

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ



к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "бакалавр"

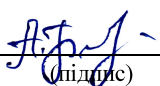
на тему: «Розробка мобільного застосунку для тайм-менеджменту
інформаційної системи ІТ-компанії»

Виконала: студентка групи 4145

 Беркунська В.М.
(підпис)

Керівник роботи:

старший викладач

(посада, науковий ступінь вчене звання)
 Павленко А.Ю.
(підпис)

м. Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 124 "Системний аналіз"
Освітня програма "Системний аналіз"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми



к.т.н, доц. Гайдаєнко О. В.

" ____ " _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти "бакалавр"

Студентці Беркунській Владиславі Максимівни

(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка мобільного застосунку для тайм-менеджменту інформаційної системи ІТ-компанії

Керівник роботи Павленко Альона Юріївна

Затверджені наказом ректора № ____-уч від " ____ " березня 2025 року.

2. Термін подання роботи:

20 червня 2025 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем,

4. Перелік питань, що належать до розробки (найменування розділів) розробка концепції мобільного застосунку для тайм-менеджменту інформаційної системи ІТ-компанії, розробка мобільного застосунку для тайм-менеджменту, реалізація мобільного застосунку для тайм-менеджменту інформаційної системи ІТ-компанії, охорона праці та техніка безпеки.

5. Перелік презентаційних матеріалів — організаційна структура користувачів додатку для тайм-менеджменту ІТ-компанії, схема запропонованої концепції, діаграма варіантів використання, діаграма діяльності створення задачі, концептуальна модель даних, фізична модель даних, інтерфейс головного екрану додатку, екран редагування задачі.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Павленко А.Ю., ст. викладач	03.02.2025	22.02.2025
2	Павленко А.Ю., ст. викладач	22.02.2025	21.03.2025
3	Павленко А.Ю., ст. викладач	21.03.2025	25.04.2025
4	Павленко А.Ю., ст. викладач	25.04.2025	07.06.2025

7. Дата видачі завдання 03 лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

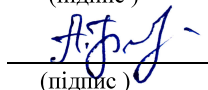
№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної галузі	13.02.2025	
2	Постановка задачі	15.02.2025	
3	Розробка концепції	22.02.2025	
4	Розробка проекту ІС	21.03.2025	
5	Реалізація проекту	25.04.2025	
6	Розробка питань з охорони праці	07.06.2025	
7	Розробка графічних матеріалів	12.06.2025	
8	Подання роботи на перевірку на плагіат	10.06.2025	
9	Подання роботи рецензенту	15.06.2025	
10	Подання роботи на захист	20.06.2025	

Студент


(підпис)

Беркунська В.М.
(ПІБ)

Керівник роботи


(підпис)

Павленко А.Ю.
(ПІБ)

АНОТАЦІЯ

Тема дипломного проєкту: «Розробка мобільного застосунку для тайм-менеджменту інформаційної системи ІТ-компанії».

Актуальність обраної теми зумовлена стрімким розвитком ІТ-сфери та зростаючою потребою в ефективному управлінні часом у процесі роботи над проєктами. В умовах багатозадачності, жорстких дедлайнів та командної взаємодії, питання продуктивності набуває особливого значення.

Мета дипломного проєкту – створення мобільного застосунку, який дозволяє покращити особисту та командну ефективність працівників ІТ-компанії шляхом впровадження інструментів тайм-менеджменту.

У рамках роботи було досліджено особливості діяльності ІТ-компаній, зокрема процеси розробки програмного забезпечення та методи управління часом, проаналізовано наявні рішення, визначено їхні недоліки та обґрунтовано потребу у спеціалізованому продукті. Розроблений застосунок містить функціонал таймера Pomodoro, ведення списку завдань, перегляду історії продуктивності та налаштувань, що відповідають специфіці роботи в ІТ-середовищі.

Дипломна робота оформлена на 87 сторінках, містить 21 рисунок, 17 таблиць, 4 додатки.

Робота написана українською мовою.

Ключові слова: тайм-менеджмент, таймер Pomodoro, мобільний застосунок, ІТ-компанія, аналітика продуктивності, керування задачами.

ABSTRACT

The topic of the diploma project: "Development of a mobile application for time management of the information system of an IT company".

The relevance of the chosen topic is due to the rapid development of the IT sphere and the growing need for effective time management in the process of working on projects. In conditions of multitasking, tight deadlines and team interaction, the issue of productivity becomes especially important.

The goal of the diploma project is to create a mobile application that allows you to improve the personal and team efficiency of IT company employees by implementing time management tools.

The work examined the specifics of IT companies' activities, in particular software development processes and time management methods, analyzed existing solutions, identified their shortcomings and justified the need for a specialized product. The developed application contains the functionality of the Pomodoro timer, maintaining a list of tasks, viewing performance history and settings that meet the specifics of working in an IT environment.

The thesis is formatted on 78 pages, contains 21 figures, 17 tables, 4 appendices.

The work is written in Ukrainian.

Keywords: time management; Pomodoro timer; mobile application; IT company; productivity analytics; task management.

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1. РОЗРОБКА КОНЦЕПЦІЇ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ТАЙМ-МЕНЕДЖМЕНТУ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІТ- КОМПАНІЙ	9
1.1 Аналіз предметної області	9
1.2 Вимоги до інформаційної системи	12
1.3. Розробка концепції інформаційної системи.....	17
РОЗДІЛ 2. РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ТАЙМ- МЕНЕДЖМЕНТУ	22
2.1. Загальносистемні рішення	22
2.2. Рішення щодо інформаційного забезпечення	27
2.3. Рішення з математичного забезпечення	38
2.4. Рішення щодо програмного забезпечення	40
2.5. Рішення щодо технічного забезпечення	46
РОЗДІЛ 3. РЕАЛІЗАЦІЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ТАЙМ- МЕНЕДЖМЕНТУ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІТ-КОМПАНІЇ ..	51
РОЗДІЛ 4. ОХОРОНА ПРАЦІ ТА ТЕХНІКА БЕЗПЕКИ	58
Список використаних джерел	66
ВИСНОВКИ	69
ДОДАТКИ.....	71
ДОДАТОК А	71
ДОДАТОК Б	81
ДОДАТОК В	83
ДОДАТОК Г	84

ВСТУП

Актуальність теми даної дипломної роботи полягає в тому, що в умовах стрімкого розвитку інформаційних технологій та зростаючої конкуренції на ринку IT-послуг, ефективне управління часом стає критичним фактором успіху як для окремих фахівців, так і для IT-компаній в цілому. Сучасні умови праці характеризуються високою інтенсивністю, багатозадачністю та необхідністю швидкого прийняття рішень, що підвищує значимість впровадження ефективних систем тайм-менеджменту в діяльність IT-компаній.

Незважаючи на наявність значної кількості мобільних застосунків для тайм-менеджменту на ринку, більшість з них не враховують специфіку роботи IT-компаній, зокрема особливості організації процесу розробки програмного забезпечення, методології управління проєктами та комунікаційні потреби команд розробників. Це створює потребу в розробці спеціалізованого мобільного застосунку, який би враховував ці особливості та ефективно інтегрувався з інформаційними системами IT-компаній.

Метою дослідження є розробка мобільного застосунку для тайм-менеджменту, який дозволить оптимізувати процеси планування та контролю робочого часу в IT-компаніях.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. визначити функціональні та нефункціональні вимоги до розроблюваного застосунку;
2. спроектувати архітектуру мобільного застосунку та інформаційної системи;
3. розробити базу даних та інтерфейс користувача;
4. реалізувати мобільний застосунок на платформі Android з використанням Java;

Об'єктом дослідження є процеси управління часом в IT-компаніях.

Предметом дослідження є методи та інструменти автоматизації тайм-менеджменту з використанням мобільних технологій.

У процесі дослідження використано комплекс загальнонаукових та спеціальних методів: системний аналіз – для дослідження особливостей інформаційних систем ІТ-компаній; порівняльний аналіз – для огляду існуючих мобільних застосунків; методи об'єктно-орієнтованого проектування – для розробки архітектури застосунку; методи програмування – для реалізації програмного коду; методи тестування програмного забезпечення – для перевірки працездатності розробленого застосунку.

Розроблений мобільний застосунок може бути впроваджений в діяльність ІТ-компаній для оптимізації процесів планування та контролю робочого часу співробітників, підвищення продуктивності роботи команд розробників та покращення комунікації між учасниками проєктів.

Робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. У першому розділі розглянуто теоретико-методологічні основи тайм-менеджменту та розробки мобільних застосунків для інформаційних систем ІТ-компаній. Другий розділ присвячено проектуванню мобільного застосунку для тайм-менеджменту. У третьому розділі описано розробку та реалізацію мобільного застосунку на платформі Android з використанням Java. Четвертий розділ містить спеціальні розділи, включаючи аналіз витрат на розробку та питання охорони праці і безпеки в надзвичайних ситуаціях.

РОЗДІЛ 1. РОЗБРОБКА КОНЦЕПЦІЇ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ТАЙМ-МЕНЕДЖМЕНТУ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІТ-КОМПАНІЙ

1.1 Аналіз предметної області

У сучасних умовах розвитку цифрових технологій інформаційні системи є ключовим компонентом функціонування ІТ-компаній. Особливу роль відіграють системи підтримки внутрішніх процесів, серед яких управління часом займає важливе місце. Тайм-менеджмент в ІТ-компанії є критичним для підвищення продуктивності команди, забезпечення своєчасного виконання проєктів, зменшення стресу та покращення балансу між роботою і особистим життям.

ІТ-компанії функціонують у динамічному середовищі, де одночасно реалізується багато паралельних проєктів. Це вимагає від працівників вміння ефективно планувати час, пріоритезувати завдання та контролювати хід їх виконання. Використання ручних методів (наприклад, таблиць, паперових блокнотів або окремих додатків без інтеграції з системою компанії) не задовольняє потреби в аналітиці, командній взаємодії та масштабованості.

Інтеграція мобільного застосунку для тайм-менеджменту з інформаційною системою ІТ-компанії дозволяє створити єдине середовище для планування задач, трекінгу часу, нагадувань, оцінки ефективності та звітності. Особливо важливим є забезпечення гнучкості для роботи в умовах віддалених команд, підтримки методологій Agile та Scrum, а також надання функцій Pomodoro, GTD (Getting Things Done).

Аналіз організаційної структури ІТ-компанії дозволяє визначити користувачів майбутньої системи, їх ролі та функціональні обов'язки.

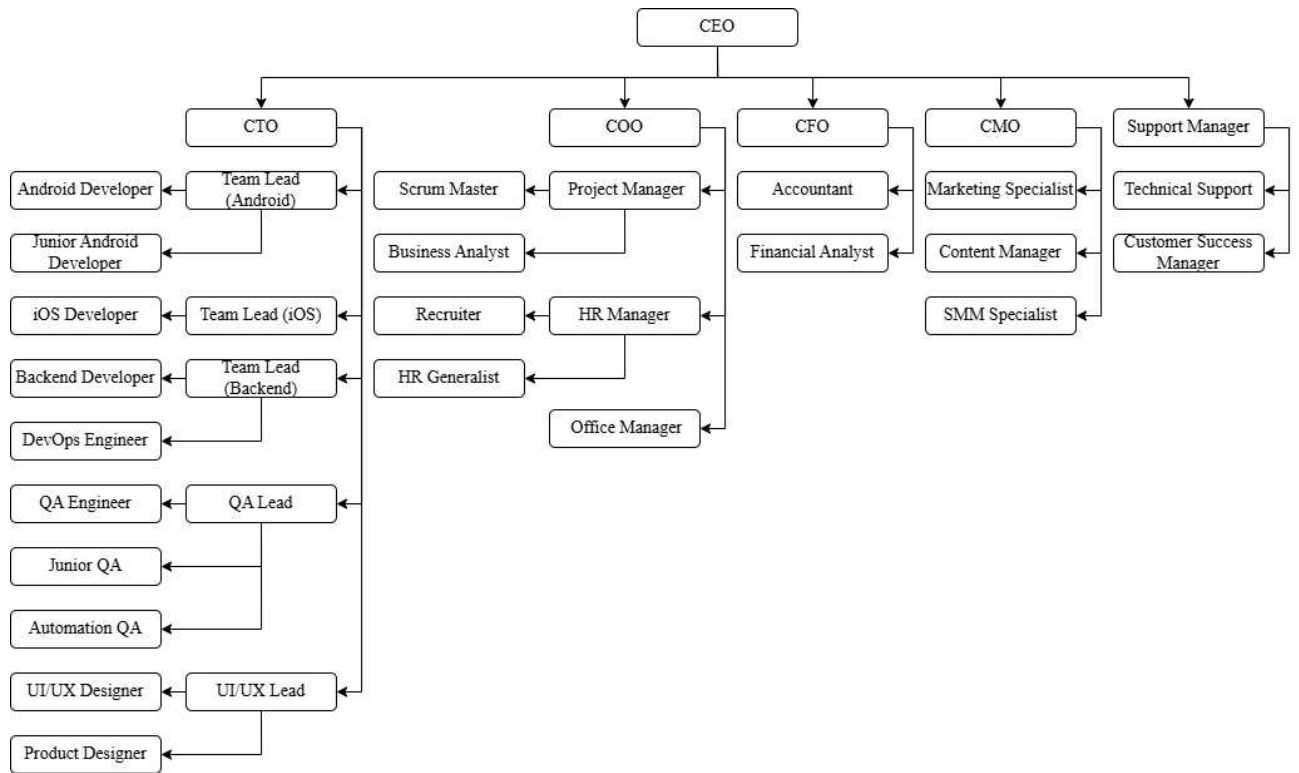


Рисунок 1.1. Організаційна структура користувачів додатку для тайм-менеджменту ІТ-компанії

Опис функцій ключових посад:

- CEO (Chief Executive Officer): формує стратегію розвитку компанії, контролює реалізацію ключових цілей, приймає стратегічні рішення, представляє компанію перед партнерами та інвесторами.
- CTO (Chief Technology Officer): відповідає за технологічну політику компанії, обирає інструменти для розробки, керує технічними командами, контролює якість продукту.
- Team Lead (Android/iOS/Backend): керує командою відповідного напрямку розробки, проводить технічні рев'ю коду, планує спринти, координує виконання задач.
- Android/iOS/Backend Developer: займається безпосередньою реалізацією функціоналу застосунку згідно з технічними завданнями.
- Junior Developer: виконує прості задачі під контролем старших колег, навчається на практиці, бере участь у підтримці існуючого коду.

- DevOps Engineer: автоматизує процеси розгортання, моніторингу та масштабування застосунку, відповідає за інфраструктуру.
- QA Lead: організовує тестування продукту, складає тест-плани, координує роботу тестувальників.
- QA Engineer / Junior QA: тестують функціонал, знаходять і фіксують помилки, створюють автоматизовані тести.
- Automation QA: створює автотести для перевірки стабільності та регресії.
- UI/UX Lead: формує дизайн-стратегію продукту, контролює відповідність UI/UX рішенням користувацького досвіду.
- UI/UX Designer: розробляє макети екранів, навігацію, типографіку, стилі.
- Product Designer: працює над логікою продукту, зручністю використання, зосереджується на поведінці користувача.
- COO (Chief Operating Officer): відповідає за адміністративну частину бізнесу, оптимізує процеси всередині компанії, забезпечує ресурсами проекти.
- Project Manager: планує проекти, слідкує за строками, комунікує з замовниками, проводить зустрічі команд.
- Scrum Master: підтримує команду у роботі за Scrum, веде спринти, прибирає перешкоди у процесі.
- Business Analyst: виявляє потреби клієнтів, аналізує вимоги, створює документацію для розробки.
- HR Manager: управляє персоналом, розвиває корпоративну культуру, вирішує кадрові питання.
- Recruiter: займається пошуком та відбором кандидатів на відкриті вакансії.
- HR Generalist: супроводжує працівника з моменту найму до звільнення, відповідає за документообіг і внутрішню комунікацію.
- Office Manager: організовує логістику, забезпечує побутові умови в офісі.

- CFO (Chief Financial Officer): керує фінансами, формує бюджет, здійснює фінансову звітність.
- Accountant: веде бухгалтерію, готує звіти, слідкує за податковою документацією.
- Financial Analyst: аналізує фінансові показники, прогнозує витрати, оцінює ефективність проєктів.
- CMO (Chief Marketing Officer): розробляє маркетингову стратегію, керує зовнішніми комунікаціями.
- Marketing Specialist: проводить рекламні кампанії, працює з аналітикою ринку.
- Content Manager: створює тексти для сайту, соціальних мереж, готує медіаматеріали.
- SMM Specialist: управляє соціальними мережами компанії, створює пости, аналізує охоплення.
- Support Manager: керує службою підтримки, формує базу знань, слідкує за якістю обслуговування клієнтів.
- Technical Support: відповідає на звернення клієнтів, допомагає з технічними проблемами.
- Customer Success Manager: працює з клієнтами після продажу, допомагає ефективно використовувати продукт, зменшує рівень відтоку клієнтів.

