МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ MVP (MINIMAL VIABLE PRODUCT) НА ПЛАТФОРМІ NO-CODE ДЛЯ АВТОМАТИЗАЦІЇ ОРГАНІЗАЦІЙНО-УПРАВЛІНСЬКИХ ПРОЦЕСІВ»**

Виконав: студент 4 курсу, групи 6.1222
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
В. О. Овчаренко
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, к.т.н Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри програмної інженерії, к.ф.-м.н. Кривохата А. Г.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри комп'ютерних наук, к.пед.н., доцент

_____ Пшенична О.С.
(підпис)

« 08 « грудня 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Овчаренку Владиславу Олексійовтчу
(прізвище, ім'я та по-батькові)

1. Тема роботи Проектування та реалізація MVP (Minimal Viable Product) на платформі No-Code для автоматизації організаційно-управлінських процесів

керівник роботи Матвіїшина Надія Вікторівна, к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 02 » грудня 2025 року № 1741-с

2. Строк подання студентом роботи 04.06.2026

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.
3. Якщо є що додати.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Теоретичні основи проєктування no-code рішень для автоматизації управлінських процесів.
2. Проєктування MVP-системи для автоматизації організаційно-управлінських процесів.
3. Реалізація MVP-системи на платформі Glide.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) презентація

6. Консультанти розділів роботи

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

7. Дата видачі завдання 05.12.2025

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	18.02.2026	
2.	Збір вихідних даних.	14.03.2026	
3.	Обробка методичних та теоретичних	26.03.2026	
	джерел.		
4.	Розробка першого та другого розділу.	14.04.2026	
5.	Розробка третього розділу.	15.05.2026	
6.	Оформлення та нормоконтроль	04.06.2026	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	19.06.2026	

Студент _____ В. О. Овчаренко
(підпис) (ініціали та прізвище)

Керівник роботи _____ Н.В.Матвіїшина
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Проєктування та реалізація MVP (мінімально життєздатного продукту) на платформі No-Code для автоматизації організаційно-управлінських процесів»: 77 с., 29 рис., 17 табл., 24 джерел, 2 додатки.

АВТОМАТИЗАЦІЯ УПРАВЛІНСЬКИХ ПРОЦЕСІВ, БАЗА ДАНИХ, ВЕБЗАСТОСУНОК, РОЛЬОВИЙ ДОСТУП, УПРАВЛІННЯ ЗАВДАННЯМИ, GLIDE, MVP, NO-CODE, PWA, WORKFLOW.

Об’єктом дослідження є процес автоматизації організаційно-управлінських процесів із використанням інформаційних технологій.

Предметом дослідження є методи та засоби проєктування і реалізації MVP-систем на основі No-Code платформ.

Метою роботи є проєктування та реалізація MVP-системи для автоматизації організаційно-управлінських процесів із використанням No-Code платформи Glide.

У роботі використано методи аналізу літературних джерел, порівняльного аналізу No-Code платформ, моделювання структури інформаційної системи, проєктування баз даних, візуального конструювання інтерфейсу та налаштування workflow-процесів у середовищі Glide.

У роботі досліджено особливості використання No-Code платформ та концепції MVP, проаналізовано сучасні No-Code рішення й обґрунтовано вибір платформи Glide. Спроєктовано структуру застосунку, базу даних та інтерфейс користувача, а також реалізовано MVP-систему для управління відділами, співробітниками, проєктами, завданнями та заявками. У застосунку реалізовано автоматизацію процесів, рольовий доступ і захист даних. Результати підтвердили ефективність No-Code підходу для швидкої розробки систем автоматизації управлінських процесів.

SUMMARY

Bachelor's Qualifying Thesis «Design and Implementation of a Minimal Viable Product (MVP) on a No-Code Platform to Automate Organizational and Management Processes»: 77 pages, 29 figures, 17 tables, 24 sources, 2 supplements.

AUTOMATION OF MANAGEMENT PROCESSES, DATABASE, WEB APPLICATION, ROLE-BASED ACCESS, TASK MANAGEMENT, GLIDE, MVP, NO-CODE, PWA, WORKFLOW.

The object of the research is the process of automating organizational and managerial processes using information technologies.

The subject of the research is the methods and tools for designing and implementing MVP systems based on No-Code platforms.

The aim of the work is to design and implement an MVP system for automating organizational and managerial processes using the No-Code platform Glide.

The research methods include literature analysis, comparative analysis of No-Code platforms, information system modeling, database design, visual interface development, and workflow configuration in the Glide environment.

The thesis explores the features of No-Code platforms and the MVP concept, analyzes modern No-Code solutions, and justifies the selection of Glide. The application structure, database, and user interface were designed, and an MVP system for managing departments, employees, projects, tasks, and requests was implemented. The system includes process automation, role-based access, and data protection mechanisms. The results confirm the effectiveness of the No-Code approach for rapid development of management process automation systems.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2
Реферат 4
Summary 5
Вступ..... 8
1 Теоретичні основи проєктування no-code рішень для автоматизації управлінських процесів 11
1.1 Поняття та особливості No-Code платформ..... 11
1.2 Сутність MVP (мінімально життєздатного продукту) та етапи його створення 13
1.3 Аналіз сучасних No-Code платформ для розробки інформаційних систем…. 16
1.3.1 Glide..... 17
1.3.2 Bubble 18
1.3.3 Webflow..... 19
1.3.4 AppSheet..... 20
2 Проєктування MVP-системи для автоматизації організаційно-управлінських процесів 23
2.1 Аналіз предметної області 23
2.2 Проєктування структури та функціоналу MVP-системи..... 26
2.3 Проєктування бази даних та логіки роботи системи 30
2.4 Проєктування інтерфейсу користувача 38
3 Реалізація MVP-системи на платформі Glide..... 42
3.1 Реалізація структури застосунку в Glide 42
3.2 Налаштування бази даних та автоматизації процесів..... 46
3.3 Реалізація системи користувачів та ролей доступу..... 49
3.4 Реалізація функціоналу управління задачами та заявками 52
3.5 Аналіз результатів розробки MVP-системи..... 56

3.6 Перспективи розвитку та вдосконалення застосунку 58
Висновки 60
Перелік посилань..... 62
Додаток А Інтерфейс розробленої програми 65
Додаток Б Розроблені таблиці для No-code платформи..... 75

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується активною цифровізацією управлінських та організаційних процесів у різних сферах діяльності. Підприємства, освітні установи, державні організації та комерційні структури дедалі частіше впроваджують цифрові рішення для автоматизації внутрішніх процесів, підвищення ефективності управління, оптимізації документообігу та координації роботи працівників. У зв’язку з цим особливої актуальності набувають технології швидкої розробки програмних продуктів, які дозволяють створювати функціональні інформаційні системи без значних витрат часу та ресурсів.

Одним із перспективних напрямків сучасної веброзробки є використання концепції No-Code. No-Code платформи дозволяють створювати вебзастосунки та інформаційні системи без традиційного програмування шляхом використання візуальних інструментів, готових компонентів та автоматизованих workflow-механізмів. Такий підхід значно спрощує процес розробки програмного забезпечення та робить його доступним не лише професійним розробникам, але й користувачам без глибоких знань мов програмування.

Особливої популярності No-Code технології набули у сфері створення MVP (Minimum Viable Product – мінімально життєздатного продукту). MVP являє собою базову версію програмного продукту з мінімальним набором функцій, достатніх для перевірки основної ідеї, тестування бізнес-процесів та отримання зворотного зв’язку від користувачів. Такий підхід дозволяє швидко оцінити ефективність цифрового рішення та мінімізувати ризики під час розробки повноцінної системи.

На сьогодні існує значна кількість No-Code платформ, серед яких особливе місце займає Glide. Дана платформа забезпечує можливість створення веб- та мобільних застосунків на основі візуального конструювання інтерфейсу,

інтеграції з таблицями даних та автоматизації процесів без написання програмного коду. Glide відзначається простотою використання, швидкістю створення MVP та можливістю реалізації базових функцій управлінських інформаційних систем.

Актуальність теми роботи зумовлена необхідністю автоматизації організаційно-управлінських процесів за допомогою сучасних цифрових технологій та потребою у швидкій розробці інформаційних систем із мінімальними витратами ресурсів. Використання No-Code платформ дозволяє суттєво скоротити терміни створення програмних продуктів, забезпечити гнучкість їх модернізації та підвищити доступність цифрових рішень для малого бізнесу, освітніх установ та організацій різного типу.

Метою роботи є проєктування та реалізація MVP-системи для автоматизації організаційно-управлінських процесів із використанням No-Code платформи Glide.

Для досягнення поставленої мети необхідно виконати такі завдання:

- дослідити особливості No-Code платформ та концепцію MVP;
- проаналізувати сучасні No-Code платформи та обрати Glide для реалізації системи;
- визначити функціональні та нефункціональні вимоги до MVP-системи;
- спроєктувати структуру системи, базу даних та інтерфейс користувача;
- реалізувати MVP-застосунок на платформі Glide;
- проаналізувати результати розробки та перспективи розвитку системи.

Об’єктом дослідження є процес автоматизації організаційно-управлінських процесів із використанням інформаційних технологій.

Предметом дослідження є методи та засоби проєктування і реалізації MVP-систем на основі No-Code платформ.

У роботі використано методи аналізу літературних джерел, порівняльного аналізу No-Code платформ, моделювання структури інформаційної системи, проєктування баз даних, візуального конструювання інтерфейсу та налаштування workflow-процесів у середовищі Glide.

Практичне значення отриманих результатів полягає у створенні функціонального MVP-застосунку для автоматизації управління задачами та організаційними процесами. Розроблена система може бути використана в діяльності невеликих підприємств, освітніх установ, студентських організацій та інших структур, де існує потреба у цифровізації процесів управління задачами, заявками та взаємодією між користувачами.

Наукова новизна роботи полягає у дослідженні можливостей використання No-Code підходу для швидкого створення MVP інформаційних систем управлінського призначення та практичній реалізації вебзастосунку на платформі Glide без використання традиційного програмування.

Результати роботи можуть бути використані як основа для подальшого розвитку повноцінної інформаційної системи, інтеграції додаткових модулів автоматизації, підключення зовнішніх сервісів та масштабування функціональних можливостей застосунку відповідно до потреб конкретної організації.

1 ТЕОРЕТИЧНІ ОСНОВИ ПРОЄКТУВАННЯ NO-CODE РІШЕНЬ ДЛЯ АВТОМАТИЗАЦІЇ УПРАВЛІНСЬКИХ ПРОЦЕСІВ

1.1 Поняття та особливості No-Code платформ

No-Code платформи є сучасним напрямом розвитку інформаційних технологій, який дозволяє створювати програмні застосунки без необхідності написання програмного коду або з мінімальним його використанням. Основна ідея таких платформ полягає у переході від традиційного програмування до візуального моделювання, де користувач формує логіку, інтерфейс і бізнес-процеси за допомогою готових блоків, шаблонів та графічних елементів. У науковій літературі No-Code розглядається як частина ширшого руху low-code/no-code, спрямованого на прискорення цифрової трансформації та зменшення залежності від професійних розробників [1].

Сутність No-Code підходу полягає у демократизації процесу створення програмного забезпечення, коли розробка цифрових продуктів стає доступною не лише для професійних програмістів, але й для користувачів без глибоких технічних знань. У такій парадигмі так звані «citizen developers» можуть самостійно створювати бізнес-застосунки, автоматизувати процеси та формувати цифрові рішення для власних потреб. Замість написання коду користувач взаємодіє з візуальним середовищем, де логіка системи задається через налаштування компонентів, зв’язків між даними та сценаріїв поведінки системи [2].

Однією з ключових особливостей No-Code платформ є використання візуального підходу до розробки, який реалізується через drag-and-drop інтерфейси. Завдяки цьому користувач може створювати інтерфейс застосунку шляхом перетягування елементів, налаштовуючи їх поведінку без необхідності програмування. Другою важливою особливістю є використання готових шаблонів та модулів, що дозволяє значно скоротити час розробки та уникнути

необхідності створення базових функцій із нуля. Крім того, сучасні No-Code платформи забезпечують можливість роботи з даними, автоматизації бізнес-процесів та інтеграції із зовнішніми сервісами через API, що розширює їх функціональність і робить їх придатними для створення повноцінних інформаційних систем [3].

Функціональні можливості No-Code платформ охоплюють широкий спектр задач, включаючи створення веб-інтерфейсів, налаштування баз даних, моделювання бізнес-логіки та автоматизацію процесів. У практичному застосуванні це означає можливість створення CRM-систем, систем управління задачами, внутрішніх порталів організацій, форм збору даних та інших цифрових інструментів. Особливо важливо, що такі платформи дозволяють реалізувати MVP-рішення у дуже короткі терміни, що є критично важливим для тестування бізнес-гіпотез і перевірки ефективності цифрових продуктів [4].

Переваги використання No-Code платформ полягають насамперед у значному прискоренні процесу розробки програмного забезпечення. Оскільки відпадає необхідність написання коду вручну, розробка стає доступнішою та швидшою, що дозволяє зменшити витрати часу та фінансових ресурсів. Крім того, No-Code підхід відкриває можливості для залучення нефахівців до процесу створення цифрових продуктів, що підвищує гнучкість організацій та сприяє інноваціям. Додатковою перевагою є можливість швидкого внесення змін у вже створені системи, що особливо важливо в умовах динамічних бізнес-вимог [5].

Разом із тим No-Code платформи мають і певні обмеження, які необхідно враховувати при їх використанні. Однією з основних проблем є залежність від конкретного постачальника платформи, що може ускладнювати міграцію системи або зміну технологічного середовища. Також обмеження можуть стосуватися масштабованості, продуктивності та гнучкості архітектури, особливо у випадках складних або високонавантажених систем. У таких ситуаціях No-Code рішення можуть бути недостатніми, і доцільним стає використання гібридного або класичного підходу до розробки [1].

Важливу роль No-Code платформи відіграють у процесах цифрової трансформації організацій. Вони дозволяють значно скоротити шлях від ідеї до реалізації програмного продукту, що забезпечує швидке реагування на зміни у бізнес-середовищі. Крім того, вони зменшують навантаження на професійних розробників, які можуть зосередитися на складніших технічних задачах, тоді як простіші рішення реалізуються бізнес-користувачами. Тобто No-Code не замінює традиційну розробку, а доповнює її, особливо у сфері створення MVP та автоматизації стандартних процесів [6].

Отже, No-Code платформи є сучасним інструментом візуальної розробки програмного забезпечення, який дозволяє створювати цифрові продукти без програмування. Вони поєднують у собі простоту використання, високу швидкість розробки та широкий набір функціональних можливостей, що робить їх ефективними для створення MVP та автоматизації організаційно-управлінських процесів. Водночас їхні обмеження визначають доцільність використання переважно для типових бізнес-задач, прототипування та швидкої перевірки ідей.

1.2 Сутність MVP (мінімально життєздатного продукту) та етапи його створення

Мінімально життєздатний продукт (MVP – Minimum Viable Product) є одним із ключових підходів сучасної цифрової розробки, який застосовується для швидкого тестування ідеї продукту в реальних умовах із мінімальними витратами ресурсів. Його сутність полягає у створенні найпростішої робочої версії продукту, яка містить лише базовий набір функцій, достатній для перевірки основної гіпотези та отримання зворотного зв’язку від користувачів. У межах концепції Lean Startup MVP розглядається не як спрощений продукт, а як інструмент навчання та перевірки цінності ідеї на практиці [7].

Головною ідеєю MVP є досягнення балансу між мінімальними витратами на розробку та максимальним обсягом отриманих знань про користувача і продукт. Замість повноцінної розробки складної системи команда створює базову версію, яка дозволяє перевірити, чи справді запропоноване рішення вирішує проблему користувача. Такий підхід є особливо важливим у випадках високої невизначеності вимог, коли ринковий попит ще не підтверджений, а функціональні потреби користувачів лише формуються [8].

Важливо підкреслити, що MVP не є неякісним або незавершеним продуктом. Навпаки, він повинен залишатися достатньо функціональним для того, щоб користувач міг реально взаємодіяти із системою та формувати обґрунтований зворотний зв’язок. Основна мета полягає не в тому, щоб створити ідеальний продукт, а в тому, щоб отримати достовірні дані про поведінку користувачів і перевірити життєздатність бізнес-гіпотези.

Окремо слід відзначити, що MVP є контекстно залежним поняттям. Мінімальний набір функцій для одного продукту може суттєво відрізнятися від іншого залежно від галузі, складності системи та потреб користувачів. У деяких випадках MVP може бути представлений у вигляді простого веб-інтерфейсу, лендингу, ручного процесу або навіть прототипу без повної автоматизації. Головним критерієм залишається здатність продукту перевірити ключову ціннісну пропозицію [7].

Практичне значення MVP полягає у зниженні ризиків розробки. Замість того щоб інвестувати значні ресурси у повноцінну систему, яка може не знайти свого користувача, команда спочатку перевіряє базову ідею. Це дозволяє уникнути критичних фінансових втрат і своєчасно скоригувати напрям розвитку продукту. Особливо важливим це є для стартапів і проєктів інноваційного характеру, де рівень невизначеності є високим [9].

Ще однією важливою перевагою MVP є перехід від суб’єктивних припущень до об’єктивних даних. Замість того щоб покладатися на думки або гіпотези, команда аналізує реальну поведінку користувачів, що взаємодіють із продуктом [8]. Це дозволяє приймати більш обґрунтовані рішення щодо

подальшого розвитку системи, її вдосконалення або навіть повного перегляду концепції.

Процес створення MVP зазвичай включає кілька послідовних етапів. Першим етапом є аналіз проблеми та визначення потреб користувачів. На цьому кроці формується розуміння того, яку саме проблему має вирішувати майбутній продукт і чому вона є важливою для цільової аудиторії. Без чіткого визначення проблеми подальша розробка втрачає зміст і може призвести до створення непотрібного функціоналу.

Другим етапом є формування ціннісної пропозиції. На цьому рівні визначається, яку саме користь отримає користувач від використання продукту та чому він має обрати саме це рішення серед альтернатив. Цей етап є критично важливим, оскільки саме ціннісна пропозиція визначає напрям подальшої розробки та набір базових функцій [10].

Третім етапом є відбір мінімального набору функцій. На цьому кроці відбувається скорочення повного функціонального бачення продукту до найнеобхідніших елементів, які дозволяють перевірити основну гіпотезу. Усі другорядні функції виключаються, оскільки вони не впливають на перевірку ключової ідеї.

Четвертим етапом є проєктування та створення прототипу або MVP-версії продукту. На цьому етапі формується структура інтерфейсу, логіка взаємодії користувача із системою та загальна архітектура рішення. У випадку No-Code підходу цей етап може бути реалізований за допомогою візуальних інструментів без написання коду, що значно прискорює процес розробки.

П’ятим етапом є запуск продукту та його тестування в реальних умовах. MVP надається обмеженій кількості користувачів, які взаємодіють із системою та формують перші відгуки. Основна мета цього етапу полягає не в масштабуванні, а в отриманні первинних даних про поведінку користувачів і ефективність запропонованого рішення [11].

Шостим етапом є збір та аналіз зворотного зв’язку. На цьому кроці досліджуються показники використання системи, поведінка користувачів, рівень

залученості та задоволеності. Отримані дані дозволяють оцінити, чи відповідає продукт очікуванням і чи потребує він змін.

Завершальним етапом є ітераційне вдосконалення продукту. На основі зібраної інформації приймається рішення про подальший розвиток: розширення функціоналу, зміна концепції або оптимізація існуючих рішень. Такий підхід забезпечує постійний розвиток продукту на основі реальних даних, а не припущень [10].

Отже, MVP є стратегічним підходом до розробки програмних продуктів, який дозволяє поєднати мінімальну функціональність із максимальною ефективністю навчання. Його сутність полягає у швидкій перевірці основної ідеї, а процес створення включає послідовні етапи від аналізу проблеми до ітераційного вдосконалення продукту. Саме тому MVP широко використовується у сучасній розробці інформаційних систем, особливо у поєднанні з No-Code платформами, які значно прискорюють його реалізацію.

1.3 Аналіз сучасних No-Code платформ для розробки інформаційних систем

No-Code платформи – це інструменти розробки програмного забезпечення, що дозволяють створювати повноцінні застосунки без написання програмного коду. Вони надають користувачам візуальний інтерфейс для проектування та конфігурування програм, використовуючи різноманітні компоненти й шаблони [12]. Ключова відмінність від традиційної розробки полягає в тому, що навіть особи без технічної освіти (так звані «citizen developers») здатні самостійно реалізовувати програмні продукти для бізнесу.

