

МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І
СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА
на тему:

АРХІТЕКТУРА ЦИФРОВОЇ БУДІВЕЛЬНОЇ ПЛАТФОРМИ

Здобувача вищої освіти першого (бакалаврського) рівня,
IV курсу, групи КБ-22-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології (за галузями)»


(підпис)

Ковалевича Владислава
Миколайовича
(прізвище, ім'я, по батькові)

Науковий керівник


(підпис)

Горбовий Артур
Юліанович, д-р техн.
наук, професор
(прізвище, ім'я, по батькові,
науковий ступінь, вчене звання)

Керівник освітньо-
професійної програми


(підпис)

Ніжегородцев Владислав
Олександрович, канд. пед.
наук, доцент
(прізвище, ім'я, по батькові,
науковий ступінь, вчене звання)

Допущено до захисту за рішенням кафедри комп'ютерних та інформаційних
технологій і систем протокол від «12» травня 2026 № 20

Завідувач кафедри


(підпис)

Філоненко Михайло
Миколайович, канд. фіз.-
мат. наук, доцент
(прізвище, ім'я, по батькові,
науковий ступінь, вчене звання)

Ірпінь – 2026

Реєстрація № 47

27.05.2026

(дата)



(підпис відповідальної особи)

Результати перевірки

ср захисту
(до захисту, на доопрацювання)

12.05.2026

(дата)



(підпис)

Результат захисту

4 (добре)
(оцінка)

08.06.2026

(дата)



(підпис)

Голова комісії:



(підпис)

О.С. Дюбанов

Члени комісії:



(підпис)

М.М. Філоненко



(підпис)

В.О. Ніжегородцев



(підпис)

А.А. Омельчук

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	27		08.06.2026 Горбовий А.Ю. Дюбанов О.С.
2. Дотримання вимог щодо оформлення	від 0 до 10	8		Філоненко М.М. Ніжегородцев В.О.
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	4		Омельчук А.А. Суховерхівка Л.П.
4. Рецензент	від 0 до 5	5		
РАЗОМ БАЛІВ	від 0 до 50	44		

Оцінювання кваліфікаційної роботи науковим керівником – 43 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
87	B	4 (добре)	08.06.2026	Дюбанов О.С. Філоненко М.М. Ніжегородцев В.О. Омельчук А.А. Суховерхівка Л.П.

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Інформаційні управляючі системи і технології (за галузями)»

Затверджую

Завідувач кафедри  Філоненко М.М.

«01» листопада 2025

Завдання на кваліфікаційну роботу
здобувачу вищої освіти

Ковалевичу Владиславу Миколайовичу

1. Тема кваліфікаційної роботи «Архітектура цифрової будівельної платформи».

Затверджена наказом ректора від «28» листопада 2026 № 1526

2. Строк здачі закінченої роботи «08» травня 2026

3. Мета роботи: обґрунтування та реалізація веб додатку цифрової будівельної платформи.

Об'єкт дослідження: процеси інформаційного забезпечення будівельних проєктів.

Предмет дослідження: архітектурні підходи, моделі даних і модулі платформи.

Орієнтовна база дослідження: Mrion Platform, підходи API-first та offline-first.

4. Вихідні дані до роботи: матеріали проєкту, кодова база, результати тестування та аудиту доступу.

5. Перелік графічного матеріалу: схема архітектури, ER-модель, схема API-запиту, схема черги синхронізації.

6. Зміст кваліфікаційної роботи (перелік питань за кожним розділом):

ВСТУП

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЦИФРОВОГО БУДІВНИЦТВА.

1.1. Основні поняття цифрової будівельної платформи. 1.2. Огляд сучасних технологій у галузі (BIM, GIS, IoT, хмарні сервіси). 1.3. Аналіз існуючих платформ та їх архітектурних підходів. 1.4. Функціональні та нефункціональні вимоги до платформи. 1.5. Методика дослідження та інструментарій.

РОЗДІЛ 2. ПРОЄКТУВАННЯ АРХІТЕКТУРИ ЦИФРОВОЇ БУДІВельНОЇ

ПЛАТФОРМИ. 2.1. Обґрунтування вибору архітектурного стилю. 2.2. Логічна архітектура системи та декомпозиція на модулі. 2.3. Проєктування моделі даних та сховищ. 2.4. Проєктування API та інтеграційних механізмів. 2.5. Забезпечення безпеки, відмовостійкості та масштабованості.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ОЦІНЮВАННЯ СИСТЕМИ. 3.1. Технологічний

стек і середовище розробки. 3.2. Реалізація серверної частини та бізнес-логіки.

3.3. Реалізація клієнтської частини та ролей користувачів. 3.4. Реалізація роботи з даними будівельних об'єктів.

7. Календарний план виконання роботи.

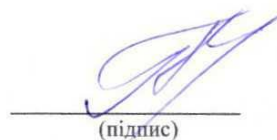
№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 30.10.2025	До 27.10.2025
2	I розділ (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	26.12.2025	24.12.2025
3	II розділ (аналіз експериментальної частини про методики досліджень)	18.02.2026	14.02.2026
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	16.04.2026	15.04.2026
5	Висновки	01.05.2026	28.04.2026
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	07.05.2026	01.05.2026
7	Отримання відгуку керівника	29.05.2026	29.05.2026
8	Отримання зовнішньої рецензії	01.06.2026	01.06.2026

8. Дата видачі завдання «01» грудня 2025

9. Керівник кваліфікаційної роботи

10. Керівник освітньо-професійної програми

11. Завдання до виконання отримав здобувач вищої освіти


(підпис)

Горбовий А.Ю.
(прізвище, ініціали)


(підпис)

Ніжегородцев В.О.
(прізвище, ініціали)


(підпис)

Ковалевич В.М
(прізвище, ініціали)

АНОТАЦІЯ

“Архітектура цифрової будівельної платформи”

Мета кваліфікаційної роботи полягає в обґрунтуванні архітектурних рішень цифрової будівельної платформи та формуванні практичних пропозицій щодо її удосконалення. У роботі досліджено предметну область цифрового будівництва, сучасні технологічні підходи до побудови платформних рішень, виконано аналіз існуючих архітектурних практик і визначено функціональні та нефункціональні вимоги до системи.

Об’єктом дослідження є процес функціонування цифрової будівельної платформи в умовах багаторольової взаємодії учасників будівельного проєкту.

Предмет дослідження — архітектурні принципи, модульна декомпозиція, модель даних, інтеграційні механізми, рольовий доступ і підходи до верифікації якості платформи. Базою дослідження є Mrion Platform, зокрема контури роботи з даними проєктів, задач, бюджетів, документів і подій QA/HSE.

За результатами дослідження обґрунтовано архітектурний підхід до розвитку платформи, визначено напрями підвищення інтеграційної стабільності, безпеки, масштабованості та продуктивності, а також сформовано практичні рекомендації для подальшої реалізації.

Кваліфікаційна робота 57 сторінок, список використаних джерел з 42 найменувань., 6 додатків.

Ключові слова: цифрова будівельна платформа, архітектура програмного забезпечення, модульна архітектура, BIM, API, RBAC, QA/HSE, модель даних.

Рік виконання кваліфікаційної роботи – 2026.

Рік захисту роботи – 2026.

ABSTRACT

“Digital Construction Platform Architecture”

The purpose of the qualification work is to substantiate architectural solutions for a digital construction platform and to develop practical proposals for its improvement. The study examines the domain of digital construction, modern technological approaches to platform design, analyzes existing architectural practices, and defines the functional and non-functional requirements for the system.

The object of research is the operation process of a digital construction platform under conditions of multi-role interaction among construction project participants.

The subject of research includes architectural principles, modular decomposition, data model design, integration mechanisms, role-based access, and approaches to platform quality verification. The research base is Mrion Platform, in particular the operational contours for project, task, budget, and document data, as well as QA/HSE events.

Based on the research results, an architectural approach to platform development is substantiated, key directions for improving integration stability, security, scalability, and performance are identified, and practical recommendations for further implementation are formulated.

The qualification work 57 pages, list of sources used with 42 names, 6 appendices.

Keywords: digital construction platform, software architecture, modular architecture, BIM, API, RBAC, QA/HSE, data model.

Year of completion of the qualification work: 2026.

Year of defense: 2026.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ.....	3
ВСТУП.....	4
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЦИФРОВОГО БУДІВНИЦТВА.....	7
1.1. Основні поняття цифрової будівельної платформи	7
1.2. Огляд сучасних технологій у галузі (BIM, GIS, IoT, хмарні сервіси)	9
1.3. Аналіз існуючих платформ та їх архітектурних підходів.....	13
1.4. Функціональні та нефункціональні вимоги до платформи	16
1.5. Методика дослідження та інструментарій.....	19
РОЗДІЛ 2. ПРОЄКТУВАННЯ АРХІТЕКТУРИ ЦИФРОВОЇ БУДІВЕЛЬНОЇ ПЛАТФОРМИ	22
2.1. Обґрунтування вибору архітектурного стилю	22
2.2. Логічна архітектура системи та декомпозиція на модулі	24
2.3. Проєктування моделі даних та сховищ.....	26
2.4. Проєктування API та інтеграційних механізмів	29
2.5. Забезпечення безпеки, відмовостійкості та масштабованості.....	32
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПЛАТФОРМИ.....	36
3.1. Технологічний стек і середовище розробки.....	36
3.2. Реалізація серверної частини та бізнес-логіки.....	38
3.3. Реалізація клієнтської частини та ролей користувачів	41
3.4. Реалізація роботи з даними будівельних об'єктів	43
ВИСНОВКИ	47
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	49
ДОДАТКИ.....	54

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

- API — Application Programming Interface (інтерфейс прикладного програмування)
- BIM — Building Information Modeling (інформаційне моделювання будівель)
- CI/CD — Continuous Integration / Continuous Delivery (безперервна інтеграція та безперервне постачання)
- CRUD — Create, Read, Update, Delete (базові операції створення, читання, оновлення та видалення даних)
- GIS — Geographic Information System (геоінформаційна система)
- HSE — Health, Safety and Environment (охорона праці, безпека та довкілля)
- IFC — Industry Foundation Classes (відкритий формат обміну BIM-даними)
- ISO/IEC — International Organization for Standardization / International Electrotechnical Commission (міжнародні стандарти у сфері інформаційних технологій)
- KPI — Key Performance Indicator (ключовий показник ефективності)
- MQTT — Message Queuing Telemetry Transport (легковаговий протокол обміну повідомленнями)
- QA — Quality Assurance (забезпечення якості)
- QA/HSE — інтегрований контур контролю якості, безпеки та екологічних аспектів
- RBAC — Role-Based Access Control (рольове керування доступом)
- REST — Representational State Transfer (архітектурний стиль побудови веб-API)

ВСТУП

Актуальність теми. Будівельна галузь переходить від розрізнених інформаційних систем до платформного підходу, у межах якого планування, виконання робіт, контроль якості, документообіг і фінансові процеси функціонують у спільному цифровому контурі. У такій моделі ключовим фактором результативності стає не окремий модуль, а цілісна архітектура платформи, що забезпечує узгодженість даних, передбачувану інтеграцію сервісів і стабільність роботи при зростанні навантаження. Додаткової ваги темі надає активне впровадження BIM-підходів у практику керування будівельною інформацією та вимоги до структурованого інформаційного обміну між учасниками проєкту. Для українського контексту важливими є також вимоги до функціонування цифрових сервісів у сфері будівництва та прозорості процедур, що безпосередньо впливає на архітектурні рішення на рівні даних, доступів і контрольних механізмів.

Стан і ступінь розробки проблеми. У наукових і прикладних працях достатньо повно висвітлено загальні принципи побудови складних програмних систем, модульної декомпозиції, еволюційної архітектури та сервісної інтеграції. Стандартизовано значну частину підходів до інформаційного моделювання об'єктів будівництва й організації обміну даними протягом життєвого циклу активу. Водночас на практиці залишаються відкритими питання щодо того, які саме архітектурні підходи доцільно застосовувати для цифрової будівельної платформи, де одночасно функціонують контури проєктного управління, бюджетування, документообігу, QA/HSE та рольового доступу. Найбільш проблемними залишаються узгодженість доменної моделі даних, контроль API-контрактів між модулями, а також баланс між безпекою і зручністю операційної роботи користувачів.

Мета дослідження — обґрунтувати архітектуру цифрової будівельної платформи та сформулювати практичні пропозиції щодо її подальшого удосконалення на прикладі Mrion Platform.

Для досягнення поставленої мети визначено такі завдання:

- дослідити поняттєвий апарат цифрових будівельних платформ, проаналізувати сучасні технології та існуючі архітектурні підходи у галузі цифрового будівництва;
- сформувані функціональні та нефункціональні вимоги до цифрової будівельної платформи та обґрунтувати методику її дослідження;
- обґрунтувати архітектурний стиль платформи, виконати декомпозицію на модулі та спроектувати модель даних;
- спроектувати API та інтеграційні механізми між підсистемами платформи з урахуванням вимог до безпеки, відмовостійкості та масштабованості;
- реалізувати серверну та клієнтську частини платформи, рольову модель доступу і роботу з даними будівельних об'єктів та оцінити результати реалізації.

Об'єкт дослідження — процес функціонування цифрової будівельної платформи в умовах багаторольової взаємодії учасників будівельного проєкту.

Предмет дослідження — архітектурні рішення, модель даних, інтеграційні механізми, рольовий доступ і підходи до верифікації якості цифрової будівельної платформи.

Методи дослідження. У роботі використано системний підхід для розгляду платформи як сукупності взаємопов'язаних модулів, методи аналізу і синтезу для формування цілісної архітектурної моделі, порівняльний аналіз для оцінки архітектурних альтернатив, структурно-функціональну декомпозицію для виділення модулів і контурів відповідальності, а також сценарний аналіз для перевірки ключових користувацьких і процесних маршрутів. Для оцінювання нефункціональних характеристик застосовано орієнтири моделі якості програмних систем, а для проєктування інтеграційного контуру — контрактний підхід до специфікації API. Під час формування вимог до захисту даних і доступу враховано сучасні підходи до управління інформаційною безпекою.

База дослідження — Mrion Platform як цифрова будівельна платформа з модульною організацією функціоналу. У межах дослідження розглянуто контури роботи з даними проєктів, задач, бюджетів, документів, а також процесами

QA/HSE та рольовою моделлю доступу (RBAC). Технологічний контекст реалізації (Next.js, TypeScript, Prisma, PostgreSQL) враховується як практична основа архітектурних рішень платформи.

Науково-практична значимість одержаних результатів. Полягає в узгодженні теоретичних принципів архітектури програмних систем із прикладними потребами цифрової будівельної платформи, де в одному середовищі інтегруються операційний, контрольний і аналітичний контури.

Особистий внесок здобувача. Автором самостійно виконано аналіз предметної області, постановку мети і завдань дослідження, систематизацію вимог до архітектури, узагальнення результатів проєктування та формування пропозицій щодо удосконалення платформи.

Апробація результатів роботи. Основні результати кваліфікаційної роботи були апробовані на двох студентських конференціях, одна з них міжнародна, що підтверджується сертифікатами учасника (дод. 3, рис. 3.1).

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЦИФРОВОГО БУДІВНИЦТВА

1.1. Основні поняття цифрової будівельної платформи

Цифрова будівельна платформа у межах цієї роботи розглядається як інтегроване програмне середовище, що об'єднує ключові процеси будівельного проекту в єдиному інформаційному контурі. На відміну від окремих прикладних систем, які автоматизують лише локальні операції, платформний підхід передбачає узгоджену роботу модулів планування, виконання, контролю, документування, бюджетування, комунікації та аналітики на спільній моделі даних[1]. Саме ця ознака відрізняє платформу від набору «розрізнених сервісів» і визначає вимоги до її архітектури [6, 28].

Базовим поняттям є архітектура цифрової платформи. У комп'ютерних науках архітектура трактується як сукупність структурних рішень: поділ системи на підсистеми, правила їх взаємодії, механізми обміну даними, моделі керування змінами та принципи еволюції програмного продукту [6]. Для будівельної предметної області це має прикладне значення, оскільки якість архітектури безпосередньо впливає на керованість строків, витрат і відповідальності між учасниками проекту. Якщо модулі не узгоджені, управлінські рішення приймаються на фрагментарних даних, що створює операційні та фінансові ризики.

Другим ключовим поняттям є доменна модель даних. Вона задає, які сутності існують у системі, як вони пов'язані та які події змінюють їх стан. Для цифрової будівельної платформи критичними є сутності проекту, етапів, задач, документів, бюджетних позицій, змін, матеріалів, підрядників, QA/HSE-подій та аналітичних зрізів. Коректна доменна модель забезпечує семантичну узгодженість даних між модулями і мінімізує дублювання інформації. У цьому контексті принципи domain-driven design залишаються методично релевантними, оскільки дозволяють формувати межі модулів за бізнес-логікою, а не за випадковими технічними ознаками [10].

Третє поняття - модульність і декомпозиція. Модульність означає, що кожний функціональний блок має чітко визначену зону відповідальності, але працює в спільному контурі платформи. У реальному контексті Mrion Platform це проявляється у наявності окремих функціональних секцій workspace (зокрема overview, tasks, budget, changes, materials, documents, qa, hse, handover, analytics, settings) та відповідних API-контурів проєктного рівня. Практична цінність такої декомпозиції полягає в керованості розвитку: локальні зміни впроваджуються у конкретному модулі без руйнування всієї системи, якщо контракти взаємодії визначені коректно [1, 12, 18]. Взаємозв'язок у межах єдиного контуру показано на рис. А.1 (дод. А). Приклад структури робочого простору платформи з групуванням модулів за процесами наведено на рис. А.2 (дод. А).

Четверте важливе поняття - інтеграційний контракт. У платформі інтеграція не зводиться до «наявності API»; вона передбачає стабільні правила запитів/відповідей, узгоджені словники статусів, версіонування та контроль сумісності. Для будівельної галузі це особливо важливо через взаємодію багатьох учасників і зовнішніх систем. Тому в архітектурному сенсі API є не лише технічним інтерфейсом, а інструментом організації міжмодульної дисципліни, де будь-яка зміна має бути передбачуваною для суміжних контурів та керування змінами [31].

