БАКАЛАВРСЬКА РОБОТА

БР. ІП - 02.00.00.000 ПЗ

Група ІП-21-1

Ворох Ростислав

2025

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ворох Ростислав Васильович

(прізвище, ім'я, по батькові)

УДК 004

(індекс)

БАКАЛАВРСЬКА РОБОТА

Розробка мобільного додатку представлення ресурсів міста

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач освітнього рівня Ворох Р.В.

(підпис, ініціали та прізвище здобувача)

Науковий керівник Вовк Роман Богданович, к.т.н., доцент

(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2025

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ ____ ” _____2025 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Вороху Ростиславу Васильовичу

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) **“ Розробка мобільного додатку представлення ресурсів міста”**

керівник проекту (роботи) Вовк Р.Б., доцент, к.т.н.

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 р. № 264/7

2. Строк подання студентом проекту (роботи) 10 червня 2025 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

4. Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз актуальності розробки платформи представлення інфраструктури міста
2. Алгоритмічна реалізація мобільного додатку представлення ресурсів міста
3. Вибір та опис шаблону проектування
4. Реалізація моделі бази даних
5. Програмна реалізація мобільного додатку представлення ресурсів міста

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Класифікація тенденцій щодо побудови платформи представлення ресурсів (рис. 1.1)
2. Загальний перелік існуючих платформ представлення міських ресурсів (рис. 1.2)
3. Графічне представлення проблем покращення платформ представлення ресурсів (рис. 1.3)
4. Jakdojade.pl – ресурс планування маршрутів громадським транспортом (рис. 1.4)
5. Ресурс асоціації громадського транспорту Гамбурга (рис. 1.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання **28 квітня 2025 р.**

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Аналіз актуальності розробки платформи представлення інфраструктури міста	04.05.2025	виконано
2	Алгоритмічна реалізація мобільного додатку представлення ресурсів міста	16.05.2025	виконано
3	Вибір та опис шаблону проектування	25.05.2025	виконано
4	Реалізація моделі бази даних	01.06.2025	виконано
5	Програмна реалізація мобільного додатку представлення ресурсів міста	05.06.2025	виконано
6	Оформлення пояснювальної записки дипломної роботи завідувачем кафедри	10.06.2025	виконано

Студент – дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Бакалаврська робота містить 75 сторінок, 27 рисунків, список використаних джерел із 37 найменуваннями, 1 додаток.

Метою роботи є розробка мобільного застосунку для представлення ресурсів міста з урахуванням сучасних підходів до проєктування інтерфейсу, вибору ефективних технологій реалізації та забезпечення інтерактивної взаємодії з користувачем.

Об'єкт дослідження - інформаційні технології для мобільного представлення інфраструктурних ресурсів міста.

Предмет дослідження - архітектурні, алгоритмічні та програмні рішення, пов'язані з розробкою мобільного додатку для інтегрованого доступу до даних про ресурси міста.

В першому розділі проведено ґрунтовний аналіз актуальності створення платформи для представлення інфраструктури міста

В другому розділі ключові технічні рішення, що стали основою для реалізації мобільного додатку з представлення ресурсів міста

В третьому розділі реалізовано макет додатку з використанням методу "ListView всередині CardView", що забезпечує зручне та структуроване представлення інформаційних блоків

Висновок: здійснено практичну реалізацію мобільного додатку для представлення міських ресурсів, зосереджену на побудові інтерфейсу користувача та забезпеченні його функціональності

КЛЮЧОВІ СЛОВА: МОБІЛЬНИЙ ДОДАТОК, РЕСУРСИ МІСТА, МІСЬКА ІНФРАСТРУКТУРА, ANDROID, ІНТЕРФЕЙС КОРИСТУВАЧА, SQLITE, CARDVIEW, WEBVIEW, ІНТЕРАКТИВНА СИСТЕМА.

ANNOTATION

The bachelor's thesis contains 75 pages, 27 figures, a list of used sources with 37 names, 1 appendix.

The purpose of the work is to develop a mobile application for presenting city resources, taking into account modern approaches to interface design, the selection of effective implementation technologies and ensuring interactive interaction with the user.

The object of research is information technologies for mobile presentation of city infrastructure resources.

The subject of research is architectural, algorithmic and software solutions related to the development of a mobile application for integrated access to data on city resources.

The first section provides a thorough analysis of the relevance of creating a platform for presenting city infrastructure

The second section presents key technical solutions that became the basis for the implementation of a mobile application for presenting city resources

The third section implements an application layout using the “ListView inside CardView” method, which provides a convenient and structured presentation of information blocks

Conclusion: a practical implementation of a mobile application for presenting city resources has been carried out, focused on building a user interface and ensuring its functionality.

KEYWORDS: MOBILE APPLICATION, CITY RESOURCES, URBAN INFRASTRUCTURE, ANDROID, USER INTERFACE, SQLITE, CARDVIEW, WEBVIEW, INTERACTIVE SYSTEM

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

XML - eXtensible Markup Language — Розширювана мова розмітки

API - Application Programming Interface — Інтерфейс прикладного програмування

SQLite - Lightweight Structured Query Language Database — Полегшена вбудована база даних на основі SQL

UI/UX - Інтерфейс та досвід користувача

HTTP - HyperText Transfer Protocol — Протокол передавання гіпертексту

JSON - JavaScript Object Notation — Формат обміну даними

ВСТУП

Стрімкий розвиток цифрових технологій у поєднанні зі зростанням потреб мешканців міст у зручному доступі до актуальної інформації щодо міської інфраструктури створює передумови для розробки інтерактивних платформ, які інтегрують різноманітні сервіси в одному середовищі. Мешканці та гості міста щоденно користуються інформацією про транспорт, заклади харчування, культурні події тощо. Проте наявні сервіси часто охоплюють лише окремі аспекти міської інфраструктури або не забезпечують достатньої зручності у користуванні.

У сучасних умовах інтенсивного розвитку міст та динамічного ритму життя населення зростає потреба у зручному та оперативному доступі до інформації про ресурси міста: транспортні маршрути, заклади харчування, культурні події, туристичні об’єкти тощо. Традиційні інформаційні джерела часто є розрізненими, малозручними або недостатньо адаптованими до потреб мобільних користувачів. У цьому контексті важливою є розробка інтуїтивно зрозумілого мобільного додатку, який об’єднує різні сервіси в єдиному цифровому середовищі.

Актуальність роботи

У контексті зростання урбанізації та цифрової трансформації міського простору актуальним стає створення ефективних цифрових інструментів для інформування мешканців і туристів щодо міської інфраструктури. Наявні сервіси часто фрагментовані, орієнтовані на окремі аспекти міського життя (транспорт, харчування, події) і не забезпечують інтегрованого користувацького досвіду. Розробка мобільного додатку, що об'єднує різні ресурси міста в єдиній зручній платформі, сприяє підвищенню доступності міських сервісів, покращенню взаємодії мешканців із міським середовищем та формуванню “розумного міста” (smart city).

Актуальність теми роботи полягає у створенні мобільного додатку, що об'єднує інформацію про ключові ресурси міста в єдиному інтерфейсі з акцентом на зручність, швидкість доступу до даних та візуальну привабливість. Такий додаток може слугувати ефективним інструментом для жителів міста, туристів і представників міських адміністрацій.

Метою роботи є розробка мобільного застосунку для представлення ресурсів міста з урахуванням сучасних підходів до проєктування інтерфейсу, вибору ефективних технологій реалізації та забезпечення інтерактивної взаємодії з користувачем.

Завдання дослідження

1. Проаналізувати існуючі сервіси та платформи для представлення міської інфраструктури.

2. Визначити технічні вимоги та обрати відповідні інструменти й мову програмування.

3. Розробити структуру бази даних та архітектуру додатку.

4. Реалізувати основні функції користувацького інтерфейсу.

5. Виконати тестування та оцінити працездатність додатку.

Об’єкт дослідження - інформаційні технології для мобільного представлення інфраструктурних ресурсів міста.

Предмет дослідження - архітектурні, алгоритмічні та програмні рішення, пов’язані з розробкою мобільного додатку для інтегрованого доступу до даних про ресурси міста.

Методи дослідження

- Аналіз існуючих інформаційних систем і платформ.

- Проєктування архітектури та бази даних.

- Методи об’єктно-орієнтованого програмування при реалізації додатку.

- Функціональне тестування для перевірки працездатності розробленого програмного забезпечення.

Наукова новизна

У роботі запропоновано концепцію мобільного додатку, що об'єднує декілька типів міських ресурсів у єдину платформу з єдиною моделлю даних і уніфікованим інтерфейсом, що дозволяє користувачам швидко знаходити необхідну інформацію про місто в одному місці.

У ході дослідження проаналізовано існуючі рішення, визначено технологічну базу, спроєктовано архітектуру системи та реалізовано функціональний мобільний додаток з урахуванням вимог користувачів і особливостей міської інфраструктури.

Практичне значення роботи полягає у створенні гнучкого, масштабованого і зручного у користуванні програмного продукту, який може бути використаний для розвитку цифрової інфраструктури міст, покращення доступу до міських сервісів і підвищення комфортності життя мешканців.

Бакалаврська робота містить 75 сторінок, 27 рисунків, 3 розділи список використаних джерел із 37 найменуваннями, 1 додаток.

РОЗДІЛ 1. АНАЛІЗ АКТУАЛЬНОСТІ РОЗРОБКИ ПЛАТФОРМИ ПРЕДСТАВЛЕННЯ ІНФРАСТРУКТУРИ МІСТА

1.1. Загальні тенденції та виклики побудови платформи представлення міських ресурсів

Розглянемо загальні тенденції та спостереження предметної області щодо розробки платформи представлення ресурсів міста (рис. 1.1).

Рисунок 1.1 – Класифікація основних тенденцій щодо побудови платформи представлення міських ресурсів

Розглянемо подані тенденції більш детально.

1. Інформаційна насиченість. Багато міст мають веб-сайти та/або мобільні додатки, які намагаються охопити якомога більше інформації. Це може бути як перевагою (великий обсяг даних), так і недоліком (складність навігації, застаріла інформація).

2. Фокус на туризм. Часто основний акцент робиться на туристичних ресурсах, що може залишати поза увагою потреби місцевих жителів (наприклад, детальний розклад громадського транспорту в реальному часі).

3. Різноманітність форматів. Інформація може бути представлена у вигляді списків, карт, фотогалерей, відгуків користувачів, новинних стрічок тощо.

4. Інтерактивність. Деякі платформи пропонують інтерактивні функції, такі як фільтри пошуку, можливість бронювання, додавання відгуків, побудова маршрутів.

5. Мобільна оптимізація. Зростає важливість мобільної доступності. Багато веб-ресурсів адаптуються під мобільні пристрої, а мобільні додатки стають все популярнішими завдяки зручності використання "на ходу".

6. Інтеграція зі сторонніми сервісами. Часто міські платформи інтегруються з іншими сервісами, такими як Google Maps, OpenStreetMap, Booking.com, TripAdvisor тощо.

7. Проблеми з актуальністю: Однією з головних проблем є підтримка актуальності інформації (графіки роботи, ціни, наявність місць тощо).

Проведемо аналіз за типами таких інформаційних ресурсів.

1. Заклади харчування (ресторани, кафе, бари).

- Веб-ресурси часто представлені у вигляді каталогів з можливістю фільтрації за типом кухні, середнім чеком, розташуванням. Можуть містити фотографії, меню, контакти, відгуки. Популярними є агрегатори відгуків (наприклад, TripAdvisor, Yelp) та сервіси онлайн-бронювання (наприклад, OpenTable).

- Мобільні додатки зручні для пошуку закладів поблизу, перегляду відгуків та фотографій, бронювання столиків. Часто мають функцію навігації.

2. Зони відпочинку (парки, сквери, музеї, кінотеатри).

- Веб-ресурси зазвичай містять інформацію про розташування, графік роботи, вартість відвідування, фотографії, опис. Для музеїв та театрів може бути інформація про поточні виставки та репертуар.

- Мобільні додатки можуть надавати інтерактивні карти з позначеними зонами відпочинку, аудіогіди для музеїв, можливість придбання квитків онлайн.

3. Розклад транспорту (автобуси, тролейбуси, трамваї, потяги).

Веб-ресурси часто представлені у вигляді таблиць з розкладом руху. Можуть містити інформацію про маршрути, зупинки, вартість проїзду. Деякі сервіси надають можливість відстеження транспорту в реальному часі.

