НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри
комп'ютерних наук
(назва кафедри)
_____ Голуб Б. Л.
(підпис) (ПІБ)
“___” _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

«Мобільний додаток інформаційної системи приватного відстеження переміщень»

Спеціальність 121 – «Інженерія програмного забезпечення»

Гарант освітньої програми
доцент к.т.н. _____ Вайганг Г.О.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи
доцент к.т.н. _____ Боярінова Ю.Є.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Крук О.О.
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____комп'ютерних наук _____
__доцент к.т.н._____ _Голуб Б. Л. __
(науковий ступінь, вчене звання) (підпис) (ПІБ)
“____” _____2025_р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту

_____Круку Олександру Олександровичу_____
(прізвище, ім’я, по батькові)

Спеціальність 121 – «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи: Мобільний додаток інформаційної системи приватного відстеження переміщень

Затверджена наказом ректора НУБіП України від “_16_” _грудня_____2024 р. №2249 “С”

Термін подання завершеної роботи на кафедру _____
(рік, місяць, число)

Вихідні дані до бакалаврської кваліфікаційної роботи

Дослідження можливостей застосування мобільних інформаційних систем для приватного відстеження переміщень. Вплив використання персонального трекінгу на підвищення обізнаності користувача щодо власної активності та мобільності.

Перелік питань, які потрібно розробити:

1) аналіз систем приватного відстеження переміщень як об’єкта дослідження
2) проєктування інформаційного та програмного забезпечення
3) розробка інформаційного та програмного забезпечення
4) тестування та введення в експлуатацію системи

Дата видачі завдання “_____”_____20___ р.

Керівник бакалаврської кваліфікаційної роботи _____ ___Боярінова Ю. Є ____
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ _____Крук О.О._____
(підпис) (прізвище та ініціали студента

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (Application Programming Interface) - інтерфейс програмування застосунків; набір функцій і протоколів, що забезпечують взаємодію між різними компонентами програмного забезпечення

GUI (Graphical User Interface) — графічний інтерфейс користувача, що забезпечує візуальну взаємодію із застосуванням

JSON (JavaScript Object Notation) — текстовий формат обміну структурованими даними, зручний для збереження та передачі через мережу

REST (Representational State Transfer) — архітектурний стиль взаємодії клієнта та сервера через інтернет-протокол HTTP, що широко використовується для створення веб-API

SDK (Software Development Kit) – набір програмних інструментів для розробки застосунків для певної платформи

SQLite — вбудована реляційна база даних, яка використовується для локального збереження інформації на мобільному пристрої

HTTP (HyperText Transfer Protocol) — протокол передачі даних між клієнтом та сервером у глобальній мережі Інтернет

.

ВСТУП

В умовах стрімкого розвитку цифрових технологій дедалі більшої актуальності набувають питання збереження приватності користувача, контролю над власними персональними даними, а також створення автономних рішень, які не потребують постійного підключення до інтернету або синхронізації з хмарними сервісами. Одним із найбільш чутливих типів інформації є геолокаційні дані, які можуть багато розповісти про звички, активність, переміщення та стиль життя користувача.

Більшість сучасних мобільних додатків для трекінгу або навігації (наприклад, Google Maps, Strava, Life360 тощо) передбачають обов'язкове збереження даних на стороні зовнішніх серверів. Це створює ризики витоку інформації, залежності від політик сторонніх компаній та неможливість повноцінного контролю над особистою історією переміщень. У зв'язку з цим виникає потреба у локальному, незалежному інструменті, який дозволяє користувачу відстежувати власні маршрути, додавати нотатки до конкретних геолокаційних точок та зберігати інформацію виключно на власному пристрої або на власному сервері через систему резервного копіювання.

Мета роботи

Метою даної дипломної роботи є створення мобільного додатку інформаційної системи приватного відстеження переміщень користувача, яка дозволяє:

- зберігати маршрути пересування на локальному пристрої;
- додавати текстові нотатки до обраних точок маршруту;
- переглядати маршрути у вигляді карти з візуалізацією;
- експортувати та імпортувати дані у форматі JSON;
- створювати резервні копії геоданих на власному сервері;
- відновлювати дані у разі втрати або перевстановлення застосунку.

Особливу увагу було приділено інтуїтивному інтерфейсу, адаптивному дизайну, а також захисту особистої інформації.

Методи та технології

- У процесі розробки було використано такі сучасні методи та засоби:
- Flutter SDK — для реалізації мобільного застосунку з кросплатформенною підтримкою Android та iOS;
- Dart — основна мова програмування клієнта;
- SQLite — для зберігання маршрутів та нотаток на мобільному пристрої;
- Provider — для реалізації реактивної архітектури стану;
- Django — для побудови серверної частини з REST API
- Django REST Framework — для реалізації функціоналу резервного копіювання та відновлення даних;
- JSON — як формат передачі геоданих між клієнтом і сервером;
- MapTiler + flutter_map + flutter_earth_globe — для відображення маршрутів у 2D та 3D-режимах;

Під час проєктування було використано методи структурного аналізу, розробки UI/UX-прототипів, а також тестування окремих компонентів системи.

Структура пояснювальної записки

Пояснювальна записка до дипломної роботи складається з п’яти основних розділів:

У першому розділі описано постановку задачі, обґрунтовано актуальність теми та визначено основні вимоги до системи;

У другому розділі розглянуто архітектурні рішення, вибір технологій, структура системи;

Третій розділ присвячено реалізації основного функціоналу мобільного застосунку та серверної частини;

У четвертому розділі подано відомості про технічні характеристики, вимоги до середовища розгортання та склад інсталяційного пакету;

П’ятий розділ містить загальні висновки, оцінку досягнутих результатів та можливі напрями подальшого розвитку.

Окремими додатками подано фрагменти коду, приклади API-запитів, а також скріншоти роботи застосунку.

1. СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області додатку відстеження переміщень

Предметна область додатку охоплює процеси збирання, збереження, аналізу та візуалізації геолокаційної інформації користувача в реальному часі або з певною періодичністю. Основною метою такого застосування є забезпечення приватного моніторингу власної активності пересування для особистого використання, з можливістю ведення історії, створення нотаток до маршрутів, а також резервного копіювання геоданих.

У сучасному світі зростає піт на інструменти, що дозволяють людям контролювати власну мобільність, вести просту аналітику на основі треків, а також зберігати важливу контекстну інформацію, пов'язану з локаціями (наприклад, у вигляді нотаток). Це може бути корисним як для повсякденної життя (запам'ятати, де був і чому), так і для професійних потреб — фіксації маршрутів кур'єрів, польових працівників тощо.

Ключові поняття предметної області:

- Користувач — особа, яка використовує застосування на мобільному пристрої для фіксації власного місцяперебування. Дані можуть бути доступні лише йому.
- Точка маршруту (локація) — запис, що містить координати GPS (широта, довгота), час фіксації та, за потреби, інші метадані (висота, швидкість).
- Маршрут (трек) — послідовність точок, зафіксованих в межах однієї сесії трекінгу.
- Записка — текстова інформація, яку користувач може прикріпити до конкретної локації або до цілого маршруту.
- Режим трекінгу — спосіб запуску процесу фіксації координат: автоматичне (постійне тло) або вручну (за кнопкою).
- База даних — внутрішнє сховище, яке містить історію переміщень, примітки, резервні копії, прив'язані до користувача.

- Інтерфейс візуалізації — карта з можливістю перегляду маршрутів, точок, приміток, а також екран історії з доступом до попередніх записів.

Таким чином, предметна область охоплює аспекти мобільного трекінгу, обробки геоданих, зручного введення контекстної інформації (нотаток) та роботи з візуалізацією, дотримуючись при цьому принципів приватності та зберігання даних лише на стороні користувача або у власному захищеному бекенді.

1.1.1 Ключові учасники системи

У системі мобільного додатку для приватного відстеження переміщень можна виокремити таких ключових учасників:

Користувач (User) — головний учасник системи, який:

- запускає/зупиняє трекінг переміщень;
- переглядає картку з маршрутами;
- додає примітки до точок або маршрутів;
- переглядає та редагує історію пересувань;
- експортує/імпортує дані;
- створює резервні копії даних

Система трекінгу (Tracking Service) — фоновий сервіс, який автоматично фіксує координати користувача під час активного режиму трекінгу.

Сервер (Backend) (опційно, якщо використовується хмарне збереження) — приймає резервні копії, надає можливість відновлення даних.

