Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка програмного забезпечення для тренувань «Gang Gym» з використанням мови Kotlin Multiplatform

Виконав: студент IV курсу, групи СП-42
спеціальності 121 – Інженерія програмного забезпечення
(шифр і назва спеціальності)

(підпис) Петрик В,В (прізвище та ініціали)

Керівник (підпис) Яворська Є.Б (прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю.М. (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М.Р. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль 2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ проф. Петрик М.Р.
(підпис) (прізвище та ініціали)
« 6 » квітня 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту _____

1. Тема роботи Розробка програмного забезпечення для тренувань «Gang Gym» з використанням мови Kotlin Multiplatform

Керівник роботи к.т.н., доцент Яворська Є.Б
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом по університету від « 06 » квітня 2026 року № 4/9 - 170 .

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи вимоги до розробки мобільного додатку, технологія Kotlin Multiplatform, сучасні засоби розробки.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області та постановка задачі. 2. Проєктування та розробка програмної системи. 3. Тестування та оцінка якості. 4. Безпека життєдіяльності, основи охорони праці. Висновки. Список використаних джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Існуючі технології реалізації подібних систем. 2. Діаграми варіантів використання системи. 3. Навігаційна структура додатку та загальна архітектура програмної системи. 4. UML-діаграма класів та діаграма послідовності додатку. 5. Результати тестування серверної і клієнтської частини та бази даних.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності	Гурик О.Я., доцент кафедри МТ		
основи охорони праці	Мариненко С.Ю., доцент кафедри МТ		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Ознайомлення з завданням до кваліфікаційної роботи		
2	Аналіз предметної області, огляд аналогів та формування вимог до додатку		
3	Проєктування архітектури Kotlin Multiplatform додатку, моделей даних та структури бази даних		
4	Розробка локального шару збереження даних, DAO, сутностей та репозиторіїв		
5	Реалізація бізнес-логіки для роботи з вправами, тренувальними програмами та сесіями тренувань		
6	Реалізація основних екранів додатку: Home, Exercise Catalog, Routine List, Workout List, Progress		
7	Тестування функціональності додатку, перевірка роботи на різних платформах та усунення помилок		
8	Написання розділу 4: «Охорона праці та безпека життєдіяльності»		
9	Оформлення висновків, списку використаних джерел та додатків		
10	Нормоконтроль, попередній захист та захист кваліфікаційної роботи		
11	Перевірка на плагіат		
12	Попередній захист кваліфікаційної роботи		
13	Захист кваліфікаційної роботи		

Студент _____ Петрик В.В
(підпис) (прізвище та ініціали)

Керівник роботи _____ Яворська Є.Б
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка кросплатформного застосунку для планування та моніторингу тренувального процесу // Кваліфікаційна робота освітнього рівня «бакалавр» // Петрик Владислав Віталійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СП-42 // Тернопіль, 2026 // С. 98, рис. – __, табл. – 0, кресл. – 0, додат. – 1, бібліогр. – 20.

Ключові слова: Kotlin Multiplatform; Compose Multiplatform; мобільний застосунок; тренувальний процес; фітнес; база даних; Ktor; Room; SQLite; PostgreSQL.

Кваліфікаційна робота присвячена дослідженню, проєктуванню та розробці кросплатформного застосунку GanG Gym для організації тренувального процесу. Актуальність роботи обумовлена зростанням потреби у зручних цифрових інструментах, які дають змогу користувачам планувати тренування, вести каталог вправ, створювати тренувальні програми, фіксувати результати занять і відстежувати власний прогрес.

У першому розділі проведено аналіз предметної області, визначено основні проблеми організації персональних тренувань та сформульовано вимоги до програмної системи.

У другому розділі спроєктовано архітектуру застосунку, структуру локальної та серверної частин, модель даних, а також обґрунтовано вибір технологічного стеку. Для реалізації клієнтської частини використано Kotlin Multiplatform і Compose Multiplatform, що забезпечує спільну бізнес-логіку та інтерфейс для Android, iOS і Desktop. Для збереження даних застосовано Room та SQLite, а серверну частину реалізовано з використанням Ktor, PostgreSQL, Flyway та механізмів автентифікації.

У третьому розділі описано реалізацію основних модулів системи: каталогу вправ, списку тренувань, шаблонів програм, календаря занять, профілю

користувача та перегляду прогресу. Окрему увагу приділено роботі з тренувальними сесіями, збереженню результатів виконання вправ і побудові зручного користувацького інтерфейсу.

У четвертому розділі розглянуто питання безпеки життєдіяльності та основ охорони праці під час роботи з комп'ютерною технікою та програмними системами.

Об'єкт дослідження — процес планування, виконання та аналізу фізичних тренувань користувача.

Предмет дослідження — методи, технології та програмні засоби розробки кросплатформних застосунків для ведення тренувального процесу з використанням Kotlin Multiplatform, Compose Multiplatform, Room, SQLite, Ktor і PostgreSQL.

ABSTRACT

Development of a cross-platform application for planning and monitoring the training process // Bachelor's Qualification Thesis // Petryk Vladyslav Vitaliiovych // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, Group SP-42 // Ternopil, 2026 // P. 98, Fig. – __, Tables – 0, Drawings – 0, Appendices – 1, References – 20.

Keywords: Kotlin Multiplatform; Compose Multiplatform; mobile application; training process; fitness; database; Ktor; Room; SQLite; PostgreSQL.

The qualification work is devoted to the research, design, and development of the cross-platform GanG Gym application for organizing the training process. The relevance of the work is determined by the growing need for convenient digital tools that allow users to plan workouts, maintain an exercise catalog, create training programs, record workout results, and monitor personal progress.

The first section analyzes the subject area, identifies the main problems of organizing personal training, and formulates the requirements for the software system.

The second section presents the application architecture, the structure of the local and server parts, the data model, and justifies the choice of the technology stack. Kotlin Multiplatform and Compose Multiplatform are used to implement the client side, providing shared business logic and user interface for Android, iOS, and Desktop. Room and SQLite are used for local data storage, while the server side is implemented using Ktor, PostgreSQL, Flyway, and authentication mechanisms.

The third section describes the implementation of the main system modules: exercise catalog, workout list, training program templates, workout calendar, user profile, and progress tracking. Special attention is paid to working with workout sessions, storing exercise results, and building a convenient user interface. The fourth section considers the issues of life safety and the basics of labor protection when working with computer equipment and software systems.

The object of the study is the process of planning, performing, and analyzing a user's physical workouts.

The subject of the study is the methods, technologies, and software tools for developing cross-platform applications for managing the training process using Kotlin Multiplatform, Compose Multiplatform, Room, SQLite, Ktor, and PostgreSQL.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – інтерфейс прикладного програмування (Application Programming Interface).

CRUD – базові операції роботи з даними: створення, читання, оновлення та видалення (Create, Read, Update, Delete).

DTO – об’єкт передавання даних між частинами програмної системи (Data Transfer Object).

HTTP – протокол передавання гіпертексту (HyperText Transfer Protocol).

JSON – текстовий формат обміну структурованими даними (JavaScript Object Notation).

JWT – стандарт токена для безпечного передавання інформації між клієнтом і сервером (JSON Web Token).

KMP – Kotlin Multiplatform, технологія для створення спільної кодової бази під різні платформи.

UI – користувацький інтерфейс (User Interface).

UX – користувацький досвід взаємодії з програмним продуктом (User Experience).

Room – бібліотека для роботи з локальною базою даних SQLite у Kotlin/Android-застосунках.

Ktor – фреймворк для створення серверних застосунків мовою Kotlin.

Koin – бібліотека для впровадження залежностей у Kotlin-застосунках.

Compose Multiplatform – декларативний фреймворк для створення кросплатформного користувацького інтерфейсу.

ЗМІСТ

ВСТУП

У сучасних умовах цифрові технології активно впроваджуються у сферу спорту, фітнесу та персонального здоров’я. Все більше користувачів прагнуть не лише виконувати фізичні вправи, а й системно планувати тренування, контролювати навантаження, зберігати історію занять та аналізувати власний прогрес. Традиційне ведення тренувального процесу у вигляді паперових записів або розрізнених нотаток є незручним, оскільки не забезпечує швидкого доступу до статистики, шаблонів тренувань та результатів виконання вправ.

Актуальність теми кваліфікаційної роботи полягає у необхідності створення зручного кросплатформного програмного засобу для організації тренувального процесу. Такий застосунок повинен забезпечувати користувачу можливість працювати з каталогом вправ, створювати тренування та тренувальні програми, планувати заняття в календарі, фіксувати результати виконання вправ і переглядати динаміку прогресу. Важливим аспектом є підтримка різних платформ, оскільки користувачі можуть взаємодіяти із системою як зі смартфона, так і з персонального комп’ютера.

Метою кваліфікаційної роботи є проєктування та розробка кросплатформного застосунку GanG Gym для планування, виконання та моніторингу тренувального процесу з використанням сучасних технологій Kotlin Multiplatform, Compose Multiplatform, локального збереження даних і серверної взаємодії.

Для досягнення поставленої мети було проаналізовано предметну область та визначено основні потреби користувачів фітнес-застосунків; сформульовано функціональні та нефункціональні вимоги до програмної системи; спроєктовано архітектуру клієнтської та серверної частин; реалізовано модулі каталогу вправ, тренувань, програм, календаря, профілю користувача та перегляду прогресу; організовано локальне збереження даних із використанням Room і SQLite; розроблено серверну частину на базі Ktor з підтримкою бази даних PostgreSQL; проведено тестування основних функцій програмного продукту.

Об’єктом дослідження є процес планування, виконання та аналізу фізичних тренувань користувача.

Предметом дослідження є методи, технології та програмні засоби створення кросплатформного застосунку для організації тренувального процесу з використанням Kotlin Multiplatform, Compose Multiplatform, Room, SQLite, Ktor і PostgreSQL.

Практичне значення одержаних результатів полягає у створенні програмного продукту, який може бути використаний для персонального ведення тренувань, формування власних програм занять, контролю виконаних вправ і перегляду спортивного прогресу. Завдяки кросплатформному підходу значна частина логіки застосунку є спільною для різних платформ, що спрощує подальший розвиток системи та зменшує витрати на підтримку окремих версій програмного забезпечення.

Розроблений застосунок GanG Gym поєднує зручний користувацький інтерфейс, структуровану модель даних та можливість розширення функціональності у майбутньому. Це дозволяє розглядати систему як основу для подальшого розвитку цифрового інструменту персонального фітнес-супроводу.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

У цьому розділі проведено аналіз предметної області, пов’язаної з організацією тренувального процесу за допомогою цифрових засобів. Розглянуто основні потреби користувачів фітнес-застосунків, визначено проблеми, які виникають під час ручного планування та обліку тренувань, а також сформульовано загальне призначення програмної системи GanG Gym. На основі аналізу предметної області буде визначено функціональні можливості майбутнього застосунку, основних користувачів системи та вимоги до її роботи.

1.1 Аналіз предметної області

У сучасному суспільстві фізична активність є важливою складовою здорового способу життя. Все більше людей займаються спортом самостійно або відвідують тренажерні зали, фітнес-студії та спортивні секції. Для досягнення стабільного результату користувачу недостатньо просто виконувати окремі вправи. Важливим є системний підхід: планування тренувань, вибір вправ відповідно до цілі, контроль навантаження, фіксація результатів і аналіз прогресу за певний період.

На практиці багато користувачів ведуть тренувальний процес у нотатках, таблицях або взагалі запам’ятовують результати без окремого обліку. Такий підхід має низку недоліків. Користувачу складно швидко знайти попередні результати, порівняти показники, оцінити динаміку розвитку або повторно використати вже створену програму тренувань. Крім того, ручне ведення записів не завжди зручне під час самого заняття, особливо коли потрібно фіксувати кількість підходів, повторень, робочу вагу, тривалість вправи або час відпочинку.

Предметна область фітнес-застосунків охоплює декілька основних напрямів: створення каталогу вправ, формування тренувань, побудову тренувальних програм, планування занять у календарі, ведення історії виконаних тренувань та перегляд прогресу. Кожен із цих напрямів є важливим для повноцінної організації

тренувального процесу. Каталог вправ дозволяє користувачу швидко обрати потрібну вправу за категорією, обладнанням, рівнем складності або цільовими м’язовими групами. Модуль тренувань дає змогу об’єднувати вправи в логічні комплекси. Календар допомагає планувати заняття, а модуль прогресу забезпечує аналіз виконаної роботи.

Існуючі фітнес-застосунки часто мають широкий функціонал, однак не завжди є зручними для користувача. Частина таких систем орієнтована на соціальні функції, платні тренувальні курси або складну аналітику, що може бути зайвим для користувача, який хоче просто створити власне тренування та відстежувати результати. Інші рішення працюють лише на одній платформі або не забезпечують достатньої гнучкості у створенні власних вправ, програм і тренувальних шаблонів. Саме тому доцільною є розробка кросплатформного застосунку, який поєднує базові інструменти планування, виконання та аналізу тренувань у єдиній системі.

Розроблювана програмна система GanG Gym призначена для користувачів, які хочуть самостійно організовувати тренувальний процес. Застосунок надає можливість переглядати каталог вправ, створювати власні вправи, формувати тренування та програми, запускати тренувальні сесії, зберігати результати виконання вправ і переглядати історію занять. Завдяки використанню Kotlin Multiplatform система може мати спільну логіку для різних платформ, зокрема Android, iOS та Desktop, що підвищує зручність використання і спрощує подальший розвиток програмного продукту.

Отже, аналіз предметної області показує, що актуальною є розробка зручного кросплатформного застосунку для планування та моніторингу тренувального процесу. Така система повинна забезпечувати структуроване збереження даних про вправи, тренування, програми та результати занять, а також надавати користувачу прості інструменти для контролю власного фізичного прогресу.

1.2 Постановка завдання та критерії успішності

Основним завданням кваліфікаційної роботи є розробка кросплатформного застосунку GanG Gym, призначеного для планування, виконання та моніторингу тренувального процесу. Програмна система повинна надати користувачу зручний інструмент для роботи з вправами, тренуваннями, тренувальними програмами, календарем занять, історією виконаних тренувань та показниками прогресу.

Під час розробки необхідно забезпечити можливість використання застосунку на кількох платформах. Для цього доцільно використати технологію Kotlin Multiplatform, яка дає змогу винести спільну бізнес-логіку, моделі даних, репозиторії та частину інтерфейсу в єдину кодову базу. Такий підхід зменшує дублювання коду, спрощує підтримку програмного продукту та дозволяє розвивати застосунок для Android, iOS і Desktop у межах одного проєкту.

Застосунок повинен містити каталог вправ із можливістю перегляду, пошуку, фільтрації, створення та редагування вправ. Для кожної вправи необхідно зберігати основну інформацію: назву, опис, категорію, обладнання, рівень складності, основні та додаткові м’язові групи, а також зображення або посилання на нього. Це дозволить користувачу швидко знаходити потрібні вправи та формувати на їх основі власні тренування.

Окремим завданням є реалізація модуля тренувань. Користувач повинен мати можливість створювати тренування з окремих блоків, додавати вправи, періоди відпочинку або готові тренувальні програми, вказувати кількість підходів, повторень, робочу вагу, тривалість виконання та час відпочинку. Після запуску тренування система повинна давати змогу фіксувати фактичні результати виконання вправ і зберігати їх для подальшого аналізу.

Також необхідно реалізувати модуль тренувальних програм, який дозволяє створювати багаторазові шаблони занять. Такі програми можуть використовуватися як основа для регулярних тренувань або як частина складніших тренувальних комплексів. Наявність цього модуля підвищує зручність застосунку,

оскільки користувачеві не потрібно щоразу створювати однакову структуру тренування з нуля.

Важливою частиною системи є календар тренувань. Він повинен відображати заплановані або виконані заняття, дозволяти користувачу переглядати тренування за конкретну дату та швидко переходити до детальної інформації про тренувальну сесію. Такий підхід робить процес контролю фізичної активності більш наочним і системним.

Для збереження даних необхідно реалізувати локальну базу даних із використанням Room та SQLite. Локальне збереження дозволить користувачу працювати із застосунком без постійного підключення до мережі. Додатково серверна частина, реалізована на базі Ktor, може забезпечувати централізоване збереження окремих даних, автентифікацію користувача та подальшу синхронізацію між пристроями.

Критеріями успішності розробки є працездатність основних модулів системи, коректне збереження та відображення даних, зручність користувацького інтерфейсу, стабільна робота застосунку на підтримуваних платформах, а також можливість подальшого розширення функціональності. Система повинна забезпечувати логічний і зрозумілий сценарій взаємодії: від створення вправи або вибору її з каталогу до виконання тренування і перегляду результатів.

Успішною реалізацію програмної системи можна вважати у випадку, якщо користувач має змогу створити власне тренування, додати до нього вправи, виконати тренувальну сесію, зберегти результати та переглянути прогрес за обраними вправами. Додатковими ознаками успішності є швидке завантаження екранів, відсутність критичних помилок під час роботи, збереження даних після перезапуску застосунку та відповідність інтерфейсу основним вимогам зручності й доступності.

1.3 Визначення акторів та варіантів використання

Для формування вимог до програмної системи необхідно визначити основних акторів, які взаємодіють із застосунком GanG Gym, а також описати варіанти використання, що відображають ключові сценарії роботи користувача. Оскільки система орієнтована насамперед на персональне ведення тренувального процесу, головним актором є користувач застосунку. Додатково можна виділити серверну частину системи як зовнішній компонент, що забезпечує збереження, обробку або синхронізацію даних у разі використання мережевих можливостей.

Основним актором системи є користувач. Він взаємодіє з інтерфейсом застосунку, переглядає каталог вправ, створює власні вправи, формує тренування, складає тренувальні програми, запускає тренувальні сесії, фіксує результати виконання вправ, переглядає календар занять, аналізує прогрес та редагує дані профілю. Для цього користувач використовує клієнтську частину застосунку, яка може бути запущена на Android, iOS або Desktop.

Другим актором доцільно визначити серверну частину системи. Вона не є користувачем у прямому розумінні, однак бере участь у взаємодії як зовнішній програмний компонент. Серверна частина може виконувати автентифікацію користувача, приймати та повертати дані через REST API, зберігати інформацію у базі даних PostgreSQL, а також забезпечувати подальшу синхронізацію даних між пристроями. У локальному режимі частина функцій може виконуватися без звернення до сервера, оскільки застосунок має власне локальне сховище на базі Room і SQLite.

Основні варіанти використання системи охоплюють усі дії, які потрібні для повноцінного ведення тренувального процесу. Користувач повинен мати змогу переглядати головний екран застосунку, переходити між основними розділами, працювати з каталогом вправ, створювати та редагувати тренування, формувати тренувальні програми, запускати тренування, вводити результати виконання вправ і переглядати статистику прогресу.

До основних варіантів використання користувача належать: перегляд каталогу вправ; пошук і фільтрація вправ; створення або редагування вправи; створення тренування; додавання вправ, відпочинку або програми до тренування; запуск тренувальної сесії; фіксація результатів виконання вправ; завершення тренування; перегляд історії тренувань; перегляд календаря занять; перегляд прогресу за вправами; редагування профілю користувача.

Серверна частина бере участь у таких варіантах використання: автентифікація користувача; отримання профілю користувача; оновлення профілю; отримання каталогу вправ; збереження змін у вправах; отримання тренувальних програм; збереження або оновлення тренувальних програм. Така взаємодія дозволяє розширити можливості застосунку за межі локального використання та підготувати систему до майбутньої синхронізації між різними пристроями.

Для наочного представлення взаємодії акторів із системою доцільно побудувати UML-діаграму варіантів використання. На ній необхідно відобразити актора “Користувач”, програмну систему GanG Gym та основні сценарії: робота з вправами, робота з тренуваннями, робота з програмами, перегляд календаря, перегляд прогресу та керування профілем. Окремо можна показати серверну частину як зовнішній компонент, що взаємодіє з варіантами використання, пов’язаними з авторизацією та збереженням даних.