Такий склад компанії дозволяє ідентифікувати різні категорії користувачів інформаційної системи, їх очікування та потреби, що враховується при формуванні вимог до функціональності мобільного застосунку для тайм-менеджменту.

1.2 Вимоги до інформаційної системи

На основі аналізу предметної області та організаційної структури компанії можна сформулювати вимоги до майбутньої інформаційної системи

тайм-менеджменту. Ці вимоги розділяються на функціональні, нефункціональні, системні та користувацькі.

Функціональні вимоги до мобільного застосунку для тайм-менеджменту IT-компаній представляють собою систематизований перелік функціональних можливостей, необхідних для задоволення специфічних потреб користувачів у контексті управління часом при розробці програмного забезпечення. Детальна специфікація функціональних вимог з аналізом поточного стану реалізації представлена у таблиці 1.1.

Таблиця 1.1 – Функціональні вимоги до мобільного застосунку

Код вимоги	Назва	Опис
FR1	Управління завданнями	Функціонал для створення, редагування та видалення завдань з детальними атрибутами
FR1.1	Створення нового завдання	Можливість додавання нового завдання з назвою, описом, прив'язкою до проекту
FR1.2	Редагування існуючого завдання	Зміна параметрів збереженого завдання
FR1.3	Видалення завдання	Можливість вилучення завдання з системи
FR1.4	Встановлення терміну виконання	Визначення дати завершення роботи над завданням
FR1.5	Встановлення пріоритету	Градація важливості завдань (низький, середній, високий)
FR1.6	Зміна статусу завдання	Відстеження прогресу виконання (нове, в роботі, завершене, заблоковане)
FR2	Управління проектами	Організація завдань у проектні структури
FR2.1	Створення нового проекту	Додавання проекту з назвою та описом
FR2.2	Редагування інформації про проект	Модифікація параметрів проекту
FR2.3	Додавання/видалення завдань до/з проекту	Управління структурою проекту
FR3	Календарне планування	Візуалізація завдань у календарному форматі
FR3.1	Відображення завдань у календарі	Представлення завдань за днями, тижнями, місяцями

FR3.2	Створення/редагування завдань з календаря	Інтеграція функціоналу управління завданнями в календарний інтерфейс
FR4	Нагадування та сповіщення	Система повідомлень про важливі події
FR4.1	Налаштування нагадувань	Встановлення часу сповіщення для завдань
FR4.2	Отримання push-сповіщень	Сповіщення про терміни, призначення, коментарі
FR5	Відстеження часу	Моніторинг часових витрат на виконання завдань
FR5.1	Ручний запуск/зупинка таймера	Управління відстеженням часу для конкретного завдання
FR6	Командна робота	Колаборативні функції для спільного управління завданнями
FR6.1	Додавання учасників до проектів	Розширення команди проекту через запрошення
FR6.2	Призначення завдань виконавцям	Розподіл відповідальності між учасниками
FR6.3	Коментування завдань	Асинхронна комунікація через коментарі
FR7	Візуалізація та звітність	Представлення даних у зручному форматі
FR7.1	Відображення списку завдань	Сортування та фільтрація завдань за різними критеріями
FR8	Синхронізація та офлайн-доступ	Забезпечення доступу до даних без підключення до мережі
FR8.1	Офлайн-режим роботи	Можливість перегляду та редагування даних без інтернету

Нефункціональні вимоги встановлюють критерії якості системи, визначаючи параметри продуктивності, надійності, безпеки та інших аспектів, що впливають на загальну ефективність та користувацький досвід.

Детальна специфікація нефункціональних вимог до мобільного застосунку для тайм-менеджменту ІТ-компаній представлена у таблиці 1.2.

Таблиця 1.2 – Нефункціональні вимоги до мобільного застосунку

Код вимоги	Категорія	Опис вимоги	Метрика	Цільове значення
NFR1	Продуктивність	Забезпечення швидкодії системи		

NFR1.1		Швидкість запуску застосунку	Час до повної готовності інтерфейсу	< 3 секунд
NFR1.2		Відзивчивість інтерфейсу	Час реакції на дії користувача	< 100 мс
NFR1.3		Швидкість синхронізації даних	Час оновлення даних при стабільному з'єднанні	< 5 секунд
NFR1.4		Завантаження списку завдань	Час від запиту до відображення списку (50 завдань)	< 1 секунди
NFR2	Надійність	Забезпечення стабільності роботи		
NFR2.1		Стабільність функціонування	Середній час між збоями	> 100 годин
NFR2.2		Збереження даних	Відсутність втрати інформації при нештатних ситуаціях	100%
NFR3	Безпека	Захист даних та забезпечення конфіденційності		
NFR3.1		Безпечна автентифікація	Використання сучасних методів захисту	Хешування паролів з "сіллю"
NFR3.2		Захист передачі даних	Шифрування комунікацій	HTTPS
NFR3.3		Захист локальних даних	Шифрування локальної бази даних	AES-256
NFR4	Масштабованість	Здатність системи обробляти зростаюче навантаження		
NFR4.1		Масштабованість архітектури	Максимальна кількість користувачів	> 1000

NFR5	Юзабіліті	Зручність використання системи		
NFR5.1		Інтуїтивність інтерфейсу	Суб'єктивна оцінка користувачами	> 4/5
NFR5.2		Швидкість освоєння	Час до самостійного виконання основних функцій	< 30 хвилин
NFR5.3		Відповідність стандартам дизайну	Дотримання Material Design	100%
NFR5.4		Ефективність створення завдання	Кількість кроків для створення простого завдання	≤ 3 кроки
NFR6	Сумісність	Функціонування на різних пристроях		
NFR6.1		Підтримка версій Android	Діапазон версій ОС	Android 7.0 - 13.0
NFR6.2		Адаптивність інтерфейсу	Коректне відображення на різних пристроях	Смартфон и та планшети
NFR7	Підтримуваність	Зручність модифікації та розширення		
NFR7.1		Якість коду	Відповідність стандартам кодування	90%
NFR7.2		Модульність архітектури	Час на додавання нової функціональності	< 40 годин

Користувацькі вимоги:

- Розробник має отримати інструмент для ефективного тайм-менеджменту, у тому числі Pomodoro;
- Тімлід повинен мати змогу контролювати задачі команди та отримувати звіти;

- Менеджер з проєктів — доступ до статистики по проєктах і командній активності;
- HR-менеджер — оцінка завантаженості співробітників для планування ресурсів;
- Тестувальник — створення задач на перевірку та моніторинг стану задач

Системні вимоги:

- Операційна система: Android 8.0 (Oreo) або вище;
- Процесор: ARM з частотою не менше 1.5 ГГц;
- Оперативна пам'ять: щонайменше 2 ГБ для стабільної роботи;
- Вільне місце на пристрої: не менше 150 МБ для встановлення та збереження локальної бази даних;
- Дозвіл екрана: мінімум 720×1280 (HD);

Таким чином, сформовані вимоги охоплюють ключові функції, необхідні для забезпечення ефективного управління часом та задачами в ІТ-компанії. Вони враховують інтереси усіх користувачів, що будуть взаємодіяти із системою.

1.3. Розробка концепції інформаційної системи

Інформаційна система тайм-менеджменту призначена для автоматизації обліку задач, відстеження часу роботи співробітників ІТ-компанії та формування звітності на основі отриманих даних.

Предметною областю даної інформаційної системи є процеси організації робочого часу в межах проєктної діяльності, що включає створення та виконання задач, застосування техніки Pomodoro, аналіз ефективності роботи працівників.

Робота ІТ-компанії з використанням інформаційної системи тайм-менеджменту проходить наступним чином. Працівник створює задачу у мобільному додатку, вказуючи назву, опис, дедлайн та інші характеристики. Потім він запускає таймер Pomodoro та виконує роботу за встановленими сесіями. Після завершення сесії дані про тривалість, статус виконання та час фіксуються в базі даних. Якщо задача призначена іншому учаснику, тимлід або менеджер має змогу контролювати її виконання. За результатами виконаних задач формуються звіти та аналітика, які доступні для перегляду відповідним користувачам.

Для реалізації мобільного застосунку обрано технології, що відповідають сучасним вимогам до Android-розробки:

- Мова програмування: Java — перевірена часом, широко підтримується та є сумісною з Android SDK;
- Середовище розробки: Android Studio — офіційне IDE для Android, яке підтримує всі необхідні інструменти (емулятор, дебагер, Gradle);
- База даних: SQLite з використанням бібліотеки Room для роботи з реляційною структурою;
- Архітектура: MVVM (Model–View–ViewModel) — забезпечує чіткий поділ логіки та уможлиблює підтримку й масштабування;
- Інші бібліотеки: LiveData та ViewModel для реактивності, Data Binding для двосторонньої прив'язки, Coroutines для багатопотоковості;
- Інструменти тестування: JUnit, Espresso для автоматизованих та UI-тестів.

Використання архітектурного підходу MVVM дозволяє забезпечити незалежність між рівнями даних, інтерфейсу та логіки обробки подій. Це значно спрощує супровід коду, пришвидшує розробку, полегшує командну роботу. ViewModel забезпечує обробку даних, які надходять від Model (через DAO) та передає їх до View без безпосередньої взаємодії між шарами.

Загальна концепція інформаційної системи:

Інформаційна система призначена для автоматизації процесів управління часом і задачами в межах IT-компанії. Користувачі матимуть змогу взаємодіяти із системою через мобільний інтерфейс, який дозволяє:

- створювати, редагувати та видаляти задачі;
- запускати таймер Pomodoro та відслідковувати сесії;
- групувати задачі за категоріями й проєктами;
- переглядати статистику у вигляді графіків та таблиць;
- отримувати сповіщення й нагадування про важливі події;
- працювати офлайн із подальшою синхронізацією.

Усі дані зберігаються локально на пристрої користувача, що забезпечує автономність роботи. Водночас структура бази даних дозволяє у майбутньому реалізувати мережеву синхронізацію та командну взаємодію через сервер.

Основні компоненти системи:

- Інтерфейс користувача (UI): адаптивний дизайн, підтримка темної теми, локалізація;
- База даних: реляційна структура з таблицями tasks, projects, users, categories, pomodoro_sessions;
- Бізнес-логіка: ViewModel обробляє дані, керує станами, передає зміни між UI і Model;
- Система сповіщень: NotificationManager, AlarmManager;
- Схема навігації: Jetpack Navigation з фрагментами (Home, Tasks, Timer, Statistics, Settings).

Концепція інформаційної системи представлена на рисунку 1.2.

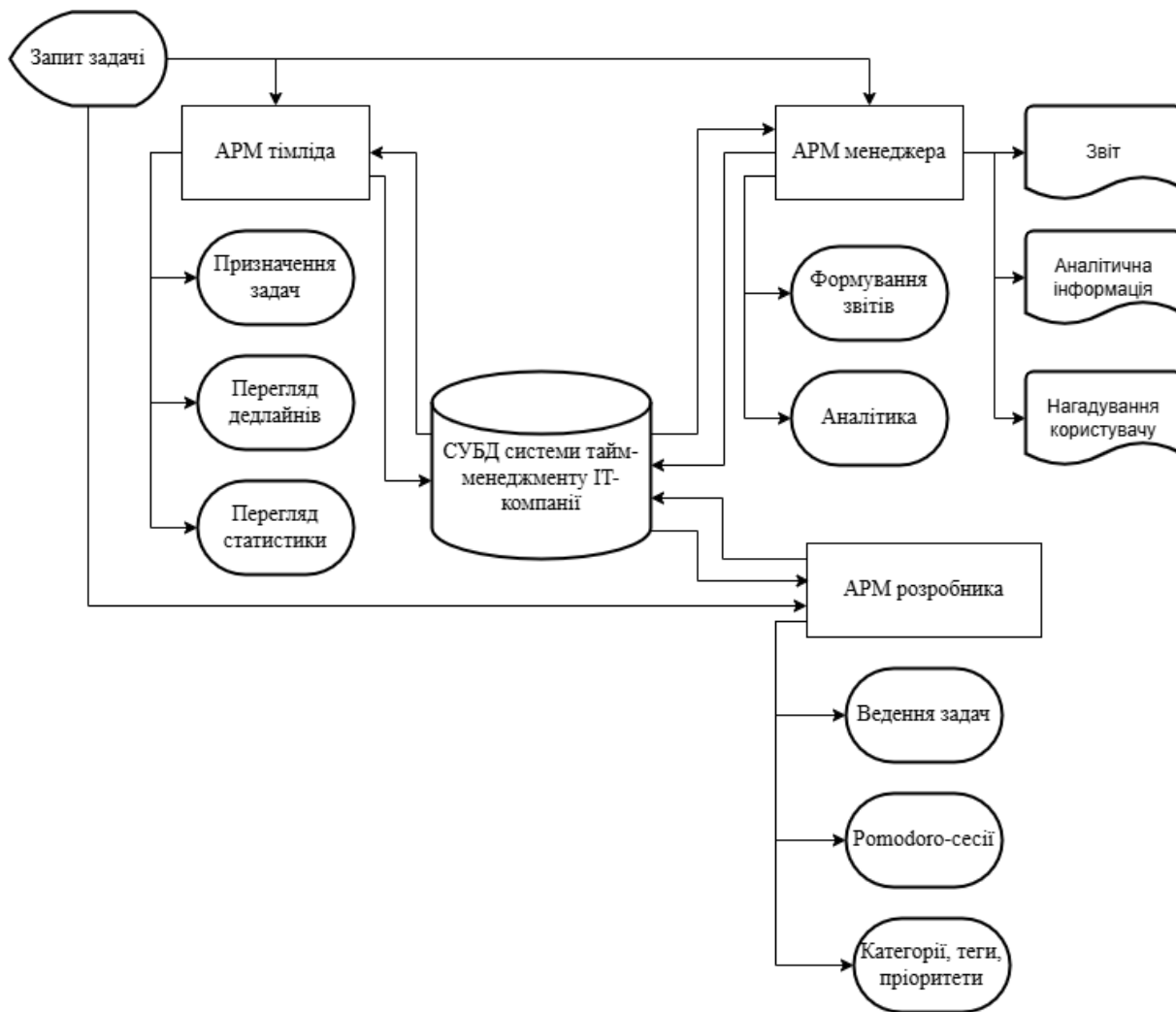


Рис. 1.2. Схема автоматизованої інформаційної системи тайм-менеджменту.

На цій схемі зображено, що кожен користувач (розробник, тімлід, менеджер, HR) має змогу працювати у своєму інтерфейсі — автоматизованому робочому місці. Усі введені та згенеровані дані зберігаються в єдиній локальній базі даних застосунку. Передбачено, що система працює автономно — користувачі мають доступ лише до власного пристрою. Такий підхід спрощує реалізацію без необхідності налаштування мережевої взаємодії, але у майбутньому можлива інтеграція із серверною частиною.

Висновок за розділом 1: у цьому розділі здійснено комплексний аналіз предметної області та організаційної структури ІТ-компанії, визначено ролі користувачів і їхні функціональні потреби. На основі отриманих даних сформульовано функціональні, нефункціональні, системні та користувацькі

вимоги й окреслено загальну концепцію інформаційної системи тайм-менеджменту.

РОЗДІЛ 2. РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ТАЙМ-МЕНЕДЖМЕНТУ

2.1. Загальносистемні рішення

Користувачами інформаційної системи є співробітники ІТ-компанії: розробники, тімліди та HR-фахівці. Кожен із них виконує у системі певний набір функцій відповідно до своїх повноважень.

Розробники створюють задачі, вказують їхній опис, дедлайн, пріоритет, категорію, а також використовують таймер Pomodoro. По завершенню сесії Pomodoro результат зберігається в базі даних, а задача оновлює свій статус. Це дозволяє аналізувати хід виконання роботи та відстежувати продуктивність.

Тімліди мають змогу призначати задачі членам команди, контролювати строки виконання та переглядати загальну статистику по задачах. HR-фахівці працюють із модулем аналізу завантаженості працівників, переглядають зведену інформацію про виконані задачі, активність працівників у Pomodoro-сесіях, а також формують звіти.

Діаграма прецедентів зображена на Рисунку 2.1.