Окремий блок понять стосується керування доступом і безпеки. У багаторольовому середовищі будівництва платформа повинна гарантувати, що користувач виконує лише ті дії, які відповідають його ролі та зоні відповідальності. Концептуально це реалізується через RBAC-модель, де права призначаються ролям, а не окремим користувачам, із подальшим аудитом подій. Для предметної області це означає захист критичних сценаріїв (погодження документів, бюджетні зміни, контроль якості та безпеки) від несанкціонованого втручання і забезпечення трасованості рішень [11, 19, 34].

Ще одне фундаментальне поняття - якість програмної системи. Для цифрової будівельної платформи якість оцінюється не лише через функціональне покриття, а через комплекс характеристик: надійність, продуктивність, сумісність, супроводжуваність, безпека, масштабованість. У межах стандартної рамки ISO/IEC

25010 це означає, що платформа має не просто «працювати», а зберігати керованість при зростанні обсягу даних, кількості учасників і складності процесів [18]. У практичному вимірі для Mrion Platform це пов'язано з необхідністю стабільної роботи проєктних контурів при одночасному веденні задач, документів, бюджету і контрольних процедур QA/HSE.

Для будівельного середовища окремо важливим є поняття інформаційного життєвого циклу. Дані проєкту не є статичними: вони створюються, погоджуються, оновлюються, архівуються і використовуються для аналітики та прийняття рішень. Тому архітектура платформи повинна підтримувати безперервність інформаційного потоку від планування до фактичного виконання та передачі об'єкта. У сучасній науково-практичній площині цей принцип узгоджується з підходами інформаційного менеджменту BIM і концепцією інтегрованого цифрового контуру проєкту [20, 21].

Отже, основні поняття цифрової будівельної платформи у межах цієї роботи формують цілісну теоретичну рамку: платформа як інтегроване середовище, архітектура як система структурних рішень, доменна модель як основа узгодженості даних, модульність як механізм керованого розвитку, інтеграційні контракти як умова стабільної взаємодії, RBAC як база безпеки та відповідальності, а якісні характеристики як критерій зрілості системи. Таке понятійне підґрунтя є необхідним для подальшого аналізу технологій у підрозділі 1.2. і для проектування архітектурних рішень у наступних розділах.

1.2. Огляд сучасних технологій у галузі (BIM, GIS, IoT, хмарні сервіси)

Цифрова трансформація будівельної галузі базується на поєднанні кількох технологічних напрямів, які працюють не ізольовано, а як єдина система. Для архітектури цифрової будівельної платформи визначальними є BIM як модельно-інформаційна основа, GIS як просторовий контур, IoT як джерело оперативних польових даних і хмарні сервіси як інфраструктура спільної роботи. Саме взаємодія

цих компонентів формує середовище, у якому дані про проєкт залишаються узгодженими на всіх етапах виконання.

Технологія BIM у сучасних дослідженнях розглядається насамперед як підхід до управління інформацією протягом життєвого циклу об'єкта, а не лише як тривимірна візуалізація. Цей підхід закріплений у стандартах серії ISO 19650, де визначено принципи організації, передачі та контролю якості інформації між учасниками будівельного процесу [2, 20, 21, 22]. Для цифрової платформи це означає необхідність єдиної структури даних, у якій проєктні сутності, задачі, документи та зміни пов'язані між собою на рівні статусів і відповідальностей.

Окреме практичне значення має інтероперабельність BIM-рішень. Умовою масштабованої платформи є обмін даними між різними програмними середовищами без втрати змісту і семантики моделей. У цьому контексті важливим є розвиток відкритих форматів, зокрема IFC, що дозволяє зменшити залежність від одного вендора та підтримувати інтеграцію між модульними системами [17]. Для платформи на кшталт Mrion це напряму пов'язано з можливістю синхронно працювати з блоками проєктного планування, документації, бюджетів і контролю виконання.

GIS-технології доповнюють BIM просторовою аналітикою. Якщо BIM фокусується на структурі об'єкта і процесі його реалізації, то GIS забезпечує прив'язку рішень до території, інфраструктури, обмежень використання землі та зовнішніх факторів. Для будівельного управління це критично у задачах вибору майданчика, планування логістики, аналізу оточення й перевірки регуляторної відповідності. Підґрунтям такого підходу є вимоги до функціонування цифрових сервісів у сфері будівництва в українському регуляторному контексті.

IoT-напряму у цифровому будівництві забезпечує перехід від періодичного ручного контролю до подієвого моніторингу в реальному або близькому до реального часі. Дані з датчиків, обладнання та польових пристроїв можуть використовуватись для фіксації стану робіт, умов безпеки, критичних інцидентів і технічних відхилень. Важливо, що цінність IoT проявляється лише тоді, коли телеметрія вбудована у доменну модель платформи: пов'язана із задачами,

журналом подій, HSE-процедурами та виконавцями. Для таких сценаріїв практично застосовуються легкі протоколи обміну, зокрема MQTT [16].

Хмарні сервіси виконують роль базової інфраструктури цифрової платформи. Вони забезпечують доступність системи для розподілених команд, еластичне масштабування обчислювальних ресурсів, централізоване оновлення функціоналу і керованість середовища розробки та експлуатації. Відповідно до моделі NIST, такі характеристики є ключовими для хмарного підходу і напряду впливають на стабільність та керованість цифрових систем [27]. У контексті будівництва це особливо важливо, оскільки учасники проєкту працюють у різних ролях і мають потребу в одночасному доступі до актуальних даних.

Для систематизації ролі кожної технології в архітектурі цифрової будівельної платформи проведено порівняльний аналіз за критеріями функціонального призначення, рівня зрілості та складності інтеграції. Узагальнені характеристики ключових технологій цифрового будівництва наведено в табл. 1.1.

Таблиця 1.1.

Порівняння технологій цифрового будівництва

Технологія	Основна функція	Роль у платформі	Рівень зрілості	Інтероперабельність	Складність інтеграції
BIM ISO 19650	Управління інформацією про об'єкт протягом усього життєвого циклу	Інформаційна основа: структура сутностей, документів, статусів	Висока	IFC, openBIM	Середня
GIS просторові дані	Прив'язка рішень до географічного та інфраструктурного контексту	Просторова аналітика: вибір майданчика, логістика, регуляторний контроль	Висока	GeoJSON, WMS, WFS	Середня
IoT MQTT, сенсори	Оперативний моніторинг стану об'єктів та умов безпеки на майданчику	Польові дані: HSE-події, технічні відхилення, телеметрія обладнання	Середня	MQTT, REST API	Висока

Продовження табл. 1.1

Технологія	Основна функція	Роль у платформі	Рівень зрілості	Інтероперабельність	Складність інтеграції
Хмарні сервіси SaaS / PaaS	Інфраструктура розподіленої роботи, масштабування, CI/CD	Доступність, еластичність ресурсів, централізоване оновлення платформи	Висока	REST, OAuth2, OpenAPI	Середня
Next.js / TS стек платформи	Типобезпечна серверно-клієнтська реалізація бізнес-логіки	Прикладний шар: API-маршрути, компоненти, контракти взаємодії	Висока	REST, GraphQL-ready	Низька
PostgreSQL / Prisma шар даних	Транзакційне сховище з контрольованою доменною схемою	Цілісність даних проєктів, бюджетів, задач, документів, QA/HSE	Висока	SQL, Prisma migrations	Низька

Джерело: складено автором на основі наукових та технічних джерел [2], [17], [20]–[22], [27], [29], [36], [37], [42].

Для Mrion Platform технологічна комбінація BIM, GIS, IoT і хмарних сервісів має практичне продовження в архітектурних вимогах: модульність, узгоджена модель даних, рольовий доступ (RBAC), простежуваність дій та надійні інтеграційні контракти між підсистемами. У такій логіці технологічний стек (Next.js, TypeScript, Prisma, PostgreSQL) виступає не самоціллю, а засобом реалізації керованої серверної логіки, типізованої взаємодії модулів і контрольованого зберігання проєктної інформації.

Отже, сучасні технології цифрового будівництва доцільно розглядати як єдиний взаємопов'язаний контур, де BIM забезпечує інформаційну цілісність, GIS — просторову обґрунтованість, IoT — оперативність даних, а хмарна інфраструктура — доступність і масштабованість платформи. Такий підхід створює методичну основу для подальшого аналізу архітектурних рішень і формування вимог до цифрової будівельної платформи в наступних підрозділах, з урахуванням потреб інтеграції, безпеки даних, продуктивності системи та підтримки масштабованого розвитку платформи.

1.3. Аналіз існуючих платформ та їх архітектурних підходів

Ринок цифрових рішень для будівництва показує, що платформи відрізняються не лише набором функцій, а насамперед архітектурною логікою, яка визначає їхню керованість, масштабованість і здатність до інтеграції. На практиці можна виділити кілька типових підходів: централізовані монолітні системи, модульні платформи з виділеними доменами та гібридні рішення, де частина функцій реалізована як окремі сервіси, а частина лишається в спільному ядрі. Для будівельної сфери вибір підходу критичний, оскільки система одночасно обробляє задачі планування, документообігу, бюджетування, контролю якості та безпеки.

Перший поширений підхід — монолітна система з єдиною базою даних і централізованою бізнес-логікою. Його перевага полягає у відносно швидкому старті та простішому адмініструванні на ранньому етапі. Проте зі зростанням кількості модулів і команд така архітектура зазвичай ускладнює внесення змін, підвищує зв'язаність компонентів і створює ризик каскадних регресій. Для будівельних платформ це особливо відчутно, коли треба одночасно змінювати правила доступу, логіку погоджень і структуру документів у кількох процесах.

Другий підхід — модульна доменна архітектура, у якій система поділяється на відносно самостійні функціональні блоки з чіткими межами відповідальності. У цьому випадку платформа краще адаптується до змін у бізнес-процесах, а розвиток окремих модулів можна планувати незалежно. З позиції архітектурної практики це відповідає принципам керованої декомпозиції та доменно-орієнтованого проєктування, де ключовою є узгодженість моделей даних і контрактів взаємодії між компонентами [4, 6, 28]. Для будівельного домену така модель більш природна, оскільки процеси QA/HSE, бюджети, задачі та документи мають різну динаміку змін.

Третій підхід — API-first архітектура з орієнтацією на стандартизовані інтерфейси обміну. За такого варіанту платформа будується як інтеграційне середовище, де внутрішні модулі й зовнішні системи взаємодіють через формалізовані контракти. Практично це підвищує передбачуваність інтеграцій, спрощує версіонування та знижує ризик технічної залежності від окремих

постачальників ПЗ. Методологічно цей підхід спирається на REST-принципи і специфікації опису API, які роблять взаємодію прозорою для команд розробки та супроводу [12, 31].

Для наочного порівняння переваг і обмежень розглянутих підходів розроблено схему, що відображає структурні відмінності між класичним монолітом, мікросервісами та модульним монолітом у контексті будівельної платформи. Порівняння архітектурних підходів із зазначенням обраного рішення для Mrion Platform наведено на рис. 1.1.

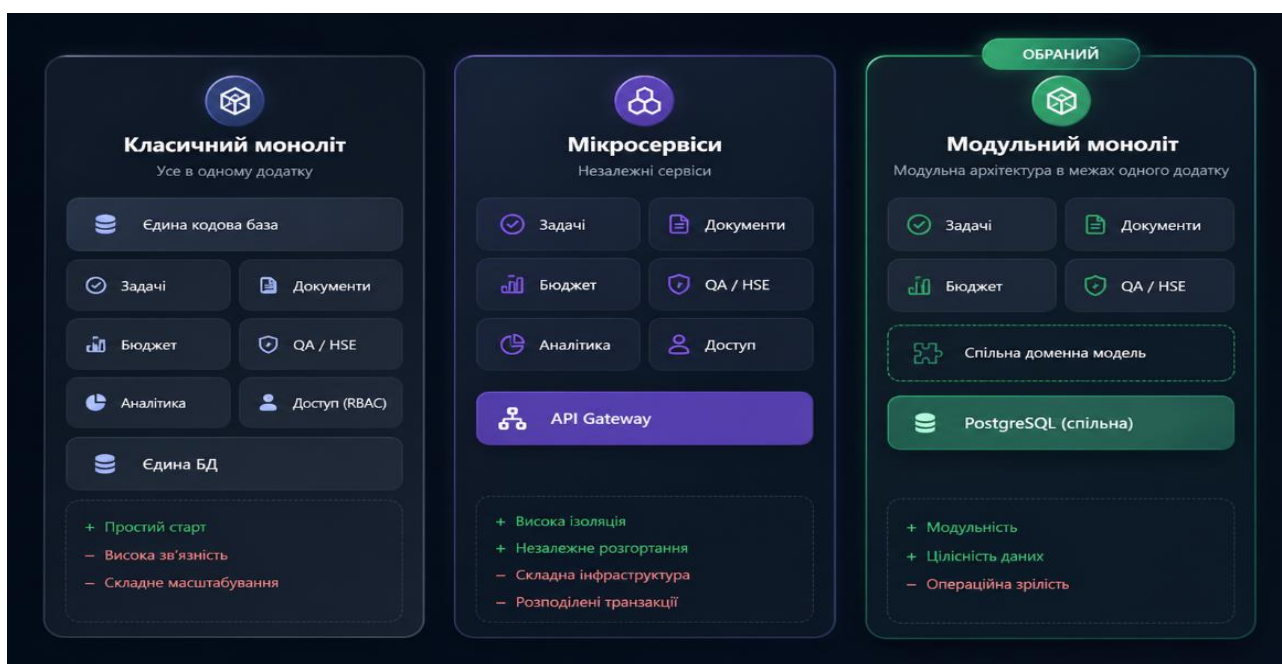


Рис. 1.1. Порівняння архітектурних підходів

Джерело: розроблено автором на основі [4, 13, 28, 38–40].

Для будівельної галузі окремо важливий баланс між інформаційною цілісністю та гнучкістю розширення. Якщо платформа орієнтована лише на документи, вона зазвичай не забезпечує належної зв'язності з календарним плануванням і бюджетним контролем. Якщо фокус лише на операційному контурі, втрачається простежуваність офіційних погоджень і юридично значущих дій. Саме тому сучасні платформи рухаються до комбінованої моделі, де документний, процесний і аналітичний контури працюють узгоджено в межах спільних правил даних і доступу.

На рівні предметної області це узгоджується з ВІМ-підходом: інформація має зберігати структуру і значення на всіх стадіях життєвого циклу, а не дублюватися між ізольованими підсистемами [20, 21]. У державному й регуляторному контексті України додаткову роль відіграє сумісність цифрових рішень із вимогами електронної системи у сфері будівництва, де важливі трасованість змін і коректний статус даних.

У практиці впровадження також видно відмінності за моделлю експлуатації: частина платформ реалізується як хмарний SaaS-сервіс, частина — як приватне або гібридне розгортання. Хмарна модель полегшує масштабування і оновлення, але висуває підвищені вимоги до управління доступом, журналювання дій і політик інформаційної безпеки. Тому архітектурна зрілість платформи визначається не тільки функціональністю, а й здатністю забезпечити контроль безпеки на рівні ролей, інтеграцій і даних [19].

У контексті Mrion Platform доцільним є саме модульний підхід із чітким доменним поділом і рольовою моделлю доступу (RBAC). Це дозволяє розвивати функціональні блоки проєктів, задач, бюджетів, документів і QA/HSE без руйнування загальної логіки системи. Водночас така архітектура вимагає дисципліни інтеграції: єдиної моделі сутностей, стабільних API-контрактів і прозорого аудиту дій користувачів. За наявності цих умов платформа може масштабуватися як за кількістю проєктів, так і за глибиною автоматизації процесів.

Отже, аналіз існуючих платформ показує, що найбільш життєздатним для цифрового будівництва є не ізольований технологічний вибір, а архітектурно збалансована модель: модульна структура, API-орієнтована взаємодія, централізовані правила безпеки та узгоджений інформаційний контур із підтримкою масштабування, аналітики, контролю змін і наскрізного управління будівельними процесами. Для подальших підрозділів це формує основу переходу від огляду рішень до конкретизації функціональних і нефункціональних вимог платформи.

1.4. Функціональні та нефункціональні вимоги до платформи

Формування вимог до цифрової будівельної платформи є ключовим етапом, який визначає не лише склад функцій, а й якість подальших архітектурних рішень. Для предметної області будівництва вимоги мають враховувати одночасно процесний контур (планування, виконання, контроль), документний контур (погодження, версійність, юридична значущість) і аналітичний контур (оцінка стану проєкту та ризиків). У практичному вимірі це означає, що платформа повинна підтримувати цілісну роботу з даними без дублювання і втрати контексту між модулями.

До функціональних вимог насамперед належить управління життєвим циклом проєкту: створення структури проєктів, етапів і задач, фіксація відповідальних осіб, контроль строків і статусів виконання. Для Mrion Platform це логічно пов'язано з модульною організацією даних, де блоки проєктів, задач, бюджетів, документів, QA/HSE та журналів подій мають узгоджено працювати в межах єдиної моделі. Окремою вимогою є підтримка ролей користувачів з різними сценаріями доступу до функцій і даних. Приклад сценарію перевірки доступу за роллю наведено на рис. В.1 (дод. В), а приклад екрану керування правами доступу - на рис. В.2 (дод. В). Узагальнену матрицю доступу за ролями подано на рис. В.3 (дод. В).

Другий важливий функціональний блок — керування документами та змінами. Платформа повинна забезпечувати реєстрацію, зберігання і контроль версій документів, прозоре погодження, простежуваність правок і зв'язок документів із задачами та етапами робіт. У контексті українського регуляторного поля це має узгоджуватись із вимогами до електронних документів та електронного документообігу. Для будівельної платформи також важливо забезпечити формування історії рішень, щоб будь-яка зміна могла бути обґрунтована і перевірена.

Третій функціональний блок — контроль якості, безпеки та інцидентів. Модулі QA/HSE повинні підтримувати фіксацію перевірок, невідповідностей, ризиків, коригувальних дій і їх виконання в межах календарного плану. Така

функціональність має бути пов'язана з ролями учасників і журналом подій, щоб керівник проєкту бачив не окремі записи, а причинно-наслідковий зв'язок між подією, відповідальним та результатом. Для платформи це створює основу керованої операційної дисципліни.

Нефункціональні вимоги визначають, наскільки система здатна стабільно працювати під навантаженням, безпечно обробляти дані та розвиватися без критичних збоїв. Базовими є вимоги до продуктивності, доступності, масштабованості, надійності, супроводжуваності й переносимості. У міжнародній практиці такі характеристики систематизуються в моделі якості ISO/IEC 25010, яка може бути використана як формальна основа для оцінювання архітектури платформи [18].