Актуальність No-Code рішень підтверджується стрімким зростанням ринку. Згідно з даними The Business Research Company, ринок No-Code платформ зріс із 22,11 млрд доларів у 2023 році до 28,11 млрд доларів у 2024 році з темпом щорічного приросту (CAGR) 27,1 %, а до 2028 року обсяг ринку може досягти

73,84 млрд доларів [13]. За прогнозом MarketsandMarkets, ринок No-Code платформ зі штучним інтелектом зросте з 4,9 млрд доларів у 2024 році до 24,7 млрд доларів у 2029 році з CAGR 38,2 % [14].

Аналітичне агентство Gartner прогнозує, що до 2025–2026 років 70 % нових корпоративних застосунків будуть створені з використанням No-Code або Low-Code технологій, порівняно з менш ніж 25 % у 2020 році [15]. Паралельно передбачається, що частка користувачів цих платформ за межами IT-відділів зросте з 60 % у 2021 році до 80 % у 2026 році. No-Code платформи функціонують завдяки таким механізмам: візуальне проектування інтерфейсів за принципом «drag-and-drop»; підключення до баз даних і API через готові модулі; автоматизація бізнес-процесів та інтеграція із зовнішніми сервісами; швидкий запуск MVP (мінімально життєздатного продукту) [16].

1.3.1 Glide

Glide (рис. 1.1) – No-Code платформа, заснована у 2018 році з місією перетворення електронних таблиць на повноцінні застосунки. Станом на 2024 рік база користувачів досягла 100 тисяч клієнтів, а дохід склав 3,7 млн доларів із приростом 65 % порівняно з попереднім роком [17].

Фундаментом Glide є структура даних зі спредшитів: кожен рядок таблиці відповідає одному запису, а стовпці – полям даних. Платформа підтримує підключення до Google Sheets, Microsoft Excel, Airtable, а також власну базу даних Glide Tables [18]. Візуальний конструктор із перетягуванням компонентів (списки, форми, кнопки, фільтри) дозволяє створювати інтерфейси навіть без дизайнерських навичок. Застосунки публікуються як Progressive Web App та доступні на будь-якому пристрої.

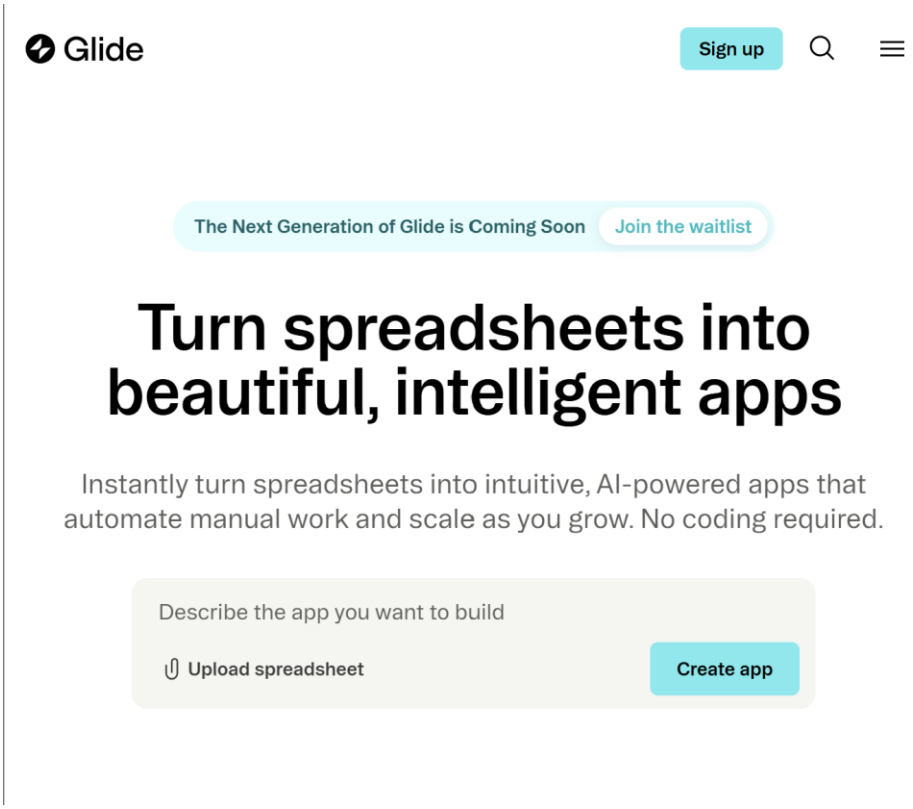


Рисунок 1.1 – Glide

У 2024 році запустили Glide AI – генерацію застосунків із підказок природною мовою та автоматизацію ручних робочих процесів. У 2025 році платформа вийшла на корпоративний рівень, пропонуючи безпечні рішення з аудитом і відповідністю вимогам регуляторів [19]. Безкоштовний план включає до 25 000 рядків даних і 2 редактори; бізнес-план стартує від 249 доларів на місяць. Обмеженням є те, що платформа найкраще підходить для структурованих даних і простих бізнес-застосунків; для складних SaaS-продуктів із нестандартною логікою гнучкості може не вистачати [18].

1.3.2 Bubble

Bubble (рис. 1.2) – одна з найпотужніших No-Code платформ повного стеку, заснована у 2012 році та профінансована на суму 106,3 млн доларів. На платформі побудовано понад 7,2 млн застосунків [20].

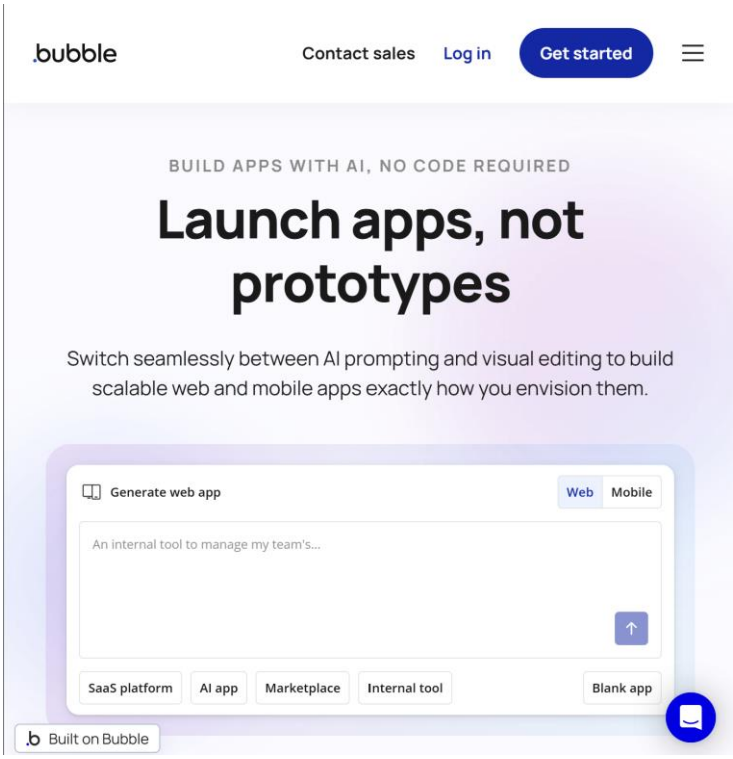


Рисунок 1.2 – Bubble

Bubble дозволяє проектувати фронтенд, будувати бекенд-процеси, керувати реляційними базами даних і інтегрувати сторонні API через візуальний редактор. З 2025 року доступний конструктор нативних iOS і Android застосунків у форматі публічної бета-версії. Однак платформа має криву навчання та систему «workload units», що може суттєво збільшити витрати при масштабуванні, а вихідний код не підлягає експорту (vendor lock-in) [21].

1.3.3 Webflow

Webflow (рис. 1.3) – No-Code платформа, орієнтована на веб-дизайн і маркетингові сайти. Станом на 2026 рік вона обслуговує понад 500 000 живих веб-сайтів і займає 1,2 % частки CMS-ринку з темпом зростання виручки 66 % у 2024 році [22]. Основна цінність – візуальне середовище дизайну, що генерує чистий HTML, CSS та JavaScript. Для корпоративних застосунків зі складною логікою чи безпечним управлінням користувачами Webflow не є оптимальним рішенням, оскільки залишається насамперед інструментом веб-дизайну [23].

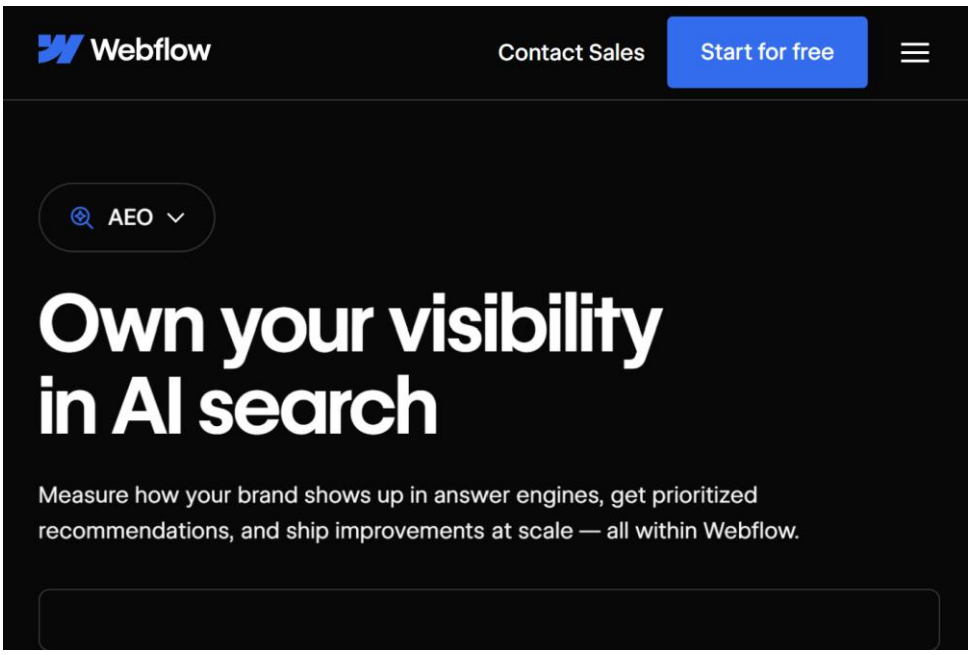


Рисунок 1.3 – Webflow

1.3.4 AppSheet

AppSheet (рис. 1.4) – No-Code платформа від Google, що дозволяє створювати мобільні та веб-застосунки безпосередньо з джерел даних (Google Sheets, Excel, SQL, Salesforce) [24].

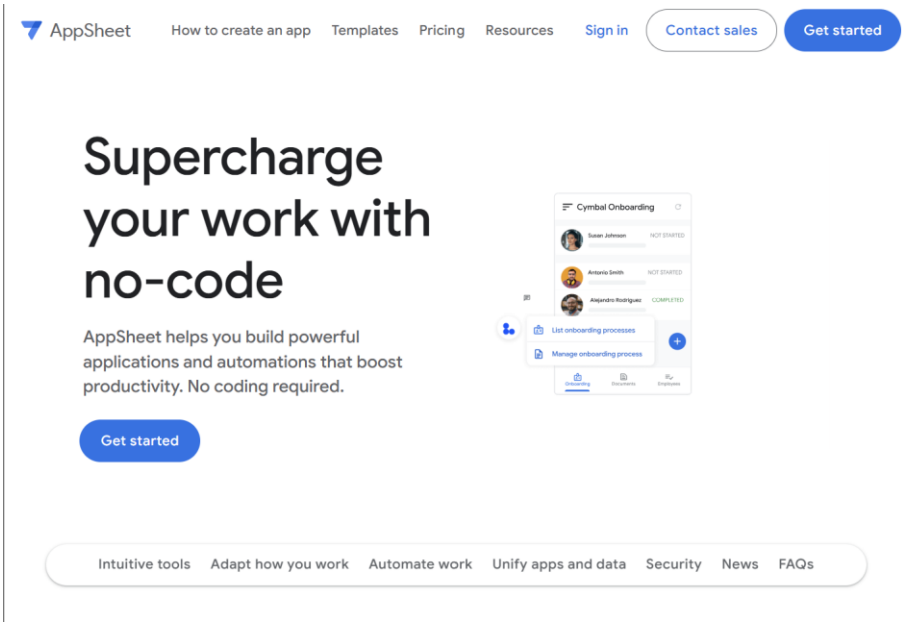


Рисунок 1.4 – AppSheet

Ключова сила – інтеграція з Google Workspace, підтримка офлайн-синхронізації та AI-функції на базі машинного навчання Google. Серед обмежень – менш гнучкий візуальний дизайн порівняно з Glide та заплутана цінова політика.

Отримані дані в результаті аналізу наведені в таблиці 1.1.

Таблиця 1.1 – Порівняння провідних No-Code платформ

Критерій	Glide	Bubble	Webflow	AppSheet
Призначення	Застосунки на основі спредшитів	Складні веб- та мобільні застосунки	Маркетингові сайти, CMS	Корпоративні застосунки з даних
Джерела даних	Google Sheets, Excel, Airtable	Внутрішня БД, API, зовнішні джерела	CMS Webflow	Google Sheets, Excel, SQL
Рівень складності	Низький	Середній–Високий	Низький–Середній	Середній
Мобільність	PWA (кросплатформний)	PWA + нативний iOS/Android (2025)	Адаптивний сайт	Android, iOS
Підтримка AI	Glide AI (генерація з промпту)	AI App Builder (прототипи)	Обмежена	ML-функції Google
Офлайн-режим	Відсутній	Відсутній	Відсутній	Наявний
Ціна (від)	Безкоштовний / $249/міс	Безкоштовний / $29/міс	Безкоштовний / $14/міс	Безкоштовний / $5/кор/міс
Кастомізація	Середня	Висока	Дуже висока	Низька
Масштабованість	Середня	Висока	Середня	Висока
Крива навчання	Низька	Висока	Середня	Середня
Vendor lock-in	Середній	Є (код не експортується)	Немає (код виводиться)	Помірний (Google)

Glide є оптимальним рішенням для команд, що прагнуть швидко перетворити наявні дані зі спредшитів на зручний застосунок. Платформа забезпечує найнижчий поріг входу, швидке розгортання та нативну інтеграцію з Google Sheets [18]. Вбудований Glide AI дає змогу генерувати застосунки з описів природною мовою [19], що робить її привабливою для не-технічних спеціалістів і малого бізнесу.

Bubble підходить для розробки складних продуктів із нестандартною логікою, де потрібен повний контроль над бекендом і базою даних. Саме на

ньому будуються повноцінні SaaS-продукти та маркетплейси, що підтверджується понад 7,2 млн застосунків, створених на платформі [20].

Webflow є ідеальним вибором для дизайнерів і маркетологів, яким потрібен максимальний контроль над зовнішнім виглядом сайту. Для задач, що виходять за межі веб-дизайну й потребують складної бізнес-логіки, платформа має суттєві обмеження [23].

AppSheet найкраще інтегрується в організаціях, що активно використовують Google Workspace, особливо для автоматизації польових операцій і управління завданнями завдяки підтримці офлайн-режиму [24].

Отже, жодна з розглянутих платформ не є універсальним рішенням для всіх випадків. Вибір No-Code інструменту повинен базуватися на типі застосунку, наявних джерелах даних, вимогах до кастомізації та технічному рівні команди. Для розробки інформаційних систем із табличною структурою даних, що орієнтовані на швидке впровадження та не потребують складного бекенду, Glide залишається одним із найбільш збалансованих рішень на ринку у 2024–2025 роках.

2 ПРОЄКТУВАННЯ MVP-СИСТЕМИ ДЛЯ АВТОМАТИЗАЦІЇ ОРГАНІЗАЦІЙНО-УПРАВЛІНСЬКИХ ПРОЦЕСІВ

2.1 Аналіз предметної області

Організаційно-управлінські процеси (ОУП) являють собою сукупність взаємопов’язаних дій, спрямованих на планування, координацію та контроль ресурсів організації для досягнення її цілей. У сучасних умовах цифрової трансформації значна частина таких процесів все ще виконується вручну або із використанням розрізнених інструментів (Excel, месенджери, паперові документи), що знижує ефективність управління, створює ризики втрати даних та ускладнює контроль виконання завдань.

У межах аналізу предметної області було встановлено, що ключовою проблемою є відсутність єдиної інформаційної системи, яка б об’єднувала всі основні управлінські процеси організації: роботу зі співробітниками, управління проєктами, контроль завдань та обробку внутрішніх запитів (табл. 2.1). Це обґрунтовує необхідність розробки MVP-системи OrgFlow на базі No-Code платформи.

Таблиця 2.1 – Основні проблеми організаційно-управлінських процесів та шляхи їх вирішення

№	Проблема	Наслідки	Рішення в рамках MVP
1	Розрізненість даних про співробітників, проєкти та задачі	Дублювання інформації, помилки, втрата даних	Створення централізованої бази даних
2	Відсутність системного контролю виконання завдань	Пропущені дедлайни, низька дисципліна	Впровадження модулю задач зі статусами та термінами
3	Неформалізований процес погодження запитів	Затримки у прийнятті рішень	Автоматизований workflow запитів
4	Відсутність прозорості навантаження працівників	Нерівномірний розподіл задач	Аналітика та зв’язок завдань із виконавцями
5	Обмежений доступ до системи поза офісом	Зниження оперативності	Кросплатформений мобільний доступ через Glide

У процесі аналізу організаційно-управлінських процесів було визначено низку типових проблем, які виникають у разі використання розрізнених інструментів (таблиць, месенджерів, паперових документів) та відсутності єдиної інформаційної системи. Ці проблеми безпосередньо впливають на ефективність управління персоналом, проєктами та задачами в організації.

По-перше, спостерігається розрізненість даних про співробітників, проєкти та задачі, що призводить до дублювання інформації, появи помилок та втрати актуальності даних. Відсутність єдиного джерела правди ускладнює координацію між підрозділами. Рішенням у межах MVP є створення централізованої бази даних, у якій всі сутності взаємопов’язані між собою.

По-друге, відсутній системний контроль виконання завдань, через що виникають пропущені дедлайни та зниження виконавської дисципліни. Це пов’язано з тим, що задачі часто фіксуються неструктуровано або в різних каналах комунікації. У запропонованому MVP ця проблема вирішується через модуль задач зі статусами, дедлайнами та відповідальними виконавцями.

Третьою проблемою є неформалізований процес погодження запитів (відпустки, ресурси, навчання тощо), що призводить до затримок у прийнятті управлінських рішень. У рамках системи передбачається автоматизований workflow погодження, який забезпечує прозорий рух запиту від ініціатора до відповідальної особи.

Четверта проблема полягає у недостатній прозорості навантаження працівників. Керівництво не має повної картини щодо того, хто і в якій мірі завантажений задачами, що призводить до нерівномірного розподілу роботи. У MVP це вирішується через аналітику та зв’язок задач із конкретними виконавцями.

П’ята проблема – обмежений доступ до системи поза межами робочого місця. Це знижує оперативність реагування на зміни та ускладнює управління в умовах мобільної роботи. Запропоноване рішення – кросплатформений доступ

до системи через Glide-застосунок, який працює на будь-яких пристроях без необхідності встановлення окремого програмного забезпечення.

У результаті аналізу предметної області було визначено три основні категорії користувачів системи OrgFlow: адміністратор, менеджер та співробітник. Кожна роль має власний рівень доступу та функціональні обов'язки, що дозволяє реалізувати модель розмежування прав у системі.

Для визначення напрямів автоматизації було проведено аналіз існуючих бізнес-процесів та їх цільового стану після впровадження MVP-системи (табл. 2.2).

Таблиця 2.2 – Аналіз бізнес-процесів «як є» та «як буде»

Процес	**As-Is (як є)**	**To-Be (як буде)**
Постановка завдань	Передача через месенджери або усно	Створення структурованих задач у системі
Контроль виконання	Ручний контроль керівником	Автоматичні статуси та дедлайни
Подача запитів	Паперові або неформальні заявки	Електронні форми запитів у застосунку
Звітність	Ручне формування Excel-звітів	Автоматичні дашборди та аналітика

Таблиця відображає порівняння поточного стану організаційно-управлінських процесів (As-Is) та їхнього цільового стану після впровадження MVP-системи (To-Be). Вона показує, що нині більшість процесів виконується вручну або через неструктуровані канали комунікації, що знижує ефективність, прозорість і контроль.

Після впровадження системи процеси переходять у цифровий формат: завдання створюються та відстежуються централізовано, контроль виконання здійснюється автоматично через статуси й дедлайни, запити оформлюються через електронні форми, а звітність формується у вигляді автоматичних аналітичних дашбордів. Це забезпечує більшу структурованість, швидкість обробки інформації та зменшення ризику помилок.

