

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

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

Кафедра інформаційних систем в економіці

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

галузь знань	12 Інформаційні технології
спеціальність	122 Комп'ютерні науки

Форма навчання: очна (денна)

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему: «Розроблення веб платформи для управління проектами та аналізу
ефективності їх виконання»

здобувача Ільєнка Олексія Сергійовича _____
(ПІБ, підпис)

Науковий керівник: к.е.н., доцент Ситник Н.Ф.
(науковий ступінь, учене звання, ПІБ)



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

Завідувач кафедри: к.е.н., доцент Тішков Б.О. _____
(підпис)

Київ 2025

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

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

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА

«Інформаційні управляючі системи і технології»

галузь знань	12 Інформаційні технології
спеціальність	122 Комп'ютерні науки

ПОГОДЖЕНО:

Керівник проєктної групи (гарант)
освітньо-професійної програми

_____ Устенко С.В.
« ____ » _____ 2025 р.

ЗАТВЕРДЖУЮ:

Завідувач кафедри інформаційних
систем в економіці

_____ Тішков Б.О.
« ____ » _____ 2025 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувача вищої освіти Ільєнка Олексія Сергійовича

денної форми навчання

на підготовку кваліфікаційної магістерської роботи

на тему: «Розроблення веб платформи для управління проєктами та аналізу ефективності їх виконання»

Тему затверджено наказом ректора Університету від «_5_» 07. 2024 р. №214ст

Кваліфікаційна магістерська робота виконується на матеріалах наукових публікацій та матеріалів з мережі інтернет

План кваліфікаційної магістерської роботи

Розділ І ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПІДХОДІВ ДО СТВОРЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ УПРАВЛІННЯ ПРОЄКТАМИ

Розділ ІІ ХАРАКТЕРИСТИКА ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ ТА МЕТОДИ І МОДЕЛІ

Розділ ІІІ РОЗРОБЛЕННЯ ПРОЄКТНИХ РІШЕНЬ ТА ЇХ РЕАЛІЗАЦІЯ

Об'єкт дослідження: інформаційні процеси, які характеризують управління проєктами в ІТ-сфері.

Предмет дослідження є методи, моделі та інструменти розробки веб-платформ для управління проєктами та оцінювання ефективності їх виконання

Мета кваліфікаційної магістерської роботи: є проєктування та створення окремих компонентів інформаційної системи для управління проєктами, яка забезпечує моніторинг їх реалізації, контроль строків і ефективності шляхом використання інтегрованої аналітики.

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

У розділі I дослідити основні бізнес-процеси з управління IT-проєктами.. Дослідити та проаналізувати сучасні методи та підходи до управління IT-проєктами, виявити проблемні місця та намітити основні шляхи їх цифровізації. Обґрунтувати необхідність побудови WEB платформи з управління IT-проєктами

У розділі II необхідно продемонструвати проєктування WEB платформи та обґрунтувати вибір методів її проєктування. Навести перелік вимог до проєкту. При моделюванні структури WEB платформи використати сучасні системи автоматизованого інжинірингу. Розробити структуру WEB платформи та описати функціонал її блоків. Розкрити питання інформаційного моделювання системи.

У розділі III розділ повинен містити наукові та практичні результати, висвітлюючи їх новизну та оригінальність при моделюванні структури WEB - платформи та її основних складових елементів. В даному розділі мають бути викладені питання організації бази даних, технічного та програмного забезпечення роботи основних проєктних рішень з підвищення ефективності IT-проєктів

**Завдання підготував
науковий керівник**



(підпис)

Ситник Н.В.

(ініціали, прізвище)

«19» травня 2025 р.

**Завдання одержав
здобувач**

(підпис)

(ініціали, прізвище)

« ____ » _____ 2025 р.

Реферат

Кваліфікаційна магістерська робота містить 98 сторінок, 3 таблиці, 33 рисунка, список використаних джерел з 50 найменувань, додатки.

«Розроблення веб платформи для управління проектами та аналізу ефективності їх виконання»

Об'єктом дослідження кваліфікаційної магістерської роботи є інформаційні системи управління проектами в IT-сфері.

Предметом дослідження є методи, моделі та інструменти розробки веб-платформ для управління проектами та оцінювання ефективності їх виконання.

Мета і завдання дослідження. Основною метою кваліфікаційної магістерської роботи є проектування та створення окремих компонентів інформаційної системи для управління проектами, яка забезпечує моніторинг їх реалізації, контроль строків і ефективності шляхом використання інтегрованої аналітики. Відповідно до поставленої мети визначені такі *завдання*:

- здійснити аналіз предметної області управління проектами в IT-сфері та існуючих систем;
- обґрунтувати вибір архітектурного підходу, технологій і методології розробки;
- визначити структуру системи та сформулювати її функціональну модель;
- розробити логічну модель бази даних та побудувати її фізичну реалізацію;
- розробити математичне забезпечення системи;
- реалізувати механізми аналітичного супроводу виконання проєктів з використанням KPI;
- реалізувати прототип інтерфейсу застосунку;
- описати програмне забезпечення для системи;
- описати технічне забезпечення для системи;
- представити результати проектування та розробки окремих компонентів інформаційної системи для управління проектами.

Теоретична, методична та практична значущість отриманих результатів. Під час дослідження було досліджено предметну область, обґрунтовано вибір гнучкої методології Scrum, архітектури SPA на основі REST API, технологічного стеку (Node.js, React, SQL Server). Побудовано інформаційні, функціональні та логічні моделі системи, спроектовано базу знань з можливістю аналітичного аналізу KPI, визначено перспективність удосконалення систем управління проектами. Практичні результати дослідження полягають у створенні проєкта ІУС та розробці окремих компонентів системи, ключовий компонент системи (БД) є основою для механізму збору та аналізу метрик ефективності виконання проєктів в компанії.

Рік виконання кваліфікаційної магістерської роботи – 2025.

Рік захисту роботи – 2025.

Ключові слова: управління проектами, інформаційна система, KPI, веб-платформа, база даних, інтерфейс, візуалізація, аналітика.

Відгук
про кваліфікаційну магістерську роботу
здобувача навчально-наукового інституту «Інститут інформаційних технологій в економіці»

освітньо-професійної програми «Інформаційні управляючі системи і технології»

Ільска Олексія Сергійовича

на тему: «Розроблення веб платформи для управління проектами та аналізу ефективності їх виконання»

1. Актуальність теми: У сучасних умовах стрімкого розвитку інформаційних технологій ефективне управління проектами є критично важливим чинником успішної діяльності будь-якої ІТ-компанії. Зростання кількості проєктів, командна робота, необхідність дотримання строків, бюджету та технічних вимог потребують цифровізації управління ІТ-проєктами. Одним із таких рішень є створення веб-платформи управління та ведення ІТ-проєктів.

2. Позитивні риси кваліфікаційної магістерської роботи: слід віднести реалізацію запитів на мові SQL в середовищі Microsoft SQL Server для аналізу задач та їх виконавців.

3. Наявність самостійних розробок автора Описана предметна область, проведений аналіз методів управління ІТ-проєктами.

4. Цінність теоретичних висновків та практичних рекомендацій: є рекомендація створення WEB платформи для управління ІТ-проєктами.

5. Наявність недоліків: Запроєктована база даних не є достатньо повною та інформативною для управління проєктами. Замість первинних даних, які потрібно використовувати для обрахунку показників якості роботи виконавців та ефективності проєктів база містить уже розраховані дані. Наприклад, база даних містить розрахункові показники KPI ефективності проєктів, тоді як не вказано хто і якими методами їх обраховує та завантажує до бази даних. В магістерській не враховані вимоги зовнішніх стейкхолдерів, зокрема замовників ІТ-проєктів. Здобувач не чітко розмежовує поняття бази даних та бази знань. В магістерській описані всі пункти методичних вказівок, але не обгрунтовано чи всі вони необхідні для розробки WEB платформи з управління ІТ-проєктами. Робота не містить ілюстративного матеріалу результатів роботи WEB платформи з управління ІТ-проєктами.

6. Загальна оцінка кваліфікаційної магістерської роботи та її допущення до захисту перед ЕК: Представлена кваліфікаційна магістерська робота виконано на мінімальному достатньому рівні та може бути допущена до захисту з відповідною оцінкою.

Науковий керівник: професор, к.е.н. доцент Ситник Н.В

22.05.2025



Рецензія

на кваліфікаційну магістерську роботу
здобувача вищої освіти

Ільєнка Олексія Сергійовича

на тему: «Розроблення веб платформи для управління проектами та аналізу
ефективності їх виконання»

Актуальність теми кваліфікаційної магістерської роботи і доцільність її розроблення: Кваліфікаційна магістерська робота присвячена розробці системи управління проектами, що є надзвичайно актуальним завданням для сучасної ІТ-індустрії, особливо з огляду на потребу ефективного планування, контролю та аналітики у процесі реалізації ІТ-проектів. Тема є своєчасною та відповідає сучасним вимогам автоматизації управлінських процесів у сфері інформаційних технологій.

Якість проведеного дослідження: Робота охоплює всі етапи життєвого циклу інформаційної системи. Особливу увагу приділено математичному забезпеченню - розроблено формули розрахунку ключових показників ефективності, інтегрованих безпосередньо в БД. Значну увагу приділено також розробці користувацького інтерфейсу та візуальній частині платформи.

Позитивні риси кваліфікаційної магістерської роботи: Реалізовано системний підхід до структурування ІС та її компонентів, детально опрацьована логіка зберігання та аналітики даних, впроваджено повноцінний математичний апарату для оцінювання KPI, реалізовано інтерактивний прототип користувацького інтерфейсу в середовищі Figma, враховано вимоги зовнішніх користувачів, належно оформлено текстову та графічну частини.

Зауваження: у роботі бажано було б реалізувати ще декілька SQL-запитів з акцентом на аналіз проектів, а не лише задач і користувачів, було б доцільно представити результати обчислень у вигляді дашборду або звітів. Незважаючи на зазначені зауваження, робота загалом відповідає вимогам до кваліфікаційних робіт.

Розроблена авторська інформаційна система для управління проектами здобувача Ільєнка О.С. виконана на високому професійному рівні, відповідає поставленим завданням, вимогам та методичному забезпеченню до спеціальності 122 «Комп'ютерні науки», рекомендується до захисту та заслуговує на оцінку «відмінно».

Рецензент:

System Analyst (Programmer)

Advanced Analytics and Data Science, s.r.o

Дрьомін І.В.

“23” травня 2025 р.

ЗМІСТ

ПЕЛЕРІК УМОВНИХ ПОЗНАЧЕНЬ.....	4
ВСТУП.....	5
РОЗДІЛ 1 ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПІДХОДІВ ДО СТВОРЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ УПРАВЛІННЯ ПРОЄКТАМИ.....	7
1.1 Аналіз існуючих інформаційних систем управління проєктами	7
1.1.1 Дослідження предметної області.....	7
1.1.2 Дослідження ринку інформаційних систем управління проєктами	8
1.2 Обґрунтування вибору підходів і технологій для створення інформаційної системи управління проєктами.....	11
РОЗДІЛ 2 ХАРАКТЕРИСТИКА ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ ТА МЕТОДИ І МОДЕЛІ	15
2.1 Структура і характеристика системи	15
2.2 Методи та моделі в інформаційній системі управління проєктами.....	18
2.2.1 Методологічна основа	19
2.2.2 Інформаційне моделювання	20
2.3 Математичне забезпечення інформаційної системи управління проєктами	28
РОЗДІЛ 3 РОЗРОБЛЕННЯ ПРОЄКТНИХ РІШЕНЬ ТА ЇХ РЕАЛІЗАЦІЯ....	32
3.1 Проектування бази даних для інформаційної системи управління проєктами	32
3.2 Проектування бази знань інформаційної системи управління проєктами та засоби інтелектуального аналізу даних	41
3.2.1 Організація та проведення інтелектуального аналізу даних з метою пошуку знань в базі даних.....	43
3.2.2 Проектування та реалізація бази знань	47
3.3. Програмне забезпечення.....	53
3.3.1 Вимоги до програмного забезпечення	55
3.3.2 Інструментальні засоби розробки для створення прикладного програмного забезпечення інформаційної системи управління проєктами ...	62
3.3.3 Архітектура програмного забезпечення	66
3.3.4 Керівництво (інструкція) користувача.....	68
3.3.5 Опис спроектованого програмного забезпечення	69
3.3.6 Користувачський інтерфейс вебсистеми.....	72
3.4. Технічне забезпечення.....	73
3.4.1 Обґрунтування вибору технічного забезпечення	76
ВИСНОВКИ	79
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	82
ДОДАТКИ.....	87

ПЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

REST API (*Representational State Transfer API*) – Програмний інтерфейс, що реалізує архітектуру REST

SPA (*Single Page Application*) – Односторінковий веб-застосунок

JWT (*JSON Web Token*) – Токен автентифікації у форматі JSON

CRUD (*Create, Read, Update, Delete*) – Базові операції над даними

UX (*User Experience*) – Досвід користувача

HTTPS (*HyperText Transfer Protocol Secure*) – Захищений протокол передавання гіпертексту

ІС – Інформаційна система

ІУСП – Інформаційна система управління проєктами

СУБД – Система управління базами даних

KPI (*Key Performance Indicators*) – Ключові показники ефективності

EVM (*Earned Value Management*) – Управління вартістю за отриманою цінністю

ВСТУП

У сучасному світі, де інформаційні технології глибоко інтегровані у всі сфери діяльності людини, процеси управління проєктами набувають усе більшої складності та масштабності. Постійне зростання обсягів робіт, необхідність гнучкої взаємодії між учасниками команд, а також високі вимоги до контролю термінів і якості виконання проєктів обумовлюють потребу у впровадженні спеціалізованих інформаційних систем. Створення ефективної веб-платформи для управління проєктами є нагальною потребою для компаній, які працюють у динамічному ринковому середовищі та прагнуть підвищити продуктивність і прозорість власної діяльності.

Метою цього кваліфікаційного магістерського проєкту є розроблення веб-платформи для управління проєктами та аналізу ефективності їх виконання з використанням сучасних технологій веб-програмування та інструментів аналітики. У процесі роботи буде здійснено аналіз предметної області, проведено огляд існуючих рішень на ринку, охарактеризовано вимоги до функціонування системи та побудовано інформаційну модель. Також буде реалізовано компоненти системи, який дозволить візуалізувати ключові показники виконання проєктів та надати користувачам інструменти для аналітичного супроводу проєктної діяльності.

Під час розробки будуть побудовані моделі та діаграми в яких буде детально описано структуру системи. Такий підхід дозволить побудувати цілісну інформаційну модель та забезпечити ефективний обмін даними між структурними компонентами системи.

Математичне забезпечення системи є невід'ємною складовою, оскільки аналіз ефективності виконання проєктів потребує розрахунку ключових показників - таких як завантаженість ресурсів, тривалість виконання завдань, відхилення від плану, тощо. Більшість обчислень буде виконуватись на рівні бази даних за допомогою SQL-запитів. Такий підхід буде оптимальним через те, що формули не мають складних функцій, та через це буде забезпечена максимальна швидкодія системи.

Для досягнення високої якості проєктного рішення буде послідовно розроблено інформаційне, програмне та технічне забезпечення системи. Функціональність платформи охоплюватиме управління завданнями, планування проєктів, відстеження виконання та побудову аналітичних звітів з використанням KPI-метрик, діаграм виконання та інших інструментів візуалізації даних.

Результатом кваліфікаційної магістерської роботи стане проєкт веб-платформи та розроблені окремі її компоненти, що опишуть систему, яка дозволить користувачам зручно управляти проєктами, контролювати строки виконання завдань, оцінювати ефективність ресурсів та своєчасно ухвалювати управлінські рішення. Для керівництва компанії така система стане засобом централізованого моніторингу стану проєктного портфеля та інструментом підтримки прийняття рішень.

Актуальність цієї теми у наш час додатково посилюється через:

1. Широке впровадження дистанційної моделі роботи та необхідність контролю над проєктами в умовах віддаленої взаємодії;
2. Підвищені ризики зриву строків виконання через нестабільну ситуацію;
3. Потребу у гнучкому ресурсному плануванні та швидкому реагуванні на зміни;

що вимагає створення зручної, масштабованої та надійної системи управління проєктами, адаптованої до сучасних викликів.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПІДХОДІВ ДО СТВОРЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ УПРАВЛІННЯ ПРОЄКТАМИ

1.1 Аналіз існуючих інформаційних систем управління проєктами

1.1.1 Дослідження предметної області

Предметна область управління проєктами охоплює сукупність методів, інструментів, процесів і засобів, які забезпечують ефективне планування, організацію, виконання, контроль та завершення проєктної діяльності в межах встановлених термінів, бюджету та з очікуваним рівнем якості. В умовах цифрової трансформації підприємств і глобалізації бізнесу зростає потреба в гнучких та інтегрованих інформаційних системах, що дозволяють автоматизувати основні етапи життєвого циклу проєкту.

Проєкт, згідно з міжнародним стандартом PMBOK (Project Management Body of Knowledge), визначається як тимчасове підприємство, спрямоване на створення унікального продукту, послуги або результату. Проєктна діяльність характеризується обмеженнями в часі, ресурсах та масштабах, що вимагає чіткого планування, визначення пріоритетів, контролю ризиків та ефективного комунікаційного супроводу.

Сучасне управління проєктами базується на сукупності підходів, були виділені три найпоширеніші з них:

1. Класичний (каскадний) підхід (Waterfall) - передбачає послідовне проходження етапів від ініціації до завершення;
2. Agile - гнучкий підхід, орієнтований на ітеративність, адаптивність та постійне вдосконалення продукту за участі користувачів;
3. Scrum, Kanban - фреймворки Agile-методології, спрямовані на організацію командної роботи в коротких спринтах, візуалізацію процесів, мінімізацію затримок та контроль якості.

Інформаційні системи, які підтримують проєктну діяльність, повинні реалізовувати основні процеси управління: управління завданнями та розкладом, координацію дій учасників команди, розподіл ресурсів, управління бюджетом, моніторинг ризиків, звітність та візуалізацію результатів.

Предметна область також охоплює аспекти вимірювання ефективності проєктів, де застосовуються як кількісні, так і якісні метрики. Серед поширених показників - ключові показники ефективності (*KPI*), управління вартістю за отриманою цінністю (*EVM*), діаграма згасання, швидкість команди, циклічний та повний час виконання завдання. Застосування таких інструментів дозволяє забезпечити прозорість процесів, вчасно виявляти відхилення та ухвалювати обґрунтовані управлінські рішення.

У зв'язку з розширенням масштабів діяльності компаній та поширенням віддаленої роботи, особливо в умовах воєнного стану, особливу актуальність набувають веб-орієнтовані платформи, які забезпечують:

- доступність з будь-якого пристрою;
- синхронізацію даних у реальному часі;
- централізоване зберігання інформації;
- безпечний обмін проєктною документацією;
- автоматизоване збирання аналітичних даних.

Таким чином, предметна область управління проєктами є багатовимірною, динамічною та технологічно орієнтованою. Вона вимагає інтеграції знань у галузі менеджменту, інформаційних технологій, UX-дизайну, а також аналізу даних, що і визначає міждисциплінарний характер кваліфікаційного дослідження.

1.1.2 Дослідження ринку інформаційних систем управління проєктами

Сучасний ринок інформаційних систем управління проєктами (ІСУП) представлений великою кількістю рішень, що надають користувачам інструменти для ефективного планування, координації, контролю та аналізу проєктної діяльності. В умовах цифровізації бізнесу та переходу до гнучких методологій

управління, попит на такі системи зростає щороку. Найбільшим попитом користуються веб-орієнтовані платформи, які забезпечують доступність з різних пристроїв, можливість віддаленої співпраці та інтеграцію з іншими корпоративними сервісами.

Для аналізу існуючих ІУСП можна розглянути такі приклади: *Jira Software*, *Trello*, *Asana*, *ClickUp*, *Zoho Project*, *Microsoft Project*, *Worksection*, *Basecamp*. Кожна з них має свою специфіку, функціональні переваги та орієнтована на різні моделі управління й розміри команд.

1. *Jira Software* - платформа компанії Atlassian, орієнтована насамперед на IT-команди. Забезпечує повну підтримку Scrum та Kanban, гнучке налаштування робочих процесів, потужну аналітику та інтеграцію з іншими інструментами розробки.
2. *Trello* - візуальна Kanban-система для управління завданнями. Virізняється простотою використання, що робить її зручною для невеликих команд і особистого планування. Обмежена у функціональності порівняно з професійними системами.
3. *Asana* - платформа для організації командної роботи, планування задач, відстеження прогресу та побудови звітів. Підтримує шаблони проєктів, календарі, діаграми Ганта, автоматизацію повторюваних процесів.
4. *ClickUp* - багатофункціональна система з високим рівнем кастомізації, підтримкою завдань, списків, цілей, часових ліній, аналітики, інтеграцій з більшістю популярних сервісів.
5. *Zoho Project* - рішення від компанії Zoho, орієнтоване на середній та великий бізнес. Має розширені функції для побудови графіків, управління ресурсами, діаграми Ганта, бюджетування та звітності.
6. *Microsoft Project* - один із найстаріших та найпотужніших інструментів для управління проєктами. Дозволяє будувати складні графіки, розраховувати критичні шляхи, працювати з ресурсами, хоча має складний інтерфейс і потребує досвіду використання.

7. Worksection - українська хмарна система управління проєктами та задачами. Має інтуїтивний інтерфейс, підтримку клієнтських кабінетів, планувальники, діаграми Ганта, звітність і інтеграцію з поштою.
8. Basecamp - платформа, орієнтована на спрощене управління невеликими командами. Забезпечує базову підтримку задач, чатів, обміну файлами, календарів, проте має обмежений інструментарій для побудови аналітики.
- В таблиці 1.1 наведено аналіз існуючих ІС та порівняння їх функціоналу.

Таблиця 1.1.

Аналіз існуючих ІС (джерело автора)

Назва платформи	Scrum / Kanban	KPI / Аналітика	Gantt / Графіки	Спільна робота	Безкоштовна версія
Jira Software	Так	Так	Частково	Так	Обмежено
Trello	Частково	Ні	Ні	Так	Так
Asana	Частково	Частково	Так	Так	Так (до 15 осіб)
ClickUp	Так	Так	Так	Так	Так
Zoho Project	Так	Так	Так	Так	Так (обмежено)
Microsoft Project	Частково	Так	Так	Частково	Ні
Worksection	Так	Частково	Так	Так	Так (до 5 осіб)
Basecamp	Ні	Ні	Ні	Так	Так (обмежено)

Враховуючи вищезгадані аспекти та проведений аналіз дозволяє зробити такі висновки:

Ринок інформаційних систем управління проєктами представлений широким спектром програмних рішень, що відрізняються функціональністю, гнучкістю налаштувань, наявністю аналітичних інструментів і рівнем інтеграції з іншими сервісами. Попри це, жодна з досліджених систем не забезпечує повного покриття

потреб користувачів щодо одночасного управління проєктами, візуалізації показників ефективності та адаптації під специфіку конкретної організації.

Отже, попри широкий вибір ІСУП на ринку, актуальним залишається створення адаптивної веб-платформи, яка поєднує зручні засоби керування проєктами з аналітичними інструментами КРІ, графіками прогресу, інтерактивною візуалізацією та можливістю масштабування відповідно до потреб конкретного підприємства.

1.2 Обґрунтування вибору підходів і технологій для створення інформаційної системи управління проєктами

Розроблення ефективної інформаційної системи управління проєктами вимагає не лише врахування специфіки предметної області, але й обґрунтованого вибору методології управління, архітектурного підходу та технологічного інструментарію. Правильне поєднання зазначених компонентів дозволяє забезпечити гнучкість, масштабованість, продуктивність та зручність користування системою.

Застосування гнучких методологій управління, таких як Agile та Scrum, дозволяє реалізовувати систему у форматі поступового вдосконалення через короткі ітерації (спринти), зосереджуючи увагу на пріоритетних функціональних блоках і забезпечуючи постійний зворотний зв'язок із користувачами. Це сприяє створенню системи, яка максимально враховує реальні потреби цільової аудиторії, дозволяє швидко реагувати на зміни вимог і оптимізувати процес розробки на кожному етапі.

З боку архітектурного підходу доцільним є використання клієнт-серверної моделі з RESTful API, що забезпечує розподіл відповідальностей між інтерфейсом користувача, логікою обробки даних і рівнем зберігання інформації. Така багаторівнева архітектура дозволяє не лише підвищити продуктивність системи, а й забезпечити її гнучке масштабування - як вертикальне (збільшення ресурсів

окремого модуля), так і горизонтальне (розділення компонентів між серверами або контейнерами).

Ключову роль у досягненні цілей системи відіграє вибір відповідного технологічного стеку. У розробці frontend частини системи рекомендовано використання React - сучасної JavaScript-бібліотеки, що дозволяє створювати динамічні, масштабовані та інтуїтивно зрозумілі інтерфейси користувача. Для реалізації backend логіки оптимальним є Node.js у поєднанні з Express.js, що забезпечують високу продуктивність, низьку затримку обробки запитів і просту інтеграцію з базою даних. Як СУБД обрано Microsoft SQL Server, що поєднує високий рівень безпеки, підтримку складних аналітичних запитів і широкі можливості адміністрування.

Правильне поєднання методологічних, архітектурних і технологічних рішень дозволяє побудувати систему, яка буде не лише стабільною з технічної точки зору, але й зручною для кінцевого користувача, адаптивною до змін у бізнес-середовищі та ефективною з погляду використання ресурсів. Такий підхід дає змогу створити інструмент, який підвищує прозорість проєктної діяльності, покращує комунікацію в командах, мінімізує управлінські ризики та сприяє досягненню поставлених цілей у визначені строки.

Зважаючи на тенденції сучасного управління проєктами, доцільно орієнтуватися на гнучкі методології (Agile), які передбачають ітеративну розробку, швидку адаптацію до змін, активну участь замовника та самоуправлінські команди. Зокрема, Scrum як найпоширеніший фреймворк Agile забезпечує ефективну реалізацію коротких спринтів, прозору оцінку прогресу завдяки таким інструментам, як Burndown Chart, Velocity та щоденні стендапи. Такий підхід є доцільним при створенні системи, яка має поступово розширювати функціональність відповідно до відгуків користувачів.

З архітектурної точки зору доцільним є впровадження клієнт-серверної моделі з RESTful API, що забезпечує розділення логіки обробки даних (бекенд) та взаємодії з користувачем (фронтенд), спрощує супровід та масштабування системи. Така архітектура дозволяє реалізувати веб-платформу як односторінковий

застосунок (SPA), де основна логіка інтерфейсу реалізується на клієнтській стороні, тоді як сервер обробляє запити до бази даних та аналітичних сервісів.