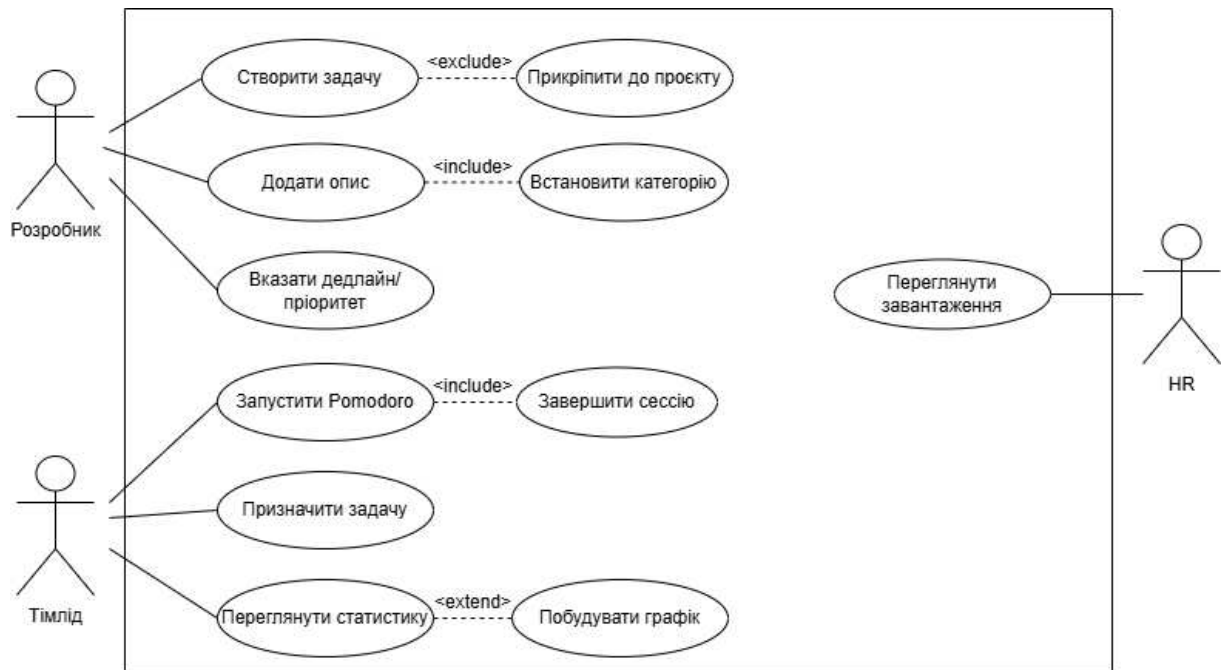


Рисунок 2.1. Діаграма прецедентів

На діаграмі прецедентів (Рис.2.1) зображено перелік дій, які виконуються користувачами системи. Деякі з них включають інші процеси (позначено як «include»), наприклад, запуск Pomodoro завжди передбачає збереження сесії. Інші можуть бути необов'язковими (extend), наприклад, додавання тегів до задачі чи побудова графіка за статистикою. Такий підхід дозволяє зробити систему гнучкою, адаптованою до потреб різних користувачів.

У таблицях 2.1–2.3 наведено типові сценарії використання інформаційної системи для різних ролей користувачів, що демонструють логіку взаємодії з ключовими функціями застосунку.

Таблиця 2.1 – Сценарій створення задачі

Мета виконання	Використовується для створення нової задачі в системі тайм-менеджменту
Виконавець дії	Розробник

Умова початку	Необхідність зафіксувати нове завдання для подальшого виконання
Тригер	Натискання на кнопку «Створити задачу» у відповідному розділі застосунку
Послідовність дій	Користувач натискає кнопку «Створити задачу», заповнює назву, опис, дедлайн, обирає пріоритет, категорію та, за потреби, додає теги. Далі натискає «Зберегти»
Результат	Інформація про задачу зберігається в локальній базі даних (Room/SQLite)

Таблиця 2.2 – Сценарій ведення Pomodoro

Мета виконання	Облік часу, витраченого на виконання задачі
Виконавець дії	Розробник
Умова початку	Завдання вже створене та вибране користувачем
Тригер	Натискання на кнопку «Старт»
Послідовність дій	Користувач натискає кнопку «Старт». Після завершення 25 хвилин система автоматично фіксує сесію та пропонує зробити перерву
Результат	Дані про сесію зберігаються в таблиці PomodoroSessions бази даних

Таблиця 2.3 – Сценарій призначення задачі

Мета виконання	Делегування задачі розробнику або іншому співробітнику
Виконавець дії	Тімлід
Умова початку	Завдання має бути розподілене між членами команди
Тригер	Вибір задачі та натискання на поле призначення виконавця
Послідовність дій	Тімлід обирає задачу зі списку, відкриває її деталі, обирає зі списку співробітника та натискає кнопку «Призначити»
Результат	У таблиці Tasks оновлюється поле user_id, що пов'язує задачу з виконавцем

Одним із ключових функціональних сценаріїв у системі тайм-менеджменту є створення нової задачі користувачем. Цей процес має важливе значення для організації робочого часу, планування активностей та забезпечення контролю над виконанням завдань.

У межах мобільного застосунку створення задачі ініціюється користувачем (наприклад, розробником), який взаємодіє з інтерфейсом системи.

Процес створення задачі охоплює взаємодію двох акторів — користувача та системи, що ілюструється на діаграмі діяльності на Рис. 2.2. Ця діаграма дозволяє детально описати логіку виконання дій, перевірку вхідних даних, обробку помилок та передачу задачі до інтерфейсу.

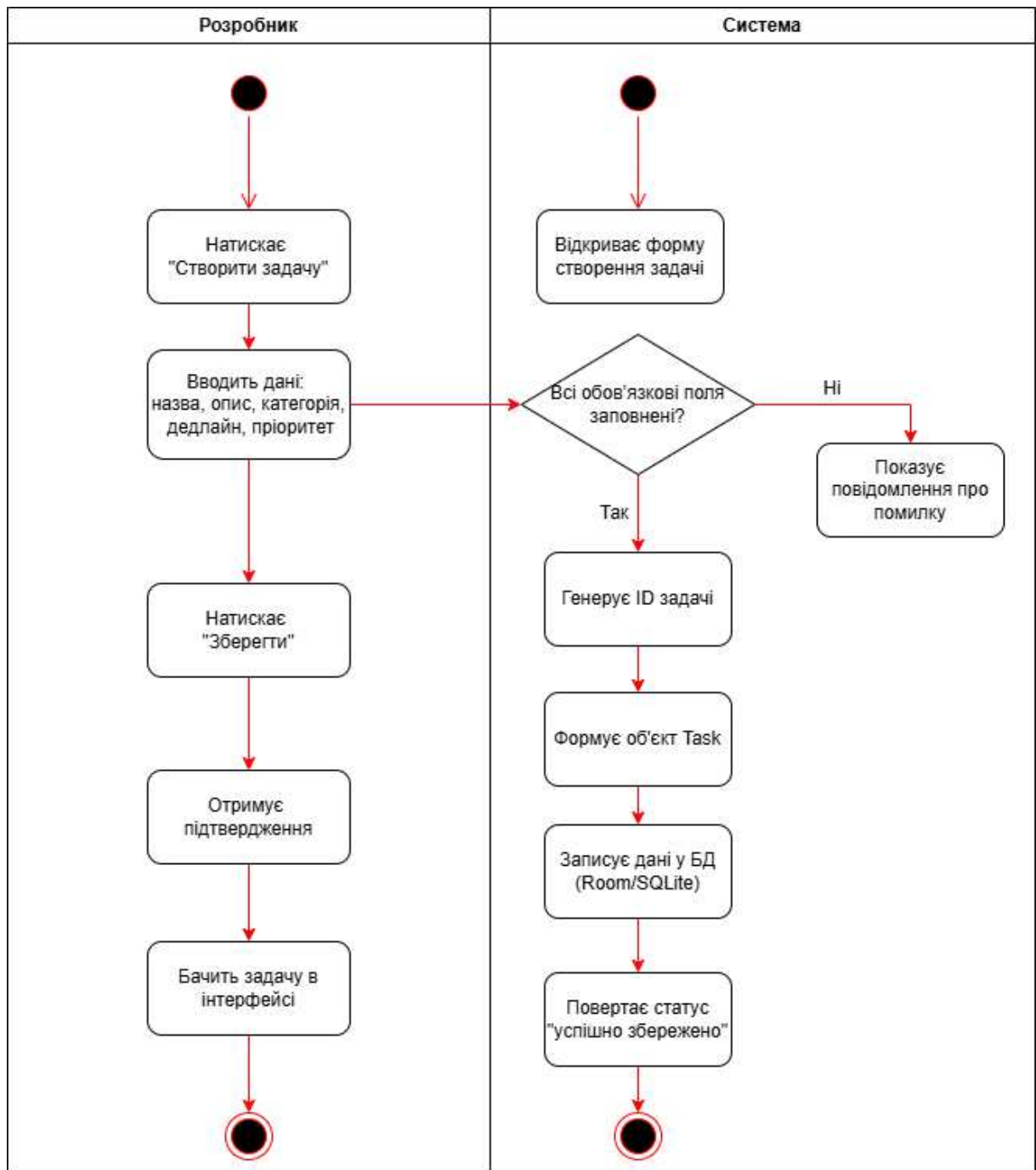


Рисунок 2.2. Діаграма діяльності створення задачі

На діаграмі діяльності показано типовий сценарій створення нового завдання у мобільному застосунку для тайм-менеджменту. У цьому процесі бере участь користувач (співробітник ІТ-компанії) та система. Ініціація відбувається через натискання кнопки «+», що відкриває форму створення задачі. Користувач вводить обов'язкову інформацію (назва), а також додаткові параметри — опис, проект, дедлайн, пріоритет, теги.

Після натискання кнопки «Зберегти» система виконує валідацію даних. Якщо обов'язкові поля не заповнено, вона повідомляє про помилку і

просить виправити. У разі успішної перевірки система створює об'єкт Task, генерує його унікальний ідентифікатор та зберігає в базі даних Room/SQLite. Потім нове завдання відображається у списку задач або календарному поданні, і користувач отримує повідомлення про успішне збереження.

Діаграма дозволяє чітко відстежити послідовність подій та розмежування відповідальності між користувачем і системою, а також враховує обробку альтернативних сценаріїв, що передбачає підвищення надійності та зручності користувацького досвіду.

2.2. Рішення щодо інформаційного забезпечення

Концептуальна модель будується на основі виявлених сутностей, їх атрибутів та зв'язків, що відображають бізнес-логіку системи тайм-менеджменту. Вона забезпечує високорівневий погляд на структуру даних без прив'язки до конкретної СУБД. Основне призначення ER-діаграми — візуалізувати, як сутності взаємодіють одна з одною, які атрибути вони мають та через які зв'язки пов'язані. Концептуальна модель сприяє перевірці коректності обраних сутностей, дозволяє виявити непотрібні чи відсутні зв'язки й служить підґрунтям для побудови наступних логічної та фізичної моделей.

Концептуальна модель даних представлена на Рисунку 2.

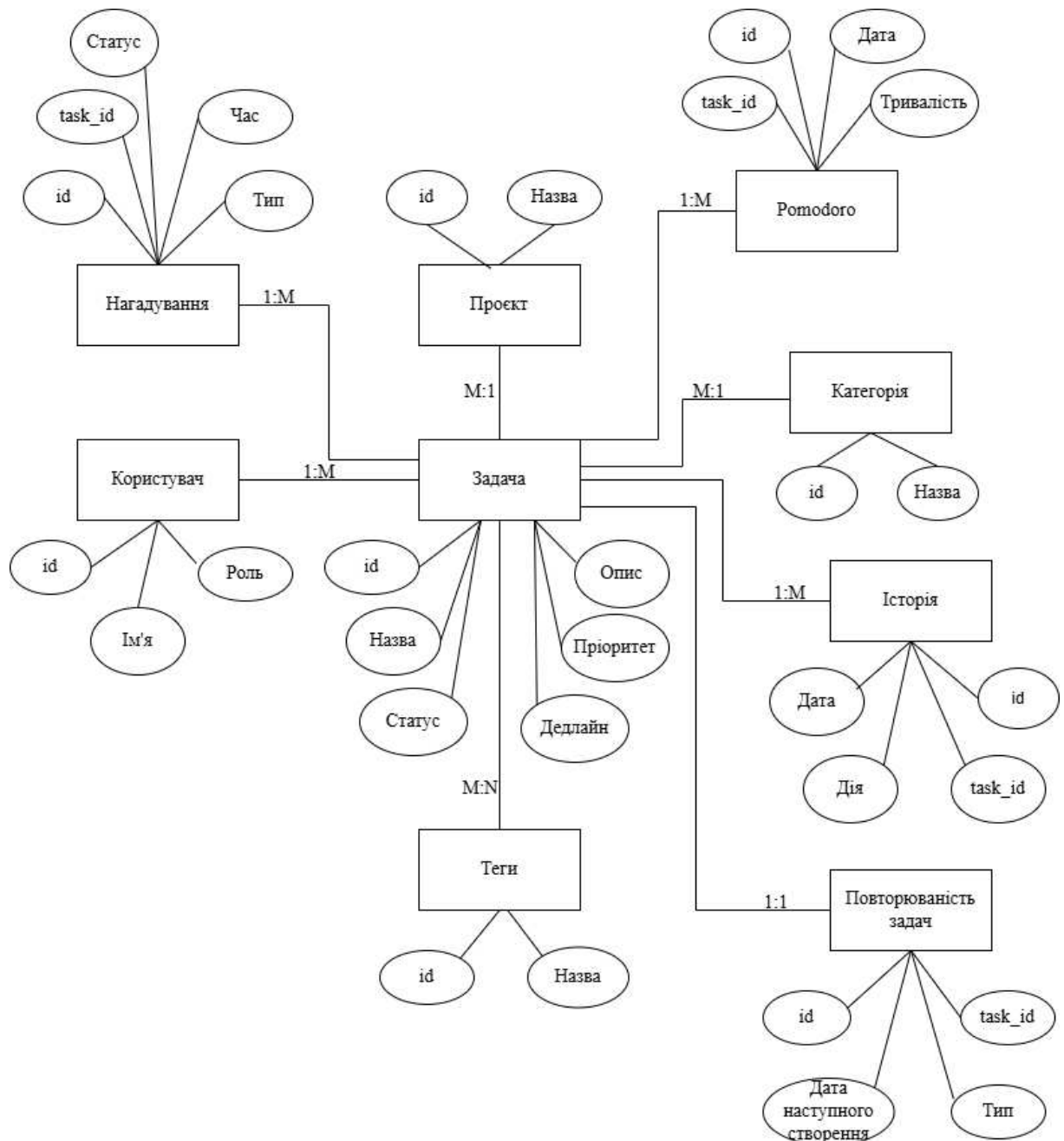


Рисунок 2.3. Концептуальна модель даних

Фізична модель враховує особливості конкретної системи управління базами даних (PostgreSQL), оптимізацію зберігання інформації та забезпечення високої продуктивності під час виконання запитів. На цьому етапі кожна сутність логічної моделі трансформується в таблицю бази даних з чітко визначеними типами полів, обмеженнями цілісності та індексами, що забезпечують швидкий доступ і коректність даних.

Основні завдання, які вирішуються під час побудови фізичної моделі:

- Визначення типів даних і розмірів полів — для кожного атрибуту обирається найбільш придатний SQL-тип (UUID, VARCHAR, TEXT, TIMESTAMP тощо), що оптимізує пам'ять і зберігає точність даних.
- Встановлення обмежень цілісності — первинні ключі (PK) гарантують унікальність записів, зовнішні ключі (FK) забезпечують коректні зв'язки між таблицями, а обмеження NOT NULL та UNIQUE запобігають появі некоректних чи дубльованих даних.
- Нормалізація структури — фізична модель приводиться до третьої нормальної форми (3NF) для усунення надлишковості та забезпечення однозначності кожного атрибуту.
- Проектування індексів — для полів, за якими здійснюється частий пошук або сортування (наприклад, ключі, зовнішні ключі, дати), створюються індекси B-tree або GIN, щоб знизити час виконання SELECT-запитів.
- Оптимізація запитів — аналізуються найважливіші сценарії використання даних (фільтрація задач, формування звітів, відстеження Pomodoro-сесій) і проектуються матеріалізовані представлення або індексовані уявлення за потреби.
- Забезпечення транзакційної цілісності — вибір механізму блокувань (MVCC) та налаштування рівнів ізоляції транзакцій гарантують стійкість даних під час одночасного доступу.
- Безпека доступу — на рівні бази даних встановлюються ролі та права користувачів, що дозволяє обмежити читання й зміну таблиць лише авторизованим об'єктам.
- Резервне копіювання та відновлення — на цьому етапі визначається стратегія бекупів (повні, інкрементальні), політики зберігання архівів та механізми реплікації для забезпечення безперервності бізнес-процесів.

Результатом виконання цього етапу є набір DDL-скриптів, що створюють у базі даних таблиці, обмеження, індекси та інші об'єкти (тригери, послідовності). Далі ці скрипти можуть бути застосовані до тестового середовища для перевірки працездатності моделі та оцінки продуктивності перед запуском у продукцію.

