ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ НАВІГАЦІЇ ПО КУЛЬТУРНИХ ОБ'ЄКТАХ МІСТА КИЄВА НА ПЛАТФОРМІ ANDROID

Лучко А.І.
Київський столичний університет імені Бориса Грінченка, м. Київ

ВСТУП

Актуальність і постановка проблеми. Взаємодія сучасної людини з міським середовищем нерозривно пов'язана з мобільними навігаційними платформами. Столиця України володіє екстремально щільним історико-культурним фондом. Продукти-гіганти на кшталт Google Maps проєктувалися переважно під потреби автомобілістів. Як наслідок, їхній графічний інтерфейс перевантажений тисячами нерелевантних точок інтересу (POI). Для пішохідного туриста цей безперервний інформаційний шум створює надмірне когнітивне навантаження.

Крім того, масові застосунки мають критичну архітектурну вразливість — абсолютну залежність від стабільного інтернет-з'єднання. Офлайн-режим у них зводиться до базового кешування растрових мап, а повноцінна маршрутизація без мережі не працює. В умовах щільної міської забудови, перебування в метрополітені чи під час перебоїв зі зв'язком це унеможливлює нормальну навігацію. Тому виникає гостра інженерна потреба у розробці вузькоспеціалізованої, автономної геоінформаційної системи. Вона має працювати за принципом Offline-first, гарантуючи безперебійний доступ до просторових даних.

Мета дослідження. Проєктування та алгоритмічна реалізація спеціалізованого мобільного застосунку для операційної системи Android, орієнтованого на автономну навігацію культурним простором Києва, з інтеграцією механізмів локального кешування, реактивної фільтрації та розв'язання задачі маршрутизації через кілька точок.

Аналіз останніх досліджень і публікацій. Розробка відмовостійких мобільних систем вимагає симбіозу картографічних та архітектурних рішень. В офіційній документації Android Developers [1] фундаментально описано проєктування клієнтського рівня за патерном MVVM. Альтернативою корпоративним мапам є проєкти Organic Maps та MAPS.ME, що базуються на відкритих даних OpenStreetMap. Проте їхня монолітна закрита архітектура унеможливлює інтеграцію глибоких кастомних рекомендаційних алгоритмів.

Оптимізація багатоточкових пішохідних маршрутів математично зводиться до класичної NP-складної задачі комівояжера (TSP). Ефективні евристичні підходи до її розв'язання, зокрема жадібний алгоритм та метод 2-opt, ґрунтовно проаналізовані в алгоритмічній базі Т. Кормена [2]. Для оптимізації рендерингу складних поліліній на мобільних пристроях стандартом де-факто залишається алгоритм Дугласа-Пекера [4].

Проєктування інтерфейсів взаємодії "на ходу" спирається на Закон Фіттса [5] та стандарти WCAG 2.1.

Короткий опис дослідження, його методів і засобів

Фундаментом розроблюваної системи стала концепція Single Activity. Замість класичної моделі з важкими переходами між вікнами, візуальну частину побудовано на декларативному фреймворку Jetpack Compose. Архітектура застосунку суворо дотримується принципу Separation of Concerns і розбита на три ізольовані рівні (Presentation, Domain, Data) [1]. За управління залежностями відповідає бібліотека Hilt, що зводить зв'язність компонентів до мінімуму. Загальну ієрархію модулів наведено на рисунку 1.