У межах технологічного стеку було обрано такі інструменти:

- Мова програмування (бекенд): *JavaScript (Node.js)* - забезпечує високопродуктивну обробку запитів у режимі реального часу, має велику екосистему модулів і підтримує роботу з REST API.
- Мова програмування (фронтенд): *JavaScript з використанням React* - сучасна бібліотека для побудови динамічних інтерфейсів користувача з компонентною архітектурою.
- Система керування базами даних: *SQL Server* - реляційна СУБД з високою продуктивністю, підтримкою транзакцій, індексуванням і можливістю обробки складних запитів.
- Інструменти аналітики та візуалізації: *Chart.js, Recharts або Apache ECharts* - бібліотеки для побудови графіків (KPI-метрики, Burndown Charts, Velocity-графіки).
- Засоби забезпечення безпеки: *JWT (JSON Web Token)* для автентифікації, HTTPS-з'єднання, контроль прав доступу на рівні API.

Для збереження гнучкості системи рекомендовано впровадити *модульну структуру* з виділенням окремих функціональних блоків:

- управління користувачами;
- управління проєктами;
- аналітичний модуль;
- модуль звітності;
- адміністративна панель.

Такий підхід до проєктування інформаційної системи базується на модульній архітектурі, що передбачає логічне розділення функціональності на окремі незалежні компоненти (модулі), кожен з яких виконує конкретне завдання - наприклад, автентифікація користувача, керування завданнями, побудова аналітики, генерація звітів тощо.

Відкритість до масштабування у цьому контексті означає, що до системи можна безперешкодно додавати нові модулі або розширювати наявні без необхідності внесення суттєвих змін до загальної структури або логіки роботи платформи.

Можливість розширення функціональності досягається завдяки впровадженню стандартизованих інтерфейсів взаємодії між модулями (REST API), що гарантує взаємозамінність і автономність кожного компонента.

Покращення супроводу та повторного використання коду забезпечується завдяки принципам компонентного підходу (особливо на стороні клієнта з використанням React), коли окремі елементи інтерфейсу або логіки можна багаторазово застосовувати в різних частинах системи, знижуючи витрати на розробку, тестування та підтримку.

Таким чином, інтегроване застосування гнучкої методології Scrum, що забезпечує адаптивне планування та швидке реагування на змінні вимоги, клієнт-серверної архітектури з використанням REST API для забезпечення ефективної взаємодії між фронтендом і бекендом, сучасного інтерфейсного фреймворку React, який дозволяє створювати динамічний, зручний для користувача інтерфейс, надійної системи керування базами даних SQL Server із підтримкою складної логіки запитів, а також аналітичних інструментів візуалізації, утворює технологічне підґрунтя для розроблення адаптивної, масштабованої та функціонально насиченої веб-платформи управління проєктами. Така система не лише автоматизує процеси планування та контролю, а й забезпечує аналітичне оцінювання ефективності реалізації проєктів, що сприяє підвищенню якості управлінських рішень.

РОЗДІЛ 2 ХАРАКТЕРИСТИКА ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ ТА МЕТОДИ І МОДЕЛІ

2.1 Структура і характеристика системи

Інформаційна система управління проєктами, яка є об'єктом цього дослідження, розробляється як засіб комплексної автоматизації процесів планування, моніторингу, координації, оцінювання та аналізу ефективності реалізації проєктів різного масштабу. Вона покликана забезпечити централізовану підтримку всіх етапів життєвого циклу проєкту - від формування завдань і визначення виконавців до контролю строків, розподілу ресурсів, побудови звітності та оцінювання результатів. Система має охоплювати не лише адміністративно-організаційні аспекти управління, але й надавати інструменти для аналітичного супроводу прийняття рішень, зокрема через обробку динамічних даних про виконання завдань, розрахунок ключових показників ефективності (KPI) та візуалізацію прогресу за допомогою інтерактивних діаграм.

Структура системи проєктується з урахуванням сучасних підходів до побудови веб-застосунків і базується на принципах модульності, масштабованості, відкритості до інтеграції та орієнтації на кінцевого користувача. Це дозволяє системі легко адаптуватися до специфіки внутрішніх бізнес-процесів організації, а також розширювати функціональність без порушення цілісності логіки. Такий підхід також створює передумови для зручного супроводу, оновлення та масштабування системи в умовах змін середовища або зростання обсягу оброблюваних даних. У результаті платформа має забезпечити надійну, гнучку та ефективну підтримку процесів управління проєктами як у малих командах, так і в середовищах корпоративного рівня.

Загальна структура системи

Система має трирівневу архітектуру, що включає такі основні компоненти:

1. Клієнтський рівень (Frontend) - відповідає за відображення інтерфейсу користувача, взаємодію із сервісами системи та візуалізацію аналітичних

показників. Реалізується за допомогою JavaScript-бібліотеки React, що забезпечує швидкодію, компонентність і високу інтерактивність.

2. Серверний рівень (Backend) - обробляє запити, реалізує бізнес-логіку, формує відповіді API та забезпечує зв'язок із базою даних. Реалізується за допомогою платформи Node.js.
3. Рівень зберігання даних (Database) - охоплює систему керування базами даних SQL Server, яка відповідає за збереження, індексацію, обробку та забезпечення цілісності структурованих даних.

Взаємодія між рівнями здійснюється за допомогою REST API, що гарантує стандартизований обмін даними, гнучкість інтеграції з зовнішніми системами та можливість використання різноманітних клієнтських застосунків.

Функціональна структура системи

Інформаційна система управління проєктами складається з таких функціональних підсистем:

- Підсистема автентифікації та авторизації - забезпечує реєстрацію, вхід до системи, розмежування прав доступу на основі ролей (адміністратор, менеджер, виконавець).
- Підсистема управління проєктами - реалізує створення, редагування та супровід проєктів, встановлення цілей, дедлайнів, призначення відповідальних осіб.
- Підсистема управління завданнями - дозволяє формувати структуру завдань, встановлювати залежності, відстежувати прогрес виконання, переглядати статуси.
- Підсистема аналітики - формує показники ефективності (KPI), генерує діаграми (Burndown Chart, Velocity), дозволяє виявляти відхилення від плану.
- Підсистема звітності - забезпечує створення звітів у форматах PDF, Excel, формує підсумкову інформацію за окремими проєктами або періодами.
- Адміністративна панель - надає інструменти для управління користувачами, налаштування параметрів системи, перегляду журналу подій.

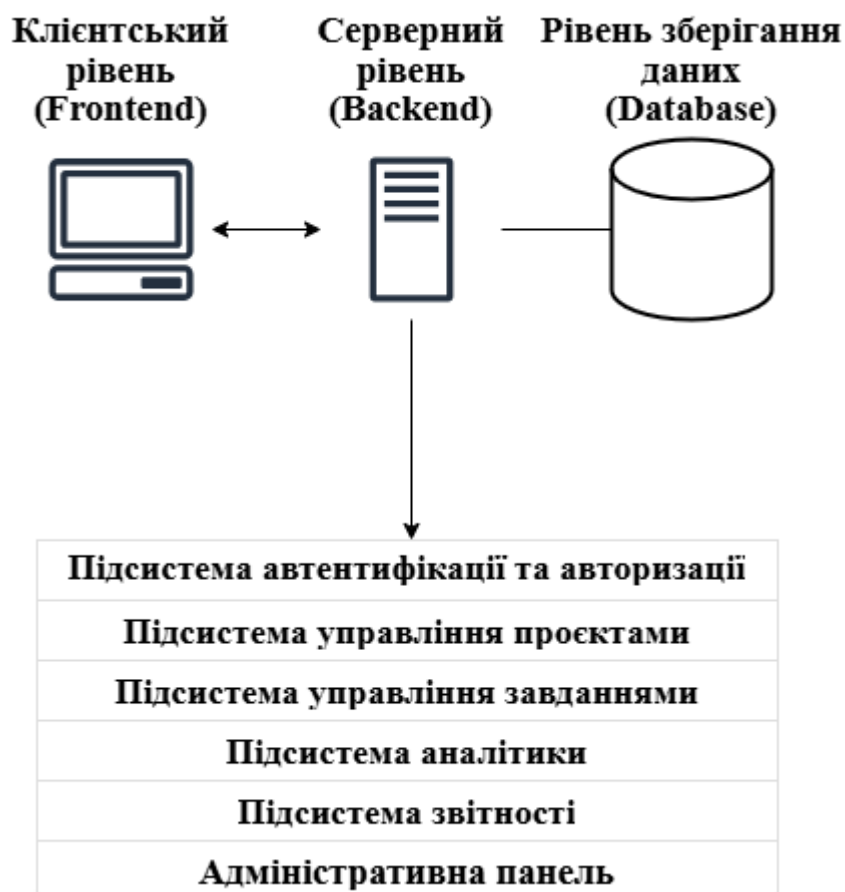


Рис. 2.1. Діаграма функціонально-рівневої архітектури (джерело автора)

Технічні характеристики

- Мова програмування (бекенд): JavaScript (Node.js)
- Мова програмування (фронтенд): JavaScript (React)
- СУБД: SQL Server
- Архітектура: клієнт-серверна, RESTful API
- Методи аутентифікації: JWT (JSON Web Token)
- Методи обміну даними: HTTPS, JSON
- Інструменти візуалізації: Chart.js / Recharts / Apache ECharts
- Інтерфейс користувача: адаптивний, компонентний, побудований за принципами UX/UI-дизайну

Таким чином, побудова системи на основі модульного та багаторівневого підходу дозволяє забезпечити високу гнучкість, надійність, масштабованість та зручність користування. Обрана архітектура підтримує подальший розвиток

системи, її адаптацію до нових функціональних вимог та інтеграцію з зовнішніми інформаційними ресурсами.

2.2 Методи та моделі в інформаційній системі управління проєктами

Розроблення сучасної інформаційної системи управління проєктами є складним інженерно-аналітичним процесом, який вимагає системного підходу до організації внутрішньої структури, обробки даних та побудови логіки взаємодії користувача із системою. У зв'язку з цим ключову роль відіграють формалізовані методи та моделі, які дозволяють не лише структурувати функціональні блоки системи, а й забезпечити її адаптивність, масштабованість та надійність у реальних умовах експлуатації. Ці методи охоплюють концептуальне, інформаційне, функціональне та логічне моделювання, а також аналітичну обробку результатів проєктної діяльності з метою подальшого вдосконалення управлінських рішень.

Застосування моделей бізнес-процесів, діаграм варіантів використання, потоків даних, а також моделей предметної області дозволяє на етапі проєктування чітко визначити межі системи, її основні підсистеми, типи користувачів, інформаційні потоки, точки прийняття рішень та критерії ефективності. Крім того, використання аналітичних методів, орієнтованих на обробку ключових показників ефективності (KPI), сприяє забезпеченню прозорості процесів управління, виявленню критичних місць та прийняттю своєчасних рішень для оптимізації ресурсів.

Вибір конкретного методу чи моделі залежить від багатьох факторів: складності організаційної структури проєкту, кількості учасників, рівня взаємозалежності між задачами, вимог до звітності та інтеграції, а також цілей, що поставлені перед системою. Наприклад, для невеликих проєктів доцільним є використання спрощених моделей управління та візуалізації, тоді як для багаторівневих програмних проєктів необхідні розгалужені інформаційні структури та методи багатовимірного аналізу.

Таким чином, методологічна та модельна база є фундаментом, що забезпечує наукову обґрунтованість і практичну ефективність функціонування інформаційної системи управління проєктами в динамічному цифровому середовищі.

2.2.1 Методологічна основа

У процесі розроблення інформаційної системи управління проєктами ключовим фактором є вибір методології управління, яка визначає логіку організації робочих процесів, взаємодії між учасниками проєкту, формування цілей і контролю за їх досягненням. У межах цього дослідження застосовано принципи гнучких методологій (Agile, Scrum), що довели свою ефективність у розробленні цифрових продуктів в умовах змінних вимог, високої невизначеності та обмеженості ресурсів.

Agile (гнучкий підхід) базується на філософії ітеративного розвитку, яка передбачає розбиття проєкту на невеликі цикли з постійним зворотним зв'язком від користувачів і постійною адаптацією до змін. Основними цінностями є індивідуальна відповідальність, співпраця з замовником, швидке реагування на зміну пріоритетів і готовність до модифікації вимог у будь-який момент реалізації проєкту.

Scrum - це структурований фреймворк реалізації Agile, який впроваджує чіткий розподіл ролей (Product Owner, Scrum Master, команда розробки), ритуали (щоденні стендапи, планування, ретроспективи, огляди спринтів) та артефакти (список завдань, спринт-беклог, інкремент продукту). Кожен проєкт розбивається на серію коротких повторюваних циклів - спринтів, зазвичай тривалістю від одного до чотирьох тижнів. У межах кожного спринту команда реалізує певний обсяг функціоналу, що дозволяє отримати частково готовий продукт і оперативно перевірити його відповідність очікуванням користувача.

Застосування Scrum-фреймворку у контексті розроблення інформаційної системи забезпечує високий рівень прозорості, оперативну ітеративну перевірку функціональності, зниження ризиків, пов'язаних із невідповідністю результатів очікуванням замовника, а також гнучкість у зміні пріоритетів відповідно до

динаміки зовнішнього середовища. Це дозволяє ефективно інтегрувати аналітичні модулі, моделі оцінювання та інтерфейсні компоненти в процес розроблення, тестування й розгортання системи.

2.2.2 Інформаційне моделювання

Інформаційне моделювання є критично важливою складовою етапу проектування інформаційної системи управління проєктами, оскільки дозволяє формалізувати логіку роботи системи, структуру даних, взаємозв'язки між компонентами та механізми взаємодії користувача із системою. Метою інформаційного моделювання є створення цілісного уявлення про архітектуру та функціонування системи до моменту її реалізації, що дає змогу виявити потенційні проблеми, оптимізувати процеси та забезпечити відповідність розробки вимогам замовника.

У межах цього дослідження використовуються такі типи інформаційних моделей:

1. Use Case Diagram (Діаграми варіантів використання)

Цей тип моделі дозволяє описати взаємодію між користувачами та системою через виконання конкретних дій - сценаріїв використання. У діаграмі виокремлюються ролі користувачів. Я виділив наступні ролі, як менеджер, виконавець, та пов'язав з відповідними функціями: створення проєктів, керування проєктами, призначення завдань, корегування завдань, оновлення статусів, додавання коментарів до виконання, подання звітів, перегляд аналітики .

Use Case Diagram слугує основою для специфікації функціональних вимог до системи, виявлення критичних точок взаємодії та перевірки повноти покриття сценаріїв.

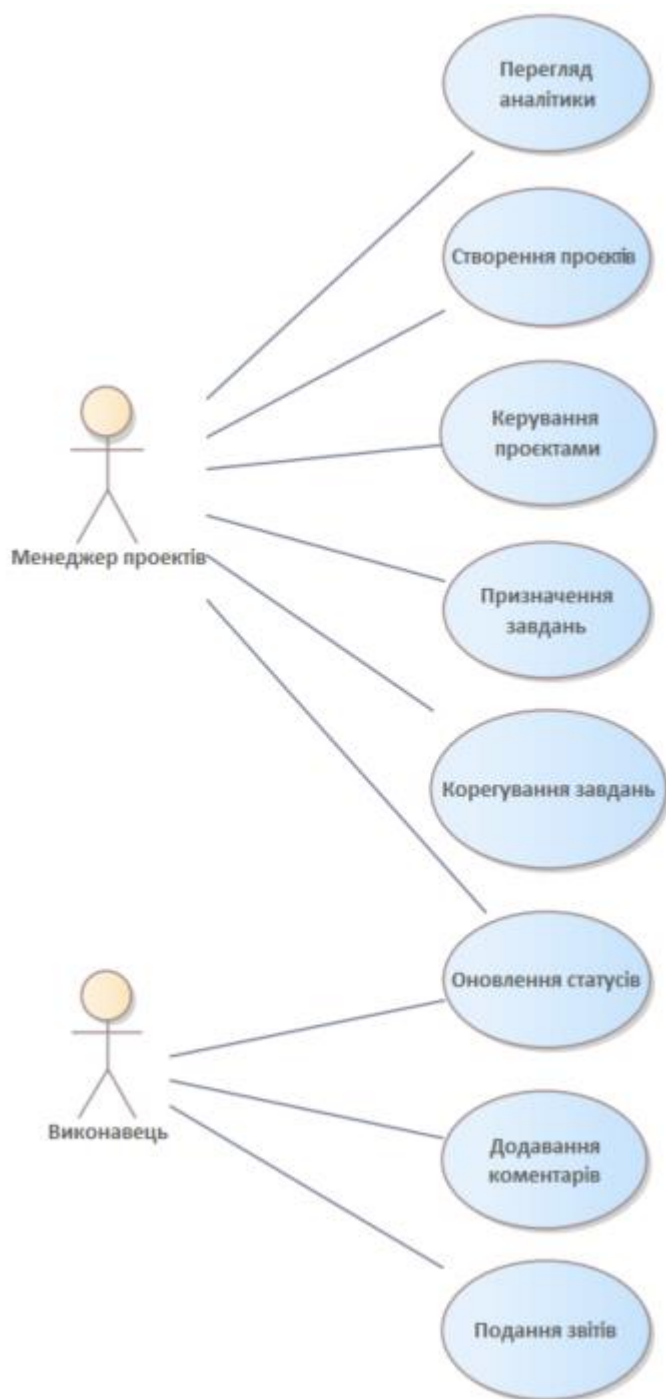


Рис. 2.2. Діаграма прецедентів (джерело автора)

2. BPMN (Business Process Model and Notation)

BPMN-діаграми дозволяють моделювати бізнес-процеси, які реалізуються або підтримуються інформаційною системою. У контексті ІУСП це процес створення нового проєкту, погодження завдань, затвердження звітів, моніторинг виконання або генерація дашбордів.

Цей тип моделі забезпечує наочне уявлення про послідовність дій, роль кожного учасника у процесі, правила переходу між етапами та точки прийняття рішень. Застосування BPMN дозволяє виявити неефективні ділянки бізнес-процесів, оптимізувати їх та узгодити логіку ІС із реальною діяльністю організації.

Було побудовано чотири типи діаграм BPMN: взаємодії, процесів, діалогів та хореографічну діаграму. Перші два типи побудовані в Bizagi, діаграма діалогів – в онлайн-інструменті Drawio, хореографічна діаграма в онлайн-інструменті Drawio. (Великі діаграми зручніше розглядати при збільшенні масштабу в MS Word)

1. Діаграма взаємодії.

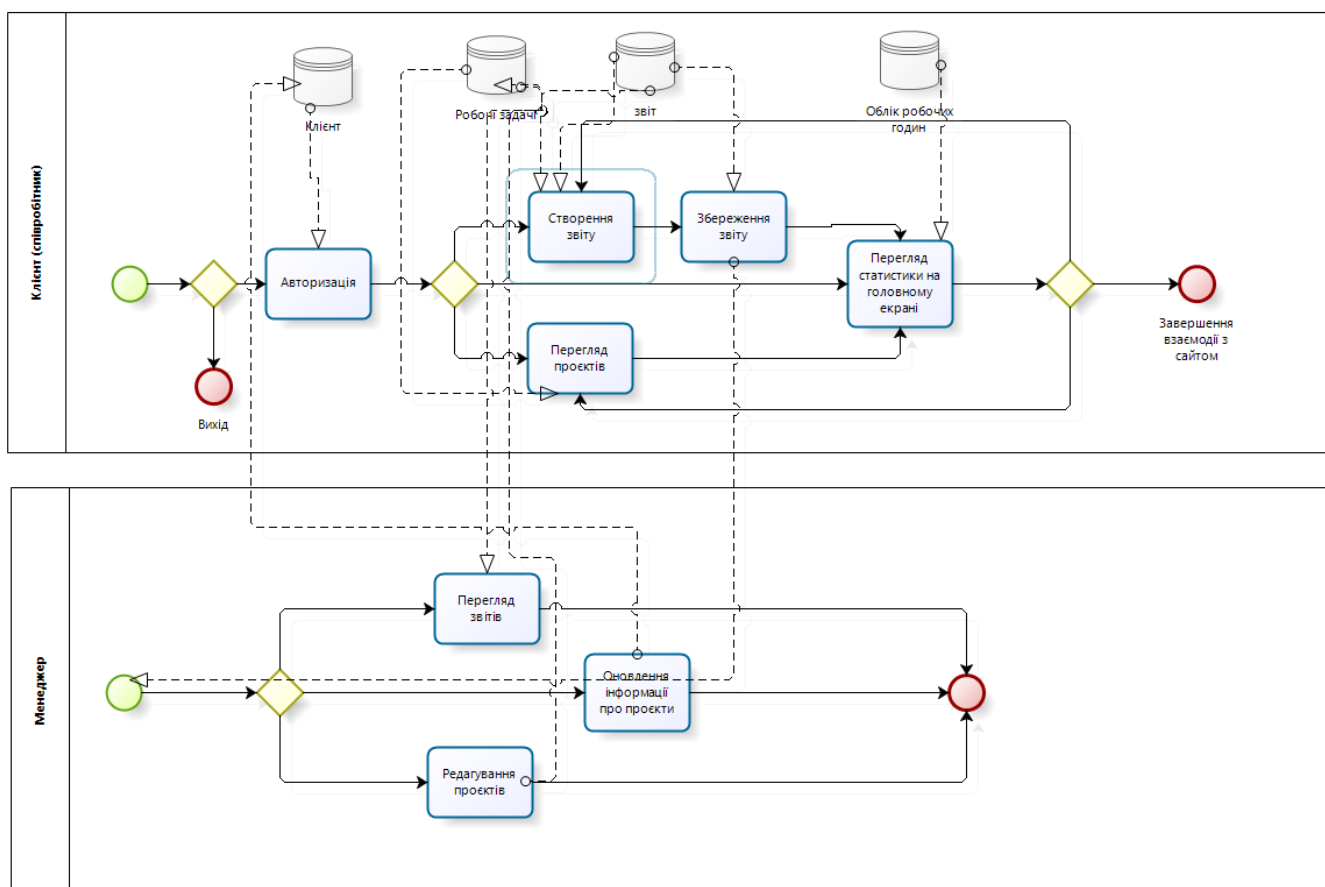


Рис. 2.3. Діаграма взаємодії (джерело автора)

2. Діаграми процесів.

2.1. Авторизація

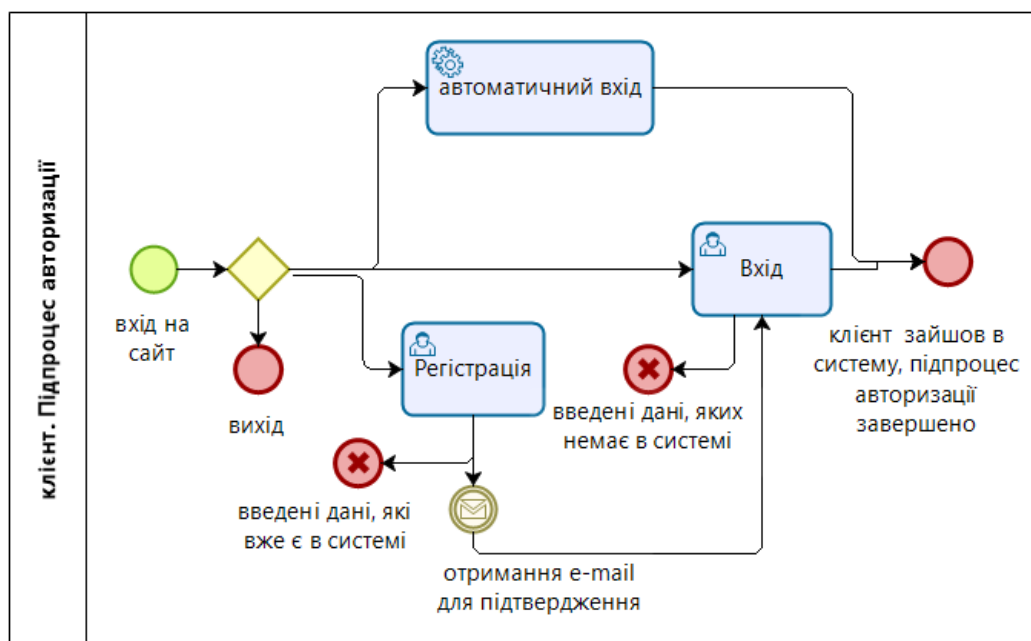


Рис. 2.4. Діаграма процесу авторизації (джерело автора)

Людина може вже при вході на сайт автоматично авторизуватись системою. Якщо автоматичного входу немає, то клієнт повинен увійти або зареєструватися. Враховані також помилки, якщо клієнт спробує увійти за неіснуючими даними в системі або зареєструватися за даними, що є в базі.

2.2. Заповнення звіту

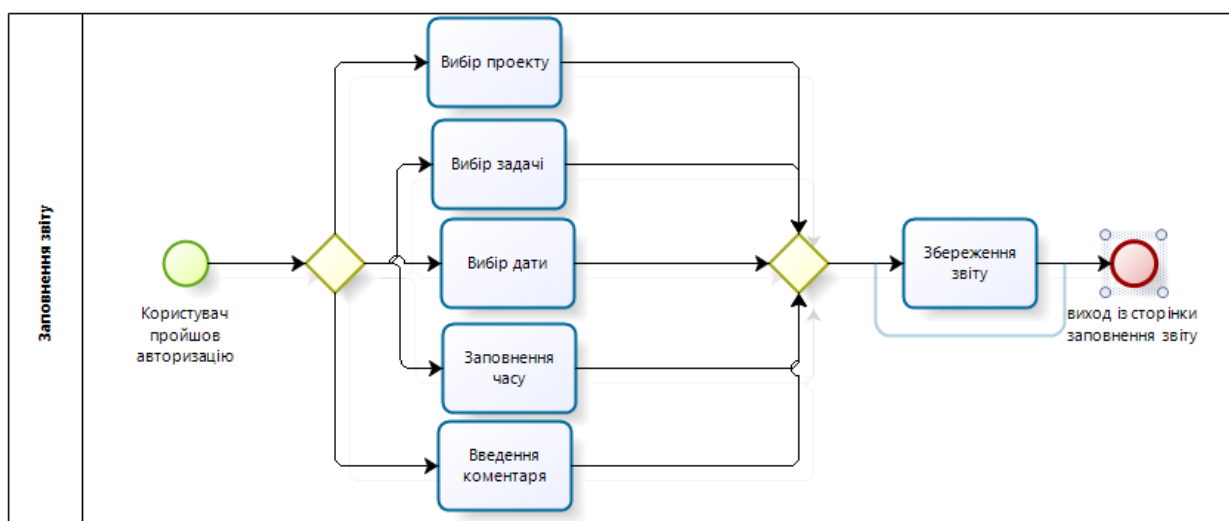


Рис. 2.5. Діаграма процесу заповнення звіту (джерело автора)

Користувач, перейшовши на сторінку заповнення звіту, може одночасно за бажанням заповнювати будь-яке з полів.

