

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Мобільний застосунок система управління
проектами»**

Виконав студент групи ІПЗк-22-1
РИЖКО Іван Русланович

Керівник роботи:
САВІЦЬКИЙ Роман Святославович

Рецензент:
ТУРЕНКО Єлизавета Єгорівна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Завідувач кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **РИЖКО Іван Русланович**

Керівник роботи: **САВІЦЬКИЙ Роман Святославович**

Тема роботи: **«Мобільний застосунок системи управління проектами»**,
затверджена Наказом закладу вищої освіти від «23» лютого 2024 р. №74с

Вихідні дані для роботи: **проєктування та розробка мобільного застосунку
системи управління проєктами**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів,
що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Савіцький Р. С.	27.02.2024р.	27.02.2024р.
2	Савіцький Р. С.	27.02.2024р.	27.02.2024р.
3	Савіцький Р. С.	27.02.2024р.	27.02.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з мобільного застосунку системи управління проєктами та пояснювальної записки. Пояснювальна записка містить: 67 сторінок, 16 ілюстрацій, 8 таблиць та 30 літературних джерел.

Метою даної роботи є створення програмних засобів для автоматизації вирішення задачі «Мобільний застосунок для управління проєктами» у рамках системи «TeamWorker». Система призначена для підвищення ефективності управління проєктами за допомогою сучасних технологій та інструментів.

Пояснювальна записка містить результати розробки автоматизованої системи. Проведено аналіз вхідних і вихідних даних задачі, описано організацію інформаційної бази, а також розроблені логічна і фізична моделі даних.

Програмні рішення задачі були розроблені в Android Studio та IntelliJ IDEA. Реалізовані програмні процедури забезпечують роботу системи в цілому та її взаємодію із загальносистемною базою даних.

КЛЮЧОВІ СЛОВА: МОБІЛЬНИЙ ЗАСТОСУНОК, УПРАВЛІННЯ ПРОЄКТАМИ, КЛІЄНТ-СЕРВЕР, КОМАНДНА РОБОТА, CRM, GET, POST, ANDROID, TEAMWORKER

					ІПЗ.КР.Б – 121 – 24 – ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		Рижко І.Р.			Мобільний застосунок системи управління проектами		Літ.	Арк.	Аркушів
Керівник.		Савіцький Р. С.						3	67
							Житомирська політехніка, група ІПЗк-22-1		
Рецензент		Туренко Є.Є.							
Зав. каф.		Вакалюк Т. А.							

ABSTRACT

The bachelor's qualification thesis consists of a mobile application for a project management system and an explanatory note. The explanatory note includes: 67 pages, 16 illustrations, 8 tables, and 30 references.

The aim of this work is to create software tools for automating the task «Android application for project management» within the «TeamWorker» system. This system is designed to enhance project management efficiency using modern technologies and tools.

The explanatory note contains the results of the development of the automated system. An analysis of the task's input and output data has been conducted, the organization of the information database is described, and the logical and physical data models are developed.

The software solutions for the task were developed using Android Studio and IntelliJ IDEA. The implemented software procedures ensure the operation of the system as a whole and its interaction with the overall system database.

KEYWORDS: MOBILE APPLICATION, PROJECT MANAGEMENT, CLIENT-SERVER, TEAMWORK, CRM, GET, POST, ANDROID, TEAMWORKER

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ.....	9
1.1 Постановка задачі.....	9
1.2 Аналіз аналогів програмного продукту	11
1.3 Вибір архітектури мобільного застосунку системи управління проєктами .	18
1.4 Обґрунтування вибору інструментальних засобів.....	20
Висновки до розділу 1	23
РОЗДІЛ 2. ПРОЄКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ	25
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи	25
2.2 Розробка бази даних системи.....	33
2.3 Проєктування та реалізація алгоритмів роботи системи	36
2.4 Реалізація мобільного застосунку управління проєктами	38
Висновки до розділу 2.....	44
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ МОБІЛЬНОГО ЗАСТОСУНКУ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ.....	45
3.1 Порядок встановлення та налаштування параметрів системи	45
3.2 Структура інтерфейсу мобільного застосунку системи управління проєктами	46
3.3 Тестування роботи програмного комплексу	54
Висновки до розділу 3.....	56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	58
ДОДАТКИ.....	61

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		5

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CRM – система управління взаємовідносинами з клієнтами (Customer Relationship Management)

ERP – система планування ресурсів підприємства (Enterprise Resource Planning)

HRM – управління людськими ресурсами (Human Resource Management)

HTTP – протокол передачі гіпертексту

JVM – віртуальна машина Java (Java Virtual Machine)

SQL – мова структурованих запитів

АС – автоматизована система

БД – база даних

ОС – операційна система

ПК – персональний комп'ютер

СУБД – система управління базою даних

СУП – система управління проектами

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність. У сучасному світі управління проєктами стає все більш важливим аспектом ефективної діяльності компаній та організацій. З розвитком технологій виникає потреба у створенні інструментів, які б забезпечували зручне та ефективне управління проєктами, особливо в умовах мобільності та дистанційної роботи. Тема мобільного додатку системи управління проєктами є надзвичайно актуальною, оскільки вона відповідає сучасним вимогам ринку, спрямованим на оптимізацію робочих процесів та підвищення продуктивності команд.

Метою цієї роботи є розробка мобільного додатку, який забезпечить зручне та інтуїтивно зрозуміле управління проєктами, дозволить координувати діяльність команди, контролювати виконання завдань та ефективно розподіляти ресурси. Додаток має спрощувати процеси планування, моніторингу та звітності, а також сприяти швидкому прийняттю рішень на основі актуальних даних.

Для досягнення поставленої мети необхідно вирішити ряд завдань, серед яких:

1. Аналіз існуючих рішень на ринку;
2. Визначення вимог користувачів до функціональності додатку;
3. Розробка архітектури та інтерфейсу користувача;
4. Реалізація програмного забезпечення та його тестування;
5. Оцінка ефективності використання додатку в реальних умовах.

Об'єктом дослідження є процес управління проєктами в умовах сучасних організацій, що характеризується високою динамічністю, необхідністю швидкого реагування на зміни та забезпеченням високої продуктивності команди.

Предметом дослідження є мобільний додаток, призначений для управління проєктами, його функціональність, архітектура, дизайн інтерфейсу та вплив на ефективність роботи команди.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

У процесі виконання роботи були використані такі методи аналізу, як системний аналіз, методи проєктування програмного забезпечення, юзабіліті-тестування, а також методи оцінки продуктивності та ефективності роботи користувачів з мобільним додатком. Ці методи дозволили всебічно дослідити предмет дослідження та отримати об'єктивні результати.

Практичне значення одержаних результатів полягає у створенні мобільного додатку, який може бути використаний як в малих, так і в великих компаніях для оптимізації процесів управління проєктами. Запропоноване рішення дозволяє підвищити продуктивність роботи команд, знизити витрати часу на виконання рутинних завдань та забезпечити більш гнучке і ефективне управління ресурсами. Результати роботи можуть бути основою для подальших досліджень та розробок у галузі управління проєктами.

Тези до даної кваліфікаційної роботи [1].

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ

1.1 Постановка задачі

У сучасному світі, де бізнес-процеси постійно ускладнюються, а конкуренція на ринку зростає, ефективне управління проєктами стає критично важливим для успіху будь-якої організації. Проєкти можуть бути різного масштабу та складності: від невеликих внутрішніх ініціатив до масштабних міжнародних програм. У будь-якому випадку, керівникам проєктів необхідні інструменти, які допомагають планувати, координувати та контролювати всі аспекти проєктної діяльності.

Системи управління проєктами (СУП) призначені для того, щоб забезпечити централізоване управління всіма процесами, пов'язаними з проєктами, зокрема плануванням, виконанням, моніторингом та завершенням проєктів. Вони дозволяють оптимізувати використання ресурсів, зменшити витрати, підвищити продуктивність команди та забезпечити своєчасне досягнення проєктних цілей. Такі системи стають невід'ємною частиною управління бізнесом у різних галузях, включаючи ІТ, будівництво, фінанси, охорону здоров'я та багато інших.

Ефективна СУП повинна відповідати ряду ключових вимог. Вона має бути гнучкою, щоб адаптуватися до специфіки різних проєктів та галузей. Крім того, система повинна бути масштабованою, щоб підтримувати як невеликі команди, так і великі організації з сотнями проєктів одночасно. Інтуїтивно зрозумілий інтерфейс та зручність у використанні також є важливими аспектами, які сприяють швидкому впровадженню системи та зменшують потребу в тривалому навчанні користувачів.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

Сучасні СУП також повинні забезпечувати можливість інтеграції з іншими корпоративними системами, такими як ERP, CRM та HRM, що дозволяє забезпечити безперервний потік інформації та оптимізувати процеси на всіх рівнях організації. Важливою складовою є забезпечення безпеки даних, адже інформація про проекти часто містить конфіденційні дані, що потребують захисту.

З розвитком технологій, системи управління проектами все частіше використовують хмарні рішення, що дозволяє забезпечити доступ до системи з будь-якої точки світу. Хмарні СУП також забезпечують масштабованість та гнучкість, дозволяючи легко адаптуватися до змін у вимогах та навантаженнях.

Ще одним важливим аспектом є аналітика та звітність. Сучасні СУП повинні надавати інструменти для аналізу даних, що дозволяють керівникам проектів приймати обґрунтовані рішення на основі актуальної інформації. Це включає в себе можливості для створення звітів, графіків, діаграм та інших візуалізацій, які допомагають оцінювати прогрес проектів, виявляти потенційні проблеми та приймати коригувальні заходи.

Таким чином, розробка системи управління проектами є складним та багатогранним завданням, що вимагає ретельного планування, аналізу вимог та високого рівня технічної компетенції. Впровадження такої системи може значно підвищити ефективність управління проектами в організації, сприяти досягненню стратегічних цілей та забезпечити конкурентні переваги на ринку. Для здійснення проектної діяльності сучасні компанії використовують спеціальні програми, які створюють єдиний інформаційний простір для управління проектами.

Цілі та завдання автоматизації управління проектами компанії включають:

1. Зростання ефективності працівників, що займаються проектною діяльністю.
2. Підвищення якості роботи керівників під час управління проектами.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

3. Зростання ефективності управління проєктами компанії в цілому.

Відповідно до теми дипломного проєкту, програма «TeamWorker» має забезпечувати управління проєктами та робочими процесами працівників. Впровадження такої системи збільшить швидкість обробки та збору необхідної інформації, підвищить продуктивність працівників.

Аналіз показав, що додаток з управління проєктами раціонально розробляти з акцентом на функціонал для рядового працівника, тому було реалізовано такі функції:

1. Рівні доступу.
2. Перегляд списку завдань працівника.
3. Зміна статусу завдання.
4. Перегляд деталей про конкретне завдання.
5. Перегляд статистики користувача за категорією.

Отже, згідно з аналізом предметної області, можна зробити висновок, що задача, яка підлягає автоматизації, належить до типових аналітично-розрахункових задач і може бути вирішена з використанням програмних та інструментальних засобів, передбачених технічним завданням, а саме з використанням елементів файл-серверних баз даних.

У результаті розробки буде автоматизовано процес роботи з проєктами та працівниками в ІТ-компаніях. Переваги проєкту будуть реалізовані за рахунок зручного користувацького інтерфейсу та впровадження додаткових засобів забезпечення безпеки, рівнів доступу та збереження даних

1.2 Аналіз аналогів програмного продукту

Ефективний контроль виконання робіт є ключовим елементом операційного процесу підприємства, оскільки він безпосередньо визначає фінансові результати. При розгляді вибору підходів до контролю важливо аналізувати як

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

позитивні аспекти, так і можливі обмеження кожного з них перед прийняттям рішення.

При контролі виконання робочих процесів використовують різні спеціальні програми, для прикладу розглянемо програму Airtable[2].

Airtable є відомою інноваційною платформою для управління базами даних, яка поєднує в собі функціонал електронних таблиць з потужністю традиційних баз даних. Ця унікальна комбінація робить Airtable надзвичайно корисною для широкого кола завдань, починаючи від простих організаційних задач і закінчуючи складними бізнес-проектами. Основна перевага Airtable полягає в її інтуїтивно зрозумілому інтерфейсі (рис. 1.1), що нагадує звичайні електронні таблиці. Це спрощує введення та обробку даних, зменшуючи необхідність у глибоких технічних знаннях.

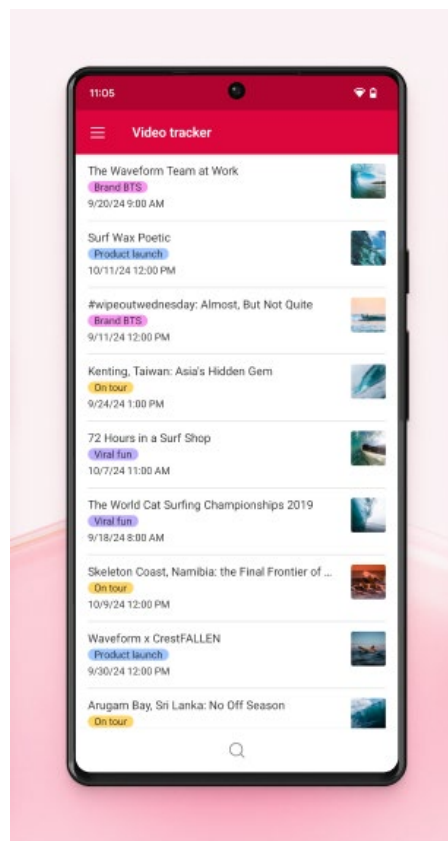


Рисунок 1.1 – Airtable

Крім основних можливостей, Airtable пропонує ряд додаткових інструментів для ефективного управління даними. Серед них — можливість зв'язування різних таблиць для створення складних баз даних, використання

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

різноманітних типів полів (таких як текстові, числові, дати та файли), а також можливість інтеграції з іншими платформами для забезпечення ще більшої функціональності. Це робить Airtable надзвичайно гнучкою і адаптивною для вирішення різних завдань в управлінні даними.