Критичною для будівельної платформи є вимога інформаційної безпеки. Оскільки система працює з договірними, фінансовими та технічними даними, необхідно забезпечити контроль доступу, захист каналів обміну, аудит дій, цілісність даних і керування інцидентами безпеки. На рівні практик це має відповідати сучасним підходам до інформаційної безпеки та API-захисту [19, 34]. Для контексту Mrion рольова модель доступу (RBAC) є обов'язковим інструментом, який обмежує операції користувача його функціональною відповідальністю.

Практичне застосування RBAC-моделі в межах платформи передбачає чітке розмежування операцій між учасниками проєкту залежно від їхньої функціональної ролі. Матрицю рольового доступу до модулів Mrion Platform із зазначенням рівня дозволених операцій наведено на рис. 1.2.

Важливою нефункціональною вимогою є інтеоперабельність. Платформа має підтримувати інтеграцію з зовнішніми сервісами і внутрішніми модулями через стабільні API-контракти та передбачуване версіонування. Це дозволяє поетапно розширювати функціональність без руйнування вже працюючих процесів і підтримує модульну еволюцію системи. Для цифрового будівництва така здатність особливо цінна, оскільки джерела даних зазвичай різномірні за форматом

і структурою. Для цифрового будівництва це забезпечує узгоджений обмін даними між учасниками, підрядниками та зовнішніми системами в межах проєкту.

Модуль	Адміністратор	Керівник проєкту	Менеджер	Виконавець	QA-інспектор	HSE-офіцер	Спостерігач
Проекти	●	●	●	●	○	○	●
Задачі	●	●	●	●	●	○	●
Документи	●	●	●	●	●	●	●
Бюджет	●	●	●	○	○	○	●
QA	●	●	●	●	●	●	●
HSE	●	●	●	●	●	●	●
Аналітика	●	●	●	○	●	●	●
Налаштування	●	●	○	○	○	○	○

Рис. 1.2. Матриця рольового доступу RBAC

Джерело: розроблено автором на основі [11, 19, 30, 33].

Окремо варто виділити вимоги до відповідності нормативному середовищу та ідентифікації користувачів. Для системи, що працює з юридично значущими діями, потрібні коректні механізми підтвердження особи, фіксації авторства змін і незаперечності операцій, що узгоджується з підходом до електронної ідентифікації та довірчих послуг. У практичному сенсі це знижує правові ризики при роботі з погодженнями, актами та іншими офіційними документами.

З технічної точки зору вимоги до продуктивної реалізації повинні підтримуватися інструментами стеку платформи: типізованою серверною логікою, керованою моделлю даних і контрольованим доступом до БД. У цьому контексті згадування Next.js, TypeScript, Prisma та PostgreSQL є доречним саме як реалізаційної основи для нефункціональних вимог до цілісності, узгодженості й супроводжуваності системи, а не як самоцілі.

Отже, функціональні вимоги визначають, що саме має робити платформа в операційній діяльності, а нефункціональні — яким чином система повинна це робити стабільно, безпечно і масштабовано. Для теми дипломної роботи та

контексту Mrion Platform їх узгоджене формулювання створює базу для наступного підрозділу, де конкретизується методика дослідження та інструментарій оцінювання архітектурних рішень.

1.5. Методика дослідження та інструментарій

У межах цієї дипломної роботи методика дослідження побудована так, щоб поєднати теоретичний аналіз архітектурних підходів із прикладною перевіркою їх придатності для цифрової будівельної платформи. Її мета полягає не лише в описі окремих технологій, а у формуванні цілісної процедури, за якою можна обґрунтовано перейти від аналізу предметної області до проектування архітектури. Такий підхід потрібний через складність будівельного домену, де одночасно взаємодіють процеси планування, виконання, документування, фінансового контролю та безпеки.

Саме така послідовність дозволяє зменшити ризик ситуації, коли "функція працює локально", але не витримує реальної експлуатації при рості навантаження або Методологічною основою дослідження є системний і архітектурний підходи. Системний підхід дозволяє розглядати платформу як взаємопов'язану сукупність модулів, даних, ролей і регламентів, а не як набір ізольованих функцій. Архітектурний підхід, у свою чергу, фокусується на межах відповідальності компонентів, контрактах взаємодії та якості еволюції рішення в часі [6, 38]. Для теми «Архітектура цифрової будівельної платформи» така комбінація є практично доцільною, оскільки дає можливість одночасно оцінювати і функціональне покриття, і технічну керованість системи.

На рівні наукових методів у роботі використано аналіз і синтез, порівняння, абстрагування та структурну декомпозицію. Аналіз і синтез застосовано для виявлення ключових сутностей предметної області та узгодження їх у єдиній моделі даних. Метод порівняння використано для зіставлення архітектурних підходів існуючих платформ і визначення їхніх сильних та слабких сторін у

контексті будівництва. Структурна декомпозиція дала змогу перейти від загальних вимог до логічно відокремлених модулів із чіткими зонами відповідальності.

Окремий блок методики становить нормативно-орієнтований аналіз. Для інформаційного менеджменту в будівництві орієнтиром виступають підходи ВІМ і правила організації даних у серії ISO 19650 [20, 21]. Для оцінювання архітектурної якості застосовано модель характеристик якості ПЗ, зокрема за критеріями надійності, супроводжуваності, безпеки та сумісності [18]. Для інтеграційного контуру враховано API-орієнтований підхід, де взаємодія між компонентами має бути формалізована на рівні контрактів [12, 31]. У частині інформаційної безпеки методика узгоджується з принципами керування доступом, аудитом дій і контролем ризиків [19].

Емпірична частина дослідження побудована на аналізі реального контексту Mrion Platform. Базою спостереження є структура модулів платформи, логіка рольового доступу (RBAC), моделі даних проєктів і задач, а також зв'язки між документним, бюджетним та QA/HSE-контурами. Додатково враховуються технічні артефакти, що відображають спосіб реалізації архітектури: організація серверної логіки, структура API, схеми доступу до даних і механізми простежуваності дій користувачів. Якщо конкретний факт не має прямого підтвердження у доступному контексті проєкту, у роботі застосовується нейтральне формулювання без категоричних висновків.

Інструментарій дослідження включає як концептуальні, так і технічні засоби. До концептуальних належать матриця вимог, таблиця відповідності «вимога - модуль», а також порівняльні схеми архітектурних альтернатив. До технічних належать засоби аналізу структури коду і моделі даних, а також інструменти перевірки цілісності API-контрактів і зв'язності сутностей. У прикладному контексті Mrion Platform технологічний стек Next.js, TypeScript, Prisma та PostgreSQL розглядається як інженерна основа реалізації прийнятої архітектури: типізована серверна логіка, контрольована доменна модель і транзакційна узгодженість даних, що забезпечують масштабованість, підтримуваність та стабільність системи.

Для об'єктивності оцінювання у методиці використовуються групи критеріїв, а не одиничний показник. Перша група критеріїв стосується функціонального покриття: наскільки платформа підтримує ключові бізнес-процеси проекту та їх рольову диференціацію. Друга група стосується архітектурної якості: зв'язаність модулів, керованість змін, простота інтеграції, простежуваність подій. Третя група стосується операційної стійкості: безпека доступу, передбачуваність оновлень, стабільність роботи з даними. Такий спосіб оцінювання знижує ризик суб'єктивних висновків і дозволяє обґрунтовано порівнювати альтернативні рішення.

Важливим елементом методики є перевірка внутрішньої узгодженості результатів між розділами роботи. Теоретичні висновки першого розділу повинні прямо переходити у проєктні рішення другого розділу, а результати прикладного аналізу - у рекомендації третього розділу. У разі розбіжностей між теоретично очікуваною і фактично реалізованою моделлю пріоритет надається перевірковим даним проєкту з подальшим поясненням причин такого відхилення.

Отже, запропонована методика дослідження поєднує теоретичну обґрунтованість, нормативну опору та практичний інструментарій аналізу. Для теми цієї дипломної роботи вона дозволяє послідовно перейти від опису предметної області до проєктування архітектури цифрової будівельної платформи, зберігаючи логіку, доказовість і прикладну цінність отриманих результатів.

РОЗДІЛ 2. ПРОЄКТУВАННЯ АРХІТЕКТУРИ ЦИФРОВОЇ БУДІВЕЛЬНОЇ ПЛАТФОРМИ

2.1. Обґрунтування вибору архітектурного стилю

Вибір архітектурного стилю для цифрової будівельної платформи визначає не тільки технічну реалізацію системи, а й те, наскільки стабільно вона буде розвиватися при зростанні кількості проєктів, користувачів і процесів. Для предметної області будівництва це особливо важливо, оскільки в межах однієї платформи одночасно взаємодіють задачі планування, документообігу, бюджетного контролю, QA/HSE-процедур і аналітики. Тому обґрунтування архітектурного стилю має враховувати не лише швидкість старту розробки, а й керованість змін, прозорість інтеграцій та безпеку доступу.

На етапі проєктування для платформи такого типу зазвичай розглядають три базові підходи: класичний моноліт, мікросервісну архітектуру і модульний моноліт. Класичний моноліт дає простішу початкову реалізацію, але зі зростанням функціональності створює надмірну зв'язаність компонентів. Повна мікросервісна модель, навпаки, забезпечує високий рівень ізоляції сервісів, однак суттєво підвищує вимоги до операційної інфраструктури, спостережуваності, тестування і підтримки розподілених транзакцій [13, 28, 39]. Для середовища, де критичною є узгодженість даних між модулями проєкту, надто ранній перехід до мікросервісів може створити більше технічних ризиків, ніж переваг.

З урахуванням цього для Mrion Platform обґрунтованим є вибір модульного моноліту з API-орієнтованою взаємодією між функціональними блоками. Такий стиль поєднує централізовану керованість даними з чіткою доменною декомпозицією. Поточна структура платформи вже відповідає цій логіці: функції розділені за предметними контурами (проєкти, задачі, бюджети, документи, QA/HSE та інші), а межі взаємодії проходять через формалізовані серверні маршрути й контракти [35, 40]. У практичному сенсі це дозволяє незалежно розвивати модулі без руйнування цілісності всієї системи.

Додатковим аргументом на користь модульного підходу є вимога до єдиної доменної моделі. Для будівельної платформи критично, щоб зміна статусу задачі, оновлення документа і корекція бюджету могли бути узгоджені в одному транзакційному контурі. Саме тому централізоване керування даними в межах спільної моделі є перевагою на етапі активного формування функціоналу[1]. Із позиції доменно-орієнтованого підходу це відповідає принципу чітких меж контекстів і контрольованої мови предметної області [10].

Важливим критерієм вибору стилю є інтеграційна дисципліна. Для подальшого розвитку платформи необхідно, щоб взаємодія модулів і зовнішніх систем залишалася передбачуваною. Це досягається через API-first підхід і стандартизований опис контрактів, що узгоджується з сучасними вимогами до проектування інтерфейсів [12, 31]. У контексті Mrion це створює технічну основу для масштабування платформи без втрати керованості.

Окремо слід врахувати вимоги якості та безпеки. Архітектурний стиль має підтримувати не тільки функціональність, а й нефункціональні характеристики: супроводжуваність, надійність, сумісність, безпечність і контроль доступу [18]. Для цифрової будівельної платформи, де обробляються фінансові, організаційні та технічні дані, необхідні централізовані механізми RBAC, аудит дій і захист API-контурів [19]. У модульному моноліті такі механізми впроваджуються більш цілісно, ніж у розподіленому середовищі, де правила доводиться дублювати між сервісами.

Технічний стек Mrion Platform також узгоджується з обраним стилем. Використання Next.js і TypeScript дає керований серверно-клієнтський контур і типізовану взаємодію між модулями [29, 42], Prisma забезпечує контрольовану роботу з доменною моделлю [37], а PostgreSQL — надійне транзакційне сховище для взаємопов'язаних сутностей [36]. У цьому випадку стек виступає не причиною вибору стилю, а засобом його якісної реалізації.

Таким чином, для поточного етапу розвитку Mrion Platform найбільш доцільним є модульний моноліт із чіткими доменними межами та API-орієнтованими контрактами. Такий вибір забезпечує баланс між швидкістю

розвитку, узгодженістю даних і керованістю архітектурних змін. Водночас він не обмежує еволюцію системи: у разі зростання навантаження або організаційної складності окремі модулі можуть бути винесені в самостійні сервіси без повної перебудови платформи [6].

2.2. Логічна архітектура системи та декомпозиція на модулі

Логічна архітектура цифрової будівельної платформи повинна відображати не тільки технічну структуру системи, а й реальну предметну логіку будівельного проекту. Для Mrion Platform це означає, що архітектурні межі мають будуватися навколо ключових доменів: проекти, задачі, бюджети, документи, контроль якості та безпеки, аналітика і службові контури адміністрування. Такий підхід дозволяє уникнути змішування відповідальностей і робить систему передбачуваною при подальшому розвитку.

Базовим принципом декомпозиції є поділ на функціональні модулі з чітко визначеними вхідними і вихідними даними. У логічній моделі кожний модуль відповідає за власний набір бізнес-операцій, але працює в узгодженому інформаційному просторі. Для практичної реалізації це означає, що зміни в одному домені не повинні неконтрольовано впливати на інші, а взаємодія між модулями має відбуватися через стабільні контракти [6, 28].

Ядром логічної архітектури є модуль проєктного управління, який задає контекст для всіх інших підсистем. Навколо нього формуються модулі задач і планування, що фіксують операційну діяльність; модуль документів, який забезпечує регламентований документообіг; бюджетний модуль, що відображає фінансовий контур; а також модулі QA/HSE, де контролюються показники якості робіт і безпеки на майданчику. Аналітичний контур виступає надбудовою, що агрегує дані з операційних модулів для прийняття управлінських рішень.

Для відображення структурної організації платформи та взаємозв'язків між її функціональними контурами розроблено схему логічної архітектури, що охоплює клієнтський шар, API-контур, доменні модулі та шар даних. Логічну архітектуру та

декомпозицію Mrion Platform на доменні модулі та принципи їх взаємодії наведено на рис. 2.1.



Рис. 2.1. Логічна архітектура та декомпозиція модулів

Джерело: розроблено автором на основі концепцій доменно-орієнтованого проєктування Domain-Driven Design: Tackling Complexity in the Heart of Software та модульної архітектури програмних систем [6, 28, 35, 40].

З позиції доменно-орієнтованого проєктування така структура відповідає принципу bounded context: кожен модуль має власну предметну модель і власні правила валідації, але інтегрується з іншими через узгоджені точки обміну [10]. Це критично для будівельної платформи, оскільки одна й та сама сутність, наприклад «етап робіт», може по-різному інтерпретуватися у задачному, документному та фінансовому контурах. Логічна декомпозиція зменшує ризик конфліктів і робить ці відмінності керованими.

Окремим елементом архітектури є модель доступу. У Mrion Platform полі користувачів (RBAC) пов'язані з доменними операціями модулів, а не лише з формальним поділом на «перегляд» і «редагування». Це дозволяє коректно розподіляти відповідальність між учасниками проєкту та підтримувати трасованість дій у критичних сценаріях. Для систем такого класу цей підхід напряму пов'язаний із вимогами безпеки й аудиту доступу [11, 19, 30].

Взаємодія модулів у логічній архітектурі реалізується через API-контракти. Така модель дає змогу формалізувати обмін подіями і даними між контурами, забезпечити версіонування інтерфейсів і контроль сумісності змін. Для платформи, що розвивається поетапно, це принципово, оскільки дозволяє масштабувати функціональність без руйнування вже стабільних частин системи [12, 31].

На рівні даних логічна декомпозиція передбачає спільні базові сутності та спеціалізовані доменні об'єкти. Спільні сутності (проект, користувач, роль, журнал подій) забезпечують цілісність платформи, а доменні сутності (задача, бюджетна позиція, документ, інцидент QA/HSE) реалізують предметну специфіку. Такий баланс між уніфікацією і спеціалізацією є необхідним для того, щоб система одночасно лишалась керованою і не втрачала глибину функціонального опрацювання.

З технічної точки зору обрана логічна архітектура узгоджується з поточним стеком Mrion Platform: Next.js і TypeScript забезпечують структуровану серверно-клієнтську взаємодію [29, 42], Prisma підтримує формалізацію доменної моделі [37], PostgreSQL — цілісність і надійність транзакційного контуру [36]. У цьому підрозділі стек розглядається як інструмент реалізації архітектурної логіки, а не як визначальний фактор декомпозиції.

Отже, логічна архітектура Mrion Platform базується на модульній декомпозиції з чіткими доменними межами, рольовим контролем доступу та контрактною взаємодією між підсистемами. Такий підхід формує стійку основу для подальшого проєктування моделі даних і інтеграційних механізмів у наступних підрозділах.

2.3. Проєктування моделі даних та сховищ

Проєктування моделі даних для цифрової будівельної платформи є критичним етапом, оскільки саме структура даних визначає узгодженість бізнес-процесів, можливість масштабування системи та якість управлінської аналітики. Для Mrion Platform модель даних повинна одночасно підтримувати операційний

контур проекту, документний обіг, бюджетний контроль, задачний цикл і блоки QA/HSE, не створюючи конфліктів між доменами. Тому під час проєктування пріоритет надається не окремим таблицям, а цілісній доменній структурі з чіткими зв'язками між сутностями.

На концептуальному рівні модель будується навколо сутності «проєкт», яка виступає кореневим контекстом для більшості операцій. Із проєктом пов'язуються етапи, задачі, документи, бюджетні позиції, події контролю якості та безпеки, журнали змін і аналітичні зрізи. Такий підхід відповідає доменно-орієнтованому принципу, за яким межі даних мають відображати реальну предметну логіку, а не зручність короткострокової реалізації [10]. У практичному сенсі це спрощує контроль цілісності і зменшує кількість неявних залежностей між модулями. Ключові зв'язки показано на рис. Д.1 (дод. Д) і рис. Д.2 (дод. Д).

Для документного і регламентного контуру важливо підтримувати версійність, статусний життєвий цикл і трасованість авторства змін. Це пов'язано не лише з внутрішніми потребами платформи, а й з вимогами до інформаційного менеджменту в будівництві [20, 21] та вимогами до функціонування цифрових сервісів у сфері будівництва [1]. У моделі даних це реалізується через зв'язок документа з проєктом, відповідальними ролями, станом погодження і подіями аудиту, що забезпечує відтворюваність прийнятих рішень.