Предметна область характеризується високою залежністю від ручних операцій, відсутністю єдиного інформаційного середовища та низьким рівнем

автоматизації. Це призводить до неефективного використання часу, помилок у даних та ускладнення управлінського контролю. Впровадження MVP-системи OrgFlow дозволяє вирішити ці проблеми шляхом централізації даних, автоматизації ключових процесів та забезпечення прозорої взаємодії між учасниками організації.

Таким чином, результати аналізу предметної області є основою для подальшого проєктування структури системи, моделювання бази даних та реалізації MVP на платформі No-code.

2.2 Проєктування структури та функціоналу MVP-системи

Проєктування MVP-системи OrgFlow здійснюється з урахуванням принципів модульної архітектури, централізованого зберігання даних та мінімально життєздатного функціоналу, достатнього для перевірки ключових гіпотез щодо ефективності управління внутрішніми процесами організації.

Основною метою даного етапу є формування логічної структури системи, визначення її функціональних компонентів, а також опис ключових сценаріїв використання та архітектурних рівнів реалізації.

Архітектура MVP-системи OrgFlow базується на модульному підході, що дозволяє розділити функціональність на окремі логічні блоки. Усі модулі взаємодіють через єдину централізовану базу даних, що забезпечує узгодженість інформації та зменшує ризики дублювання даних.

До складу системи входять п’ять ключових модулів:

- модуль відділів;
- модуль співробітників;
- модуль проєктів;
- модуль завдань;
- модуль запитів.

Ці компоненти формують єдину екосистему управління організаційними процесами, де кожен модуль виконує окрему функціональну роль, але водночас взаємодіє з іншими через спільну структуру даних.

Модуль «Відділи» відповідає за організаційну структуру компанії та управління підрозділами. Модуль «Співробітники» забезпечує ведення профілів працівників та їх прив'язку до відповідних відділів. Модуль «Проєкти» використовується для створення та супроводу проєктів, тоді як модуль «Завдання» реалізує оперативне управління виконанням робіт. Модуль «Запити» забезпечує процеси погодження та внутрішньої комунікації.

Функціональні вимоги (табл. 2.3) визначають мінімально необхідний набір можливостей кожного модуля в межах MVP-версії системи. Вони сформовані за пріоритетною методологією MoSCoW, що дозволяє розділити функції на обов'язкові, бажані та другорядні.

Таблиця 2.3 – Функціональні вимоги до модулів системи OrgFlow

Модуль	**Основні функції**	**Пріоритет (MoSCoW)**
Відділи	CRUD відділів, призначення керівника	Must have
Співробітники	Профілі, контакти, прив'язка до відділу	Must have
Проєкти	Створення, статус, прогрес, бюджет	Must have
Завдання	Призначення, статуси, пріоритети, дедлайни	Must have
Запити	Подача, погодження, типи запитів	Should have
Аналітика	Зведення по навантаженню	Could have
Сповіщення	Нагадування про дедлайни	Could have

Таблиця відображає функціональний склад MVP-системи та пріоритизацію її модулів відповідно до підходу MoSCoW. Основу системи формують модулі, що мають статус Must have, оскільки вони забезпечують ключові процеси управління організацією: ведення відділів, облік співробітників, управління проєктами та виконання завдань. Саме ці компоненти є критичними для працездатності системи.

Модуль «Запити» віднесений до категорії Should have, оскільки він розширює функціональність системи та підтримує процеси внутрішнього погодження, але не є критичним для базового запуску MVP.

Модулі «Аналітика» та «Сповіщення» мають статус Could have і розглядаються як додаткові можливості, що підвищують зручність користування системою. Вони можуть бути реалізовані на подальших етапах розвитку продукту без впливу на базову функціональність.

Для опису взаємодії користувачів із системою було побудовано діаграму варіантів використання (рис. 2.1), яка відображає основні ролі та їхні дії в межах MVP-системи.



Рисунок 2.1 – Use Case діаграма взаємодії користувачів із системою OrgFlow

На верхньому рівні знаходиться адміністратор, який має повний доступ до всіх даних, включаючи налаштування системи, управління відділами та перегляд аналітики. На середньому рівні розташований менеджер, який відповідає за створення проєктів, призначення завдань, а також погодження запитів співробітників. На нижньому рівні знаходиться співробітник, який взаємодіє лише зі своїми завданнями та може подавати запити (відпустка, ресурси, навчання). Така структура забезпечує чіткий розподіл відповідальності та контроль доступу до інформації.

Архітектура MVP-системи OrgFlow (табл. 2.4) побудована за багаторівневим принципом, що дозволяє розділити відповідальність між рівнями представлення, бізнес-логіки та зберігання даних. Такий підхід забезпечує масштабованість системи та спрощує її подальший розвиток.

Таблиця 2.4 – Архітектурні рівні MVP-системи OrgFlow

Рівень	**Компоненти**	**Опис**
Рівень представлення	Glide UI (екрани, списки, форми)	Інтерфейс взаємодії користувача
Рівень логіки	Computed Columns, Relations, Actions, Workflows	Обробка даних та бізнес-логіка
Рівень доступу	Role-based Access	Розмежування прав користувачів
Рівень даних	Google Sheets / Glide Tables	Централізоване зберігання даних

Таблиця відображає багаторівневу архітектуру MVP-системи, де кожен рівень виконує окрему функціональну роль. Рівень представлення відповідає за інтерфейс користувача, через який здійснюється взаємодія з системою. Рівень логіки забезпечує обробку даних і реалізацію бізнес-правил.

Рівень доступу визначає права користувачів і контролює їхні можливості в системі, а рівень даних відповідає за централізоване зберігання та структуру інформації. У сукупності ці рівні формують цілісну архітектуру, що забезпечує стабільну роботу MVP.

Отже, у межах даного розділу було спроєктовано структуру MVP-системи OrgFlow, визначено її модульну архітектуру, функціональні вимоги, сценарії використання та загальну технічну побудову. Запропонована модель забезпечує баланс між простотою реалізації та функціональною достатністю, що відповідає концепції MVP та дозволяє реалізувати систему з можливістю подальшого масштабування.

2.3 Проєктування бази даних та логіки роботи системи

Проєктування бази даних є ключовим етапом розробки MVP-системи, оскільки саме воно визначає структуру зберігання інформації, взаємозв’язки між сутностями та ефективність подальшої обробки даних. Для системи управління організаційними процесами було обрано реляційний підхід, який дозволяє чітко структурувати дані у вигляді взаємопов’язаних таблиць.

Концептуальна модель (рис. 2.2) системи відображає основні сутності предметної області та зв’язки між ними. У центрі моделі знаходиться організаційна структура, яка включає відділи, співробітників, проєкти, завдання та запити.

Як видно з ER-моделі, система побудована за принципом ієрархічної взаємодії сутностей. Відділи виступають базовим організаційним рівнем, який об’єднує співробітників та проєкти. У свою чергу, проєкти містять набір завдань, а співробітники можуть бути як виконавцями завдань, так і ініціаторами запитів

Логічна модель бази даних деталізує ER-діаграму та визначає конкретні атрибути кожної сутності, а також їх функціональне призначення в межах системи. Усі таблиці взаємопов’язані між собою та утворюють єдину структуру даних, яка забезпечує підтримку основних бізнес-процесів MVP-системи.

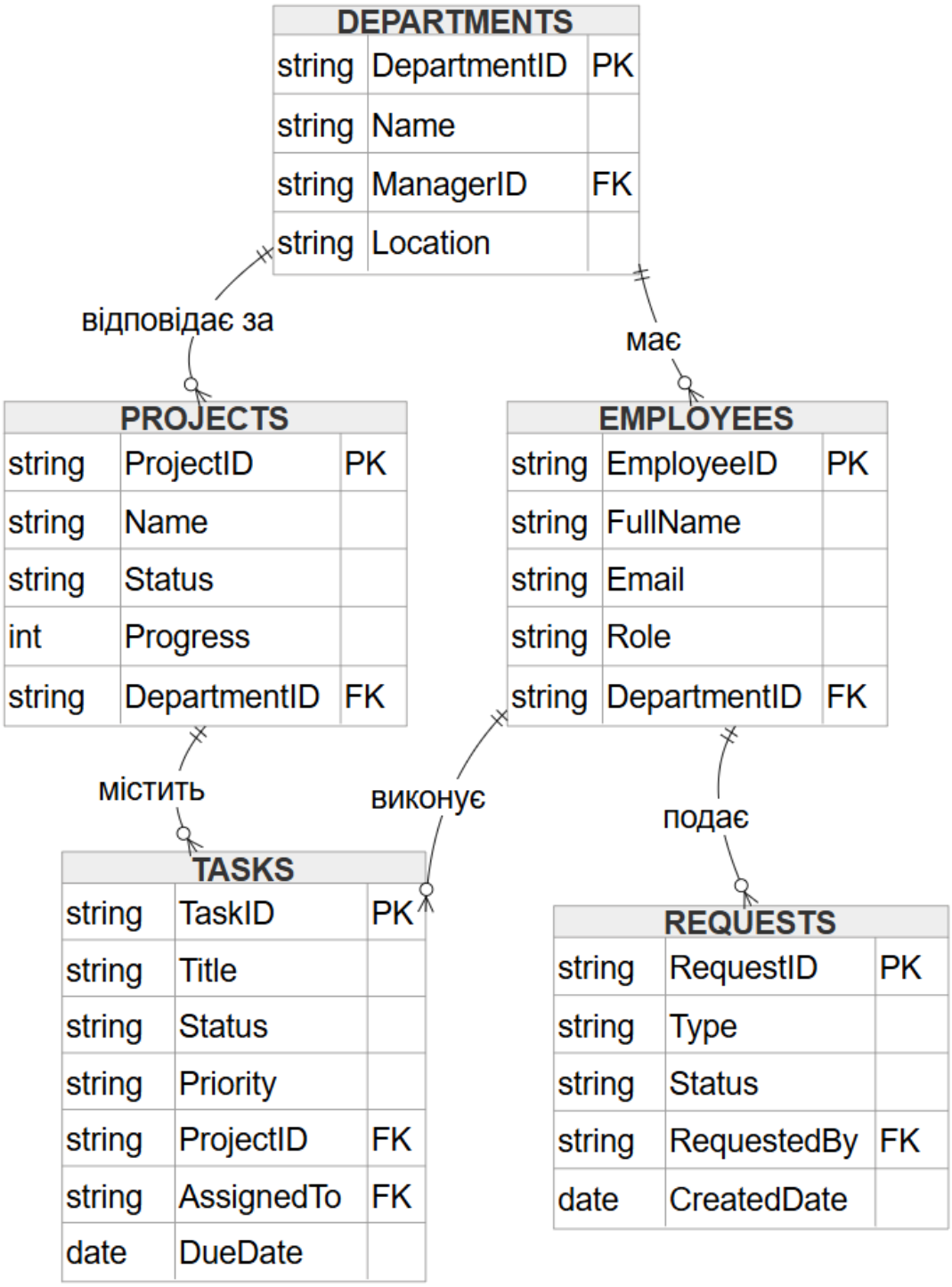


Рисунок 2.2 – ER-діаграма бази даних системи

Таблиця «Departments» (табл. 2.5) є однією з базових сутностей системи, оскільки саме вона формує організаційну структуру підприємства. Поле DepartmentID використовується як первинний ключ і забезпечує унікальність кожного запису, що дозволяє однозначно ідентифікувати конкретний відділ у системі. Назва відділу (Name) використовується для відображення структурних

підрозділів у користувацькому інтерфейсі та забезпечує зрозумілу навігацію для користувачів. Поле ManagerID реалізує зв’язок із таблицею співробітників і визначає відповідального керівника, що дозволяє моделювати ієрархічну структуру управління. Локація (Location) додає додатковий описовий рівень і використовується для сегментації відділів за фізичним або функціональним розташуванням.

Таблиця 2.5 – «Departments» (Відділи)

Поле	Тип	Опис
DepartmentID	Text (PK)	Унікальний ідентифікатор відділу
Name	Text	Назва відділу
ManagerID	Text (FK)	Ідентифікатор керівника
Location	Text	Локація або підрозділ

Таблиця «Employees» (табл. 2.6) використовується для зберігання всієї інформації про користувачів системи та їх належність до організаційної структури. EmployeeID виконує функцію первинного ключа та дозволяє унікально ідентифікувати кожного співробітника.

Поле FullName застосовується для відображення повного імені користувача в інтерфейсі та звітності. Email використовується як основний контактний канал та потенційно може виконувати роль логіну для авторизації. Role визначає рівень доступу користувача в системі та є ключовим елементом реалізації role-based access control.

DepartmentID забезпечує зв’язок із таблицею відділів і визначає структурну належність співробітника, що дозволяє коректно організувати ієрархію та розподіл задач.

Таблиця 2.6 – «Employees» (Співробітники)

Поле	Тип	Опис
EmployeeID	Text (PK)	Унікальний ідентифікатор співробітника
FullName	Text	Повне ім’я
Email	Email	Корпоративна електронна адреса
Role	Text	Роль у системі (Admin, Manager, Employee)
DepartmentID	Text (FK)	Прив’язка до відділу

Таблиця «Projects» (табл. 2.7) відповідає за управління проєктною діяльністю в системі та є проміжною ланкою між відділами і завданнями. ProjectID забезпечує унікальність кожного проєкту. Name використовується для ідентифікації проєкту в інтерфейсі та звітності. Status дозволяє відслідковувати життєвий цикл проєкту, від планування до завершення. Progress відображає ступінь виконання у відсотках і використовується для аналітики. DepartmentID визначає, який саме відділ відповідає за реалізацію проєкту, що забезпечує організаційний контроль та структурованість даних.

Таблиця 2.7 – «Projects» (Проєкти)

Поле	Тип	Опис
ProjectID	Text (PK)	Унікальний ідентифікатор проєкту
Name	Text	Назва проєкту
Status	Choice	Статус (Planned / Active / Completed)
Progress	Number	Відсоток виконання
DepartmentID	Text (FK)	Відповідальний відділ

Таблиця «Tasks» (табл. 2.8) є центральною операційною сутністю системи, оскільки саме через неї реалізується виконання основних бізнес-процесів. TaskID виступає у ролі унікального ідентифікатора завдання. Title використовується для короткого опису суті роботи. Status відображає поточний етап виконання завдання та дозволяє контролювати робочий процес. Priority визначає важливість і терміновість виконання, що допомагає в управлінні навантаженням. ProjectID встановлює зв’язок із відповідним проєктом, забезпечуючи структуровану ієрархію. AssignedTo визначає відповідального виконавця, а DueDate задає часові рамки виконання.

Таблиця 2.8 – «Tasks» (Завдання)

Поле	Тип	Опис
TaskID	Text (PK)	Унікальний ідентифікатор завдання
Title	Text	Назва завдання
Status	Choice	Статус (To Do / In Progress / Done)
Priority	Choice	Пріоритет (Low / Medium / High)
ProjectID	Text (FK)	Прив’язка до проєкту
AssignedTo	Text (FK)	Виконавець
DueDate	Date	Дедлайн

Таблиця «Requests» (табл. 2.9) використовується для обробки внутрішніх запитів співробітників і підтримує адміністративні процеси системи. RequestID забезпечує унікальність кожного запиту. Type визначає його категорію, що дозволяє системі диференціювати різні бізнес-сценарії. Status відображає поточний стан обробки запиту. RequestedBy встановлює зв’язок із таблицею співробітників і дозволяє визначити автора запиту. CreatedDate фіксує момент створення, що є важливим для контролю та аудиту.

Таблиця 2.9 – «Requests» (Запити)

Поле	Тип	Опис
RequestID	Text (PK)	Унікальний ідентифікатор запиту
Type	Text	Тип запиту (відпустка, погодження тощо)
Status	Text	Статус (Pending / Approved / Rejected)
RequestedBy	Text (FK)	Ініціатор запиту
CreatedDate	Date	Дата створення

Для забезпечення цілісності даних у системі реалізовано набір зв’язків (табл. 2.10) типу «один-до-багатьох», що дозволяє моделювати реальні організаційні процеси.

Таблиця 2.10 – Зв’язки в базі даних

Зв’язок	Тип	Опис
Departments – Employees	1:N	Один відділ містить багато співробітників
Departments – Projects	1:N	Один відділ керує кількома проєктами
Projects – Tasks	1:N	Один проєкт складається з багатьох завдань
Employees – Tasks	1:N	Співробітник може виконувати багато завдань
Employees – Requests	1:N	Співробітник може створювати багато запитів

Зв’язки між таблицями реалізовані за принципом «один-до-багатьох», що забезпечує логічну ієрархію даних та відображає реальні організаційні процеси. Зокрема, один відділ може містити багато співробітників, а також керувати кількома проєктами, що дозволяє структурувати систему на рівні підрозділів. У свою чергу, кожен проєкт складається з множини завдань, що відображає декомпозицію проєктної діяльності на конкретні робочі елементи.

Крім того, співробітники можуть бути пов’язані з великою кількістю завдань, що забезпечує гнучкість у розподілі навантаження та управлінні виконанням робіт. Також кожен співробітник має можливість створювати численні запити, що формує окремий адміністративний контур взаємодії із системою. У сукупності такі зв’язки забезпечують цілісність даних, виключають дублювання інформації та підтримують узгоджену роботу всіх модулів системи.

Для моделювання процесу виконання завдань у системі було розроблено діаграму станів (рис. 2.3), яка відображає повний життєвий цикл завдання від моменту його створення до завершення або повернення на доопрацювання.



Рисунок 2.3 – Діаграма станів «Життєвий цикл завдання»

Як видно з діаграми, процес починається зі створення завдання, після чого воно переходить у початковий статус To Do, що означає його формування та постановку в чергу виконання. На цьому етапі визначаються основні параметри завдання, такі як відповідальний виконавець, пріоритетність та дедлайн, що дозволяє структуровано організувати подальшу роботу.

Далі завдання переходить у стан In Progress, який відображає фазу активного виконання. У цей період користувач безпосередньо працює над реалізацією поставленого завдання, а система фіксує прогрес його виконання.

Після завершення основної роботи завдання переходить у статус Review, де відбувається перевірка результату менеджером або відповідальною особою. Це критичний етап контролю якості, на якому оцінюється відповідність виконаного результату початковим вимогам.

У разі успішної перевірки завдання переводиться у фінальний стан Done, що означає його повне завершення. Водночас у випадку виявлення помилок або невідповідностей завдання може бути повернуте на попередній етап для доопрацювання, що забезпечує ітеративність процесу та підвищує якість виконання робіт.

Окрім управління завданнями, система передбачає окремий бізнес-процес погодження внутрішніх запитів (рис. 2.4). Для його відображення використовується діаграма послідовності, яка демонструє взаємодію між співробітником, системою та менеджером.

Процес починається з того, що співробітник ініціює створення запиту, наприклад на відпустку або інший внутрішній ресурс. Після подання запиту система автоматично присвоює йому статус Pending, що означає очікування розгляду, та одночасно надсилає сповіщення менеджеру про появу нового запиту. Наступним етапом є розгляд запиту менеджером, який аналізує його зміст та приймає управлінське рішення. У разі схвалення запиту система змінює його статус на Approved, після чого співробітник отримує відповідне сповіщення про позитивне рішення.