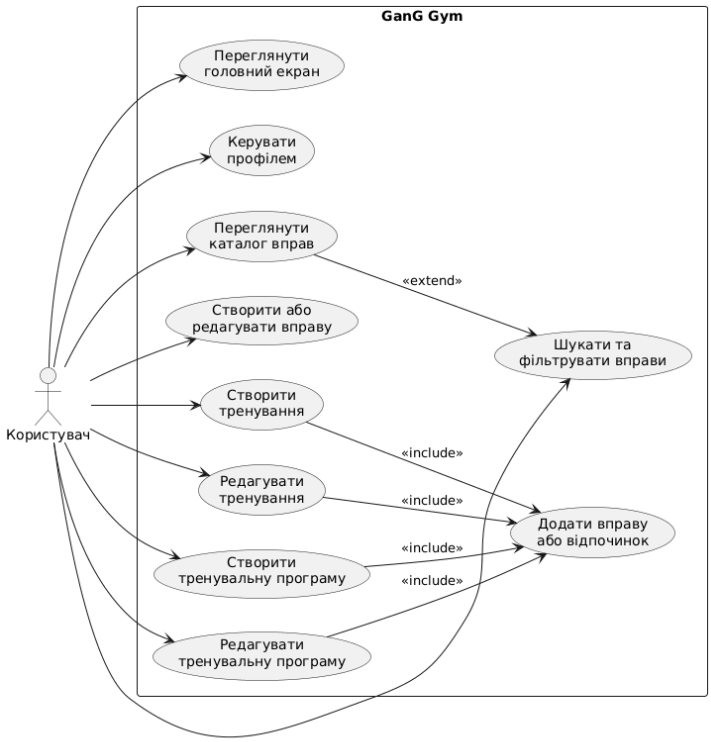

Рисунок 1.1 — Діаграма варіантів використання користувача в системі

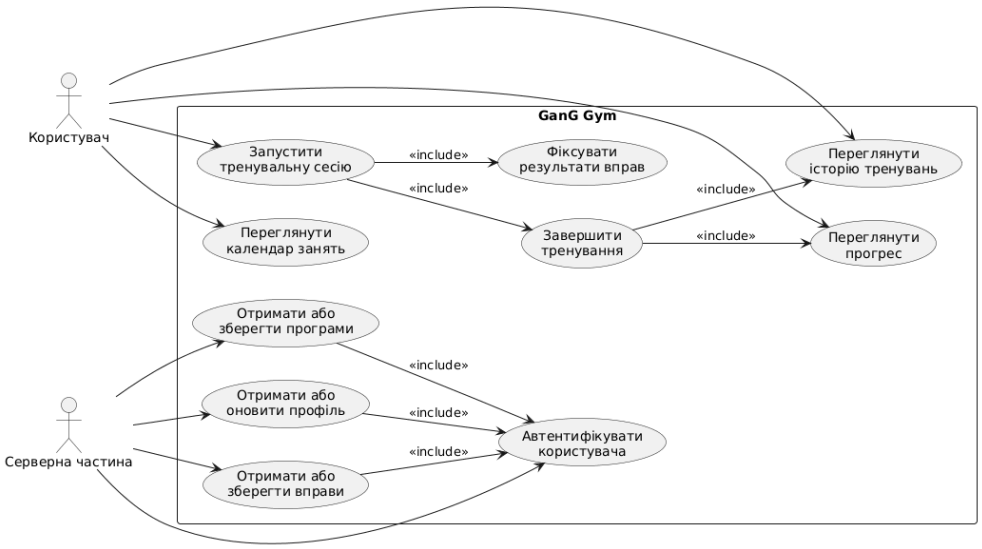

Рисунок 1.2 — Діаграма варіантів використання тренувальної сесії та серверної взаємодії

Дві діаграми варіантів використання дозволяють чіткіше визначити межі системи та розділити її функціональність на логічні частини. Перша діаграма

відображає основні дії користувача під час роботи з інтерфейсом застосунку, зокрема керування вправами, тренуваннями, тренувальними програмами та профілем. Друга діаграма демонструє сценарії, пов’язані з виконанням тренувальної сесії, переглядом історії та прогресу, а також взаємодією із серверною частиною. Такий поділ спрощує сприйняття структури системи та допомагає перейти до формування конкретних функціональних вимог, що будуть використані під час проєктування і реалізації програмного продукту.

1.4 Опис ключових варіантів використання

Після визначення акторів системи доцільно детальніше описати основні варіанти використання застосунку GanG Gym. Такий опис дозволяє зрозуміти логіку взаємодії користувача з програмним продуктом, визначити послідовність дій у типових сценаріях та уточнити функціональні вимоги до системи. Основними сценаріями є робота з каталогом вправ, створення тренувань і тренувальних програм, запуск тренувальної сесії, фіксація результатів, перегляд календаря та аналіз прогресу.

Першим ключовим варіантом використання є робота з каталогом вправ. Користувач відкриває відповідний розділ застосунку та отримує список доступних вправ. Для зручності передбачено пошук за назвою та фільтрацію за категорією, обладнанням і рівнем складності. Після вибору вправи користувач може переглянути її опис, основні параметри та зображення. У разі потреби він може створити нову вправу або відредагувати вже наявну, вказавши назву, опис, категорію, обладнання, складність і м’язові групи.

Другим важливим сценарієм є створення тренування. Користувач переходить до розділу тренувань, натискає кнопку створення нового тренування та заповнює основні дані: назву, опис, категорію, обладнання, рівень складності й доступність. Після цього він формує структуру тренування, додаючи до нього вправи, блоки відпочинку або готові тренувальні програми. Для кожної вправи можна вказати

кількість підходів, повторень, робочу вагу, цільову тривалість і час відпочинку. Після збереження тренування стає доступним у загальному списку.

Окремим варіантом використання є створення тренувальної програми. Цей сценарій схожий на створення тренування, однак програма використовується як багаторазовий шаблон. Користувач може сформувати послідовність вправ і блоків відпочинку, зберегти її та надалі використовувати під час створення тренувань. Такий підхід спрощує організацію регулярних занять, оскільки користувачу не потрібно щоразу вручну формувати однакову структуру тренування.

Найважливішим сценарієм роботи системи є запуск тренувальної сесії. Користувач обирає потрібне тренування та запускає його виконання. Під час тренування система відображає список вправ і параметри, які потрібно виконати. Користувач поступово вносить фактичні результати: кількість повторень, використану вагу, тривалість виконання або інші доступні показники. Після завершення тренування він може додати примітку та зберегти сесію. Збережені результати використовуються для формування історії та розрахунку прогресу.

Ще одним ключовим варіантом використання є перегляд календаря занять. Користувач відкриває календар і бачить тренування, пов’язані з конкретними датами. Після вибору дня система відображає список виконаних або запланованих тренувань за цю дату. Користувач може переглянути коротку інформацію про тренувальну сесію, час виконання, кількість підходів, загальну кількість повторень і власні примітки. Такий сценарій дозволяє швидко оцінити регулярність занять і знайти потрібний запис у тренувальній історії.

Останнім важливим сценарієм є перегляд прогресу. Система аналізує збережені результати тренувальних сесій та формує показники для кожної вправи. Користувач може побачити загальну кількість підходів, повторень, найкращу зафіксовану вагу, сумарний час виконання та останній результат. Це дозволяє відстежувати зміни у фізичній підготовці, порівнювати поточні показники з попередніми та приймати рішення щодо подальшого тренувального навантаження.

Описані варіанти використання охоплюють основні сценарії взаємодії користувача із системою GanG Gym. Вони формують основу для подальшої

специфікації функціональних вимог, проєктування структури даних, побудови інтерфейсу користувача та реалізації програмної логіки застосунку.

1.5 Специфікація вимог до програмного забезпечення

Специфікація вимог до програмного забезпечення визначає основні можливості, обмеження та якісні характеристики програмної системи GanG Gym. Вимоги сформовано на основі аналізу предметної області, визначених акторів і ключових варіантів використання. Вони описують, які функції повинен виконувати застосунок, якою має бути його поведінка у типових сценаріях, а також які умови необхідно забезпечити для стабільної та зручної роботи користувача.

Програмна система GanG Gym повинна забезпечувати повний цикл організації тренувального процесу: роботу з каталогом вправ, створення тренувань і тренувальних програм, запуск тренувальних сесій, фіксацію результатів, перегляд календаря занять і аналіз прогресу. Окрім цього, система повинна мати зручний користувацький інтерфейс, підтримувати локальне збереження даних і бути придатною для подальшого розширення.

Вимоги до програмного забезпечення поділяються на функціональні та нефункціональні. Функціональні вимоги визначають конкретні дії, які система повинна виконувати. Нефункціональні вимоги описують якісні характеристики програмного продукту, зокрема продуктивність, надійність, зручність використання, безпеку та підтримку різних платформ.

1.5.1 Функціональні вимоги

Функціональні вимоги описують основні можливості, які повинна надавати програмна система користувачу. Для застосунку GanG Gym ці вимоги пов’язані з веденням тренувального процесу, збереженням даних, роботою з вправами, тренуваннями, програмами та результатами занять.

Система повинна надавати користувачу можливість переглядати головний екран застосунку, з якого можна перейти до основних розділів: каталогу вправ,

тренувань, тренувальних програм, календаря, прогресу та профілю користувача. Навігація між розділами має бути зрозумілою та доступною з основного інтерфейсу.

Застосунок повинен забезпечувати роботу з каталогом вправ. Користувач має мати змогу переглядати список вправ, виконувати пошук за назвою, застосовувати фільтри за категорією, обладнанням і рівнем складності. Для кожної вправи система повинна відображати назву, опис, зображення або посилання на нього, категорію, обладнання, складність, основні та додаткові м’язові групи.

Система повинна дозволяти створювати та редагувати вправи. Під час створення вправи користувач повинен мати змогу ввести назву, опис, обрати категорію, обладнання, рівень складності та м’язові групи. Після збереження нова вправа має відображатися у каталозі та бути доступною для додавання до тренувань або тренувальних програм.

Застосунок повинен забезпечувати створення та редагування тренувань. Користувач має мати змогу вказати назву тренування, опис, категорію, обладнання, складність і параметри доступності. До тренування повинні додаватися окремі вправи, блоки відпочинку або готові тренувальні програми. Для вправ у складі тренування необхідно передбачити налаштування кількості підходів, повторень, робочої ваги, цільової тривалості та часу відпочинку.

Система повинна підтримувати створення тренувальних програм. Тренувальна програма розглядається як шаблон, який користувач може повторно використовувати під час створення тренувань. Користувач повинен мати змогу додавати до програми вправи та блоки відпочинку, редагувати її структуру, зберігати зміни та переглядати створені програми у відповідному розділі.

Однією з основних функцій системи є запуск тренувальної сесії. Користувач повинен мати змогу обрати тренування та розпочати його виконання. Під час тренувальної сесії система повинна відображати вправи та їхні планові параметри, а також дозволяти вводити фактичні результати: кількість повторень, використану вагу, тривалість виконання та примітки.

Після завершення тренування система повинна зберігати результати тренувальної сесії. Збережені дані мають використовуватися для формування історії тренувань, календаря занять і статистики прогресу. Користувач повинен мати змогу переглянути раніше виконані тренування та детальну інформацію про результати конкретної сесії.

Застосунок повинен містити календар занять. У календарі користувач має бачити тренування, пов’язані з конкретними датами. Після вибору дати система повинна відображати список тренувань за цей день, час виконання, кількість підходів, загальну кількість повторень та інші доступні показники.

Система повинна забезпечувати перегляд прогресу користувача. На основі збережених результатів тренувань застосунок має формувати показники за вправами: загальну кількість підходів, загальну кількість повторень, найкращу зафіксовану вагу, сумарний час виконання та останній результат. Це дозволяє користувачу оцінювати динаміку власних тренувань.

Також система повинна надавати можливість роботи з профілем користувача. У профілі можуть зберігатися ім’я, нікнейм, стать, дата народження, зріст, електронна пошта та фото користувача. Користувач повинен мати змогу переглядати та змінювати ці дані.

Окремою функціональною вимогою є підтримка локального збереження даних. Система повинна зберігати інформацію про вправи, тренування, програми, тренувальні сесії та результати у локальній базі даних, щоб користувач міг працювати із застосунком без постійного підключення до мережі.

Серверна частина системи повинна забезпечувати обробку запитів через API, роботу з профілем користувача, вправами та тренувальними програмами. Для захищених запитів має передбачатися механізм автентифікації користувача.

1.5.2 Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики програмної системи GanG Gym. Вони не описують окремі функції застосунку, але визначають, наскільки зручно, стабільно, безпечно та ефективно система повинна працювати.

Застосунок повинен мати зручний і зрозумілий користувацький інтерфейс. Основні дії користувача мають виконуватися без зайвих переходів, а структура екранів повинна бути логічною та послідовною. Користувач повинен швидко знаходити потрібні розділи, створювати тренування, запускати сесію та переглядати результати без необхідності додаткового навчання.

Система повинна забезпечувати кросплатформність. Основна логіка застосунку має бути реалізована таким чином, щоб її можна було використовувати на Android, iOS та Desktop. Використання Kotlin Multiplatform і Compose Multiplatform дозволяє зменшити дублювання коду та забезпечити однакову поведінку системи на різних платформах.

Програмний продукт повинен бути продуктивним. Екрани застосунку мають відкриватися без значних затримок, списки вправ, тренувань і програм повинні відображатися коректно навіть за наявності великої кількості записів. Пошук і фільтрація вправ мають виконуватися швидко та не блокувати роботу інтерфейсу.

Система повинна бути надійною. Дані користувача не повинні втрачатися після закриття або перезапуску застосунку. У разі помилки під час введення або збереження даних система повинна повідомляти користувача про проблему та не допускати пошкодження вже збереженої інформації.

Важливою вимогою є підтримка локальної роботи. Користувач повинен мати можливість працювати з основними функціями застосунку без постійного підключення до інтернету. Локальна база даних має забезпечувати збереження вправ, тренувань, програм і результатів тренувальних сесій на пристрої користувача.

Серверна частина повинна бути масштабованою та придатною до подальшого розвитку. Архітектура API має дозволяти додавати нові функції, наприклад синхронізацію між пристроями, хмарне збереження тренувань або розширену авторизацію, без повної перебудови системи.

Система повинна забезпечувати базовий рівень безпеки. Доступ до захищених серверних ресурсів має виконуватися лише після автентифікації

користувача. Дані профілю та тренувальна інформація повинні оброблятися таким чином, щоб запобігти несанкціонованому доступу.

Програмний код повинен бути підтримуваним і структурованим. Моделі даних, репозиторії, презентери, екрани інтерфейсу та серверні маршрути мають бути розділені за призначенням. Такий підхід спрощує тестування, виправлення помилок і подальше розширення функціональності.

Застосунок повинен бути придатним до тестування. Основні модулі системи мають перевірятися за допомогою функціонального та автоматизованого тестування. Особливу увагу необхідно приділити перевірці роботи з базою даних, створенню тренувань, запуску тренувальних сесій, збереженню результатів і коректному відображенню прогресу.

Отже, сформульовані функціональні та нефункціональні вимоги визначають основу для подальшого проєктування програмної системи GanG Gym. Вони дозволяють перейти до вибору архітектури, побудови моделі даних, реалізації основних модулів та перевірки відповідності готового застосунку поставленим завданням.

1.6 Висновки до розділу 1

У першому розділі було проведено аналіз вимог до програмної системи GanG Gym, призначеної для планування, виконання та моніторингу тренувального процесу. Розглянуто предметну область фітнес-застосунків і визначено основні проблеми, які виникають під час ручного ведення тренувань, зокрема складність пошуку попередніх результатів, незручність повторного використання тренувальних шаблонів та відсутність зручного інструменту для аналізу прогресу.

У процесі аналізу було визначено, що розроблювана система повинна забезпечувати користувачу повний цикл роботи з тренувальними даними: перегляд і створення вправ, формування тренувань і програм, запуск тренувальних сесій, збереження результатів, перегляд календаря занять та аналіз фізичного прогресу. Окрему увагу приділено необхідності кросплатформної реалізації, оскільки

користувачі можуть працювати із застосунком як з мобільних пристроїв, так і з персонального комп’ютера.

Було визначено основних акторів системи та варіанти використання. Головним актором є користувач, який взаємодіє з інтерфейсом застосунку та виконує основні дії, пов’язані з організацією тренувального процесу. Додатково розглянуто серверну частину як зовнішній програмний компонент, що може забезпечувати автентифікацію, обробку API-запитів, збереження профілю, вправ і тренувальних програм.

Також було описано ключові варіанти використання системи: роботу з каталогом вправ, створення тренувань, створення тренувальних програм, запуск тренувальної сесії, фіксацію результатів, перегляд календаря та аналіз прогресу. Ці сценарії дали змогу конкретизувати поведінку системи та визначити основні функції, які мають бути реалізовані у програмному продукті.

На основі проведеного аналізу сформульовано функціональні та нефункціональні вимоги до програмного забезпечення. Функціональні вимоги описують можливості, які система повинна надавати користувачу, а нефункціональні визначають якісні характеристики застосунку: зручність інтерфейсу, продуктивність, надійність, локальну роботу, безпеку, підтримуваність, придатність до тестування та кросплатформність.

Отже, результати першого розділу формують основу для подальшого проєктування та реалізації програмної системи GanG Gym. Визначені вимоги, актори та варіанти використання дозволяють перейти до вибору архітектури, побудови моделі даних, розробки користувацького інтерфейсу та реалізації основних модулів застосунку.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

У цьому розділі розглянуто процес проєктування та розробки програмної системи GanG Gym. На основі вимог, визначених у першому розділі, обґрунтовано вибір підходу до розробки, спроєктовано загальну архітектуру застосунку, структуру клієнтської та серверної частин, модель збереження даних і основні програмні модулі. Окрему увагу приділено вибору технологічного стеку, організації кросплатформної кодової бази, реалізації локальної бази даних, серверного API та побудові користувацького інтерфейсу.

Під час проєктування системи враховано необхідність підтримки кількох платформ, зокрема Android, iOS та Desktop, а також можливість подальшого розширення функціональності. Для цього застосовано архітектурний підхід із розділенням відповідальностей між шарами даних, доменної логіки, презентаційної логіки та інтерфейсу користувача. Така структура дозволяє зробити програмний продукт більш зрозумілим, підтримуваним і придатним до тестування.

У межах розділу буде описано вибір процесу розробки, архітектуру системи, структуру бази даних, UML-діаграми, вибір мови програмування та середовища розробки, реалізацію основних класів і методів, а також особливості створення інтерфейсу користувача. Результати цього розділу є основою для подальшого тестування, впровадження та оцінки працездатності розробленого застосунку.

2.1 Вибір процесу розробки

Під час створення програмної системи GanG Gym важливим етапом є вибір процесу розробки, який визначає послідовність виконання робіт, спосіб уточнення вимог, організацію реалізації функціональності та перевірку результатів. Оскільки застосунок містить декілька взаємопов’язаних модулів, зокрема каталог вправ, тренування, тренувальні програми, календар, профіль користувача, прогрес і серверну частину, розробку доцільно виконувати поступово, з поетапною перевіркою працездатності кожної частини.

Для реалізації програмної системи обрано ітеративний підхід до розробки, який відповідає принципам Agile. Такий підхід дозволяє розділити створення застосунку на окремі етапи та після кожного з них отримувати працездатну частину програмного продукту. Це є особливо важливим для кросплатформного застосунку, оскільки необхідно поступово перевіряти роботу спільної логіки, локальної бази даних, користувацького інтерфейсу та платформозалежних компонентів.

На початковому етапі було виконано аналіз предметної області та сформовано основні вимоги до системи. Після цього визначено ключові сценарії використання, акторів і структуру майбутнього застосунку. На основі цих даних спроєктовано загальну архітектуру системи, модель даних і набір основних модулів, які потрібно реалізувати.

