Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Мобільний застосунок «Путівник по місту Харків»

(тема)

Виконав:

здобувач 3 року навчання,

групи КІУКІу-22-1

Даніл СТОЯНОВ
(власне ім’я, прізвище)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма

Комп’ютерна інженерія
(повна назва освітньої програми)

Керівник: доц. Тетяна ФІЛІМОНЧУК
(посада, власне ім’я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім’я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Стоянову Данілу Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи Мобільний застосунок «Путівник по місту Харків»

затверджена наказом по університету від “ 26 ” травня 2025 р. № 425 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 14 липня 2025 р.

3. Вхідні дані до роботи

1. Мова програмування Dart
2.Фраймворк Flutter
3. Середовище розробки Visual Studio Code
4. Бібліотеки: flutter_map, flutter_bloc, http, sqllite, geolocartor, url_launcher
5. База даних SQLite
6. Інструменти картографії: OpenStreetMap, Google Maps

4. Перелік питань, що потрібно опрацювати у роботі

1. Вступ
2. Аналіз предметної області
3. Аналіз технологій, що використовуються при розробці мобільного застосунку
4. Опис програмної реалізації застосунку
5. Інструкція користувача
6. Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд-презентація – 10 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз існуючих методів вирішення задачі	10.06.2025-13.06.25	
2	Вибір програмного забезпечення та	14.06.2025-17.06.25	
	інструментів розробки		
3	Проєктування архітектури застосунку	18.06.25-21.06.25	
4	Розробка логіки застосунку	22.06.25-28.06.25	
5	Розробка графічного інтерфейсу користувача	29.06.25-02.07.25	
6	Тестування застосунку	03.07.25-05.07.25	
7	Подання кваліфікаційної роботи керівникам	06.07.25-09.07.25	
	для попереднього захисту		
8	Подання кваліфікаційної роботи на	10.07.25-11.07.25	
	рецензування		

Дата видачі завдання " 09 " червня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Тетяна ФІЛІМОНЧУК
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 75 с., 23 рис., 1 табл. 2 дод., 26 джерел.

МОБІЛЬНИЙ ЗАСТОСУНОК, FLUTTER, DART, НАВІГАЦІЯ, ФІЛЬТРАЦІЯ, КАРТА, GPS, SQLITE, OPENSTREETMAP, GOOGLE MAPS, UX, UI, BLOC, CLEAN ARCHITECTURE.

Метою кваліфікаційної роботи є створення мобільного застосунку-путівника по місту Харків для мешканців та туристів.

У процесі розробки реалізовано мобільний застосунок з інтерактивною картою, каталогом локацій, пошуком на карті визначних місць, ресторанів та кафе, побудовою маршрутів та відображенням актуальної інформації про місця. Застосунок створено з використанням мови Dart та фреймворку Flutter.

Архітектура побудована за принципами Clean Architecture та BLoC, що забезпечує модульність, масштабованість та тестованість. Для зберігання даних використано локальну базу SQLite, а навігація реалізована з інтеграцією OpenStreetMap та Google Maps API. Також реалізовано підтримку побудови піших маршрутів, фільтрації за типами місць та обробки поточної геолокації користувача.

Особливу увагу приділено зручному інтерфейсу (UI) та досвіду користувача (UX), адаптованому для мобільних пристроїв та швидкої взаємодії з картою та списками.

Результат кваліфікаційної роботи демонструє застосування сучасних технологій та підходів у розробці мобільних застосунків, а сам проєкт може слугувати основою для розширення туристичних цифрових сервісів міста Харків.

ABSTRACT

Bachelor’s thesis: 75 pages, 23 figures, 1 table, 2 appendices, 26 sources.

MOBILE APPLICATION, FLUTTER, DART, NAVIGATION, FILTERING, MAP, GPS, SQLITE, OPENSTREETMAP, GOOGLE MAPS, UX, UI, BLOC, CLEAN ARCHITECTURE.

The purpose of the qualification work is to create a mobile application-guide to the city of Kharkiv for residents and tourists.

During the development process, a mobile application with an interactive map, a catalog of locations, a map search for landmarks, restaurants and cafes, route building and display of current information about places was implemented. The application was created using the Dart language and the Flutter framework.

The architecture is built according to the principles of Clean Architecture and BLoC, which ensures modularity, scalability and testability. A local SQLite database is used to store data, and navigation is implemented with the integration of OpenStreetMap and Google Maps API. Support for building walking routes, filtering by place types and processing the user's current geolocation is also implemented.

Special attention was paid to a convenient interface (UI) and user experience (UX), adapted for mobile devices and quick interaction with the map and lists.

The result of the qualification work demonstrates the application of modern technologies and approaches in the development of mobile applications, and the project itself can serve as the basis for expanding the tourist digital services of the city of Kharkiv.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ОС – операційна система

AOT – компілятор (англ., Ahead-of-Time)

API – інтерфейс прикладного програмування (англ., Application Programming Interface)

BLoC – компонент бізнес-логіки (англ., Business Logic Component)

DRY – принцип програмування «не повторюйся» (англ., Don't Repeat Yourself)

GPS – глобальна система позиціонування (англ., Global Positioning System)

HTTP – протокол передавання гіпертексту (англ., HyperText Transfer Protocol)

HTTPS – захищений протокол передавання гіпертексту (англ., HyperText Transfer Protocol Secure)

iOS – операційна система iPhone (англ., iPhone Operating System)

JIT – компіляція під час виконання (англ., Just-In-Time)

KISS – принцип програмування «коротко та просто» (англ., Keep It Simple, Stupid)

macOS – операційна система для комп’ютерів Mac (англ., Macintosh Operating System)

OSRM – машина маршрутизації з відкритим кодом (англ., Open Source Routing Machine)

UI – інтерфейс користувача (англ., User Interface)

URL – уніфікований локатор ресурсу (англ., Uniform Resource Locator)

UX – користувацький досвід (англ., User Experience)

ВСТУП

У сучасному світі технології стали невід’ємною частиною практично кожної сфери діяльності людини. Виробництво, освіта, медицина, транспорт, культура – усі ці галузі зазнали значного впливу цифрових інновацій, що не лише покращують зручність та ефективність їх використання, але й сприяють появі нових підходів до вирішення щоденних завдань.

Цифрові пристрої, мобільне програмне забезпечення та програмні платформи стали невід'ємними супутниками людини, пропонуючи інструменти, які змінюють спосіб життя, роботи та відпочинку.

Одним із найбільш динамічних напрямків на даний час є розробка мобільних застосунків, які стають універсальними помічниками для вирішення різноманітних задач: від організації повсякденних справ та планування часу до пошуку нових маршрутів для подорожей, дослідження невідомих місць. Як висновок можливо зазначити, що мобільні застосунки спрощують життя, надаючи швидкий доступ до інформації та можливість взаємодії з нею [1].

Розробка мобільного застосунку для отримання геолокаційної інформації міста Харкова, спрямована на створення інноваційного рішення для мешканців та гостей міста. Путівник об'єднає в собі інформацію про туристичні маршрути, культурні пам’ятки, місця відпочинку, транспортну інфраструктуру та популярні заклади. Унікальна особливість даного програмного продукту полягатиме у поєднанні зручного інтерфейсу з функціональністю, що дозволить користувачам швидко знаходити необхідні дані та отримувати рекомендації, враховуючи їхні вподобання та потреби.

Сучасні мобільні технології та інтеграція геолокаційних сервісів відкривають можливості для створення інструментів, які не лише полегшують навігацію по місту, але й пропонують персоналізований підхід. Використовуючи дані про місцезнаходження, мобільний застосунок

надаватиме інформацію про цікаві місця неподалік, маршрути до них та актуальні події [2].

Реалізація цього проєкту сприятиме підвищенню туристичної привабливості Харкова, зміцненню його іміджу як сучасного європейського міста та забезпечить зручність для його мешканців. Мобільний путівник допоможе популяризувати місцеві заклади та привернути до них нових клієнтів, а також допоможе туристам легко орієнтуватися в місті, знаходити цікаві місця та планувати свої подорожі.

Дане рішення стане прикладом того, як технології можуть не лише покращити досвід користувачів, але й зробити значний внесок у розвиток локальної інфраструктури. Мобільний застосунок з інформацією про місто Харків – це крок уперед до інтеграції цифрових рішень у міський простір, який вдосконалює взаємодію людини з її оточенням та робить її більш комфортною та доступною.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис задачі

Протягом виконання кваліфікаційної роботи слід приділити увагу вирішенню низки задач:

- провести аналіз існуючих аналогів на даний час геолокаційних застосунків з метою розуміння недоліків;
- розробити основні компоненти інтерфейсу майбутнього застосунку, зокрема головний екран, інтерактивну карту, каталог локацій і сторінку з інформацією про пам’ятку;
- розробка застосунку для надання інформації про визначні місця Харкова;
- забезпечення користувачів детальною інформацією про кожну пам’ятку;
- забезпечення користувачів актуальною інформацією про ресторани та кафе Харкова;
- створення сучасного дизайну застосунку відповідно до актуальних стандартів;
- покращення досвіду користувача (UX) через інтерактивні анімації та інтуїтивну навігацію, доповнивши інструментами пошуку пам’яток із гнучкою фільтрацією за категоріями;
- створення кодової бази, яка є структурованою, масштабованою та легко підтримуваною;
- забезпечити інтеграцію з навігаційним сервісом Google Maps для прокладання маршрутів до пам’яток та інших місць у випадку відмови користувача використовувати OpenStreetMap;

1.2 Аналіз існуючих рішень

У сучасному світі мобільні застосунки-путівники стали невід'ємною частиною туристичних подорожей та щоденного життя мешканців міст. Вони дозволяють отримувати інформацію про культурні, історичні та розважальні місця, будувати маршрути, знаходити ресторани, готелі та користуватися громадським транспортом.

Для створення ефективного продукту важливо провести аналіз існуючих рішень і виділити їхні сильні та слабкі сторони. Аналіз дозволяє зрозуміти, які функції вже реалізовані в інших проєктах, а які є нішевими та можуть стати ключовими для продукту, що розробляється.

Виявлення слабких сторін конкурентів допоможе уникнути типових помилок та створити більш зручний та функціональний кінцевий продукт. Аналіз відгуків користувачів існуючих путівників також дозволить зрозуміти, які функції користувачі вважають найбільш важливими, а які потребують вдосконалення. Глибокий аналіз існуючих мобільних путівників є необхідним етапом розробки нового продукту. Він дозволить не тільки зрозуміти потреби користувачів, але й створити унікальний та конкурентоспроможний продукт, який задовольнить найвищі вимоги сучасних користувачів.

1.2.1 Мобільний застосунок «Kiev»

Мобільний застосунок «Kiev» – є туристичним путівником по місту Київ. Він має інформацію про доступні екскурсії, пошук готелів, карти метро та автомобільних доріг, статичну карту міста. Інтерфейс виглядає простим з мінімалістичним дизайном із акцентом на зображення (рисунок 1.1) та візуальні елементи [3]. Проте мобільний застосунок має певні недоліки. При завантаженні та першому використанні застосунку він має незручності та недоліки в інтерфейсі, при натисканні на певні блоки з малюнками або

текстом для більш детального перегляду визначного місця виникають проблемі. Тобто малюнки не відображаються користувачеві або знаходяться в неправильному місці такі проблемі трапляються не завжди, але це приносить дискомфорт при користуванні застосунком.

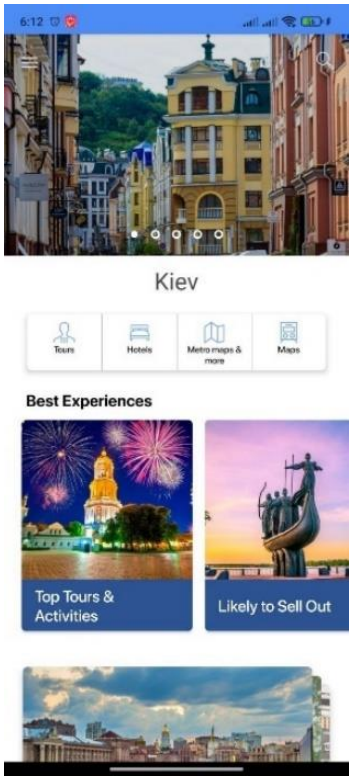


Рисунок 1.1 – Головна сторінка мобільного застосунку «Kiev»

Аналіз відгуків користувачів, який є в магазині Google Play, говорить про те, що карта метро не актуальна на теперішній час, також не працює розділ з екскурсіями, та мало інформації про визначні об’єкти.

1.2.2 Мобільний застосунок «LvivGuide»

Мобільний застосунок «LvivGuide» – це путівник по місту Львів. Він пропонує все необхідне для комфортного перебування в місті: від детальних карт з маршрутами до рекомендацій щодо ресторанів та розваг. Інтуїтивний інтерфейс з яскравими фотографіями (рисунок 1.2) дозволяє легко знайти потрібну інформацію. Програма містить інтерактивну карту міста, пошук

житла та цікаві локації [4]. При використанні цього застосунку були знайдені певні недоліки, а саме дизайн інтерфейсу окремого визначного місця може бути перевантаженим на деяких пристроях, що вносить незручності в його використанні. Також при натисканні на кнопку для відкриття карти з позначкою, де знаходиться окремий об’єкт була проблема – вона не відкривалась, а сам застосунок перестав реагувати на будь які дії і тому його довелося закрити та відкрити його заново для подальшого використання.

Також проаналізувавши відгуки в магазині Google Play від інших користувачів застосунком було виявлено, що карта працює з певними проблемами, а саме вона зовсім не відкривається або відкривається але не відображає позначки з визначними місцями.

Рисунок 1.2 – Головна сторінка мобільного застосунку «LvivGuide»

1.2.3 Мобільний застосунок «Kolomyia: guide and routes»

Мобільний застосунок «Kolomyia: guide and routes» виглядає добре організованим (рисунок 1.3) та візуально привабливим. В горі екрану є

чіткий заголовок «List of locations», що одразу інформує користувача про зміст сторінки. Використання вкладок «Popular», «Monuments», «Architecture», «Religious» дозволяє швидко перемикатися між категоріями [5]. На першому екрані видно лише назви місць та фотографії і часто не вистачає ключової інформації, наприклад, опису локацій.

Усі картки виглядають однаково, що може бути незручним для користувачів, які хочуть швидко виділити щось важливе і не видно, яка саме вкладка активна (це може заплутати користувача). Також не вистачає рядка пошуку або фільтрів для більш гнучкого пошуку потрібної локації.

