КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ

Кафедра комп'ютерної інженерії

До захисту допущено: «На правах рукопису»
Завідувач кафедри _____ Юрій Бойко
« _ » _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему:

«ЗАСТОСУНОК ПО ЗБОРУ І СИНХРОНІЗАЦІЇ ІНФОРМАЦІЇ РОЗВІДУВАЛЬНОГО ХАРАКТЕРУ»

Виконав:
студент 4-го курсу бакалаврату
денної форми навчання
спеціальності 123 Комп'ютерна інженерія
ОНП «_____»
Євтушенко Максим Сергійович _____

Науковий керівник:
кандидат фізико-математичних наук, доцент
Сугакова Олена Володимирівна _____
асистент
Погорєлов Р. В. _____

Рецензент:
_____

Засвідчую, що у цій бакалаврській роботі
немає запозичень з праць інших авторів без
відповідних посилань
Студент _____

Робота допущена до захисту в ЕК рішенням кафедри _____
від «__»_____ 2023 р., протокол № __.

Завідувач кафедри _____,
кандидат фізико-математичних наук, доцент
Бойко Юрій Володимирович

(підпис)

Київ – 2023

РЕФЕРАТ

Дипломна робота за об’ємом складає 34 сторінки, містить 3 рисунки, 1 додаток, використано 5 інформаційних джерел.

Розглянуто технології, що були застосовані при написанні додатку: GPS, метод тріангуляції, сервіс Google Maps, фреймворк Flutter та застосування мови Dart.

Було створено мобільний застосунок для запису, збереження та відтворення безпечних маршрутів пересування потенційно замінованими територіями.

Ключові слова: DART, FLUTTER, GPS, GOOGLE MAPS

ЗМІСТ

1. Перелік умовних позначень, скорочень і термінів

ОС – операційна система;

SDK (Software Development Kit) – набір засобів розробки, утиліт і документації, що дають можливість створювати програми за визначеною технологією або для певної платформи;

API (Application Programming Interface) – набір чітко визначених методів взаємодії між програмами;

UI (User Interface) – інтерфейс користувача;

Фреймворк – програмне середовище, створене для спрощення та прискорення розробки програмного забезпечення;

Віджет – компонент графічного інтерфейсу;

Білінговий акаунт – обліковий запис, що використовується для здійснення фінансових операцій;

Нативний код – код, написаний мовою програмування, специфічною для певної платформи чи середовища.

2. ВСТУП

Від початку російсько-української війни значна частина територій України стала небезпечною для пересування через ймовірні мінування чи наявність вибухонебезпечних об’єктів.

Міни, розтяжки, нерозірвані боєприпаси – масове явище біля лінії фронту та на звільнених територіях і становить велику загрозу для життя військових та цивільних. Розмінування займає роки, а поблизу лінії зіткнення – взагалі неможливе.

Для даної проблеми необхідне рішення, що зможе швидко убезпечити пересування людей потенційно замінованими територіями. Таким рішенням може бути мобільний застосунок, що матиме доступ до бази перевірених маршрутів, складених саперними підрозділами, та відображатиме їх на карті. Це дозволить безпечно пересуватись між населеними пунктами, оборонними рубежами чи вивозити поранених з лінії фронту, без необхідності шукати інформацію про шлях чи провідника.

Актуальність роботи полягає в необхідності простого і швидкого способу вирішення задачі безпечного пересування в місцевостях, де велись бойові дії та безпосередньо біля лінії фронту.

Метою випускної кваліфікаційної роботи є створення застосунку для запису, збереження та відтворення безпечних маршрутів пересування потенційно замінованими територіями.

3. Постановка задачі

В першу чергу, для створення даного застосунку, необхідно визначитись з його ключовими завданнями:

- визначати геопозицію пристрою;
- давати змогу користувачу зберігати безпечний маршрут;
- відображати карту з геолокацією користувача та обраним маршрутом;
- збережені маршрути мають бути доступні всім користувачам;
- додаток має бути доступним для використання на різних пристроях та операційних системах.

На основі ключових завдань можна визначитись з основними технологіями:

- для визначення геолокації необхідне використання технології GPS;
- для відображення карти слід використати сервіс Google Maps, що доступний на будь-якому мобільному пристрої;
- для збереження маршрутів та їх доступності будь-якому користувачу, необхідно використати хмарну базу даних;
- для забезпечення максимальної доступності застосунку слід скористатись кросплатформеним фреймворком.

4. Огляд технологій

4.1. GPS

Глобальна система позиціонування (GPS) — це навігаційна система, заснована на супутниковій технології. Його основна техніка передбачає вимірювання відстаней між приймачем і декількома одночасно спостережуваними супутниками, а положення супутників прогнозуються та транслюються користувачеві разом із сигналом GPS. За кількома відомими положеннями (супутників) і виміряними відстанями між приймачем і супутниками можна визначити положення приймача. Зміна позиції, яку також можна визначити, є

швидкістю приймача. Найважливішими застосуваннями GPS є позиціонування та навігація.

4.1.1 Геоцентрична система координат закріплена на Землі

Система координат, орієнтована на Землю (англ. Earth-Centred Earth-Fixed – ECEF) корисна для визначення позиції об'єкта на земній поверхні. ECEF є правою декартовою системою координат (x, y, z). Її початок співпадає з центром мас Землі, а вісь z збігається з середньою віссю обертання Землі, вісь x вказує на Гринвіцький меридіан, а вісь y спрямована на завершення правої системи.

