і великих підприємств часто супроводжується багатьма складностями. Ці труднощі мають як технічний, так і організаційний характер, що робить їх особливо чутливими у масштабних структурах із чітко сформованими внутрішніми процесами. Проблеми, що виникають в таких випадках, не завжди зумовлені недоліками самих програмних продуктів, часто, причина полягає у невідповідності логіки їх роботи до вже існуючих бізнес-процесів, корпоративної культури або

вимог до безпеки. [38, с. 177-178, 180]

1. Невідповідність існуючим бізнес-процесам

Одна з найпоширеніших проблем полягає в тому, що готові платформи мають власну структуру даних і логіку організації проєктів, яка рідко ідеально збігається з тим, як компанія організовує свою роботу. Наприклад, у платформі може бути жорстко задана модель "проєкт → завдання → підзавдання", тоді як підприємство використовує іншу ієрархію (наприклад, за напрямками, клієнтами або зонами відповідальності). [39, с. 9-10]

Це призводить до необхідності «обгинати» робочі процеси під платформу, а не навпаки. У результаті - або відбувається вимушена трансформація внутрішньої структури (що викликає супротив у персоналу), або інструмент залишається напіввикористаним. [39, с. 10]

2. Обмеження кастомізації

Багато платформ дозволяють лише поверхневе налаштування - зміну кольорів, структури полів або шаблонів. Але якщо компанія має власну специфіку - наприклад, працює за унікальною схемою погодження або має внутрішню термінологію - виникає потреба в глибшому втручанні. У SaaS-продуктах така можливість або повністю відсутня, або потребує звернення до технічної підтримки, часто - платної, з обмеженим часом реагування. [40]

Таким чином, логічно, що компанії змушені створювати окремі інструменти (боти, скрипти, проміжні API-зв’язки), щоб компенсувати відсутність необхідної функціональності. А це в свою чергу підвищить вартість володіння системою та додасть складності в адмініструванні та технічному підтримуванні.

3. Конфіденційність та контроль над даними

Питання безпеки в корпоративному середовищі - одне з найбільш критичних. Багато компаній, особливо ті, що працюють з персональними даними клієнтів, конфіденційними фінансовими документами або розробками, мають внутрішні політики, які прямо забороняють зберігати інформацію на сторонніх серверах. [41, с. 4-5]

Тим часом більшість готових рішень працюють саме за моделлю хмарного

зберігання, з використанням дата-центрів за межами країни. Навіть якщо юридично це допустимо, такі обмеження можуть створювати формальні підстави для відмови від впровадження. [41, с. 4-5]

4. Низький рівень складності адаптації персоналу

Великі підприємства рідко є гомогенними в контексті цифрової компетентності. У межах однієї організації можуть працювати як технічно грамотні спеціалісти, так і працівники, що мають лише базові навички користування ПК. Для таких користувачів - перенасичений інтерфейс або складна логіка платформи може бути серйозною перешкодою, і в результаті - гальмується впровадження. виникає неосознаний саботаж або пасивний спротив, зростає навантаження на технічну підтримку. [42, с. 17]

Навіть найкраща система втрачає сенс, якщо її не використовують ефективно. Парадоксально, але часто саме «зайва складність» стає причиною відмови від інструменту через кілька місяців після запуску. [42, с. 18-19]

Тому, згідно всіх попередніх пунктів – можна зробити висновок, що готові системи управління завданнями безперечно мають свою нішу і часто є ефективним інструментом для команд до 20–30 осіб. Однак перед масштабним використанням в корпоративному середовищі вони потребують ретельного аналізу, адаптації й технічної підтримки, що підвищує складність та вартість володіння.

Ці фактори пояснюють, чому багато компаній - особливо з високими вимогами до безпеки, автономності та гнучкості - все частіше розглядають альтернативу у вигляді власноруч розроблених або open-source рішень, які можна адаптувати під внутрішню логіку без обмежень постачальника. Такий підхід дозволяє зменшити залежність від зовнішніх сервісів, покращити контроль над даними і в довгостроковій перспективі - значно знизити експлуатаційні витрати.

2.3 Аналіз типових недоліків SaaS-підходу

Модель програмного забезпечення як послуги (Software as a Service, SaaS), безперечно, змінила правила гри на ринку цифрових продуктів. Вона дозволила

компаніям позбутися витрат на інфраструктуру, отримати швидкий доступ до функціональних сервісів, а розробникам - будувати масштабовані бізнеси на регулярних підписках. Проте за цією зручністю приховуються обмеження, які дедалі більше критикуються. Особливо - з боку підприємств, що працюють у сегментах середнього та великого бізнесу. [43, с. 145]

Далі розглянемо ключові недоліки SaaS-підходу, які варто враховувати при ухваленні рішення щодо вибору платформи для корпоративного управління проєктами та колаборації.

1. Відсутність повного контролю над системою

Найбільш очевидний і, фундаментальний недолік SaaS-рішень - це відсутність контролю над самим програмним середовищем. Користувач отримує лише доступ до обмеженого інтерфейсу, у межах якого йому дозволено взаємодіяти з системою, а усі інші аспекти - зберігання даних, обробка запитів, логіка роботи бекенду - лишаються закритими. Це створює низку проблем, а саме - неможливість внести зміни у бізнес-логіку без погодження з постачальником (і зазвичай постачальник не погодиться, або “відморозиться”), відсутність доступу до повного аудиту системи, неможливість інспекції або вдосконалення алгоритмів, що опрацьовують критично важливу інформацію. [36, с. 320-321]

У випадку корпоративного програмного забезпечення це особливо болюче, адже багато бізнес-процесів є унікальними, а типові сценарії, передбачені розробником платформи, можуть частково або повністю не відповідати реальним потребам.

2. Вендер-локінг (vendor lock-in)

Однією з найменш очевидних, але надзвичайно небезпечних пасток SaaS-підходу є явище «vendor lock-in» - тобто прив’язка до конкретного постачальника. Варто лише поглянути на типову ситуацію: компанія кілька років використовує SaaS-платформу, накопичує у ній терабайти даних, формує внутрішню культуру взаємодії на її основі, та раптом провайдер - підвищує ціни, або змінює політику зберігання даних, або обмежує функціональність у базовому плані. або - що найгірше - припиняє підтримку чи зникає з ринку. [44, с. 92-93]

У такій ситуації бізнес втрачає гнучкість. Міграція на іншу платформу стає довгою, складною, та часто просто дорогою. Замість того, щоб бути господарем власних процесів, компанія опиняється у позиції залежного клієнта, змушеного приймати будь-які умови, що диктує сервіс. [44, с. 93]

3. Ризики конфіденційності та захисту даних

Попри те, що більшість відомих SaaS-провайдерів декларують високі стандарти безпеки (наприклад, відповідність вимогам SOC 2, ISO 27001), використання сторонніх серверів для зберігання корпоративної інформації завжди пов’язане з певним ризиком, адже у випадку витоку, зловживання внутрішнім доступом або кібератаки відповідальність - як юридична, так і репутаційна - лягає передусім на компанію, а не на розробника платформи. Особливо небезпечним це є в галузях, де обробляються персональні дані (HR, медицина), фінансова інформація (банківська сфера, аудит) або комерційні секрети. [45, с. 635-636; 46]

Питання захисту даних стає ще більш чутливим у міжнародному контексті - наприклад, коли сервери розташовані в юрисдикції, що не гарантує відповідного рівня правової захищеності (це стосується як країн із надто лояльними до державного втручання законами, так і таких, де контроль відсутній узагалі). [45, с. 635-636]

4. Обмеження у кастомізації функціоналу

Ще один системний недолік SaaS - це його уніфікована природа. Продукти розробляються з розрахунку на «середнього» користувача: інтерфейс, набір функцій, обмеження - усе адаптоване під максимально широке коло споживачів [2 с. 38]. Але в реальному бізнесі майже не буває "типових" сценаріїв [38, с. 180].

У корпоративному середовищі, де часто необхідна інтеграція з внутрішніми сервісами (ERP, CRM, фінансовими системами), така негнучкість стає критичною.

5. Вартість, яка зростає непропорційно до використання

На перший погляд, SaaS-сервіси виглядають економічно привабливими: відсутність витрат на серверне обладнання, адміністрування, оновлення ПЗ. Проте ця перевага часто виявляється ілюзорною, особливо на довготривалій дистанці, адже модель щомісячної або щорічної підписки передбачає регулярну оплату навіть у ті

періоди, коли використання функціоналу мінімальне. Із зростанням кількості співробітників або проєктів, витрати зростають експоненційно - вартість розраховується не за обсягом фактичного використання, а за кількістю користувачів або модулів, навіть якщо частина з них активна лише епізодично. [6]

Більш того, доступ до критично важливого функціоналу (звіти, ролі, безпека, інтергації з API) часто відкривається лише в межах вищих тарифних планів, які можуть коштувати в рази більше базового пакету. Таким чином, компанія поступово виявляє себе у фінансовій пастці: або працювати з обмеженим функціоналом, або постійно нарощувати витрати без гарантії ефективної віддачі. [44, с. 95]

6. Залежність від зовнішніх факторів

Ще один недооцінений аспект SaaS-моделі - її залежність від зовнішніх умов, зокрема:

Якість інтернет-з’єднання - у разі відсутності доступу до мережі (наприклад, через технічні збої, кібератаки, регіональні обмеження) - доступ до системи в офісі буде неможливий.

Доступність серверів постачальника - хоча більшість провайдерів декларують високу відмовостійкість, історія знає чимало випадків тривалих збоїв навіть у великих компаніях - AWS, Microsoft, Google. [47; 48]

Юридичні та політичні ризики - у випадку зміни законодавства або санкційної політики (як це сталося, наприклад, з низкою продуктів у РФ або Ірані), компанія може втратити доступ до інструменту, на який покладалась у щоденній роботі. [49]

На відміну від локальних або автономних open-source рішень, SaaS-продукти не гарантують безперервної роботи за будь-яких умов, і цей фактор необхідно враховувати, особливо в галузях, де операційна безперервність має першочергове значення.

7. Нестача прозорості й залежність від внутрішніх рішень провайдера

Навіть за умови повної стабільності платформи, користувач SaaS-сервісу залишається позбавленим впливу на її розвиток. Рішення про додавання чи видалення функціоналу, зміни інтерфейсу, структури меню, логіки автоматизації ухвалюються виключно розробниками, на основі комерційних або маркетингових

міркувань.