Мобільні додатки є дуже зручними для перегляду розкладу, відстеження транспорту на карті, планування маршрутів з урахуванням громадського транспорту.

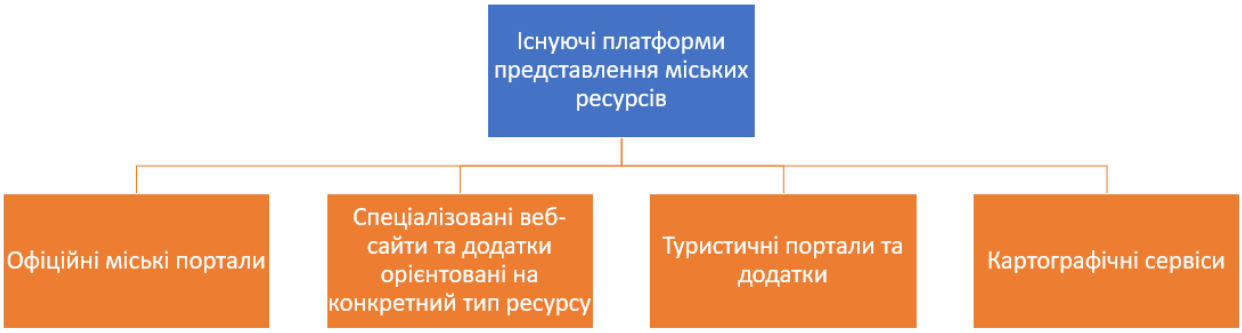

Рисунок 1.2 – Загальний перелік існуючих платформ представлення міських ресурсів

Приклади існуючих рішень можуть бути специфічними для інших міст, але ілюструють загальні підходи (рис. 1.2):

Офіційні міські портали часто містять розділи з інформацією про всі типи міських ресурсів. Їхньою перевагою є офіційність інформації, але вони можуть бути менш зручними у використанні порівняно зі спеціалізованими платформами.

Спеціалізовані веб-сайти та додатки орієнтовані на конкретний тип ресурсу (наприклад, сайт з відгуками про ресторани, додаток з розкладом автобусів). Зазвичай мають більш глибоку функціональність для своєї ніші.

Туристичні портали та додатки зосереджені на інформації, корисній для туристів. Можуть включати рекомендації щодо закладів харчування, місць для відвідування, транспортних опцій.

Картографічні сервіси (Google Maps, Citymapper) інтегрують інформацію про різні міські ресурси, включаючи заклади, зони відпочинку та громадський транспорт. Часто надають навігацію та інформацію в реальному часі.

Рисунок 1.3 – Графічне представлення проблем та можливостей для покращення платформ представлення міських ресурсів

Проблеми та можливості для покращення (рис. 1.3):

1. Розрізненість інформації. Інформація про різні міські ресурси може бути розкидана по різних веб-сайтах та додатках, що ускладнює її пошук.

2. Недостатня актуальність. Розклади транспорту, графіки роботи закладів можуть бути не оновлені.

3. Обмежена функціональність. Деякі платформи можуть мати базовий набір функцій і не пропонувати інтерактивних можливостей (наприклад, онлайн-бронювання, відстеження транспорту).

4. Можливість створення єдиної інтегрованої платформи. Розробка веб-сайту або мобільного додатку, який би об'єднував інформацію про всі основні міські ресурси, був би дуже корисним для мешканців та гостей міста.

5. Впровадження системи зворотного зв'язку. Надання користувачам можливості повідомляти про неточності або залишати відгуки допомогло б підтримувати актуальність інформації.

6. Використання відкритих даних (Open Data). Забезпечення доступу до відкритих даних про міські ресурси дозволило б стороннім розробникам створювати корисні сервіси.

1.2. Аналіз існуючих сервісів і платформ представлення інфраструктури міста

1.2.1. Аналіз онлайн ресурсів планування громадського транспорту

Спочатку розглянемо сервіси що пов’язані з питаннями громадського транспортного зв’язку в містах.

Jakdojade.pl – це веб-сайт та мобільний додаток, який допомагає планувати маршрути громадським транспортом у багатьох містах Польщі. Він пропонує:

- Актуальні розклади руху, тобто інформаціб про автобуси, трамваї, метро, приміські поїзди та навіть деякі міжміські маршрути.

- Пошук маршрутів. Можливість знайти оптимальний маршрут з урахуванням часу в дорозі, пересадок та вартості проїзду.

- Відстеження транспорту в реальному часі. У деяких містах можна бачити, де саме зараз знаходиться автобус чи трамвай.

- Купівля квитків. У багатьох містах можна придбати квитки на проїзд прямо в додатку.

- Інформація про затримки. Додаток повідомляє про затримки в русі транспорту.

- Українська версія. Доступна українська мова інтерфейсу.

- Офлайн-доступ до розкладів. Можливість переглядати збережені розклади без доступу до Інтернету.

- Навігація. Функція навігації для пішохідних переходів між зупинками.

- Інтеграція з картами. Відображення маршрутів на карті.

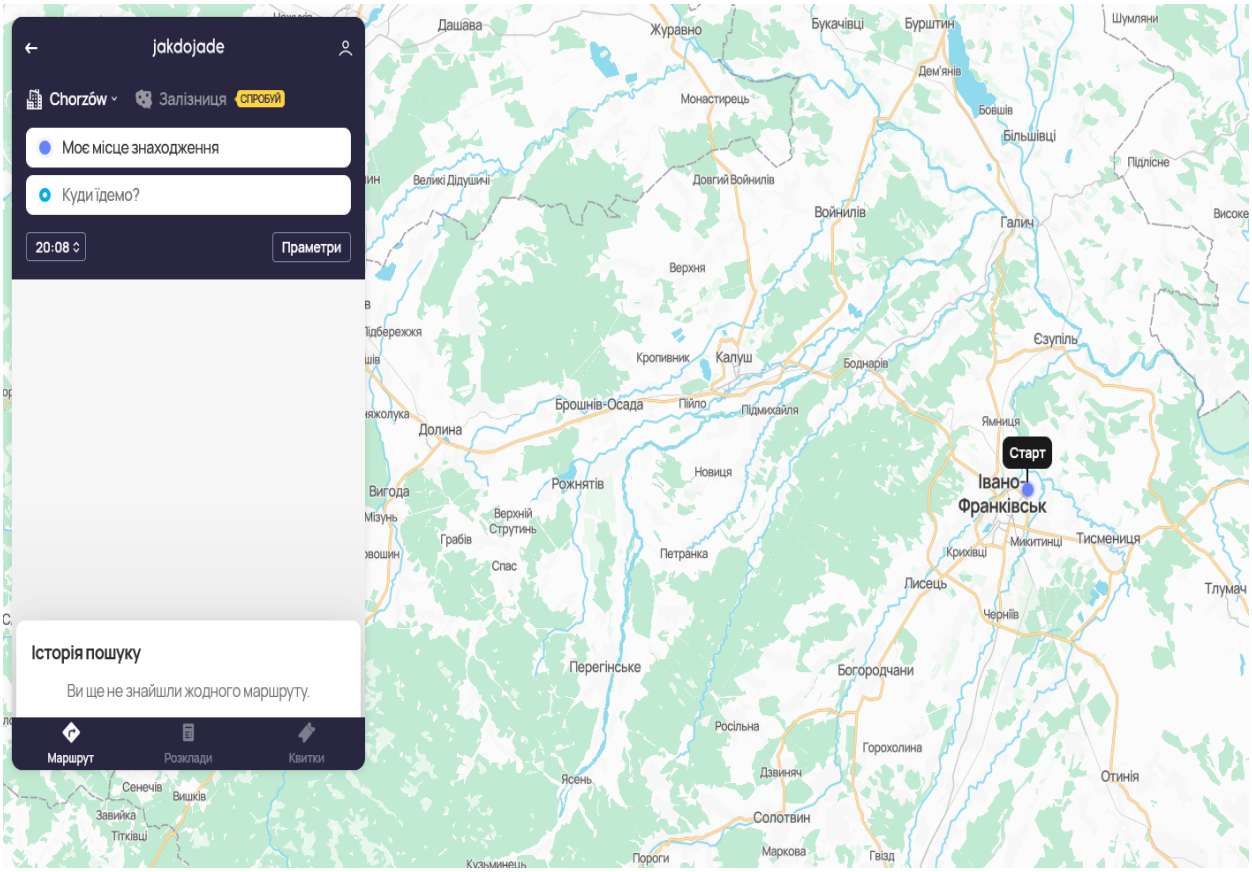

Рисунок 1.4 - Jakdojade.pl – ресурс планування маршрутів громадським транспортом у багатьох містах Польщі

Jakdojade.pl вважається одним з найточніших і найпопулярніших сервісів для планування маршрутів громадським транспортом в Польщі.

Розглянемо планування маршрутів громадським транспортом в містах Німеччини на прикладі міста Гамбург. Веб-сайт присвячений Hamburger Verkehrsverbund (HVV), асоціації громадського транспорту Гамбурга.

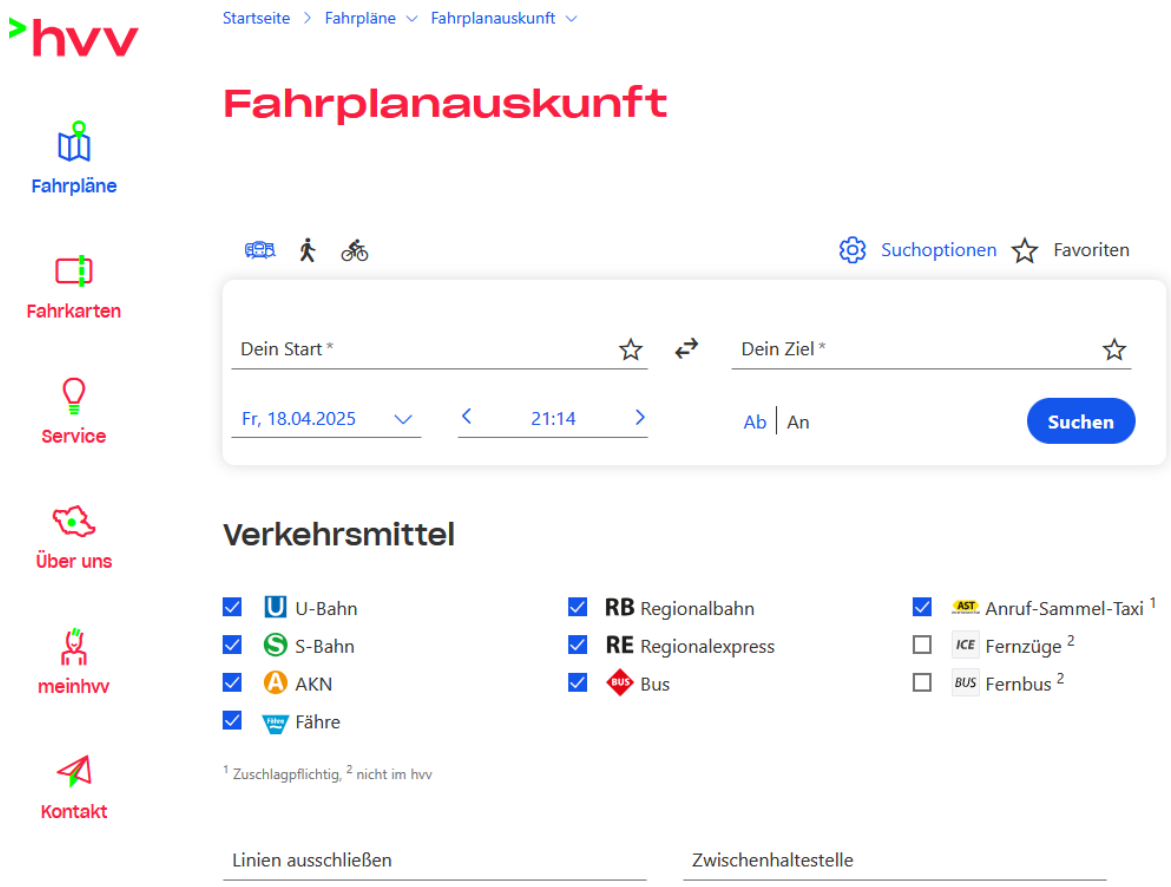

Рисунок 1.5 – Ресурс асоціації громадського транспорту Гамбурга

Веб-сайт надає різноманітні послуги та інформацію, пов’язану з громадським транспортом у Гамбурзі та прилеглих районах. Деякі ключові функції включають розділ Fahrplanauskunft (Інформація про розклад), де користувачі можуть знайти розклади, Abfahrtszeiten (час відправлення) та сполучення. Веб-сайт також пропонує інформацію про Fahrkarten (квитки), включаючи hvv Deutschlandticket, Einzel-/Tageskarten (одиночні/денні квитки), а також варіанти для різних груп, таких як Kinder / Schüler*innen

(діти/школярі) та Azubis (стажисти). Крім того, сайт містить детальну інформацію про тарифну систему hvv, пункти обслуговування та програми. Він також містить новини та оновлення, такі як запровадження Wunschausstieg (гнучкі зупинки) у вечірні та нічні години та продовження MetroBus-Linie 4.