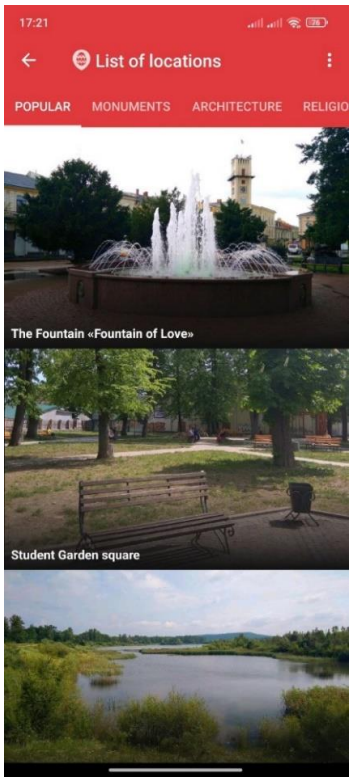


Рисунок 1.3 – Сторінка мобільного застосунку «Kolomyia: guide and routes»

1.2.4 Мобільний застосунок «Львівщина»

Мобільний застосунок-путівник «Львівщина» має зручний інтерфейс із логічним розташуванням елементів та чіткою структурою. Приємний дизайн та якісні фотографії пам’яток роблять користування сервісом комфортним і приємним [6]. Великим плюсом є розділення за категоріями, такими як

пам’ятки, музеї та туристичні маршрути, що полегшує пошук потрібної інформації. Також застосунок підтримує кілька мов (українську та англійську), що робить його корисним для туристів. Програма містить інтерактивну карту з позначеними локаціями та фотографіями об’єктів, що значно спрощує орієнтування. Крім того, є детальні описи місць, що дозволяє дізнатися більше про історію та значення об’єктів (рисунок 1.4). Навігація інтуїтивно зрозуміла завдяки доступним вкладкам «Довкола», «Мапа», «Пленер» та «Головна».

Однак у застосунку є й деякі недоліки. Відсутність фільтрів при відкритті карти ускладнює пошук об’єктів за інтересами, такими як музеї, парки, храми, театри, що змушує користувача кожного разу натискати на мітку на карті, щоб дізнатися деталі. Також невідомо, чи є можливість прокладати маршрути на карті, що могло б бути корисним для туристів.

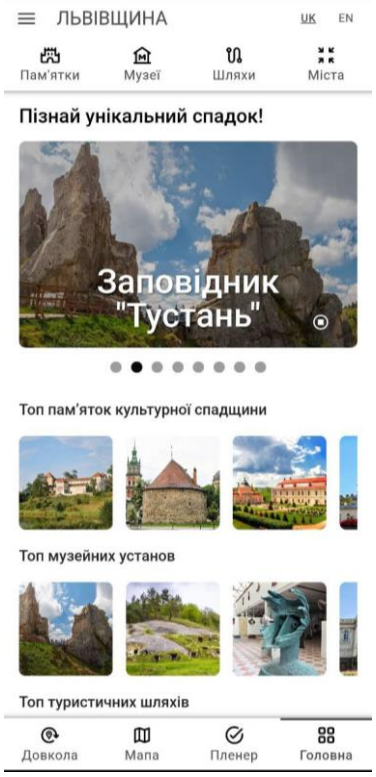


Рисунок 1.4 – Головна сторінка мобільного застосунку «Львівщина»

Ще одним важливим недоліком є відсутність інформації про місця для відпочинку, такі як ресторани та бари, що також важливо для туристів. Крім

того, якщо у застосунку міститься багато тексту без акцентів та виділення ключових моментів, користувачам може бути складно знайти найважливішу інформацію. Загалом застосунок виглядає якісним та корисним, але є можливості для його покращення.

1.3 Результат аналізу існуючих рішень

При детальному розгляді мобільних застосунків з функціоналом путівника по місту було виявлено наступні недоліки:

- деякі застосунки мають незручний інтерфейс, а саме при навігації по застосунку або при ознайомлені з певним визначним об’єктом;
- в деяких застосунках погано працює карта з позначками визначних міст;
- недостатньо інформації про історію міста або про певне визначне місце;
- багато застосунків мають застарілий дизайн і невиразну графіку, що негативно впливає на загальне враження від використання;
- карти в деяких застосунках містять неточну інформацію про розташування об'єктів, що може призвести до того, що користувач потрапить не туди, куди планував.

1.4 Постановка задачі кваліфікаційної роботи

З огляду на опис задачі, потрібно розробити кросплатформний мобільний застосунок, який матиме наступний набір можливостей:

- перегляд списку визначних міст;
- перегляд списку ресторанів та кафе;
- фільтрація визначних об’єктів, ресторанів та кафе;
- відображення позначок визначних об’єктів на карті;
- відображення позначок ресторанів та кафу на карті;

- відображення інформації про окремі визначні місця;
- відображення інформації про історію Харкова.
- пошук місцезнаходження користувача та центрування карті відповідно до користувача;
- будувати маршрути до визначних місць відповідно до місцезнаходження користувача;
- забезпечити користувачеві вибір між Goolgle Maps або OpenStreetMap при прокладанні маршруту;
- надати користувачеві інформацію про його маршрут, а саме: відобразити маршрут на карті у вигляді лінії, довжину маршруту та приблизній час в дорозі до пункту призначення.

При розробці мобільної платформи важливо дотримуватися перевірених принципів, які довели свою ефективність на практиці, таких як DRY, KISS.

DRY (Don't Repeat Yourself) та KISS (Keep It Simple, Stupid) – два фундаментальні принципи, які лежать в основі ефективної розробки програмного забезпечення. Дотримання цих принципів дозволяє створювати якісніший, масштабований код, який легко підтримується [7].

Суть принципу DRY у тому, що кожна функціональність має бути реалізована лише в одному місці: якщо у коді зустрічається якийсь фрагмент, який виконує певну задачу, то цей фрагмент повинен використовуватися скрізь, де потрібно виконання цієї задачі (і бажано не дублюватися).

KISS означає, що рішення має бути максимально простим: складний код важче зрозуміти, підтримувати та модифікувати в подальшому.

Принципи DRY та KISS є фундаментальними для створення якісного програмного забезпечення. Їх дотримання дозволить створити конкурентний проєкт.

2 АНАЛІЗ ТЕХНОЛОГІЙ, ЩО ВИКОРИСТОВУЮТЬСЯ ПРИ РОЗРОБЦІ МОБІЛЬНОГО ЗАСТОСУНКУ

2.1 Огляд засобів розробки

2.1.1 Мова програмування Dart

Мова програмування Dart – це сучасний, високорівневий та компільований інструмент розробки, створений компанією Google та вперше представлений у 2013 році. Основне призначення Dart – створення високопродуктивних програм, включаючи мобільні та веб додатки, а також високонавантажені сайти. Dart поєднує особливості синтаксису таких мов, як C, Python, Java та JavaScript, підтримуючи об'єктно-орієнтовану парадигму програмування [8].

Свого часу перед Google стояла задача – розробити заміну JavaScript, який, на їхню думку, містив фундаментальні недороблення. Завдяки підтримці великої технологічної компанії та перспектив подальшого розвитку, мова Dart стала однією з найбільш затребуваних мов програмування в сучасному світі.

Dart ефективно використовує два типи компіляції – Just-In-Time (JIT) та Ahead-Of-Time (AOT), що робить її універсальним та зручним інструментом для розробки.

JIT-компіляція застосовується під час розробки та тестування. Цей метод компілює код «на льоту», прискорюючи цикл розробки, дозволяючи оперативно вносити зміни та перевіряти їх за допомогою функції Hot Reload, яка дозволяє миттєво перезавантажувати прототип програми без втрати стану, що прискорює процес налагодження.

Однак, програми, які використовують лише JIT-компіляцію, можуть запускатися довше і демонструвати менш стабільну продуктивність на етапі

використання. AOT-компіляція використовується для підготовки фінальних версій програм. Цей метод компілює код заздалегідь у машинний чи нативний код платформи, забезпечуючи швидке виконання програм та стабільність роботи. AOT-компіляція робить програми більш продуктивними та зменшує час їх запуску, що важливо для створення високоякісного досвіду користувача.

Dart успішно поєднує переваги обох компіляцій: JIT-компіляція прискорює процес розробки та налагодження, а AOT-компіляція забезпечує високу продуктивність готової програми. Така гнучкість робить Dart ідеальним вибором для кросплатформної розробки з використанням фреймворку Flutter, дозволяючи розробникам досягати швидких результатів на етапі створення та відмінної продуктивності на етапі експлуатації.

Слід зазначити, що мова Dart набула широкої популярності завдяки ключовій ролі в роботі з фреймворком Flutter, який використовується для створення мобільних застосунків на ОС Android, iOS. Як основна мова програмування для Flutter, Dart забезпечує високу продуктивність, зручність розробки та швидкий цикл тестування.

2.1.2 Фреймворк Flutter

Flutter – це відкритий фреймворк для розробки кросплатформних застосунків. Його основна особливість полягає у використанні єдиної кодової бази для створення програм на Android, iOS, вебплатформах та інших системах, що значно спрощує та прискорює процес розробки, дозволяючи скоротити витрати та уникнути написання окремих кодів для кожної платформи [9].

Основною мовою програмування є Dart, яка забезпечує високу продуктивність кінцевих продуктів та багаті можливості розробки. Програми на Flutter складаються з віджетів, які дозволяють створювати адаптивні та функціональні інтерфейси користувача. Flutter використовує власний

графічний рушій (Skia або Impeller), який забезпечує відтворення інтерфейсів незалежно від платформи. Завдяки цьому, програми можуть функціонувати швидко та плавно, без застосування JavaScript, що робить їх роботу більш ефективною в порівнянні з такими фреймворками, як React Native.

Ключовою функцією Flutter є гаряче перезавантаження (Hot Reload), яке дозволяє розробникам миттєво бачити зміни у коді без перезапуску програми, що значно прискорює процес налагодження, тестування та налаштування інтерфейсу користувача. Завдяки цьому розробники можуть більше часу приділяти поліпшенню логіки програми та інтерфейсу.

Flutter надає велику бібліотеку віджетів, які дозволяють створювати інтерфейси, що відповідають рекомендаціям Google та Apple, а також адаптувати їх до унікальних вимог застосунків. Віджети повністю рендеряться всередині фреймворку Flutter, що дає повний контроль над зовнішнім виглядом та поведінкою інтерфейсів.

Також слід зазначити, що фреймворк пропонує автоматизовані інструменти для тестування та дозволяє мінімізувати необхідність ручної перевірки, що прискорює розробку застосунків, забезпечуючи їхню готовність до виробничого використання. Flutter – потужний інструмент для створення швидких, якісних та кросплатформних рішень.

Він поєднує високу продуктивність, гнучкість віджетів та зручність розробки, що робить його популярним вибором для Android, iOS та вебплатформ. Flutter входить до десятки найкращих репозиторіїв на GitHub, пропонуючи універсальні рішення для сучасних програм.

2.1.3 Редактор коду Visual Studio Code

Visual Studio Code (VS Code) – це редактор коду, розроблений компанією Microsoft [10]. Він став одним із самих популярних редакторів завдяки функціоналу, гнучкості та великій спільноті. VS Code підтримує безліч мов програмування та інструментів, що робить його універсальним

інструментом для різних типів розробки, від веб розробки до розробки мобільних та десктопних програм. Однією з головних рис VS Code є підтримка розширень: тисячі плагінів дозволяють інтегрувати редактор з різними фреймворками, інструментами та технологіями.

VS Code надає потужні інструменти для налагодження коду, включаючи можливість ставити точки зупинки, покрокове налагодження, аналіз стеку викликів та змінних. Ці інструменти працюють як для локальних програм, так і для віддалених серверів, що робить налагодження зручним та ефективним.

VS Code працює на ОС Windows, macOS та Linux, що робить його доступним для розробників на будь-якій платформі. Редактор має зручний та мінімалістичний інтерфейс, який можна повністю налаштувати під потреби розробника. Користувачі можуть змінювати теми оформлення, налагоджувати панелі інструментів, а також використовувати різні гарячі клавіші та налаштування користувача, щоб працювати швидше.

2.1.4 Інструмент тестування мобільних застосунків Android Emulator

Android Emulator – це потужний інструмент для розробників, що дозволяє тестувати мобільні застосунки у віртуальному середовищі. Він забезпечує можливість імітувати різноманітні пристрої Android, включаючи смартфони, планшети, розумні годинники, телевізори та навіть пристрої зі складними екранами. Це особливо зручно для перевірки роботи застосунків на пристроях із різними екранами, характеристиками та версіями Android.

Емулятор дозволяє запускати застосунок так, ніби він працює на реальному пристрої, що включає перевірку інтерфейсу, функціональності та інтеграції з операційною системою. Він підтримує широкий діапазон версій Android, від старих до найновіших, що допомагає переконатися, що застосунок працює стабільно на різних версіях операційних систем. Android Emulator підтримує імітацію таких функцій, як камера, GPS, акселерометр,

гіроскоп, датчики відбитків пальців, датчики наближення та інші, що дає змогу тестувати роботу застосунків, які залежать від апаратних можливостей пристрою.

Крім того, Android Emulator може використовуватися разом із Visual Studio Code, що дає змогу розробникам тестувати застосунки безпосередньо з редактора. Завдяки підтримці розширень та інтеграції з емулятором, налагодження, запуск і тестування застосунків у VS Code стає зручним та ефективним. Використання емулятора прискорює процес тестування, оскільки не потрібно постійно переключатися між реальними пристроями, а гнучкість налаштувань дає змогу швидко створювати різні конфігурації для тестування застосунків у різних умовах.

2.1.5 Система контролю версій Git

Системи контролю версій – це спеціалізовані програмні інструменти, які дозволяють зберігати різні версії файлів проєкту та повертатися до них у разі необхідності. Вони спрощують процес організації командної роботи над програмним забезпеченням, надаючи можливість створення окремих гілок для кожного розробника або функціоналу та подальшого об’єднання цих гілок у загальний проєкт за потреби.

Git є однією з найпопулярніших систем контролю версій, яка дозволяє відстежувати всі зміни, внесені до файлів проєкту, протягом його розробки [11]. Завдяки розподіленій архітектурі, кожна копія репозиторію містить всю історію, що робить Git ідеальним інструментом для спільної роботи, де учасники можуть паралельно працювати над різними аспектами проєкту без ризику конфліктів.

GitHub, у свою чергу, є популярною платформою для хостингу репозиторіїв Git [12], яка надає зручний веб інтерфейс для управління репозиторіями, спрощуючи роботу з кодом і координацію між учасниками команди. Крім основних функцій зберігання та керування кодом, GitHub

пропонує додаткові можливості: відстеження помилок, організацію завдань та створення документації для проєктів.

У комплексі Git та GitHub стають незамінними інструментами для розробників, які працюють над командними проєктами, забезпечуючи зручний доступ до інструментів управління кодом, співпраці та ефективного контролю за змінами.

2.2 Архітектурні підходи та принципи

2.2.1 Принцип Clean Architecture

Clean Architecture, є однією з найбільш ефективних методологій проєктування програмної архітектури, запропонованою Робертом Мартіном. Вона має на меті забезпечити створення гнучких, добре тестованих та простих у супроводі систем, де доменна логіка – основна бізнес-частина програми залишається незалежною від технічних деталей реалізації, таких як користувацький інтерфейс, бази даних або сторонні фреймворки. Головним принципом цієї архітектури є правило залежностей, згідно з яким внутрішні шари системи не повинні залежати від зовнішніх, що забезпечує чітку ієрархію та модульність структури. Такий підхід дозволяє легко адаптувати програму до змін технологій, не порушуючи його фундаментальної логіки [13].