3. Діаграма діалогів (Lucidchart)

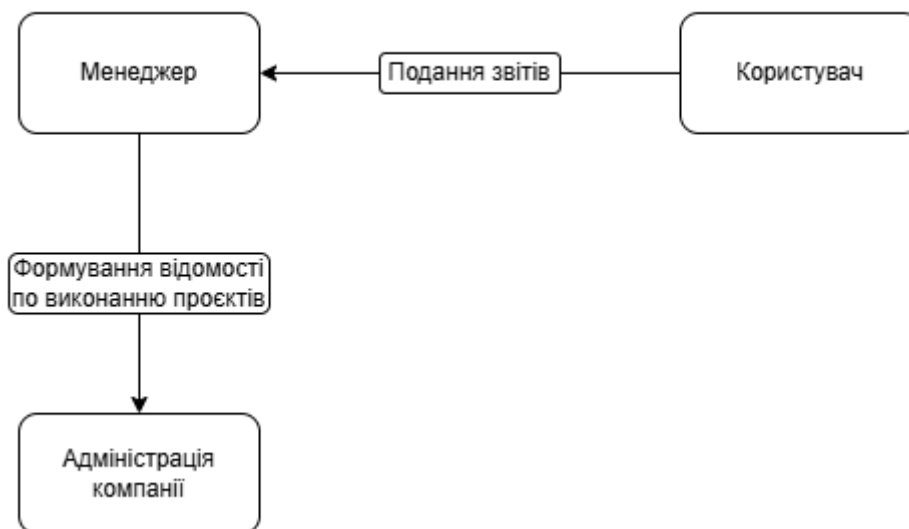


Рис. 2.6. Діаграма діалогів (джерело автора)

Менеджер з користувачем поєднані питанням надсилання звіту. А з адміністрацією – передача інформації про відомість з виконання проєктів.

4. Хореографічна діаграма (Drawio)



Рис. 2.7. Хореографічна діаграма (джерело автора)

Дана діаграма дає нам наступну інформацію: спочатку клієнт надсилає інформацію про результати роботи з задачею. Далі менеджер отримав відповідний звіт. Наступним кроком менеджер формує необхідну інформацію до адміністрації компанії. А адміністрація приймає звіт менеджера.

3. DFD (Data Flow Diagram)

DFD-діаграми використовуються для моделювання потоків інформації між підсистемами, модулями та зовнішніми елементами. Цей тип моделі демонструє, які дані надходять у систему, як вони обробляються, куди передаються і в якому вигляді зберігаються.

DFD дозволяє ієрархічно структурувати логіку обробки інформації, формалізувати зв'язки між функціональними блоками та визначити вхідні/вихідні повідомлення, що циркулюють у системі. Це є надзвичайно важливим для забезпечення цілісності інформаційної моделі та перевірки її коректності.

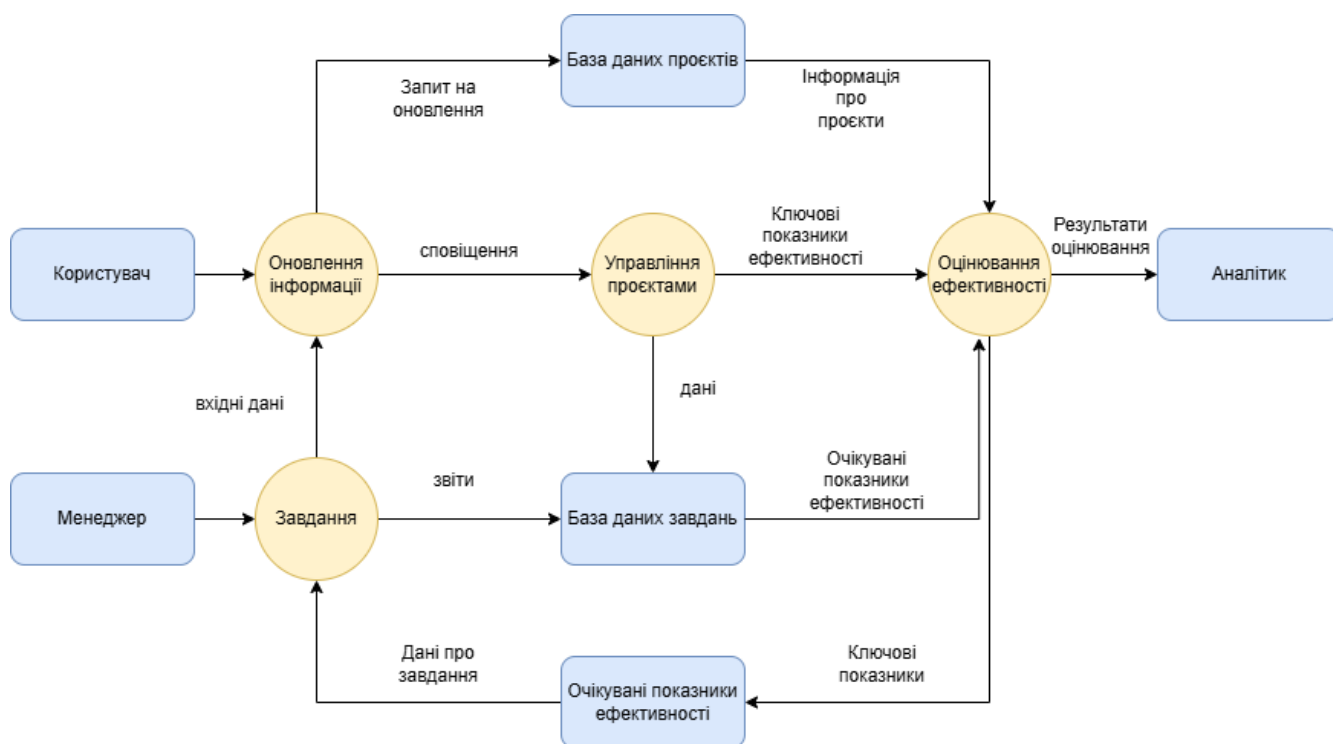


Рис. 2.7. Діаграма потоку даних (джерело автора)

На діаграмі зображено потоки даних між основними компонентами системи: зовнішніми користувачами (менеджер, співробітник, адміністратор), внутрішніми процесами (керування проєктами, завданнями, аналітикою, звітністю), сховищами

даних (користувачі, проєкти, завдання, KPI) та каналами передачі інформації. Стрілки вказують напрямки циркуляції даних, забезпечуючи наочне уявлення про логіку функціонування системи на рівні обробки інформації.

4. ER-діаграми (Entity-Relationship Diagram)

ER-моделі описують структуру даних, що зберігаються у базі даних, через визначення сутностей, їх атрибутів та типів зв'язків між ними.

Ці моделі є основою для побудови логічної та фізичної схеми бази даних, забезпечення нормалізації та ефективної організації збереження даних. Наявність чітко сформованої ER-моделі сприяє стабільності структури даних, спрощенню запитів до СУБД та гарантує узгодженість інформації в межах усієї системи.

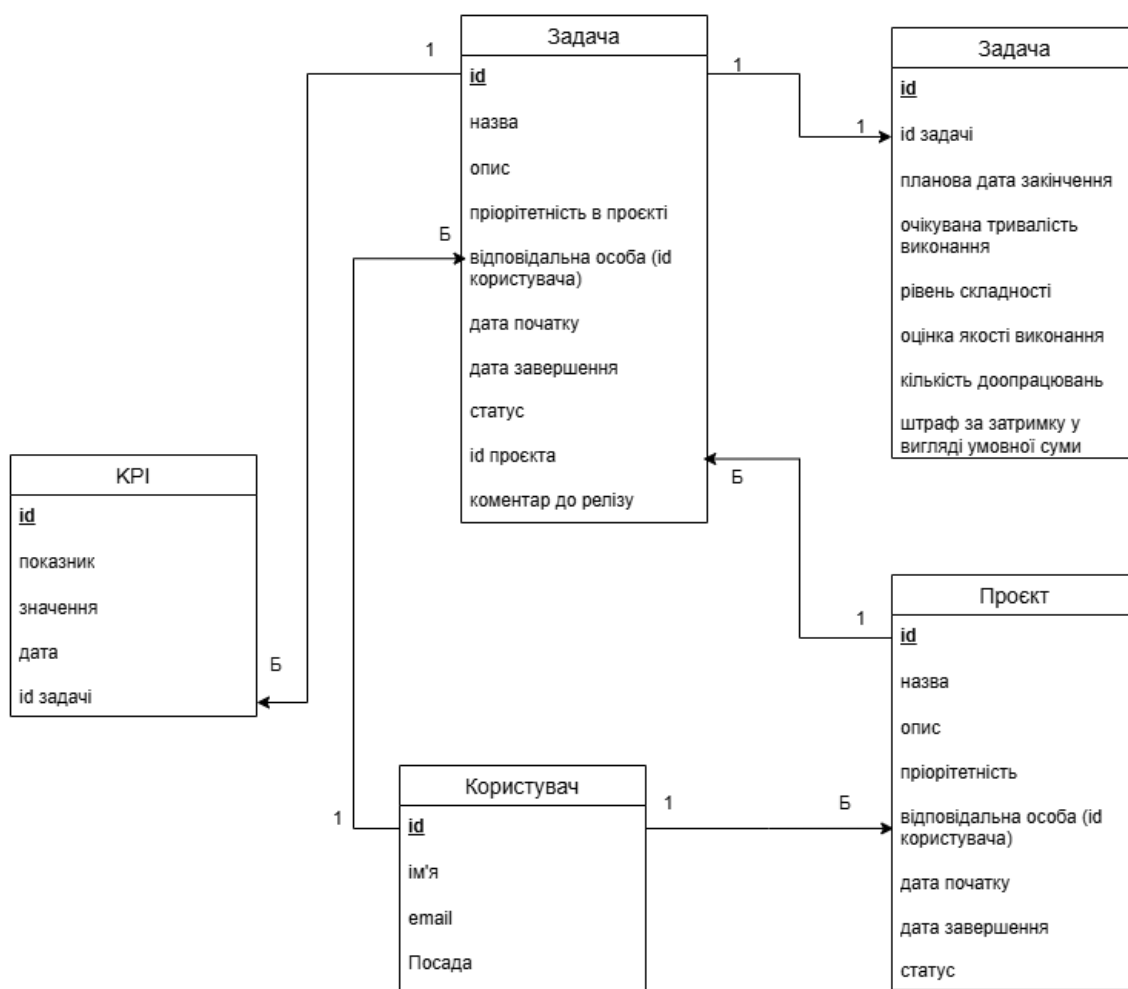


Рис. 2.8. Інфологічна модель БД (джерело автора)

У цій моделі є чотири основні сутності: «Задача», «Проект», «Користувач» та «КРІ». Кожна сутність має свої атрибути, які відповідають конкретним характеристикам даних. Зв'язки між сутностями показуються зв'язками на діаграмі. Наприклад, «Проект» має зв'язок з «Користувачем» за допомогою зовнішнього ключа «ID працівника», що зв'язує кожен запис проєкта з конкретним працівником. Аналогічно, "Проект" та «КРІ» також мають зв'язки з «Користувачем» за допомогою відповідних зовнішніх ключів.

Таким чином, у результаті проведеного інформаційного моделювання було сформовано комплексне уявлення про функціональну, логічну та інформаційну структуру системи управління проєктами. Кожен із використаних типів моделей виконує власну роль у процесі проєктування: діаграми прецедентів (Use Case) дозволили структурувати сценарії взаємодії користувачів із системою; BPMN-діаграми забезпечили формалізацію бізнес-процесів і послідовності операцій; діаграма потоків даних (DFD) відобразила маршрути циркуляції інформації між підсистемами та базами даних; а ER-діаграма стала основою для побудови логічної структури бази даних.

Застосування зазначених моделей дало змогу істотно підвищити прозорість архітектурного дизайну системи, оскільки кожен елемент її структури було чітко формалізовано та логічно взаємопов'язано. Візуалізація та аналіз моделей також сприяли виявленню потенційно критичних точок у взаємодії компонентів, де можуть виникати функціональні конфлікти або інформаційні втрати. Крім того, інформаційне моделювання забезпечило узгодженість між функціональними вимогами, логікою обробки даних і специфікою інтерфейсної частини, що є запорукою цілісності та стабільності системи.

Інформаційне моделювання стало не лише етапом технічного планування, а й інструментом аналітичної валідації проєктних рішень, що дозволяє впевнено перейти до етапу реалізації системи з чітко визначеними ролями, потоками даних та структурою бази даних.

2.3 Математичне забезпечення інформаційної системи управління проєктами

Математичне забезпечення є фундаментальним компонентом у структурі інформаційної системи управління проєктами, оскільки саме воно забезпечує кількісну оцінку ефективності реалізації задач, проєктів та діяльності окремих учасників. Завдяки формалізованим математичним моделям відбувається перехід від зберігання сирих операційних даних до їх аналітичної інтерпретації у вигляді ключових показників ефективності (Key Performance Indicators - KPI).

З метою реалізації повноцінного математичного аналізу, в структурі бази даних передбачено окрему таблицю `task_metrics`, яка містить первинні оцінні та планові атрибути, необхідні для коректного розрахунку KPI: планову дату завершення задачі (`planned_deadline`), очікувану тривалість задачі (`estimation_hours`), складність (`complexity`), кількість доопрацювань (`rework_count`), задоволеність замовника (`satisfaction`) та штраф за прострочення (`delay_penalty`). Ці атрибути формують вхідні дані для побудови математичних моделей.

До основних показників, які використовуються для оцінювання ефективності виконавців і проєктів, належать:

- `estimation_hours` (E) - очікувана тривалість задачі
- Actual (τ) - фактична тривалість виконання задачі: $\text{actual} = \text{tend} - \text{tstart}$
- Delay (δ) - відхилення від планової дати завершення: $\text{delay} = \text{tend} - \text{tplan}$
- Ontime (θ) - своєчасність виконання задачі: $\text{ontime} = 1$, якщо $\text{tend} \leq \text{tplan}$, інакше 0
- Performance (η) - продуктивність працівника: кількість завершених задач / середня тривалість виконання
- Priority (π) - частка задач з високим пріоритетом у виконаних
- Satisfaction (σ) - середній рівень оцінки задоволеності за виконані задачі
- Rework (ψ) - середня кількість доопрацювань на одну задачу
- Penalty (ϕ) - сума штрафів за затримки задач

Оцінювання ефективності діяльності відбувається шляхом агрегації вищевказаних показників у межах користувача або проєкту. Всі розрахунки здійснюються автоматизовано на рівні бази даних із використанням SQL-запитів, у тому числі збережених процедур та представлень (VIEW). Це дозволяє здійснювати динамічний аналіз даних у режимі реального часу.

Формули:

$$(2.1) \quad \tau = tend - tstart$$

Фактична тривалість виконання задачі в днях.

$$(2.2) \quad \delta = tend - tplan$$

Відхилення від планового дедлайну. Значення $\delta > 0$ вказує на прострочення.

$$(2.3) \quad \theta = \begin{cases} 1, \text{ якщо } tend \leq tplan \\ 0, \text{ якщо } tend > tplan \end{cases}$$

Індикатор своєчасності виконання задачі (бінарна змінна).

$$(2.4) \quad \eta = \frac{Ntasks}{Avg(\tau)}$$

де $Ntasks$ - кількість виконаних задач користувача, а $Avg(\tau)$ - середня тривалість задач.

Продуктивність виконавця - кількість задач, виконаних у середній тривалості.

$$(2.5) \quad \pi = \frac{N_{priority=1}}{Ntasks}$$

Частка задач з пріоритетом 1 серед усіх завершених задач виконавця.

$$(2.6) \quad \sigma = \frac{\sum_N \sigma}{N}$$

Середня оцінка задоволеності замовника за задачами.

$$(2.7) \quad \psi = \frac{\sum_N \text{rework_count}}{N}$$

Середня кількість доопрацювань задачі (редизайнів, переробок).

$$(2.8) \quad \varphi = \frac{\sum_N \text{delay_penalty}}{N}$$

Загальна сума штрафів за перевищення термінів задач (якщо $tend > tplan$).

$$(2.9) \quad \varepsilon = \tau - E$$

Відхилення фактичної тривалості виконання від очікуваної (оціненої) на етапі планування.

Враховуючи раніше розглянуті розразунки показників необхідних для оцінки продуктивності співробітників під час виконання робочих задач, можна зробити формулу KPI на основі зваженої суми нормованих показників:

$$(2.10) \quad KPI = w_1 \cdot \bar{\theta} + w_2 \cdot \left(1 - \frac{\bar{\delta}}{\delta_{\max}}\right) + w_3 \cdot \sigma + w_4 \cdot \left(1 - \frac{\bar{\psi}}{\psi_{\max}}\right) + w_5 \cdot \left(1 - \frac{\bar{\phi}}{\phi_{\max}}\right)$$

де:

- $\bar{\theta}$ — середній індикатор своєчасного виконання задач (від 0 до 1),
- $\bar{\delta}$ — середнє відхилення від плану (затримка), нормалізується через $\delta_{\max}$ — максимальне допустиме запізнення,
- σ — середній рівень задоволеності (від 0 до 1),
- $\bar{\psi}$ — середнє число доопрацювань, нормалізується через $\psi_{\max}$ — припустиму межу,
- $\bar{\phi}$ — загальна сума штрафів, нормалізується на $\phi_{\max}$ (максимальний припустимий штраф).
- w_i — вага кожного показника (вплив на загальний KPI),

Кожен $w_i \in [0;1]$ і $\sum w_i = 1$ (наприклад, всі рівні — по 0.2).

Завдяки впровадженню формалізованого математичного апарату система управління проєктами набуває не лише інструментів для об'єктивного моніторингу стану виконання задач, а й повноцінного функціоналу аналітичної підтримки управлінських рішень. Зокрема, розрахунок показників фактичної тривалості задач (τ), відхилення від плану (δ), своєчасності виконання (θ), продуктивності виконавців (η), кількості доопрацювань (ψ), задоволеності замовників (σ) та суми штрафів (ϕ) забезпечує багатовимірну оцінку ефективності роботи як окремих користувачів, так і проєктів загалом. Формула KPI, побудована як зважена функція від зазначених нормалізованих показників, дає змогу інтегрувати різні аспекти якості у єдиний індикатор продуктивності, що зручно для ранжування персоналу, проєктів або підрозділів.

Такі можливості відкривають шлях до реалізації системи прогнозування критичних ситуацій, раннього виявлення ризиків (наприклад, через порушення дедлайнів, зростання кількості доопрацювань або зниження задоволеності), а також до динамічного перерозподілу ресурсів у проєктному середовищі. Крім того, математичні моделі легко адаптуються до подальшого розширення системи — наприклад, впровадження модулів машинного навчання на базі зібраних КРІ для автоматичного прогнозування затримок, ідентифікації найефективніших виконавців, або генерації персоналізованих рекомендацій для менеджерів. У перспективі це дозволяє трансформувати систему з інструменту контролю у повноцінну платформу підтримки прийняття рішень у сфері управління проєктами.

РОЗДІЛ 3 РОЗРОБЛЕННЯ ПРОЕКТНИХ РІШЕНЬ ТА ЇХ РЕАЛІЗАЦІЯ

3.1 Проектування бази даних для інформаційної системи управління проєктами

Проектування бази даних є критично важливим і водночас фундаментальним етапом у процесі створення інформаційної системи управління проєктами. Саме база даних виконує роль центрального компонента, який забезпечує надійне зберігання, впорядкування, структуризацію та швидкий доступ до інформації, що генерується та використовується системою в процесі її функціонування. Вона виступає ядром, через яке відбувається взаємодія між різними підсистемами, а також між користувачами та прикладною логікою системи. Без правильно спроектованої бази даних було б неможливо забезпечити узгодженість між операційними процесами, аналітичними запитами, механізмами авторизації та управління доступом.

З точки зору інформаційної безпеки та консистентності, база даних повинна гарантувати цілісність даних, що досягається шляхом застосування відповідних обмежень, нормалізації структури та використання зовнішніх ключів для підтримки зв'язків між таблицями. Крім того, важливим є забезпечення узгодженості та уніфікації даних у межах усієї системи, що дозволяє уникнути дублювання, зменшити ризик логічних конфліктів і підвищити ефективність виконання запитів до СУБД.

На підставі аналізу функціональних вимог до майбутньої системи, а також на основі результатів інформаційного моделювання, наведеного у попередньому розділі, було побудовано інфологічну модель бази даних, яка відображає ключові сутності предметної області, їхні атрибути, а також типи логічних зв'язків між цими сутностями. Модель дозволяє формалізувати логіку взаємодії між користувачами, задачами, проєктами та показниками ефективності (KPI), що є критично важливим для подальшого впровадження функціональних та аналітичних компонентів системи. Саме ця модель стала основою для подальшого переходу до

фізичного проєктування бази даних, її реалізації та інтеграції з прикладною частиною платформи.

Інфологічна модель, зображена на рис. 2.8, включає чотири ключові сутності: **Користувач**, **Проект**, **Задача** та **КРІ**. Кожна сутність має унікальний ідентифікатор (*id*), набір атрибутів, які характеризують об'єкти відповідного класу, а також міжсуб'єктні зв'язки, що відображають логіку взаємодії між даними.

- **Користувач** (*id, ім'я, email, посада*) - описує учасників системи, включаючи менеджерів, виконавців та адміністративний персонал. Ця сутність має зв'язки типу «один до багатьох» із сутностями **Задача** та **Проект**, оскільки один користувач може бути відповідальним за виконання кількох задач та реалізацію кількох проєктів.
- **Проект** (*id, назва, опис, пріоритетність, відповідальна особа, дата початку, дата завершення, статус*) - визначає об'єкт управління, що об'єднує множину задач. Має зв'язок із **Користувачем**, який виступає відповідальною особою за весь проєкт, та зв'язок із **Задачами**, що належать до цього проєкту.
- **Задача** (*id, назва, опис, пріоритетність, відповідальна особа, дата початку, дата завершення, статус, id проєкту, коментар до релізу*) - є базовим елементом реалізації проєкту. Зв'язана з **Проектом**, як його структурна одиниця, та з **Користувачем**, що відповідає за виконання конкретної задачі.
- **КРІ** (*id, показник, значення, дата, id задачі*) - представляє ключові показники ефективності виконання задач. Має зв'язок типу «багато до одного» з сутністю **Задача**, що дозволяє аналізувати продуктивність на рівні окремих елементів проєкту.
- **Метрики задачі** (*task_metrics*) (*id, task_id, planned_deadline, estimation_hours, complexity, satisfaction, rework_count, delay_penalty*) - забезпечують зберігання первинних планових, оцінкових та якісних характеристик задач. Зв'язані з таблицею **Задача** через зовнішній ключ *task_id*. Дозволяють проводити глибокий аналіз ефективності виконання задач, враховуючи як планові (*deadline, estimation*), так і фактичні показники (*затримка, якість*),

складність тощо). Наявність цієї таблиці є необхідною умовою для об'єктивного обчислення КРІ, оскільки вона містить вихідні значення для математичних моделей аналізу.

Наявність логічно структурованих зв'язків між сутностями дозволяє забезпечити підтримку цілісності даних, оптимізацію запитів та ефективне виконання операцій з аналітики. Наприклад, зв'язки між КРІ та задачами дозволяють побудувати узагальнені звіти ефективності в межах кожного проєкту або за періодом часу. Аналогічно, зв'язок між задачами та користувачами забезпечує автоматизоване формування списку відповідальних осіб та сповіщення про дедлайни.

Основаючись на попередній інфологічній моделі, побудуємо даталогічну модель БД у вигляді реляційної схеми для конкретної системи керування базами даних, такої як SQL Server:

Таблиця "Користувач":

DESKTOP-2HJ35MA...ienko - Diagram_0*		DESKTOP-2HJ35MA.k...ilienko - dbo.user*	
	Имя столбца	Тип данных	Разрешить знач...
🔑	id	numeric(10, 0)	☐
	name	nchar(35)	☐
	email	nchar(30)	☐
	position	nchar(50)	☐

Рис. 3.1. Об'єкт «Користувач» (джерело автора)

Таблиця "Задача":

DESKTOP-2HJ35MA...ienko - Diagram_0*		DESKTOP-2HJ35MA.k...ilienko - dbo.user*	
	Имя столбца	Тип данных	Разрешить знач...
🔑	id	numeric(10, 0)	☐
	name	nchar(40)	☐
	description	text	☒
	priority	numeric(2, 0)	☐
	user_id	numeric(10, 0)	☐
	start_date	date	☐
	end_date	date	☐
	status	nchar(20)	☐
	project_id	numeric(10, 0)	☐
	notes	text	☒

Рис. 3.2. Об'єкт «Задача» (джерело автора)

Таблиця "Проект":

DESKTOP-2HJ35MA.k...ilenko - Diagram_0*		DESKTOP-2HJ35MA.k...ilenko - dbo.user*	
	Имя столбца	Тип данных	Разрешить значения ...
PK	id	numeric(10, 0)	☐
	name	nchar(40)	☐
	description	text	☒
	priority	numeric(2, 0)	☐
	user_id	numeric(10, 0)	☐
	start_date	date	☐
	end_date	date	☐
	status	nchar(20)	☐

Рис. 3.3. Об'єкт «Проект» (джерело автора)

Таблиця " Ключові показники ефективності ":

DESKTOP-2HJ35MA.k...ilenko - dbo.user*		DESKTOP-2HJ35MA.k...ilenko - dbo.task*	
	Имя столбца	Тип данных	Разрешить знач...
PK	id	numeric(10, 0)	☐
	indicator	nchar(15)	☐
	value	float	☐
	data	date	☐
	task_id	numeric(10, 0)	☐

Рис. 3.4. Об'єкт «Ключові показники ефективності» (джерело автора)

Таблиця " Метрики задачи ":

	Имя столбца	Тип данных	Разрешить знач...
PK	id	int	☐
	task_id	numeric(10, 0)	☐
	planned_deadline	date	☒
	estimation_hours	numeric(5, 2)	☒
	complexity	tinyint	☒
	satisfaction	tinyint	☒
	rework_count	int	☒
	delay_penalty	numeric(5, 2)	☒

Рис. 3.5. Об'єкт «Метрики задачі» (джерело автора)

У цій даталогічній моделі кожна сутність стає окремою таблицею, а атрибути стають полями в цих таблицях. Зв'язки між сутностями відображаються за допомогою зовнішніх ключів. Наприклад, поле "Користувач_ID" у таблицях "Проект", "Задача" є зовнішнім ключем, який посилається на поле "ID" у таблиці "Користувач".

Контрольні приклади з заповненими таблицями продемонстровані у Додатку А.

