**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Західноукраїнський національний університет**
**Факультет комп'ютерних інформаційних технологій**
Кафедра інформаційно-обчислювальних систем і управління

**ВОРОЖБИТ Руслан Олександрович**

# Кросплатформенний мобільний застосунок для виконання спортивних вправ та самоконтролю / Cross-platform mobile application for performing sports exercises and self-control

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-42
Р. О. Ворожбит
_____

Науковий керівник:
к.т.н., доцент І. В. Турченко
_____

Кваліфікаційну роботу
допущено до захисту:
«___» _____ 20___ р.
В.о. завідувача кафедри
_____ Н.В. Дзюбановська

**ТЕРНОПІЛЬ – 2026**

**Факультет комп'ютерних інформаційних технологій**
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.В. Дзюбановська
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
ВОРОЖБИТУ Руслану Олександровичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
**Кросплатформенний мобільний застосунок для виконання спортивних вправ та самоконтролю / Cross-platform mobile application for performing sports exercises and self-control**

керівник роботи к.т.н., доцент І. В. Турченко
затверджений наказом по університету від 01 грудня 2025 року № 892.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити
- провести аналіз предметної області та огляд існуючих додатків для занять спортом, виявити їхні переваги та обмеження;
- обґрунтувати вибір технологій для вирішення задач з побудови моделей даних та їх представлення користувачу;
- спроєктувати архітектуру мобільного застосунку, розробити структуру бази даних, REST API та візуальний мобільний застосунок для представлення даних користувачу;
- реалізувати систему у вигляді мобільного застосунку з модулями вправ, колекціями вправ, статистикою виконання вправ;
- провести тестування мобільного застосунку та оцінити стабільність роботи застосунку.

5. Перелік графічного матеріалу у роботі
- схема алгоритму напівавтоматичного підрахунку повторень;
- схема алгоритму контролю часу

6. Консультанти розділів кваліфікаційної роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання 01 грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів кваліфікаційної роботи | Примітка |
|---|---|---|---|
| 1 | Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи | до 01.01. 2026 р. | |
| 2 | Написання 1 розділу кваліфікаційної роботи | до 01.02. 2026 р. | |
| 3 | Написання 2 розділу кваліфікаційної роботи | до 15.03.2026 р. | |
| 4 | Написання 3 розділу кваліфікаційної роботи | до 01.05.2026 р. | |
| 5 | Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником | до 15.05.2026 р. | |
| 6 | Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів | до 25.05.2026 р. | |
| 7 | Перевірка кваліфікаційної роботи на оригінальність тексту | до 30.05.2026 р. | |
| 8 | Оформлення кваліфікаційної роботи та отримання допуску до захисту | до 10.06.2026 р. | |
| 9 | Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії | до 10.06.2026 р. | |

Студент _____ Р. О. Ворожбит
підпис

Керівник роботи _____ к.т.н., доцент І. В. Турченко
підпис

# АНОТАЦІЯ

Кваліфікаційна робота на тему «Кросплатформенний мобільний застосунок для виконання спортивних вправ та самоконтролю» на здобуття освітнього ступеня «Бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» виконана обсягом 57 сторінок і містить 16 рисунків, 2 таблиці, 3 додатки та список з 25 використаних джерел.

Метою роботи є розробка прототипу мобільного застосунку, що забезпечує користувачам можливість проходження спортивних вправ у зручному інтерактивному форматі. Мобільний застосунок реалізує функціонал формування тренувальних програм, відображення інструкцій та рекомендацій, а також ведення статистики виконаних занять.

Методи дослідження включають аналіз наукової та методичної літератури, системний аналіз вимог до мобільних застосунків, UML-моделювання, проєктування архітектури програмного забезпечення та програмну реалізацію прототипу з використанням фреймворку React Native.

Результати роботи: створено прототип мобільного застосунку для занять спортом, який забезпечує кросплатформену роботу на Android та iOS, містить модулі для формування тренувальних програм, перегляду вправ та моніторингу прогресу користувача. Мобільний застосунок має інтуїтивно зрозумілий інтерфейс та може бути використаний широким колом користувачів для підтримки фізичної активності.

Результати роботи можуть бути застосовані у практиці розробки мобільних застосунків спортивного спрямування, а також як основа для подальшого розширення функціоналу (наприклад, інтеграції з фітнес-пристроями).

Ключові слова: МОБІЛЬНИЙ ЗАСТОСУНОК, СПОРТИВНІ ВПРАВИ, REACT NATIVE, КРОСПЛАТФОРМЕНІСТЬ, ТРЕНУВАЛЬНІ ПРОГРАМИ, ФІЗИЧНА АКТИВНІСТЬ.

# ANNOTATION

The qualification work entitled “Cross-platform mobile application for performing sports exercises and self-control” for the Bachelor’s degree in Computer Science (specialty 122 “Computer Science”, educational program “Computer Science”) comprises 57 pages and includes 16 figures, 2 tables, 3 appendices, and a list of 25 references.

The aim of the work is to develop a prototype of a mobile application that provides users with the ability to perform sports exercises in a convenient interactive format. The application implements functionality for creating training programs, displaying instructions and recommendations, as well as tracking statistics of completed sessions.

The research methods include analysis of scientific and methodological literature, system analysis of requirements for mobile applications, UML modeling, software architecture design, and prototype implementation using the React Native framework.

The results of the qualification work include the development of a prototype mobile application for sports activities, which ensures cross-platform operation on Android and iOS, contains modules for training program creation, exercise viewing, and user progress monitoring. The application features an intuitive interface and can be used by a wide range of users to support physical activity.

The results of the research can be applied in the practice of developing mobile applications for sports purposes and serve as a basis for further functional expansion (e.g., integration with fitness devices).

Keywords: MOBILE APPLICATION, SPORTS EXERCISES, REACT NATIVE, CROSS-PLATFORM, TRAINING PROGRAMS, PHYSICAL ACTIVITY.

ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АПІ – програмний інтерфейс застосунку (Application Programming Interface)

БД – база даних

ІС – інформаційна система

REST – стиль архітектури програмного забезпечення (Representational State Transfer)

SPA – односторінковий застосунок (Single Page Application)

UML – уніфікована мова моделювання (Unified Modeling Language)

JSON – формат обміну даними (JavaScript Object Notation)

# ВСТУП

Сучасний спосіб життя характеризується високим рівнем гіподинамії, надмірним використанням цифрових технологій у повсякденній діяльності та зростанням кількості хронічних захворювань, пов’язаних із малорухливістю. Всесвітня організація охорони здоров’я наголошує, що регулярна фізична активність є одним із ключових чинників профілактики серцево-судинних хвороб, діабету та ожиріння, а також сприяє покращенню психоемоційного стану людини. Проте значна частина населення стикається з проблемою недостатньої мотивації, браком часу чи відсутністю доступу до професійних тренерів. Це створює потребу у цифрових рішеннях, які можуть допомогти організувати тренувальний процес, контролювати прогрес та підтримувати мотивацію користувачів.

Розвиток мобільних технологій та поширення смартфонів відкрили нові можливості для створення інноваційних додатків у сфері спорту та здорового способу життя. Такі додатки здатні інтегруватися з натільними пристроями, збирати дані про фізичну активність, формувати індивідуальні програми тренувань та надавати рекомендації з вправ. У зв’язку з цим розробка мобільного застосунку для занять спортом, який поєднує функції планування, моніторингу та мотивації, є актуальним науково-практичним завданням.

Об’єктом дослідження є процеси організації та підтримки фізичної активності користувачів за допомогою мобільних технологій.

Предметом дослідження є методи та моделі персоналізованої підтримки занять спортом у мобільному застосунку.

Метою роботи є розробка кросплатформенного мобільного застосунку для виконання спортивних вправ та самоконтролю.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати сучасні тенденції у сфері мобільних додатків для спорту та здорового способу життя;

- здійснити огляд існуючих рішень та визначити їхні переваги й обмеження;

- обґрунтувати вибір методів аналізу даних та алгоритмів персоналізації тренувань;

- розробити архітектуру мобільного застосунку, алгоритмічне забезпечення та структуру бази даних;

- реалізувати прототип мобільного застосунку з модулями планування тренувань, моніторингу активності та формування рекомендацій;

- провести тестування мобільного застосунку.

Практичне значення результатів полягає у створенні прототипу мобільного застосунку для занять спортом, який може використовуватися широким колом користувачів для підтримки фізичної активності, контролю прогресу та формування індивідуальних рекомендацій. Результати роботи також можуть бути застосовані у навчальному процесі як приклад використання сучасних інформаційних технологій у сфері спорту та здоров’я.

Результати роботи опубліковано в матеріалах міжнародної науково-практичної конференції студентів і молодих вчених «ICSPM 2026: Інтелектуальні обчислювальні системи та управління проєктами, м. Тернопіль, 21–22 травня 2026 р. (додаток Б).

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1 Аналіз основних підходів до тренувань

Тренувальний процес у сучасній спортивній науці розглядається як систематизована діяльність, що має чітку структуру, спрямовану на розвиток фізичних якостей, адаптацію організму до навантажень та досягнення визначених результатів. Його організація базується на трьох фундаментальних засадах: фізіології, біомеханіці та педагогіці.

Фізіологія визначає реакції організму на фізичне навантаження. Ключові аспекти:

- адаптація м’язової системи: розвиток силових і витривалих властивостей через гіпертрофію та зміни у складі м’язових волокон;

- енергетичні процеси: використання аеробних та анаеробних шляхів для забезпечення роботи м’язів;

- відновлення: баланс між катаболічними та анаболічними процесами, що забезпечує прогресивний розвиток.

Біомеханіка аналізує рухи та навантаження з точки зору ефективності й безпеки:

- раціональність рухів: правильна техніка виконання вправ мінімізує ризик травм.

- оптимізація навантаження: вибір амплітуди, швидкості та траєкторії рухів для досягнення максимальної ефективності;

- механічна адаптація: зміцнення суглобів, зв’язок і сухожиль у відповідь на систематичні навантаження.

Педагогіка забезпечує методичну організацію тренувань:

- індивідуалізація: врахування віку, статі, рівня підготовки та цілей спортсмена;

- поступовість: прогресивне збільшення навантаження для уникнення перевантаження;

- варіативність: зміна вправ, інтенсивності та обсягу для запобігання адаптації та підтримки мотивації;

- циклічність: побудова тренувального процесу у вигляді мікро-, мезо- та макроциклів, що дозволяє планувати пікові навантаження та відновлення.

Тренувальний процес не є хаотичним набором вправ, а являє собою цілісну систему, де кожен елемент (навантаження, відпочинок, техніка, мотивація) взаємопов'язаний. Ефективність досягається завдяки інтеграції:

- фізіологічних реакцій (адаптація організму);
- біомеханічних закономірностей (раціональність рухів);
- педагогічних принципів (методична організація).

Таким чином, науковий підхід до тренувального процесу передбачає його розгляд як багаторівневої системи, що поєднує біологічні, механічні та педагогічні аспекти. Лише комплексне застосування цих принципів забезпечує ефективний розвиток фізичних якостей, стійку адаптацію організму та досягнення високих спортивних результатів.