Фізична модель представлена на Рисунку 2.4

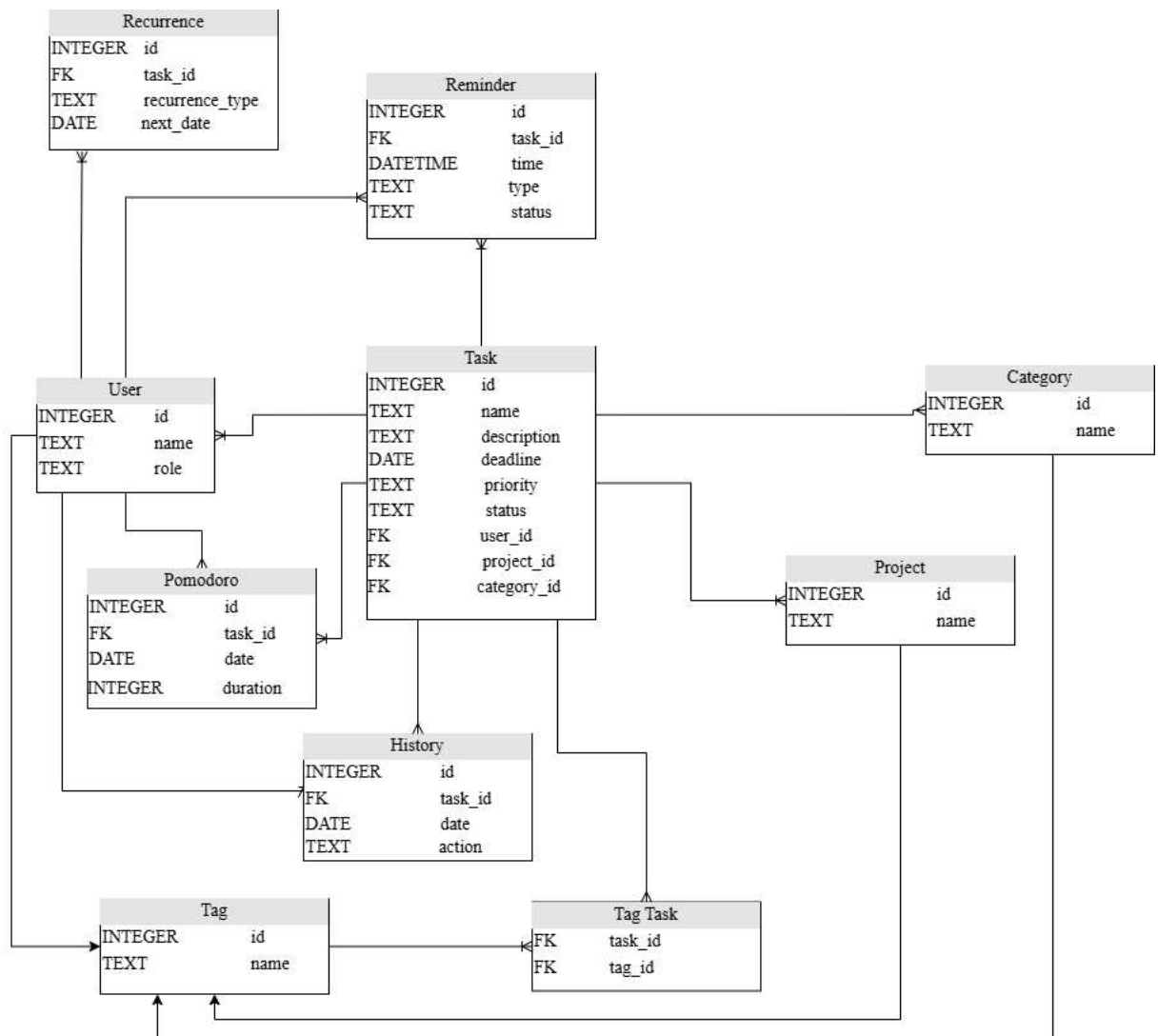


Рисунок 2.4. Фізична модель даних додатку для тайм менеджменту

База даних починається з об'єкту User.

Опис таблиці «User» представлений на таблиці 2.4

Таблиця 2.4 – Опис таблиці "User"

Назва поля	Тип даних	Обмеження	Опис
id	UUID	PK, NOT NULL	Унікальний ідентифікатор користувача
name	VARCHAR(255)	NOT NULL	Ім'я користувача
role	VARCHAR(100)	NOT NULL	Роль в системі (розробник, тімлід тощо)

Сутність «Користувач» є основною в системі, вона моделює осіб, які взаємодіють із додатком. Кожному користувачу присвоюється унікальний UUID, який використовується як первинний ключ. Поле name дозволяє зберегти повне ім'я або псевдонім, необхідний для персоналізації інтерфейсу. Атрибут role визначає набір прав і функціонал, доступний конкретному користувачу, забезпечуючи різні рівні доступу (наприклад, адмін, користувач, гість).

Опис таблиці «Category» представлений на таблиці 2.5

Таблиця 2.5 – Опис таблиці «Category»

Назва поля	Тип даних	Обмеження	Опис
id	UUID	PK, NOT NULL	Унікальний ідентифікатор категорії
name	VARCHAR(100)	NOT NULL, UNIQUE	Назва категорії (технічна, особиста)

Сутність «Категорія» використовується для класифікації задач на логічні групи за контекстом чи тематикою. Поле id є унікальним ідентифікатором категорії, що гарантує цілісність посилань. Атрибут name задає читабельне позначення категорії, яке відображається користувачу при створенні та фільтрації задач. Унікальність назви дозволяє уникати дублювання та забезпечує однозначність класифікації.

Опис таблиці «Task» представлений на таблиці 2.6

Таблиця 2.6 – Опис таблиці "Task"

Назва поля	Тип даних	Обмеження	Опис
id	UUID	PK, NOT NULL	Унікальний ідентифікатор задачі
name	VARCHAR(255)	NOT NULL	Назва задачі
description	TEXT	NULLABLE	Опис задачі
deadline	TIMESTAMP WITH TIME ZONE	NULLABLE	Дата та час дедлайну
status	VARCHAR(50)	NOT NULL, DEFAULT 'todo'	Статус (todo, in_progress, done)

user_id	UUID	FK → User.id, NOT NULL	Ідентифікатор користувача
project_id	UUID	FK → Project.id, NULLABLE	Ідентифікатор проєкту
category_id	UUID	FK → Category.id, NULLABLE	Ідентифікатор категорії
created_at	TIMESTAMP WITH TIME ZONE	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Час створення
updated_at	TIMESTAMP WITH TIME ZONE	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Час останнього оновлення
completed_at	TIMESTAMP WITH TIME ZONE	NULLABLE	Час виконання
parent_task_id	UUID	FK → Task.id, NULLABLE	Батьківська задача

Сутність «Задача» є ключовою складовою системи, вона описує одиничну роботу, що має бути виконана. Поле id служить унікальним ідентифікатором задачі, який використовується для зв'язків із іншими сутностями. Атрибути name, description та deadline визначають базові характеристики задачі. Поля priority і status дозволяють вказати важливість та поточний стан, що необхідно для коректного планування та моніторингу.

Опис таблиці «Project» представлений на таблиці 2.7

Таблиця 2.7 – Опис таблиці "Project"

Назва поля	Тип даних	Обмеження	Опис
id	UUID	PK, NOT NULL	Унікальний ідентифікатор проєкту
name	VARCHAR(255)	NOT NULL, UNIQUE	Назва проєкту

Сутність «Проект» об'єднує задачі в логічні групи для централізованого управління та звітності. Поле id дозволяє однозначно ідентифікувати проєкт у системі. Атрибут name задає просту назву проєкту, яку відображають у списках та заголовках. Використовуючи цей об'єкт, можна зв'язати задачі, відстежувати їх прогрес у межах конкретного проєкту та генерувати агреговані звіти.

Опис таблиці «Pomodoro» представлений на таблиці 2.8

Таблиця 2.8 – Опис таблиці "Pomodoro"

Назва поля	Тип даних	Обмеження	Опис
id	UUID	PK, NOT NULL	Ідентифікатор сесії Pomodoro
task_id	UUID	FK → Task.id, NOT NULL	Ідентифікатор задачі
date	TIMESTAMP WITH TIME ZONE	NOT NULL	Дата та час початку сесії
duration	INTEGER	NOT NULL	Тривалість у хвилинах

Сутність «Pomodoro» використовується для фіксації інтервалів роботи за методикою Pomodoro. Кожна сесія ідентифікується унікальним id. Поле task_id пов'язує сесію з конкретною задачею, дозволяючи контролювати час

виконання. Атрибути date та duration забезпечують інформацію про часові рамки кожного інтервалу.

Опис таблиці «History» представлений на таблиці 2.9

Таблиця 2.9 – Опис таблиці «History»

Назва поля	Тип даних	Обмеження	Опис
id	UUID	PK, NOT NULL	Ідентифікатор запису історії
task_id	UUID	FK → Task.id, NOT NULL	Ідентифікатор задачі
date	TIMESTAMP WITH TIME ZONE	NOT NULL	Дата та час зміни
action	VARCHAR(255)	NOT NULL	Опис дії (створено, оновлено)

Сутність «History» відстежує всі зміни задач із вказівкою дати та описом дій. Поле id є унікальним ключем, що дозволяє ідентифікувати запис. Атрибут task_id пов'язує запис історії з відповідною задачею. Поле action описує дію (створення, оновлення, завершення) для аудит-трейсу.

Опис таблиці «Reminder» представлений на таблиці 2.10

Таблиця 2.10 – Опис таблиці "Reminder"

Назва поля	Тип даних	Обмеження	Опис
id	UUID	PK, NOT NULL	Ідентифікатор нагадування
task_id	UUID	FK → Task.id, NOT NULL	Ідентифікатор задачі
time	TIMESTAMP WITH TIME ZONE	NOT NULL	Час надсилання нагадування

type	VARCHAR(50)	NOT NULL	Тип (push, in-app)
status	VARCHAR(50)	NOT NULL, DEFAULT 'pending'	Статус (pending, sent, cancelled)

Сутність «Reminder» забезпечує автоматичне надсилення сповіщень користувачу. Кожне нагадування має унікальний id. Поле task_id встановлює зв'язок з відповідною задачею. Атрибути time, type та status визначають час, метод сповіщення та поточний стан надсилення.

Опис таблиці «Recurrence» представлений на таблиці 2.11

Таблиця 2.11 – Опис таблиці "Recurrence"

Назва поля	Тип даних	Обмеження	Опис
id	UUID	PK, NOT NULL	Ідентифікатор правила повторення
task_id	UUID	FK → Task.id, NOT NULL	Ідентифікатор задачі
recurrence_type	VARCHAR(50)	NOT NULL	Тип повторення (daily, weekly)
next_date	DATE	NOT NULL	Дата наступного повторення

Сутність «Recurrence» використовується для визначення правил повторення задач. Поле id є первинним ключем запису. Атрибут task_id зв'язує правило з задачею. Поля recurrence_type та next_date визначають (наприклад, щодня, щотижня) та дату наступного створення задачі.

Опис таблиці «Tag» представлений на таблиці 2.12

Таблиця 2.12 – Опис таблиці "Tag"

Назва поля	Тип даних	Обмеження	Опис
id	UUID	PK, NOT NULL	Ідентифікатор тега
name	VARCHAR(100)	NOT NULL, UNIQUE	Назва тега

Сутність «Tag» призначена для гнучкої класифікації задач за довільними мітками. Поле id унікально ідентифікує тег. Атрибут name встановлює назву тега, що використовується в інтерфейсі. Таким чином, користувачі можуть маркувати задачі для подальшого фільтрування.

Опис таблиці «Tag_Task» представлений на таблиці 2.13

Таблиця 2.13 – Опис таблиці "Tag_Task"

Назва поля	Тип даних	Обмеження	Опис
task_id	UUID	FK → Task.id, NOT NULL	Ідентифікатор задачі
tag_id	UUID	FK → Tag.id, NOT NULL	Ідентифікатор тега

Сутність «Tag_Task» реалізує зв'язок багато-до-багатьох між задачами та тегами. Поля task_id та tag_id є складовими первинного ключа. Таким чином, кожна комбінація задачі та тега зберігається окремо. Це забезпечує можливість багатократного позначення однієї і тієї ж задачі різними тегами.

Фізична модель даних передбачає зберігання цих таблиць у PostgreSQL з чітко визначеними PK, FK, обмеженнями та індексами для забезпечення цілісності й продуктивності.

2.3. Рішення з математичного забезпечення

Математичне забезпечення розробки автоматизованих систем або її елементів є сукупністю засобів і методів для побудови моделей проєктованих об'єктів і процесів управління. У сучасній практиці комп'ютерне моделювання застосовується для вирішення широкого спектра задач у різних галузях, зокрема при автоматизації планування, моніторингу та оптимізації робочих процесів.

Алгоритм рішення задач формалізує послідовність операцій від обробки вхідних даних до формування вихідного результату. Приклад структури алгоритму:

1. Ініціалізація:

- Завантажити вхідні параметри: тривалості задач d_i , дедлайни D_i .
- Установити початкові значення змінних: $s_i = 0, x_i = 0$.

2. Основний розрахунок:

- Для кожної задачі обчислити можливий час початку s_i за умови $s_i + d_i \leq D_i$.
- Визначити бінарну змінну $x_i = 1$, якщо задача виконується, інакше $x_i = 0$.

3. Оцінка цільової функції: $F = \sum_{i=1}^n w_i x_i$.

4. Контроль достовірності обчислень:

- Перевірка співвідношення $s_i + d_i \leq D_i + M(1 - x_i)$, де M — велике константне число.
- Обчислення контрольно-діагностичних співвідношень, наприклад: $\sum_i x_i \leq N_{max}$, де N_{max} — максимально допустима кількість одночасних задач.

5. Формування вихідної звітності:

- Результати сортуються за зростанням ідентифікаторів id_i .
- Дані групуються за проєктами та категоріями.

У алгоритмі використано умовні позначення реквізитів та сигналів, графічні і табличні блоки з посиланням на масиви даних. За необхідності уточнюється точність обчислень (наприклад,).

У якості методу моделювання вибрано дискретне ітераційне моделювання з кроком Δt , що дозволяє:

- Відстежувати стан системи у дискретні моменти $t_k = k\Delta t$.
- Реалізувати модель у вигляді матриці переходів: $S(t_k + 1) = f(S(t_k), U(t_k))$, де S_t — вектор станів, U_t — вектор керуючих сигналів.
- Обробляти події пакетами, забезпечуючи асинхронність та реактивність.

Переваги цього методу:

- Простота реалізації в програмному коді або блок-схемі;
- Гнучкість у зміні структури моделі;
- Здатність включати перевірочні співвідношення для контролю коректності.

Недоліки:

- Залежність точності від обраного кроку Δt ;
- Можлива висока обчислювальна вартість при дрібних кроках.

Процес побудови математичної моделі:

1. Формалізація предметної області — виділення ключових об'єктів (задачі, ресурси, користувачі) та визначення їх атрибутів.
2. Визначення реквізитів і сигналів — складання переліку параметрів d_i , D_i , керуючих сигналів та умов їх активації.
3. Запис математичних формул з уніфікованими позначеннями, використаними в інших розділах (наприклад, для опису сутностей бази даних).
4. Верифікація моделі — перевірка контрольних співвідношень, щоб виявити помилки на етапі розрахунків: $\sum_i x_i d_i \leq T_{max}$, де T_{max} — загальний часовий ресурс.
5. Документування — фіксація алгоритму, формул та інструкцій з організації вихідних документів.

2.4. Рішення щодо програмного забезпечення

Визначення оптимального набору технологій представляє собою основоположний етап у процесі розробки мобільного застосунку, що здійснює критичний вплив на продуктивність розробки, функціональні можливості, масштабованість та підтримуваність кінцевого продукту. Технологічний стек для мобільного застосунку тайм-менеджменту формувався виходячи з комплексного аналізу вимог до системи, сучасних тенденцій у розробці застосунків для платформи Android та необхідності забезпечення стійкої працездатності в корпоративному середовищі ІТ-компаній.

Фундаментальним вибором для розробки застосунку стала платформа Android — найпоширеніша мобільна операційна система у світі, що забезпечує широке охоплення потенційних користувачів. Перевага даної платформи обґрунтовується її відкритістю, наявністю потужного SDK (Software Development Kit) та зрілої екосистеми інструментів розробки. Контекст корпоративного середовища, де пристрої на базі Android часто

використовуються як основні робочі інструменти, також підтверджує доцільність даного вибору.