Даталогічна модель БД:

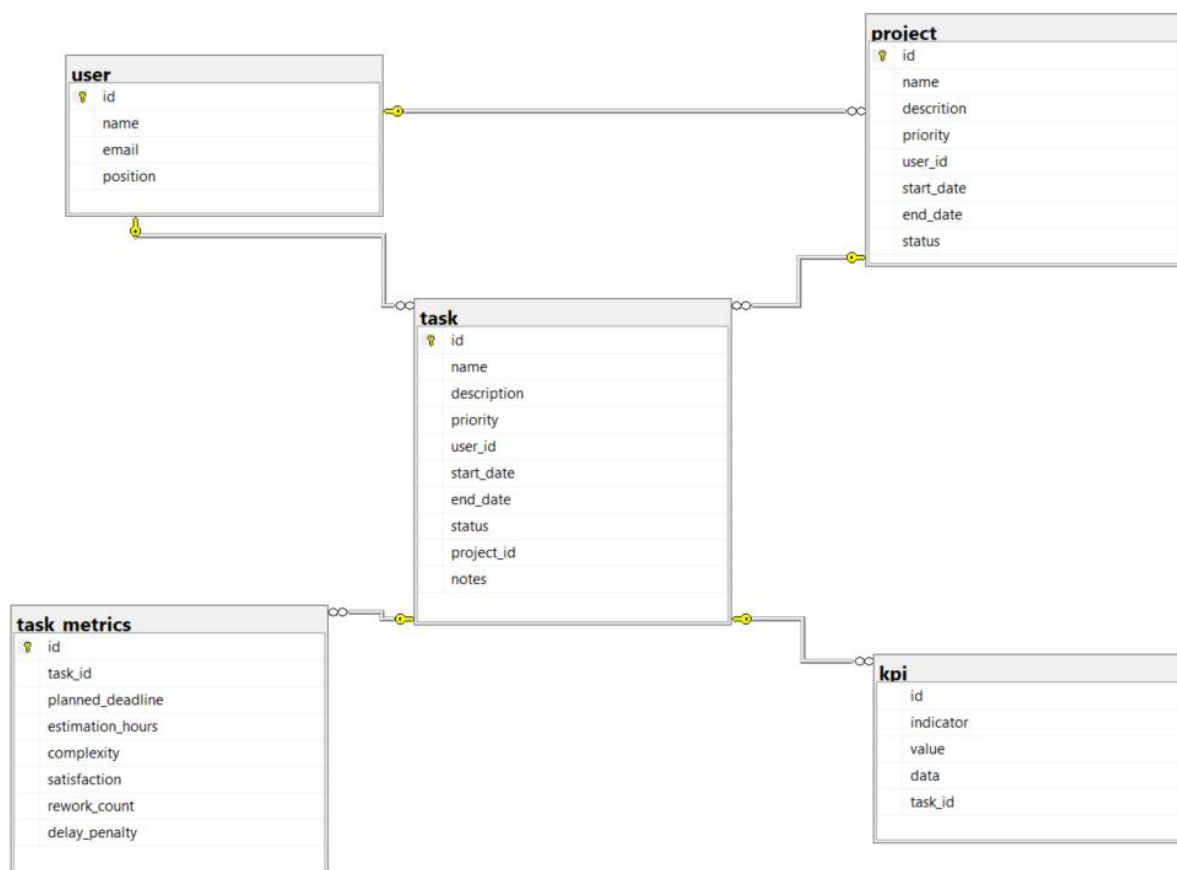


Рис. 3.6. Даталогічна модель БД (джерело автора)

Ця даталогічна модель відображає інфологічну модель в термінах конкретної системи керування базами даних (у даному випадку - SQL Server) і може бути використана для створення та управління базою даних у відповідній системі.

Також на мою думку невід’ємною частиною проектування БД є інтеграція математичного забезпечення. Такий підхід дозволить оптимізувати систему та підвищити швидкість обробки запитів користувачів.

В першу чергу потрібно визначитись з можливими варіантами інтеграції математичного забезпечення. Я розглянув такі рішення:

Варіант 1: Представлення (VIEW)

Побудова уніфікованих уявлень (view), які агрегують необхідні поля з таблиць task, task_metrics, user, project. У цих представленнях реалізуються формули з розділу математичного забезпечення:

Приклад:

```
CREATE VIEW vw_user_kpi_summary AS
```

(запит на основі попередньо наданої формули KPI)

Переваги:

- Автоматичне оновлення при зміні вихідних даних
- Придатність для використання у звітах, BI-додатках
- Мінімальний обсяг пам’яті

Варіант 2: Збережені процедури (Stored Procedures)

Використовуються для розрахунку KPI за запитом із параметрами (наприклад, user_id, date_range). Результат — динамічна таблиця або тимчасовий результат.

Приклад:

```
EXEC sp_GetUserPerformance @user_id = 7, @start_date = '2024-01-01'
```

Переваги:

- Гнучкість параметризації
- Підтримка бізнес-логіки та складних обчислень

- Можливість вбудованого кешування

Варіант 3: Модифікація таблиць – матеріалізовані поля

Створення додаткових колонок у таблицях, наприклад:

- kpi_score NUMERIC
- is_ontime BIT
- productivity FLOAT

Вони заповнюються через тригери або регулярні job-и.

Переваги:

- Максимальна продуктивність при вибірці
- Зберігання історичних значень KPI

Недоліки:

- Потреба в синхронізації при зміні даних
- Надлишковість інформації

Варіант 4: Аналітичні функції через Common Table Expressions (CTE)

Використання SQL-запитів із WITH ... AS для формування гнучких тимчасових підрахунків. Це може бути частиною складніших об'єднаних запитів.

Аналізувавши 4 попередні варіанти було прийнято рішення скомбінувати кілька підходів:

- основні розрахунки KPI — через VIEW та STORED PROCEDURES
- критично важливі, часто використовувані значення (наприклад, kpi_score) — зберігаються у таблицях з періодичним оновленням.

З метою реалізації математичного забезпечення безпосередньо у структурі бази даних, у СУБД було створено уніфіковане представлення (VIEW), яке забезпечує автоматизований розрахунок ключових показників ефективності (KPI) на основі

зібраних первинних даних у таблицях task, task_metrics, kpi та user. Для збереження нормалізованості структури та уникнення дублювання інформації, результати обчислень KPI інтегруються через поле value таблиці kpi, де зберігається підсумковий коефіцієнт ефективності, обчислений відповідно до математичних формул, викладених у розділі математичного забезпечення.

Обчислення здійснюється динамічно за допомогою представлення vw_user_kpi, яке базується на приєднаних таблицях та обчислює зважену метрику ефективності за формулою KPI:

$$KPI = 0.2 \cdot \theta + 0.2 \cdot (1 - \delta) + 0.2 \cdot \sigma + 0.2 \cdot (1 - \psi) + 0.2 \cdot (1 - \phi)$$

Кожен показник береться у нормалізованому вигляді та агрегується на рівні користувача. Для гнучкості реалізації, обчислення побудоване як віртуальний представлення, що оновлюється при кожному зверненні та дозволяє аналітично оцінювати динаміку продуктивності в реальному часі.

SQL-код для створення представлення vw_user_kpi:

```
CREATE VIEW vw_user_kpi AS
SELECT
u.id AS user_id,
u.name AS user_name,
-- 1. Своєчасність виконання задач
AVG(CASE WHEN t.end_date <= tm.planned_deadline THEN 1.0 ELSE 0.0 END) AS
ontime_rate,

-- 2. Нормалізована затримка (delay_score ∈ [0.0; 1.0])
CASE
WHEN AVG(DATEDIFF(DAY, tm.planned_deadline, t.end_date)) < 0 THEN 1.0
WHEN AVG(DATEDIFF(DAY, tm.planned_deadline, t.end_date)) > 10 THEN 0.0
ELSE 1.0 - (AVG(DATEDIFF(DAY, tm.planned_deadline, t.end_date)) / 10.0)
END AS delay_score,

-- 3. Нормалізована середня задоволеність (по 10-бальній шкалі)
AVG(tm.satisfaction * 1.0) / 10.0 AS satisfaction_score,

-- 4. Зворотна шкала доопрацювань (макс. 5 правок)
1.0 - (AVG(tm.rework_count * 1.0) / 5.0) AS rework_score,

-- 5. Зворотна шкала штрафів (макс. 1000 грн на задачу)
1.0 - (AVG(tm.delay_penalty * 1.0) / 1000.0) AS penalty_score,

-- Підсумковий KPI як зважене середнє
(
```

```

0.2 * AVG(CASE WHEN t.end_date <= tm.planned_deadline THEN 1.0 ELSE 0.0
END) +
0.2 * (
CASE
WHEN AVG(DATEDIFF(DAY, tm.planned_deadline, t.end_date)) < 0
THEN 1.0
WHEN AVG(DATEDIFF(DAY, tm.planned_deadline, t.end_date)) > 10
THEN 0.0
ELSE 1.0 - (AVG(DATEDIFF(DAY, tm.planned_deadline, t.end_date))
/ 10.0)
END
) +
0.2 * (AVG(tm.satisfaction * 1.0) / 10.0) +
0.2 * (1.0 - (AVG(tm.rework_count * 1.0) / 5.0)) +
0.2 * (1.0 - (AVG(tm.delay_penalty * 1.0) / 1000.0))
) AS value
FROM dbo.[user] u
JOIN dbo.task t ON t.user_id = u.id
JOIN dbo.task_metrics tm ON tm.task_id = t.id
WHERE t.status = 'виконано'
GROUP BY u.id, u.name;

```

Результати що відображає представлення можна побачити в додатку Б

Наступним кроком була реалізована **збережена процедура** для збереження згенерованих метрик.

```

CREATE PROCEDURE sp_InsertUserKPI
AS
BEGIN
SET NOCOUNT ON;
INSERT INTO dbo.kpi (user_id, indicator, value, data)
SELECT
v.user_id,
'user_kpi' AS indicator,
v.value,
GETDATE() AS data
FROM vw_user_kpi v
END;

```

Для використання процедури потрібно виконати: EXEC sp_InsertUserKPI;

Описана реалізація математичного забезпечення безпосередньо в базі даних дозволяє досягти високого рівня автоматизації обчислення ключових показників ефективності (KPI) в інформаційній системі управління проєктами. Завдяки застосуванню уніфікованого представлення vw_user_kpi, усі необхідні обчислення (своєчасність, затримки, задоволеність, доопрацювання та штрафи) виконуються динамічно на рівні СУБД, без потреби в зовнішній аналітичній обробці. Такий

підхід забезпечує актуальність результатів, гнучкість аналізу в режимі реального часу та зручну інтеграцію з інтерфейсними модулями для візуалізації метрик або формування звітів.

Додатковим кроком стало впровадження збереженої процедури `sp_InsertUserKPI`, яка дозволяє формально зафіксувати значення KPI у таблиці `kpi`, забезпечуючи їх архівування, історичну аналітику та повторне використання у процесах контролю, звітності або ухвалення управлінських рішень. Такий комплексний підхід інтеграції математичного апарату в структуру бази даних дозволяє не лише проводити аналітичну обробку, а й оптимізувати навантаження на клієнтську частину платформи, зменшуючи час відповіді системи на типові запити, а також створює передумови для масштабування рішення без втрати продуктивності.

3.2 Проектування бази знань інформаційної системи управління проєктами та засоби інтелектуального аналізу даних

У сучасних умовах реалізації багаторівневих цифрових проєктів особливого значення набуває впровадження в інформаційну систему бази знань, яка виконує функцію накопичення, узагальнення та інтерпретації критично важливої інформації про проєктну діяльність. У поєднанні з засобами інтелектуального аналізу даних (Data Mining) така база створює аналітичне підґрунтя для прийняття обґрунтованих управлінських рішень, оцінювання продуктивності, прогнозування ефективності команди та ідентифікації потенційних проблем на ранніх етапах життєвого циклу проєкту.

У межах даного проєкту реалізація бази знань та аналітичного шару здійснюється з використанням Microsoft SQL Server, що є надійною та високопродуктивною системою управління базами даних із широкими можливостями підтримки бізнес-аналітики, вбудованих функцій аналітичної обробки та інструментів роботи з великими обсягами даних. Вибір саме цієї СУБД обумовлений її здатністю ефективно реалізовувати складні аналітичні запити,

підтримкою віконних функцій, процедур зберігання, тригерів, а також вбудованих засобів побудови звітності (SSRS) і обробки багатовимірних даних (SSAS).

База знань проєктної діяльності містить:

- агреговані історичні дані про виконання задач та проєктів (строки, статуси, обсяг виконаних завдань, кількість і типи змін);
- ключові показники ефективності (KPI) з прив'язкою до задач, користувачів і часових проміжків;
- структуровані зв'язки між типами задач, рівнем їх пріоритетності та результативністю виконання;
- динамічні характеристики проєктів, що дозволяють проводити порівняльний аналіз.

Усі ці дані зберігаються у вигляді спеціалізованих звітних таблиць (data marts), що формуються на основі операційних даних системи. Їх підготовка здійснюється за допомогою T-SQL-скриптів, які реалізують логіку трансформації, обчислення агрегатів та формування зв'язків у базі знань.

Для реалізації інтелектуального аналізу даних у системі використовується комбінація таких методів:

- описова аналітика (зведення, гістограми);
- кореляційний аналіз (вплив пріоритетності задач на час виконання);
- класифікація задач за складністю або типом виконавця;
- прогнозування KPI на основі історичних трендів.

Отримані результати подаються користувачам у вигляді інтерактивних звітів та дашбордів, реалізованих через зовнішні клієнтські бібліотеки (*Chart.js*, *Apache ECharts*) або засоби звітності SQL Server (наприклад, SQL Server Reporting Services - SSRS).

Таким чином, база знань, інтегрована з інтелектуальними засобами аналізу, виконує функцію аналітичного ядра інформаційної системи управління проєктами, слугуючи основою для підтримки стратегічних та тактичних управлінських рішень. Її значення полягає не лише в акумуляції структурованої інформації про стан задач, проєктів і командної діяльності, а й у здатності формувати глибоке аналітичне розуміння процесів, що відбуваються в організації. Завдяки наявності детальних даних про ключові показники ефективності (KPI), динаміку виконання завдань, завантаженість виконавців, історію змін та відхилень - система здатна здійснювати постфактум оцінку проєктної діяльності, виявляючи сильні й слабкі сторони реалізованих ініціатив.

Крім того, за допомогою вбудованих інструментів аналізу та візуалізації, система надає змогу формувати обґрунтовані прогнози щодо строків реалізації майбутніх проєктів, ресурсної потреби, ймовірності перевищення бюджетів або порушення дедлайнів. Це створює підґрунтя для проактивного управління, де рішення ухвалюються не лише на основі фактичних даних, а й із урахуванням передбачуваних сценаріїв. Такий підхід дозволяє своєчасно ідентифікувати ризики, зокрема перевантаження виконавців, затримки у виконанні критичних задач, неефективний розподіл пріоритетів, і відповідно - реалізовувати превентивні заходи.

У результаті впровадження бази знань із підтримкою інтелектуального аналізу даних відбувається підвищення загального рівня ефективності управління проєктами, посилення обґрунтованості рішень, зниження рівня невизначеності та забезпечення адаптивності організації до динамічних змін зовнішнього та внутрішнього середовища.

3.2.1 Організація та проведення інтелектуального аналізу даних з метою пошуку знань в базі даних

Інтелектуальний аналіз даних (Data Mining) виступає невід'ємним та ключовим компонентом аналітичного шару інформаційної системи управління

проектами, оскільки забезпечує якісно новий рівень роботи з інформацією. Його завдання полягає не лише у збиранні та обробці великих обсягів даних, але й у виявленні неочевидних залежностей, закономірностей, повторюваних структур і тенденцій, які залишаються поза межами традиційної аналітики. Саме завдяки цим можливостям система здатна надати управлінцю не просто набір звітів, а практично орієнтовані знання, що безпосередньо впливають на прийняття стратегічних і тактичних рішень.

На відміну від класичного підходу, коли інформаційна система обмежується лише пасивним зберіганням та відтворенням даних, застосування інтелектуального аналізу дозволяє реалізувати активну модель функціонування, у якій платформа самостійно формує аналітичні висновки, ідентифікує ризики, прогнозує критичні ситуації та пропонує рекомендації для подальших дій. Таким чином, система стає інструментом прогнозно-діагностичного характеру, здатним виявляти потенційні загрози ще до їх виникнення, оптимізувати розподіл трудових і часових ресурсів та обґрунтовано коригувати хід виконання проєктів.

Завдяки використанню методів кластеризації, класифікації, асоціативних правил, регресійного аналізу, а також візуалізації багатовимірних даних, забезпечується не лише глибоке осмислення накопиченої інформації, а й *формування бази знань*, яка служить джерелом безперервного вдосконалення управлінських процесів. У підсумку, інтелектуальний аналіз даних не лише розширює функціональні можливості ІС, а й перетворює її на адаптивну аналітичну платформу, що сприяє підвищенню продуктивності команди, дотриманню дедлайнів і загальному зростанню ефективності реалізації проєктів.

Організація інтелектуального аналізу даних у межах розробленої інформаційної системи базується на використанні Microsoft SQL Server як централізованої платформи зберігання та обробки даних. Ця СУБД є не лише надійним середовищем для збереження інформації, але й потужним аналітичним інструментом завдяки розширеним можливостям мови запитів Transact-SQL (T-SQL). Саме використання T-SQL дозволяє реалізовувати складні алгоритми аналізу даних без необхідності залучення зовнішніх програмних рішень, що, своєю чергою,

забезпечує скорочення часу обробки та зменшення обсягу передавання інформації між серверами і клієнтами.

Завдяки використанню віконних функцій (OVER, PARTITION BY, RANK, LAG/LEAD тощо), агрегатних обчислень, підзапитів, табличних змінних, CTE (Common Table Expressions) та збережених процедур, платформа SQL Server забезпечує ефективну реалізацію операцій багатовимірного аналізу, класифікації даних, обчислення динамічних показників і побудови логічно обґрунтованих звітів.

Аналітична інфраструктура, сформована на основі SQL Server, дозволяє:

- виконувати ретроспективний аналіз даних, тобто аналізувати хронологію виконання задач, відповідність строкам, обсяг перенесених завдань;
- здійснювати порівняння ефективності роботи різних учасників у межах одного або кількох проєктів;
- виявляти нормативні відхилення між плановими і фактичними значеннями показників (час виконання, пріоритети, обсяг завдань);
- будувати адаптивні звіти в режимі реального часу, які формуються автоматично на основі запитів до бази знань і відображаються у вигляді графіків, таблиць, зведень та діаграм.

Завдяки зосередженню логіки аналізу на рівні СУБД, зменшується навантаження на прикладні модулі, підвищується швидкодія системи, забезпечується централізація зберігання та обробки критично важливої інформації. Це, своєю чергою, робить систему надійною, масштабованою та здатною до обробки великих масивів даних без втрати ефективності.

Для практичного застосування SQL-аналітики було поставлено два аналітичних завдання:

1. Оцінити продуктивність кожного користувача (виконавця) за такими критеріями:

- Загальна кількість задач, які він виконав;
- Середній час виконання задач (у днях);

- Частка задач, виконаних вчасно (до `end_date` $\leq$ дедлайну - для прикладу, використаємо `end_date` як кінцеву планову дату);
- Кількість задач з високим пріоритетом (наприклад, `priority = 1` як "високий").

T-SQL-запит для виконання завдання з створенням представлення (VIEW):

```
CREATE VIEW vw_UserTaskPerformance AS
SELECT
    u.id AS user_id,
    u.name AS user_name,
    COUNT(t.id) AS total_tasks,
    AVG(DATEDIFF(DAY, t.start_date, t.end_date)) AS avg_completion_days,
    SUM(CASE
        WHEN t.end_date <= GETDATE() THEN 1
        ELSE 0
    END) * 100.0 / COUNT(t.id) AS percent_on_time,
    SUM(CASE
        WHEN t.priority = 1 THEN 1
        ELSE 0
    END) AS high_priority_tasks
FROM
    dbo.[user] u
LEFT JOIN
    dbo.task t ON t.user_id = u.id
WHERE
    t.status = N'виконано'
GROUP BY
    u.id, u.name;
```

Переглянути представлення можна у додатку В.

2. Оцінити якість виконання задач за допомогою показників КРІ за такими критеріями:

- Кількість показників КРІ, зафіксованих для кожної задачі;
- Середнє значення КРІ по задачі (наприклад, продуктивність, ефективність, рівень виконання);
- Дата останнього оновлення КРІ;
- Поточний статус задачі та її пріоритет;
- Виконавець, відповідальний за виконання задачі.

Цей аналіз дозволяє виявити, наскільки ефективно виконуються задачі, які метрики фіксуються найчастіше, чи досягаються очікувані значення KPI та чи є відхилення, що можуть впливати на загальний рівень ефективності проєктів.

T-SQL-запит для виконання завдання з створенням представлення (VIEW):

```
CREATE VIEW vw_TaskKPIAnalysis AS
SELECT
    t.id AS task_id,
    t.name AS task_name,
    u.name AS assigned_user,
    COUNT(k.id) AS total_kpi_records,
    AVG(CAST(k.value AS FLOAT)) AS avg_kpi_value,
    MAX(k.data) AS last_kpi_date,
    t.status,
    t.priority
FROM
    dbo.task t
JOIN
    dbo.[user] u ON t.user_id = u.id
LEFT JOIN
    dbo.kpi k ON t.id = k.task_id
GROUP BY
    t.id, t.name, u.name, t.status, t.priority;
```

Переглянути представлення можна у додатку Г.

3.2.2 Проєктування та реалізація бази знань

База знань є ключовим інформаційно-аналітичним компонентом інформаційної системи управління проєктами, що виконує стратегічну функцію централізованого накопичення, структуризації, збереження та обробки фактологічних і аналітичних даних, які формуються в ході планування, виконання та контролю проєктних завдань. На відміну від стандартного підходу, орієнтованого виключно на зберігання транзакційної інформації (наприклад, створення задач або зміна статусу), база знань акумулює дані, що мають довготривалу аналітичну цінність: ключові показники ефективності (KPI), історію рішень, зв'язки між задачами й виконавцями, закономірності змін у продуктивності тощо.

Функціонування бази знань забезпечує якісний перехід від пасивного накопичення даних до їх осмисленого й цілеспрямованого використання. Зокрема, дані стають джерелом для аналізу результативності виконання задач, ідентифікації відхилень від планових показників, прогнозування потенційних ризиків та формування рекомендацій для підвищення ефективності роботи проєктних команд. Такий підхід створює підґрунтя для підтримки прийняття рішень на основі даних (data-driven decision making), що відповідає сучасним практикам цифрового управління проєктами.

Крім того, база знань забезпечує стандартизацію інформаційного простору, у якому всі структурні елементи системи - задачі, проєкти, користувачі, метрики, часові інтервали - узгоджені між собою, формують єдину онтологію даних і доступні для подальшої обробки за допомогою інтелектуальних алгоритмів. Це створює передумови для впровадження моделей машинного навчання, предиктивної аналітики та сценарного моделювання, які базуються саме на повноцінній, структурованій базі знань.

Логічна структура бази знань

Проєктована база знань реалізується у межах СУБД Microsoft SQL Server та базується на існуючій логічній моделі даних, яка включає сутності:

- task - задачі проєктів;
- user - виконавці та менеджери;
- project - проєкти;
- kpi - ключові показники ефективності (Key Performance Indicators), прив'язані до задач.

Центральну роль у базі знань відіграє таблиця kpi, яка фіксує метрики оцінювання задач у динаміці. Саме вона становить основу для побудови аналітичних уявлень, у яких дані агрегуються, обчислюються та подаються у вигляді узагальнених звітів.

Інфологічна модель реалізації знань

На основі даних з таблиць task і kpi у процесі реалізації створюються представлення (VIEW), які функціонують як віртуальні аналітичні таблиці,

забезпечуючи доступ до знань без дублювання даних. Наприклад, у підрозділі 3.2.1 було створено представлення `vw_TaskKPIAnalysis`, що агрегує показники по задачах, і `vw_UserTaskPerformance`, яке виконує аналіз продуктивності користувачів.

Крім представлень, для підвищення ефективності роботи з базою знань використовуються:

- збережені процедури (Stored Procedures) - для оновлення зведених таблиць, формування аналітичних зведень;
- індекси - для пришвидшення пошуку по задачах, користувачах та показниках;
- тригерна логіка - для автоматичного запису нових KPI або фіксації подій зміни статусу задачі.

Фізичне розгортання та інтеграція

База знань розгортається на рівні тієї ж СУБД, що й операційна частина системи, із чітким логічним поділом:

- операційні таблиці (`task`, `user`, `project`, `kpi`) слугують джерелом даних;
- аналітичні представлення (`vw_*`) забезпечують швидкий доступ до агрегованої інформації;
- клієнтські інтерфейси (дашборди, звіти) взаємодіють саме з базою знань для побудови візуалізацій.

Реалізація бази знань здійснюється шляхом створення спеціалізованої аналітичної надбудови над операційною базою даних, яка включає представлення (VIEW), збережені процедури (Stored Procedures), індекси та інші механізми оптимізації. Ця надбудова орієнтована на забезпечення гнучкого та швидкого доступу до зведеної, агрегованої та аналітичної інформації, яка підтримує управлінські рішення в межах системи.

У рамках даного розділу кваліфікаційної магістерської роботи було розроблено дві збережені процедури (Stored Procedure):

1. Збережена процедура для динамічного зведеного звіту **KPI по проєкту**.

Ця процедура дає змогу аналітику або менеджеру викликати деталізований KPI-звіт по будь-якому проєкту за ідентифікатором.

```

CREATE PROCEDURE sp_GetProjectKPIReport
    @project_id NUMERIC
AS
BEGIN
    SELECT
        p.name AS project_name,
        t.name AS task_name,
        u.name AS assigned_user,
        k.indicator,
        k.value,
        k.data
    FROM dbo.project p
    JOIN dbo.task t ON p.id = t.project_id
    JOIN dbo.[user] u ON t.user_id = u.id
    LEFT JOIN dbo.kpi k ON t.id = k.task_id
    WHERE p.id = @project_id
    ORDER BY k.data DESC;
END;

```

Призначення та функціональність:

Ця процедура реалізує механізм динамічного отримання повного КРІ-звіту для конкретного проєкту, ідентифікованого параметром @project_id. На основі зв'язків між таблицями project, task, user та kpi процедура повертає деталізований перелік задач з відповідними показниками КРІ, включаючи назву задачі, виконавця, показник, його значення та дату фіксації.

Основні переваги:

- Використовується як основа для менеджерської звітності по проєктах.
- Дає змогу аналізувати якісні характеристики виконання задач у межах проєкту.
- Полегшує вивантаження звітів для подальшого аналізу або візуалізації.