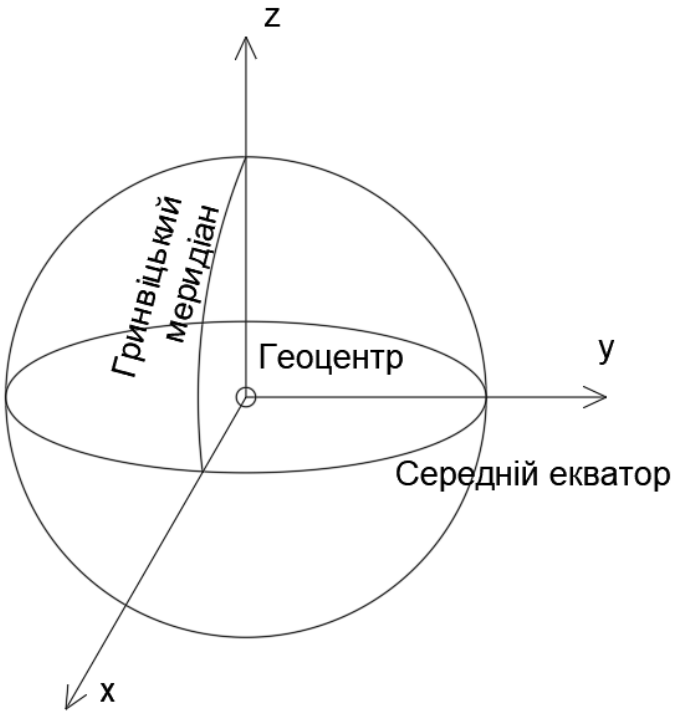

Рис.1 Геоцентрична система координат

Іншими словами, вісь z вказує на середній полюс обертання Землі, що визнано міжнародною конвенцією та названо міжнародним умовним початком (англ. Conventional International Origin – CIO). Площину xy називають середньою екваторіальною площиною, а площину xz – середнім нульовим меридіаном.

Геоцентрична система може бути представлена і в сферичній системі координат (r, ϕ, λ), де r – радіус точки (x, y, z). ϕ та λ – геоцентричні широта і довгота. λ відраховується на схід від нульового меридіана. Зв'язок між декартовою та сферичною системами:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r\ \cos\phi\ \cos\lambda \\ r\ \cos\phi\ \sin\lambda \\ r\ \sin\phi \end{pmatrix},\ \text{або}\ \begin{cases} r = \sqrt{x^2 + y^2 + z^2} \\ \tan\lambda = y/x \\ \tan\phi = z/\sqrt{x^2 + y^2} \end{cases}.$$

Еліпсоїдальна або геодезична система координат (φ, λ, h) також може бути визначена на основі координат геоцентричної, проте необхідні два додаткові параметри – велика (a) та мала (b) піввісі або велика піввісь (a) та сплющення $(f = \frac{a-b}{a})$. φ, λ та h – відповідно геодезичні довгота, широта та висота. Зв'язок між декартовою та геодезичною системами:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} (N+h)\ \cos\varphi\ \cos\lambda \\ (N+h)\ \cos\varphi\ \sin\lambda \\ (N(1-e^2)+h)\ \sin\varphi \end{pmatrix},\ \text{або}\ \begin{cases} \tan\varphi = \frac{z}{\sqrt{x^2+y^2}}\left(1 - e^2\frac{N}{N+h}\right)^{-1} \\ \tan\lambda = y/x \\ h = \frac{\sqrt{x^2+y^2}}{\cos\varphi} - N \end{cases},$$

де $N = \frac{a}{\sqrt{1-e^2\sin^2\varphi}}$, $e = \frac{\sqrt{a^2-b^2}}{a}$.

N – радіус кривизни в основній вертикалі, e – перший ексцентриситет.

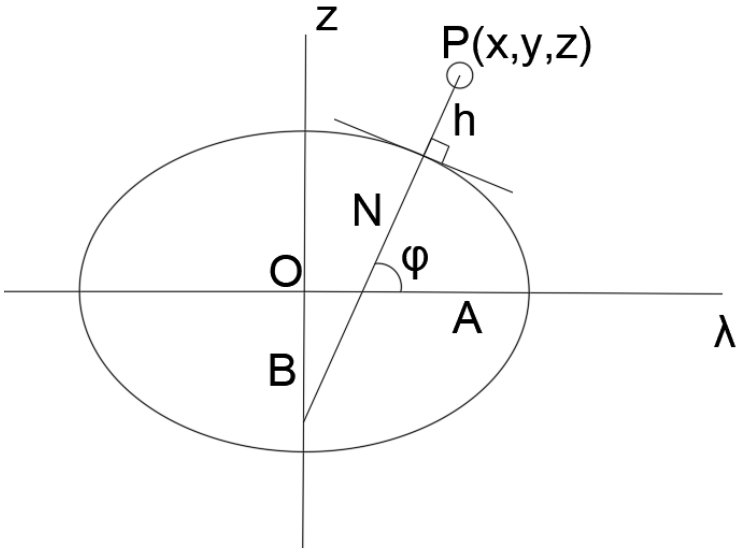

Рис.2 Геометричний зміст радіуса кривизни в основній вертикалі

Співвідношення між геоцентричною та геодезичною широтами ϕ та φ визначається за формулою: $\tan\phi = (1 - e^2\frac{N}{N+h})\tan\varphi$. [1]

4.1.2. Тріангуляція GPS

Тріангуляція є ключовим принципом, на якому базується технологія GPS для визначення місцезнаходження. Вона використовується для вимірювання відстаней між GPS-приймачем і супутниками для подальшого обчислення координат.