RATP (https://www.ratp.fr/) - офіційний сайт оператора громадського транспорту Парижа та його околиць. Пропонує планування маршрутів, розклади, карти, інформацію про трафік та квитки. Має також мобільний додаток Bonjour RATP.

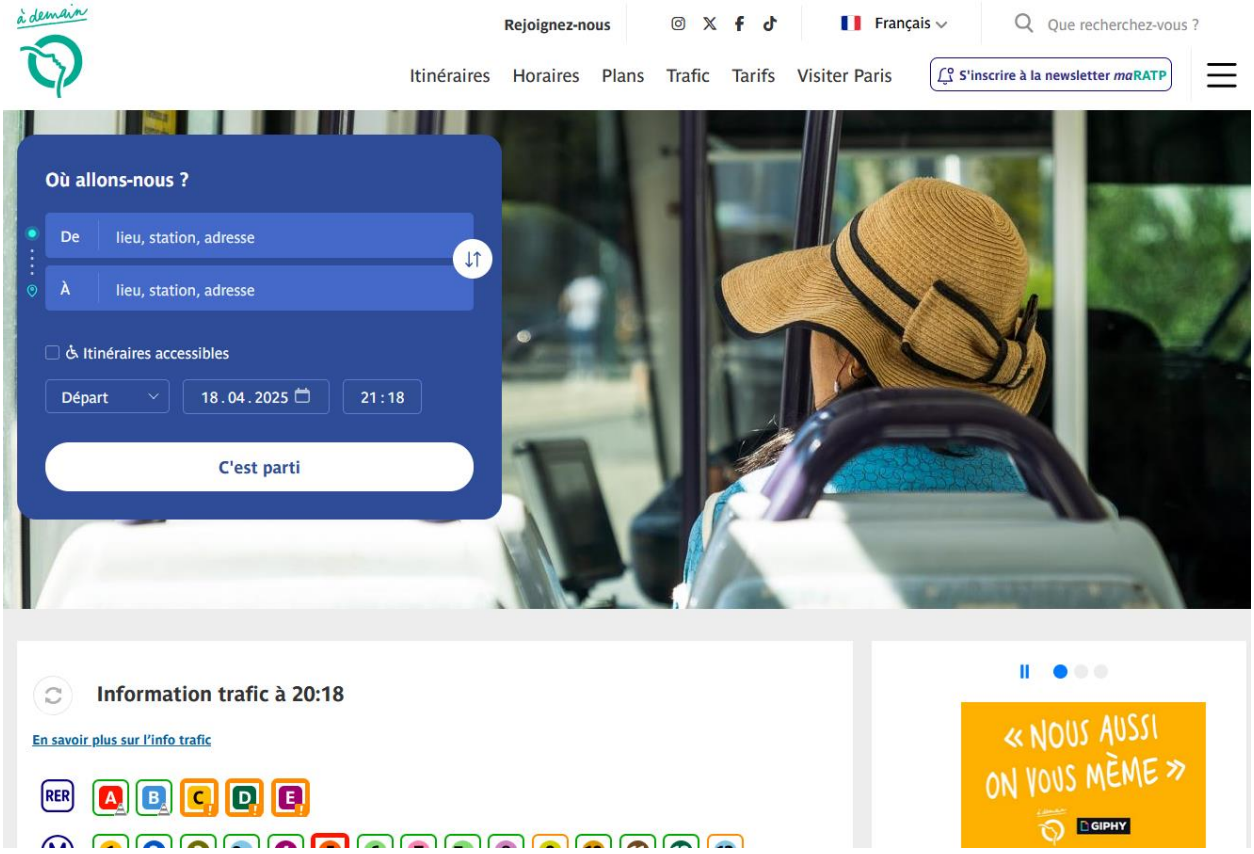

Рисунок 1.6 - Офіційний сайт оператора громадського транспорту Парижа та його околиць

Веб-сайт RATP пропонує різноманітні інструменти та інформацію для користувачів громадського транспорту. Сайт дозволяє користувачам планувати свої подорожі, вказуючи початкову та кінцеву точку. Він пропонує

можливість вказати дату та час відправлення або прибуття, а також може надати варіанти доступних маршрутів.

Користувачі можуть отримати доступ до розкладів для різних видів транспорту, включаючи метро, автобуси, трамваї та RER. На веб-сайті представлені карти систем метро, автобусів і трамваїв. Сайт пропонує оновлення трафіку в режимі реального часу, інформуючи користувачів про будь-які збої або затримки в мережі.

Користувачі можуть підписатися на інформаційну розсилку maRATP, щоб отримувати сповіщення та оновлення. Також є спеціальний номер (3117) для повідомлення про проблеми з безпекою. Веб-сайт також містить інформацію про поточні та майбутні роботи на лініях.

1.2.2. Аналіз ресурсів щодо закладів харчування в місті

Yelp – це онлайн-платформа, яка дозволяє користувачам знаходити та оцінювати місцеві заклади та послуги. Розглянемо основні функції та можливості Yelp. Yelp містить велику базу даних з інформацією про різні компанії, такі як ресторани, магазини, розважальні заклади, салони краси, медичні центри та багато іншого.

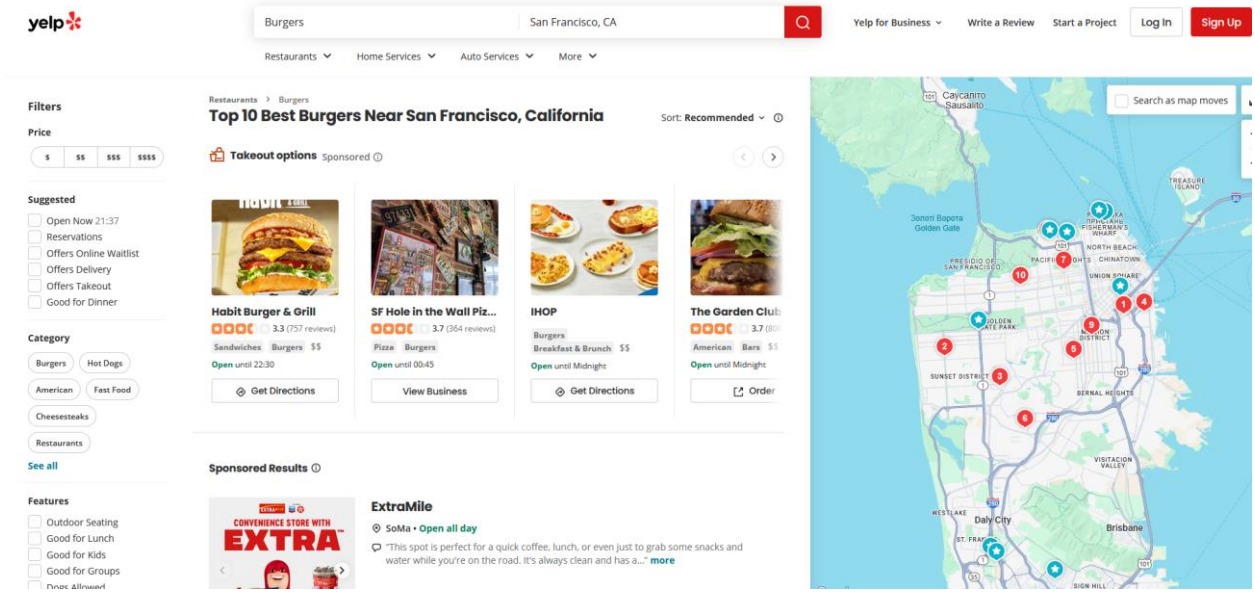

Рисунок 1.7 - Онлайн-платформа Yelp

Користувачі можуть залишати відгуки та ставити оцінки закладам, що допомагає іншим людям приймати обґрунтовані рішення. Yelp дозволяє шукати заклади за назвою, категорією, місцем розташуванням, ціною та іншими критеріями. Існують також фільтри для уточнення результатів пошуку.

Кожна сторінка закладу містить важливу інформацію, таку як адреса, години роботи, контактні дані, меню (для ресторанів), фотографії та посилання на веб-сайт. Yelp використовує служби на основі місцезнаходження, щоб допомагати користувачам знаходити заклади поблизу. Компанії можуть взаємодіяти з користувачами, відповідаючи на відгуки. Користувачі можуть завантажувати фотографії закладів, а також зберігати заклади в закладки для подальшого використання.

Time Out – це веб-сайт, який слугує вичерпним путівником по найкращих містах світу, надаючи інформацію про те, чим зайнятися, де поїсти та випити, про мистецтво та культуру, подорожі, фільми та музику.

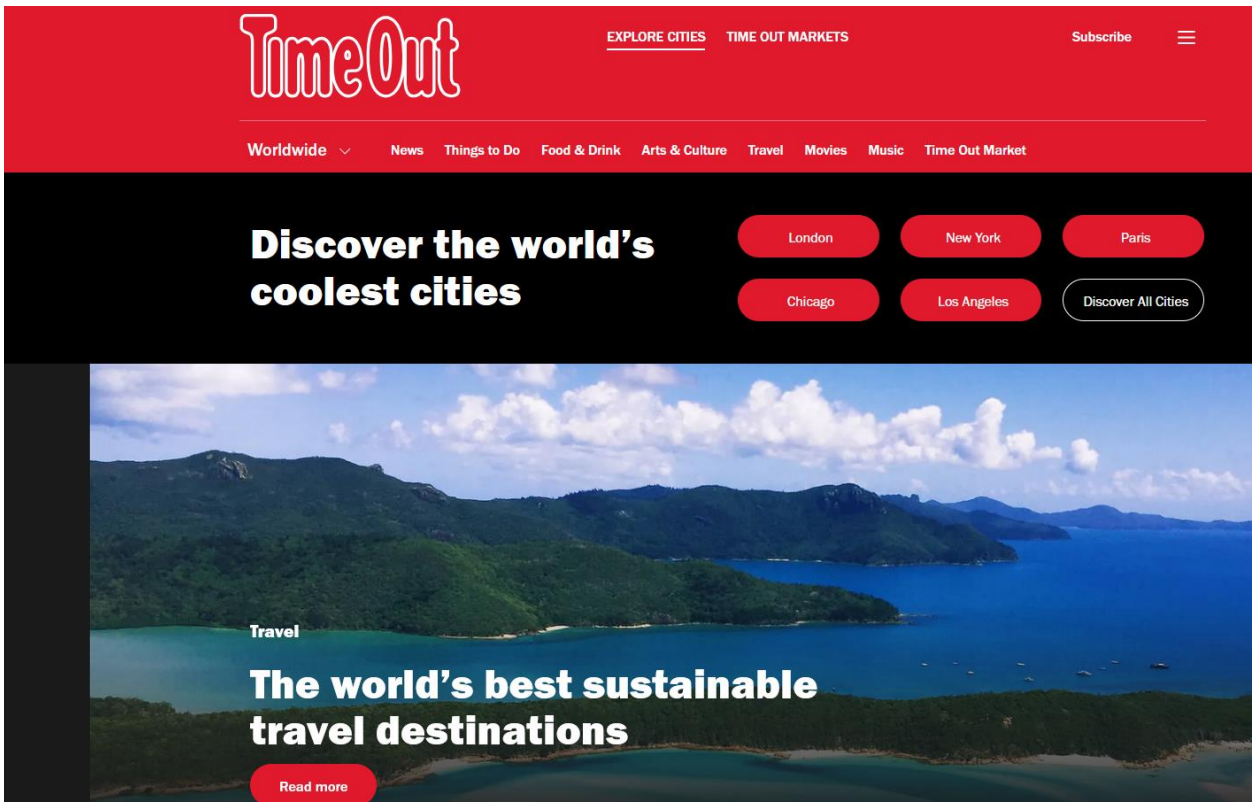

Рисунок 1.8 – Міський гід Time Out

Проведемо короткий огляд його функцій та можливостей Time Out пропонує путівники по численних містах світу, включаючи популярні напрямки, такі як Лондон, Нью-Йорк, Париж та Токіо. Ці путівники містять інформацію про місцеві пам'ятки, ресторани, бари, події та культурні заходи.