2. Збережена процедура для отримання аналітичного звіту з продуктивності користувачів.

```

CREATE PROCEDURE sp_GetUserTaskPerformance
    @start_date DATE = NULL,
    @end_date DATE = NULL,
    @status NVARCHAR(20) = NULL

```

```

AS
BEGIN
    SET NOCOUNT ON;

    SELECT
        u.id AS user_id,
        u.name AS user_name,
        COUNT(t.id) AS total_tasks,
        AVG(DATEDIFF(DAY, t.start_date, t.end_date)) AS
avg_completion_days,
        SUM(CASE
            WHEN t.end_date <= GETDATE() THEN 1
            ELSE 0
        END) * 100.0 / NULLIF(COUNT(t.id), 0) AS percent_on_time,
        SUM(CASE
            WHEN t.priority = 1 THEN 1
            ELSE 0
        END) AS high_priority_tasks
    FROM
        dbo.[user] u
    LEFT JOIN
        dbo.task t ON t.user_id = u.id
    WHERE
        (@start_date IS NULL OR t.start_date >= @start_date)
        AND (@end_date IS NULL OR t.end_date <= @end_date)
        AND (@status IS NULL OR t.status = @status)
    GROUP BY
        u.id, u.name
    ORDER BY
        avg_completion_days ASC;
END;

```

Призначення та функціональність

Ця процедура надає гнучкий інструмент для моніторингу продуктивності співробітників, що беруть участь у виконанні задач у системі. Вона дозволяє згенерувати зведений звіт по кожному користувачу, який містить загальну кількість виконаних задач, середню тривалість виконання, відсоток задач, виконаних у межах дедлайну, а також кількість задач з високим пріоритетом.

Ключові особливості:

- Параметри @start_date, @end_date, @status дозволяють фільтрувати вибірку за часовими межами та статусом задач (наприклад, лише завершені).

- Обчислення за допомогою вбудованих T-SQL функцій забезпечує швидку обробку даних безпосередньо на рівні СУБД.
- Є ефективною основою для щотижневих/щомісячних аналітичних звітів, що можуть інтегруватися в дашборди або системи сповіщень.

Результати реалізації

У результаті реалізації бази знань у межах інформаційної системи управління проектами було сформовано цілісну аналітичну підсистему, яка забезпечує систематизовану обробку, зберігання та узагальнення ключових показників діяльності, пов'язаних із виконанням проєктів та задач. Архітектура цієї підсистеми базується на технологіях Microsoft SQL Server, що дозволяє поєднувати надійне зберігання даних з високопродуктивною аналітикою на рівні бази даних.

По-перше, забезпечено довготривале зберігання та структуровану фіксацію ключових показників ефективності (KPI), які прив'язуються до задач і регулярно оновлюються на основі фактичних даних виконання. Дані з таблиці kpi агрегуються і виводяться через спеціалізовані представлення (vw_TaskKPIAnalysis) і збережені процедури (sp_GetProjectKPIReport), що дозволяє здійснювати глибокий аналіз результативності на рівні окремих задач або проєктів.

По-друге, впроваджено централізований механізм побудови звітів і візуалізацій. Створення представлення vw_UserTaskPerformance та збереженої процедури sp_GetUserTaskPerformance забезпечило можливість отримання узагальнених звітів за визначеними критеріями (середня тривалість виконання задач, відсоток виконання у межах строків, пріоритетність завдань тощо). Ці об'єкти можуть використовуватись у графічних інтерфейсах, звітних платформах або BI-системах.

По-третє, реалізоване рішення стало дієвим інструментом підтримки прийняття управлінських рішень. Дані, що зберігаються в базі знань, дозволяють керівникам аналізувати поточний стан виконання проєктів, виявляти вузькі місця у виконанні задач, оцінювати завантаження співробітників і визначати динаміку зміни ефективності роботи команди.

По-четверте, проєктована структура бази знань створює платформу для подальшого розвитку аналітичної складової системи. Логічна модель та інструментальна база SQL Server забезпечують можливість інтеграції алгоритмів машинного навчання, автоматичної класифікації задач, прогнозування ризиків невиконання дедлайнів та створення рекомендаційних підсистем на основі історичних даних.

Таким чином, розроблена база знань не є лише сховищем даних, а виступає комплексним аналітичним середовищем, що формує інтелектуальну основу системи управління проєктами. Вона дозволяє здійснювати управління на основі об'єктивних, перевірених і постійно оновлюваних даних, сприяючи підвищенню прозорості, керованості та ефективності всієї проєктної діяльності організації.

3.3. Програмне забезпечення

Програмне забезпечення виступає центральним елементом функціонування інформаційної системи управління проєктами, оскільки саме воно реалізує прикладну логіку, інтегрує всі функціональні модулі системи та забезпечує взаємодію кінцевих користувачів із даними. Його функції охоплюють повний цикл взаємодії з інформаційною системою: від введення даних, обробки та аналізу до виведення результатів у зручній формі за допомогою графічного інтерфейсу та аналітичних панелей.

Однією з ключових функціональних властивостей програмного забезпечення є забезпечення доступу до бази знань, що акумулює інформацію про проєкти, задачі, KPI, користувачів і результати виконання. За допомогою прикладної логіки здійснюється керування проєктами та задачами, зміна їхніх статусів, розподіл обов'язків між учасниками команди, а також формування звітів і візуалізація показників ефективності. Усе це відбувається у реальному часі та з урахуванням ролей користувачів, їхніх прав доступу та поточних параметрів взаємодії.

З технічної точки зору, архітектура програмного забезпечення побудована за принципами модульності, що передбачає чіткий поділ функціональності на незалежні компоненти (керування проєктами, задачами, аналітикою,

користувачами, автентифікацією тощо). Це значно полегшує обслуговування, тестування та масштабування системи. У межах мікромодульної організації кожен компонент виконує ізольовану функцію, але водночас взаємодіє з іншими через стандартизовані інтерфейси.

Крім того, реалізація клієнт-серверної архітектури забезпечує фізичний та логічний поділ на рівні обробки даних (сервер), бізнес-логіки (додаток) та інтерфейсу користувача (клієнт). Такий підхід дозволяє гнучко масштабувати систему у разі зростання кількості користувачів або обсягу даних, а також безперешкодно додавати нові функціональні модулі без порушення цілісності загальної архітектури.

Таким чином, реалізоване програмне забезпечення виконує роль інтеграційного середовища, що об'єднує всі логічні й інформаційні компоненти системи, забезпечуючи злагоджену роботу, надійність функціонування та зручність у використанні для кінцевих користувачів.

У межах цього проєкту програмне забезпечення реалізовано як веб-платформа, що поєднує інтуїтивно зрозумілий інтерфейс користувача, прикладну логіку та серверну частину з доступом до бази даних. Для розробки використано стек сучасних технологій, що включає:

- Frontend (клієнтська частина): реалізовано з використанням JavaScript-бібліотеки React, яка забезпечує компонентну структуру інтерфейсу, повторне використання елементів, динамічне оновлення стану та високу швидкодію. Інтерфейс адаптований для різних ролей користувачів (менеджер, виконавець) і дозволяє керувати проєктами, створювати задачі, призначати виконавців, оновлювати статуси, а також переглядати звіти й аналітику.
- Backend (серверна частина): реалізована на базі Node.js із використанням Express.js для маршрутизації запитів та забезпечення RESTful-архітектури.

Серверна логіка відповідає за авторизацію, обробку запитів до бази даних, валідацію даних, генерацію звітів та формування аналітики.

- СУБД (сервер даних): для зберігання, обробки й аналітики використовується Microsoft SQL Server, що дозволяє реалізовувати як транзакційні, так і аналітичні запити, працювати з представленнями (VIEW), збереженими процедурами (Stored Procedure) та забезпечувати надійну підтримку цілісності даних.
- API-шар: між клієнтською та серверною частинами функціонує REST API, який забезпечує передачу структурованих JSON-повідомлень, підтримує авторизацію через HTTPS, обробку помилок і стандартизовані відповіді для фронтенду.
- Система візуалізації даних: для аналітичних панелей, дашбордів і діаграм використано бібліотеки Chart.js та Apache ECharts, що дозволяють реалізовувати інтерактивну аналітику (діаграми продуктивності, розподіл задач, тренди KPI тощо).

Програмне забезпечення побудовано за принципами трирівневої архітектури, де рівень презентації, бізнес-логіки та зберігання даних розділено, що забезпечує легкість модифікації та масштабування. Упроваджені механізми аутентифікації та авторизації на основі токенів доступу дозволяють забезпечити контроль за правами користувачів і безпечну взаємодію з даними.

Загальна реалізація програмного забезпечення дозволила інтегрувати всі компоненти системи - інтерфейс, аналітику, управління задачами та звітністю - у єдине середовище, орієнтоване на зручність користувача, гнучкість у налаштуваннях та підтримку управлінських рішень у реальному часі.

3.3.1 Вимоги до програмного забезпечення

На етапі розроблення програмного забезпечення для інформаційної системи управління проєктами одним із ключових завдань є формалізація вимог, які

визначають межі функціонування системи, її очікувану поведінку та якісні характеристики. Цей етап має критичне значення, оскільки саме вимоги формують основу для технічного проєктування, вибору архітектурних рішень, інструментальних засобів розробки, а також впливають на процеси тестування, розгортання та супроводу системи.

Вимоги до програмного забезпечення мають комплексний характер і поділяються на дві основні категорії: функціональні та нефункціональні. Функціональні вимоги описують, що саме має виконувати система, тобто перелік операцій, які мають бути доступні користувачеві, логіку обробки інформації, взаємодію з іншими компонентами, поведінку інтерфейсу тощо. У контексті системи управління проєктами до таких вимог належать: керування проєктами й задачами, фіксація показників KPI, генерація звітів, аналіз ефективності виконання тощо.

Натомість нефункціональні вимоги визначають якість виконання функціоналу та охоплюють такі аспекти, як продуктивність, масштабованість, надійність, безпека, сумісність із платформами, доступність та ергономічність. Вони забезпечують відповідність системи технічним обмеженням середовища, вимогам безпеки, очікуванням кінцевих користувачів щодо зручності використання, а також галузевим стандартам якості програмного забезпечення.

Визначення та документація вимог є необхідною передумовою для цілеспрямованої розробки, оскільки вони слугують критерієм валідації та верифікації проєктних рішень, а також інструментом комунікації між аналітиками, розробниками, тестувальниками та замовником. Відтак, у межах цього підрозділу систематизовано основні вимоги до програмного забезпечення як складової частини розроблюваної інформаційної системи.

Функціональні вимоги.

Основні функціональні вимоги до інформаційної системи управління проєктами охоплюють ключові модулі, необхідні для забезпечення повного життєвого циклу проєкту - від створення до завершення, а також підтримки взаємодії між усіма учасниками проєктного процесу.

1. Авторизація та автентифікація користувачів:

- Реєстрація нових користувачів із введенням ідентифікаційних та контактних даних.
- Реалізація входу до системи через логін і пароль із валідацією облікових даних.
- Розмежування прав доступу до функціоналу відповідно до ролі користувача: **менеджер** або **виконавець**.
- Захист від несанкціонованого доступу через сесійну перевірку та підтримку шифрованого з'єднання (HTTPS).
- Можливість відновлення пароля або повторної активації облікового запису.

2. Управління проєктами:

- Створення нових проєктів із зазначенням назви, опису, дати початку, дати завершення та рівня пріоритету.
- Призначення відповідального менеджера та учасників команди.
- Можливість редагування параметрів активного проєкту та видалення завершених або неактуальних проєктів.
- Відображення статусу проєкту (наприклад: «активний», «виконаний», «архівований»).
- Ведення журналу змін у проєкті для прозорості управлінських дій.

3. Управління задачами:

- Створення задач у межах певного проєкту із зазначенням назви, опису, дедлайну, пріоритету та відповідального виконавця.
- Редагування задач: оновлення опису, зміна статусу (наприклад: «заплановано», «у роботі», «виконано»), додавання приміток і коментарів.
- Автоматичне сповіщення виконавців про нові задачі або зміни в них.
- Можливість перегляду історії змін кожної задачі.
- Формування переліку задач у вигляді таблиці, списку або канбан-дошки з фільтрами за проєктом, статусом, виконавцем.

4. Формування звітів:

- Автоматичне або ручне створення звітів про виконання задач по кожному проєкту.
- Зведена аналітика про виконання задач окремими виконавцями або командами.
- Формування звітів за ключовими показниками ефективності (KPI), з можливістю експорту у формат PDF, Excel.
- Історичне порівняння результатів по проєктах і задачах у межах обраного періоду.
- Доступ керівника до узагальнених звітів та можливість побудови графіків продуктивності.

5. Візуалізація аналітики:

- Графічне представлення даних за допомогою інтерактивних діаграм, гістограм, кругових графіків, лінійних трендів.
- Побудова дашбордів у режимі реального часу, що відображають основні метрики ефективності проєктної діяльності.
- Можливість фільтрації аналітики за часом, користувачем, проєктом або статусом задач.
- Відображення індексів виконання задач (наприклад: відсоток завершення, середній час виконання, пріоритетне навантаження).
- Інтеграція з модулем KPI для візуального відстеження результатів по ключових показниках у межах кожної задачі.

Нефункціональні вимоги:

Нефункціональні вимоги визначають **якісні характеристики програмного забезпечення**, що забезпечують його надійність, продуктивність, масштабованість та зручність використання в реальних умовах експлуатації. У контексті системи управління проєктами ці вимоги є ключовими для стабільної роботи платформи, безпечного зберігання даних та підтримки високої ефективності користувацької взаємодії.

1. Продуктивність:

- Веб-інтерфейс повинен забезпечувати відгук на користувацькі дії не довший за 1 секунду при стандартному навантаженні.
- База даних повинна обробляти запити на отримання звітів і KPI не довше 3 секунд для обсягів до 1000 задач.
- Система має забезпечувати мінімальне блокування при паралельному доступі користувачів до спільних ресурсів.

2. Масштабованість:

- Архітектура має підтримувати горизонтальне та вертикальне масштабування, що дозволяє розширювати обчислювальні ресурси без зміни логіки системи.
- Повинна бути можливість додавання нових функціональних модулів без порушення вже реалізованого функціоналу.
- Система повинна витримувати паралельну роботу щонайменше 50 користувачів без втрати продуктивності.

3. Безпека:

- Усі з'єднання з сервером повинні здійснюватися через захищений протокол HTTPS.
- Дані авторизації користувачів зберігаються у вигляді хешованих значень із використанням алгоритму SHA-256 або еквівалентного.
- Реалізований механізм ролей та доступу до функціоналу на основі принципу найменших привілеїв.
- Система повинна бути захищена від типових векторів атак, таких як SQL-ін'єкції, міжсайтове виконання скриптів (XSS), CSRF.

4. Надійність:

- Програмне забезпечення повинно автоматично зберігати зміни, у разі втрати зв'язку з сервером - відображати повідомлення та запобігати втраті даних.
- У разі збою або помилки система повинна забезпечити журналювання подій для подальшого аналізу.
- Повинна бути реалізована резервна копія бази даних із регулярним оновленням (не рідше одного разу на добу).

5. Зручність використання (юзабіліті):

- Інтерфейс користувача повинен бути інтуїтивно зрозумілим, з мінімумом дій для виконання основних операцій.
- Забезпечується адаптивність інтерфейсу для різних розширень екрана (включаючи планшети та ноутбуки).
- Візуальні елементи повинні бути стилістично узгодженими, з дотриманням єдиного дизайну, логіки розміщення та кольорових схем.
- Повинна бути реалізована контекстна допомога та повідомлення про помилки/успішні дії користувача.

6. Сумісність:

- Програмне забезпечення повинно коректно працювати в актуальних версіях браузерів: Google Chrome, Mozilla Firefox, Microsoft Edge.
- Серверна частина має підтримувати розгортання на хмарних та локальних серверах, зокрема у середовищах Windows Server або Docker.

Вимоги до середовища розгортання:

- Серверна частина: Node.js (версія ≥ 16), Express.js, SQL Server 2019.
- Клієнтська частина: React (версія ≥ 18), бібліотеки візуалізації Chart.js або Apache ECharts.
- Інфраструктура: Хостинг з підтримкою Node.js, MSSQL, HTTPS (наприклад, Azure, AWS або локальний сервер).
- Інтерфейс обміну даними: REST API з використанням JSON-формату.

У процесі визначення вимог до програмного забезпечення важливо враховувати не лише технічні обмеження, внутрішню структуру підприємства та особливості бізнес-процесів, а й очікування зовнішніх стейкхолдерів — передусім замовників ІТ-проектів. Замовники виступають не лише ініціаторами задач, але й кінцевими бенефіціарами результатів, тому їхні потреби мають істотний вплив на архітектуру та функціонал системи.

У сучасній практиці управління ІТ-проєктами замовники дедалі частіше вимагають інтеграції в робочий процес — як шляхом надання зворотного зв'язку щодо виконання задач, так і через прямий моніторинг стану проєкту. Це обумовлює необхідність реалізації відкритих, клієнтоорієнтованих інтерфейсів взаємодії, з можливістю фільтрації даних, динамічного перегляду ключових метрик та формування звітів у зручному форматі. При цьому зростає потреба у налаштуванні ролей і прав доступу таким чином, щоби замовник міг бачити лише релевантну інформацію, не порушуючи цілісність внутрішніх процесів компанії-виконавця.

Крім того, зовнішні стейкхолдери очікують від системи високої адаптивності до змін вимог, що є типовим у сфері розробки ПЗ. З цієї причини інформаційна система повинна підтримувати не тільки стандартне створення та виконання задач, але й гнучке управління специфікаціями, повторними погодженнями, варіантами реалізації та трасуванням змін. Також важливим аспектом є впровадження механізмів інформування (email-, web- або мобільних повідомлень) щодо критичних подій проєкту — таких як зміна пріоритету, затримка задачі або перевищення бюджету — що дозволяє замовнику своєчасно реагувати на відхилення.

До ключових **вимог зовнішніх стейкхолдерів** належать:

- Прозорість виконання задач: необхідність у наочному відображенні поточного статусу завдань, графіку виконання, термінів завершення та відповідальних осіб.
- Якість зворотного зв'язку: можливість залишати оцінки, коментарі та зауваження до виконаних задач, що слугують основою для аналізу задоволеності клієнтів.
- Доступність аналітичної інформації: формування звітів про хід реалізації проєктів, відстеження ключових показників ефективності (KPI), оцінка ризиків та дотримання строків.

- Гнучкість взаємодії: можливість узгодження та редагування вимог до задач, створення нових запитів і взаємодія з менеджером проєкту безпосередньо через веб-інтерфейс.
- Захищеність даних: гарантія конфіденційності інформації, переданої замовником, та обмеження доступу до проєктної документації, відповідно до політик безпеки.

Інтеграція цих вимог у процес розроблення системи забезпечує не лише підвищення довіри до платформи з боку клієнтів, але й покращення якості управлінських рішень. Врахування потреб зовнішніх стейкхолдерів дозволяє зробити систему більш адаптивною, клієнтоорієнтованою та конкурентоспроможною у сфері управління ІТ-проєктами.

3.3.2 Інструментальні засоби розробки для створення прикладного програмного забезпечення інформаційної системи управління проєктами

Розроблення програмного забезпечення інформаційної системи управління проєктами вимагає комплексного підходу до вибору інструментальних засобів, які забезпечують повноцінну реалізацію функціональних вимог системи. Проєктування охоплює побудову повної архітектурної моделі, яка включає три ключові рівні: клієнтську частину (інтерфейс взаємодії з користувачем), серверну логіку (обробка запитів, контроль доступу, маршрутизація) та рівень зберігання й аналітичної обробки даних (база знань, збережені процедури, представлення, запити). Для кожного з цих рівнів було обрано спеціалізовані засоби розробки, що відповідають сучасним галузевим стандартам.

Клієнтська частина потребує інструментів, здатних забезпечити високу інтерактивність, адаптивність та зручність у користуванні. У цьому контексті перевага була надана бібліотекам з компонентною архітектурою, що дозволяють створювати гнучкий та масштабований інтерфейс, який динамічно реагує на зміни

в даних. Окрему увагу було приділено засобам побудови інтерактивної аналітики та візуалізації інформації - діаграм, графіків, показників KPI тощо.

На рівні серверної логіки ключовими критеріями при виборі інструментів стали продуктивність, підтримка REST-архітектури, можливість масштабування, а також підтримка сучасних механізмів безпеки (автентифікація, шифрування, контроль доступу). Обрані засоби дозволяють організувати централізовану обробку бізнес-логіки та ефективну взаємодію з базою даних через стандартизовані API-інтерфейси.

Рівень зберігання даних, представлений реляційною СУБД, потребує високої стійкості, підтримки складних SQL-запитів, оптимізації аналітичних операцій та можливості створення логічно пов'язаних структур даних. З цією метою була обрана промислова система керування базами даних, яка забезпечує гнучке моделювання сутностей, реалізацію представлень (VIEW), збережених процедур (Stored Procedure) та використання індексів для підвищення продуктивності.

При виборі засобів розробки враховувалися такі ключові вимоги, як:

- масштабованість - здатність системи адаптуватися до зростання навантаження;
- продуктивність - забезпечення швидкої реакції на дії користувача та обробки запитів;
- гнучкість розгортання - можливість налаштування середовища як на локальних, так і на хмарних платформах;
- сумісність із сучасними стандартами веб-технологій - підтримка REST API, адаптивна верстка, безпечний протокол HTTPS тощо;
- інтеграційна здатність - можливість взаємодії компонентів між собою через стандартизовані інтерфейси.

Таблиця 3.1.

Стек технологій програмного забезпечення (джерело автора)

Рівень архітектури	Застосовані технології	Призначення / функціональність
Клієнтська частина (Frontend)	React	Побудова динамічного та компонентного інтерфейсу користувача
	HTML5, CSS3, JavaScript	Веб-розмітка, стилізація елементів, базова логіка на стороні клієнта
	Chart.js / Apache ECharts	Візуалізація аналітики: графіки, діаграми, тренди KPI
Серверна частина (Backend)	Node.js	Розгортання серверної логіки, обробка запитів
	Express.js	Реалізація REST API, маршрутизація, обробка HTTP-запитів
	JWT, bcrypt.js	Автентифікація, хешування паролів, керування сесіями
База даних (Data layer)	Microsoft SQL Server	Реляційне зберігання даних, транзакційна обробка
	SQL, T-SQL	Написання запитів, створення представлень, збережених процедур, індексів
	SQL Server Management Studio	Моделювання структури БД, адміністрування, створення діаграм
Засоби розробки та супроводу	Visual Studio Code	Основне середовище розробки з підтримкою розширень для JS, Node.js, Git
	Postman	Тестування REST API, перевірка маршрутів та авторизації
	Git / GitHub	Система контролю версій, ведення історії змін, командна розробка
Протоколи та стандарти	HTTPS, REST, JSON	Безпечна передача даних, структура API, формат обміну інформацією

У таблиці 3.1 наведено повний стек технологій, що був використаний під час розроблення інформаційної системи управління проєктами. Структура таблиці відповідає трирівневій архітектурі програмного забезпечення та охоплює інструменти для реалізації кожного з її компонентів: клієнтської частини, серверної логіки, бази даних, а також допоміжних засобів розробки. Зокрема, для створення інтерфейсу користувача застосовано React, що забезпечує динамічну побудову

компонентів та високу адаптивність інтерфейсу. Для графічної візуалізації аналітичних даних використано бібліотеки Chart.js та Apache ECharts, що дозволяють реалізовувати інтерактивні графіки й діаграми у форматі реального часу.

Серверна логіка реалізована з використанням Node.js і фреймворку Express.js, які забезпечують обробку запитів, маршрутизацію, а також реалізацію RESTful API для обміну даними з клієнтською частиною. Для керування доступом і безпекою застосовано токенну автентифікацію (JWT) та хешування паролів через bcrypt.js. У ролі системи управління базами даних обрано Microsoft SQL Server, який забезпечує надійне зберігання, обробку та агрегацію даних, що використовуються у звітах і KPI-аналітиці. Робота з базою даних здійснюється через середовище SQL Server Management Studio. Додатково до процесу розробки були інтегровані засоби контролю версій (Git, GitHub), тестування API (Postman) та програмного середовища (Visual Studio Code), що забезпечили ефективність командної роботи, підтримку коду та якісне тестування функціональності.

У результаті попереднього аналізу вимог до системи, оцінки альтернативних технологічних рішень та врахування практичних обмежень середовища розроблення було сформовано оптимальний стек технологій, що дозволяє реалізувати всі необхідні функціональні компоненти системи в стислі терміни з дотриманням вимог до продуктивності, безпеки та зручності експлуатації. Обрані інструменти забезпечують високу швидкість розробки, що є критичним чинником для створення прикладного ПЗ у рамках академічного чи виробничого проєкту. Водночас вони гарантують стабільність функціонування у процесі тривалої експлуатації, мінімізацію технічних ризиків і можливість швидкого усунення потенційних збоїв.

Стек розробки було побудовано таким чином, щоб забезпечити гнучкість масштабування та подальшого вдосконалення системи, включаючи впровадження нових функціональних модулів, розширення бази знань або зміну механізмів візуалізації аналітики. Особливу увагу було приділено безпеці даних, зокрема

захисту каналів передачі, автентифікації користувачів і зберіганню конфіденційної інформації відповідно до сучасних практик веб-розробки.

У логічній структурі стеку кожен компонент відповідає за певний рівень архітектури: клієнтська частина (frontend) реалізує інтерактивний інтерфейс користувача; серверна частина (backend) обробляє бізнес-логіку, відповідає за автентифікацію, обробку запитів та обмін даними з базою; рівень бази даних (database) відповідає за зберігання, агрегацію та аналітичну обробку інформації. Крім того, використовуються допоміжні інструменти розробки, тестування, контролю версій і документування, що сприяють ефективному керуванню життєвим циклом програмного забезпечення.

3.3.3 Архітектура програмного забезпечення

Архітектура програмного забезпечення інформаційної системи управління проектами є визначальним чинником, що впливає на якість, надійність, масштабованість та підтримуваність розроблюваної системи. З метою досягнення високого рівня структурованості, модульності та адаптивності була обрана **трирівнева клієнт-серверна архітектура**, яка відповідає сучасним вимогам до веборієнтованих систем.

На першому рівні - **рівні презентації (Presentation Layer)** - функціонує клієнтська частина, реалізована з використанням бібліотеки React. Вона відповідає за формування графічного інтерфейсу користувача, обробку подій, валідацію форм і відображення аналітичних даних. Через REST API клієнт надсилає запити до серверної частини, отримує відповіді у форматі JSON та оновлює інтерфейс без перезавантаження сторінки, що забезпечує високу інтерактивність.

Другий рівень - **рівень логіки застосунку (Application Layer)** - представлений серверною частиною на Node.js з використанням Express.js. Він виконує роль посередника між інтерфейсом користувача та базою даних: обробляє запити, виконує авторизацію, забезпечує маршрутизацію, застосовує бізнес-правила та формує відповідні відповіді клієнту. На цьому рівні також реалізовано

механізми автентифікації через JWT, шифрування облікових даних, обробку помилок і ведення журналів активності.