Бюджетний контур у структурі даних проєктується як окремий домен із власними сутностями плану, факту, змін і підстав коригування. Важливо, щоб фінансові записи не існували ізольовано від задачного та документного блоків, оскільки в реальній практиці бюджетна зміна майже завжди має процесну причину і документальне підтвердження. Саме тому для платформи доцільно фіксувати зв'язки між бюджетною операцією, джерелом зміни і відповідальною роллю користувача.

У контурі QA/HSE модель даних повинна зберігати інциденти, перевірки, коригувальні дії та їх статус виконання у зв'язку з конкретним об'єктом робіт. Для управлінського аналізу важливо не лише накопичувати події, а й забезпечувати причинно-наслідкову зв'язність: що саме сталося, у якому модулі,

ким зафіксовано, який результат виконання. Така структура дозволяє перейти від формального журналювання до практичного контролю ризиків [26].

На логічному рівні модель даних для Mrion Platform доцільно реалізовувати як транзакційно узгоджений реляційний контур, де PostgreSQL виконує роль основного операційного сховища. Вибір реляційної моделі обґрунтовується потребою в жорстких зв'язках, обмеженнях цілісності та надійній обробці взаємозалежних операцій, що особливо важливо для проєктно-бюджетних і документних сценаріїв [24]. Реляційний підхід у цьому випадку краще підтримує контрольовану еволюцію схеми, ніж розрізнені сховища без єдиної моделі транзакцій.

Фізичне проєктування сховища передбачає розмежування операційного і аналітичного використання даних. Операційний шар орієнтований на стабільні CRUD-сценарії та контроль доступу за ролями, тоді як аналітичний шар потребує індексування, агрегування і періодичних зрізів для управлінських показників. Для збереження продуктивності важливо, щоб аналітичні запити не руйнували стабільність транзакційного контуру. У практичній архітектурі це досягається регламентами запитів, вибірковою денормалізацією звітних представлень і контролем важких обчислень.

Важливим елементом моделі є блок доступу та аудиту. Сутності користувачів, ролей і дозволів повинні бути пов'язані з бізнес-операціями так, щоб будь-яку критичну зміну можна було перевірити за журналом подій. Така організація даних узгоджується з підходами до інформаційної безпеки і дозволяє підтримувати RBAC-модель не декларативно, а через перевірювані факти виконаних дій. [19]

На рівні прикладної реалізації роль структурного посередника між доменною моделлю та БД виконує Prisma, що дає можливість формалізувати схему, контрольовано проводити міграції і підтримувати цілісність типів у серверній логіці на TypeScript. У зв'язці з Next.js це забезпечує прогнозовану роботу API-шару, де контракти взаємодії можуть бути узгоджені зі схемою даних і політиками

доступу [31, 37]. Для довгострокової підтримки системи це зменшує ризик розходження між моделлю коду і фактичною структурою сховища.

Отже, проєктування моделі даних та сховищ для Mrion Platform базується на доменній декомпозиції, реляційній цілісності, рольовому контролі та керованій еволюції схеми. Такий підхід забезпечує узгоджену роботу модулів проєктів, задач, бюджетів, документів і QA/HSE, а також створює надійну основу для подальшого проєктування API й інтеграційних механізмів у наступному підрозділі.

2.4. Проєктування API та інтеграційних механізмів

Проєктування API для цифрової будівельної платформи є ключовим елементом архітектури, оскільки через інтерфейси взаємодії забезпечується зв'язність між модулями, зовнішніми сервісами та користувацькими застосунками. Для Mrion Platform API-контур має підтримувати сценарії проєктного управління, документного обігу, бюджетного контролю, QA/HSE-процесів та аналітичних запитів. Тому базовими вимогами є передбачуваність контрактів, чітка структура ресурсів і стабільність endpoint-ів у часі.

На логічному рівні доцільним є ресурсно-орієнтований підхід, у якому API відображає доменну модель платформи: проєкти, задачі, документи, бюджети та події контролю якості. Це спрощує керування доступом, версіонування й трасованість дій, що узгоджується з REST-принципами та сучасними підходами до опису API-контрактів [12, 31]. Кожний маршрут має визначені права доступу та правила валідації даних. Схему інтеграційної взаємодії модулів через API наведено на рис. Б.1 (дод. Б), а приклад формату запиту/відповіді — на рис. Б.2 (дод. Б).

Важливою вимогою є уніфікація форматів запиту й відповіді. Доцільно застосовувати єдину структуру службових полів, коди помилок, схеми пагінації та стандартний підхід до фільтрації й сортування. Це підвищує передбачуваність клієнтської взаємодії та зменшує витрати на супровід API при розширенні функціоналу [41].

Інтеграційні механізми в Mrion Platform доцільно реалізовувати як комбінацію синхронної та подієвої взаємодії. Синхронний API-контур забезпечує транзакційні бізнес-операції, тоді як подієвий контур використовується для повідомлень про статусні переходи, QA/HSE-інциденти та фонові аналітичні оновлення. Для телеметричних або польових інтеграцій можуть використовуватися легкі протоколи подієвого обміну, зокрема MQTT [16]. Такий підхід дозволяє уникати перевантаження транзакційного API другорядними процесами.

Для унаочнення міжмодульної та зовнішньої взаємодії розроблено схему API-контрактів, що відображає транзакційний і подієвий контури та правила інтеграційної дисципліни між компонентами платформи. Схему API-контрактів та інтеграційних механізмів Mrion Platform наведено на рис. 2.2.

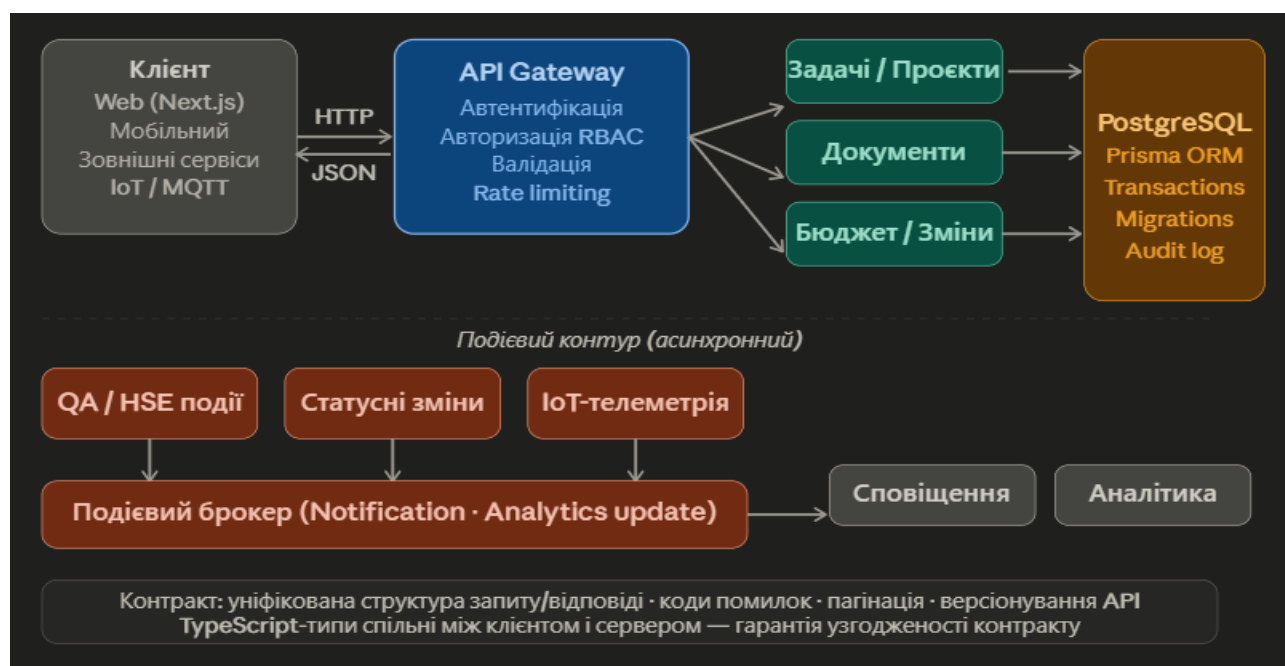


Рис. 2.2. Схема API-контрактів та інтеграційного шару

Джерело: розроблено автором на основі принципів REST-архітектури Roy Thomas Fielding [12], а також підходів до побудови мікросервісних і модульних систем [28, 39].

Окрему роль відіграє інтероперабельність із зовнішніми цифровими системами будівництва. З урахуванням BIM/GIS-контексту API має підтримувати інтеграцію з просторовими і модельними даними без руйнування внутрішньої доменної логіки. Для цього важливо розмежовувати внутрішню модель даних і

зовнішні інтеграційні представлення, а також контролювати трансформації форматів на рівні інтеграційного шару [17]. У такій схемі платформа зберігає цілісність власної моделі та одночасно підтримує розширений обмін.

Критичною частиною проєктування API є безпека взаємодії. Для кожного endpoint-а повинні бути визначені політики автентифікації, авторизації, валідації даних і журналювання. У контексті рольової моделі (RBAC) права доступу мають перевірятися не на рівні інтерфейсу в цілому, а на рівні конкретної операції над конкретним ресурсом. Такий підхід відповідає вимогам до захисту API і допомагає мінімізувати ризики несанкціонованих змін [19, 33, 34]. Для юридично значущих сценаріїв додатково важлива фіксація авторства і часу виконання дій.

У практичній реалізації на стеку Next.js, TypeScript, Prisma і PostgreSQL API-проєктування повинно підтримувати узгодженість між контрактом, серверною логікою і схемою даних. TypeScript забезпечує типову цілісність на межі інтерфейсів [42], Prisma — контроль доменних зв'язків і міграцій [37], PostgreSQL — транзакційну надійність у критичних сценаріях [36]. За такого підходу зміни API не відокремлюються від еволюції моделі даних, а проходять керований цикл перевірки сумісності.

Для зниження технічного ризику доцільно застосовувати версіонування API і регламент керування змінами контрактів. Кожна зміна, що впливає на клієнтів або суміжні модулі, повинна мати чітко визначений перехідний період, оновлену специфікацію і сценарії тестування сумісності. Така дисципліна особливо важлива для платформи з великою кількістю функціональних модулів, де локальна правка одного маршруту може впливати на кілька бізнес-контурів одночасно [9].

Отже, проєктування API та інтеграційних механізмів для Mrion Platform базується на контрактному підході, доменній узгодженості, комбінованій синхронно-подієвій взаємодії та рольовому захисті операцій. Це забезпечує кероване розширення системи, стабільну міжмодульну інтеграцію і надійну основу для наступного підрозділу, присвяченого безпеці, відмовостійкості та масштабованості платформи.

2.5. Забезпечення безпеки, відмовостійкості та масштабованості

Для цифрової будівельної платформи питання безпеки, відмовостійкості та масштабованості є не допоміжними, а системоутворюючими. У контексті Mrion Platform це особливо важливо, оскільки система одночасно працює з організаційними, технічними й фінансовими даними проєкту, а також підтримує рольову взаємодію великої кількості учасників. Відповідно, архітектурні рішення мають забезпечувати не лише функціональність модулів, а й стабільну роботу платформи в умовах зростання навантаження, змін процесів і потенційних ризиків безпеки.

Базовим принципом інформаційної безпеки є модель «мінімально необхідних прав», реалізована через RBAC-контур. Для практичного застосування це означає, що доступ надається не за формальною належністю до підрозділу, а за конкретною відповідальністю у межах проєктного процесу. Такий підхід знижує ймовірність несанкціонованих дій і підвищує прозорість аудиту. На рівні методології він узгоджується з сучасними стандартами керування інформаційною безпекою [2, 11, 19].

Окремо має бути забезпечений захист API-поверхні, оскільки саме API є основною точкою міжмодульної та зовнішньої взаємодії. Критичними тут є валідація вхідних даних, контроль доступу до ресурсів, обмеження частоти запитів, журналювання подій і кероване опрацювання помилок. Із практичної точки зору це дозволяє зменшити ризики експлуатації типових вразливостей прикладного рівня [32, 33, 34]. Для платформи з великою кількістю бізнес-операцій така дисципліна є обов'язковою умовою стабільного розвитку.

У контурі даних важливо забезпечити цілісність і відновлюваність інформації. Це досягається через транзакційний режим роботи БД, контрольовані міграції схеми, регулярні резервні копії та перевірювані процедури відновлення. Для Mrion Platform, де зв'язки між задачами, документами, бюджетом і QA/HSE є взаємозалежними, втрата цілісності навіть окремого фрагмента даних може мати каскадний вплив на управлінські рішення. Тому стійкість сховища повинна

оцінюватися не лише за фактом збереження даних, а й за здатністю швидко повернути систему в консистентний стан [24].

Відмовостійкість платформи в архітектурному сенсі означає здатність підтримувати ключові функції при часткових збоях інфраструктури або сервісних компонентів. Для цього доцільно застосовувати ізоляцію критичних сценаріїв, контроль тайм-аутів, повторні спроби для безпечних операцій та механізми деградації сервісу без повної втрати доступності. У практиці проектного середовища важливо, щоб навіть при локальних інцидентах користувачі могли виконувати базові операції: перегляд стану задач, роботу з документами і фіксацію подій контролю.

Масштабованість для Mrion Platform слід розглядати у двох площинах: функціональній і навантажувальній. Функціональна масштабованість означає можливість додавати нові модулі або розширювати наявні без руйнування існуючих контрактів. Навантажувальна масштабованість пов'язана зі зростанням кількості проектів, користувачів, API-запитів і аналітичних обчислень. Для обох випадків критичною є модульна декомпозиція та контрольована еволюція інтерфейсів, що узгоджується із загальними принципами архітектурної якості [6, 18]. Додатково важливими є моніторинг продуктивності, автоматизоване балансування навантаження та прогнозування ресурсних потреб системи.

Розглянуті характеристики масштабованості є частиною ширшого переліку нефункціональних вимог, що визначають якість платформи в цілому. Для їх систематизації застосовано модель ISO/IEC 25010, а повний перелік вимог із пріоритетами та засобами реалізації наведено в табл. 2.1.

Таблиця 2.1.

Нефункціональні вимоги за ISO/IEC 25010

Характеристика	Підхарактеристика	Вимога для платформи	Пріоритет	Засіб реалізації в стеку
Безпека	Контроль доступу, конфіденційність, аудит	RBAC-модель з перевіркою прав на рівні операцій; журнал усіх критичних дій; захищений API-контур	Критичний	Middleware авторизації, JWT, логування Prisma
Надійність	Відмовостійкість, відновлюваність	Транзакційна цілісність при збоях; автовідновлення без втрати даних; резервне копіювання	Критичний	PostgreSQL transactions, backup-процедури
Продуктивність	Ефективність часу відповіді, використання ресурсів	Відповідь API ≤ 500 мс при типовому навантаженні; аналітичні запити не впливають на транзакційний контур	Високий	Індекси PostgreSQL, кешування, read-оптимізація
Сумісність	Інтероперабельність, сумісне існування	Стабільні API-контракти; підтримка інтеграцій без порушення внутрішньої моделі; версіонування	Високий	OpenAPI-специфікація, версіонування маршрутів
Масштабованість	Функціональна та навантажувальна	Нові модулі без перебудови ядра; підтримка зростання кількості проєктів, ролей і подій	Середній	Модульна архітектура, контейнеризація
Переносимість	Адаптованість, встановлюваність	Можливість розгортання в різних хмарних середовищах без залежності від одного вендора	Середній	Docker, CI/CD pipeline, env-конфігурація

Джерело: складено автором на основі ISO/IEC 25010:2011 [18].

Інфраструктурний шар також впливає на стійкість і масштабування системи. Практики контейнеризації та керування контейнерами дозволяють стандартизувати середовища виконання і керувати ресурсами більш прогнозовано [7, 25]. У поєднанні з моніторингом метрик продуктивності та стабільності це

формує основу для проактивного управління платформою, коли проблеми фіксуються до того, як переходять у критичні інциденти [16]. Екран моніторингу стабільності API з табличним поданням ключових метрик наведено на рис. Г.2 (дод. Г).

Для оцінювання рівня захищеності та стійкості доцільно використовувати операційні критерії: частота інцидентів доступу, час виявлення і локалізації збою, відсоток успішного відновлення сервісів, стабільність API при піковому навантаженні, а також частка змін, що проходять без регресій у суміжних модулях. Такий підхід дозволяє перевести безпеку і відмовостійкість із декларативної площини у вимірюваний контур керування [36].

Отже, для Mrion Platform забезпечення безпеки, відмовостійкості та масштабованості має реалізовуватися як єдина архітектурна стратегія: рольовий контроль доступу, захищений API-контур, транзакційна цілісність даних, керована інфраструктура та постійний моніторинг. Саме така стратегія створює практичну основу для стабільної експлуатації платформи і переходу до розділу 3, де розглядається реалізація та тестування прийнятих рішень.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПЛАТФОРМИ

3.1. Технологічний стек і середовище розробки

Реалізація цифрової будівельної платформи потребує такого технологічного стеку, який одночасно забезпечує швидкість розробки, керованість доменної логіки і стабільність експлуатації. Для Mrion Platform вибір інструментів сформовано з урахуванням особливостей предметної області: багатомодульна структура, рольовий доступ користувачів, велика кількість взаємопов'язаних сутностей і потреба в контрольованій еволюції функціоналу. Тому стек розглядається не як набір окремих технологій, а як єдина інженерна система.

Базовим каркасом прикладного рівня виступає Next.js, який дозволяє будувати структуровану серверно-клієнтську архітектуру з уніфікованим маршрутизатором та зрозумілою організацією API-шару [29]. Для платформи такого типу це важливо, оскільки веб-інтерфейс, бізнес-логіка і серверні інтеграції мають працювати узгоджено в межах одного проектного контуру. Використання TypeScript доповнює цей підхід за рахунок типобезпечності: контракти між модулями стають перевірюваними на етапі розробки, що знижує ризик помилок при масштабуванні функціональності [42].

На рівні моделі даних Mrion Platform спирається на Prisma як керований шар доступу до БД [37]. Практичне значення цього рішення полягає в тому, що доменні сутності та зв'язки між ними формалізуються в єдиній схемі, а міграції виконуються контрольовано і прозоро. У поєднанні з PostgreSQL це дозволяє підтримувати транзакційну цілісність для взаємозалежних сценаріїв, де одночасно залучені задачі, документи, бюджетні операції і події QA/HSE [36]. З погляду архітектурної стійкості такий підхід відповідає вимогам до керованого data-intensive контуру [24].