Архітектура будується у формі концентричних кіл, кожне з яких виконує окрему роль. Найвнутрішніший шар містить сутності, які представляють об'єкти предметної області та містять загальні бізнес-правила. Наступний шар реалізує конкретні сценарії використання системи. Третій рівень займається адаптацією даних для взаємодії між внутрішньою логікою та зовнішніми компонентами, такими як інтерфейс або система зберігання даних. Останній, зовнішній шар, включає всі технічні деталі: графічний інтерфейс, API, бази даних та інші реалізації, через які користувач взаємодіє

з застосунком. Важливим є те, що кожен наступний шар не може безпосередньо впливати на попередній, що гарантує незалежність ключової логіки від технічних реалізацій.

Використання Clean Architecture забезпечує численні переваги, серед яких масштабованість, легкість супроводу, висока тестованість та чітке розділення відповідальностей між компонентами. Завдяки цьому така архітектура особливо корисна при розробці складних програм, де важливе тривале підтримування та регулярні оновлення. У контексті мобільного застосунку використання Clean Architecture дозволяє чітко організувати логіку програми, відокремити доменну модель від зовнішніх сервісів, або серверу застосунку, а також забезпечити гнучкість при подальших змінах [14]. Це спрощує процес розробки, покращує якість коду та забезпечує зручність внесення нових функцій, що є надзвичайно важливим для успішного функціонування сучасного цифрового продукту.

2.2.2 Архітектурний патерн BloC

BLoC (Business Logic Component) – це архітектурний патерн, який використовується для розробки мобільних застосунків з метою розділення бізнес-логіки та UI, що покращує масштабованість, тестованість та підтримку коду [15].

В BLoC архітектурі використовується принцип потоків передачі даних між шарами програми, що дозволяє ефективно керувати станом та оновленнями UI. Потоки дозволяють асинхронно обробляти події та змінювати стан без прямого втручання UI, що підвищує реактивність та динамічність програми. Кожен BLoC можна протестувати незалежно, що значно спрощує процес тестування та покращує якість коду. Крім того, використання потоків дозволяє обробляти зміни в даних та інтерфейси програми в реальному часі, забезпечуючи швидке оновлення UI. Це особливо корисно у програмах з великим обсягом даних або при роботі з зовнішніми

джерелами даних, такими як сервери або бази даних. За допомогою потоків можна ефективно керувати асинхронними операціями, такими як завантаження даних із сервера або обробка запитів користувача.

BLoC сприяє кращому поділу відповідальності між компонентами, покращуючи організацію коду, що дозволяє легше оновлювати, проводити рефакторинг або розширювати програму, додаючи нові модулі або змінюючи існуючі без ризику порушити роботу інших частин системи.

2.3 Використання GPS для навігації та трекінгу

Модуль Global Positioning System (GPS) широко використовується в мобільних застосунках для визначення географічного положення пристрою. Ця функція відкриває величезний спектр можливостей для розробників та забезпечує користувачам доступ до ряду корисних функцій.

GPS є основою для навігаційних застосунків, таких як Google Maps або Waze, дозволяючи користувачам отримувати точні маршрути, інструкції по ходу руху, розрахунок відстаней та часу прибуття.

Технологія працює завдяки сигналам від супутників GPS, які передають точні координати пристрою. Мобільні застосунки можуть інтегрувати GPS за допомогою API, наприклад, Google Maps API для Android та інших операційних систем (рисунок 2.1).

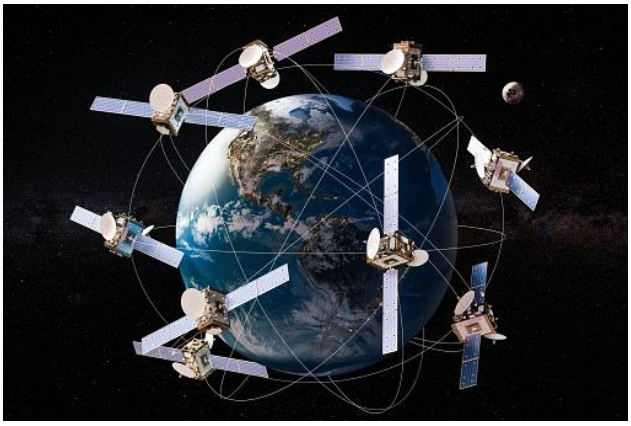


Рисунок 2.1 – Модуль GPS

Використання GPS має численні переваги: точне визначення місцезнаходження, підвищення зручності користування застосунком, широкий спектр застосувань у різних сферах, від транспорту до розваг та здоров'я. Однак є і певні недоліки, такі як значне споживання енергії, зниження точності у міських умовах, ризики для конфіденційності, а також залежність від погодних умов та доступності сигналу.

Розробники повинні враховувати ці особливості, запитувати дозвіл користувача на використання GPS, оптимізувати енерговитрати, комбінувати GPS з іншими джерелами даних для підвищення точності та забезпечувати обробку можливих помилок. Правильне використання GPS дозволяє створювати функціональні та зручні застосунки, що значно покращують якість життя користувачів.

2.4 Використання бібліотек та залежностей

Використання зовнішніх бібліотек (рисунок 2.2) у середовищі Flutter є важливим етапом при створенні сучасних мобільних додатків. Цей підхід дозволяє значно скоротити час розробки, забезпечити високий рівень продуктивності та створити масштабовану, добре структуровану архітектуру.

При формуванні переліку залежностей враховувалися такі критерії: можливість повторного використання коду, модульна побудова системи, чітке розділення логіки та інтерфейсу, а також простота подальшого супроводу й оновлення проєкту.

Нижче наведено основні бібліотеки, які були використані в процесі реалізації мобільного застосунку, та їхнє призначення:

- flutter_bloc, bloc – реалізують архітектурний підхід BLoC для управління станом. Це дозволяє відокремити бізнес-логіку від графічного інтерфейсу, що сприяє покращенню читабельності коду, полегшує тестування та підтримку [16];

- flutter_map [17], latlong2 [18] – забезпечують інтеграцію з

картографічним сервісом OpenStreetMap. За допомогою цих бібліотек реалізовано функціонал динамічних маркерів, спливаючих підказок та обчислень на карті, що є ключовими для мобільного путівника;

- sqflite [19] – використовується як локальна система зберігання даних;
- geolocator [20] – забезпечує отримання поточних координат користувача, що необхідно для відображення об’єктів поруч та побудови маршрутів;
- url_launcher [21] – відповідає за відкриття зовнішніх посилань, що важливо для взаємодії з іншими сервісами;
- http [22] – використовується для обміну даними із сервером за протоколом HTTP, що дозволяє легко інтегрувати зовнішні API;
- android_intent_plus [23] – забезпечує взаємодію з системними функціями Android, що сприяє глибшій інтеграції з операційною системою;
- package_info_plus [24], device_info_plus [25] – надають доступ до службової інформації про застосунок і пристрій, що може використовуватися для діагностики або реалізації додаткової логіки;
- cupertino_icons [26] – набір піктограм, сумісних із дизайном iOS, що важливо для забезпечення кросплатформової послідовності інтерфейсу.

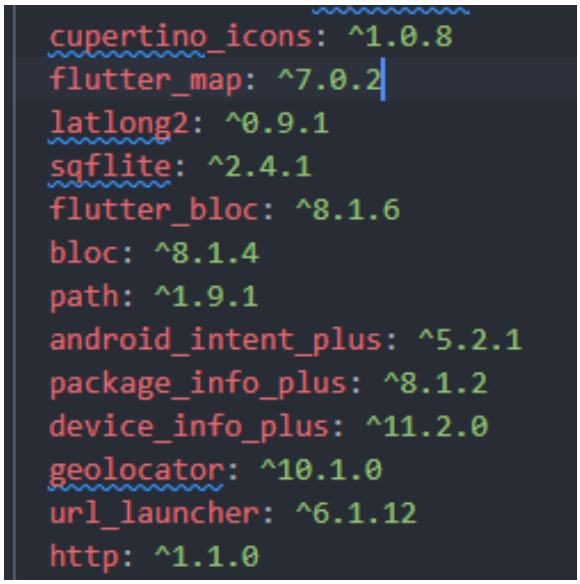


Рисунок 2.2 – Список програмних залежностей, зазначених у конфігураційному файлі pubspec.yaml

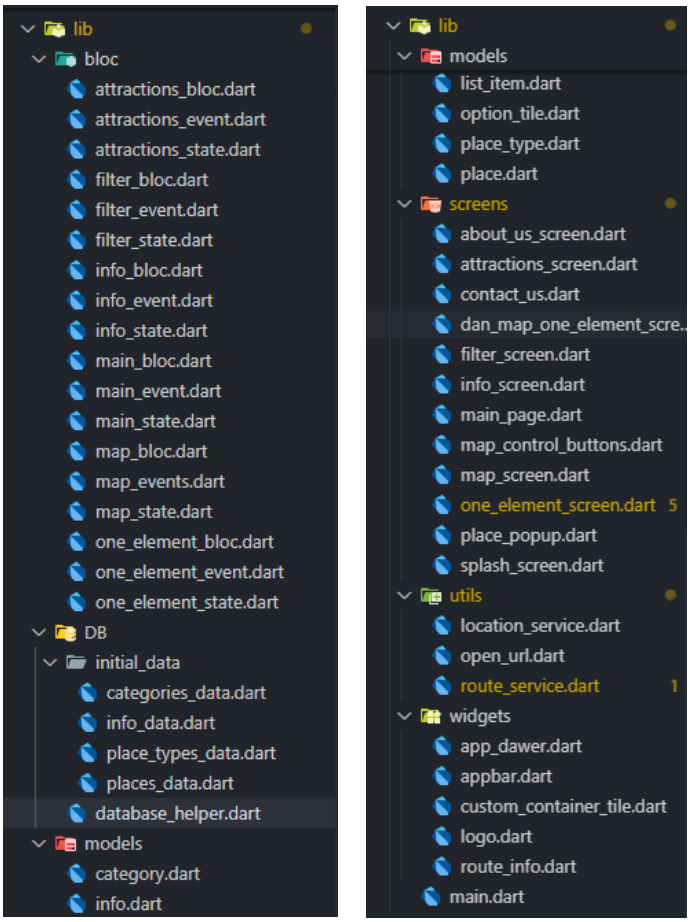
Обраний набір бібліотек забезпечив не лише реалізацію окремих функціональних модулів, але й сформував цілісну архітектурну модель, яка відповідає сучасним стандартам розробки програмного забезпечення. Усі обрані бібліотеки мають активну спільноту, регулярні оновлення, добре документовані API та високий рівень стабільності, що гарантувало безперебійне виконання ключових функцій додатка.

Таким чином, вибір залежностей дозволив не лише втілити задуманий функціонал, але й закласти основу для подальшого розвитку застосунку, забезпечивши гнучкість, стабільність та відповідність сучасним вимогам до розробки мобільного програмного забезпечення.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ЗАСТОСУНКУ

3.1 Структура застосунку

Мобільний застосунок реалізовано відповідно до принципів Clean Architecture, що забезпечує чітке розділення шарів відповідальності, незалежність бізнес-логіки від зовнішніх факторів (таких як UI, мережні запити чи локальна база даних), полегшує тестування та подальший супровід проєкту. Така організація коду дозволяє ефективно масштабувати застосунок, оскільки зміни в одному шарі не впливають на інші частини системи.



а) б)

Рисунок 3.1 – Структура проєкту: а) каталоги bloc та DB; б) каталоги model, screens, utils та widgets та файл main.dart

У якості основного архітектурного патерну використано BLoC (Business Logic Component), який забезпечує керування станами та подіями без прив’язки до конкретних віджетів, що дозволяє уникати зайвої залежності між логікою програми та її графічним інтерфейсом, а також спрощує процес тестування.

Загальна структура застосунку побудована за модульним принципом, де кожен каталог (рисунок 3.1) відповідає за окремий аспект функціональності, що значно полегшує розуміння проєкту новими розробниками та автоматизує процеси розробки, тестування та супроводу. Основні структурні моделі складаються з таких каталогів та файлів:

Каталог bloc – ключова частина застосунка, що відповідає за обробку бізнес-логіки, управління станами та взаємодію між UI та внутрішніми компонентами (рисунок 3.1 а). Кожен функціональний блок а саме: info, one_element, map, filter, attractions, main, має свій власний BLoC, що забезпечує чітке розділення відповідальностей. Для кожного BLoC передбачено три основні файли:

- *_event.dart – визначення можливих подій, які може викликати користувач або система;

- *_state.dart – перелік можливих станів, у які може переходити цей блок;

- *_bloc.dart – сама реалізація логіки, що обробляє події та видає відповідні стани.

Каталог DB – цей модуль реалізує доступ до локальної бази даних, що використовується для зберігання інформації про туристичні об’єкти, категорії, типи місць (рисунок 3.1 а). Використання локальної БД дозволяє користувачам отримувати доступ до контенту навіть без підключення до інтернету. Для роботи з БД використано бібліотеку sqflite, яка забезпечує надійне та ефективне зберігання даних на пристрої користувача. У каталозі знаходяться:

- database_helper.dart – клас, що реалізує ініціалізацію, створення

таблиць, виконання запитів SELECT, INSERT, UPDATE, DELETE;

- підкаталог initial_data/, у якому зберігаються початкові дані: список категорій, інформаційні сторінки, приклади туристичних місць, які завантажуються при першому запуску програми.

Каталог models містить визначення моделей даних, які використовуються в різних частинах застосунку, від отримання інформації з бази даних до представлення їх у вигляді списків або детального перегляду (рисунок 3.1). Всі моделі реалізовані як immutable (незмінні) об'єкти, що забезпечує безпеку при роботі з даними. Серед основних моделей є:

- category.dart, place.dart, info.dart – доменні моделі, що представляють сутності з бази даних;

- list_item.dart, option_tile.dart – допоміжні структури, що використовуються для побудови списків та елементів інтерфейсу.

Модуль screens реалізує всі екрани мобільного застосунку, через які користувач взаємодіє з функціональністю програми (рисунок 3.1 б). Екрани побудовані таким чином, щоб вони могли легко взаємодіяти з BLoC-компонентами, отримувати стани та реагувати на події. Кожен екран відповідає за відображення певного функціоналу та взаємодію з користувачем. Виділено такі екрани:

- головні: main_page.dart, map_screen.dart, attractions_screen.dart, info_screen.dart;

- детальні: dan_map_one_element_screen.dart, place_popup.dart, one_element_screen.dart – екрани, що відображають інформацію про конкретний об’єкт;

- допоміжні: filter_screen.dart, splash_screen.dart, about_us_screen.dart, contact_us.dart.

