Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту

Кафедра інтелектуальних програмних систем і технологій

До захисту допущено
Кафедрою ІПСіТ протокол №___ від_____

завідувач кафедри _____ Олег ОЛЕШКО
(підпис) *(ім'я, прізвище)*

«___»_____ 2026 р.

Кваліфікаційна робота
здобувача першого (бакалаврського) рівня вищої освіти

«Розробка клієнт-серверного мобільного застосунку для моніторингу та аналізу спортивних активностей»
(назва роботи)

Спеціальність (спеціалізація) 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

Виконавець _____ Гладкоскок Максим
(підпис) *(ім'я, прізвище)*

Науковий керівник _____ Коршенко Владислав
(підпис) *(ім'я, прізвище)*

Харків – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра Математичного моделювання та аналізу даних
Рівень вищої освіти (освітньо-кваліфікаційний рівень) Бакалавр
Галузь знань: F – Інформаційні технології
Спеціальність: F2 – Комп'ютерні науки
Освітня програма: _Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри _ММ та АД_
Струков Володимир Михайлович
«_13__» _жовтня_2025_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЕКТ)

Гладкоскок Максим Олексійович
(прізвище, ім'я, по батькові студента)

1. Тема роботи: Розробка клієнт-серверного мобільного застосунку для моніторингу та аналізу спортивних активностей.

   керівник роботи Коршенко Владислав Сергійович, викладач кафедри ММ та АД. _____,
   (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "___"_____20__року
№___

2. Строк подання студентом роботи_____

3. Перелік питань, які потрібно розробити
   1) Огляд та аналіз існуючих програмних рішень для відстеження фізичної активності.
   2) Формування функціональних та нефункціональних вимог до мобільного застосунку на платформах Android та iOS.
   3) Проєктування архітектури клієнт-серверної системи та структури реляційної бази даних.
   4) Розробка серверної частини з використанням фреймворку Laravel та реалізація REST API.
   5) Документування розробленого програмного інтерфейсу (API) за стандартом OpenAPI 3.0.

6) Проєктування інтерфейсу користувача та розробка клієнтської частини засобами Flutter і Dart.
7) Програмна реалізація модулів геолокації (GPS-маршрут), розрахунку статистичних даних (дистанція, темп, швидкість) та фонової роботи застосунку.
8) Тестування розробленого програмного забезпечення та (конвертація в .apk)

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Аналіз предметної області, огляд аналогів та затвердження технічного завдання.	15.10.2025 – 10.11.2025
2	Проєктування архітектури системи, бази даних та клієнт-серверної взаємодії.	11.11.2025 – 15.12.2025
3	Розробка бекенду (PHP/Laravel), реалізація та документування REST API.	16.12.2025 – 10.02.2026
4	Розробка UI/UX та базового функціоналу мобільного клієнта (Flutter/Dart).	11.02.2026 – 20.03.2026
5	Інтеграція модулів GPS-трекінгу, фонової роботи та синхронізації з сервером.	21.03.2026 – 20.04.2026
6	Комплексне тестування, виправлення помилок та оптимізація системи.	21.04.2026 – 10.05.2026
7	Оформлення тексту пояснювальної записки кваліфікаційної роботи.	11.05.2026 – 25.05.2026
8	Підготовка презентаційних матеріалів, графічної частини та нормоконтроль.	26.05.2026 – 05.06.2026

5. Дата видачі завдання 13.10.2025

Студент Гладкоскок М.О. _____
ініціали, прізвище підпис

Керівник роботи Коршенко В.С. _____
ініціали, прізвище підпис

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг роботи складає 65 сторінок, із яких 53 сторінок основної частини, 21 рисунків, 10 таблиць, 20 найменувань списку використаних джерел та 1 додаток.

Мета роботи є розробка клієнт-серверного мобільного застосунку для моніторингу та аналізу спортивних активностей, що забезпечує запис тренувань, побудову GPS-маршруту, збереження статистичних показників та взаємодію із серверною частиною.

Об'єкт дослідження – процес моніторингу спортивної активності користувачів із використанням мобільних інформаційних технологій.

Предмет дослідження – методи, технології та програмні засоби розробки клієнт-серверного мобільного застосунку для збору, збереження, обробки та аналізу даних спортивної активності.

Визначення функціональних можливостей мобільного застосунку для ефективного моніторингу спортивної активності, зокрема запису тренувань, перегляду історії занять, аналізу статистики, системи рейтингу користувачів та підтримки профілю користувача.

Область застосування – моніторинг фізичної активності, аналіз спортивних показників користувача, підтримка здорового способу життя та подальший розвиток мобільних систем спортивного моніторингу.

Ключові слова: МОБІЛЬНИЙ ЗАСТОСУНОК, СПОРТИВНА АКТИВНІСТЬ, МОНІТОРИНГ, GPS, FLUTTER, DART, REST API, ГЕОЛОКАЦІЯ, КЛІЄНТ-СЕРВЕРНИЙ ЗАСТОСУНОК.

ABSTRACT

The explanatory note for the bachelor's qualification work consists of an introduction, four chapters, conclusions, a list of references, and appendices. The total volume of the work is 65 pages, including 53 pages of the main part, 21 figures, 10 tables, 20 references, and 1 appendice.

The purpose of the work is the development of a client-server mobile application for monitoring and analyzing sports activities, which provides workout recording, GPS route tracking, storage of statistical indicators, and interaction with the server side.

The object of research is the process of monitoring sports activity using mobile information technologies.

The subject of research is methods, technologies, and software tools for developing a client-server mobile application for collecting, storing, processing, and analyzing sports activity data.

Determination of the functional capabilities of a mobile application for effective sports activity monitoring, including workout recording, activity history viewing, statistics analysis, user ranking, and profile support.

Application area – monitoring physical activity, analysis of sports indicators, support for a healthy lifestyle, and further development of mobile sports monitoring systems.

Keywords: MOBILE APPLICATION, SPORTS ACTIVITY, MONITORING, GPS, FLUTTER, DART, REST API, GEOLOCATION, CLIENT-SERVER APPLICATION.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ 5

ВСТУП 6

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ 8

1.1 Особливості моніторингу спортивної активності. 8

1.2 Аналіз сучасних застосунків для відстеження спортивних активностей. 13

1.3 Порівняння технологій реалізації мобільних застосунків. 15

1.4 Постановка задачі дослідження. 19

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА КЛІЄНТ-СЕРВЕРНОГО МОБІЛЬНОГО ЗАСТОСУНКУ 19

2.1 Загальна архітектура системи. 19

2.2 Проєктування структури даних застосунку. 20

2.3 Проєктування навігації та структури інтерфейсу. 23

2.4 Організація взаємодії з REST API. 33

2.5 Організація керування станом застосунку. 35

2.6 Обґрунтування вибору технологій та програмних засобів. 37

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ МОБІЛЬНОГО ЗАСТОСУНКУ 40

3.1 Реалізація функціональних можливостей застосунку. 40

3.2 Тестування мобільного застосунку. 46

3.3 Аналіз результатів роботи системи. 49

ВИСНОВОК 51

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 53

ДОДАТКИ 55

Додаток А 55

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface;

REST – Representational State Transfer;

GPS – Global Positioning System;

JSON – JavaScript Object Notation;

HTTP – HyperText Transfer Protocol;

UI – User Interface;

UX – User Experience;

OSM – OpenStreetMap;

Dart – мова програмування для розробки мобільних застосунків;

Flutter – фреймворк для розробки кросплатформених мобільних застосунків;

REST API – програмний інтерфейс взаємодії між клієнтською та серверною частинами застосунку.

ВСТУП

У сучасних умовах розвитку інформаційних технологій мобільні застосунки стали важливою складовою повсякденного життя людини. Особливого поширення набули цифрові рішення, спрямовані на підтримку здорового способу життя, моніторинг фізичної активності та аналіз спортивних показників. Використання мобільних пристроїв із вбудованими засобами геолокації, доступом до мережі Інтернет та можливістю обробки даних у режимі реального часу дозволяє створювати функціональні системи контролю спортивної діяльності користувачів.

Популяризація занять спортом, бігом, велоспортом та активним способом життя зумовила підвищений попит на програмні рішення, що забезпечують запис тренувань, збереження історії активностей, аналіз статистичних показників і формування мотиваційних механізмів у вигляді рейтингів та досягнень. Існуючі застосунки для моніторингу спортивної активності надають широкий набір функцій, однак часто характеризуються надмірною складністю інтерфейсу, обмеженим доступом до окремих можливостей або залежністю від платних підписок.

Актуальність теми кваліфікаційної роботи обумовлена необхідністю створення сучасного клієнт-серверного мобільного застосунку, що дозволяє користувачеві здійснювати моніторинг спортивної активності, аналізувати результати тренувань, переглядати статистичні показники та взаємодіяти із системою в зручному та зрозумілому інтерфейсі. Додатковою перевагою є використання засобів геолокації для побудови маршруту переміщення, збереження параметрів активності та формування історії тренувань.

У межах кваліфікаційної роботи розглядається розробка клієнт-серверного мобільного застосунку для моніторингу та аналізу спортивних активностей, що забезпечує реєстрацію та авторизацію користувачів, запис тренувань із використанням GPS-координат, побудову маршруту на карті,

збереження статистичних показників, перегляд історії активностей, систему рейтингу користувачів та підтримку багатомовного інтерфейсу.

Об’єктом дослідження є процес моніторингу та аналізу спортивної активності користувачів із використанням мобільних інформаційних технологій.

Предметом дослідження є методи, програмні засоби та технології розробки клієнт-серверного мобільного застосунку для збору, обробки, збереження та візуалізації даних спортивних активностей.

Метою роботи є розробка клієнт-серверного мобільного застосунку для моніторингу та аналізу спортивних активностей, який забезпечує запис маршрутів тренувань, відображення статистики користувача, перегляд історії занять та взаємодію із серверною частиною для збереження і обробки даних.

Для досягнення поставленої мети необхідно виконати такі **завдання**:

– провести аналіз предметної області та існуючих програмних рішень для моніторингу спортивної активності;

– дослідити сучасні технології розробки мобільних клієнт-серверних застосунків;

– спроєктувати структуру мобільного застосунку, навігацію та основні програмні модулі;

– реалізувати систему реєстрації, авторизації та керування профілем користувача;

– реалізувати механізм запису спортивної активності із використанням геолокації та побудови маршруту;

– реалізувати збереження, перегляд, сортування та фільтрацію історії тренувань;

– реалізувати систему статистики та рейтингу користувачів;

– провести тестування функціональних можливостей застосунку та оцінити результати його роботи.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Особливості моніторингу спортивної активності.

У сучасних умовах розвитку інформаційних технологій значного поширення набули цифрові системи моніторингу фізичної активності, які дозволяють користувачам контролювати параметри тренувань, аналізувати результати фізичних навантажень та відстежувати зміни спортивних показників у динаміці. Активне використання мобільних пристроїв, оснащених вбудованими датчиками, геолокацією та доступом до мережі Інтернет, сприяло появі програмних засобів, здатних автоматизувати процес збору та аналізу інформації про фізичну активність людини. Дослідження показують, що використання мобільних застосунків позитивно впливає на мотивацію користувачів до занять спортом та сприяє підтримці регулярної фізичної активності.

Моніторинг спортивної активності являє собою процес збору, обробки, збереження та аналізу даних про фізичні навантаження користувача. До основних параметрів, які зазвичай фіксуються сучасними мобільними застосунками, належать дистанція, тривалість тренування, швидкість пересування, темп, маршрут, кількість виконаних вправ та інші статистичні показники. Для отримання таких даних використовуються вбудовані можливості смартфонів, зокрема GPS-модулі, акселерометри, гіроскопи та інші сенсори, що забезпечують можливість визначення місцезнаходження та оцінювання рухової активності користувача.

Сучасні системи спортивного моніторингу орієнтовані не лише на збереження інформації про тренування, але й на забезпечення зручної взаємодії користувача із даними. У більшості застосунків реалізовано механізми візуалізації маршрутів на карті, перегляду історії тренувань, аналізу статистики та порівняння результатів за певні періоди часу. Важливою особливістю є можливість накопичення інформації, що дозволяє

користувачеві оцінювати власний прогрес, аналізувати ефективність тренувального процесу та формувати індивідуальні спортивні цілі.

Окрему роль у сучасних системах спортивного моніторингу відіграє клієнт-серверна архітектура, за якої мобільний застосунок взаємодіє із серверною частиною для збереження, синхронізації та обробки інформації про фізичну активність. Такий підхід забезпечує централізоване зберігання даних, доступ до історії тренувань із різних пристроїв та можливість подальшого аналізу статистичних показників.