Мова програмування Java обрана як основний інструмент розробки, що відповідає вимогам дипломного завдання та представляє собою одну з офіційно підтримуваних мов для Android-розробки. Java характеризується об'єктно-орієнтованою парадигмою, сильною типізацією, багатою стандартною бібліотекою та величезною екосистемою зовнішніх фреймворків та інструментів. Незважаючи на зростаючу популярність Kotlin як рекомендованої Google мови для Android-розробки, Java залишається повністю підтримуваною платформою та має ряд переваг, включаючи більший розмір спільноти, значну кількість документації та прикладів, а також широку розповсюдженість у корпоративному секторі. Для проекту планується використання сучасної версії JDK (Java Development Kit) 11 або вище, що забезпечує доступ до новітніх мовних можливостей при збереженні сумісності з платформою Android.

Середовище розробки Android Studio вибрано як основний інструмент для написання, тестування та збірки застосунку. Дане IDE (Integrated Development Environment) є офіційно рекомендованим Google для Android-розробки та надає інтегровані інструменти для написання та аналізу коду, проектування інтерфейсів, зневадження, профілювання продуктивності, тестування та підготовки застосунків до публікації. Android Studio базується на потужній платформі IntelliJ IDEA, що забезпечує широкі можливості для рефакторингу, автозавершення коду та навігації по проекту, що суттєво підвищує продуктивність розробки.

Система збірки Gradle обрана як стандартний та найбільш підтримуваний інструмент для автоматизації процесу складання Android-застосунків. Gradle забезпечує гнучке управління залежностями, налаштування процесу збірки та створення різних варіантів застосунку для різних середовищ (розробка, тестування, продакшн). Декларативний підхід

до опису процесу збірки за допомогою DSL (Domain Specific Language) на основі Groovy або Kotlin дозволяє створювати ефективні та підтримувані скрипти збірки, що включають всі необхідні етапи компіляції, упаковки та оптимізації застосунку.

Таблиця 2.14 – Обраний технологічний стек та його обґрунтування

Компонент стеку	Обране рішення	Обґрунтування вибору
Платформа	Android	Широка розповсюдженість, відкритість, корпоративне використання
Мова програмування	Java (JDK 11+)	Вимога завдання, зрілість, велика екосистема, підтримка Android Studio
IDE	Android Studio	Офіційне IDE, інтегровані інструменти для розробки під Android
Система збірки	Gradle	Стандарт для Android, гнучке управління залежностями
Архітектурний патерн	MVVM	Розділення відповідальності, тестованість, інтеграція з Jetpack
Управління станом UI	Jetpack ViewModel, Jetpack LiveData	Збереження стану при змінах конфігурації, реактивне оновлення UI, врахування життєвого циклу
Локальна БД	SQLite + Jetpack Room	Стандарт для Android, надійність, зручний ORM, перевірка запитів на етапі компіляції
Навігація	Jetpack Navigation Component	Спрощення навігації, візуальний редактор, обробка deep links
Фонові роботи	Jetpack WorkManager	Надійне виконання відкладених завдань (синхронізація, сповіщення)

Мережева взаємодія	Retrofit + OkHttp	Ефективний тип-безпечний HTTP-клієнт для REST API
Dependency Injection	Hilt	Спрощена реалізація DI для Android, інтеграція з Jetpack
Календар	Material CalendarView (Applandeo)	Material Design, гнучкість кастомізації, підтримка подій, діапазонів, Java-прикладі
Сповіщення	Android Notification API (NotificationCompat) + AlarmManager/WorkManager	Стандартний API, сумісність, гнучкість налаштувань, планування
Тестування	JUnit, Mockito, Espresso	Стандартні інструменти для юніт-, інтеграційного та UI-тестування

Архітектурний патерн Model-View-ViewModel (MVVM) обрано як основу для організації коду застосунку, що відповідає рекомендаціям Google щодо розробки сучасних Android-застосунків. Цей патерн забезпечує чітке розділення відповідальності між різними компонентами системи: представленням (View), що відповідає за відображення інтерфейсу користувача; моделлю представлення (ViewModel), що містить логіку представлення та підготовку даних для відображення; та моделлю (Model), що представляє бізнес-логіку та дані застосунку. Перевагами MVVM є підвищена тестованість коду завдяки відсутності прямих залежностей між ViewModel та View, стійкість до змін конфігурації пристрою, а також гармонійна інтеграція з компонентами Android Jetpack.

Для ефективної реалізації архітектури MVVM та забезпечення інших ключових аспектів функціональності застосунку обрано набір бібліотек Android Jetpack – колекції бібліотек, інструментів та рекомендацій від Google, спрямованих на спрощення розробки якісних Android-застосунків. Компонент ViewModel з Jetpack забезпечує збереження даних

представлення при змінах конфігурації пристрою, таких як поворот екрану, перехід у мультивіконний режим або зміна мови системи. LiveData, також з набору Jetpack, надає можливість створення спостережуваних потоків даних, що враховують життєвий цикл компонентів View та забезпечують автоматичне оновлення інтерфейсу при зміні даних.

Для роботи з локальною базою даних обрано SQLite як стандартну та найбільш підтримувану реляційну СУБД для платформи Android. Для спрощення взаємодії з SQLite та реалізації об'єктно-реляційного відображення використовується бібліотека Room з набору Jetpack, що надає зручний API для визначення схеми бази даних, виконання запитів та отримання результатів у вигляді Java-об'єктів. Room забезпечує перевірку SQL-запитів на етапі компіляції, інтеграцію з LiveData для реактивного оновлення UI при зміні даних у базі та підтримку асинхронних операцій без блокування головного потоку.

Для організації навігації між екранами застосунку обрано Jetpack Navigation Component, що спрощує реалізацію переходів між фрагментами, передачу параметрів та управління back stack. Ця бібліотека надає візуальний редактор для проектування навігаційних графів, типобезпечну передачу аргументів між екранами та підтримку deep links для можливості переходу до конкретних екранів застосунку з зовнішніх джерел.

Для виконання фонових задач, таких як синхронізація даних з сервером або планування сповіщень, обрано Jetpack WorkManager. Ця бібліотека забезпечує надійне виконання відкладених та періодичних задач з урахуванням обмежень системи та стану пристрою. WorkManager автоматично обирає оптимальний спосіб виконання задачі (JobScheduler, AlarmManager або Firebase JobDispatcher) залежно від версії Android та доступних сервісів, що спрощує розробку та підвищує сумісність з різними пристроями.

Для мережевої взаємодії з серверним API обрано комбінацію бібліотек Retrofit та OkHttp. Retrofit надає декларативний підхід до опису REST API через Java-інтерфейси та автоматизує процес серіалізації/десеріалізації даних, обробку запитів та перетворення відповідей. OkHttp, як низькорівневий HTTP-клієнт, забезпечує ефективне управління з'єднаннями, кешування та обробку помилок. Разом ці бібліотеки надають потужний та гнучкий інструментарій для взаємодії з RESTful API, необхідний для синхронізації даних застосунку з центральним сервером.

Для впровадження залежностей (Dependency Injection) обрано Hilt – бібліотеку, побудовану на основі Dagger 2 та спеціально адаптовану для Android. Hilt спрощує процес впровадження залежностей через автоматичну генерацію компонентів, прив'язку до життєвого циклу Android-компонентів та інтеграцію з іншими бібліотеками Jetpack. Використання DI через Hilt дозволяє зменшити зв'язаність компонентів, підвищити тестованість коду та спростити управління залежностями у масштабі всього застосунку.

Для реалізації календарного інтерфейсу обрано бібліотеку Material CalendarView від Applandeo, що пропонує гнучкі можливості для кастомізації, підтримку відображення подій на календарі, вибір діапазонів дат та реалізацію відповідно до принципів Material Design. Ця бібліотека надає детальну документацію та приклади використання на Java, що спрощує її інтеграцію в проект.

Для сповіщень користувача про наближення термінів виконання завдань та інші події обрано стандартний Android Notification API через бібліотеку сумісності NotificationCompat, що забезпечує створення сповіщень, сумісних з різними версіями Android. Для планування сповіщень використовується комбінація AlarmManager для точних за часом нагадувань та WorkManager для гнучкого планування з урахуванням стану пристрою та системних обмежень.

Для тестування застосунку обрано стандартний набір інструментів, що включає JUnit для юніт-тестування окремих компонентів, Mockito для створення мок-об'єктів та симуляції поведінки залежностей при тестуванні, та Espresso для автоматизованого тестування користувацького інтерфейсу. Ця комбінація забезпечує можливість тестування на різних рівнях, від окремих класів до комплексних сценаріїв взаємодії користувача з застосунком.

При виборі бібліотек та компонентів технологічного стеку приділялася особлива увага балансу між функціональністю та впливом на розмір і продуктивність застосунку. Використання рекомендованих Google бібліотек Jetpack забезпечує найкращу сумісність з платформою та слідування сучасним практикам розробки. Для специфічних функціональних можливостей, таких як календарний інтерфейс, обрані перевірені рішення з активною підтримкою та детальною документацією. Загалом, обраний технологічний стек створює надійну основу для розробки ефективного, масштабованого та підтримуваного мобільного застосунку для тайм-менеджменту, що відповідає сучасним стандартам Android-розробки та специфічним потребам ІТ-компаній.

2.5. Рішення щодо технічного забезпечення

Для забезпечення роботи автоматизованої системи тайм-менеджменту необхідне обладнання для розробки, розгортання та користування додатком:

Розробницьке середовище:

- Персональний комп'ютер класу Intel i5/i7 або еквівалент із чотириядерним процесором;
- Оперативна пам'ять не менше 16 ГБ;
- SSD-диск об'ємом не менше 256 ГБ;

- Монітор, клавіатура, маніпулятор (миша/трекпад).
- Серверна інфраструктура (якщо застосовується):
- Віртуальний або фізичний сервер під управлінням Linux (Ubuntu 20.04 LTS);
- Процесор не менше 4 ядер;
- Оперативна пам'ять не менше 8 ГБ;
- Дисковий простір SSD від 100 ГБ;
- Підключення до мережі з мінімальною пропускну здатністю 100 Мбіт/с.

Мережеве обладнання:

- Wi-Fi маршрутизатор з підтримкою стандартів IEEE 802.11 b/g/n/ac, DHCP;
- Ethernet-кабель (Cat5e або вище) для підключення розробницької платформи до мережі;
- При необхідності VPN-сервер для захищеного доступу до внутрішніх сервісів.

Додаткові пристрої:

- Принтер (USB або мережевий) для друку звітної документації;
- Сканер документів для оцифрування необхідних паперових форм;
- USB-кабель для підключення мобільних пристроїв до комп'ютера під час налагодження.

Носії даних для розробницького та серверного зберігання можуть використовуватися двох основних типів — HDD (жорсткі диски) та SSD (твердотільні накопичувачі). HDD базуються на магнітних дисках, що обертаються, а SSD використовують флеш-пам'ять NAND без рухомих частин.

Переваги SSD: висока швидкість послідовного та довільного читання/запису, мінімальні затримки при доступі, стійкість до вібрацій і температурних коливань. SSD-носії забезпечують прискорене завантаження ОС і додатків, що особливо важливо під час розробки та тестування. Типові ємності SSD варіюються від 120 ГБ до 8 ТБ, при цьому вартість за гігабайт поступово знижується.

Переваги HDD: нижча ціна за гігабайт, велика доступна ємність (до 16 ТБ і більше), відпрацьована технологія з доведеною надійністю в умовах тривалих записів. Проте HDD мають повільніші операції довільного доступу та споживають більше енергії під час обертання шпинделя.

SSD мають вищу надійність завдяки відсутності механічних елементів та індикації циклів програмування-стирання (P/E). Типи NAND (SLC, MLC, TLC, QLC) відрізняються щільністю зберігання та витривалістю: SLC забезпечує 100 000+ циклів P/E, MLC — до 10 000, TLC — 3 000–5 000, QLC — 1 000–2 000. Енергоспоживання SSD в режимі очікування близько 0,1–0,5 Вт, що знижує загальні витрати на експлуатацію.

HDD у активному режимі можуть споживати 5–10 Вт, а в режимі очікування 1–3 Вт. Як результат, при великому парку серверів SSD дозволяють зменшити експлуатаційні витрати та теплове навантаження на ЦОД.

Для ОС, середовища розробки та баз даних рекомендується використовувати SSD ємністю від 256 ГБ, щоб забезпечити швидкий старт та високу продуктивність I/O.

Для зберігання архівів та резервних копій вибрати HDD ємністю від 2 ТБ, оскільки вони забезпечують економічне зберігання великих обсягів даних.

Якщо бюджет дозволяє, застосовувати підсистему NVMe SSD для критичних завдань із високою інтенсивністю операцій запису/читання.

Мобільні гаджети є ключовим каналом доступу до системи тайм-менеджменту в офісних або віддалених умовах. Сучасні смартфони та планшети працюють на базі ARM-процесорів (Cortex-A серії, Apple A-серії тощо), що забезпечують високий рівень багато- та одноядерної продуктивності при оптимальному рівні енергоспоживання. Більшість пристроїв оснащені не менше ніж 3–6 ГБ оперативної пам'яті LPDDR4(X) для безперебійної роботи мультизадачності.

Екран і інтерфейс: Рекомендовано сенсорні дисплеї IPS або OLED з роздільною здатністю щонайменше HD (720×1280) або FHD (1080×1920) для чіткого відображення інтерфейсу. Список multitouch підтримує мінімум 5 торкань одночасно, що дозволяє реалізувати жестові команди у додатку.

Зберігання даних: Вбудований накопичувач eMMC 5.1 або UFS 2.x об'ємом від 32 ГБ забезпечує достатній простір для кешування локальних даних, офлайн-режимів та журналу Pomodoro-сесій. Для вимогливих користувачів рекомендується підтримка microSD-карт класу UHS-I для розширення сховища.

Комунікаційні модулі: GSM/3G/4G LTE модулі з підтримкою частот 800/1800/2600 МГц гарантують стабільне з'єднання із сервером. Wi-Fi 802.11 a/b/g/n/ac забезпечує швидку передачу великих обсягів даних під час синхронізації. Bluetooth 5.0 дозволяє інтеграцію з носимими пристроями (розумні годинники, фітнес-браслети) для розширення функціоналу додатку.

Сенсори та функції: Пристрої повинні мати акселерометр, гіроскоп та датчик освітленості для адаптації інтерфейсу під навколишні умови. Геолокація (GPS, GLONASS) допомагає відстежувати місце виконання задач і автофіксувати контекст активності.

Живлення та автономність: Акумулятори Li-Ion/Li-Po ємністю не менше 3000 мА·год забезпечують роботу додатку протягом одного

робочого дня (8–10 годин активного використання). Підтримка швидкої зарядки (Quick Charge, Power Delivery) рекомендована для мінімізації простою.

Безпека: Використання біометричної автентифікації (відбиток пальця, розпізнавання обличчя) та апаратного шифрування сховища (AES-256) підвищують рівень захисту конфіденційних даних користувача.

Висновок за розділом 2: у цьому розділі сформовано комплексні рішення для майбутньої системи тайм-менеджменту: визначені ролі й сценарії роботи користувачів, побудовані концептуальну та фізичну модель даних, розроблено математичні алгоритми, обрано оптимальний стек програмних компонентів і докладно описані вимоги до технічного обладнання. Отримані напрацювання закладають міцну основу для подальшої реалізації гнучкого, надійного та масштабованого мобільного додатку.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ТАЙМ-МЕНЕДЖМЕНТУ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІТ-КОМПАНІЇ

Проектування користувацького інтерфейсу та досвіду взаємодії представляє собою визначальний фактор у успішній імплементації мобільних застосунків, при цьому для інструментів тайм-менеджменту даний аспект набуває особливої значущості з огляду на специфіку завдань, пов'язаних з організацією часу та управлінням активностями.

Мінімалізм та візуальна ясність складають ключовий принцип проектування, спрямований на зниження когнітивного навантаження та підвищення ефективності взаємодії з застосунком. Інтерфейс максимально звільняється від надлишкових елементів, що потенційно відволікають увагу користувача, з фокусом на забезпеченні чіткої навігаційної структури, однозначної ідентифікації інтерактивних елементів та оптимальної щільності інформації на екрані.

Візуальна ієрархія елементів інтерфейсу, що формується через управління розміром, кольором, контрастністю та просторовим розташуванням, спрямовує увагу користувача та полегшує навігацію в інформаційному просторі застосунку. Структурування контенту з виділенням пріоритетних елементів дозволяє користувачеві швидко орієнтуватися та знаходити потрібну інформацію без необхідності послідовного сканування всього екрану. Для застосунку тайм-менеджменту візуальна ієрархія набуває додаткової значущості в контексті відображення пріоритетів завдань, термінів виконання та статусів, де візуальне кодування може суттєво прискорити обробку інформації.