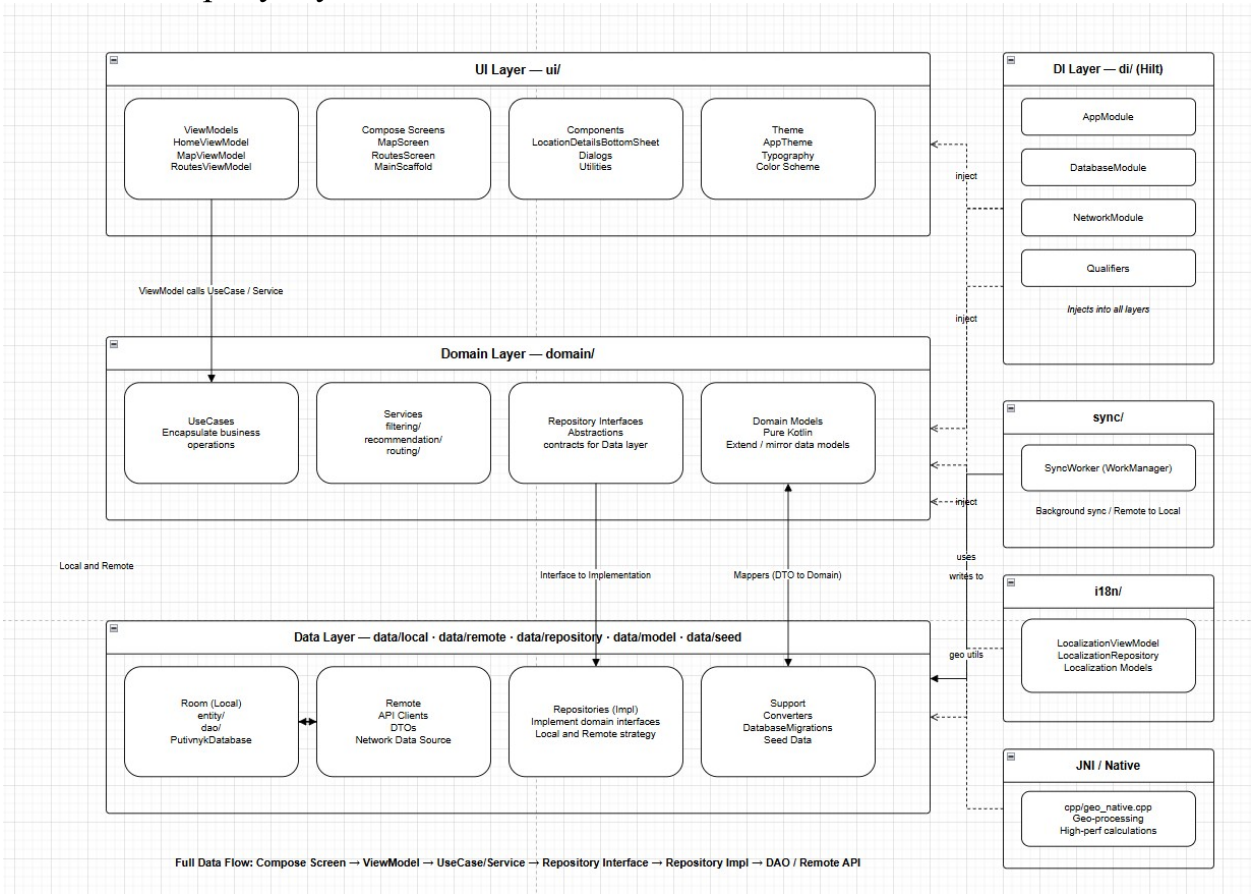

Рис. 1. Структурна схема застосунку

Усю роботу з локальним сховищем переведено на рейки реляційної бази даних SQLite через ORM-обгортку Room [7]. Каталог локацій зведено до строгої нормальної форми (6 взаємопов'язаних таблиць). Щоб уникнути лінійного сканування бази під час "живого пошуку" (Live Search), текстові поля проіндексовано за допомогою віртуальних таблиць FTS4 (Full-Text Search). Завдяки цьому швидкість відфільтровування даних впала до 2-3 мілісекунд, що забезпечує неблокуючий інтерфейс (Non-blocking UX). За тіньову синхронізацію масивів даних із віддаленим сервером відповідає системний планувальник WorkManager.

Картографічне ядро спирається на нативний рушій MapLibre GL Native [8]. Окремим архітектурним викликом став безперервний математичний розрахунок дистанцій між об'єктами. Розрахунок ортодромій за формулою Гаверсинуса вимагає інтенсивних обчислень. Щоб не перевантажувати процесор смартфона (віртуальну машину JVM), цю математику написано мовою C++ і скомпільовано у нативну бібліотеку, яка спілкується з основним кодом через JNI-міст (Java Native Interface).

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

У процесі розробки створено готовий програмний продукт, який вирішує ключові задачі міської туристичної логістики. Інтерфейс жорстко підпорядковано моторній доступності: зони кліку (Hit Box) маркерів та кнопок програмно розширено до 48x48 dp [6] для усунення хибних натискань на ходу. Реалізовано модуль геопросторових тригерів (Geofencing), який розраховує наближення до об'єкта та генерує контекстні сповіщення без потреби постійно дивитися на екран.