Третій рівень - **рівень доступу до даних (Data Layer)** - реалізований на основі СУБД Microsoft SQL Server. Він відповідає за збереження структурованих даних про користувачів, проекти, задачі, KPI та інші сутності. У межах цього рівня функціонують збережені процедури, представлення (VIEW) для аналітики, індекси для оптимізації запитів і логічні зв'язки між таблицями відповідно до інфологічної моделі, розробленої раніше.

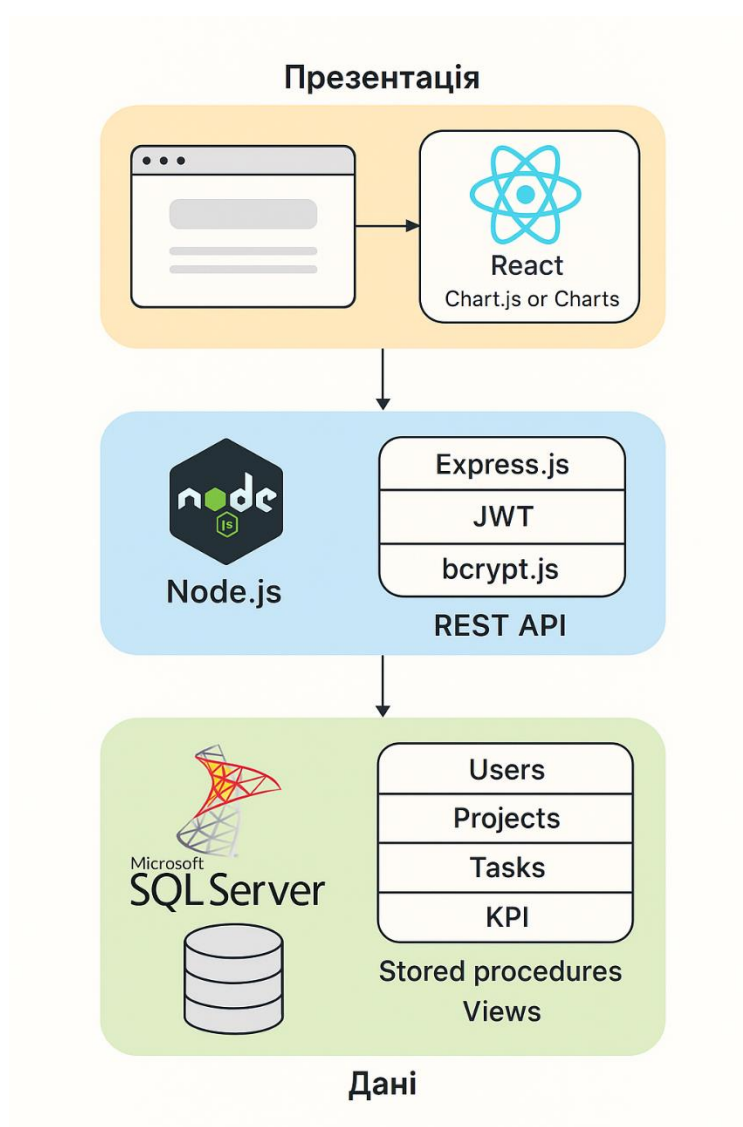


Рис. 3.7. Архітектурна діаграма (джерело автора)

Важливою особливістю обраної архітектури є її **модульність**, що дозволяє здійснювати незалежну розробку, тестування й оновлення окремих компонентів

без втручання в решту системи. Крім того, клієнт-серверна модель дозволяє масштабувати систему як горизонтально (додавання серверів додатку), так і вертикально (покращення ресурсів бази даних або фронтенду). Це забезпечує гнучкість у випадку розширення функціональності, зростання навантаження або інтеграції зі сторонніми сервісами.

3.3.4 Керівництво (інструкція) користувача

Інформаційна система управління проектами орієнтована на широке коло користувачів, які взаємодіють із нею в межах своїх повноважень і функціональних ролей. З метою забезпечення ефективного використання системи було розроблено інструкцію користувача, що містить опис базових операцій, етапів взаємодії з інтерфейсом, а також типові сценарії використання.

1. Авторизація та вхід у систему

- Для початку роботи користувач має перейти на головну сторінку веб-платформи.
- У формі входу необхідно вказати логін (email) та пароль, після чого натиснути кнопку «Увійти».
- У разі відсутності облікового запису передбачено можливість реєстрації (для співробітників) або створення адміністрацією (для менеджерів).
- У разі неправильного введення даних система виводить повідомлення про помилку.

2. Головна панель та навігація

- Після авторизації користувач потрапляє на головну панель, де доступні модулі: «Мої задачі», «Проекти», «Звіти», «KPI», «Профіль».
- Інтерфейс реалізований у вигляді адаптивного меню з підтримкою фільтрації, пошуку та перегляду задач у вигляді списку або канбан-дошки.

3. Робота з задачами

- Користувач має можливість переглядати призначені йому задачі, відсортовані за пріоритетом або дедлайном.

- Для кожної задачі можна оновити статус («заплановано», «у роботі», «виконано»), додати коментар або уточнити опис.
- Завершення задачі супроводжується записом про час виконання та фіксацією змін у базі даних.

4. Управління проєктами (для менеджера)

- Менеджер має змогу створювати нові проєкти, додаючи опис, строки, відповідальних осіб.
- У межах кожного проєкту можна створювати задачі, розподіляти їх між учасниками та відстежувати прогрес виконання.
- Доступна аналітика за проєктами – у вигляді KPI-метрик, діаграм виконання та звітів по задачах.

5. Аналітика та KPI

- У модулі аналітики користувач (переважно менеджер або аналітик) може переглядати зведені дані по ефективності проєктів і виконавців.
- Інтерфейс підтримує візуалізацію у вигляді графіків, гістограм та кругових діаграм (на основі Chart.js / ECharts).
- За потреби дані можна експортувати у форматах PDF або Excel.

6. Профіль та безпека

- Кожен користувач може переглядати й редагувати особисті дані (email, посаду).
- Реалізована функція зміни пароля та виходу із системи.
- Сесія автоматично завершується після тривалого періоду неактивності.

Таким чином, система побудована таким чином, щоб забезпечити максимальну зручність взаємодії, мінімальну криву навчання та високу ефективність виконання проєктних завдань. Інструкція користувача дозволяє швидко адаптуватися до роботи в системі навіть при мінімальному технічному досвіді, а інтерфейс орієнтований на інтуїтивну взаємодію та послідовне виконання дій.

3.3.5 Опис спроектованого програмного забезпечення

Спроектоване програмне забезпечення є центральним компонентом інформаційної системи управління проєктами, що реалізує прикладну логіку системи, забезпечує інтерактивну взаємодію користувачів із даними та підтримує процеси управління, моніторингу й аналітики в межах життєвого циклу проєкту. ПЗ побудоване відповідно до принципів модульності, багаторівневої архітектури та відкритості до масштабування, що забезпечує стабільну та ефективну роботу системи у середовищі з високими вимогами до надійності, продуктивності та гнучкості.

Система реалізована у вигляді веборієнтованої платформи, яка функціонує за схемою клієнт-серверної архітектури. Клієнтська частина (frontend) побудована на основі бібліотеки React та реалізує динамічний інтерфейс користувача з підтримкою інтерактивних елементів, асинхронної обробки подій та адаптивного відображення інформації. Серверна частина (backend), розроблена з використанням Node.js та Express.js, відповідає за маршрутизацію, логіку обробки запитів, автентифікацію користувачів і взаємодію з базою знань через REST API.

Важливою складовою спроектованого програмного забезпечення є реляційна база даних, яка забезпечує централізоване зберігання, структурування, обробку й аналітичне опрацювання інформації. База даних реалізована на платформі Microsoft SQL Server, що дозволило впровадити складну модель зв'язків між сутностями, використовуючи зовнішні ключі, індекси, збережені процедури, представлення (VIEW) та механізми забезпечення цілісності даних.

Інфологічна модель бази даних включає чотири основні сутності: користувачі, проєкти, задачі та KPI. Всі ці об'єкти мають чітко визначені атрибути, первинні ключі та логічні зв'язки між собою, що дозволяє гнучко фіксувати розподіл ролей, призначення задач, оцінку продуктивності та агрегування показників ефективності. Для підвищення ефективності аналітики розроблено набір представлень та аналітичних процедур, які дозволяють здійснювати вибірку даних за різними критеріями: за виконавцем, проєктом, часовими діапазонами, статусами виконання тощо.

База даних також виступає основою для реалізації інтелектуального аналізу (data mining), оскільки накопичена інформація дозволяє формувати прогнози, виявляти закономірності та будувати управлінські моделі. Такий підхід підвищує адаптивність системи, дозволяє менеджерам приймати обґрунтовані рішення на основі історичних та поточних даних, а також відкриває можливості для подальшої інтеграції елементів машинного навчання.

Таким чином, спроектоване програмне забезпечення виконує функцію інтегрованої платформи управління проєктною діяльністю, яка поєднує в собі декілька ключових компонентів. По-перше, воно забезпечує реалізацію логіки управління проєктами: створення проєктів, формування задач, призначення відповідальних осіб, встановлення пріоритетів і дедлайнів, контроль виконання та завершення робіт. Цей функціональний блок охоплює повний життєвий цикл задачі - від планування до звітності - з фіксацією усіх змін у базі даних.

По-друге, інтерфейсна частина платформи забезпечує зручну взаємодію з користувачем, реалізовану за допомогою адаптивного та динамічного фронтенду. Це дозволяє користувачам швидко орієнтуватися в системі, працювати з великою кількістю задач, фільтрувати дані, візуалізувати аналітику та отримувати миттєвий зворотний зв'язок на свої дії. Завдяки чіткому розмежуванню ролей та контекстно-чутливому відображенню функціоналу інтерфейс є інтуїтивно зрозумілим як для рядових виконавців, так і для менеджерів та аналітиків.

По-третє, аналітична складова системи базується на ефективно реалізованій реляційній базі даних, що виконує роль не лише сховища, а й ядра аналітичного обчислення. За допомогою збережених процедур, представлень і запитів SQL рівня агрегації, користувачі можуть отримувати зведену інформацію про прогрес виконання проєктів, дотримання строків, рівень продуктивності команди, а також ключові показники ефективності (KPI). Така інтеграція операційної та аналітичної логіки в єдиній системі дозволяє не лише реагувати на події в реальному часі, а й здійснювати стратегічне планування, прогнозування ризиків та підвищення ефективності організаційних процесів.

У сукупності ці компоненти створюють надійне, масштабоване та інформативне рішення, яке може бути ефективно впроваджене у практику управління проєктами в IT-сфері, освіті, консалтингу та інших галузях, де важливими є контроль, прозорість і аналітична підтримка управлінських рішень.

3.3.6 Користувацький інтерфейс вебсистеми

Користувацький інтерфейс (UI) є невід’ємним елементом ефективної інформаційної системи управління проєктами, оскільки саме через нього реалізується взаємодія кінцевого користувача з функціоналом системи. Враховуючи багаторівневу структуру ролей користувачів (менеджери, виконавці, аналітики, адміністратори), інтерфейс повинен враховувати специфіку їхніх завдань та відповідних сценаріїв використання системи.

Менеджери потребують зручного доступу до інструментів планування, призначення задач, моніторингу виконання проєктів, генерації звітів і аналітичних дашбордів. Для них важлива можливість швидко переглянути статуси задач, терміни завершення, пріоритети, а також показники ефективності працівників. Виконавці, зі свого боку, мають отримати просту й швидку можливість перегляду призначених задач, оновлення їх статусу, додавання коментарів та звітів. Інтерфейс має бути лаконічним, зосередженим на актуальних задачах і мінімалістичним у навігації, аби не відволікати від основного функціоналу.

Таким чином, розробка інтерфейсу повинна відбуватись за принципами *user-centered design* — з орієнтацією на потреби кожної цільової ролі користувача, підтримкою послідовної логіки взаємодії, зрозумілою навігацією, зворотним зв’язком на кожен запит користувача та адаптивним розміщенням функціональних блоків. Це дозволить забезпечити високу ефективність використання системи, зменшити час на навчання персоналу та підвищити загальну задоволеність від роботи з платформою.

У межах даного проєкту було створено прототип користувацького інтерфейсу вебсистеми за допомогою інструменту Figma. Макети відображають ключові функціональні екрани, серед яких:

- екран авторизації користувача з можливістю створення профілю;
- головна панель з навігацією між модулями KPI, Projects, Users, Tasks;
- сторінка управління проєктами з візуалізацією статусу задач у вигляді канбан-дошки;
- форма створення нового проєкту;
- таблиця перегляду задач з фільтрацією та кнопкою оновлення;
- модуль оцінювання KPI з можливістю розрахунку показників і збереження результатів у базу даних.

Кольорова палітра та компоновання елементів було підібрано відповідно до принципів UI/UX-дизайну: акцентні елементи виділено, шрифти забезпечують зручність читання, навігаційне меню є фіксованим і доступним з будь-якої сторінки. Прототип дозволяє ітеративно вдосконалювати систему з урахуванням зворотного зв'язку від тестувальників та користувачів.

У подальшому інтерфейс планується реалізувати за допомогою технології React із використанням компонентної бібліотеки (наприклад, Material UI або Bootstrap), що забезпечить масштабованість, інтерактивність і відповідність сучасним стандартам фронтенд-розробки. Зображення макетів наведені в Додатку Д.

3.4. Технічне забезпечення

Технічне забезпечення інформаційної системи управління проєктами є фундаментальним елементом, що забезпечує її повноцінну експлуатацію, стабільну роботу в умовах багатокористувацького доступу, а також здатність до подальшого масштабування та інтеграції з іншими сервісами. У контексті даної системи технічне забезпечення охоплює сукупність обчислювальних, мережевих, дискових і захисних ресурсів, які безпосередньо підтримують роботу серверної частини, клієнтських підключень та бази даних.

На відміну від програмного забезпечення, що відповідає за логіку обробки даних, візуалізацію інформації та взаємодію з користувачем, технічне забезпечення визначає фізичні межі й характеристики функціонування системи. До його завдань

належить обробка запитів у реальному часі, надійне зберігання великого обсягу даних, забезпечення високої швидкості доступу до бази даних, стабільне підключення до мережі, а також комплексна безпека даних - включаючи захист від втрати, несанкціонованого доступу та зовнішніх загроз.

Крім того, технічна інфраструктура повинна забезпечувати високу доступність системи незалежно від кількості активних користувачів, географічного розташування або навантаження на обчислювальні ресурси. Саме тому проєктування й розгортання системи передбачає врахування відповідності технічних рішень реальним умовам експлуатації, рівню критичності даних, обсягам трафіку та можливостям масштабування, як вертикального (розширення потужності одного вузла), так і горизонтального (додавання вузлів до інфраструктури).

Архітектура розробленої системи орієнтована на веб-середовище з розміщенням на сервері загального призначення. Для розгортання серверної частини програмного забезпечення доцільним є використання віртуального або фізичного серверу, що функціонує під управлінням сучасної операційної системи (наприклад, Windows Server або Ubuntu Server). Системні вимоги для стабільної роботи бекенд-сервісів (Node.js + Express.js), СУБД SQL Server та сервера веб-додатку включають:

- **Процесор:** мінімум 4 ядра (рекомендовано Intel i5 або AMD Ryzen 5);
- **Оперативна пам'ять:** 8–16 ГБ (залежно від навантаження);
- **Сховище:** SSD 256–512 ГБ з резервом під бекапи;
- **Мережеве підключення:** стабільне широкосмугове з'єднання з пропускнуою здатністю не менше 100 Мбіт/с;
- **Інструменти захисту:** брандмауер, SSL/TLS-сертифікати для HTTPS-з'єднань, резервне копіювання даних.

Клієнтська частина системи (frontend) доступна через сучасні веб-браузери (Google Chrome, Mozilla Firefox, Microsoft Edge), що не вимагає встановлення

додаткового ПЗ на стороні користувача. Таким чином, мінімізується залежність від апаратних характеристик клієнтських машин: для комфортної роботи достатньо персонального комп'ютера або ноутбука з мінімальними ресурсами ($\text{CPU} \geq 2$ ядра, $\text{RAM} \geq 4$ ГБ, стабільне інтернет-з'єднання).

Особливу увагу під час розробки було приділено забезпеченню відмовостійкості та розширюваності системи. За рахунок модульної архітектури та уніфікованих API, система може бути легко масштабована шляхом розгортання додаткових серверів або переведенням компонентів до хмарної інфраструктури (наприклад, на базі Azure або AWS). Також реалізовано механізми автоматичного резервного копіювання бази даних, які можуть бути інтегровані з хмарними сервісами зберігання (OneDrive, Google Drive, Dropbox тощо).

Таким чином, запропоноване технічне забезпечення системи є достатньо гнучким і універсальним для розгортання як у локальному, так і в хмарному середовищі, що дозволяє максимально адаптувати інфраструктуру до специфіки роботи конкретної організації. У разі локальної інсталяції рішення може бути інтегроване в існуючі серверні потужності підприємства, що знижує витрати на оренду хмарних сервісів та дозволяє зберігати повний контроль над обробкою конфіденційних даних. Такий підхід особливо актуальний для установ із суворими політиками безпеки або обмеженим доступом до інтернету.

Водночас, у разі розгортання у хмарному середовищі, платформа легко масштабується за рахунок автоматичного розподілу ресурсів, високої доступності та вбудованих засобів захисту, які пропонуються провайдерами (AWS, Microsoft Azure, Google Cloud Platform). Хмарне розміщення також спрощує адміністрування, прискорює розгортання нових компонентів та знижує вимоги до власної IT-інфраструктури компанії. Це робить систему привабливою для малих та середніх підприємств, стартапів, а також організацій, що працюють у віддаленому або змішаному форматі.

Завдяки підтримці як традиційної серверної моделі, так і хмарної архітектури, розроблена система демонструє високу адаптивність до умов експлуатації, що дозволяє її ефективно впроваджувати в різних галузях - від ІТ-компаній і освітніх установ до державних та корпоративних організацій. Гнучкість у виборі технічної інфраструктури забезпечує баланс між продуктивністю, вартістю володіння, рівнем доступності та відповідністю стандартам безпеки, що є критично важливими чинниками для сучасних інформаційних систем.

3.4.1 Обґрунтування вибору технічного забезпечення

Вибір технічного забезпечення для інформаційної системи управління проектами здійснювався на основі ряду ключових критеріїв, що визначають ефективність, надійність та економічну доцільність експлуатації системи. Основними критеріями виступали: очікувані обсяги оброблюваних даних, кількість одночасних користувачів, вимоги до безпеки та захисту інформації, а також можливість масштабування інфраструктури в разі зростання навантаження. Особлива увага приділялась вибору серверного обладнання та каналів зв'язку, які повинні забезпечувати високу пропускну здатність і мінімальний час відгуку.

Система орієнтована на роботу в середовищі клієнт-серверної архітектури, що передбачає необхідність у централізованому серверному рішенні з розгортанням бази даних, серверу прикладної логіки та механізмів безпеки (брандмауер, SSL, резервне копіювання). Для забезпечення стабільної роботи запропоновано використання серверу середнього рівня потужності з багатоядерним процесором (щонайменше 8 потоків), оперативною пам'яттю від 32 ГБ та SSD-накопичувачем обсягом не менше 1 ТБ. Таке обладнання гарантує надійне функціонування системи за умови середнього рівня навантаження з перспективою масштабування.

Крім серверної частини, були визначені мінімальні вимоги до клієнтських пристроїв, які мають забезпечити комфортну роботу з інтерфейсом через браузер. Для цього достатньо сучасного ПК або ноутбука з 4-ядерним процесором, 8 ГБ

оперативної пам'яті та підтримкою сучасних веб-браузерів (Chrome, Firefox, Edge). Важливим аспектом є й підтримка адаптивного інтерфейсу, що дозволяє використовувати систему на мобільних пристроях, що розширює її доступність.

Таблиця 3.2.

Специфікація рекомендованого технічного забезпечення (джерело автора)

Компонент середовища	Тип/роль обладнання	Рекомендовані характеристики	Призначення
Сервер бази даних	Локальний або хмарний сервер	CPU: ≥ 4 ядра; RAM: ≥ 16 ГБ; SSD: ≥ 512 ГБ; ОС: Windows Server / Linux (Ubuntu)	Розміщення Microsoft SQL Server, зберігання та обробка даних
Сервер застосунку (Backend)	Веб-сервер або хостинг-сервіс	CPU: ≥ 4 ядра; RAM: ≥ 8 ГБ; ОС: Linux; Node.js $\geq v18$; Express.js	Обробка HTTP-запитів, логіка API, зв'язок з базою даних
Сервер фронтенду	(опціонально) CDN / веб-хостинг	HTML/CSS/JS статичний хостинг або SPA-білд React; інтеграція через Nginx або Apache	Відображення інтерфейсу користувача
Клієнтський пристрій	Робоча станція (ПК, ноутбук)	CPU: ≥ 2 ядра; RAM: ≥ 4 ГБ; Браузер: Chrome, Firefox, Edge (останні версії); Дисплей $\geq 1280 \times 720$	Доступ користувача до веб-інтерфейсу системи
Резервне сховище	Хмарне / мережеве сховище (NAS)	Обсяг: ≥ 1 ТБ; Тип: SSD або HDD; Захист: автоматичне резервне копіювання, шифрування, періодичне архівування	Створення бекапів БД і логів
Мережеве підключення	Ethernet / Інтернет-канал	Пропускна здатність: ≥ 100 Мбіт/с; Надійність: безперервний доступ, UPS, VPN-захист	Стабільна робота клієнт-серверної архітектури

З огляду на вимоги до резервування даних і забезпечення безперервності роботи, передбачається використання хмарних або локальних рішень для створення щоденних резервних копій бази даних та журналів змін. У разі використання моделі хостингу в хмарі (наприклад, AWS, Azure, GCP), також

враховуються параметри розподіленого зберігання та балансування навантаження, що дозволяє уникнути простоїв та втрати даних. Для більшої наглядності була створена Таблиця 3.2, на якій можна ознайомитись з рекомендованим технічним забезпеченням.

ВИСНОВКИ

В умовах сучасних інформаційних технологій управління проєктами стає дедалі складнішим, і традиційні засоби (ручні підходи, таблиці тощо) часто не забезпечують необхідної ефективності та прозорості процесів. Неефективне ручне управління задачами призводить до втрати даних, помилок та суб'єктивних рішень. Водночас відсутність аналітичного компонента в системах управління не дозволяє своєчасно виявляти критичні тренди та ухвалювати обґрунтовані управлінські рішення. За даними сучасних досліджень, аналітика відіграє вирішальну роль у проєктному менеджменті, оскільки забезпечує прозорість ходу робіт та покращує процес ухвалення рішень. Таким чином, актуальність теми обумовлена необхідністю автоматизації процесів управління проєктами та впровадження інструментів аналітики для підвищення продуктивності роботи команди і якості управлінських рішень.

Відповідно, метою роботи було розробити веб-платформу управління проєктами з інтегрованим аналітичним модулем. Для цього було обрано багатоварштову (клієнт-серверну) архітектуру, що розділяє інтерфейс, бізнес-логіку та рівень даних, забезпечуючи незалежний розвиток і масштабування компонентів системи. Вибрані технології відповідають вимогам безпеки та продуктивності і дозволяють ефективно реалізувати функціональні вимоги. Аналітичний модуль реалізовано у вигляді набору звітів і візуалізацій, заснованих на даних із єдиної бази: такий підхід корелюється з рекомендаціями щодо ролі аналітики у сучасному менеджменті, який підкреслює значний вплив аналізу даних на ефективність проєктів. В результаті розроблена система вирішує основні завдання, поєднуючи автоматизацію рутинних операцій і можливість глибокого аналізу даних.

Робота складається з трьох розділів, що послідовно розкривають поставлену проблему. У першому розділі проведено огляд предметної області та аналіз існуючих систем управління проєктами: описано основні бізнес-процеси, сформульовано функціональні вимоги до платформи та виявлено недоліки ручних методів управління, що підтверджують необхідність прозорої аналітики. У другому

розділі обґрунтовано вибір підходів та технологій розробки: проаналізовано можливості клієнтських і серверних веб-фреймворків, обрано СУБД для побудови бази даних, а також розглянуто засоби візуалізації даних для побудови діаграм. Описано обрану архітектуру системи та складовий набір модулів. У третьому розділі викладено процес практичної реалізації системи: розроблено модель «бази знань» для формалізації структурних даних про проєкти, спроектовано та реалізовано реляційну базу даних для зберігання інформації про задачі, ресурси та користувачів, а також спроектовано функціональні модулі (управління задачами й ролями, введення даних, генерація аналітичних звітів та діаграм).

Розроблена система пройшла перевірки функціональності та продуктивності. Було виконано набір контрольних SQL-запитів до бази даних з метою перевірки правильності формування звітних даних: результати запитів підтвердили коректність структури бази даних і можливість отримання необхідних звітів та діаграм. Таким чином, тестування підтвердило працездатність системи та відповідність реалізації заявленим вимогам.

Отже, основними досягненнями роботи є створення проєкта та окремих компонентів веб-платформи, яка забезпечує автоматизоване управління проєктними задачами і містить потужний аналітичний блок. Розроблена платформа автоматизує рутинні процеси планування та контролю, дозволяючи відмовитися від неефективних електронних таблиць і паперових документів, що часто призводять до помилок та дублювання інформації. Єдина БД і уніфікований інтерфейс забезпечать підвищену прозорість роботи команди і раціональне використання ресурсів. Наявність аналітичних звітів та дашбордів надасть можливість у реальному часі відстежувати ключові показники проєкту, що, як підкреслюють дослідники, істотно покращує якість прийняття управлінських рішень і збільшує шанси успіху проєкту. Практична значимість отриманого рішення полягає у готовності платформи до впровадження у реальних організаціях: вона може бути використана для оптимізації співпраці між учасниками проєктів та

для оперативного аналізу їхнього стану, що відповідає сучасним вимогам цифровізації бізнес-процесів.