Процес тріангуляції полягає в одночасному отриманні сигналів від трьох чи більше супутників. Від кожного супутника надходить сигнал з точним часом відправлення. Приймач отримує ці сигнали та вимірює час, за який вони надходять з супутника до приймача.

Затримка сигналу залежить від швидкості поширення світла і може бути перетворена на відстань між супутником і приймачем, використовуючи відому швидкість поширення світла. Це дозволяє визначити сферу, в якій знаходиться GPS-приймач.

Приймач отримує сигнали від трьох або більше супутників та визначає відстані до кожного супутника. Це дозволяє визначити три сфери, в яких розташовані супутники.

Точне місцезнаходження приймача може бути визначене там, де ці три сфери перетинаються, утворюючи точку тріангуляції. Це можливо завдяки вимірюванню відстаней до трьох супутників та факту, що приймач знаходиться на поверхні Землі. Одержані від супутників координати відображають географічне місцезнаходження приймача на земній поверхні. Цей процес відбувається швидко і дозволяє отримати точні координати.

Для точнішого визначення місцезнаходження, GPS-приймач може використовувати виміри від більшої кількості супутників. Чим більше супутників використовується приймачем, тим точнішим буде місцезнаходження.

Однак, існує декілька недоліків тріангуляції в системі GPS, що можуть призводити до неточностей. Наприклад, вплив атмосферних умов, таких як розсіювання сигналу, може впливати на точність вимірів. Крім того, можуть виникати проблеми через ефект множинного відбиття сигналу, коли сигнал від

супутника досягає приймача різними шляхами. Для усунення даних проблем, сучасні GPS-приймачі використовують методи корекції. Один з них – корекція сигналу за допомогою системи диференціальної корекції, згідно якої дані від керованої станції коригуються і передаються приймачу для поліпшення точності. Інші методи включають використання додаткових супутників з інших систем навігації, що покращують доступність і точність навігації.

Загалом, тріангуляція є головним принципом технології GPS. Вона дозволяє визначати координати приймача з високою точністю, забезпечуючи широкий спектр застосувань в різних галузях, від навігації та військових операцій до спорту і екологічних досліджень.

4.2. Google Maps

Google Maps – це онлайн-карта та глобальна географічна інформаційна система, що дозволяє користувачам переглядати карти, знаходити місця та прокладати маршрути з однієї точки до іншої. Сервіс з'явився на ринку у 2005 році і став одним з найпопулярніших сервісів компанії Google.

Однією з ключових технічних характеристик Google Maps є геодезична точність. Платформа використовує глобальну систему позиціонування (GPS), яка забезпечує точність визначення місцезнаходження до кількох метрів. Окрім GPS, Google Maps також використовує інші методи позиціонування, такі як Wi-Fi, Bluetooth та мобільна мережа, що дозволяє забезпечити ще більшу точність.

Ще однією з важливих характеристик Google Maps є швидкість та ефективність обробки даних. Платформа працює в режимі реального часу та швидко відображає зміни на мапі, такі як зміна маршруту або додавання нових місць. Це можливо завдяки використанню технології веб-програмування та баз даних, що дозволяють зберігати великі обсяги інформації та оперативно обробляти запити користувачів.

Окремим важливим елементом Google Maps є база даних, що містить велику кількість географічних даних, таких як дороги, будівлі, річки, озера та інші

об'єкти. Сервісом використовуються складні алгоритми для обробки цих даних та відображення їх на мапі. Крім того, Google Maps має ряд інших технічних характеристик, які забезпечують зручність та функціональність сервісу. Наприклад, мапа має різні режими відображення, які дозволяють користувачам переглядати місцезнаходження з різних ракурсів, таких як супутниковий знімок або карта, залежно від їх потреб. Також, Google Maps має можливість додавання та редагування інформації користувачами, що дозволяє забезпечити більш повну та актуальну інформацію про місцезнаходження.

Google Maps є однією з найбільш технологічно розвинених та ефективних онлайн-картографічних платформ. Її технічні характеристики дозволяють забезпечити високу точність, швидкість та зручність використання для користувачів. Можливість інтеграції з іншими сервісами та додатками робить Google Maps ще більш універсальним та зручним для використання в різних сферах життя.

Важливою особливістю Google Maps є мобільний застосунок, що надає користувачам доступ до географічних даних на мобільних пристроях. Це дозволяє отримувати інформацію про геолокацію та маршрути навіть під час руху, що робить сервіс незамінним інструментом для подорожей та навігації.

Google Maps знаходить застосування в різних наукових дослідженнях, наприклад в географічних, геологічних та кліматичних дослідженнях чи для демографічних аналізів. Завдяки доступності, широкому спектру використання та багатофункціональності, сервіс став не тільки популярним і корисним серед користувачів, але і цінним інструментом для наукових досліджень.