Важливою особливістю стеку є його відповідність модульній архітектурі. Функціональні домени платформи реалізуються як відокремлені логічні блоки з чіткими межами відповідальності, а взаємодія між ними проходить через API-контракти. Це узгоджується з практиками сучасного проектування архітектури

програмних систем, де ключовими є декомпозиція, контрактна взаємодія і контроль еволюції рішень [6, 28]. Для будівельного контексту такий підхід зменшує ризик каскадних змін у суміжних модулях.

Середовище розробки орієнтоване на відтворюваність і командну роботу. У практичному процесі це означає стандартизовану конфігурацію запуску, контроль версій коду, формалізований цикл інтеграції змін і автоматизовані перевірки ключових сценаріїв. Для підтримки стабільності застосовуються інструменти контейнеризації та CI-процеси, що дозволяють синхронізувати локальні, тестові й робочі середовища [7, 13, 14, 15, 36]. Така організація особливо важлива для платформного продукту, який розвивається ітеративно і потребує частих релізів без втрати якості [5, 8, 9].

Для відображення організації процесу розробки та порядку просування змін між середовищами розроблено схему CI/CD-пайплайну, що охоплює етапи від коміту розробника до розгортання в продакшн-середовищі. Схему CI/CD-пайплайну та середовища розробки Mrion Platform наведено на рис. 3.1.

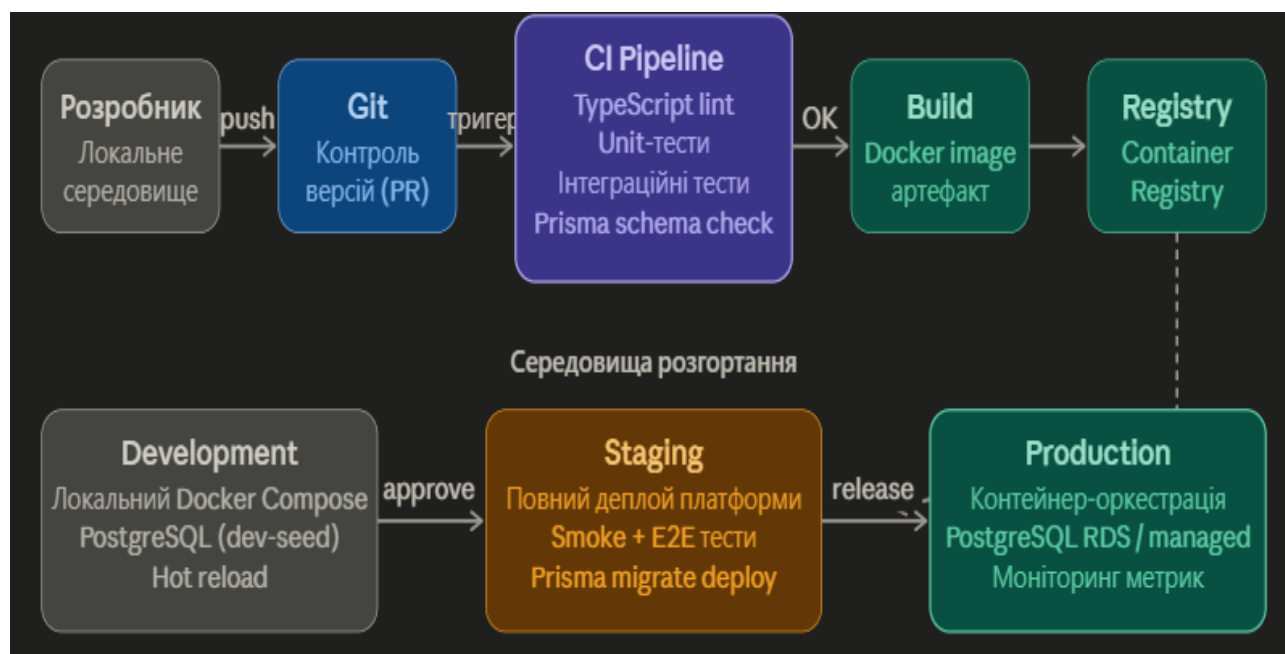


Рис. 3.1. Схема CI/CD-пайплайну та середовища розробки

Джерело: розроблено автором на основі підходів DevOps та CI/CD [5, 8, 9], документації Docker Docs, GitHub Actions documentation, GitLab Docs, Kubernetes Documentation та Prisma Documentation.

Окрему роль у середовищі розробки відіграє спостережуваність системи. Для практичної експлуатації важливо не лише фіксувати факт помилки, а й розуміти її контекст: де вона виникла, який модуль залучений, яка роль виконувала операцію і який вплив це має на бізнес-процес. Тому стек доповнюється моніторинговим контуром, що дозволяє відстежувати метрики продуктивності, навантаження і стабільності сервісів [16]. Це створює підґрунтя для проактивного управління якістю платформи.

З урахуванням специфіки будівельної платформи критичним елементом середовища є безпечний контур доступу. Технологічна конфігурація повинна підтримувати RBAC-модель, аудит дій і захищену роботу API, оскільки система оперує фінансовими, документними та операційними даними проєкту. У цьому аспекті стек має відповідати актуальним підходам до управління інформаційною безпекою і захисту прикладних інтерфейсів [19, 28, 34].

Практичний ефект обраного стеку проявляється в балансі між гнучкістю і керованістю. З одного боку, платформа може швидко розширювати модулі та сценарії роботи; з іншого — зміни проходять через контрольовані контракти, типізацію і формалізовану модель даних. Саме така комбінація є критичною для довготривалого розвитку системи, де кількість доменних сутностей і бізнес-правил постійно зростає.

Отже, технологічний стек і середовище розробки Mrion Platform сформовані як взаємопов'язаний архітектурний контур: Next.js і TypeScript забезпечують структуровану прикладну реалізацію, Prisma і PostgreSQL - керовану модель даних, а інфраструктурні практики CI/CD, контейнеризації та моніторингу - стабільність розвитку й експлуатації. Така основа є достатньою для переходу до наступного підрозділу, де деталізується реалізація серверної частини та бізнес-логіки.

3.2. Реалізація серверної частини та бізнес-логіки

Серверна частина цифрової будівельної платформи виконує роль керуючого контуру, у якому поєднуються правила предметної області, контроль доступу,

узгоджена робота з даними і міжмодульна взаємодія. Для Mrion Platform це особливо важливо, оскільки більшість критичних операцій проходить саме через серверний шар: створення і ведення проєктів, життєвий цикл задач, погодження документів, бюджетні зміни та фіксація подій QA/HSE. Відповідно, якість реалізації серверної логіки напряду впливає на стабільність усієї системи.

Архітектурно серверна частина реалізована за модульним принципом, де кожний домен має власний набір маршрутів і операцій, але працює в межах єдиного проєктного контексту. Така організація дозволяє відокремити відповідальність модулів і знизити ризик каскадних змін при розвитку функціоналу. Практично це означає, що зміни в задачному або документному контурі не повинні порушувати бюджетні чи аналітичні сценарії без явного перегляду контрактів взаємодії [6, 28].

На рівні прикладної реалізації серверний шар будується на Next.js API-маршрутах у поєднанні з TypeScript [29, 42]. Це дає можливість типізувати вхідні параметри, структури відповідей і внутрішні сервісні операції, що зменшує кількість помилок на стиках модулів. Для платформи з великою кількістю доменних сутностей така типова дисципліна є критичною, оскільки бізнес-логіка постійно розширюється і має залишатися керованою.

Логіка обробки запитів у серверній частині реалізується за послідовністю «валідація -> авторизація -> доменна операція -> транзакційний запис -> аудит події». Такий порядок дозволяє забезпечити коректність виконання кожної дії і відтворюваність результату в операційному журналі. Для бізнес-процесів будівельного проєкту це важливо, бо будь-яка критична зміна (статус задачі, погодження документа, корекція бюджету) повинна мати зрозумілу історію виконання і відповідального виконавця. Погодження документа наведено на рис. Е.1 (дод. Е), а обробку QA/HSE-події - на рис. Е.2 (дод. Е).

Окрему роль відіграє доменна бізнес-логіка. У Mrion Platform вона охоплює правила переходів статусів, перевірки допустимості дій за ролями, контроль взаємозв'язків між сутностями та обмеження на конфліктні операції. Із позиції доменно-орієнтованого підходу це відповідає принципу, що бізнес-правила мають

бути зафіксовані в серверному контурі, а не розпорошені між клієнтом і БД [3, 10]. Такий підхід підвищує передбачуваність системи і спрощує супровід.

Підсистема доступу реалізується через RBAC-політику, де права визначаються за роллю користувача в межах конкретного проєкту. Це дозволяє уникнути ситуацій, коли користувач має загальний доступ до модуля, але не повинен виконувати окремі критичні операції. Для серверної логіки це означає, що перевірка прав виконується на рівні endpoint-а і доменної операції, а результат перевірки фіксується в аудиті. Така практика відповідає сучасним вимогам до захисту API і керування інформаційною безпекою [11, 19, 34].

Контур роботи з даними реалізований через Prisma як шар доступу до PostgreSQL [36, 37]. Це забезпечує структуровану взаємодію з моделлю БД, контроль міграцій і транзакційну узгодженість у сценаріях, де змінюються кілька взаємопов'язаних сутностей. Для практичних процесів Mrion Platform це критично у випадках, коли одна дія користувача впливає одночасно на задачний, документний і бюджетний контури. У таких операціях серверна логіка повинна або завершити весь ланцюг змін успішно, або відкотити його без часткового пошкодження стану даних [24].

Для інтеграції між модулями і зовнішніми споживачами серверна частина використовує контрактний API-підхід. Це означає, що структура запитів і відповідей, коди помилок і правила версіонування мають бути стабільними й документованими. У практиці реалізації це знижує ризик несумісності клієнтських компонентів і полегшує контроль еволюції endpoint-ів у часі [12, 31].

Окремо слід відзначити роль серверного журналювання і моніторингу. Для операційної стійкості платформи важливо мати не лише технічні логи, а й подієві записи доменної логіки: хто, коли і яку дію виконав, у якому модулі та з яким результатом. Це дозволяє швидше локалізувати причини відхилень у бізнес-процесах і підвищує якість управлінських рішень у проєктному контурі [16].

Отже, реалізація серверної частини та бізнес-логіки Mrion Platform базується на модульній архітектурі, типобезпечному API-шарі, рольовому контролі доступу, транзакційній роботі з даними і контрактній інтеграції. Така організація забезпечує

керованість складних будівельних процесів у цифровому середовищі та створює основу для наступного підрозділу, присвяченого реалізації клієнтської частини і ролей користувачів.

3.3. Реалізація клієнтської частини та ролей користувачів

Клієнтська частина цифрової будівельної платформи є операційним середовищем, у якому реалізується щоденна робота учасників будівельного проєкту. Саме через інтерфейс користувачі формують задачі, ведуть документообіг, контролюють бюджетні зміни, фіксують QA/HSE-події та отримують аналітичні зрізи. Тому під час реалізації фронтенду для Mrion Platform важливо забезпечити не тільки зручність інтерфейсу, а й архітектурну узгодженість із серверними контрактами та доменною моделлю даних.

Загальна логіка клієнтського контуру побудована за модульним принципом, де структура екранів відповідає основним доменам платформи: проєкти, задачі, документи, бюджет, QA/HSE, повідомлення, аналітика і налаштування. Така декомпозиція підтримує керованість предметної області та знижує ризик помилкових дій через змішування процесів в одному робочому просторі. Це дає можливість користувачу працювати в межах чітко визначеного сценарію без втрати контексту проєкту [18,20]. Ілюстративні приклади екранів інтерфейсу наведено на рис. Ж.1–Ж.4 (дод. Ж).

Технологічно клієнтська частина реалізується на базі Next.js та TypeScript [29, 42], що дозволяє поєднати компонентну організацію інтерфейсу з типобезпечною взаємодією з API. Для платформи, де велика частка бізнес-операцій проходить через форми, фільтри, статусні переходи й асинхронні запити, типізація є критичною умовою стабільності. Вона зменшує кількість розбіжностей між клієнтським поданням і серверною моделлю, а також спрощує супровід при розвитку функціоналу.

Ключовим елементом реалізації є рольова поведінка інтерфейсу. У Mrion Platform RBAC-модель повинна відображатися не тільки в перевірках на сервері, а

й на рівні клієнта: доступні розділи, кнопки дій, сценарії погодження та статусні операції мають відповідати ролі користувача у конкретному проєкті. Такий підхід знижує операційний шум і підвищує безпекову дисципліну, оскільки користувач не взаємодіє з функціями, які не входять до його зони відповідальності [11, 19].

Реалізація ролей у клієнтському шарі доцільно будується за сценарним принципом. Для керівних ролей пріоритетними є оглядові панелі, контроль відхилень і агреговані метрики; для виконавчих ролей — робота із задачами, документами та статусами; для ролей контролю якості та безпеки — швидка фіксація інцидентів і контроль коригувальних дій. Така диференціація інтерфейсу підвищує його практичну придатність у щоденній роботі.

З метою ілюстрації рольово-орієнтованої організації клієнтського шару розроблено макет інтерфейсу модуля задач, що демонструє структуру робочого простору, статусну модель задач та диференціацію доступних дій залежно від ролі користувача. Макет інтерфейсу модуля задач Mrion Platform для ролі «Менеджер проєкту» наведено на рис. 3.2.

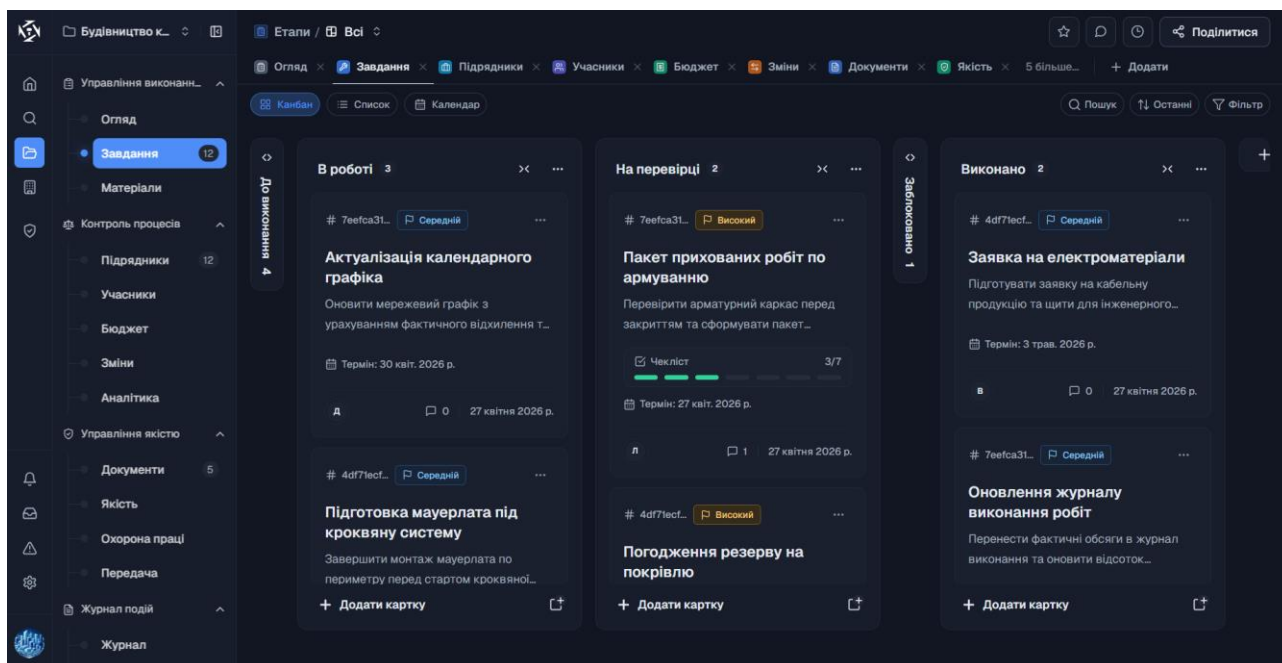


Рис. 3.2. Макет інтерфейсу модуля задач

Джерело: авторська розробка відповідно до концепції цифрової платформи управління будівельними проєктами [2].

Окрему увагу приділено узгодженості станів інтерфейсу: завантаження, часткове оновлення даних, валідаційні помилки, конфлікти редагування, результат асинхронних операцій. У будівельних процесах користувач повинен чітко розуміти, чи зміна дійсно зафіксована в системі, чи потребує повторного виконання. Саме тому клієнтська частина має відображати не лише «успішний» сценарій, а й контрольовану поведінку в граничних ситуаціях, що напряду впливає на достовірність операційного контуру.

Для підтримки цілісності між фронтом і сервером важливо дотримуватися контрактного підходу: структура запитів, формат відповідей, коди помилок і правила версіонування повинні бути стабільними. Це зменшує ризик регресій при зміні endpoint-ів і спрощує розвиток клієнтських модулів у довгостроковій перспективі [12, 31]. У межах Mrion Platform такий підхід є критичним через велику кількість взаємопов'язаних функціональних контурів.

Із нефункціонального боку реалізація клієнтської частини повинна забезпечувати стабільність рендерингу, передбачувану продуктивність і адаптивність інтерфейсу. В умовах роботи з масивами задач, документів і подій, низька швидкодія фронту безпосередньо знижує ефективність прийняття рішень. Тому якість клієнтської реалізації оцінюється не тільки візуально, а і за критеріями надійності та операційної придатності [18].

Отже, клієнтська частина Mrion Platform реалізується як модульний рольово-орієнтований інтерфейс, синхронізований із серверною бізнес-логікою і доменною моделлю даних. Така організація забезпечує контрольовану взаємодію користувачів із ключовими процесами платформи, підвищує безпеку дій і створює практичну основу для подальшого розгляду реалізації роботи з даними будівельних об'єктів.

3.4. Реалізація роботи з даними будівельних об'єктів

Робота з даними будівельних об'єктів у цифровій платформі повинна забезпечувати одночасно повноту інформації, її узгодженість між модулями та зручність практичного використання на всіх етапах проєкту. Для Mrion Platform це

означає, що дані про об'єкти, етапи, задачі, документи, бюджети й контрольні події мають оброблятися як єдиний інформаційний контур, а не як ізольовані фрагменти в різних підсистемах. Саме така логіка дозволяє перейти від формального накопичення даних до керованого прийняття рішень.