Застосування Airtable розповсюджується на широкий спектр сфер, включаючи управління проєктами, CRM-системи для ведення взаємин з клієнтами, а також індивідуальні бізнес-проєкти. Система дозволяє створювати налаштовані рішення, ідеально підходять для специфічних бізнес-процесів чи особистих проєктів. Завдяки своїм унікальним можливостям і простоті використання, Airtable здобула величезну популярність серед користувачів, що цінують поєднання легкості електронних таблиць із потужністю баз даних.

Варто зазначити, що, незважаючи на переваги, Airtable має деякі обмеження. Наприклад, вона залежить від наявності інтернет-з'єднання, оскільки функціонує у форматі хмарного сервісу, що може обмежувати доступність у офлайн-режимі. Крім того, в порівнянні зі спеціалізованими CRM-системами, Airtable може не мати всіх необхідних функцій, наприклад, повноцінної автоматизації процесів продажу чи детальної аналітики.

Розглянемо програму "Asana" (рис. 1.2), що відома своїм потужним інструментарієм для керування проєктами, забезпечуючи ефективний контроль над виконанням завдань [3]. Використовуючи Asana, можна створювати нові проєкти, приєднувати користувачів, формулювати завдання і розподіляти їх серед учасників команд.

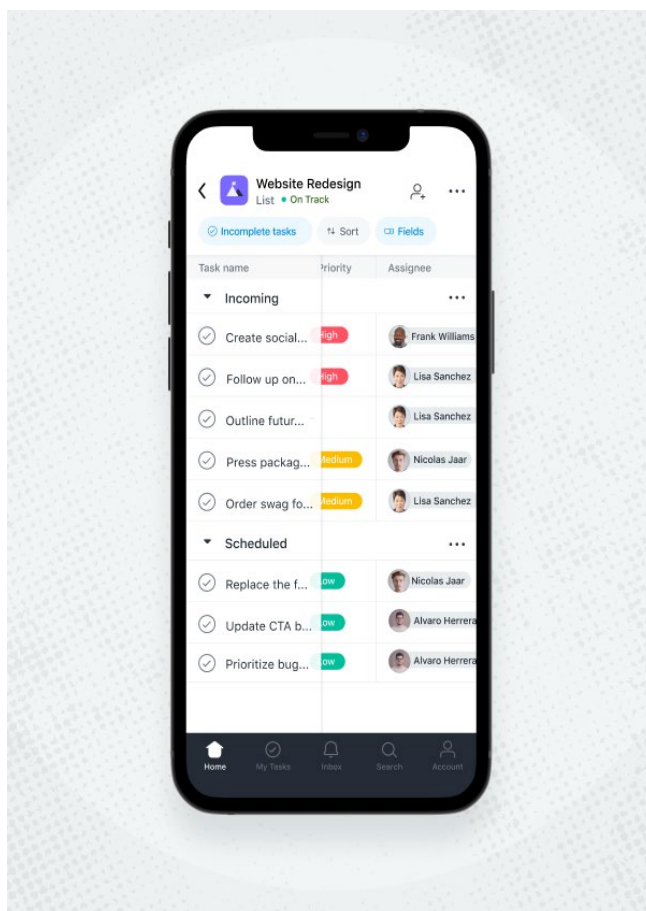


Рисунок 1.2 – Asana

Asana, як інструмент для управління проєктами, пропонує широкі можливості для організації робочих процесів і координації командних зусиль. Адроїд-додаток забезпечує зручний доступ до всіх функцій і функціональних можливостей платформи, що робить його важливим інструментом для професійного використання у рухливому режимі.

Однією з ключових переваг Asana на андроїд є здатність до інтеграції з іншими інструментами і платформами. Це дозволяє користувачам легко синхронізувати і взаємодіяти з даними з різних джерел, що сприяє збільшенню продуктивності та ефективності управління проєктами. Dodatok також надає можливість переглядати і редагувати завдання в режимі реального часу, що сприяє оперативному прийняттю рішень і вчасному реагуванню на зміни у проєктах.

Однак, використання Asana на андроїд може виявитися не зовсім інтуїтивним для нових користувачів. Специфічний дизайн і інтерфейс додатку

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

можуть потребувати певного часу для оволодіння, особливо для тих, хто не має досвіду з подібними системами управління завданнями. Також варто врахувати, що великі обсяги даних чи складність проєктів можуть зробити додаток менш зручним у використанні, вимагаючи додаткового часу на налаштування та усвідомлення всіх його можливостей.

Загалом, Asana є потужним інструментом для організації та управління проєктами, який забезпечує зручний доступ до важливих функцій на мобільному пристрої. Можливості сервісу роблять його незамінним помічником для менеджерів проєктів та командних лідерів, що цінують ефективність і прозорість у робочих процесах.

Jira – це високоефективний інструмент для управління проєктами, який знайшов широке застосування серед розробників програмного забезпечення і команд, що працюють у сфері технологій [4]. Він пропонує розширені можливості для організації завдань, відстеження їх виконання і забезпечення звітності, що дозволяє учасникам команди ефективно співпрацювати над проєктами (рис. 1.3).



Рисунок 1.3 – Jira

Однією з ключових переваг мобільного додатку Jira є його здатність до інтеграції з різними іншими інструментами розробки програмного забезпечення, такими як GitHub і Bitbucket. Це спрощує процес обміну даними і співпраці між командами, особливо в умовах розподіленої роботи, а також у режимі реального часу.

Jira підтримує різні методології управління проєктами, такі як Scrum, Kanban та інші, що дає змогу адаптувати його до конкретних потреб команди і характеру проєкту. Інтерфейс Jira простий у використанні, що робить його доступним для новачків, але в той же час він має достатньо гнучкості для задоволення потреб більш складних проєктів.

Однак для того, щоб повною мірою використовувати всі можливості Jira на андроїд, може знадобитися певний час для ознайомлення з усіма його функціями і налаштуваннями. Деякі користувачі можуть вважати інтерфейс Jira трохи перевантаженим через велику кількість налаштувань і опцій, що потребують ретельного освоєння.

У підсумку, Jira на андроїд є потужним інструментом для керування проєктами, який пропонує широкі можливості для організації та управління різноманітними проєктами. Він дозволяє командам ефективно працювати над завданнями і забезпечувати їх успішне виконання, забезпечуючи при цьому зручний інтерфейс і можливості для адаптації під специфічні потреби користувачів.

На основі отриманих даних була створена таблиця, в якій проведено порівняльний аналіз трьох популярних мобільних додатків для управління проєктами: Airtable, Asana та Jira на платформі Android. Цей аналіз спрямований на виявлення ключових відмінностей між цими інструментами, визначення їх переваг і можливих обмежень. (табл. 1.1):

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		16

Таблиця 1.1

Порівняння існуючих аналогів

	Airtable	Asana	Jira
Інтуїтивно зрозумілий інтерфейс	+	+	-
Швидкість та продуктивність	+	+	-
Базові функції	+	+	+
Бізнес-звітність	-	+	+
Ієрархія співробітників	-	+	+
Доступність (ціна)	-	+	-
Складність в освоєнні	+	+-	-
Додатковий функціонал	-	+-	+

Airtable, Asana і Jira володіють унікальними перевагами та обмеженнями. Airtable виділяється легкістю використання, швидкістю і доступністю за вигідною ціною, проте має обмеження у звітності та ієрархії співробітників. Asana пропонує збалансоване рішення з гнучкою структурою ієрархії та розширеною звітністю для бізнесу, при цьому зберігаючи простоту використання та конкурентоспроможну цінову політику. Jira відома своїми потужними інструментами для колективної роботи, глибокою аналітикою та звітністю, але вона має складний інтерфейс і високий рівень складності у

використанні, що може стати викликом для нових користувачів і вимагати значних витрат на навчання і впровадження.

Зважаючи на порівняльний аналіз, функціональні вимоги до системи управління проєктами спрямовані на створення інтуїтивно зрозумілого інтерфейсу з чіткою навігацією, що спрощує роботу користувачів та мінімізує час навчання. Висока швидкість роботи та продуктивність системи є критично важливими для забезпечення ефективного управління даними та завданнями. Необхідно включити базові функції, такі як створення та редагування записів, календарні інструменти та систему звітності, що дозволяє аналізувати ключові показники бізнесу. Ієрархія співробітників та керування правами доступу також мають бути реалізовані для оптимального управління внутрішніми процесами.

1.3 Вибір архітектури мобільного застосунку системи управління проєктами

При виборі архітектури для мобільного застосунку системи управління проєктами на платформі Android важливо врахувати кілька ключових аспектів, зокрема: ефективність роботи, безпеку даних, зручність інтерфейсу користувача та можливість масштабування системи. У цьому розділі розглянемо вибір технологій та архітектурних рішень, що найкращим чином відповідають цим вимогам.

Java є однією з найпоширеніших мов програмування для розробки Android-додатків [5]. Її об'єктно-орієнтована природа дозволяє створювати надійні та ефективні програми. Використання Java забезпечує широкі можливості для оптимізації та розширення функціоналу застосунку.

XML (Extensible Markup Language) використовується для розмітки користувацьких інтерфейсів у Android-додатках. Це дозволяє чітко визначати структуру сторінок та взаємодію елементів інтерфейсу, що спрощує розробку та підтримку додатку.

Для реалізації серверної частини застосунку було використано Java з фреймворком Spring Boot [6]. Spring Boot дозволяє швидко створювати

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

самостійні, готові до використання додатки, що мають вбудований вебсервер і забезпечують просте керування конфігурацією.

Spring Security було використано для забезпечення безпеки додатку [7]. Він надає різноманітні механізми аутентифікації і авторизації користувачів, захист від атак і управління сесіями, що важливо для забезпечення конфіденційності даних та стійкості до вторгнень.

Spring MVC (Model-View-Controller) використовуватиметься для реалізації вебкомпонентів додатку [8]. Цей паттерн дозволяє чітко відокремлювати логіку додатку (модель), представлення даних (вид) і обробку запитів (контролер), що спрощує розробку і підтримку проєкту. Архітектуру MVC зображено на рисунку 1.4.

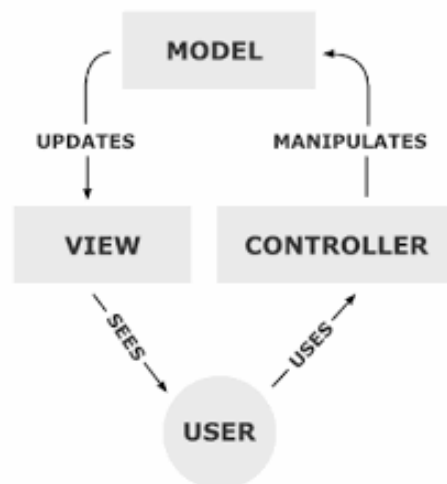


Рисунок 1.4 – Архітектура MVC

Для мобільного застосунку системи управління проєктами використано клієнт-серверну архітектуру.

Клієнтська частина реалізована на Java з використанням XML для опису інтерфейсу користувача. Це забезпечує швидкий доступ до даних і зручність для користувача.

Серверна частина побудована на Java з використанням Spring Boot для створення RESTful API [9]. Spring Security гарантує безпеку передачі даних між клієнтом та сервером, а Spring MVC забезпечує ефективну обробку запитів і відповідей.

База даних реалізована на MySQL, що забезпечує структуроване зберігання і ефективний доступ до інформації про проекти, завдання та користувачів [10]. Використання SQL дозволяє виконувати різноманітні операції з даними, включаючи створення таблиць, вставку даних та складні запити, що забезпечують оптимальну роботу з базою даних у всіх компонентах системи.

Ця архітектурна модель дозволяє ефективно керувати проектами, забезпечує зручний інтерфейс для користувачів та надійність системи. Вона також забезпечує можливість масштабування і розширення функціональності за необхідності.

1.4 Обґрунтування вибору інструментальних засобів

Вибір інструментів для розробки додатків був здійснений з урахуванням передових технологій та можливостей. Android Studio використовується для створення мобільних додатків на Java, забезпечуючи розширені можливості для створення ефективних інтерфейсів. Це інтегроване середовище дозволяє зручно керувати станом додатків, взаємодіяти з серверами та надавати багатофункціональність для користувачів.

Android Studio відоме своєю простотою використання та здатністю швидко розширювати функціонал, що зробило його популярним серед розробників мобільних додатків [11].

Сьогодні Android Studio є одним із ключових інструментів для створення мобільних додатків, що надає широкий спектр можливостей, включаючи розробку користувацьких інтерфейсів за допомогою XML, управління компонентами, роботу з мережами, підтримку архітектурних компонентів та інтеграцію з серверними API.

Основні переваги Android Studio полягають у зручному інтерфейсі, високій продуктивності завдяки передовим методам оптимізації та вбудованій підтримці тестування. Оволодіння Android Studio може зайняти трохи часу для ознайомлення з його особливостями та урахування специфікацій проекту.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

Android Studio ідеально підходить для розробки як простих мобільних додатків, так і складних корпоративних проєктів, які потребують надійного інструменту з високою продуктивністю.