Ще однією особливістю сучасних застосунків для спортивного моніторингу є використання елементів мотивації користувача. До них можна віднести системи рейтингів, відображення особистих досягнень, ведення історії результатів та соціальні механізми взаємодії між користувачами. Практика показує, що наявність інструментів самоаналізу та візуалізації прогресу позитивно впливає на регулярність занять фізичною активністю.

Таким чином, моніторинг спортивної активності є важливою складовою сучасних мобільних технологій, що поєднує можливості збору, аналізу та візуалізації даних про фізичні навантаження користувача. Використання мобільних застосунків дозволяє не лише контролювати результати тренувань, але й сприяє підвищенню мотивації до занять спортом, формуванню здорових звичок та підтримці активного способу життя. Для узагальнення основних функціональних можливостей систем моніторингу спортивної активності сформовано таблицю 1.1, у якій наведено їх короткий опис.

Таблиця 1.1 - Функціональні можливості систем моніторингу спортивної активності

Функціональна можливість	**Опис**
GPS-моніторинг	Визначення маршруту пересування користувача під час тренування
Облік дистанції	Розрахунок пройденої відстані

Аналіз швидкості	Обчислення середньої швидкості та темпу
Історія тренувань	Збереження інформації про попередні активності
Статистика	Відображення показників фізичної активності
Рейтинг користувачів	Порівняння результатів між користувачами
Профіль користувача	Зберігання персональних даних

Отже, наведені у таблиці 1.1 функціональні можливості є базовими для сучасних систем моніторингу спортивної активності. Найбільш важливими серед них є запис маршруту пересування користувача, визначення статистичних показників тренування, збереження історії занять та можливість аналізу отриманих результатів. Використання таких функцій дозволяє користувачам здійснювати контроль фізичної активності, аналізувати власний прогрес та підтримувати регулярність тренувань.

Під час функціонування клієнт-серверного мобільного застосунку важливу роль відіграють протоколи зв’язку та механізми передачі даних між мобільним клієнтом і серверною частиною. Для забезпечення обміну інформацією про спортивну активність користувача використовується взаємодія через REST API, що дозволяє надсилати HTTP-запити для отримання, збереження, оновлення та видалення даних.

Передача інформації між компонентами системи здійснюється у форматі JSON, що забезпечує зручність структурування, обробки та обміну даними між мобільним застосунком і сервером. Такий підхід дозволяє забезпечити сумісність між різними компонентами системи, централізоване збереження даних та стабільну взаємодію із серверною частиною.

На рисунку 1.1 представлено загальну схему взаємодії клієнт-серверного мобільного застосунку під час передачі та обробки даних.

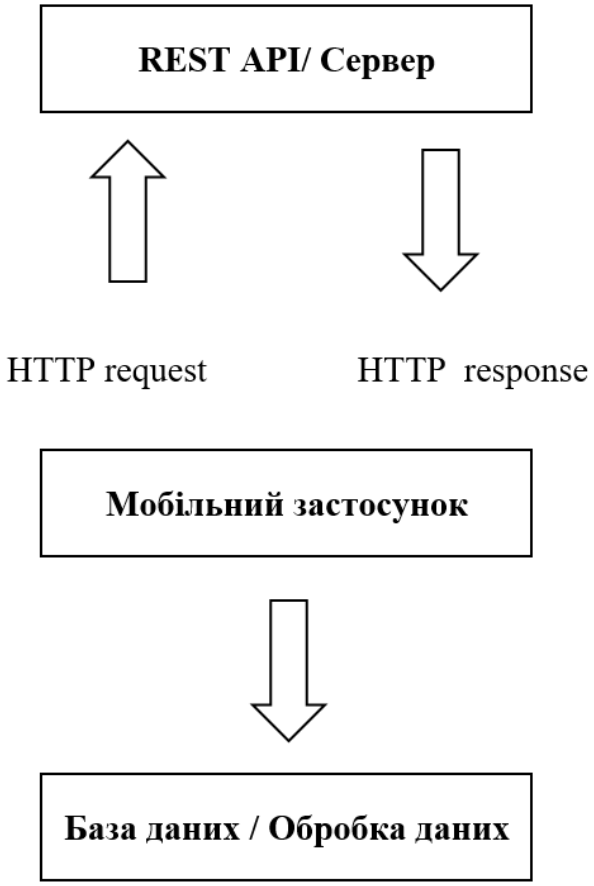


Рисунок 1.1 Загальна схема взаємодії клієнт-серверного мобільного застосунку.

Мобільний застосунок надсилає HTTP-запити до серверної частини через REST API, після чого сервер виконує обробку отриманих даних, взаємодіє з базою даних та повертає відповідь користувачу. Такий підхід забезпечує централізоване зберігання інформації, обробку даних спортивної активності та синхронізацію між компонентами системи.

Клієнт-серверний мобільний застосунок для моніторингу спортивної активності передбачає постійну взаємодію між мобільним клієнтом та серверною частиною. Сервер використовується для збереження інформації

про користувача, історії тренувань, статистичних показників та інших даних, необхідних для функціонування системи.

Під час роботи застосунок може надсилати запити для реєстрації та авторизації користувача, отримання інформації про профіль, збереження спортивної активності, перегляду статистики та історії тренувань. У відповідь сервер повертає необхідні дані для їх подальшого відображення в інтерфейсі мобільного застосунку.

Використання мобільних застосунків для моніторингу спортивної активності має низку переваг. Зокрема, користувач отримує можливість автоматизованого збору інформації про фізичну активність, перегляду статистичних показників, аналізу прогресу та збереження історії тренувань. Крім цього, використання геолокації дозволяє будувати маршрути пересування та визначати основні параметри тренування.

Водночас існують певні обмеження, пов’язані із точністю GPS-відстеження, залежністю від доступу до мережі Інтернет та технічними характеристиками мобільного пристрою. На якість роботи також можуть впливати помилки визначення геолокації, обмеження серверної взаємодії та енергоспоживання мобільного пристрою.

Таблиця 1.2 демонструє основні переваги використання мобільних застосунків для моніторингу спортивної активності. Використання таких систем дозволяє користувачам здійснювати контроль фізичних навантажень, аналізувати результати тренувань та підтримувати мотивацію до регулярних занять спортом.

Таблиця 1.2 - Переваги використання систем моніторингу спортивної активності

Переваги	**Опис**
Автоматизація	Автоматичний збір інформації про тренування.
Аналіз статистики	Можливість аналізувати дистанцію, швидкість, темп.

GPS-моніторинг	Побудова маршруту та визначення переміщення.
Історія тренувань	Збереження попередніх спортивних активностей.
Мотивація	Система рейтингу та контроль прогресу.
Доступність	Використання на мобільних пристроях у будь-який час.

1.2 Аналіз сучасних застосунків для відстеження спортивних активностей.

Сучасні мобільні застосунки для моніторингу спортивної активності надають користувачам можливість відстежувати результати тренувань, аналізувати фізичні показники та контролювати особистий прогрес. Найбільш поширеними рішеннями у цій сфері є Strava, Nike Run Club та Google Fit, які забезпечують широкий набір функціональних можливостей для аналізу фізичної активності [6–8].

Strava є одним із найпоширеніших застосунків для відстеження спортивної активності, орієнтованим насамперед на біг та велоспорт [6]. Застосунок дозволяє записувати маршрути із використанням GPS, зберігати історію тренувань, переглядати статистику активності, а також взаємодіяти із соціальними функціями, такими як рейтинги, досягнення та обмін результатами між користувачами.

Nike Run Club орієнтований переважно на бігові тренування та надає інструменти для контролю дистанції, темпу, швидкості й тривалості фізичної активності [7]. Особливістю застосунку є наявність тренувальних програм, аудіосупроводу під час занять та рекомендацій для покращення спортивних результатів.

Google Fit є універсальним рішенням для моніторингу фізичної активності, що дозволяє збирати та аналізувати інформацію про фізичні навантаження користувача [8]. Застосунок підтримує інтеграцію із різними сервісами та пристроями, забезпечує збір статистичних даних і дозволяє відстежувати загальну активність користувача.

Для більш детального порівняння функціональних можливостей сучасних застосунків доцільно виконати їх аналіз за основними критеріями, що представлено у таблиці 1.3.

Таблиця 1.3 - Порівняння сучасних застосунків для моніторингу спортивної активності.

Функція	**Strava**	**Nike Run Club**	**Google Fit**
GPS-маршрут	+	+	+
Історія тренувань	+	+	+
Аналіз статистики	+	+	+
Рейтинг користувачів	+	-	-
Профіль користувача	+	+	+
Авторизація	+	+	+
Багатомовність	+	+	+
Запис активності	+	+	+

Отже, проведений аналіз сучасних застосунків для моніторингу спортивної активності показав, що найбільш функціонально розвиненим рішенням є Strava, яка поєднує можливості GPS-відстеження маршрутів, збереження історії тренувань, аналізу статистичних показників, рейтингу користувачів та соціальної взаємодії [6]. Саме тому під час розробки мобільного застосунку основну увагу було зосереджено на реалізації ключових функціональних можливостей, характерних для Strava, зокрема запису спортивної активності, побудови маршруту, перегляду історії тренувань, системи рейтингу, профілю користувача та аналізу статистичних показників.

У той же час було враховано окремі особливості інших популярних застосунків, таких як Nike Run Club та Google Fit, що дозволило визначити найбільш доцільні підходи до реалізації інтерфейсу та функціональних можливостей системи.

1.3 Порівняння технологій реалізації мобільних застосунків.

Розробка сучасних мобільних застосунків для моніторингу спортивної активності передбачає використання програмних технологій, що забезпечують збір, обробку, збереження та візуалізацію інформації про фізичну активність користувача. До ключових вимог таких систем належать підтримка геолокації, взаємодія із серверною частиною, обробка даних у режимі реального часу, зручний інтерфейс користувача та можливість відображення статистичної інформації [1].

Для реалізації мобільних застосунків можуть використовуватися різні технології та фреймворки, серед яких найбільш поширеними є нативна розробка для Android та iOS, а також кросплатформені рішення. Нативна розробка передбачає використання окремих мов програмування та інструментів для кожної мобільної платформи, однак потребує більших витрат часу на створення та підтримку програмного продукту [1].

Одним із сучасних підходів до створення мобільних застосунків є використання кросплатформених технологій, які дозволяють реалізувати єдину програмну систему для декількох платформ. Серед таких технологій поширення набув Flutter – фреймворк для створення мобільних застосунків, що використовує мову програмування Dart та дозволяє реалізовувати сучасний інтерфейс користувача, підтримувати швидку взаємодію та забезпечувати повторне використання програмного коду [2].

Для застосунків спортивного моніторингу особливе значення має підтримка геолокації та картографічних сервісів. Використання GPS дозволяє здійснювати визначення маршруту пересування користувача, обчислення пройденої дистанції, швидкості руху та інших параметрів фізичної активності [4]. Крім цього, важливою складовою є організація взаємодії між мобільним застосунком та серверною частиною за допомогою REST API, що забезпечує передачу інформації, авторизацію користувача та збереження статистичних даних [3].

Порівняльну характеристику основних технологій реалізації мобільних застосунків наведено у таблиці 1.4.

Таблиця 1.4 – Порівняння технологій реалізації мобільних застосунків

Технологія	**Переваги**	**Недоліки**
Native Android (Java/Kotlin)	Висока продуктивність	Окрема розробка для Android
Native iOS (Swift)	Оптимізація під iOS	Окрема розробка для iOS
React Native	Кросплатформеність	Залежність від сторонніх бібліотек
Flutter	Єдина кодова база, висока швидкодія, сучасний UI	Потребує вивчення Dart

Отже, аналіз сучасних технологій реалізації мобільних застосунків показав, що використання кросплатформених рішень є доцільним для створення систем спортивного моніторингу. З огляду на необхідність реалізації сучасного інтерфейсу, підтримки геолокації, клієнт-серверної взаємодії та повторного використання коду, для реалізації застосунку було обрано Flutter і мову програмування Dart.

1.4 Постановка задачі дослідження.

Проведений аналіз предметної області, сучасних мобільних застосунків для моніторингу спортивної активності та технологій їх реалізації показав актуальність створення мобільних систем, що забезпечують зручний контроль фізичної активності користувача, аналіз статистичних показників та збереження історії тренувань.

На основі проведеного аналізу встановлено, що сучасні застосунки спортивного моніторингу повинні забезпечувати запис фізичної активності користувача, визначення маршруту пересування із використанням GPS, збереження результатів тренувань, відображення статистичних показників та підтримку взаємодії із серверною частиною. Також важливими є зручність інтерфейсу користувача, швидкість роботи системи та можливість централізованого зберігання даних.