Бібліотека widgets – це бібліотека перевикористаних віджетів, що забезпечують єдиний стиль інтерфейсу. Використання загальних компонентів дозволяє уникати дублювання коду, полегшити процес зміни дизайну та підвищити читабельність. Серед них:

- appbar.dart, app_drawer.dart – стандартні елементи навігації;

- custom_container_tile.dart, route_info.dart, logo.dart – спеціалізовані віджети, що використовуються для формування списків, маршрутів, логотипів.

Каталог utils містить набір технічних утиліт та сервісів, які забезпечують роботу програми (рисунок 3.1 б). Утиліти абстраговані від інших модулів, що дозволяє легко їх замінювати або розширювати. Ці функції відповідають за додаткову функціональність, яка не пов’язана напряму з логікою окремих екранів, але є важливими для роботи всього застосунку. Зокрема:

- location_service.dart – сервіс, що отримує поточну геолокацію користувача через GPS;

- route_service.dart – реалізація побудови маршруту між точками за допомогою сторонніх API;

- open_url.dart – функціонал для відкриття зовнішніх посилань у браузері або інших програмах.

main.dart – це головний файл програми, що виступає точкою входу. У ньому виконується ініціалізація всіх необхідних BLoC-компонентів, налаштування теми та маршрутизації, а також завантаження початкових екранів (рисунок 3.1 б). Під час запуску мобільного застосунку виконується послідовна ініціалізація таких екранів:

- splash_screen.dart – екран привітання, який відображається під час завантаження мобільного застосунку;

- main_page.dart – головна сторінка;

- map_screen.dart – карта з відображенням туристичних об’єктів.

3.2 Взаємодія компонентів

Взаємодія компонентів у застосунку реалізована у вигляді чітко визначеного потоку даних між шарами архітектури (рисунок 3.2). Інтерфейс

користувача (екрани та віджети) не працює безпосередньо з даними, а взаємодіє з BLoC-компонентами, які отримують події від UI та повертають відповідні стани з підготовленими даними. Ці компоненти логіки, у свою чергу, звертаються до бази даних через спеціальний сервіс database_helper, зчитуючи або оновлюючи інформацію. Уся інформація в застосунку передається у вигляді об’єктів, визначених у моделях (models/*.dart), що забезпечує єдину структуру даних на всіх рівнях. Утиліти (utils/*.dart) – такі як сервіси геолокації чи маршрутизації – також викликаються з BLoC або UI та надають допоміжний функціонал, що не належить до основної бізнес-логіки. Завдяки такій взаємодії забезпечується централізоване керування станом, уникнення дублювання коду та максимальна ізольованість модулів.

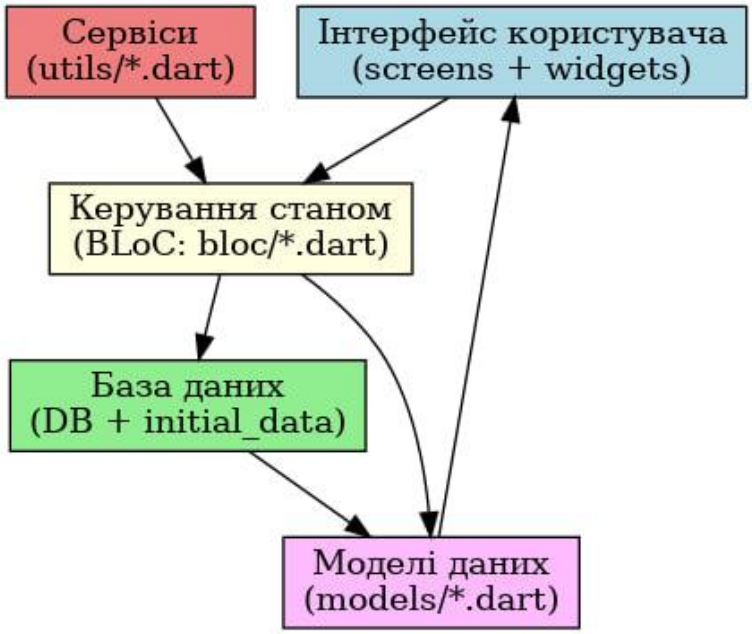


Рисунок 3.2 – Схема взаємодії основних компонентів застосунку

3.3 Основні елементи та функції застосунку

3.3.1 Клас CustomContainerTile

Клас CustomContainerTile – це віджет користувача, що реалізує універсальний елемент списку з іконкою зліва, текстовим заголовком та

стрілкою справа (лістинг 3.1). Він використовується на головному екрані MainPage для побудови зрозумілого та візуально послідовного інтерфейсу навігації. Основна перевага цього компонента – забезпечення єдиного стилю та можливість повторного використання в різних частинах застосунку.

Віджет має параметри title, leadingIcon, trailingIcon та onTap, що дозволяє динамічно задавати назву елементу, відповідні іконки та функцію, яка виконується при натисканні.

Візуально компонент оформлений із застосуванням тіні BoxShadow та кольорового фону, що підвищує естетичність та зручність взаємодії користувача з елементами інтерфейсу.

Лістинг 3.1 – Клас CustomContainerTile

```
class CustomContainerTile extends StatelessWidget {
  final String title; final IconData leadingIcon;
  final IconData trailingIcon;
final VoidCallback onTap;
  @override Widget build(BuildContext context) {
    final theme = Theme.of(context);
    return Container(height: 60,
      margin: EdgeInsets.symmetric(vertical: 8.0),
      decoration: BoxDecoration(
        color: Color.fromARGB(255, 208, 233, 247),
        boxShadow: [BoxShadow(
            color: Colors.black26,offset: Offset(0, 3),
            blurRadius: 5.0,spreadRadius: 1.0,),],),
      child: Center(child: ListTile(
          title: Text(
title, style: theme.textTheme.bodyMedium),
          leading: Icon(leadingIcon),
          trailing: Icon(trailingIcon), onTap: onTap,),),);}}
```

3.3.2 Метод _onLoadPlaceTypes

Метод _onLoadPlaceTypes виконує обробку події LoadPlaceTypes, яка відповідає за завантаження типів місць із локальної бази даних. Він використовується у фільтраційному екрані для визначення, які категорії об'єктів слід показувати користувачу на основі контексту (наприклад,

«Карта», «Визначні місця», «Ресторани та бари»). У процесі виконання методу надсилається стан FilterLoading, після чого здійснюється отримання усіх типів місць із бази даних за допомогою класу DatabaseHelper (лістинг 3.2). Далі відбувається фільтрація даних відповідно до назви категорії, переданої у події. На основі відфільтрованих даних формується мапа filterSelections, яка містить інформацію про те, які фільтри наразі активні. Потім визначається, чи всі фільтри обрані allSelected, та в разі успішного завершення процесу надсилається стан FilterLoaded. Якщо під час виконання виникає помилка, генерується стан FilterError.

Лістинг 3.2 – Метод _onLoadPlaceTypes

```
void _onLoadPlaceTypes(
LoadPlaceTypes event,
 Emitter<FilterState> emit) async {
  emit(FilterLoading());
 try {final data = await
DatabaseHelper.instance.getPlaceTypes();
    List<Map<String, dynamic>> filtered = data;
    if (event.title.contains('Визначні')) {
      filtered = data.where((e) =>
        ['Парки', 'Музеї', 'Храми', 'Театри',
'Пам\'ятники'].contains(e['name'])).toList();
    } else if (event.title == 'Ресторани та бари') {
      filtered = data.where((e) =>
        ['Ресторани', 'Бари та паби', 'Кафе', 'Fast
food'].contains(e['name'])).toList();}
final sel = {for (var p in filtered)
 p['id']: event.currentFilters[p['id']] ?? false};
    emit(FilterLoaded(
filtered, sel,
 sel.values.every((v) => v)));}
 catch (_) {emit(FilterError('Failed to load place types'));}}
```

3.3.3 Методи _zoomIn та _zoomOut

Методи _zoomIn та _zoomOut реалізують зміну масштабу карти – відповідно збільшення та зменшення. Перед виконанням кожен з методів перевіряє, чи карта готова до взаємодії isMapReady. Якщо умова виконується, поточний рівень масштабу currentZoom зчитується з контролера карти

mapController. У випадку методу _zoomIn, якщо масштаб ще не досяг максимального значення maxZoom, він збільшується на фіксовану величину zoomStep, з обмеженням у межах допустимих значень за допомогою clamp (лістинг 3.3). Аналогічно, метод _zoomOut зменшує масштаб, якщо поточне значення перевищує мінімальний допустимий рівень minZoom. Обидва методи використовують функцію move, щоб оновити положення та масштаб карти.

Лістинг 3.3 – Методи _zoomIn та _zoomOut

```
void _zoomIn() {if (!isMapReady) return;final currentZoom =
mapController.camera.zoom;
 if (currentZoom < maxZoom) {
mapController.move(
mapController.camera.center,(currentZoom +
zoomStep).clamp(minZoom, maxZoom),);}}
void _zoomOut() {
if (!isMapReady) return; final currentZoom =
mapController.camera.zoom;
 if (currentZoom > minZoom)
{mapController.move(mapController.camera.center,(currentZoom -
zoomStep).clamp(minZoom, maxZoom),);}}
```

3.3.4 Клас DatabaseHelper

Клас DatabaseHelper відповідає за взаємодію з локальною базою даних SQLite, яка використовується у застосунку для зберігання даних про категорії, типи місць, самі місця, а також інформаційні статті (лістинг 3.4). Це синглтон-клас, він має приватний конструктор та єдину точку доступу через DatabaseHelper.instance.

Лістинг 3.4 – Клас DatabaseHelper

```
class DatabaseHelper {
  static final instance = DatabaseHelper._privateConstructor();
  static Database? _db;DatabaseHelper._privateConstructor();
  Future<Database> get database async {if (_db != null) return
_db!;final path = join(await getDatabasesPath(), 'guide.db');
    _db = await openDatabase(path,version: 1,onCreate:
_onCreate,onOpen: (db) => _populateInitialData(db),);return
```

```
_db!;}
  Future<void> _onCreate(Database db, int version) async {
    await db.execute('''CREATE TABLE Categories (id INTEGER
PRIMARY KEY, name TEXT);''');
    await db.execute('''CREATE TABLE PlaceTypes (id INTEGER
PRIMARY KEY, name TEXT);''');
    await db.execute('''CREATE TABLE Places (...);''');
    await db.execute('''CREATE TABLE Info (...);''');}
```

Під час ініціалізації база створюється або відкривається з файлу guide.db. Метод _onCreate виконує створення таблиць Categories, PlaceTypes, Info та Places (таблиця 3.1). Після відкриття викликається метод _populateInitialData, який перевіряє наявність початкових даних і, за потреби, заповнює таблиці інформацією з файлів categoriesData, placeTypesData, placesData та infoData (лістинг 3.5). Клас також надає методи для отримання даних:

- getPlaces() – повертає всі місця у вигляді списку ListItem;
- getPlaceTypes() – повертає всі типи місць у вигляді списку мап;
- getInfoById(id) – повертає один запис типу Info за ID;
- getPlaceById(id) – повертає конкретне місце за ID.

Таблиця 3.1 – Структура таблиці Places

Поле	Тип даних	Опис
id	INTEGER (PK)	Унікальний ідентифікатор місця
category_id	INTEGER (FK)	Посилання на категорію Categories
place_type_id	INTEGER (FK)	Посилання на тип місця PlaceTypes
name	TEXT	Назва місця
description	TEXT	Опис
address	TEXT	Адреса місця
latitude	REAL	Географічна широта
longitude	REAL	Географічна довгота
image_url	TEXT	Посилання на зображення
contact_info	TEXT	Контактна інформація (телефон, сайт)

Лістинг 3.5 – Метод _populateInitialData

```
Future<void> _populateInitialData(Database db) async {
    if ((await db.rawQuery('SELECT COUNT(*) FROM
Categories')).first['COUNT(*)'] == 0) {
      for (var c in categoriesData) await
db.insert('Categories', c);}}
  Future<List<ListItem>> getPlaces() async {
    final data = await (await database).query('Places');
    return data.map((e) => ListItem.fromMap(e)).toList();}
  Future<List<Map<String, dynamic>>> getPlaceTypes() async =>
      (await (await database).query('PlaceTypes'));
  Future<Info> getInfoById(int id) async {
    final data = await (await database).query('Info', where: 'id
= ?', whereArgs: [id]); if (data.isNotEmpty) return
Info.fromMap(data.first);throw Exception('Info not found');}
  Future<ListItem?> getPlaceById(int id) async {
    final data = await (await database).query('Places', where:
'id = ?', whereArgs: [id]);return data.isNotEmpty ?
ListItem.fromMap(data.first) : null;}}
```

3.3.5 Клас MainBloc

Клас MainBloc реалізує бізнес-логіку навігації у програмі з використанням патерну BLoC (лістинг 3.6). MainBloc відповідає за обробку подій, пов’язаних з переходом між головними екранами застосунку та відкриттям зовнішніх посилань. Основні події, що обробляються:

- NavigateToMap – здійснює перехід на екран карти (/map);
- NavigateToAttractions – відкриває екран з переліком місць за категоріями (/attractions);
- NavigateToInfo – відкриває детальну інформацію за певним ідентифікатором (/info);
- OpenUrl – відкриває зовнішнє посилання за допомогою утиліти openUrl().

Обробка подій відбувається через методи _onNavigateToMap, _onNavigateToAttractions, _onNavigateToInfo, _onOpenUrl, які викликають функцію emit для оновлення стану MainState та ініціації навігації або дії. Цей підхід відокремлює логіку від інтерфейсу, забезпечуючи гнучкість, масштабованість та зручність тестування застосунку.