Java є однією з найпопулярніших мов програмування завдяки своїм ключовим перевагам, таким як відкритий вихідний код та широке використання в різних сферах, включаючи корпоративні застосунки, мобільні додатки і вебсервіси. Основною перевагою Java є її платформа-незалежність, що дозволяє виконувати програми на будь-якій операційній системі, яка підтримує JVM (Java Virtual Machine) [12]. Це робить Java дуже привабливою для розробників, які прагнуть створювати програмне забезпечення, що може працювати на різних платформах без необхідності вносити зміни в код.

Java також відома своєю надійністю і безпекою, що робить її ідеальною для розробки великих і критично важливих систем. Система управління пам'яттю і вбудований механізм збору сміття допомагають уникнути багатьох типів помилок, які можуть виникати при ручному управлінні пам'яттю в інших мовах програмування [13].

Однією з важливих характеристик Java є велика стандартна бібліотека, яка надає розробникам широкий набір готових до використання функцій і класів, що значно прискорює процес розробки. Крім того, наявність розвиненої екосистеми інструментів і фреймворків, таких як Spring, Hibernate і Maven, дозволяє створювати складні і ефективні програми з меншими зусиллями.

Java також має активну і велику спільноту розробників, що сприяє швидкому вирішенню проблем, обміну знаннями і досвідом. Багато компаній і організацій надають підтримку для Java, забезпечуючи її постійний розвиток і вдосконалення.

Завдяки своїй універсальності, Java використовується в різних галузях, включаючи фінанси, банківську справу, телекомунікації, охорону здоров'я та багато інших. Мобільні додатки, створені для операційної системи Android, також значною мірою написані на Java, що додатково підкреслює її значущість

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

у сучасному світі програмування. Щодо середовища розробки, Android Studio забезпечує комфортне створення мобільних додатків на Java з використанням XML для створення інтерфейсів та інших компонентів додатку. Інструмент підтримує широкий набір функцій, включаючи автодоповнення коду, візуальне налагодження та підтримку різних фреймворків.

У розробці серверної частини системи використовується фреймворк Spring для Java, що забезпечує розширені можливості для створення вебдодатків. Він надає потужні інструменти для розробки різноманітних додатків, зокрема корпоративних рішень та мікросервісів.

У контексті розробки мобільних додатків для Android, Java залишається одним з ключових інструментів, хоч Kotlin стає все більш популярним [14]. Мова програмування Java довго була основною мовою для Android-розробки і має багато переваг, включаючи зрілість екосистеми та велике ком'юніті розробників, що створюють бібліотеки та інструменти, які полегшують процес створення додатків.

База даних відіграє важливу роль у зберіганні та обробці інформації, забезпечуючи ефективний доступ та маніпулювання даними. Вона структурована спеціальним чином для забезпечення швидкості операцій, високої надійності та можливості одночасного доступу для багатьох користувачів [15]. В сучасних інформаційних системах бази даних є центральним елементом, оскільки вони дозволяють зберігати великі обсяги даних у структурованому вигляді, забезпечують можливість виконання складних запитів та аналітики, а також гарантують цілісність і безпеку даних.

MySQL Workbench є потужним інструментом для роботи з базами даних, який дозволяє виконувати різноманітні завдання, від проєктування структури даних до адміністрування [16]. Цей інструмент забезпечує можливості для резервного копіювання, перенесення та моніторингу баз даних, що робить його незамінним для розробників мобільних додатків на Java. MySQL Workbench пропонує зручний графічний інтерфейс, який спрощує створення і управління

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

базами даних. Він дозволяє проєктувати моделі баз даних, генерувати SQL-скрипти, а також аналізувати і оптимізувати запити.

Для розробників мобільних додатків важливо мати ефективні інструменти для управління базами даних, оскільки дані відіграють критичну роль у функціонуванні додатків. MySQL Workbench забезпечує зручний спосіб взаємодії з базою даних, дозволяючи швидко створювати та модифікувати таблиці, індекси, тригери та інші об'єкти баз даних. Крім того, цей інструмент підтримує інтеграцію з іншими інструментами розробки, що полегшує роботу в команді і забезпечує більш ефективний процес розробки.

Однією з ключових функцій MySQL Workbench є можливість візуального проєктування баз даних. Розробники можуть створювати діаграми сутностей і зв'язків (ER-діаграми), що допомагають уявити структуру бази даних і взаємозв'язки між її елементами [17]. Це особливо корисно на етапі проєктування, коли необхідно ретельно продумати архітектуру системи і забезпечити її масштабованість і ефективність.

MySQL Workbench також забезпечує потужні інструменти для адміністрування баз даних. Розробники можуть виконувати резервне копіювання і відновлення баз даних, що є критичним для забезпечення безпеки даних. Моніторинг продуктивності баз даних дозволяє виявляти і усувати вузькі місця в роботі системи, забезпечуючи високу продуктивність і надійність.

Використання MySQL Workbench у розробці мобільних додатків на Java дозволяє забезпечити високу якість і ефективність процесу розробки. Цей інструмент надає всі необхідні можливості для управління базами даних, від проєктування до адміністрування, що робить його незамінним для сучасних розробників. Завдяки MySQL Workbench, розробники можуть зосередитися на створенні функціональних і надійних мобільних додатків, не витрачаючи зайвий час на вирішення технічних питань, пов'язаних з управлінням базами даних.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до розділу 1

У першому розділі було чітко сформульовано мету розробки мобільного додатку для системи управління проєктами і проведено детальний аналіз існуючих рішень на ринку, таких як Airtable, Jira і Asana. Аналіз дозволив ідентифікувати особливості, переваги і недоліки, що сприяло кращому розумінню потреб і вимог для подальшої розробки власного продукту.

Також було обрано архітектуру для мобільного додатку, включаючи використання таких технологій, як Java, Android SDK та інструменти розробки, такі як Android Studio. Цей вибір був обґрунтований їхньою сумісністю, продуктивністю і популярністю в галузі мобільної розробки.

Описано також обрану архітектуру, включаючи серверну частину з використанням Spring Framework та управління базами даних за допомогою MySQL Workbench. Використання цього інструменту дозволяє ефективно проєктувати, адмініструвати та оптимізувати бази даних, що є критично важливим для забезпечення надійної роботи мобільного додатку.

Для розробки обрано середовище Android Studio, яке надає широкі можливості для написання, відлагодження і управління кодом, а також інтеграцію з іншими інструментами, такими як MySQL Workbench.

Загалом, цей розділ встановив основу для подальшої розробки мобільного додатку системи управління проєктами, який має забезпечити широкий функціонал і зручність у використанні.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		24

РОЗДІЛ 2. ПРОЄКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

Аналіз вимог включає ретельне дослідження потреб та очікувань користувачів щодо функцій і характеристик мобільного додатку для управління проєктами. Основна мета – з'ясувати, які саме функції необхідні для задоволення користувачів. Ми визначили конкретні функціональні вимоги, що описують особливості та можливості програми.

До ключових функцій належать: створення і управління проєктами, завданнями та підзадачами, відстеження прогресу, призначення відповідальних осіб, встановлення термінів і нагадувань, інтеграція з календарями та іншими системами управління часом [18]. Окрім того, враховувалися нефункціональні вимоги, які стосуються продуктивності, стабільності та зручності використання. Вимоги до продуктивності включають швидке завантаження і обробку даних, ефективне використання ресурсів пристрою та мінімальний вплив на інші додатки. Для стабільності додаток повинен бути надійним, з мінімальною кількістю помилок та збоїв, а також мати функцію автоматичного відновлення після збоїв.

Зручність використання передбачає інтуїтивно зрозумілий інтерфейс, легкий для нових користувачів, та доступність для користувачів з різними потребами. Додаток має бути адаптивним до різних розмірів екранів і підтримувати різні орієнтації пристроїв.

Аналіз вимог також охоплював безпекові аспекти для захисту конфіденційної інформації користувачів і забезпечення безпечного обміну даними, включаючи шифрування, використання безпечних протоколів передачі даних та механізми аутентифікації і авторизації.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

Отже, аналіз вимог дозволив нам чітко зрозуміти, які функції і характеристики необхідні для створення ефективного і зручного мобільного додатку для управління проєктами. Визначення функціональних і нефункціональних вимог забезпечить розробку продукту, що відповідатиме очікуванням користувачів та забезпечить оптимальну функціональність і задоволення від його використання.

Функціональні вимоги

1. Можливість реєстрації та авторизації користувачів. Система має забезпечити можливість користувачам створювати облікові записи і входити в них для використання функціоналу програми.
2. Система має надавати детальну статистику щодо проєктів, завдань і працівників, включаючи виконання строків, ефективність, продуктивність та завантаженість.
3. Система має забезпечувати можливість перегляду завдань для робітників з детальною інформацією, такою як опис завдання, пріоритет, терміни виконання та відповідальні особи. Користувачі повинні мати можливість змінювати статус завдання.

Нефункціональні вимоги

1. Сприйняття
 - a. Відповідь на звичайні запити має займати не більше 3 секунд, а на складні — не більше 15 секунд.
 - b. Інтерфейс повинен бути легким у використанні і інтуїтивно зрозумілим, щоб користувачі могли без зусиль здійснювати покупки та користуватися сервісом.
2. Надійність
 - a. Час, що витрачається на обслуговування платформи, не повинен перевищувати 10% від загального часу роботи.
 - b. Середній час безвідмовної роботи повинен складати 60 робочих днів.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

- с. Максимальна частка помилок та дефектів в роботі платформи — не більше однієї помилки на 1000 запитів користувача.

3. Продуктивність

- а. Система має бути швидкою і масштабованою, здатною обробляти великий потік користувачів і запитів без помітних затримок.

4. Можливість експлуатації

- а. Система повинна мати можливість масштабуватися для відповіді на зростаючий обсяг користувачів, не знижуючи швидкості роботи.
- б. Оновлення версій повинні відбуватися в часи мінімального навантаження на платформу, з мінімізацією перерв у роботі сайту під час завантаження оновлень.

Часові характеристики реакції системи

1. Інстантний відгук на запити користувачів. Система має надавати швидкий доступ до інформації та послуг.
2. Негайне оновлення задач. Система має миттєво відображати зміни, внесені адміністратором, такі як додавання нових задач чи оновлення вже виконаних.
3. Мінімізація часу реагування на помилки. Система повинна негайно реагувати на будь-які помилки чи проблеми під час взаємодії з користувачем, щоб швидко вирішувати проблеми та надавати необхідну допомогу.

Система пропонує різноманітні сценарії використання для задоволення різноманітних потреб користувачів, враховуючи, що характеристики оперативного реагування мобільного застосунку для системи управління проектами залежать від таких факторів, як обладнання користувача, обсяг і складність запитів до бази даних, а також оптимізація самої системи.

Діаграма варіантів використання представляє візуальну взаємодію між системою та користувачами (рис. 2.1) [19] .

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

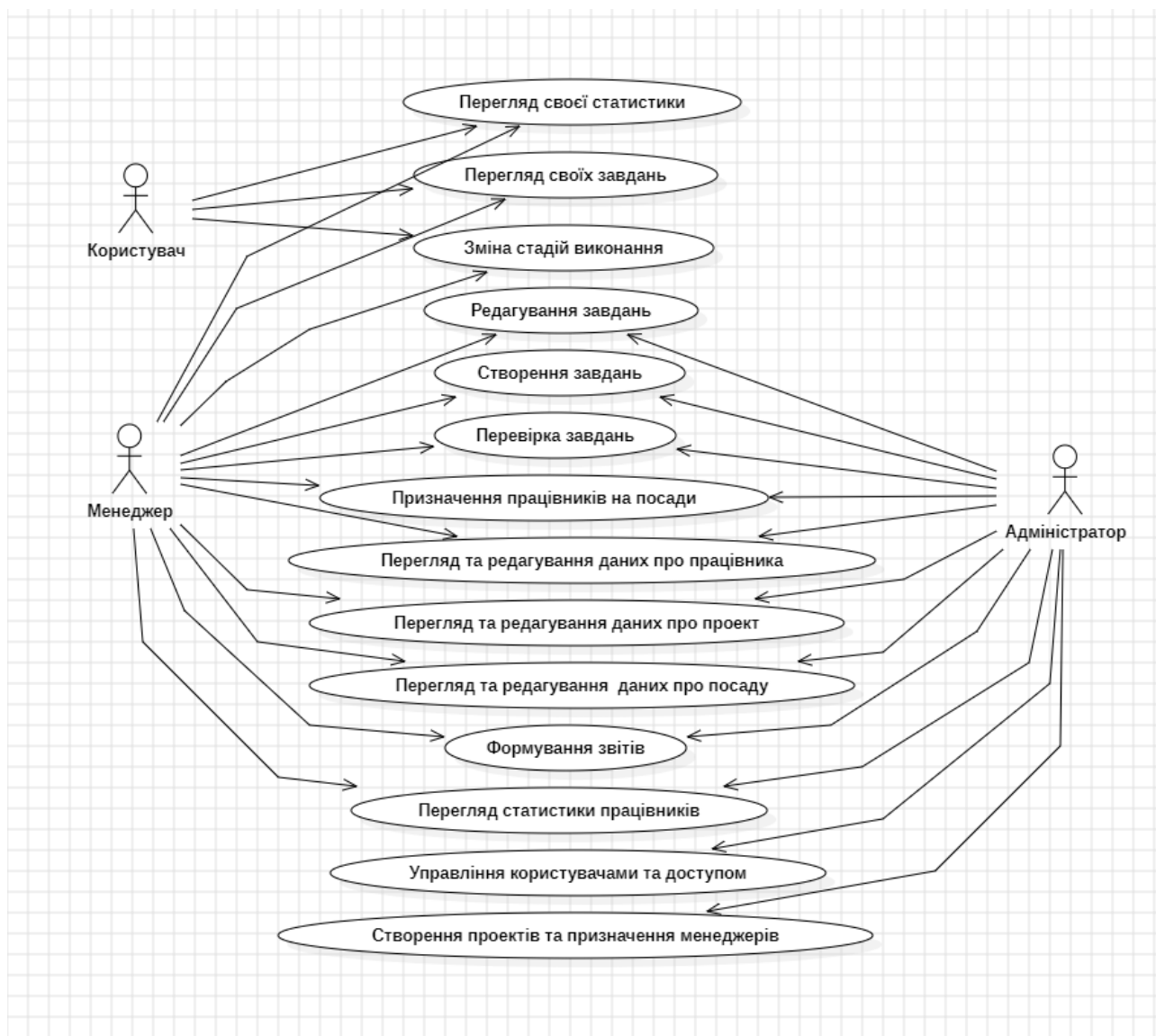


Рисунок 2.1 – Діаграма варіантів використання

Основні актори:

Користувач – цей актор має можливість виконувати такі дії, як перегляд своїх завдань, зміни статусів завдань, перегляд статистики по виконаних завданнях.