Наступним етапом стала реалізація базової структури проєкту Kotlin Multiplatform. Було організовано окремі модулі для Android, iOS, Desktop, спільної клієнтської логіки, ядра системи та серверної частини. Такий поділ дозволив відокремити платформозалежний код від спільної логіки та забезпечити можливість повторного використання значної частини програмного коду на різних платформах.

Подальша розробка виконувалася за функціональними блоками. Спочатку було реалізовано моделі даних, локальну базу даних і репозиторії для роботи з вправами, тренуваннями, тренувальними програмами та тренувальними сесіями. Після цього створено презентаційну логіку, яка відповідає за підготовку даних для інтерфейсу користувача. На наступному етапі розроблено екрани застосунку: головний екран, каталог вправ, список тренувань, тренувальні програми, календар, прогрес і профіль користувача.

Окремою ітерацією було реалізовано серверну частину системи. Вона побудована з використанням Ktor і містить маршрути для роботи з профілем користувача, вправами та тренувальними програмами. Для збереження серверних даних використано PostgreSQL, а для керування змінами структури бази даних застосовано Flyway. Також передбачено механізм автентифікації, який у

локальному режимі може працювати через заголовок розробника, а в розгорнутому середовищі — через Firebase.

Перевагою ітеративного процесу розробки є можливість поступово перевіряти коректність реалізації та вносити зміни без повної перебудови системи. Наприклад, після реалізації каталогу вправ можна було окремо перевірити пошук, фільтрацію та редагування вправ, а після створення модуля тренувань — перевірити додавання вправ, запуск сесії та збереження результатів. Такий підхід зменшує ризик накопичення помилок і полегшує тестування.

Обраний процес розробки також добре підходить для навчального проєкту, оскільки дозволяє логічно розподілити роботу між етапами аналізу, проєктування, реалізації та тестування. Кожна ітерація завершується отриманням конкретного результату, який можна оцінити, продемонструвати та використати як основу для наступного етапу.

Отже, для розробки програмної системи GanG Gym було обрано ітеративний процес розробки з використанням принципів Agile. Він забезпечує гнучкість, поступове уточнення вимог, поетапну реалізацію функціональності та можливість регулярної перевірки працездатності застосунку на різних платформах.

2.2 Проєктування архітектури системи

Архітектура програмної системи GanG Gym спроєктована з урахуванням кросплатформності, модульності та можливості подальшого розширення. Основною ідеєю архітектури є розділення застосунку на декілька логічних частин: клієнтську частину, спільну бізнес-логіку, локальне сховище даних, серверну частину та базу даних сервера. Такий підхід дозволяє зменшити дублювання коду, спростити підтримку системи та забезпечити однакову поведінку застосунку на різних платформах.

Клієнтська частина системи реалізована на основі Kotlin Multiplatform та Compose Multiplatform. Завдяки цьому значна частина коду, зокрема моделі даних, репозиторії, презентаційна логіка та екрани інтерфейсу, може використовуватися

спільно для Android, iOS та Desktop. Окремі платформозалежні модулі відповідають лише за ті функції, які залежать від конкретної операційної системи, наприклад створення локальної бази даних, роботу з фотографіями або запуск застосунку на відповідній платформі.

У клієнтській частині використано багатошарову архітектуру. Шар інтерфейсу користувача відповідає за відображення екранів і обробку дій користувача. Презентаційний шар містить класи, які формують стан екранів і керують логікою взаємодії з користувачем. Доменний шар описує основні сценарії роботи з даними, наприклад отримання вправ, тренувань, програм або результатів тренувальних сесій. Шар даних відповідає за роботу з репозиторіями, локальною базою даних і перетворенням сутностей бази даних у моделі, зручні для використання в застосунку.

Для локального збереження даних використовується база даних SQLite через бібліотеку Room. У ній зберігаються вправи, тренування, блоки тренувань, тренувальні програми, результати виконання вправ і тренувальні сесії. Локальне сховище дозволяє користувачу працювати із застосунком без постійного підключення до інтернету. Це є важливим для фітнес-застосунку, оскільки користувач може використовувати його безпосередньо під час тренування, коли мережеве з’єднання може бути нестабільним або відсутнім.

Серверна частина системи реалізована окремим модулем на основі фреймворку Ktor. Вона надає API для роботи з профілем користувача, вправами та тренувальними програмами. Сервер також містить механізм автентифікації, який дозволяє обмежити доступ до захищених маршрутів. У режимі розробки може використовуватися спрощена автентифікація через спеціальний заголовок користувача, а для розгорнутого середовища передбачена інтеграція з Firebase.

Для збереження серверних даних використовується PostgreSQL. Структура серверної бази даних керується за допомогою Flyway-міграцій, що дозволяє послідовно змінювати схему бази даних і контролювати її версії. Доступ до бази даних організовано через репозиторії, які відокремлюють логіку роботи з даними

від серверних маршрутів. Це робить код більш структурованим і спрощує тестування серверної частини.

Взаємодія користувача із системою відбувається через клієнтський застосунок. Користувач відкриває потрібний розділ, наприклад каталог вправ або список тренувань, після чого інтерфейс звертається до презентаційного шару. Презентаційний шар отримує дані через відповідні use case або репозиторії, а репозиторії працюють із локальною базою даних. У разі використання серверних можливостей клієнтська частина може надсилати запити до серверного API, яке обробляє їх, перевіряє автентифікацію та взаємодіє з PostgreSQL.

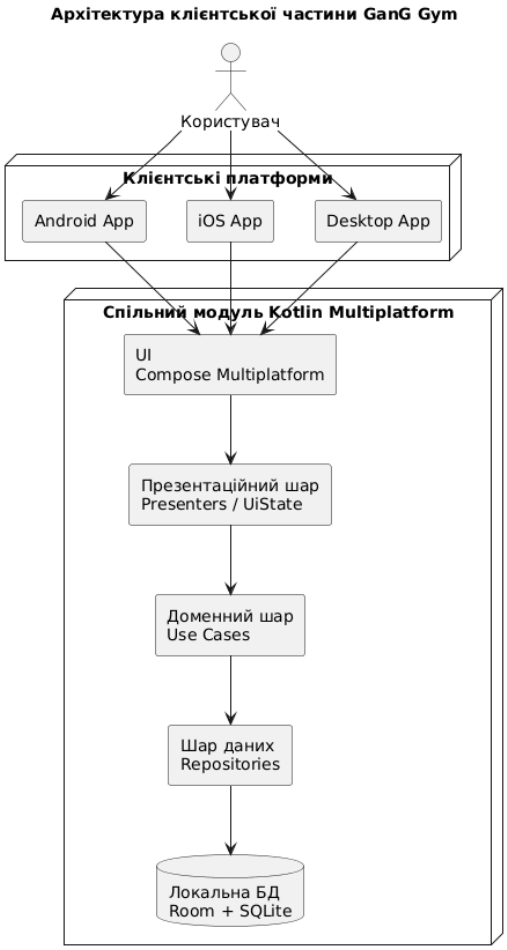

Рисунок 2.1 — Архітектура клієнтської частини

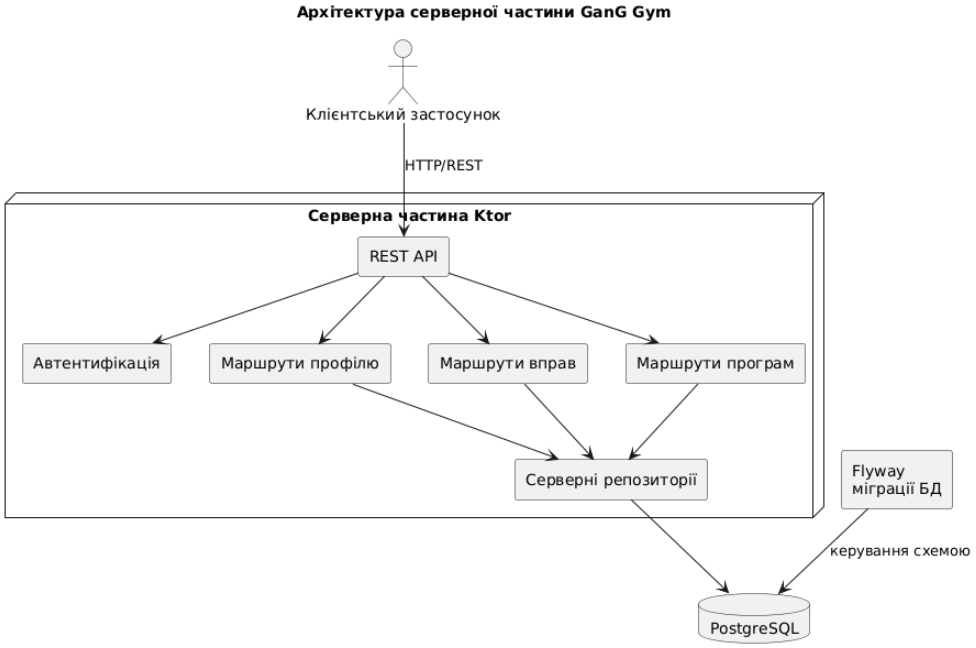

Рисунок 2.2 — Архітектура серверної частини

Загальна архітектура системи передбачає чітке розділення відповідальностей між її компонентами. Клієнтський застосунок відповідає за взаємодію з користувачем і локальне збереження тренувальних даних. Серверна частина відповідає за обробку мережевих запитів, автентифікацію та централізоване збереження окремих даних. База даних клієнта забезпечує автономну роботу застосунку, а серверна база даних створює основу для подальшої синхронізації та розширення функціональності.

Обрана архітектура є придатною для подальшого розвитку системи. У майбутньому до неї можна додати повну синхронізацію між пристроями, хмарне збереження тренувальної історії, систему авторизації користувачів, рекомендації щодо тренувань або соціальні функції. Завдяки модульній структурі такі зміни можуть бути реалізовані без повної перебудови програмного продукту.

Отже, архітектура GanG Gym побудована як кросплатформна модульна система, що поєднує спільну клієнтську логіку, локальну базу даних, серверний API та серверну базу даних. Такий підхід забезпечує зручність використання, підтримуваність, масштабованість і технічну основу для подальшого розвитку застосунку.

2.3 Побудова схем бази даних

Для збереження даних у програмній системі GanG Gym передбачено дві схеми бази даних: локальну клієнтську базу даних та серверну базу даних. Локальна база даних використовується безпосередньо в застосунку та забезпечує автономну роботу користувача без постійного підключення до мережі. Серверна база даних призначена для централізованого збереження профілю користувача, вправ і тренувальних програм, а також створює основу для подальшої синхронізації між пристроями.

Локальна база даних реалізована за допомогою бібліотеки Room та вбудованої бази SQLite. Вона містить основні сутності, необхідні для ведення тренувального процесу: вправи, тренування, блоки тренувань, тренувальні програми, блоки програм, тренувальні сесії та результати виконання вправ. Така структура дозволяє зберігати як шаблони тренувань, так і фактичні результати користувача після виконання занять.

Основною таблицею для роботи з вправами є exercises. У ній зберігається інформація про назву вправи, опис, зображення, категорії, обладнання, основні та додаткові м’язові групи, рівень складності, власника запису, дату створення та оновлення. Також таблиця містить службові поля для синхронізації, зокрема стан синхронізації, очікувану операцію та час останньої синхронізації.

Таблиця workouts призначена для збереження тренувань. Вона містить назву, опис, категорії, обладнання, складність, орієнтовну тривалість, тип доступності, джерело створення та службові поля синхронізації. Кожне тренування може складатися з кількох блоків, які зберігаються у таблиці workout_blocks. Блок тренування може представляти окрему вправу, відпочинок, вкладену програму або інший структурний елемент. Для кожного блоку зберігається порядок виконання, тип, примітка, пов’язана вправа або програма, кількість повторень, робоча вага, тривалість, кількість раундів і параметри відпочинку.

Для багаторазових тренувальних шаблонів використовується таблиця routines. Вона зберігає основні дані про тренувальну програму: власника, назву,

опис, дату створення та оновлення. Структура програми описується таблицею routine_blocks, яка за своїм призначенням подібна до workout_blocks, але належить не конкретному тренуванню, а тренувальній програмі. Це дозволяє створювати шаблони, які надалі можна використовувати під час формування тренувань.

Фактичні результати виконання тренувань зберігаються у таблиці workout_sessions. Вона містить інформацію про власника сесії, пов’язане тренування, час початку, час завершення, самопочуття користувача та примітки. Детальні результати окремих вправ зберігаються у таблиці exercise_results. Для кожного результату фіксується тренувальна сесія, вправа, номер підходу, кількість повторень, вага, тривалість виконання, час завершення та текстова примітка.

Між таблицями локальної бази даних передбачено зв’язки, які забезпечують цілісність даних. Таблиця workout_blocks пов’язана з таблицею workouts, тому під час видалення тренування автоматично видаляються всі його блоки. Аналогічно таблиця routine_blocks пов’язана з routines, а таблиця exercise_results — з workout_sessions. Для пришвидшення пошуку створено індекси за ключовими полями, зокрема ідентифікаторами тренувань, програм, сесій та вправ.

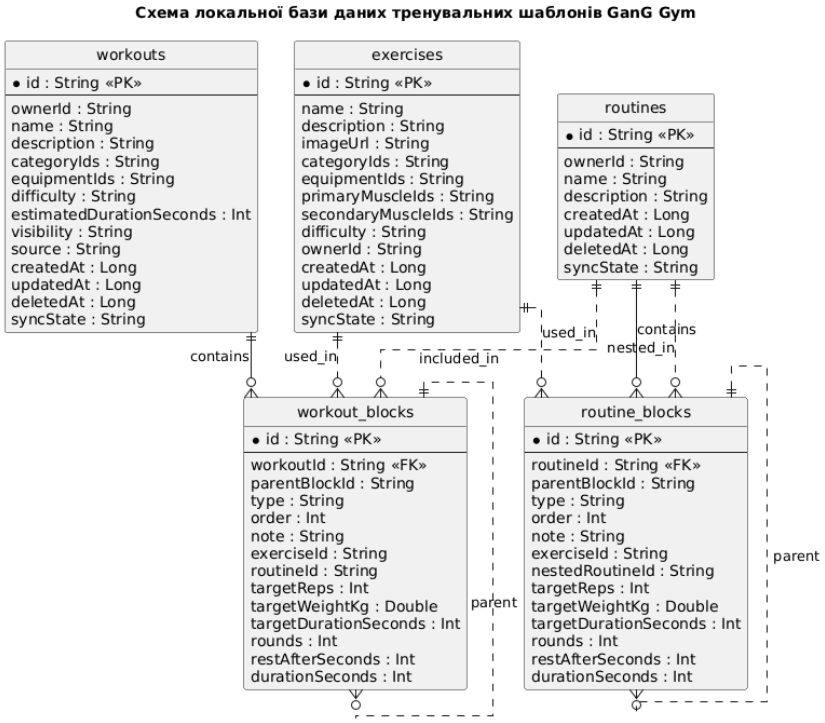

Рисунок 2.3 — Схема локальної бази даних тренувальних шаблонів

Схема локальної бази даних тренувальних шаблонів відображає структуру збереження вправ, тренувань і тренувальних програм. Таблиця exercises містить основну інформацію про вправи, які можуть використовуватися як у тренуваннях, так і в програмах. Таблиця workouts зберігає загальні дані про тренування, а таблиця workout_blocks описує його внутрішню структуру: вправи, відпочинок або вкладені програми. Аналогічно таблиця routines зберігає тренувальні програми, а routine_blocks визначає набір блоків, з яких складається кожна програма. Така структура дозволяє створювати як окремі тренування, так і багаторазові шаблони для регулярного використання.

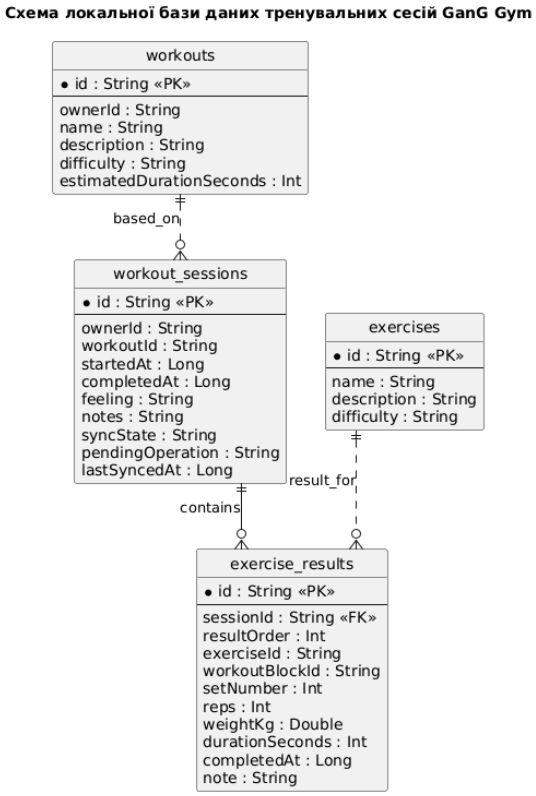

Рисунок 2.4 — Схема локальної бази даних тренувальних сесій

Схема локальної бази даних тренувальних сесій відображає збереження фактичних результатів виконання тренувань. Таблиця workout_sessions містить інформацію про конкретну сесію: користувача, пов’язане тренування, час початку, час завершення, самопочуття та примітки. Таблиця exercise_results зберігає результати виконання окремих вправ у межах тренувальної сесії, зокрема номер

підходу, кількість повторень, робочу вагу, тривалість виконання та час завершення. Завдяки такій структурі застосунок може формувати історію тренувань, відображати заняття у календарі та розраховувати показники прогресу користувача.

Серверна база даних реалізована на основі PostgreSQL. Її структура керується за допомогою Flyway-міграцій, що дозволяє поступово змінювати схему бази даних і контролювати її версії. Серверна схема містить таблиці user_profiles, exercises, routines та routine_blocks.

Таблиця user_profiles використовується для збереження профілю користувача. Вона містить ідентифікатор користувача, ім’я, нікнейм, електронну пошту, посилання на фото, стать, дату народження, зріст, дату створення, дату оновлення та дату видалення. Для нікнейма й електронної пошти створено унікальні індекси, що запобігають дублюванню цих значень.

Серверна таблиця exercises зберігає інформацію про вправи, які можуть належати конкретному користувачу або бути загальнодоступними. У ній передбачено поля для назви, опису, зображення, категорій, обладнання, м’язових груп і рівня складності. Для контролю коректності даних використовуються обмеження, наприклад перевірка непорожньої назви та допустимого значення складності.

Таблиці routines і routine_blocks відповідають за збереження тренувальних програм на сервері. Таблиця routines містить загальну інформацію про програму, а routine_blocks описує її структуру. Блоки програми можуть бути вправами, періодами відпочинку, вкладеними програмами або суперсетами. Для блоків передбачено перевірки коректності значень, зокрема порядку виконання, кількості повторень, ваги, тривалості та часу відпочинку.