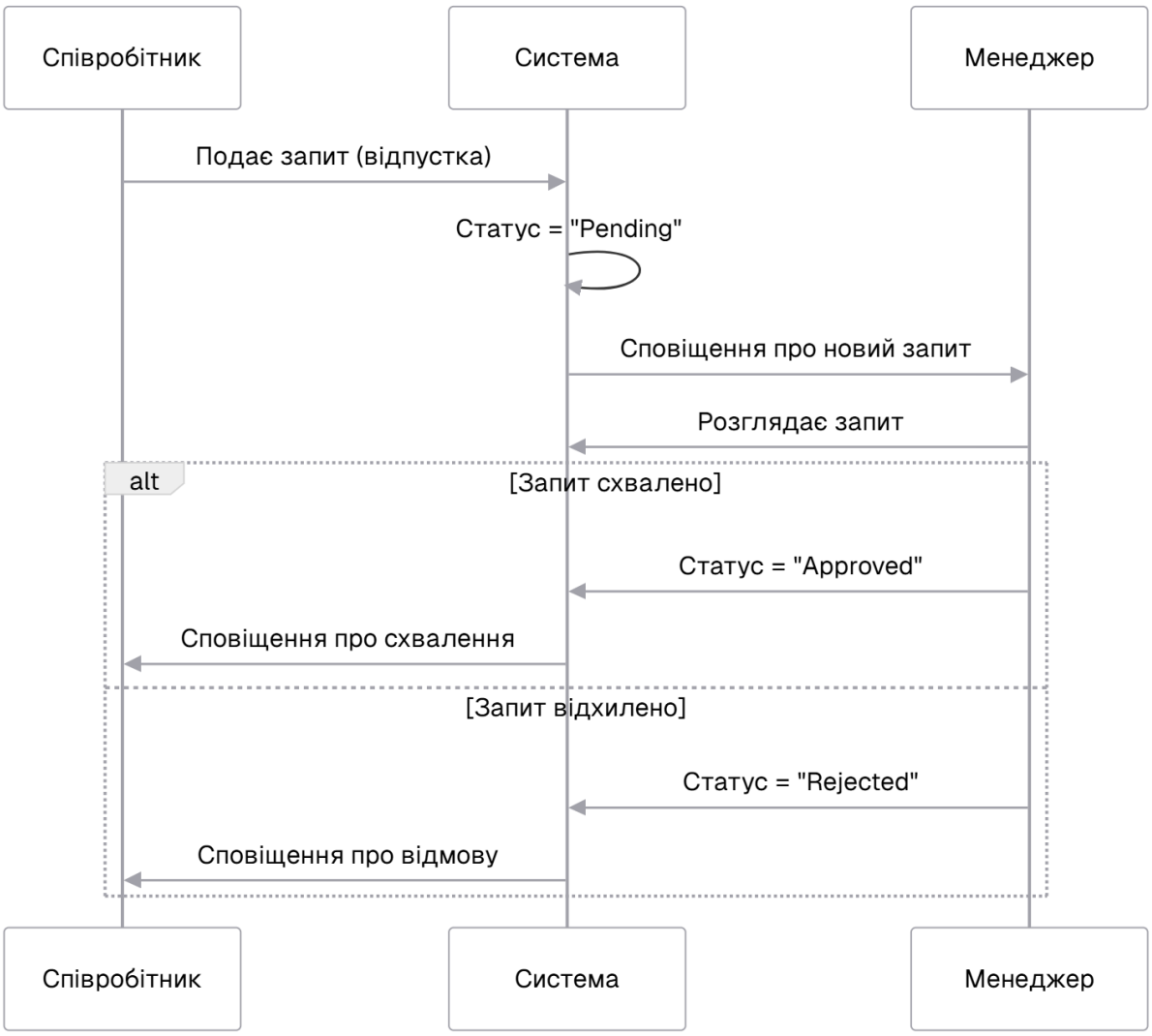


Рисунок 2.4 – Діаграма послідовності «Погодження запиту»

Якщо ж запит не відповідає встановленим критеріям або потребує відхилення, його статус змінюється на Rejected, і система також інформує співробітника про результат розгляду. Таким чином забезпечується повний замкнений цикл взаємодії між учасниками процесу, а також прозорість і контроль усіх етапів погодження.

Таким чином, розроблена структура бази даних та логіка роботи системи забезпечують цілісність інформації, підтримують основні бізнес-процеси організації та дозволяють реалізувати масштабовану MVP-архітектуру. Взаємозв’язки між сутностями побудовані таким чином, щоб мінімізувати надлишковість даних та забезпечити ефективну обробку інформації в режимі реального часу.

2.4 Проєктування інтерфейсу користувача

У процесі проєктування інтерфейсу користувача було сформовано карту екранів (рис. 2.5), яка відображає загальну структуру навігації системи та взаємозв’язок між основними функціональними модулями. Вона демонструє, як користувач переходить між ключовими розділами застосунку – від авторизації до роботи з завданнями, проєктами, співробітниками та запитами.

Ця діаграма відображає загальну структуру навігації застосунку та взаємозв’язок між його основними екранами. Після входу в систему користувач потрапляє на головний екран (Dashboard), який виконує роль центральної точки доступу до всіх функціональних модулів. Звідси здійснюється перехід до ключових розділів: завдань, проєктів, співробітників, запитів та особистого профілю.

Кожен із основних розділів має власні підрівні деталізації. У модулі завдань користувач може переглядати детальну інформацію про конкретне завдання, у розділі проєктів – відкривати деталі проєкту, у блоці співробітників – переглядати профіль окремого працівника. Модуль запитів передбачає як перегляд детальної інформації про конкретний запит, так і створення нового запиту через окрему форму. Така структура забезпечує логічну ієрархію інтерфейсу та зручну навігацію між рівнями даних.

Проєктування інтерфейсу користувача базується на сучасних принципах UX-дизайну, основною метою яких є забезпечення простоти, ефективності та зручності використання системи. Інтерфейс розроблено за підходом mobile-first, що дозволяє коректно працювати з ним як на мобільних пристроях, так і на настільних комп’ютерах, забезпечуючи адаптивність та гнучкість у використанні.

Принцип простоти реалізовано через мінімалістичну структуру навігації, у якій користувач взаємодіє лише з основними модулями системи без перевантаження зайвими елементами. Візуальна ієрархія досягається завдяки використанню карткових компонентів, акцентів на ключових даних та кольорового виділення статусів і пріоритетів, що дозволяє швидко орієнтуватися в інформації.

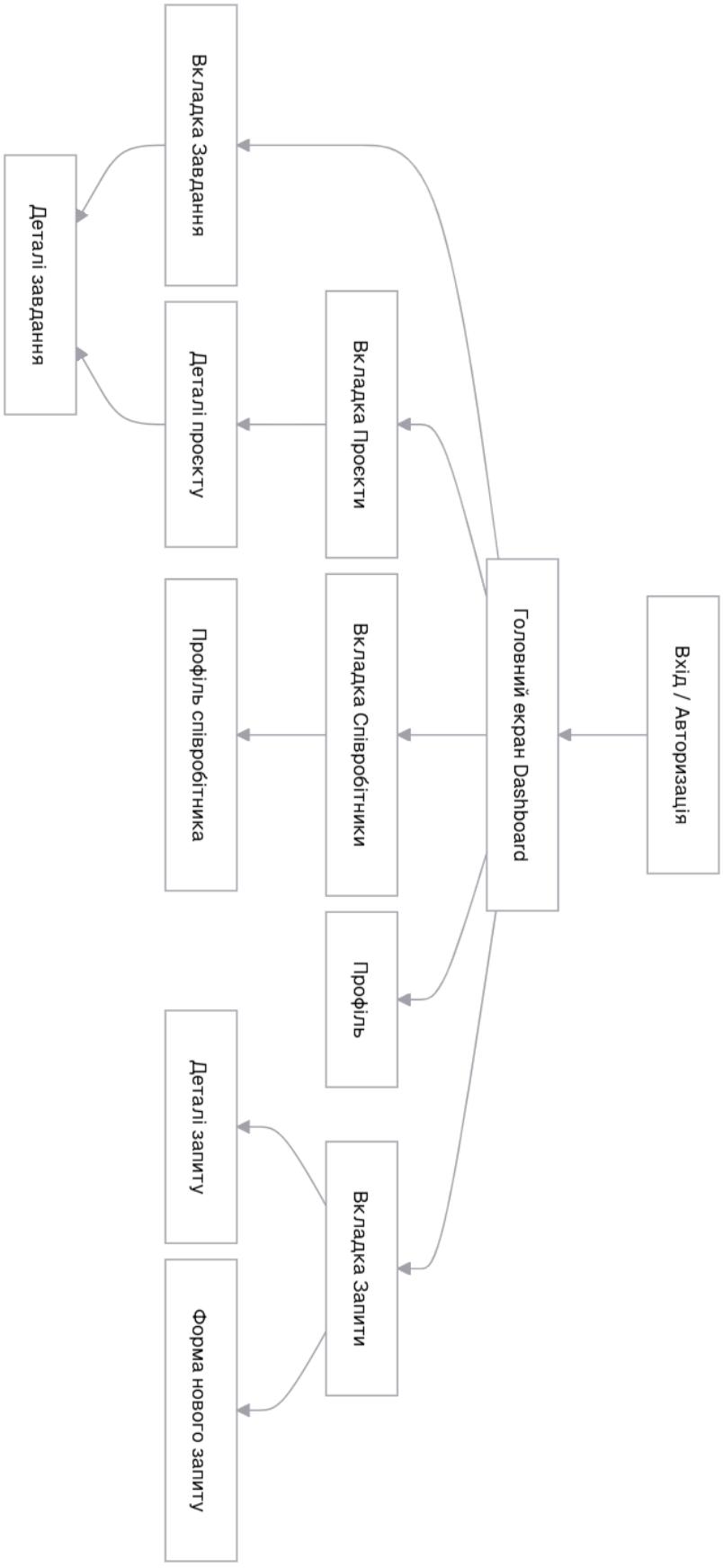


Рисунок 2.5 – Карта екранів застосунку

Зворотний зв’язок у системі реалізовано через миттєві повідомлення у форматі toast-сповіщень, які інформують користувача про результат виконаної дії, а також підвищують прозорість взаємодії із системою. Консистентність інтерфейсу забезпечується використанням єдиного стилю компонентів у всіх розділах застосунку, що формує цілісне та зрозуміле середовище.

Головний екран системи (Dashboard) виконує роль центрального вузла взаємодії користувача із застосунком. У верхній частині розміщується блок привітання, який формує персоналізований досвід використання системи. Далі йде блок KPI-карток, що відображають ключові показники ефективності, зокрема кількість активних завдань і проєктів. Центральна частина екрану містить список завдань на поточний день, що дозволяє користувачу швидко переходити до виконання роботи, а нижче розміщується блок останніх запитів для контролю адміністративних процесів. Завершує структуру нижня навігаційна панель, яка забезпечує швидкий доступ до основних модулів системи.

Для покращення сприйняття інформації в інтерфейсі застосовується система кольорового кодування, що дозволяє швидко ідентифікувати статуси завдань та їх пріоритетність без додаткового аналізу тексту. Червоний колір використовується для критичних станів, таких як високий пріоритет або прострочені завдання, жовтий – для середнього рівня пріоритету, а зелений – для низького пріоритету або виконаних завдань. Такий підхід підвищує швидкість сприйняття інформації та зменшує когнітивне навантаження користувача.

Логіка взаємодії користувача із системою побудована на розділенні ролей (рис. 2.6), що дозволяє адаптувати інтерфейс під конкретні потреби менеджера та співробітника. Після входу в систему користувач потрапляє на головний екран, після чого система визначає його роль і формує відповідний сценарій роботи.

Менеджер отримує доступ до функцій управління, зокрема створення завдань та погодження запитів, тоді як співробітник працює зі своїми

завданнями, оновлює їх статуси та подає запити. Такий поділ забезпечує чітку структуру прав доступу та оптимізує робочі процеси.

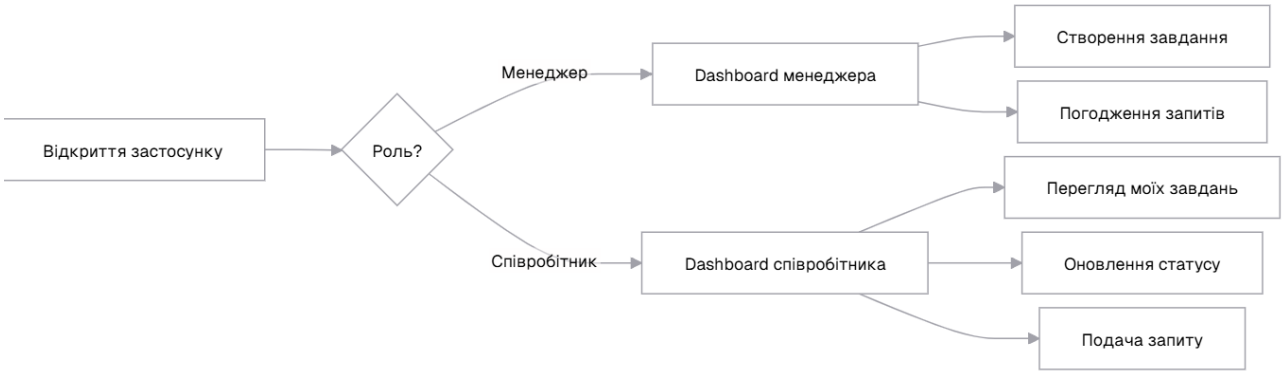


Рисунок 2.6 – Навігаційний потік користувача (User Flow)

У результаті проєктування інтерфейсу користувача було сформовано структуровану та логічну систему взаємодії користувача із застосунком. Інтерфейс побудовано на принципах простоти, адаптивності та консистентності, що забезпечує зручну навігацію між основними модулями системи та швидкий доступ до ключової функціональності.

Особливу увагу приділено організації головного екрана, який виконує роль центрального елементу взаємодії, а також візуальній ієрархії та кольоровому кодуванню, що покращує сприйняття інформації. Запропонований підхід дозволяє підвищити ефективність роботи користувачів і забезпечує інтуїтивно зрозумілий інтерфейс.

3 РЕАЛІЗАЦІЯ MVP-СИСТЕМИ НА ПЛАТФОРМІ GLIDE

3.1 Реалізація структури застосунку в Glide

Реалізація MVP системи «OrgManager» стала першим практичним етапом побудови інформаційної системи та була спрямована на перенесення попередньо спроєктованої логічної моделі в робоче середовище. Для цього було використано платформу Glide, яка дозволяє створювати повноцінні вебзастосунки на основі табличних даних без написання коду. На цьому етапі ключовим завданням було сформувати структуру застосунку, забезпечити коректне підключення джерел даних та автоматичне створення навігації між основними модулями системи.

Після імпорту таблиць система автоматично сформувала базову архітектуру застосунку (рис. 3.1), яка була поділена на основні та допоміжні екрани. Інтерфейс розробленої системи в додатку А наведено. Це дозволило одразу перейти до логічної організації функціональних модулів та визначення ролей кожного розділу. Загальна структура застосунку представлена у вигляді навігаційної моделі.

Як видно з наведеної структури, застосунок організовано за принципом модульності, де кожен розділ відповідає окремій сутності системи. Основна навігація об’єднує ключові бізнес-процеси організації, тоді як додаткове меню забезпечує доступ до допоміжних функцій, зокрема профілю користувача. Такий підхід дозволяє забезпечити логічну цілісність системи та спрощує взаємодію користувача з інтерфейсом.

Після формування загальної структури наступним кроком стала деталізація навігаційних екранів (рис. 3.2) та визначення їх функціонального призначення. Кожен екран було налаштовано відповідно до типу даних, які він обробляє, що забезпечило узгоджену роботу всієї системи.

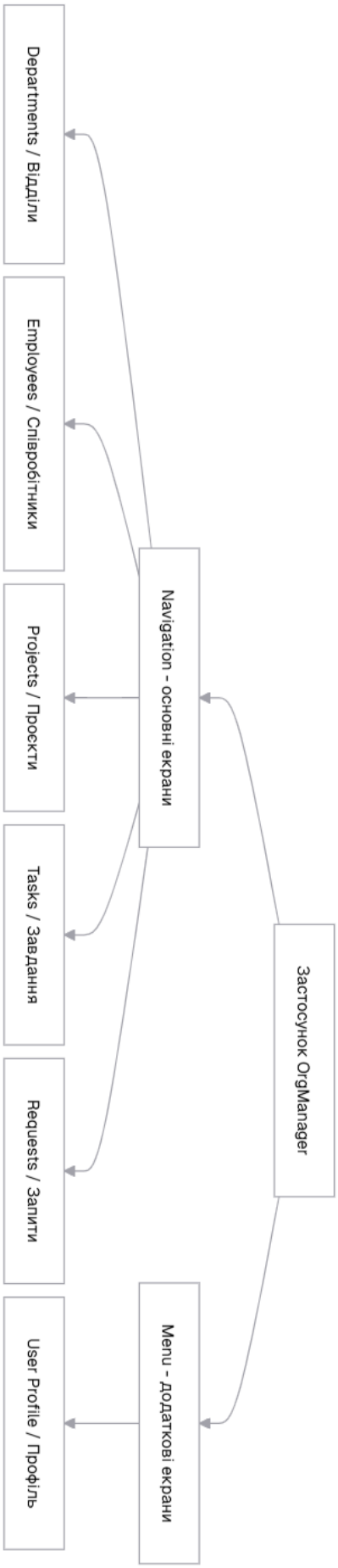


Рисунок 3.1 – Загальна структура застосунку

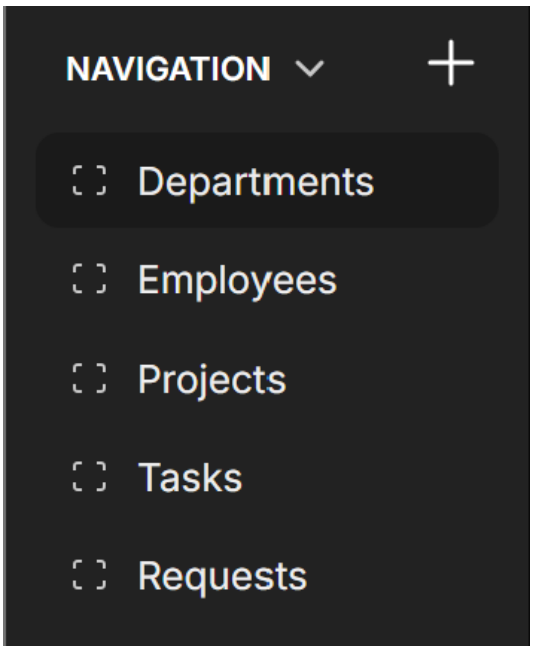


Рисунок 3.2 – Конфігурація навігаційних екранів застосунку

У межах цієї структури екран Departments використовується для відображення організаційної структури підприємства, тоді як Employees забезпечує роботу з інформацією про співробітників. Модуль Projects відповідає за управління проєктами, Tasks – за контроль виконання завдань, а Requests – за обробку внутрішніх запитів. Таким чином, кожен розділ виконує чітко визначену функцію, що сприяє логічному розподілу навантаження між компонентами системи.

Після визначення структури навігації було здійснено налаштування відображення даних у межах кожного екрана. Особлива увага приділялася компоненту Collection, який є основним способом візуалізації табличних даних у Glide. Для кращого розуміння розглянемо налаштування на прикладі екрану Departments.

Після визначення загальної структури навігації наступним кроком стало налаштування відображення даних у кожному з екранних модулів. Особлива увага була приділена компоненту Collection, оскільки саме він у середовищі Glide є основним інструментом для візуалізації табличних даних та забезпечує зручну роботу користувача зі списками записів.

Для прикладу було розглянуто налаштування екрану Departments, де джерелом даних виступає відповідна таблиця Departments. Система автоматично підключає всі записи з цієї таблиці та формує з них структурований список. У межах відображення основне поле DepartmentName використовується як головний заголовок елемента, що дозволяє швидко ідентифікувати назву відділу. Додатково поле Location виконує роль описового елемента та відображає інформацію про розташування підрозділу, що додає контекст і полегшує орієнтацію користувача в структурі організації. Поле Status, у свою чергу, використовується як метаінформація і дозволяє візуально визначати поточний стан запису, наприклад його активність або неактивність.

Важливо зазначити, що Glide надає можливість гнучкого налаштування способів відображення даних у компоненті Collection. Це дозволяє адаптувати інтерфейс під різні типи інформації та сценарії використання системи, залежно від її функціонального призначення.

Завдяки наявності різних стилів відображення – таких як списки, картки, таблиці або канбан-дошки (рис. 3.3) – система може ефективно працювати як із простими довідниковими даними, так і зі складними операційними процесами. Такий підхід забезпечує універсальність інтерфейсу та підвищує зручність взаємодії користувача із системою.

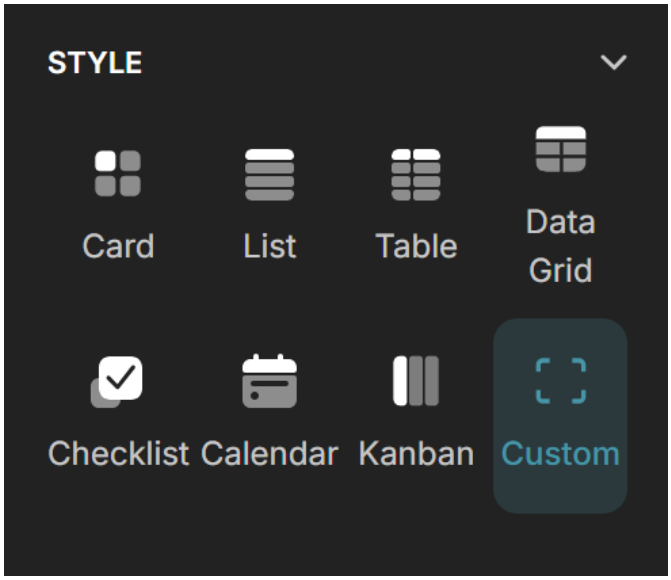


Рисунок 3.3 – Стилі в Glide

У підсумку реалізація структури застосунку в Glide дозволила трансформувати логічну модель бази даних у повноцінний функціональний інтерфейс із чіткою навігацією, модульною архітектурою та адаптивною системою відображення даних. Наступним логічним етапом є деталізація взаємодії між сутностями та реалізація механізмів розмежування доступу користувачів у системі.