## 1.2 Огляд сучасних застосунків для занять спортом

Для вивчення інтеграції сучасних мобільних технологій та веб-систем у процеси фізичної реабілітації та спортивного тренування було проведено огляд літератури та існуючих технологічних рішень. Аналіз базується на дослідженні архітектурних підходів у кросплатформовій розробці (React Native), системі управління контентом (CMS на React) та впливу цифровізації на ефективність тренувального процесу.

Сьогодні ринок Health & Fitness пропонує кілька лідерів, які задають стандарти інтерфейсів та функціоналу. Ці додатки демонструють успішну реалізацію колекцій вправ та управління інвентарем. Нижче наведено детальну харрактеристику додатків.

Nike Training Club (NTC) - еталонний приклад структуризації контенту. Мобільний застосунок використовує складні колекції (наприклад, "Йога для бігунів" або "Тренування без знаряддя"), де кожна вправа має чітку відеоінструкцію та опис. Це ілюструє необхідність гнучкого фільтра за наявним обладнанням [14]. Рисунки 1.1 - 1.3 ілюструють основні копії екранів додатка Nike Training Club.

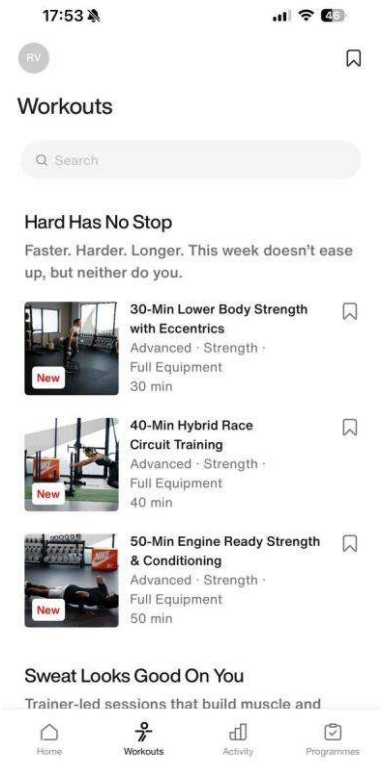


Рисунок 1.1 - Екран з тренуваннями

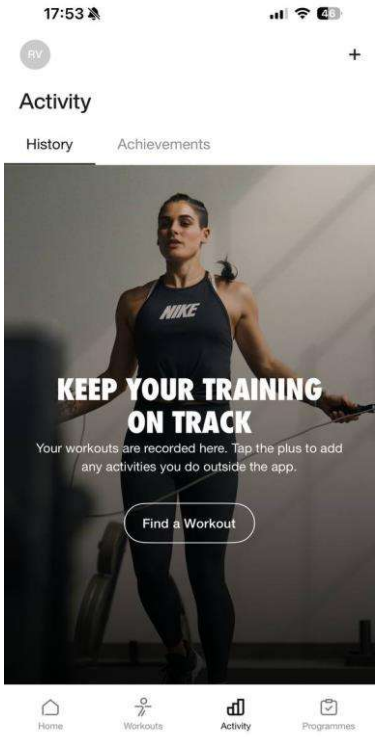


Рисунок 1.2 - Екран з історією вправ

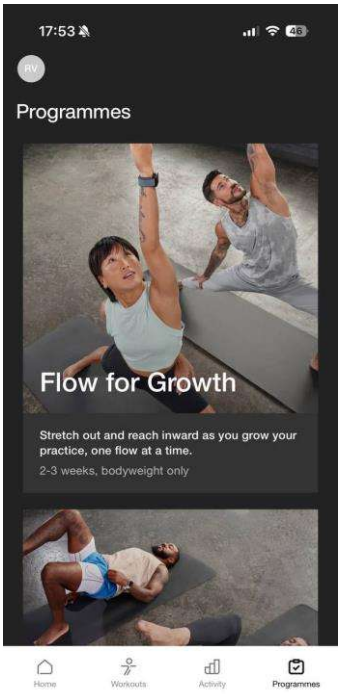


Рисунок 1.3 - Екран з програмами для тренувань

Мобільний застосунок Fitbod фокусується на алгоритмічному створенні тренувань. Його ключова особливість – динамічне налаштування списку вправ залежно від того, яке знаряддя користувач вибрав у профілі (Gym, Home, Bodyweight). Це підкреслює актуальність вашої системи менеджменту інвентарю [15]. Рисунки 1.4 - 1.7 ілюструють основні копії екранів додатка Fitbod.

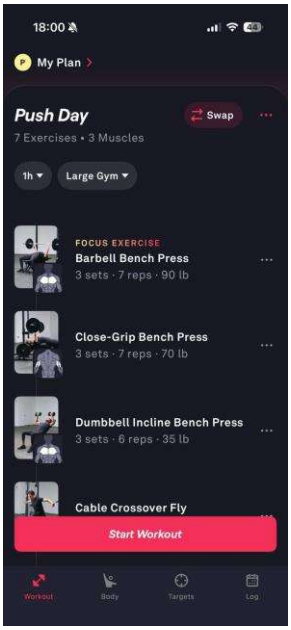


Рисунок 1.4 - Екран з вправами

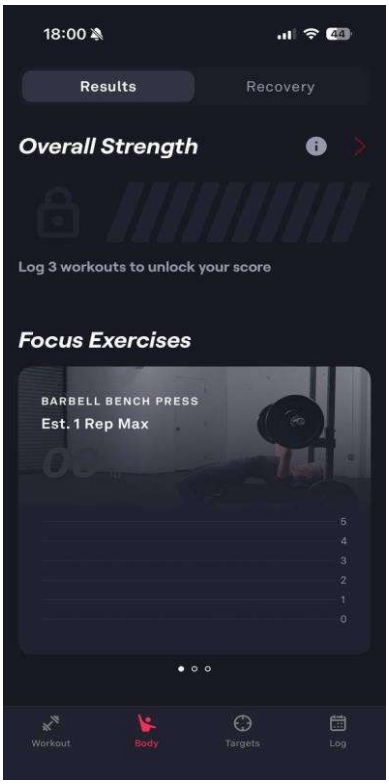


Рисунок 1.5 - Екран з результатами тренувань

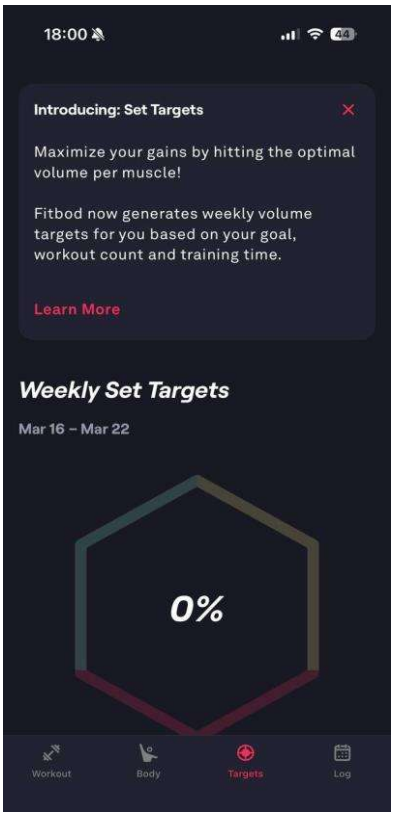


Рисунок 1.6 - Екран з цілями на тренування

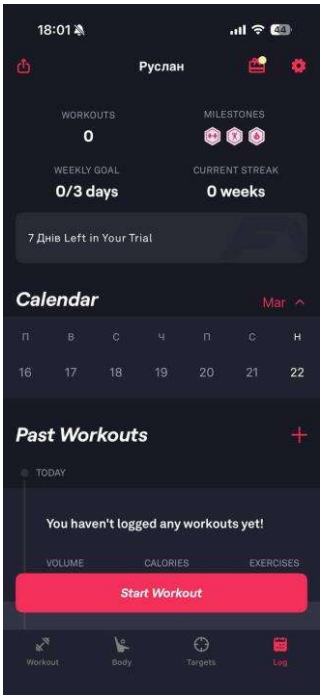


Рисунок 1.7 - Екран з записами тренувань у календарі

Strong – популярний інструмент для запису даних тренувань, який демонструє важливість зручної бібліотеки вправ. Він дозволяє користувачам створювати власні вправи та групувати їх, що є прямою паралеллю до вашого модуля "Створення вправ" Рисунки 1.8 - 1.11 ілюструють основні копії екранів додатка Strong.