4.3. Мова Dart

Dart – це об'єктно-орієнтована мова програмування, розроблена компанією Google, з метою створення ефективної, швидкої, безпечної та простої мови програмування, яка б добре підходила для розробки веб-додатків та мобільних додатків. Dart використовується як мова програмування для

фреймворка Flutter, що дозволяє розробляти кросплатформені мобільні додатки для Android та iOS. Dart надає мову та середовище виконання, на яких працюють програми Flutter, але Dart також підтримує багато основних завдань розробника, як от форматування, аналіз і тестування коду. [3]

Ключові позиції мови Dart:

- повністю об’єктно-орієнтована мова програмування;
- містить автоматичний сміттєзбірник, тому відсутні зайві витрати пам’яті та не потрібно про них турбуватись;
- підтримує загальні мовні функції, такі як інтерфейси, міксини, абстрактні класи, узагальнені генерики та статична типізація;
- правильна система типізації (та що відкидає неправильні програми);
- ізоляція для паралелізму;
- може використовувати попередню компіляцію в нативний код, для досягнення максимальної продуктивності;
- підтримує велику базу пакетів, що надають відповідну функціональність, додатково до самої мови;
- підтримується багатьма популярними інструментами розробника;
- програми можна скомпілювати у знімки, які містять увесь програмний код і залежності, попередньо проаналізовані та готові до виконання під час запуску, а знімки також широко використовуються при використанні ізолятів.

4.4. Flutter

Flutter – фреймворк для розробки мобільних (і не тільки) застосунків, розроблений компанією Google. Головною відмінністю даного фреймворку від подібних є самостійне відтворення елементів інтерфейсу користувача, а не використання нативних засобів платформи. Наприклад, при запиті на відображення кнопки, Flutter сам візуалізує її, а не надсилає запит про це до ОС.

Концептуально Flutter складається з чотирьох частин:

- Dart – мова програмування;
- кодова база переважно C++ і використовується графічний движок Skia, що є відкритою графічною бібліотекою, написаною мовою C++;
- базова бібліотека, що є інтерфейсом, над SDK платформ;
- віджети. [2]

Особливістю побудови інтерфейсу користувача у Flutter є віджети, точніше те, що все є віджетом. Віджетами є не тільки очевидні кнопки чи зображення на екрані, а ще й навіть відступ навколо фото чи тема, яку використовує програма. Віджети також є частиною програмного коду. Враховуючи, що все є віджетом, то логічним наслідком є те, що код у Flutter значною мірою являє собою ієрархію віджетів, що має назву «дерево віджетів».

Усі віджети є класами Dart і мають лише одну конкретну вимогу: забезпечувати метод build. А цей метод повертає інші віджети, лише віджети найнижчого рівня повертають примітивні типи даних. Крім цієї вимоги, на рівні коду будь-який віджет – це звичайний клас мовою Dart (який принципово не відрізняється від класів у інших мовах програмування).

Віджет у Flutter розширює один зі стандартних класів, що надає сам Flutter. Розширений клас визначає яким буде віджет на фундаментальному рівні. Є 2 головні класи: StatelessWidget та StatefulWidget.

StatelessWidget або віджет без стану, він ніколи не змінюється. Наприклад, іконки, невеликі зображення, текст тощо.

StatefulWidget містить у собі поняття стану, тобто певним змінюється, після взаємодії з користувачем. Наприклад, чекбокс чи текстове поле. При написанні такого віджету необхідно створювати два класи: клас самого віджету з підтримкою стану та клас власне стану.

Головна різниця віджетів «зі станом» та «без стану» полягає в тому, що зміну першого Flutter автоматично обробляє, а другого – ні. Таким чином, при

зміні стану віджету «зі станом» він повторно відтворюється, коли це потрібно. Цей процес керується ядром Flutter і не потребує програмної реалізації в коді програми.

Підсумовуючи, можна виділити дві речі. По-перше, Flutter UI будується шляхом створення віджетів, що призводить до створення «дерева віджетів». У той час, код самих віджетів є об’єктно-орієнтованим, побудова інтерфейсу користувача – це композиційна парадигма. Більшість віджетів Flutter самі по собі є досить простими і їх композиція дозволяє створювати надійний інтерфейс.

По-друге, інтерфейс користувача будується в програмному коді. Тобто не потрібно використовувати окрему мову для розмітки інтерфейсу, на відміну від веб-розробки. Перевагою Flutter, перед багатьма альтернативами, є необхідність вивчення і використання лише однієї мови. [2]

5. Проектування і реалізація

5.1. Google Maps Platform

Для застосунку використано Platform Google Maps (набір API та SDK), яка дозволяє вбудовувати Google Maps у мобільні додатки та веб-сторінки або отримувати дані з карт.

Продукти платформи захищені від несанкціонованого використання шляхом обмеження викликів API лише тими, які надають належні облікові дані для автентифікації. Ці облікові дані мають форму ключа API – унікального буквено-цифрового рядка, який пов'язує білінговий акаунт Google з проектом і конкретним API або SDK. [4]

Для отримання доступу до Google Maps та підключення їх до застосунку було виконано 3 кроки:

1) створено обліковий запис Google Cloud та новий проект;
2) згенеровано ключ API Google Maps;

3) додано Google Maps до застосунку, імпортувавши відповідний SDK до програми та використовуючи отриманий раніше API Google Maps.

5.2. База даних

В якості бази даних для застосунку було використано Cloud Firestore. Це гнучка, масштабована база даних для мобільних, веб- та серверних розробок на основі Firebase та Google Cloud. Вона підтримує синхронізацію даних між клієнтськими додатками за допомогою слухачів у режимі реального часу та пропонує офлайн-підтримку для мобільних і веб-додатків, щоб можна було створювати адаптивні додатки, які працюють незалежно від затримок у мережі або підключення до Інтернету. Cloud Firestore також пропонує бездоганну інтеграцію з іншими продуктами Firebase та Google Cloud, включаючи Cloud Functions. [5]

Для підключення Cloud Firestore до проекту:

1) створено новий проект у Firebase Console;
2) в меню проекту обрано Firestore Database та дотримуючись інструкції підключено до проекту;
3) надано доступ застосунку до Firestore, додавши до проекту відповідний JSON-файл з сертифікатом з Firebase Console та програмно викликавши необхідні методи.