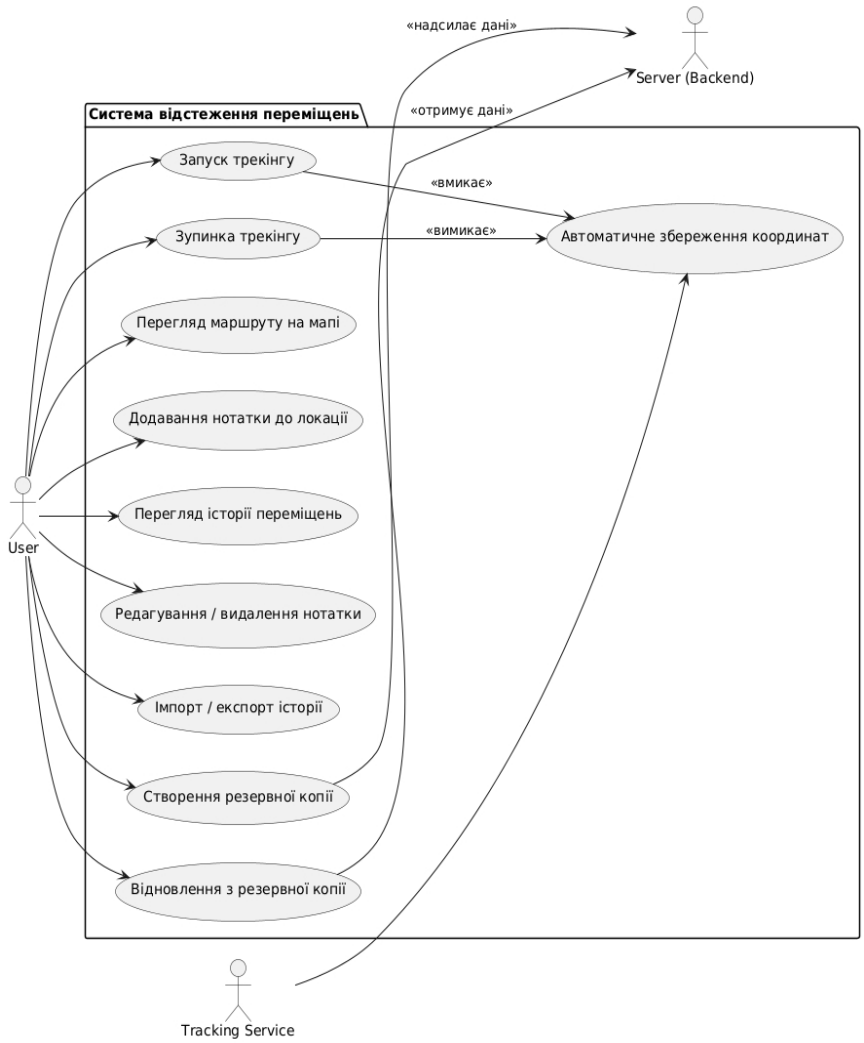


Рис. 1.1 - Діграма прецедентів системи відстеження переміщень

1.2 Аналіз вимог до додатку

Розроблений мобільний приклад забезпечує комплексний функціонал для відстеження переміщень користувача в реальному часі. Основною метою системи є збір, збереження, аналіз та візуалізація геолокаційних даних, що дозволяє формувати інтерактивні карти маршрутів. Застосування фіксує координати, що відображає пройдені шляхи на карті, а також дозволяє користувачам додавати текстові заметки до конкретних точок маршруту чи обраних локацій.

Крім мобільного компонента система включає серверну частину, реалізовану у вигляді централізованого хмарного сервера. Сервер виконує ряд важливих функцій: збереження даних користувачів, їх обробку, регулярне резервне копіювання, а також синхронізацію між пристроями. Такий підхід

забезпечує збереження історії переміщень навіть у разі втрати або заміни мобільного пристрою, а також дозволяє швидко відновити дані за потреби.

Серверна інфраструктура підтримує роботу з обліковими записами користувачів, забезпечує доступ до даних лише авторизованим особам, гарантує конфіденційність інформації та її захист згідно з актуальними стандартами. Усі маршрути, разом із прикріпленими до них примітками, можуть бути експортовані у вигляді резервних копій, а також імпортовані при повторній інсталяції застосунку чи зміні пристрою.

Таким чином, запропонований програмний продукт поєднує зручність мобільного інтерфейсу з надійністю та масштабованістю серверного зберігання, надаючи користувачеві гнучкий інструмент для приватного відстеження переміщень, з можливістю створення індивідуальних карт, нотаток та безпечного доступу до своєї геолокаційної історії.

1.2.1 Функціональні вимоги

1. Реєстрація та авторизація користувачів

- Створення облікового запису : користувач повинен мати можливість зареєструватися в системі, вказавши адресу електронної пошти та пароль.
- Вхід до системи : після реєстрації користувач має можливість авторизуватися, використовуючи свої облікові дані.
- Захист сесії : після входу система повинна видавати токен доступу (JWT), який використовується для підтвердження шкірного подальшого запиту.

2. Робота з геоданими

- Прийом маршрутів від мобільного додатку : система має приймати GPS-координати, які надсилаються клієнтським застосуванням у вигляді масиву точок (timestamp, широта, довгота).
- Збереження координат у БД : усі отримані маршрути повинні зберігатися в базі даних разом із часовими мітками та ID користувача.

- Заметки до маршрутів : користувач може створювати текстові примітки, прив'язані або до конкретної точки маршруту, або до всього маршруту за дату.
- Редагування, перегляд та видалення нотаток : реалізована можливість перегляду списку нотаток, внесення змін, а також їх повного видалення.

3. Резервне копіювання та відновлення

- Створення резервної копії : користувач має можливість створити резервну копію своїх маршрутів та нотаток та завантажити її на сервер.
- Відновлення даних : реалізовано функціонал відновлення всіх збережених даних (координат, маршрутів, нотаток) з резервної копії.

4. Експорт/імпорт даних

- Експорт у файл : користувач може експортувати свої маршрути та заметки у файл (наприклад, JSON або CSV) для зберігання локально.
- Імпорт із файлу : підтримується завантаження файлів резервної копії назад у систему для відновлення.
-

1.2.2 Нефункціональні вимоги

1. Безпека

Усі запити до API мають бути захищені за допомогою токенів доступу (JWT).

Паролі користувачів мають зберігатись у базі даних у вигляді хешу з використанням сучасних алгоритмів (наприклад, bcrypt або Argon2).

Должна бути реалізована обробка помилок авторизації, обмеження кількості спроб входу (rate limiting), а також CSRF-захист для вебінтерфейсу (якщо є).

2. Приватність

Доступ до персональних даних користувача мають виключно він сам. Сторонні користувачі не мають змоги переглядати маршрути чи примітки інших.

Кожен запит до сервера повинен перевіряти авторизацію та відповідність користувача до запитуваних даних.

3. Масштабованість

Архітектура повинна дозволяти розширення без зміни основної логіки (напр., додавання нових типів нотаток чи зберігання медіа).

Система має бути готова до збільшення кількості користувачів та обсягу геоданих без значного падіння продуктивності.

За необхідності можлива реалізація кешування, горизонтального масштабування, розділення на сервіси.

4. Надійність

Передбачено періодичне резервне копіювання бази даних.

Усі критичні операції (запис маршрутів, відновлення даних) мають виконуватися в межах транзакцій, щоб уникнути часткової втрати даних.

Сервер повинен мати логування помилок та систему моніторингу працездатності.

5. Продуктивність

API має повертати відповіді в межах кількох сотень мілісекунд за типових умов.

Фонові процеси (наприклад, обробка великих маршрутів або імпорт даних) мають виконуватися асинхронно.

6. Платформозалежність

Клієнт : мобільний приклад створено за допомогою Flutter , що забезпечує кроссплатформеність (Android/iOS).

Сервер : бекенд реалізовано на Python із використанням Django REST Framework – гнучкого та надежного фреймворку для створення API.

1.3 Моделювання предметної області

Моделювання предметної області є критично важливим етапом під час проектування інформаційної системи, оскільки дозволяє чітко визначити основні сутності, їх атрибути та взаємозв'язки між ними. Це забезпечує логічну цілісність системи, спрощує подальшу реалізацію програмної логіки та сприяє ефективній роботі з базами даних та програмними інтерфейсами.

Для мобільного додатку приватного відстеження переміщень було виділено п'ять ключових сутностей, які відображають основні об'єкти взаємодії користувача з системою та структуру збереження інформації. Кожна з них має власне призначення, набір атрибутів і визначену роль у загальній архітектурі застосування.

User (Користувач) — базова сутність, що представляє особу, яка використовує додаток. Зберігає дані авторизації та виступає власником усіх пов'язаних з ним даних (маршрутів, точок, нотаток, резервних копій).

TrackedRoute (Маршрут) — логічна одиниця пересування, що складається з послідовності географічних точок, початкової години та розрахованого відстані. Користувач може мати кілька маршрутів, які зберігаються незалежно один від одного.

Location (Локація) — окрема координатна точка (широта, довгота, мітка часу), яка зберігається або в контексті маршруту, або автономно, залежно від налаштувань збору даних.

LocationNote (Нотатка) — текстова примітка, прив'язана до певної географічної точки або маршруту. Служити для збереження контекстної інформації, думок чи завдань користувача, пов'язаних із конкретними місцями.

Backup (Резервна копія) — сутність, що описує збереження повної копії даних користувача на сервері. Містить шлях до файлу та мітку часу створення копії. Служити для відновлення інформації у разі втрати чи перенесення даних на інше пристрій.

Таке моделювання дозволяє створити гнучку, масштабовану архітектуру, в якій шкірна сутність логічно ізольована, але має зрозумілі зв'язки з іншими компонентами. Це полегшує розширення функціональності системи в майбутньому та забезпечує високий рівень відповідності між бізнес-логікою та структурою даних.

1.4 Огляд інформаційних джерел та існуючих рішень

1.4.1 Аналіз існуючих рішень

На сучасному ринку існує велика кількість мобільних застосунків, що реалізують функції трекінгу переміщень, проте більшість із них орієнтовані або на спортивне використання, або на комерційний моніторинг. Проведено аналіз найпопулярніших рішень, які частково чи повністю перетинаються за функціональністю із запропонованим додатком.

Google Timeline (у складі Google Maps)

Призначення: Автоматичне відстеження переміщень користувача, прив'язане до акаунта Google.

Висока точність, інтеграція з іншими Google-сервісами, збереження історії.

Недоліки: Закритий код, обмежений контроль за даними, відсутність ручного редагування точок та маршрутів, повна залежність від хмари.

Висновок: Сервіс не орієнтований на конфіденційність чи кастомізацію. Не підходить для приватного використання.

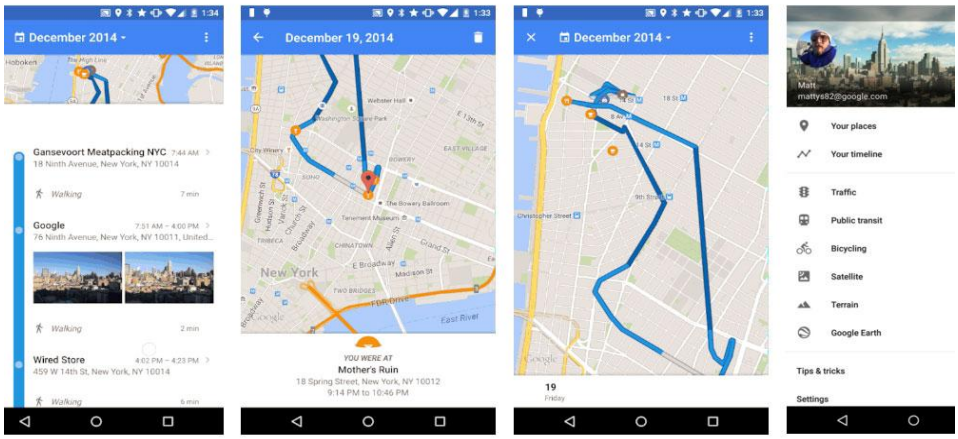


Рис. - 1.2 Інтерфейс Google Timeline

Strava, Komoot, Geo Tracker

Призначення: Відстеження маршрутів для активностей (біг, велоспорт, піші походи).