Формування структури інтерфейсу починається зі створення низькодеталізованих каркасів (wireframes) основних екранів, що визначають розташування функціональних блоків, навігаційних елементів та інформаційних компонентів. Базовий набір екранів включає головний екран зі списком завдань або дашбордом, екрани створення та редагування

завдань, календарні представлення в різних масштабах (день, тиждень, місяць), екрани проектів та налаштувань застосунку.

Для мобільного застосунку тайм-менеджменту актуальним є застосування певних UI/UX патернів, що зарекомендували себе в даній предметній області. Навігаційна структура реалізується через нижню панель навігації (Bottom Navigation Bar), що забезпечує швидкий доступ до основних розділів застосунку та відповідає сучасним рекомендаціям з проектування мобільних інтерфейсів. Додаткові механізми навігації включають системи вкладок для перемикання між режимами відображення в календарі (день, тиждень, місяць) та фільтри для різних представлень списку завдань.



Рисунок 3.1 . Нижня панель навігації у застосунку

Відображення завдань організовується переважно у вигляді списків з використанням компонента RecyclerView, що забезпечує ефективну візуалізацію великих наборів даних з можливістю кастомізації представлення окремих елементів. Для швидкого доступу до функції створення нового завдання застосовується патерн Плаваючої кнопки дії (Floating Action Button), позиціонованої в нижній частині екрана та візуально виділеної для привернення уваги.

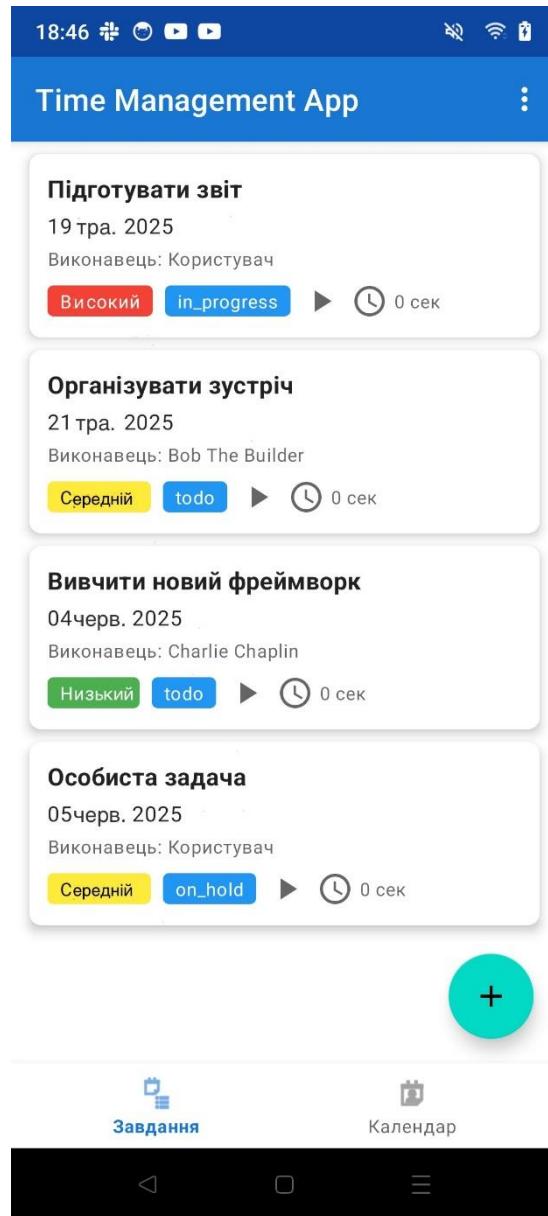


Рисунок 3.2 . Головний екран застосунку

Календарний інтерфейс реалізується із застосуванням стандартних представлень (місяць, тиждень, день) з можливістю переходу до детальної інформації про завдання на вибрану дату.

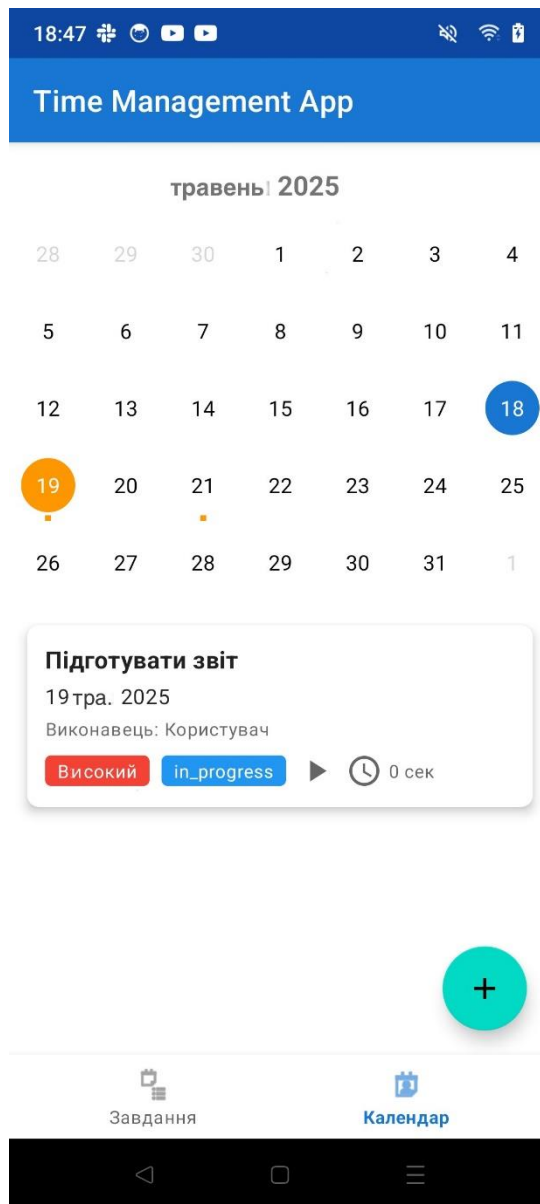


Рисунок 3.3 . Календарний інтерфейс

Система візуальних індикаторів, що включає кольорове кодування пріоритетів, відмітки для завдань з наближенням термінів та індикатори завантаженості днів у календарі, підвищує інформативність інтерфейсу та дозволяє користувачеві швидко оцінювати ситуацію без необхідності детального аналізу текстової інформації. Реалізація інтерфейсу редагування завдань включає стандартні форми введення з підтримкою валідації даних, випадаючі списки для вибору проектів та виконавців, календарні селектори для встановлення термінів та систему управління відстеженням часу з кнопками запуску, паузи та зупинки таймера.

18:47

←

Редагувати задачу

Назва завдання

Назва завдання

Підготувати звіт

Опис завдання

Зібрати дані та підготувати щомісячний звіт для керівництва

Проект:

Робота

▼

Виконавець:

Користувач

▼

Дата виконання

19 тра. 2025

ЗМІНИТИ

Завдання

Календар

Рисунок 3.4 . Екран редагування задачі (1)

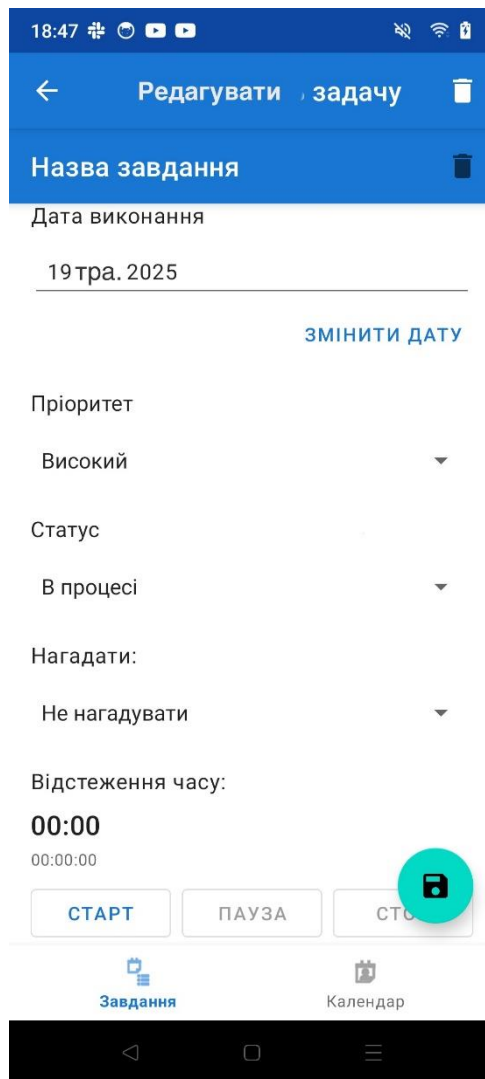


Рисунок 3.5 . Екран редагування задачі (2)

Таким чином, проектування користувацького інтерфейсу та досвіду взаємодії для мобільного застосунку тайм-менеджменту базується на врахуванні специфічних потреб ІТ-фахівців, застосуванні сучасних принципів дизайну та використанні перевірених UI/UX патернів, що забезпечує створення інтуїтивно зрозумілого, ефективного та естетично привабливого продукту.

Висновок за розділом 3: у цьому розділі визначено ключові принципи UI/UX — мінімалізм, візуальна ієрархія та чітка навігація — і на їхній основі створено каркаси та макети основних екранів із використанням перевірених патернів (Bottom Navigation Bar, RecyclerView, FAB, календар), а також впроваджено кольорове кодування пріоритетів і термінів. Отримані рішення

гарантують інтуїтивно зрозумілий, ефективний та привабливий інтерфейс мобільного додатку.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ ТА ТЕХНІКА БЕЗПЕКИ

Мобільний застосунок для тайм-менеджменту призначено для використання у межах офісних та віддалених умов праці співробітників ІТ-компаній, зокрема — програмістів, тестувальників, тімлідів і HR-фахівців. Використання програми не передбачає підвищеної фізичної активності або спеціалізованого обладнання, однак вимагає дотримання загальних правил безпечної організації робочого місця при роботі з комп'ютерною технікою.

Умови експлуатації застосунку повинні відповідати нормам охорони праці, а саме — вимогам щодо ергономіки, мікроклімату, освітлення, електробезпеки та протипожежної безпеки. Робоче місце користувача має бути облаштоване так, щоб мінімізувати навантаження на зорову, нервову та опорно-рухову системи. Застосунок не потребує використання додаткових фізичних пристроїв (окрім стандартного смартфона або ПК), однак довготривале використання пристрою потребує періодичних перерв відповідно до рекомендацій ДСанПіН 3.3.2.007-98.

При використанні застосунку в офісних умовах рекомендовано дотримуватись таких параметрів:

- температура повітря — 20–24°C;
- вологість — 40–60%;
- рівень освітленості — не менше 500 лк;
- відстань від очей до екрана — 60–70 см;
- тривалість безперервної роботи — не більше 60 хвилин поспіль з наступною перервою 5–10 хвилин.

Освітлення повинно бути рівномірним, без засліплюючих блисків. Джерела світла бажано розташовувати збоку від робочого місця. Перевагу слід надавати лампам з нейтральною колірною температурою (4000–5000

К). Також рекомендується використання моніторів із захисним покриттям та налаштуванням яскравості відповідно до умов приміщення.

Крісло користувача повинно мати регулювання по висоті, підтримку попереку та підлокітники. Монітор розміщується на рівні очей, клавіатура — на відстані 10–15 см від краю столу. При дотриманні цих рекомендацій знижується ризик професійних захворювань, зокрема остеохондрозу, зорового напруження та карпального тунельного синдрому.

Для віддалених користувачів важливо забезпечити такі ж умови вдома — або максимально наближені до них. При тривалому використанні гаджетів (смартфонів або планшетів) рекомендовано застосовувати підставки, фільтри від блиску, а також періодично змінювати положення тіла. Робоче місце вдома повинно мати природне освітлення, або штучне — із достатньою освітленістю.

Якщо застосунок використовується зі смартфона, потрібно дотримуватись рекомендованої відстані до екрана (30–40 см), змінювати фокус зору, уникати положення лежачи або зігнутої спини. Для покращення продуктивності рекомендується використовувати бездротову клавіатуру і мишу при роботі з планшетом або ноутбуком.

Ергономіка має важливе значення для збереження працездатності користувача. Стіл повинен бути достатньо широким для розміщення обладнання та документів. Висота столу має становити 72–75 см. Висота сидіння крісла повинна забезпечувати кут у 90° у колінних і ліктьових суглобах. Спина повинна мати опору. Мишка розміщується поруч із клавіатурою.

Мікроклімат — ще один важливий аспект. Робоче приміщення повинно регулярно провітрюватись. Рекомендоване застосування зволожувачів або кондиціонерів. Температура — не нижче 20°C взимку і не вище 24°C влітку. Рівень шуму не повинен перевищувати 50 дБА.

Важливо запровадити раціональний режим праці та відпочинку. Рекомендується використовувати техніку Pomodoro, що включає 25 хвилин активної роботи і 5 хвилин перерви. Через кожні 2 години варто робити довшу перерву — до 15 хвилин. Протягом цієї перерви бажано виконати вправи для очей, шиї, рук.

Візуальна гімнастика: моргання, фокусування погляду на віддалених об'єктах, масаж повік. Фізична активність: кругові рухи плечима, нахили голови, обертання зап'ястями, розтягування ніг. Також ефективна коротка прогулянка або зміна активності (наприклад, читання документації).

При використанні застосунку необхідно дотримуватись базових правил електробезпеки. Усі пристрої мають бути справними, із заводською ізоляцією. Заборонено користуватись несправними зарядними пристроями, кабелями із пошкодженою ізоляцією, підключати смартфон до розетки в умовах підвищеної вологості.

Розетки повинні бути заземленими. Не рекомендується використовувати потрійники, саморобні подовжувачі. Під час грози рекомендується вимикати електроніку з мережі. Робоче місце має бути вільним від рідин, займистих матеріалів та зайвих проводів.

Умови експлуатації застосунку не створюють підвищеної пожежної небезпеки, однак користувач повинен дотримуватись загальних вимог протипожежної безпеки. Не можна залишати увімкнене обладнання без нагляду, особливо при зарядці.

В офісному середовищі приміщення мають бути обладнані вогнегасниками. Поблизу робочого місця не повинно бути легкозаймистих матеріалів. Всі електроприлади слід вимикати після завершення роботи. Шляхи евакуації повинні бути вільними. Для домашнього використання бажано мати мінімальний набір засобів пожежогасіння — хоча б вогнегасник типу ВСЕ-2 або пожежну ковдру.

У разі пожежі або аварійної ситуації користувач повинен припинити використання пристрою, відключити електроживлення (в офісі — за допомогою ПЗВ, вдома — з розетки), повідомити відповідальні служби. У разі повітряної тривоги — негайно пройти до укриття.

Важливо бути ознайомленим із планом евакуації, мати доступ до аптечки першої допомоги, знати базові правила дій у разі травм, опіків, отруєнь, перегріву. Необхідно зберігати спокій, уникати паніки і діяти відповідно до інструкцій органів цивільного захисту.

Робоче місце користувача (наприклад, розробника або тестувальника) повинно включати:

- Стіл шириною не менше 120 см із матовою поверхнею;
- Ергономічне крісло з регулюванням висоти, глибини та нахилу спинки;
- Підставку під ноги (при необхідності);
- Освітлення — настільна лампа з регулюванням напрямку світла;
- Монітор на підставці або піднятий до рівня очей;
- Бездротову мишу та клавіатуру;
- Органайзер для документів, блокнотів та канцелярії.

Усі елементи мають бути розміщені так, щоби не викликати перенапруження м'язів, уникати скручування тулуба або нахилу голови вниз. Перевага надається природним матеріалам, належній вентиляції й добрій акустиці приміщення.

Крім фізичного комфорту, важливо враховувати й психологічну безпеку. Постійне використання тайм-менеджменту, дедлайни, віддалена робота можуть призводити до хронічної втоми або емоційного вигорання. З цією метою:

- Рекомендується впроваджувати внутрішню систему перерв, нагадування про відпочинок;
- Керівництво компанії має надавати доступ до психологічної підтримки (наприклад, корпоративного психолога);
- Робоче середовище повинно бути вільним від токсичної комунікації;
- Працівники повинні вміти відмежовуватись від роботи після завершення робочого дня.

У кожному офісному приміщенні повинна бути аптечка першої допомоги, яка містить:

- стерильні бинти, пластирі, вата;
- перекис водню, хлоргексидин;
- антисептики для рук;
- жарознижувальні препарати;
- ножиці, пінцет;
- гумові рукавички.

Персонал має бути ознайомлений з розташуванням аптечки та вміти надати домедичну допомогу. У випадку домашнього використання мобільного застосунку бажано мати базову аптечку поруч з робочим місцем.