5.3. Інтерфейс користувача

Користувацький інтерфейс реалізовано мовою Dart. Інтерфейс складається з трьох основних частин:

1. Навігаційна панель:
   a. заголовок – рядок для введення імені користувача;
   b. пункти “Map” та “Routes”, що при натисканні на них відкривають відповідні вікна.

2. Вікно “Map” – містить інтерфейс Google Maps, з відображенням геолокації пристрою та відповідною кнопкою, що переміщує камеру до цього місця. Крім того, в правому нижньому кутку знаходиться кнопка початку/завершення запису маршруту:
   a. початок запису – відкривається додаткове вікно для введення назви маршруту, після натискання кнопки “Start” додаткове вікно закривається та починається запис маршруту,
   b. завершення запису – при повторному натисканні на кнопку запис припиняється та відправляється до бази даних.
3. Вікно “Routes” – містить список всіх наявних у базі даних маршрутів. Кожен пункт списку складається з назви, імені користувача що його додав та кнопки відтворення. При натисканні кнопки відкривається нове вікно Google Maps, з відтвореним маршрутом у вигляді лінії.

5.4. Використані пакети

Flutter підтримує використання пакетів, наданих іншими розробниками до екосистем Flutter та Dart. Це дозволяє швидко створити додаток без необхідності розробляти все з нуля.

Пакет – це збірник коду, який містить в собі готові до використання рішення для певної задачі. Пакети можуть містити готові віджети, функції, класи і додаткові ресурси, що можуть значно спростити процес розробки.

При створенні застосунку було використано 5 пакетів:

- google_maps_flutter – надає доступ до функціоналу Google Maps, використовується для відображення мап на екрані, налаштування параметрів карти та відображення маршрутів на карті.
- geolocator – забезпечує легкий доступ до сервісів локації для конкретної платформи, дозволяє постійно зчитувати геолокацію пристрою, налаштовуючи точність та період зчитування.

- permission_handler – надає крос-платформений (iOS, Android) API для запиту дозволів, наприклад доступ до камери, геопозиції пристрою, контактів тощо, і перевірки їхнього статусу. Також можна відкрити налаштування програми на пристрої, щоб користувач міг надати дозвіл, якщо заборонив його раніше.
- cloud_firestore – використовується для використання API Cloud Firestore.
- firebase_core – використовується для використання Firebase Core API, який дозволяє підключатися до декількох додатків Firebase.

6. Опис програмного коду та функціоналу застосунку

Програмний код застосунку написаний у чотирьох файлах з розширенням .dart, що відповідають за свою частину функціоналу.

Головним є файл main.dart, з якого запускається весь застосунок. Також описано частину інтерфейсу, що відповідає за бокову навігаційну панель що містить поле для введення імені користувача та два пункти “Map” і “Routes”, які відповідають за переходи на відповідні віна застосунку. Окрім цього, під час запуску застосунку, перевіряється підключення до бази даних та перевірка/запит дозволу на доступ до геолокації пристрою.

У файлі map.dart повністю описаний функціонал вікна “Map”, а саме відображення карти, її налаштування, а також кнопка початку/завершення запису маршруту. Вікно “Map” першим відкривається, після запуску застосунку. При старті запису спливає спеціальне вікно для введення назви нового маршруту, а при завершенні записаний маршрут відправляється до бази даних.

Всередині файлу routesDB.dart описано клас RoutesDB, що використовується для збору даних необхідних для записаного маршруту, їх перетворення у потрібний формат та відправки до бази даних.

Файл routesList.dart відповідає за вікно з записаними раніше маршрутами, що зберігаються в базі даних, та їх відтворення. Вікно складається зі списку маршрутів та кнопок відтворення. При натисканні на кнопку відкривається вікно з мапою, на якій відтворено обраний маршрут.

Крім чотирьох файлів з кодом до проекту був доданий файл google-services.json, що використовується для доступу застосунку до сервісу Google Maps та містить ключі доступу до акаунту Google Cloud Platform.

7. Реалізація і застосування проекту

Значна частина територій України наразі не є безпечною для пересування, через потенційну наявність вибухонебезпечних предметів. В першу чергу це стосується деокупованих територій та прифронтових зон. Ідея застосунку полягає в збереженні та відтворенні перевірених безпечних маршрутів.

Безпечні маршрути мають вноситись в базу даних саперними підрозділами. Для цього, під час використання застосунку, в основному вікні “Map” спочатку потрібно натиснути на кнопку початку/завершення запису. Після цього в спливаючому вікні вноситься назва нового маршруту і починається запис. Для завершення повторно натискається та сама кнопка і тоді записаний шлях відправляється до бази даних.

Для того, щоб скористатись записаним маршрутом, користувач має перейти на вікно “Routes”. В цьому вікні відображено список всіх наявних маршрутів, що зберігаються в базі даних. Після знаходження потрібного маршруту, необхідно натиснути на відповідну кнопку відтворення, праворуч від назви, далі буде відкрито нове вікно з відображеним маршрутом, яким можна безпечно пересуватись.