У результаті навіть добре налагоджені робочі процеси можуть порушитися після чергового «оновлення», до якого бізнес не був готовий. Такі зміни часто відбуваються без детального попередження або з обмеженою можливістю повернення до попередньої версії. У складних середовищах це викликає ланцюгову реакцію: [44, с. 1-2]

- Потреба у повторному навчанні персоналу.
- Порушення вже встановленої документації та регламентів.
- Несумісність з кастомними елементами, створеними під стару логіку.

Таким чином, компанія змушена постійно підлаштовуватись під платформу, втрачаючи частину контролю над власними процесами.

Таким чином аналіз показує, що попри зручність, швидкий старт та мінімальні технічні вимоги, SaaS-модель має серйозні системні обмеження, особливо для компаній, які цінують гнучкість, безпеку, автономність і довготривалу передбачуваність своєї інфраструктури.

Підхід «один продукт для всіх» працює ефективно лише в межах невеликих команд або короткострокових проєктів. Коли ж мова йде про повноцінну цифрову екосистему підприємства, з індивідуальними процесами, багаторівневою ієрархією, специфічними вимогами до захисту даних - використання SaaS може перетворитися з рішення на проблему.

На цьому фоні - альтернативні підходи, зокрема, відкриті, самостійно впроваджувані рішення з відкритим кодом - виглядають дедалі привабливішими. Вони не тільки дозволяють досягти повного контролю над функціональністю, але й створюють можливості для масштабування, модульної побудови системи, гнучкого налаштування, локального зберігання даних і мінімізації залежності від зовнішніх факторів. [8, с. 10-11]

Таким чином, SaaS - це інструмент. Але, як і будь-який інструмент, він повинен бути застосований у відповідному контексті. І якщо цей контекст - складне корпоративне середовище з високими вимогами до гнучкості й безпеки - SaaS часто

виявляється недостатнім.

2.4 Переваги open-source підходу в контексті безпеки та гнучкості

Open-source - це не лише філософія доступності та прозорості. У контексті корпоративного використання - особливо в сегменті інструментів управління проєктами та колаборації, це стратегічний вибір, який прямо впливає на безпеку, гнучкість, стабільність та довгострокову незалежність від зовнішніх провайдерів [8, с. 4-5, 18-19]. У цьому підрозділі розглянемо ключові переваги open-source-підходу, що роблять його дедалі актуальнішим у сучасному бізнес-середовищі.

1. Повний контроль над системою

Однією з найважливіших переваг відкритого коду є можливість повністю контролювати весь життєвий цикл програмного продукту - від інсталяції до масштабування, від кастомізації до архівування. Компанія більше не залежить від політики чи «настроїв» стороннього постачальника: усі файли, конфігурації, бази даних знаходяться у її безпосередньому розпорядженні. [40]

Цей контроль означає - можливість редагувати логіку системи під власні потреби, реалізацію внутрішніх сценаріїв роботи, які просто недоступні в SaaS-моделях. зміну або доповнення функцій без обмежень, оновлення - лише тоді, коли це зручно, а не коли цього вимагає постачальник [18, с. 216].

У корпоративному світі, де стабільність і передбачуваність - фактори не менш важливі, ніж інновації, такий контроль має фундаментальне значення.

2. Безпека та автономність даних

На відміну від SaaS-платформ, у open-source-системах можна самостійно визначати, де й як зберігатимуться дані [1, с. 2492]. Якщо компанія розміщує сервер на власній інфраструктурі - то це означає: повну автономність від сторонніх дата-центрів, локальне шифрування під власним контролем, обмеження зовнішнього доступу до чутливої інформації, підвищену відповідальність за резервне копіювання - але і повну гнучкість у його реалізації, та багато інших переваг.

У разі критичної ситуації - зникнення мережі, блокування IP-адрес, виходу з

ладу хмарних платформ - система може продовжувати працювати в локальному середовищі. Це особливо важливо для державних установ, банківських організацій, медичних закладів та інших структур з підвищеними вимогами до конфіденційності та стабільності.

3. Відсутність vendor lock-in

Open-source - це фактично антипод концепції "vendor lock-in". Рішення, побудоване на відкритому коді, не прив’язує компанію до конкретного постачальника або платформи. У будь-який момент: можна змінити хостинг, делегувати підтримку іншій команді або найняти розробників з “третього” підприємтсва, розробити додаткові модулі на власний розсуд, інтегрувати систему з іншими продуктами без обмежень API чи ліцензійних бар’єрів. [50, с. 2-3]

У стратегічній перспективі це дає надзвичайно цінну технологічну незалежність, яка особливо важлива у часи швидкоплинних змін - як на ринку ПЗ, так і у сфері законодавчого регулювання.

4. Гнучкість у розробці та кастомізації

Жоден готовий SaaS-продукт не зможе забезпечити такого рівня адаптації, як open-source рішення. Мова не лише про можливість змінити логотип, кольорову схему чи набір полів у формі. Йдеться про: модифікацію структури бази даних, переписування контролерів і сервісних об’єктів, зміну архітектури модулів, додавання нестандартних сценаріїв роботи, включно з інтеграцією з фізичними пристроями, локальними API чи навіть зовнішніми CRM, в силу того, що потенційний розробки чи користувач, по значенню “відкрите джерело” — має доступ читати та змінювати код в будь якому елементи програмного забезпечення.

Така гнучкість дозволяє побудувати систему, що ідеально лягає на наявні бізнес-процеси, а не змушує адаптувати організацію під продукт.

5. Сприяння внутрішньому розвитку команди

Інший, можливо неочевидний, але надзвичайно цінний аспект open-source підходу - це розвиток внутрішньої IT-команди. Компанії, що впроваджують власне open-source-рішення, фактично створюють простір для зростання своїх спеціалістів: DevOps-інженери працюють із реальними задачами деплою, CI/CD, бекенд-

розробники - з бізнес-логікою і налаштуванням контролерів, аналітики - з архітектурою структури даних і запитами до бази, дизайнери - з UI/UX у власному стилі, без обмежень фреймворку.

На відміну від готового продукту, де є лише можливість «натиснути кнопку» або «змінити налаштування», open-source - це відкритий простір для глибокої інженерної роботи.

6. Прозорість та спільнота

Коли вихідний код системи відкритий - його можуть переглядати, тестувати, обговорювати тисячі людей по всьому світу [51, с. 2-3]. У результаті:

- Швидше виявляються баги та вразливості.
- Оновлення часто надходять ззовні - від учасників спільноти.
- Можна скористатись готовими рішеннями, опублікованими іншими користувачами.
- Наявна незалежна експертиза, що підвищує довіру до продукту.

Такі переваги особливо важливі у складних корпоративних системах, де стабільність і безпека мають першорядне значення. Прозорість стає не просто перевагою, а запорукою довіри до продукту, який можна перевірити самостійно, а не сподіватися на закриту техпідтримку. [51, с. 22-23]

7. Економічна ефективність у довгостроковій перспективі

Поширеним упередженням є те, що open-source рішення автоматично означає «безкоштовно». Це не зовсім так. Хоча базовий код доступний вільно, впровадження потребує витрат: на інфраструктуру, підтримку, налаштування, розробку додаткового функціоналу. Проте принципова відмінність полягає в структурі цих витрат.

У SaaS-моделі основна частина витрат - це підписка, яка повторюється щомісячно, а отже, має кумулятивний характер [43, с. 142-143]. У разі open-source – очевидно, що перші інвестиції вищі (налаштування, запуск), але відсутні регулярні витрати на ліцензії. Зростання користувачів не тягне за собою автоматичне зростання вартості. З часом підтримка стає внутрішньою компетенцією компанії, що знижує залежність від сторонніх послуг.

8. Можливість створення конкурентної переваги

І нарешті, ще один, логічно виходящий з попередньї інформації фактор - можливість перетворити внутрішню систему на інструмент стратегічної переваги. Open-source дає не лише контроль і гнучкість, а й можливість створення унікального рішення, яке неможливо скопіювати чи повторити «з коробки». [18, с. 216]

Це дозволяє:

- Автоматизувати внутрішні процеси до рівня, недосяжного у готових продуктах.
- Створити оригінальні інтеграції, що забезпечують більш плавну взаємодію систем.
- Демонструвати прозорість і технологічну компетентність перед партнерами або клієнтами.

У певних галузях (наприклад, консалтинг, IT, інжиніринг) власна open-source-система може навіть стати частиною комерційної пропозиції - як показник незалежності, експертизи та готовності до глибокої кастомізації під клієнта.

Виходячи з вище перечисленої інформації, можна дійти висновку, що програмне забезпечення з відкритим вихідним кодом (open-source) у корпоративному середовищі - це вже не тільки прояв ідеалізму чи прихильності до відкритих технологій. Це зважений, стратегічний вибір, який забезпечує контроль над функціональністю, гарантію безпеки та локального зберігання даних, гнучкість у розробці та інтеграціях, незалежність від сторонніх провайдерів, економічну доцільність у довгостроковій перспективі.

Зрозуміло, open-source не є панацеєю. Він вимагає відповідального підходу, готовності до глибшого залучення IT-команди, певного технічного рівня всередині компанії. Проте для організацій, що прагнуть, або потребують стабільного, контрольованого та гнучкого середовища для управління процесами та колаборацією - це рішення, яке не лише відповідає поточним потребам, а й формує стійкий технологічний фундамент на майбутнє.

2.5 Обґрунтування вибору технологічного стеку VILT

Одним із ключових етапів створення будь-якої сучасної веб-системи є вибір технологічного стеку - сукупності мов програмування, фреймворків, бібліотек і допоміжних інструментів, на яких ґрунтується архітектура продукту. Правильний вибір у випадку даного продукту є важливим задля того, щоб якомога легше та швидше ІТ-департаменти компаній змогли почати роботу з кастомізації продукту, без лишнього часу на навчання.

У процесі проєктування власної корпоративної open-source платформи менеджменту та колаборації, мною було вирішено будувати систему на основі технологічного стеку VILT (Vue.js + Inertia.js + Laravel + Tailwind CSS). Цей підхід обрано не випадково: він забезпечує як високу швидкість розробки, так і технологічну зрілість на рівні продуктивних корпоративних рішень.