На концептуальному рівні реалізація роботи з даними базується на проєктному контексті: кожний будівельний об'єкт належить до конкретного проєкту і пов'язаний із його структурою виконання. У межах цього контексту фіксуються ключові характеристики об'єкта, етапи реалізації, пов'язані задачі, статуси перевірок, документні підтвердження і фінансові зміни. Такий підхід забезпечує простежуваність стану об'єкта в часі та дозволяє бачити причинно-наслідкові зв'язки між подіями.

Практична реалізація передбачає використання єдиної моделі ідентифікації сутностей, у якій кожен об'єкт має стабільний ідентифікатор і зв'язки з доменними модулями платформи. Це дає змогу уникнути дублювання записів, втрати контексту при інтеграції і некоректних об'єднань у звітних вибірках. Для будівельної платформи з великою кількістю операційних сценаріїв така цілісність є критичною умовою достовірності даних [24].

Важливим елементом є статусна модель даних об'єкта. Реалізація повинна підтримувати чітко визначені стани та дозволені переходи між ними, щоб користувачі різних ролей працювали в єдиній процесній логіці. У практиці Mgrion Platform це стосується не лише об'єкта як одиниці обліку, а і пов'язаних сутностей: задач, документів, контролю якості й безпеки. Уніфікована статусна логіка зменшує ризик суперечностей між модулями та спрощує аудит змін.

Для документного контуру робота з даними об'єктів реалізується через прив'язку документів до конкретного етапу або події по об'єкту. Це дозволяє отримати повний ланцюг підтверджень: від постановки задачі до фактичного виконання і контрольної перевірки. Така модель узгоджується з вимогами до керованого інформаційного менеджменту в будівництві [20, 21] та вимогами до функціонування цифрових сервісів у сфері будівництва. У прикладному вимірі це підвищує юридичну і процесну відтворюваність рішень.

У бюджетному контурі дані об'єктів використовуються для зв'язку між фактичними роботами та фінансовими операціями. Реалізація повинна забезпечити, щоб бюджетна зміна мала посилення на конкретний об'єкт, підставу зміни та відповідальний рольовий контекст. Це дозволяє уникати ситуацій, коли фінансова подія існує без процесного обґрунтування, і підвищує контрольованість витрат у межах проєкту.

Для QA/HSE-модулів реалізація даних об'єкта має підтримувати швидку реєстрацію інцидентів, перевірок і коригувальних дій із прив'язкою до місця виникнення події та відповідального виконавця. Така деталізація є критичною для аналізу повторюваних ризиків і планування профілактичних заходів [26]. У результаті платформа переходить від реактивного усунення наслідків до проактивного контролю якості та безпеки. Екран аналітики KPI проєкту, що забезпечує управлінський контроль ключових показників ефективності у розрізі модулів платформи, наведено на рис. Г.3 (дод. Г).

На технічному рівні обробка даних об'єктів реалізується через узгоджену взаємодію API, серверної бізнес-логіки і моделі БД. У контексті стеку Next.js, TypeScript, Prisma і PostgreSQL це означає типізовані контракти обміну [29, 42], контрольовані валідації, транзакційну фіксацію змін і підтримку цілісності зв'язків між сутностями [36, 37]. Для складних міжмодульних операцій такий підхід знижує ризик часткового запису або втрати консистентності при збоях. Журнал виконання тестових сценаріїв, що підтверджує коректність обробки даних об'єктів у ключових модулях, наведено на рис. Г.1 (дод. Г).

Наведені технічні рішення перевірено в межах комплексного тестування платформи, що охоплювало функціональні, інтеграційні, безпекові та навантажувальні сценарії[23]. Зведені результати тестування ключових сценаріїв із фіксацією часу відповіді та статусу виконання наведено в табл. 3.1.

Окремо потрібно забезпечити рольовий контроль доступу до даних будівельних об'єктів. Не всі ролі повинні мати однаковий обсяг перегляду й редагування, особливо в частині бюджетних і контрольних подій. Тому доступ реалізується на рівні операцій над сутностями, а не лише на рівні модулів.

Таблиця 3.1.

Результати тестування ключових сценаріїв

#	Сценарій тестування	Модуль	Тип	Час відповіді	Результат	Примітки
1	Створення проєкту з призначенням ролей учасників	Проекти	Функціональний	124 мс	Пройдено	RBAC-перевірка коректна
2	Зміна статусу задачі з фіксацією в журналі аудиту	Задачі	Функціональний	89 мс	Пройдено	Аудит-запис збережено
3	Погодження документа — повний цикл статусів	Документи	Інтеграційний	210 мс	Пройдено	Версія та автор зафіксовані
4	Транзакційна зміна бюджету з пов'язаним документом	Бюджет	Інтеграційний	187 мс	Пройдено	Транзакція атомарна
5	Реєстрація QA-інциденту з коригувальною дією	QA	Функціональний	143 мс	Пройдено	Зв'язок з об'єктом збережено
6	Фіксація HSE-події з прив'язкою до виконавця	HSE	Функціональний	156 мс	Пройдено	Роль перевірено коректно
7	Спроба доступу до бюджету з роллю «Виконавець»	Доступ	Безпека	41 мс	Відхилено (403)	RBAC працює правильно
8	Аналітичний запит при 50 одночасних користувачах	Аналітика	Навантажувальний	448 мс	Прийнято	У межах ≤ 500 мс
9	Міграція схеми з перевіркою сумісності даних	Prisma	Регресійний	—	Пройдено	Наявні дані збережено

Джерело: складено автором на основі результатів функціонального, інтеграційного та навантажувального тестування.

Отже, реалізація роботи з даними будівельних об'єктів у Mrion Platform базується на цілісній проєктній моделі, статусній узгодженості, рольовому контролі та транзакційній надійності обробки.

ВИСНОВКИ

У кваліфікаційній роботі досягнуто поставлену мету: обґрунтовано архітектуру цифрової будівельної платформи та сформовано практичні пропозиції щодо її удосконалення на прикладі Mrion Platform. Відповідно до визначених завдань отримано такі результати.

Досліджено понятійний апарат цифрових будівельних платформ, проаналізовано сучасні технології (BIM, GIS, IoT, хмарні сервіси) та існуючі архітектурні підходи у галузі. Встановлено, що для платформ цього класу принциповими є єдині правила інформаційного обміну, цілісність життєвого циклу даних і архітектурна дисципліна між модулями. BIM-орієнтований підхід формує методичну основу структурованого управління будівельною інформацією, практична цінність якого зростає за умови коректної інтеграції з платформними механізмами контролю доступу.

Сформовано функціональні та нефункціональні вимоги до платформи, що охоплюють вимоги до якості, безпеки, масштабованості й експлуатаційної надійності. Обґрунтовано методику дослідження на основі системного підходу, порівняльного аналізу архітектурних альтернатив і сценарного аналізу користувацьких маршрутів.

Обґрунтовано модульний архітектурний стиль із чітким розмежуванням відповідальності підсистем як такий, що забезпечує знижену зв'язаність і керованість змін. Виконано декомпозицію платформи на функціональні модулі та спроектовано модель даних, що узгоджує контури проєктів, задач, бюджетів, документів і подій QA/HSE з підтримкою цілісності операційного та аналітичного контурів.

Спроектовано API та інтеграційні механізми між підсистемами на основі контрактного підходу, що забезпечує передбачувану взаємодію модулів і прозорість змін. Підтверджено необхідність раннього вбудовування політик безпеки та рольового керування доступом (RBAC) у базові архітектурні рішення як обов'язкової умови відмовостійкості та масштабованості платформи.

Впроваджено серверну та клієнтську частини платформи, рольову модель доступу і роботу з даними будівельних об'єктів. Оцінка результатів реалізації показала, що архітектурна стійкість системи визначається насамперед якістю узгодження доменної логіки, API-контрактів і ролей користувачів, а не лише вибором технологічного стеку. Верифікація рішень у реальних сценаріях експлуатації підтверджена як обов'язкова умова переходу від технічно працездатної системи до керованої цифрової платформи.

Запропоновані підходи мають пряме прикладне значення для розвитку Mgrion Platform: підвищення керованості міжмодульної взаємодії, покращення якості даних, посилення контролю доступу та зниження інтеграційних ризиків. Отримані результати можуть бути використані як методична основа для масштабування подібних платформних рішень у сфері цифрового будівництва.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ковалевич В.М., Горбовий А.Ю. Архітектура цифрової будівельної платформи // Теоретична та практична концептуалізація розвитку фінансово-кредитних механізмів під впливом цифровізації : матеріали Міжнародної науково-практичної конференції (м. Ірпінь, 19–20 березня 2026 р.). Ірпінь, 2026. С. 881–883.
2. Ковалевич В.М., Горбовий А.Ю. Інтелектуальна цифрова платформа управління будівельними проєктами // Інноваційні цифрові технології: від алгоритмів до безпечних систем : матеріали науково-практичної конференції (м. Ірпінь, 21 травня 2026 р.). Ірпінь, 2026.
3. Кривий В.Л., Олецкий О.В. Основи програмної інженерії : навчальний посібник. Київ : Видавничий дім «Академія», 2015. 320 с.
4. Al-Qora'n L., Al-Said Ahmad A. Modular Monolith Architecture in Cloud Environments: A Systematic Literature Review // Future Internet. 2025. Vol. 17, no. 11. Art. 496. DOI: <https://doi.org/10.3390/fi17110496>.
5. Ali J.M. DevOps and Continuous Integration/Continuous Deployment (CI/CD) Automation // Advances in Engineering Innovation. 2023. Vol. 4. P. 38–42. URL: <https://aei.ewapub.com/article/view/8286> (дата звернення: 04.03.2026).
6. Bass L., Clements P., Kazman R. Software Architecture in Practice. 4th ed. Boston: Addison-Wesley, 2021. 624 p.
7. Docker Docs [Електронний ресурс]. URL: <https://docs.docker.com/> (дата звернення: 05.03.2026).
8. Donca I.-C., Stan O.P., Misaros M., Gota D., Miclea L. Method for Continuous Integration and Deployment Using a Pipeline Generator for Agile Software Projects // Sensors. 2022. Vol. 22, no. 12. Art. 4637. DOI: <https://doi.org/10.3390/s22124637>.
9. DORA. Accelerate: State of DevOps 2023 [Електронний ресурс]. Google Cloud DevOps Research and Assessment. URL: <https://dora.dev/research/2023/dora-report/> (дата звернення: 06.03.2026).

10. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Boston : Addison-Wesley, 2003. 560 p.
11. Ferraiolo D. F., Kuhn D. R. Role-Based Access Controls // Proceedings of the 15th National Computer Security Conference. Baltimore, MD, 1992. P. 554–563. URL: <https://csrc.nist.gov/files/pubs/conference/1992/10/13/rolebased-access-controls/final/docs/ferraiolo-kuhn-92.pdf> (дата звернення: 08.03.2026).
12. Fielding R.T. Architectural Styles and the Design of Network-based Software Architectures : dissertation. University of California, Irvine, 2000. URL: <https://ics.uci.edu/~fielding/pubs/dissertation/top.htm> (дата звернення: 09.03.2026).
13. Ford N., Richards M., Sadalage P., Dehghani Z. Software Architecture: The Hard Parts. Sebastopol : O'Reilly Media, 2021. 459 p.
14. GitHub Actions documentation [Електронний ресурс]. URL: <https://docs.github.com/en/actions> (дата звернення: 10.03.2026).
15. GitLab Docs [Електронний ресурс]. URL: <https://docs.gitlab.com/> (дата звернення: 11.03.2026).
16. Grafana Documentation [Електронний ресурс]. URL: <https://grafana.com/docs/> (дата звернення: 11.03.2026).
17. IFC 4.3 Formally Approved and Published as an ISO Standard [Електронний ресурс]. buildingSMART International, 2024. URL: <https://www.buildingsmart.org/ifc-4-3-formally-approved-and-published-as-an-iso-standard/> (дата звернення: 12.03.2026).
18. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models [Електронний ресурс]. URL: <https://www.iso.org/standard/35733.html> (дата звернення: 13.03.2026).
19. ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection — Information security management systems — Requirements [Електронний ресурс]. URL: <https://www.iso.org/standard/27001> (дата звернення: 15.03.2026).

20. ISO 19650-1:2018. Organization and digitization of information about buildings and civil engineering works, including BIM — Information management using BIM — Part 1: Concepts and principles [Электронный ресурс]. URL: <https://www.iso.org/standard/68078.html> (дата звернення: 16.03.2026).
21. ISO 19650-2:2018. Organization and digitization of information about buildings and civil engineering works, including BIM — Information management using BIM — Part 2: Delivery phase of the assets [Электронный ресурс]. URL: <https://www.iso.org/standard/68080.html> (дата звернення: 17.03.2026).
22. ISO 29481-1:2025. Building information models — Information delivery manual — Part 1: Methodology and format [Электронный ресурс]. URL: <https://www.iso.org/standard/88515.html> (дата звернення: 20.03.2026).
23. Jest Documentation [Электронный ресурс]. URL: <https://jestjs.io/docs/getting-started> (дата звернення: 21.03.2026).
24. Kleppmann M. Designing Data-Intensive Applications. Sebastopol : O'Reilly Media, 2017. 616 p.
25. Kubernetes Documentation [Электронный ресурс]. URL: <https://kubernetes.io/docs/> (дата звернення: 21.03.2026).
26. Liu N., Guo D., Song Z., Zhong S., Hu R. BIM-based digital platform and risk management system for mountain tunnel construction // Scientific Reports. 2023. Vol. 13. Art. 7585. DOI: <https://doi.org/10.1038/s41598-023-34525-w>.
27. Mell P., Grance T. The NIST Definition of Cloud Computing [Электронный ресурс]. NIST SP 800-145, 2011. URL: <https://csrc.nist.gov/pubs/sp/800/145/final> (дата звернення: 21.04.2026).
28. Newman S. Building Microservices. 2nd ed. Sebastopol : O'Reilly Media, 2021. 620 p.
29. Next.js Documentation [Электронный ресурс]. URL: <https://nextjs.org/docs> (дата звернення: 23.03.2026).
30. NIST SP 800-53 Rev. 5 (upd.1). Security and Privacy Controls for Information Systems and Organizations [Электронный ресурс]. URL: <https://csrc.nist.gov/pubs/sp/800/53/r5/upd1/final> (дата звернення: 27.03.2026).

31. OpenAPI Specification v3.2.0 [Электронный ресурс]. OpenAPI Initiative, 2025. URL: <https://spec.openapis.org/oas/latest> (дата звернения: 01.04.2026).
32. OWASP Application Security Verification Standard 4.0 [Электронный ресурс]. URL: <https://owasp.org/www-project-application-security-verification-standard/> (дата звернения: 02.04.2026).
33. OWASP Cheat Sheet Series — Authentication Cheat Sheet [Электронный ресурс]. URL: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html (дата звернения: 03.04.2026).
34. OWASP Top 10 API Security Risks — 2023 [Электронный ресурс]. URL: <https://owasp.org/API-Security/editions/2023/en/0x11-t10/> (дата звернения: 21.04.2026).
35. Pappula K.K. Modular Monoliths in Practice: A Middle Ground for Growing Product Teams // International Journal of Emerging Trends in Computer Science and Information Technology. 2022. Vol. 3, no. 4. P. 53–63. DOI: <https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P106>.
36. PostgreSQL Documentation [Электронный ресурс]. URL: <https://www.postgresql.org/docs/> (дата звернения: 21.04.2026).
37. Prisma Documentation [Электронный ресурс]. URL: <https://www.prisma.io/docs> (дата звернения: 05.04.2026).
38. Richards M., Ford N. Fundamentals of Software Architecture: An Engineering Approach. Sebastopol : O'Reilly Media, 2020. 419 p.
39. Richardson C. Microservices Patterns. Shelter Island : Manning Publications, 2018. 520 p.
40. Su R., Li X. Modular Monolith: Is This the Trend in Software Architecture? // Proceedings of the 1st International Workshop on New Trends in Software Architecture (SATrends '24). Lisbon, 2024. P. 10–13. DOI: <https://doi.org/10.1145/3643657.3643911>.
41. Swagger / OpenAPI Tools Documentation [Электронный ресурс]. SmartBear Software. URL: <https://swagger.io/docs/> (дата звернения: 09.04.2026).

42. TypeScript Documentation [Электронный ресурс].

URL: <https://www.typescriptlang.org/docs/> (дата звернення: 11.04.2026).