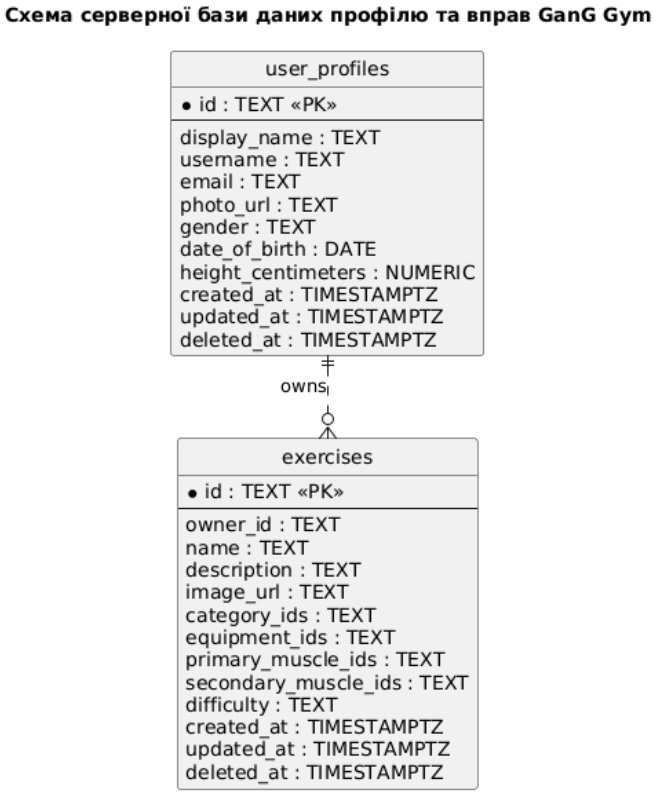

Рисунок 2.5 — Схема серверної бази даних профілю та вправ

Серверна схема профілю та вправ містить таблицю user_profiles, у якій зберігаються основні дані користувача: ім’я, нікнейм, електронна пошта, фото, стать, дата народження та зріст. Для контролю унікальності в базі даних передбачено унікальні індекси для нікнейма та електронної пошти. Таблиця exercises використовується для збереження вправ, які можуть належати конкретному користувачу або бути загальними. Поле owner_id дозволяє пов’язати вправу з користувачем, а службові поля created_at, updated_at і deleted_at забезпечують контроль часу створення, оновлення та м’якого видалення записів.

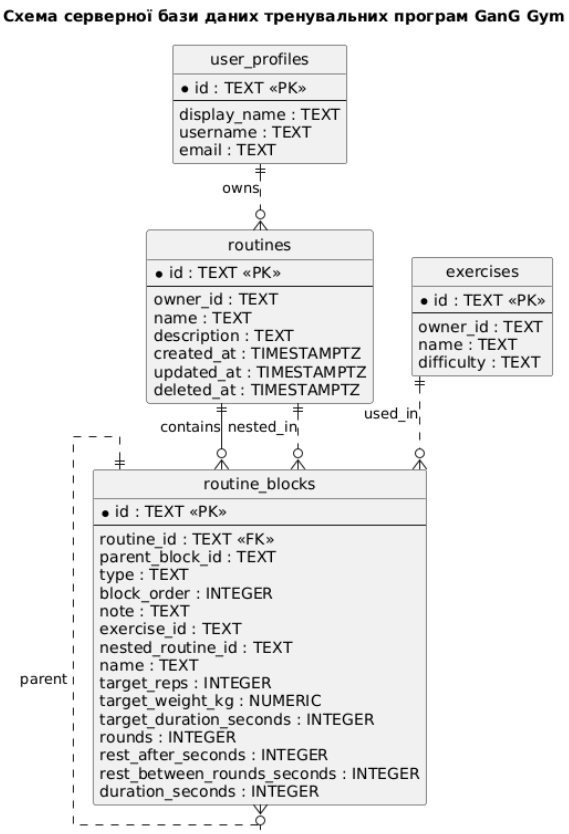

Рисунок 2.6 — Схема серверної бази даних тренувальних програм

Серверна схема тренувальних програм відображає структуру збереження багаторазових шаблонів занять. Таблиця routines містить загальні дані про програму: власника, назву, опис, дату створення, дату оновлення та дату видалення. Детальна структура програми зберігається у таблиці routine_blocks, де кожен запис описує окремий блок програми. Блок може бути вправою, відпочинком, вкладеною програмою або суперсетом. Для цього передбачено поля type, exercise_id, nested_routine_id, parent_block_id, а також параметри навантаження: кількість повторень, вага, тривалість, кількість раундів і час відпочинку. Такий підхід дозволяє зберігати як прості тренувальні програми, так і складніші вкладені структури.

Отже, побудована структура баз даних забезпечує збереження всіх основних даних, необхідних для роботи системи GanG Gym. Локальна база даних відповідає за автономну роботу застосунку та збереження тренувальної історії користувача, а серверна база даних забезпечує централізоване збереження профілю, вправ і

тренувальних програм. Такий підхід дозволяє поєднати швидку локальну роботу з можливістю подальшого розвитку серверної синхронізації.

2.4 Побудова UML-діаграм класів

Для опису внутрішньої структури програмної системи GanG Gym доцільно використати UML-діаграми класів. Вони дозволяють наочно показати основні сутності застосунку, їхні властивості, взаємозв’язки та роль у загальній архітектурі. Побудова таких діаграм є важливим етапом проєктування, оскільки допомагає перейти від опису вимог і схем бази даних до структури програмного коду.

У системі GanG Gym основними класами предметної області є Exercise, Workout, TrainingPlan, WorkoutSession та допоміжні ідентифікатори. Клас Exercise описує вправу та містить інформацію про її назву, опис, категорію, обладнання, рівень складності й м’язові групи. Клас Workout відповідає за тренування, яке може складатися з окремих вправ, блоків відпочинку або готових тренувальних програм. Клас TrainingPlan використовується для опису багаторазових тренувальних шаблонів, які користувач може застосовувати під час формування тренувань.

Окреме місце у структурі системи займає клас WorkoutSession, який описує фактичне виконання тренування. На відміну від Workout, що є шаблоном або планом заняття, WorkoutSession містить інформацію про конкретну виконану сесію: час початку, час завершення, результати вправ, примітки та інші показники. Саме ці дані надалі використовуються для побудови історії тренувань, календаря та відображення прогресу користувача.

Для відокремлення бізнес-логіки від джерел даних у системі використовуються репозиторії. Наприклад, ExerciseRepository, WorkoutRepository, RoutineRepository та WorkoutSessionRepository визначають інтерфейси роботи з відповідними сутностями. Їхні локальні реалізації взаємодіють із DAO-класами та локальною базою даних Room. Завдяки такому підходу інші частини застосунку не залежать безпосередньо від способу збереження даних.

Презентаційна логіка системи реалізована за допомогою класів-презентерів. Для кожного основного екрана передбачено окремий презентер і стан інтерфейсу. Наприклад, ExerciseCatalogPresenter формує стан каталогу вправ, обробляє пошук, фільтрацію, створення та редагування вправ. WorkoutListPresenter відповідає за список тренувань, створення тренування, запуск сесії та збереження результатів. RoutineListPresenter керує тренувальними програмами, CalendarPresenter відповідає за відображення занять у календарі, а ProgressPresenter формує дані для перегляду прогресу.

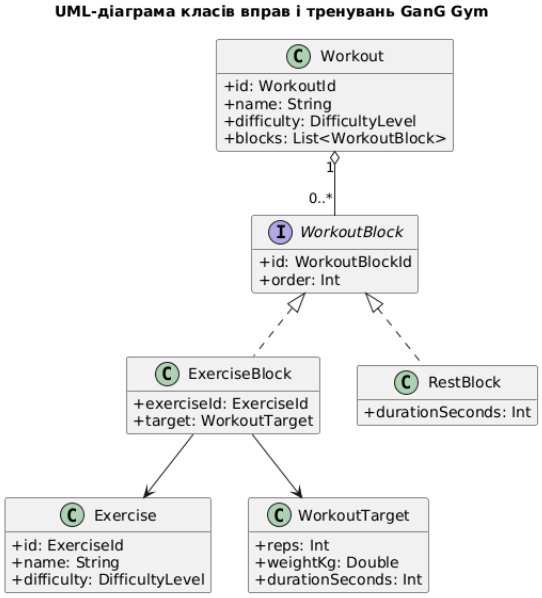

Рисунок 2.7 — UML-діаграма класів вправ і тренувань

Діаграма класів вправ і тренувань відображає базову структуру створення тренування. Клас Exercise описує окрему вправу, а клас Workout представляє тренування, що складається з набору блоків. Блоки тренування реалізовані через інтерфейс WorkoutBlock, який може мати різні конкретні типи. ExerciseBlock використовується для додавання вправи до тренування, а RestBlock — для опису періоду відпочинку. Клас WorkoutTarget містить планові параметри виконання вправи, зокрема кількість повторень, робочу вагу або тривалість.

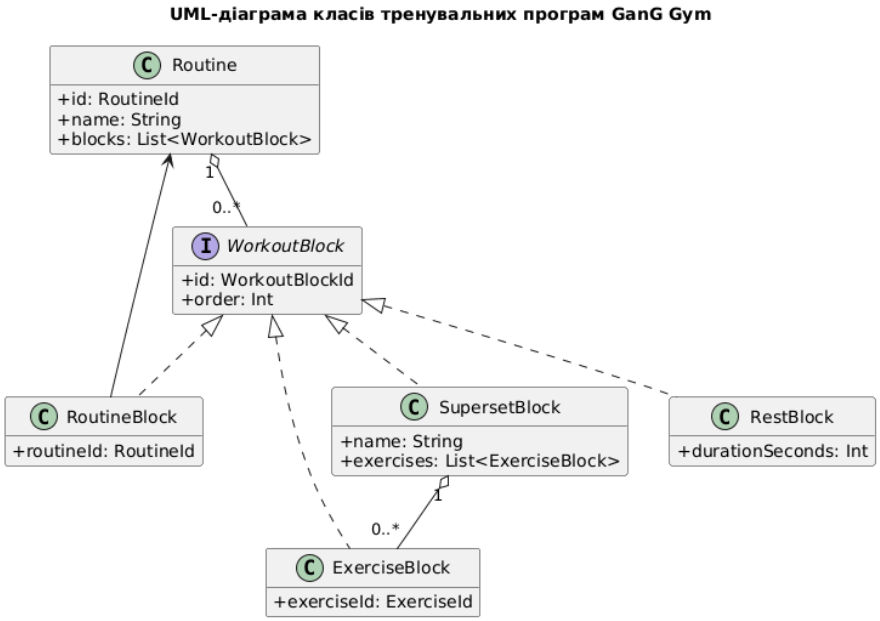

Рисунок 2.8 — UML-діаграма класів тренувальних програм

Діаграма класів тренувальних програм показує структуру багаторазових шаблонів занять. Клас Routine описує тренувальну програму, яка складається з блоків WorkoutBlock. До складу програми можуть входити окремі вправи, відпочинок, вкладені програми або суперсети. Такий підхід дозволяє створювати як прості послідовності вправ, так і складніші тренувальні структури з повторним використанням готових програм.

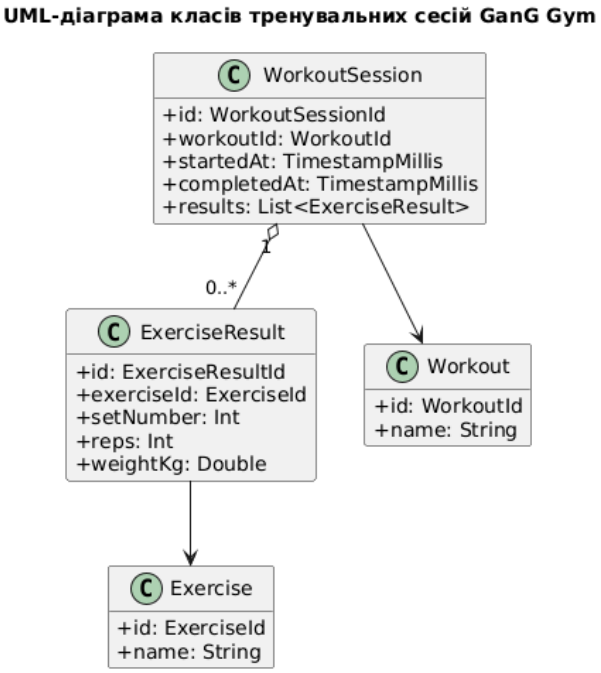

Рисунок 2.9 — UML-діаграма класів тренувальних сесій

Діаграма класів тренувальних сесій описує збереження фактичних результатів виконання тренування. Клас WorkoutSession представляє одну виконану сесію та пов’язується з тренуванням, на основі якого вона була запущена. Клас ExerciseResult містить результат виконання конкретної вправи в межах сесії, зокрема номер підходу, кількість повторень і робочу вагу. Ці дані використовуються для формування історії тренувань, календаря занять і показників прогресу користувача.

Діаграми доменної моделі розділено на декілька частин для кращої читабельності та простішого сприйняття структури системи. Перша діаграма відображає класи, пов’язані з вправами та тренуваннями, зокрема Exercise, Workout, WorkoutBlock, ExerciseBlock, RestBlock і WorkoutTarget. Друга діаграма описує структуру тренувальних програм, які складаються з блоків і можуть містити вправи, відпочинок, вкладені програми або суперсети. Третя діаграма демонструє класи, що використовуються для збереження виконаних тренувальних сесій і результатів вправ. Такий поділ дозволяє окремо показати шаблони тренувань,

багаторазові програми та фактичні результати користувача, не перевантажуючи одну UML-діаграму великою кількістю зв’язків.

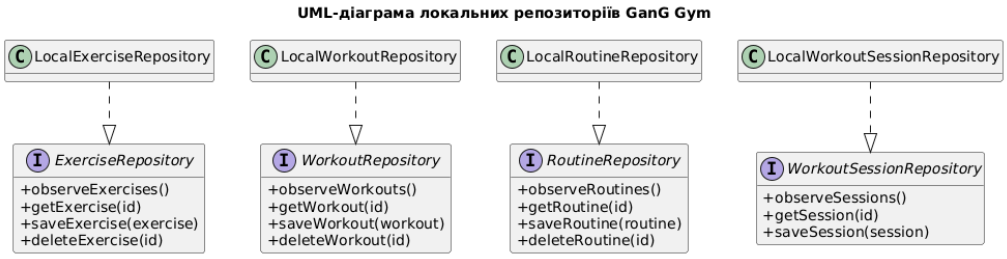

Рисунок 2.10 — UML-діаграма локальних репозиторіїв

Діаграма локальних репозиторіїв показує інтерфейси, через які інші частини клієнтського застосунку працюють із даними. Для кожної основної сутності передбачено окремий репозиторій: вправ, тренувань, тренувальних програм і тренувальних сесій. Локальні реалізації цих репозиторіїв приховують деталі роботи з базою даних Room і повертають доменні моделі, зручні для використання у презентаційному та доменному шарах.

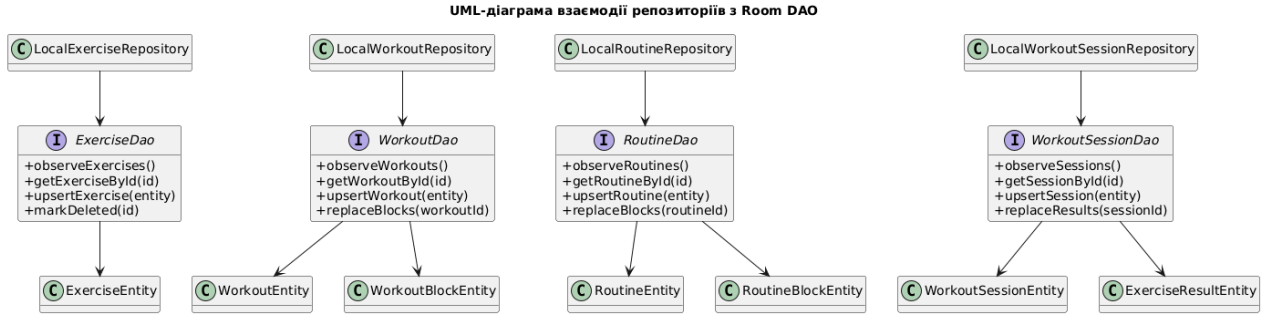

Рисунок 2.11 — UML-діаграма взаємодії репозиторіїв з Room DAO

Діаграма взаємодії репозиторіїв з Room DAO відображає шлях доступу до локальної бази даних. Репозиторії не працюють із таблицями напряму, а звертаються до відповідних DAO-інтерфейсів. DAO виконують запити до таблиць і повертають entity-об’єкти. Далі ці об’єкти перетворюються у доменні моделі за допомогою mapper-класів. Такий підхід відокремлює структуру бази даних від бізнес-логіки застосунку.

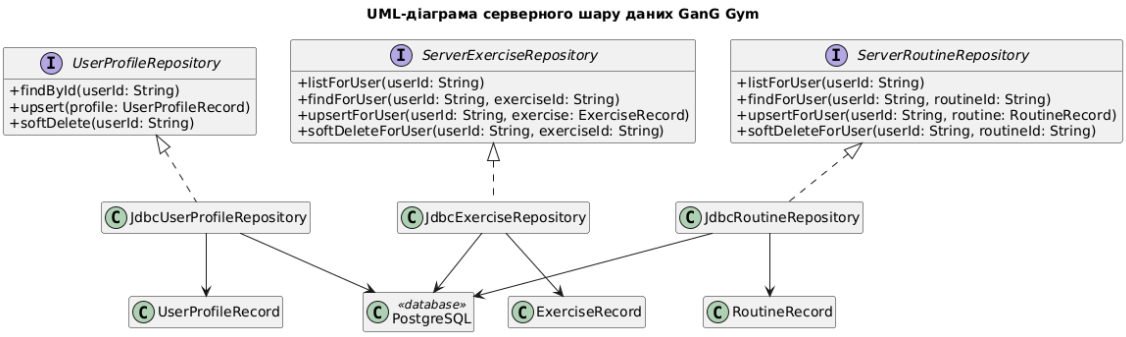

Рисунок 2.12 — UML-діаграма серверного шару даних

Діаграма серверного шару даних показує структуру роботи серверної частини з базою PostgreSQL. Для профілю користувача, вправ і тренувальних програм створено окремі repository-інтерфейси. Їхні JDBC-реалізації виконують запити до бази даних і працюють із record-класами, які описують серверні представлення даних. Завдяки такому поділу маршрути Ktor не залежать від конкретного способу доступу до бази даних, що спрощує тестування та подальшу підтримку серверної частини.

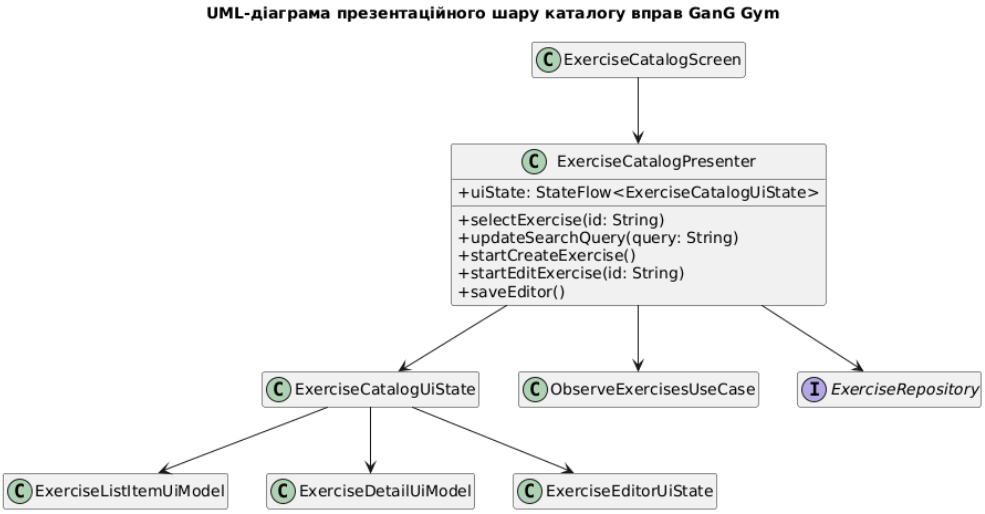

Рисунок 2.13 — UML-діаграма презентаційного шару каталогу вправ

Діаграма презентаційного шару каталогу вправ показує взаємодію між екраном, презентером і станом інтерфейсу. ExerciseCatalogScreen відповідає за відображення даних, а ExerciseCatalogPresenter формує ExerciseCatalogUiState,

обробляє пошук, фільтрацію, вибір вправи, створення та редагування записів. Дані надходять через ObserveExercisesUseCase і ExerciseRepository, що відокремлює інтерфейс від шару збереження даних.