Веб-сайт рекламує Time Out Market, які пропонують кулінарні та культурні враження в різних містах. Користувачі можуть підписатися на розсилку новин, щоб отримувати оновлення про найкраще в місті, безкоштовні події та цікаві місця від Time Out.

Веб-сайт містить популярні статті та відео, пов'язані з подорожами, їжею, культурою та розвагами. Time Out створює оригінальний відеоконтент, наприклад, серіал "24 години в:", який демонструє різні міста та враження. Веб-сайт містить посилання на профілі Time Out у соціальних мережах, таких як Instagram, TikTok, Facebook, Twitter та YouTube. Time Out пропонує свій контент кількома мовами, орієнтуючись на глобальну аудиторію.

Foursquare City Guide – це мобільний додаток, розроблений, щоб допомогти користувачам знаходити та досліджувати цікаві місця, ресторани, розваги та інші заклади в їхньому поточному місцезнаходженні або в будь-якому іншому місті світу. Основна цінність додатку полягає у використанні рекомендацій від спільноти користувачів, а також персоналізованих пропозицій на основі вподобань.

Розглянемо ключові функції та характеристики Foursquare City Guide.

Користувачі можуть шукати конкретні місця, категорії закладів (наприклад, "італійські ресторани", "кав'ярні", "музеї") або види діяльності ("жива музика", "нічні клуби").

Додаток базується на відгуках, оцінках, підказках (tips) та списках, створених іншими користувачами Foursquare. Це дозволяє отримувати уявлення про якість та особливості різних місць з перших рук.

Foursquare аналізує історію відвідувань користувача, його збережені місця та вподобання, щоб надавати більш релевантні та персоналізовані

рекомендації. Чим більше ви використовуєте додаток, тим краще він розуміє ваші смаки.

Рисунок 1.9 – Додаток Foursquare City Guide

Користувачі можуть створювати власні списки цікавих місць (наприклад, "Кращі кав'ярні міста", "Місця, які варто відвідати") та ділитися ними з іншими. Існують також готові списки від інших користувачів та редакторів Foursquare. Користувачі можуть залишати короткі текстові підказки про конкретні аспекти місця, наприклад, "спробуйте їхній фірмовий бургер" або "гарний вид з тераси".

Додаток дозволяє переглядати фотографії місць, завантажені іншими користувачами, що допомагає скласти візуальне уявлення про заклад. Для кожного закладу надається важлива інформація, така як адреса, години роботи, контактний телефон, веб-сайт, ціновий діапазон та наявні зручності (наприклад, Wi-Fi, парковка).

Foursquare інтегрований з картами, що дозволяє легко знайти розташування обраного місця та прокласти до нього маршрут. Користувачі можуть підписуватися на своїх друзів, щоб бачити їхні відгуки та відвідані місця, отримуючи таким чином персональні рекомендації від знайомих.

Foursquare City Guide є корисним інструментом для тих, хто хоче досліджувати нові місця у своєму місті або під час подорожей, спираючись на досвід та рекомендації інших користувачів. Він поєднує в собі елементи соціальної мережі та локального пошукового сервісу.

1.2.3. Аналіз платформ планування відпочинку, екскурсій, подій в місті

Eventbrite: (https://www.eventbrite.com/) - сайт для пошуку та купівлі квитків на різноманітні події, такі як концерти, виставки, майстер-класи, фестивалі тощо. Eventbrite – це онлайн-платформа, яка дозволяє користувачам знаходити та купувати квитки на широкий спектр подій.

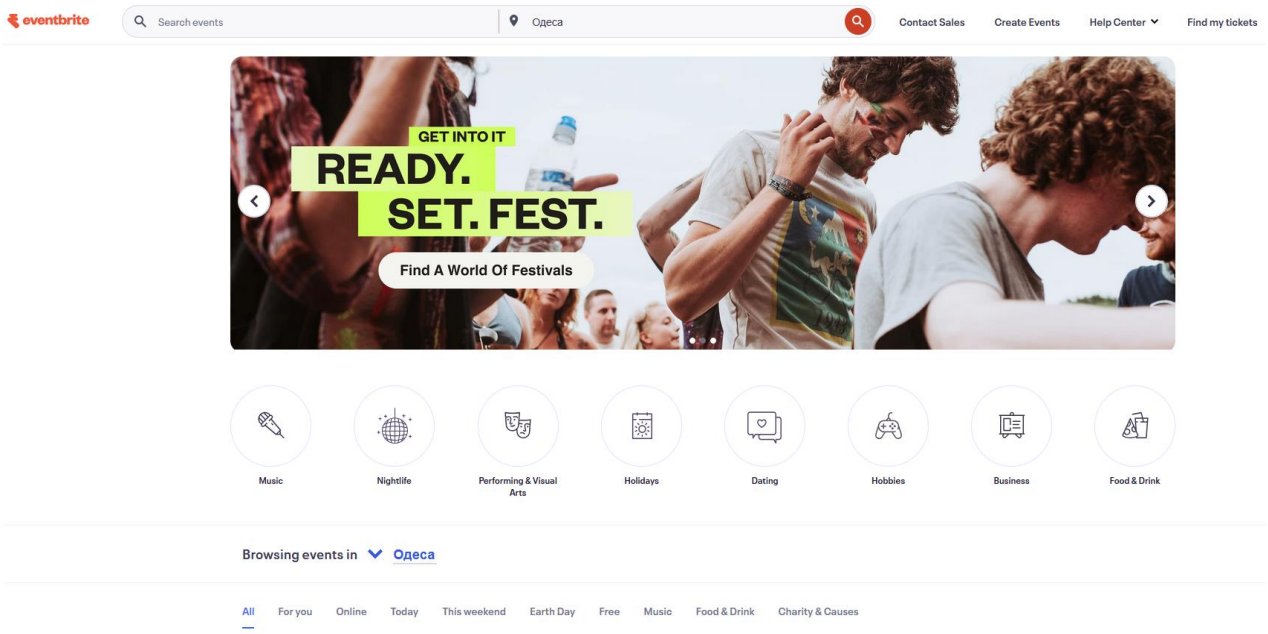

Рисунок 1.10 – Онлайн-платформа Eventbrite

Розглянемо детальний опис її функціональності:

- Пошук подій. Eventbrite пропонує потужний пошук, що дозволяє знаходити події за ключовими словами, категоріями (наприклад, музика, мистецтво, бізнес, їжа, мода), місцем розташуванням, датою, ціною та іншими критеріями.

- Категорії подій. Платформа охоплює велику кількість категорій, включаючи концерти, фестивалі, виставки, конференції, семінари, майстер-класи, спортивні заходи, театральні вистави, кінопокази, кулінарні події, модні покази, благодійні заходи та багато іншого.

- Рекомендації. Eventbrite може пропонувати персоналізовані рекомендації на основі ваших попередніх пошуків, придбаних квитків та вподобань.

- Інформація про подію. Для кожної події надається детальна інформація, включаючи опис, дату та час, місце проведення, програму (якщо є), список спікерів (для конференцій), фотографії та відео (якщо є), інформацію про організатора та відгуки (якщо є).

- Інтерактивна карта. Місцезнаходження події відображається на карті, що дозволяє легко знайти шлях до місця проведення.

- Купівля квитків. Eventbrite пропонує зручний процес купівлі квитків онлайн. Користувачі можуть вибирати кількість квитків, тип квитків (наприклад, VIP, загальний вхід), та безпечно оплачувати їх за допомогою різних методів оплати (наприклад, кредитна картка, PayPal).

- Мобільні квитки. Після покупки квитки зазвичай доступні у вигляді електронних квитків, які можна показати на екрані телефону або роздрукувати.

- Управління квитками. Користувачі можуть переглядати свої замовлення, завантажувати квитки, переносити квитки іншим людям (якщо це дозволено організатором) та отримувати сповіщення про зміни у події.

- Інтеграція з соціальними мережами: Користувачі можуть ділитися інформацією про події в соціальних мережах.

Для організаторів Eventbrite також пропонує інструменти для організаторів подій, включаючи створення сторінок подій, продаж квитків, просування подій та управління відвідувачами.

Airbnb Experiences: (https://www.airbnb.com/experiences) - розділ Airbnb, де місцеві жителі пропонують унікальні враження та активності для туристів та місцевих. Airbnb Experiences – це розділ на платформі Airbnb, який відрізняється від основного сервісу оренди житла. Замість проживання, він пропонує користувачам можливість бронювати унікальні заходи та активності, які проводять місцеві жителі. Ці "враження" (experiences) можуть варіюватися від екскурсій та майстер-класів до пригодницьких походів та кулінарних класів.

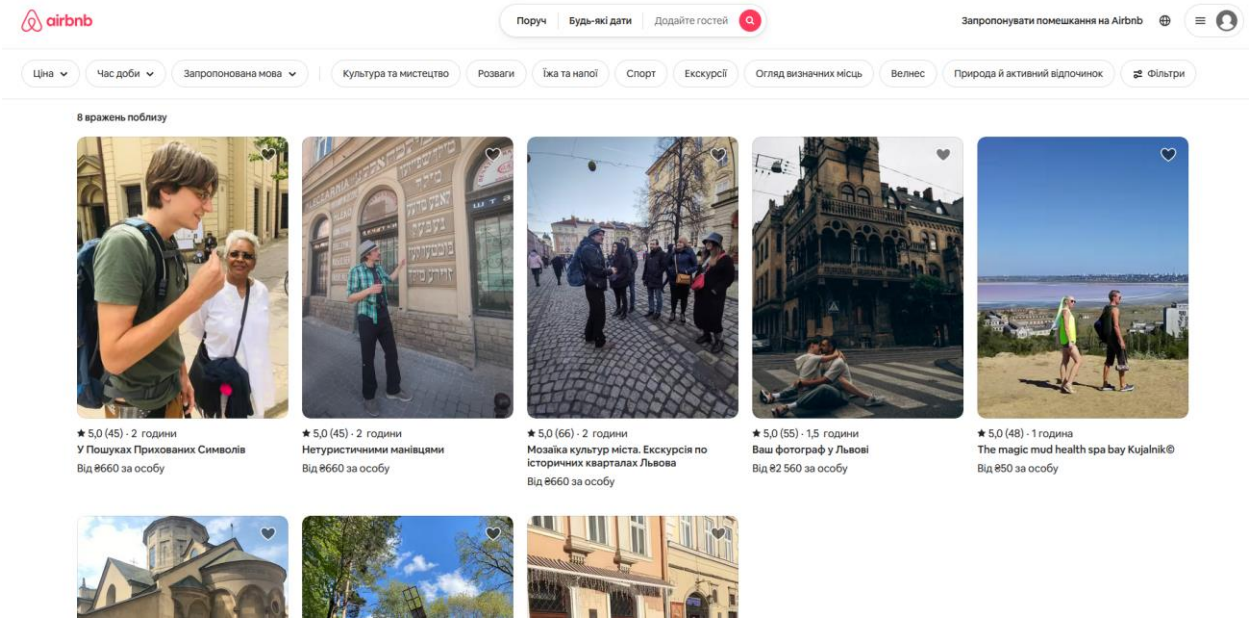

Рисунок 1.11 – Вигляд платформи Airbnb Experiences

Розглянемо ключові особливості та характеристики Airbnb Experiences. Головна особливість полягає в тому, що враження організовують та проводять місцеві експерти, ентузіасти або професіонали у своїй галузі. Це дає можливість отримати автентичний погляд на місто чи регіон та дізнатися про нього зсередини.

Спектр пропонованих вражень надзвичайно широкий. Він може включати:

- Екскурсії: Пішохідні, велосипедні, тематичні екскурсії містом, історичними місцями, природними заповідниками тощо.

- Майстер-класи: Кулінарні уроки, художні майстер-класи (живопис, кераміка), ремісничі заняття, фотографія тощо.

- Пригоди: Походи в гори, сплави на байдарках, скелелазіння, катання на конях, водні види спорту.