Менеджер має розширені права порівняно зі звичайним користувачем. Він може переглядати та редагувати дані по працівниках, а також отримувати звіти та статистику по проектам.

Адміністратор має найвищий рівень доступу. Він відповідає за адміністрування системи, включаючи управління користувачами та доступами, налаштування системних параметрів, перегляд і редагування даних проектів та створення звітів.

На зображенні показані наступні функції та дії:

Користувач:

1. Перегляд своїх завдань.
2. Зміна статусу завдань.
3. Перегляд статистики виконання завдань.

Менеджер:

1. Перегляд завдань.
2. Зміна статусу завдань.
3. Редагування завдань.
4. Перегляд та редагування даних про працівників.
5. Перегляд та редагування даних про проекти.
6. Перегляд та редагування даних про посади.
7. Перегляд звітів та статистики.

Адміністратор:

1. Управління користувачами та доступами.
2. Налаштування системних параметрів.
3. Перегляд звітів по всіх користувачам і проектам.
4. Перегляд завдань.
5. Редагування завдань.
6. Перегляд та редагування даних про працівників.
7. Перегляд та редагування даних про проекти.
8. Перегляд та редагування даних про посади.
9. Перегляд звітів та статистики.

Хоча діаграма представляє весь функціонал системи «TeamWorker», мобільний додаток для Android реалізує лише функції для звичайного користувача. Це рішення було прийнято з міркувань раціональності, оскільки адміністрування системи на мобільному пристрої є менш зручним і ефективним. Таким чином, мобільний додаток зосереджується на забезпеченні зручності та

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		29

ефективності для кінцевого користувача, дозволяючи йому керувати своїми завданнями та переглядати особисту статистику.

Короткий опис прецедентів:

1. Реєстрація. Користувач системи реєструється на сайті, щоб отримати доступ до основних функцій (див. рис. 2.1).
2. Вхід в систему. Користувач системи входить у свій обліковий запис для доступу до персоналізованих функцій (див. рис. 2.1).
3. Вихід із системи, Користувач системи виходить зі свого облікового запису.
4. Перегляд завдань.
5. Перегляд особистої статистики..

Діаграма варіантів використання забезпечує всебічне та наочне уявлення про функціональні можливості системи. Вона визначає та документує різні сценарії використання, стаючи важливим інструментом для аналізу системи, збору вимог та проєктування.

Об'єктно-орієнтована структура. Такий підхід дозволяє створювати модульний дизайн системи, де різні компоненти представлені як об'єкти з певними атрибутами та методами [20]. Кожен об'єкт в системі виконує чітко визначені функції, що дозволяє розділити відповідальність між різними частинами системи. Завдяки інкапсуляції функціональності в об'єкти, система сприяє повторному використанню коду, забезпечуючи при цьому модульність та розширюваність. Це означає, що існуючі об'єкти можна використовувати в інших частинах системи або в інших проєктах, що значно зменшує час і зусилля, необхідні для розробки нових функціональностей.

На узагальненій діаграмі об'єктно-орієнтованої структури мобільного застосунку для управління проєктами (рис. 2.2) зображені взаємозв'язки та взаємодія між різними об'єктами системи. Діаграма показує, як різні об'єкти взаємодіють один з одним, обмінюючись даними та виконуючи спільні задачі. Кожна сутність представляє певний компонент або модуль програмного

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

продукту зі своїми атрибутами та методами, що відповідають за виконання конкретних функцій. Діаграма слугує візуальним представленням архітектури системи, допомагаючи краще зрозуміти її загальну структуру та взаємозв'язки. Вона також сприяє кращому плануванню та управлінню проєктом, оскільки дозволяє чітко бачити, як різні частини системи взаємодіють одна з одною та як зміни в одній частині можуть вплинути на інші частини системи.

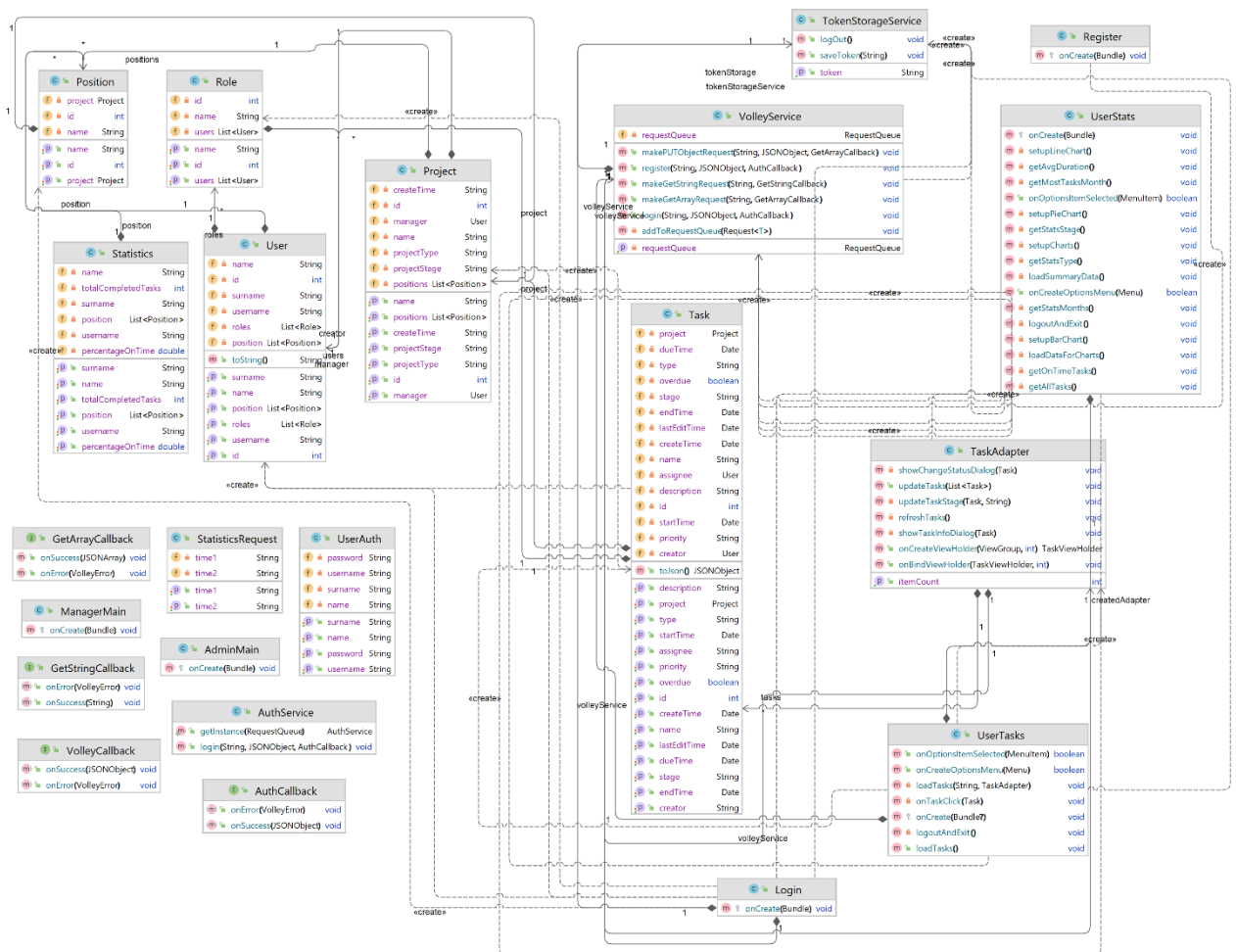


Рисунок 2.2 – Діаграма класів

Position (Посада): Представляє робочу посаду в рамках проєкту. Містить атрибути, такі як name (назва), id, project (проєкт) та пов'язані users (користувачі).

Role (Роль): Визначає роль користувача в проєкті, включаючи атрибути, такі як id, name (назва) та список пов'язаних positions (посад).

Project (Проект): Представляє сутність проєкту з атрибутами, такими як `id`, `name` (назва), `description` (опис), `projectType` (тип проєкту) та асоціаціями з `positions` (посадами) та `tasks` (завданнями).

Statistics (Статистика): Збирає статистичні дані, пов'язані з активністю користувача, такими як `percentageOnTime` (кількість вчасно).

User (Користувач): Представляє користувача з атрибутами, такими як `id`, `username` (ім'я користувача), `password` (пароль) та асоціаціями з `roles` (ролями), `tasks` (завданнями), `projects` (проєктами) та `positions` (посадами).

VolleyService (Сервіс Volley): Керує HTTP-запитами з методами, такими як `makePUTRequest`, `makeGETRequest`, `register` та `addRequestQueue`.

TokenStorageService (Сервіс зберігання токенів): Керує зберіганням токенів, надаючи методи `saveToken` (зберегти токен) та `getToken` (отримати токен).

Register (Реєстрація): Керує реєстрацією користувачів з через запити `VolleyService`.

UserStats (Статистика користувача): Клас сторінки статистики. Виводить дані про роботу працівника, з такими методами як `setupCharts` (налаштувати графіки).

Task (Завдання): Представляє завдання в рамках проєкту, включаючи атрибути, такі як `id`, `name` (назва), `project` (проєкт), `description` (опис), `dateStart` (дата початку), `dateEnd` (дата завершення) та `priority` (пріоритет).

TaskAdapter (Адаптер завдань): Адаптує дані завдань для відображення, з методами `showChangeStatusDialog`, `updateList`, `notifyDataSetChanged` та обробкою взаємодій із завданнями.

UserTasks (Завдання користувачів): Керує відображенням завдань, специфічними для користувачів, надаючи метод `loadTasks`.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

Login (Вхід): Керує входом користувача.

GetArrayCallback (Колбек для отримання масиву): Визначає інтерфейс колбеку для керування відповідями масиву з методом onSuccess.

GetStringCallback (Колбек для отримання рядка): Визначає інтерфейс колбеку для керування відповідями рядка з методом onSuccess.

StatisticsRequest (Запит статистики): Представляє запит на статистичні дані, включаючи атрибути, такі як time1, time2 та username (ім'я користувача).

AuthService (Сервіс автентифікації): Керує сервісами автентифікації з методом getInstance.

AuthCallback (Колбек автентифікації): Визначає інтерфейс колбеку для відповідей автентифікації з методами onSuccess та onError.

VolleyCallback (Колбек Volley): Визначає інтерфейс колбеку для відповідей Volley з методами onSuccess та onError.

2.2 Розробка бази даних системи

Для зберігання файлів та інформації в Android-додатку для управління проектами використовується MySQL, що є системою управління базами даних, яка базується на реляційній моделі. В даному контексті, ER-модель (модель сутність-зв'язок) зазвичай реалізується у вигляді бази даних [21]. При використанні реляційної бази даних, дані організовуються у таблиці, де кожен рядок представляє окремий екземпляр певної сутності. У таблицях містяться різноманітні поля, серед яких деякі є індексами, що слугують для вказівки на відповідні записи в інших таблицях. Індокси виконують функцію покажчиків, забезпечуючи фізичну реалізацію зв'язків між сутностями в базі даних.

Android-додаток для управління проектами використовує MySQL не лише для базового зберігання даних, але й для забезпечення ефективного доступу та маніпуляції інформацією. ER-модель (рис. 2.3), яка лежить в основі структури

бази даних, дозволяє чітко визначати сутності, їх атрибути та взаємозв'язки між ними. У реляційній моделі бази даних, дані організовані у вигляді таблиць, що складаються з рядків та стовпців [22]. Кожен рядок відповідає конкретному екземпляру сутності, а стовпці представляють атрибути цієї сутності. Деякі з цих стовпців містять спеціальні значення – індекси, які посилаються на інші записи у базі даних, тим самим реалізуючи логічні зв'язки між різними сутностями. Ця структура дозволяє легко виконувати складні запити, отримувати необхідні дані та забезпечувати цілісність інформації в системі.