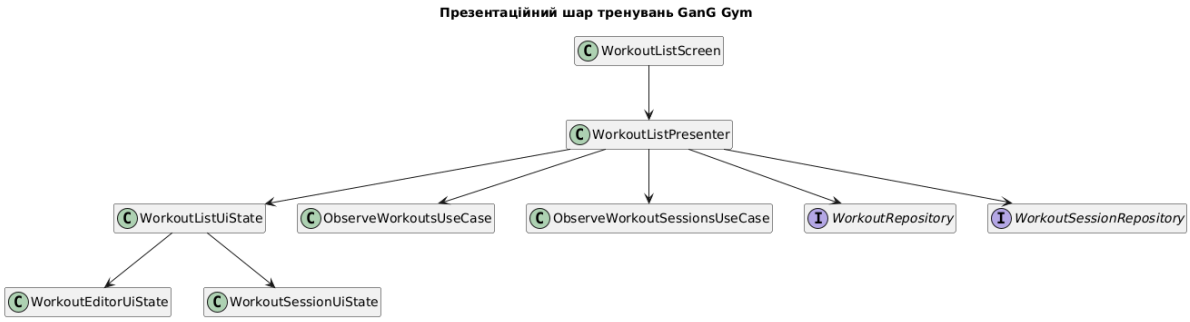

Рисунок 2.14 — UML-діаграма презентаційного шару тренувань

Рисунок 2.15 — UML-діаграма презентаційного шару тренувальних програм

Отже, UML-діаграми класів дозволяють наочно описати програмну структуру GanG Gym. Вони демонструють поділ системи на доменну модель, шар даних і презентаційний шар, а також показують основні зв’язки між класами. Це спрощує розуміння архітектури застосунку та створює основу для подальшого опису реалізації основних класів і методів.

2.5 Вибір мови та середовища розробки

Для реалізації програмної системи GanG Gym було обрано мову програмування Kotlin. Такий вибір зумовлений тим, що Kotlin є сучасною мовою, яка активно використовується для розробки Android-застосунків, серверних систем і кросплатформних рішень. Мова має лаконічний синтаксис, підтримує об’єктно-орієнтований і функціональний підходи, забезпечує null-safety та добре інтегрується з екосистемою Java. Це дозволяє створювати надійний, читабельний і підтримуваний програмний код.

Однією з головних причин вибору Kotlin є підтримка технології Kotlin Multiplatform. Вона дозволяє використовувати спільну кодову базу для різних платформ, зокрема Android, iOS та Desktop. У межах проєкту GanG Gym це дало змогу винести спільні моделі, репозиторії, доменну логіку, презентаційний шар та значну частину інтерфейсу користувача в один модуль. Завдяки цьому зменшується дублювання коду та спрощується подальша підтримка застосунку.

Для створення користувацького інтерфейсу використано Compose Multiplatform. Це декларативний фреймворк, який дозволяє будувати інтерфейс на основі стану застосунку. Такий підхід добре підходить для GanG Gym, оскільки екрани застосунку постійно реагують на зміну даних: список вправ оновлюється після створення або редагування вправи, тренувальна сесія змінюється під час введення результатів, а екран прогресу перераховує показники на основі збережених тренувань. Compose Multiplatform також дозволяє створювати єдиний стиль інтерфейсу для різних платформ.

Для реалізації серверної частини обрано фреймворк Ktor. Він також базується на Kotlin і добре підходить для створення легких REST API. У системі GanG Gym серверна частина використовується для обробки запитів до профілю користувача, вправ і тренувальних програм. Ktor дозволяє зручно налаштовувати маршрути, серіалізацію, автентифікацію та взаємодію з базою даних.

Для локального збереження даних у клієнтському застосунку використано Room і SQLite. SQLite є вбудованою базою даних, яка не потребує окремого сервера

та добре підходить для мобільних і настільних застосунків. Room забезпечує зручний рівень абстракції над SQLite, дозволяючи описувати таблиці у вигляді Kotlin-класів і працювати з ними через DAO-інтерфейси. Це спрощує роботу з локальними даними та зменшує ризик помилок під час написання SQL-запитів.

Для серверного збереження даних використано PostgreSQL. Ця система керування базами даних є надійною, поширеною та підходить для збереження структурованої інформації. У проєкті PostgreSQL використовується для профілів користувачів, вправ і тренувальних програм. Для керування змінами структури серверної бази даних застосовано Flyway, який дозволяє виконувати міграції та контролювати версії схеми.

Як основне середовище розробки використано IntelliJ IDEA. Воно має якісну підтримку Kotlin, Gradle, Kotlin Multiplatform, Compose Multiplatform і Ktor. Середовище надає інструменти для навігації по коду, автодоповнення, рефакторингу, запуску тестів і збирання проєкту. Для Android-частини також може використовуватися Android Studio, оскільки вона базується на IntelliJ IDEA та має розширені інструменти для роботи з Android SDK.

Для iOS-частини проєкту використовується Xcode, оскільки запуск і тестування iOS-застосунку потребує відповідного середовища Apple. У структурі GanG Gym iOS-модуль містить точку входу SwiftUI, яка підключає спільний Kotlin Multiplatform інтерфейс. Такий підхід дозволяє поєднати нативну iOS-оболонку зі спільною логікою та UI, реалізованими у Kotlin.

Для автоматизації збирання проєкту використано Gradle. Він керує залежностями, модулями, завданнями компіляції та запуском тестів. У проєкті Gradle використовується для збирання Android, Desktop, shared, core і server модулів. Завдяки цьому весь проєкт має єдину систему збирання, що спрощує підтримку та запуск окремих частин програмної системи.

Отже, вибір Kotlin, Kotlin Multiplatform, Compose Multiplatform, Ktor, Room, SQLite, PostgreSQL, Flyway, Gradle, IntelliJ IDEA, Android Studio та Xcode є обґрунтованим для розробки GanG Gym. Обраний технологічний стек дозволяє створити кросплатформний застосунок із локальним збереженням даних,

серверною частиною, зручним інтерфейсом і можливістю подальшого розвитку системи.

2.6 Реалізація основних класів та методів

Реалізація програмної системи GanG Gym виконана з урахуванням багатошарової архітектури, описаної у попередніх підрозділах. Основні класи системи поділено на доменні моделі, класи локального збереження даних, репозиторії, презентери користувацького інтерфейсу та серверні маршрути. Такий поділ дозволяє відокремити бізнес-логіку від інтерфейсу користувача і способу збереження даних.

Основу системи становлять доменні класи, розміщені у модулі core. До них належать Exercise, Workout, Routine, TrainingPlan, WorkoutSession та ExerciseResult. Клас Exercise описує окрему вправу та містить її назву, опис, категорії, обладнання, м’язові групи і рівень складності. Клас Workout описує тренування, що складається з набору блоків. Для опису блоків використано інтерфейс WorkoutBlock, який має кілька реалізацій: ExerciseBlock, RestBlock, RoutineBlock і SupersetBlock. Завдяки цьому тренування може містити не лише вправи, а й відпочинок, вкладені програми та складніші структури.

Клас Routine використовується для представлення тренувальної програми, яка може багаторазово використовуватися під час створення тренувань. Клас WorkoutSession описує фактичне виконання тренування користувачем. Він містить час початку і завершення заняття, примітки, самопочуття та список результатів вправ. Клас ExerciseResult зберігає результат одного підходу або виконання вправи: кількість повторень, вагу, тривалість та номер підходу.

Лістинг 2.1 — Фрагмент реалізації доменних класів тренування та тренувальної сесії

```
 data class Workout(
val id: WorkoutId,
```

```
    val ownerId: UserId,
    val name: String,
    val description: String = "",
    val difficulty: DifficultyLevel = DifficultyLevel.NOT_SET,
    val estimatedDurationSeconds: Int? = null,
    val visibility: WorkoutVisibility = WorkoutVisibility.PRIVATE,
    val source: WorkoutSource = WorkoutSource.CUSTOM,
    val blocks: List<WorkoutBlock> = emptyList(),
    val audit: AuditInfo,
)

sealed interface WorkoutBlock {
    val id: WorkoutBlockId
    val order: Int
    val note: String
}

data class ExerciseBlock(
    override val id: WorkoutBlockId,
    override val order: Int,
    override val note: String = "",
    val exerciseId: ExerciseId,
    val target: WorkoutTarget = WorkoutTarget(),
    val rounds: Int = 1,
    val restAfterSeconds: Int = 0,
) : WorkoutBlock

data class RestBlock(
    override val id: WorkoutBlockId,
    override val order: Int,
    override val note: String = "",
    val durationSeconds: Int,
) : WorkoutBlock

data class WorkoutTarget(
```

```
    val reps: Int? = null,
    val weightKg: Double? = null,
    val durationSeconds: Int? = null,
)

data class WorkoutSession(
    val id: WorkoutSessionId,
    val ownerId: UserId,
    val workoutId: WorkoutId? = null,
    val startedAt: TimestampMillis,
    val completedAt: TimestampMillis? = null,
    val feeling: WorkoutFeeling? = null,
    val notes: String = "",
    val results: List<ExerciseResult> = emptyList(),
)

data class ExerciseResult(
    val id: ExerciseResultId,
    val exerciseId: ExerciseId,
    val workoutBlockId: WorkoutBlockId? = null,
    val setNumber: Int,
    val reps: Int? = null,
    val weightKg: Double? = null,
    val durationSeconds: Int? = null,
    val completedAt: TimestampMillis? = null,
    val note: String = "",
)
```

Для збереження даних у локальній базі використовуються entity-класи Room. Наприклад, WorkoutEntity зберігає основну інформацію про тренування, а WorkoutBlockEntity описує окремі блоки цього тренування. Оскільки доменна модель і структура бази даних не повністю збігаються, у системі реалізовано mapper-функції. Вони перетворюють entity-об’єкти у доменні моделі та навпаки. Наприклад, метод WorkoutWithBlocks.toDomain() формує об’єкт Workout разом із

його блоками, а метод Workout.toBlockEntities() перетворює блоки тренування у список записів для таблиці workout_blocks.

Окрему увагу приділено реалізації різних типів блоків тренування. Під час перетворення WorkoutBlockEntity у доменну модель система перевіряє значення поля type і створює відповідний клас: ExerciseBlock, RestBlock, RoutineBlock або SupersetBlock. Для суперсетів додатково обробляються дочірні блоки, що дозволяє зберігати складні тренувальні структури в реляційній базі даних.

Лістинг 2.2 — Фрагмент перетворення блоків тренування між Room Entity та доменною моделлю

```
     private const val BLOCK_TYPE_EXERCISE = "EXERCISE"
private const val BLOCK_TYPE_REST = "REST"
private const val BLOCK_TYPE_ROUTINE = "ROUTINE"
private const val BLOCK_TYPE_SUPERSET = "SUPERSET"

fun WorkoutWithBlocks.toDomain(): Workout =
    workout.toDomain(
        blocks = blocks.toDomainBlocks(parentBlockId = null),
    )

private fun List<WorkoutBlockEntity>.toDomainBlocks(
    parentBlockId: String?,
): List<WorkoutBlock> =
    filter { it.parentBlockId == parentBlockId }
        .sortedBy { it.order }
        .mapNotNull { entity ->
            entity.toDomainBlock(
                childBlocks    =    toDomainBlocks(parentBlockId    =
entity.id),
            )
        }

private fun WorkoutBlockEntity.toDomainBlock(
```

```
    childBlocks: List<WorkoutBlock>,
): WorkoutBlock? =
    when (type) {
        BLOCK_TYPE_EXERCISE -> ExerciseBlock(
            id = WorkoutBlockId(id),
            order = order,
            note = note,
            exerciseId = ExerciseId(requireNotNull(exerciseId)),
            target = WorkoutTarget(
                reps = targetReps,
                weightKg = targetWeightKg,
                durationSeconds = targetDurationSeconds,
            ),
            rounds = rounds ?: 1,
            restAfterSeconds = restAfterSeconds ?: 0,
        )

        BLOCK_TYPE_REST -> RestBlock(
            id = WorkoutBlockId(id),
            order = order,
            note = note,
            durationSeconds = durationSeconds ?: 0,
        )

        BLOCK_TYPE_ROUTINE -> RoutineBlock(
            id = WorkoutBlockId(id),
            order = order,
            note = note,
            routineId = RoutineId(requireNotNull(routineId)),
        )

        BLOCK_TYPE_SUPERSET -> SupersetBlock(
            id = WorkoutBlockId(id),
            order = order,
            note = note,
```

```
            name = name.orEmpty(),
            rounds = rounds ?: 1,
            restBetweenRoundsSeconds = restBetweenRoundsSeconds ?: 0,
            exercises                                                =
childBlocks.filterIsInstance<ExerciseBlock>(),
        )

        else -> null
    }
```

Доступ до локальної бази даних організовано через DAO-інтерфейси та репозиторії. DAO-класи містять SQL-запити для отримання, вставки, оновлення та видалення записів. Репозиторії, у свою чергу, приховують деталі роботи з DAO і надають зручний інтерфейс для інших частин застосунку. Наприклад, LocalWorkoutSessionRepository реалізує методи observeSessions(), getSession() і saveSession(). Під час збереження тренувальної сесії репозиторій спочатку зберігає саму сесію, потім видаляє старі результати цієї сесії та записує оновлений список результатів вправ.

Такий підхід дозволяє уникнути дублювання застарілих результатів і гарантує, що кожна тренувальна сесія має актуальний набір виконаних вправ. Також у репозиторіях використовується стан синхронізації, наприклад PENDING_CREATE, PENDING_UPDATE або PENDING_DELETE, що створює основу для подальшої синхронізації локальних і серверних даних.

Лістинг 2.3 — Реалізація методу збереження тренувальної сесії

```
 class LocalWorkoutSessionRepository(
    private val workoutSessionDao: WorkoutSessionDao,
) : WorkoutSessionRepository {

    override fun observeSessions(): Flow<List<WorkoutSession>> =
        workoutSessionDao.observeSessions()
            .map { sessions ->
                sessions.map { it.toDomain() }
```

```
            }

    override suspend fun getSession(
        id: WorkoutSessionId,
    ): WorkoutSession? =
        workoutSessionDao.getSessionById(id.value)?.toDomain()

    override suspend fun saveSession(
        session: WorkoutSession,
    ) {
        val syncState =
            if (workoutSessionDao.getSessionById(session.id.value) ==
null) {
                LocalSyncState.PENDING_CREATE
            } else {
                LocalSyncState.PENDING_UPDATE
            }

        workoutSessionDao.upsertSession(
            session.toEntity(syncState = syncState),
        )

        workoutSessionDao.deleteResultsForSession(session.id.value)

        workoutSessionDao.upsertResults(
            session.toResultEntities(),
        )
    }
}
```

Презентаційний шар реалізовано за допомогою presenter-класів. Вони отримують дані з use case та репозиторіїв, формують стан екрана і реагують на дії користувача. Наприклад, ExerciseCatalogPresenter відповідає за каталог вправ: обробляє пошук, фільтрацію, вибір вправи, створення нового запису та

редагування існуючого. Його результатом є ExerciseCatalogUiState, який передається на екран ExerciseCatalogScreen.

Клас WorkoutListPresenter має ширшу відповідальність, оскільки керує списком тренувань, редактором тренування, запуском тренувальної сесії, введенням результатів і збереженням завершеної сесії. Під час запуску тренування presenter створює чернетку сесії, формує список результатів для вправ і дозволяє користувачу змінювати фактичні значення повторень, ваги та тривалості. Після завершення заняття ці дані перетворюються у доменний клас WorkoutSession і передаються до репозиторію для збереження.

Окремим прикладом презентаційної логіки є CalendarPresenter. Він поєднує дані про тренування, вправи та виконані сесії, після чого формує стан календаря. Метод побудови стану групує тренувальні сесії за датами, визначає вибраний день, підраховує кількість тренувань і формує список результатів для відображення користувачу. Завдяки цьому календар не зберігає окрему дубльовану інформацію, а будується на основі вже наявної історії тренувань.

Лістинг 2.4 — Фрагмент формування стану календаря тренувань

```
 private fun buildCalendarUiState(
    sessions: List<WorkoutSession>,
    workouts: List<Workout>,
    exercises: List<Exercise>,
    month: CalendarMonth,
    selected: CalendarDate,
    selectedSessionId: String?,
    today: CalendarDate,
): CalendarUiState {
    val sessionDateCounts = sessions
        .groupingBy { session ->
            session.displayTimestampMillis()
                .toCalendarDate()
                .key
        }
```

```
            .eachCount()

        val selectedSessions = sessions
            .filter { session ->
                session.displayTimestampMillis()
                    .toCalendarDate() == selected
            }
            .sortedBy { it.displayTimestampMillis() }

        val workoutNames = workouts.associateBy(
            keySelector = { it.id.value },
            valueTransform = { it.name },
        )

        val exerciseNames = exercises.associateBy(
            keySelector = { it.id.value },
            valueTransform = { it.name },
        )

        return CalendarUiState(
            isLoading = false,
            monthTitle = month.title,
            selectedDateLabel = selected.longLabel,
            selectedWorkoutCountLabel = "Workouts:
${selectedSessions.size}",
            days = month.days(
                today = today,
                selected = selected,
                sessionDateCounts = sessionDateCounts,
            ),
            selectedWorkouts = selectedSessions.map { session ->
                session.toUiModel(workoutNames = workoutNames)
            },
            selectedWorkout = selectedSessionId
                ?.let { id ->
```

```
                sessions.firstOrNull { session -> session.id.value ==
id }
            }
            ?.toDetailUiModel(
                workoutNames = workoutNames,
                exerciseNames = exerciseNames,
            ),
    )
}
```

Серверна частина реалізована за допомогою Ktor. Основні маршрути розділено за функціональними модулями: профіль користувача, вправи та тренувальні програми. Наприклад, у маршрутах вправ реалізовано отримання списку вправ, отримання конкретної вправи за ідентифікатором, створення або оновлення вправи та м’яке видалення. Перед виконанням захищених операцій сервер перевіряє автентифікованого користувача. Якщо дані запиту некоректні, сервер повертає відповідь із помилкою, наприклад для порожнього ідентифікатора, порожньої назви вправи або недопустимого рівня складності.

Лістинг 2.5 — Фрагмент реалізації серверних маршрутів для роботи з вправами

```
 fun Route.exerciseRoutes() {
route("/exercises") {
    get {
        val user = call.requireAuthenticatedUser() ?: return@get

        call.respond(
            call.application.exerciseRepository()
                .listForUser(user.id)
                .map { it.toResponse() },
        )
    }
```

```
        get("/{id}") {
            val user = call.requireAuthenticatedUser() ?: return@get
            val exerciseId = call.parameters["id"].orEmpty()

            val exercise = call.application.exerciseRepository()
                .findForUser(user.id, exerciseId)

            if (exercise == null) {
                call.respond(
                    status = HttpStatusCode.NotFound,
                    message = ErrorResponse(
                        code = "exercise_not_found",
                        message = "Exercise was not found.",
                    ),
                )
                return@get
            }

            call.respond(exercise.toResponse())
        }

        put("/{id}") {
            val user = call.requireAuthenticatedUser() ?: return@put
            val exerciseId = call.parameters["id"].orEmpty()
            val request = call.receive<UpsertExerciseRequest>()

            val validationError = request.validate(exerciseId)
            if (validationError != null) {
                call.respond(HttpStatusCode.BadRequest,
validationError)
                return@put
            }

            val exercise = request.toRecord(
                id = exerciseId,
```

```
                ownerId = user.id,
                now = Instant.now(),
            )

            val                    updatedExercise                    =
call.application.exerciseRepository()
                .upsertForUser(user.id, exercise)

            call.respond(updatedExercise.toResponse())
        }

        delete("/{id}") {
            val    user    =    call.requireAuthenticatedUser()    ?:
return@delete
            val exerciseId = call.parameters["id"].orEmpty()

            val deleted = call.application.exerciseRepository()
                .softDeleteForUser(user.id, exerciseId)

            if (!deleted) {
                call.respond(
                    status = HttpStatusCode.NotFound,
                    message = ErrorResponse(
                        code = "exercise_not_found",
                        message = "Exercise was not found.",
                    ),
                )
                return@delete
            }

            call.respond(HttpStatusCode.NoContent)
        }
    }
}
```

Таким чином, реалізація основних класів і методів GanG Gym побудована навколо чіткого розділення відповідальностей. Доменні класи описують предметну область, entity-класи відповідають за структуру локальної бази даних, mapper-функції виконують перетворення даних, репозиторії організовують доступ до сховища, презентери формують стан інтерфейсу, а серверні маршрути забезпечують роботу REST API. Така структура робить програмний продукт зрозумілим, підтримуваним і готовим до подальшого розширення.