- Культурні заходи: Відвідування місцевих вистав, концертів, традиційних церемоній.

- Велнес: Заняття йогою, медитацією, спа-процедури.

- Дегустації: Винні тури, дегустації місцевих продуктів, пива, кави.

- Фотосесії: Професійні фотосесії в унікальних локаціях.

- Соціальні активності: Вечері з місцевими, знайомства з культурою через особисте спілкування.

Часто враження проводяться для невеликих груп людей, що забезпечує більш інтимну та інтерактивну атмосферу. Як і з житлом на Airbnb, користувачі можуть залишати відгуки та ставити оцінки за отримані враження, що допомагає іншим при виборі. Процес бронювання є простим та зручним, здійснюється безпосередньо через платформу Airbnb.

Airbnb Experiences також є можливістю для місцевих жителів поділитися своїми знаннями, пристрастями та культурою з іншими, а також отримати додатковий дохід.

Таким чином, Airbnb Experiences – це платформа, яка з'єднує мандрівників та місцевих жителів через спільні активності, пропонуючи більш глибоке та автентичне знайомство з містом чи регіоном, ніж традиційні туристичні послуги.

Viator (TripAdvisor): (https://www.viator.com/) - ще одна платформа для бронювання екскурсій та вражень. Viator – це онлайн-платформа, яка належить TripAdvisor і спеціалізується на пропонуванні та бронюванні

екскурсій, турів, квитків на атракції та інших видів туристичних вражень по всьому світу.

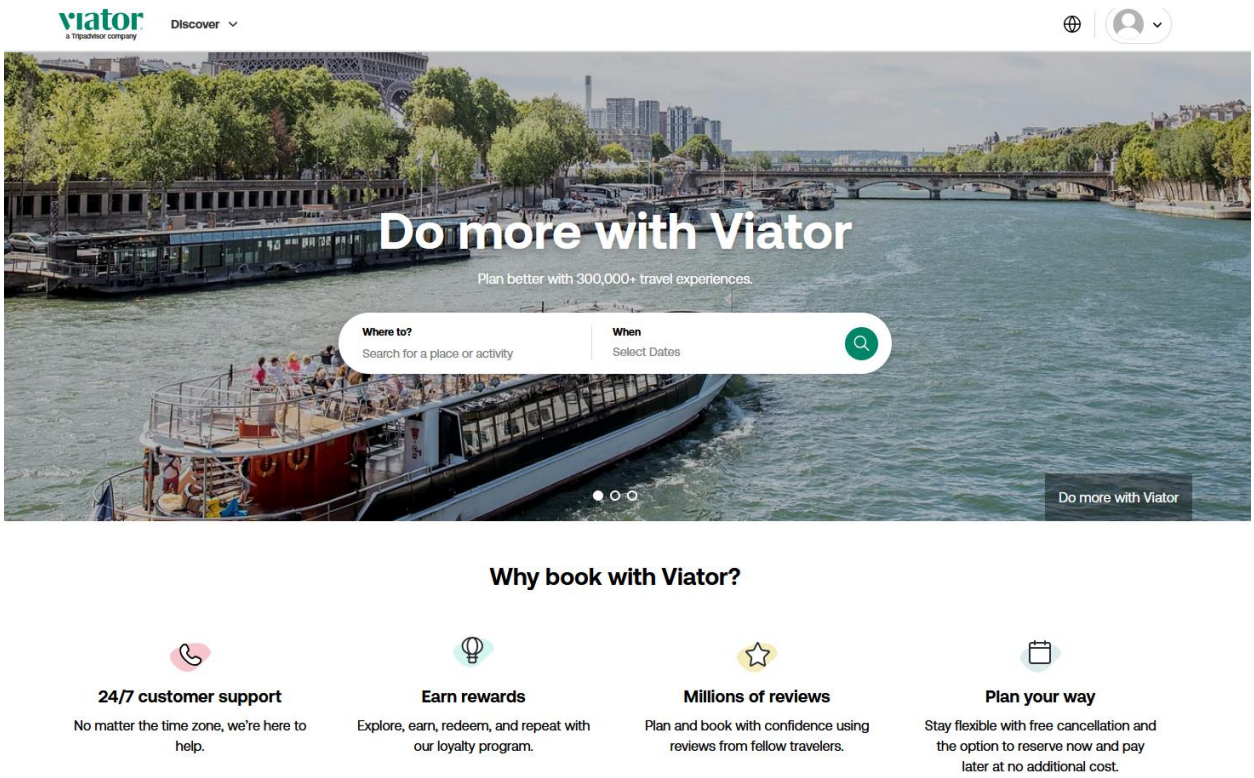

Рисунок 1.12 - Онлайн-платформа Viator

Розглянемо основні характеристики та можливості Viator. Viator має дуже широкий вибір екскурсій, турів, активностей та квитків, охоплюючи практично всі популярні туристичні напрямки. Ви можете знайти все: від оглядових екскурсій містом та відвідування музеїв до екстремальних пригод та кулінарних майстер-класів.

Платформа пропонує різні види вражень, включаючи:

- Екскурсії (пішохідні, автобусні, велосипедні, тематичні)

- Тури (багатоденні, одноденні, індивідуальні, групові)

- Квитки на атракції (музеї, парки розваг, зоопарки)

- Активності (водні види спорту, катання на конях, польоти на повітряній кулі)

- Кулінарні враження (дегустації, кулінарні класи)

- Шоу та концерти

- Трансфери

Viator використовує систему відгуків та рейтингів TripAdvisor, що дозволяє користувачам приймати обґрунтовані рішення, базуючись на досвіді інших туристів. Кожна пропозиція має детальний опис, включаючи маршрут, тривалість, інформацію про гіда, що включено у вартість, фотографії та відгуки.

Процес бронювання є простим та безпечним. Можна переглянути наявність, вибрати дату та час, і оплатити онлайн. Viator часто пропонує гарантію найкращої ціни, тобто якщо ви знайдете ту саму екскурсію дешевше, вам повернуть різницю.

Багато пропозицій мають гнучкі умови скасування, що дозволяє повернути кошти, якщо ваші плани зміняться. Viator має мобільний додаток, який дозволяє зручно переглядати та бронювати екскурсії на ходу. Платформа доступна багатьма мовами.

Viator є хорошим вибором для тих, хто шукає широкий вибір екскурсій та активностей по всьому світу з можливістю порівняння цін, перегляду відгуків та зручного бронювання.

GetYourGuide: (https://www.getyourguide.com/) - сайт для бронювання екскурсій, турів, квитків на атракції та інших видів активностей у різних містах. GetYourGuide – це онлайн-платформа, яка пропонує користувачам широкий вибір екскурсій, турів, квитків на різноманітні атракції та інші туристичні активності по всьому світу. За своїм функціоналом та призначенням вона дуже схожа на Viator, але має свої особливості.

Проаналізуємо основні характеристики та можливості GetYourGuide.

GetYourGuide пропонує мільйони пропозицій у тисячах напрямків по всьому світу. Ви можете знайти практично будь-яку активність, яка вас цікавить, від оглядових екскурсій та відвідування історичних пам'яток до екстремальних пригод та кулінарних майстер-класів.

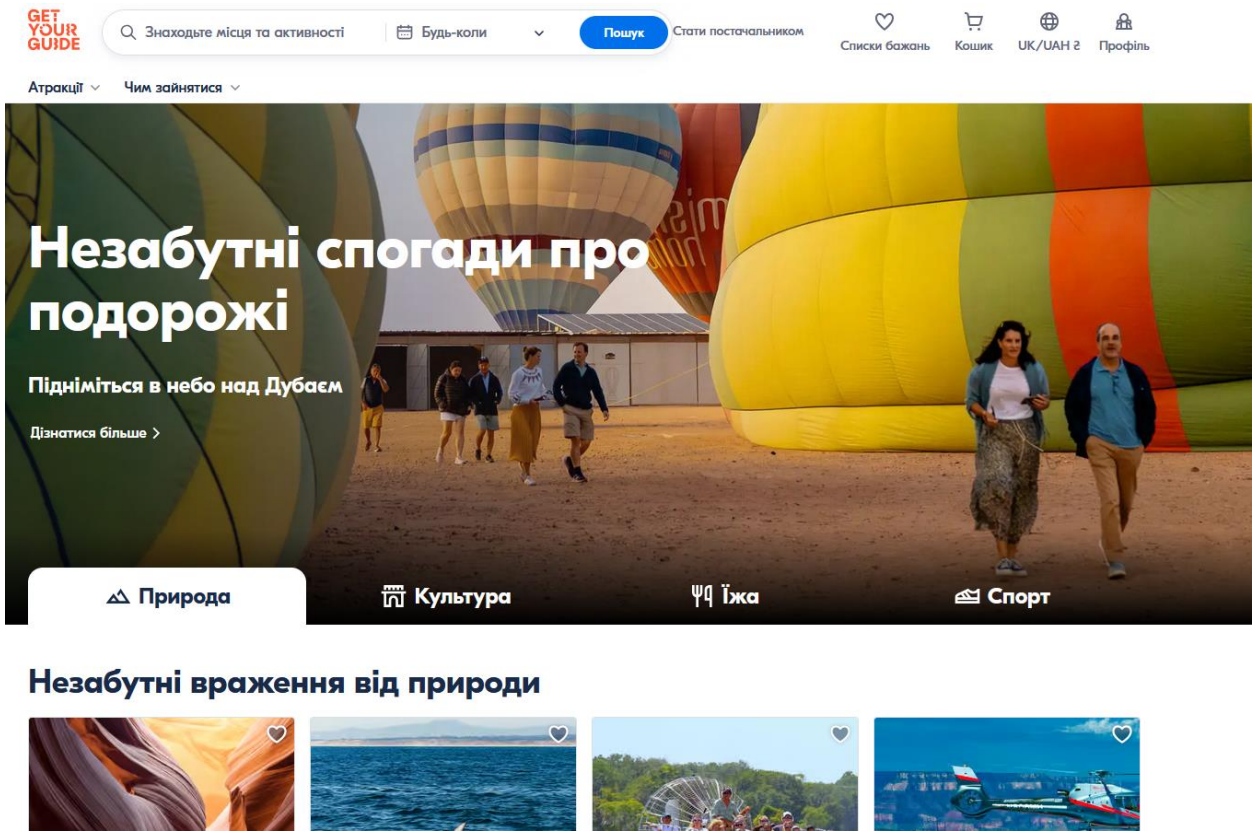

Рисунок 1.13 - Платформа для бронювання екскурсій, турів, квитків GetYourGuide

Активності на GetYourGuide поділені на зручні категорії, такі як:

- Екскурсії та огляди міст
- Квитки на атракції (музеї, зоопарки, тематичні парки)
- Тури на природі та активний відпочинок
- Водні види спорту
- Кулінарні тури та дегустації
- Культурні та історичні екскурсії
- Шоу та концерти
- Трансфери та транспорт

Кожна пропозиція містить вичерпний опис, включаючи маршрут, тривалість, що входить у вартість, інформацію про гіда, доступні мови, фотографії та відгуки клієнтів. Бронювання відбувається онлайн у кілька

кліків. Користувачі можуть легко вибрати дату, час, кількість учасників та оплатити за допомогою різних платіжних методів.

Система відгуків дозволяє користувачам оцінювати свій досвід та ділитися враженнями, допомагаючи іншим приймати зважені рішення. GetYourGuide часто пропонує гарантію найкращої ціни, обіцяючи повернути різницю, якщо ви знайдете ту саму пропозицію за нижчою ціною в іншому місці. Багато пропозицій мають гнучкі умови скасування, що дозволяє отримати повне відшкодування при скасуванні заздалегідь (зазвичай за 24-48 годин до початку).

GetYourGuide має зручний мобільний додаток для iOS та Android, що дозволяє бронювати та керувати своїми бронюваннями в дорозі. Платформа доступна багатьма мовами, що робить її зручною для користувачів з усього світу.

Основна відмінність GetYourGuide може полягати в акценті на інтуїтивно зрозумілому інтерфейсі та можливості легко знайти та забронювати саме те, що вам потрібно, завдяки добре продуманій системі фільтрів та категорій.