Технологічний стек VILT є сучасним поєднанням засобів розробки повнофункціональних веб-застосунків, де:

- Vue.js відповідає за побудову інтерфейсу (frontend). [21]
- Inertia.js забезпечує взаємодію між клієнтською та серверною частинами без необхідності повноцінного REST API. [22]
- Laravel є дуже популярною серверною платформою, яка реалізує логіку застосунку, обробку запитів, роботу з базами даних. [23]
- Tailwind CSS використовується як гнучкий утилітарний CSS-фреймворк для побудови адаптивного дизайну. [7]

Поєднання цих компонентів дозволяє створити монолітну, але реактивну веб-систему, що зберігає продуктивність класичного сервера, але виглядає й працює як сучасний SPA (Single Page Application). [52, с. 1-2]

Vue.js - це сучасний JavaScript-фреймворк, який поєднує гнучкість у розробці з продуктивністю та масштабованістю. Цей фреймоврк надає численні переваги - легкість навчання та читабельність коду (це дозволяє забезпечити швидке залучення нових учасників до розробки), декларативна побудова інтерфейсу - компонентна архітектура Vue сприяє модульності, що особливо важливо при проєктуванні

складних систем з багатьма сутностями, підтримка реактивності - зміни даних миттєво відображаються на екрані, що підвищує зручність розробки системи з боку розробника (більше не потрібно вручну лізти в DOM), широка екосистема - плагіни, бібліотеки, інтеграції - усе це пришвидшує розробку і розширює функціональність без надмірного навантаження на команду [21].

Крім того, Vue.js добре масштабується - від невеликих панелей управління до повномасштабних корпоративних систем.

Одним із інноваційних елементів у стеку VILT є використання Inertia.js. Це популярна бібліотека, яка дозволяє створювати динамічні інтерфейси без необхідності реалізації повноцінного REST або GraphQL API [22].

Замість традиційного клієнт-серверного розмежування, Inertia дозволяє використовувати серверні маршрути Laravel для передачі сторінок Vue у вигляді компонентів, що забезпечує скорочення часу розробки, оскільки відпадає потреба у дублюванні логіки в API, Кращу продуктивність, оскільки запити обробляються через внутрішні механізми Laravel, простоту дебагінгу та тестування, бо розробник працює в єдиному просторі логіки. [22]

Laravel - один з найпопулярніших та найсучасніших PHP-фреймворків, який вирізняється сучасним підходом до архітектури, безпекою, документацією та активною спільнотою. Саме Laravel реалізує логіку бізнес-процесів у розроблюваній системі [23]:

- Чітке MVC-структурування коду забезпечує організованість [24].
- Вбудовані механізми авторизації, валідації, обробки запитів знижують потребу у власних реалізаціях.
- Міграції та сидери дають змогу швидко розгортати БД у новому середовищі.
- Eloquent ORM дозволяє працювати з базою даних у максимально зручній, об’єктно-орієнтованій формі.
- Підтримка подій, черг, задач, логування, що є критично важливим для багатокористувацьких систем.

Використання Tailwind CSS як базового інструменту для верстки дозволяє

швидко розробляти верстку, проте, він не є корінним елементом системи, і за бажанням розробника – може бути замінений на інші стилістичні фреймоврки CSS. [7]

Варто зазначити, що вибір стеку VILT не був прийнятий імпульсивно або виключно на основі зручності. На початкових етапах розробки були також розглянуті інші популярні варіанти, зокрема:

- MERN (MongoDB, Express.js, React, Node.js) - привабливий варіант з високою популярністю у спільноті JavaScript, але вимагає повного занурення у односторінкову архітектуру, має вищу складність налаштування бекенду, особливо для комплексної системи з розвиненою авторизацією та ролями, а також має дуже мало “функціоналу за замовчуванням (на відміну від Laravel)” - що суттєво сповільнило б як початкову розробку автором даної курсової роботи, так і наступні покращення користувачами/розробниками такої системи. [53]
- Laravel + Blade (класичний підхід) - менш динамічний у візуальній взаємодії з користувачем, потребує повного перезавантаження сторінки при кожній дії, що вже не відповідає очікуванням сучасного UX. [54]
- Vue + REST API - традиційний спосіб побудови SPA, який передбачає створення повноцінного API між Vue і Laravel. Такий підхід більш масштабований і в деякій мірі можливо і кращий з технічної точки зору, але потребує подвоєної логіки обробки даних і значно ускладнює та затримує розробку та тестування. (Що погано сказалось би як на початковій розробці даної системи, так і в майбутньому для клієнта через більшу потенційну самовартість впроваджень поліпшень та змін).

У результаті аналізу було вирішено - що VILT це оптимальне рішення для розробки системи-об’єкту даної роботи, яка повинна відповідати всім вимогам викладеним в попередніх розділах.

Особисто я мав досвід роботи з Laravel і Vue у попередніх навчальних і позанавчальних проєктах. Саме тому рішення обрати знайомі інструменти було не лише практичним, а й стратегічно доцільним: це дозволило сконцентруватися на

реалізації бізнес-логіки, а не на подоланні технічних труднощів, пов’язаних з вивченням нових технологій.

Важливо зазначити, що усі компоненти стеку - з відкритим вихідним кодом, активною спільнотою, потужною документацією. Це особливо важливо у контексті побудови open-source продукту, адже забезпечує його доступність, зрозумілість для інших розробників, а отже - потенційну підтримку, удосконалення та адаптацію третіми сторонами.

Також додатково було вирішено використовувати: TypeScript для фронтенд-частин, адже він забезпечує типізацію, що значно знижує кількість помилок під час розробки і полегшує подальшу підтримку коду [55, с. 2-5], Eloquent ORM – систему, як активно застосовує складні зв’язки між моделями (проекти, задачі, користувачі), що спрощує обробку даних та робить код більш підтримуваним.

Вплані UX – було вирішено зразу вшити підтримку темної теми, що відповідає сучасним очікуванням від веб-інтерфейсу.

Отже, вибір стеку VILT (Vue.js, Inertia.js, Laravel, Tailwind CSS) у процесі розробки корпоративної open-source системи управління менеджментом і колаборацією є цілком обґрунтованим як з інженерної, так і з прикладної точки зору. Він забезпечує баланс між швидкістю розробки, зручністю в обслуговуванні, гнучкістю кастомізації та сучасним користувацьким досвідом.

Технології, обрані для реалізації, є відомими, перевіреними і водночас відкритими, що відповідає і філософії open-source, і технічним вимогам до проєкту. Їх використання дало змогу створити продукт, який одночасно легко впровадити. просто підтримувати і, що найважливіше, - адаптувати до реальних потреб конкретної компанії без будь-яких ліцензійних або технологічних обмежень.

РОЗДІЛ 3. РОЗРОБКА КОРПОРАТИВНОЇ СИСТЕМИ КОЛАБОРАЦІЇ ТА МЕНЕДЖМЕНТУ

3.1 Архітектура та модульність створеного продукту

Після вибору технологічного стеку наступним етапом у розробці корпоративної онлайн-системи менеджменту та колаборації стало проєктування її архітектури. Враховуючи орієнтацію на open-source підхід і можливість масштабування, було прийнято рішення створити систему модульного типу, де кожен функціональний блок розробляється як відносно незалежна сутність. Такий підхід дозволить, з одного боку, легко адаптувати систему під потенційно нові потреби конкретного підприємства, а з іншого - розвивати її поступово, не порушуючи загальної логіки платформи.

В основу архітектури покладено чітку декомпозицію функцій, типову для систем управління завданнями: користувачі, проєкти, завдання, коментарі, доступи, тощо. Основний акцент зроблено не на кількості функцій, а на якісному розділенні відповідальностей між модулями [25].

Загалом програмний продукт складається з наступних основних модулів/систем/частин: користувачі та система ролей/доступу (ACL), проєкти, завдання, логи часу, коментарі, прості нотифікації, вебхуки

Кожен модуль та системи є логічно завершеними, мають власну модель(-і) у базі даних (завдяки ORM Eloquent, що забезпечує незалежність від СУБД), відповідні контролери, маршрути та Vue-компоненти для візуалізації, тощо.

- Модуль користувачів та автентифікації

У даній реалізації користувачі зберігаються у стандартній таблиці users, яка розширена додатковими полями (наприклад, name, role_id, timestamps, тощо). Кожен користувач має роль, яка визначає його рівень доступу до функцій системи.

За замовчуванням - існує 4 ролі – суперадмін (зі всіма правами), проджект-менеджер (з правами створення/читання/оновлення/видалення (CRUD) завдань,

проєктів (лише своїх, або тих, до яких його додали), а також логувати та коментувати), розробник (з правами бачити завдання/проєкти до яких його додали, додавати логи та коментарі), та клієнт (з правами читати проєкти/завдання до яких його додали, та коментувати завдання).

Але також можливо створювати користувацькі ролі, з будь якою назвою та набором прав. Право створювати ролі – це теж окреме налаштування (або надалі – “право”) ролі, яке за замовчуванням є лише у суперадміністратора.

Ролі зберігаються окремо (roles), а зв’язок реалізовано через foreign key в модулі users (role_id). Усі перевірки доступу здійснюються на рівні контролерів, або Policy, що є вбудованою простою системою безпеки у Laravel.

- Модуль проєктів

Проєкт - це ключова бізнес-одиниця в даній системі. Він об’єднує завдання, учасників, та все “між тим”, хоча й сама структура проєкту проста (наявні ліше id, назва, статус, та власник)

Кожен проєкт має свою внутрішню ієрархію завдань, що реалізовано через зв’язки один-до-багатьох (project_id поле у таблиці tasks). Також кожен проєкт має список учасників (project_user таблиця), що дозволяє реалізувати модель доступу до даних у межах конкретного проєкту.

- Модуль завдань

Завдання є структурною одиницею менеджменту. Кожне завдання може бути прикріпленим до проєкту, призначеним одному або кільком користувачам, містити статус, пріоритет, опис, коментарі, тощо.

Для зручності фронтенду всі завдання реалізовані як Vue-компоненти, що динамічно оновлюються за допомогою реактивності, а зміни вносяться через Ajax-запити без перезавантаження сторінки.