Смартфони та планшети є основним середовищем для роботи з мобільним застосунком. Тривале використання таких пристроїв може викликати:

- зорове перенапруження;
- підвищене навантаження на шию (так званий «текстовий синдром»);
- сухість очей, головний біль;
- розлади сну при вечірній роботі з яскравим екраном.

Для зниження негативного впливу необхідно:

- використовувати функцію «нічного режиму» або фільтр синього світла;
- обирати шрифти середнього розміру;
- не тримати смартфон нижче рівня грудей;
- не працювати з ним у темряві.

В умовах поширеної практики віддаленої або гібридної роботи виникає низка додаткових факторів ризику, які варто враховувати. Зокрема:

- зниження фізичної активності та тривале сидіння без перерв;
- відсутність належно організованого робочого місця вдома;
- ізоляція, відсутність соціальної взаємодії, що впливає на психоемоційний стан;
- проблеми зі сном, порушення режиму дня;
- складність у розмежуванні особистого й робочого часу.

Для мінімізації цих ризиків рекомендується:

- створити чіткий робочий графік з часом початку і завершення роботи;
- регулярно виходити з дому (прогулянки, спорт);
- залишатися на зв'язку з командою — щоденні стендапи, неформальні чати;
- організувати повноцінне робоче місце (стілець, стіл, освітлення);
- дотримуватись правил цифрової гігієни: не працювати у ліжку, не перевіряти пошту вночі тощо.

Роботодавець відіграє ключову роль у формуванні безпечного середовища праці, навіть у випадках віддаленої зайнятості. Зокрема, компанія може:

- проводити інструктажі з охорони праці для всіх працівників;
- забезпечити працівників необхідними засобами (наприклад, знижками на ергономічні крісла або лампи);
- розробити політику психічного здоров'я (анонімні консультації, гаряча лінія);
- надавати підтримку у разі надзвичайних ситуацій (наприклад, під час воєнного стану чи пандемії);
- фінансувати курси першої допомоги або забезпечити аптечки.

Також важливо розробити внутрішні регламенти щодо максимальної тривалості роботи за екраном, мінімальних вимог до домашнього робочого місця, режиму праці та відпочинку, включення перерв у планування завдань.

Таким чином, експлуатація мобільного застосунку для тайм-менеджменту є безпечною за умови дотримання норм охорони праці. Програмний продукт не створює техногенних чи фізичних загроз для здоров'я, а ризики, пов'язані з сидячою роботою або навантаженням на зір, повністю компенсуються відповідною організацією умов праці.

Дотримання режиму праці, мікрокліматичних умов, ергономіки, правил електро- та пожежної безпеки дозволяє уникнути порушень здоров'я під час експлуатації. Рекомендації з профілактики професійних захворювань, психоемоційної гігієни та цифрової безпеки є важливими для сталого використання мобільного застосунку у сучасному середовищі праці.

Урахування нормативних актів, адаптація середовища до потреб користувачів, а також усвідомлена поведінка самих працівників дозволяють забезпечити високий рівень безпеки, що є необхідною умовою для ефективного використання ІТ-продуктів у професійній діяльності.

Висновок за розділом 4: у цьому розділі окреслено всі необхідні вимоги з охорони праці та безпеки — від ергономіки й мікроклімату

робочого місця до правил електробезпеки та протипожежних заходів із посиланням на чинні нормативні документи. Дотримання цих рекомендацій гарантує мінімізацію ризиків професійних захворювань, зорового та психоемоційного навантаження при роботі з мобільним застосунком в офісі та вдома.

Список використаних джерел

1. 15 інструментів тайм-менеджменту в IT: як ефективно управляти ..., <https://web-academy.ua/blog/pm-ba-pdm/15-instruments-for-time-management>
2. What is mobile time management apps? – Focuskeeper Glossary, <https://focuskeeper.co/glossary/what-is-mobile-time-management-apps>
3. Ефективний тайм-менеджмент для керівників: Балансування оперативних та стратегічних задач - IT-послуги, <https://itez.com.ua/blog/effective-time-management-for-executives-balancing-operational-and-strategic-tasks.html>
4. Pomodoro Technique for Design and User Experience Teams - Lark, доступ отримано травня 8, 2025, https://www.larksuite.com/en_us/topics/goal-setting-techniques-for-functional-teams/pomodoro-technique-for-design-and-user-experience-teams
5. Implementing the Pomodoro Technique in Software Development: Case Studies and Best Practices - DEV Community, доступ отримано травня 8, 2025, <https://dev.to/teamcamp/implementing-the-pomodoro-technique-in-software-development-case-studies-and-best-practices-4hj9>
6. A brief introduction to GTD (Getting Things Done) - Endjin, доступ отримано травня 8, 2025, <https://endjin.com/blog/2020/02/a-brief-introduction-to-gtd-getting-things-done/>
7. Інформаційна архітектура — Вікіпедія, доступ отримано травня 8, 2025, https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0
8. How to Build a Team Management App - Trovve, доступ отримано травня 8, 2025, <https://trovve.com/2024/01/24/how-to-build-a-team-management-app/>
9. Database schema for project management system - Surfside Media, доступ отримано травня 8, 2025, <https://www.surfsidemedia.in/post/database->

[schema-for-project-management-system](#)

10. Notifications overview | Views - Android Developers, доступ отримано травня 8, 2025, <https://developer.android.com/develop/ui/views/notifications>
11. The Mobile App Architecture Guide for 2024 - Net Solutions, доступ отримано травня 8, 2025, <https://www.netsolutions.com/hub/mobile-app-development/architecture/>
12. The Importance of UX/UI Design for Mobile Apps ... - Developer Nation, доступ отримано травня 8, 2025, <https://www.developernation.net/blog/the-importance-of-ux-ui-design-for-mobile-apps-how-to-create-an-engaging-user-experience/>
13. Guide to app architecture | App architecture | Android Developers, доступ отримано травня 8, 2025, <https://developer.android.com/topic/architecture>
14. MVVM (Model View ViewModel) Architecture Pattern in Android - GeeksforGeeks, доступ отримано травня 8, 2025, <https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/>
15. Understanding Client-Server Architecture Basics - SynchroNet, доступ отримано травня 8, 2025, <https://synchronet.net/client-server-architecture/>
16. This is simple notes app that follows MVVM architectural design pattern and uses android jetpack components - GitHub, доступ отримано травня 8, 2025, <https://github.com/noob-gamedestroyer/MVVM-Notes-app>
17. Time Tracking for Android Studio | Code Time - Software.com, доступ отримано травня 8, 2025, <https://www.software.com/code-editors/android-studio>
18. Calendar provider overview | Identity - Android Developers, доступ отримано травня 8, 2025, <https://developer.android.com/identity/providers/calendar-provider>
19. How to make Offline database with online synchronization in android app using JAVA. Please suggest me code or any related tutorial - Stack Overflow, <https://stackoverflow.com/questions/78160414/how-to-make-offline-database-with-online-synchronization-in-android-app-using-java>

20. Composite Design Pattern in Java - GeeksforGeeks, доступ отримано травня 8, 2025, <https://www.geeksforgeeks.org/composite-design-pattern-in-java/>

ВИСНОВКИ

У результаті виконання дипломного проєкту було реалізовано розробку мобільного застосунку для тайм-менеджменту, що відповідає сучасним вимогам ІТ-сектора й дозволяє ефективно організовувати робочий час у командному та індивідуальному режимах.

1. Проведено докладний аналіз предметної області та вивчено ключові підходи до управління часом, а також специфіку інформаційних систем ІТ-компаній, що дало змогу сформулювати обґрунтовану мету та завдання дослідження. На цьому етапі визначено цільову аудиторію, проаналізовано існуючі рішення на ринку та виявлено їхні недоліки, що потребували створення спеціалізованого інструменту.
2. Здійснено модельне проектування даних: сформовано концептуальну ER-діаграму та фізичну модель із детальним описом полів, ключів і обмежень цілісності, що забезпечує надійність та продуктивність зберігання даних.
3. Для реалізації алгоритмів планування задач використано комбінацію математичних методів оптимізації. Розроблені процедури ітераційного моделювання дозволяють адаптуватися до обмежень дедлайнів і ресурсів, забезпечуючи гнучкість у побудові розкладу та контроль достовірності обчислень.
4. Обрано сучасний технологічний стек: для клієнтської частини — платформа Android з використанням Jetpack-компонентів.
5. Значну увагу приділено розробці інтерфейсу користувача та UX-дизайну: реалізовано інтуїтивні екрани створення і перегляду задач, таймер Pomodoro, історію продуктивності й аналітичні дашборди. Додатково впроваджено механізми офлайн-кешування та синхронізації даних для роботи в умовах нестабільного з'єднання.

Результати дипломного проекту можуть бути використані як основа для впровадження аналогічних систем у корпоративних чи стартапових середовищах, а також адаптовані під різні вертикалі (освіта, продажі, медицина) для підвищення продуктивності та якості роботи користувачів.

ДОДАТКИ

ДОДАТОК А

Посібник користувача мобільного застосунку для тайм-менеджменту інформаційної системи ІТ-компанії

1. Встановлення та запуск застосунку

1. Завантажити APK-файл із офіційного репозиторію або магазинів Google Play / App Store.
2. Встановити застосунок, підтвердивши права доступу до хронометра, пам'яті та сповіщень.
3. Відкрийте застосунок, натиснувши іконку на робочому екрані.

2. Головний екран (Завдання)

Головний екран відображає список задач у вигляді карток (зображено на Рисунку А1).

- Назва завдання та дата виконання (формат: "19 травня 2025").
- Виконавець (наприклад, Bob The Builder або Користувач).
- Пріоритет: кольорові бейджі "Високий" (червоний), "Середній" (жовтий), "Низький" (зелений).
- Статус: бейджі "todo", "in_progress", "on_hold" у блакитному кольорі.

Фільтрація та сортування

Натисніть значок меню з трьома крапками у правому верхньому куті екрану(Рисунок А2):

- Сортувати за датою
- Сортувати за пріоритетом
- Призначені мені

Виберіть потрібний пункт, система автоматично оновить список задач.

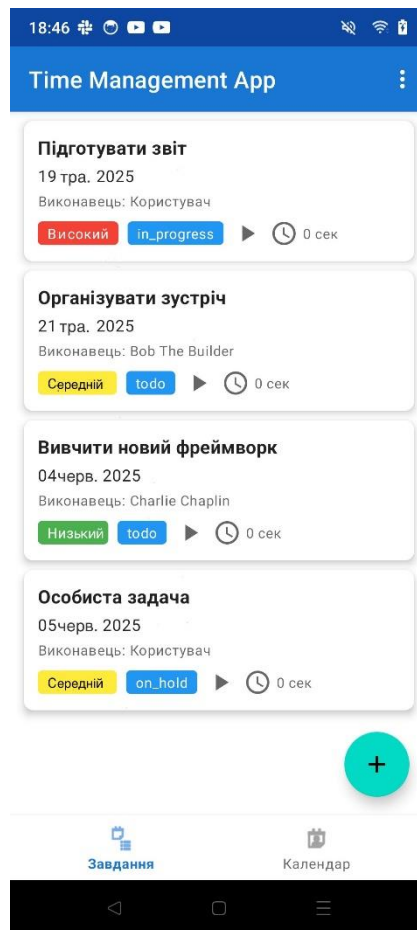


Рисунок А1. Головний екран застосунку

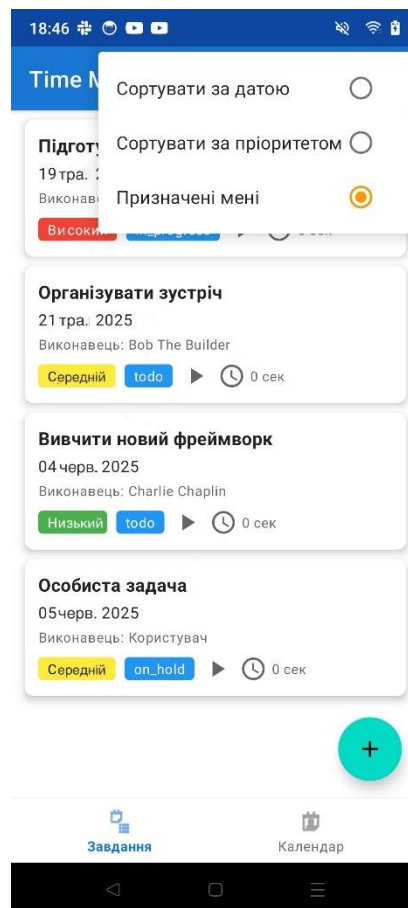


Рисунок А2. Меню сортування задач

3. Створення нової задачі

Натисніть круглу кнопку «+» у правому нижньому куті головного екрану.

1. Назва задачі: введіть короткий опис (обов'язково).
2. Опис задачі: деталізуйте мету або кроки виконання.
3. Проект: виберіть із списку ("Робота", "Особистий проект" чи "Без проекту")(Рисунок А3).

Рисунок А3. Випадаючий список вибору проекту

4. Виконавець: оберіть із випадаючого списку наявних користувачів або "Не назначений"(Рисунок А4).

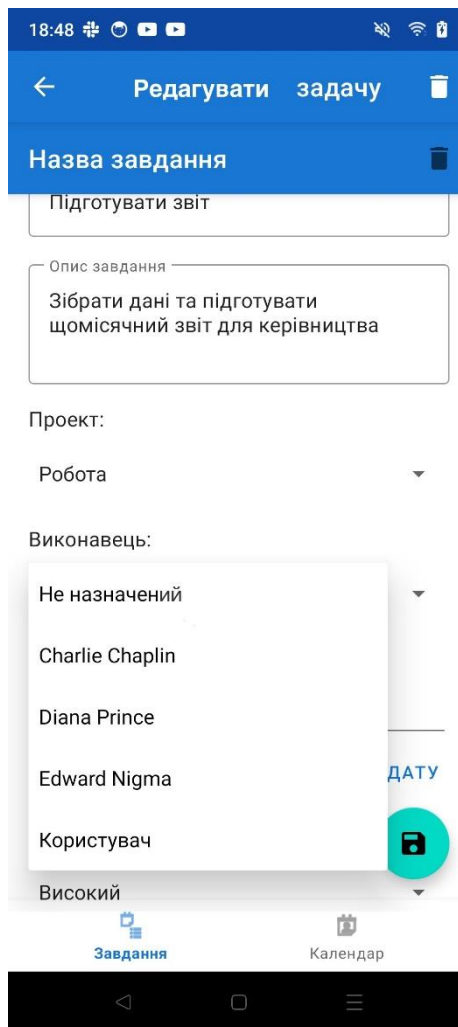


Рисунок А4. Випадаючий список виконавців проекту

5. Дата виконання: відображається під полем "Дата виконання". Натисніть «Змінити дату» для вибору через календар(Рисунок А5).