Переваги: Детальний запис треків, аналітика, візуалізація на карті.

Недолік: Спортивна спрямованість, орієнтація на публічний обмін маршрутами, частково платний функціонал.

Висновок: Не передбачено гнучке додавання нотаток чи приватне збереження на пристрої без синхронізації.

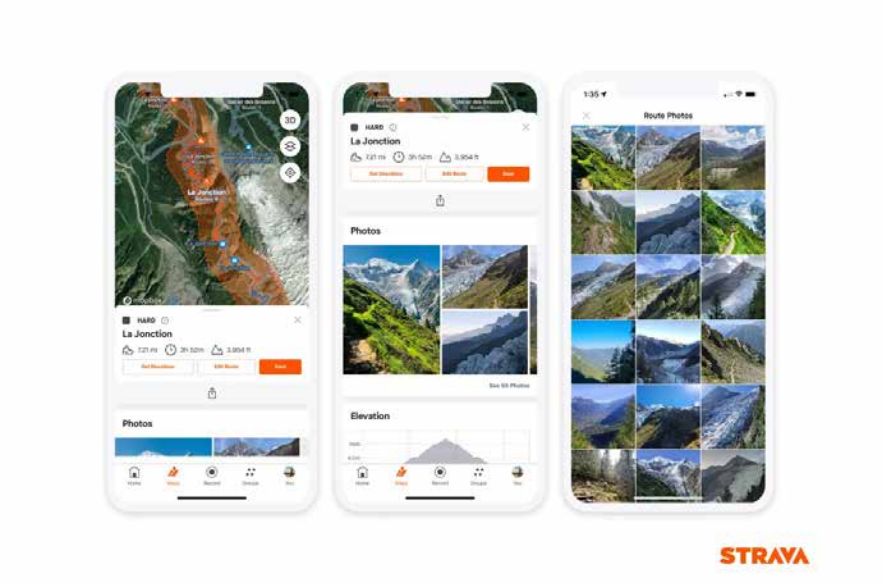


Рис. 1.3 - Інтерфейси Strava, Komoot, Geo Tracker

Life360, Find My (Apple)

Призначення: Сімейне або корпоративне стеження за місцезнаходженням користувачів у реальному часі.

Переваги: Реальний моніторинг, push-сповіщення.

Недолік: Повна відсутність функціоналу для роботи з примітками, маршрутом як історією, прив'язка до облікового запису.

Висновок: Використовуються для нагляду, а не для приватного аналізу переміщень.

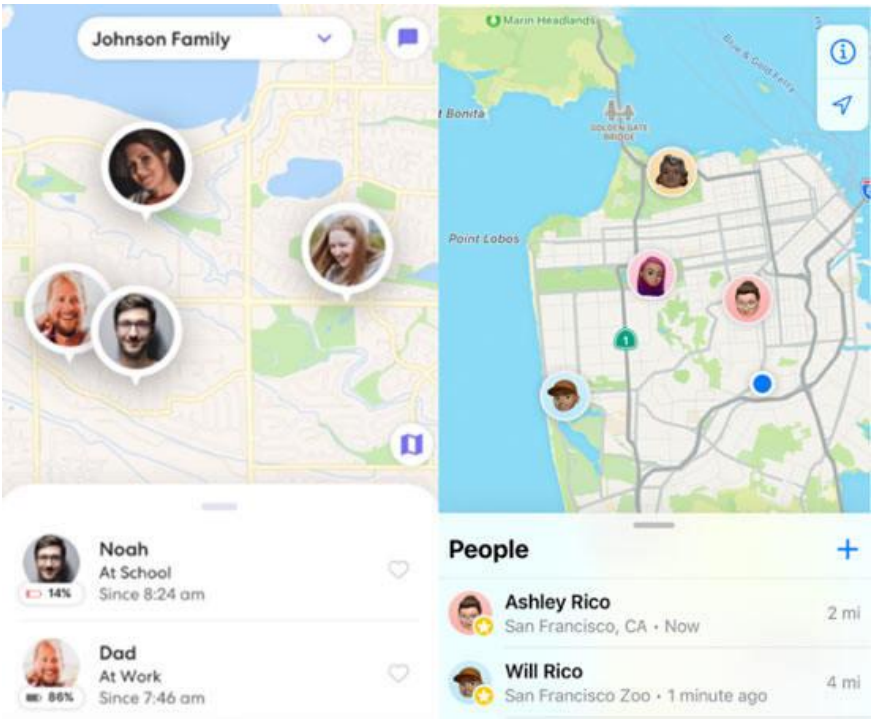


Рис 1.4 - Інтерфейс Life360, Find My (Apple)

OpenTracks (Android, Open Source)

Призначення: Відкритий застосунок для трекінгу та збереження маршрутів.

Переваги: працює офлайн, зберігає дані локально, підтримує експорт GPX/CSV, відкритий код.

Недоліки: відсутність підтримки нотаток, інтерфейс без гнучкої кастомізації, обмежена підтримка резервного копіювання.

Висновок: айближчий аналог, проте не підтримує прив'язку текстової інформації до точок або інтерфейсу заміток.

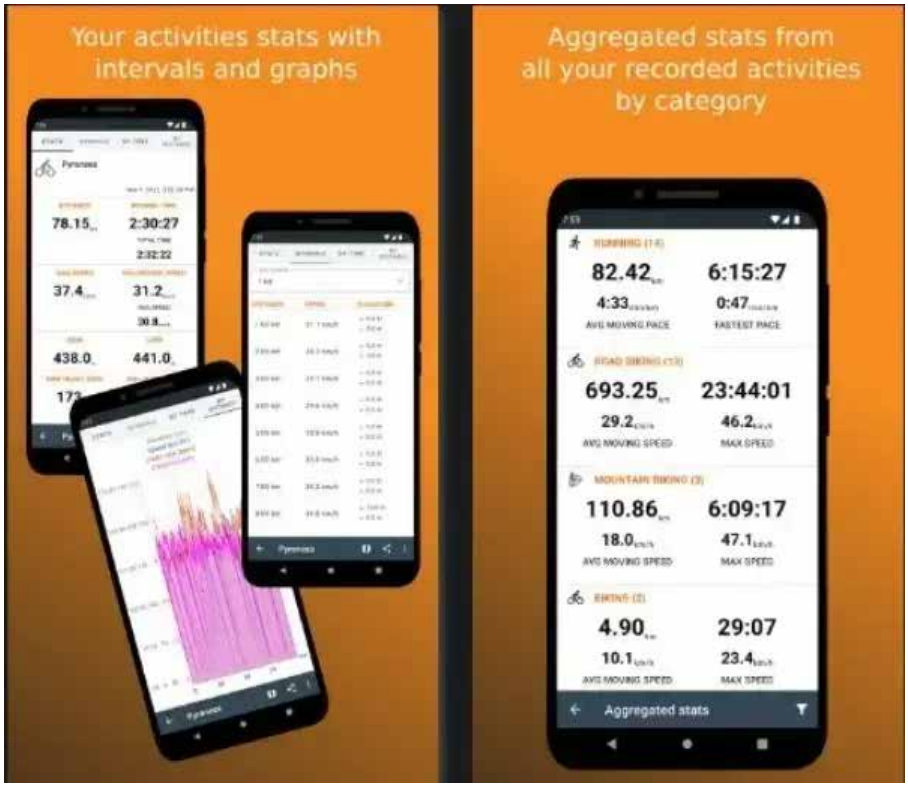


Рис. 1.5 - Інтерфейс OpenTracks (Android, Open Source)

1.4.2 Основні недоліки існуючих рішень

На сучасному ринку мобільних застосунків представлено чимало рішень, що забезпечують функції трекінгу геолокації та збереження маршрутів користувача. Серед найпопулярніших – Google Timeline, Strava, Life360, OpenTracks тощо. Кожен із цих сервісів має свої переваги: точність фіксації координат, візуалізація маршрутів, підтримка GPS-експорту, інтеграція з іншими сервісами чи платформами.

Однак при глибшому аналізі виявляється, що жодний з них не надає одночасно повного набору функцій, які можуть бути критично важливими для окремого користувача, що прагне контролювати власні переміщення з дотриманням високого рівня конфіденційності.

- не підтримують додавання контексту до координат (текстові нотатки чи коментарі);
- не забезпечують повноцінного резервного копіювання та відновлення даних за межами внутрішніх сервісів;
- обмежені у функціоналі експорту/імпорту даних у відкритих форматах;

- орієнтовані або на спортивне використання, або на сімейний моніторинг, що не завжди відповідає завданням приватного трекінгу.

Нижче наведено порівняльну таблицю деяких поширених рішень із коротким описом їх сильних і слабких сторін.

Таблиця 1.1

Таблиця порівняння існуючих рішень

Застосунок	Переваги (+)	Недоліки (–)
Google Timeline	+ Висока точність+ Автоматичне відстеження+ Інтеграція з Google Maps	– Немає локального збереження– Відсутність кастомізації– Не підтримує нотатки
Strava / Komoot / Geo Tracker	+ Детальні маршрути+ Гарна візуалізація+ Експорт у GPX/CSV	– Орієнтовані на спорт– Немає текстових нотаток– Частково платні
Life360 / Find My	+ Реальний моніторинг+ Сповіщення+ Кросплатформеність	– Відсутність маршрутів і нотаток– Залежність від обліковок
OpenTracks	+ Відкритий код+ Працює офлайн+ Експорт у GPX/CSV	– Відсутні нотатки– Мінімальний інтерфейс– Обмежені резервні копії

1.4.3 Мотивація створення власного рішення

На сучасному етапі розвитку цифрових технологій на ринку представлено велику кількість мобільних застосунків, що реалізують функції збору та обробки геолокаційних даних. Серед них найбільш відомими є Google Timeline, Strava, Life360, а також ряд спеціалізованих трекерів. Проте більшість із них орієнтовані на конкретні вузькі сценарії — спортивні тренування, моніторинг членів родини, соціальні функції, обмін маршрутами тощо.