Лістинг 3.6 – Клас MainBloc

```
class MainBloc extends Bloc<MainEvent, MainState> {
MainBloc() : super(MainInitial()) {
on<NavigateToMap>(_onNavigateToMap);
on<NavigateToAttractions>(_onNavigateToAttractions);
on<NavigateToInfo>(_onNavigateToInfo);
on<OpenUrl>(_onOpenUrl);}
void _onNavigateToMap(NavigateToMap e, Emitter<MainState> emit)
{emit(MainNavigation('/map', arguments: {'title': 'Карта'}));}
void _onNavigateToAttractions(NavigateToAttractions e,
Emitter<MainState> emit) {
emit(MainNavigation('/attractions', arguments: {'title':
e.title}));} void _onNavigateToInfo(NavigateToInfo e,
Emitter<MainState> emit) {
emit(MainNavigation('/info', arguments: {'infoId': e.infoId}));}
void _onOpenUrl(OpenUrl e, Emitter<MainState> emit)
{openUrl(e.url);}}
```

3.3.6 Клас _MapScreenState

Клас _MapScreenState реалізує основну логіку роботи екрану карти в мобільному застосунку. Він відповідає за відображення карти з позначенням визначних місць, роботу з геолокацією користувача, побудову маршрутів до обраних об’єктів, керування станом фільтрації, та інтерфейсні елементи керування (лістинг Б.1). Ключові можливості класу:

- ініціалізація карти та геопозиції: під час запуску екрану завантажуються дані на карту та визначається місце перебування користувача (рисунок 3.3 а);

- відстеження локації: реалізовано підписку на зміну координат, а також можливість вмикати або вимикати відстеження руху;

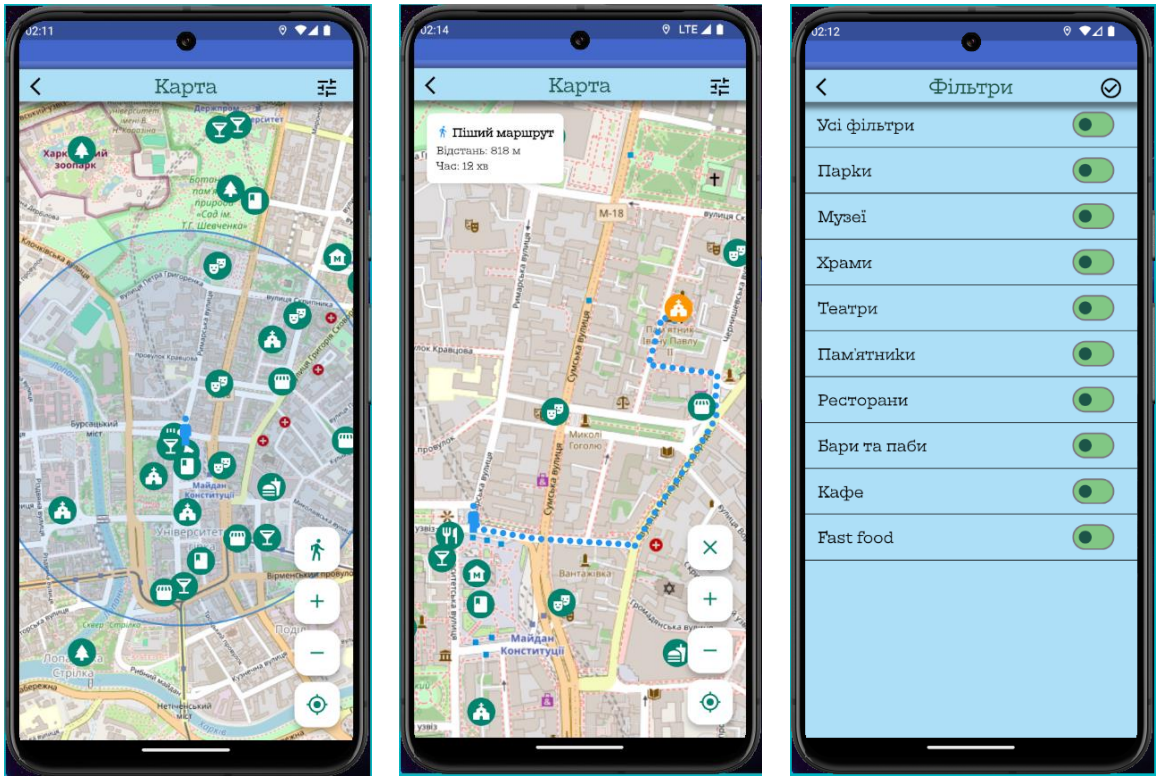
- робота з маршрутами: за допомогою служби RouteService будується пішохідний маршрут від поточної позиції користувача до обраної точки (рисунок 3.3 б);

- фільтрація об’єктів: користувач може обрати типи об'єктів для відображення на мапі, фільтруючи маркери (рисунок 3.3 в);

- динамічне відображення: залежно від стану завантаження відображається прогрес, повідомлення про помилки або сам інтерфейс карти

з накладеннями;

- інтерактивні елементи: кнопки для збільшення/зменшення масштабу, центрування карти, запуску побудови маршруту (рисунок 3.3 а, б).



а) б) в)

Рисунок 3.3 – Екран «Карта»: а) початковий стан карти; б) стан карти при побудові маршруту; в) вкладка «Фільтри » для карти

Метод getIconPath відповідає за визначення візуального представлення типу місця на карті за допомогою відповідної іконки з бібліотеки Icons у Flutter (лістинг 3.7). Цей метод отримує ідентифікатор типу місця (наприклад, парк, музей, кафе тощо) та повертає відповідну іконку, яка потім використовується при візуалізації маркерів на мапі (рисунок 3.3 а, б). Це дозволяє користувачу швидко візуально розпізнати тип об’єкта на карті.

Лістинг 3.7 – Метод getIconPath

```
Object getIconPath(int placeTypeId) {
switch (placeTypeId) {
case 1: return Icons.park; //парк
```

```
case 2: return Icons.museum;//музей
case 3: return Icons.church;//храм
case 4: return Icons.theater_comedy;//театр
case 5: return Icons.book;//пам'ятка-історичний об'єкт
case 6: return Icons.restaurant;//ресторан
case 7: return Icons.local_bar;
case 8: return Icons.storefront_rounded;/кафе
case 9: return Icons.fastfood;//заклад швидкого харчування
default: return Icons.location_on;//загальна іконка}}
```

3.3.7 Клас PlacePopopWithRoute

Клас PlacePopupWithRoute відповідає за відображення діалогового вікна з детальною інформацією про обране місце на карті (рисунок 3.4). Основна мета цього компонента – надати користувачеві швидкий доступ до короткого опису, зображення, адреси, контактної інформації, а також можливості побудови маршруту (лістинг Б.2).

Рисунок 3.4 – Діалогове вікно з інформацією про обране місце на карті

У діалозі відображається заголовок місця та його зображення, при цьому передбачено резервне зображення у разі помилки завантаження. Також

виводиться короткий опис з можливістю переходу до повного перегляду на окремому екрані OneElementScreen.

Якщо присутні адреса або контактні дані, вони теж відображаються у відповідному форматі. Внизу діалогового вікна розташовані дві інтерактивні кнопки: «Закрити» – для закриття вікна з можливістю скасування побудови маршруту, та «Прокласти маршрут» – для виклику відповідного callback-функціоналу і побудови маршруту до місця.

3.3.8 Клас RouteService

Клас RouteService відповідає за побудову пішохідного маршруту між двома географічними координатами, використовуючи публічний API сервісу OSRM (Open Source Routing Machine). Він реалізує HTTP-запити до маршрутизатора через безпечний протокол HTTPS і містить логіку обробки помилок, повторних спроб та парсингу отриманої відповіді (лістинг Б.3).

Метод buildWalkingRoute намагається побудувати маршрут, перевіряючи кілька профілів (foot, walking) і автоматично повторює запити у разі тимчасових збоїв. Відповідь API аналізується методом _parseRouteResponse, який отримує геометрію маршруту та обчислює відстань та тривалість (лістинг 3.8), яка далі буде використовуватися в класі RouteInfoWidget. Клас також містить допоміжні методи для перевірки коректності координат, побудови URL-запиту та форматування отриманих значень відстані й часу для зручного відображення у інтерфейсі користувача.

Лістинг 3.8 – Метод формування відстані та часу для відображення на екрані

```
static String formatDistance(double? distance) {
if (distance == null || distance <= 0) return '0 м';
return distance < 1000? '${distance.round()} м'
: '${(distance / 1000).toStringAsFixed(1)} км';}
static String formatDuration(double? duration) {
if (duration == null || duration <= 0) return '0 хв';
final minutes = ((6.8 * duration) / 60).round();
if (minutes < 60) return '$minutes хв';
final h = (minutes / 60).floor();
```

```
final m = minutes % 60;
return m == 0 ? '${h}г' : '${h}г ${m}хв';}}
```

3.3.9 Клас RouteInfoWidget

Клас RouteInfoWidget реалізує віджет, який відображає коротку інформацію про побудований маршрут у вигляді невеликої картки поверх карти (рисунок 3.5). Він показує тип маршруту (піший), загальну відстань та приблизний час у дорозі (лістинг 3.9). Віджет автоматично форматуватиме числові значення за допомогою методів RouteService.formatDistance та RouteService.formatDuration. Компонент стилізовано відповідно до загального дизайну з тінями, заокругленими краями та піктограмою.

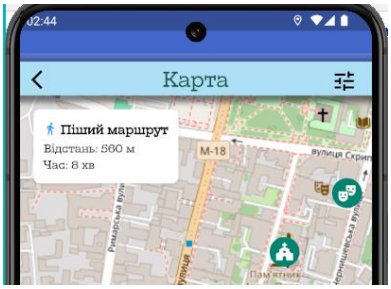


Рисунок 3.5 – Інформаційний віджет про маршрут

Лістинг 3.9 – Клас RouteInfoWidget

```
class RouteInfoWidget extends StatelessWidget {final double
distance;final double duration;const RouteInfoWidget({required
this.distance,required this.duration,Key? key,}) : super(key:
key);
@override Widget build(BuildContext context) {return
Positioned(top: 16,left: 16,child: Container(padding:
EdgeInsets.all(12),decoration: BoxDecoration(color:
Colors.white,borderRadius: BorderRadius.circular(8),boxShadow:
[BoxShadow(color: Colors.black12,blurRadius: 4,offset: Offset(0,
2),),],),child: Column(crossAxisAlignment:
CrossAxisAlignment.start,mainAxisSize:
MainAxisSize.min,children: [Row(children:
[Icon(Icons.directions_walk, color: Colors.blue, size:
16),SizedBox(width: 4),Text('Піший маршрут',style:
TextStyle(fontWeight: FontWeight.bold, fontSize:
15),),],),SizedBox(height: 4),Text('Відстань:
${RouteService.formatDistance(distance)}',style:
TextStyle(fontSize: 13),),Text('Час:
```

```
${RouteService.formatDuration(duration)}',style:
TextStyle(fontSize: 13),),],),),);}}
```

3.3.10 Клас Locationservice

Клас LocationService відповідає за отримання та моніторинг геолокації користувача в застосунку (лістинг Б.4). Він забезпечує як однократне визначення поточного місця розташування, так і безперервне відстеження координат за допомогою Geolocator. Основна мета – централізоване надання координат через потік Stream<LatLng> іншим компонентам, які потребують геоданих (наприклад, карті або маршрутам). Ключові можливості:

- отримання поточного місцезнаходження користувача;
- постійне оновлення координат через потоковий інтерфейс locationStream (лістинг 3.10);
- обробка відмов доступу до локації або помилок з GPS;
- фонове оновлення координат за таймером, якщо потік Geolocator не працює (лістинг 3.11);
- повна зупинка та очищення ресурсів через методи stopLocationTracking та dispose (лістинг 3.12).

Лістинг 3.10 – Метод startLocationTracking

```
void startLocationTracking() {
if (_positionStream != null) return;
_positionStream = Geolocator.getPositionStream(
locationSettings: LocationSettings(
accuracy: LocationAccuracy.high,
distanceFilter: 10,),).listen((pos) =>
_locationController?.add(LatLng(pos.latitude, pos.longitude)),
onError: (e) {print("Location stream error:
$e");_startLocationTimer();},);}
```

Лістинг 3.11 – Метод _startLocationTimer

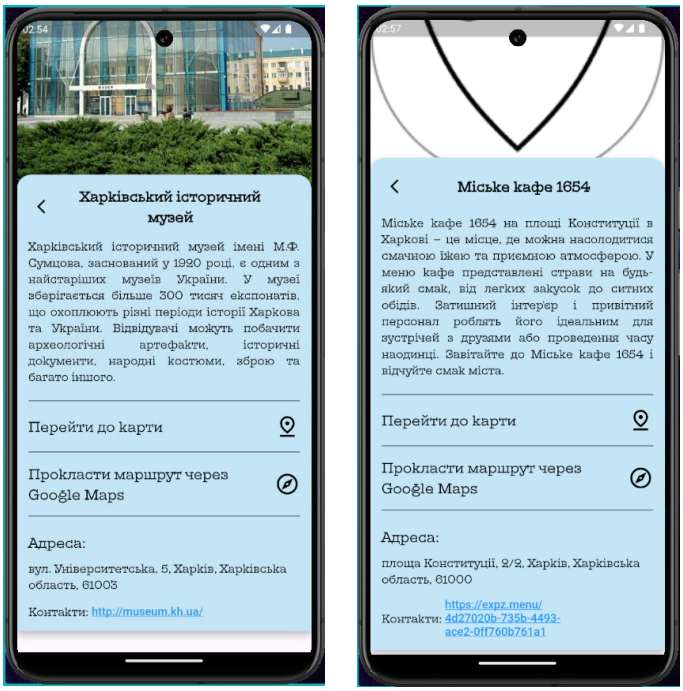
```
void _startLocationTimer() {
_locationTimer?.cancel();
_locationTimer = Timer.periodic(
Duration(seconds: 10),(_) => getCurrentLocation(),);}
```

Лістинг 3.12 – Методи stopLocationTracking та dispose

```
void stopLocationTracking() {_positionStream?.cancel();
_positionStream = null;_locationTimer?.cancel();
_locationTimer = null;}
void dispose() {stopLocationTracking();
_locationController?.close();_locationController = null;}
```

3.3.11 Клас OneElementScreen

Клас OneElementScreen відповідає за відображення повної інформації про обране місце на окремому екрані (рисунок 3.6). Цей екран надає користувачу розширений опис елемента, фотографію, адресу, контакти, а також інтерактивні функції, пов’язані з геолокацією (лістинг Б.5).



а) б)

Рисунок 3.6 – Екрани окремих місць: а) Екран «Харківський історичний музей»; б) Екран «Міське кафе 1654»

На основі Bloc архітектури, екран спочатку завантажує інформацію про елемент за його id через подію FetchOneElement, після чого відображає результат:

- велике зображення (або резервне, якщо помилка);
- заголовок і опис місця;
- адреса та контактні дані (з підтримкою клікабельних посилань);
- кнопки для:
  1) переходу на карту для взаємодії з цим місцем;
  2) побудови маршруту через Google Maps.

3.3.12 Клас AppDrawer

Клас AppDrawer, який реалізує віджет користувача бокового меню у мобільному застосунку (рисунок 3.7 а), забезпечує навігацію між основними розділами програми та містить пункти меню: «Зв’язок з нами» та «Про нас» (лістинг 3.13). Основна область бокового меню містить логотип та список пунктів, кожен елемент має іконку, текстовий заголовок та обробник події натискання.

При виборі пункту «Зв’язок з нами» запускається email-клієнт за допомогою функції launchEmailGmail(context), а при виборі «Про нас» відбувається перехід на екран AboutUsScreen (рисунок 3.7 б), в якому знаходиться інформація про розробника та мета проєкту.