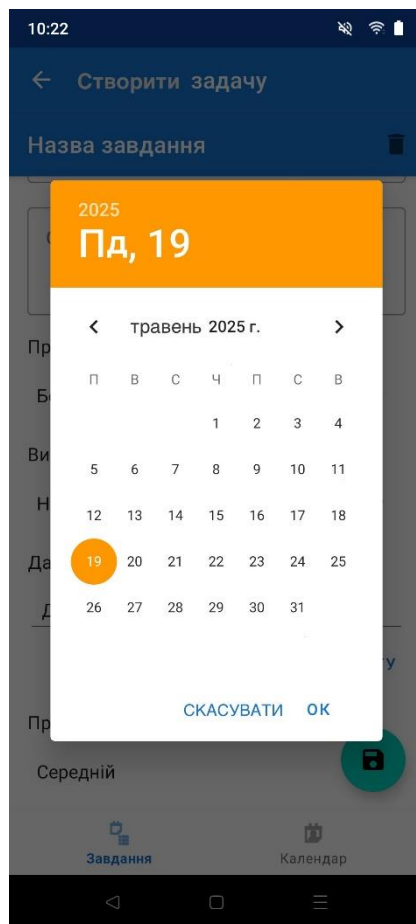


Рисунок А5. Вибір дати через календар

6. Пріоритет: виберіть один із трьох рівнів (Рисунок А6):

- Низький
- Середній
- Високий

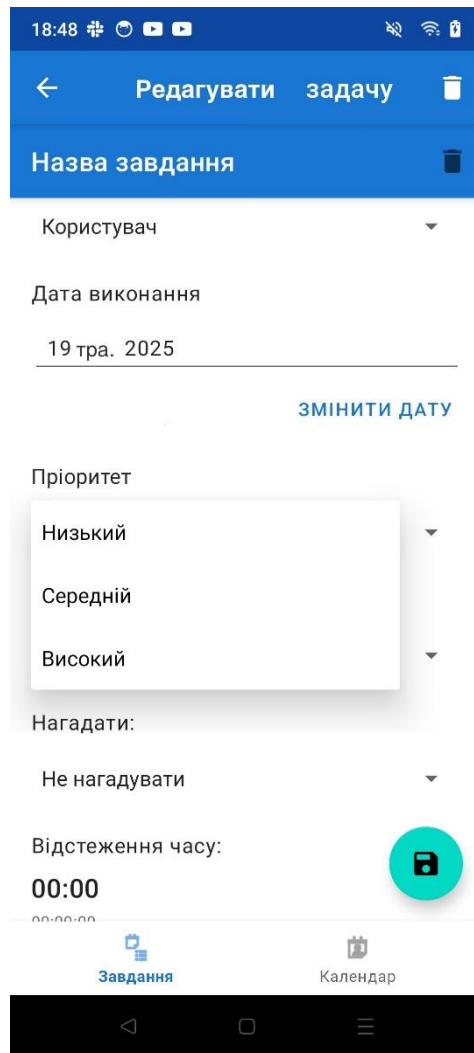


Рисунок А6. Випадаючий список з рівнями пріоритету завдання

7. Статус: задайте початковий статус (Рисунок А7):

- До виконання
- В процесі
- Виконано
- Відкладено
- Скасовано

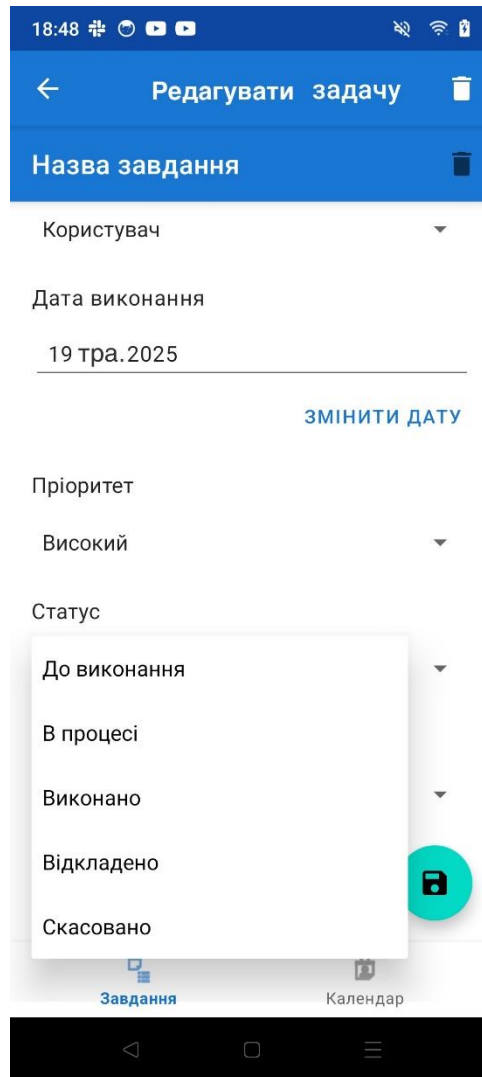


Рисунок А7. Випадаючий список вибору статусу завдання

8. Нагадати: виберіть опцію нагадування(Рисунок А8.):

- Не нагадувати
- В день події (о 09:00)
- За 15 хвилин
- За 30 хвилин
- За 1 годину
- За 1 день

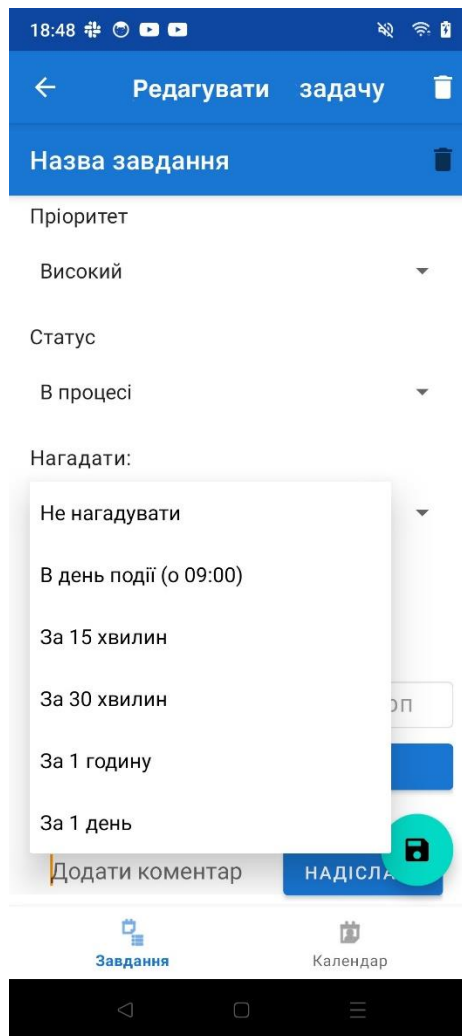


Рисунок А8. Випадаючий список вибору опції нагадування
Після заповнення натисніть кнопку «Зберегти».

4. Редагування та видалення задачі

1. На головному екрані торкніться картки задачі, яку потрібно змінити.
2. Відкриється екран Редагувати задачу (Рисунок А9)
3. Змініть необхідні поля: назву, опис, проєкт, виконавця, дату, пріоритет, статус чи нагадування.
4. Натисніть кнопку «Зберегти» внизу.
5. Щоб видалити задачу, натисніть значок кошику у верхньому правому куті та підтвердіть.

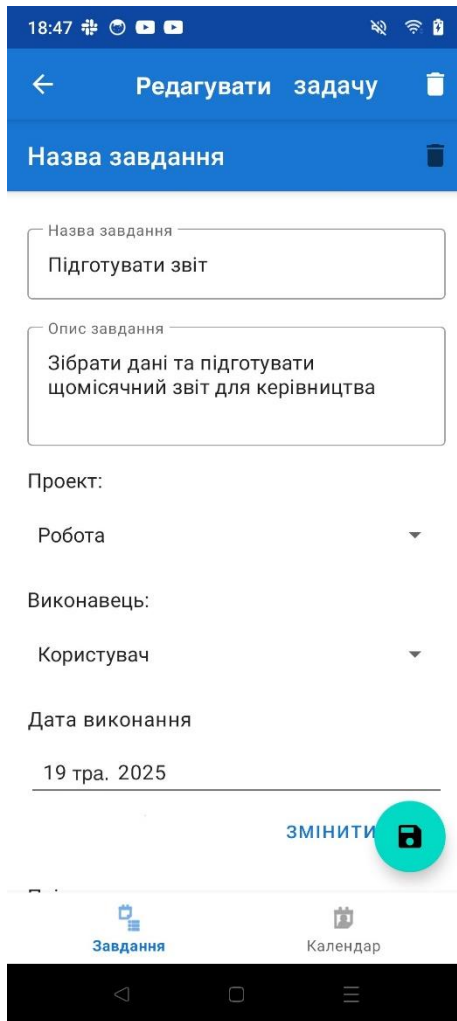


Рисунок А9. Екран редагування задачі

5. Календар

Перейдіть у вкладку Календар (Рисунок А10).

- Місячний перегляд із підсвітленням дат із задачами.
- Синя крапка — поточна дата, жовте підкреслення — на цю дату запланована задача.
- Нижче календаря відображається список задач на обрану дату.
- Щоб змінити дату, торкніться іншого дня у календарі; список автоматично оновиться.

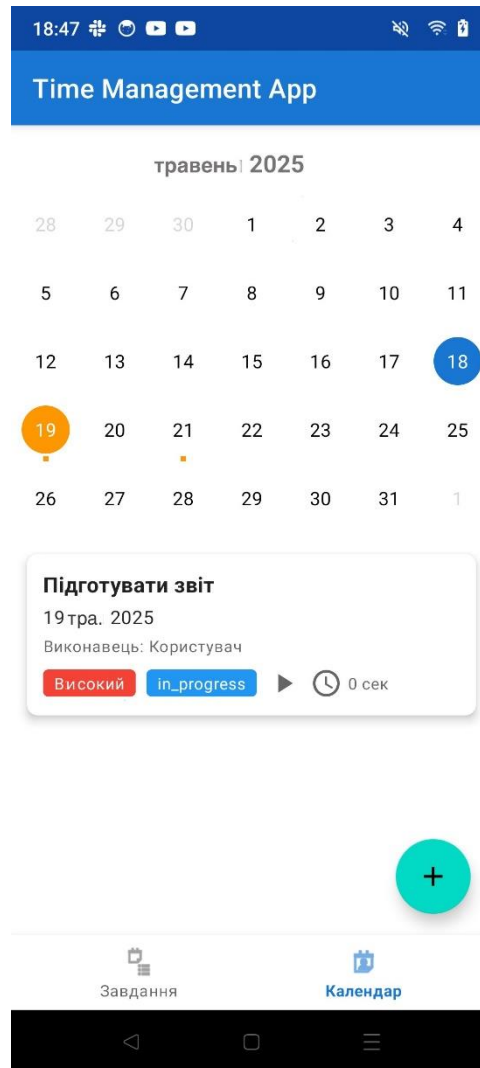


Рисунок А10. Вкладка «Календар»

Цей посібник складено на основі реальних скріншотів застосунку та відповідає актуальній версії інтерфейсу.

ДОДАТОК Б

```
package com.example.timemanagementapp.data.local.entity;

import androidx.room.Entity;
import androidx.room.PrimaryKey;
import androidx.room.ColumnInfo;
import androidx.room.ForeignKey;
import androidx.room.TypeConverters;
import
    com.example.timemanagementapp.data.local.converter.DateCon
    verter;
import java.util.Date;
import java.util.UUID;
import androidx.annotation.NonNull;

@Entity(tableName = "tasks",
    foreignKeys = {
        @ForeignKey(entity = Project.class,
            parentColumns = "project_id",
            childColumns = "project_id",
            onDelete = ForeignKey.SET_NULL),
        @ForeignKey(entity = User.class,
            parentColumns = "user_id",
            childColumns = "assignee_user_id",
            onDelete = ForeignKey.SET_NULL),
        @ForeignKey(entity = Task.class,
            parentColumns = "task_id",
            childColumns = "parent_task_id",
            onDelete = ForeignKey.CASCADE)
    })
@TypeConverters(DateConverter.class)
public class Task {
    @PrimaryKey
    @NonNull
    @ColumnInfo(name = "task_id")
    public String taskId;

    @ColumnInfo(name = "project_id", index = true)
    public String projectId;

    @NonNull
```

```

public String title;
public String description;

@ColumnInfo(name = "assignee_user_id", index = true)
public String assigneeUserId;
@NotNull
@ColumnInfo(name = "creator_user_id")
public String creatorUserId;
@ColumnInfo(name = "due_date")
public Date dueDate;
public int priority;

@NotNull
public String status;
@ColumnInfo(name = "is_recurring")
public boolean isRecurring;
@ColumnInfo(name = "recurrence_rule")
public String recurrenceRule;
@ColumnInfo(name = "parent_task_id", index = true)
public String parentTaskId;
@NotNull
@ColumnInfo(name = "created_at")
public Date createdAt;
@NotNull
@ColumnInfo(name = "updated_at")
public Date updatedAt;

@ColumnInfo(name = "completed_at")
public Date completedAt;
@ColumnInfo(name = "reminder_offset_millis")
public Long reminderOffsetMillisBeforeDueDate;
@ColumnInfo(name = "time_spent_millis")
public long timeSpentMillis;
@ColumnInfo(name = "time_tracking_start_time_millis")
public Long timeTrackingStartTimeMillis;
public Task(@NotNull String title, @NotNull String
    creatorUserId) {
    this.taskId = UUID.randomUUID().toString();
    this.title = title;
    this.creatorUserId = creatorUserId;
    this.priority = 2;
    this.status = "todo";
    this.isRecurring = false;
    Date now = new Date();

```

```

        this.createdAt = now;
        this.updatedAt = now;
    }

    // ...

```

ДОДАТОК В

DAO (Data Access Objects):

Інтерфейси з анотаціями Room (@Dao, @Insert, @Update, @Delete, @Query) для визначення операцій доступу до даних. DAO інкапсулює логіку SQL-запитів.

Приклад TaskDao.java:

```

package com.example.timemanagementapp.data.local.dao;

import androidx.lifecycle.LiveData;
import androidx.room.Dao;
import androidx.room.Insert;
import androidx.room.OnConflictStrategy;
import androidx.room.Query;
import androidx.room.Update;
import androidx.room.Delete;
import com.example.timemanagementapp.data.local.entity.Task;
import java.util.List;

@Dao
public interface TaskDao {
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void insertTask(Task task);

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void insertAllTasks(List<Task> tasks);

    @Update
    void updateTask(Task task);

    @Delete
    void deleteTask(Task task);

    @Query("DELETE FROM tasks")
    void deleteAllTasks();

    @Query("SELECT * FROM tasks WHERE task_id = :taskId")
    LiveData<Task> getTaskById(String taskId);

```

```

@Query("SELECT * FROM tasks ORDER BY due_date ASC")
LiveData<List<Task>> getAllTasksSortedByDueDate();

@Query("SELECT * FROM tasks WHERE project_id = :projectId
ORDER BY priority DESC, due_date ASC")
LiveData<List<Task>> getTasksForProject(String projectId);

@Query("SELECT * FROM tasks WHERE assignee_user_id =
:userId AND status != 'done' ORDER BY due_date ASC")
LiveData<List<Task>> getActiveTasksForUser(String userId);

@Query("SELECT * FROM tasks ORDER BY priority DESC,
due_date ASC")
LiveData<List<Task>> getAllTasksSortedByPriority();
}

```

ДОДАТОК Г

Доповнення TaskRepository.java:

```

package com.example.timemanagementapp.data;

import android.app.Application;
import android.content.Context;
import android.content.SharedPreferences;
import androidx.lifecycle.LiveData;
import com.example.timemanagementapp.data.local.AppDatabase;
import com.example.timemanagementapp.data.local.dao.TaskDao;
import
    com.example.timemanagementapp.data.local.dao.ProjectDao;
import com.example.timemanagementapp.data.local.dao.UserDao;
import
    com.example.timemanagementapp.data.local.dao.TaskCommentDa
    o;
import com.example.timemanagementapp.data.local.entity.Task;
import
    com.example.timemanagementapp.data.local.entity.Project;
import com.example.timemanagementapp.data.local.entity.User;
import
    com.example.timemanagementapp.data.local.entity.TaskCommenn
    t;
import com.example.timemanagementapp.data.remote.ApiService;

```



```

import
    com.example.timemanagementapp.data.remote.RetrofitClient;
import com.example.timemanagementapp.data.remote.dto.TaskDto;
import
    com.example.timemanagementapp.data.remote.dto.ProjectDto;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import retrofit2.Call;
import retrofit2.Response;
import android.util.Log;

public class TaskRepository {
    private static final String TAG = "TaskRepository";
    private static final String PREFS_NAME = "SyncPrefs";
    private static final String KEY_LAST_SYNC_TASKS =
        "lastSyncTasks";
    private static final String KEY_LAST_SYNC_PROJECTS =
        "lastSyncProjects";

    private TaskDao taskDao;
    private ProjectDao projectDao;
    private UserDao userDao;
    private TaskCommentDao taskCommentDao;

    private ApiService apiService;
    private SharedPreferences sharedPreferences;

    private LiveData<List<Task>> allTasks;
    private LiveData<List<Project>> allProjects;
    private LiveData<List<User>> allUsers;
    private ExecutorService executorService;

    public TaskRepository(Application application) {
        AppDatabase database =
            AppDatabase.getInstance(application);
        taskDao = database.taskDao();
        projectDao = database.projectDao();
        userDao = database.userDao();
    }

```

```

        taskCommentDao = database.taskCommentDao();

        // Ініціалізація ApiService (потрібно реалізувати
        RetrofitClient)
        apiService =
        RetrofitClient.getInstance(application).getApiService();
        sharedPreferences =
        application.getSharedPreferences(PREFS_NAME,
        Context.MODE_PRIVATE);

        allTasks = taskDao.getAllTasksSortedByDueDate();
        allProjects = projectDao.getAllProjects();
        allUsers = userDao.getAllUsers();
        executorService = Executors.newFixedThreadPool(3);
    }

    \
    // public void insert(Task task) { ... }
    // public void update(Task task) { ... }
    // ...

    public void insertSynced(Task task) {
        // task.setSyncStatus(SyncStatus.CREATED);
        executorService.execute(() -> {
            taskDao.insertTask(task);
            pushSingleTaskToServer(task);
        });
    }

    public void updateSynced(Task task) {

        task.setSyncStatus(SyncStatus.UPDATED);
        executorService.execute(() -> {
            taskDao.updateTask(task);
            pushSingleTaskToServer(task);
        });
    }

    public void deleteSynced(Task task) {
        executorService.execute(() -> {
            taskDao.deleteTask(task);
            deleteSingleTaskFromServer(task.getTaskId());
        });
    }

```

```
}  
// ...
```