Значна частина таких застосунків передбачає централізоване зберігання даних у хмарі або на серверах сторонніх компаній. Це, у свою чергу, ставить під сумнів приватність і контроль користувача над власною інформацією. Крім того, більшість рішень не передбачають можливості контекстної роботи з маршрутами, зокрема додавання текстових нотаток до окремих точок, редагування маршрутів або експорту історії переміщень у зручному форматі.

Наявні open source-альтернативи, такі як OpenTracks, GPSLogger, OwnTracks, частково вирішують окремі технічні задачі, проте не пропонують цілісного рішення, яке би одночасно поєднувало:

- повну автономність та офлайн-роботу без реєстрацій;
- збереження маршрутів і прив’язку нотаток до геоточек;
- візуалізацію даних на мапі;
- резервне копіювання та відновлення інформації у разі потреби;
- гнучкий імпорт та експорт у форматі JSON.

У зв’язку з цим постала необхідність розробки власного застосунку, який враховує потреби користувача в конфіденційності, гнучкості та зручності використання. Основними вимогами до системи стали: локальне збереження всіх даних без сторонніх сервісів, можливість залишати нотатки до маршрутів, візуальна простота інтерфейсу та підтримка резервного копіювання через власний сервер.

Таким чином, запропоноване рішення заповнює нішу персонального цифрового інструменту для ведення приватної історії переміщень — без необхідності підключення до сторонніх платформ або компромісів щодо особистих даних.

1.5 Постановка завдання

Метою даного проекту є розробка мобільного застосунку для приватного відстеження переміщень користувача з можливістю ведення історії, додавання контекстних нотаток, візуалізації маршрутів на карті та резервного копіювання даних. Застосунок має працювати автономно (без постійного підключення до

Інтернету), зберігати інформацію локально на пристрої, а також надавати за бажанням базову можливість синхронізації із сервером для резервного збереження.

Основні завдання, які необхідно реалізувати:

Реалізувати трекінг переміщень:

- автоматичне збирання координат у фоновому режимі;
- фіксація часу, швидкості, тривалості та тривалості маршруту.
    - Забезпечити збереження маршрутів та локацій:
- створення маршрутів як послідовностей геоточок;
- локальне збереження у базі даних (SQLite).

Реалізувати роботу з примітками:

- додавання текстових нотаток до окремих точок або до маршруту;
- редагування, перегляд та видалення приміток.
    - Візуалізація даних на карті:
- відображення пройдених маршрутів;
- маркери нотаток та поточної позиції.

Інтерфейс користувача:

- мінімалістичний, інтуїтивно зрозумілий дизайн;
- навігація між екранами: історія, трекінг, акаунт, карта.

Резервне копіювання та відновлення:

- створення копії історії даних у файл;
- збереження копії на сервер;
- можливість відновлення при втраті чи зміні пристрою.

Імпорт / експорт даних:

- підтримка JSON або CSV;
- зручне перенесення даних між пристроями.

Приватність і безпека:

- повна автономність — жодних сторонніх сервісів без згоди;
- доступ до даних має лише власник пристрою;
- шифрування або захист копій паролем (опційно).

1.5.1 Очікуваний результат:

Створення кроссплатформеного мобільного застосування (на базі Flutter), який дозволяє користувачеві:

- контролювати власні переміщення;
- зберігати важливу контекстну інформацію;
- переглядати історію маршрутів та нотаток;
- самостійно керувати своїми даними, не порушуючи конфіденційності.

2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

На початковому етапі проектування додатку було побудовано ER-діаграму , що відображає логічну структуру бази даних. У розробленій моделі виділено такі сутності: User, TrackedRoute, Location, LocationNote, Backup. Між ними встановлено зв'язки типу один-до-багатьох (1:N).

Відповідність нормальним формам:

Усі таблиці задовольняють першу нормальну форму (1NF): кожне поле містить атомарне значення.

Дані не повторюються в межах однієї таблиці, всі залежності функціональні — отже, реалізовано іншу нормальну форму (2NF).

У таблицях немає транзитивних залежностей, кожен неключовий атрибут залежить лише від первинного ключа — дотримано третьої нормальної форми (3NF).

Таким чином, модель є реляційною , оптимізованою для зберігання та обробки структурованих даних. Вона не містить елементів нелереляційної структури (наприклад, вкладених масивів чи ієрархічних зв'язків на полях).

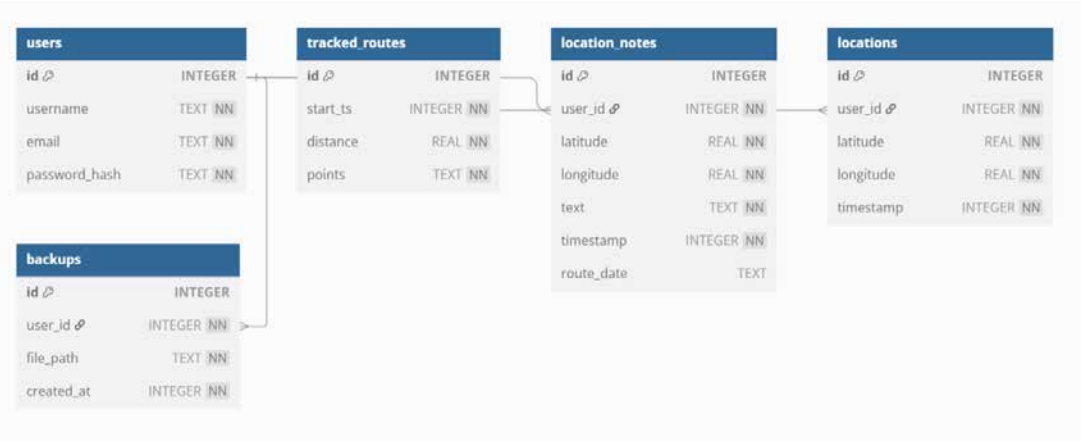


Рис. 2.1 - Er діаграма додатку відслідковування переміщень

Таблиця 2.1

Таблиця сутностей та їх опису

№	Назва таблиці	Ключові поля	Призначення таблиці
1	users	id, username, email, password_hash	Зберігає інформацію про користувачів системи (логін, пошта, пароль у хеші).
2	tracked_routes	id, start_ts, distance, points, user_id	Містить маршрути користувача: час початку, довжину та список координат (JSON).
3	locations	id, user_id, latitude, longitude, timestamp	Зберігає окремі геолокаційні точки, отримані під час трекінгу.
4	location_notes	id, user_id, latitude, longitude, text, timestamp, route_date	Містить нотатки, які користувач може додавати до точок або маршрутів.
5	backups	id, user_id, file_path, created_at	Зберігає інформацію про резервні копії даних користувача (шлях до файлу, час).

2.2 Діаграма класів та кооперацій

На етапі проектування прикладного програмного забезпечення було побудовано діаграму класів, яка відображає структуру основних об'єктів системи та їх зв'язки. Кожен клас відповідає одній із сутностей предметної області, що було визначено раніше: користувач, маршрут, локація, нотатка та резервна копія.

Між класами встановлено асоціації типу «один до багатьох», зокрема: один користувач може мати багато маршрутів, нотаток, точок геолокації та резервних копій. Усі класи містять базові атрибути, необхідні для зберігання

інформації, а також можуть мати допоміжні методи, наприклад, для перетворення в JSON.

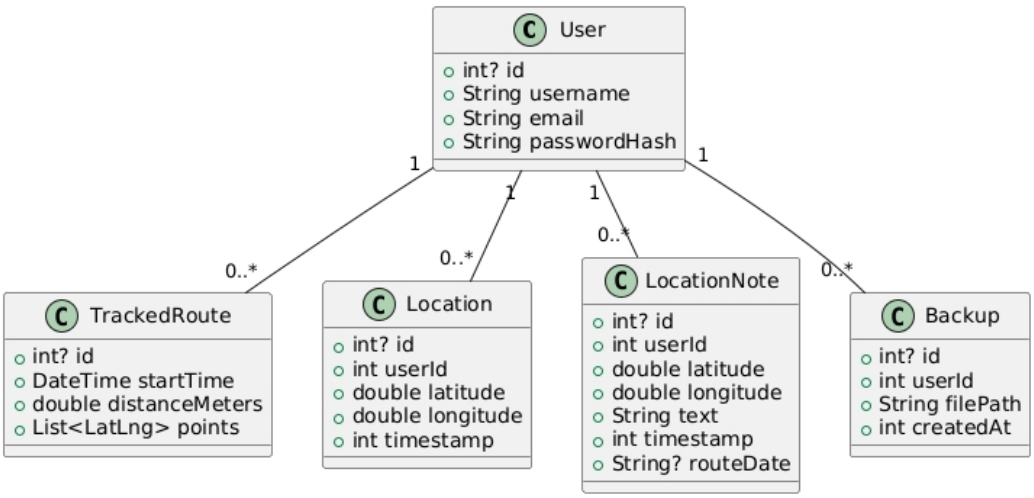


Рис 2.2 - Діаграма класів та кооперативів

2.3 Діаграма пакетів

Для впорядкування структури застосування під час розробки було виконане розбиття на логічні пакети. Кожен пакет містить окрему функціональну частину системи та відповідає за певний аспект взаємодії.

Було виділено такі основні пакети:

- data- класи моделей, що описують структуру даних;
- services— сервіси, які взаємодіють з базою даних, сервером та геолокаційними API;
- screens- екрани користувацького інтерфейсу (мапа, історія, акаунт);
- providers— постачальники стану (наприклад, трекінгу та нотаток), що використовуються у Flutter через Provider.