1.3. Актуальність і мета проекту побудови платформи представлення ресурсів міста

Після аналізу багатьох ресурсів ми побачили, що переважно вони спеціалізуються на одному типу надання послуг. Мета цього проекту — створити єдине джерело інформації, яке люди можуть використовувати для доступу до різних ресурсів міста. Людям потрібен швидкий доступ до ресурсів, таких як залізниці, тарифи на міський транспорт, екстрені контакти, місця для пікніків, школи. Оскільки Android є найбільш використовуваною мобільною операційною системою, це було б чудовим способом створити додаток, який використовує цю операційну систему для охоплення

максимальної аудиторії. Основна ідея — створити прозору та надійну систему, якою люди можуть користуватися щодня без проблем.

Андроїд-додаток допоможе людям отримувати інформацію про ресурси у місті. Основна мета додатку — надати належну систему для розкладу транспорту, а також додаток включає іншу інформацію, таку як міський транспорт, правила дорожнього руху, новини, екстрені контакти, поліцейські дільниці, місця для пікніків, кінотеатри та ресторани. Ця створена платформа дозволить дізнатися більше про місто.

1.4. Висновки до розділу

В даному розділі здійснено аналіз існуючих рішень показав, що більшість наявних онлайн-сервісів мають вузьку спеціалізацію — зокрема, інструменти планування маршрутів громадського транспорту, сервіси пошуку закладів харчування чи культурних подій. Водночас відсутня єдина інтегрована система, яка б охоплювала різні сфери міського життя в одному зручному інтерфейсі.

Розгляд особливостей функціонування подібних ресурсів дозволив окреслити основні вимоги до платформи: інтерактивність, актуальність даних, геолокаційна підтримка, можливість багаторівневої фільтрації та персоналізації інформації.

Таким чином, було підтверджено доцільність та актуальність розробки єдиної платформи представлення інфраструктури міста, яка об'єднуватиме ключові інформаційні ресурси для зручності мешканців і гостей міста. Сформульовано мету подальшої роботи — створення системи, що забезпечить інтеграцію та візуалізацію різнопланових міських сервісів у доступному цифровому форматі.

РОЗДІЛ 2. АЛГОРИТМІЧНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ ПРЕДСТАВЛЕННЯ РЕСУРСІВ МІСТА ТА ВИБІР ПРОГРАМНИХ ТЕХНОЛОГІЙ

2.1. Вибір та опис інтстументів для реалізації

2.1.1. Android SDK (Software Development Kit) та Android Studio

Android SDK (Software Development Kit) — це комплекс інструментів для розробників, необхідний для створення додатків під операційну систему Android. Його можна розглядати як набір будівельних блоків та інструкцій, які дозволяють програмістам писати, тестувати та налагоджувати програмне забезпечення для пристроїв на базі Android.

Основні компоненти Android SDK включають:

- Бібліотеки Android API. Набір попередньо написаного коду, який надає доступ до функціональності пристрою (камера, GPS, контакти тощо) та інтерфейсу Android.

- Інструменти розробки. Компілятори, налагоджувачі, емулятори пристроїв, інструменти для пакування та підписання додатків.

- Документація. Детальні описи API, посібники та приклади коду, що допомагають розробникам у процесі створення додатків.

- Приклади коду. Готові фрагменти коду, які демонструють використання різних API та функцій Android.

- Емулятори. Програмні середовища, які імітують роботу реальних Android-пристроїв на комп'ютері розробника, дозволяючи тестувати додатки без фізичного пристрою.

- Інструменти для оновлення та керування SDK. Утиліти для завантаження нових версій SDK, встановлення додаткових компонентів та керування встановленими пакетами.

Android SDK надає розробникам гнучкість та контроль над процесом створення додатків, дозволяючи використовувати різні мови програмування (хоча переважно Java та Kotlin) та інтегрувати різноманітні бібліотеки та фреймворки.

Android Studio — це офіційне інтегроване середовище розробки (IDE) для створення Android-додатків, розроблене Google на базі IntelliJ IDEA. Це потужний та зручний інструмент, який об'єднує всі необхідні засоби для розробки в одному інтерфейсі, значно спрощуючи та прискорюючи процес створення додатків.

Основні можливості Android Studio включають:

- Інтелектуальний редактор коду. Підсвічування синтаксису, автодоповнення коду, перевірка помилок в режимі реального часу, рефакторинг коду.

- Візуальний редактор макетів (Layout Editor). Інтуїтивно зрозумілий інтерфейс для створення графічного інтерфейсу користувача за допомогою перетягування елементів (drag-and-drop) або редагування XML-коду.

- Вбудовані інструменти для налагодження (Debugger). Покрокове виконання коду, встановлення точок зупинки, перегляд значень змінних для виявлення та усунення помилок.

- Інтеграція з Android SDK. Автоматичне керування встановленими версіями SDK та доступ до всіх його інструментів.

- Швидкий емулятор. Вбудований емулятор Android з різними конфігураціями пристроїв для тестування додатків.

- Інструменти для профілювання. Аналіз продуктивності додатку, використання пам'яті та енергоспоживання.

- Підтримка Gradle. Автоматизація процесу збірки проєкту та керування залежностями.

- Інтеграція з системою контролю версій (Git). Зручна робота з репозиторіями Git безпосередньо з IDE.

- Інструменти для тестування. Підтримка юніт-тестів, інтеграційних тестів та UI-тестів.

- Підтримка Kotlin. Повна підтримка сучасної мови програмування Kotlin, яка офіційно рекомендована Google для розробки під Android.

Android SDK — це набір інструментів та бібліотек, необхідних для розробки. Це як набір окремих деталей та інструкцій. Android Studio — це інтегроване середовище розробки (IDE), яке об'єднує ці інструменти SDK в одному зручному інтерфейсі, надаючи розробнику комплексне робоче місце. Це як майстерня, де всі необхідні інструменти вже організовані та готові до використання.

Для розробки Android-додатків необхідно мати встановлений Android SDK. Android Studio є рекомендованим та найбільш зручним інструментом для роботи з цим SDK, значно спрощуючи та оптимізуючи процес розробки. Хоча теоретично можна використовувати інші IDE або інструменти командного рядка для роботи з SDK, Android Studio надає найбільш повний та інтегрований досвід розробки.

На рисунку 2.1 показано уривок головної сторінки веб-сайту. Усі пакети Android SDK включені в цей інструмент.

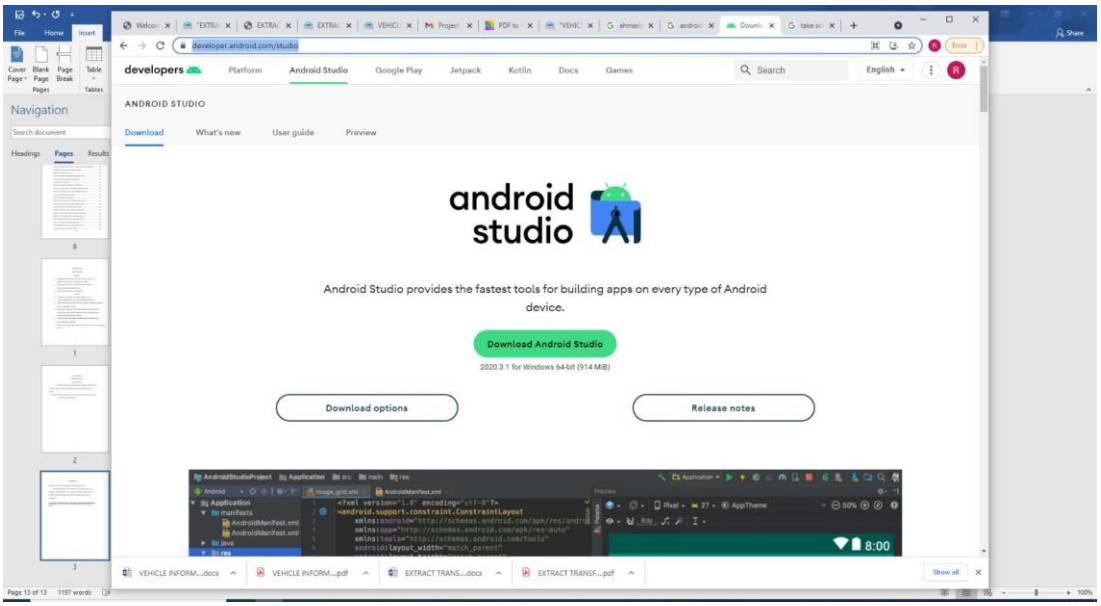

Рисунок 2.1 - Сторінка завантаження Android Studio

Інструменти Android SDK, використовувані разом з Android Studio, надають гнучкість для створення додатку та інтерфейсу користувача за нашого власного зручності. Щоб завантажити Android Studio, треба перейти на веб-сайт розробників Android Studio та завантажити його для користувачів MAC OS та Windows залежно від операційної системи [3].

2.1.2. База даних SQLite

SQLite — це відкрита SQL база даних, яка зберігає дані у текстовому файлі на пристрої. Android має вбудовану реалізацію бази даних SQLite [6]. "Lite" у SQLite означає легковісність, оскільки вона зберігає дані у вигляді текстового файлу, а не в хмарі або на сервері. Ця база даних підтримує всі функції реляційної бази даних (MYSQL, PostgreSQL), і тому запити здійснюються за допомогою загальних SQL-запитів. Нам не потрібно створювати спеціальний з'єднання з JDBC (Java Database Connectivity) або ODBC (Open Database Connectivity) і ми можемо використовувати вбудовані класи SQL для виконання наших операцій.

SQLite — це вбудована, реляційна база даних, що зберігається у файлі. На відміну від традиційних клієнт-серверних СУБД (систем керування базами даних), таких як MySQL або PostgreSQL, SQLite не потребує окремого серверного процесу для роботи. Замість цього, вся база даних (включаючи таблиці, індекси, тригери та дані) зберігається в одному звичайному файлі на диску.

Наведемо ключові характеристики та особливості SQLite:

- Вбудована (Embedded): Бібліотека SQLite безпосередньо інтегрується в програму, яка її використовує. Це означає, що програма сама керує файлом бази даних, без необхідності зв'язку з окремим сервером.

- Безсерверна (Serverless): Відсутність окремого серверного процесу робить SQLite дуже простою у використанні та налаштуванні. Не потрібно встановлювати, налаштовувати та адмініструвати сервер СУБД.

- Самодостатня (Self-Contained): Вся база даних зберігається в одному файлі, що полегшує її перенесення, копіювання та резервне копіювання.

- Кроссплатформенна (Cross-Platform): Бібліотека SQLite доступна для багатьох операційних систем (Windows, macOS, Linux, Android, iOS тощо) та мов програмування.

- Повністю ACID-сумісна (Atomicity, Consistency, Isolation, Durability): Гарантує надійність транзакцій. Кожна транзакція є атомарною (або виконується повністю, або не виконується взагалі), база даних завжди перебуває у консистентному стані, транзакції ізольовані одна від одної, а зміни є стійкими після їх завершення.

- Підтримує стандартний SQL (з деякими особливостями): SQLite підтримує більшість стандартних SQL-команд (SELECT, INSERT, UPDATE, DELETE, CREATE TABLE, ALTER TABLE, індекси, тригери, представлення тощо), але має деякі відмінності у синтаксисі та функціональності порівняно з більш потужними СУБД.

- Відсутність вбудованої підтримки багатокористувацького доступу: Оскільки SQLite є файловою базою даних, одночасний запис кількома процесами може призвести до проблем з цілісністю даних. Хоча існують механізми для часткового вирішення цієї проблеми (наприклад, WAL - Write-Ahead Logging), SQLite не призначена для висококонкурентних багатокористувацьких середовищ.

SQLite можна використовувати для наступних проектів:

- Мобільні додатки (Android, iOS): Завдяки своїй легкості та вбудованості, SQLite є популярним вибором для локального зберігання даних у мобільних додатках.

- Вбудовані системи: Використовується в різноманітних вбудованих пристроях, таких як телевізори, роутери, медичне обладнання тощо.

- Невеликі настільні додатки: Для зберігання конфігурації, локальних даних користувача тощо.

- Веб-браузери: Деякі веб-браузери використовують SQLite для зберігання історії відвідувань, cookie-файлів та інших локальних даних.

- Тестування та розробка: Легкість налаштування робить SQLite зручною для використання в якості тимчасової бази даних під час тестування та розробки.

Переваги SQLite наступні:

- Простота використання та налаштування.

- Не потребує окремого сервера.

- Легка та має низькі вимоги до ресурсів.

- Кроссплатформенність.

- ACID-сумісність.

Інтерфейс БД SQLite подано на рисунку 2.2.