Таким чином, використання даного застосунку дозволить безпечно пересуватись без необхідності шукати провідника або інформацію про шлях.

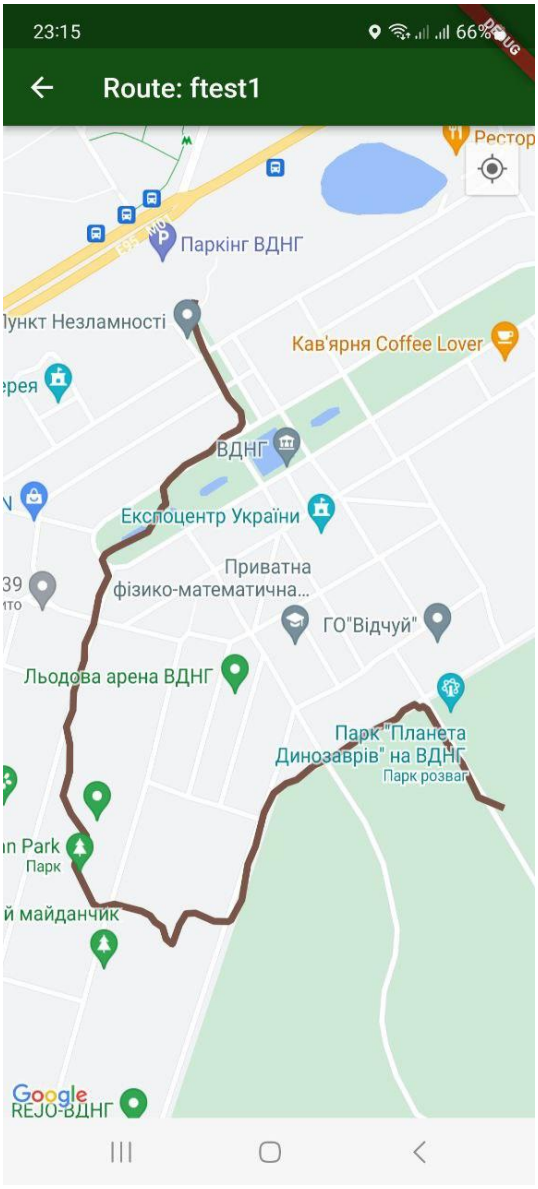

Рис.3 Приклад відтворення записаного маршруту

8. ВИСНОВКИ

В ході виконання дипломної роботи були розглянуті теоретичні матеріали та програмні засоби, що необхідні для створення застосунку.

Для програмної реалізації додатку були обрані фреймворк Flutter та мова програмування Dart, що використовується фреймворком. Даний фреймворк набуває популярності, серед розробників мобільних застосунків, а також має кросплатформені засоби, для легкої реалізації програми на різних пристроях з різними операційними системами, що є надзвичайно важливо в умовах необхідності працездатності на будь-якому пристрої.

Для зберігання даних було обрано Firebase Cloud Firestore – надійна хмарна база даних, що доступна з будь-якого місця в Інтернеті. Firestore має дуже простий та інтуїтивно зрозумілий інтерфейс користувача, що дозволяє легко зберігати та витягувати дані.

Було розроблено мобільний застосунок для запису, збереження та відтворення безпечних маршрутів пересування потенційно замінованими територіями.

9. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Xu, G., & Xu, Y. (2016). Applications of GPS theory and algorithms. In GPS (pp. 313-340). Springer, Berlin, Heidelberg.
2. Zammetti, F. (2019). Practical Flutter. Berkeley, CA: Apress.
3. Dart overview – dart.dev – Режим доступу: URL: [https://dart.dev/overview]
4. Google Maps Platform Documentation – Режим доступу: URL: [https://developers.google.com/maps/documentation]
5. Cloud Firestore Documentation – Режим доступу: URL: [https://firebase.google.com/docs/firestore]