Метою дослідження є розробка клієнт-серверного мобільного застосунку для моніторингу спортивних активностей, функціональні можливості якого базуються на принципах роботи сучасних застосунків спортивного моніторингу, зокрема Strava.

Для досягнення поставленої мети необхідно вирішити такі задачі:

– виконати аналіз сучасних застосунків для моніторингу спортивної активності;

– дослідити технології реалізації мобільних застосунків спортивного моніторингу;

– реалізувати систему реєстрації та авторизації користувачів;

– реалізувати механізм запису спортивної активності із використанням геолокації;

– забезпечити побудову маршруту пересування користувача на карті;

– реалізувати збереження історії тренувань та перегляд статистичних показників;

– реалізувати систему рейтингу користувачів та профіль користувача;

– забезпечити взаємодію мобільного застосунку із серверною частиною через REST API.

Для кращого розуміння послідовності виконання дослідження доцільно представити основні етапи реалізації мобільного застосунку для моніторингу спортивної активності, що наведені на рисунку 1.4.



Рисунок 1.4 – Основні етапи реалізації дослідження

Наведена схема відображає основні етапи реалізації дослідження, починаючи від аналізу предметної області та існуючих аналогів до розробки, реалізації функціональних можливостей і тестування мобільного застосунку.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА КЛІЄНТ-СЕРВЕРНОГО ЗАСТОСУНКУ.

2.1 Загальна архітектура системи.

Розроблений мобільний застосунок MiniStrava реалізовано на основі клієнт-серверної архітектури, що забезпечує взаємодію між мобільним клієнтом, серверною частиною та механізмами збереження інформації. Такий підхід дозволяє централізовано обробляти дані, зберігати результати спортивної активності користувачів та забезпечувати доступ до функціональних можливостей системи через REST API.

Клієнтська частина системи реалізована за допомогою фреймворку Flutter із використанням мови програмування Dart. Використання Flutter дозволило створити мобільний застосунок із сучасним інтерфейсом користувача, підтримкою геолокації та взаємодією із серверною частиною в режимі обміну даними через HTTP-запити. Серверна частина системи побудована на базі Laravel та забезпечує обробку запитів, авторизацію користувачів, збереження спортивної активності, формування статистичних показників і рейтингу користувачів.

У структурі клієнтської частини застосунку виділено декілька функціональних модулів, кожен із яких відповідає за окрему частину роботи системи. Основу структури становлять модулі взаємодії з API, моделі даних, екрани застосунку, маршрути навігації, компоненти інтерфейсу та стилізація застосунку.

Для взаємодії із серверною частиною використовується бібліотека Dio, що забезпечує виконання HTTP-запитів до REST API та обмін даними у форматі JSON. Керування станом застосунку реалізовано за допомогою Provider, а для роботи з геолокацією та відображення маршруту використовуються бібліотеки Geolocator і FlutterMap. Для кращого розуміння

структури клієнтської частини доцільно представити загальну організацію модулів застосунку, що наведена на рисунку 2.1.

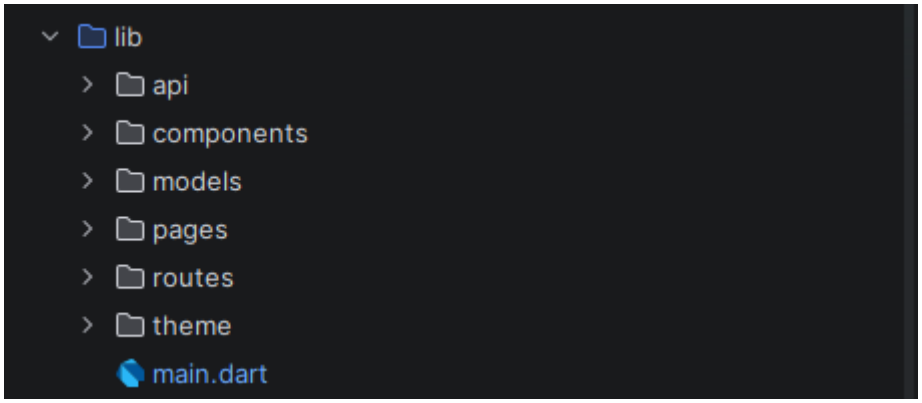


Рисунок 2.1 – Загальна структура клієнтської частини мобільного застосунку

Наведена структура демонструє модульний підхід до реалізації клієнтської частини застосунку. Модуль `api` відповідає за взаємодію із серверною частиною, `models` містить структури даних, `pages` забезпечує реалізацію екранів мобільного застосунку, `routes` реалізує навігацію між сторінками, `theme` відповідає за оформлення інтерфейсу, а `components` містить повторно використовувані елементи інтерфейсу.

Отже, використання клієнт-серверної архітектури забезпечує модульність, масштабованість та зручність подальшого розвитку системи, а також дозволяє ефективно реалізувати механізми моніторингу спортивної активності користувача.

2.2 Проєктування структури даних застосунку.

Важливим етапом розробки мобільного застосунку є проєктування структури даних, що забезпечує збереження інформації про користувачів, спортивні активності, статистичні показники та GPS-маршрути. Для реалізації системи MiniStrava використовується структура даних, побудована за принципом взаємопов’язаних сутностей, що забезпечує централізоване збереження інформації та швидкий доступ до неї.

Основу структури даних складають сутності користувача, спортивної активності, географічних координат маршруту, механізмів авторизації та статистичних показників. Такий підхід дозволяє організувати логічний зв’язок між тренуваннями користувача та даними про маршрут пересування, що є необхідним для побудови карти активності та аналізу результатів тренування.

На рисунку 2.2 представлено структуру бази даних мобільного застосунку MiniStrava.

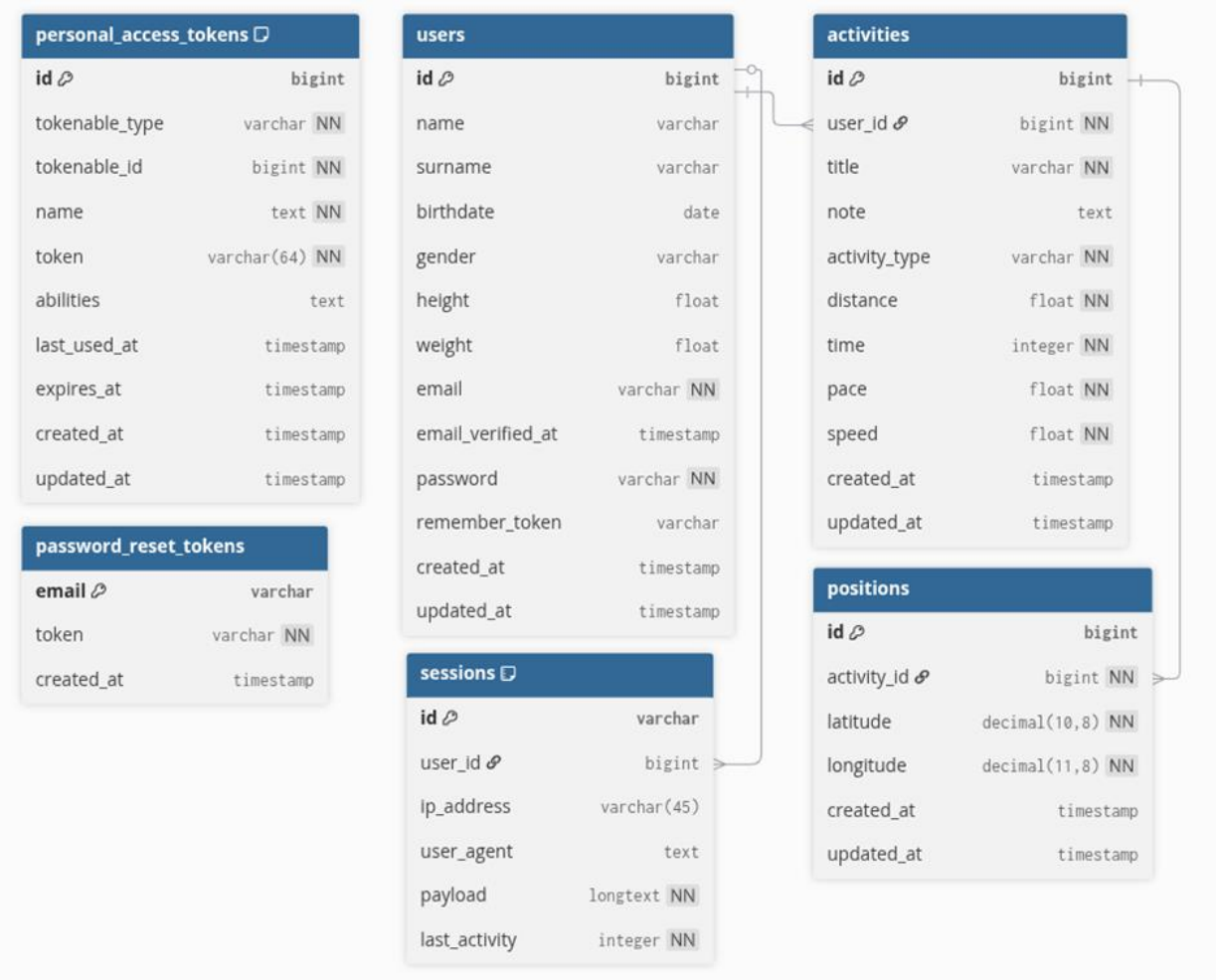


Рисунок 2.2 – Структура бази даних мобільного застосунку MiniStrava

Структура бази даних містить декілька взаємопов’язаних сутностей, які забезпечують збереження інформації про користувачів, спортивні активності, маршрути пересування та механізми авторизації. Основною сутністю є таблиця users, що містить персональні дані користувача, зокрема ім’я,

електронну пошту, дату народження, антропометричні показники та інформацію для авторизації.

Сутність activities використовується для збереження інформації про спортивні тренування користувача, включаючи тип активності, дистанцію, час виконання, швидкість та темп пересування. Для забезпечення можливості побудови маршруту використовується таблиця positions, яка містить GPS-координати та забезпечує зв’язок із конкретною активністю через ідентифікатор тренування.

Для реалізації механізмів авторизації та підтримки роботи користувачів у системі використовуються таблиці personal_access_tokens, password_reset_tokens та sessions, які забезпечують керування доступом, відновлення пароля та збереження інформації про активні сесії користувачів.

Для кращого розуміння структури даних основні сутності системи наведено у таблиці 2.1.

Таблиця 2.1 – Основні сутності системи

Сутність	**Призначення**
Users	Зберігання інформації про користувачів системи
Activities	Дані про спортивні тренування користувача
Positions	GPS-координати маршруту пересування
Sessions	Зберігання інформації про активні сесії
User Profile	Дані профілю користувача
User Stats	Статистичні показники користувача
Ranking Entry	Дані рейтингу користувачів

Наведена структура даних забезпечує ефективне збереження та обробку інформації про спортивну активність користувачів. Використання окремої сутності для координат маршруту дозволяє зберігати GPS-точки у вигляді послідовності географічних координат, що дає можливість відображати маршрут на карті та виконувати подальший аналіз фізичної активності користувача.

Крім серверної структури даних, у клієнтській частині застосунку використовуються моделі activity, ranking_entry, user_profile та user_stats, які забезпечують представлення отриманої інформації у мобільному застосунку та її подальшу обробку.

Отже, структура даних застосунку MiniStrava забезпечує зручне збереження, передачу та аналіз інформації про спортивні активності користувача, що є необхідною умовою реалізації функціональних можливостей системи.

2.3 Проєктування навігації та структури інтерфейсу.

Одним із важливих етапів розробки мобільного застосунку є проєктування навігації та структури інтерфейсу користувача, оскільки саме інтерфейс забезпечує взаємодію користувача із функціональними можливостями системи. У процесі розробки застосунку MiniStrava особливу увагу приділено простоті навігації, зручності взаємодії та швидкому доступу до основних функцій моніторингу спортивної активності.

Структура навігації застосунку побудована на основі розділення функціональних можливостей на окремі екрани, між якими реалізовано переходи за допомогою маршрутизації GoRouter. Основними елементами навігації є нижнє меню, сторінки авторизації та екрани роботи із тренуваннями, статистикою й профілем користувача.

Такий підхід забезпечує логічне розділення функціональних можливостей системи та спрощує взаємодію користувача із застосунком. Кожен екран відповідає за виконання окремого набору завдань, що позитивно

впливає на зручність використання та подальшу підтримку програмного забезпечення.

Для кращого розуміння структури переходів між основними сторінками застосунку доцільно представити схему навігації, що наведена на рисунку 2.3.