Прокладання багатоточкових маршрутів реалізовано через конвеєр оптимізації. Спочатку інтелектуальний модуль генерує матрицю відстаней. Далі застосовується жадібний алгоритм "найближчого сусіда". Оскільки цей метод часто створює неоптимальні петлі на маршруті, результати додатково пропускаються через евристичний алгоритм локального пошуку 2-opt [2], який розплутує самоперетини. Логіку усунення перетинів проілюстровано на блок-схемі (рис. 2). Для забезпечення надійної автономної роботи застосунку реалізовано підсистему локального зберігання просторових даних на базі SQLite з використанням абстракції Room. Пошук точок інтересу (POI) здійснюється через механізм повнотекстового пошуку FTS4, що гарантує миттєву видачу результатів навіть на пристроях із низькою обчислювальною потужністю, повністю виключаючи мережеві затримки. Архітектура мобільного клієнта побудована за патерном MVVM (Model-View-ViewModel) із дотриманням принципів Clean Architecture. Це дозволило чітко ізолювати бізнес-логіку від шару представлення. Фонова тіньова синхронізація локальної бази даних із віддаленим джерелом виконується за допомогою WorkManager, який автоматично керує транзакціями під час появи стабільного підключення до мережі.

Проєктування графічного інтерфейсу користувача здійснено за декларативним підходом за допомогою фреймворку Jetpack Compose. Повна відмова від класичної XML-розмітки на користь парадигми Code-First дозволила суттєво зменшити обсяг шаблонного коду та усунути проблеми розсинхронізації між внутрішнім станом програми (State) та візуальними компонентами. Агресивне кешування у поєднанні з нативним рушієм MapLibre GL Native забезпечує плавний рендеринг картографічної підкладки зі стабільним показником 60 FPS, навіть при одночасному

відображенні складних багатоточкових поліліній маршрутів та великої кількості маркерів культурних об'єктів.

Під час профілювання на тестових пристроях було виявлено, що рендеринг "сирих" відповідей від сервера OSRM [3] (масиви на тисячі координат) викликає критичне переповнення оперативної пам'яті. Цю проблему усунуто впровадженням модифікованого алгоритму Дугласа-Пекера [4] з допуском відхилення 5-7 метрів. Він автоматично видаляє проміжні точки на прямих відрізках дороги, стискаючи вагу маршруту на 60-80% зі збереженням стабільних 60 FPS при скролінгу карти.