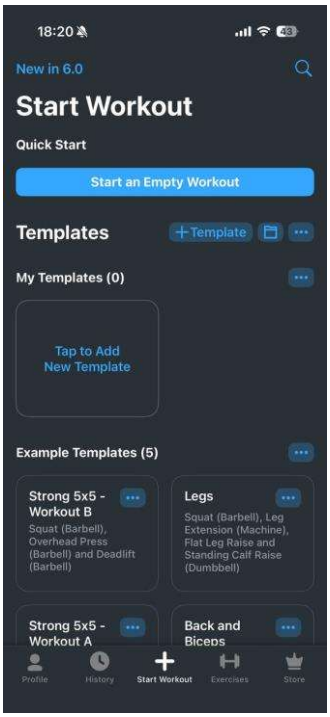


Рисунок 1.8 - Екран для старту роботи над вправами

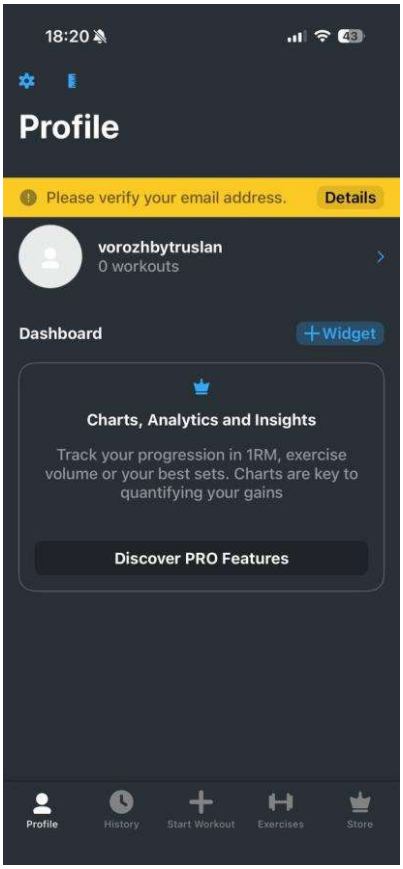


Рисунок 1.9 - Екран з профілем користувача

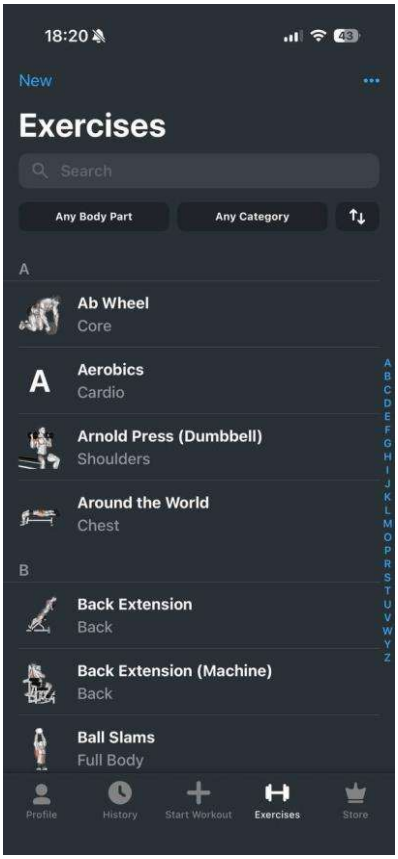


Рисунок 1.10 - Екран з доступними вправами

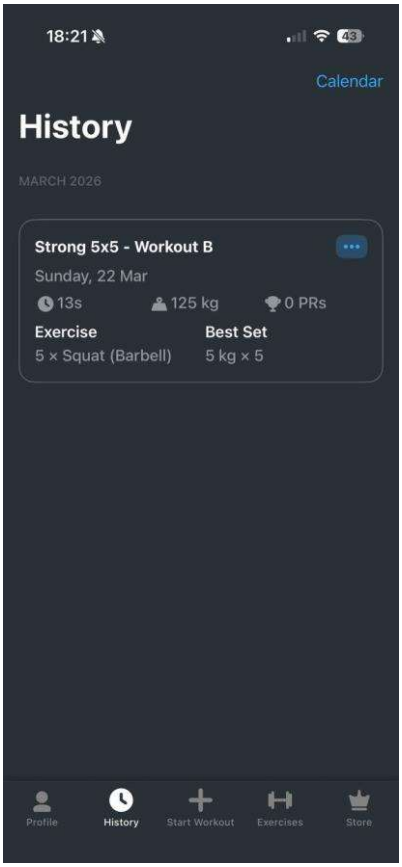


Рисунок 1.11 - Екран з історією виконання вправ

Традиційно спортивні додатки стикалися з проблемою розрізненості даних: мобільний клієнт часто мав обмежену логіку, а оновлення контенту потребувало випуску нової версії в App Store чи Google Play. Однак використання React Native кардинально змінює цей підхід. Завдяки спільній бізнес-логіці та можливості динамічного отримання даних через API, мобільний застосунок стає гнучким інструментом, що миттєво адаптується до змін у базі даних, внесених через адміністративну панель на React [2].

Огляд літератури з розробки інтерфейсів показує, що впровадження компонентного підходу дозволяє підвищити залученість користувачів. Наприклад, використання бібліотек візуалізації даних у веб-інтерфейсі дозволяє тренерам або адміністраторам бачити теплові карти популярності вправ, що допомагає в оптимізації колекцій.

У наукових працях з методики спорту відзначається нестача систем, які забезпечують довгострокову послідовність тренувань без втрати їх контексту. Більшість додатків пропонують або занадто жорсткі плани, або хаотичний набір

вправ.

Розробка рішення, представленого в роботі, заповнює ці прогалини через наступні інновації:

а) семантичне групування (React Admin). Веб-платформа дозволяє не просто додавати вправи, а створювати складні зв'язки між типом знаряддя та ефективністю вправи для конкретної групи м'язів;

б) контекстуальна адаптивність (React Native). Мобільний застосунок аналізує наявний у користувача інвентар і миттєво фільтрує доступні колекції, що критично важливо для підтримки регулярності тренувань у різних локаціях;

в) масштабованість бази даних: На відміну від статичних списків, архітектура даного проекту передбачає розширення через "атомарні одиниці" (вправи), які можуть бути рекомбіновані у нескінченну кількість програм.

Незважаючи на переваги, розробка стикається з викликами:

а) синхронізація медіа-контенту:  передача відеоінструкцій високої якості на мобільні пристрої потребує оптимізації кешування та використання CDN;

б) уніфікація знаряддя: різні виробники та типи обладнання вимагають створення чіткої таксономії в адмін-панелі, щоб уникнути плутанини (наприклад, різниця між гантелями та гирями у контексті певних вправ);

в) ергономіка вводу: процес створення вправи адміністратором має бути максимально автоматизованим, щоб мінімізувати ручну роботу.

Проєкт розробки інтелектуальної спортивної екосистеми на основі React Native та React має на меті подолати недоліки існуючих рішень шляхом впровадження методів структурованого управління контентом та гнучкої фільтрації.

Розроблений мобільний додаток адаптується до широкого спектру сценаріїв (від домашніх тренувань до професійних залів) завдяки спеціально спроектованим модулям "Вправи-Колекції-Знаряддя".

Розроблений мобільний застосунок встановлює новий стандарт зручності та методичної точності в сучасних системах цифрового фітнесу, забезпечуючи безшовну інтеграцію між адміністратором контенту та кінцевим споживачем. Така

інтеграція підвищує загальну зручність у використанні, трансформуючи класичний підхід до тренувань у гнучкий інтерактивний досвід.

## 1.3 Обґрунтування вибору технологій для вирішення поставлених проблем

React Native – це сучасний фреймворк від компанії Meta, який дозволяє створювати мобільні додатки для Android та iOS з використанням єдиної кодової бази на JavaScript/TypeScript. Його ключова особливість полягає у можливості застосування компонентного підходу та повторного використання коду, що значно скорочує час і ресурси на розробку.

Переваги React Native для спортивних додатків:

- висока продуктивність: завдяки використанню нативних компонентів React Native забезпечує плавну роботу інтерфейсу, що критично важливо для додатків з інтерактивними графіками, таймерами та трекерами активності;

- модульність: структура дозволяє легко інтегрувати додаткові функції (наприклад, відстеження пульсу, GPS-навігацію, інтеграцію з фітнес-браслетами);

- швидке оновлення: технологія Hot Reloading дає можливість оперативно тестувати зміни без повної перекомпіляції застосунку;

- екосистема: велика кількість бібліотек та плагінів для роботи з сенсорами, графіками, базами даних, що особливо корисно для спортивних додатків.

Кросплатформеність означає можливість створення одного мобільного застосунку, який працює на різних операційних системах, це надає такі плюси:

- єдина кодова база: розробник пише код один раз, а мобільний застосунок функціонує на Android та iOS. Це зменшує витрати часу та коштів;

- уніфікований інтерфейс: користувачі отримують схожий досвід незалежно від платформи, що підвищує зручність використання;

- швидкий вихід на ринок: завдяки кросплатформеності можна одночасно охопити більшу аудиторію спортсменів та користувачів фітнес-додатків.

- гнучкість інтеграцій: React Native дозволяє додавати нативні модулі для специфічних функцій (наприклад, HealthKit для iOS чи Google Fit для Android), що

забезпечує повну функціональність мобільного застосунку.

Вибір React Native для розробки спортивного застосунку є обґрунтованим з точки зору:

- економічної ефективності (зменшення витрат на розробку та підтримку);

- технологічної універсальності (можливість інтеграції з різними пристроями та сервісами);

- педагогічної цінності (створення доступного інструменту для масового користувача, що сприяє популяризації фізичної активності).

Таким чином, використання React Native у розробці спортивного застосунку забезпечує оптимальний баланс між продуктивністю, кросплатформеністю та економічною доцільністю, що робить його одним із найкращих рішень для реалізації подібних проєктів.

Варто також зазначити, що наразі на ринку React Native є одним з найбільш прогресивних рішень на ринку. Для підкріплення цієї думки пропоную порівняння з іншими доступними рішеннями:

- React Native: мова програмування: JavaScript/TypeScript; переваги стеку: велика екосистема бібліотек та плагінів, використання нативних компонентів забезпечує високу продуктивність, Hot Reloading для швидкого тестування змін, легке інтегрування з існуючими веб-технологіями; недоліки: у складних графічних інтерфейсах може поступатися Flutter, частина функціоналу потребує нативних модулів;

- Flutter: мова програмування: Dart; переваги: власний рендеринг-движок (Skia), що забезпечує однаковий вигляд на Android та iOS, висока продуктивність у графічно насичених додатках, велика кількість готових UI-компонентів; недоліки: менша кількість бібліотек у порівнянні з JavaScript-екосистемою, вища крива навчання через нову мову (Dart), додатки можуть бути «важчими» за розміром;

- Xamarin: мова програмування: C#; переваги: глибока інтеграція з екосистемою Microsoft, можливість використання спільної логіки для мобільних та десктопних додатків, підтримка нативних API через Xamarin.Forms; недоліки: менша популярність серед розробників, ніж React Native чи Flutter, обмежена

кількість готових UI-компонентів, вища залежність від Microsoft-екосистеми.

Аргументація вибору React Native для спортивного застосунку:

- кросплатформеність охоплює користувачів Android та iOS, що критично для масового спортивного застосунку;

- JavaScript, має широку базу розробників, що знижує витрати на команду;

- гнучкість інтеграцій надає можливість підключення нативних модулів для роботи з сенсорами, GPS, HealthKit та Google Fit;

- швидкість розробки завдяки Hot Reloading та великій кількості бібліотек;

- React Native надає достатню швидкість роботи для додатків, які потребують інтерактивних графіків, таймерів та трекерів, але не є настільки графічно складними, щоб вимагати Flutter.

У порівнянні з Flutter та Xamarin, React Native є оптимальним вибором для розробки спортивного застосунку завдяки поєднанню: широкої екосистеми, економічної доцільності, простоти інтеграції з нативними сервісами, та швидкості розробки.

Подібний синергетичний ефект дозволяє суттєво оптимізувати процес проектування, мінімізуючи часові витрати на написання специфічного для кожної платформи коду та зміщуючи фокус безпосередньо на реалізацію унікальної бізнес-логіки фітнес-платформи.

Крім того, уніфікація архітектурних рішень забезпечує стабільне масштабування продукту в майбутньому, що є критично важливим для динамічного середовища мобільних технологій. Для більш детального аналітичного огляду та наочного відображення відмінностей між сучасними кросплатформеними фреймворками, нижче наведено структуроване зіставлення їхніх технічних та експлуатаційних параметрів.

Комплексний аналіз вищезазначених технологій дозволяє чітко окреслити межі ефективного застосування, враховуючи специфіку архітектури мобільних операційних систем та вимоги до швидкодії інтерфейсу користувача. Для більш детального аналітичного огляду, глибокого теоретичного обґрунтування та наочного відображення ключових відмінностей між сучасними

кросплатформеними фреймворками, виникає необхідність у систематизації їхніх технічних, функціональних та експлуатаційних параметрів.

У таблиці 3.1 наведено зведену характеристику інструментальних засобів, використаних при розробці.