Також фронт-енд “реагує” на права користувача, то показує йому можливість змінити назву/опис/т.п. прямо на сторінці, за умови – що у такого користувача є на це право.

- Модуль логів часу

Теж дуже важливою частиною системи є (опціональне) логування часу

працівниками, що дозволяє надалі визначити продуктивність працівників, оцінювати майбутні схожі завдання, та/або передавати ці логи клієнтам, якщо, умовно, компанія займається розробкою ПЗ для клієнтів, з оплатою “погодинно”.

- Модуль коментарів

Кожне завдання підтримує обговорення у вигляді коментарів. Коментарі зберігаються в таблиці task_comments і мають зв’язок із завданням (task_id) і автором (user_id).

- Нотифікації

У системі реалізовано базову модель сповіщень, де при завантаженні дешборд-сторінки – також сервером будуть передані останні зміни в завданнях/проєктах, сповіщення про новий коментар/завдання, додання до проєкту, тощо.

- Модуль вебхуків

Важливою частиною цієї системи із самого початку проектування – була підтримка вебхуків, для легкої інтеграції з сторонніми сервісами.

Загальна логіка взаємодії модулів

Усі модулі взаємодіють між собою через взаємозв’язки у базі даних (реляційна модель) та відповідні API через Inertia.js.

Проєктна архітектура системи базується на принципах модульності, гнучкості та простоти розширення, що є ключовими вимогами для сучасних корпоративних онлайн-систем. Кожен модуль реалізовано у спосіб, який дозволяє: автономно підтримувати його, інтегрувати з іншими модулями без дублювання логіки та легко масштабувати функціонал у подальших версіях системи.

Цей підхід дозволив не просто створити “прототип”, а закласти фундамент для повноцінного продукту, який здатен конкурувати з SaaS-рішеннями, залишаючись при цьому відкритим, гнучким та керованим з боку самої компанії-кінцевого користувача.

3.2 Процес розробки та реалізації

Було створено новий git-репозиторій, та новий Laravel проєкт. Першим етапом було створення всіх моделей, міграцій, схем баз даних, й так далі всіх модулів, описаних в пункті 3.1.

Почато розробляти Laravel проєкт було використовуючи Breeze Startup Kit, який дозволяє швидко та просто почати новий проєкт, маючи готову базову автентифікацію, функціональність скидування паролю, підтвердження email, тощо. [27]

Після чого, спочатку інтегрувано inertia.js, а після – почата розробка саме фронт-енд частини. Було вирішено виділити по 3 сторінки (створення, перегляд конкретної моделі, та перегляд всіх моделей) для двох найголовніших модулів – проєкти та завдання. Зміна та видалення моделей – було вирішено реалізувати прямо на сторінці перегляду моделі.

Спочатку було створено сторінки перегляду та вибору всіх проєктів (див. Рис. 1) та сторінку перегляду деталей та редагування проєкту (див. Рис. 2), в якій, за принципом розбиття на сторінки, автоматично завантажуються всі проєкти (або ті, які користувач може бачити зі своїми правами). Також було додано фільтри та поле пошуку, а також прості елементи інтерфейсу, такі як – рахувач всіх завдань, активних проєктів, тощо.

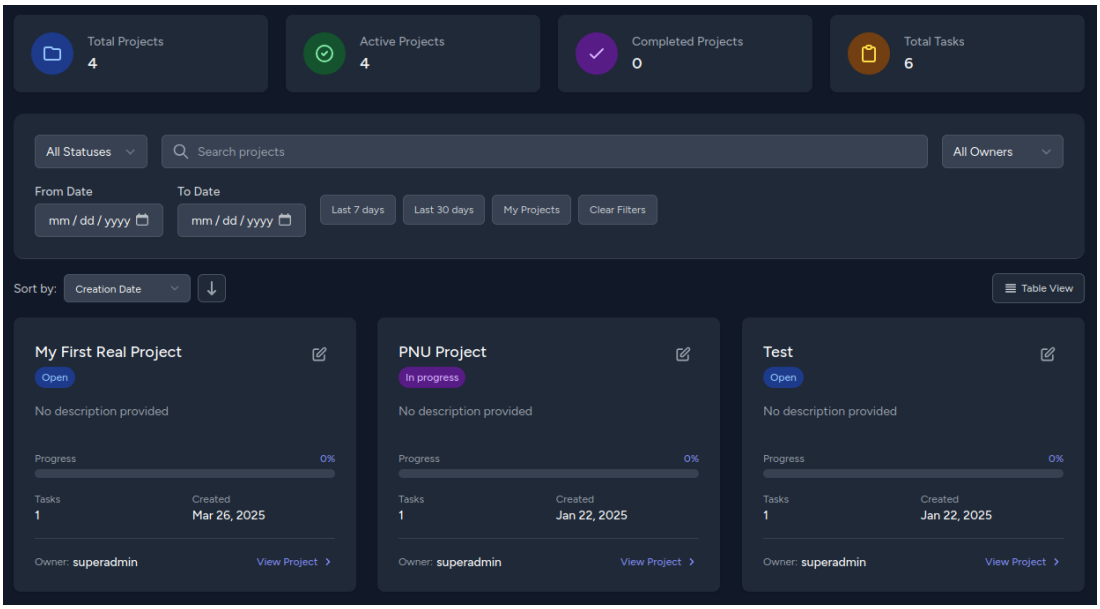

Рисунок 1 – Сторінка вибору проєктів

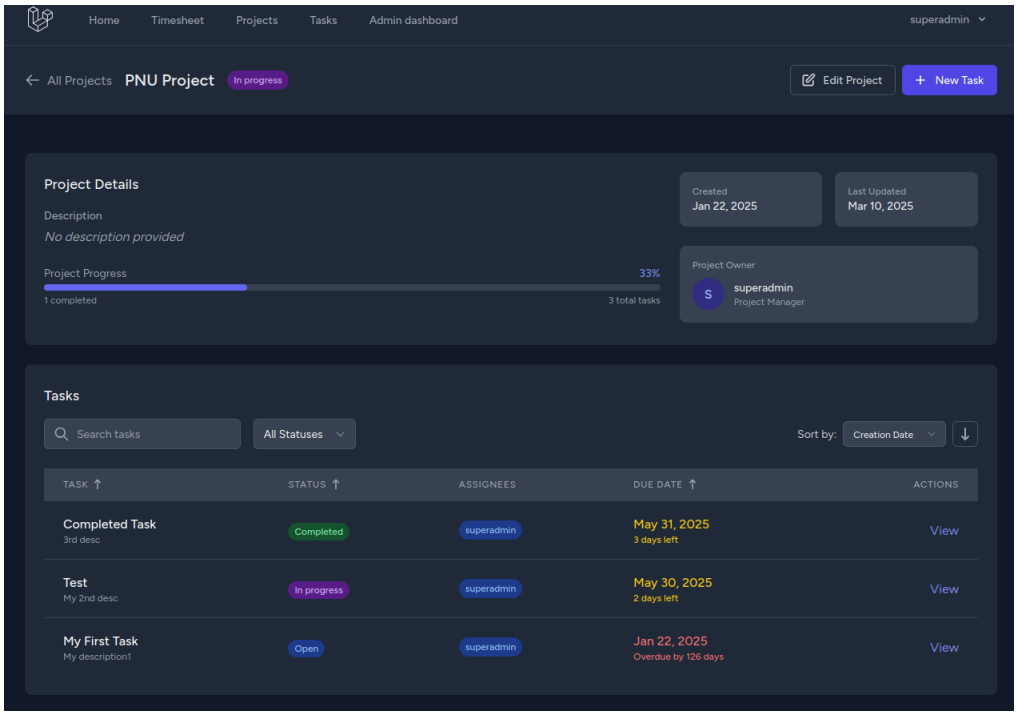

Рисунок 2 – Сторінка огляду окремого проєкту

Після цього – було в схожій манері створено сторінки масового (див. Рис. 3) та по-одинокого (див. Рис. 4) перегляду завдань.

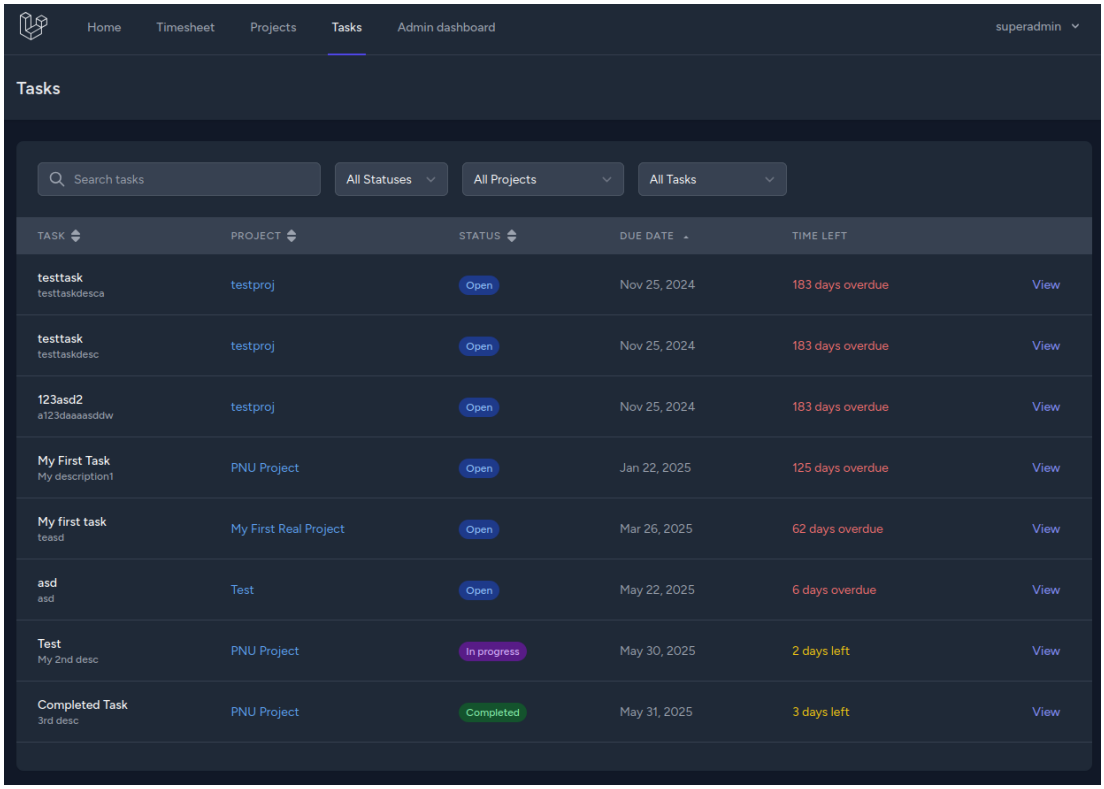

Рисунок 3 – Сторінка масового перегляду завдань.