Рисунок 2.3 – Загальна схема навігації мобільного застосунку MiniStrava

Наведена схема демонструє загальну структуру переходів між екранами мобільного застосунку. Основним елементом взаємодії є нижня панель навігації, що забезпечує швидкий доступ до функцій запису активності, перегляду історії тренувань, рейтингу користувачів та профілю.

Початковим етапом взаємодії користувача із системою є стартовий екран, що забезпечує перехід до реєстрації нового облікового запису або авторизації користувача у системі. Після успішного входу користувач отримує доступ до основних функціональних можливостей застосунку та може здійснювати навігацію між окремими розділами без повторної авторизації.

Використання централізованої системи маршрутизації дозволяє забезпечити логічну послідовність переходів між екранами та спрощує керування навігаційною структурою застосунку. Такий підхід відповідає сучасним рекомендаціям щодо проєктування мобільних інтерфейсів, оскільки сприяє підвищенню зручності використання, скороченню кількості дій для доступу до функціональних можливостей системи та покращенню загального користувацького досвіду.

Крім того, розподіл функціональних можливостей між окремими екранами забезпечує кращу структурованість застосунку та спрощує його

подальший розвиток. Кожен екран відповідає за виконання конкретного набору завдань, що дозволяє зменшити складність інтерфейсу та зробити взаємодію із системою більш зрозумілою для користувача.

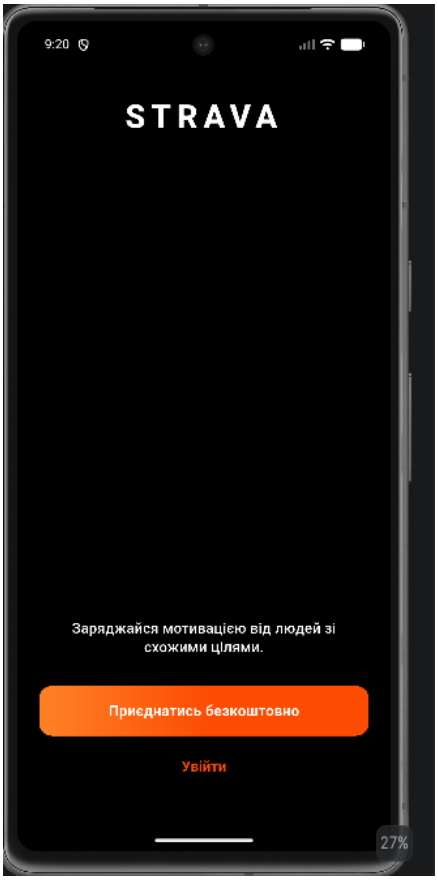


Рисунок 2.4 – Стартовий екран мобільного застосунку

Стартовий екран реалізовано у мінімалістичному стилі із використанням темної кольорової схеми та акцентного помаранчевого кольору. Інтерфейс забезпечує швидкий доступ до функцій створення нового облікового запису та авторизації користувача.

Під час проєктування стартового екрана основну увагу приділено простоті використання та зручності взаємодії користувача із системою. На екрані відображаються лише основні елементи керування, необхідні для початку роботи із застосунком, що дозволяє зменшити інформаційне навантаження на користувача.

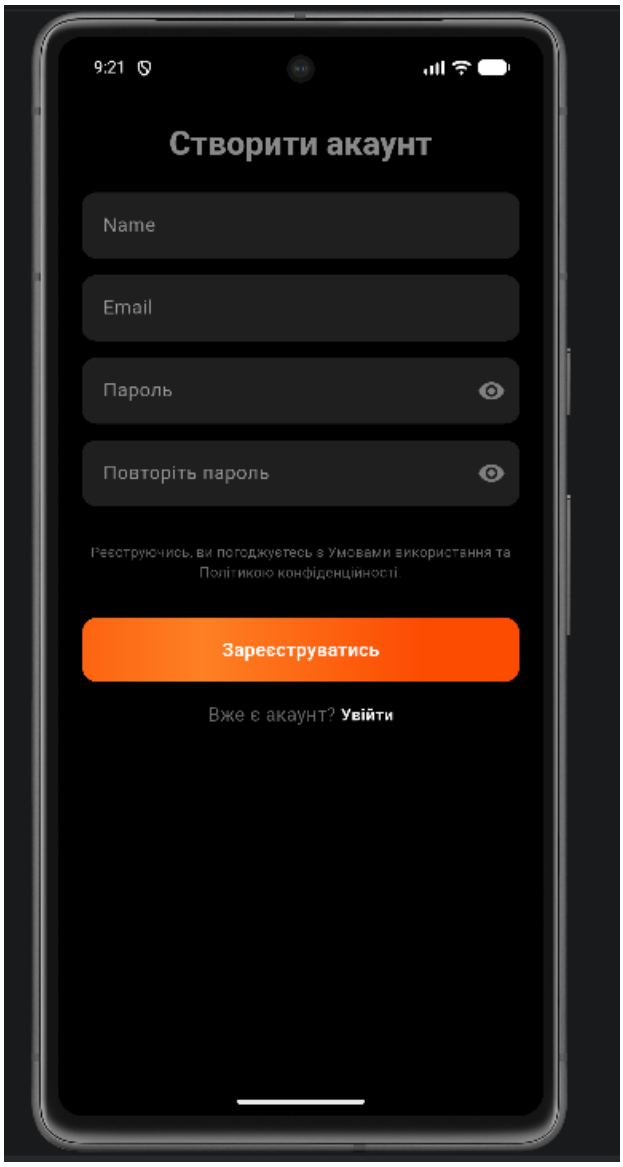


Рисунок 2.5 – Екран реєстрації користувача.

Екран реєстрації забезпечує можливість створення нового облікового запису шляхом введення основної інформації користувача, зокрема імені, електронної пошти та пароля. Для підвищення зручності взаємодії реалізовано механізм підтвердження пароля та перехід до сторінки авторизації.

Під час проєктування екрана реєстрації особливу увагу приділено простоті та зрозумілості процесу створення облікового запису. Поля введення розташовані у логічній послідовності, що дозволяє користувачу швидко заповнити необхідні дані та завершити процедуру реєстрації. Для зменшення

кількості помилок передбачено окреме поле підтвердження пароля, яке забезпечує додаткову перевірку правильності введених даних.

Використання єдиного стилю оформлення, темної кольорової схеми та акцентних елементів інтерфейсу забезпечує візуальну цілісність застосунку та покращує сприйняття інформації. Після успішного завершення реєстрації користувач отримує можливість авторизуватися у системі та перейти до використання функцій моніторингу спортивної активності.

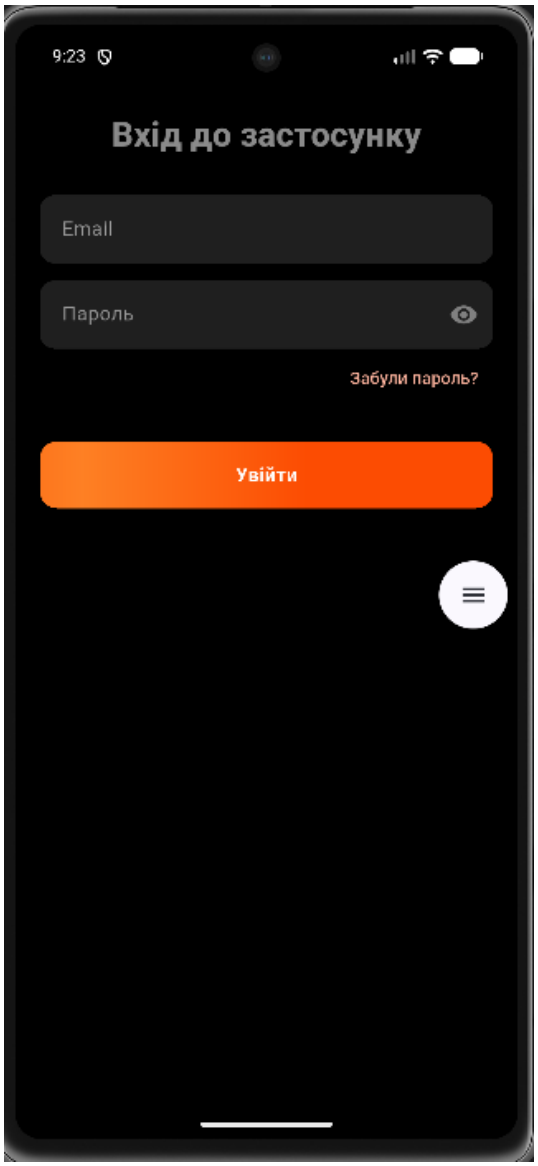


Рисунок 2.6 – Екран авторизації користувача

Екран авторизації призначений для входу користувача до системи із використанням електронної пошти та пароля. Крім основної функції входу, реалізовано механізм відображення введеного пароля та можливість відновлення доступу до облікового запису.



Рисунок 2.7 – Екран запису спортивної активності

Екран запису спортивної активності є одним із ключових елементів застосунку та забезпечує збір інформації про фізичну активність користувача. Інтерфейс містить карту маршруту, інформаційний блок статистичних

показників, зокрема тривалості, дистанції, темпу та швидкості, а також елементи керування початком і завершенням тренування.

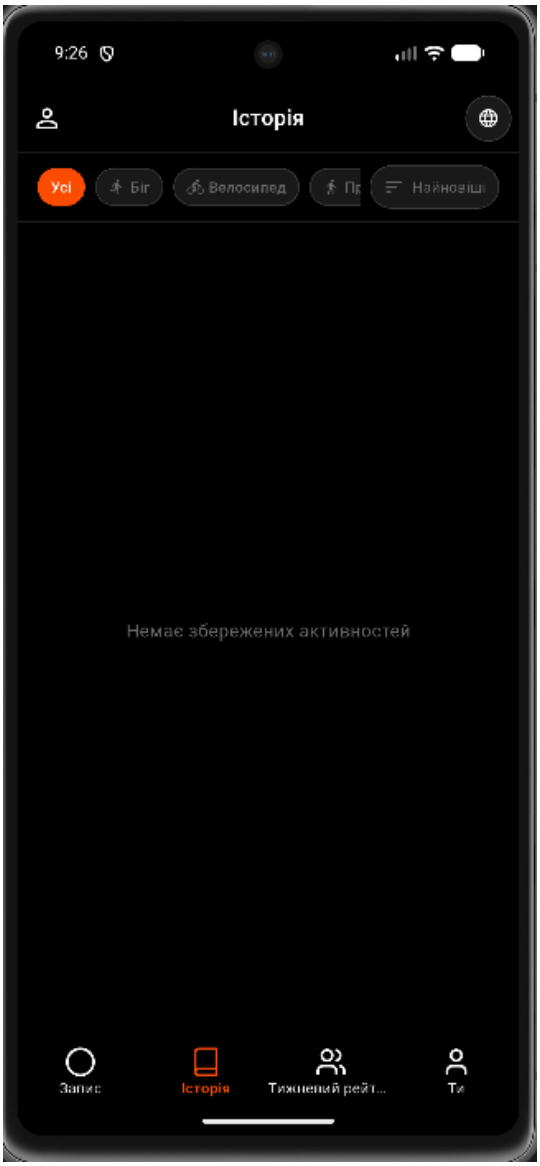


Рисунок 2.8 – Екран історії спортивних активностей

Екран історії тренувань забезпечує можливість перегляду результатів попередніх активностей користувача. Для спрощення навігації реалізовано механізми сортування та фільтрації тренувань за типом фізичної активності.

На екрані відображається список збережених тренувань із можливістю швидкого доступу до детальної інформації про кожну активність. Для підвищення зручності використання передбачено фільтрацію за видами спортивної діяльності, зокрема бігом, ходьбою та велоспортом. Це дозволяє

користувачу швидко знаходити необхідні записи серед накопиченої історії тренувань.

Функція збереження історії активностей є важливою складовою систем спортивного моніторингу, оскільки надає можливість аналізувати результати попередніх тренувань, відстежувати прогрес та оцінювати зміни фізичних показників упродовж тривалого часу. Використання механізмів сортування та фільтрації сприяє підвищенню ефективності роботи з даними та покращує загальний користувацький досвід.



Рисунок 2.9 – Екран тижневого рейтингу користувачів

Екран рейтингу використовується для відображення статистичних показників фізичної активності користувачів у межах визначеного проміжку часу. Реалізований механізм дозволяє виконувати порівняння власної активності та переглядати персональні результати за останній тиждень.