Таблиця 3.1 – Інструментальні засоби розробки

| Технологія / Інструмент | Актуальна версія (2026) | Призначення |
|---|---|---|
| React Native | v0.78 | Кросплатформена розробка мобільного клієнта (Android та iOS) |
| TypeScript | v5.8 | Мова програмування зі статичною типізацією для зменшення кількості помилок |
| React Navigation | v7 | Управління навігацією між екранами мобільного застосунку |
| Redux Toolkit | v2.5 | Управління глобальним станом (сесії, вправи, результати) |
| Axios | v1.7 | HTTP-клієнт для взаємодії з сервером із підтримкою перехоплювачів (JWT) |
| Node.js | v22 / v23 (LTS) | Серверна платформа для виконання JavaScript |
| NestJS | v11 | Розробка серверної частини, створення REST API |
| MongoDB | v8.0 | Документно-орієнтована база даних для зберігання даних |
| Mongoose | v8.9 | Бібліотека для інтеграції Node.js з MongoDB |
| Visual Studio Code | v1.98+ | Середовище розробки з розширеннями (ESLint, Prettier) |
| Git | v2.48+ | Система контролю версій |

## 1.4 Формалізація задачі та вимог до мобільного застосунку

Мобільний застосунок для підтримки занять спортом має забезпечувати користувачеві індивідуалізовані рекомендації, контроль виконання тренувальних програм та інтеграцію з сенсорними пристроями. Основна задача полягає у створенні програмного комплексу, здатного:

- дозволяти користувачу формувати вправи самостійно, для більшої кількості користувацьких сценаріїв;

- здійснювати моніторинг прогресу та вести статистику виконання вправ;

- забезпечувати зручний інтерфейс для взаємодії користувача з мобільним застосунком.

Для формалізації задачі пропоную розглянути мобільний застосунок як багаторівневу систему оптимізації:

- вхідні дані: антропометричні показники (зріст, вага, вік), фізіологічні параметри (пульс, рівень активності), цілі користувача;

- обробка: алгоритми аналізу даних, моделі машинного навчання для прогнозування результатів, правила адаптації тренувальних програм;

- вихідні дані: індивідуалізовані рекомендації щодо тренувань, харчування та відновлення;

- критерії оптимізації: максимізація ефективності тренувань при мінімізації ризику перевантаження та травм.

Функціональні вимоги:

- персоналізація: мобільний застосунок повинен враховувати індивідуальні особливості користувача;

- адаптивність: можливість корекції тренувальних програм у залежності від прогресу та змін фізичного стану;

- інтеграція: підтримка роботи з зовнішніми пристроями (фітнес-браслети, смарт-годинники, датчики);

- аналітика: формування звітів про динаміку фізичних показників;

- інтерфейс: інтуїтивно зрозумілий дизайн, що забезпечує простоту використання.

Нефункціональні вимоги:

- надійність: мобільний застосунок повинен працювати стабільно при різних умовах використання.

- масштабованість: можливість обробки великої кількості користувачів одночасно;

- безпека: захист персональних даних користувача відповідно до міжнародних стандартів;

- продуктивність: швидка обробка даних у реальному часі.

- кросплатформеність: підтримка роботи на Android та iOS завдяки використанню React Native.

Формалізація задачі та вимог до мобільного застосунку дозволяє визначити ключові параметри її функціонування та критерії ефективності. Такий підхід забезпечує створення програмного продукту, здатного не лише автоматизувати процес тренувань, але й підвищити їх результативність завдяки індивідуалізації та адаптивності.

# 2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ, АЛГОРИТМІЧНОГО ТА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

Проєктування алгоритмічного та інформаційного забезпечення спортивного мобільного засутосунку базується на розробці концептуальної моделі функціонування мобільного застосунку, описі алгоритмів роботи основних модулів та створенні структури бази даних для збереження інформації про користувачів і тренувальні програми.

Отримані результати охоплюють такі ключові напрями:

- алгоритмічне забезпечення визначає логіку формування тренувальних програм, алгоритми відображення вправ та рекомендацій, а також механізми підрахунку повторень і контролю часу виконання. Описано алгоритми ведення статистики занять та побудови графіків прогресу користувача;

- спроєктовано структуру бази даних, яка містить таблиці для збереження даних про користувачів, вправи, тренувальні програми та результати їх виконання;

- побудовано UML-діаграми (діаграму класів, діаграму послідовності та діаграму компонентів), що відображають архітектурну взаємодію між модулями мобільного застосунку та потоки даних.

Цей комплекс рішень закладає основу для подальшої програмної реалізації застосунку, визначаючи його архітектуру, алгоритмічну логіку та інформаційну модель, які у сукупності забезпечують функціональність, масштабованість і зручність використання мобільного застосунку.

## 2.1 Опис компонентів: інтерфейс користувача, модуль бізнес-логіки, модуль роботи з даними, API для інтеграції.

У цьому підрозділі розглядається архітектурна структура спортивного мобільного застосунку, яка визначає його функціональність та взаємодію між основними складовими. Опис компонентів дозволяє зрозуміти, як саме реалізується робота мобільного застосунку – від інтерфейсу користувача до

обробки даних на серверній стороні. Архітектура мобільного застосунку побудована за принципом розділення відповідальностей [7]:

- інтерфейс користувача забезпечує зручну взаємодію з мобільним застосунком та відображення актуальної інформації;

- модуль бізнес-логіки реалізує правила формування тренувальних програм, контроль навантаження та обробку результатів;

- модуль роботи з даними відповідає за збереження профілів користувачів, історії тренувань та результатів у базі даних;

- API для інтеграції забезпечує зв'язок між мобільним клієнтом та бекендом, реалізованим на Node.js/NestJS, з використанням MongoDB як системи управління даними [8, 9].

Такий підхід гарантує масштабованість, узгодженість та надійність роботи мобільного застосунку, а також створює основу для подальшого розширення його функціоналу.

Інтерфейс реалізований у мобільній версії для Android та iOS за допомогою React Native [1, 2]. Основні екрани:

- логін / реєстрація – створення акаунта та авторизація користувача;

- профіль користувача – персональні дані, історія тренувань;

- список вправ – каталог вправ, створених тренером;

- колекції вправ – групування вправ у тренувальні програми;

- екран проходження вправ – покрокове виконання тренування з таймером та підрахунком повторень;

- статистика користувача – графіки та звіти про прогрес (кількість занять, виконані вправи).

Модуль бізнес-логіки, який відповідає за основну логіку в мобільному застосунку:

- формування програм: вправи та програми створює лише тренер, користувач отримує доступ до них;

- контроль навантаження: рівень інтенсивності визначає користувач або його тренер (кількість повторень, тривалість вправ);

- обробка даних: логіка відображення вправ, рекомендацій та побудови статистики.

Модуль роботи з даними, який відповідає за обробку та збереження інформації:

- профіль користувача: дані акаунта(електронна пошта, аватар);

- результати тренувань: кількість повторень, час виконання;

- синхронізація: дані залишаються актуальними між різними пристроями завдяки зберіганню даних на стороні сервера.

- база даних: використовується MongoDB [5], яка містить колекції для користувачів, вправ, програм та результатів.

API для інтеграції, яке забезпечує взаємодію між мобільним клієнтом та серверною частиною:

- REST API: використовується для обміну даними між клієнтом і бекендом;

- серверна частина: реалізована на Node.js з використанням NestJS [3].

Основні ресурси API:

- авторизація та реєстрація користувачів;

- отримання списку вправ та програм;

- збереження результатів тренувань;

- формування та передача статистики користувача.

Інтеграція зі сторонніми сервісами не передбачена, мобільний застосунок працює автономно. Таким чином архітектура спортивного мобільного застосунку складається з чотирьох ключових компонентів: інтерфейсу користувача, модуля бізнес-логіки, модуля роботи з даними та REST API. Взаємодія між ними забезпечує повний цикл роботи мобільного застосунку – від входу користувача до аналізу його прогресу, зберігаючи дані доступними на різних пристроях.

## 2.2 Алгоритмічне забезпечення мобільного застосунку

У мобільному застосунку для занять спортом формування тренувальних програм є ключовим елементом, що визначає ефективність та зручність використання мобільного застосунку. Програма будується на основі поєднання

загальних принципів, правил побудови занять, варіативності вправ та алгоритмічної реалізації.

Загальні принципи побудови програми

- цільова орієнтація: кожна тренувальна програма створюється відповідно до індивідуальної мети користувача. Це може бути зниження маси тіла, нарощування м'язової маси, розвиток витривалості чи підтримка загальної фізичної форми. Мобільний застосунок враховує обраний напрям і формує набір вправ, які найбільш ефективно сприяють досягненню поставленої мети;

- прогресивність: навантаження у програмі зростає поступово. Це стосується кількості повторень, тривалості виконання вправ та інтенсивності занять. Такий підхід забезпечує адаптацію організму та запобігає перевантаженню;

- баланс: тренувальні програми включають різні типи вправ – силові, кардіо та функціональні. Це дозволяє гармонійно розвивати м'язову силу, витривалість та координацію рухів;

- індивідуалізація: мобільний застосунок враховує вік, рівень фізичної підготовки та можливі обмеження користувача (наприклад, травми чи медичні рекомендації). Це дозволяє створювати безпечні та ефективні програми.

Варіативність вправ:

- ротація вправ: щоб уникнути монотонності та підтримувати мотивацію, мобільний застосунок пропонує різні варіанти вправ для однієї групи м'язів. Наприклад, замість класичних присідань можуть бути використані випади чи вправи з еспандером;

- рівні складності: кожна вправа має кілька рівнів виконання – початковий, середній та просунутий. Це дозволяє поступово ускладнювати програму відповідно до прогресу користувача;

- адаптація: при досягненні певних результатів (наприклад, збільшення кількості повторень чи покращення витривалості) мобільний застосунок автоматично пропонує більш складні варіанти вправ [15];

- комбінування: користувач може отримати комплексні програми, які поєднують різні типи навантажень. Наприклад, силові вправи можуть доповнюватися кардіо-компонентами, що забезпечує всебічний розвиток [14].

Алгоритмічна реалізація:

- модуль генерації програми: формує список вправ на основі рекомендацій тренера, цілей користувача та його рівня підготовки. Алгоритм враховує історію занять та пропонує оптимальну послідовність вправ;

- модуль контролю прогресу: аналізує виконані тренування, відстежує динаміку показників (кількість повторень, тривалість, частота занять) та пропонує корекцію навантаження;

- модуль рекомендацій: на основі зібраних даних підказує користувачеві оптимальні вправи, їх послідовність та інтенсивність. Він враховує історію тренувань, рівень складності та індивідуальні особливості користувача.

Таким чином, логіка формування тренувальних програм у спортивному мобільному застосунку базується на принципах цільової орієнтації, прогресивності, балансу та індивідуалізації. Варіативність вправ і алгоритмічна реалізація забезпечують ефективність занять, підтримку мотивації та довготривалу адаптацію організму до фізичних навантажень.

У мобільному застосунку реалізовано напівавтоматичний підрахунок повторень. Мобільний застосунок пропонує інтерактивний лічильник, де користувач може натискати кнопку після кожного повторення або завершення підходу. Алгоритм підрахунку повторень включає:

- ініціалізацію вправи (задання цільової кількості повторень);

- фіксацію кожного повторення (ручним або автоматичним способом);

- перевірку досягнення цільового значення;

- збереження результату у базі даних для подальшої статистики.

Також було продумано механізм контролю часу виконання вправ:

- таймер вправи: для кожної вправи задається тривалість виконання (наприклад, планка – 60 секунд);

- зворотний відлік: користувач бачить таймер, який відраховує залишок часу до завершення вправи;

- сигнали завершення: після закінчення часу мобільний застосунок подає звуковий або візуальний сигнал;

- фіксація результату: фактичний час виконання зберігається у базі даних, що дозволяє аналізувати прогрес.