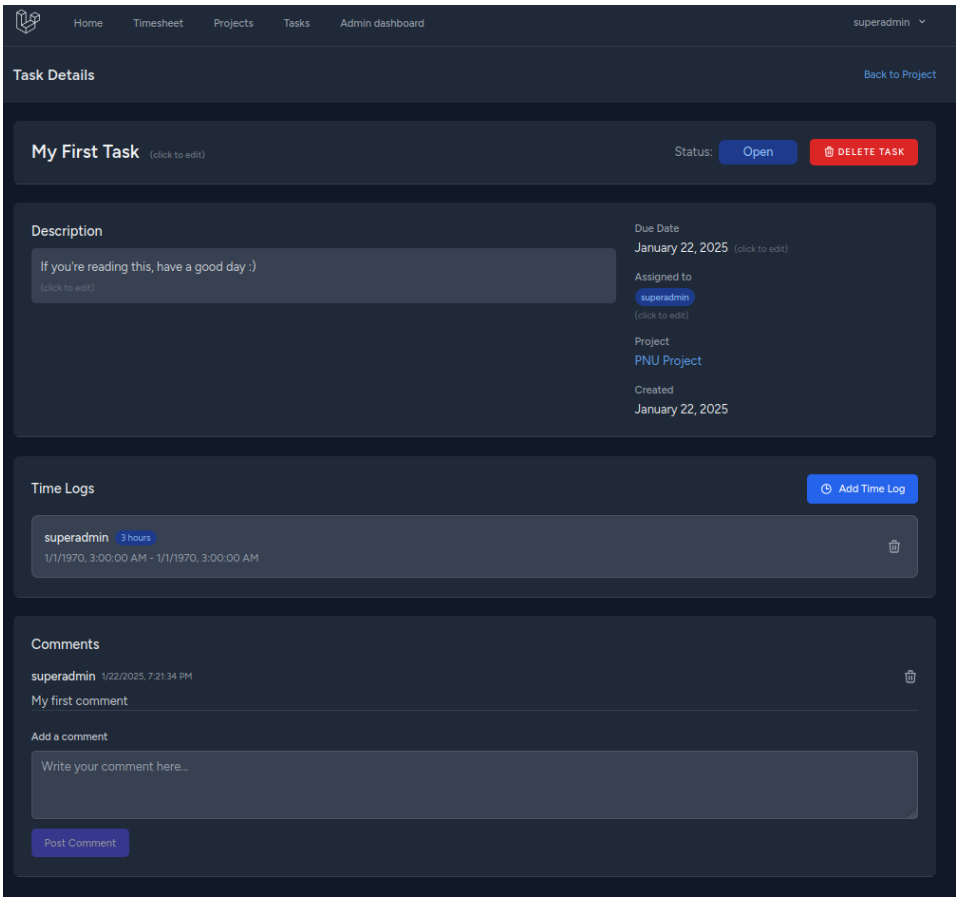

Рисунок 4 - Сторінка перегляду/редагування завдання

Саме на сторінці перегляду завдання, також було розміщено елементи таких модулів як “коментар”, “логи часу”, тощо. Також було вирішено не створювати окрему сторінку редаугвання завдання задля економії часу та розробки, та покращення досвіду користувача. Всі елементи, що можуть бути редаговані – можуть бути натиснуті користувачем прямо на сторінці, після чого замість них – в DOM створиться input елемент. Після уведення нових даних, та натисканні курсором поза input елементом – така інформація автоматично відобразиться як нова на сторінці, та відправиться на сервер, для збереження.

Також було створено “багато-функціональну” адмін-панель, де напряму можна б було одночасно – створити/редагувати/видалити/подивитись користувачів, ролі, вебхуки, загальні налаштування сайту, тощо.

Після завершення фронтенду/веб-інтерфейсу, та базового налаштування моделей – було розроблено повноцінну ACL систему, на основі ролей користувачів, яка працювала на стороні сервера, і відповідно – не дозволяла обійти обмеження шляхом маніпуляції фронтендом.

Вебхуки, як важлива частина системи - реалізовані в адмін панелі. Необхідно лишень обрати модель (проєкт, завдання, користувач, роль, лог, коментар), встановити HTTP-шлях, опціонально встановити Header’и, та надалі отримувати на цей HTTP-шлях сповіщення про події відповідної моделі. Тип події буде переданий в тілі HTTP запиту (створення/оновлення/видалення).

3.3 Функціональність системи: основні модулі та логіка роботи

Функціональність розробленої корпоративної системи онлайн-менеджменту та колаборації охоплює ключові потреби сучасної організації у сфері цифрового управління проєктами. Система побудована таким чином, щоб користувач мав змогу не лише виконувати конкретні задачі, а й відчувати логічну завершеність кожного етапу роботи: від постановки завдання - до його виконання.

Після входу в систему користувач потрапляє на інтерактивну домашню сторінку (див. Рис. 5), яка адаптується до ролі та прав доступу (як і інші сторінки). Менеджер – бачить активність своїх проєктів, виконавець – лише завдання, що потребують уваги.

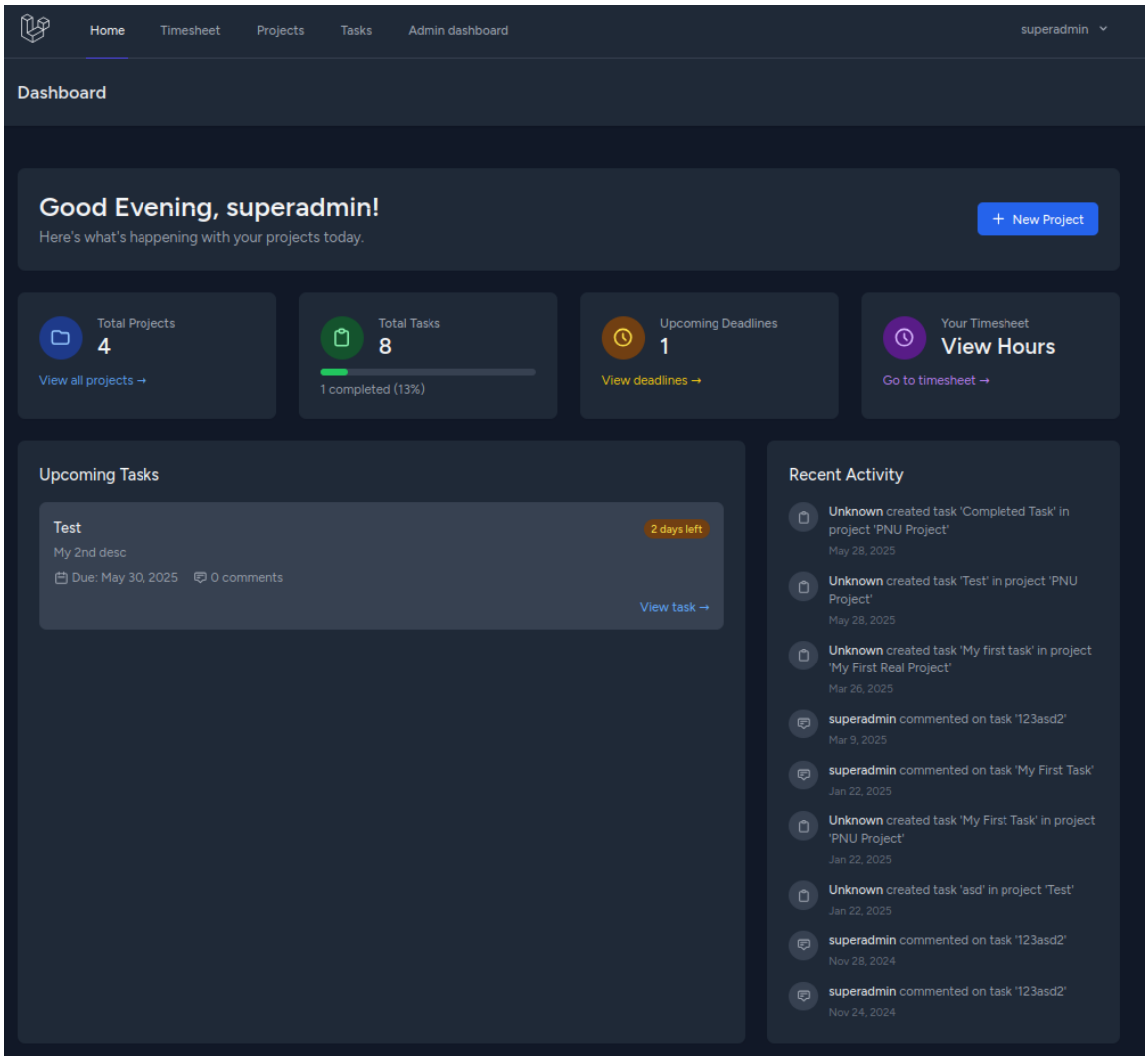

Рисунок 5 - Домашня сторінка

Процес авторизації реалізовано на основі стандартних механізмів Laravel (Laravel Breeze), з використанням хешування паролів через bcrypt, middleware-захистом маршрутів і CSRF-захистом форм. Після входу система відразу формує індивідуальний інформаційний простір, який включає: сьогоднішні завдання, зміни в проєктах, нагадування про дедлайни.

Це дозволяє уникнути «інформаційного шуму» і сфокусувати користувача на тому, що важливо саме зараз.

Користувач із роллю менеджера може створити новий проєкт, вказавши його назву, опис, відповідальних, тощо. Створення проєкту супроводжується додаванням учасників (через форму з автозаповненням), вибором статусу: активний, призупинений, завершений, та створенням перших завдань.