Таке розбиття забезпечує модульність коду, спрощує супровід проекту та дозволяє уникнути дублювання логіки.

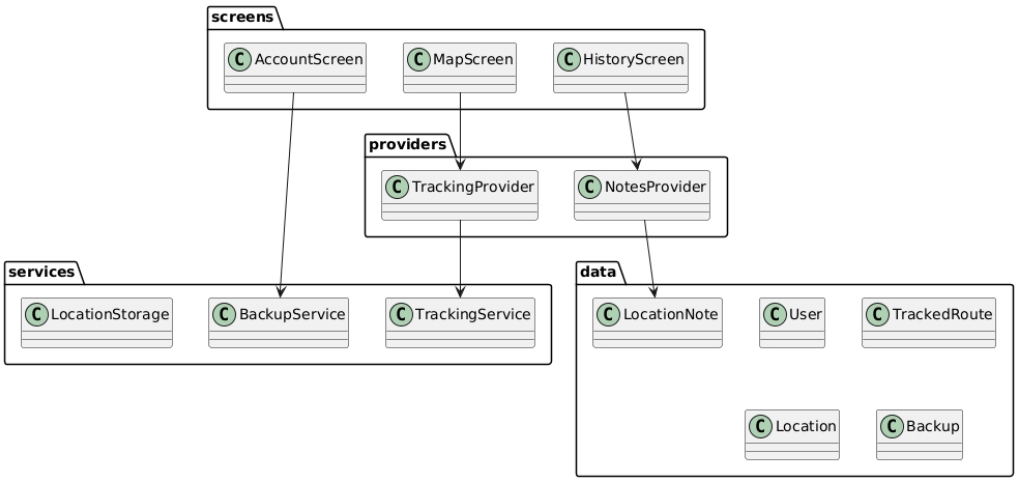


Рис 2.3 - Діаграма пакетів

2.4 Діаграма компонентів

На завершальному етапі проекту було побудовано діаграму компонентів, яка демонструє взаємодію між основними частинами системи.

Програмна система складається з двох головних компонентів:

- мобільного застосування , що реалізований на Flutter;
- серверної частини , що побудована на базі Django.
- Мобільний застосування включає в себе:
- інтерфейс користувача;
- управління станом (через Provider);
- локальну базу даних SQLite;
- мережевий клієнт для обміну даними із сервером.

Серверна частина містить REST API, через які мобільний приклад здійснює авторизацію користувача, завантаження резервних копій та їх відновлення. Дані на сервері зберігаються у базі PostgreSQL.

Дана архітектура дозволяє розмежувати відповідальність між клієнтом та сервером, забезпечити автономність роботи в офлайн-режимі та реалізувати синхронізацію лише за потреби.

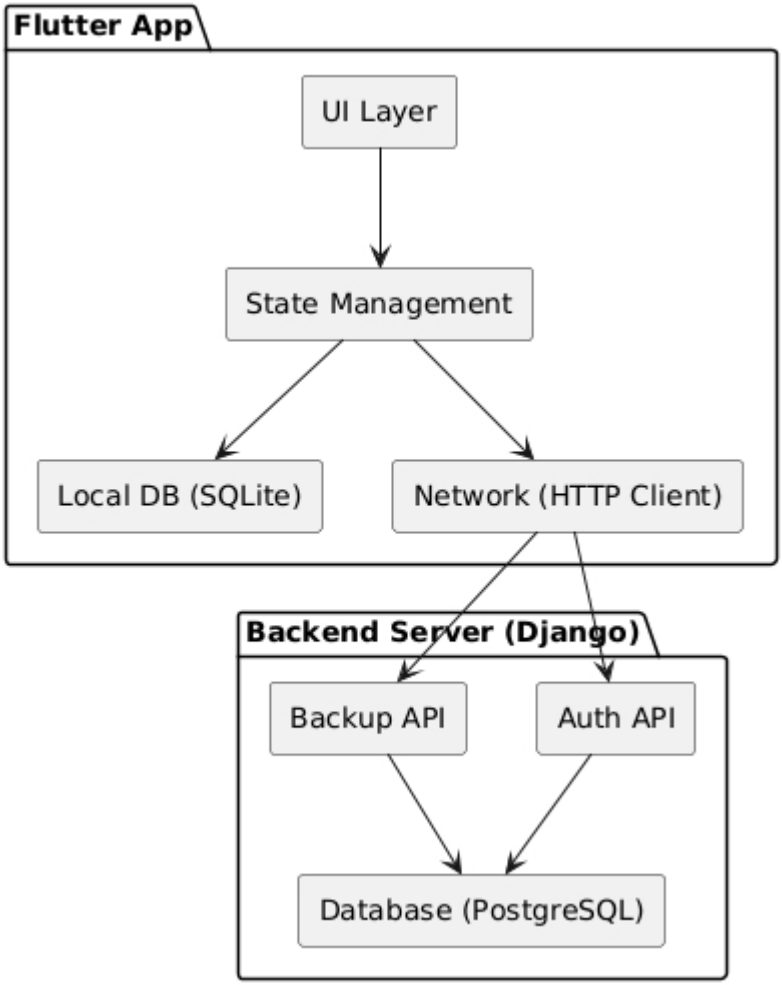


Рис 2.4 - Діаграма компонентів

3. РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Система управління інформаційною базою

У проєктованій системі використовуються дві системи управління базами даних:

SQLite — для локального зберігання даних на мобільному пристрої

PostgreSQL — для централізованого збереження резервних копій на сервері

Локальна СУБД — SQLite

SQLite є вбудованою реляційною базою даних, яка ідеально підходить для використання у мобільних застосунках. Вона не потребує встановлення окремого сервера, забезпечує швидкий доступ до даних та має повну підтримку SQL.

У мобільному застосунку SQLite використовується для зберігання всієї поточної інформації користувача:

- маршрутів (tracked_routes);
- координатних точок (locations);
- нотаток (location_notes);
- облікових даних (users);
- інформації про резервні копії (backups).

Доступ до бази реалізований за допомогою Dart-сервісів, що виконують CRUD-операції та перетворення даних між об’єктами і SQL-таблицями.

Серверна СУБД — PostgreSQL

Для зберігання резервних копій використовується серверна база даних PostgreSQL, яка розгорнута на бекенді, створеному за допомогою Django. PostgreSQL є потужною об'єктно-реляційною СУБД з високою надійністю, продуктивністю та широкими можливостями для масштабування.

На сервері зберігаються лише ті дані, які були явно надіслані з мобільного застосунку — зокрема, файли резервного копіювання та службова інформація, пов’язана з ними. Це забезпечує:

- централізоване збереження копій
- можливість відновлення даних у разі втрати доступу до пристрою;
- розмежування локальних і хмарних даних з фокусом на приватність.
- Взаємодія між базами

Обидві СУБД працюють незалежно одна від одної. Основна частина роботи відбувається з локальною базою (SQLite), що дозволяє забезпечити автономність та швидкодію застосунку. При бажанні користувача відбувається передача даних на сервер, де вони зберігаються у PostgreSQL як резервна копія. Це дозволяє поєднати переваги локального зберігання (приватність, офлайн-доступ) із надійністю серверного архівування.

3.2 Розробка інформаційної бази

Локальна база даних (SQLite)

У мобільному застосунку реалізовано структуру локальної бази даних за допомогою SQLite. Створено п’ять основних таблиць:

- users — зберігає облікові записи користувачів;
- tracked_routes — містить дані маршрутів (час, довжина, точки у форматі JSON);
- locations — зберігає координатні точки з міткою часу;
- location_notes — дозволяє прив’язати текстові нотатки до конкретної локації або дати маршруту;
- backups — описує створені локальні резервні копії.

Розробка фізичної структури передбачає створення первинних ключів, зовнішніх ключів, а також необхідних індексів для швидкого доступу до даних.

Приклад створення таблиці tracked_routes у SQLite:

```
CREATE TABLE tracked_routes (
 id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
  start_ts INTEGER NOT NULL,
  distance REAL NOT NULL,
  points TEXT NOT NULL
);
```

Серверна база даних (PostgreSQL)

На сервері, реалізованому на базі Django + PostgreSQL, створено спрощену модель для зберігання лише найнеобхідніших даних, пов’язаних із резервним копіюванням:

- users — (синхронізується або створюється автоматично під час запиту);
- backups — таблиця, в якій зберігається інформація про файл резервної копії, дату її створення та власника.

Приклад створення таблиці backups у PostgreSQL:

```
CREATE TABLE backups (
  id SERIAL PRIMARY KEY,
  user_id INTEGER NOT NULL REFERENCES users(id),
  file_path TEXT NOT NULL,
  created_at TIMESTAMP NOT NULL
);
```

Особливості реалізації

Дані синхронізуються з сервером лише за ініціативою користувача.

Усі важливі операції з базою даних реалізовані у вигляді сервісів, які взаємодіють із таблицями через SQL-запити або ORM.

Забезпечено підтримку зовнішніх ключів, унікальності записів та цілісності даних.

Розроблена структура інформаційної бази дозволяє ефективно зберігати та обробляти інформацію як у локальному середовищі мобільного додатку, так і на сервері. Вона забезпечує швидкий доступ до даних, підтримку автономного режиму та надійне резервне копіювання у випадку втрати або перенесення даних на інший пристрій.

3.2 Вибір інструментарію для створення прикладного програмного забезпечення

Для реалізації прикладного програмного забезпечення було обрано набір сучасних інструментів, які забезпечують кросплатформеність, високу продуктивність, зручність розробки та підтримку всіх необхідних функцій для роботи з геолокацією, базами даних і мережею. Система складається з двох частин — мобільного клієнта та серверної частини, для кожної з яких було підібрано відповідний стек технологій.

Мобільний клієнт (Flutter)

Для розробки мобільного додатку обрано Flutter — фреймворк з відкритим кодом від Google, який дозволяє створювати високопродуктивні додатки під Android та iOS з єдиною кодовою базою.