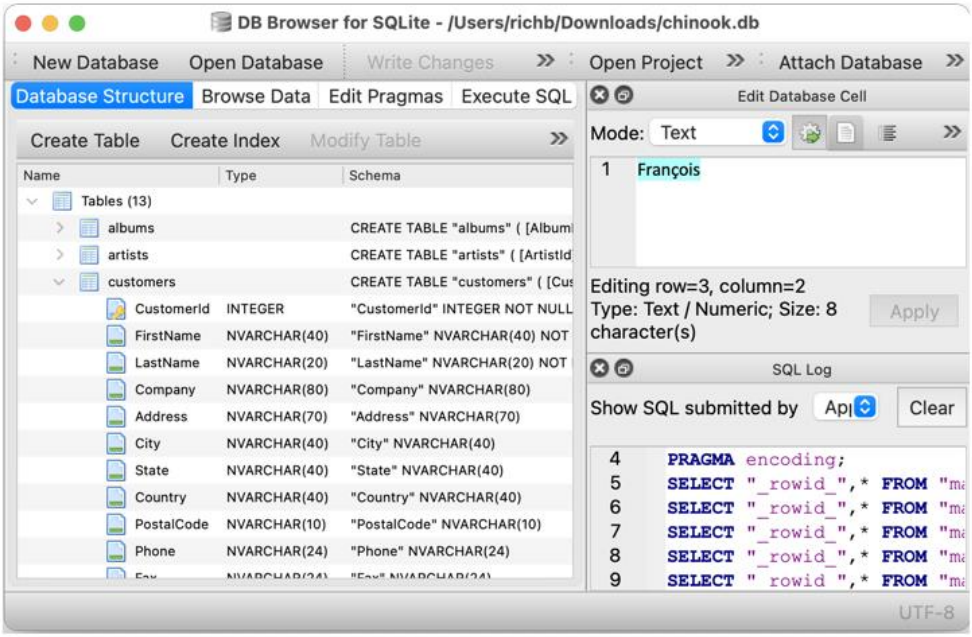

Рисунок 2.2 – Інтерфейс SQLite

2.1.3. Мова розмітки XML

XML (Extensible Markup Language) — це мова розмітки, яка використовується для визначення макетів у Android. У Android ми можемо визначити макет у двох способах:

- Оголошення елементів інтерфейсу користувача в XML

Цей додаток використовує оголошення елементів інтерфейсу користувача в XML для створення та оформлення видів. Цей тип розробки інтерфейсу користувача в Android пропонує простий XML-словник, який включає види та фрагменти, що відповідають класам View та їх підкласам, таким як віджети та макети. Це проектування здійснюється шляхом ручного написання атрибутів для певного класу. На рисунку 2.3 показано, як ми використовували ImageView, інтегрований у LinearLayout, щоб продемонструвати зображення, які ми використовували на головній сторінці додатку.

```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="120dp"
    android:paddingRight="5dp"
    android:paddingLeft="0dp"
    android:layout_weight="1">
    <ImageView
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:src="@drawable/express"
        android:id="@+id/express"
        style="@style/DefaultButton"
        android:paddingLeft="5dp"
        android:layout_weight="1"/>
    <ImageView
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:src="@drawable/auto"
        android:id="@+id/auto"
        android:paddingLeft="5dp"
        style="@style/DefaultButton"
        android:layout_weight="1"/>
    <ImageView
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:src="@drawable/traffic_violation_fines"
        android:id="@+id/traffic_violation_fines"
        style="@style/DefaultButton"
        android:paddingLeft="5dp"
        android:layout_weight="1"/>
</LinearLayout>
```

Рисунок 2.3 - Оголошення елементів інтерфейсу користувача в XML

- Редактор макетів Android Studio

Інший спосіб створення інтерфейсу користувача додатку — використовувати інтерфейс перетягування. Android Studio пропонує вбудовану екосистему, яка додає код для певного виду, коли користувач перетягує та переміщує вид у консоль редактора.

На рисунку 2.4 показано редактор макетів для функції перетягування.

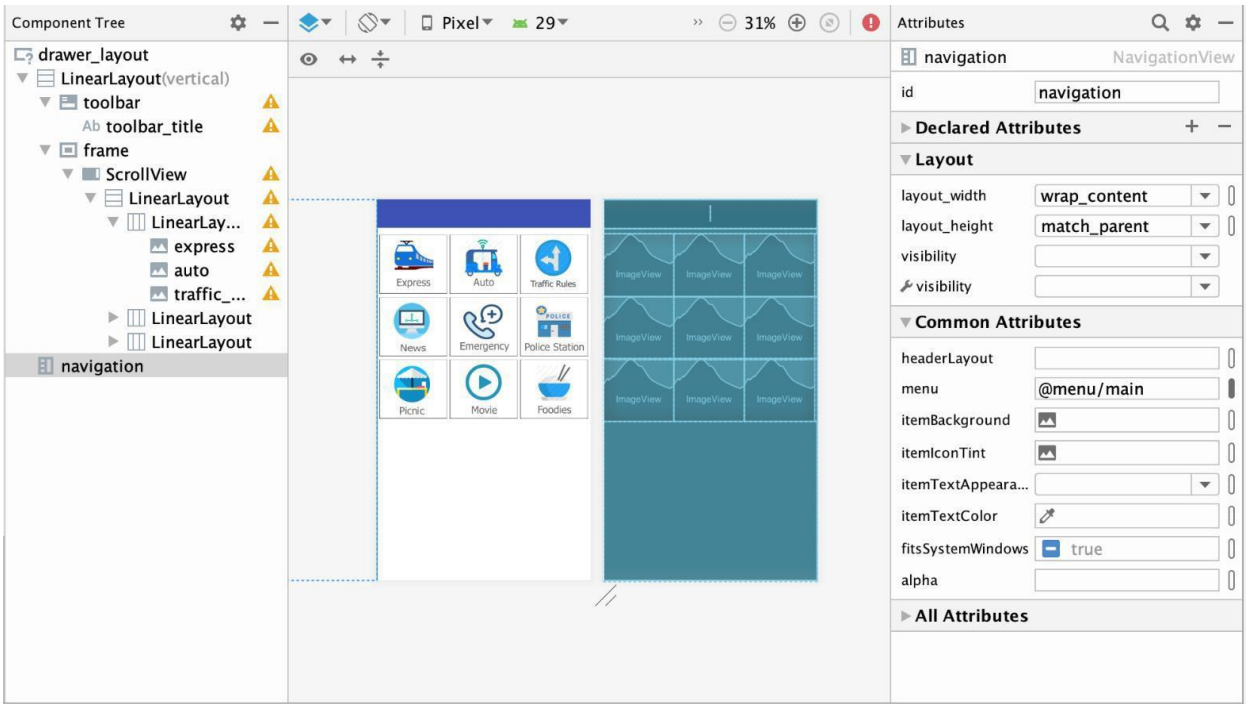

Рисунок 2.4 – Редактор макета Android Studio

2.2. Вибір та опис шаблону проектування

Шаблон проектування MVC (Model View Controller) використовується в модулі щодо залізниць додатку. Шаблони проектування є основою будь-якого процесу розробки програмного забезпечення. Щоб написати хороший додаток, потрібно використовувати хорошу граматику і використовувати відповідний шаблон проектування. Шаблони проектування допомагають покращити ефективність додатку та збільшити його тестуваність. Модель, вид і контролер — три складові цього шаблону. MVC сприяє модульній

розробці завдяки цим особливостям, і він значно покращує ефективність розробки системи, підтримуваність та повторне використання коду, що може задовольнити потреби проектування більш складних багатошарових систем [1]. Цей шаблон є прикладом тристоронньої факторизації, коли об'єкти різних класів беруть на себе дії, пов'язані з доменною областю додатку (модель), відображенням стану програми (вид) та взаємодією користувача з моделлю та видом (контролер) [4]. На рисунку 2.5 показано просту демонстрацію шаблону проектування MVC.

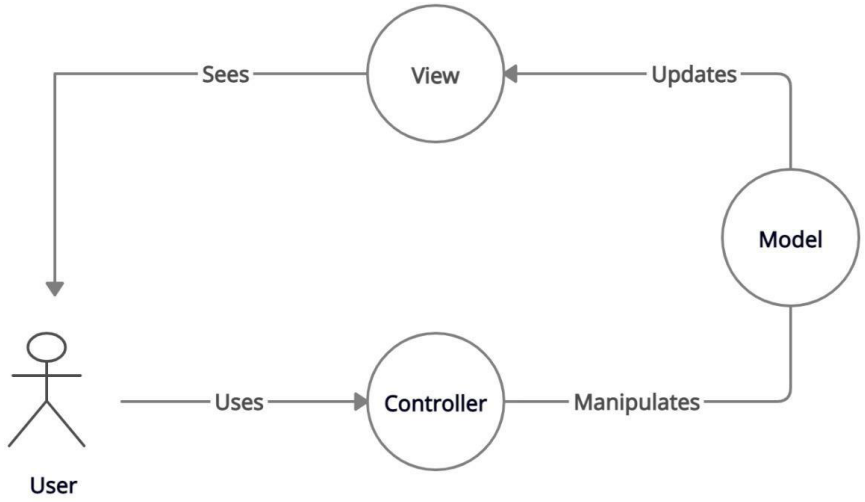

Рисунок 2.5 – Шаблон проектування Model View Controller

Модель містить лише дані додатку, але не логіку. У Java це називається класом POJO (Plain Old Java Objects). Поведінка, необхідна моделям, — це здатність мати залежних і здатність транслювати повідомлення про зміни своїм залежним (вид і контролер) у цьому випадку [4].

Модель представляє дані, бізнес-логіку та правила роботи з даними. Вона відповідає за управління структурою даних, їх зберігання, обробку та логіку програми.

Основні функції:

- Зберігання даних (наприклад, у базі даних, файлах чи пам’яті).

- Обробка запитів на отримання, оновлення чи видалення даних.

- Забезпечення цілісності даних і виконання бізнес-правил.

- Повідомлення Вигляду або Контролера про зміни даних (зазвичай через механізми спостереження, наприклад, події чи сигнали).

На рисунку 2.6 показано приклад класу даних для модуля залізниць, який містить змінні екземпляра, такі як призначення, тип і назва поїзда. Він не містить жодної бізнес-логіки, тим самим повністю інкапсулюючи клас даних. Цей клас даних відповідає за отримання та встановлення даних з бази даних за допомогою методів getter і setter, описаних у коді нижче. Клас даних передає дані контролеру (Railways Adapter), який потім передає їх виду. Відтак вид відповідає за їх правильне відображення користувачеві. Оскільки модель даних не має напрямку з видом, кажуть, що модель і вид у шаблоні MVC слабко зв'язані, на відміну від інших шаблонів проектування, таких як MVP (Model-View-Presenter) або MVVM (Model-View-ViewModel).

```
package com.rushabh.nasikapp.ahmedabadexplorer.Express;
class Express {
    private String desc;
    private String title;
    private String type;
    public String getType() {
        return this.type;
    }
    public void setType(String type) {
        this.type = type;
    }
    public String getTitle() {
        return this.title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getDesc() {
        return this.desc;
    }
    public Express(String title, String desc, String type() {
        this.title = title;
        this.desc = desc;
        this.type = type;
    }
}
```

Рисунок 2.6 – Приклад класу для модуля залізниць

View (Вигляд) - відповідає за відображення даних користувачу. Це інтерфейс, з яким взаємодіє користувач, представлений у вигляді графічного інтерфейсу, веб-сторінки чи іншого візуального представлення.

Функції:

- Відображення даних, отриманих від Моделі, у зрозумілому для користувача форматі.

- Забезпечення інтерактивності (наприклад, кнопки, форми).

- Оновлення інтерфейсу при зміні даних у Моделі.

- Зазвичай не містить бізнес-логіки, а лише відображає те, що передає Модель.

Усі діяльності та файли макетів XML у проекті є частиною виду в шаблоні проектування MVC. Простіше кажучи, вид — це те, що користувач бачить на екрані мобільного пристрою. Вид представляє дані моделі користувачеві. Він знає, як отримати дані моделі, але не знає, що ці дані означають або що користувач може зробити, щоб маніпулювати ними.

Контролер є посередником між Моделлю та Виглядом. Він обробляє дії користувача, взаємодіє з Моделлю для оновлення даних і передає необхідну інформацію Вигляду для відображення.

Функції:

- Обробка вхідних даних від користувача (наприклад, кліки, введення тексту).