Інтерфейс проєкту містить кілька вкладок:

- Завдання (Tasks) - перелік усіх задач, фільтр за статусами, виконавцями, дедлайнами.
- Учасники - список із можливістю додавати/видаляти користувачів.
- Хронологія (Audit Trail) - список дій користувачів у межах проєкту.

Усі дані оновлюються асинхронно без перезавантаження сторінки, що забезпечує плавний UX навіть при роботі з великою кількістю елементів.

Кожне завдання створюється або безпосередньо через проект, або окремо, після чого прикріплюється до проєкту.

Основні параметри завдання це Заголовок, опис, дедлайн, відповідальний(-і), статус виконання (виконано, готово до тестування, на паузі, тощо).

Користувачі можуть коментувати завдання, змінювати їхній статус (відповідно до прав), позначати як виконані, делегувати іншим учасникам. Усі зміни логуються і зберігаються в історії, що дозволяє відслідковувати хід роботи та уникати непорозумінь у команді.

Обмін інформацією - одна з центральних функцій системи. Коментарі реалізовано як асинхронний чат, прив’язаний до кожного завдання.

Система сповіщень автоматично повідомляє про нові коментарі, зміну статусу задач, дедлайнів або нові призначення.

Кожен користувач може редагувати свій особистий профіль (ім’я, email, тощо), а також вибирати кольорову тему та мову сайту змінюючи налаштування власного браузера.

Інтерфейс адаптований до мобільних пристроїв - завдяки Tailwind CSS система коректно відображається на планшетах і смартфонах, що важливо для сучасного способу роботи «на ходу».

Завдяки адаптивному інтерфейсу та зрозумілим сценаріям дій, система практично не потребує додаткового навчання - логіка навігації інтуїтивно знайома технічно грамотному користувачеві, навіть якщо він уперше працює з такого типу програмами.

Функціональність системи побудована з урахуванням того, що в майбутньому вона може бути:

- розширена новими модулями (наприклад, календар, тайм-трекінг, інтеграція з поштою).
- локалізована під нові ринки - підтримка мультимовності вже закладена.
- переналаштована під різні типи організацій - наприклад, для IT-компанії, виробничої фірми або креативного агентства.
- інтегрована з іншими системами (ERP, CRM, SSO тощо).

Гнучка архітектура, описана в попередньому підрозділі, забезпечує можливість додавати або вимикати модулі без необхідності глобальної перебудови системи. Наприклад, можливо створити мікросервіс для роботи з документами або окремий API для додаткових сервісів.

Створена система реалізує ключові функції системи корпоративного менеджменту й колаборації - від управління задачами до організації командної взаємодії та звітності. Її функціональність відповідає базовим сценаріям корпоративної роботи та водночас залишає широке поле для адаптації під конкретні бізнес-процеси.

Особливої уваги заслуговує те, що функціонал не є перевантаженим: користувачеві не доводиться пробиратися крізь десятки кнопок і полів - кожен елемент інтерфейсу виконує чітко визначену функцію. Це дозволяє не лише ефективно працювати в межах платформи, а й - що не менш важливо - забезпечує прийняття системи користувачами без внутрішнього супротиву, що часто є ключовим чинником успіху IT-рішень у реальних компаніях, а також, завдяки open-source “природі” застосунку – відсутні всі так чи інакше “рекламні”, або “внутрішні” елементи, що також додатково навантажують практично всі наявні SaaS рішення аналогічних систем.

3.4 Перспективи масштабування та інтеграції системи у внутрішню інфраструктуру компаній

Однією з ключових переваг розробленої системи є її відкритість до подальшого розвитку. На відміну від «закритих» SaaS-продуктів, де користувач має справу з уже визначеним і часто жорстко зафіксованим функціоналом, власна система, побудована на відкритому коді, є гнучкою за своєю природою. Її можна масштабувати не лише технічно (в плані навантаження), а й функціонально - адаптуючи під специфіку будь-якої організації. У цьому підрозділі детально розгляну, як саме це можливо реалізувати, які напрями розвитку є найбільш перспективними, і яким чином система може бути інтегрована у вже наявні цифрові екосистеми компаній.

На рівні інфраструктури система вже зараз спроєктована так, аби витримувати зростання користувачів, проєктів, задач і запитів. Масштабування можливе у кількох напрямках.

Горизонтальне масштабування: перенесення системи на кластер серверів із балансуванням навантаження. Laravel та Vue.js це дозволяють завдяки підтримці контейнеризації (Docker) і сумісності з інструментами типу Nginx Load Balancer або Traefik.

Розділення сервісів: при потребі система може бути перепроєктована у мікросервісну архітектуру, де аналітика, повідомлення, керування задачами функціонуватимуть окремо - з підключенням через API або Message Queue (наприклад, RabbitMQ). Проте важливо зазначити що хоч така модифікація і є можливою, вона не є “очікуванною” для даної системи, тому впровадження такого масштабування потребуватимуме значної зміни системи. Потенційно – альтернативні “вилки” даної системи можуть бути розвинені паралельно, та підтримувати таку функціональність.

Завдяки використанню Laravel та Vue - фреймворків із широким промисловим застосуванням - немає обмежень щодо серверної архітектури, що дозволяє легко розміщувати систему на VPS чи в хмарі (AWS, DigitalOcean, Hetzner тощо).

Також система зараз реалізує набір базових функцій, які забезпечують ефективну командну роботу: проєкти, задачі, коментарі, доступи, нотифікації. Однак структура коду й архітектура дозволяють розширити її в таких напрямах:

- Тайм-трекінг - підрахунок часу, витраченого на кожну задачу, з подальшим експортом у вигляді табелів, які можна імпортувати в бухгалтерські чи HR-системи.
- Календарна синхронізація - двостороннє підключення до Google Calendar або Outlook, що дозволить відображати дедлайни й зустрічі в одному інтерфейсі.
- Інтеграція з системами документообігу - приклад: Nextcloud або Zoho WorkDrive - для зручної роботи з договорами, актами, ТЗ безпосередньо в межах задачі.
- Розширена аналітика - підключення зовнішніх BI-платформ (Metabase, Redash) або внутрішня реалізація KPI-дошки з оцінкою ефективності команди.
- Мобільний застосунок - через API або PWA-технологію (Progressive Web App), що зробить систему повністю доступною на смартфонах.

Кожен із цих напрямів може бути реалізований як окремий модуль – практично без втручання в базовий код системи, що забезпечує гнучкість у впровадженні нового функціоналу відповідно до бізнес-потреб та зменшує необхідність попереднього ознайомлення потенційних розробників із кодом системи.

В умовах, коли компанії вже використовують набір внутрішніх систем (CRM, ERP, бухгалтерію, корпоративну пошту тощо), критично важливо, щоб нова платформа могла інтегруватися без руйнування наявної цифрової екосистеми. У цьому сенсі розроблена система пропонує кілька очевидних переваг.

- Наявність основного REST API (або можливість швидко реалізувати додаткове), яке дозволяє підключати зовнішні системи, як-от Zoho CRM, 1C, Bitrix24 тощо.

- Нативна підтримка вебхуків через веб-інтерфейс, що дозволить швидко розробляти та експерементувати із “виходящими” інтеграціями.
- Підтримка єдиної авторизації (SSO) - можлива реалізація інтеграції через OAuth2, що дозволить користувачам входити в систему за допомогою корпоративних облікових записів (Google Workspace, Microsoft 365 тощо).
- Імпорт/експорт даних у форматах CSV/Excel - для оперативного перенесення задач, користувачів або історії з інших платформ (Trello, Jira).

Іншими словами, платформа не вимагає «починати з нуля», а дозволяє вписатися у вже діючий IT-ландшафт в найкоротших термінах, що є критично важливим для середніх і великих компаній із усталеними процесами.

Ключова ідея - мінімум жорстких рамок, максимум свободи для адаптації.

Побудована система - це не просто MVP або черговий таск-менеджер. Це основа для створення повноцінного корпоративного цифрового середовища, яке може рости, змінюватись, інтегруватися, і при цьому - залишатися повністю у власності та під контролем компанії.

Можливість масштабування - як технічного, так і функціонального - у поєднанні з відкритою архітектурою та простотою підтримки забезпечує стійкість системи до змін і викликів, що особливо актуально в умовах цифрової трансформації бізнесу.

3.5 Цінність рішення для корпоративного використання

Базовий принцип, покладений в основу системи - це створення повністю контрольованого, кастомізованого і відкритого простору для організації внутрішньої роботи компанії. У цьому сенсі розроблена система є не просто черговим таск-трекером - вона виконує роль ядра цифрової взаємодії, довкола якого можуть будуватися й інші бізнес-процеси: документація, облік, тайм-менеджмент, управління якістю тощо. А найголовніша – сама система теж може піддаватись змінам та ітераціям, що дозволяє створити дійсно унікальну систему під практично будь який бізнес-процес.

Ключові елементи цінності:

- Функціональна повнота для внутрішньої роботи: задачі, проєкти, ролі, статуси, тощо - усе в одному місці.
- Гнучкість у розширенні: можливість адаптувати систему під конкретну галузь або модель бізнесу.
- Відкритий код: гарантія прозорості, незалежності від провайдерів, можливість вільної модифікації.
- Контроль над даними: розгортання локально або у хмарі - на розсуд компанії, відповідно до політик безпеки.
- Швидкість впровадження: завдяки знайомим технологіям (Laravel, Vue.js) система може бути розгорнута та за необхідності покращена за кілька годин.

Інакше кажучи, мова йде про інструмент, який, не диктуватиме, як працювати, а дасть змогу впорядкувати вже наявну логіку бізнес-процесів у цифровому форматі.

3.6 Порівняльні переваги над SaaS-рішеннями

У попередніх розділах уже було проаналізувалися типові обмеження SaaS-підходу. Також варто зосередитись на тому, чому саме open-source система, розроблена в межах цієї роботи, виграє на практиці.