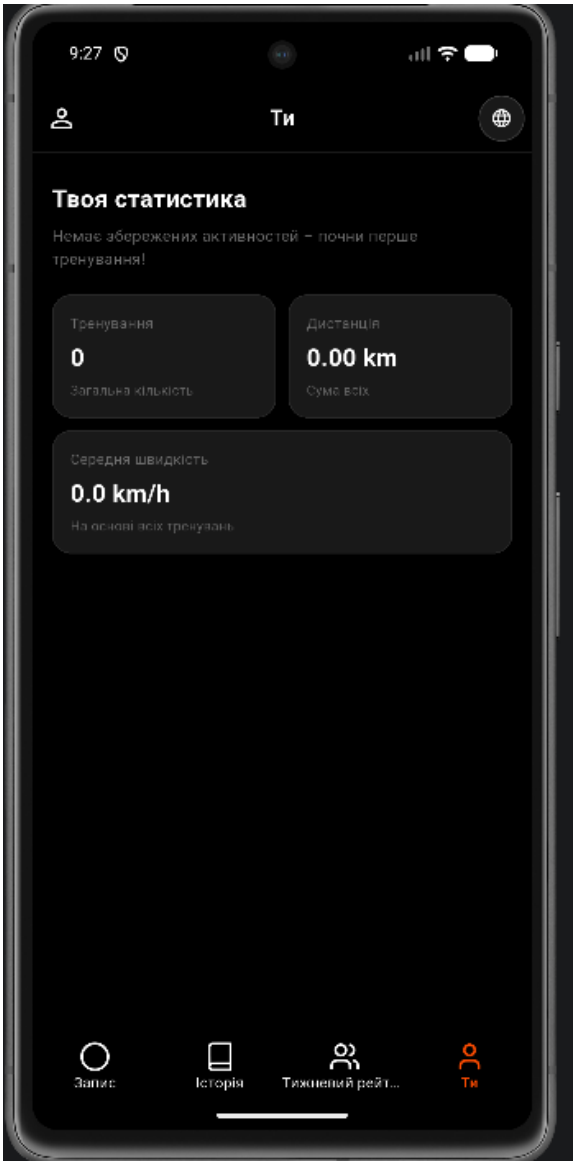


Рисунок 2.10 – Екран статистики користувача

Екран статистики користувача забезпечує перегляд узагальненої інформації про результати тренувань, включаючи кількість активностей, сумарну дистанцію та середню швидкість. Використання інформаційних блоків дозволяє покращити сприйняття статистичних показників.

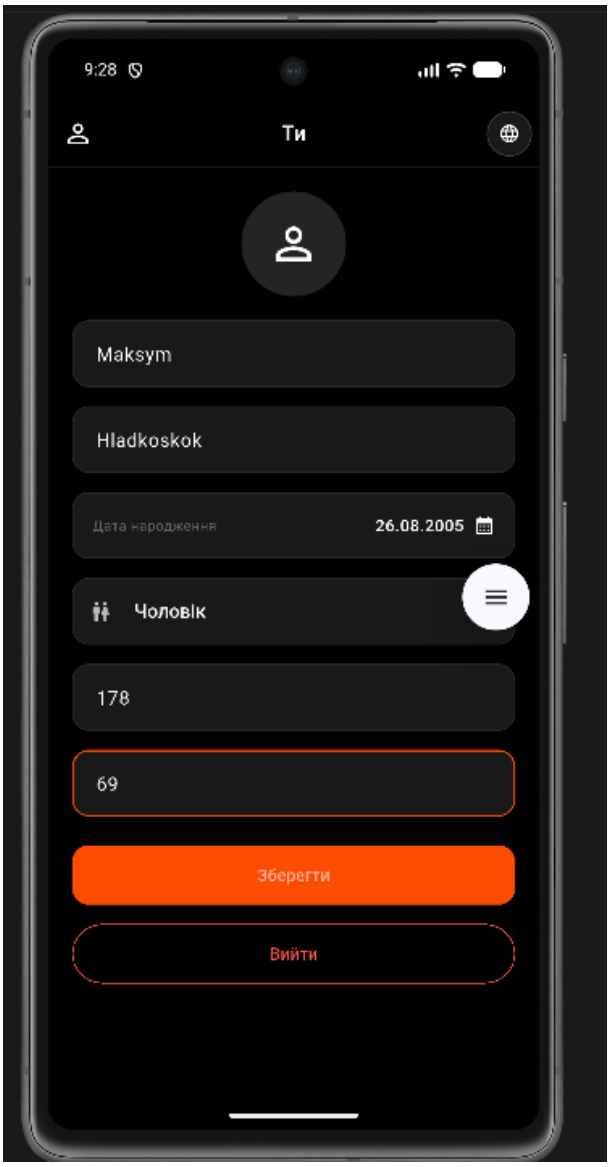


Рисунок 2.11 – Екран редагування профілю користувача

Екран профілю користувача забезпечує можливість редагування персональних даних, зокрема імені, дати народження, статі, антропометричних показників та інших параметрів, що використовуються для персоналізації роботи застосунку та формування статистичних даних.

Отже, структура інтерфейсу та навігації мобільного застосунку MiniStrava забезпечує зручний доступ до основних функціональних можливостей системи, спрощує взаємодію користувача із застосунком та дозволяє ефективно організувати процес моніторингу спортивної активності.

2.4 Організація взаємодії з REST API.

Для забезпечення взаємодії між клієнтською та серверною частинами мобільного застосунку MiniStrava використовується архітектурний підхід REST API, що забезпечує передачу інформації між компонентами системи за допомогою HTTP-запитів. Використання REST API дозволяє реалізувати централізоване збереження даних, механізми авторизації користувачів, отримання статистичної інформації, формування рейтингу та роботу із тренуваннями.

У клієнтській частині застосунку реалізація взаємодії із сервером виконується за допомогою бібліотеки `Dio`, яка використовується для надсилання HTTP-запитів та отримання відповідей від серверної частини. Запити між компонентами системи здійснюються у форматі JSON, що забезпечує структурований обмін даними та спрощує процес їх подальшої обробки.

На серверній стороні REST API реалізовано із використанням Laravel та забезпечено набір маршрутів для авторизації, отримання інформації про користувача, збереження спортивних активностей, формування рейтингу та роботи зі статистичними показниками. Взаємодія між мобільним застосунком і серверною частиною наведена на рисунку 2.12.

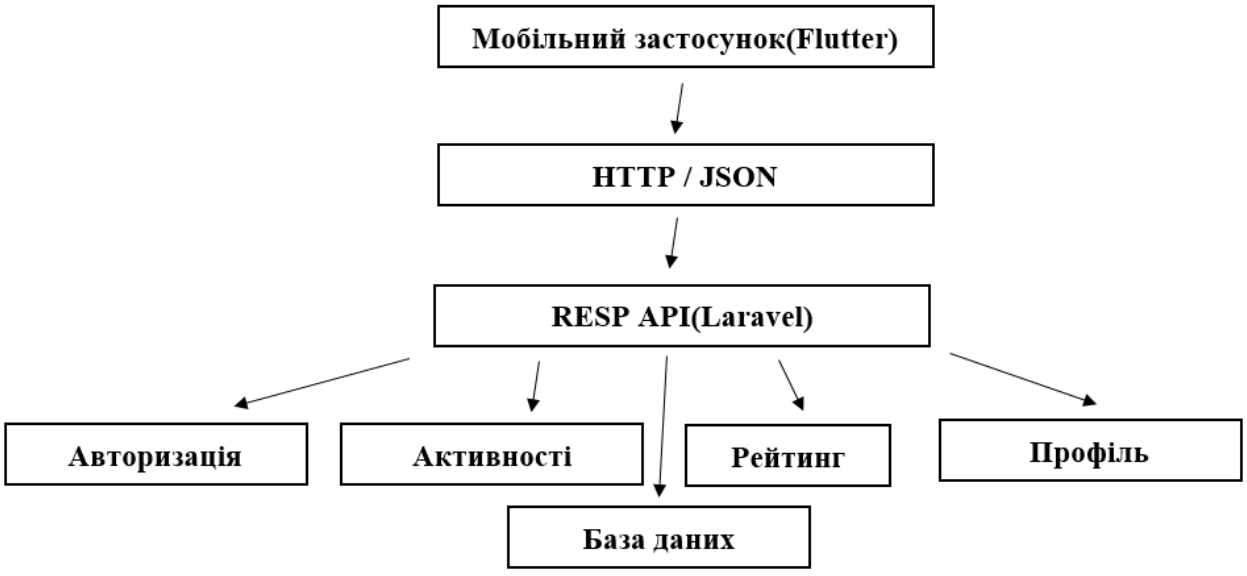


Рисунок 2.12 – Організація взаємодії мобільного застосунку із REST API

Наведена схема демонструє механізм взаємодії між мобільним застосунком та серверною частиною. Клієнтська частина формує HTTP-запити до REST API, після чого сервер виконує обробку інформації, взаємодіє з базою даних та повертає відповідь у форматі JSON.

Таблиця 2.2 - Основні запити REST API мобільного застосунку

Маршрут API	**Метод**	**Призначення**
/auth/login	POST	Авторизація користувача
/auth/register	POST	Реєстрація нового користувача
/user	GET	Отримання інформації про профіль
/activities	GET	Отримання списку тренувань
/activities	POST	Збереження спортивної активності
/ranking	GET	Отримання рейтингу користувачів

Для організації роботи із REST API у структурі застосунку використовується окремий модуль `api`, що містить компоненти взаємодії із серверною частиною. Зокрема, `activity_repository` забезпечує роботу із даними спортивних активностей, `ranking_repository` використовується для формування рейтингу, `user_repository` забезпечує отримання інформації про користувача, а `AuthProvider` реалізує механізми авторизації та керування станом облікового запису.

Отже, використання REST API забезпечує стабільну взаємодію між мобільним застосунком та серверною частиною, централізоване збереження

інформації та можливість масштабування системи шляхом розширення функціональних можливостей серверної частини.

2.5 Організація керування станом застосунку

Під час розробки мобільного застосунку MiniStrava важливим завданням є організація ефективного керування станом застосунку, оскільки система передбачає роботу з даними користувача, авторизацією, статистичними показниками, рейтингом та спортивними активностями. Для забезпечення централізованого керування даними у застосунку використано підхід **Provider**, що дозволяє організувати взаємодію між інтерфейсом користувача та джерелами даних.

Використання **Provider** забезпечує можливість оновлення інтерфейсу в реальному часі після зміни даних користувача або отримання нової інформації із серверної частини. Такий підхід дозволяє уникнути дублювання логіки, спрощує структуру програмного коду та забезпечує централізоване керування станом системи.

У структурі застосунку реалізовано декілька компонентів керування станом, що відповідають за окремі функціональні частини системи. Зокрема, **AuthProvider** використовується для реалізації авторизації користувача та керування інформацією про активну сесію, **user_provider** забезпечує отримання та оновлення інформації профілю користувача, а **ranking_provider** відповідає за формування рейтингової інформації та її відображення в інтерфейсі застосунку.

Для кращого розуміння процесу керування станом застосунку доцільно представити загальну схему взаємодії між інтерфейсом користувача, компонентами керування станом та REST API, що наведена на рисунку 2.13.

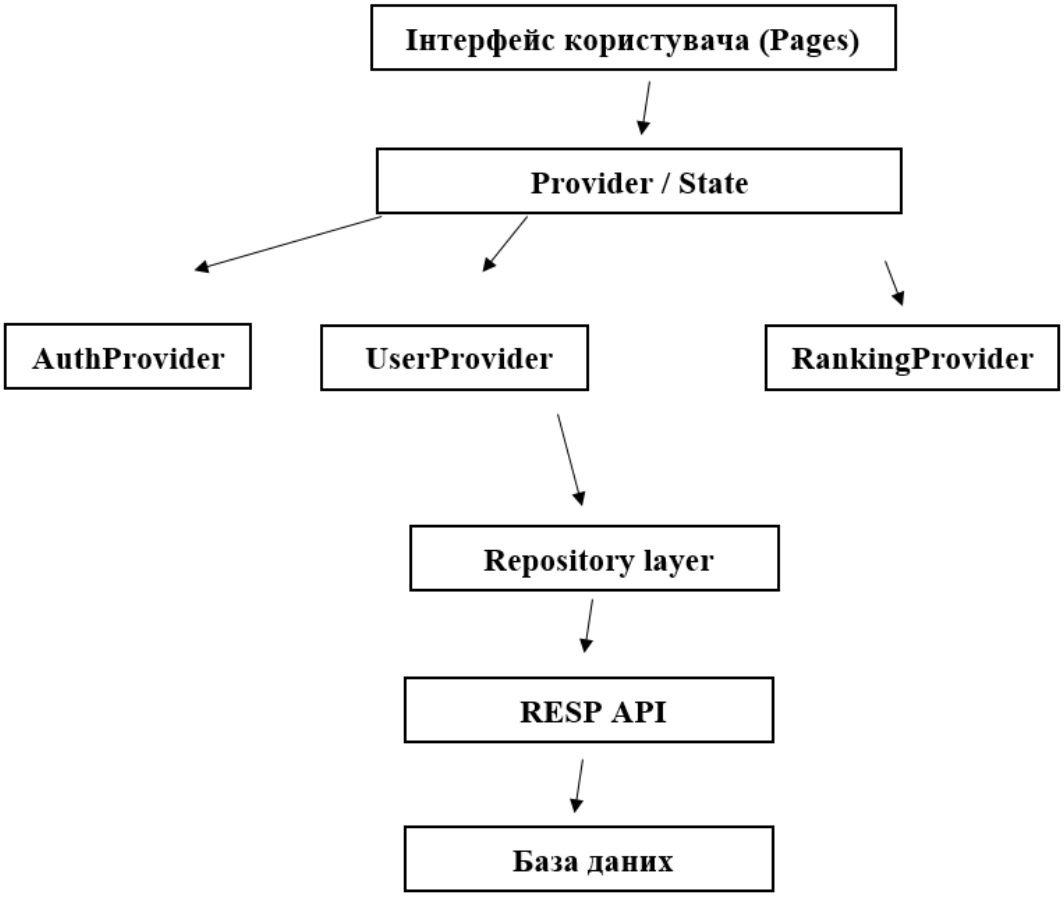


Рисунок 2.13 – Організація керування станом мобільного застосунку

Наведена схема демонструє механізм організації керування станом у мобільному застосунку. Інтерфейс користувача взаємодіє із компонентами Provider, які відповідають за отримання, оновлення та збереження даних. У разі необхідності отримання інформації провайдери звертаються до компонентів взаємодії із REST API, що забезпечують надсилання HTTP-запитів та отримання відповідей від серверної частини.