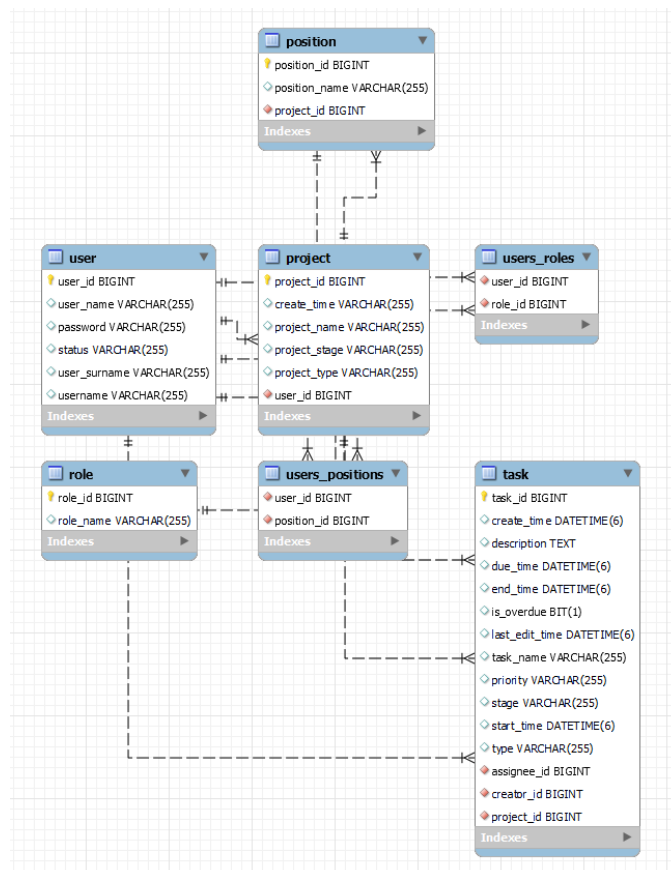


Рисунок 2.3 – ER-модель

Розглянемо структуру бази даних для заданої предметної області.

Основними сутностями цієї структури є:

1. Користувачі – містить дані про користувачів програми;
2. Ролі – містить дані про ролі користувачів для реалізації гнучких рівнів доступу;
3. Посади – містить данні про посади на проєктах компанії;
4. Проєкти – містить інформацію про проєкти компанії;

5. Завдання – містить детальну інформацію про всі завдання для працівників в компанії;
6. Посади користувачів – таблиця, що містить дані про користувачів програми.

Структура полів таблиці User наведено у таблиці 2.1.

Таблиця 2.1

Назва поля	Тип	Інформаційний зміст
user_id	Числовий	id користувача
login	Текстовий	логін користувача
password	Текстовий	пароль користувача
name	Текстовий	ім'я користувача
surname	Текстовий	прізвище користувача
status	Текстовий	статус користувача

Ролі – таблиця, що містить інформацію про ролі користувача.

Структура полів таблиці Roles наведено у таблиці 2.2.

Таблиця 2.2

Назва поля	Тип	Інформаційний зміст
Role_id	Числовий	Код ролі
Role_name	Текстовий	Назва ролі

Посади – таблиця, що містить інформацію про посади користувачів.

Структура полів таблиці Position наведено у таблиці 2.3.

Таблиця 2.3

Назва поля	Тип	Інформаційний зміст
Position_id	Числовий	Код посади
Position_name	Текстовий	Назва посади
Project_id	Числовий	Код проєкту

Проекти – таблиця, що містить дані про проєкти.

Структура полів таблиці Project наведено у таблиці 2.4.

Таблиця 2.4

Назва поля	Тип	Інформаційний зміст
Project_id	Числовий	id проєкту
Project_name	Текстовий	назва проєкту
Create_time	Текстовий	Дата створення
Project_stage	Текстовий	стадія проєкту
Project_type	Текстовий	тип проєкту

Завдання – таблиця, що містить інформацію про всі завдання компанії.

Структура полів таблиці Task наведено у таблиці 2.5.

Таблиця 2.5

Назва поля	Тип	Інформаційний зміст
Task_id	Числовий	Код завдання
Create_time	Дата	Час створення
Description	Текстовий	Опис
Due_time	Дата й час	Дедлайн завдання
End_time	Дата й час	Дата закінчення виконання
Is_overdue	Логічний	Дедлайн минув
Last_edit_time	Дата й час	Дата останнього змінення
Task_name	Текстовий	Назва завдання
Priority	Текстовий	Пріоритет виконання
Stage	Текстовий	Стадія виконання
Start_time	Дата й час	Дата початку виконання завдання
type	Текстовий	Тип завдання
Assignee_id	Числовий	Код виконавця завдання
Creator_id	Числовий	Код працівника що створив завадання
Project_id	Числовий	Код проєкту

2.3 Проєктування та реалізація алгоритмів роботи системи

Система управління проєктами являється досить простою для розуміння, і має не важкий алгоритм роботи. Алгоритм роботи з платформою представлено на рисунку 2.4.

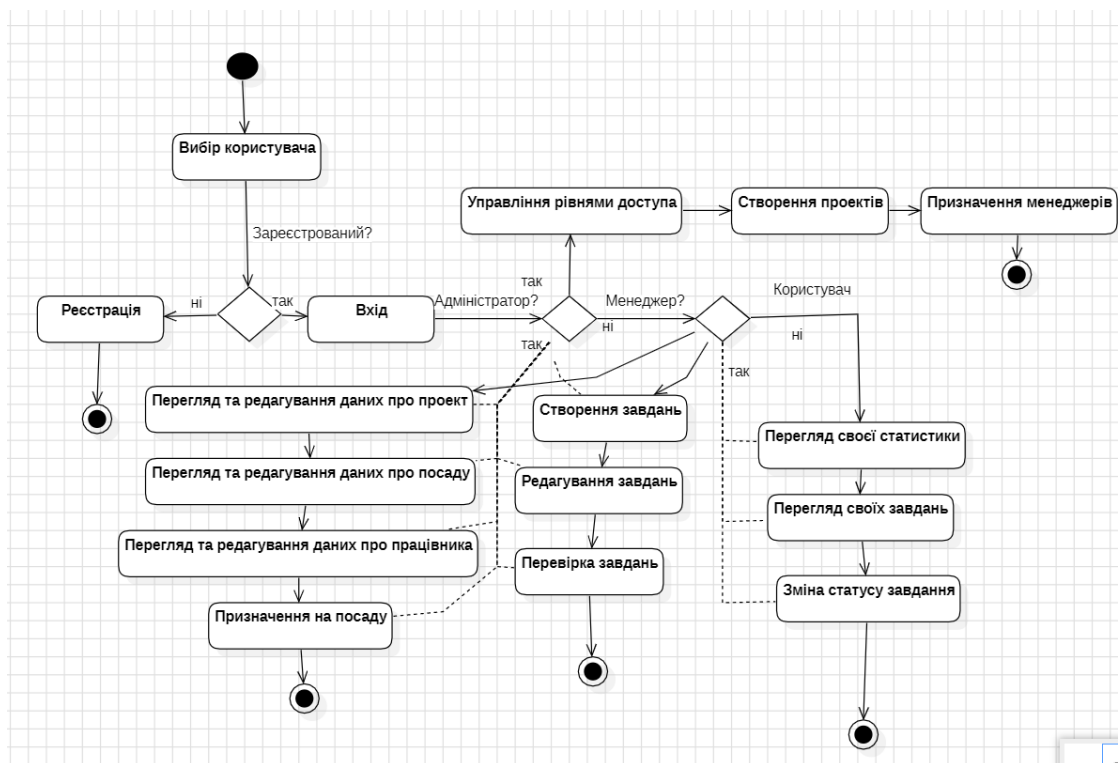


Рисунок 2.4 – Діаграма активності

Якщо авторизувався адміністратор:

- створення нових завдань;
- редагування завдань;
- перегляд статистики виконання завдань;
- перегляд завдань;
- перегляд та редагування даних про працівників;
- перегляд та редагування даних про проєкти;
- перегляд звітів та статистики;
- створення проєктів та призначення менеджерів;
- управління користувачами та доступами;
- перегляд та редагування всіх даних системи;
- налаштування системних параметрів;
- перегляд звітів по всім користувачам і проєктам.

Якщо авторизувався менеджер:

- створення нових завдань;
- редагування завдань;

- зміна статусу завдання;
- перегляд статистики виконання завдань;
- перегляд завдань;
- перегляд своїх завдань;
- редагування завдань;
- перегляд та редагування даних про працівників;
- перегляд та редагування даних про проекти;
- перегляд звітів та статистики.

Якщо авторизувався користувач:

- перегляд своїх завдань;
- перегляд своєї статистики;
- зміна статусу завдання.

Як було зазначено вище з міркувань зручності та раціональності для розробки мобільного застосунку системи керування проектами, було обрано під реалізацію лише функціонал ролі користувача. Реалізація тільки цього функціоналу дозволяє створити більш простий і інтуїтивно зрозумілий інтерфейс, орієнтований на зручність користувачів. Таким чином, користувачі можуть швидко освоїтися з додатком і почати ефективно працювати з проектами без зайвих ускладнень.

2.4 Реалізація мобільного застосунку управління проектами

Volley – це популярна бібліотека для Android, розроблена для спрощення роботи з мережевими запитами [23]. Вона забезпечує ефективну та асинхронну обробку HTTP-запитів, дозволяючи виконувати їх у фоновому режимі, що покращує продуктивність і реактивність додатків. Volley підтримує роботу з різними типами запитів, включаючи GET, POST, PUT, DELETE, і дозволяє легко обробляти відповіді у форматах JSON, XML, а також звичайних рядках [24].

Ключовими особливостями Volley є зручна обробка кешування та автоматичне повторення запитів у разі виникнення помилок. Це дозволяє

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		38

зменшити затримки і забезпечити більш стабільну роботу додатків у мережах з нестабільним з'єднанням. Volley також пропонує інтерфейс для додавання заголовків і параметрів до запитів, що полегшує роботу з API, які вимагають авторизації або додаткових даних.

Метод для виконання запиту на логін:

```
public void login(String url, JSONObject jsonRequest, final AuthCallback
callback) {
    JSONObjectRequest jsonObjectRequest = new
JsonObjectRequest(Request.Method.POST,
    url,
    jsonRequest,
    response -> {
        try {
            callback.onSuccess(response);
        } catch (JSONException e) {
            throw new RuntimeException(e);
        }
    },
    error -> {
        String errorMessage = error.getMessage();
        if(errorMessage!=null) {
            Log.e("Volley Error", errorMessage);
        } else Log.d("Volley Error", "");
        callback.onError(error);
    }
    );
    addToRequestQueue(jsonObjectRequest);
}
```

Метод для виконання POST-запиту на логін з передачею JSON-даних та обробкою результатів через callback.

Метод для GET-запиту, що повертає масив JSON-об'єктів:

```
public void makeGetArrayRequest(String url, final GetArrayCallback callback)
{
    StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
    response -> {
        try {
            JSONArray jsonArrayResponse = new JSONArray(response);
            callback.onSuccess(jsonArrayResponse);
        } catch (JSONException e) {
            e.printStackTrace();
        }
    },
    error -> callback.onError(error)) {
@Override
public Map<String, String> getHeaders() throws AuthFailureError {
    Map<String, String> headers = new HashMap<>();
    String token = tokenStorageService.getToken();
    headers.put("Authorization", "Bearer " + token);
    return headers;
}
```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

```

@Override
protected Response<String> parseNetworkResponse (NetworkResponse
response) {
    String parsed;
    try {
        parsed = new String(response.data, "UTF-8");
    } catch (UnsupportedEncodingException e) {
        parsed = new String(response.data);
    }
    return Response.success(parsed,
HttpHeaderParser.parseCacheHeaders(response));
}

};

requestQueue.add(stringRequest);
}

```

Метод для виконання PUT-запиту з JSON-об'єктом

```

public void makePUTObjectRequest(String url, JSONObject jsonObject,
GetArrayCallback callback) {
    JSONObjectRequest request = new JSONObjectRequest(Request.Method.PUT,
url, jsonObject, new Response.Listener<JSONObject>() {
@Override
public void onResponse(JSONObject response) {
    JSONArray jsonArray = new JSONArray();
    jsonArray.put(response);
    try {
        callback.onSuccess(jsonArray);
    } catch (JSONException e) {
        throw new RuntimeException(e);
    }
}
}, error -> callback.onError(error)) {
@Override
public Map<String, String> getHeaders() throws AuthFailureError {
    Map<String, String> headers = new HashMap<>();
    String token = tokenStorageService.getToken();
    headers.put("Authorization", "Bearer " + token);
    return headers;
}
});
requestQueue.add(request);
}
}

```

Метод для виконання PUT-запиту з JSON-об'єктом. Результат обробляється як масив JSON-об'єктів через callback та використовує токен для авторизації.

Клас VolleyService відповідає за здійснення мережевих запитів за допомогою бібліотеки Volley. Кожен метод цього класу представляє собою певний тип HTTP-запиту (POST, GET, PUT) і має свою унікальну реалізацію [25]. Ось докладний опис кожного запиту.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		40

Метод login виконує POST-запит для аутентифікації користувача. Він приймає URL-адресу, JSON-об'єкт із даними для запиту та callback, який викликається після завершення запиту. Запит створюється за допомогою JsonObjectRequest, що відправляє дані на сервер і очікує JSON-відповідь. У разі успіху викликається метод onSuccess callback'a, а в разі помилки — onError.

Метод register також здійснює POST-запит для реєстрації користувача. Він схожий на метод login, але тут дані користувача відправляються на інший URL-адресу. Після успішного отримання відповіді серверу, викликається метод onSuccess callback'a, а у випадку помилки — onError.

Метод makeGetArrayRequest здійснює GET-запит для отримання масиву JSON-даних. Цей запит використовує StringRequest, який очікує отримати рядкову відповідь. У разі успіху рядкова відповідь перетворюється на JSONArray, і викликається метод onSuccess callback'a. Якщо запит завершується помилкою, викликається метод onError. Перед відправленням запиту додаються заголовки, які включають токен аутентифікації, отриманий від tokenStorageService.