Обрані технології та пакети:

- Мова програмування: Dart
- Геолокація: geolocator, flutter_foreground_task
- Візуалізація маршрутів на карті: flutter_map, latlong2, flutter_earth_globe
- Зберігання даних: sqflite (SQLite) + path_provider
- Управління станом: provider
- Імпорт/експорт: json_serializable, csv, file_picker
- Інтерфейс: Material UI, кастомні компоненти, адаптовані під мобільні пристрої

Цей набір дозволяє реалізувати як реальний трекінг переміщень у фоні, так і зручну візуалізацію та роботу з нотатками.

Серверна частина (Django + PostgreSQL)

Для обробки запитів на резервне копіювання даних, автентифікацію та збереження копій на сервері використовується веб-фреймворк Django з мовою програмування Python.

Обрані технології:

- Фреймворк: Django + Django REST Framework
- СУБД: PostgreSQL
- Автентифікація: JWT (через djangorestframework-simplejwt)
- Зберігання файлів: локально або на S3-сумісному сховищі

Цей стек забезпечує надійність, масштабованість і простоту розгортання серверного API, необхідного лише для резервного копіювання.

Обґрунтування вибору

Обраний інструментарій дозволяє:

- створити сучасний, легкий і швидкий мобільний застосунок з багатим UI;
- працювати з геолокацією у фоновому режимі;
- зберігати дані офлайн і синхронізувати їх із сервером лише за потреби;
- забезпечити розширюваність, безпеку та модульність.

Такий підхід дозволяє задовольнити як технічні, так і функціональні вимоги до системи, забезпечуючи високу якість розробки та зручність використання.

3.3 Алгоритмізація та програмування програмних модулів

На цьому етапі було реалізовано основні програмні модулі, що відповідають за ключові функції застосунку: відстеження переміщень, збереження маршрутів, роботу з нотатками, імпорт/експорт даних, а також створення резервних копій. Для кожного з модулів розроблено відповідну логіку, побудовано алгоритмічні діаграми та виконано програмну реалізацію.

Модуль трекінгу переміщень

Цей модуль відповідає за періодичне зчитування геолокаційних координат у фоновому режимі та формування маршруту у вигляді списку точок з координатами та мітками часу.

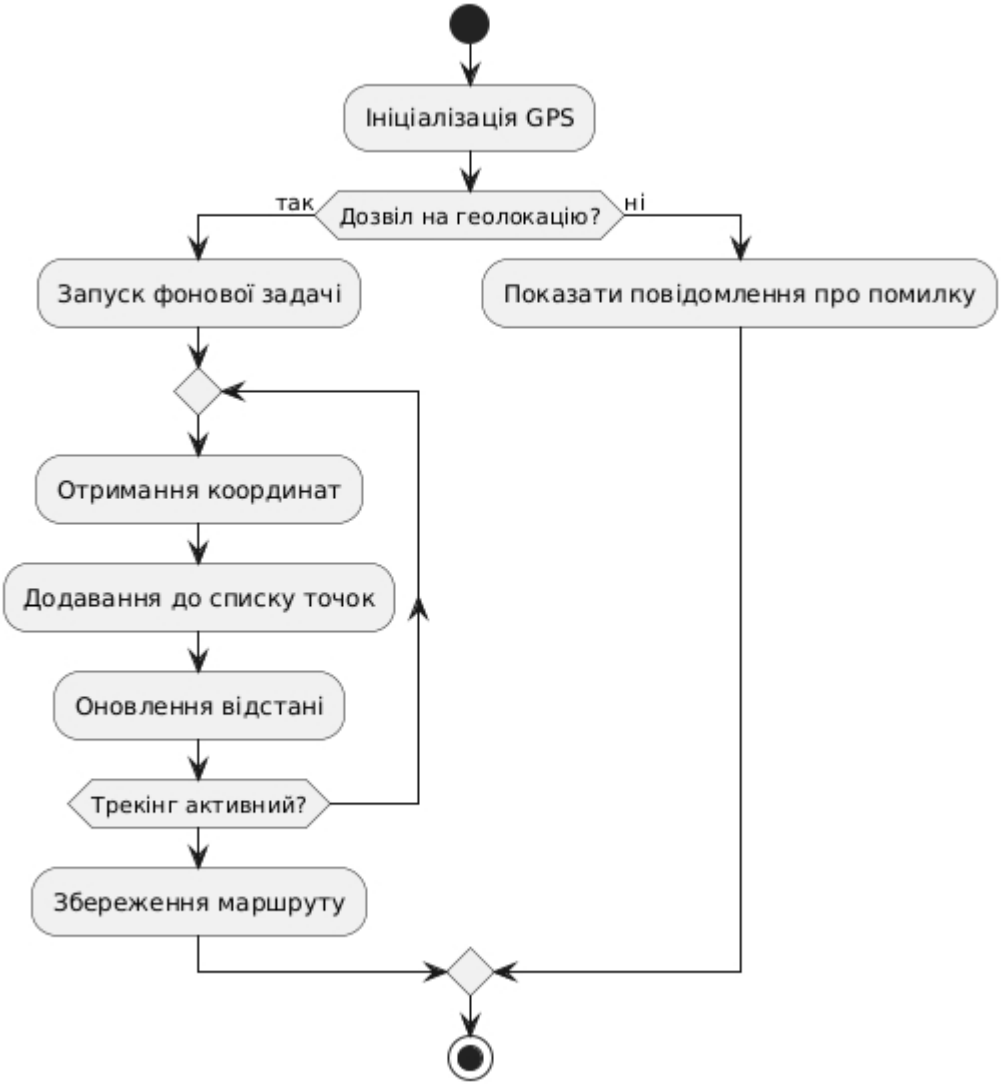


Рис 3.1 – Алгоритм трекінгу переміщень

Модуль створення нотаток

Цей модуль дозволяє користувачу прикріпити текстову нотатку до певної координати або дати маршруту. Нотатка зберігається в таблиці location_notes.

Рис 3.2 - Алгоритм створення нотатки

Модуль резервного копіювання

Цей модуль дозволяє створити файл із копією всіх даних (маршрути, точки, нотатки) та надіслати його на сервер.

Рис. 3.3 - Алгоритм резервного копіювання

Модулі реалізовані у вигляді Dart-класів з чітким розділенням відповідальностей:

- TrackingService — керує GPS-модулем, записом точок та створенням маршрутів;
- NoteService — обробляє створення, редагування та збереження нотаток;
- BackupService — відповідає за експорт, імпорт і взаємодію з сервером;

Використання структурованої архітектури дозволило забезпечити зрозумілу побудову логіки та можливість розширення в майбутньому.

4. РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Тестування є обов’язковим етапом життєвого циклу програмного забезпечення і виконується з метою перевірки працездатності системи, виявлення помилок, а також підтвердження відповідності функціональності встановленим вимогам. У межах проєкту мобільного додатку для приватного відстеження переміщень було проведено кілька типів тестування, орієнтованих на ключові модулі системи.

4.1.1 Функціональне тестування

Виконувалося вручну на фізичних пристроях з операційною системою Android. Метою було перевірити, чи коректно працює кожен модуль системи.

Таблиця 4.1

Таблиця результатів перевірки модулів системи

Тестований модуль	Перевірка	Результат
Трекінг координат	Чи фіксуються точки маршруту під час руху	Успішно
Створення маршруту	Чи зберігається маршрут після завершення	Успішно
Додавання нотатки	Чи додається та зберігається нотатка до локації	Успішно
Відображення на мапі	Чи правильно будується маршрут на карті	Успішно
Експорт / імпорт	Чи зчитуються та відновлюються дані з JSON-файлу	Успішно
Резервне копіювання	Чи створюється файл та надсилається на сервер	Успішно

4.1.2 Інтерфейсне тестування (UI)

На рис. 4.1 зображено форму авторизації.

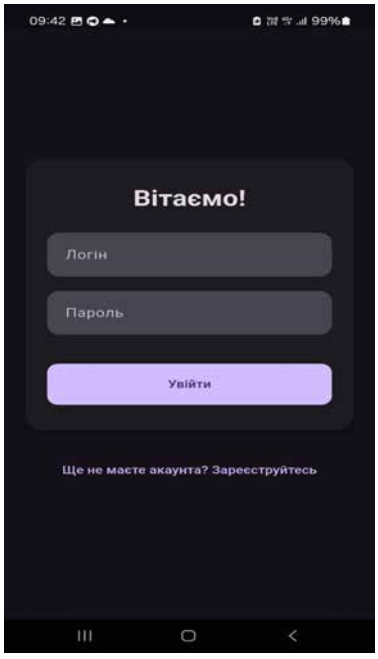


Рис. 4.1 - Форма авторизації

При відсутності облікового запису користувач може зареєструватися (рис. 4.2).

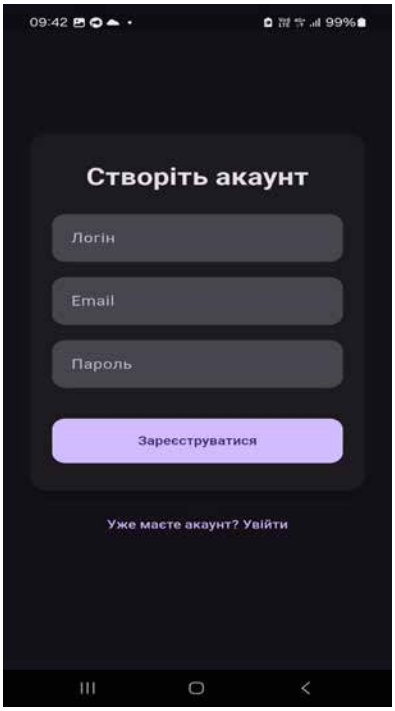


Рис. 4.2 - Форма реєстрації

Після введеня даних та натисканні кнопки “Увійти” відкривається сторінка трекінгу (рис.4.3). Яка містить мапу та кнопки управління, також можна виконуват дії кліками, скролами та іншими діями.Кнопки управління: перемикання типів карт, центрування, зміна масштабу.