3.2 Налаштування бази даних та автоматизації процесів

Після завершення проєктування логічної моделі системи наступним етапом стала її практична реалізація у середовищі Glide Data Editor, де вся структура бази даних була відтворена у вигляді взаємопов’язаних таблиць. Усі сутності попередньо були імпортовані до платформи та автоматично синхронізовані із застосунком «OrgManager», що дозволило одразу сформувати цілісну основу для подальшого налаштування інтерфейсу та реалізації бізнес-логіки.

У процесі роботи було створено п’ять основних таблиць, які відповідають ключовим бізнес-сутностям системи. Кожна з них у середовищі Glide реалізована як окремий Data Sheet, що підтримує як зберігання базових атрибутів, так і розширення через обчислювані стовпці та зв’язки між даними. Важливо зазначити, що після імпорту система автоматично розпізнала ключові ідентифікатори, завдяки чому стало можливим одразу налаштовувати Relation-зв’язки без необхідності додаткового програмування або ручної нормалізації даних.

Базовою сутністю виступає таблиця Departments, яка містить інформацію про організаційні підрозділи. Вона включає унікальний ідентифікатор DepartmentID, що використовується як Primary Key для зв’язків з іншими таблицями системи. Додаткові поля, зокрема назва відділу та його розташування,

застосовуються в інтерфейсі для візуального відображення структури організації та забезпечують користувачу швидку навігацію по підрозділах.

Далі було інтегровано таблицю Employees, яка містить дані про співробітників і є ключовою для реалізації ролей, доступів та прив’язки виконавців до завдань. Під час налаштування в Glide було створено зв’язок із таблицею Departments через DepartmentID, що дозволило автоматично групувати співробітників за підрозділами та використовувати ці дані для побудови динамічних списків і карток у застосунку.

Наступною сутністю стала таблиця Projects, яка відповідає за управління проєктами. Після її імпорту було налаштовано зв’язок із таблицею Tasks через ProjectID, що дозволило реалізувати логічну структуру «проєкт – набір завдань». Такий підхід забезпечив автоматичну агрегацію даних без необхідності додаткової обробки на рівні логіки застосунку.

Центральною таблицею операційного рівня є Tasks, оскільки саме вона відображає основні бізнес-процеси системи. У процесі налаштування особлива увага приділялася зв’язкам із проєктами та співробітниками: через поле AssignedTo було реалізовано прив’язку до виконавця, а через ProjectID – до відповідного проєкту. Це дало можливість використовувати Tasks як основу для динамічного відображення робочого навантаження та фільтрації даних у різних частинах інтерфейсу.

Останньою було додано таблицю Requests, яка використовується для обробки внутрішніх запитів користувачів. Вона пов’язується зі співробітниками через поле RequestedBy, що дозволяє відстежувати ініціаторів запитів, формувати історію звернень та забезпечувати прозорість процесів погодження в системі.

У підсумку всі таблиці були об’єднані в єдину логічну структуру, яка в Glide автоматично підтримує цілісність даних і забезпечує коректну взаємодію між сутностями без необхідності ручного програмування. Окремо слід зазначити, що повна структура таблиць, включно з детальним описом полів, типів даних та зв’язків, наведена в додатку Б.

Після формування базових таблиць наступним кроком стало налаштування зв’язків між ними безпосередньо у Glide Data Editor. Для цього було використано вбудований тип стовпців Relation, який дозволяє встановлювати логічні зв’язки між записами різних таблиць на основі ключових полів. У процесі реалізації система використовувала унікальні ідентифікатори (наприклад DepartmentID та EmployeeID) для автоматичного зіставлення даних, що дозволило уникнути дублювання та забезпечити нормалізовану структуру бази.

Зокрема, було реалізовано зв’язок між Departments та Employees, який відображає логіку «один відділ – багато співробітників». Аналогічно було налаштовано зв’язок між Projects і Tasks, що формує структуру проєкту як контейнера для завдань.

Додатково було створено зв’язки між Tasks та Employees для визначення виконавців, а також між Requests та Employees для фіксації авторів запитів. Усі ці зв’язки працюють динамічно, тобто будь-які зміни в даних автоматично відображаються в інтерфейсі застосунку.

Після цього було реалізовано обчислювані стовпці (Computed Columns), які дозволяють виконувати автоматичну обробку даних безпосередньо в Glide. Для цього використовувалися вбудовані механізми, такі як Rollup, If-Then-Else та Math. Наприклад, поле TaskCount було налаштоване для підрахунку кількості завдань, прив’язаних до конкретного співробітника, що дозволяє оцінювати навантаження в реальному часі. Аналогічно, ProjectProgress обчислює відсоток виконання завдань у межах проєкту та використовується для візуалізації прогресу.

Окремо було реалізовано поле IsOverdue, яке автоматично визначає прострочені завдання на основі порівняння дедлайну зі статусом виконання, а також DaysLeft, що розраховує кількість днів до завершення завдання та використовується для пріоритизації робіт.

Фінальним етапом стала автоматизація процесів за допомогою модуля Glide Workflows, який дозволяє реалізовувати подієво-орієнтовану логіку без написання коду. У системі було налаштовано сценарії, що автоматично

спрацьовують при створенні завдань, зміні їх статусу або подачі запитів. Наприклад, при створенні нового завдання система автоматично встановлює статус «To Do», фіксує дату створення та надсилає сповіщення виконавцю, що забезпечує миттєву реакцію системи на бізнес-події.

Також при зміні статусу завдання на «Done» автоматично фіксується дата завершення, що дозволяє надалі аналізувати ефективність виконання робіт. Аналогічно для Requests реалізовано workflow, де запити переходять у статус «Pending», а після погодження або відхилення система змінює статус і надсилає відповідні сповіщення користувачам.

Таким чином, реалізована в Glide структура бази даних разом із автоматизацією процесів перетворює систему на динамічне середовище, де всі зміни відбуваються автоматично та синхронізовано, що значно підвищує ефективність роботи застосунку.

3.3 Реалізація системи користувачів та ролей доступу

Реалізація підсистеми користувачів та контролю доступу в застосунку «OrgManager» стала важливим етапом, який забезпечив персоналізовану взаємодію користувачів із системою та розмежування прав відповідно до їхніх ролей. Дана функціональність була реалізована у середовищі Glide із використанням вбудованих механізмів автентифікації, рольових полів та умовної видимості компонентів.

Першим кроком стала реалізація механізму входу до системи, який базується на Email-авторизації. Після введення електронної пошти користувача система автоматично виконує пошук відповідного запису в таблиці Employees та ідентифікує користувача як активного учасника системи. Таким чином, кожен акаунт напряму пов’язується з конкретним співробітником через поле Email, що дозволяє уникнути додаткових складних механізмів реєстрації та забезпечує швидкий доступ до системи.

Після успішної авторизації користувач переходить до перевірки ролі, яка визначає подальшу логіку відображення інтерфейсу та доступних функцій.

Наступним етапом стало формування рольової моделі, яка визначає рівень доступу кожного користувача до функціоналу системи. У межах застосунку було виділено три основні ролі: Admin, Manager та Employee, кожна з яких має власний набір прав і обмежень (рис. 3.4).

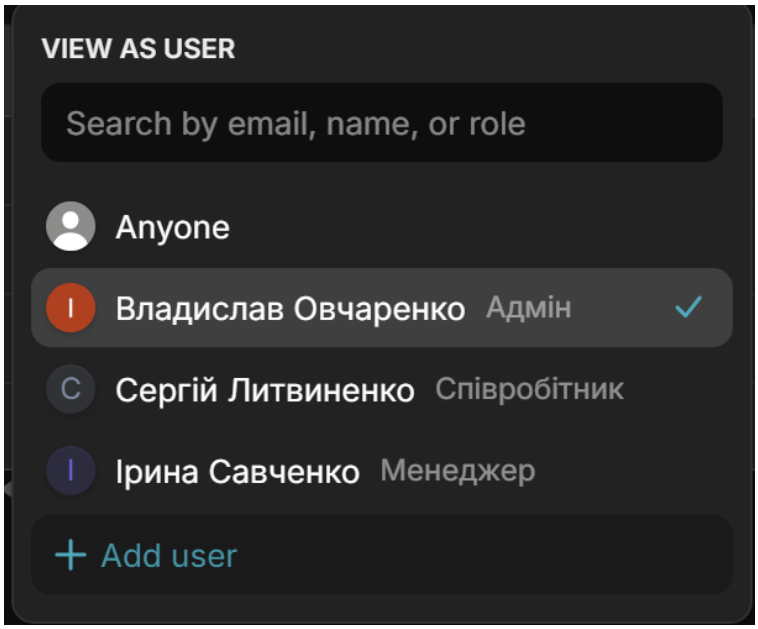


Рисунок 3.4 – Визначені користувачі в системі для тестування

Адміністратор системи має повний доступ до всіх модулів, включаючи редагування, видалення та управління структурою даних. Менеджер виконує функції керування проєктами, створення завдань і погодження запитів, тоді як співробітник має обмежений доступ і працює лише зі своїми завданнями та запитами.

Ця логіка реалізована у Glide через поєднання User Profiles, Role column та умовної видимості елементів інтерфейсу, що дозволяє динамічно змінювати доступність функцій залежно від ролі користувача.

Після визначення рольової моделі було здійснено налаштування механізму доступу до окремих елементів інтерфейсу. У Glide це реалізовано через Visibility

Conditions, які дозволяють приховувати або відображати компоненти залежно від значення ролі користувача.

Для ролі Admin активовано повний доступ до всіх екранів і функцій, включаючи можливість редагування та видалення записів. Роль Manager отримала доступ до створення та призначення завдань, а також до погодження запитів. Користувачі з роллю Employee обмежені переглядом лише власних завдань і можливістю подачі запитів.

Таким чином, система забезпечує багаторівневий контроль доступу без необхідності написання додаткового серверного коду, повністю використовуючи вбудовані можливості Glide.

Окрему увагу було приділено забезпеченню конфіденційності даних користувачів. Для цього було використано механізм Row Owners, який дозволяє обмежити доступ до записів на рівні рядків таблиці.

Зокрема, у таблиці Requests було налаштовано прив’язку через поле RequestedByEmail, що гарантує, що кожен користувач бачить лише власні запити. Аналогічно у таблиці Tasks застосовано поле AssignedToEmail, яке забезпечує відображення лише тих завдань, що призначені конкретному співробітнику.

Завдяки цьому підходу дані не передаються іншим користувачам і не завантажуються в їхній інтерфейс, що значно підвищує рівень безпеки та приватності в межах застосунку.

Приклад вигляду налаштованих даних про робітника наведено на рисунку 3.5.

У результаті реалізація системи користувачів і ролей доступу дозволила створити гнучку модель управління правами, яка забезпечує безпечну, персоналізовану та логічно структуровану взаємодію користувачів із системою «OrgManager».

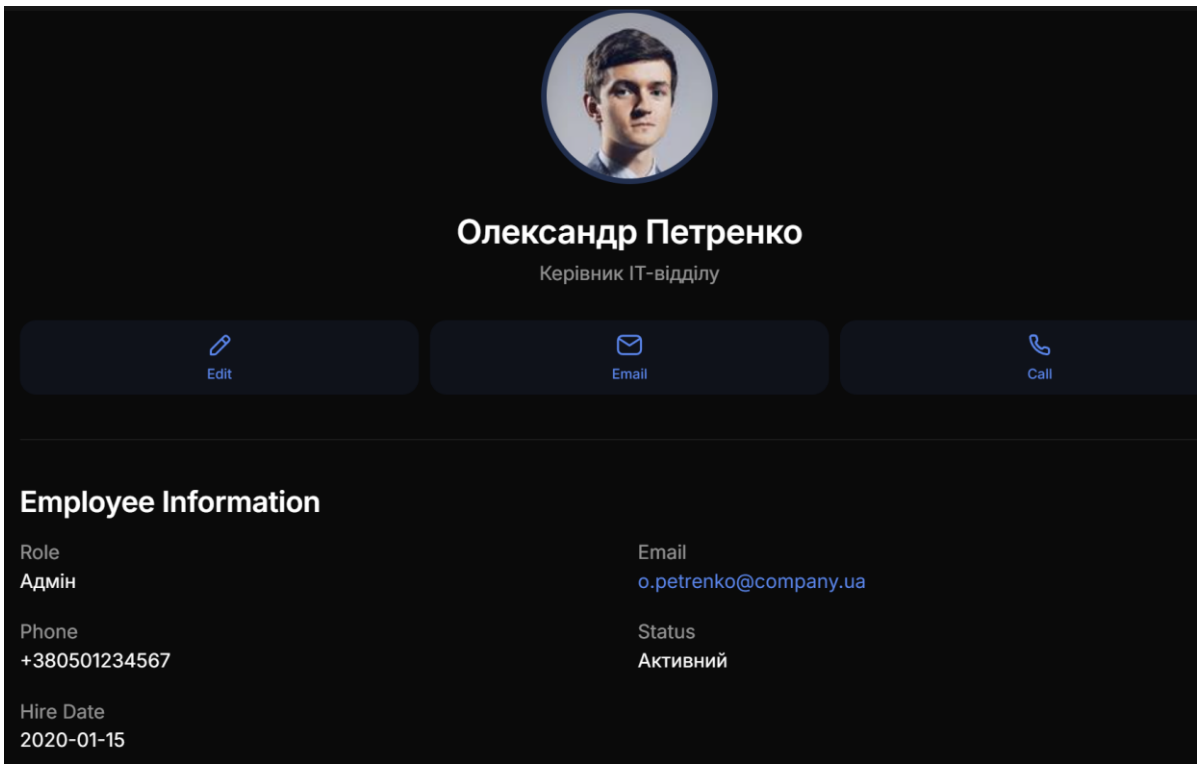


Рисунок 3.5 – Сторніка даних про адмін користувача

3.4 Реалізація функціоналу управління задачами та заявками

Після налаштування структури даних, зв’язків і рольової моделі доступу наступним етапом стала реалізація прикладного функціоналу системи, який безпосередньо забезпечує щоденну роботу користувачів із завданнями та заявками. Основна увага в цьому модулі була зосереджена на тому, щоб перетворити статичні дані на інтерактивні процеси, що відображають життєвий цикл задач і запитів у межах застосунку.

Реалізація модуля Tasks у середовищі Glide починалася з формування екрану відображення завдань, який було побудовано на основі таблиці Tasks. Після підключення джерела даних система автоматично синхронізувала записи, що дозволило одразу перейти до налаштування інтерфейсу та логіки роботи з елементами списку.

Для зручності користувача екран завдань було реалізовано у двох режимах відображення – Checklist та Kanban. Перемикання між ними дозволяє адаптувати

подання інформації залежно від потреб: у першому випадку завдання відображаються у вигляді списку (рис. 3.6), а в другому – як візуальні картки, згруповані за статусами життєвого циклу (To Do, In Progress, Review, Done).

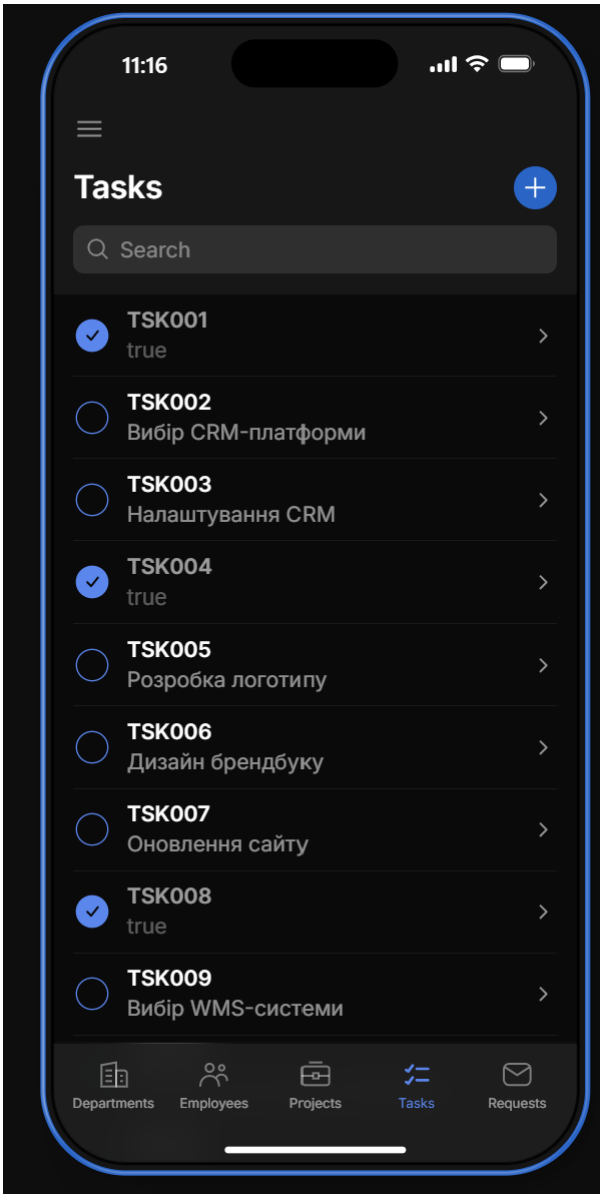


Рисунок 3.6 – Приклад Checklist в завданнях

У процесі налаштування кожен елемент інтерфейсу був пов’язаний із відповідними полями таблиці Tasks. Зокрема, кнопка «Add» була реалізована через Form Component і використовується для створення нового завдання, після чого запис автоматично додається до таблиці. Поле статусу було налаштоване як Choice Component, що дозволяє змінювати стан завдання безпосередньо з

інтерфейсу користувача. Додатково застосовано умовне форматування, яке візуально виділяє пріоритетні та критичні завдання, а також Date Component для контролю дедлайнів. Призначення виконавця реалізовано через Relation Picker, який підтягує дані з таблиці співробітників.

Таким чином, інтерфейс завдань у Glide став не просто відображенням даних, а повноцінним інструментом керування життєвим циклом задач.

Паралельно зі списковим відображенням було реалізовано канбан-дошку, яка є альтернативним способом візуалізації тих самих даних (рис. 3.7). У середовищі Glide Kanban було налаштовано на основі поля Status, що дозволило автоматично групувати завдання за етапами виконання.

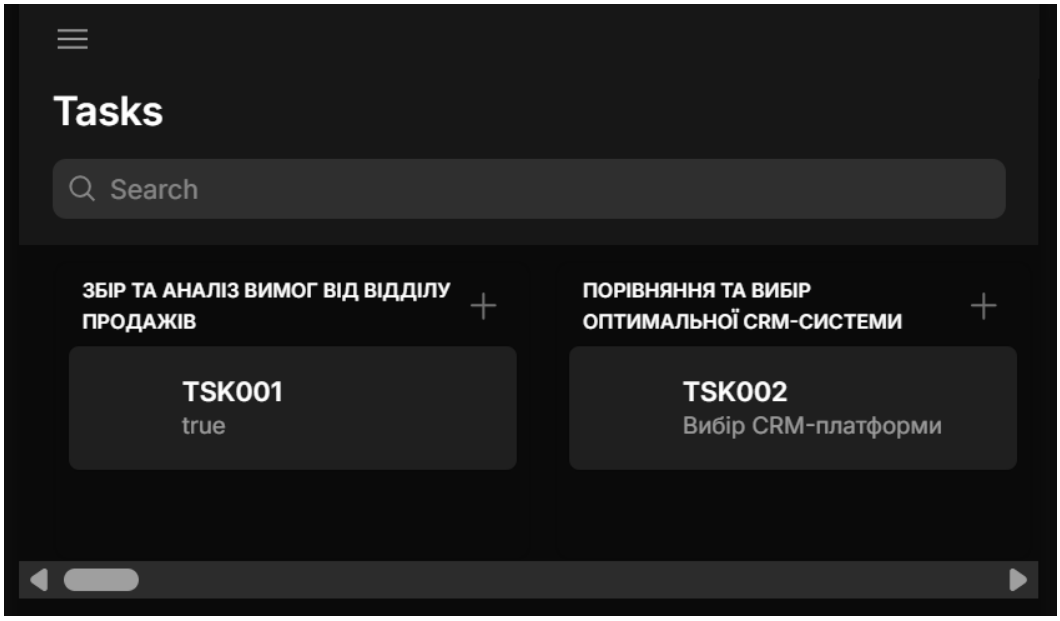


Рисунок 3.7 – Приклад Kanban в завданнях

У результаті користувач отримав можливість працювати із завданнями у вигляді карток, які можна візуально переміщувати між колонками станів. Такий підхід забезпечує більш інтуїтивне розуміння поточного навантаження та дозволяє швидко оцінювати прогрес виконання робіт без необхідності відкривати кожен запис окремо.