На рисунку 2.1 представлено схему алгоритму напівавтоматичного підрахунку повторень.

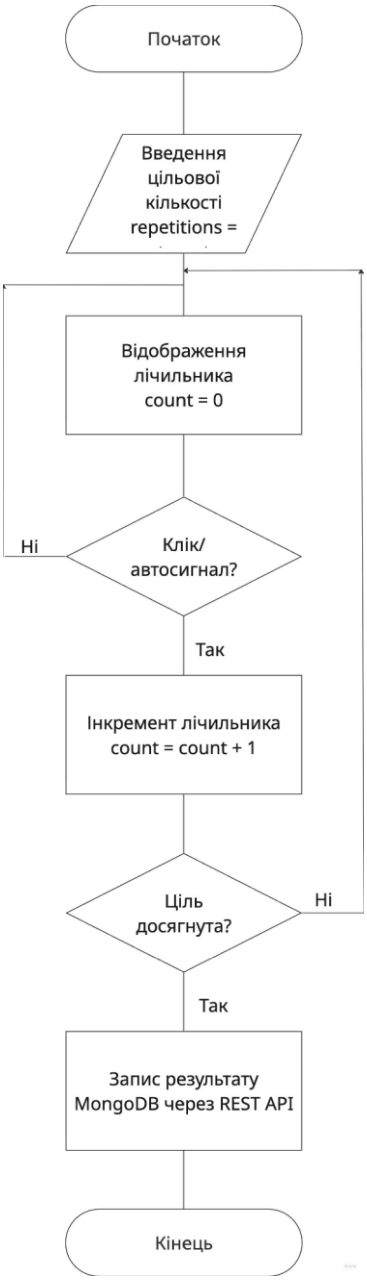


Рисунок 2.1 - Схема алгоритму напівавтоматичного підрахунку повторень

Алгоритм контролю часу:

- запуск таймера при старті вправи;

- відображення залишку часу на екрані;

- сигналізація про завершення вправи;

- збереження фактичного часу виконання у профілі користувача.

На рисунку 2.2 представлено схему алгоритму контролю часу.


Рисунок 2.2 - Схема алгоритму контролю часу

Також, для того, щоб модуль не був ізольованим від інших модулів, було розроблено інтеграцію цього модуля з іншими модулями:

- модуль статистики: отримує дані про кількість повторень та час виконання, будує графіки прогресу;

- модуль бізнес-логіки: перевіряє відповідність виконаних результатів до цільових показників програми.

Технічна реалізація:

- UI: інтерактивні лічильники повторень та таймери на екрані проходження вправ;

- бекенд: збереження даних у MongoDB через REST API (кількість повторень, час виконання, дата тренування);

- алгоритми: прості цикли та таймери для контролю часу, обробка подій для підрахунку повторень [21].

## 2.3 Інформаційне забезпечення мобільного застосунку

Інформаційне забезпечення мобільного застосунку охоплює сукупність структур даних, форматів обміну інформацією та механізмів її зберігання, що забезпечують функціонування мобільного застосунку відповідно до визначених вимог.

Для зберігання інформації використовується документно-орієнтована база даних MongoDB, яка дозволяє гнучко описувати різнотипні об'єкти предметної області.

База даних містить п'ять основних колекцій:

- users – профілі користувачів (email, хешований пароль, ім'я, аватар, дата реєстрації);

- exercises – вправи (назва, опис, тривалість, кількість повторень, тип, група м'язів, рівень складності, зображення);

- collections – тренувальні програми (назва, опис, рівень, перелік ідентифікаторів вправ);

- workout_sessions – сесії тренувань (ідентифікатор користувача, ідентифікатор колекції, час початку та завершення, масив результатів);

- exercise_results – результати виконання вправ (ідентифікатор сесії,

ідентифікатор вправи, кількість виконаних повторень, фактичний час виконання, дата).

Зв'язки між колекціями реалізовано через посилання (ObjectId): документ workout_session містить посилання на user та collection, а документ exercise_result – на workout_session та exercise. Такий підхід забезпечує цілісність даних при мінімальній надмірності.

Взаємодія між мобільним клієнтом та серверною частиною здійснюється через REST API з використанням формату JSON (JavaScript Object Notation). JSON обрано через його легковагність, широку підтримку в екосистемі JavaScript/TypeScript та зручність серіалізації об'єктів MongoDB. Усі запити та відповіді мають визначену схему, що забезпечує передбачуваність інтерфейсу взаємодії.

Безпека інформації забезпечується на кількох рівнях. Паролі користувачів зберігаються у хешованому вигляді з використанням алгоритму bcrypt. Автентифікація реалізована на основі JSON Web Tokens (JWT): після успішного входу сервер видає токен, який клієнт додає до заголовків усіх наступних запитів. На стороні NestJS доступ до захищених ресурсів контролюється через механізм Guard, що перевіряє дійсність токена.

На стороні мобільного клієнта глобальний стан застосунку управляється за допомогою Redux Toolkit. Це дозволяє зберігати в оперативній пам'яті пристрою поточну сесію користувача, завантажені списки вправ та колекцій, а також результати тренування до моменту їх синхронізації з сервером. У разі відсутності мережевого з'єднання мобільний клієнт використовує кешовані дані, а збереження нових результатів виконується після відновлення доступу до інтернету.

Основний інформаційний потік у мобільному застосунку організований таким чином: після автентифікації користувача клієнт завантажує каталог вправ та колекцій із сервера та зберігає їх у Redux-стані. Під час виконання тренування результати (кількість повторень, час) фіксуються локально. Після завершення сесії сформований об'єкт WorkoutSession із вкладеними ExerciseResult передається на сервер одним POST-запитом, що мінімізує кількість мережевих звернень і знижує навантаження на з'єднання.

Таким чином, інформаційне забезпечення мобільного застосунку побудоване на принципах мінімальної надмірності, безпеки зберігання персональних даних та стійкості до переривань мережевого з'єднання, що забезпечує надійну та зручну роботу мобільного застосунку в умовах реального використання.

## 2.4 Побудова UML-діаграм для візуалізації архітектури застосунку

UML-діаграми є потужним інструментом для формального опису архітектури програмного забезпечення [7]. У рамках кваліфікаційної роботи побудовано три ключові типи діаграм: діаграму варіантів використання, діаграму класів та діаграму послідовності. Кожна з них відображає окремий аспект функціонування мобільного застосунку та дозволяє краще зрозуміти взаємодію між її компонентами.


Рисунок 2.1 - Діаграма варіантів використання

Діаграма варіантів використання визначає функціональні можливості мобільного застосунку з точки зору кінцевого користувача. Основними акторами є: незареєстрований користувач, авторизований користувач та адміністратор мобільного застосунку. Незареєстрований користувач може переглянути загальну

інформацію про мобільний застосунок, виконати реєстрацію або авторизацію. Авторизований користувач має доступ до: перегляду каталогу вправ, виконання тренувальних програм, збереження результатів, перегляду власної статистики та налаштування профілю. Адміністратор може створювати та редагувати вправи і колекції, керувати обліковими записами користувачів та аналізувати загальну статистику.

Діаграма класів відображає статичну структуру мобільного застосунку та взаємозв'язки між класами. Основні класи мобільного застосунку:

- User (Користувач): поля – id, email, password, name, avatar, createdAt; методи – register(), login(), updateProfile(), getStatistics();

- Exercise (Вправа): поля – id, name, description, duration, repetitions, type, muscleGroup, difficulty; методи – getDetails(), getInstructions();

- Collection (Колекція): поля – id, title, description, exercises[], level; методи – getExercises(), addExercise(), removeExercise();

- WorkoutSession (Сесія тренування): поля – id, userId, collectionId, startTime, endTime, results[]; методи – start(), complete(), saveResult();

- ExerciseResult (Результат вправи): поля – id, sessionId, exerciseId, repetitionsDone, timeDone, date; методи – save(), getHistory().

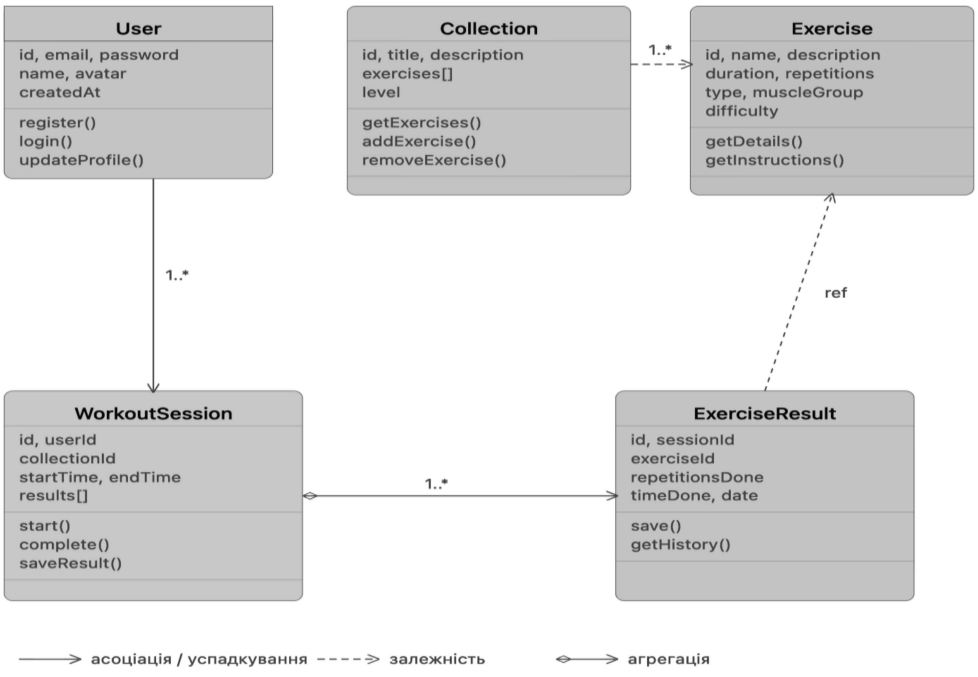


Рисунок 2.2 - Діаграма класів

Між класами визначено такі зв'язки: User має нуль або більше WorkoutSession (один до багатьох); WorkoutSession містить один або більше ExerciseResult (агрегація); Collection включає один або більше Exercise (асоціація); ExerciseResult посилається на конкретну Exercise.

Діаграма послідовності описує взаємодію між компонентами мобільного застосунку в часі для конкретного сценарію. Розглянемо сценарій «Виконання тренувальної програми»:

1) користувач відкриває список колекцій у мобільному застосунку. React Native клієнт надсилає GET-запит до REST API (/api/collections);

2) NestJS контролер отримує запит, передає його до сервісного шару CollectionService, який звертається до MongoDB і повертає список колекцій у форматі JSON;

3) користувач обирає конкретну колекцію та починає тренування. Мобільний застосунок запускає WorkoutSession та відображає вправу з таймером;

4) після виконання кожної вправи результати (кількість повторень, час) зберігаються локально. Після завершення сесії надсилається запит з результатами;

5) сервер зберігає сесію та результати до бази даних.