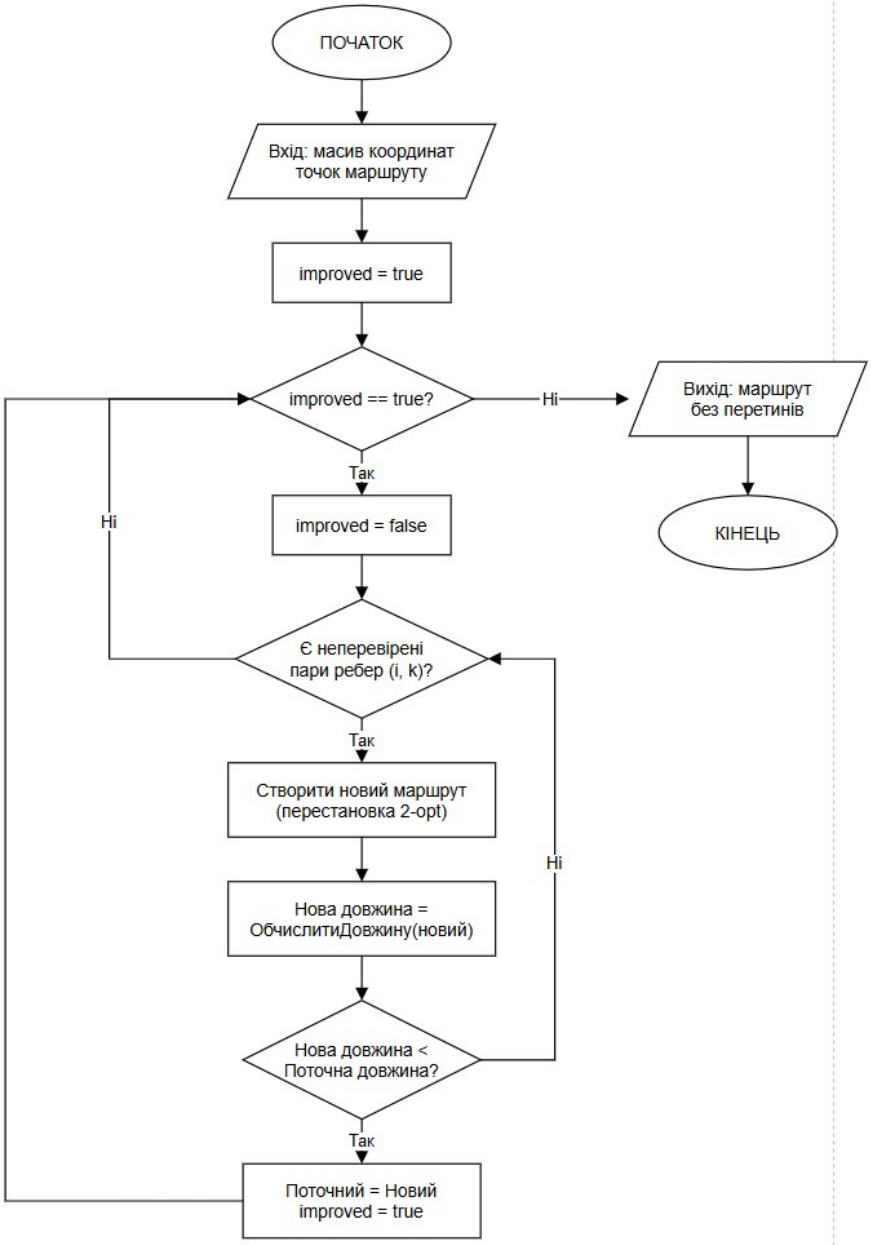

Рис.2. Блок-схема алгоритму усунення самоперетинів маршруту (2-opt)

Для підтримки іноземних туристів розроблено автономний модуль перекладу на базі машинного навчання. Замість використання хмарних API, застосовано технологію Google ML Kit. Переклад текстів відбувається

безпосередньо на процесорі пристрою за трирівневою каскадною схемою (ресурси, локальний кеш бази даних, ML-модель). Структурну схему механізму локалізації наведено на рисунку 3.