Перехід між статусами в канбан-дошці синхронізується з основною таблицею Tasks, що забезпечує актуальність даних у всіх режимах відображення.

Окремим функціональним блоком було реалізовано модуль Requests (рис. 3.8), який відповідає за створення та погодження внутрішніх заявок користувачів. Його побудова базується на формі Glide Data Form, через яку співробітник подає запит, заповнюючи відповідні поля.

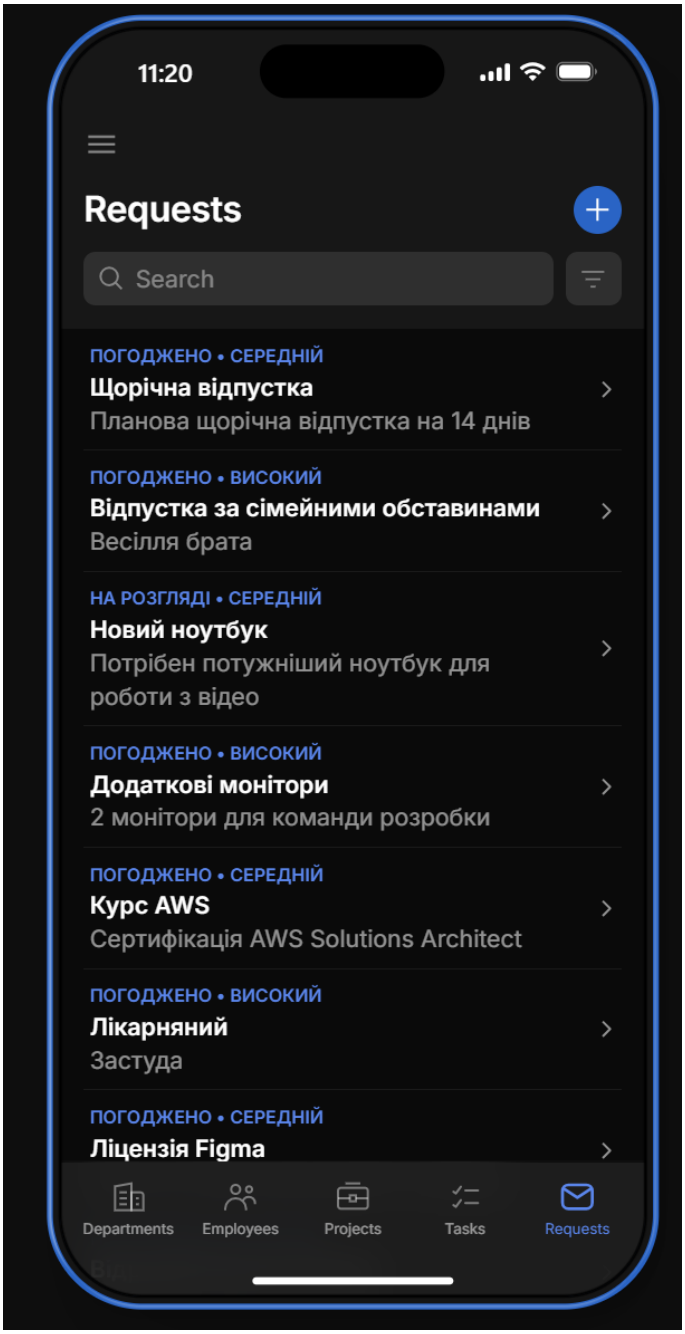


Рисунок 3.8 – Модуль Requests

Після надсилання форми в системі автоматично створюється новий запис у таблиці Requests зі статусом «Pending», що означає очікування розгляду. Далі

заявка потрапляє до списку менеджера, де вона може бути або погоджена, або відхилена.

У разі зміни статусу система автоматично оновлює запис і фіксує результат розгляду. Таким чином реалізується повний цикл обробки заявки – від її створення до фінального рішення.

Для забезпечення інтерактивності системи було налаштовано набір стандартних дій Glide Actions, які прив’язані до ключових сценаріїв роботи із завданнями та заявками. Зокрема, дія Approve Request змінює статус заявки на «Approved», тоді як Reject Request встановлює значення «Rejected». Аналогічно, для завдань реалізовано дію Complete Task, яка переводить їх у стан «Done» і може супроводжуватися відображенням сповіщення користувачу.

Також було реалізовано дію Assign Task, яка забезпечує прив’язку виконавця до конкретного завдання через оновлення відповідного relation-поля. Усі ці дії працюють у зв’язці з базою даних і миттєво синхронізуються з інтерфейсом, що забезпечує цілісність і актуальність інформації в системі.

У підсумку реалізація функціоналу управління задачами та заявками перетворила систему на інтерактивне середовище, де кожна дія користувача одразу відображається у структурі даних і впливає на подальшу логіку роботи застосунку.

3.5 Аналіз результатів розробки MVP-системи

Додатково для узагальнення результатів розробки було виконано оцінку ступеня реалізації запланованого функціоналу MVP-системи. Аналіз показав, що всі ключові компоненти, визначені на етапі проєктування, були повністю впроваджені у робочу версію застосунку «OrgManager» із використанням можливостей платформи Glide.

Усі основні функціональні блоки були реалізовані відповідно до початкових вимог і інтегровані в єдину систему без необхідності використання

зовнішнього програмного коду. Це дозволило забезпечити цілісність архітектури та швидку взаємодію між модулями.

Зведений стан реалізації функціоналу подано в таблиці 3.1.

Таблиця 3.1 - Зведений стан реалізації функціоналу

Запланована функція	**Статус реалізації**	**Платформний інструмент**
Централізована БД	Реалізовано	Glide Tables
Управління відділами	Реалізовано	Collection (List)
Облік співробітників	Реалізовано	Collection + Relations
Управління проєктами	Реалізовано	Cards + Rollup
Управління завданнями	Реалізовано	Kanban/Checklist
Обробка заявок	Реалізовано	Forms + Workflows
Рольовий доступ	Реалізовано	Roles + Row Owners
Мобільний доступ	Реалізовано	PWA

Як видно з таблиці, усі заплановані функції були успішно реалізовані в межах MVP без відхилень від проєктної структури. Особливо важливо, що кожен функціональний модуль був не лише створений, але й інтегрований у загальну систему через єдину базу даних та механізми зв’язків між таблицями.

Отримані результати свідчать про те, що розроблена MVP-система повністю виконує поставлені цілі та відповідає вимогам, сформованим на етапі проєктування. Було забезпечено централізацію даних, реалізовано автоматизацію ключових бізнес-процесів, впроваджено рольову модель доступу та створено зручний інтерфейс для роботи користувачів.

Окремо варто відзначити, що використання платформи Glide дозволило реалізувати повноцінну інформаційну систему у форматі no-code рішення, де логіка взаємодії, структура даних та бізнес-процеси були побудовані через візуальні інструменти без класичного програмування.

У підсумку MVP-система «OrgManager» може розглядатися як стабільна основа для подальшого розширення функціоналу, масштабування та переходу до більш складної корпоративної інформаційної системи.

3.6 Перспективи розвитку та вдосконалення застосунку

Після завершення реалізації MVP-системи «OrgManager» було визначено основні напрями її подальшого розвитку та вдосконалення. Оскільки створений застосунок охоплює базовий функціонал управління організаційними процесами, наступним логічним етапом є його розширення до повноцінної корпоративної інформаційної системи з більш глибокою автоматизацією, аналітикою та інтеграціями.

У першу чергу перспективним напрямом є розширення функціоналу аналітики та звітності. Наразі система забезпечує базове відображення даних через Rollup- та обчислювані поля, однак у подальшому доцільно впровадити дашборди управлінського рівня з візуалізацією ключових показників ефективності (KPI), навантаження співробітників, виконання проєктів та динаміки заявок. Це дозволить перетворити систему з інструмента обліку на повноцінну платформу підтримки управлінських рішень.

Наступним важливим напрямом є розширення автоматизації бізнес-процесів. Поточна реалізація Workflows покриває базові сценарії створення та погодження завдань і заявок, однак у перспективі можливо впровадження більш складних багаторівневих процесів погодження, умовних сценаріїв виконання дій та автоматичних нагадувань. Додатково доцільним є впровадження системи тригерів, що реагують на зміни даних у реальному часі та запускають ланцюжки автоматичних операцій.

Окремим напрямом розвитку є інтеграція із зовнішніми сервісами. Зокрема, система може бути підключена до електронної пошти, месенджерів або календарів для автоматичного сповіщення користувачів, синхронізації подій та планування завдань. Також перспективною є інтеграція з API сторонніх систем управління проєктами або HR-платформами для обміну даними.

Важливим етапом вдосконалення є масштабування системи та оптимізація структури даних. У міру збільшення кількості користувачів і записів доцільно переглянути архітектуру бази даних, оптимізувати зв’язки між таблицями та

впровадити більш гнучку модель збереження історичних даних. Це дозволить підвищити продуктивність системи та забезпечити стабільну роботу при зростанні навантаження.

Також перспективним є покращення користувацького інтерфейсу. Хоча Glide забезпечує швидке створення UI, у майбутньому можливе впровадження більш кастомізованого дизайну, адаптивних компонентів та персоналізованих панелей користувача. Це підвищить зручність роботи та дозволить краще адаптувати систему під різні ролі.

Окремо варто відзначити потенціал впровадження елементів штучного інтелекту. У перспективі система може аналізувати навантаження співробітників, прогнозувати строки виконання завдань, автоматично пріоритезувати задачі та надавати рекомендації менеджерам щодо оптимізації робочих процесів.

У підсумку можна зазначити, що реалізована MVP-система «OrgManager» має значний потенціал для подальшого розвитку. Її архітектура, побудована на принципах модульності та зв’язності даних, дозволяє поступово розширювати функціонал без необхідності кардинальної перебудови системи, що створює основу для її еволюції у повноцінну корпоративну платформу управління.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було здійснено комплексне дослідження, проєктування та реалізацію MVP-системи для автоматизації організаційно-управлінських процесів із використанням No-Code платформи Glide. Результати роботи підтверджують доцільність застосування No-Code підходу для швидкої розробки функціональних інформаційних систем, що не потребують традиційного програмування.

На першому етапі було розглянуто теоретичні основи No-Code розробки та концепцію MVP, що дозволило визначити ключові принципи створення мінімально життєздатного продукту, орієнтованого на швидке тестування гіпотез і перевірку бізнес-логіки. Також було проаналізовано сучасні No-Code платформи та обґрунтовано вибір Glide як оптимального інструменту для реалізації системи завдяки його гнучкості, підтримці баз даних, вбудованим компонентам інтерфейсу та можливостям автоматизації.

На етапі проєктування було проаналізовано предметну область управлінських процесів, визначено функціональні та нефункціональні вимоги до системи, а також спроєктовано структуру MVP, що включає модулі управління відділами, співробітниками, проєктами, завданнями та заявками. Особливу увагу було приділено проєктуванню бази даних, визначенню сутностей та зв’язків між ними, а також розробці логіки взаємодії користувачів із системою.

Практична реалізація була виконана у середовищі Glide, де було створено багатотабличну базу даних із використанням Relations, Computed Columns та автоматичних Workflow-процесів. Було налаштовано інтерфейс застосунку з використанням компонентів Collection, Kanban та Forms, що забезпечило зручне відображення та управління даними. Додатково реалізовано рольову модель доступу та механізм Row Owners, який забезпечує захист даних та розмежування прав користувачів.

Аналіз результатів розробки показав, що всі заплановані функції MVP були успішно реалізовані, зокрема централізоване зберігання даних, управління завданнями та заявками, рольовий доступ і автоматизація ключових бізнес-процесів. Це підтверджує ефективність використаного підходу та відповідність системи визначеним вимогам.

Таким чином, розроблена MVP-система демонструє можливість швидкого створення функціональних управлінських застосунків на основі No-Code технологій. Отримані результати можуть бути використані як основа для подальшого розширення системи та впровадження її у реальні організаційні середовища.

ПЕРЕЛІК ПОСИЛАНЬ

1. Ajimati M. O., Carroll N., Maher M. Adoption of low-code and no-code development: A systematic literature review and future research agenda. *Journal of Systems and Software*. 2024. Vol. 222. Art. 112300. DOI : https://doi.org/10.1016/j.jss.2024.112300.

2. No-code платформа: знайомство з технологіями no-code. Creatio. URL : https://www.creatio.com/ua/no-code/what-is-no-code (дата звернення : 26.03.2026).

3. Khan T., Downie A. What is no-code? IBM. URL : https://www.ibm.com/think/topics/no-code (дата звернення : 26.03.2026).

4. Різниця між low-code та no-code: порівняння та можливості кожної технології. Scriptum. URL : https://scriptum.ua/riznyczya-mizh-low-code-ta-no-code-porivnyannya-ta-mozhlyvosti-kozhnoyi-tehnologiyi/ (дата звернення : 26.03.2026).

5. Zeichick A. What is No-Code Development? Building apps without coding. *Oracle Middle East Regional*. 2025. URL : https://www.oracle.com/ middleeast/application-development/no-code/ (дата звернення : 26.03.2026).

6. Koç O., Yücedağ İ., Şentürk Ü. The Impact of Artificial Intelligence Enhanced No-Code Software Development Platforms on Software Processes : A Literature Review. *Düzce Üniversitesi Bilim ve Teknoloji Dergisi*. 2025. Vol. 13, No. 1. P. 383–401. DOI : https://doi.org/10.29130/dubited.1554356.

7. McDonald K. What is a Minimum Viable Product (MVP)? Agile Alliance. 2023. URL : https://agilealliance.org/glossary/mvp/ (дата звернення : 26.03.2026).

8. Lortie J., Cox K., DeRosset S., Thompson R., Kelly S. Unpacking the minimum viable product (MVP) : a framework for use, goals and essential elements. *Journal of Small Business and Enterprise Development*. 2024. Vol. 32, No. 1. P. 212–235. DOI : https://doi.org/10.1108/jsbed-02-2024-0075.

9. Launched Y. A. MVP Development Stages explained: How to launch a startup? *URLAUNCHED*. 2024. URL : https://www.urlaunched.com/blog/mvp-development-stages-explained-how-to-launch-a-startup (дата звернення : 26.03.2026).

10. Ries E. What is an MVP? Lean Startup Co. 2026. URL : https://leanstartup.co/resources/articles/what-is-an-mvp/ (дата звернення : 26.03.2026).

11. Rehkopf M. What is a Minimum Viable Product (MVP)? How to Get Started. Atlassian. 2026. URL : https://www.atlassian.com/agile/product-management/minimum-viable-product (дата звернення : 26.03.2026).

12. miisoft. No-code платформи – найповніший каталог, порівняння, відгуки. URL : https://miisoft.com.ua/category/no-code-software/ (дата звернення : 26.03.2026).

13. EINPresswire. No-Code Development Platforms Market Sees Exponential Growth, Projected to Reach $73.84 Billion by 2028. URL : https://www.einpresswire.com/article/721480405 (дата звернення : 26.03.2026).

14. MarketsandMarkets. No-code AI Platforms Market Surges to $24.7 billion by 2029. 2025. URL : https://lite.aol.com/entertainment/story/0022/20251212/9596860.htm (дата звернення : 26.03.2026).

15. Quixy. Game-Changing Top 60 No-Code Low-Code Citizen Development Statistics. URL : https://quixy.com/blog/no-code-low-code-citizen-development-statistics-facts/ (дата звернення : 26.03.2026).

16. gtrix. Low-code і No-code платформи – альтернатива класичній розробці? URL : https://gtrix.com.ua/blog/low-code-i-no-code-alternativa-klasichniy-rozrobtsi/ (дата звернення : 26.03.2026).

17. We Rise by Lifting Others. Story of Glide : No-code platform for building mobile and web apps. 2025. URL : https://www.werisebyliftingothers.in/2025/09/story-of-glide-no-code-platform-for.html (дата звернення : 26.03.2026).

18. AppMySite. Glide No-Code App Builder Review (2025). URL : https://blog.appmysite.com/glide-no-code-app-builder-review/ (дата звернення : 26.03.2026).

19. Linktly. Glide Review 2026: Features, Pricing, Pros, Cons. URL : https://www.linktly.com/development-software/glide-promo-code/ (дата звернення : 26.03.2026).

20. iCoderz. Bubble Pros and Cons – Should You Use This No-Code Platform? URL : https://www.icoderzsolutions.com/blog/bubble-pros-and-cons/ (дата звернення : 26.03.2026).

21. Blaze. Bubble.io Review: Is It Worth The Cost in 2025? URL : https://www.blaze.tech/post/bubble-reviews (дата звернення : 26.03.2026).

22. Involve Digital. No-Code Web Apps with Webflow 2026. URL : https://www.involvedigital.com/insights/no-code-web-apps-webflow-guide (дата звернення : 26.03.2026).

23. RapidDevelopers. Webflow Review 2024: Pros, Cons, Security. URL : https://www.rapidevelopers.com/review/webflow (дата звернення : 26.03.2026).

24. Blaze. AppSheet Review (2025): Features, Pricing, & Who It’s Best For. URL : https://www.blaze.tech/post/appsheet-review (дата звернення : 26.03.2026).