Лістинг 3.13 – Клас AppDrawer

```
class AppDrawer extends StatelessWidget {
@override
Widget build(BuildContext context) {final theme =
Theme.of(context);return Drawer(child: Column(children:
[Container(height: 80,color: Color.fromARGB(221, 66, 102, 202),
//0xFF009246),Expanded(child: Container(color:
Color.fromARGB(48, 52, 184, 245),   //0xFFF6FFED child:
Column(children: [const
DrawerHeader(padding:EdgeInsets.only(bottom: 40),decoration:
BoxDecoration(color: Color.fromARGB(46, 16,
147,207),),child:Logo(),),Expanded(child: ListView(padding:
EdgeInsets.zero,children: [ListTile(leading:
Icon(Icons.border_color_outlined,size: 30, color: const
Color.fromARGB(255, 46, 46, 46)), title: Text('Зв\'язок з
нами',style: theme.textTheme.bodySmall),onTap: ()
{launchEmailGmail(context);},),ListTile(leading:
```

```
Icon(Icons.info_outline,size: 30, color:
Colors.black),title:Text('Про нас', style:
theme.textTheme.bodySmall),onTap: ()
{Navigator.push(context,MaterialPageRoute(builder: (context) =>
AboutUsScreen(),),);},),],),),],),),),],),);}}
```

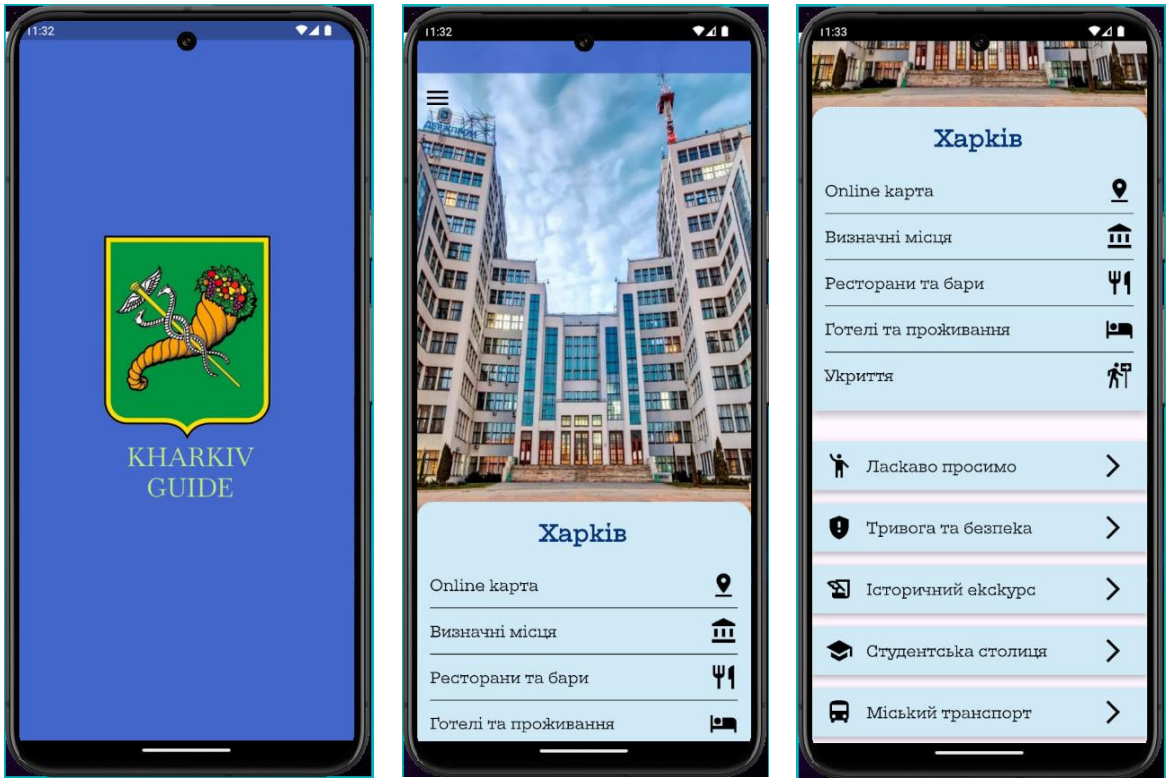
а) б)

Рисунок 3.7 – Екрани «Бокового віджету»: а) початкове бокове меню; б) пункт меню «Про нас»

4 ІНСТРУКІЯ КОРИСТУВАЧА

Щоб запустити мобільний застосунок потрібно натиснути на іконку застосунку з назвою «Kharkiv Guide». Після цього відкриється стартовий екран (splash screen), який сигналізує про завантаження програми (рисунок 4.1 а). Через кілька секунд користувач потрапляє на головний екран (рисунок 4.1 б, в), який є центральною точкою взаємодії з застосунком.

На головному екрані зображено список пунктів кожен з яких веде до певного розділу застосунку. Список включає пункти з назвою: «Online карта», «Визначні місця», «Ресторани та бари», «Готелі та проживання», «Ласкаво просимо», «Історичний екскурс», «Студентська столиця» та інші (рисунок 4.1 в). Користувач може обрати будь-який пункт зі списку для перегляду детальної інформації, навігації містом або взаємодії з картою.

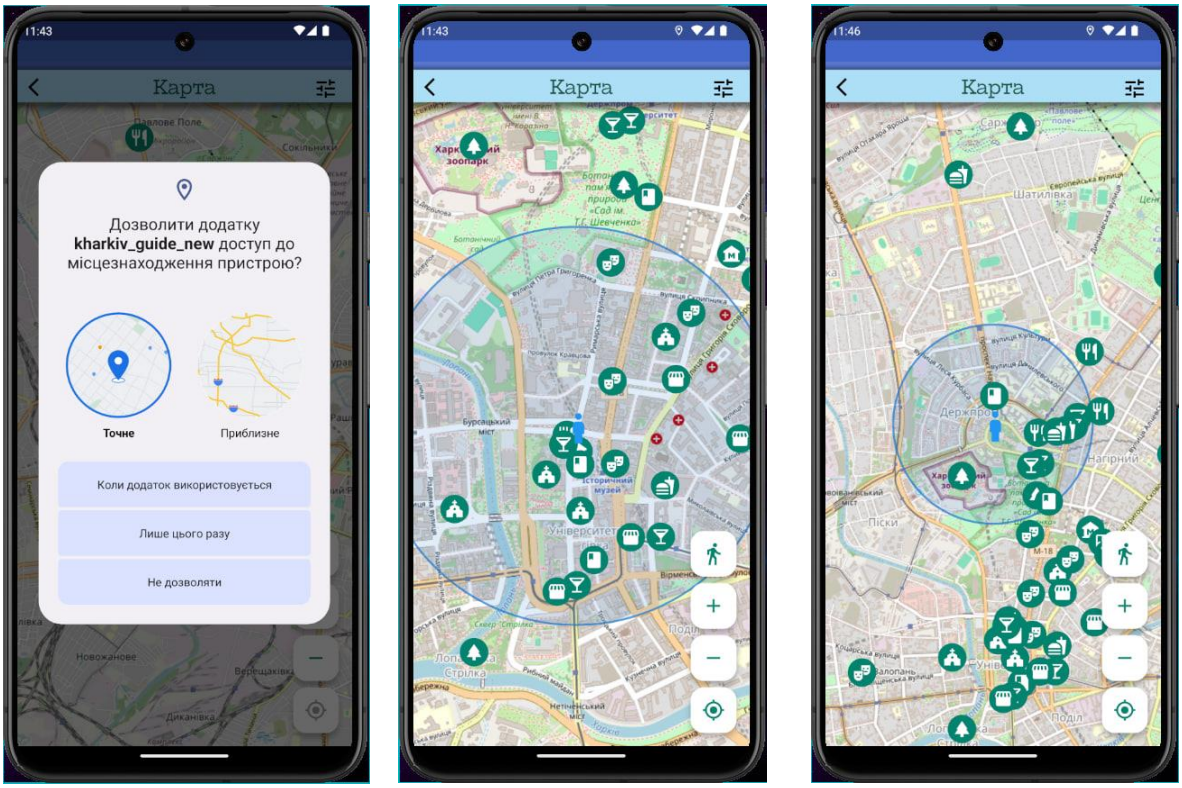


а) б) в)

Рисунок 4.1 – Екрани під час запуску застосунку: а) стартовий екран; б) перша частина «Головного екрану»; в) продовження «Головного екрану»

4.1 Online карта

Після натискання на плитку «Online карта», у разі першого запуску застосунку, система запропонує користувачеві надати доступ до геолокації пристрою (рисунок 4.2 а). Для коректної роботи функцій, пов’язаних із визначенням місцезнаходження та побудовою маршрутів, необхідно надати дозвіл. Якщо такий дозвіл уже надано, користувач одразу переходить до екрану «Карта». На цьому екрані розміщено інтерактивну карту міста Харків, де нанесено маркери, що позначають визначні місця, заклади харчування та відпочинку, а також поточне місцезнаходження користувача. Для кожної категорії передбачено унікальний дизайн маркера, що полегшує візуальне сприйняття (рисунок 4.2 б, в).



а) б) в)

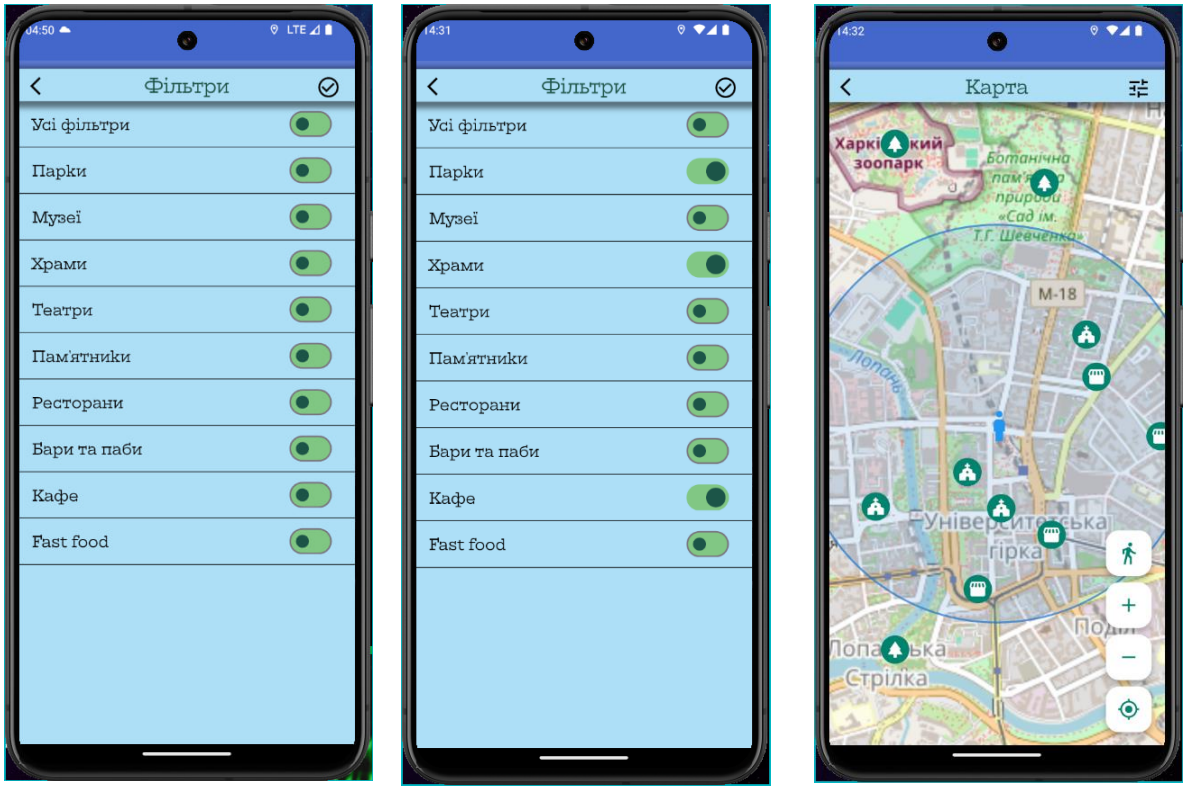
Рисунок 4.2 – Екран «Карта»: а) вікно щоб надати доступ до геолокації пристрою; б) початковий маштаб карти »; в) зміненний маштаб карти

Місцезнаходження користувача позначено синьою іконкою у вигляді фігури людини, навколо якої відображається радіус у 750 метрів – зона для

орієнтації та зручної навігації. Користувач може взаємодіяти з картою за допомогою жестів (перетягування, масштабування) або через навігаційні кнопки, розташовані у нижньому правому куті екрана (рисунок 4.2 б, в). До основних кнопок належать:

- центрування карти відповідно до поточного розташування користувача;
- зменшення та збільшення масштабу;
- побудова маршруту до вибраного місця та скасування маршруту, якщо він більше не потрібен.

У правому верхньому куті екрана розміщена кнопка фільтрації (рисунок 4.2 б, в), яка дозволяє налаштувати відображення об’єктів на карті. При її натисканні з’являється вікно з варіантами фільтрів, де користувач може обрати категорії локацій наприклад, визначні місця або заклади харчування та відпочинку, які він хоче бачити на карті (рисунок 4.3 а, б).



а) б) в)

Рисунок 4.3 – Екран «Фільтри»: а) початковий стан; б) активовані фільтри «Парки», «Храми», «Кафе»; в) екран «Карта» після виконання фільтрів

У застосунку реалізовано наступні фільтри: «Усі фільтри», «Парки», «Музеї», «Храми», «Театри», «Пам’ятники», «Ресторани», «Бари та паби», «Кафе», «Fast Food».

Активувати фільтр можна, натиснувши перемикач поруч із назвою відповідної категорії. Щоб застосувати зміни, необхідно підтвердити вибір, натиснувши іконку галочки у верхній частині вікна (рисунок 4.3 б, в).

За замовчуванням при відкритті екрана «Карта» відображаються всі місця. Якщо користувач раніше застосовував певні фільтри, але бажає знову бачити повний список об’єктів – достатньо увімкнути перемикач біля фільтра «Усі фільтри». Це дозволяє швидко повернути повну картину місць без скидання налаштувань вручну.