ДОДАТКИ

Додаток А

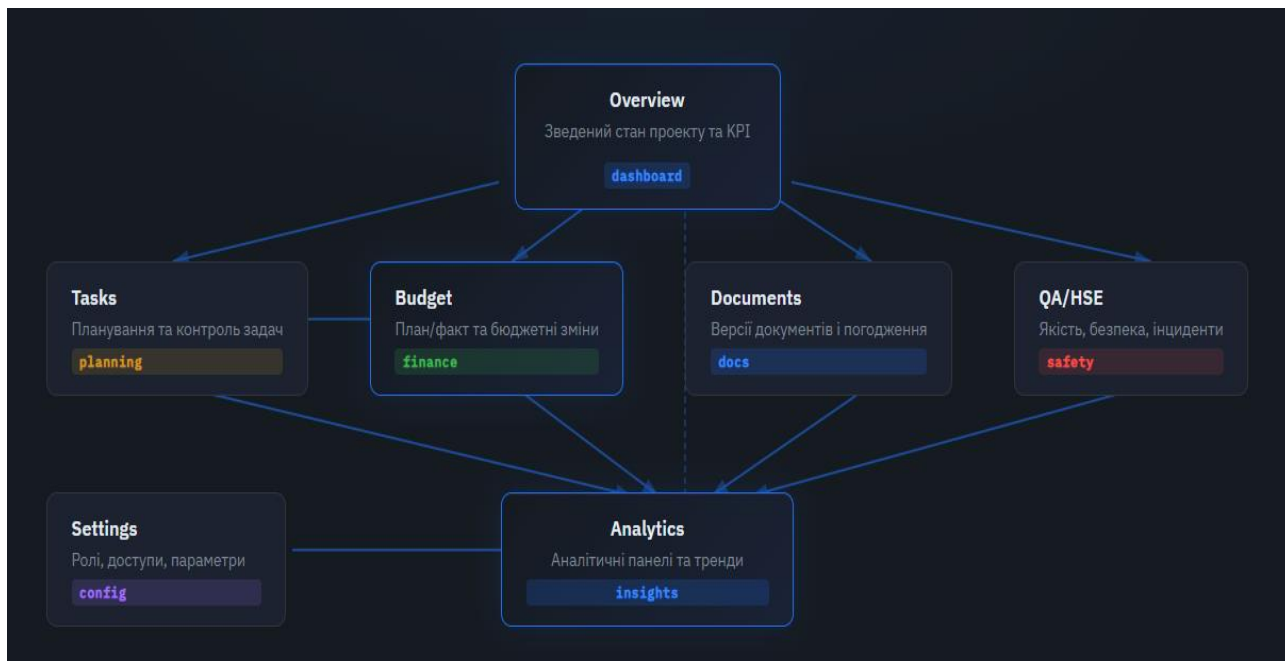


рис. А.1 — Загальна логіка функціональних модулів

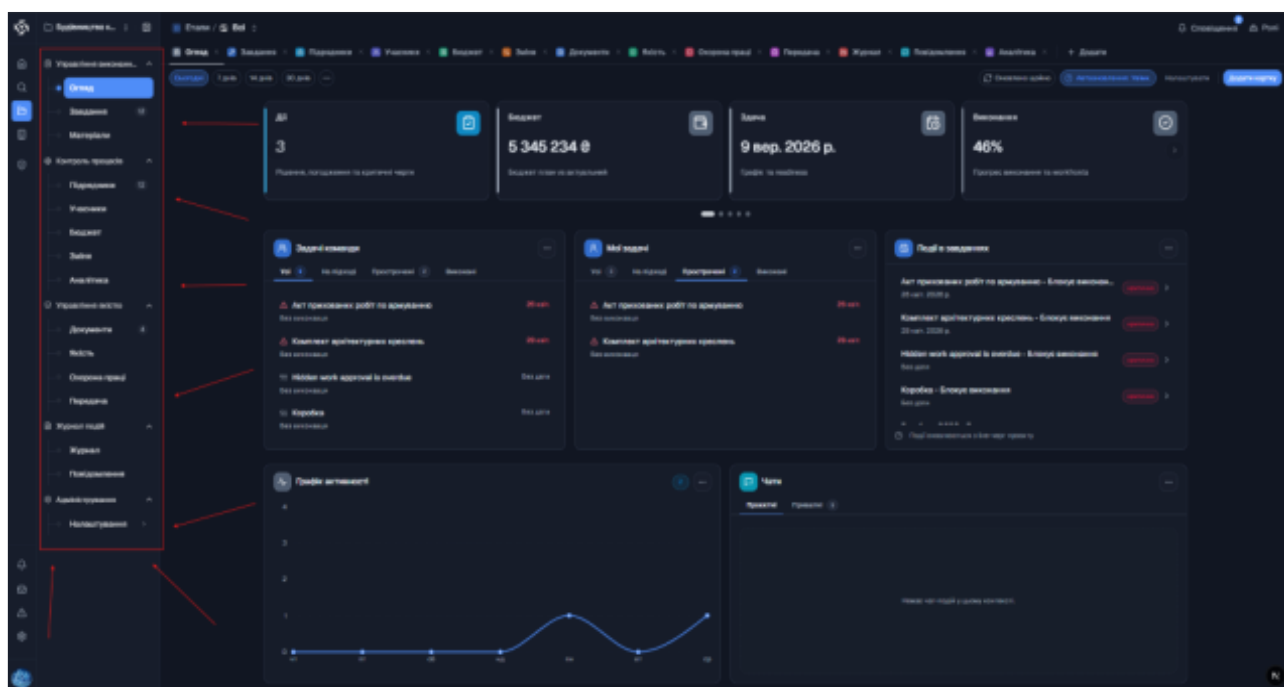


рис. А.2 — Екран структури робочого простору проекту

Додаток Б

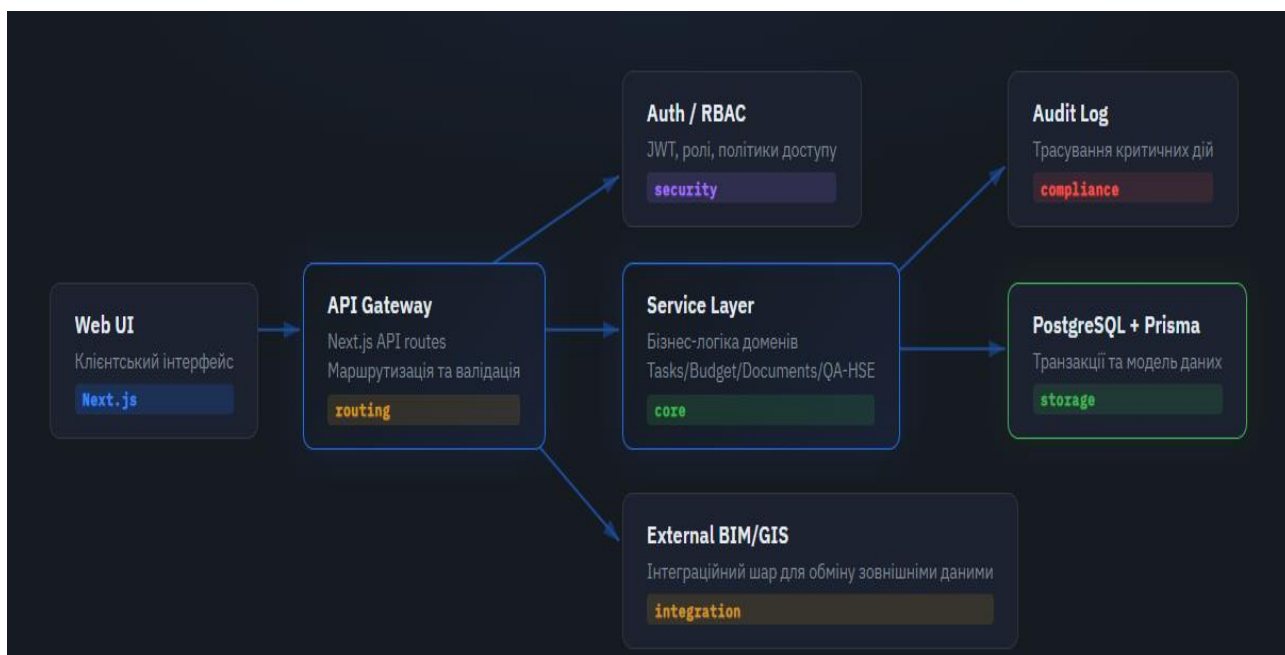


Рис. Б.1 Схема інтеграційної взаємодії модулів через API

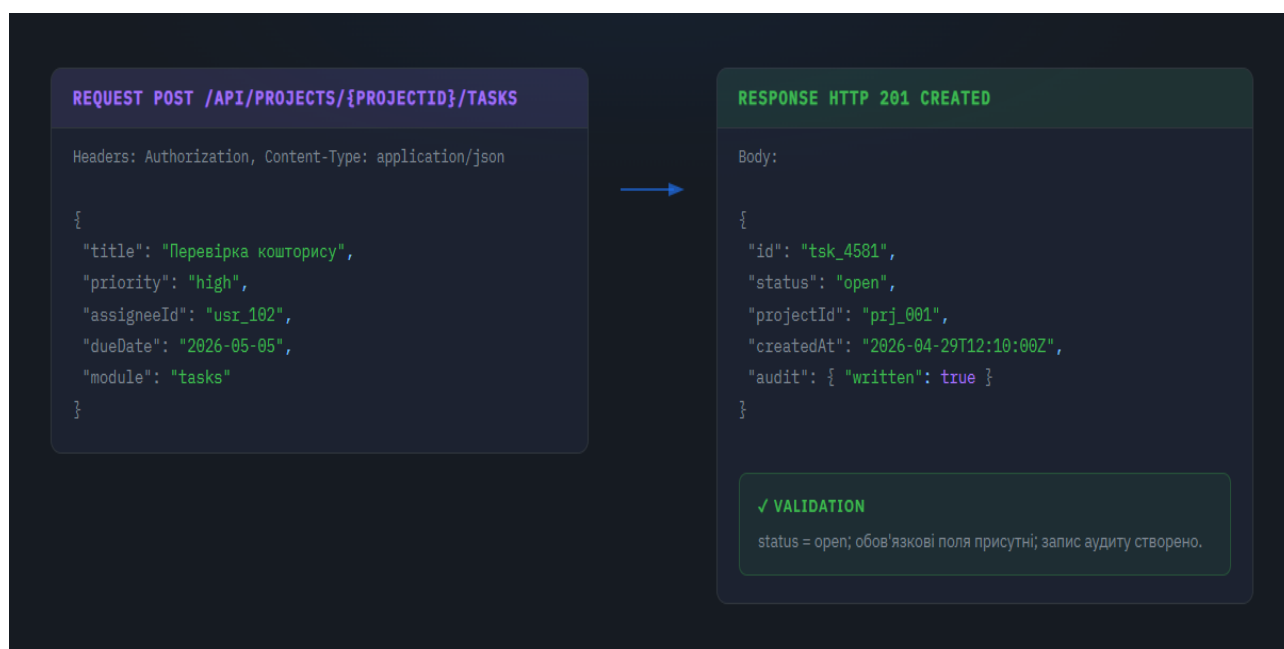


Рис. Б.2 Приклад формату запиту/відповіді API

Додаток В

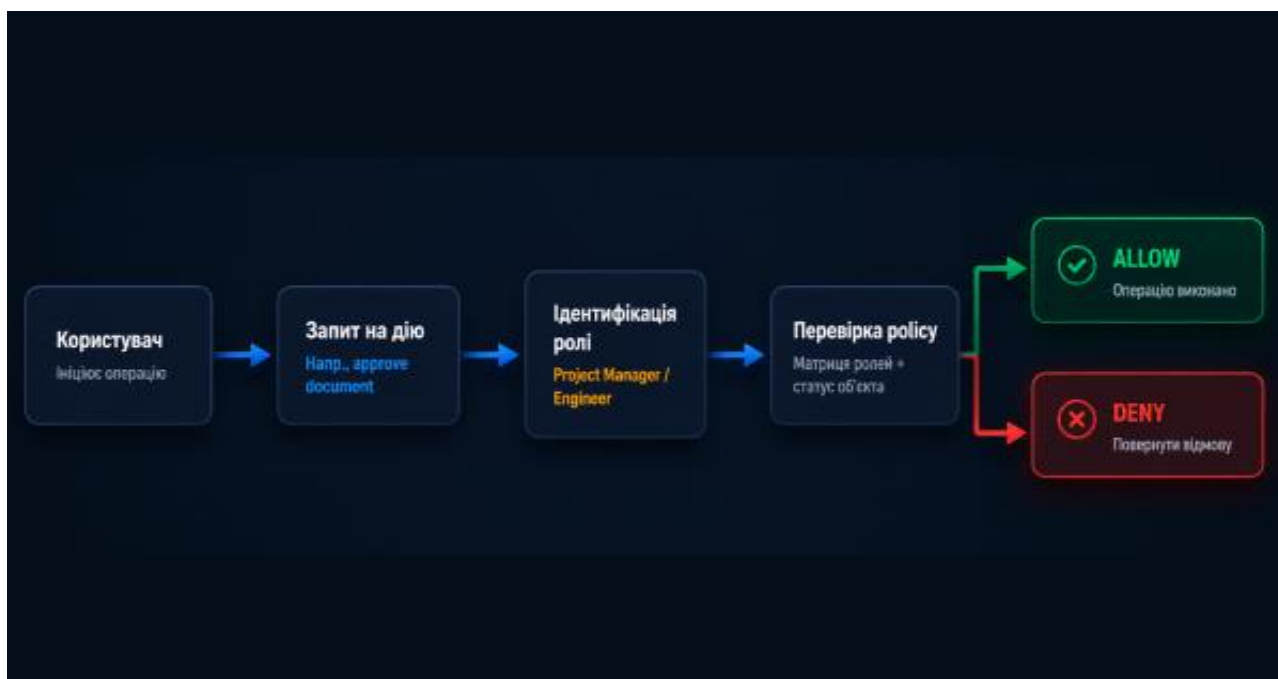


Рис. В.1 Сценарій перевірки доступу користувача до операції

Роль	Опис	Права	Учасники	Тип	...
Учасник організації	Базова присутність в організації без управлінських прав.	4 права	10	Системна	...
Адміністратор організації	Операційне адміністрування організації без прав власника.	6 прав	10	Системна	...
Власник організації	Повний контроль організації, включно з власністю та бізнесом.	4 права	1	Системна	...
Співробітник	Лише перегляд даних проекту без права внесення змін.	1 право	10	Системна	...
Прोजіктер	Виконання власних задач, фіксація виконання і поточної оцінки.	1 право	10	Системна	...
Вектор	Координація б'юджету і звітність про виконання робіт на майданчику.	1 право	10	Системна	...
Керівник майданчика	Керівництво виконанням робіт на майданчику, інспектори та координація.	1 право	10	Системна	...

Рис. В.2 Екран керування правами доступу

Продовження додатка В

Роль	OVERVIEW	TASKS	BUDGET	DOCUMENTS	QA/HSE	ANALYTICS	SETTINGS
Адміністратор платформи	+	+	+	+	+	+	+
Керівник проекту (Project Manager)	+	+	+	+	+	+	±
Інженер проекту	+	+	±	+	±	±	-
Фахівець QA/HSE	+	±	-	±	+	±	-
Фінансовий/кошторисний фахівець	±	±	+	±	-	±	-
Спостерігач/Читач	±	-	-	±	-	-	-

Рис В.3 Матриця доступу до основних модулів

Додаток Г

Час	Рівень	Категорія	Подія/опис	Ідентифікатор	Ідентифікатор
07.04.2026, 00:35:06	Info	search,telemetry	Search telemetry event: sidebar.save	alicksadler2@gmail.com	-
07.04.2026, 00:34:41	Info	search,telemetry	Search telemetry event: sidebar.query	alicksadler2@gmail.com	-
07.04.2026, 00:34:37	Info	search,telemetry	Search telemetry event: sidebar.empty	alicksadler2@gmail.com	-
07.04.2026, 00:34:36	Info	search,telemetry	Search telemetry event: sidebar.empty	alicksadler2@gmail.com	-
07.04.2026, 00:34:36	Info	search,telemetry	Search telemetry event: sidebar.empty	alicksadler2@gmail.com	-
07.04.2026, 00:34:36	Info	search,telemetry	Search telemetry event: sidebar.empty	alicksadler2@gmail.com	-
07.04.2026, 00:34:36	Info	search,telemetry	Search telemetry event: sidebar.empty	alicksadler2@gmail.com	-
07.04.2026, 00:29:19	Info	search,telemetry	Search telemetry event: sidebar.query	alicksadler2@gmail.com	-
07.04.2026, 00:29:19	Info	search,telemetry	Search telemetry event: sidebar.query	alicksadler2@gmail.com	-
07.04.2026, 00:29:19	Info	search,telemetry	Search telemetry event: sidebar.query	alicksadler2@gmail.com	-

Рис. Г.1 Екран журналу виконання тестових сценаріїв

Час	Контекст	Загальна запитівка	Деградація	Найдовший блок	Блок	Ідентифікатор
28.04.2026, 01:52:24	Завантаження	4208 мс	Рівень	Стан системи (3433 мс)	8 блоків	d8f948c0-394L
20.04.2026, 19:34:49	Завантаження	4151 мс	Рівень	Стан системи (3379 мс)	8 блоків	0cf7be38-86eL
20.04.2026, 19:31:25	Завантаження	3983 мс	Рівень	Стан системи (3203 мс)	8 блоків	edb35682-74dL
19.04.2026, 18:35:32	Завантаження	3863 мс	Рівень	Стан системи (3063 мс)	8 блоків	19707ce5-553L
19.04.2026, 18:34:24	Завантаження	4652 мс	Рівень	Стан системи (3878 мс)	8 блоків	48551096-0a6L
17.04.2026, 09:45:06	Завантаження	4489 мс	Рівень	Стан системи (3708 мс)	8 блоків	4acfa584-9f9L
17.04.2026, 08:48:56	Завантаження	4421 мс	Рівень	Стан системи (3651 мс)	8 блоків	b33014b0-77bL
17.04.2026, 08:42:52	Завантаження	3966 мс	Рівень	Стан системи (3205 мс)	8 блоків	e0807f6d-73eL
12.04.2026, 00:44:17	Завантаження	12554 мс	Критично	Стан системи (7478 мс)	8 блоків	b6fb2d6b-870L
12.04.2026, 00:44:17	Завантаження	12551 мс	Критично	Стан системи (6772 мс)	8 блоків	49ae801b-1d0L

рис. Г.2 — Екран моніторингу стабільності API (табличне подання метрик)