Використання такого підходу дозволяє відокремити логіку обробки даних від компонентів користувацького інтерфейсу, що спрощує підтримку та подальший розвиток програмного забезпечення.

Для кращого представлення функціонального призначення компонентів керування станом у таблиці 2.3 наведено їх основні функції.

Таблиця 2.3 - Основні компоненти керування станом застосунку

Компонент	**Призначення**
AuthProvider	Авторизація користувача та керування сесією
user_provider	Отримання та оновлення інформації профілю користувача
ranking_provider	Формування та оновлення рейтингової інформації
activity_repository	Отримання та збереження спортивних активностей
user_repository	Взаємодія із даними користувача
ranking_repository	Отримання інформації про рейтинг

Використання механізму **Provider** дозволяє забезпечити логічне розділення компонентів інтерфейсу та бізнес-логіки застосунку, спрощує підтримку програмного коду та забезпечує швидке оновлення даних у мобільному інтерфейсі після їх зміни.

Отже, реалізація керування станом на основі **Provider** забезпечує ефективну організацію взаємодії між компонентами системи, підвищує гнучкість програмної архітектури та спрощує масштабування функціональних можливостей мобільного застосунку MiniStrava.

2.6 Обґрунтування вибору технологій та програмних засобів

Під час розробки мобільного застосунку MiniStrava важливим етапом є вибір технологій та програмних засобів, що забезпечують реалізацію функціональних можливостей системи, стабільність роботи застосунку, зручність подальшої підтримки та можливість масштабування програмного продуґкту.

Для реалізації клієнтської частини застосунку було обрано фреймворк **Flutter**, що дозволяє створювати сучасні мобільні застосунки із

використанням єдиної кодової бази. Використання **Flutter** забезпечує високу швидкодію інтерфейсу, підтримку гнучкого проєктування елементів взаємодії та зручну інтеграцію із зовнішніми бібліотеками.

Мовою програмування для реалізації застосунку обрано **Dart**, що є основною мовою розробки для **Flutter** та забезпечує підтримку об'єктно-орієнтованого програмування, зручну роботу із асинхронними процесами та взаємодію із мережевими сервісами.

У якості середовища розробки використано **Android Studio**, що забезпечує зручні інструменти для створення, тестування, налагодження та запуску мобільного застосунку. Середовище розробки також дозволяє виконувати емуляцію роботи застосунку та спрощує процес керування структурою проєкту.

Для організації взаємодії із серверною частиною застосунку використовується бібліотека **Dio**, що забезпечує виконання HTTP-запитів до REST API та підтримує роботу із відповідями сервера у форматі JSON. Реалізація механізму навігації між сторінками застосунку виконується за допомогою **GoRouter**, що дозволяє організувати структуру переходів між екранами та забезпечити централізоване керування маршрутами.

Отримання географічних координат користувача реалізовано із використанням бібліотеки **Geolocator**, що забезпечує доступ до GPS-модуля мобільного пристрою та дозволяє здійснювати моніторинг переміщення користувача під час тренувань. Для відображення карти та маршруту спортивної активності використовується **FlutterMap** у поєднанні із картографічними сервісами **OpenStreetMap**.

Для керування станом застосунку використовується Provider, що дозволяє забезпечити централізоване оновлення даних інтерфейсу та організувати взаємодію між компонентами системи.

Для кращого представлення використаних технологій у таблиці 2.4 наведено їх основне призначення.

Таблиця 2.4 – Використані технології та програмні засоби

Технологія	**Призначення**
Flutter	Розробка мобільного застосунку
Dart	Мова програмування
Android Studio	Середовище розробки
Dio	Взаємодія із REST API
Provider	Керування станом застосунку
GoRouter	Організація навігації
Geolocator	Отримання GPS-координат
FlutterMap	Відображення карти маршруту
OpenStreetMap	Картографічні дані
Flutter Localizations	Підтримка локалізації інтерфейсу

Наведені технології забезпечують реалізацію основних функціональних можливостей мобільного застосунку, зокрема авторизації користувачів, GPS-моніторингу спортивної активності, роботи із картографічними сервісами, взаємодії із серверною частиною та керування інтерфейсом користувача.

Отже, вибір Flutter, Dart, REST API та допоміжних бібліотек є доцільним для реалізації мобільного застосунку MiniStrava, оскільки забезпечує гнучкість розробки, підтримку сучасного інтерфейсу, можливість масштабування та стабільну взаємодію між компонентами системи. Крім того, використані технології надають широкий набір готових інструментів для реалізації геолокації, мережевої взаємодії, керування станом та побудови користувацького інтерфейсу. Це дозволяє скоротити час розробки, підвищити якість програмного продукту та забезпечити можливість його подальшого розширення новими функціональними можливостями.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Реалізація функціональних можливостей застосунку.

У процесі розробки мобільного застосунку MiniStrava було реалізовано набір функціональних можливостей, що забезпечують авторизацію користувача, GPS-моніторинг спортивної активності, збереження результатів тренувань, формування рейтингу та персоналізацію профілю. Реалізація системи здійснювалася із використанням клієнт-серверного підходу, що забезпечує взаємодію між мобільним застосунком та серверною частиною через REST API.

Однією із базових функціональних можливостей застосунку є механізм реєстрації та авторизації користувача. Для реалізації цього механізму використовується компонент AuthProvider, що забезпечує взаємодію із серверною частиною, керування станом авторизації та оновлення інтерфейсу після успішного входу користувача до системи.

Після успішної авторизації користувач отримує доступ до всіх функціональних можливостей застосунку. Дані облікового запису використовуються для отримання персональної статистики, збереження спортивних активностей та формування рейтингу користувачів. Реалізований механізм авторизації також забезпечує можливість завершення сесії користувача та повторного входу до системи без втрати збережених даних.

Крім механізму авторизації, у застосунку реалізовано функції запису тренувань із використанням GPS-координат, автоматичного розрахунку дистанції, швидкості та темпу руху. Отримані результати можуть бути збережені на сервері та надалі відображаються у розділі історії тренувань, статистики користувача та системі рейтингу.

Для кращого представлення процесу авторизації доцільно використати схему, наведену на рисунку 3.1.

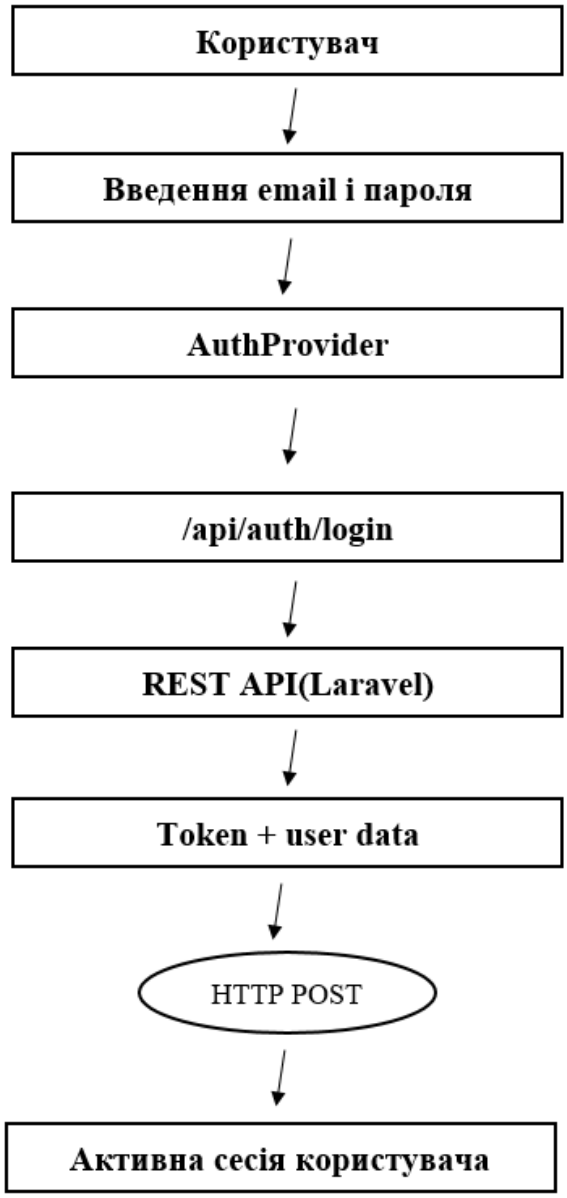


Рисунок 3.1 – Схема процесу авторизації користувача

Процес авторизації реалізовано через надсилання POST-запиту до маршруту /api/auth/login із використанням бібліотеки Dio. Після перевірки введених облікових даних серверна частина повертає токен доступу та інформацію про користувача, що використовується для формування активної сесії та подальшої взаємодії із захищеними маршрутами REST API.

Для прикладу реалізації HTTP-запиту авторизації у лістингу 3.1 наведено фрагмент програмного коду.

```
final response = await api.dio.post(
  '/api/auth/login',
  data: {
    'email': email,
    'password': password,
  },
  options: Options(
    headers: {
      'Accept': 'application/json',
    },
  ),
);
```

Лістинг 3.1 – Реалізація HTTP-запиту авторизації користувача

Однією із ключових функціональних можливостей застосунку є реалізація GPS-моніторингу спортивної активності. Для реалізації цієї функції використовується компонент RecordController, побудований на основі ChangeNotifier, що забезпечує централізоване керування процесом тренування та автоматичне оновлення статистичних показників.

Компонент відповідає за запуск і завершення тренування, обробку геолокаційних даних, підрахунок пройденої дистанції та формування маршруту пересування користувача. У процесі роботи RecordController взаємодіє із сервісами геолокації мобільного пристрою та забезпечує оновлення даних у режимі реального часу.

Завдяки використанню механізму керування станом усі зміни статистичних показників автоматично відображаються в інтерфейсі користувача без необхідності ручного оновлення екрана. Це забезпечує зручне відстеження основних параметрів тренування під час виконання спортивної активності.

Для отримання географічних координат використовується бібліотека Geolocator, що забезпечує доступ до GPS-модуля мобільного пристрою. Перед початком тренування система виконує перевірку доступності служб геолокації та запитує необхідні дозволи на доступ до місцеположення користувача.

Після отримання необхідних дозволів застосунок розпочинає збір координат у режимі реального часу. Отримані дані використовуються для побудови маршруту користувача, визначення пройденої дистанції та розрахунку основних статистичних показників тренування.

Організацію GPS-моніторингу тренування наведено на рисунку 3.2.