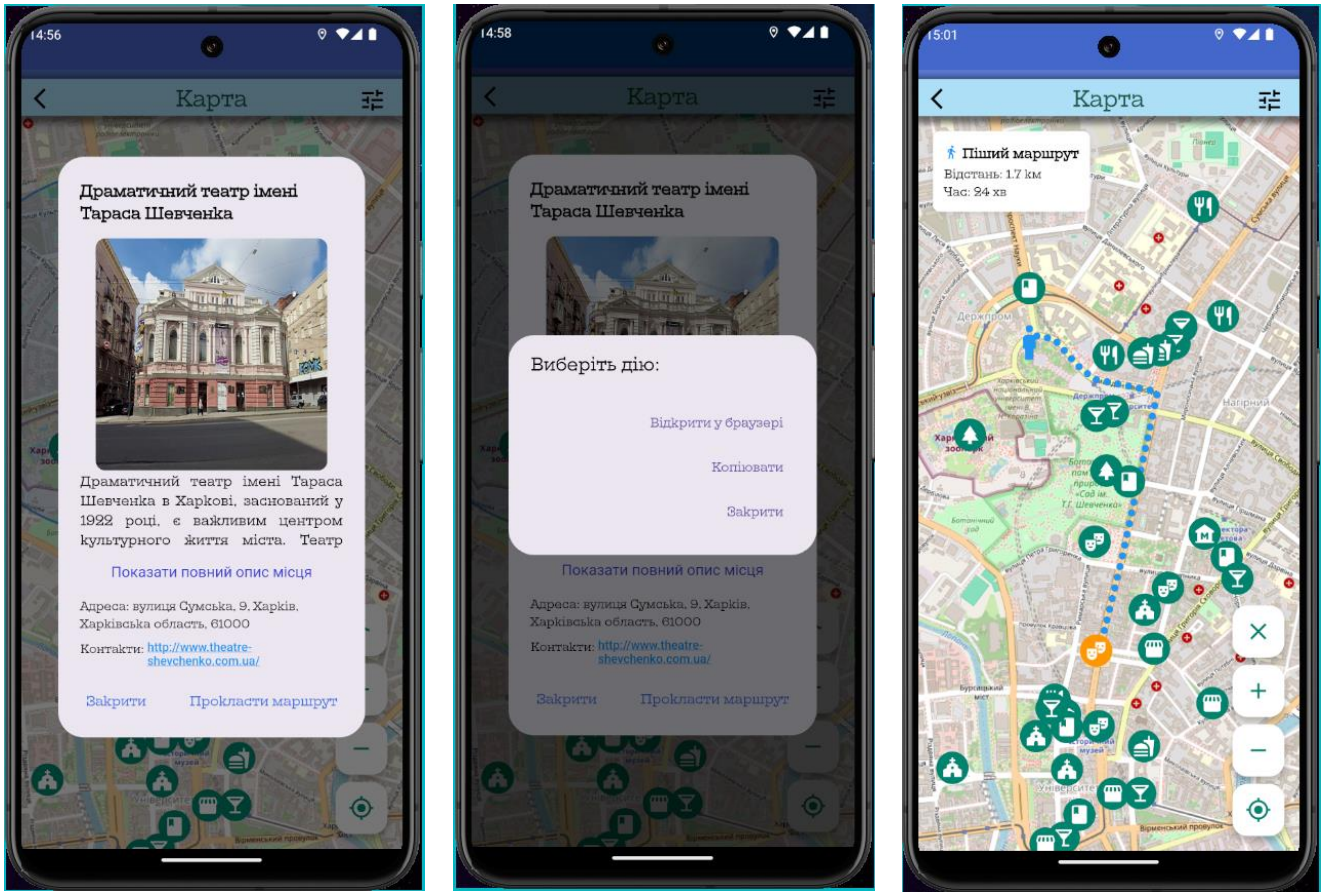
Коли користувач натискає на маркер певного місця на карті, з’являється діалогове вікно з короткою інформацією про обрану локацію (рисунок 4.4 а). У вікні відображається назва, зображення, стислий опис, адреса, а також контактні дані.

Контакти подані у вигляді активного посилання, при натисканні на яке відкривається браузер або пропонується скопіювати посилання для подальшого використання (рисунок 4.4 б). У цьому вікні передбачено кілька функціональних кнопок (рисунок 4.3 а):

- «Показати повний опис місця» – відкриває екран із детальною інформацією про локацію;
- «Закрити» – закриває діалогове вікно;
- «Прокласти маршрут» – запускає побудову маршруту від поточного місцезнаходження користувача до обраного місця, автоматично закриваючи вікно.

Після побудови маршруту маркер місця змінює колір на помаранчевий, що візуально позначає його як активне. Сам маршрут відображається у вигляді синьої пунктирної лінії, яка допомагає краще орієнтуватися (рисунок 4.4 в). У верхній лівій частині екрана з’являється інформаційне вікно, яке показує приблизну відстань до обраного місця та орієнтовний час у

дорозі (рисунок 4.4 в). Щоб скасувати маршрут, потрібно натиснути іконку хрестика, яка розміщена як верхня кнопка у блоці навігаційних кнопок у правій частині екрана.

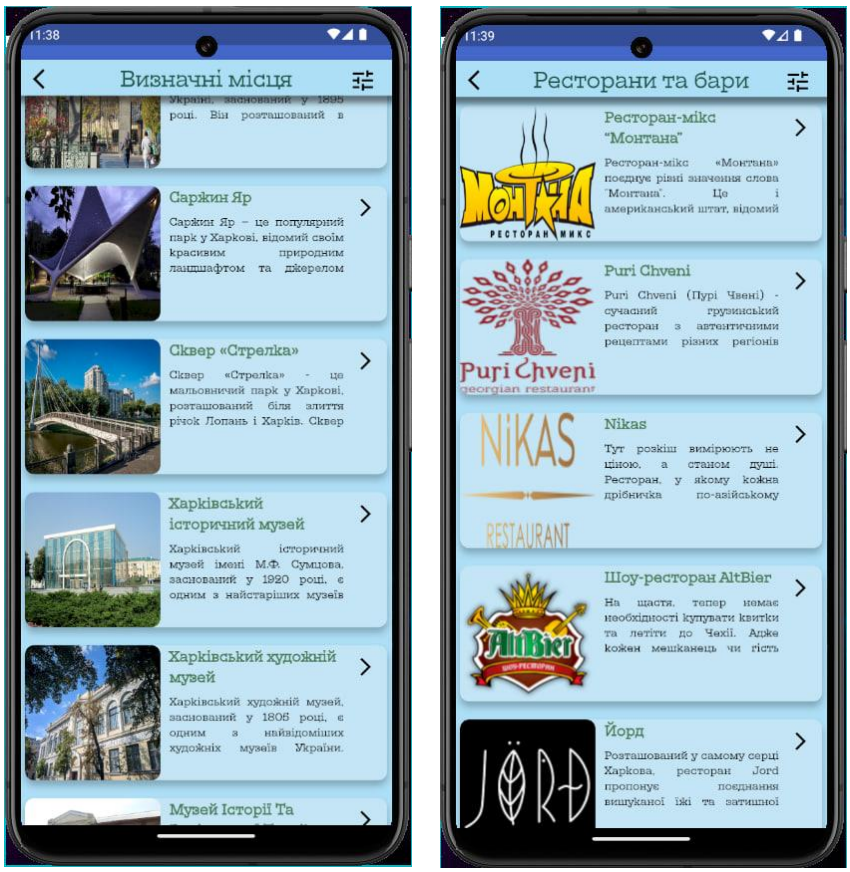


а) б) в)

Рисунок 4.4 – Взаємодія з маркером на екрані «Карта»: а) коротка інформація про місце; б) вікно для взаємодії з посиланням; в) екран «Карта» після виконання кнопки «Прокласти маршрут»

4.2 Перегляд каталогів місць за категоріями

При натисканні на плитку «Визначні місця» або «Ресторани та бари» здійснюється перехід до відповідного списку локацій. Кожен елемент у списку містить назву, короткий опис, зображення та кнопку переходу до повної інформації (рисунок 4.5). Користувач може скористатися фільтрацією, натиснувши на іконку фільтра у правому верхньому куті екрана (рисунок 4.5).



а) б)

Рисунок 4.5 – Екрани каталогів місць: а) екран «Визначні місця»; б) екран «Ресторани та бари»

Після цього відкривається діалогове вікно з перемикачами, які дозволяють обрати, які саме об’єкти будуть відображатися на карті. Набір доступних фільтрів залежить від обраної категорії:

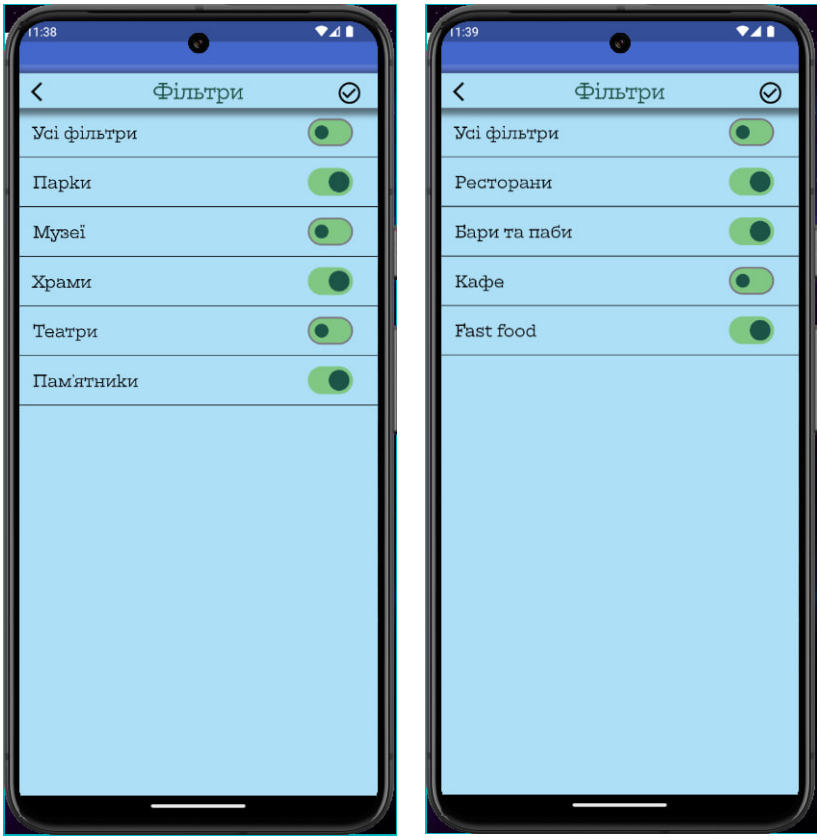
- для визначних місць доступні такі фільтри (рисунок 4.6 а):
  1) «Усі фільтри»;
  2) «Парки»;
  3) «Музеї»;
  4) «Храми»;
  5) «Театри»;
  6) «Пам’ятники»;
- для закладів харчування та відпочинку (рисунок 4.6 б):
  1) «Усі фільтри»;

2) «Ресторани»;

3) «Бари та паби»;

4) «Кафе»;

5) «Fast Food».

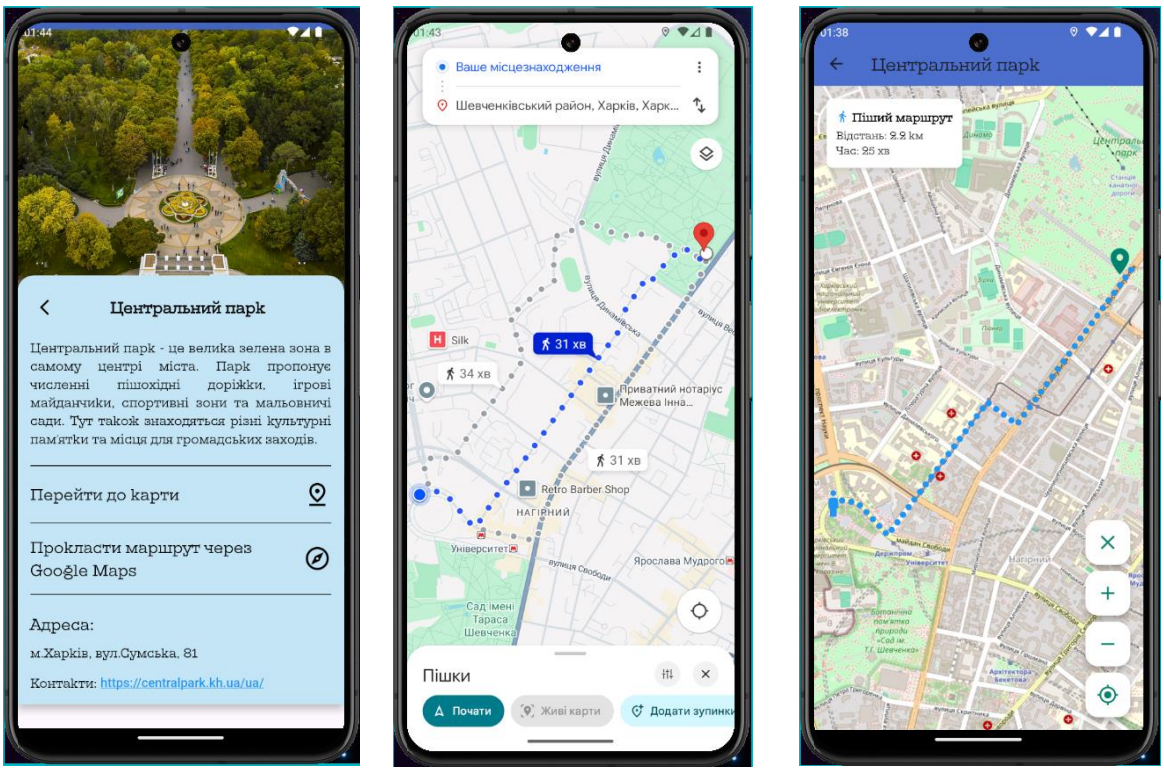


а) б)

Рисунок 4.6 – Екрани фільтрації: а) для каталогу «Визначні місця»; б) для каталогу «Ресторани та бари»

Для активації фільтра потрібно натиснути на відповідний перемикач, а потім натиснути кнопку підтвердження (галочку) у верхній частині вікна. Якщо користувач хоче побачити всі місця повторно, достатньо вибрати лише пункт «Усі фільтри».

При натисканні на кнопку переходу до повної інформації відкривається детальний екран з повною інформацією про місце (рисунок 4.7 а): повна назва, зображення, докладний опис, що включає історичні факти, культурне значення, адреса та контактні дані.



а) б) в)

Рисунок 4.7 – Детальний екран місця: а) початковий екран для «Центральній парк»; б) застосунок «Google Maps» з прокладеним маршрутом до «Центральній парк»; в) екран «Карта» для «Центральній парк»

На цьому екрані (рисунок 4.7) доступні дві основні дії: «Перейти до карти» та «Прокласти маршрут через Google Maps». При натисканні кнопки «Прокласти маршрут через Google Maps» запускається офіційний застосунок Google Maps із побудованим маршрутом до цього місця (рисунок 4.7 б). При натисканні кнопки «Перейти до карти» відкриває екран із картою, на якій вже нанесено маркер обраного місця та маркер користувача (рисунок 4.7 в).

Функціональність карти аналогічна до «Online карта», за винятком того, що фільтрація тут недоступна. Як і у звичайному режимі карти, маршрут позначається синьою лінією з точок, яка візуально показує шлях. Також у верхньому лівому куті з'являється вікно з інформацією про маршрут: орієнтовна відстань та час у дорозі (рисунок 4.7 в). Щоб скасувати маршрут, необхідно натиснути іконку хрестика, яка розміщена як верхня кнопка у блоці навігаційних кнопок у правій частині екрана.