У перспективі розвиток системи може включати розширення функціональності та адаптацію до нових вимог. Зокрема, доцільним є впровадження прогнозування ключових показників ефективності (KPI) на основі методів машинного навчання і штучного інтелекту, оскільки в сучасному проєктному менеджменті ШІ використовується для автоматизації рутинних завдань та передбачення ризиків. Розробка мобільної версії платформи значно підвищить її доступність та зручність для користувачів, адже мобільні додатки для управління проєктами сприяють покращенню комунікації і оптимізації робочих процесів. Також перспективними є розширення набору ролей користувачів (наприклад, додавання ролей аналітиків і керівників різних рівнів) і інтеграція платформи зі сторонніми сервісами через API – це дозволить синхронізувати дані з іншими інструментами і розширити аналітичні можливості системи. Реалізація цих напрямів подальшого розвитку забезпечить ще вищий рівень автоматизації, гнучкості та масштабованості створеного рішення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Андрушків, Б. М. Інформаційні системи і технології в управлінні організацією: навч. посіб. / Б. М. Андрушків, І. М. Гнатишин. - Львів: ЛНУ, 2020. - 344 с.
2. Anthony, W. P. Human Resource Management Systems: Strategies, Tactics, and Techniques / Anthony, W. P., Perrewew, J. D., Kacmar, K. M. - 2nd ed. - Cincinnati: South-Western, 2002. - 898 p.
3. Баклановський, О. Г. Інформаційні системи і технології в управлінні підприємствами: навч. посіб. / О. Г. Баклановський. - Київ: Центр учбової літератури, 2019. - 280 с.
4. Береза, А. М. Проектування систем оброблення інформації: навч. посіб. / А. М. Береза, І. А. Козак. - К.: КНЕУ, 2008. - 448 с.
5. Бойко, А. І. Системи підтримки прийняття рішень: монографія / А. І. Бойко. - Х.: ХНУРЕ, 2019. - 216 с.
6. Васильєва, Н. В. Інформаційні технології в менеджменті: навч. посіб. / Н. В. Васильєва. - К.: КНЕУ, 2020. - 224 с.
7. Гаврилюк, О. О. Управління проєктами: теорія і практика: навч. посіб. / О. О. Гаврилюк. - Львів: Новий Світ-2000, 2019. - 360 с.
8. ДСТУ ISO 21500:2016. Керівництво з управління проєктами. - [Чинний від 01.01.2017]. - К.: ДП «УкрНДНЦ», 2016. - 32 с.
9. Han, J., Pei, J., & Kamber, M. Data Mining: Concepts and Techniques. - 3rd ed. - Boston: Elsevier, 2011. - 744 p.
10. ISO/IEC 25010:2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models.
11. Kavanagh, M. J. HR Information Systems: A Practical Approach / Kavanagh, M. J., Johnson, R. D. - New York: McGraw-Hill, 2012. - С. 3–27.

12. Kavanagh, M. J. Human Resource Information Systems: Basics, Applications, and Future Directions / Kavanagh, M. J., Thite, M. - 3rd ed. - Thousand Oaks: SAGE Publications, 2017. - 592 p.
13. Котляр, В. І. Розробка інформаційних систем: проектування, реалізація, впровадження: навч. посіб. / В. І. Котляр. - К.: Кондор, 2018. - 296 с.
14. Microsoft SQL Server – Documentation [Електронний ресурс]. - Режим доступу: <https://learn.microsoft.com/sql/> , вільний. - Дата звернення: 17.05.2025.
15. Моделі і методи проектування інформаційних систем [Електронний ресурс]. - Режим доступу: https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20160217112601/260038/index.html , вільний. - Дата звернення: 17.05.2025.
16. Мотуз, В. Г. Інформаційні технології та системи управління: навч. посіб. / В. Г. Мотуз, М. В. Руденко. - К.: КНЕУ, 2021. - 378 с.
17. Node.js – Documentation [Електронний ресурс]. - Режим доступу: <https://nodejs.org/en/docs/> , вільний. - Дата звернення: 17.05.2025.
18. Пономаренко, В. С. Інформаційні системи в управлінні персоналом: навч. посіб. / В. С. Пономаренко, І. В. Журавльова, І. Л. Латишева. - Харків: ХНЕУ, 2008. - 336 с.
19. Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK Guide). - 7th ed. - USA: PMI, 2021. - 370 p.
20. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ СИСТЕМ [Електронний ресурс]. - Режим доступу: https://pidru4niki.com/12800528/bankivska_sprava/programne_zabezpechennya_kompyuternih_informatsiynih_sistem , вільний. - Дата звернення: 17.05.2025.
21. Пушкар, М. С. Проектування систем автоматизації: навч. посіб. / М. С. Пушкар, С. М. Проценко. - 2013. - [Електронний ресурс]. - Режим доступу:

- <http://ir.nmu.org.ua/bitstream/handle/123456789/3536/CD218.pdf> , вільний. - Дата звернення: 17.05.2025.
22. React – Official Documentation [Електронний ресурс]. - Режим доступу: <https://reactjs.org> , вільний. - Дата звернення: 17.05.2025.
23. REST API Design Guidelines [Електронний ресурс]. - Режим доступу: <https://restfulapi.net/> , вільний. - Дата звернення: 17.05.2025.
24. Saunders, C. S. Managing Human Resources in the Software Industry / C. S. Saunders. - New York: Routledge, 2016. - 336 p.
25. Ситник, Н. В. Проектування баз і сховищ даних: навч. посіб. / Н. В. Ситник. - К.: КНЕУ, 2004. - 348 с.
26. Ситник, Н. В. Проектування баз і сховищ даних: навч.-метод. посіб. для самост. вивч. дисц. / Н. В. Ситник, М. Т. Краснюк. - К.: КНЕУ, 2005. - 264 с.
27. Trello – Project Management Tool [Електронний ресурс]. - Режим доступу: <https://trello.com> , вільний. - Дата звернення: 17.05.2025.
28. Asana – Official Site [Електронний ресурс]. - Режим доступу: <https://asana.com> , вільний. - Дата звернення: 17.05.2025.
29. Jira Software – Atlassian [Електронний ресурс]. - Режим доступу: <https://www.atlassian.com/software/jira> , вільний. - Дата звернення: 17.05.2025.
30. ClickUp – Project Management Platform [Електронний ресурс]. - Режим доступу: <https://clickup.com> , вільний. - Дата звернення: 17.05.2025.
31. Zoho Projects – Project Management Software [Електронний ресурс]. - Режим доступу: <https://www.zoho.com/projects/> , вільний. - Дата звернення: 17.05.2025.
32. Microsoft Project – Product Page [Електронний ресурс]. - Режим доступу: <https://www.microsoft.com/en-us/microsoft-365/project> , вільний. - Дата звернення: 17.05.2025.

33. Worksection – Українська система управління проєктами [Електронний ресурс]. - Режим доступу: <https://worksection.com/ua/> , вільний. - Дата звернення: 17.05.2025.
34. Basecamp – Project Management & Team Communication [Електронний ресурс]. - Режим доступу: <https://basecamp.com> , вільний. - Дата звернення: 17.05.2025.
35. Krug, S. Don't Make Me Think: A Common Sense Approach to Web Usability. – 3rd ed. – San Francisco: New Riders, 2014. – 216 p.
36. Norman, D. The Design of Everyday Things. – Revised and Expanded Edition. – New York: Basic Books, 2013. – 368 p.
37. Lidwell, W., Holden, K., Butler, J. Universal Principles of Design. – 2nd ed. – Beverly: Rockport Publishers, 2010. – 272 p.
38. Garrett, J. J. The Elements of User Experience: User-Centered Design for the Web and Beyond. – 2nd ed. – Berkeley: New Riders, 2010. – 208 p.
39. Kalbach, J. Mapping Experiences: A Complete Guide to Customer Alignment Through Journeys, Blueprints, and Diagrams. – 2nd ed. – O'Reilly Media, 2020. – 432 p.
40. Interaction Design Foundation. What is User Experience (UX) Design? [Електронний ресурс]. – Режим доступу: <https://www.interaction-design.org/literature/topics/ux-design> , вільний. – Дата звернення: 19.05.2025.
41. Nielsen, J. Usability Engineering. – Cambridge: Academic Press, 1994. – 362 p.
42. UXPin. The Guide to Prototyping [Електронний ресурс]. – Режим доступу: <https://www.uxpin.com/studio/ebooks/guide-to-prototyping/>, вільний. – Дата звернення: 17.05.2025.
43. Figma – Collaborative Interface Design Tool [Електронний ресурс]. – Режим доступу: <https://www.figma.com/>, вільний. – Дата звернення: 19.05.2025.
44. Material Design Guidelines – Google [Електронний ресурс]. – Режим доступу: <https://m3.material.io/> , вільний. – Дата звернення: 19.05.2025.

45. Bootstrap – The most popular HTML, CSS, and JS library [Електронний ресурс]. – Режим доступу: <https://getbootstrap.com/>, вільний. – Дата звернення: 19.05.2025.
46. UX Design Patterns – UI Design Inspiration [Електронний ресурс]. – Режим доступу: <https://uimovement.com/>, вільний. – Дата звернення: 19.05.2025.
47. Friedman, V. Smashing UX Design: Foundations for Designing Online User Experiences. – Chichester: Wiley, 2012. – 400 p.
48. Wroblewski, L. Mobile First. – New York: A Book Apart, 2011. – 132 p.
49. Cederholm, D. Responsive Web Design. – A Book Apart, 2011. – 150 p.
50. Shneiderman, B., Plaisant, C., Cohen, M. et al. Designing the User Interface: Strategies for Effective Human-Computer Interaction. – 6th ed. – Boston: Pearson, 2016. – 672 p.

ДОДАТКИ

Додаток А

	id	name	email	position
▶	1	Іван Петренко	ivan.petrenko@example.com	Менеджер проєктів
	2	Олена Коваль	olena.koval@example.com	Frontend-розробник
	3	Андрій Сидоренко	andriy.syd@example.com	Backend-розробник
	4	Марія Бондар	mariya.bondar@example.com	Аналітик
	5	Юрій Павленко	yurii.pavlenko@example.com	DevOps-інженер
	6	Світлана Литвин	svitlana.l@example.com	Tester
	7	Павло Романенко	pavlo.rom@example.com	UI/UX дизайнер
	8	Наталія Снігур	natalia.sn@example.com	Business-аналітик
	9	Віктор Чумак	victor.ch@example.com	Fullstack Developer
	10	Ігор Нікітін	ihor.nikitin@example.com	Project Manager
	11	Оксана Ткаченко	oksana.tk@example.com	QA Engineer
	12	Роман Демченко	roman.dem@example.com	Data Analyst

Рис. Д.1. Таблиця «Працівники» (джерело автора)

	id	name	description	priority	user_id	start_date	end_date	status
▶	1	CRM-система	Клієнтська CRM-система для малого бізнесу	1	1	2024-11-01	2025-01-15	активний
	2	Навчальна платформа	Система керування освітніми курсами	2	10	2024-10-01	2025-02-28	активний
	3	Портал підтримки	Інтерфейс служби техпідтримки	2	1	2024-09-15	2024-12-01	виконано
	4	Бюджетний трекер	Web-додаток для фінансового планування	1	10	2024-08-01	2024-11-15	активний
	5	Інвентаризація складу	Система моніторингу товарів	3	1	2024-06-10	2024-09-30	виконано
	6	Хмарний сервіс файлів	Прототип файлового сховища	1	1	2024-07-01	2024-10-15	архівований

Рис. Д.2. Таблиця «Проекти» (джерело автора)

	id	name	description	priority	user_id	start_date	end_date	status	project_id	notes
▶	1	Дизайн UI	Створити макети	2	2	2024-11-05	2024-11-15	виконано	1	Готово
	2	API логін	Бекенд логіки входу	1	3	2024-11-10	2024-11-25	виконано	1	ОК
	3	Аналіз вим...	Специфікація функцій	1	4	2024-10-15	2024-10-30	виконано	2	Файл надіслано
	4	Форма реє...	Frontend для signur	2	2	2024-11-20	2024-12-01	виконано	1	UX-оцінка позитивна
	5	Тестування...	Unit-тестування бекенду	1	6	2024-11-25	2024-12-05	виконано	1	95% покриття
	6	Форма фід...	Контактна форма	3	2	2024-12-01	2024-12-10	виконано	2	Завершено
	7	Налаштува...	GitHub Actions	2	5	2024-10-01	2024-10-05	виконано	3	Автоматизовано
	8	Прототип г...	UI шаблон	2	7	2024-09-20	2024-09-25	виконано	3	UX схвалено
	9	Визначенн...	Показники ефективності	1	8	2024-08-01	2024-08-05	виконано	4	Уточнено з керівництвом
	10	Звітність	Генерація PDF-звітів	2	11	2024-08-10	2024-08-20	виконано	4	Експорт налаштовано
	11	База товар...	Каталог продукції	1	3	2024-06-15	2024-07-01	виконано	5	Повна база
	12	Сторінка ав...	React компонент	1	2	2024-07-05	2024-07-15	виконано	6	Завершено
	13	Інтеграція	Підключення до бекенду	2	9	2024-07-10	2024-07-20	виконано	6	Налагоджено
	14	Міграція да...	Збереження у SQL Server	2	3	2024-07-20	2024-07-30	виконано	6	Проблем не виявлено
	15	Оцінка про...	Порівняння метрик	1	12	2024-10-01	2024-10-10	виконано	2	Методика погоджена
	16	Побудова г...	D3.js інтеграція	3	7	2024-10-10	2024-10-15	виконано	2	Інтерактивні візуалізації
	17	Моніторин...	Збір логів	2	5	2024-11-01	2024-11-05	виконано	4	Навантаження в нормі
	18	Тестування...	Postman сценарії	2	6	2024-11-06	2024-11-12	виконано	4	Всі пройдено

Рис. Д.3. Таблиця «Задачі» (джерело автора)

	id	indicator	value	data	task_id
►	1	Час виконання (дн)	10	2024-11-15	1
	2	Якість коду	90	2024-11-25	2
	3	Точність аналізу	95	2024-10-30	3
	4	UX-оцінка	85	2024-12-01	4
	5	Кодове покриття (%)	95	2024-12-05	5
	6	Терміновість	100	2024-12-10	6
	7	CI/CD впровадження	88	2024-10-05	7
	8	Задоволеність користувача	92	2024-09-25	8
	9	Чіткість KPI	93	2024-08-05	9
	10	Наявність звітів	87	2024-08-20	10
	11	Повнота бази	98	2024-07-01	11
	12	Якість front-end	89	2024-07-15	12
	13	API стабільність	91	2024-07-20	13
	14	Успішність міграції	96	2024-07-30	14
	15	Оцінка ефективності	94	2024-10-10	15
	16	Інтерактивність графіків	86	2024-10-15	16
	17	Стабільність моніторингу	90	2024-11-05	17
	18	Якість автотестів	88	2024-11-12	18

Рис. Д.4. Таблиця «Ключові показники ефективності» (джерело автора)

	id	task_id	planned_d...	estimation_hours	complexity	satisfaction	rework_cou...	delay_penalty
►	10	1	2024-11-09	27,00	2	9	0	18,00
	11	2	2024-11-15	22,00	2	10	0	42,00
	12	3	2024-10-19	16,00	2	7	1	41,00
	13	4	2024-11-23	26,00	3	10	2	37,00
	14	5	2024-11-29	39,00	2	9	2	50,00
	15	6	2024-12-06	37,00	2	8	0	46,00
	16	7	2024-10-04	26,00	1	9	1	0,00
	17	8	2024-09-24	20,00	3	7	2	0,00
	18	9	2024-08-08	37,00	2	7	2	0,00
	19	10	2024-08-14	27,00	3	9	2	42,00
	20	11	2024-06-22	33,00	2	8	1	23,00
	21	12	2024-07-11	14,00	3	7	1	29,00
	22	13	2024-07-13	27,00	3	10	2	16,00
	23	14	2024-07-23	17,00	2	7	0	44,00
	24	15	2024-10-07	21,00	2	8	1	10,00
	25	16	2024-10-15	24,00	1	7	1	0,00
	26	17	2024-11-04	34,00	2	9	0	0,00
	27	18	2024-11-11	36,00	2	10	0	0,00
★	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рис. Д.5. Таблиця «Метрики задачі» (джерело автора)

Додаток Б

	user_id	user_name	ontime_rate	delay_score	satisfaction_score	rework_score	penalty_score	value
1	2	Олена Коваль	0.000000	0.500000	0.850000	0.850000	0.967500	0.633500
2	3	Андрій Сидоренко	0.000000	0.200000	0.833333	0.933334	0.963667	0.586067
3	4	Марія Бондар	0.000000	0.000000	0.700000	0.800000	0.959000	0.491800
4	5	Юрій Павленко	0.000000	0.900000	0.900000	0.900000	1.000000	0.740000
5	6	Світлана Литвин	0.000000	0.700000	0.950000	0.800000	0.975000	0.685000
6	7	Павло Романенко	0.500000	1.000000	0.700000	0.700000	1.000000	0.780000
7	8	Наталія Снігур	1.000000	1.000000	0.700000	0.600000	1.000000	0.860000
8	9	Віктор Чумак	0.000000	0.300000	1.000000	0.600000	0.984000	0.576800
9	11	Оксана Ткаченко	0.000000	0.400000	0.900000	0.600000	0.958000	0.571600
10	12	Роман Демченко	0.000000	0.700000	0.800000	0.800000	0.990000	0.658000

Рис. Д.6. Представлення «Розрахунок КРІ» (джерело автора)

Додаток В

	user_id	user_name	total_tasks	avg_completion_days	percent_on_time	high_priority_tasks
1	2	Олена Коваль	4	10	100.00000000000000	1
2	3	Андрій Сидоренко	3	13	100.00000000000000	2
3	4	Марія Бондар	1	15	100.00000000000000	1
4	5	Юрій Павленко	2	4	100.00000000000000	0
5	6	Світлана Литвин	2	8	100.00000000000000	1
6	7	Павло Романенко	2	5	100.00000000000000	0
7	8	Наталія Снігур	1	4	100.00000000000000	1
8	9	Віктор Чумак	1	10	100.00000000000000	0
9	11	Оксана Ткаченко	1	10	100.00000000000000	0
10	12	Роман Демченко	1	9	100.00000000000000	1

Рис. Д.7. Представлення «Продуктивність кожного користувача» (джерело автора)

Додаток Г

	task_id	task_name	assigned_user	total_kpi_records	avg_kpi_value	last_kpi_date	status	priority
1	1	Дизайн UI	Олена Коваль	1	10	2024-11-15	виконано	2
2	2	API логін	Андрій Сидоренко	1	90	2024-11-25	виконано	1
3	3	Аналіз вимог	Марія Бондар	1	95	2024-10-30	виконано	1
4	4	Форма реєстрації	Олена Коваль	1	85	2024-12-01	виконано	2
5	5	Тестування API	Світлана Литвин	1	95	2024-12-05	виконано	1
6	6	Форма фідбеку	Олена Коваль	1	100	2024-12-10	виконано	3
7	7	Налаштування CI/CD	Юрій Павленко	1	88	2024-10-05	виконано	2
8	8	Прототип головної	Павло Романенко	1	92	2024-09-25	виконано	2
9	9	Визначення KPI	Наталія Снігур	1	93	2024-08-05	виконано	1
10	10	Звітність	Оксана Ткаченко	1	87	2024-08-20	виконано	2
11	11	База товарів	Андрій Сидоренко	1	98	2024-07-01	виконано	1
12	12	Сторінка авторизації	Олена Коваль	1	89	2024-07-15	виконано	1
13	13	Інтеграція API	Віктор Чумак	1	91	2024-07-20	виконано	2
14	14	Міграція даних	Андрій Сидоренко	1	96	2024-07-30	виконано	2
15	15	Оцінка продуктивності	Роман Демченко	1	94	2024-10-10	виконано	1
16	16	Побудова графіків	Павло Романенко	1	86	2024-10-15	виконано	3
17	17	Моніторинг серверів	Юрій Павленко	1	90	2024-11-05	виконано	2
18	18	Тестування API v2	Світлана Литвин	1	88	2024-11-12	виконано	2

Рис. Д.8. Представлення «Якість виконання задач за показниками KPI» (джерело автора)

Додаток Д

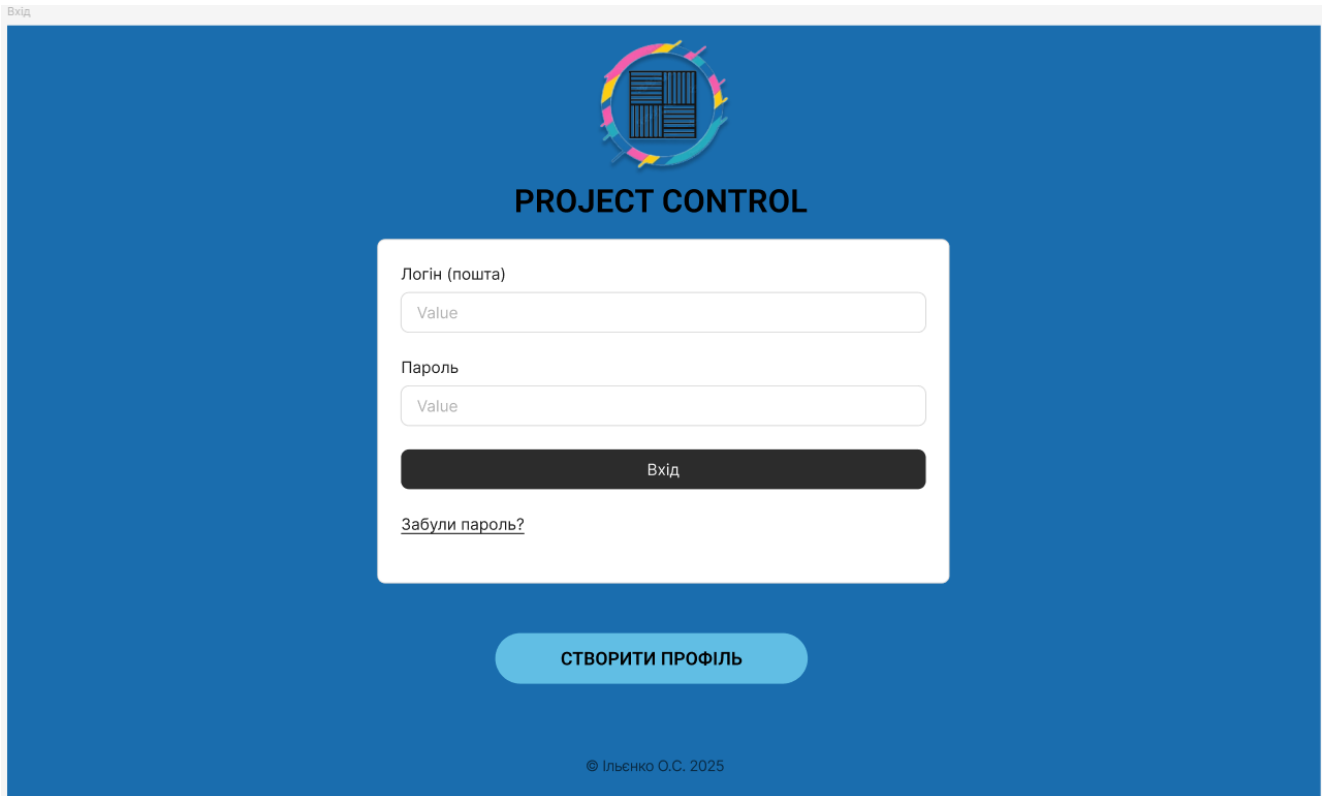


Рис. Д.9. Екран «Авторизація» (джерело автора)

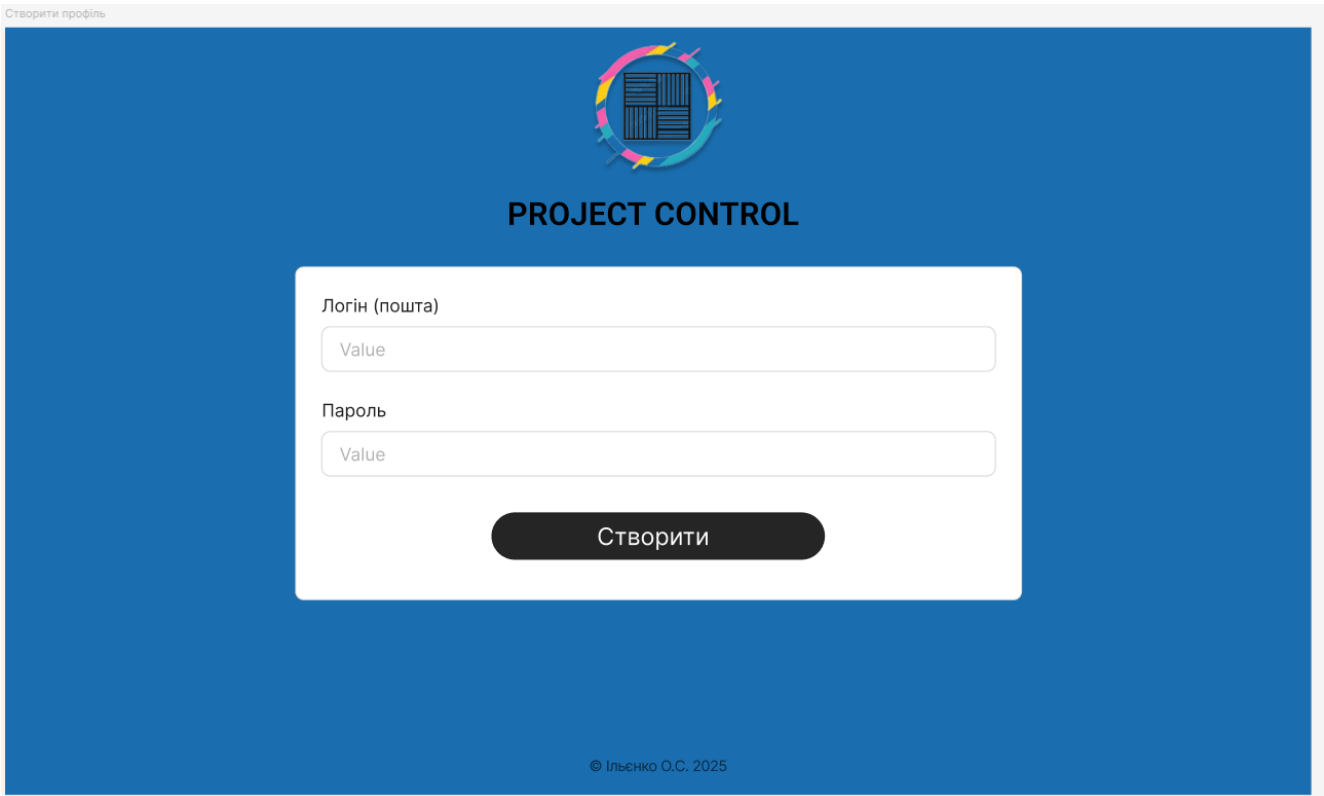


Рис. Д.10. Екран «Реєстрація» (джерело автора)