Додаток – Код програми

Файл main.dart:

```
import 'package:flutter/material.dart';
import 'package:my_dyplom_app/map.dart';
import 'package:permission_handler/permission_handler.dart';
import 'package:firebase_core/firebase_core.dart';
import 'routesList.dart';

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(const MainApp());
}

class MainApp extends StatelessWidget {
  const MainApp();

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: HomePage(),
    );
  }
}

class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  var currentPage = DrawerSections.map; // Активна сторінка
  PermissionStatus _permissionStatus =
      PermissionStatus.denied; // Статус дозволів
  String userName = 'User name'; // Ім'я користувача
```

```
// Перевірка та запит дозволів
Future<void> checkPermission() async {
  var status = await Permission.locationWhenInUse.status;
  if (status.isGranted) {
    setState(() {
      _permissionStatus = status;
    });
  } else if (status.isDenied || status.isRestricted) {
    setState(() {
      _permissionStatus = status;
    });
    await Future.delayed(Duration(seconds: 5));
    checkPermission();
  } else if (status.isPermanentlyDenied) {
    openAppSettings();
  }
}

// Початковий стан сторінки, при запуску
@override
void initState() {
  super.initState();
  checkPermission();
}

@override
Widget build(BuildContext context) {
  var _container; // Відповідає за відображення активноїсторінки
  String _title = 'My app'; // Назва сторінки (текст в AppBar)
  // Перемикання сторінки та тексту в AppBar
  if (currentPage == DrawerSections.map) {
    _container = MapViewWidget(userName: userName);
    _title = 'Map';
  } else if (currentPage == DrawerSections.routes) {
    _container = RoutesListScreen();
    _title = 'Routes';
  }

  return Scaffold(
```

```
    appBar: AppBar(
      backgroundColor: Color.fromARGB(255, 20, 80, 20),
      title: Text(_title),
    ),
    body: SafeArea(child: _container),
    drawer: SafeArea(
      child: Column(
        children: [
          myDrawerHead(), // Заголовок з іменем користувача
          Expanded(child: myDrawerList()), // Список з вкладками
        ],
      ),
    ),
  );
}

// Заголовок з іменем користувача
Widget myDrawerHead() {
  // Контролер TextFormField
  TextEditingController _textController =
      TextEditingController(text: userName);
  // Виставлння курсора в кінець рядка
  _textController.selection =
      TextSelection.collapsed(offset: _textController.text.length);

  return Container(
    color: Color.fromARGB(255, 5, 80, 5),
    width: double.infinity,
    height: 130,
    child: Row(
      children: [
        Icon(
          Icons.person,
          size: 100,
        ),
        Expanded(
          child: Padding(
            padding: const EdgeInsets.symmetric(horizontal: 16.0),
            child: TextFormField(
```

```
                autofocus: true,
                controller: _textController,
                style: TextStyle(
                  fontSize: 30,
                ),
                onChanged: (newText) {
                  setState(() {
                    userName = _textController.text;
                  });
                },
              ),
            ),
          ),
        ],
      ),
    );
}

// Список з вкладками
Widget myDrawerList() {
  return Container(
    color: Color.fromARGB(255, 25, 150, 25),
    child: Column(
      children: [
        SizedBox(
          height: 5,
        ),
        menuItem(0, 'Map', Icons.map,
            currentPage == DrawerSections.map ? true : false),
        SizedBox(
          height: 10,
        ),
        menuItem(1, 'Routes', Icons.route,
            currentPage == DrawerSections.routes ? true : false),
      ],
    ),
  );
}
```

```
  // Віджет для опису кожного пункту меню
  Widget menuItem(int id, String title, IconData icon, bool selected) {
    return Material(
            color: selected ? Color.fromARGB(255, 5, 100, 5) :
Colors.transparent,
      child: InkWell(
        // Виставлення сторінки, при натисканні на пункт меню
        onTap: () {
          Navigator.pop(context);
          setState(() {
            if (id == 0) {
              currentPage = DrawerSections.map;
            } else if (id == 1) {
              currentPage = DrawerSections.routes;
            }
          });
        },
        child: Padding(
          padding: const EdgeInsets.symmetric(vertical: 20),
          child: Row(
            children: [
              Padding(
                padding: const EdgeInsets.symmetric(horizontal: 15),
                child: Icon(
                  icon,
                  size: 25,
                ),
              ),
              Expanded(
                  child: Text(
                title,
                style: TextStyle(
                  fontSize: 18,
                ),
              ))
            ],
          ),
        ),
      ),
```

```
    );
  }
}

enum DrawerSections {
  map,
  routes,
}
```

Файл map.dart:

```
import 'package:flutter/material.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'package:geolocator/geolocator.dart';
import 'routesDB.dart';

// Віджет основної карти
class MapViewWidget extends StatefulWidget {
  final String userName;
  const MapViewWidget({Key? key, required this.userName});

  @override
  State<MapViewWidget> createState() => _MapViewWidgetState();
}

class _MapViewWidgetState extends State<MapViewWidget> {
  late GoogleMapController mapController;
  final LatLng _basePos = const LatLng(50.3828, 30.4719);

  bool _isRecording = false; // Чи ведеться запис маршруту

  late String _routeName;
  List<LatLng> _routePoints = [];
  var subscription;
  final LocationSettings locationSettings =
      LocationSettings(accuracy: LocationAccuracy.best);

  void _onMapCreated(GoogleMapController controller) =>
      mapController = controller;
```

```
// Функція для почтку запису маршруту
void _startRecording() async {
  final name = await showDialog(
    context: context,
    builder: (context) => AlertDialog(
      // Виклик вікна для введення назви маршруту
      title: Text('Enter route name'),
      content: TextField(
        onChanged: (value) => _routeName = value,
        decoration: InputDecoration(hintText: 'Route name'),
      ),
      actions: [
        TextButton(
          onPressed: () => Navigator.pop(context),
          child: Text('CANCEL'),
        ),
        ElevatedButton(
          onPressed: () {
            if (_routeName.isNotEmpty) {
              Navigator.pop(context, _routeName);
            }
          },
          child: Text('START'),
        ),
      ],
    ),
  );

  if (name != null) {
    setState(() {
      _isRecording = true;
    });
    // Отримання поточної геолокації з високою точністю
    final Position position = await Geolocator.getCurrentPosition(
      desiredAccuracy: LocationAccuracy.high,
    );
    // Додавання позиції до списку точок
    final routePoint = LatLng(position.latitude, position.longitude);
```

```
      _routePoints.add(routePoint);
      // Виставлення камери на поточку геолокацію
      final CameraPosition initialCameraPosition = CameraPosition(
        target: routePoint,
        zoom: 18,
      );
      mapController

.animateCamera(CameraUpdate.newCameraPosition(initialCameraPosition));

      // Початок слідкування за геолокацією та довання точок до списку
      subscription =
          Geolocator.getPositionStream(locationSettings: locationSettings)
              .listen((Position position) {
                final LatLng routePoint = LatLng(position.latitude,
position.longitude);
        setState(() {
          _routePoints.add(routePoint);
        });
      });
    }
  }

  // Зупинка запису маршруту
  void _stopRecording() {
    setState(() {
      _isRecording = false;
    });

    // Припинення слідкування за геолокацією
    subscription.cancel();

    // Відправка збереженого маршруту до БД
    if (_routePoints.length > 1) {
      RoutesDB newRoute = RoutesDB(
          userName: widget.userName,
          routeName: _routeName,
          routePoints: _routePoints);
      newRoute.addRoute();
```

```
    }

    // Очищення списку точок
    _routePoints = [];
  }

  @override
  Widget build(BuildContext context) {
    return Stack(
      children: [
        // Карта та її налаштування
        GoogleMap(
            initialCameraPosition: CameraPosition(target: _basePos, zoom:
18),
          onMapCreated: _onMapCreated,
          myLocationEnabled: true,
          myLocationButtonEnabled: true,
          zoomControlsEnabled: false,
        ),

        // Кнопка для початку/завершення запису
        Positioned(
          bottom: 16,
          right: 16,
          child: FloatingActionButton(
            onPressed: _isRecording ? _stopRecording : _startRecording,
            child: Icon(_isRecording ? Icons.stop : Icons.play_arrow),
          ),
        ),
      ],
    );
  }
}
```