2.7 Розробка інтерфейсу користувача

Розробка інтерфейсу користувача є важливим етапом створення програмної системи GanG Gym, оскільки саме через графічний інтерфейс користувач взаємодіє з основними функціями застосунку. Під час проєктування інтерфейсу було враховано, що система призначена для швидкого використання під час тренування, тому екрани мають бути зрозумілими, не перевантаженими зайвою інформацією та зручними для навігації.

Інтерфейс застосунку реалізовано з використанням Compose Multiplatform. Це дозволило створити спільні екрани для різних платформ і забезпечити однакову логіку відображення даних на Android, iOS та Desktop. Основою побудови інтерфейсу є декларативний підхід, за якого зовнішній вигляд екрана залежить від поточного стану UiState. Коли змінюються дані, наприклад список вправ або результати тренувальної сесії, інтерфейс автоматично оновлюється відповідно до нового стану.

Головний екран застосунку виконує роль навігаційного центру. З нього користувач може перейти до каталогу вправ, списку тренувань, тренувальних програм, календаря, прогресу або профілю. У нижній частині інтерфейсу розміщено навігаційну панель, яка дає змогу швидко перемикатися між основними розділами. Така структура робить застосунок зручним для щоденного використання, оскільки користувач не витрачає час на пошук потрібної функції.

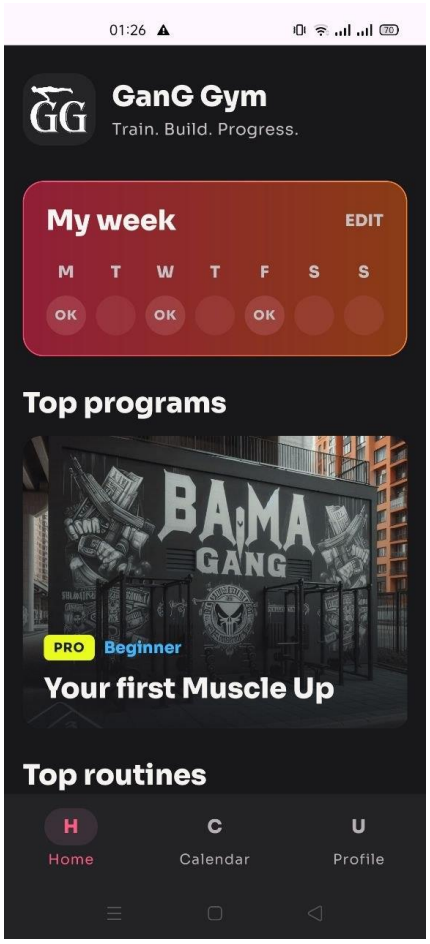

Рисунок 2.16 — Головний екран застосунку

Екран каталогу вправ призначений для перегляду, пошуку та редагування вправ. Користувач бачить список доступних вправ із коротким описом, рівнем складності, категорією та обладнанням. Для зручності передбачено пошукове поле та фільтри, які дозволяють швидко знайти потрібну вправу за категорією, обладнанням або складністю. Після вибору вправи відкривається детальна інформація, де відображаються опис, м’язові групи та додаткові характеристики.

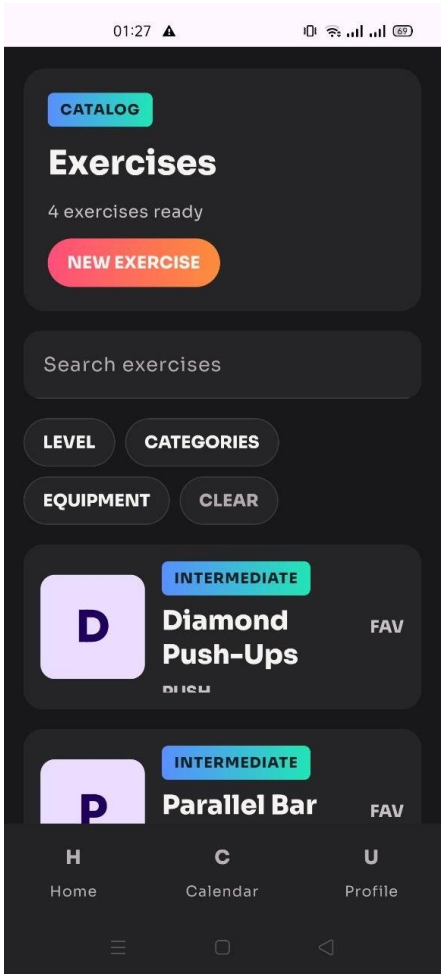

Рисунок 2.17 — Інтерфейс каталогу вправ

Окремо реалізовано режим створення та редагування вправи. У цьому режимі користувач може ввести назву, опис, посилання на зображення, обрати категорію, обладнання, основні та додаткові м’язові групи, а також рівень складності. Такий підхід дозволяє не обмежуватися стандартним набором вправ, а формувати власний каталог відповідно до індивідуальних потреб користувача.

Екран тренувань дає змогу переглядати створені тренування, відкривати детальну інформацію, редагувати структуру тренування та запускати тренувальну сесію. У режимі редагування користувач може додавати вправи, блоки відпочинку або тренувальні програми. Для кожної вправи передбачено налаштування кількості раундів, повторень, ваги, тривалості та часу відпочинку. Це дозволяє гнучко формувати тренування різного рівня складності.

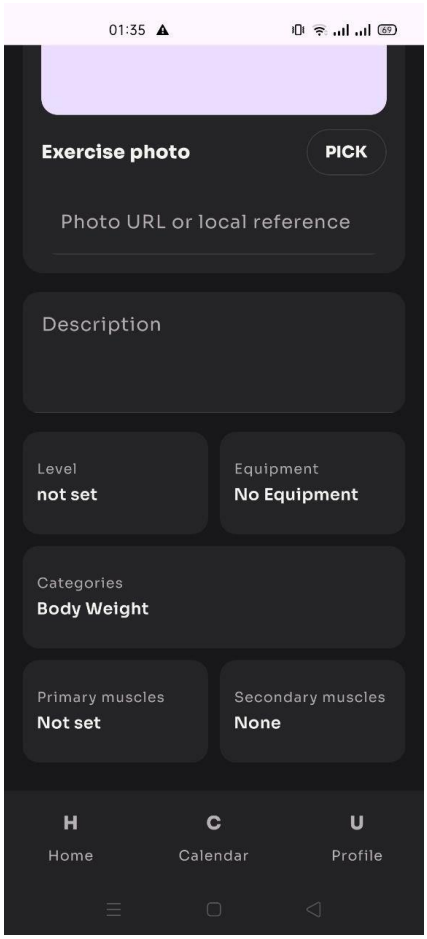

Рисунок 2.18 — Інтерфейс створення та редагування тренування

Під час запуску тренувальної сесії інтерфейс змінюється відповідно до сценарію виконання тренування. Користувач бачить список вправ і може вводити фактичні результати: кількість повторень, робочу вагу або тривалість виконання. Після завершення тренування дані зберігаються в історії та використовуються для подальшого аналізу прогресу. Такий екран має бути максимально простим, оскільки користувач взаємодіє з ним безпосередньо під час фізичного навантаження.

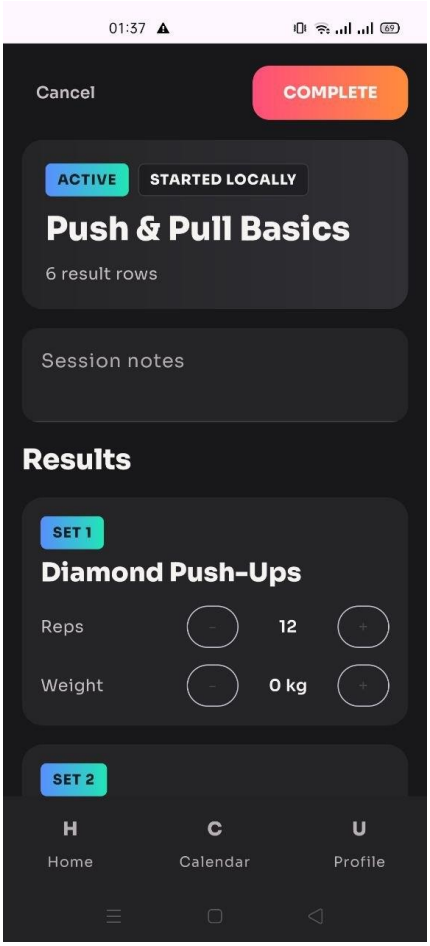

Рисунок 2.19 — Інтерфейс виконання тренувальної сесії

Екран тренувальних програм призначений для створення багаторазових шаблонів занять. Його структура подібна до екрана тренувань, однак основний акцент зроблено на формуванні послідовності вправ і відпочинку, яку можна використовувати повторно. Це спрощує організацію регулярних тренувань і дозволяє користувачу швидко додавати готові блоки до нових занять.

Календар тренувань використовується для перегляду виконаних занять за датами. На екрані відображається поточний місяць, вибрана дата та список тренувальних сесій за цей день. Для кожної сесії показується назва тренування, час виконання, кількість підходів, загальна кількість повторень і примітки. Такий інтерфейс дозволяє користувачу оцінити регулярність занять і швидко знайти потрібне тренування в історії.

Рисунок 2.20 — Інтерфейс календаря тренувань

Екран прогресу призначений для аналізу результатів користувача. На ньому відображаються показники за вправами: загальна кількість підходів, повторень, найкраща зафіксована вага, сумарний час виконання та останній результат. Також може відображатися графік зміни показників, що робить аналіз тренувальної динаміки більш наочним. Цей розділ допомагає користувачу оцінювати ефективність тренувань і коригувати подальше навантаження.

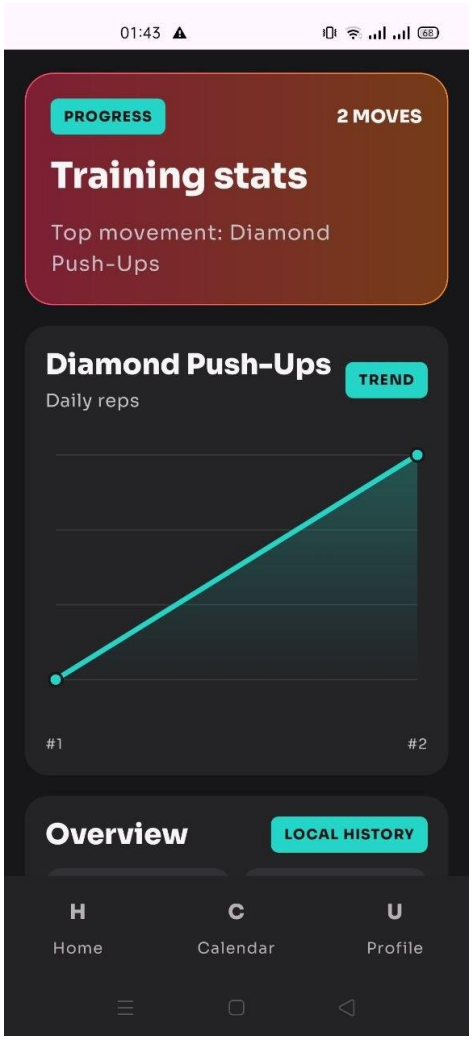

Рисунок 2.21 — Інтерфейс перегляду прогресу користувача

Екран профілю містить персональні дані користувача: ім’я, нікнейм, стать, дату народження, зріст, електронну пошту та фото. Користувач може переглядати та змінювати ці дані. Наявність профілю дозволяє персоналізувати застосунок і створює основу для подальшої реалізації авторизації та синхронізації між пристроями.

Під час розробки інтерфейсу важливу роль відіграє узгодженість візуального стилю. У застосунку використано єдину тему оформлення, однакові принципи побудови екранів, повторювані елементи керування та зрозумілі підписи. Завдяки цьому користувач швидко звикає до логіки застосунку й може легко переходити між різними розділами.

Отже, інтерфейс користувача GanG Gym розроблено з урахуванням зручності, простоти навігації та потреб користувача під час тренувального процесу. Реалізація за допомогою Compose Multiplatform дала змогу створити спільний кросплатформний інтерфейс, який взаємодіє з презентаційним шаром через UiState-моделі та забезпечує швидке оновлення даних на екрані.

2.8 Висновки до розділу 2

У другому розділі було виконано проєктування та розробку програмної системи GanG Gym, призначеної для планування, виконання та моніторингу тренувального процесу. На основі вимог, сформульованих у першому розділі, визначено загальну архітектуру системи, структуру клієнтської та серверної частин, схеми баз даних, основні класи, методи та принципи побудови користувацького інтерфейсу.

Для реалізації застосунку було обрано ітеративний процес розробки, що дозволило поступово створювати та перевіряти окремі функціональні модулі. Такий підхід дав змогу поетапно реалізувати каталог вправ, тренування, тренувальні програми, тренувальні сесії, календар, прогрес, профіль користувача та серверну частину.

Було спроєктовано модульну архітектуру системи, у якій клієнтська частина реалізована з використанням Kotlin Multiplatform і Compose Multiplatform, а серверна частина — на основі Ktor. Клієнтський застосунок містить спільну бізнес-логіку, презентаційний шар, інтерфейс користувача та локальну базу даних. Серверна частина забезпечує REST API, автентифікацію та роботу з централізованою базою PostgreSQL.

У межах розділу побудовано схеми локальної та серверної баз даних. Локальна база даних Room/SQLite використовується для збереження вправ, тренувань, тренувальних програм, блоків, тренувальних сесій і результатів виконання вправ. Серверна база даних PostgreSQL містить профілі користувачів,

вправи, тренувальні програми та їхні блоки. Для керування змінами серверної схеми використано Flyway-міграції.

Також було побудовано UML-діаграми класів, які відображають доменну модель, шар даних і презентаційний шар. Через велику кількість класів діаграми було розділено на декілька частин, що дозволило краще показати окремі компоненти системи: вправи та тренування, тренувальні програми, тренувальні сесії, локальні репозиторії, Room DAO, серверний шар даних і презентери інтерфейсу.

Реалізація основних класів і методів виконана з дотриманням принципу розділення відповідальностей. Доменні класи описують предметну область, entity-класи відповідають за структуру локальної бази даних, mapper-функції виконують перетворення між моделями, репозиторії організовують доступ до даних, презентери формують стан інтерфейсу, а серверні маршрути забезпечують обробку API-запитів.

Для створення інтерфейсу користувача використано Compose Multiplatform. Було розроблено екрани головної навігації, каталогу вправ, тренувань, тренувальних програм, виконання тренувальної сесії, календаря, прогресу та профілю користувача. Інтерфейс побудовано на основі UiState-моделей, що забезпечує реактивне оновлення даних і зрозумілу взаємодію користувача із застосунком.

Отже, у другому розділі було сформовано технічну основу програмної системи GanG Gym. Спроєктована архітектура, структура баз даних, UML-моделі, вибраний технологічний стек та реалізовані програмні компоненти забезпечують можливість стабільної роботи застосунку, його подальшого тестування та розширення функціональності.

3 ТЕСТУВАННЯ ТА ВЕРИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У цьому розділі розглянуто процес тестування та верифікації програмної системи GanG Gym. Після реалізації основних модулів застосунку необхідно перевірити, чи відповідає розроблене програмне забезпечення вимогам, визначеним у першому розділі, а також чи коректно працюють основні сценарії взаємодії користувача із системою.

Тестування програмної системи GanG Gym охоплює перевірку клієнтської та серверної частин. Для клієнтського застосунку важливо перевірити роботу каталогу вправ, створення і редагування тренувань, формування тренувальних програм, запуск тренувальної сесії, збереження результатів, відображення календаря, перегляд прогресу та роботу профілю користувача. Для серверної частини необхідно перевірити коректність API-маршрутів, автентифікацію, обробку помилкових запитів і взаємодію з базою даних.

Основною метою тестування є виявлення помилок у роботі системи, перевірка стабільності збереження даних, оцінка відповідності функціональним і нефункціональним вимогам, а також підтвердження того, що застосунок може бути використаний за призначенням. Особливу увагу приділено сценаріям, які є критичними для користувача: створення вправи, формування тренування, виконання сесії та перегляд результатів.

У межах цього розділу буде описано види та план тестування, проведено функціональне тестування основних модулів системи, розглянуто автоматизоване тестування, наведено вимоги до розгортання програмного забезпечення та виконано верифікацію відповідності готового продукту поставленим вимогам. Результати тестування дозволяють оцінити якість реалізації GanG Gym і підтвердити готовність системи до практичного використання.

3.1 Тестування програмної системи

Тестування програмної системи GanG Gym є важливим етапом перевірки якості розробленого програмного забезпечення. Його основна мета полягає у встановленні відповідності застосунку функціональним і нефункціональним вимогам, визначеним на етапі аналізу, а також у виявленні можливих помилок у роботі основних модулів системи.

Оскільки GanG Gym складається з клієнтської частини, локальної бази даних і серверного API, тестування охоплює декілька напрямів. Перевіряється коректність роботи інтерфейсу користувача, збереження та отримання даних із локальної бази Room/SQLite, формування станів екранів, обробка тренувальних сесій, робота календаря та прогресу, а також функціонування серверних маршрутів Ktor.

Під час тестування особлива увага приділяється ключовим сценаріям використання: створенню вправи, формуванню тренування, створенню тренувальної програми, запуску тренувальної сесії, збереженню результатів, перегляду календаря та аналізу прогресу. Саме ці сценарії визначають основну цінність програмного продукту для користувача.

3.1.1 Види та план тестування

Для перевірки програмної системи GanG Gym було застосовано кілька видів тестування. Такий підхід дозволяє оцінити не лише окремі функції застосунку, а й загальну стабільність системи під час виконання типових сценаріїв користувача.

Першим видом є функціональне тестування. Воно спрямоване на перевірку того, чи правильно система виконує основні функції, передбачені вимогами. У межах функціонального тестування перевіряється робота каталогу вправ, пошуку та фільтрів, створення і редагування вправ, створення тренувань і програм, запуск тренувальної сесії, введення результатів, перегляд календаря та показників прогресу.

Другим видом є тестування локального збереження даних. Воно передбачає перевірку коректності роботи з базою даних Room/SQLite. Під час цього

тестування перевіряється, чи зберігаються створені вправи, тренування, програми та результати тренувальних сесій, чи не зникають вони після перезапуску застосунку, а також чи правильно оновлюються або видаляються записи.

Третім видом є тестування серверної частини. Воно охоплює перевірку REST API, реалізованого за допомогою Ktor. Перевіряються маршрути для роботи з профілем користувача, вправами та тренувальними програмами, коректність автентифікації, обробка некоректних запитів, повернення відповідних HTTP-статусів і взаємодія з базою даних PostgreSQL.

Четвертим видом є автоматизоване тестування. Воно використовується для перевірки окремих частин програмної логіки без ручного проходження всіх сценаріїв. Автоматизовані тести дозволяють швидше виявляти регресії після внесення змін у код. У проєкті передбачено тести для спільного модуля та серверної частини, що дає змогу перевіряти логіку застосунку на різних рівнях.

Також було враховано тестування зручності використання. Його метою є перевірка того, наскільки зрозуміло користувачу взаємодіяти із застосунком. Під час такого тестування оцінюється логіка навігації, доступність основних дій, зрозумілість форм створення вправ і тренувань, а також зручність введення результатів під час виконання тренувальної сесії.