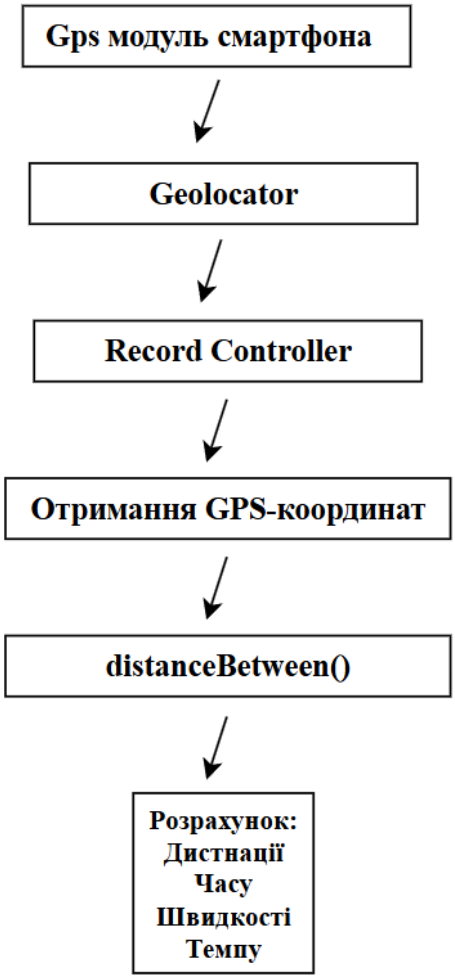


Рисунок 3.2 – Організація GPS-моніторингу тренування

У процесі тренування застосунок здійснює безперервне отримання GPS-координат через потік геолокаційних даних (getPositionStream). Для підвищення точності використовується високий рівень визначення координат (LocationAccuracy.best), а оновлення місцеположення виконується через кожні три метри переміщення користувача.

На основі отриманих координат система виконує формування маршруту пересування користувача та автоматичний розрахунок статистичних показників, зокрема тривалості тренування, дистанції, середньої швидкості та темпу. Отримані координати послідовно зберігаються у вигляді набору точок маршруту, що дозволяє відобразити фактичний шлях переміщення користувача на карті після завершення тренування.

Крім побудови маршруту, система виконує розрахунок відстані між сусідніми GPS-точками та накопичує загальну пройдену дистанцію. Розраховані показники використовуються для формування статистики тренування та подальшого збереження результатів на сервері.

У лістингу 3.2 наведено приклад отримання GPS-координат користувача.

```
_posSub = Geolocator.getPositionStream(
  locationSettings: const LocationSettings(
    accuracy: LocationAccuracy.best,
    distanceFilter: 3,
  ),
```

Лістинг 3.2 – Отримання GPS-координат користувача

Для збереження результатів тренувань використовується компонент `ActivityRepository`, що забезпечує взаємодію із серверною частиною через REST API. Після завершення тренування застосунок формує структуру активності, яка містить тип тренування, дистанцію, тривалість, швидкість, темп та GPS-маршрут користувача.

Процес збереження результатів тренування наведено на рисунку 3.4.

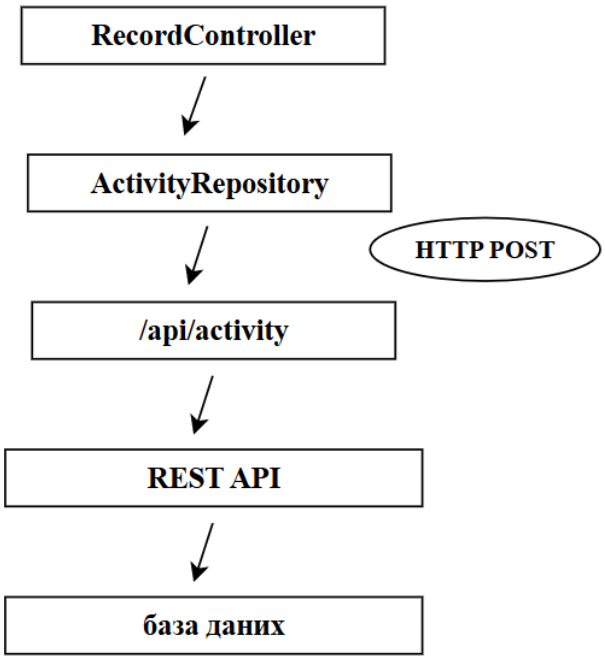


Рисунок 3.4 – Процес збереження спортивної активності

Перелік основних параметрів, що передаються під час збереження спортивної активності на сервері, наведено в таблиці 3.1.

Таблиця 3.1 – Поля даних спортивної активності

Дані	**Призначення**
distance	Пройдена дистанція
time	Тривалість тренування
speed	Середня швидкість
pace	Темп руху
activity_type	Тип активності
track	GPS-маршрут користувача

Реалізований механізм також забезпечує перегляд історії тренувань, отримання детальної інформації про окремі активності та видалення тренувань через відповідні маршрути REST API.

Для підвищення мотивації користувачів у застосунку реалізовано механізм тижневого рейтингу, що побудований із використанням компонента RankingProvider. Отримання рейтингових даних здійснюється через RankingRepository, а оновлення інтерфейсу реалізується із використанням механізму notifyListeners().

Крім цього, у застосунку реалізовано механізм керування профілем користувача через компонент UserProvider, що забезпечує отримання, оновлення та збереження персональних даних користувача, включаючи ім’я, прізвище, дату народження, стать, зріст і масу тіла.

Таким чином, у мобільному застосунку MiniStrava реалізовано комплекс функціональних можливостей, що забезпечують авторизацію користувача, GPS-моніторинг спортивної активності, запис і збереження тренувань, формування рейтингу, перегляд статистичних показників та керування профілем користувача.

3.2 Тестування мобільного застосунку.

Після завершення реалізації основних функціональних можливостей мобільного застосунку MiniStrava було виконано тестування коректності роботи системи. Основною метою тестування була перевірка працездатності механізмів авторизації, взаємодії із серверною частиною, GPS-моніторингу, обробки статистичних даних та керування профілем користувача.

Під час тестування особливу увагу приділено перевірці взаємодії мобільного застосунку із REST API серверної частини. У процесі виконання тестів підтверджено коректне виконання HTTP-запитів авторизації, завершення активної сесії користувача, оновлення профілю та отримання статистичних показників. Результати взаємодії із серверною частиною наведено на рисунку 3.5.

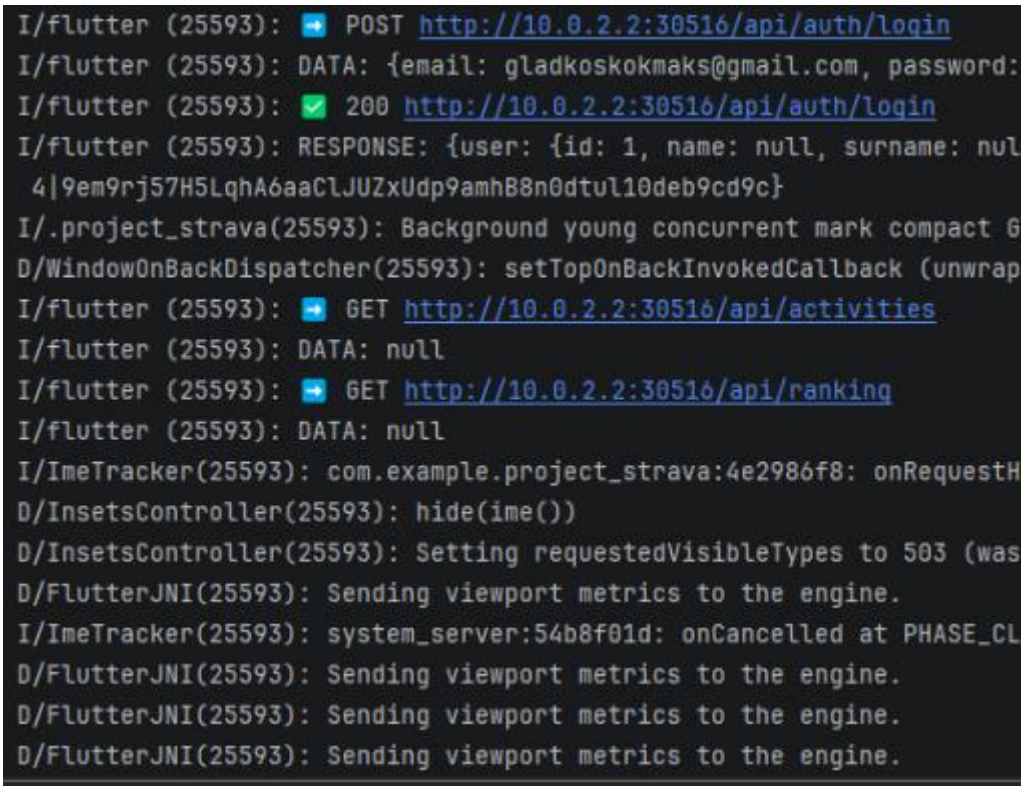


Рисунок 3.5 – Результат тестування взаємодії мобільного застосунку із REST API.

У процесі тестування підтверджено успішне виконання HTTP-запитів типу POST, PUT та GET, а також отримання відповіді сервера із кодом 200 OK, що підтверджує коректність обміну даними між клієнтською та серверною частинами застосунку.

Окремо було виконано тестування механізму GPS-моніторингу спортивної активності. Під час запуску тренування перевірено ініціалізацію компонента RecordController, отримання географічних координат користувача та роботу механізму перевірки доступності служб геолокації. Результати тестування наведено на рисунку 3.6 та 3.7.

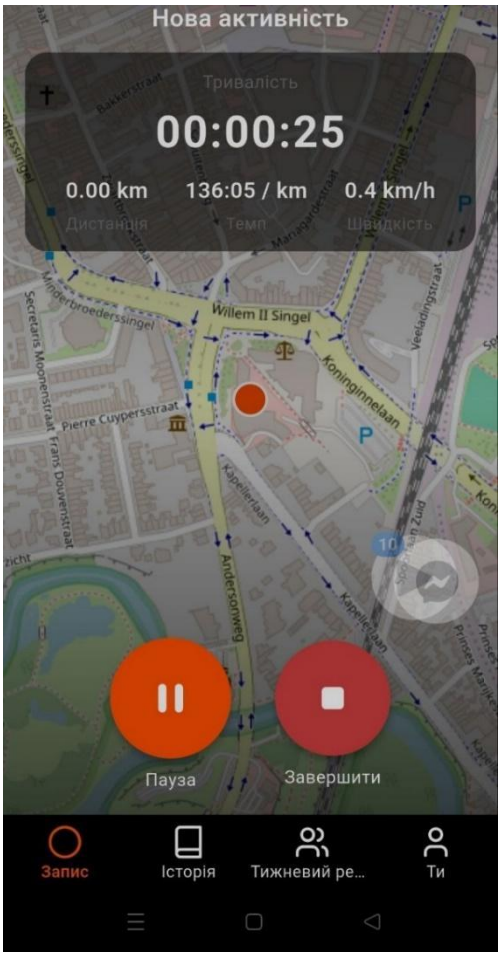


Рисунок 3.6 – Результат тестування GPS-моніторингу спортивної активності

```
D/WindowOnBackDispatcher(25593): setTopOnBackInvokedCallback (unwrapped): android.app.
I/flutter (25593): ✅ 200 http://10.0.2.2:30516/api/ranking
I/flutter (25593): RESPONSE: []
I/flutter (25593): ✅ 200 http://10.0.2.2:30516/api/activities
I/flutter (25593): RESPONSE: []
I/flutter (25593): ➡️ GET http://10.0.2.2:30516/api/user/stats
I/flutter (25593): DATA: null
I/flutter (25593): ✅ 200 http://10.0.2.2:30516/api/user/stats
I/flutter (25593): RESPONSE: {activities_number: 0, distance_total: 0, speed_avg: 0}
I/flutter (25593): ➡️ GET http://10.0.2.2:30516/api/user
I/flutter (25593): DATA: null
D/WindowOnBackDispatcher(25593): setTopOnBackInvokedCallback (unwrapped): io.flutter.e
I/ImeTracker(25593): com.example.project_strava:ee7d9fc7: onRequestShow at ORIGIN_CLIE
D/InsetsController(25593): show(ime())
```

Рисунок 3.7 – Результат тестування GPS-моніторингу спортивної активності

Під час тестування також перевірено сценарій відсутності доступу до служб геолокації, у якому система коректно виконує обробку помилки та

повідомляє користувача про недоступність GPS-модуля, що підтверджує стабільність роботи застосунку у різних умовах використання.

Результати тестування основних функціональних можливостей застосунку наведено у таблиці 3.2.

Таблиця 3.2 – Результати тестування функціональних можливостей застосунку

№	Компонент	Перевірка	Результат
1	Авторизація	Вхід у систему	Успішно
2	REST API	Обмін даними із сервером	Успішно
3	Історія тренувань	Отримання списку активностей	Успішно
4	Рейтинг	Завантаження рейтингових даних	Успішно
5	Статистика	Отримання статистики користувача	Успішно
6	GPS-моніторинг	Отримання координат та запис маршруту	Успішно
7	Профіль	Оновлення персональних даних	Успішно

Таким чином, проведене тестування підтвердило коректність роботи основних функціональних можливостей мобільного застосунку MiniStrava, стабільність взаємодії із серверною частиною та працездатність механізмів моніторингу спортивної активності.