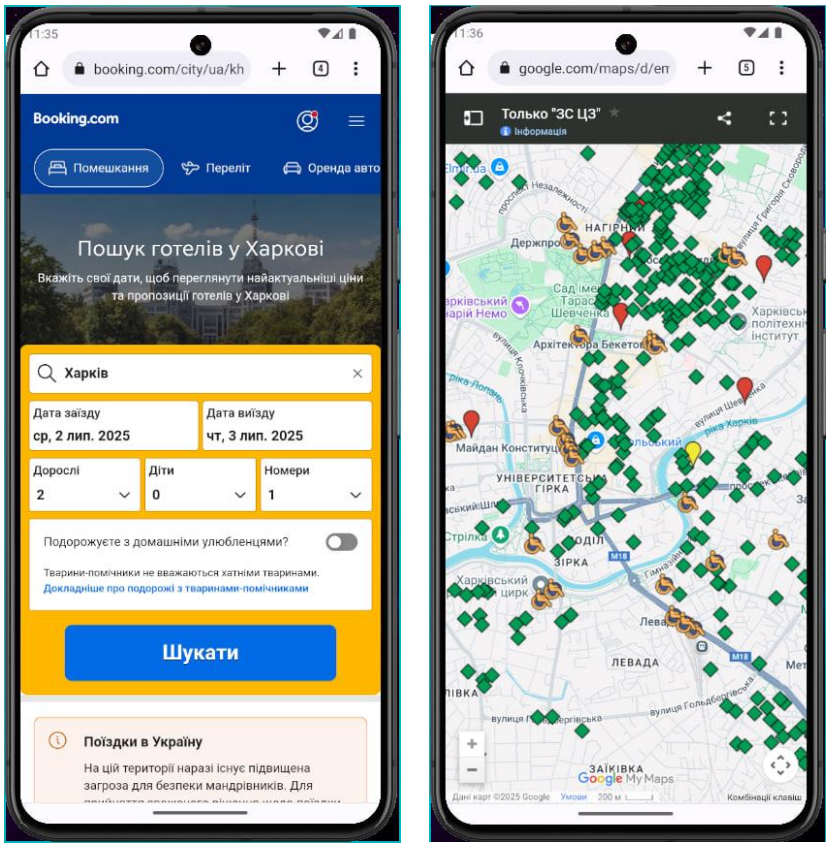
4.3 Перехід до зовнішніх ресурсів

У мобільному застосунку передбачено можливість швидкого доступу до перевірених зовнішніх онлайн-сервісів, що надають актуальну інформацію пов’язану з містом Харків. Для цього на головному екрані розміщено відповідні плитки:

- «Готелі та проживання» – перехід до сторінки з переліком готелів Харкова на сервісі Booking.com (рисунок 4.8 а);

- «Укриття» – перехід до інтерактивної карти з укриттями, реалізованої у Google Maps (рисунок 4.8 б).

-



а) б)

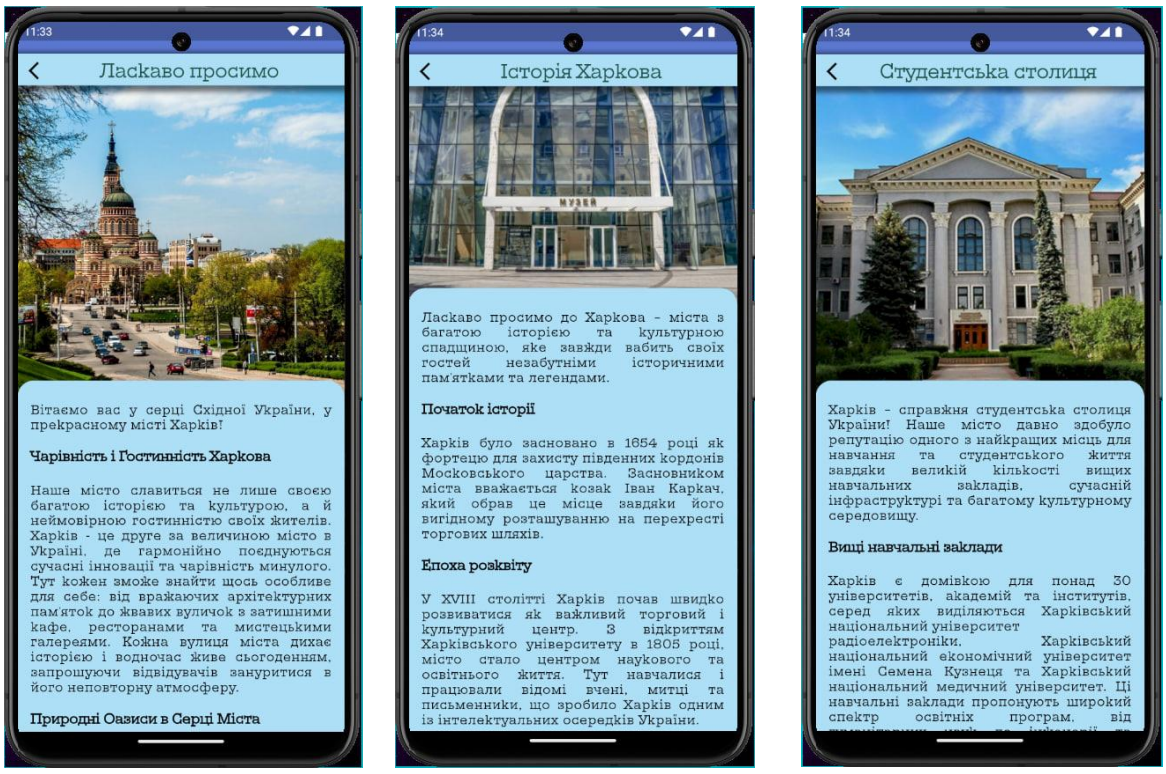
Рисунок 4.8 – Зовнішні ресурси: а) екран «Готелі та проживання»; б) екран «Укриття»

Після натискання на відповідний пункт застосунок автоматично відкриває зовнішній браузер зі сторінкою обраного сервісу, що дозволяє уникнути дублювання інформації в застосунку та гарантує отримання

найактуальніших даних напряму з надійних джерел. Функція переходу до зовнішніх ресурсів не потребує авторизації та не зберігає персональні дані користувача, що забезпечує простоту та безпеку використання.

4.4 Перегляд інформаційних сторінок

У застосунку доступні кілька інформаційних сторінок, що знайомлять користувача з містом Харків. Вони охоплюють важливі теми, такі як історія, культура, громадський транспорт, питання безпеки та загальні відомості про місто. Ці розділи створені спеціально для того, щоб надати корисну довідкову інформацію в доступній та зручній формі (рисунок 4.9).



а) б) в)

Рисунок 4.9 – Інформаційні сторінки: а) екран «Ласкаво просимо»; б) екран «Історичний екскурс»; в) екран «Студентська столиця»

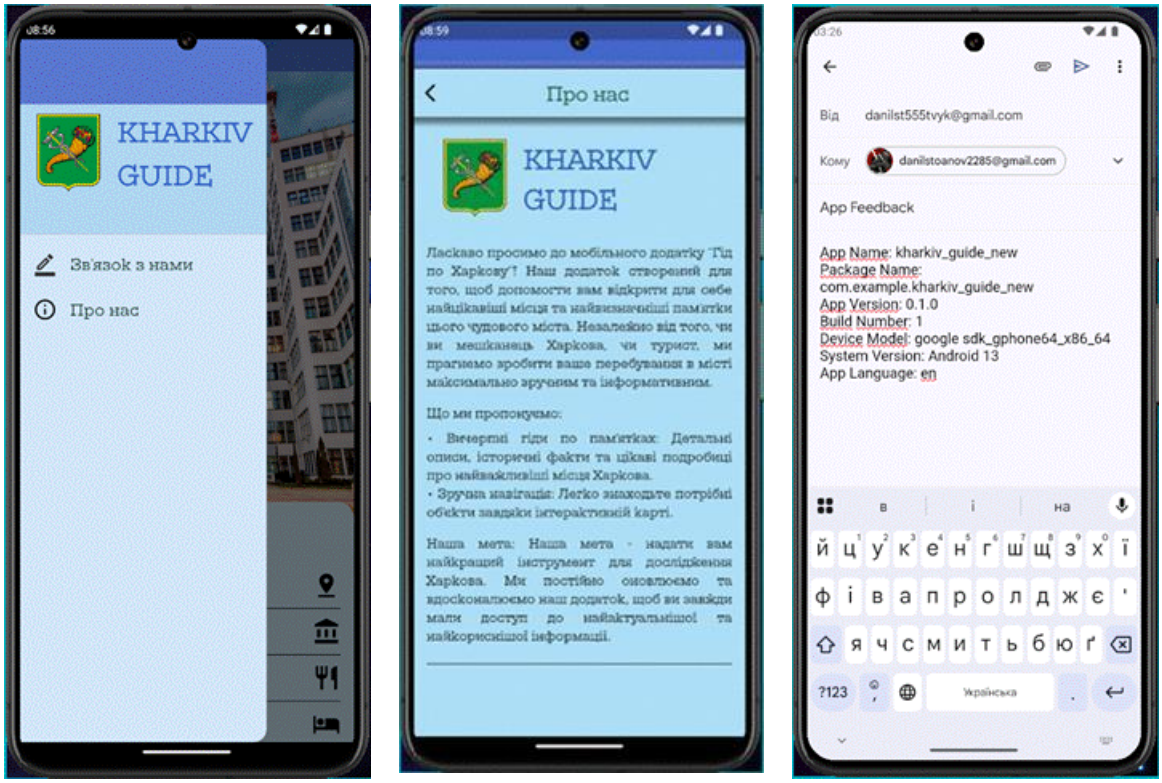
Переходити до кожної сторінки можна через головне меню, натискаючи на відповідні плитки з назвами: «Ласкаво просимо», «Тривога та безпека», «Історичний екскурс», «Студентська столиця», «Міський

транспорт». Після цього відкривається екран з розгорнутою текстовою інформацією, супроводженою зображенням і заголовком теми (рисунок 4.9).

Сторінку можна вільно прокручувати вниз, щоб ознайомитися з усім її вмістом. Для повернення до попереднього екрану достатньо скористатися кнопкою «Назад» у верхній панелі або стандартною навігацією пристрою.

4.5 Перехід до бокового меню

У верхньому лівому куті головного екрана розміщено кнопку у вигляді трьох горизонтальних ліній – це і є виклик бокового меню. Натиснувши на неї, користувач відкриває панель з додатковими розділами, які дозволяють швидко перейти до окремих частин застосунку (рисунок 4.10 а).



а) б) в)

Рисунок 4.10 – Екрани «Бокового віджету»: а) початкове бокове меню; б) пункт меню «Про нас»; в) пункт меню «Зв’язок з нами»

У меню доступні такі опції, як «Зв’язок з нами» – для звернення до розробників (рисунок 4.10 в), і «Про нас» – де коротко описано мету

застосунку та його можливості (рисунок 4.10 б). Під час вибору пункту «Зв’язок з нами» відкривається поштовий застосунок, Gmail. У новому листі автоматично заповнюється адреса одержувача, тема, а також додається технічна інформація про пристрій та версію програми. Користувачу залишається лише написати повідомлення або надіслати готовий лист із запитом, відгуком чи побажанням (рисунок 4.10 в).

Таке рішення дозволяє швидко й без зайвих кроків зв’язатися з розробником. Саме меню має простий та зручний дизайн – воно не заважає перегляду контенту і легко закривається повторним натисканням або жестом. Це забезпечує комфортну навігацію і швидкий доступ до важливої інформації.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено мобільний застосунок «Путівник по місту Харків», призначений для інформування мешканців та гостей міста про цікаві місця, культурні об’єкти, заклади харчування та інші важливі локації. Застосунок забезпечує зручну навігацію по карті з підтримкою геолокації, фільтрацію об’єктів за категоріями, побудову маршрутів та перегляд інформаційних сторінок з текстовими описами.

У процесі реалізації було використано сучасні підходи до розробки мобільного програмного забезпечення, зокрема фреймворк Flutter, архітектуру BLoC, принципи Clean Architecture, а також SQLite для зберігання локальної бази даних. Особливу увагу приділено зручності інтерфейсу (UI) та позитивному досвіду користувача (UX), адаптованому для мобільних пристроїв.

Розроблений застосунок протестовано на відповідність функціональним вимогам. В результаті отримано стабільний та інтуїтивно зрозумілий продукт, який може бути корисним як для туристів, так і для місцевих мешканців.

Отримані в ході роботи знання з мобільної розробки, архітектури застосунків, роботи з картографічними сервісами та базами даних були успішно застосовані на практиці, що свідчить про досягнення поставленої мети кваліфікаційної роботи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Мобільні застосунки: їх види та особливості. URL: https://highload.today/uk/mobilni-zastosunki-yih-vidi-ta-osoblivosti.

2. How to Create a Location-Based App. URL: https://codeit.us/blog/how-to-create-a-location-based-app?utm_source=chatgpt.com.

3. Kiev Travel Guide – Apps on Google Play. URL: https://play.google.com/store/apps/details?id =com.etips.kiev.travel.guide.

4. Lviv Guide – Apps on Google Play. URL: https://play.google.com/store/apps/details?id =com.guidexp.lvivguide.

5. Коломия: путівник і маршрути – Apps on Google Play. URL: https://play.google.com/store/apps/details?id =com.kolomyiaapp.android.places.

6. Львівщина – Apps on Google Play. URL: https://play.google.com/store/apps/details?id=com.komkor.lvivarea.

7. DRY, YAGNI, KISS and SINE – 4 Most Important Software Development Principles. URL: https://mattilehtinen.com/articles/4-most-important-software-development-principles-dry-yagni-kiss-and-sine.

8. What is the Dart Programming Language? A Guide to its Applications. URL: https://emeritus.org/blog/coding-dart-programming-language.

9. Flutter – Build apps for any screen. URL: https://flutter.dev.

10. Visual Studio Code: Code Editing. Redefined. URL: https://code.visualstudio.com.

11. Gittutorial Documentation – Git. URL: https://git-scm.com/docs.

12. What is GitHub. URL: https://www.geeksforgeeks.org/what-is-github-and-how-to-use-it/?ysclid=lx4igehn7n590881322.

13. The Clean Architecture. URL: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html.

14. Guide to app architecture, App architecture, Android Developers. URL: https://developer.android.com/topic/architecture?

15. Freeman E., Robson E., Ba B., Sierra K. Head First Design Patterns: Building Extensible and Maintainable Object-Oriented Software. Publisher: O'Reilly Media; 2nd edition, 2021. 669 p.

16. Flutter package: flutter_bloc. URL: https://pub.dev/packages/flutter_bloc.

17. Flutter package: flutter_map. URL: https://pub.dev/packages/flutter_map.

18. Flutter package: latlong2. URL: https://pub.dev/packages/latlong2.

19. Flutter package: sqflite. URL: https://pub.dev/packages/sqflite.

20. Flutter package: geolocator. URL: https://pub.dev/packages/geolocator.

21. Flutter package: url_launcher. URL: https://pub.dev/packages/url_launcher.

22. Flutter package: http. URL: https://pub.dev/packages/http.

23. Flutter package: android_intent_plus. URL: https://pub.dev/packages/android_intent_plus.

24. Flutter package: package_info_plus. URL: https://pub.dev/packages/package_info_plus.

25. Flutter package: device_info_plus. URL: https://pub.dev/packages/device_info_plus.

26. Flutter package: cupertino_icons. URL: https://pub.dev/packages/cupertino_icons.