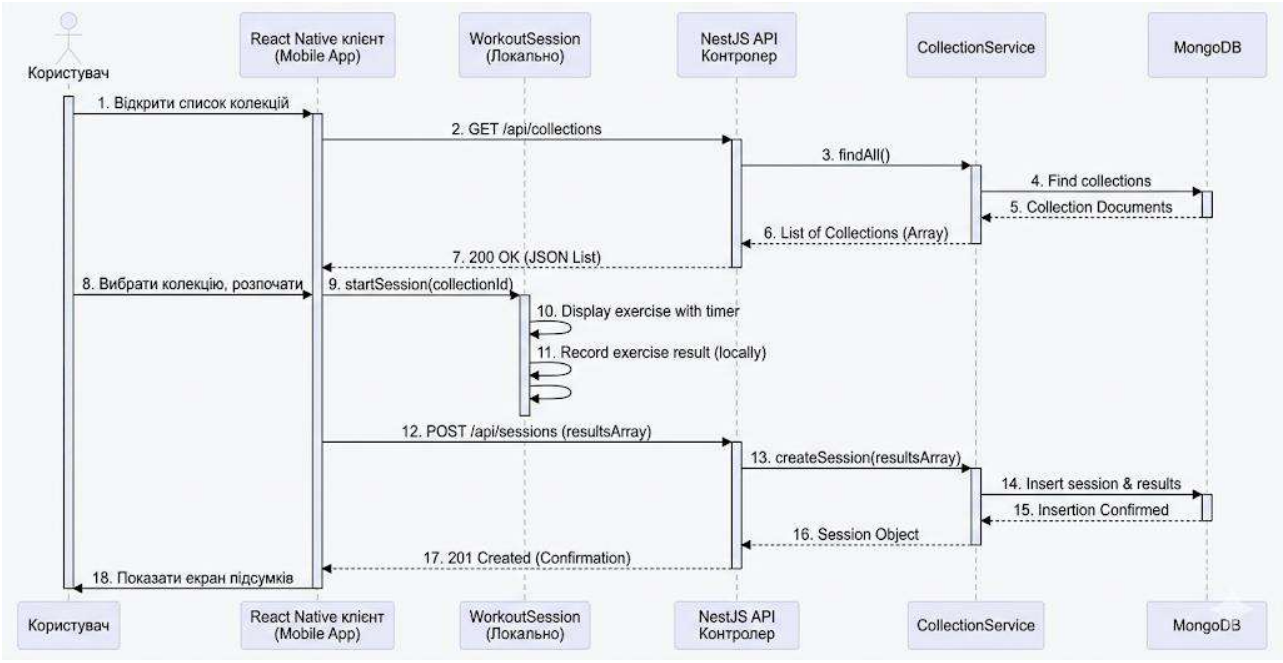


Рисунок 2.3 - Діаграма виконання тренувальної програми

Побудовані UML-діаграми дозволяють чітко розмежувати відповідальності між компонентами мобільного застосунку, визначити потоки даних та спростити процес програмної реалізації. Вони також слугують документацією, яка полегшує подальшу підтримку та розширення функціоналу застосунку. Використання уніфікованої мови моделювання в рамках даного проекту виступає фундаментальним етапом переддевелоперського аналізу, що дозволяє трансформувати абстрактні функціональні вимоги до фітнес-платформи у чіткі інженерні специфікації та графічні паттерни. Завдяки декомпозиції складної логіки взаємодії користувача з тренувальними модулями на окремі об'єктно-орієнтовані структури, вдається мінімізувати ризики виникнення архітектурних помилок на ранніх стадіях написання вихідного коду.

Розроблені графічні моделі, що включають діаграми прецедентів, класів та послідовностей, забезпечують наочне представлення внутрішньої архітектури системи як для розробників, так і для потенційних архітекторів рішень. Це створює надійну теоретичну базу для реалізації принципу модульності, спрощує інтеграцію сторонніх сервісів, таких як системи відстеження біометричних показників або геолокаційні модулі, та дозволяє чітко спрогнозувати поведінку програмного забезпечення у нестандартних сценаріях використання. Крім того, наявність деталізованого UML-опису значно знижує поріг входження для нових спеціалістів, які можуть бути залучені до супроводу або масштабування даного продукту в майбутньому.

Таким чином, графічне моделювання архітектурних рішень перетворюється з суто формального етапу проектування на ефективний інструмент контролю якості та оптимізації розробки. Візуалізація структурних зв'язків та динамічних процесів всередині мобільного застосунку дозволяє досягти високого рівня системної інтеграції, гарантуючи при цьому гнучкість, відмовостійкість та відповідність розробленого продукту сучасним стандартам індустрії цифрового фітнесу. Для детальнішого ознайомлення з побудованою архітектурною моделлю та взаємозв'язками між ключовими підсистемами програмного продукту, нижче наведено відповідний графічний матеріал.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

## 3.1 Програмна реалізація мобільного застосунку

Програмна реалізація мобільного застосунку здійснена відповідно до спроєктованої архітектури та включає клієнтську частину на React Native, серверну частину на NestJS та базу даних MongoDB. У цьому підрозділі розглядаються ключові аспекти реалізації основних модулів мобільного застосунку.

Клієнтська частина організована за принципом розділення відповідальностей: компоненти інтерфейсу, логіка управління станом, сервіси взаємодії з API та навігація винесені у відповідні директорії. Серверна частина побудована за модульним принципом NestJS: кожен ресурс мобільного застосунку (users, exercises, collections, sessions) має власний модуль з контролером, сервісом та DTO-класами для валідації вхідних даних.

Модуль автентифікації реалізовано з використанням JWT-токенів. При реєстрації користувача пароль хешується бібліотекою bcrypt перед збереженням у MongoDB. При авторизації сервер перевіряє відповідність введеного паролю збереженому хешу та у разі успіху генерує JWT-токен з терміном дії. На стороні клієнта токен зберігається у Redux-стані та автоматично додається до заголовка Authorization кожного HTTP-запиту через перехоплювач (interceptor) бібліотеки Axios. Захист серверних маршрутів забезпечується JwtAuthGuard, що перевіряє дійсність токена перед обробкою запиту.

Каталог вправ завантажується з сервера при першому відкритті відповідного екрану та кешується у Redux-стані для зменшення кількості мережевих запитів. GET-запит до кінцевої точки /api/exercises повертає масив об'єктів вправ у форматі JSON. На стороні клієнта реалізована фільтрація за групою м'язів та рівнем складності – фільтрація виконується локально на основі вже завантажених даних, що забезпечує миттєву реакцію інтерфейсу без додаткових запитів до сервера. Аналогічно організована робота з колекціями: список тренувальних програм завантажується одним запитом і зберігається у стані застосунку.

Екран виконання тренування є центральним функціональним модулем мобільного застосунку. При вході до екрану ініціалізується об'єкт WorkoutSession,

що містить ідентифікатор обраної колекції та масив для збору результатів. Для вправ із фіксованою тривалістю запускається таймер зворотного відліку, реалізований через setInterval з кроком одна секунда. Для силових вправ відображається інтерактивний лічильник повторень, що збільшується при натисканні на відповідну кнопку. Після завершення кожної вправи результат (кількість виконаних повторень або фактичний час виконання) зберігається локально через Redux action saveExerciseResult. Між вправами автоматично запускається таймер відпочинку тривалістю, яку обирає адмін. Після завершення останньої вправи у колекції накопичені результати передаються на сервер одним POST-запитом до /api/sessions, після чого користувач перенаправляється на екран підсумків тренування.

Статистика користувача формується на стороні сервера: при отриманні GET-запиту до /api/statistics/:userId сервіс агрегує дані з колекції workout_sessions та exercise_results для відповідного користувача. Повертається об'єкт з загальною кількістю тренувань, сумарним часом занять, розподілом активності за днями тижня та переліком найчастіше виконуваних вправ. На стороні клієнта отримані дані використовуються для побудови графіків і діаграм на екрані статистики.

Усі мережеві запити інкапсульовані у сервісний шар клієнтської частини. Сервісні функції викликаються з Redux Thunk-обробників, що дозволяє централізовано управляти станами завантаження (loading), успіху (success) та помилки (error) для кожної операції. У разі отримання від сервера статус-коду 401 (неавторизований) перехоплювач Axios автоматично перенаправляє користувача на екран авторизації.

Серверна частина на NestJS отримує HTTP-запити через контролери, делегує бізнес-логіку до сервісного шару та взаємодіє з MongoDB через ODM-бібліотеку Mongoose. Для кожної колекції бази даних визначено Schema-схему Mongoose, що задає структуру документів та типи полів. Вхідні дані у POST-запитах валідуються за допомогою DTO-класів з бібліотеки class-validator, що запобігає збереженню некоректних або неповних даних.

Таким чином, програмна реалізація мобільного застосунку охоплює повний цикл роботи мобільного застосунку – від автентифікації користувача до

збереження та аналізу результатів тренувань. Модульна організація коду як на клієнтській, так і на серверній стороні забезпечує зручність подальшої підтримки та розширення функціоналу мобільного застосунку.

## 3.2 Розробка графічного інтерфейсу користувача мобільного застосунку

Графічний інтерфейс мобільного застосунку розроблений з урахуванням принципів зручності використання (usability) та інтуїтивності навігації. Дизайн базується на концепції мінімалізму: кожен екран виконує одну чітку функцію, а візуальний стиль витриманий у темній кольоровій гамі з акцентними елементами.

Структура навігації застосунку побудована на двох рівнях. Перший рівень – автентифікаційний стек (Authentication Stack), що включає екрани входу та реєстрації. Після успішної авторизації користувач переходить до другого рівня – основного стеку (Main Stack) з вкладковою навігацією (Tab Navigation), де розміщені головні розділи застосунку.

Екран реєстрації та авторизації є першим, що бачить новий користувач. Він містить логотип застосунку, поля введення електронної пошти та паролю, кнопки «Увійти» та «Зареєструватися», а також посилання на відновлення паролю. Форма валідується у реальному часі: при введенні некоректних даних відображається відповідне повідомлення без необхідності надсилати форму.

Головний екран (Home Screen) вітає користувача та відображає короткий огляд активності: кількість тренувань за поточний тиждень, останню виконану колекцію вправ та рекомендовану програму на сьогодні. Реалізовано горизонтальний скрол банерів з рекомендованими колекціями.

Екран каталогу вправ (Exercises Screen) відображає повний список вправ, доступних у мобільному застосунку. Реалізовано фільтрацію за групою м'язів (ноги, груди, спина, руки, прес, кардіо) та рівнем складності (початковий, середній, просунутий). Кожна вправа представлена карткою з назвою, ілюстрацією та коротким описом. При натисканні на картку відкривається детальний екран вправи з повним описом техніки виконання, кількістю підходів та повторень.

Екран колекцій (Collections Screen) містить перелік тренувальних програм, впорядкованих за рівнем складності. Кожна колекція відображає: назву програми, загальну тривалість, кількість вправ та цільові групи м'язів. Користувач може обрати колекцію та перейти до її виконання.