3.3 Аналіз результатів роботи системи.

У результаті виконання кваліфікаційної роботи було розроблено мобільний застосунок MiniStrava для моніторингу спортивної активності користувачів. Реалізований застосунок забезпечує запис тренувань, GPS-

моніторинг маршруту, збереження статистичних показників, формування рейтингу користувачів та керування персональним профілем.

У процесі розробки реалізовано клієнт-серверну архітектуру, що забезпечує взаємодію між мобільним застосунком та серверною частиною через REST API. Використання бібліотеки Dio дозволило організувати стабільний обмін інформацією між компонентами системи, а механізм керування станом на основі Provider забезпечив коректне оновлення інтерфейсу та синхронізацію даних користувача.

Під час реалізації механізму спортивного моніторингу використано бібліотеку Geolocator, що дозволила реалізувати отримання GPS-координат користувача у режимі реального часу, визначення пройденої дистанції, обчислення швидкості та темпу руху. Для візуалізації маршруту використано FlutterMap, що забезпечує відображення маршруту тренування на карті.

Результати тестування підтвердили коректність реалізації механізмів авторизації, взаємодії із серверною частиною, GPS-моніторингу, збереження тренувань, роботи профілю користувача та формування рейтингу. Під час перевірки встановлено, що застосунок стабільно виконує обробку HTTP-запитів, забезпечує коректне оновлення даних та підтримує роботу основних функціональних можливостей.

До основних переваг реалізованого застосунку можна віднести підтримку GPS-моніторингу, автоматичний розрахунок статистичних показників, централізоване збереження результатів тренувань, наявність рейтингової системи та персоналізацію профілю користувача. Разом із цим система має можливості для подальшого розширення, зокрема реалізації соціальних функцій, push-сповіщень, розширеної аналітики спортивної активності та інтеграції із зовнішніми сервісами.

Таким чином, поставлену мету роботи досягнуто, а реалізований мобільний застосунок MiniStrava може бути використаний як основа для подальшого розвитку систем моніторингу спортивної активності користувачів.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було досліджено особливості моніторингу спортивної активності, виконано аналіз сучасних мобільних застосунків для відстеження фізичної активності та проаналізовано технології реалізації мобільних систем спортивного моніторингу. На основі проведеного аналізу визначено основні вимоги до розробки мобільного застосунку, орієнтованого на запис тренувань, GPS-моніторинг та збереження статистичних показників користувача.

У процесі виконання роботи було спроєктовано та реалізовано клієнт-серверний мобільний застосунок MiniStrava, функціональність якого базується на принципах роботи сучасних систем спортивного моніторингу. Реалізовано механізми авторизації та реєстрації користувача, запису спортивної активності, GPS-відстеження маршруту, збереження статистичних даних, формування рейтингу та керування персональним профілем.

Для реалізації клієнтської частини використано фреймворк Flutter, що дозволив створити кросплатформний мобільний застосунок із сучасним інтерфейсом користувача. Для організації взаємодії із серверною частиною застосовано REST API та бібліотеку Dio, що забезпечує передачу даних між компонентами системи. Керування станом застосунку реалізовано за допомогою механізму Provider, а для роботи із геолокацією використано бібліотеку Geolocator. Відображення маршруту спортивної активності реалізовано із використанням FlutterMap.

Під час реалізації системи забезпечено можливість автоматичного отримання GPS-координат користувача, розрахунку дистанції, швидкості, темпу та тривалості тренування. Реалізовано механізм збереження спортивних активностей та перегляду історії тренувань, а також формування статистичних показників користувача.

Проведене тестування підтвердило працездатність основних функціональних можливостей застосунку та коректність взаємодії між

клієнтською і серверною частинами системи. У процесі тестування підтверджено стабільну роботу механізмів авторизації, GPS-моніторингу, отримання статистичних даних, керування профілем та обробки HTTP-запитів.

Таким чином, поставлену мету кваліфікаційної роботи досягнуто, а завдання дослідження виконано. Розроблений мобільний застосунок MiniStrava може використовуватися як основа для подальшого розвитку систем моніторингу спортивної активності та розширення функціональних можливостей, зокрема впровадження соціальних функцій, push-сповіщень, розширеної аналітики та інтеграції із зовнішніми сервісами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Flutter Documentation. URL: https://docs.flutter.dev/ (дата звернення – 18.12.2025).
2. Dart Documentation. URL: https://dart.dev/ (дата звернення – 18.12.2025).
3. REST API Tutorial. URL: https://restfulapi.net/ (дата звернення – 18.12.2025).
4. Geolocator Package Documentation. URL: https://pub.dev/packages/geolocator. (дата звернення – 18.12.2025).
5. Flutter Map Documentation. URL: https://docs.fleaflet.dev/. (дата звернення – 18.12.2025).
6. Strava. Official Website. URL: https://www.strava.com/. (дата звернення – 13.04.2026).
7. Nike Run Club. Official Website. URL: https://www.nike.com/nrc-app (дата звернення – 13.04.2026).
8. Google Fit. Official Website. URL: https://www.google.com/fit/ (дата звернення – 13.04.2026).
9. Research on mobile health and physical activity tracking (scientific articles / Google Scholar / PubMed). (дата звернення – 18.12.2025).
10. Dio Package Documentation. URL: https://pub.dev/packages/dio (дата звернення – 13.04.2026).
11. Provider Package Documentation. URL: https://pub.dev/packages/provider (дата звернення – 13.04.2026).
12. GoRouter Documentation. URL: https://pub.dev/packages/go_router (дата звернення – 13.04.2026).
13. Flutter Localization Documentation. URL: https://docs.flutter.dev/ui/accessibility-and-internationalization/internationalization (дата звернення – 18.12.2025).

14. LatLong2 Package Documentation. URL: https://pub.dev/packages/latlong2 (дата звернення – 13.04.2026).
15. OpenStreetMap Official Website. URL: https://www.openstreetmap.org/ (дата звернення – 13.04.2026).
16. JSON Official Website. URL: https://www.json.org/json-en.html (дата звернення – 13.04.2026).
17. MDN Web Docs. HTTP Request Methods. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods (дата звернення – 13.04.2026).
18. Android Studio Documentation. URL: https://developer.android.com/studio (дата звернення – 13.04.2026).
19. Flutter State Management Documentation. URL: https://docs.flutter.dev/data-and-backend/state-mgmt/simple (дата звернення – 13.04.2026).
20. Geolocation API Concepts. URL: https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API (дата звернення – 19.12.2025).

ДОДАТОК А

Лістинг А.1 – Реалізація авторизації користувача

```
import 'package:dio/dio.dart';
import 'package:flutter/material.dart';
import 'api.dart';

class AuthProvider with ChangeNotifier {
  bool _isLoggedIn = false;
  bool get isLoggedIn => _isLoggedIn;

  String? _email;
  String? get email => _email;

  String? _token;

  Future<void> register({
    required String name,
    required String email,
    required String password,
    required String passwordConfirmation,
  }) async {
    final api = ApiService();

    final response = await api.dio.post(
      '/api/auth/register',
      data: FormData.fromMap({
        'name': name,
        'email': email,
        'password': password,
        'password_confirmation': passwordConfirmation,
      }),
      options: Options(headers: {
        'Accept': 'application/json',
      }),
    );
```

```
  final token = response.data['token'] as String;
  final userId = response.data['user']['id'] as int;

  _saveSession(
    token: token,
    email: email,
    userId: userId,
  );
}
Future<void> login(String email, String password) async {
  final api = ApiService();

  final response = await api.dio.post(
    '/api/auth/login',
    data: {
      'email': email,
      'password': password,
    },
    options: Options(
      headers: {
        'Accept': 'application/json',
      },
    ),
  );

  final token = response.data['token'] as String;
  final user = response.data['user'];

  _saveSession(
    token: token,
    email: user['email'],
    userId: user['id'],
  );
}
```

```
Future<void> logout() async {
  final api = ApiService();

   {
    await api.dio.post(
      '/api/auth/logout',
      options: Options(
        headers: {
          'Accept': 'application/json',
        },
      ),
    );
  }

  _isLoggedIn = false;
  _email = null;
  _token = null;

  api.clearToken();
  notifyListeners();
}

void _saveSession({
  required String token,
  required String email,
  required int userId,
}) {
  _token = token;
  _email = email;
  _isLoggedIn = true;

  ApiService().setToken(token);
  notifyListeners();
}
```

```
  Future<void> sendResetPasswordEmail(String email) async {
    final api = ApiService();

    await api.dio.post(
      '/api/auth/send-reset-password-email',
      data: {
        'email': email,
      },
      options: Options(
        headers: {
          'Accept': 'application/json',
        },
      ),
    );
  }
}
```

Лістинг А.2 – Реалізація GPS-моніторингу тренування

```
  Future<void> startNew() async {
    debugPrint('RecordController: START NEW');

    final ok = await _ensurePermission();
    if (!ok) return;

    final now = DateTime.now();
    _isRecording = true;
    _isPaused = false;
    _startTime = now;
    _elapsed = Duration.zero;
    _distanceMeters = 0;
    _lastPosition = null;
    _path.clear();
    notifyListeners();
```

```
    showMessage(msgStarted);

    _timer?.cancel();
    _timer = Timer.periodic(const Duration(seconds: 1), (_) {
      if (!_isRecording || _isPaused) return;
      _elapsed += const Duration(seconds: 1);
      notifyListeners();
    });

    _posSub?.cancel();
    _posSub = Geolocator.getPositionStream(
      locationSettings: const LocationSettings(
        accuracy: LocationAccuracy.best,
        distanceFilter: 3,
      ),
    ).listen((pos) {
      if (!_isRecording || _isPaused) return;

      _path.add(pos);

      if (_lastPosition != null) {
        final d = Geolocator.distanceBetween(
          _lastPosition!.latitude,
          _lastPosition!.longitude,
          pos.latitude,
          pos.longitude,
        );
        if (d > 0) {
          _distanceMeters += d;
        }
      }
      _lastPosition = pos;
      notifyListeners();
    });
  }
```

Лістинг А.3 – Реалізація збереження спортивної активності

```
Future<Activity> createActivity(Activity activity) async {
    final response = await _dio.post(
      '/api/activity',
      data: {
        'title': activity.name,
        'note': activity.note,
        'activity_type': _mapActivityType(activity.type),
        'distance': activity.distanceMeters,
        'time': activity.duration.inSeconds,
        'pace': activity.paceSecondsPerKm / 60,
        'speed': activity.avgSpeedKmh,
        'track': activity.path
            .map((p) => {
          'latitude': p.lat,
          'longitude': p.lon,
        })
            .toList(),
      },
      options: Options(
        headers: {'Accept': 'application/json'},
      ),
    );

    return Activity.fromApiJson(response.data);
  }
```

Лістинг А.4 – Реалізація отримання історії тренувань

```
Future<List<Activity>> fetchActivities() async {
  final response = await _dio.get(
    '/api/activities',
    options: Options(
      headers: {
        'Accept': 'application/json',
      },
    ),
  );

  final List data = response.data as List;

  return data.map<Activity>((json) {
    final createdAt = DateTime.parse(json['created_at']);

    return Activity(
      id: json['id'],
      start: createdAt,
      end: createdAt.add(
        Duration(seconds: json['time'] ?? 0),
      ),
      distanceMeters: (json['distance'] as num).toDouble(),
      name: json['title'],
      note: json['note'],
      type: ActivityType.values.firstWhere(
            (e) => e.name == json['activity_type'],
        orElse: () => ActivityType.run,
      ),
      path: (json['positions'] as List)
          .map<TrackPoint>(
            (p) => TrackPoint(
          lat: (p['latitude'] as num).toDouble(),
          lon: (p['longitude'] as num).toDouble(),
```

```
        ),
      )
          .toList(),
    );
  }).toList();
}
```

Лістинг А.5 – Реалізація рейтингової системи

```
Future<void> loadRanking() async {
  _loading = true;
  _error = null;
  notifyListeners();

  try {
    _entries = await _repo.fetchWeeklyRanking();
  } on DioException catch (e) {
    _error = e.response?.data['message'] ?? 'Failed to load
ranking';
  } finally {
    _loading = false;
    notifyListeners();
  }
}
```

Лістинг А.6 – Реалізація керування профілем користувача

```
Future<void> loadProfile() async {
  _loading = true;
  notifyListeners();

  _profile = await _repo.fetchProfile();

  _loading = false;
  notifyListeners();
}
```

```
Future<void> saveProfile({
  required String? name,
  required String? surname,
  required DateTime? birthdate,
  required String? gender,
  required double? height,
  required double? weight,
}
```