План тестування програмної системи GanG Gym включає послідовну перевірку основних модулів. Спочатку перевіряється запуск застосунку та відображення головного екрана. Далі тестується каталог вправ: перегляд списку, пошук, фільтрація, створення та редагування вправ. Після цього перевіряється модуль тренувань: створення тренування, додавання блоків, редагування параметрів і збереження. Наступним етапом тестується створення тренувальних програм і повторне використання їх у тренуваннях.

Після перевірки шаблонів виконується тестування тренувальної сесії. Користувач запускає тренування, вводить фактичні результати вправ, завершує сесію та перевіряє, чи збережено результати. Далі перевіряється календар тренувань, де виконана сесія має відображатися у відповідній даті. Після цього

тестується екран прогресу, який повинен коректно розраховувати показники на основі збережених результатів.

Окремим етапом плану є перевірка серверної частини. Для цього виконуються запити до API профілю, вправ і тренувальних програм. Перевіряються як успішні сценарії, так і помилкові ситуації: відсутність автентифікації, неправильний ідентифікатор запису, порожня назва вправи або недопустиме значення рівня складності.

Після завершення тестування результати порівнюються з функціональними та нефункціональними вимогами. Якщо система виконує основні сценарії, коректно зберігає дані, стабільно працює на підтримуваних платформах і повертає очікувані відповіді API, програмний продукт можна вважати таким, що пройшов базову перевірку якості.

3.1.2 Функціональне тестування

Функціональне тестування GanG Gym проводилося для перевірки відповідності реалізованих можливостей функціональним вимогам. Основну увагу приділено ключовим сценаріям роботи користувача: перегляду каталогу вправ, створенню тренувань і програм, запуску тренувальної сесії, збереженню результатів, перегляду календаря та аналізу прогресу.

Спочатку було перевірено запуск застосунку та навігацію між основними розділами. Після відкриття системи користувач має доступ до головного екрана, каталогу вправ, тренувань, програм, календаря, прогресу та профілю. Переходи між екранами виконуються коректно.

Далі протестовано каталог вправ. Перевірено перегляд списку, пошук, фільтрацію, відкриття детальної інформації, створення та редагування вправ. У разі введення коректних даних вправа зберігається у каталозі, а при порожній назві система не дозволяє виконати збереження.

Наступним етапом перевірено створення тренувань і тренувальних програм. Користувач може додавати вправи, блоки відпочинку, змінювати кількість

повторень, вагу, тривалість і час відпочинку. Створені тренування та програми зберігаються і доступні для повторного використання.

Окремо протестовано тренувальну сесію. Після запуску тренування система формує список вправ для виконання, користувач вводить фактичні результати, а після завершення сесії дані зберігаються у локальній базі. Надалі ці результати відображаються в історії, календарі та на екрані прогресу.

Також перевірено серверну частину. Для API вправ протестовано отримання списку, отримання вправи за ідентифікатором, створення або оновлення запису та видалення. У разі некоректних даних сервер повертає відповідні помилки, наприклад 400 Bad Request або 404 Not Found.

3.1.3 Навантажувальне тестування

Навантажувальне тестування програмної системи GanG Gym проводилося з метою перевірки стабільності роботи застосунку та серверної частини під час збільшення кількості запитів і обсягу даних. Оскільки основна частина функціональності застосунку працює локально, особливу увагу приділено швидкості відкриття екранів, роботі зі списками вправ і тренувань, збереженню тренувальних сесій та обробці запитів серверного API.

Для клієнтської частини було перевірено поведінку застосунку при збільшенні кількості записів у локальній базі даних. Тестування включало роботу з розширеним каталогом вправ, створенням кількох тренувань, додаванням великої кількості блоків до тренування та збереженням декількох тренувальних сесій. Під час перевірки застосунок продовжував коректно відкривати основні екрани, виконувати пошук, фільтрацію та відображати результати без критичних затримок.

Окремо перевірено модуль тренувальної сесії. Під час тестування до тренування додавалася значна кількість вправ і підходів, після чого запускалася сесія та зберігалися результати. Система коректно формувала список результатів, записувала їх у локальну базу даних і надалі використовувала для відображення в історії, календарі та модулі прогресу.

Для серверної частини навантажувальне тестування виконувалося шляхом багаторазового надсилання запитів до API вправ і тренувальних програм. Перевірялися сценарії отримання списку записів, отримання конкретного запису за ідентифікатором, створення або оновлення запису та м’яке видалення. Серверна частина на базі Ktor коректно обробляла послідовні запити, повертала очікувані HTTP-статуси та не втрачала цілісність даних.

Під час тестування також враховано роботу бази даних PostgreSQL. Завдяки використанню індексів для основних полів, зокрема ідентифікатора власника запису, сервер може ефективно отримувати дані конкретного користувача. Механізм Flyway-міграцій забезпечує стабільну структуру бази даних і дозволяє уникнути невідповідності між кодом серверної частини та схемою збереження даних.

Результати навантажувального тестування показали, що GanG Gym стабільно працює в умовах типового навантаження, характерного для персонального фітнес-застосунку. Локальна база даних забезпечує швидке збереження і читання тренувальних даних, а серверна частина коректно обробляє основні API-запити. Отже, продуктивність системи є достатньою для практичного використання та подальшого розширення функціональності.

3.1.4 Автоматизоване тестування

Автоматизоване тестування GanG Gym використано для перевірки окремих частин програмної логіки без необхідності ручного проходження всіх сценаріїв. Такий підхід дозволяє швидко перевіряти працездатність системи після внесення змін у код і зменшує ризик появи регресій.

Основну частину автоматизованих тестів реалізовано для серверного модуля. Для цього використано тестове середовище Ktor, яке дає змогу запускати застосунок у тестовому режимі та виконувати HTTP-запити до API без окремого розгортання сервера. У тестах перевіряються базові маршрути /, /health і /api/v1/health, які повинні повертати успішний статус і службову інформацію про API.

Окремо протестовано маршрути профілю користувача. Перевірено, що запит до /api/v1/me без автентифікаційних даних повертає помилку доступу, а запит із заголовком X-User-Id повертає профіль користувача. Також перевірено оновлення профілю та обробку некоректної дати народження.

Для модуля вправ автоматизовані тести перевіряють створення вправи, отримання списку вправ, завантаження вправи за ідентифікатором, відхилення порожньої назви та заборону перезапису вправи іншим користувачем. Аналогічні перевірки реалізовано для тренувальних програм, тренувань і тренувальних сесій. У тестах перевіряється не лише успішне створення записів, а й обробка помилкових даних, наприклад неправильного типу блоку, недопустимого значення доступності або некоректного часу завершення сесії.

Також реалізовано тест для перевірки конфігурації автентифікації. Він підтверджує, що Firebase-провайдер не може бути створений без необхідних облікових даних. Це дозволяє виявити помилки конфігурації ще на етапі запуску тестів.

Для спільного клієнтського модуля передбачено базові тести, які перевіряють працездатність тестового середовища для common та desktop-частин. Надалі цей набір може бути розширений тестами для доменної логіки, mapper-функцій, репозиторіїв і presenter-класів.

Автоматизовані тести запускаються за допомогою Gradle. Для серверної частини використовується команда ./gradlew :server:test, а для спільного desktop-модуля — ./gradlew :app:shared:jvmTest. Виконання цих тестів дає змогу перевірити, що основні API-сценарії, автентифікація та базова логіка проєкту працюють очікувано.

Отже, автоматизоване тестування підтвердило працездатність основних серверних маршрутів, коректність обробки помилкових запитів і базову готовність клієнтського спільного модуля до подальшого розширення тестового покриття.

3.2 Розгортання програмної системи та системні вимоги

Розгортання програмної системи GanG Gym залежить від того, яка частина застосунку запускається: клієнтська або серверна. Проєкт має модульну структуру і складається з Android-застосунку, iOS-застосунку, Desktop-версії, спільного Kotlin Multiplatform модуля, core-модуля та серверної частини на базі Ktor. Збирання і запуск окремих частин системи виконується за допомогою Gradle.

Для запуску Android-версії необхідно мати встановлене середовище Android Studio або IntelliJ IDEA з підтримкою Android SDK. Android-застосунок збирається за допомогою Gradle-команди ./gradlew :app:androidApp:assembleDebug. Після збирання застосунок може бути встановлений на емулятор або фізичний Android-пристрій. Мінімальна підтримувана версія Android визначається параметром minSdk, а цільова версія — параметром targetSdk у конфігурації проєкту.

Desktop-версія застосунку запускається на JVM і може використовуватися на персональному комп’ютері. Для її запуску необхідно мати встановлений JDK та Gradle Wrapper, який уже входить до структури проєкту. Desktop-застосунок можна запустити командою ./gradlew :app:desktopApp:run. Для режиму активної розробки може використовуватися команда hot reload, яка дозволяє швидше перевіряти зміни інтерфейсу.

Для запуску iOS-версії необхідне середовище Xcode, оскільки збірка і запуск iOS-застосунків виконуються через інструменти Apple. У структурі проєкту передбачено окремий модуль iosApp, який містить точку входу для iOS-застосунку та підключає спільну Kotlin Multiplatform логіку.

Серверна частина GanG Gym реалізована на базі Ktor і запускається окремо від клієнтських застосунків. Для локального запуску сервера використовується команда ./gradlew :server:run. У режимі розробки сервер може працювати без підключення Firebase та PostgreSQL, використовуючи спрощений механізм автентифікації через заголовок X-User-Id. Це зручно для локального тестування API та перевірки основних маршрутів.

Для повноцінного розгортання серверної частини необхідно налаштувати змінні середовища. До них належать параметри вибору провайдера автентифікації, дані для підключення до Firebase, рядок підключення до PostgreSQL, ім’я користувача бази даних, пароль і додаткові параметри пулу з’єднань. Для керування структурою серверної бази даних використовується Flyway, який автоматично застосовує міграції та створює необхідні таблиці.

Мінімальні системні вимоги для розробки та запуску проєкту включають персональний комп’ютер з операційною системою Windows, macOS або Linux, встановлений JDK, Gradle Wrapper, IntelliJ IDEA або Android Studio, Android SDK для Android-версії та Xcode для iOS-версії. Для серверної частини у розгорнутому середовищі додатково потрібна база даних PostgreSQL і доступ до змінних середовища.

Для користувача клієнтської частини основними вимогами є наявність пристрою з підтримуваною операційною системою та достатнім обсягом пам’яті для встановлення застосунку і збереження локальних даних. Оскільки основна інформація про тренування зберігається локально, застосунок може виконувати базові функції без постійного підключення до інтернету. Мережеве з’єднання необхідне лише для функцій, пов’язаних із серверним API та майбутньою синхронізацією.

Отже, система GanG Gym може бути розгорнута як локально для розробки й тестування, так і в більш повному середовищі з серверною базою даних і автентифікацією. Використання Gradle, Kotlin Multiplatform, Ktor, Room, SQLite та PostgreSQL забезпечує гнучкість розгортання, підтримку кількох платформ і можливість подальшого масштабування програмного продукту.

3.3 Верифікація програмної системи

Верифікація програмної системи GanG Gym проводилася з метою перевірки відповідності реалізованого програмного продукту вимогам, визначеним на етапі аналізу. На відміну від тестування окремих функцій, верифікація спрямована на

загальну оцінку того, чи правильно система реалізує поставлене завдання та чи може використовуватися за призначенням.

Під час верифікації було проаналізовано основні функціональні вимоги до системи. Застосунок забезпечує роботу з каталогом вправ, дозволяє створювати та редагувати вправи, формувати тренування, додавати до них вправи й блоки відпочинку, створювати тренувальні програми та використовувати їх повторно. Також реалізовано запуск тренувальної сесії, введення фактичних результатів, збереження історії занять, перегляд календаря та аналіз прогресу користувача.

Було перевірено відповідність системи вимогам до збереження даних. Локальна база даних Room/SQLite забезпечує збереження вправ, тренувань, програм, тренувальних сесій і результатів виконання вправ. Дані не втрачаються після перезапуску застосунку та можуть використовуватися іншими модулями системи, наприклад календарем або екраном прогресу.

Окремо перевірено відповідність серверної частини вимогам до API. Сервер на базі Ktor забезпечує маршрути для роботи з профілем користувача, вправами, тренувальними програмами, тренуваннями та тренувальними сесіями. Захищені маршрути вимагають автентифікації користувача, а некоректні запити обробляються з поверненням відповідних HTTP-статусів і повідомлень про помилки.

Нефункціональні вимоги також були враховані під час верифікації. Інтерфейс застосунку є зрозумілим і побудований за єдиним стилем, навігація між розділами є логічною, а основні дії користувача виконуються без зайвих переходів. Завдяки використанню Kotlin Multiplatform і Compose Multiplatform застосунок має спільну кодову базу для різних платформ, що відповідає вимозі кросплатформності.

Верифікація також підтвердила підтримуваність архітектури. Система поділена на доменні моделі, репозиторії, локальну базу даних, презентери, екрани інтерфейсу та серверні маршрути. Такий поділ спрощує тестування, внесення змін і подальше розширення функціональності. У майбутньому до системи можна

додати повну синхронізацію між пристроями, розширену авторизацію, рекомендації щодо тренувань або соціальні функції без повної перебудови проєкту.

Результати верифікації показали, що програмна система GanG Gym відповідає основним вимогам, визначеним у першому розділі. Вона реалізує необхідні сценарії для планування, виконання та аналізу тренувань, забезпечує локальне збереження даних, має серверну частину для подальшого розвитку та може використовуватися як основа для персонального цифрового супроводу тренувального процесу.

3.4 Висновки до розділу 3

У третьому розділі було розглянуто процес тестування та верифікації програмної системи GanG Gym. Основною метою цього етапу була перевірка відповідності реалізованого програмного продукту функціональним і нефункціональним вимогам, визначеним під час аналізу предметної області.

Було описано загальний підхід до тестування системи, визначено основні види перевірок і план тестування. Особливу увагу приділено ключовим сценаріям роботи користувача: перегляду каталогу вправ, створенню вправ, формуванню тренувань і програм, запуску тренувальної сесії, збереженню результатів, перегляду календаря та аналізу прогресу.

У межах функціонального тестування підтверджено, що основні модулі застосунку працюють коректно. Система дозволяє користувачу створювати та редагувати вправи, формувати тренування, додавати до них вправи й блоки відпочинку, створювати тренувальні програми, виконувати тренувальні сесії та зберігати результати. Дані тренувальних сесій коректно використовуються для відображення історії занять, календаря та показників прогресу.

Під час навантажувального тестування встановлено, що система стабільно працює в умовах типового навантаження, характерного для персонального фітнес-застосунку. Локальна база даних Room/SQLite забезпечує швидке збереження й

отримання даних, а серверна частина на базі Ktor коректно обробляє API-запити до основних ресурсів.

Автоматизоване тестування дало змогу перевірити працездатність серверних маршрутів, автентифікацію, обробку некоректних запитів і базову готовність спільного клієнтського модуля до подальшого розширення тестового покриття. Використання тестового середовища Ktor дозволило перевіряти API без окремого розгортання сервера.

Також було виконано верифікацію програмної системи. Вона підтвердила, що GanG Gym відповідає основним вимогам, сформульованим у першому розділі. Застосунок забезпечує повний цикл роботи з тренувальним процесом: від створення вправ і тренувань до виконання сесій, збереження результатів і перегляду прогресу.

Отже, результати тестування та верифікації підтвердили працездатність, стабільність і практичну придатність програмної системи GanG Gym. Розроблений застосунок може використовуватися для персонального планування й аналізу тренувань, а його архітектура дозволяє розширювати функціональність у майбутньому.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

Розробка програмного забезпечення для тренувань «Gang Gym» з використанням мови Kotlin Multiplatform належить до видів інтелектуальної праці, що потребують високої концентрації уваги, тривалої роботи за комп'ютером та значного нервово-емоційного навантаження. Під час створення кросплатформного застосунку «Gang Gym» використовувалися сучасні програмні засоби та комп'ютерна техніка, що зумовлює необхідність дотримання вимог безпеки життєдіяльності та охорони праці [21]. У цьому розділі розглянуто поняття діяльності та її видів у контексті безпеки праці розробника програмного забезпечення, а також заходи психофізіологічного розвантаження для підтримання здоров'я та працездатності працівника.

4.1 Діяльність. Її види та розуміння в безпеці праці

Поняття "діяльність" є одним із ключових у сфері безпеки праці та охорони здоров'я працівників. Діяльність – це цілеспрямована активність людини, спрямована на досягнення певної мети та задоволення потреб [22]. У контексті безпеки праці під діяльністю розуміють сукупність дій, операцій і процесів, що виконуються працівником у ході виробничого процесу та пов'язані з певними ризиками для його здоров'я і безпеки.

Розробка програмного забезпечення для застосунку «Gang Gym» охоплює широкий спектр видів діяльності: проєктування архітектури мобільного застосунку, написання програмного коду мовою Kotlin Multiplatform, тестування модулів, аналіз вимог та документування. Кожен із цих видів діяльності має специфічні характеристики з точки зору безпеки праці та потребує відповідних захисних заходів.

За характером навантаження на організм людини розрізняють фізичну та розумову діяльність [3]. Фізична діяльність пов'язана з м'язовими навантаженнями, переміщенням вантажів, виконанням механічних операцій та вимірюється у таких

показниках, як важкість праці. Розумова (інтелектуальна) діяльність передбачає переробку інформації, прийняття рішень, концентрацію уваги та творче мислення – вона характеризується показниками напруженості праці.

Праця розробника програмного забезпечення відноситься до розумової діяльності з переважанням напруженості. Відповідно до класифікації праці за важкістю та напруженістю [24], робота програміста класифікується як така, що характеризується значним нервово-емоційним навантаженням, тривалим зосередженням уваги, монотонністю та потребою у прийнятті рішень в умовах обмеженого часу. Створення модулів для платформ iOS та Android у межах єдиної кодової бази Kotlin Multiplatform потребує одночасного врахування особливостей різних операційних систем, що ще більше підвищує рівень розумового навантаження.

У безпеці праці виокремлюють такі основні види діяльності залежно від умов виконання:

– розумова діяльність – обробка інформації, аналіз, проєктування, програмування;

– операторська діяльність – управління автоматизованими системами через інтерфейс;

– творча діяльність – розробка нових алгоритмів, дизайн, нестандартні рішення;

– управлінська діяльність – координація роботи команди, планування задач;

– педагогічна та комунікативна діяльність – навчання, технічна підтримка.

Під час розробки «Gang Gym» розробник здійснює переважно розумову та творчу діяльність. Зокрема, реалізація спільної бізнес-логіки для Android і iOS у Kotlin Multiplatform вимагає глибокого розуміння механізмів interop, управління залежностями через Gradle та врахування платформоспецифічних обмежень. Це зумовлює підвищену напруженість праці та необхідність відповідних профілактичних заходів.

Розуміння видів діяльності в контексті безпеки праці дозволяє правильно оцінити умови праці та визначити відповідні заходи захисту. Для програмістів, що

розробляють мобільні застосунки, ключовими питаннями охорони праці є: організація ергономічного робочого місця, регламентація тривалості безперервної роботи за екраном, профілактика захворювань опорно-рухового апарату та органів зору, а також забезпечення психофізіологічного розвантаження [21].

Відповідно до нормативних вимог [25], умови праці розробника програмного забезпечення повинні відповідати таким критеріям: температура в приміщенні 22–24 °С, відносна вологість повітря 40–60 %, рівень штучного освітлення на робочій поверхні не менше 300–500 лк, рівень шуму не більше 50 дБА. Організація праці повинна передбачати регламентовані перерви для відпочинку тривалістю 10–15 хвилин через кожні 1,5–2 години роботи.

Таким чином, усвідомлення видів діяльності та їх особливостей у безпеці праці дозволяє комплексно підходити до організації трудового процесу розробника «Gang Gym» та забезпечити безпечні умови праці відповідно до чинного законодавства України.