Екран виконання тренування (Workout Screen) є ключовим функціональним екраном застосунку. Він відображає поточну вправу з анімованою ілюстрацією або відео, назву вправи та опис техніки, таймер зворотного відліку (для вправ із фіксованим часом), лічильник повторень (для силових вправ), кнопки «Пауза» та «Наступна вправа», а також індикатор загального прогресу тренування у вигляді смужки прогресу.

Під час виконання вправи користувач може вручну відзначати виконані повторення натисканням на велику круглу кнопку по центру екрану. При завершенні вправи автоматично розпочинається таймер відпочинку із налаштованою тривалістю паузи між підходами.

Екран статистики (Statistics Screen) надає користувачеві аналітичний огляд тренувальної активності. Відображаються: загальна кількість виконаних тренувань, сумарний час занять, улюблені вправи та колекції, графік активності за останні 30 днів (кількість тренувань по днях тижня), діаграма розподілу вправ за групами м'язів.

Екран профілю (Profile Screen) дозволяє переглянути та відредагувати особисті дані користувача: ім'я, аватар, електронну пошту. Також тут розміщено налаштування сповіщень, кнопку виходу з облікового запису та посилання на умови використання.

Усі компоненти інтерфейсу реалізовані з дотриманням принципу відповідності платформі: на Android використовуються Material Design-елементи, на iOS – елементи у стилі Human Interface Guidelines. Це досягається завдяки нативним компонентам React Native, які автоматично адаптуються до цільової платформи.

Представлення кодової реалізації можна знайти в додатку А.

## 3.3 Тестування мобільного застосунку та оцінка його стабільності

Тестування є невід'ємною частиною процесу розробки програмного забезпечення. У рамках даного проєкту проведено кілька рівнів тестування: модульне (unit testing), інтеграційне та системне тестування.

Модульне тестування було застосовано для перевірки окремих функцій та компонентів мобільного застосунку. Для серверної частини на NestJS використано фреймворк Jest, що є стандартним інструментом тестування для Node.js-додатків. Тестувалися: логіка формування тренувальних сесій, алгоритми підрахунку статистики, валідація вхідних даних у DTO-класах, функції обчислення прогресу користувача.

Для мобільного клієнта проведено тестування ключових Redux-редюсерів та селекторів за допомогою Jest. Перевірялася коректність оновлення стану при різних діях: авторизації, завантаженні вправ, збереженні результатів тренування.

Інтеграційне тестування охопило взаємодію між мобільним клієнтом та REST API. Для цього використовувався інструмент Postman, за допомогою якого тестувалися всі кінцеві точки API:

- POST /api/auth/register – реєстрація нового користувача;
- POST /api/auth/login – авторизація та отримання JWT-токена;
- GET /api/exercises – отримання списку вправ;
- GET /api/collections – отримання списку колекцій;
- POST /api/sessions – збереження результатів тренувальної сесії;
- GET /api/statistics/:userId – отримання статистики користувача.

Усі кінцеві точки показали коректну роботу: повертали очікувані статус-коди (200, 201, 400, 401, 404) та структуровані JSON-відповіді відповідно до специфікації.

Системне тестування проводилось на реальних пристроях: смартфоні Android (Samsung Galaxy A54, Android 14) та симуляторі iOS (iPhone 15, iOS 17). Тестувалися такі сценарії використання:

- реєстрація та авторизація нового користувача;
- перегляд каталогу вправ та застосування фільтрів;

- вибір тренувальної програми та її повне виконання;

- перегляд статистики після завершення тренування;

- редагування профілю користувача;

- коректна робота при відсутності інтернет-з'єднання (відображення збережених даних).

Результати системного тестування наведено у таблиці 3.2.

Таблиця 3.2 – Результати системного тестування

| Сценарій використання | Очікуваний результат | Фактичний результат | Статус |
|---|---|---|---|
| Реєстрація та авторизація нового користувача | Успішне створення облікового запису та отримання JWT-токена | JWT-токен успішно зберігається, перенаправлення на головний екран. | Успішно |
| Перегляд каталогу вправ та фільтрів | Коректне відображення списку з можливістю фільтрації за категоріями | Дані завантажуються менш ніж за 1 секунду, фільтри працюють без затримок. | Успішно |
| Вибір програми та її виконання | Запуск WorkoutSession, відображення вправ із таймером | Перехід до сесії відбувається миттєво, таймер працює стабільно. | Успішно |
| Перегляд статистики після тренування | Оновлення та коректне відображення | Дані успішно зберігаються (локально та на сервері), екран | Успішно |
| Редагування профілю користувача | Зміна даних (ім'я, налаштування) та їх збереження у базі даних. | Дані оновлюються в базі та синхронізуються з Redux-станом. | Успішно |
| Робота в автономному режимі (Offline) | Відображення збережених локально даних за відсутності мережі. | Мобільний клієнт коректно використовує кешовані дані без помилок. | Успішно |

Реєстрація та авторизація – пройдено успішно на обох платформах, користувач зміг перейти в додаток та побачити свій профіль. Перегляд каталогу вправ – пройдено, фільтрація працює коректно. Виконання тренувальної програми – пройдено, таймер та лічильник повторень функціонують без збоїв. Збереження результатів – пройдено, дані коректно зберігаються у MongoDB. Перегляд статистики – пройдено, графіки відображаються коректно. Робота без мережі – часткова: відображаються раніше завантажені дані, збереження нових результатів відбувається при відновленні з'єднання, так як повного кешування в додатку реалізовано не було.

Тестування продуктивності показало, що час завантаження списку вправ (50 записів) становить у середньому 320 мс при наявності стабільного з'єднання з інтернетом, що є досить стандартним результатом в порівнянні з іншими застосунками. Вказаний часовий інтервал повністю задовольняє сучасні вимоги до UI/UX-дизайну, оскільки затримка є практично непомітною для кінцевого користувача і не викликає відчуття "зависання" інтерфейсу під час динамічної взаємодії з контентом.

Час повного холодного запуску (Cold Start) мобільного застосунку становить близько 2,1 секунди на операційній системі Android та 1,8 секунди на платформі iOS. Подібна різниця у швидкодії між двома популярними екосистемами є цілком очікуваною і зумовлена специфікою архітектурної реалізації підсистем керування пам'яттю та механізмами ініціалізації процесів усередині кожної з ОС.

Показники споживання оперативної пам'яті (RAM) під час активного тренувального процесу та паралельного трекінгу активності фіксуються в межах 180 МБ. Варто детальніше пояснити таку структуру розподілу пам'яті: досить велика частина виділеного ОЗП витрачається безпосередньо на забезпечення життєдіяльності середовища виконання та запуск внутрішніх механізмів інтерпретації коду, аналогічних принципам роботи Node.js у веб-інфраструктурі (зокрема, роботу JavaScript-рушія Hermes або JavaScriptCore, що створюють необхідний міст (bridge) між кросплатформеним кодом та нативними API пристрою).

# ВИСНОВКИ

У результаті виконання кваліфікаційної роботи розроблено прототип кросплатформеного мобільного застосунку для виконання спортивних вправ та самоконтролю, що повністю відповідає поставленим завданням.

У процесі роботи вирішено такі завдання:

1. Проведено аналіз предметної області та огляд сучасного ринку мобільних додатків для занять спортом. Розглянуто провідні рішення (Nike Training Club, Fitbod, Strong), визначено їхні переваги та обмеження. Встановлено, що більшість існуючих додатків не забезпечують гнучкого управління контентом та не дозволяють налаштовувати тренувальні програми відповідно до наявного обладнання.

2. Обґрунтовано вибір технологічного стеку для розробки. Визначено, що React Native є оптимальним фреймворком для кросплатформеної мобільної розробки завдяки широкій екосистемі, підтримці нативних компонентів та великій кількості кваліфікованих розробників. NestJS та MongoDB обрано для реалізації серверної частини.

3. Спроєктовано архітектуру мобільного застосунку відповідно до принципу розділення відповідальностей: інтерфейс користувача (React Native), модуль бізнес-логіки, модуль роботи з даними та REST API. Розроблено структуру бази даних з колекціями для користувачів, вправ, тренувальних програм та результатів.

4. Побудовано UML-діаграми: діаграму варіантів використання, діаграму класів та діаграму послідовності, які відображають функціональні вимоги до мобільного застосунку, статичну структуру та динаміку взаємодії компонентів.

5. Реалізовано прототип мобільного застосунку з такими модулями: авторизація та реєстрація користувачів, каталог вправ з фільтрацією, колекції тренувальних програм, екран виконання тренування з таймером та лічильником повторень, статистика активності користувача, профіль користувача.

6. Проведено тестування мобільного застосунку на рівнях модульного, інтеграційного та системного тестування. Тести виконано на пристроях Android та iOS. Підтверджено стабільність роботи мобільного застосунку, коректну взаємодію

між клієнтською та серверною частинами, відповідність функціональних можливостей вимогам технічного завдання.

7. Практична цінність розробленого прототипу полягає у можливості його використання широким колом користувачів для самостійної організації тренувального процесу, контролю прогресу та підтримки мотивації. Кросплатформена реалізація забезпечує охоплення як Android, так і iOS аудиторії.

8. Перспективами подальшого розвитку мобільного застосунку є: інтеграція з портативними пристроями (Apple Health, Google Fit), додавання функції відеоінструкцій для кожної вправи, реалізація соціальних функцій (рейтинги, поширення досягнень), впровадження персоналізованих рекомендацій на основі машинного навчання.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Facebook Open Source. React Documentation: Build Every Platform. URL: https://react.dev/
2. Meta Platforms Inc. React Native: Learn once, write anywhere. URL: https://reactnative.dev/
3. NestJS Group. Documentation: A progressive Node.js framework. URL: https://docs.nestjs.com/
4. Amazon Web Services. AWS Documentation: Cloud Computing Services. URL: https://docs.aws.amazon.com/
5. MongoDB, Inc. MongoDB Manual: The document database. URL: https://www.mongodb.com/docs/
6. Sanity.io. Sanity Documentation: Structured Content and Headless CMS. URL: https://www.sanity.io/docs
7. Мартін Р. Чиста архітектура. Мистецтво розробки програмного забезпечення. Фабула, 2019. 352 с.
8. Ньюмен С. Створення мікросервісів. O'Reilly Media, 2015. 368 с.
9. Фоулер М. Шаблони архітектури корпоративних програмних додатків. Addison-Wesley, 2002. 560 с.
10. Kim G., Humble J. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. IT Revolution Press, 2016. 480 p.
11. Playwright Documentation. End-to-End Testing for Modern Web Apps. URL: https://playwright.dev/
12. Detox Documentation. Gray box end-to-end testing and automation framework for mobile apps. URL: https://wix.github.io/Detox/
13. Docker Inc. Docker Overview: Containerization and Development. URL: https://docs.docker.com/
14. Nike Training Club (NTC). Case Study: Mobile UI/UX Research. URL: https://www.nike.com/ntc-app

15. Fitbod Inc. Engineering Blog: Adaptive Workout Algorithms. URL: https://fitbod.me/

16. Sood A. Serverless Media Processing with AWS. Packt Publishing, 2021. 312 p.

17. Redis Documentation. Data structures and caching strategies. URL: https://redis.io/docs/

18. Vercel Documentation. Deployment and CI/CD for Frontend applications. URL: https://vercel.com/docs

19. GitHub Actions Documentation. Automating your workflow with CI/CD. URL: https://docs.github.com/en/actions

20. Prisma ORM. Database toolkit for TypeScript and Node.js. URL: https://www.prisma.io/docs

21. Reanimated Documentation. Advanced Physics-Based Animation Engine for React Native. URL: https://docs.swmansion.com/react-native-reanimated/

22. Mongoose ODM Documentation. URL: https://mongoosejs.com/docs/

23. Maestro Documentation. Mobile UI Automation Framework. URL: https://maestro.mobile.dev/

24. Ворожбит Р., Турченко І.В. Кросплатформенний мобільний застосунок для виконання спортивних вправ та самоконтролю. ICSPM 2026: Intelligent Computing Systems and Project Management / Інтелектуальні обчислювальні системи та управління проєктами (Тернопіль, 21-22 травня 2026 р). Тернопіль: ЗУНУ, 2026. С. 129-131.

25. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп'яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

Додаток А

# ПРОГРАМНИЙ КОД КЛЮЧОВИХ КОМПОНЕНТІВ МОБІЛЬНОГО ЗАСТОСУНКУ

## А.1 Компонент екрану виконання тренування (WorkoutScreen.tsx)

```
import { View, Text, TouchableOpacity, StyleSheet } from 'react-native';
import { useSelector, useDispatch } from 'react-redux';
import { saveExerciseResult } from '../store/workoutSlice';

interface WorkoutScreenProps {
  route: any;
  navigation: any;
}

const WorkoutScreen: React.FC<WorkoutScreenProps> = ({ route, navigation }) => {
  const { collection } = route.params;
  const dispatch = useDispatch();
  const [currentIndex, setCurrentIndex] = useState(0);
  const [repetitions, setRepetitions] = useState(0);
  const [timeLeft, setTimeLeft] = useState(0);
  const [isResting, setIsResting] = useState(false);
  const timerRef = useRef<any>(null);
  const currentExercise = collection.exercises[currentIndex];

  useEffect(() => {
    if (currentExercise?.duration) {
      setTimeLeft(currentExercise.duration);
      startTimer();
    }
    return () => clearInterval(timerRef.current);
  }, [currentIndex]);

  const startTimer = () => {
    timerRef.current = setInterval(() => {
      setTimeLeft(prev => {
        if (prev <= 1) {
          clearInterval(timerRef.current);
```

```
        handleExerciseComplete();
        return 0;
      }
      return prev - 1;
    });
  }, 1000);
};
const handleRepetition = () => setRepetitions(prev => prev + 1);
const handleExerciseComplete = () => {
  dispatch(saveExerciseResult({
    exerciseId: currentExercise.id,
    repetitionsDone: repetitions,
    timeDone: currentExercise.duration - timeLeft,
  }));
  if (currentIndex < collection.exercises.length - 1) {
    setIsResting(true);
    setTimeout(() => {
      setIsResting(false);
      setCurrentIndex(prev => prev + 1);
      setRepetitions(0);
    }, 30000);
  } else {
    navigation.navigate('WorkoutSummary');
  }
};

return (
  <View style={styles.container}>
    <Text style={styles.exerciseName}>{currentExercise?.name}</Text>
    {currentExercise?.duration ? (
      <Text style={styles.timer}>{timeLeft}c</Text>
    ) : (
      <TouchableOpacity style={styles.repButton} onPress={handleRepetition}>
        <Text style={styles.repCount}>{repetitions}</Text>
      </TouchableOpacity>
    )}
    <TouchableOpacity style={styles.nextButton} onPress={handleExerciseComplete}>
      <Text>Завершити вправу</Text>
    </TouchableOpacity>
  </View>
```

```
  );
};
```

A.2 Контролер REST API для управління вправами (exercises.controller.ts)

```
import { Controller, Get, Post, Body, Param, UseGuards } from '@nestjs/common';
import { ExercisesService } from './exercises.service';
import { CreateExerciseDto } from './dto/create-exercise.dto';
import { JwtAuthGuard } from '../auth/jwt-auth.guard';

@Controller('exercises')
export class ExercisesController {
  constructor(private readonly exercisesService: ExercisesService) {}

  @Get()
  @UseGuards(JwtAuthGuard)
  findAll() {
    return this.exercisesService.findAll();
  }

  @Get(':id')
  @UseGuards(JwtAuthGuard)
  findOne(@Param('id') id: string) {
    return this.exercisesService.findOne(id);
  }

  @Post()
  @UseGuards(JwtAuthGuard)
  create(@Body() createExerciseDto: CreateExerciseDto) {
    return this.exercisesService.create(createExerciseDto);
  }
}
```

Додаток Б

Копія опублікованих результатів

**Міністерство освіти і науки України**
**Західноукраїнський національний університет**
**Факультет комп'ютерних інформаційних технологій**
**Кафедра інформаційно-обчислювальних систем і управління**

**Матеріали**

І Міжнародної науково-практичної конференції студентів і молодих вчених

**«ICSPM 2026: Intelligent Computing Systems and Project Management / Інтелектуальні обчислювальні системи та управління проєктами»**

**21–22 травня 2026 р.**


м. Тернопіль - 2026

2

УДК 004.5: 004.77

# КРОСПЛАТФОРМЕНИЙ МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ВИКОНАННЯ СПОРТИВНИХ ВПРАВ ТА САМОКОНТРОЛЮ

Ворожбит Руслан Олександрович,
здобувач першого (бакалаврського) рівня вищої освіти,
vorozhbytr21@gmail.com

Турченко Ірина Василівна,
к.т.н., доцент, доцент кафедри інформаційно-обчислювальних систем і управління,
itu@wunu.edu.ua
ORCID: 0000-0002-9441-6669
Західноукраїнський національний університет

Сучасний розвиток цифрових технологій суттєво впливає на сферу фізичної активності та здорового способу життя. Зокрема, все більшої популярності набувають мобільні застосунки для організації тренувального процесу, які дають змогу користувачам виконувати спортивні вправи самостійно без необхідності відвідування тренажерних залів або залучення персонального тренера. Особливої актуальності такі сервіси набули після пандемії COVID-19, коли значна частина користувачів почала активно використовувати цифрові платформи для занять спортом у домашніх умовах [1].

Основною перевагою сучасних мобільних застосунків для фітнесу є можливість швидкого доступу до структурованих тренувальних програм, відстеження власного прогресу, отримання інструкцій щодо техніки виконання вправ та фіксації результатів у режимі реального часу. Такі системи сприяють підвищенню мотивації користувачів та забезпечують доступність спортивних послуг для людей з обмеженим часом або фінансовими ресурсами для відвідування спортивних закладів [2].

Попри активний розвиток цифрових сервісів у сфері фітнесу, існуючі рішення мають низку недоліків. Частина мобільних застосунків характеризується перевантаженим інтерфейсом, складною навігацією або надмірною комерціалізацією – значний обсяг функціоналу доступний лише за платною підпискою. Крім цього, окремі сервіси не забезпечують належного рівня гнучкості у формуванні тренувальних програм відповідно до наявного обладнання користувача або його рівня фізичної підготовки [3].

Аналіз сучасних мобільних фітнес-застосунків, серед яких розглядалися Nike Training Club, Fitbod та Strong [4, 5], вказують на те, що більшість сучасних сервісів мають подібний набір базових функцій: каталог вправ, тренувальні програми, таймер та лічильник повторень, збереження результатів (таблиця 1). Водночас значна частина платформ недостатньо орієнтована на кросплатформену доступність та відкрите управління контентом.

Таблиця 1 – Порівняльні характеристики застосунків

| Функціональна можливість | Nike Training Club | Fitbod | Strong |
|---|---|---|---|
| Каталог вправ | + | + | + |
| Тренувальні програми | + | + | частково |
| Таймер і лічильник повторення | + | + | + |
| Збереження результатів | + | + | + |
| Статистика активності | + | + | + |
| Персоналізація програм | частково | + | + |
| Нагадування про тренування | + | + | + |
| Відеоінструкції до вправ | + | немає | немає |
| Безкоштовний доступ | частково | частково | частково |
| Кросплатформена реалізація | + | + | + |

Розробка прототипу кросплатформеного мобільного застосунку для виконання спортивних вправ та самоконтролю, який забезпечує користувачам можливість перегляду каталогу вправ, виконання структурованих тренувальних програм, відстеження власного прогресу та збереження результатів через зручний і зрозумілий інтерфейс, є актуальною задачею сучасних інформаційних технологій у сфері цифрового здоров'я та фітнесу.

У процесі дослідження було визначено основні вимоги до проєктованого застосунку. Одним із ключових аспектів є створення інтуїтивно зрозумілого інтерфейсу, який дозволить користувачу швидко виконувати основні дії: обрати тренувальну програму, виконати вправу з таймером або лічильником повторень та переглянути власну статистику. Особлива увага надається мінімізації кількості кроків під час взаємодії із системою та адаптації інтерфейсу до платформ Android та iOS.

Запропонований застосунок орієнтований на реалізацію сучасного технологічного стеку для кросплатформеної мобільної розробки. Клієнтська частина реалізована на React Native з використанням TypeScript та Redux Toolkit для управління станом. Серверна частина побудована на основі NestJS з MongoDB як системою управління базами даних. Така архітектура забезпечує масштабованість системи та незалежне розгортання клієнтської і серверної складових.

Функціональна структура сервісу передбачає наявність основних модулів: каталогу вправ з фільтрацією за групою м'язів та рівнем складності, колекцій тренувальних програм, екрану виконання тренування з таймером зворотного

відліку та лічильником повторень, статистики активності користувача та персонального кабінету.

Діаграма варіантів використання, що визначає функціональні можливості системи з точки зору кінцевого користувача, представлена на рисунку 1.


Рисунок 1 – Діаграма варіантів використання

Тестування розробленого прототипу проводилося на пристроях Android (Samsung Galaxy A54, Android 14) та симуляторі iOS (iPhone 15, iOS 17). За результатами системного тестування підтверджено коректну роботу всіх основних сценаріїв використання: авторизації, перегляду каталогу, виконання тренувальної програми та збереження результатів. Середній час завантаження списку вправ становив 320 мс, час запуску застосунку – близько 2,1 секунди на Android та 1,8 секунди на iOS.

Висновки: результати проведеного дослідження підтверджують актуальність розробки спеціалізованих мобільних застосунків для самостійної організації тренувального процесу та демонструють важливість якісного технічного проєктування для забезпечення стабільної та зручної взаємодії користувача із системою. Розроблений прототип є основою для подальшої реалізації повноцінного комерційного продукту у сфері цифрових фітнес-послуг.

## Список використаних джерел

1. World Health Organization. Physical activity. 2022. URL: https://www.who.int/news-room/fact-sheets/detail/physical-activity

2. Appinventiv. Fitness App Development: A Complete Guide. 2024. URL: https://appinventiv.com/blog/fitness-app-development/

3. Masiello E., Friedman J. Mastering React Native. Packt Publishing, 2017. 420 p.

4. Nike Training Club. URL: https://www.nike.com/ntc-app

5. Fitbod: Weight Lifting & Fitness. URL: https://fitbod.me