Файл routesBD.dart:

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
```

```
// Окремий клас для формування та відправки маршруту до БД
class RoutesDB {
  String? userName;
  String? routeName;
  List<LatLng>? routePoints;

  RoutesDB({this.userName, this.routeName, this.routePoints});

  // Метод для відправки маршуту до БД
  addRoute() {
    try {
       // Перетворення List<LatLng> у List<Map<String, dynamic>>, що може бути збереженим в БД
      List<Map<String, dynamic>> nrP = routePoints!
          .map((point) => {
                "latitude": point.latitude,
                "longitude": point.longitude,
              })
          .toList();
      // Відправка маршруту
      FirebaseFirestore.instance
          .collection('routes')
          .add({
            'userName': userName,
            'routeName': routeName,
            'routePoints': nrP,
          })
          // Повідомлення про успіх або провал відправки
          .then((value) => print('Sended!'))
          .catchError((e) => print('Fail!: $e'));
    } catch (e) {
      print('Error adding route: $e');
    }
  }
}
```

Файл routesList.dart:

```
import 'package:cloud_firestore/cloud_firestore.dart';
```

```
import 'package:flutter/material.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';

// Відображення списку маршрутів, що знаходяться в БД
class RoutesListScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return StreamBuilder<QuerySnapshot>(
      stream: FirebaseFirestore.instance
          .collection('routes')
          .snapshots(), // Переглядання колекції routes в БД
      builder: (context, snapshot) {
        if (snapshot.hasError) {
          // Перевірка на помилку
          return Center(
            child: Text('Error: ${snapshot.error}'),
          );
        }

        if (snapshot.connectionState == ConnectionState.waiting) {
          // Повідомлення, якщо дані не встигли завантажитись
          return Center(
            child: Text('Loading...'),
          );
        }

         final data = snapshot.requireData; // Зберігання даних снапшота в змінну

        // Побудова списку
        return ListView.builder(
          itemCount: data.size,
          itemBuilder: (context, index) {
            // Збереження даних у відповідні змінні
            final _routeName = data.docs[index]['routeName'];
            final _userName = data.docs[index]['userName'];
            // Перетворення List<Map<String, dynamic>> у List<LatLng>
                                final    List<LatLng>    _routePoints    =
data.docs[index]['routePoints']
```

```
                .map<LatLng>(
                                    (point)  =>  LatLng(point['latitude'],
point['longitude']))
                .toList();

            // Створення об'єктів списку
            return ListTile(
              title: Text(_routeName),
              subtitle: Text(_userName),
              // Кнопка для відтворення збереженого маршруту
              trailing: IconButton(
                icon: Icon(Icons.play_arrow),
                onPressed: () {
                  // Функція для відтворення маршруту
                  showRouteOnMap(context, _routeName, _routePoints);
                },
              ),
            );
          },
        );
      },
    );
  }
}

// Функція для відтворення маршруту
void showRouteOnMap(
    BuildContext context, String _routeName, List<LatLng> routePoints) {
  // Початкова позиція камери
  final CameraPosition _initialCameraPosition = CameraPosition(
    target: routePoints.first,
    zoom: 16.0,
  );

  // Подува маршруту на основі збережених точок
  final Set<Polyline> _polylines = {
    Polyline(
      polylineId: PolylineId('route'),
      color: Colors.brown,
```

```
      width: 5,
      points: routePoints,
    )
  };

  // Карта та її налаштування
  final GoogleMap _map = GoogleMap(
    initialCameraPosition: _initialCameraPosition,
    polylines: _polylines,
    myLocationEnabled: true,
    myLocationButtonEnabled: true,
    zoomControlsEnabled: false,
  );

  // Відкривання нового вікна
  Navigator.of(context).push(
    MaterialPageRoute(
      builder: (context) => Scaffold(
        appBar: AppBar(
            title: Text('Route: $_routeName'),
            backgroundColor: Color.fromARGB(255, 20, 80, 20),
            // Кнопка для повернення до списку маршрутів
            leading: IconButton(
              icon: Icon(Icons.arrow_back),
              onPressed: () => Navigator.of(context).pop(),
            )),
        body: _map, // Виклик карти з маршрутом
      ),
    ),
  );
}
```