ДОДАТОК А

Інтерфейс розробленої програми

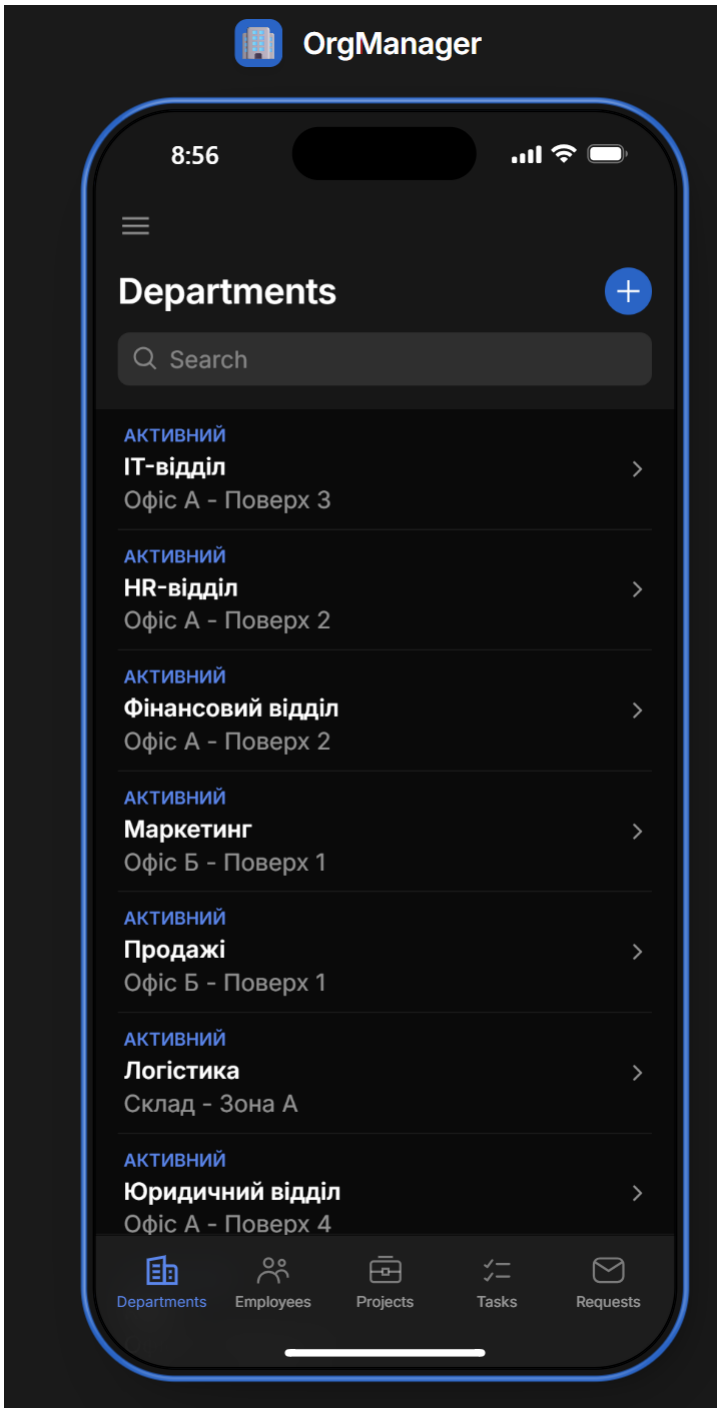


Рисунок А.1 – Сторінка відділів

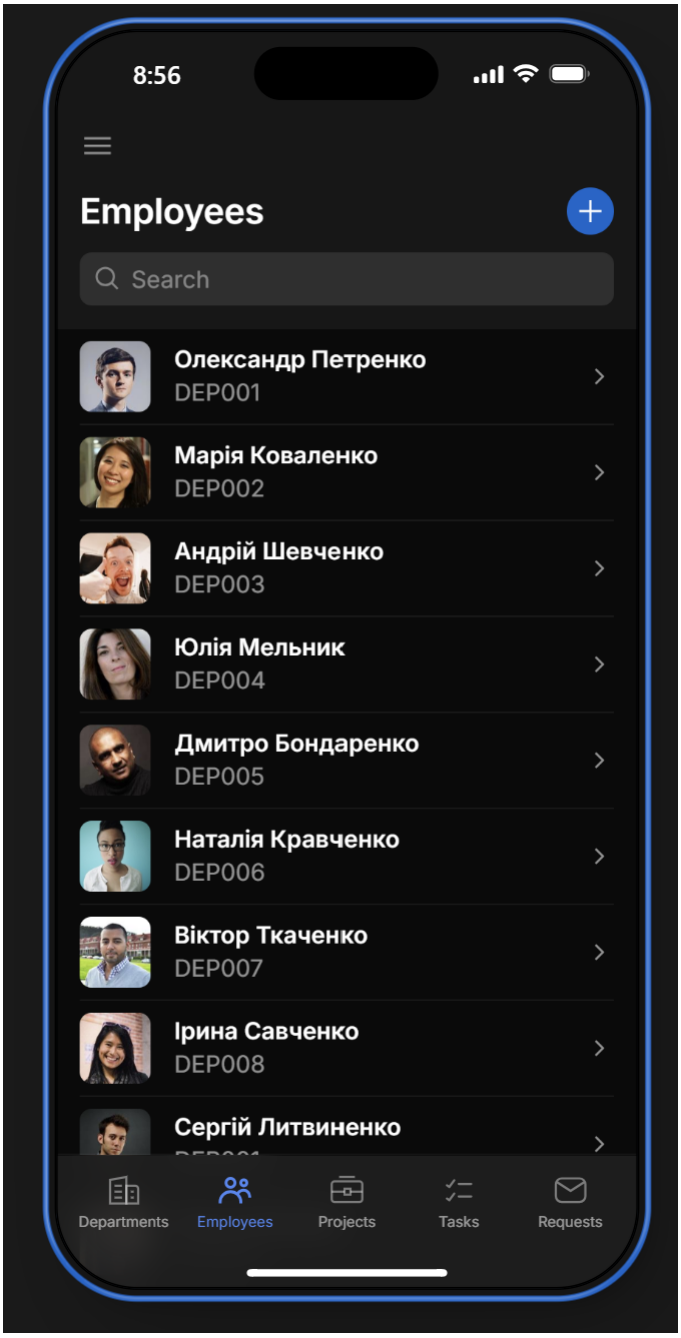


Рисунок А.2 – Сторінка працівників

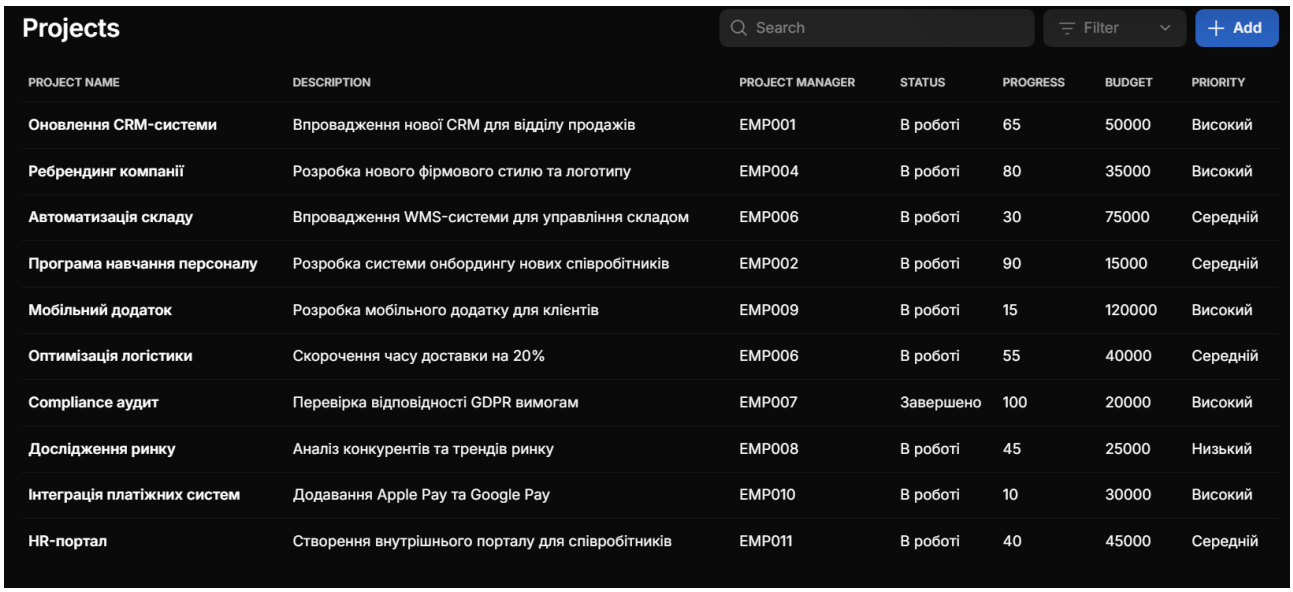


Рисунок А.3 – Сторінка проєктів

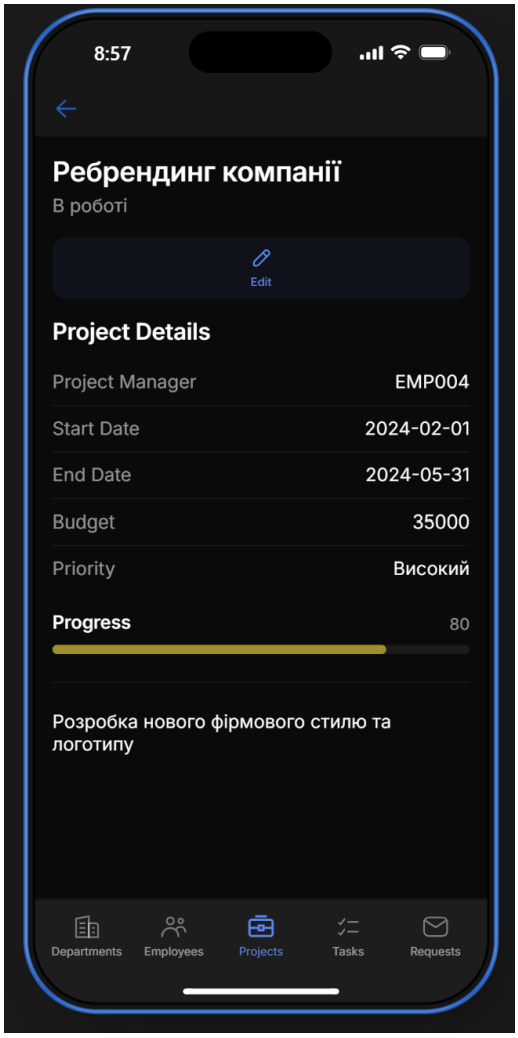


Рисунок А.4 – Сторінка опису проєкту

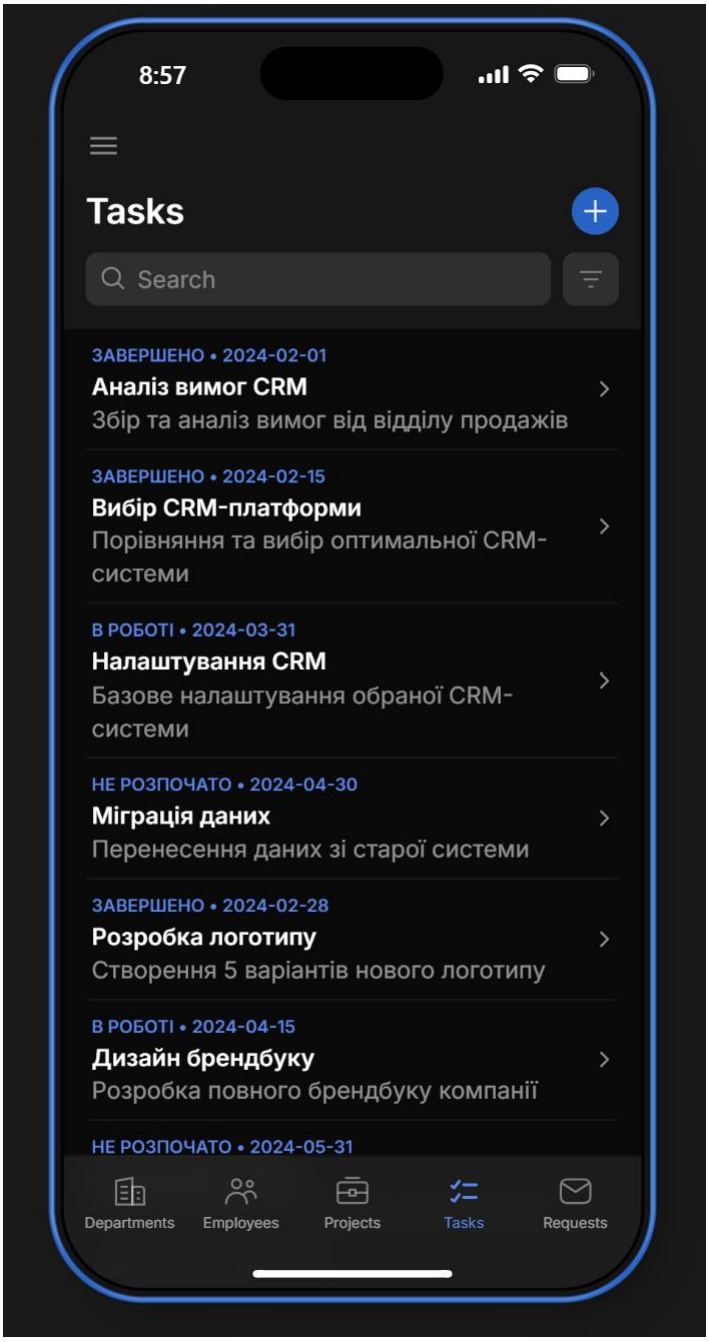


Рисунок А.5 – Сторінка завдань

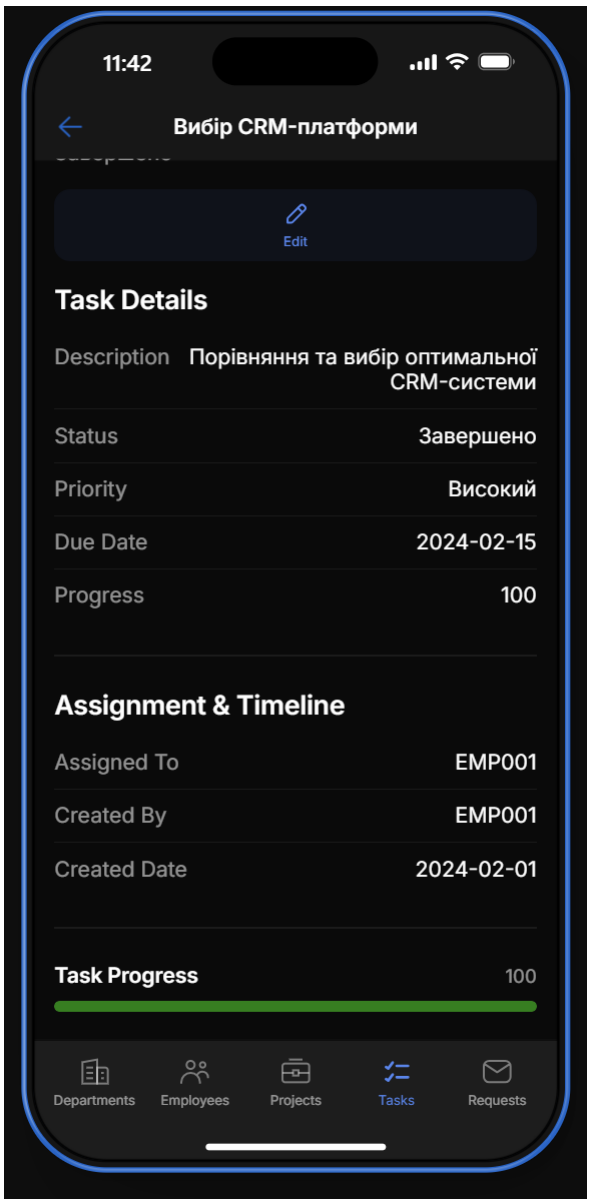


Рисунок А.6 – Сторінка опису завдання

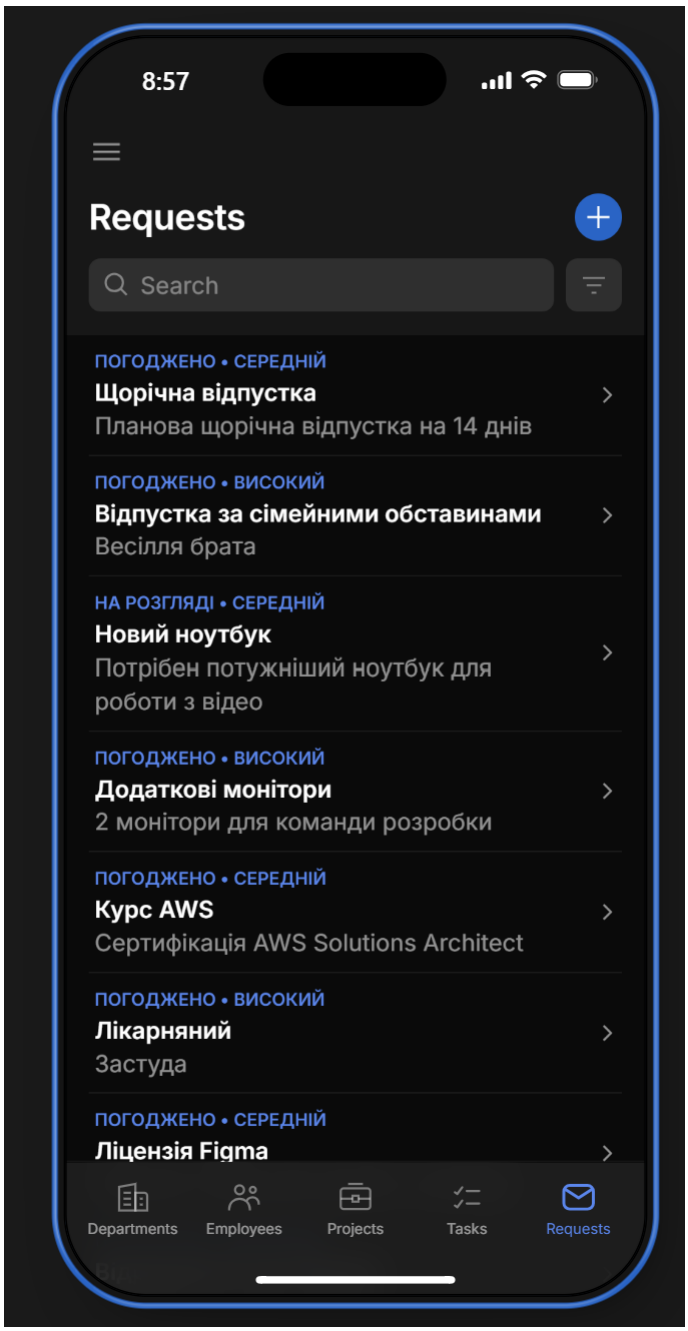


Рисунок А.7 – Сторінка запросів

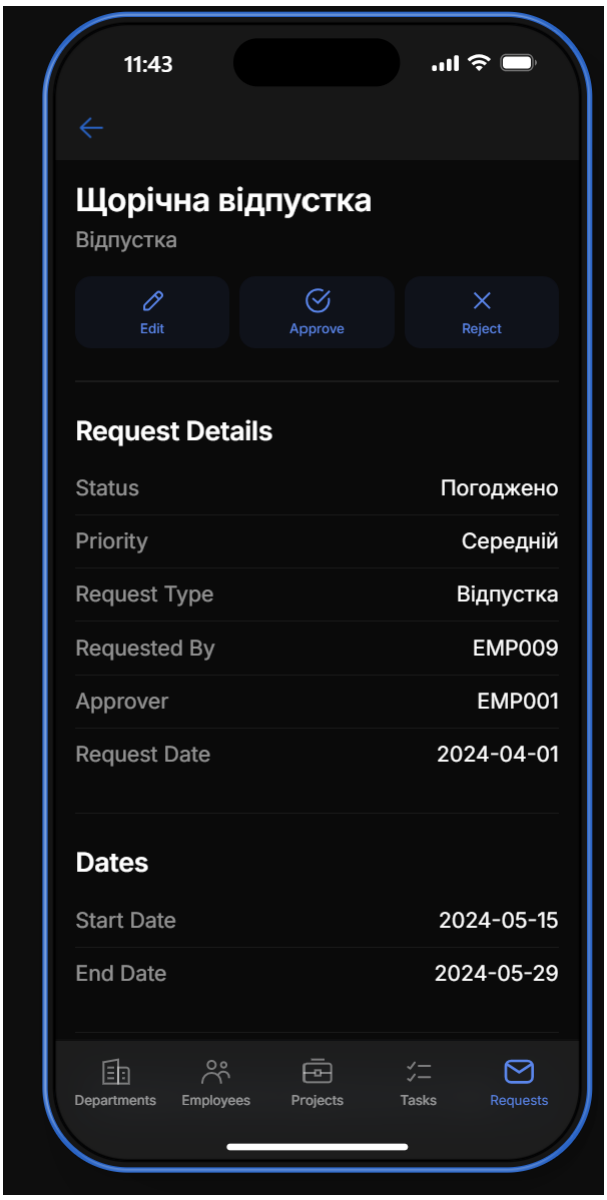


Рисунок А.8 – Сторінка опису запросу

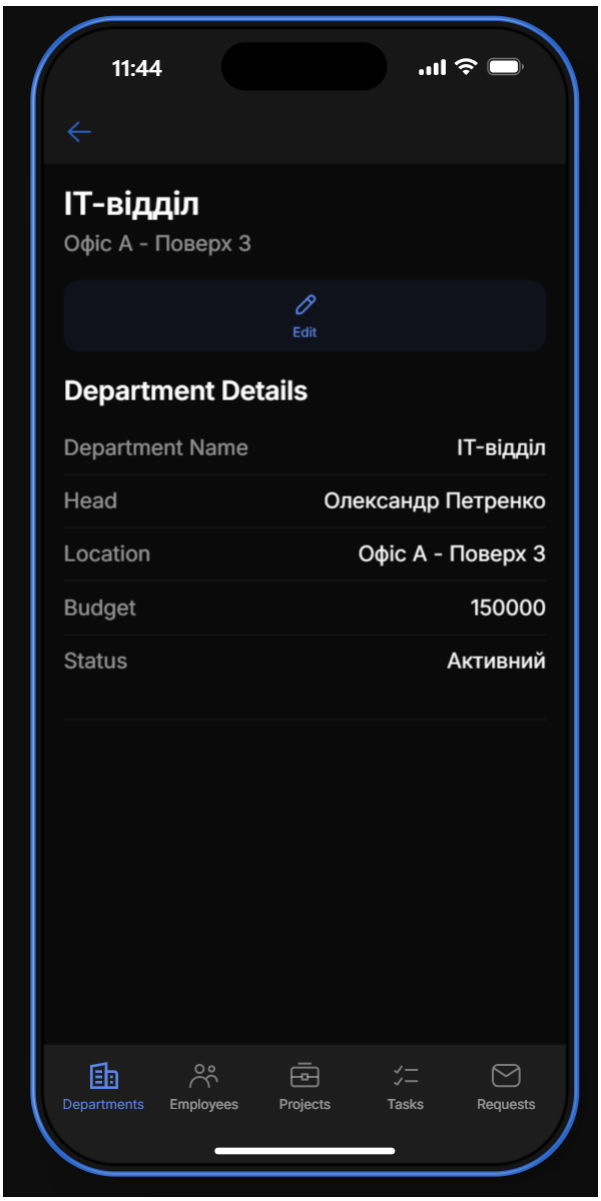


Рисунок А.9 – Сторінка даних про відділ

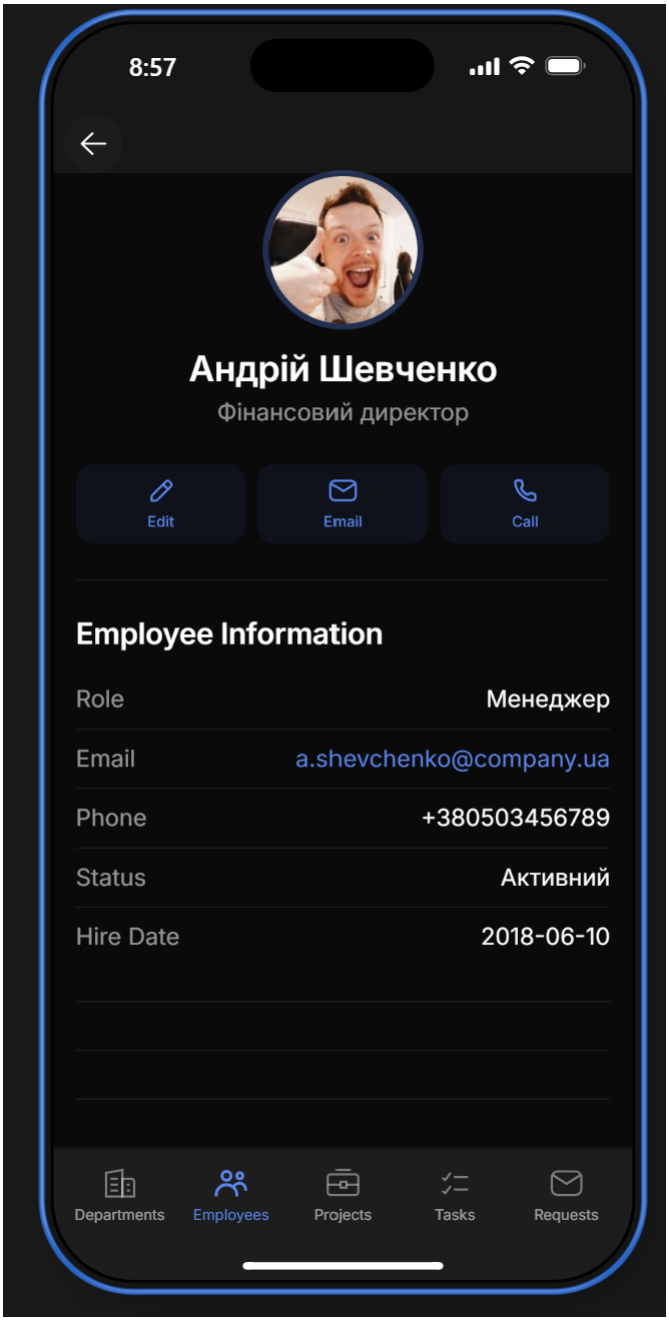


Рисунок А.10 – Сторінка даних про робітника

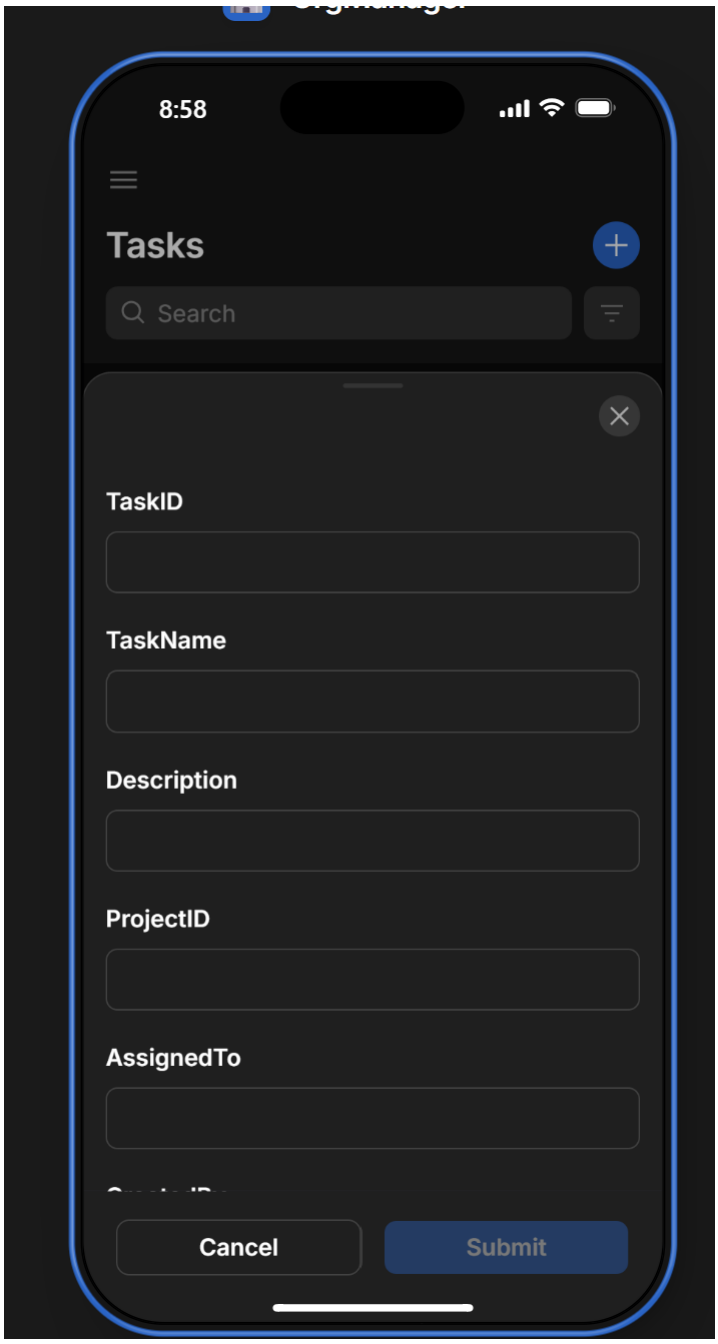


Рисунок А.11 – Сторінка створення завдання

ДОДАТОК Б

Розроблені таблиці для No-code платформи

	A	B	C	D	E	F
1	DepartmentID	DepartmentNam	Head	Location	Budget	Status
2	DEP001	IT-відділ	Олександр Петренко	Офіс А - Поверх 3	150000	Активний
3	DEP002	HR-відділ	Марія Коваленко	Офіс А - Поверх 2	80000	Активний
4	DEP003	Фінансовий відд	Андрій Шевченко	Офіс А - Поверх 2	120000	Активний
5	DEP004	Маркетинг	Юлія Мельник	Офіс Б - Поверх 1	100000	Активний
6	DEP005	Продажі	Дмитро Бондаренко	Офіс Б - Поверх 1	200000	Активний
7	DEP006	Логістика	Наталія Кравченко	Склад - Зона А	90000	Активний
8	DEP007	Юридичний відд	Віктор Ткаченко	Офіс А - Поверх 4	70000	Активний
9	DEP008	R&D	Ірина Савченко	Офіс В - Поверх 2	180000	Активний

Рисунок Б.1 – Departments

	A	B	C	D	E	F	G	H	I	J	K	L
1	EmployeeID	FullName	Email	Phone	DepartmentID	Position	Role	HireDate	Status	Photo		
2	EMP001	Олександр Петренко	o.petrenko@com	+380501234567	DEP001	Керівник IT-відд	Адмін	2020-01-15	Активний	https://randomuser.me/api/portraits/men/1.jpg		
3	EMP002	Марія Коваленко	m.kovalenko@c	+380502345678	DEP002	HR-директор	Менеджер	2019-03-20	Активний	https://randomuser.me/api/portraits/women/2.jpg		
4	EMP003	Андрій Шевченко	a.shevchenko@	+380503456789	DEP003	Фінансовий дир	Менеджер	2018-06-10	Активний	https://randomuser.me/api/portraits/men/3.jpg		
5	EMP004	Юлія Мельник	y.melnyk@comp	+380504567890	DEP004	Маркетинг-дире	Менеджер	2021-02-01	Активний	https://randomuser.me/api/portraits/women/4.jpg		
6	EMP005	Дмитро Бондаренко	d.bondarenko@(	+380505678901	DEP005	Керівник продаж	Менеджер	2019-09-15	Активний	https://randomuser.me/api/portraits/men/5.jpg		
7	EMP006	Наталія Кравченко	n.kravchenko@c	+380506789012	DEP006	Керівник логісти	Менеджер	2020-04-22	Активний	https://randomuser.me/api/portraits/women/6.jpg		
8	EMP007	Віктор Ткаченко	v.tkachenko@co	+380507890123	DEP007	Головний юрист	Менеджер	2017-11-30	Активний	https://randomuser.me/api/portraits/men/7.jpg		
9	EMP008	Ірина Савченко	i.savchenko@co	+380508901234	DEP008	Керівник R&D	Менеджер	2021-05-18	Активний	https://randomuser.me/api/portraits/women/8.jpg		
10	EMP009	Сергій Литвиненко	s.lytvynenko@c	+380509012345	DEP001	Старший розроб	Співробітник	2021-08-01	Активний	https://randomuser.me/api/portraits/men/9.jpg		
11	EMP010	Ольга Попова	o.popova@comp	+380500123456	DEP001	DevOps-інженер	Співробітник	2022-01-10	Активний	https://randomuser.me/api/portraits/women/10.jpg		
12	EMP011	Максим Козак	m.kozak@compa	+380501122334	DEP001	Frontend-розроб	Співробітник	2022-03-15	Активний	https://randomuser.me/api/portraits/men/11.jpg		
13	EMP012	Анна Гриценко	a.grytsenko@co	+380502233445	DEP002	HR-менеджер	Співробітник	2021-06-20	Активний	https://randomuser.me/api/portraits/women/12.jpg		
14	EMP013	Павло Іваненко	p.ivanenko@com	+380503344556	DEP004	SMM-спеціаліст	Співробітник	2022-02-28	Активний	https://randomuser.me/api/portraits/men/13.jpg		
15	EMP014	Катерина Сидоренко	k.sydorenko@c	+380504455667	DEP005	Менеджер з про	Співробітник	2021-11-12	Активний	https://randomuser.me/api/portraits/women/14.jpg		
16	EMP015	Роман Федоренко	r.fedorenko@cor	+380505566778	DEP003	Бухгалтер	Співробітник	2020-07-05	Активний	https://randomuser.me/api/portraits/men/15.jpg		
17	EMP016	Тетяна Морозова	t.morozova@cor	+380506677889	DEP006	Логіст	Співробітник	2022-04-18	Активний	https://randomuser.me/api/portraits/women/16.jpg		
18	EMP017	Олег Черненко	o.chernenko@c	+380507788990	DEP008	Дослідник	Співробітник	2021-09-25	Активний	https://randomuser.me/api/portraits/men/17.jpg		
19	EMP018	Світлана Яковенко	s.yakovenko@c	+380508899001	DEP007	Юрист	Співробітник	2022-05-30	Активний	https://randomuser.me/api/portraits/women/18.jpg		

Рисунок Б.2 – Employees

	A	B	C	D	E	F	G	H	I	J	K
1	ProjectID	ProjectName	Description	DepartmentID	ProjectManager	StartDate	EndDate	Budget	Progress	Status	Priority
2	PRJ001	Оновлення CRM-систе	Впровадження нової CRM для в	DEP001	EMP001	2024-01-15	2024-06-30	50000	65	В роботі	Високий
3	PRJ002	Ребрендинг компанії	Розробка нового фірмового сти	DEP004	EMP004	2024-02-01	2024-05-31	35000	80	В роботі	Високий
4	PRJ003	Автоматизація складу	Впровадження WMS-системи д	DEP006	EMP006	2024-03-01	2024-09-30	75000	30	В роботі	Середній
5	PRJ004	Програма навчання пе	Розробка системи онбордингу н	DEP002	EMP002	2024-01-10	2024-04-30	15000	90	В роботі	Середній
6	PRJ005	Мобільний додаток	Розробка мобільного додатку д	DEP001	EMP009	2024-04-01	2024-12-31	120000	15	В роботі	Високий
7	PRJ006	Оптимізація логістики	Скорочення часу доставки на 2	DEP006	EMP006	2024-02-15	2024-07-31	40000	55	В роботі	Середній
8	PRJ007	Compliance аудит	Перевірка відповідності GDPR в	DEP007	EMP007	2024-03-15	2024-05-15	20000	100	Завершено	Високий
9	PRJ008	Дослідження ринку	Аналіз конкурентів та трендів р	DEP008	EMP008	2024-04-10	2024-06-10	25000	45	В роботі	Низький