Продовження додатка Г

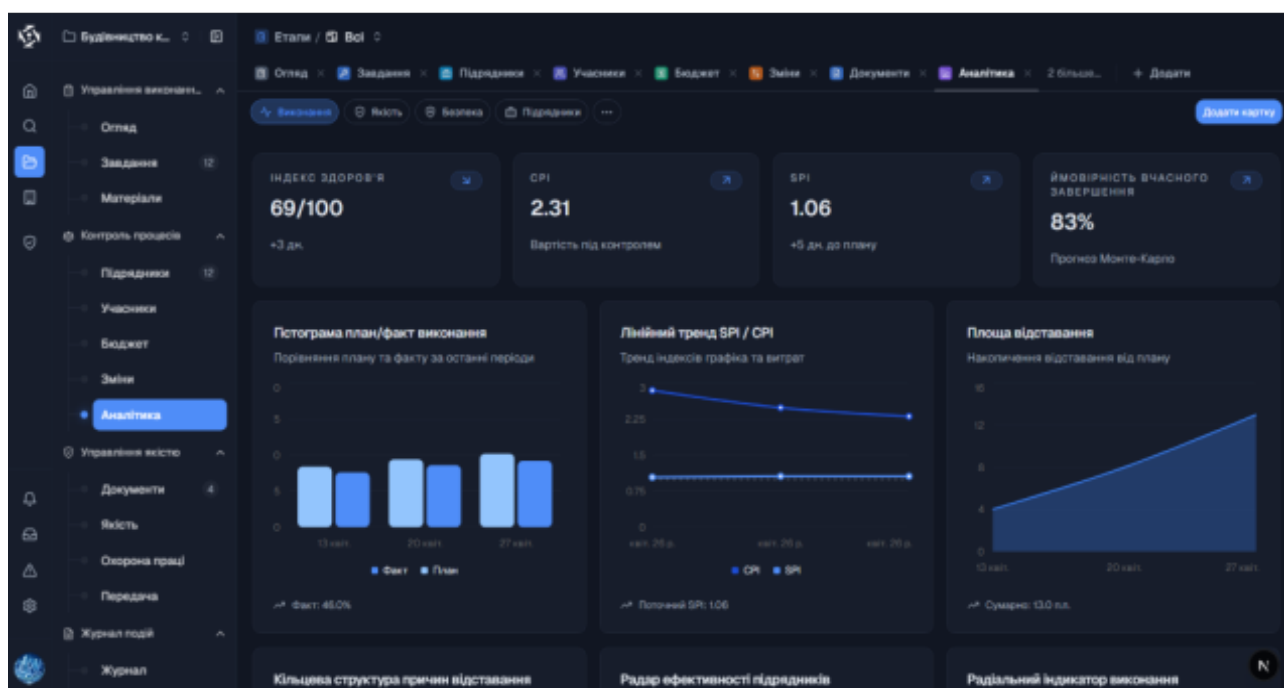


Рис. Г.3 Екран аналітики KPI проекту

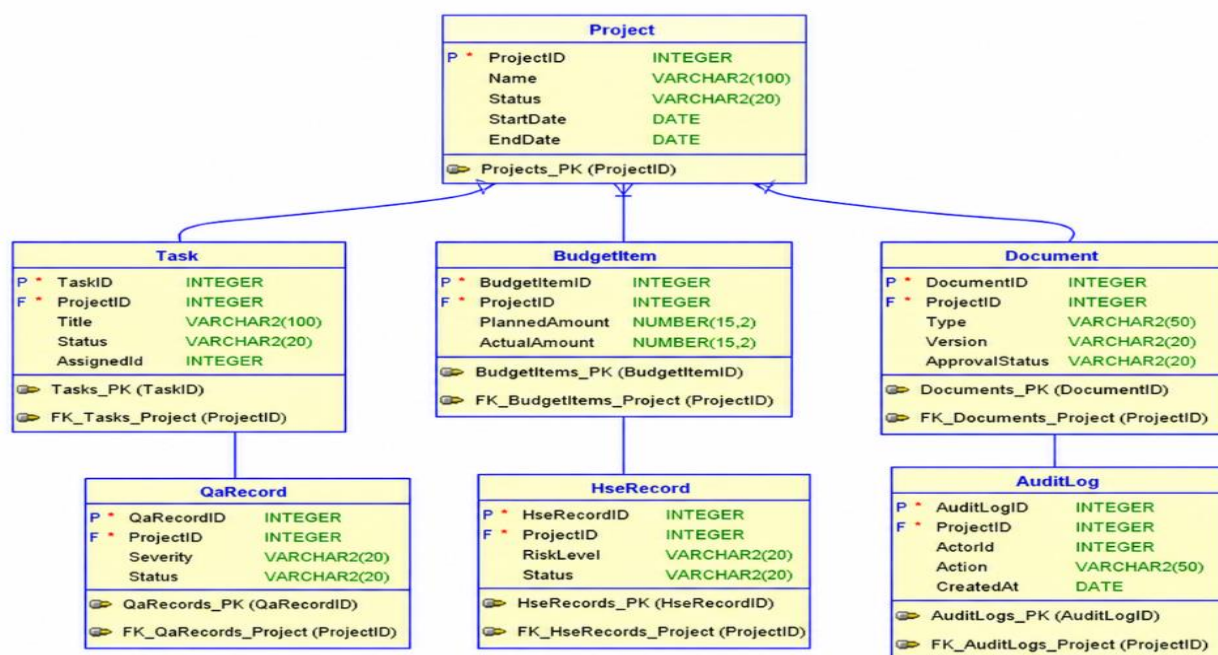


Рис. Д.1 Узагальнена ER-схема доменних сутностей

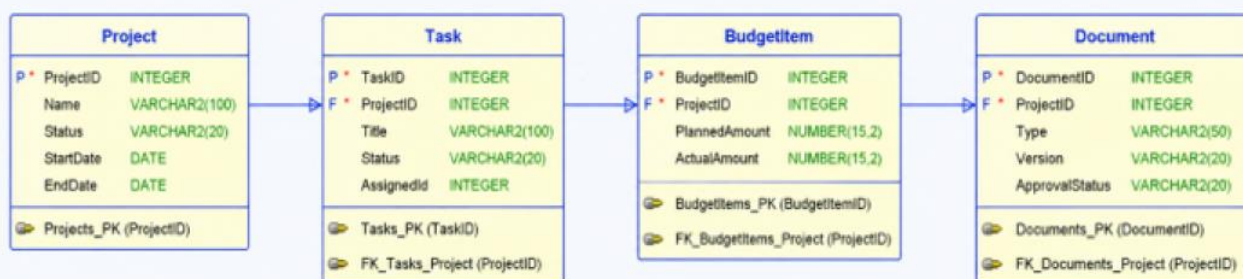


Рис. Д.2 Приклад зв'язків Project -> Task -> BudgetItem -> Document

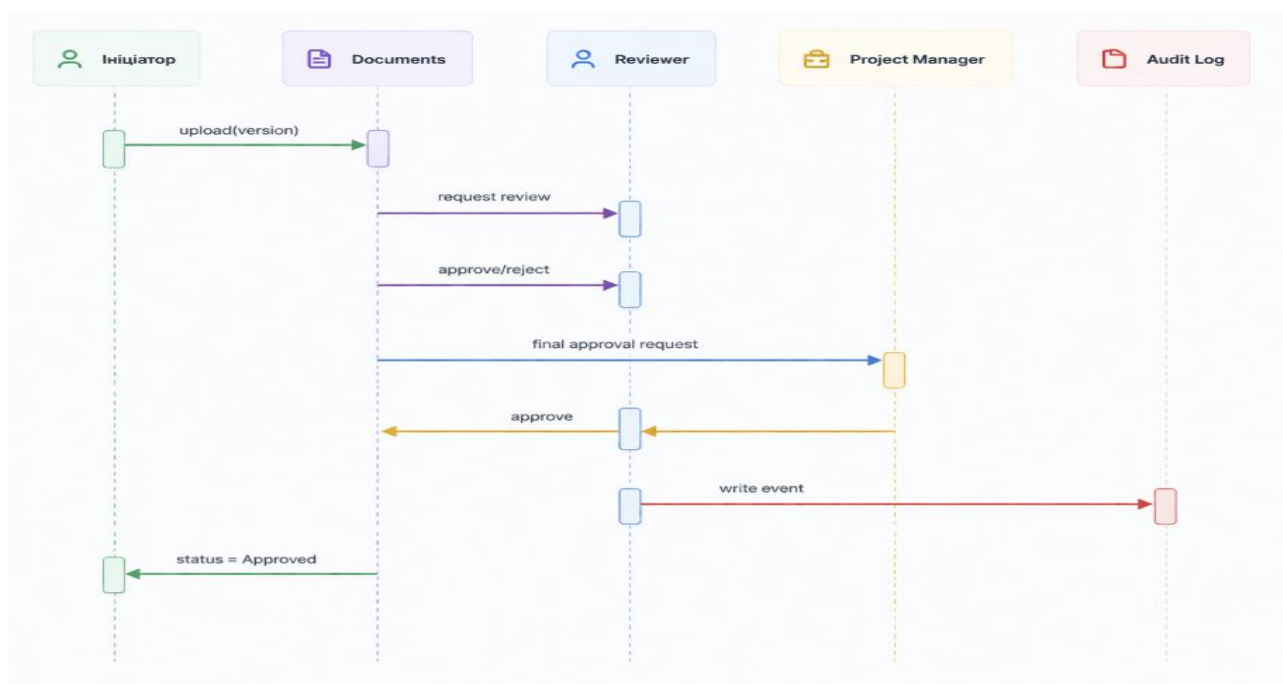


Рис. Е.1 Діаграма послідовності погодження документа

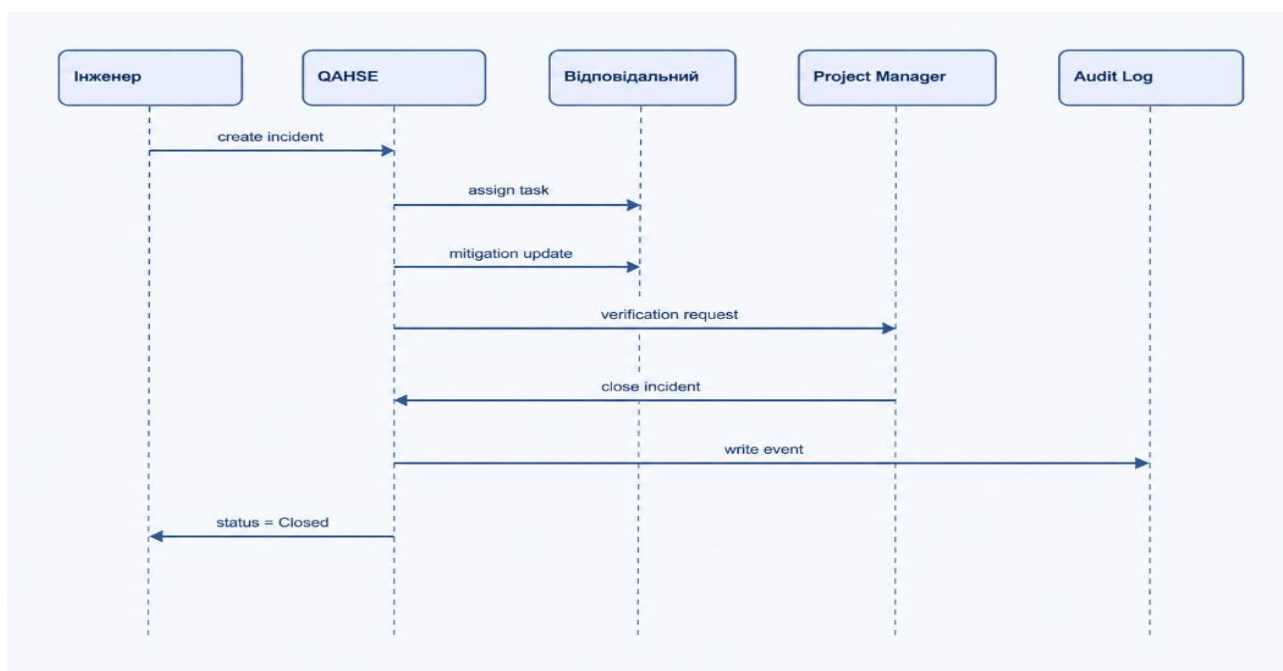


Рис. Е.2 Діаграма послідовності обробки QA/HSE-події

Додаток Ж

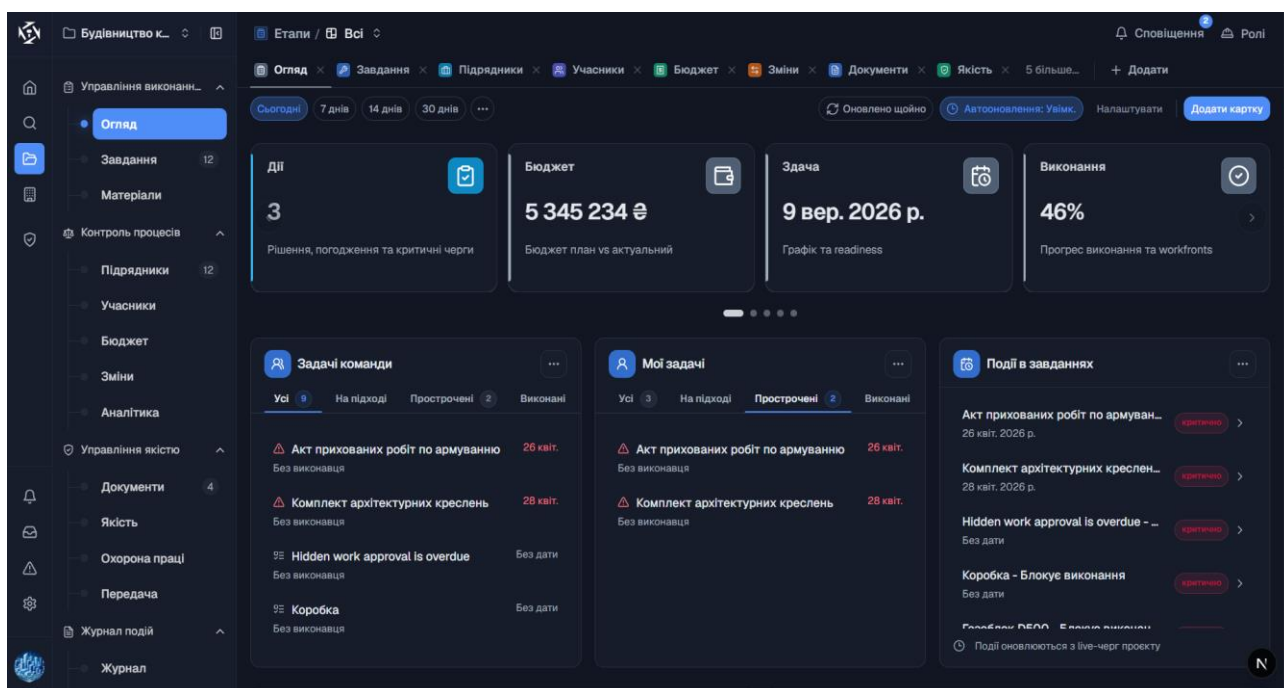


Рис. Ж.1 Екран огляду проєкту

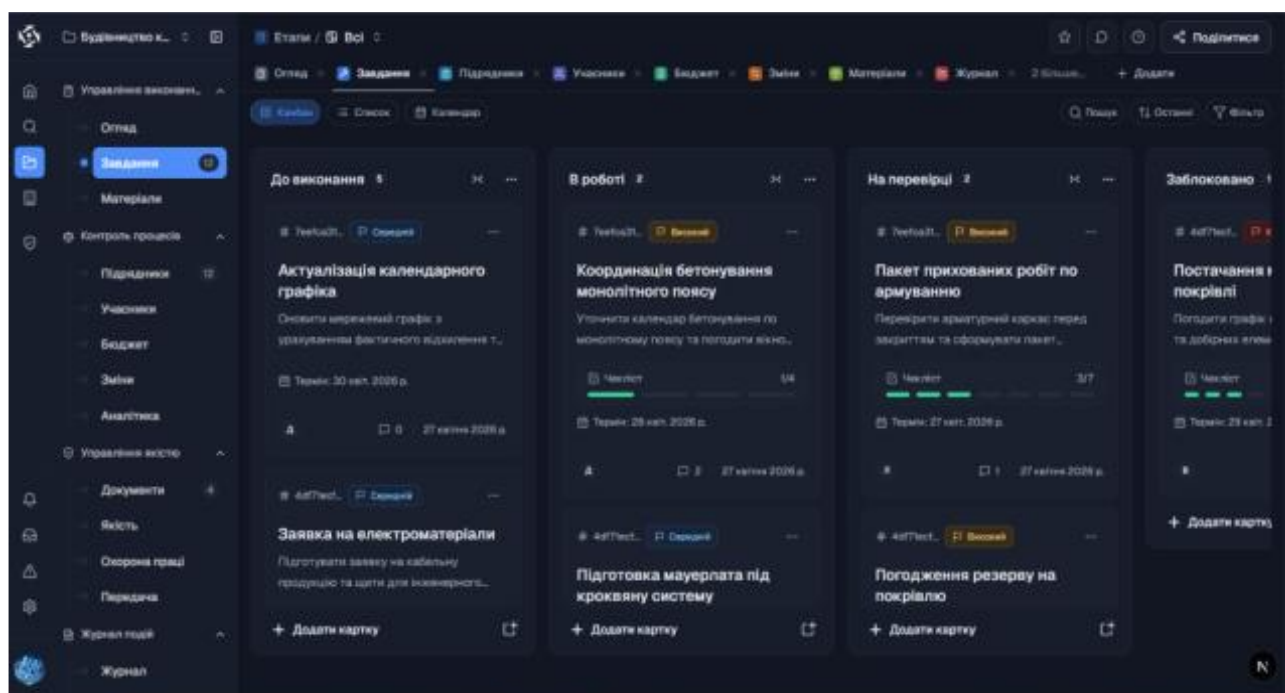


Рис. Ж.2 Екран управління задачами

Продовження додатка Ж

Уд	Бюджет	Одобрення	МЕР	Артикул WBS	Назва	Кошторис	План	Факт	БАС	Налаштування
1	Бюджет					840 000 грн	1 240 000 грн	560 000 грн	1 240 000 грн	...
1.1	Оренда техніки	Чарнетка			ТОВ «ПавлоРоман»	420 000 грн	620 000 грн	280 000 грн	80 000 грн	...
2	Одобрення					3 960 000 грн	4 640 000 грн	2 920 000 грн	4 640 000 грн	...
2.1	Підрад на коробку	Чарнетка			ТОВ «Моноліт-Буд»	1 980 000 грн	2 320 000 грн	1 460 000 грн	1 120 000 грн	...
3	МЕР					4 800 000 грн	11 660 000 грн	3 480 000 грн	11 660 000 грн	...
3.1	Монолітні роботи	Чарнетка			ТОВ «Бетон-Сервіс»	1 980 000 грн	4 150 000 грн	1 460 000 грн	0 грн	...
3.2	Матеріали покриття	Чарнетка			ТОВ «Дан-Проф»	420 000 грн	1 680 000 грн	280 000 грн	0 грн	...

Рис. Ж.3 Екран бюджетного модуля

Документ	Тип	Підписано	Статус	Версія
Комплект архітектурних ... R2	Креслення	—	На підписі	R2
Акт прихованих робіт по ... R1	Акт прихованих р...	—	Прострочено	R1
Дозвіл на виконання буд... R2	Дозвіл	—	Підписано	R2
Резюме бюджету R2	Документ	—	На підписі	R2
Чекліст підготовки до п... R0	Документ	—	На погодженні	R0

Рис. Ж.4 Екран документації та погодження



Рис. 3.1 Сертифікат участі у міжнародній студентській конференції