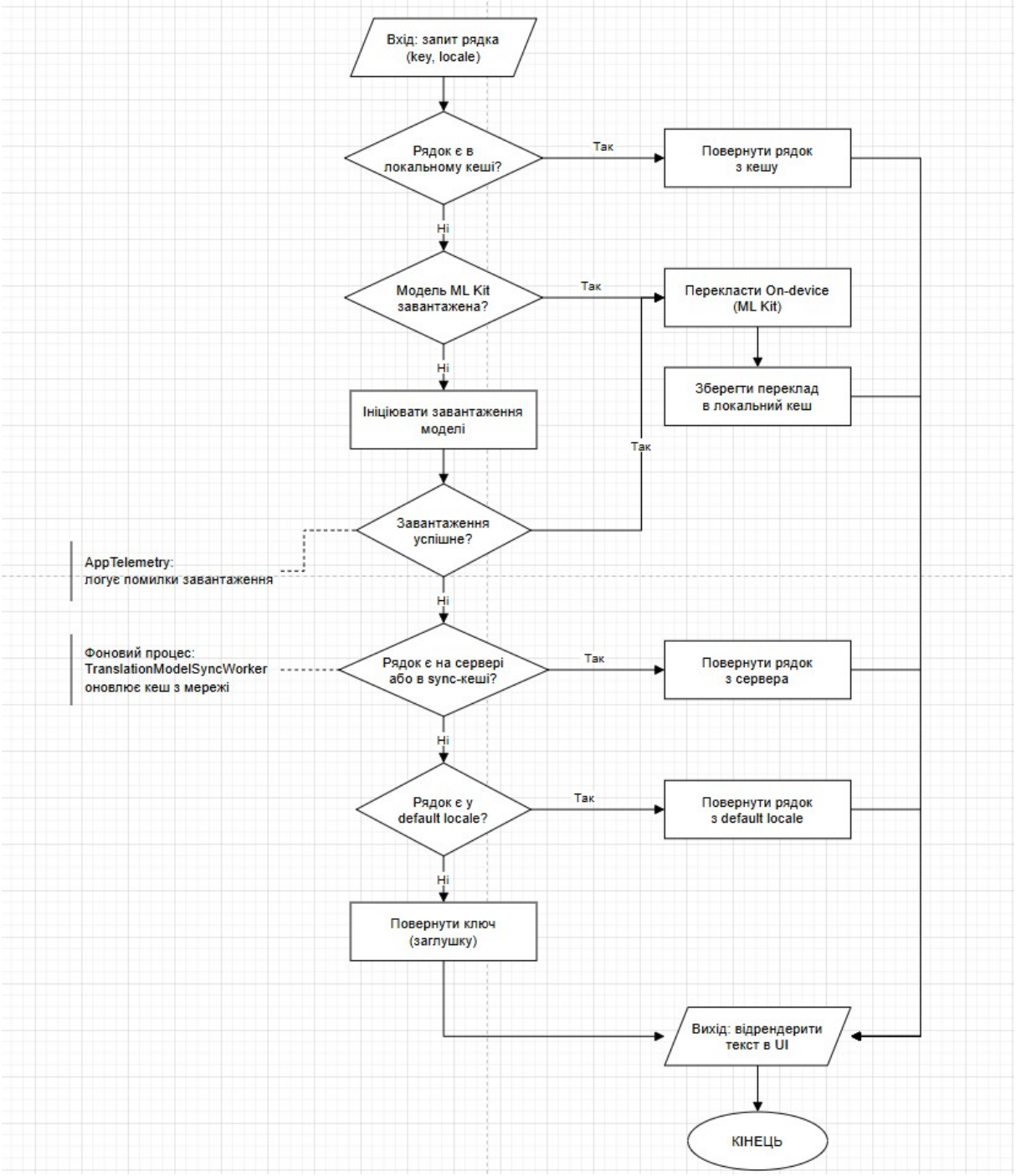

Рис. 3. Структурна схема каскадного механізму локалізації

ВИСНОВКИ

Підсумком виконання кваліфікаційної роботи є створення повністю функціонального мобільного застосунку для автономної навігації культурними просторами Києва. Інтеграція Offline-first підходу, ORM-бази Room з FTS4-індексацією та нативного картографічного рушія дозволила

забезпечити стовідсоткову працездатність системи в умовах відсутності мережі. Впровадження математичного апарату (евристика 2-opt, алгоритм Дугласа-Пекера, обчислення Гаверсинуса на рівні C++) вирішило проблему NP-складних багатоточкових маршрутів, радикально скоротивши навантаження на мобільний процесор та акумулятор. Розроблена архітектура є масштабованою і може бути інтегрована в інші муніципальні геоінформаційні системи.

ДЖЕРЕЛА

1. Guide to app architecture. Android Developers. URL: https://developer.android.com/topic/architecture (дата звернення: 30.03.2026).

2. Кормен Т., Лейзерсон Ч., Рівест Р., Стайн К. Вступ до алгоритмів. 3-тє вид. Київ : К.І.С., 2019. 1288 с.

3. Open Source Routing Machine API. Project OSRM. URL: http://project-osrm.org/docs/v5.24.0/api/ (дата звернення: 30.03.2026).

4. Douglas D. H., Peucker T. K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. The Canadian Cartographer. 1973. Vol. 10, no. 2. P. 112–122.

5. Fitts's Law: The Importance of Size and Distance in UI Design. Interaction Design Foundation. URL: https://www.interaction-design.org/literature/article/fitts-s-law-the-importance-of-size-and-distance-in-ui-design (дата звернення: 30.03.2026).

6. Accessibility in Compose. Android Developers. URL: https://developer.android.com/jetpack/compose/accessibility (дата звернення: 30.03.2026)

7. Save data in a local database (Room). Android Developers. URL: https://developer.android.com/training/data-storage/room (дата звернення: 30.03.2026).

8. MapLibre GL Native. Official documentation. URL: https://maplibre.org/maplibre-native/ (дата звернення: 30.03.2026).

9. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., Hlushak, O., Zhyltsov, O., Ilich, L., Kobets, N., Kovaliuk, T., Kuchakovska, H., Lytvyn, O., Lytvyn, P., Mashkina, I., Morze, N., Nosenko, T., Proshkin, V., Radchenko, S., & Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science. https://doi.org/10.28925/9720213284km