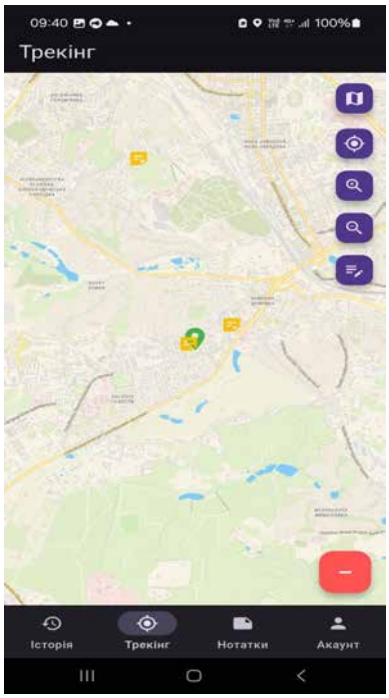


Рис. 4.3 - Сторінка трекінгу

Клікнувши на кнопку із записником користувач запускає режим нотаток, клікнувши на бажану ділянку на мапі Він зможе прив’язати до неї

нотатку (рис. 4.4), яка в свою чергу буде відображатися на мапі оранжевою іконкою.

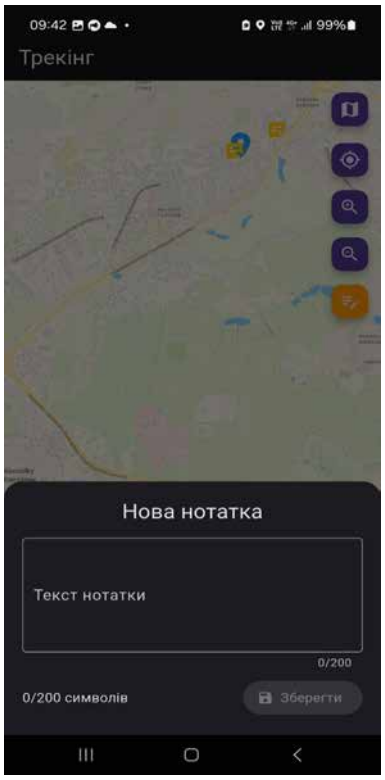


Рис. 4.4 - Сторінка добавлення нотатки

Також перелік нотаток з можливістю їх перегляду, редагування та видалення (рис.4.5) можна знайти вибравки в меню пункт “Нотатки”.

Рис. 4.5 - Сторінка добавлення нотатки

Також користувачу доступна функція перегляду історії переміщень по датах (рис. 4.6).

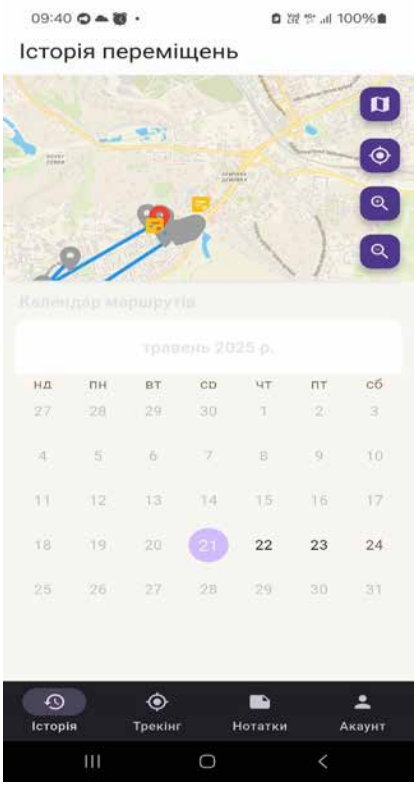


Рис. 4.6 - Сторінка історії переміщень

Користувач має декілька варіантів збереження даних своїх переміщень, вони представленні в налаштуваннях додатку (рис.4.7). А саме імпорт/експорт геоданих та резервне збереження на сервері. В налаштуваннях користувач також може оновити/відредагувати свої дані профілю, змінити кольорову тему.

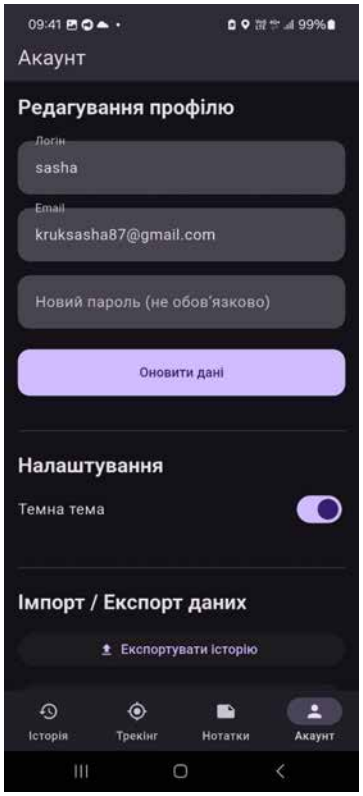


Рис. 4.7 - Сторінка налаштувань

У результаті тестування встановлено, що система працює стабільно, всі модулі функціонують відповідно до технічних вимог. Виявлені незначні зауваження (затримки при великій кількості точок маршруту).

4.2 Вимоги до апаратного та програмного забезпечення

Розроблена система резервного копіювання геоданих передбачає взаємодію клієнтської частини, реалізованої як мобільний додаток на Flutter, та серверної частини на основі фреймворку Django. Система повинна функціонувати стабільно на типових пристроях, які відповідають мінімальним технічним хар.

4.2.1 Апаратні вимоги

Клієнтська частина (мобільний пристрій):

Процесор: ARMv7 або новий, рекомендовано ARM64

Оперативна пам'ять: не менше 2 ГБ

Вільне місце для зберігання: від 100 МБ

Операційна система: Android 8.0 або вище, iOS 12.0 або вище

Інтернет-з'єднання: стабільне з'єднання з мережею для надсилання/отримання резервних копій

Серверна частина (комп'ютер/сервер):

Процесор: двоядерний (або більше) з тактовою частотою від 2.0 ГГц

Оперативна пам'ять: від 1 ГБ

Вільне місце на диску: від 500 МБ для зберігання резервних копій

Сіткове з'єднання: постійне підключення до мережі з пропускною здатністю не менше 10 Мбіт/с

4.2.2 Програмні вимоги

Клієнтська частина (Flutter):

Flutter SDK: версія 3.10 або новіша

Dart SDK: версія 3.2 або новіша

Android SDK або Xcode: для компіляції мобільного додатку

Платформа розробки: Windows, macOS або Linux з підтримкою Flutter

Серверна частина (Django):

Операційна система: Ubuntu 20.04 або новіша, Windows 10/11, macOS

Python: версія 3.10 або новіша

Django: версія 4.2 або новіша

Django REST Framework: для створення API

SQLite або PostgreSQL: для зберігання даних

Інструменти керування середовищем: pip, venv (або Anaconda)

Вебсервер для розгортання: стандартний runserverабо Apache/nginx з WSGI (у разі розгортання на хостингу)

Система може бути розгорнута на локальному сервері або у хмарній середовищі (наприклад, на VPS або у внутрішній мережі підприємства).

4.3 Склад інсталяційного пакету

Інсталяційний пакет призначений для надання повного комплекту засобів, необхідних для запуску клієнтської та серверної частин системи резервного копіювання геоданих.

Клієнтська частина (мобільний додаток):

- Файл location_tracker.apk: інсталяційний файл для Android-пристроїв
- Файл assets/backup_instructions.txt: текстова інструкція з використання функцій резервного копіювання
- Файл README.md: опис функціональних можливостей застосування, контактні дані розробника, приклади використання
- Серверна частина (бекап-сервер на Django):

Проект Django з такими каталогами:

- backup/— додаток для обробки API запитів на резервне копіювання
- users/- модуль реєстрації та ідентифікації користувачів (якщо передбачено)
- media/backups/- директорія для збереження резервних JSON-файлів
- Файл manage.py: скрипт управління проектом Django
- Файл requirements.txt: список потрібних Python-залежностей
- Файл README.md: інструкція з налаштування середовища, запуску сервера, форматів запитів до API
- Файл .env.example: шаблон змінних середовищ (наприклад, секретні ключі, конфігурації бази даних)

ВИСНОВКИ

У ході виконання дипломної роботи розроблено повнофункціональну програмну систему для приватного відстеження переміщень користувача, яка включає мобільний додаток, реалізований засобами Flutter та серверну частину, побудовану на основі Django. Основна мета, що полягала у створенні ефективного засобу для фіксації маршрутів, прив'язки до них текстових нотаток та можливості збереження/відновлення даних через резервні копії, була повністю досягнута.

Розроблена система вирішує актуальне завдання — забезпечення збереження та приватного аналізу особистих геолокаційних даних без залучення сторонніх сервісів. Завдяки кроссплатформенним можливостям Flutter, приложение є доступним для пристроїв з Android та iOS, а серверна частина легко розгортається на будь-якій середовищі, що підтримує Python.

У процесі роботи досягнуто наступних результатів:

проведено аналіз аналогічних рішень та сформульовано вимоги до майбутньої системи з урахуванням потреб користувача в зручному, наочному та безпечному інструменті відстеження переміщень;

спроектовано архітектуру клієнт-серверного рішення, що включає локальне зберігання треків, додавання нотаток до локацій, візуалізацію маршрутів на карті, резервне копіювання та відновлення через веб-сервіс;

реалізовано сучасний, інтуїтивно зрозумілий інтерфейс користувача з використанням гнучкої кольорової схеми у світлих тонах, підтримкою інтерактивних карток (у тому числі 3D-глобуса) та календаря маршрутів;

створено ефективний механізм локального зберігання даних із використанням SQLite, що дозволяє працювати з додатком без постійного інтернет-з'єднання;

розроблено серверний API, який дозволяє передавати дані заметок у форматі JSON, зберігати резервні копії на сервері та завантажувати їх на клієнт за потреби;

проведено тестування всіх компонентів системи, перевірено коректність збереження та відновлення даних, стабільність роботи застосування, ефективність взаємодії з сервером.