4.2 Психофізіологічне розвантаження для працівників

Психофізіологічне розвантаження – це комплекс заходів, спрямованих на відновлення фізичної та психічної працездатності людини шляхом зниження рівня нервово-емоційного напруження, усунення психічної втоми та відновлення функціонального стану організму [26]. Для розробника програмного забезпечення, задіяного у створенні застосунку «Gang Gym», психофізіологічне розвантаження є невід'ємною складовою організації трудового процесу.

Інтенсивна інтелектуальна праця, пов'язана з розробкою кросплатформного застосунку засобами Kotlin Multiplatform, зумовлює накопичення нервово-психічного напруження. Тривала робота з великими обсягами коду, відлагодження взаємодії між платформами та постійне прийняття архітектурних рішень призводять до розвитку розумової втоми, зниження концентрації уваги та підвищення ризику помилок. Саме тому своєчасне та якісне психофізіологічне

розвантаження є важливою умовою збереження здоров'я та продуктивності праці [27].

Основними видами психофізіологічного розвантаження для працівників розумової праці є:

– активний відпочинок – виконання фізичних вправ, прогулянки на свіжому повітрі під час обідніх перерв;

– психологічне розвантаження – використання кімнат психологічного розвантаження, сеанси релаксації та медитації;

– виробнича гімнастика – спеціальний комплекс вправ для очей, шиї, спини та кистей рук;

– аутотренінг та дихальні вправи – методи саморегуляції для зняття нервового напруження;

– переключення видів діяльності – чергування розумової роботи з менш напруженими завданнями.

Важливою складовою психофізіологічного розвантаження для програмістів є виробнича гімнастика. Оскільки робота над «Gang Gym» передбачає тривале перебування у статичній позі, виконання спеціальних фізичних вправ дозволяє усунути м'язову напругу, покращити кровообіг та знизити рівень розумової втоми. Рекомендується виконувати вправи для очей кожні 45–60 хвилин роботи, а загальну фізичну розминку – не рідше двох разів протягом робочого дня [28].

Психологічне розвантаження на виробництві може забезпечуватися шляхом облаштування спеціальних кімнат відпочинку, де працівник може провести регламентовані перерви в комфортних умовах. Доведено, що короткочасне перебування у тихому, затишному середовищі з елементами природи (рослини, спокійні кольори інтер'єру, природне освітлення) сприяє швидшому відновленню психічного стану [26]. Для команди розробників застосунку «Gang Gym», зокрема тих, хто відповідає за складні технічні рішення в Kotlin Multiplatform, такі умови є особливо важливими.

Методи аутотренінгу та дихальних вправ можуть застосовуватися безпосередньо на робочому місці. Техніки діафрагмального дихання, прогресивної

м'язової релаксації та усвідомленості (mindfulness) дозволяють знижувати рівень тривожності та стресу, покращувати концентрацію уваги та відновлювати емоційну рівновагу [29]. Їх перевагою є те, що вони не потребують спеціального обладнання та можуть бути виконані протягом 5–10 хвилин під час регламентованих перерв.

Режим праці та відпочинку є визначальним фактором психофізіологічного стану працівника. Відповідно до нормативних вимог [25], для осіб, що виконують роботу з підвищеним нервово-емоційним навантаженням, встановлюються додаткові перерви для відпочинку.

Рекомендований режим роботи програміста передбачає:

– початкову перерву через 1,5–2 години після початку роботи тривалістю 10 хвилин;

– обідню перерву тривалістю не менше 30–60 хвилин;

– перерву в другій половині дня тривалістю 10–15 хвилин;

– обмеження тривалості безперервної роботи за монітором до 2 годин.

Не менш важливим є дотримання режиму дня за межами робочого часу. Достатня тривалість сну (7–8 годин), регулярне харчування, фізична активність та відмова від шкідливих звичок є основою для підтримання належного психофізіологічного стану. Для розробника, залученого до проєкту «Gang Gym», де тематика застосунку безпосередньо пов'язана із фізичними тренуваннями та здоровим способом життя, дотримання цих принципів є особливо доречним [23].

Таким чином, психофізіологічне розвантаження є комплексним і системним підходом до збереження здоров'я та підтримання працездатності розробників програмного забезпечення. Регулярне застосування заходів розвантаження, дотримання режиму праці та відпочинку, а також використання методів саморегуляції дозволяють забезпечити ефективну та безпечну роботу над створенням застосунку «Gang Gym» з використанням Kotlin Multiplatform.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було спроєктовано та розроблено програмну систему GanG Gym, призначену для планування, виконання та моніторингу тренувального процесу. Розроблений застосунок надає користувачу можливість працювати з каталогом вправ, створювати тренування і тренувальні програми, запускати тренувальні сесії, зберігати результати виконання вправ, переглядати календар занять та аналізувати власний прогрес.

У першому розділі було проведено аналіз предметної області та визначено актуальність створення цифрового інструменту для організації тренувань. Встановлено, що ручне ведення тренувального процесу є незручним і не забезпечує швидкого доступу до історії занять, повторного використання програм та аналізу результатів. На основі цього було сформульовано мету роботи, визначено основних акторів, варіанти використання, функціональні та нефункціональні вимоги до програмної системи.

У другому розділі виконано проєктування та розробку GanG Gym. Було обрано ітеративний процес розробки, спроєктовано архітектуру клієнтської та серверної частин, побудовано схеми локальної і серверної баз даних, а також UML-діаграми основних компонентів системи. Для реалізації клієнтської частини використано Kotlin Multiplatform і Compose Multiplatform, що дозволило створити спільну кодову базу для Android, iOS та Desktop. Для локального збереження даних застосовано Room і SQLite, а серверну частину реалізовано на базі Ktor з використанням PostgreSQL та Flyway.

У процесі реалізації було створено основні модулі системи: каталог вправ, тренування, тренувальні програми, тренувальні сесії, календар, прогрес і профіль користувача. Для організації доступу до даних використано репозиторії, DAO-інтерфейси та mapper-функції, які забезпечують перетворення між доменними моделями та структурами бази даних. Презентаційний шар побудовано на основі presenter-класів і UiState-моделей, що забезпечує зрозумілу взаємодію між логікою застосунку та інтерфейсом користувача.

У третьому розділі було проведено тестування та верифікацію програмного забезпечення. Функціональне тестування підтвердило коректність основних сценаріїв: створення вправ, формування тренувань, запуск тренувальної сесії, збереження результатів, перегляд календаря та аналіз прогресу. Навантажувальне тестування показало, що система стабільно працює в умовах типового навантаження. Автоматизовані тести підтвердили працездатність серверних маршрутів, автентифікації та базових компонентів системи.

У четвертому розділі було розглянуто питання безпеки життєдіяльності та основ охорони праці під час роботи з комп'ютерною технікою й інформаційними системами. Визначено основні фактори, які можуть впливати на користувача під час тривалої роботи за комп'ютером, а також наведено рекомендації щодо безпечної організації робочого місця.

Практичне значення отриманих результатів полягає у створенні кросплатформного застосунку, який може бути використаний для персонального ведення тренувань. GanG Gym дозволяє користувачу структурувати тренувальний процес, зберігати історію занять і відстежувати прогрес. Завдяки модульній архітектурі система може бути розширена в майбутньому, зокрема шляхом додавання синхронізації між пристроями, хмарного збереження даних, рекомендацій щодо тренувань або соціальних функцій.

Отже, поставлену мету кваліфікаційної роботи досягнуто. Розроблена програмна система GanG Gym відповідає визначеним вимогам, реалізує основні сценарії організації тренувального процесу та може бути використана як основа для подальшого розвитку цифрового інструменту персонального фітнес-супроводу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kotlin Documentation. URL: https://kotlinlang.org/docs/home.html (дата звернення: 20.06.2026).
2. Kotlin Multiplatform Documentation. URL: https://kotlinlang.org/docs/multiplatform.html (дата звернення: 20.06.2026).
3. Compose Multiplatform Documentation. URL: https://kotlinlang.org/compose-multiplatform/ (дата звернення: 20.06.2026).
4. Jetpack Compose Documentation. URL: https://developer.android.com/compose (дата звернення: 20.06.2026).
5. Android Developers Documentation. URL: https://developer.android.com/docs (дата звернення: 20.06.2026).
6. Room Persistence Library Documentation. URL: https://developer.android.com/training/data-storage/room (дата звернення: 20.06.2026).
7. SQLite Documentation. URL: https://www.sqlite.org/docs.html (дата звернення: 20.06.2026).
8. Ktor Documentation. URL: https://ktor.io/docs/ (дата звернення: 20.06.2026).
9. Ktor Server Documentation. URL: https://ktor.io/docs/server-create-a-new-project.html (дата звернення: 20.06.2026).
10. Gradle User Manual. URL: https://docs.gradle.org/current/userguide/userguide.html (дата звернення: 20.06.2026).
11. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/ (дата звернення: 20.06.2026).
12. Flyway Documentation. URL: https://documentation.red-gate.com/fd (дата звернення: 20.06.2026).

13. Koin Documentation. URL: https://insert-koin.io/docs/ (дата звернення: 20.06.2026).
14. Kotlinx Coroutines Documentation. URL: https://kotlinlang.org/docs/coroutines-overview.html (дата звернення: 20.06.2026).
15. Kotlinx Serialization Documentation. URL: https://kotlinlang.org/docs/serialization.html (дата звернення: 20.06.2026).
16. Firebase Admin SDK Documentation. URL: https://firebase.google.com/docs/admin/setup (дата звернення: 20.06.2026).
17. Material Design 3 Documentation. URL: https://m3.material.io/ (дата звернення: 20.06.2026).
18. IntelliJ IDEA Documentation. URL: https://www.jetbrains.com/help/idea/getting-started.html (дата звернення: 20.06.2026).
19. Android Studio Documentation. URL: https://developer.android.com/studio/intro (дата звернення: 20.06.2026).
20. Xcode Documentation. URL: https://developer.apple.com/documentation/xcode (дата звернення: 20.06.2026).
21. Охорона праці та безпека життєдіяльності в інформаційно-комунікаційній сфері : підручник / О. П. Ткачук, В. В. Березуцький, Т. В. Слободська та ін. – Київ : Каравела, 2021. – 312 с.
22. Безпека життєдіяльності : підручник / О. І. Запорожець, І. М. Пістун, Г. Г. Гогіташвілі та ін. – 2-ге вид., перероб. і доп. – Київ : Центр учбової літератури, 2020. – 448 с.
23. Гігієна та фізіологія праці : підручник / В. А. Гжегоцький, Г. Т. Канівська, Б. А. Назаренко. – Львів : Світ, 2020. – 280 с.
24. Охорона праці : підручник / К. Н. Ткачук, М. О. Халімовський, В. В. Зацарний та ін. – 2-ге вид., перероб. і доп. – Київ : Основа, 2018. – 448 с.
25. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями : НПАОП 0.00-7.15-18 : затв. Наказом Міністерства

соціальної політики України від 14.02.2018 р. № 207. – Київ : Форт, 2018. – 16 с.

26. Психологія праці та управління : навч. посіб. / Л. В. Помаз, О. М. Кривоконь. – Харків : НТУ «ХПІ», 2019. – 210 с.

27. Основи охорони праці та безпеки життєдіяльності : підручник / В. Ц. Жидецький. – Львів : УАД, 2019. – 346 с.

28. Санітарний регламент для підприємств та установ, що експлуатують обчислювальну техніку : ДСанПіН 3.3.2.007-98. – Офіц. вид. – Київ : Міністерство охорони здоров'я України, 1998. – 24 с.

29. Фізіологія людини : підручник / В. І. Філімонов. – 3-тє вид., випр. – Київ : ВСВ «Медицина», 2019. – 488 с.

ДОДАТКИ

Додаток А

Тези конференції

IX Міжнародна студентська науково - технічна конференція
"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ. АКТУАЛЬНІ ПИТАННЯ"

УДК 004.42
Петрик В. В. – ст. гр. СП-42
Тернопільський національний технічний університет імені Івана Пулюя

АРХІТЕКТУРНІ РІШЕННЯ ДЛЯ ПОБУДОВИ КРОСПЛАТФОРМНОГО ЗАСТОСУНКУ МОНІТОРИНГУ ТРЕНУВАЛЬНОГО ПРОЦЕСУ

Науковий керівник: к.т.н., доцент Є. Б. Яворська

Petryk V.
Ternopil Ivan Puluj National Technical University

ARCHITECTURAL SOLUTIONS FOR BUILDING A CROSS-PLATFORM APPLICATION FOR MONITORING THE TRAINING PROCESS

Supervisor: PhD, Associate Professor Ye. B. Yavorska

Ключові слова: Kotlin Multiplatform, Compose Multiplatform, фітнес-застосунок, тренувальний процес, моніторинг прогресу.
Keywords: Kotlin Multiplatform, Compose Multiplatform, fitness application, training process, progress monitoring.

У сучасних умовах розвитку цифрових технологій дедалі більшої актуальності набувають програмні засоби для підтримки здорового способу життя, планування фізичної активності та контролю результатів тренувань. Багато користувачів ведуть облік тренувань у нотатках, електронних таблицях або взагалі не зберігають історію занять. Такий підхід ускладнює аналіз прогресу, повторне використання тренувальних програм і системне планування фізичного навантаження.

Одним із перспективних напрямів розв'язання цієї проблеми є створення кросплатформного застосунку, який поєднує каталог вправ, конструктор тренувань, тренувальні програми, календар занять і модуль аналізу прогресу. Використання Kotlin Multiplatform дозволяє реалізувати спільну бізнес-логіку для Android, iOS та Desktop, що зменшує дублювання коду і спрощує подальшу підтримку програмного продукту.

Запропонована програмна система GanG Gym побудована на основі модульної архітектури. Клієнтська частина містить спільний Kotlin Multiplatform модуль, у якому реалізовано доменні моделі, репозиторії, презентаційну логіку та інтерфейс користувача на базі Compose Multiplatform. Окремі платформозалежні компоненти відповідають за запуск застосунку, роботу з локальною базою даних і доступ до можливостей конкретної операційної системи.

Для локального збереження даних використовується Room та SQLite. У локальній базі даних зберігаються вправи, тренування, блоки тренувань, тренувальні програми, тренувальні сесії та результати виконання вправ. Такий підхід забезпечує можливість автономної роботи застосунку без постійного підключення до мережі, що є важливим під час використання системи безпосередньо в процесі тренування.

Серверна частина системи реалізована за допомогою Ktor. Вона забезпечує REST API для роботи з профілем користувача, вправами та тренувальними програмами. Для збереження серверних даних використовується PostgreSQL, а керування змінами структури бази даних виконується за допомогою Flyway-міграцій. Така архітектура створює основу для подальшої реалізації синхронізації між пристроями та централізованого збереження даних користувача.

Особливу увагу під час розробки приділено організації тренувальної сесії. Користувач може створити тренування, додати до нього вправи або блоки відпочинку, вказати кількість повторень, робочу вагу, тривалість виконання та час відпочинку. Після запуску тренування система формує список вправ для виконання, дозволяє фіксувати фактичні результати та зберігає їх у локальній базі даних.

Збережені результати використовуються для побудови історії тренувань, відображення занять у календарі та формування показників прогресу. Модуль прогресу дає змогу переглядати загальну кількість підходів, повторень, найкращу зафіксовану вагу, сумарний час виконання та останній результат за окремими вправами. Це дозволяє користувачу оцінювати ефективність тренувань і коригувати подальше навантаження.

У процесі розробки було проведено функціональне та автоматизоване тестування основних модулів системи. Перевірено створення вправ, формування тренувань і програм, запуск тренувальної сесії, збереження результатів, роботу календаря, модуля прогресу та серверних API-маршрутів. Результати тестування підтвердили працездатність системи та відповідність реалізованого функціоналу поставленим вимогам.

Перспективи подальшого розвитку проєкту полягають у впровадженні повної синхронізації між пристроями, розширенні аналітичного модуля, додаванні персоналізованих рекомендацій щодо тренувань і створенні соціальних функцій для взаємодії користувачів. Це дозволить перетворити застосунок GanG Gym на повноцінну платформу цифрового супроводу тренувального процесу.

Література:

1. Kotlin Multiplatform Documentation. URL: https://kotlinlang.org/docs/multiplatform.html
2. Compose Multiplatform Documentation. URL: https://kotlinlang.org/compose-multiplatform/
3. Ktor Documentation. URL: https://ktor.io/docs/
4. Room Persistence Library Documentation. URL: https://developer.android.com/training/data-storage/room
5. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/

Додаток Б

Лістинг Б.1 — Збереження тренувальної сесії

```
class LocalWorkoutSessionRepository(
    private val workoutSessionDao: WorkoutSessionDao,
) : WorkoutSessionRepository {

    override fun observeSessions(): Flow<List<WorkoutSession>> =
        workoutSessionDao.observeSessions()
            .map { sessions ->
                sessions.map { it.toDomain() }
            }

    override suspend fun getSession(
        id: WorkoutSessionId,
    ): WorkoutSession? =
        workoutSessionDao.getSessionById(id.value)?.toDomain()

    override suspend fun saveSession(
        session: WorkoutSession,
    ) {
        val syncState =
            if (workoutSessionDao.getSessionById(session.id.value) ==
null) {
                LocalSyncState.PENDING_CREATE
            } else {
                LocalSyncState.PENDING_UPDATE
            }

        workoutSessionDao.upsertSession(
            session.toEntity(syncState = syncState),
        )
```

```
        workoutSessionDao.deleteResultsForSession(session.id.value)

        workoutSessionDao.upsertResults(
            session.toResultEntities(),
        )
    }
}
```

Лістинг Б.2 — Формування результатів для екрана прогресу

```
private fun List<WorkoutSession>.toProgressItems(
    exercises: List<Exercise>,
): List<ExerciseProgressUiModel> {
    val exerciseNames = exercises.associateBy(
        keySelector = { it.id.value },
        valueTransform = { it.name },
    )

    return flatMap { session -> session.results }
        .groupBy { it.exerciseId.value }
        .map { (exerciseId, results) ->
            val sortedResults = results.sortedByDescending {
                it.completedAt?.value ?: 0L
            }

            ExerciseProgressMetrics(
                exerciseId = exerciseId,
                exerciseName    =    exerciseNames[exerciseId]    ?:
exerciseId,
                totalSets = results.size,
                totalReps = results.sumOf { it.reps ?: 0 },
                bestWeight   =   results.mapNotNull   {   it.weightKg
}.maxOrNull(),
```

```
                totalTimeSeconds = results.sumOf { it.durationSeconds
?: 0 },
                lastResult = sortedResults.firstOrNull(),
            )
        }
        .sortedWith(
            compareByDescending<ExerciseProgressMetrics>            {
it.totalReps }
                .thenByDescending { it.totalSets }
                .thenBy { it.exerciseName },
        )
        .map { it.toUiModel() }
}
```

Лістинг Б.3 — Перетворення показників прогресу в модель інтерфейсу

```
private           fun           ExerciseProgressMetrics.toUiModel():
ExerciseProgressUiModel =
    ExerciseProgressUiModel(
        exerciseId = exerciseId,
        exerciseName = exerciseName,
        totalSetsLabel = "$totalSets sets",
        totalRepsLabel = "$totalReps reps",
        bestWeightLabel = bestWeight.toBestWeightLabel(),
        totalTimeLabel = totalTimeSeconds.toTotalTimeLabel(),
        lastResultLabel = lastResult?.toResultLabel() ?: "No result",
    )

private fun Double?.toBestWeightLabel(): String =
    when {
        this == null -> "No weight"
        this <= 0.0 -> "Bodyweight"
        this % 1.0 == 0.0 -> "Best ${toInt()} kg"
        else -> "Best $this kg"
```

```
    }

private fun ExerciseResult.toResultLabel(): String =
    listOfNotNull(
        reps?.let { "$it reps" },
        weightKg?.takeIf { it > 0.0 }?.toWeightLabel(),
        durationSeconds?.takeIf { it > 0 }?.toDurationLabel(),
    )
        .takeIf { it.isNotEmpty() }
        ?.joinToString(" - ")
        ?: "Result saved"
```