Критерій	SaaS-сервіси (Trello, Jira, Asana)	Розроблена система
Доступ до коду	Відсутній	Повний, відкритий, безумовний
Можливість модифікації	Лімітований або відсутнійAPI, без зміни внутрішньої логіки	Повна кастомізація
Зберігання даних	На серверах провайдера	Локально або у вибраній хмарі
Вартість у перспективі	Зростає пропорційно до кількості користувачів	Зростає з масштабуванням, яке повністю керується

Критерій	SaaS-сервіси (Trello, Jira, Asana)	Розроблена система
Адаптація під бізнес-модель	В межах шаблонів системи	Повна свобода структурування процесів кінцевим користувачем.
Інтеграції	Через API, часто обмежені	API або повна інтеграція на рівні бекенду
Підтримка	Через технічну підтримку постачальника, дуже обмежені спільноти	Open-source спільноти (які зазвичай є великим та ефективними, бо не обмежуються форумами постачальника)

Таблиця 1 – Таблиця порівнянь розроблюваної системи та інших схожих систем

Це порівняння не є абстрактним. Воно підтверджується практикою, адже чим більша компанія, тим менше їй підходить шаблонне рішення - і тим більше вона потенційно виграє від контролю над внутрішніми інструментами.

ВИСНОВОК

У процесі написання цієї дипломної роботи було здійснено комплексне дослідження теми корпоративних онлайн систем менеджменту та колаборації з фокусом на їх практичне значення, недоліки існуючих рішень та потенціал створення власного open-source продукту.

На першому етапі роботи проведено глибокий аналіз ринку готових інструментів, таких як Trello, Jira, Zoho Projects, Microsoft Planner, тощо. У ході дослідження з’ясувалося, що хоча всі ці платформи виконують заявлену функціональність, вони мають істотні обмеження у контексті корпоративного використання: жорстка логіка, обмежена кастомізація, непрозора модель зберігання даних, відсутність контролю над інфраструктурою та зростаюча вартість при масштабуванні.

Це підвело до наступного логічного етапу: вибір альтернативи у вигляді власного open-source рішення, яке дозволяє:

- Розміщувати дані в обраному середовищі (включно з локальними серверами).
- Забезпечити гнучкість, розширюваність і масштабованість без обмежень постачальника.
- Забезпечувати весь базовий функціонал, звичний для інших популярних систем колаборації та менеджменту.

У процесі реалізації було обрано стек технологій VILT (Vue.js, Inertia.js, Laravel, Tailwind CSS), який забезпечив баланс між швидкістю та простотою розробки, зрозумілою структурою та малою затратою зусиль на створення інтерфейсу, зручного для кінцевого користувача.

На базі цього стеку створено систему, яка включає:

- Управління проєктами, задачами, ролями, статусами, користувачами, ролями, доступами.
- Внутрішню комунікацію через коментарі.

- Базову систему сповіщень.
- Базову аналітику та звітність.
- Контроль доступу та ролей.
- Адаптивний, сучасний інтерфейс.

Система реалізована у вигляді відкритого програмного забезпечення, яке може бути розгорнуте у будь-якій компанії без ліцензійних обмежень, з можливістю подальшого розширення або інтеграції в існуючу інфраструктуру.

Практичне значення створеного продукту полягає у тому, що він може бути адаптований до потреб будь-якої організації, незалежно від галузі, суттєво знижує витрати на ПЗ у довгостроковій перспективі, дозволяє бізнесу залишатися незалежним від сторонніх сервісів, підвищує прозорість, відповідальності й керованості внутрішніх процесів, а також може стати основою для створення власної цифрової екосистеми компанії, або цінним її доповненням.

Окремо варто відзначити, що під час розробки були враховані не лише технічні, а й людські аспекти впровадження: простота інтерфейсу адаптація термінології, поступовість навчання. Це дозволяє системі не лише «працювати», а й бути реально прийнятою командою, що у практиці є чи не найважливішим фактором успіху.

Перспективи подальшого розвитку

Проєкт, реалізований у межах цієї дипломної роботи, є першим кроком до побудови повноцінного корпоративного середовища, яке:

- Може бути розширене модулями тайм-трекінгу, календаря, інтеграції з CRM та ERP.
- Потенційно підтримує мобільні версії (через PWA або нативні застосунки).
- Потенційно підтримувати систему “розширень” що ще більше спростить подальше налагодження системи, та потенційно – дозволить налаштувати систему до задовільного рівня взагалі без необхідності в кодуванні застосунку.

У цьому сенсі робота не лише завершується, а й відкриває нові напрями: для глибшої розробки, для реального впровадження у компаніях, а, можливо, і для перетворення на повноцінний продукт, здатний конкурувати з комерційними

аналогами.

Цей досвід є не лише навчальним етапом, а й практичною реалізацією ідеї, яка може мати реальне застосування у бізнес-середовищі - незалежно від масштабу чи галузі. В епоху цифрових трансформацій, володіння своїм інструментом - це не розкіш, а конкурентна перевага. І саме така система її забезпечує.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gröber L., Mrowczynski R., Vijay N., Muller D. A., Dabrowski A., Krombholz K. To Cloud or not to Cloud: A Qualitative Study on Self-Hosters' Motivation, Operation, and Security Mindset // Proceedings of the 32nd USENIX Security Symposium, August 9–11, 2023, Anaheim, CA, USA. — USENIX Association, 2023. — P. 2491–2506. — ISBN 978-1-939133-37-3. — URL: https://www.usenix.org/conference/usenixsecurity23/presentation/grober (дата звернення: 30.05.2025).

2. Мадні А. З., Зіверс М. Інтеграція систем: ключові погляди, досвід та виклики / Azad M. Madni, Michael Sievers // Systems Engineering. — 2014. — Vol. 17, No. 1. — P. 37–51. — DOI: 10.1002/sys.21249. — URL: https://onlinelibrary.wiley.com/doi/10.1002/sys.21249 (дата звернення: 30.05.2025).

3. Bafoutsou G., Mentzas G. Review and functional classification of collaborative systems // International Journal of Information Management. — 2002. — Vol. 22, No. 4. — P. 281–305. — DOI: 10.1016/S0268-4012(02)00013-0. — URL: https://www.sciencedirect.com/science/article/pii/S0268401202000130 (дата звернення: 30.05.2025).

4. Prahalad C. K., Ramaswamy V. Co-Creation Experiences: The Next Practice in Value Creation // Journal of Interactive Marketing. — 2004. — Vol. 18, No. 3. — P. 5–14. — DOI: 10.1002/dir.20015. — URL: https://onlinelibrary.wiley.com/doi/10.1002/dir.20015 (дата звернення: 30.05.2025).

5. Koskela L., Howell G. The Theory of Project Management: Explanation to Novel Methods // Proceedings of the 10th International Group for Lean Construction Conference (IGLC-10), August 2002, Gramado, Brazil. — 2002. — P. 1–11. — URL: https://www.researchgate.net/publication/228918258_The_theory_of_project_management_explanation_to_novel_methods (дата звернення: 30.05.2025).

6. Brusgaard H. The pitfalls of hidden costs in SaaS // Keepit Blog. — 2024. —

URL: https://www.keepit.com/blog/saas-hidden-costs/ (дата звернення: 30.05.2025).

7. Tailwind Labs. Tailwind CSS. — 2025. — URL: https://tailwindcss.com/ (дата звернення: 30.05.2025).

8. Bonaccorsi A., Rossi C. Why Open Source Software Can Succeed // LEM Working Paper Series, No. 2002/15, Sant'Anna School of Advanced Studies, Laboratory of Economics and Management, Pisa, Italy, July 2002. — 31 p. — URL: https://hdl.handle.net/10419/89290 (дата звернення: 30.05.2025).

9. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. — Sebastopol, CA: O'Reilly Media, 2017. — 616 p. — ISBN 978-1-4493-7332-0. — URL: https://dataintensive.net/ (дата звернення: 30.05.2025).

10. Baturay M. H., Birtane M. Responsive Web Design: A New Type of Design for Web-based Instructional Content // Procedia - Social and Behavioral Sciences. — 2013. — Vol. 106. — P. 2275–2279. — DOI: 10.1016/j.sbspro.2013.12.259. — URL: https://www.sciencedirect.com/science/article/pii/S187704281305455X (дата звернення: 30.05.2025).

11. Atlassian. Trello. — 2025. — URL: https://trello.com/uk (дата звернення: 30.05.2025).

12. Asana, Inc. Asana. — 2025. — URL: https://asana.com (дата звернення: 30.05.2025).

13. Atlassian. Jira. — 2025. — URL: https://www.atlassian.com/software/jira (дата звернення: 30.05.2025).

14. ClickUp. ClickUp — The everything app for work. — 2025. — URL: https://clickup.com (дата звернення: 30.05.2025).

15. Microsoft Corporation. Microsoft Planner. — 2025. — URL: https://www.microsoft.com/uk-ua/microsoft-365/planner/microsoft-planner (дата звернення: 30.05.2025).

16. Zoho Corporation. Zoho Projects — 2025. — URL: https://www.zoho.com/projects (дата звернення: 30.05.2025).

17. Srinivasan P. Trello vs. Google Tasks: Which Task Manager is Best? //

ClickUp Blog. — 2024. — URL: https://clickup.com/blog/trello-vs-google-tasks/ (дата звернення: 30.05.2025).

18. von Hippel E., von Krogh G. Open Source Software and the “Private-Collective” Innovation Model: Issues for Organization Science // Organization Science. — 2003. — Vol. 14, No. 2. — P. 209–223. — DOI: 10.1287/orsc.14.2.209.14992. — URL: https://doi.org/10.1287/orsc.14.2.209.14992 (дата звернення: 30.05.2025).

20. Janssen M., Joha A. Challenges for Adopting Cloud-Based Software as a Service (SaaS) in the Public Sector // ECIS 2011 Proceedings. European Conference on Information Systems, June 2011. — URL: http://aisel.aisnet.org/ecis2011/80 (дата звернення: 30.05.2025).

21. Vue.js. Introduction — The Progressive JavaScript Framework. — 2025. — URL: https://vuejs.org/guide/introduction.html (дата звернення: 30.05.2025).

22. Inertia.js. The Modern Monolith. — 2025. — URL: https://inertiajs.com (дата звернення: 30.05.2025).

23. Laravel. Laravel 10.x — Офіційна документація. — 2025. — URL: https://laravel.com/docs/10.x (дата звернення: 30.05.2025).

24. MDN Web Docs. MVC. — 2025. — URL: https://developer.mozilla.org/en-US/docs/Glossary/MVC (дата звернення: 30.05.2025).