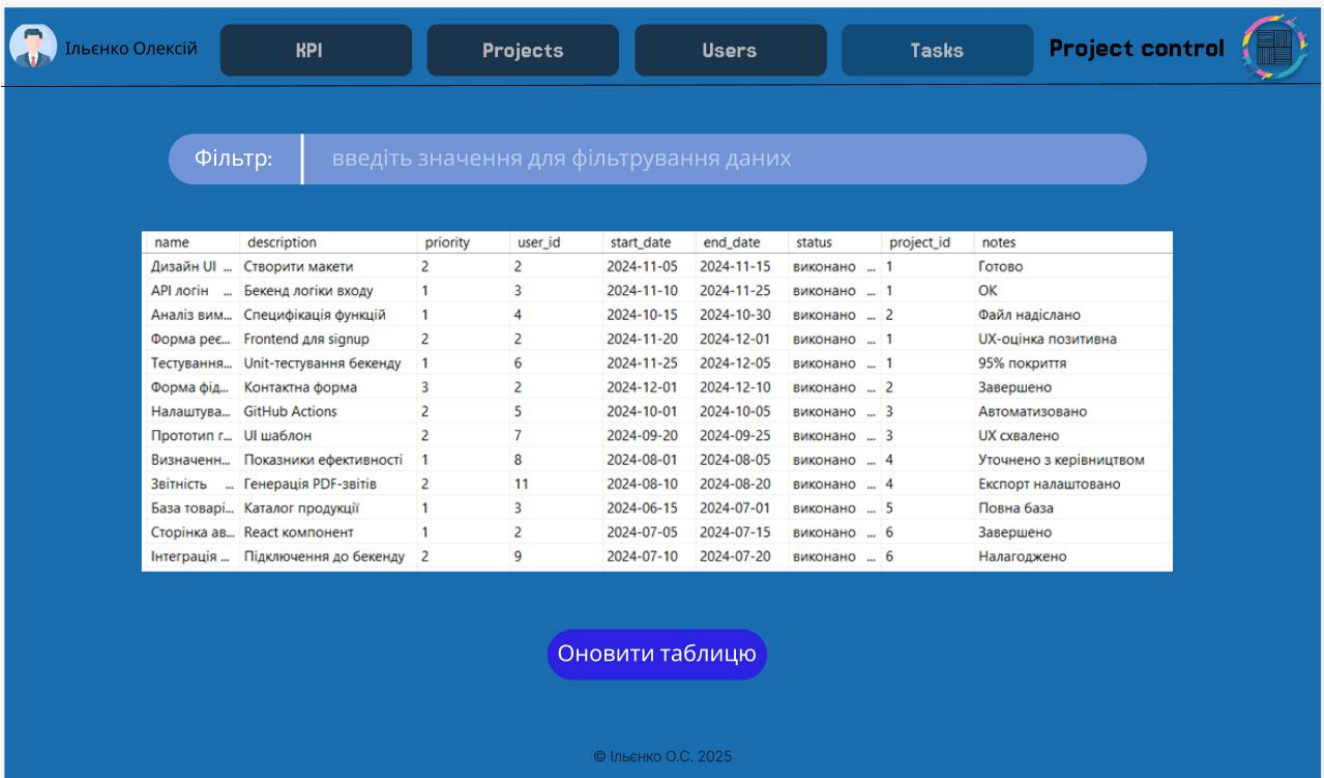


Рис. Д.11. Екран «Задачі» (джерело автора)

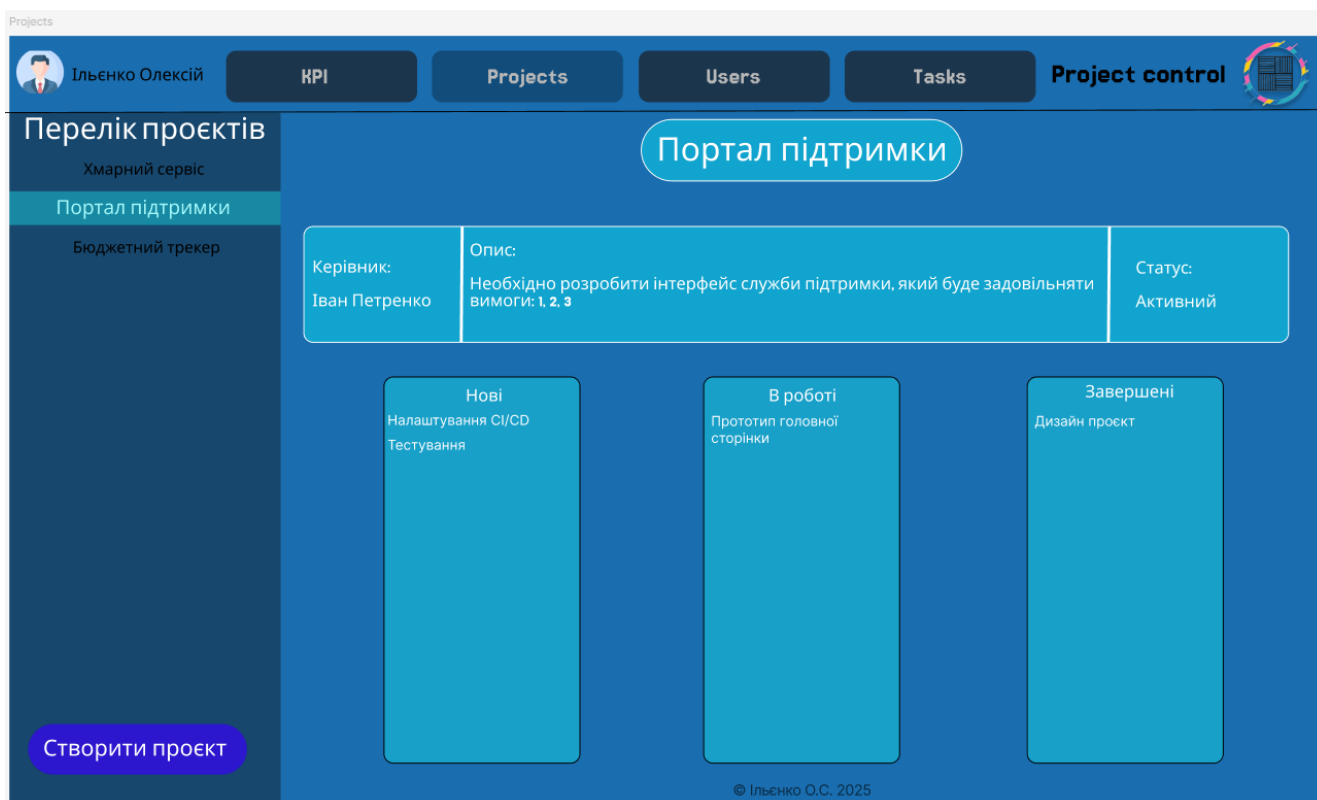


Рис. Д.12. Екран «Проекти та їх наповнення» (джерело автора)

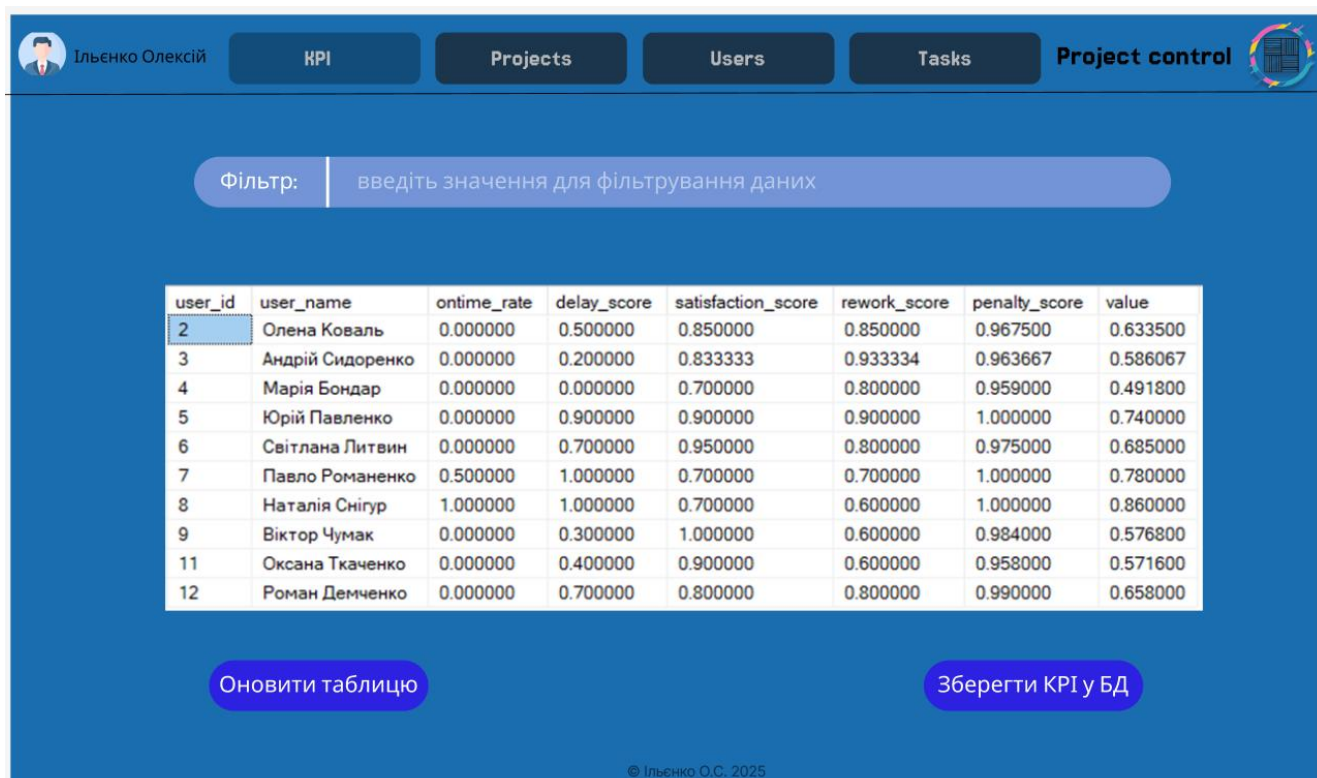


Рис. Д.13. Екран «Ключові показники ефективності» (джерело автора)

Створення проєкту

Ільєнко Олексій KPI Projects Users Tasks Project control

Створити проєкт

Назва проєкту

Керівник

Опис проєкта

Дата почтаку

Дата закінчення

Пріоритет

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Зберегти

© Ільєнко О.С. 2025

Рис. Д.17. Екран «Створення проєкту» (джерело автора)

Дата звіту

5/19/2025

Дата редагування

Звіт не був оцінений

Звіт подібності

метадані

Назва організації

Kyiv National Economic University named after Vadym Hetman KNEU

Заголовок

Розроблення веб платформи для управління проектами та аналізу ефективності їх виконання

Автор

Науковий керівник / Експерт

Ільєнко Олексій СергійовичСитник Н.В.

підрозділ

кафедра інформаційних систем в економіці

Обсяг знайдених подібностей

2.74%

2.74%

КП 1

25

Довжина фрази для коефіцієнта подібності 2

0.73%

0.73%

КП 2

12736

Кількість слів

0.60%

0.60%

КЦ

103600

Кількість символів

Тривога

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

Заміна букв		1
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)		18

Подібності за списком джерел

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

10 найдовших фраз

Копір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розроблення інформаційної управляючої системи для оптимізації діяльності цифрового інноваційного хабу 5/16/2025 Kyiv National Economic University named after Vadym Hetman KNEU (кафедра інформаційних систем в економіці)	40 0.31 %
2	https://ir.kneu.edu.ua/bitstreams/6ccdb206-8499-47f8-b15b-5d42b9b14ce2/download	28 0.22 %
3	https://ir.kneu.edu.ua/bitstreams/6ccdb206-8499-47f8-b15b-5d42b9b14ce2/download	25 0.20 %

4	Розроблення інформаційної управляючої системи для оптимізації діяльності цифрового інноваційного хабу 5/16/2025 Kyiv National Economic University named after Vadym Hetman KNEU (кафедра інформаційних систем в економіці)	20 0.16 %
5	Розроблення інформаційної управляючої системи для оптимізації діяльності цифрового інноваційного хабу 5/16/2025 Kyiv National Economic University named after Vadym Hetman KNEU (кафедра інформаційних систем в економіці)	17 0.13 %
6	https://ir.kneu.edu.ua/bitstreams/6ccdb206-8499-47f8-b15b-5d42b9b14ce2/download	14 0.11 %
7	Розроблення інформаційної управляючої системи для оптимізації діяльності цифрового інноваційного хабу 5/16/2025 Kyiv National Economic University named after Vadym Hetman KNEU (кафедра інформаційних систем в економіці)	14 0.11 %
8	Маруняк_стаття 10/14/2024 Publishing House "Helvetica" (Видавничий дім "Гельветика")	12 0.09 %
9	https://ir.kneu.edu.ua/bitstreams/40f69492-bf2d-4dcb-b3e1-7da1833a5b34/download	12 0.09 %
10	КОМБІНОВАНИЙ ІНСТРУМЕНТАРІЙ ОРГАНІЗАЦІЙНО-ТЕХНОЛОГІЧНОГО ТА ЦИФРОВОГО АДМІНІСТРУВАННЯ ПРОЄКТАМИ БУДІВНИЦТВА 6/8/2024 Kyiv National University of Construction and Architecture (Kyiv National University of Construction and Architecture)	11 0.09 %

з бази даних RefBooks (0.08 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
джерело: https://arxiv.org/		
1	Uncertainty Annotated Databases - A Lightweight Approach for Approximating Certain Answers (extended version) Aaron Huber,Boris Glavic,Su Feng,Oliver Kennedy;	10 (1) 0.08 %

з домашньої бази даних (1.18 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розроблення інформаційної управляючої системи для оптимізації діяльності цифрового інноваційного хабу 5/16/2025 Kyiv National Economic University named after Vadym Hetman KNEU (кафедра інформаційних систем в економіці)	150 (12) 1.18 %

з програми обміну базами даних (0.65 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	123-Бз-2025-Фашевський ОО 2/25/2025 Ukrainian national aviation university (ФКНТ Кафедра інтелектуальних кібернетичних систем)	18 (2) 0.14 %

2	КОМБІНОВАНИЙ ІНСТРУМЕНТАРІЙ ОРГАНІЗАЦІЙНО-ТЕХНОЛОГІЧНОГО ТА ЦИФРОВОГО АДМІНІСТРУВАННЯ ПРОЄКТАМИ БУДІВНИЦТВА 6/8/2024 Kyiv National University of Construction and Architecture (Kyiv National University of Construction and Architecture)	16 (2) 0.13 %
3	ФККПІ_2020_121_ОриничЛВ 7/11/2024 Ukrainian national aviation university (Ukrainian national aviation university)	15 (2) 0.12 %
4	ВЕБСЕРВІС ОПТИМІЗАЦІЇ РОБОТИ ІНТЕРНЕТ-МАГАЗИНУ ЕЛЕКТРОНІКИ 6/23/2024 Rivne Professional college of National University of Life and Environmental sciences of Ukraine (ВСП „Рівненський фаховий коледж НУБіП України“)	12 (2) 0.09 %
5	Маруняк_стаття 10/14/2024 Publishing House "Helvetica" (Видавничий дім "Гельветика")	12 (1) 0.09 %
6	2025_Дудла_В.В._21МБКН_1 2/19/2025 Tavria State Agrotechnological University (Кафедра комп'ютерних наук)	10 (1) 0.08 %
з Інтернету (0.83 %)		
ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://ir.kneu.edu.ua/bitstreams/6ccdb206-8499-47f8-b15b-5d42b9b14ce2/download	89 (6) 0.70 %
2	https://ir.kneu.edu.ua/bitstreams/40f69492-bf2d-4dcb-b3e1-7da1833a5b34/download	12 (1) 0.09 %
3	https://ua-referat.com/uploaded/kvalifikacija-bakalavrseka-robota/index5.html	5 (1) 0.04 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)
------------------	-------	---------------------------------------

Тези на тему- Методологія проектування баз даних

Ільєнко Олексій Сергійович

здобувач вищої освіти Інституту інформаційних систем в економіці

Київський національний економічний університет імені Вадима Гетьмана,

Україна

МЕТОДОЛОГІЯ ПРОЕКТУВАННЯ БАЗ ДАНИХ

В процесі розробки бази даних як складової частини інформаційної системи розробникам необхідно пройти ряд етапів, багато з яких описані в попередніх розділах і будуть розглядатися далі. Ці етапи складають методологію розробки бази даних, формуючи максимально повний опис технологій подання, зберігання і обробки даних комп'ютерними системами.

Для розробки баз даних також існує методологія проектування, яка зачіпає ряд етапів, від аналізу предметної області та документів до опису процедур обробки та подання даних. Розробка бази даних ґрунтується на попередньому аналізі предметної області, на виділених основних і допоміжних функціях, а також описі оточення кожної функції. Використання цієї інформації дає розробнику обґрунтування використання при моделюванні необхідних сутностей та подання в них потрібних атрибутів, організовуючи між сутностями правильні зв'язку. [1]

Методи, виділені в розробці бази даних: класичний (від документів) і функціонально-об'єктний (від функцій і об'єктів), - реалізують єдину методологію проектування бази даних, визначаючи, в залежності від обраного підходу, ключовий елемент аналізу - документ або об'єкт відповідно.

Розвиток класичного підходу при проектуванні реляційної бази даних ґрунтується на тому факті, що основним елементом роботи користувача в предметній області є документ, що містить необхідну інформацію, яку доцільно зберігати в базі даних з наступним поданням в електронних і паперових документах. Цей підхід практично ніяк не враховує особливості роботи з даними в рамках окремих функцій користувача інформаційної системи, для якої розробляється база даних. Розробники згадували про функціональну корисність бази даних тільки в момент розробки процедур обробки даних і реалізації програмного додатка.

Основу функціонально-об'єктного підходу до розробки бази даних становить усвідомлення потреб користувачів системи у виконанні окремих функціональних завдань, які, як з цеглинок, складають комплексну функціональну середу діяльності користувача. Цей підхід, виділяючи окремі функції предметної області і розділяючи їх на основні та допоміжні, дає можливість розробнику підходити до процесу проектування, розглядаючи функції у вигляді самостійних блоків (чорний ящик) і об'єднуючи їх в "будова", що називається "інформаційна система".[2]

Переходячи до процесу моделювання бази даних, розробник повинен розглянути функціональні структури даних, які фактично представляють собою моделі бази даних, орієнтовані на роботу однієї конкретної функції предметної області. Природно, що немає сенсу створювати для єдиної інформаційної системи безліч окремих моделей баз даних під кожну функцію, якщо це не передбачено особливими вимогами до реалізації інформаційної системи.

Подальший процес моделювання даних передбачає використання об'єктів предметної області в якості елементів для створення сутностей, а атрибути, що описують виділені об'єкти, є атрибутами відповідних сутностей. В результаті процесу моделювання структур даних для функцій розробник отримує модель бази даних, яку можна надалі реалізовувати в СУБД.

В даному процесі, на відміну від класичного підходу, основу складає ключова сутність, яка відповідає основному об'єкту обраної функції предметної області, інформацію про який необхідно зберігати в базі даних. В результаті розгляду атрибутивного складу ключової сутності

можуть утворитися об'єктні атрибути, які в предметній області представляються самостійними об'єктами. Такі атрибути повинні в моделі бази даних представлятися у вигляді окремих сутностей, пов'язаних із сутністю, з якої вони виділені. Цей процес аналізу атрибутів і виділення нових сутностей триває до тих пір, поки у знову виділяються сутності не залишаться тільки прості атрибути.

Однак, незалежно від обраного підходу до розробки бази даних, реалізації моделей баз даних буде ідентичною, і всі відмінності підходів представляються особливостями визначення елементів моделей баз даних та правилами в процесах створення таких моделей. Так, при класичному підході розробник спочатку знає про всі сутності, які повинні з'явитися в моделі бази даних, і застосовує методи нормалізації, щоб отримати ефективну структуру бази даних. У функціонально-об'єктних підходах розробник володіє знаннями про функції, що реалізуються в предметній області, і об'єктах, які є основними (ключовими) для кожної функції. Застосовуючи методики аналізу об'єктів, він формує необхідні сутності та моделі баз даних для кожної окремої функції, об'єднуючи їх в рамках єдиної моделі бази даних предметної області.

Висновок:

Якщо розглянути ці два підходи, то неважко помітити, що процеси розробки бази даних практично не відрізняються один від одного. Це дозволяє говорити, що методологія розробки майже не змінилася при зміні підходу і, після аналізу предметної області, коли розробник підходить до розгляду структур даних, етапність методології проектування вимагає аналізу моделей даних, визначення зв'язків і застосування відповідних нотацій для представлення структур даних.

Нотацій для представлення моделей бази даних досить велика кількість, і спрямовані вони на уявлення структур даних, орієнтоване на подальше використання в СУБД і відповідних реалізаціях інформаційних систем. Особливо важливо правильно вибрати нотацію моделювання бази даних, коли мова йде про вибір підходу до реалізації програмного додатка: процедурне (транзакційних) або об'єктно-орієнтоване додаток. Особливих відмінностей у підходах моделювання тут не спостерігається, але уявлення моделей баз даних буде різним, з огляду на необхідність зв'язування елементів і структур даних з об'єктними елементами програмного додатка.

Список використаних джерел

1. Проектування баз і сховищ даних [Електронний ресурс] : навч.-метод. посіб. для самост. вивч. дисц. / Н. В. Ситник, М. Т. Краснюк ; М-во освіти і науки України, Київ. нац. екон. ун-т. - Електрон. текстові дані. - К. : КНЕУ, 2005. - 264 с.
2. Проектування баз і сховищ даних [Електронний ресурс] : навч. посіб. / Н. В. Ситник ; М-во освіти і науки України, Київ. нац. екон. ун-т. - Електрон. текстові дані. - К. : КНЕУ, 2004. - 348 с.

Науковий Керівник: Устенко Станіслав Веніамінович

доктор екон. наук, професор, професор Інституту інф. сист. і в економіці

Київський національний економічний університет імені Вадима Гетьмана,

Україна

Тези на тему- Особливості застосування тригерів у базах даних

Ільєнко Олексій Сергійович

здобувач вищої освіти Інституту інформаційних систем в економіці

Київський національний економічний університет імені Вадима Гетьмана,

Україна

ОСОБЛИВОСТІ ЗАСТОСУВАННЯ ТРИГЕРІВ У БАЗАХ ДАНИХ

У процесі функціонування бази даних важливу роль відіграють механізми, які дозволяють автоматизувати обробку даних у відповідь на певні події. Одним із таких ключових механізмів є тригери. Тригери забезпечують автоматичне виконання визначених дій у момент внесення змін до таблиць бази даних, що дозволяє реалізувати логіку, яка спрацьовує без втручання користувача або прикладної програми. Завдяки тригерам стає можливим централізоване забезпечення цілісності даних, контроль за дотриманням бізнес-правил та реалізація механізмів історизації або логування.

Тригери створюються на основі подій, які ініціюють їхнє спрацьовування. Такими подіями є операції вставки (INSERT), оновлення (UPDATE) або видалення (DELETE) записів. Крім того, тригери поділяються на два основних типи за часом виконання: ті, що спрацьовують до виконання дії (BEFORE), і ті, що виконуються після (AFTER). Це дає можливість не лише реалізувати додаткову перевірку даних до того, як зміни будуть внесені в таблицю, а й виконати допоміжні дії після завершення основної операції, наприклад, оновити пов'язані записи або записати зміну в окрему таблицю аудиту.[1]

У тригерах розробник має доступ до спеціальних змінних, які дозволяють оперувати як старими значеннями (ключове слово OLD), так і новими (NEW). Завдяки цьому відкривається широкий простір для реалізації складної логіки - від перевірки умов цілісності до реалізації каскадних оновлень або блокування змін у залежності від ролі користувача. Наприклад, у фінансовій системі можна реалізувати тригер, який забороняє зміну критичних полів транзакції після її затвердження, або автоматично створює запис у журналі змін при оновленні балансу рахунку.

Важливо відзначити, що тригери можуть бути на рівні рядків (FOR EACH ROW) або на рівні всієї таблиці. У першому випадку вони виконуються для кожного окремого запису, який піддається зміні, що зручно при необхідності деталізованої обробки. У другому - тригер виконується один раз на всю операцію, що може бути ефективнішим для агрегованої обробки або перевірки.[2]

Однак при всіх перевагах використання тригерів слід дотримуватися обережності. Їхня надмірна кількість, складна логіка або рекурсивні виклики можуть ускладнити підтримку системи та знижувати її продуктивність. Особливо це стосується ситуацій, коли кілька тригерів впливають на одну й ту саму таблицю або спрацьовують ланцюгово. Невірно реалізовані тригери можуть призвести до помилок або навіть до блокування транзакцій. Тому при створенні тригерів необхідно дотримуватися принципів прозорості, передбачуваності та мінімізації побічних ефектів.

З методологічної точки зору, тригери слід розглядати як допоміжний, але дуже потужний інструмент підтримки логіки даних на рівні СУБД. На відміну від перевірок у прикладному коді, вони гарантують, що встановлені правила дотримуються незалежно від того, через який інтерфейс або додаток виконується зміна даних. Це особливо важливо в системах із багатьма точками доступу до бази - наприклад, у корпоративних ІТ-системах або вебсервісах, де змінити дані можуть як внутрішні процеси, так і зовнішні інтеграції.

Проектування тригерів також пов'язане з загальними принципами розробки інформаційної системи. Наприклад, якщо в системі реалізовано об'єктно-орієнтований підхід, то тригери можуть служити аналогом методів класів, що відповідають за реакцію на зміну стану об'єкта. У класичних транзакційних системах вони можуть виконувати роль контролерів бізнес-логіки на рівні бази даних. Незалежно від обраної архітектури, тригери дозволяють знизити дублювання коду, зменшити ризики помилок і забезпечити уніфіковану реакцію на події в системі.

Підсумовуючи, можна стверджувати, що тригери є невід'ємним елементом сучасних реляційних баз даних. Їх правильне використання дозволяє створити надійну, керовану й послідовну систему обробки даних, яка забезпечує реалізацію бізнес-логіки безпосередньо на рівні СУБД. Однак необхідно пам'ятати про обмеження цього інструменту, зокрема - про ризики надмірного ускладнення логіки, взаємозалежність подій та можливі проблеми продуктивності. Тому тригери мають проектуватися з урахуванням загальної архітектури системи, вимог до цілісності даних і очікуваного навантаження на базу. Грамотно реалізовані тригери значно підвищують якість системи, зменшують витрати на її супровід та роблять обробку даних більш ефективною й безпечною.

Список використаних джерел

3. Проектування баз і сховищ даних [Електронний ресурс] : навч.-метод. посіб. для самост. вивч. дисц. / Н. В. Ситник, М. Т. Краснюк ; М-во освіти і науки України, Київ. нац. екон. ун-т. - Електрон. текстові дані. - К. : КНЕУ, 2005. - 264 с.
4. Проектування баз і сховищ даних [Електронний ресурс] : навч. посіб. / Н. В. Ситник ; М-во освіти і науки України, Київ. нац. екон. ун-т. - Електрон. текстові дані. - К. : КНЕУ, 2004. - 348 с.