Метод makeGetStringRequest виконує GET-запит для отримання рядкових даних з сервера. Він також використовує StringRequest. У разі успіху відповідь передається у метод onSuccess callback'a. У разі помилки викликається метод onError. Заголовки запиту також містять токен аутентифікації.

Метод makePUTObjectRequest здійснює PUT-запит для оновлення даних на сервері. Цей запит використовує JsonObjectRequest для відправлення JSON-об'єкта і очікування JSON-відповіді. Після успішного завершення запиту, відповідь обгортається у JSONArray і передається у метод onSuccess callback'a. У разі помилки викликається метод onError. Як і у попередніх методах, до запиту додаються заголовки з токеном аутентифікації.

Кожен з цих методів має єдину структуру: створює запит відповідного типу, додає його у чергу запитів та обробляє успішні відповіді і помилки за допомогою відповідних callback'ів.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

Функції в класі UserStats призначені для створення та налаштування різних типів графіків для відображення статистичних даних користувача. Використовується бібліотека MPAndroidChart для створення графіків.

MPAndroidChart – це потужна бібліотека для створення графіків на Android. Вона підтримує різні типи графіків, такі як лінійні графіки, гістограми, кругові діаграми тощо [26]. Бібліотека надає можливості для налаштування графіків, включаючи стилізацію, анімацію та інтерактивність.

Функція setupLineChart налаштовує лінійний графік для відображення кількості завдань за місяцями.

```
private void setupLineChart() {
    tasksPerMonthChart.getDescription().setEnabled(false);
    tasksPerMonthChart.setDrawGridBackground(false);
    tasksPerMonthChart.setTouchEnabled(true);
    tasksPerMonthChart.setDragEnabled(true);
    tasksPerMonthChart.setScaleEnabled(true);
    tasksPerMonthChart.setPinchZoom(true);

    XAxis xAxis = tasksPerMonthChart.getXAxis();
    xAxis.setPosition(XAxis.XAxisPosition.BOTTOM);
    xAxis.setGranularity(1f);
    xAxis.setLabelCount(6, true);
    xAxis.setDrawGridLines(false);

    YAxis leftAxis = tasksPerMonthChart.getAxisLeft();
    leftAxis.setDrawGridLines(false);
    leftAxis.setAxisMinimum(0f);
    leftAxis.setGranularity(1f);
    leftAxis.setValueFormatter(new ValueFormatter() {
        @Override
        public String getFormattedValue(float value) {
            return String.valueOf((int) value);
        }
    });
    tasksPerMonthChart.getAxisRight().setEnabled(false);

    tasksPerMonthChart.getRendererLeftYAxis().getPaintGrid().setColor(Color.BLACK);

    tasksPerMonthChart.getRendererLeftYAxis().getPaintAxisLine().setColor(Color.BLACK);
    tasksPerMonthChart.setScaleYEnabled(true);
    tasksPerMonthChart.setScaleXEnabled(true);
    tasksPerMonthChart.invalidate();
}
```

Ця функція налаштовує лінійну діаграму для відображення даних. Вона вимикає непотрібні елементи, такі як опис і фонові сітка, а також активує можливості взаємодії користувача з діаграмою, такі як торкання, перетягування, масштабування та збільшення/зменшення. Вісь X розташовується внизу,

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

відображає фіксовану кількість міток і не показує сітку. Ліва вісь Y налаштовується для відображення цілочисельних значень, а права вісь вимикається. Додатково налаштовуються кольори осей, і в кінці діаграма оновлюється з новими параметрами.

Висновки до розділу 2

Основні кроки у створенні мобільного додатку для управління проєктами включають вивчення різних сценаріїв його використання з урахуванням вимог. На підставі цього аналізу було розроблено архітектуру платформи, що передбачає моделювання структури проєкту та взаємодію між його компонентами. Створено базу даних для зберігання інформації про користувачів, проєкти та завдання. Детально описано алгоритм роботи з додатком та послідовність його використання. Функціональні можливості додатку охоплюють створення та управління проєктами, призначення завдань, формування звітів, надсилання повідомлень учасникам команди, додавання нотаток і коментарів, а також керування дедлайнами та пріоритетами завдань..

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ МОБІЛЬНОГО ЗАСТОСУНКУ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ

3.1 Порядок встановлення та налаштування параметрів системи

Для успішного розгортання застосунку необхідно виконати кілька ключових кроків, кожен з яких має важливе значення для забезпечення стабільної та безперебійної роботи системи. Почнемо з налаштування серверної частини, на якій працюватиме бекенд застосунку. Це включає підготовку серверного середовища, що охоплює встановлення необхідного програмного забезпечення, налаштування конфігураційних файлів та запуск відповідних служб. Важливо забезпечити стабільне та безперебійне з'єднання з базою даних, оскільки це є критичною умовою для збереження та отримання даних застосунку. Сервер повинен бути налаштований таким чином, щоб він міг ефективно обробляти запити від мобільного застосунку, забезпечуючи швидку та надійну роботу системи, з мінімальними затримками і високим рівнем безпеки.

Після успішного налаштування та запуску серверної частини, наступним кроком є встановлення застосунку на мобільний пристрій користувача. Це процес, що включає кілька важливих етапів. Спершу потрібно завантажити APK файл, який містить всі необхідні компоненти та ресурси застосунку [27]. Після завантаження файлу на мобільний пристрій, користувач повинен внести зміни у налаштування свого телефону, дозволяючи встановлення додатків з невідомих джерел. Це важливо, оскільки за замовчуванням більшість пристроїв блокують встановлення програм, які не були завантажені з офіційного магазину додатків.

Після того як відповідні налаштування були змінені, користувач може перейти до безпосередньої інсталяції APK файлу. Після успішної інсталяції

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

застосунок буде готовий до використання. Користувач зможе запускати застосунок, який автоматично встановить з'єднання з сервером для обміну даними.

3.2 Структура інтерфейсу мобільного застосунку системи управління проєктами

Інтерфейс мобільного застосунку системи управління проєктами спроектований з урахуванням потреб користувачів, щоб забезпечити максимальну зручність та ефективність використання. Кожен елемент інтерфейсу ретельно продуманий, щоб спростити навігацію та забезпечити швидкий доступ до всіх необхідних функцій. Застосунок розроблено таким чином, щоб користувачі могли легко орієнтуватися, навіть якщо вони вперше використовують цей інструмент. Це досягається завдяки інтуїтивному дизайну, зрозумілим іконкам та логічній структурі меню.

Основні компоненти інтерфейсу включають різні екрани та елементи, які виконують специфічні функції. Наприклад, головний екран служить центральним хабом, з якого користувач може отримати доступ до всіх інших розділів застосунку. Тут розміщені панель навігації і статистика. Панель навігації дозволяє швидко перемикатися між екранами, такими як «Завдання».

Модульна структура системи зображена на рисунку 3.1

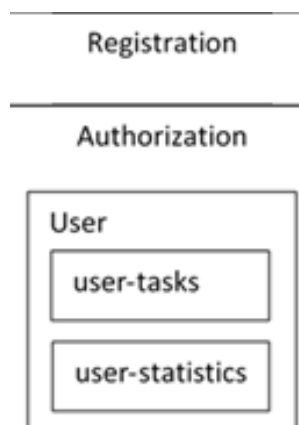


Рисунок 3.1 – Модульна структура системи

Модульна структура мобільного застосунку системи управління проєктами представлена на діаграмі, яка складається з трьох основних модулів: Registration, Authorization, User.

Authorization – цей модуль відповідає за процес аутентифікації користувачів. Він забезпечує захищений вхід в систему, перевіряючи облікові дані користувачів (логін та пароль) перед наданням доступу до основних функцій додатку.

User – основний модуль, який включає в себе підмодулі для роботи користувачів із завданнями та статистикою. Після успішної авторизації користувач переходить до цього модуля, який забезпечує доступ до основних функцій системи управління проєктами.

User-tasks – підмодуль, що відповідає за управління завданнями користувача. Він дозволяє створювати, редагувати, видаляти та переглядати завдання. Також в цьому підмодулі можуть бути реалізовані функції для призначення завдань іншим членам команди, встановлення пріоритетів і дедлайнів, а також відстеження виконання завдань.

User-statistics – підмодуль, що забезпечує користувача доступом до статистичних даних про його діяльність. Він надає інформацію про виконання завдань, прогрес проєктів, особисту продуктивність та інші аналітичні дані, які допомагають оцінити ефективність роботи користувача.

Ця модульна структура забезпечує чітке розділення функціональності додатку, що сприяє його зручності у використанні, легкості у підтримці та можливості подальшого розширення. Кожен модуль має свої чітко визначені задачі і взаємодіє з іншими модулями через добре визначені інтерфейси.

Екран activity_login містить стартову форму проєкту, через яку користувач може увійти в програму, ввівши свій логін і пароль. Якщо користувач не зареєстрований, він може зареєструватися за допомогою форми на екрані activity_register

Екран activity_login зображений на рисунку 3.2.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

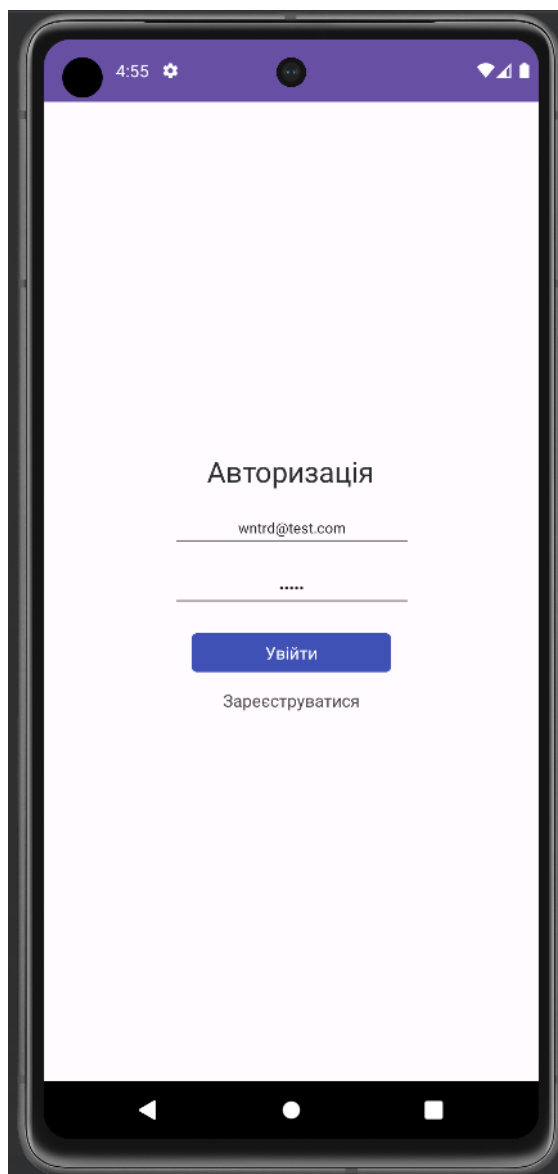


Рисунок 3.2 – Вигляд екрана activity_login

Екран activity_register зображений на рисунку 3.3.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

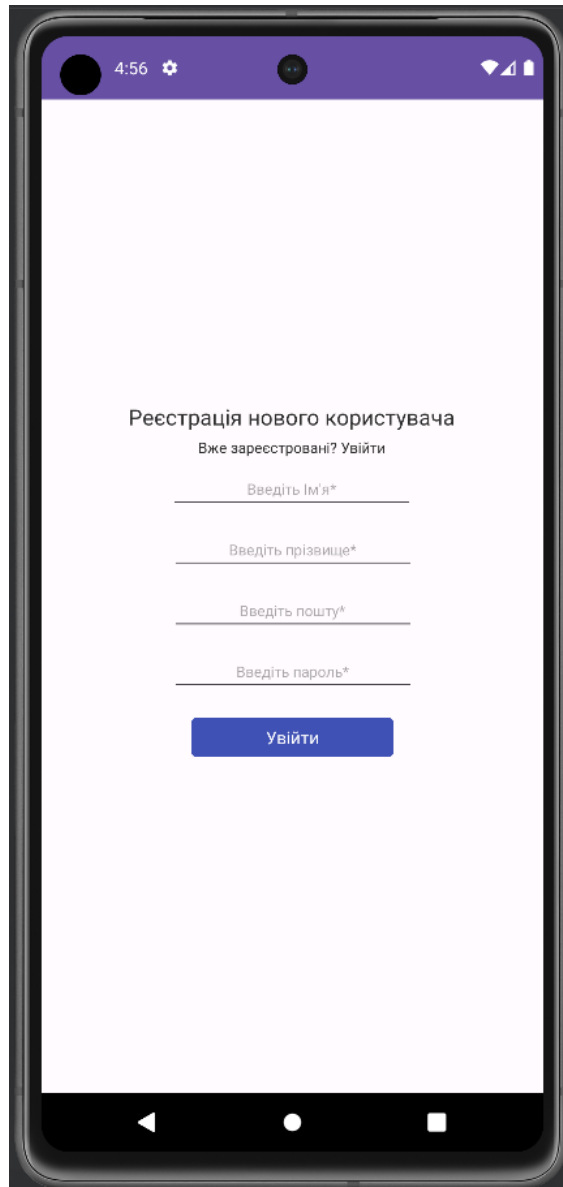


Рисунок 3.3 – Вигляд екрана activity_register

На екрані статистики користувач може детально ознайомитися з своїм навантаженням та успішністю. Тут представлена вся необхідна статистична інформація про завдання, включаючи типи завдань, стадії виконання, кількість виконаних завдань по місяцях, та інші важливі деталі.