25. Codd E. F. A Relational Model of Data for Large Shared Data Banks // Communications of the ACM. — 1970. — Vol. 13, No. 6. — P. 377–387. — DOI: 10.1145/362384.362685. — URL: https://dl.acm.org/doi/10.1145/362384.362685 (дата звернення: 30.05.2025).

26. Aston B. 36 Best Online Collaboration Tools For Teams In 2025 // People Managing People. — 2025. — URL: https://peoplemanagingpeople.com/tools/the-best-collaboration-tools/ (дата звернення: 30.05.2025).

27. Laravel. Laravel Breeze — Пакет для аутентифікації в Laravel 11.x. — 2025. — URL: https://laravel.com/docs/11.x/starter-kits#laravel-breeze (дата звернення: 30.05.2025).

28. Sharma N. 47 Workplace Collaboration Statistics and Trends in 2025 // ProofHub. — 2024. — URL: https://www.proofhub.com/articles/workplace-collaboration-

statistics (дата звернення: 30.05.2025).

29. Xerox. Що таке цифровізація і навіщо вона потрібна? — 2025. — URL: https://www.xerox.com/uk-ua/services/insights/shcho-take-tsifrov-zats-ya-nav-shcho-vona-potr-bna (дата звернення: 30.05.2025).

30. Attaran M., Attaran S., Kirkland D. The Need for Digital Workplace: Increasing Workforce Productivity in the Information Age // International Journal of Enterprise Information Systems. — 2019. — Vol. 15, No. 1. — P. 1–30. — DOI: 10.4018/IJEIS.2019010101. — URL: https://www.researchgate.net/publication/329844969_The_Need_for_Digital_Workplace_Increasing_Workforce_Productivity_in_the_Information_Age (дата звернення: 30.05.2025).

31. de Crombrugghe de Picquendaele L. Analysis of an innovative business model: the freemium and its applications : магістерська робота / Л. де Кромбругге де Піккандаль. — Louvain-la-Neuve : NOVA – School of Business and Economics, 2016. — 54 с. — URL: https://thesis.dial.uclouvain.be/entities/masterthesis/a0633008-3940-4262-8c2a-715e98a7d1e7 (дата звернення: 30.05.2025).

32. Froud E. The Rise of Project Management Software: A History // The Digital Project Manager. — 2024. — URL: https://thedigitalprojectmanager.com/topics/history-of-project-management-software/ (дата звернення: 30.05.2025).

33. Native Teams. Motivating Global Teams: Common Challenges & Solutions // Native Teams. — 2025. — URL: https://nativeteams.com/blog/motivating-global-teams (дата звернення: 30.05.2025).

34. D Danaiata, C Hurbean. SaaS - Better Solution for Small and Medium-Sized Enterprises // Applied Economics, Business and Development. — 2010. — Vol. 8, No. 1. — P. 29–34. — ISSN 1790-5109. ISBN: 978-960-474-184-7 — URL: https://www.appliedeconomics.biz (дата звернення: 30.05.2025).

35. Lee S. 7 Data Collaboration Trends Driving Software and Tech Advances // Number Analytics. — 2025. — URL: https://www.numberanalytics.com/blog/7-data-collaboration-trends-software-tech-advances (дата звернення: 30.05.2025).

36. Davis F. D. Perceived Usefulness, Perceived Ease of Use, and User Acceptance

of Information Technology // MIS Quarterly. — 1989. — Vol. 13, No. 3. — P. 319–340. — DOI: 10.2307/249008. — URL: http://www.jstor.org/stable/249008 (дата звернення: 30.05.2025).

37. Pearson S. Privacy, Security and Trust in Cloud Computing // HP Laboratories, HPL-2012-80R1. — 2012. — 37 c. — URL: https://d1wqtxts1xzle7.cloudfront.net/33984274/Privacy__Security_and_Trust_in_Cloud_Computing_HP-libre.pdf?1403141678=&response-content-disposition=inline%3B+filename%3DPrivacy_Security_and_Trust_in_Cloud_Comp.pdf&Expires=1748576397&Signature=f1Rwda6V58hVpgIygdMBwQzZgYCBs9XGtBLkOQzIdwN3Hs7j8Jjgk2zmXv5QiGbC1-SKWRyWGLpFs~K1~rmAHu17gRwrwD9PEEH0f4-5gzfJ4CQFS3Hpwo4dA4xXSonl-DR1PkVC8oTA8In1Isf7fbdsEx3Pua79gG53Tu4SnOU2xiszGHMRyj9gKyUuc3qslV4c97Pplc~Iux8RKTsu29FLrqFbnPwRDR4tJHp6qXhnRi9AmKzrMyAHOmcUPSgyGWEwC9P5wVQxT30D-KkHo9aVqwO2yX9MUTBDQlBSlkb~fnFD43oMEVQx4ajKB88dR1eAQVOp2tD1p3Uq8tc2~w__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA (дата звернення: 30.05.2025).

38. Markus M. L., Tanis C. The Enterprise System Experience: From Adoption to Success // R. W. Zmud, M. F. Price (eds.). Framing the Domains of IT Management: Projecting the Future Through the Past. — Pinnaflex Educational Resources, Inc., 2000. — P. 173–206.

39. Markus M. L. Technochange Management: Using IT to Drive Organizational Change // Journal of Information Technology. — 2004. — Vol. 19, No. 1. — P. 4–20. — DOI: 10.1057/palgrave.jit.2000002. — URL: https://link.springer.com/article/10.1057/palgrave.jit.2000002 (дата звернення: 30.05.2025).

40. Benlian A., Hess T. Opportunities and risks of software-as-a-service: Findings from a survey of IT executives // Decision Support Systems. — 2011. — Vol. 52, No. 1. — DOI: 10.1016/j.dss.2011.07.007. — URL: https://www.sciencedirect.com/science/article/abs/pii/S0167923611001278 (accessed: 30.05.2025).

41. Subashini S., Kavitha V. A survey on security issues in service delivery models of cloud computing // Journal of Network and Computer Applications. — 2011. — Vol. 34, No. 1. — P. 1–11. — DOI: 10.1016/j.jnca.2010.07.006. — URL: https://www.sciencedirect.com/science/article/abs/pii/S1084804510001281 (дата звернення: 30.05.2025).

42. Bostrom R. P., Heinen J. S. MIS Problems and Failures: A Socio-Technical Perspective. Part I: The Causes // MIS Quarterly. — 1977. — Vol. 1, No. 3. — P. 17–32. — URL: https://www.jstor.org/stable/248710 (дата звернення: 30.05.2025).

43. Choudhary V. Comparison of Software Quality Under Perpetual Licensing and Software as a Service // Journal of Management Information Systems. — 2007. — Vol. 24, No. 2. — P. 141–165. — DOI: 10.2753/MIS0742-1222240206. — URL: https://www.tandfonline.com/doi/abs/10.2753/MIS0742-1222240206 (дата звернення: 30.05.2025).

44. Park G., Comuzzi M., van der Aalst W. M. P. Analyzing Process-Aware Information System Updates Using Digital Twins of Organizations // Business & Information Systems Engineering. — 2022. — Vol. 64, No. 1. — P. 23–38. — DOI: 10.1007/s12599-021-00721-9. — URL: https://link.springer.com/article/10.1007/s12599-021-00721-9 (дата звернення: 30.05.2025).

45. Díaz de León Guillén M. Á., Morales-Rocha V., Fernández Martínez L. F. A systematic review of security threats and countermeasures in SaaS // Journal of Computer Security. — 2020. — Vol. 28, No. 5. — P. 635–653. — DOI: 10.3233/JCS-200002. — URL: https://content.iospress.com/articles/journal-of-computer-security/jcs200002 (дата звернення: 30.05.2025).

46. Woody's Express Parcels. IT Policies & Procedures: General Data Protection Regulation (GDPR). — 2018. — 8 с. — URL: https://woodys-express.com/privacy (дата звернення: 30.05.2025).

47. Microsoft. Історія технічного стану Azure. — URL: https://azure.status.microsoft/status/history/ (дата звернення: 30.05.2025).

48. Amazon Web Services. Summary of the Amazon Kinesis Event in the Northern Virginia (US-EAST-1) Region. — 2020. — URL:

https://aws.amazon.com/message/11201/ (дата звернення: 30.05.2025).

49. Google. Про обмеження щодо використання Google Ads у деяких країнах. — 2020. — URL: https://support.google.com/google-ads/answer/6163740?hl=uk (дата звернення: 30.05.2025).

50. Zhu K., Zhou Z. Lock-In Strategy in Software Competition: Open-Source Software vs. Proprietary Software // Information Systems Research. — 2011. — Articles in Advance, pp. 1–10. — DOI: 10.1287/isre.1110.0358. — URL: https://doi.org/10.1287/isre.1110.0358 (дата звернення: 30.05.2025).

51. Clarke R., Dorwin D., Nash R. Is Open Source Software More Secure? — Homeland Security / Cyber Security, 2005. — 33 с. — URL: https://courses.cs.washington.edu/courses/csep590/05au/whitepaper_turnin/oss(10).pdf (дата звернення: 30.05.2025).

52. Li N., Zhang B. The Research on Single Page Application Front-end development Based on Vue // Journal of Physics: Conference Series. — 2021. — Article 012030. — DOI: 10.1088/1742-6596/1883/1/012030. — URL: https://doi.org/10.1088/1742-6596/1883/1/012030 (дата звернення: 30.05.2025).

53. Bawane M., Gawande I., Joshi V., Nikam R., Bachwani S. A. A Review on Technologies used in MERN stack // International Journal for Research in Applied Science & Engineering Technology (IJRASET). — 2022. — Vol. 10, Issue 1. — P. 479–488. — ISSN 2321-9653. — URL: https://www.ijraset.com/fileserve.php?FID=39868 (дата звернення: 30.05.2025).

54. Laravel. Blade Templating Engine — Laravel 12.x Documentation. — 2025. — URL: https://laravel.com/docs/12.x/blade (дата звернення: 30.05.2025).

55. Gowda P. A. N. TypeScript vs. JavaScript: A Comparative Analysis // International Journal for Multidisciplinary Research (IJFMR). — 2019. — Vol. 1, Issue 2. — DOI: 10.36948/ijfmr.2019.v01i02.22779. — URL: https://www.researchgate.net/publication/388462409_TypeScript_vs_JavaScript_A_Comparative_Analysis (дата звернення: 30.05.2025).