Особливістю реалізованої системи є підтримка прив'язки текстових нотаток до довільних географічних точок незалежно від активного маршруту, а також їх логічна групація за датами. Це надає користувачеві гнучкість у фіксації подій, спогадів чи службової інформації, що пов'язана з конкретними місцями.

Крім цього, важливо відзначити, що система створена з урахуванням принципів конфіденційності : жодні дані не передаються без згоди користувача, резервні копії зберігаються в контроленій среді, доступ до якої можна омежити через авторизацію.

- автоматичне резервне копіювання за розкладом;
- шифрування даних;
- синхронізація між кількома пристроями;
- статистичний аналіз переміщень;
- візуалізація приміток у вигляді хронологічної стрічки подій.

Таким чином, дипломна робота реалізувала повний цикл створення сучасного клієнт-серверного застосування — від постановки завдання та проектування до реалізації, тестування та документування. Отриманий результат має як практичну цінність для кінцевих користувачів, так і потенціал для подальших досліджень та комерційного використання.

Розроблену систему можна рекомендувати як основу для побудови персоналізованих цифрових щоденників, сервісів фіксації маршрутів, а також мобільних додатків у сфері логістики, охорони здоров'я, польових робіт чи туризму.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Official documentation of Flutter [Електронний ресурс]. - Режим доступу: https://flutter.dev
2. Dart Programming Language Documentation [Електронний ресурс]. - Режим доступу: https://dart.dev
3. Django Project Documentation [Електронний ресурс]. - Режим доступу: https://docs.djangoproject.com/
4. Django REST framework – Web APIs for Django [Електронний ресурс]. - Режим доступу: https://www.django-rest-framework.org/
5. SQLite Documentation [Електронний ресурс]. - Режим доступу: https://www.sqlite.org/docs.html
6. MapTiler API Documentation [Електронний ресурс]. - Режим доступу: https://docs.maptiler.com/cloud/api/
7. Сидоренко В.В. Архітектура програмного забезпечення. - Харків: ХНУРЕ, 2020. - 274 с. https://ela.kpi.ua/server/api/core/bitstreams/6179ebd1-56db-429a-b930-211917fdff7e/content
8. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. https://www.iso.org/standard/35733.

ДОДАТОК А

SQL ЗАПИТИ

Сторінок – 2

2025

```
CREATE TABLE users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT NOT NULL,
    email TEXT NOT NULL,
    password_hash TEXT NOT NULL
);

CREATE TABLE tracked_routes (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER NOT NULL,
    start_ts INTEGER NOT NULL,
    distance REAL NOT NULL,
    points TEXT NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id)
);

CREATE TABLE location_notes (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER NOT NULL,
    latitude REAL NOT NULL,
    longitude REAL NOT NULL,
    text TEXT NOT NULL,
    timestamp INTEGER NOT NULL,
    route_date TEXT,
    FOREIGN KEY (user_id) REFERENCES users(id)
);

CREATE TABLE locations (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER NOT NULL,
    latitude REAL NOT NULL,
    longitude REAL NOT NULL,
    timestamp INTEGER NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id)
);
CREATE TABLE backups (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER NOT NULL,
    file_path TEXT NOT NULL,
```

```
    created_at INTEGER NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

ДОДАТОК Б

КОД DATABASE_HELPER

Сторінок – 3

2025

```
class DatabaseHelper {
  static final DatabaseHelper instance = DatabaseHelper._internal();
  static Database? _database;

  // Приватний конструктор (singleton)
  DatabaseHelper._internal();

  // Версія БД (змінюй, якщо додаєш таблиці або поля)
  static const int _dbVersion = 3;

  Future<Database> get database async {
    if (_database != null) return _database!;
    _database = await _initDatabase();
    return _database!;
  }

  // Ініціалізація БД
  Future<Database> _initDatabase() async {
    final dbPath = await getDatabasesPath();
    final path = join(dbPath, 'tracking_app.db');

    return await openDatabase(
      path,
      version: _dbVersion,
      onConfigure: (db) async {
        await db.execute('PRAGMA foreign_keys = ON');
      },
      onCreate: _onCreate,
      onUpgrade: _onUpgrade,
    );
  }

  // Створення таблиць при першому запуску
  Future<void> _onCreate(Database db, int version) async {
    await db.execute('''
      CREATE TABLE users (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        username TEXT NOT NULL UNIQUE,
        email TEXT NOT NULL UNIQUE,
```

```
      password_hash TEXT NOT NULL
    )
  ''');

  await db.execute('''
    CREATE TABLE locations (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      user_id INTEGER NOT NULL,
      latitude REAL NOT NULL,
      longitude REAL NOT NULL,
      timestamp INTEGER NOT NULL,
      FOREIGN KEY(user_id) REFERENCES users(id) ON DELETE CASCADE
    )
  ''');

  await db.execute('''
    CREATE TABLE location_notes (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      user_id INTEGER NOT NULL,
      latitude REAL NOT NULL,
      longitude REAL NOT NULL,
      text TEXT NOT NULL,
      timestamp INTEGER NOT NULL
    )
  ''');
}

// Оновлення бази даних (наприклад, при зміні версії)
Future<void> _onUpgrade(Database db, int oldVersion, int newVersion) async {
  if (oldVersion < 2) {
    await db.execute('''
      CREATE TABLE location_notes (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        user_id INTEGER NOT NULL,
        latitude REAL NOT NULL,
        longitude REAL NOT NULL,
        text TEXT NOT NULL,
        timestamp INTEGER NOT NULL
      )
```

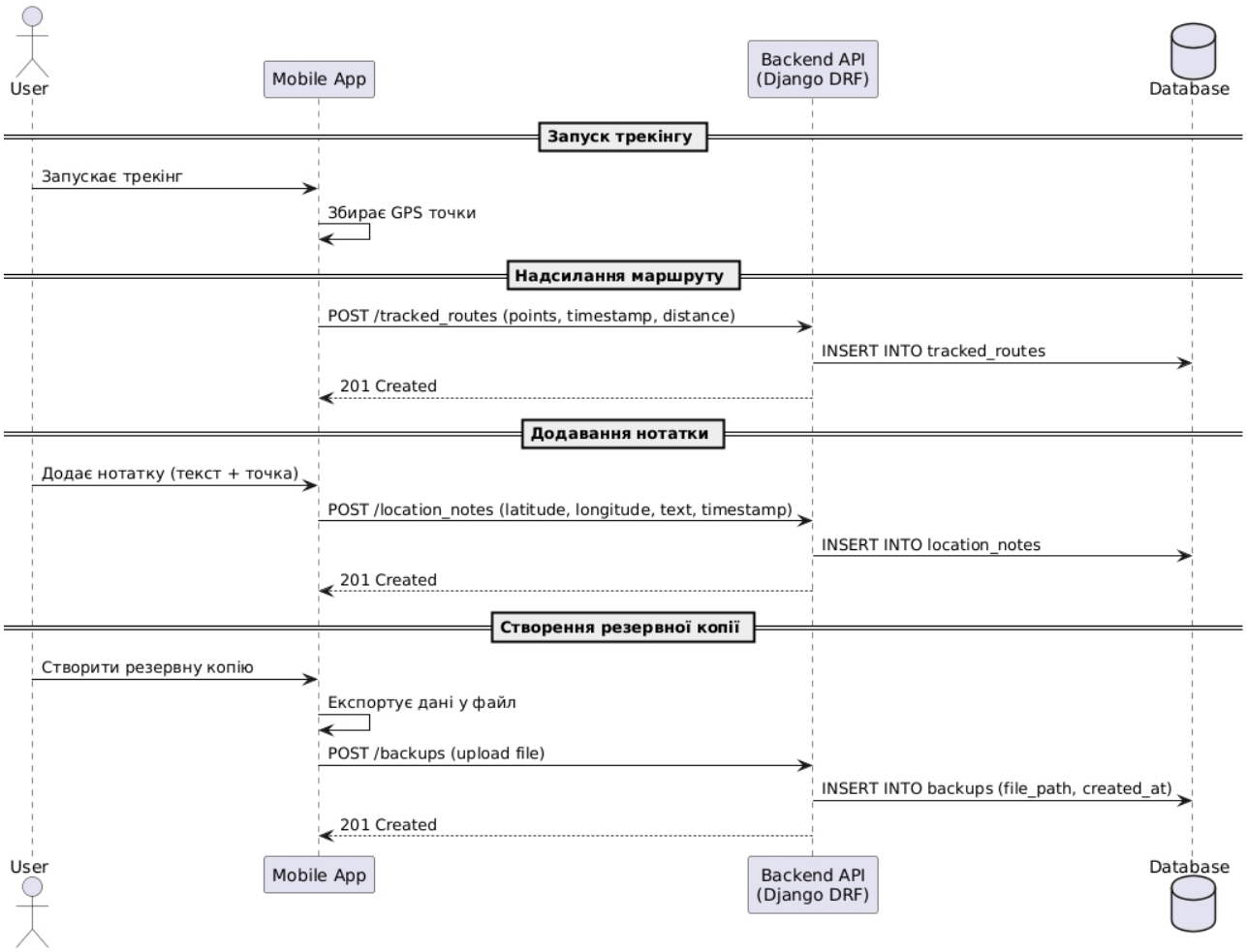
```
      ''');
    }
    if (oldVersion < 3) {
      await db.execute("ALTER TABLE location_notes ADD COLUMN route_date
TEXT");
    }

  }
```

ДОДАТОК Б

ДІАГРАМА ПОСЛІДОВНОСТІ

Сторінок – 1

2025


User
Mobile App
Backend API
(Django DRF)
Database
Запуск трекінгу
Запускає трекінг
Збирає GPS точки
Надсилання маршруту
POST /tracked_routes (points, timestamp, distance)
INSERT INTO tracked_routes
201 Created
Додавання нотатки
Додає нотатку (текст + точка)
POST /location_notes (latitude, longitude, text, timestamp)
INSERT INTO location_notes
201 Created
Створення резервної копії
Створити резервну копію
Експортує дані у файл
POST /backups (upload file)
INSERT INTO backups (file_path, created_at)
201 Created
User
Mobile App
Backend API
(Django DRF)
Database