10	PRJ009	Інтеграція платіжних си	Додавання Apple Pay та Google	DEP001	EMP010	2024-05-01	2024-07-31	30000	10	В роботі	Високий
11	PRJ010	HR-портал	Створення внутрішнього портал	DEP001	EMP011	2024-03-20	2024-08-31	45000	40	В роботі	Середній
12											

Рисунок Б.3 – Projects

	A	B	C	D	E	F	G	H	I	J	K
1	TaskID	TaskName	Description	ProjectID	AssignedTo	CreatedBy	CreatedDate	DueDate	Priority	Status	Progress
2	TSK001	Аналіз вимог CRM	Збір та аналіз вимог від відділ	PRJ001	EMP009	EMP001	2024-01-15	2024-02-01	Високий	Завершено	100
3	TSK002	Вибір CRM-платформ	Порівняння та вибір оптималь	PRJ001	EMP001	EMP001	2024-02-01	2024-02-15	Високий	Завершено	100
4	TSK003	Налаштування CRM	Базове налаштування обрано	PRJ001	EMP010	EMP001	2024-02-15	2024-03-31	Високий	В роботі	70
5	TSK004	Міграція даних	Перенесення даних зі старої с	PRJ001	EMP009	EMP001	2024-04-01	2024-04-30	Високий	Не розпочато	0
6	TSK005	Розробка логотипу	Створення 5 варіантів нового	PRJ002	EMP013	EMP004	2024-02-01	2024-02-28	Високий	Завершено	100
7	TSK006	Дизайн брендбуку	Розробка повного брендбуку к	PRJ002	EMP013	EMP004	2024-03-01	2024-04-15	Середній	В роботі	60
8	TSK007	Оновлення сайту	Впровадження нового дизайн	PRJ002	EMP011	EMP004	2024-04-15	2024-05-31	Середній	Не розпочато	0
9	TSK008	Аудит складських про	Аналіз поточних процесів на с	PRJ003	EMP016	EMP006	2024-03-01	2024-03-31	Високий	Завершено	100
10	TSK009	Вибір WMS-системи	Аналіз та вибір системи управ	PRJ003	EMP006	EMP006	2024-04-01	2024-04-30	Високий	В роботі	50
11	TSK010	Розробка програми на	Створення модулів онбординг	PRJ004	EMP012	EMP002	2024-01-10	2024-02-28	Середній	Завершено	100
12	TSK011	Створення відеомате	Запис навчальних відео	PRJ004	EMP012	EMP002	2024-03-01	2024-03-31	Середній	Завершено	100
13	TSK012	Тестування програми	Пілотне тестування з новими	PRJ004	EMP012	EMP002	2024-04-01	2024-04-30	Середній	В роботі	80
14	TSK013	Прототип мобільного	Розробка UI/UX прототипу	PRJ005	EMP011	EMP009	2024-04-01	2024-05-15	Високий	В роботі	40
15	TSK014	Backend API	Розробка серверної частини д	PRJ005	EMP009	EMP009	2024-05-01	2024-08-31	Високий	Не розпочато	0
16	TSK015	Аналіз маршрутів дос	Оптимізація логістичних марш	PRJ006	EMP016	EMP006	2024-02-15	2024-03-31	Середній	Завершено	100
17	TSK016	Впровадження GPS-т	Встановлення систем відстеж	PRJ006	EMP016	EMP006	2024-04-01	2024-05-31	Середній	В роботі	45
18	TSK017	GDPR аудит документ	Перевірка політик конфіденціл	PRJ007	EMP018	EMP007	2024-03-15	2024-04-15	Високий	Завершено	100
19	TSK018	Оновлення договорів	Внесення змін відповідно до С	PRJ007	EMP007	EMP007	2024-04-15	2024-05-15	Високий	Завершено	100
20	TSK019	Опитування клієнтів	Проведення онлайн-опитуван	PRJ008	EMP017	EMP008	2024-04-10	2024-05-10	Низький	В роботі	60
21	TSK020	Аналіз конкурентів	Дослідження 10 основних кон	PRJ008	EMP017	EMP008	2024-05-01	2024-05-31	Низький	В роботі	30
22	TSK021	Інтеграція Apple Pay	Технічна інтеграція Apple Pay	PRJ009	EMP010	EMP010	2024-05-01	2024-06-15	Високий	В роботі	15
23	TSK022	Тестування платежів	QA тестування платіжних сист	PRJ009	EMP009	EMP010	2024-06-15	2024-07-31	Високий	Не розпочато	0
24	TSK023	Дизайн HR-порталу	Розробка макетів порталу	PRJ010	EMP011	EMP011	2024-03-20	2024-04-20	Середній	Завершено	100
25	TSK024	Розробка модуля відп	Система подачі заявок на відп	PRJ010	EMP011	EMP011	2024-04-20	2024-06-30	Середній	В роботі	35
26	TSK025	Підготовка звіту Q1	Квартальний фінансовий звіт	PRJ003	EMP015	EMP003	2024-04-01	2024-04-15	Високий	Завершено	100
27											

Рисунок Б.4 – Tasks

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	RequestID	RequestType	Title	Description	RequestedBy	ApproverID	RequestDate	StartDate	EndDate	Status	Priority	Comments	
2	REQ001	Відпустка	Щорічна відпустка	Планова щорічна відп	EMP009	EMP001	2024-04-01	2024-05-15	2024-05-29	Погоджено	Середній	Погоджено. Приємного відпочинку!	
3	REQ002	Відпустка	Відпустка за сімейни	Весілля брата	EMP012	EMP002	2024-04-10	2024-04-25	2024-04-28	Погоджено	Високий		
4	REQ003	Ресурси	Новий ноутбук	Потрібен потужніший н	EMP013	EMP004	2024-04-05			На розгляді	Середній	Очікуємо затвердження бюджету	
5	REQ004	Ресурси	Додаткові монітори	2 монітори для команд	EMP001	EMP003	2024-03-20			Погоджено	Високий	Замовлення оформлено	
6	REQ005	Навчання	Курс AWS	Сертифікація AWS Sol	EMP010	EMP001	2024-04-12	2024-05-01	2024-07-31	Погоджено	Середній	Бюджет виділено	
7	REQ006	Відпустка	Лікарняний	Застуда	EMP015	EMP003	2024-04-15	2024-04-15	2024-04-19	Погоджено	Високий	Одужуйте!	
8	REQ007	Ресурси	Ліцензія Figma	Ліцензія Figma для дис	EMP011	EMP001	2024-04-08			Погоджено	Середній	Оформлено річну підписку	
9	REQ008	Відрядження	Відрядження до Льво	Зустріч з клієнтом	EMP014	EMP005	2024-04-18	2024-04-22	2024-04-24	На розгляді	Високий		
10	REQ009	Навчання	Конференція IT Arena	Участь у конференції І	EMP009	EMP001	2024-04-20	2024-09-26	2024-09-28	На розгляді	Низький		
11	REQ010	Ресурси	Офісні меблі	Ергономічне крісло	EMP017	EMP008	2024-04-14			Відхилено	Низький	Бюджет вичерпано. Розглянемо в Q3	
12	REQ011	Відпустка	Відпустка без збереж	Особисті справи	EMP016	EMP006	2024-04-22	2024-05-10	2024-05-12	На розгляді	Середній		
13	REQ012	Ресурси	Підписка на сервіси	Підписка на аналітичні	EMP017	EMP008	2024-04-25			На розгляді	Середній	Semrush + SimilarWeb	
14	REQ013	Навчання	Курс Project Managen	PMP сертифікація	EMP012	EMP002	2024-04-28	2024-06-01	2024-09-30	Погоджено	Середній		
15	REQ014	Відрядження	Виставка LogiMAT	Участь у логістичній ви	EMP006	EMP003	2024-04-30	2024-05-20	2024-05-23	Погоджено	Високий	Затверджено бюджет на 3 особи	
16	REQ015	Відпустка	Декретна відпустка	Відпустка по догляду з	EMP018	EMP007	2024-05-02	2024-06-01	2024-12-01	На розгляді	Високий	Документи передані в HR	
17													

Рисунок Б.5 – Requests