Екран activity_user_stats зображений на рисунках 3.4, 3.5.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

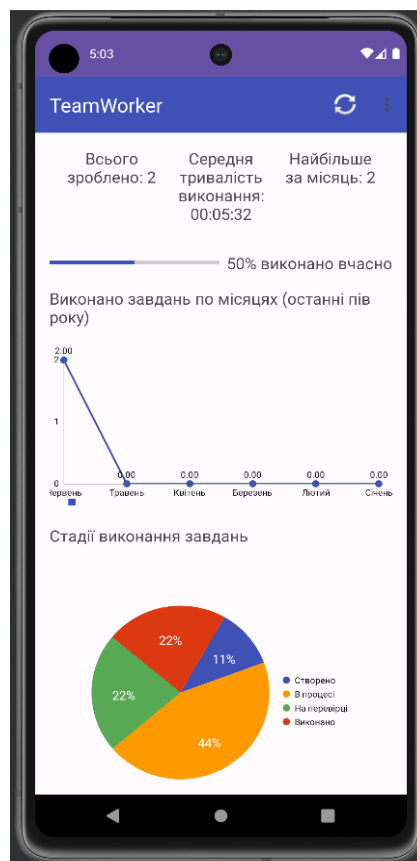


Рисунок 3.4 – Вигляд першої половини екрану activity_user_stats

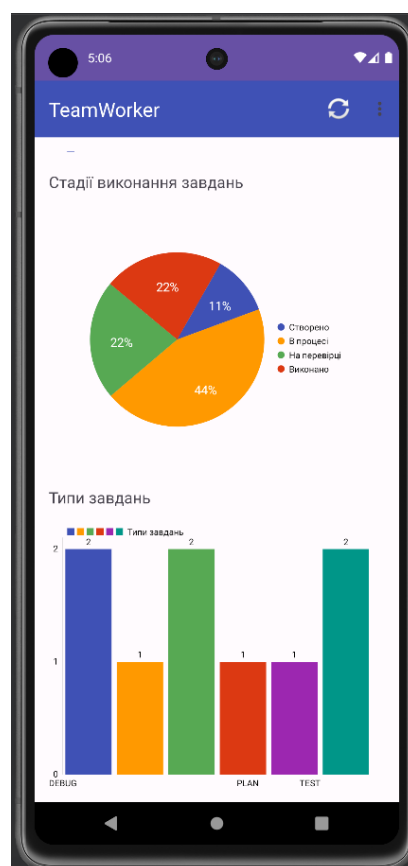


Рисунок 3.5 – Вигляд другої частини екрану activity_user_stats.

При взаємодії з секціями кругової діаграми або стовпцями стовпчатої діаграми отримуємо впливаюче вікно, яке вказує, що за дані користувач переглядає (рис. 3.6).

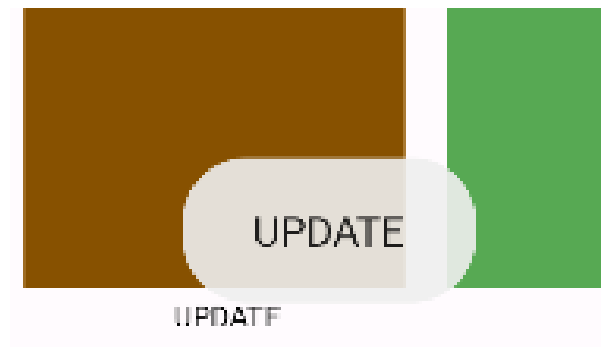


Рисунок 3.6 – Натиснуто на стовпець.

Екран завдання надає інформацію про завдання користувача. Тут користувач може переглянути опис завдання, вимоги, критерії успіху та примітки. Також зазначені учасники, які відповідають за виконання завдання, поточний статус та крайній термін виконання. Це дозволяє користувачам тримати всі завдання під контролем і своєчасно виконувати їх.

Екран activity_user_tasks зображений на рисунку 3.7.



Рисунок 3.7 – Вигляд екрану activity_user_tasks.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк. 50
Змн.	Арк.	№ докум.	Підпис	Дата		

Щоб отримати детальну інформацію про завдання, користувачу потрібно натиснути кнопку з іконкою знака питання. Деталі відображаються на діалоговому вікні dialog_task_info.

Діалогове вікно dialog_task_info зображений на рисунку 3.8.

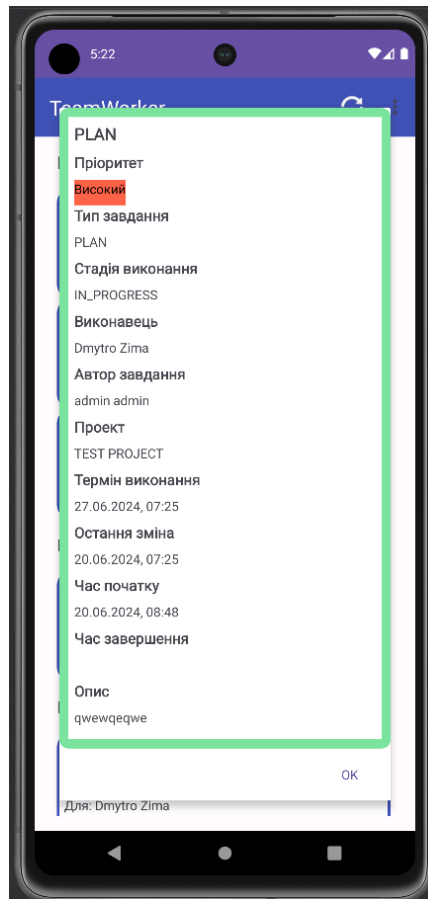


Рисунок 3.8 – Вигляд діалогового вікна dialog_task_info.

За зміну статусу відповідає діалогове вікно dialog_change_status.

Діалогове вікно dialog_change_status зображений на рисунку 3.9.

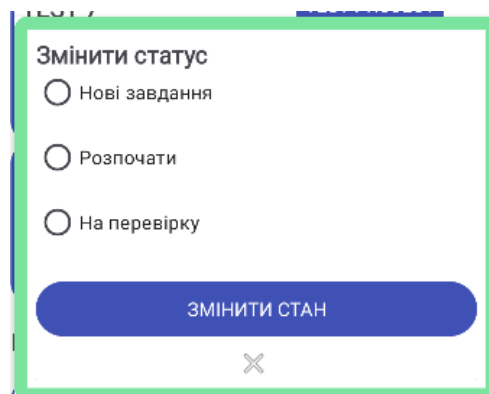


Рисунок 3.9 – Вигляд діалогового вікна dialog_task_info

Ця структура забезпечує користувачам інтуїтивне і зрозуміле використання мобільного застосунку для ефективного управління проектами, що сприяє підвищенню продуктивності та організованості в роботі.

3.3 Тестування роботи програмного комплексу

Тестування є критично важливим етапом при розробці додатків на Android, оскільки воно забезпечує високу якість кінцевого продукту та задовольняє очікування користувачів [28]. Ретельне тестування допомагає виявити та виправити помилки на ранніх стадіях розробки, що значно зменшує витрати на їх виправлення на пізніших етапах. Це також гарантує, що додаток працює стабільно та безпечно в різних умовах і на різних пристроях, що є важливим для забезпечення позитивного користувацького досвіду.

Тестування продуктивності є ще одним важливим аспектом, який не можна ігнорувати [29]. Воно дозволяє оцінити, як додаток поводить себе під навантаженням, перевірити його швидкість відгуку та ефективність використання ресурсів. Важливо, щоб додаток не тільки правильно функціонував, але й працював швидко та без збоїв, навіть при високих навантаженнях. Це забезпечує користувачам приємний досвід використання і зменшує ймовірність видалення додатка через його низьку продуктивність.

В цілому, тестування є невід'ємною частиною процесу розробки Android-додатків [30]. Воно допомагає створювати надійні, безпечні та зручні у використанні додатки, що відповідають високим стандартам якості. Інвестування часу та ресурсів у тестування окупається завдяки зменшенню кількості помилок, підвищенню задоволеності користувачів та збереженню репутації розробника.

Таблиця тестування роботи сайту наведено у таблиці 3.1.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		52

Таблиця 3.1

Тестування роботи сайту

Назва тестування	Кроки тестування	Тестові дані	Очікуваний результат	Фактичний результат	Pass / Fail
1.	2.	3.	4.	5.	6.
Тестування входу користувача	1.Відкрити сторінку входу 2.Ввести правильний логін 3.Ввести правильний пароль 4.Натиснути кнопку "Увійти"	Логін: wntrd@mail.com Пароль:wntrd	Користувач успішно входить в систему і перенаправляється на головну сторінку	Як і очікувалося	Pass
Тестування реєстрації	1.Відкрити сторінку реєстрації 2.Ввести дані нового користувача. 3.Натиснути кнопку "Зареєструватися"	Логін: ryzhko@mail.com Пароль: ryzko	Користувач успішно реєструється і перенаправляється на сторінку підтвердження	Як і очікувалося	Pass
Тестування роботи меню	1.Відкрити меню 2.Натиснути кнопку "Завдання"	Немає тестових даних.	Відбувається перехід на екран завдань	Як і очікувалося	Pass
Тестування відображення деталей завдання	1.Відкрити сторінку "Завдання" 2.Натиснути кнопку "Деталі"	Немає тестових даних.	Завдання успішно відображається в діалоговому вікні	Як і очікувалося	Pass
Тестування виходу з системи	1.Натиснути кнопку "Вийти"		Користувач успішно виходить з системи	Як і очікувалося	Pass
Тестування зміни статусу	1.Відкрити сторінку "Завдання" 2. Натиснути кнопку зміни статусу 3. Обрати статус	Новий статус.	Статус успішно змінено, сторінка оновилась	Як і очікувалося	Pass

Тестування кнопки оновлення сторінки	1.Додати завдання через веб версію. 3.Натиснути кнопку оновлення	Нове завдання	Сторінка оновлюється, завдання з'являється в категорії "Нові завдання"	Як і очікувалося	Pass
--------------------------------------	---	---------------	--	------------------	------

Висновки до розділу 3

Розробка мобільного застосунку для системи управління проєктами на платформі Android пройшла успішно, завдяки чіткій структурі процесу та дотриманню найкращих практик програмування. У процесі розробки була врахована низка важливих аспектів, що забезпечило високу якість кінцевого продукту. В першу чергу, велика увага приділялася створенню інтуїтивного та зручного інтерфейсу, який спрощує навігацію та забезпечує користувачам легкий доступ до всіх функцій системи управління проєктами.

Ключовим моментом стало забезпечення сумісності застосунку з різними пристроями та версіями операційної системи Android. Це було досягнуто завдяки ретельному плануванню та впровадженню адаптивного дизайну, що дозволяє додатку коректно працювати на широкому спектрі пристроїв. Крім того, застосунок був оптимізований для ефективного використання ресурсів, що забезпечує швидкодію та стабільну роботу навіть при високих навантаженнях.

Особливу увагу було приділено забезпеченню безпеки даних користувачів. Впроваджені механізми аутентифікації та авторизації гарантують захист особистої інформації та даних проєктів, що є критично важливим для користувачів, які довіряють системі управління проєктами свої конфіденційні дані.

Важливим етапом розробки було тестування застосунку, яке дозволило виявити та виправити помилки на ранніх стадіях, що значно підвищило якість кінцевого продукту. Використання автоматизованих тестів допомогло

забезпечити стабільність і надійність додатку при внесенні змін до коду, що скоротило час на тестування та виправлення помилок.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи бакалавра було розроблено мобільний застосунок для системи керування проєктами на платформі Android. Головною метою цього проєкту було створення інтуїтивно зрозумілого та функціонального інструменту, який би забезпечував ефективне управління проєктами та завданнями для користувачів.

Розробка застосунку включала кілька ключових етапів, кожен з яких сприяв досягненню високої якості кінцевого продукту. Спочатку був проведений аналіз вимог користувачів та визначено основні функції, які повинні бути реалізовані у застосунку. На основі цього аналізу було створено детальний проєкт інтерфейсу, що забезпечує зручну та зрозумілу навігацію.

Реалізація функціональності застосунку здійснювалася з використанням сучасних технологій та методологій розробки програмного забезпечення. Особлива увага приділялася адаптивності дизайну, щоб забезпечити коректну роботу застосунку на різних пристроях з різними розмірами екранів і версіями операційної системи Android. Було впроваджено механізми аутентифікації та авторизації, що гарантують безпеку даних користувачів.

Розроблений мобільний застосунок для системи керування проєктами відповідає всім вимогам, що були поставлені на початку роботи. Він є зручним у використанні, надійним та функціональним, що дозволяє користувачам ефективно керувати своїми проєктами та завданнями, підвищуючи їх продуктивність та організованість. Результати роботи демонструють успішне застосування теоретичних знань та практичних навичок, отриманих під час навчання, і можуть бути використані як основа для подальшого розвитку та вдосконалення системи управління проєктами.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		56

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Рижко І. Р., Савіцький Р. С. Використання серверу Spring Boot при розробці Android додатків. Подано до друку, 13-17 травня 2024 р. Житомир.
2. Airtable. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.airtable.com/>
3. Asana. [Електронний ресурс] – Режим доступу до ресурсу: <https://asana.com/>
4. Jira. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.atlassian.com/software/jira>
5. Документація Java. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.java.com/>
6. Документація Spring Boot. [Електронний ресурс] – Режим доступу до ресурсу: <https://spring.io/projects/spring-boot>
7. Документація Spring Security. [Електронний ресурс] – Режим доступу до ресурсу: <https://spring.io/projects/spring-security>
8. Документація Spring MVC. [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html>
9. Що таке RESTful API?. [Електронний ресурс] – Режим доступу до ресурсу: <https://devzone.org.ua/post/shcho-take-restful-api>
10. Документація MySQL. [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.mysql.com/doc/>
11. Android Studio: переваги та особливості. [Електронний ресурс] – Режим доступу до ресурсу: <https://qagroup.com.ua/publications/android-studio-perevagy-ta-osoblyvosti/>
12. Документація JVM. [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.oracle.com/javase/specs/jvms/se8/html/>
13. What are the Benefits of Java? [Електронний ресурс] – Режим доступу до ресурсу: <https://www.developer.com/java/java-benefits/>

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		

14. Kotlin vs. Java For Android. [Електронний ресурс] – Режим доступу до ресурсу: <https://lampa.dev/blog/kotlin-vs-java-for-android-development-ultimate-comparison>
15. Основи розробки баз даних. [Електронний ресурс] – Режим доступу до ресурсу: <https://support.microsoft.com/uk-ua/topic/%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B8-%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B8-%D0%B1%D0%B0%D0%B7-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5>
16. MySQL Workbench [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mysql.com/products/workbench/>
17. Зв'язки в базах даних: що це і як вони працюють [Електронний ресурс] – Режим доступу до ресурсу: <https://foxminded.ua/zviazok-u-bazi-danykh/>
18. Що таке управління проектами: як здійснюється і що в себе включає? [Електронний ресурс] – Режим доступу до ресурсу: <https://blog.agrokebety.com/shcho-take-upravlinnya-proektamy>
19. What is Use Case Diagram? [Електронний ресурс] – Режим доступу до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>
20. Що таке об'єктно-орієнтоване програмування: принципи, переваги та недоліки. [Електронний ресурс] – Режим доступу до ресурсу: <https://goit.global/ua/articles/shcho-take-ob-iektno-orientovane-prohramuvannia-pryntsypy-perevahy-ta-nedoliky/>
21. Відображення моделі «сутність-зв'язок» на базу даних. [Електронний ресурс] – Режим доступу до ресурсу: <https://ua5.org/database/2003-vidobrazhennya-modeli-sutnist-zvyazok-na-bazu-danyh.html>
22. Реляційні бази даних усе, що необхідно про них знати. [Електронний ресурс] – Режим доступу до ресурсу: <https://foxminded.ua/reliatsiini-bazy-danykh/>

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		

- 23.Репозиторій бібліотеки Volley. [Електронний ресурс] – Режим доступу до ресурсу: <https://google.github.io/volley/>
- 24.Документація JSON. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.json.org/json-uk.html>
- 25.HTTTP Методи запиту. [Електронний ресурс] – Режим доступу до ресурсу: https://w3schoolsua.github.io/tags/ref_httpmethods.html#gsc.tab=0
- 26.Репозиторій бібліотеки MPAndroidChart. [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/PhilJay/MPAndroidChart>
- 27.What Is an APK File? [Електронний ресурс] – Режим доступу до ресурсу: <https://www.lifewire.com/apk-file-4152929>
- 28.Fundamentals of testing Android apps [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.android.com/training/testing/fundamentals>
- 29.ТЕСТУВАННЯ ПРОДУКТИВНОСТІ. [Електронний ресурс] – Режим доступу до ресурсу: <https://training.qatestlab.com/blog/technical-articles/performance-testing/>
- 30.Чому тестування необхідне? [Електронний ресурс] – Режим доступу до ресурсу: [https://qalight.ua/baza-znaniy/chomu-testuvannya-neobhidne/#:~:text=%D0%A1%D0%B0%D0%BC%D0%B5%20%D1%82%D0%BE%D0%BC%D1%83%20%D0%B1%D1%83%D0%B4%D1%8C%2D%D1%8F%D0%BA%D0%B8%D0%B9%20%D0%BF%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82,Software\).](https://qalight.ua/baza-znaniy/chomu-testuvannya-neobhidne/#:~:text=%D0%A1%D0%B0%D0%BC%D0%B5%20%D1%82%D0%BE%D0%BC%D1%83%20%D0%B1%D1%83%D0%B4%D1%8C%2D%D1%8F%D0%BA%D0%B8%D0%B9%20%D0%BF%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82,Software).)

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		

Реалізація запитів на сервер

Лістинг VolleyService.java

```

package services;

import android.content.Context;
import android.util.Log;

import com.android.volley.AuthFailureError;
import com.android.volley.NetworkResponse;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.HttpHeaderParser;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.UnsupportedEncodingException;
import java.util.HashMap;
import java.util.Map;
import callback.AuthCallback;
import callback.GetArrayCallback;
import callback.GetStringCallback;

public class VolleyService {
    private RequestQueue requestQueue;
    private Context context;
    private TokenStorageService tokenStorageService;

    public VolleyService(Context context) {
        this.context = context;
        requestQueue = getRequestQueue();
        tokenStorageService = new TokenStorageService(context);
    }

    private RequestQueue getRequestQueue() {
        if (requestQueue == null) {
            requestQueue =
Volley.newRequestQueue(context.getApplicationContext());
        }
        return requestQueue;
    }

    private <T> void addToRequestQueue(Request<T> req) {
        getRequestQueue().add(req);
    }

    public void login(String url, JSONObject jsonRequest, final AuthCallback
callback) {
        JsonObjectRequest jsonObjectRequest = new
JsonObjectRequest(Request.Method.POST,
            url,
            jsonRequest,
            response -> {
                try {
                    callback.onSuccess(response);
                } catch (JSONException e) {

```

```

        throw new RuntimeException(e);
    }
},
error -> {
    String errorMessage = error.getMessage();
    if(errorMessage!=null) {
        Log.e("Volley Error", errorMessage);
    } else Log.d("Volley Error", "");
    callback.onError(error);
}
);
addToRequestQueue(jsonObjectRequest);
}

public void register(String url, JSONObject data, AuthCallback callback) {

    JsonObjectRequest request = new JsonObjectRequest(Request.Method.POST,
url, data,
        new Response.Listener<JSONObject>() {
            @Override
            public void onResponse(JSONObject response) {

                try {
                    callback.onSuccess(response);
                } catch (JSONException e) {
                    throw new RuntimeException(e);
                }
            }
        },
        new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {

                callback.onError(error);
            }
        }
    ));

    getRequestQueue().add(request);
}

public void makeGetArrayRequest(String url, final GetArrayCallback
callback) {
    StringRequest stringRequest = new StringRequest(Request.Method.GET,
url,
        response -> {
            try {
                JSONArray jsonArrayResponse = new JSONArray(response);
                callback.onSuccess(jsonArrayResponse);
            } catch (JSONException e) {
                e.printStackTrace();
            }
        },
        error -> callback.onError(error)) {
        @Override
        public Map<String, String> getHeaders() throws AuthFailureError {
            Map<String, String> headers = new HashMap<>();
            String token = tokenStorageService.getToken();
            headers.put("Authorization", "Bearer " + token);
            return headers;
        }
        @Override
        protected Response<String> parseNetworkResponse(NetworkResponse
response) {
            String parsed;

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        try {
            parsed = new String(response.data, "UTF-8");
        } catch (UnsupportedEncodingException e) {
            parsed = new String(response.data);
        }
        return Response.success(parsed,
            HttpHeaders.parseCacheHeaders(response));
    }

    };

    requestQueue.add(stringRequest);
}

public void makeGetStringRequest(String url, final GetStringCallback
callback) {
    StringRequest stringRequest = new StringRequest(Request.Method.GET,
url,
        response -> {
            try {
                callback.onSuccess(response);
            } catch (JSONException e) {
                e.printStackTrace();
            }
        },
        callback::onError) {
        @Override
        public Map<String, String> getHeaders() throws AuthFailureError {
            Map<String, String> headers = new HashMap<>();
            String token = tokenStorageService.getToken();
            headers.put("Authorization", "Bearer " + token);
            return headers;
        }
    };
    requestQueue.add(stringRequest);
}

public void makePUTObjectRequest(String url, JSONObject jsonObject,
GetArrayCallback callback) {
    JSONObjectRequest request = new JSONObjectRequest(Request.Method.PUT,
url, jsonObject, new Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject response) {
            JSONArray jsonArray = new JSONArray();
            jsonArray.put(response);
            try {
                callback.onSuccess(jsonArray);
            } catch (JSONException e) {
                throw new RuntimeException(e);
            }
        }
    }, error -> callback.onError(error)) {
        @Override
        public Map<String, String> getHeaders() throws AuthFailureError {
            Map<String, String> headers = new HashMap<>();
            String token = tokenStorageService.getToken();
            headers.put("Authorization", "Bearer " + token);
            return headers;
        }
    };
    requestQueue.add(request);
}
}

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		

Сторінка логіну

Лістинг Login.java

```
package auth.login;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.Volley;
import com.example.teamworker.R;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import auth.register.Register;
import callback.AuthCallback;
import services.TokenStorageService;
import admin.AdminMain;
import services.VolleyService;
import manager.ManagerMain;
import model.Position;
import model.Project;
import model.Role;
import model.User;
import user.UserStats;

public class Login extends AppCompatActivity {

    private TokenStorageService tokenStorage;
    private EditText usernameEditText;
    private EditText passwordEditText;
    private Button loginButton;
    private TextView regLink;
    private VolleyService volleyService;
    private static final String AUTH_API =
"http://192.168.56.1:8080/api/v1/auth/";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        usernameEditText = findViewById(R.id.login_login);
        passwordEditText = findViewById(R.id.login_password);
```

```

loginButton = findViewById(R.id.login_loginButton);
regLink = findViewById(R.id.login_regLink);
volleyService = new VolleyService(this);
tokenStorage = new TokenStorageService(this);

regLink.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(Login.this, Register.class);
        startActivity(intent);
    }
});

loginButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String username = usernameEditText.getText().toString();
        String password = passwordEditText.getText().toString();

        JSONObject credentials = new JSONObject();
        try {
            credentials.put("username", username);
            credentials.put("password", password);
        } catch (JSONException e) {
            e.printStackTrace();
        }

        String loginUrl = AUTH_API + "login";

        volleyService.login(loginUrl, credentials, new AuthCallback()
        {
            @Override
            public void onSuccess(JSONObject response) throws
JSONException {
                try {
                    String token = response.getString("token");
                    String username = response.getString("username");
                    tokenStorage.saveToken(token);
                    String userUrl =
"http://192.168.56.1:8080/api/v1/users/get/" + username;
                    JSONObjectRequest userRequest = new
JsonObjectRequest(Request.Method.GET, userUrl, null,
                    new Response.Listener<JSONObject>() {
                        @Override
                        public void onResponse(JSONObject
userResponse) {
                            Log.d("RESPONSE",
userResponse.toString());

                            try {
                                User user = new User();

                                user.setId(userResponse.getInt("id"));

                                user.setUsername(userResponse.getString("username"));

                                user.setName(userResponse.getString("name"));

                                user.setSurname(userResponse.getString("surname"));

                                JSONArray positionsArray =
userResponse.getJSONArray("position");
                                List<Position> positions = new
ArrayList<>();

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        positionsArray.length(); i++) {
            = positionsArray.getJSONObject(i);

            Position position = new
            Position();
            position.setId(positionObject.getInt("id"));
            position.setName(positionObject.getString("name"));

            positionObject.getJSONObject("project");
            Project project = new
            Project();
            project.setId(projectObject.getInt("id"));
            project.setName(projectObject.getString("name"));
            project.setCreateTime(projectObject.getString("createTime"));
            project.setProjectStage(projectObject.getString("projectStage"));
            project.setProjectType(projectObject.getString("projectType"));

            position.setProject(project);

            positions.add(position);
        }
        user.setPosition(positions);

        JSONArray rolesArray =
        List<Role> roles = new

        for (int i = 0; i <
            JSONObject roleObject =

            Role role = new Role();

            roles.add(role);
        }
        user.setRoles(roles);

        Log.d("USER_OBJECT",
        user.toString());

        roles.get(0).getName();

        (userRole.equals("ROLE_ADMIN")) {

        if (roles.size() > 0) {
            String userRole =

            Intent intent;
            if

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		

```

                                intent = new
Intent(Login.this, AdminMain.class);
                                } else if
                                intent = new
                                } else {
                                intent = new
                                }

intent.putExtra("USER_OBJECT", user);
                                Toast.makeText(Login.this,
"Вітаємо, " + user.getName(), Toast.LENGTH_SHORT).show();
                                startActivity(intent);
                                } else {
                                Toast.makeText(Login.this,
"Невідома роль", Toast.LENGTH_SHORT).show();
                                }

                                } catch (JSONException e) {
                                throw new RuntimeException(e);
                                }
                                },
                                new Response.ErrorListener() {
                                @Override
                                public void
onErrorResponse(VolleyError error) {
                                Toast.makeText(Login.this,
"Помилка отримання даних користувача: " + error.getMessage(),
Toast.LENGTH_SHORT).show();
                                }
                                }) {
                                @Override
                                public Map<String, String> getHeaders() {
                                Map<String, String> headers = new
HashMap<>();
                                headers.put("Authorization", "Bearer " +
token);
                                return headers;
                                }
                                };

Volley.newRequestQueue(Login.this).add(userRequest);

                                } catch (JSONException e) {
                                e.printStackTrace();
                                Toast.makeText(Login.this, "Помилка сервера",
Toast.LENGTH_SHORT).show();
                                }
                                }

                                @Override
                                public void onError(VolleyError error) {
                                Toast.makeText(Login.this, "Невірний логін або
пароль", Toast.LENGTH_SHORT).show();
                                }
                                });
                                }
                                });
                                }
                                }

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		