- Виклик відповідних методів Моделі для оновлення чи отримання даних.

- Передача даних із Моделі до Вигляду для відображення.

- Координація взаємодії між Моделлю та Виглядом.

Контролер діє як на модель, так і на вид. Він контролює потік даних в об'єкт моделі та оновлює вид, коли дані змінюються. Він утримує вид і модель окремо. У розділі залізниць адаптер діє як міст між діяльністю та

класом моделі. Модель отримує дані з бази даних SQLite, а контролер потім передає дані виду для їх відображення користувачеві.

На рисунку 2.7 показано діаграму випадків використання високого рівня розділу залізниць додатку. Користувач надає вхідні дані про пункти призначення зупинок виду. Вид з’єднується з контролером, контролер потім просить модель (запит до бази даних SQLite) отримати дані, запитані видом. Модель повертає дані контролеру, який потім надсилає їх виду. У цьому процесі ми можемо бути впевнені, що бізнес-логіка відокремлена від логіки виду. Отже, у майбутньому, коли бізнес-логіка зміниться, класи виду залишаться неушкодженими. Це фактично роз'єднання моделі та виду контролером.

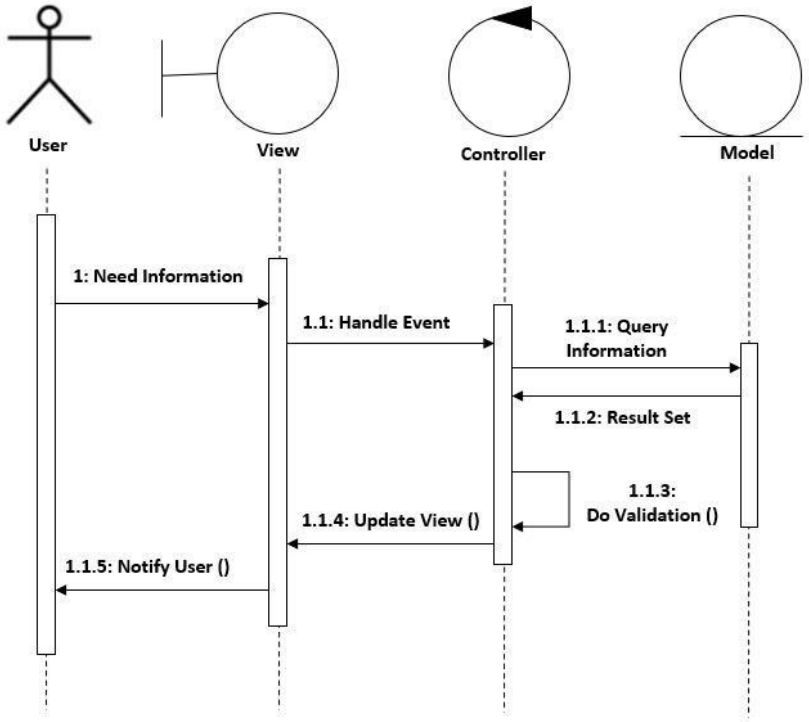

Рисунок 2.7 - MVC Use Case діаграма

Основними перевагами MVC є наступні:

- Кожен компонент має чітко визначену роль, що полегшує розробку, тестування та підтримку.

- Легко змінювати або замінювати один компонент (наприклад, Вигляд) без впливу на інші.

- Модель можна використовувати з різними Виглядами чи Контролерами.

Отже, шаблон MVC є потужним інструментом для створення структурованих і масштабованих додатків, зокрема представлення ресурсів міста. Він дозволяє чітко розділити логіку даних, інтерфейс і обробку дій користувача, що сприяє ефективній розробці та підтримці.

2.3. Реалізація моделі бази даних

Як ми обговорювали раніше, розроблюване програмне забезпечення використовує базу даних SQLite, яка є реляційною базою даних. Android має вбудовану реалізацію бази даних SQLite. Ми розглянемо, як ми використовували базу даних SQLite для нашої користі, дослідивши операції CRUD (Створення, Видалення, Оновлення, Видалення).

```
public void createTables(SQLiteDatabase database) {
    try {
        database.execSQL("create table if not exists express (_id integer p
            "trainid integer," +
            "trainname text," +
            " stationname TEXT," +
            " arrival TIME," +
            " dateplus integer," +
            "sun TEXT, " +
            "mon TEXT, " +
            "tue TEXT, " +
            "wed TEXT, " +
            "thu TEXT, " +
            "fri TEXT, " +
            "sat TEXT)");
    }
    catch(SQLException e){
        System.out.println(e.fillInStackTrace());
    }
}
```

Рисунок 2.8 – Вікно створення бази даних

Усі бібліотеки SQLite знаходяться в пакеті android.database.sqlite. Щоб створити базу даних, просто використовуйте метод openOrCreateDatabase() з ім'ям бази даних та режимом як параметрами. Він повертає екземпляр бази даних SQLite, який ви повинні отримати у власному об'єкті [6]. На рисунку 2.8 ми створюємо таблицю з назвою express, викликаючи метод execSQL() класу SQLiteDatabase. Оскільки блок коду може викликати виняток, якщо таблиця не існує або будь-яка колонка в таблиці написана з помилкою, ми огортаємо його блоком try і catch, щоб обробити виняток, якщо він виникне.

Щоб вставити дані в базу даних, ми використовуємо метод execSQL(), визначений у класі SQLiteDatabase. Це вставить деякі значення в нашу таблицю в нашій базі даних. Інший метод, який виконує ту саму роботу, але приймає додаткові параметри, наведений як execSQL(String sql, Object[] bindArgs) [6]. На рисунку 2.9 показано використання execSQL() для вставки даних у таблицю з назвою express.

```
import android.database.sqlite.SQLiteDatabase;

public class DatabaseHelper {
    private SQLiteDatabase database;

    public void insertTrainData() {
        database.execSQL(
            "INSERT INTO express (" +
                "trainid, trainname, stationname, arrival, dateplus, sun, mon, tue, wed,
            ") VALUES (" +
                "19203, " +
                "'Bandra Terminus Mumbai Bhavnagar', " +
                "'Bandra Terminus', " +
                "'13:15', " +
                "'0', " +
                "'-', " +
                "'Mon', " +
                "'-', " +
                "'-', " +
                "'-', " +
                "'-', " +
                "'-'" +
            ")"
        );
    }
}
```

Рисунок 2.9 – Вікно додавання даних в базу даних

Щоб отримати дані з бази даних, ми можемо використовувати об'єкт класу Cursor у Java. Ми використовуємо метод класу RawQuery, і він

повертає результат набору результатів з курсором, вказаним на таблицю. Ми можемо переміщувати курсор вперед і переміщувати дані вздовж.

Реалізація класу Cursor показана на рисунку 2.10. Якщо кількість об'єктів курсора більше нуля, ми проходимо всі значення об'єкта курсора; в іншому випадку ми просто відображаємо сповіщення з текстом "Назва поїзда не знайдена".

```
Cursor cursor = db.rawQuery(
    "SELECT DISTINCT dl.trainid AS trainid, dl.trainname AS trainname " +
    "FROM express dl " +
    "WHERE dl.trainname LIKE '" + cs + "%'",
    null
);

if (cursor.getCount() > 0) {
    if (cursor.moveToFirst()) {
        do {
            String trainNo = cursor.getString(1);
            String trainName = cursor.getString(1); // Можлива помилка: той самий індекс
            Cursor cursor1 = db.rawQuery(
                "SELECT _id, stationname, arrival, dateplus " +
                "FROM express " +
                "WHERE trainid = " + trainNo,
                null
            );
            cursor1.moveToFirst();
            String src = cursor1.getString(1);
            cursor1.moveToLast();
            express_array.add(
                new Array_Express(
                    trainName,
                    trainNo,
                    src,
                    cursor1.getString(1)
                )
            );
             cursor1.close(); // Рекомендується закривати курсор
        } while (cursor.moveToNext());
    }
    cursor.close(); // Рекомендується закривати курсор
} else {
    Toast toast = Toast.makeText(
        ListofExpress.this,
        "Назва поїзда не знайдена!",
        Toast.LENGTH_SHORT
    );
    toast.setGravity(Gravity.CENTER, 0, 0);
    toast.show();
}
```

Рисунок 2.10 – Вигляд вікна отримання даних з бази даних

Таблиця: express — основна таблиця, яка зберігає інформацію про поїзди, їхні маршрути та розклад.

Поля:

- _id: Унікальний ідентифікатор запису (стандарт для Android SQLite, автоінкрементний).

- trainid: Ідентифікатор поїзда (числовий).

- trainname: Назва поїзда (текст).

- stationname: Назва станції (текст).

- arrival: Час прибуття (текст, наприклад, '13:15').

- dateplus: Поле, ймовірно, для коригування дати чи позначки (текст, наприклад, '0').

- sun, mon, tue, wed, thu, fri, sat: Поля для позначення днів тижня, коли поїзд курсує (текст, наприклад, 'Mon' або '-').

Зв’язки в базі даних.

Один trainid може мати кілька записів у таблиці express (наприклад, для різних станцій на маршруті), оскільки код використовує moveToFirst і moveToLast для отримання початкової та кінцевої станцій. Це вказує на те, що таблиця зберігає розклад поїзда з інформацією про зупинки.

2.4. Висновки до розділу

В даному розділі розглянуто ключові технічні рішення, що стали основою для реалізації мобільного додатку з представлення ресурсів міста. Було обґрунтовано вибір інструментів розробки, серед яких Android SDK та Android Studio як основні засоби для створення мобільного застосунку під операційну систему Android.

З метою зберігання та обробки даних про інфраструктурні об’єкти міста було обрано легку вбудовану базу даних SQLite, що є ефективним рішенням для мобільних додатків з локальним зберіганням інформації. Для

структурного опису інтерфейсу застосовано мову розмітки XML, яка дозволяє забезпечити адаптивність і зрозумілу візуальну побудову елементів UI.

Окрему увагу приділено вибору шаблону проєктування — структурного підходу до організації коду, який забезпечує гнучкість, масштабованість і полегшує супровід додатку.

На завершення було реалізовано модель бази даних, яка дозволяє зберігати, фільтрувати та обробляти інформацію про об’єкти міської інфраструктури. Вказаний підхід є базисом для подальшої реалізації логіки взаємодії користувача з даними у мобільному застосунку.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ ПРЕДСТАВЛЕННЯ РЕСУРСІВ МІСТА

3.1. Реалізація макету за допомогою методу “ListView всередині CardView”

Як ми обговорювали раніше, ми використовуємо XML для проектування фронтенду нашого додатку. Розглянемо, як ми використовуємо деякі ключові елементи XML у нашому додатку для покращення дизайну.

ListView всередині CardView — це сучасний спосіб створення макетів у Android, який робить інтерфейс користувача привабливим. CardView — це вид, який відображає види один на одному. Він застосовний до Android SDK версії 21 і вище. В даній роботі використано цей інструмент у розділах екстрених ситуацій, ресторанів, кінотеатрів та залізниць додатку.

У контексті розробки мобільних додатків під Android, ListView всередині CardView - це поширений спосіб відображення списку елементів (ListView), де кожен елемент або група елементів обернена в контейнер CardView.

Давайте розберемо кожен компонент окремо, а потім розглянемо їх комбінацію.

Особливості CardView наступні:

- Контейнерний елемент. CardView - це компонент UI (інтерфейсу користувача), який використовується для створення карток з закругленими кутами та тінню.

- Візуальне відокремлення. Він надає візуальне відокремлення вмісту, що робить його більш організованим та легким для сприйняття користувачем.

- Ефект підняття. Тінь, яку генерує CardView, створює ефект підняття над фоном, що додає глибини інтерфейсу.

- Групування інформації. CardView часто використовується для групування пов'язаної інформації, наприклад, заголовка, зображення та короткого опису.

Особливості ListView:

- Компонент для відображення списків. ListView - це компонент UI, який використовується для відображення прокручуваних списків однотипних елементів.

- Ефективне відображення великих обсягів даних. ListView оптимізований для відображення великих наборів даних, оскільки він повторно використовує (переробляє) елементи, які стають невидимими під час прокручування, що економить пам'ять та підвищує продуктивність.

- Адаптер для даних: Для відображення даних у ListView використовується спеціальний об'єкт - адаптер, який бере дані з джерела (наприклад, масиву, списку, бази даних) та перетворює їх у представлення для кожного елемента списку.

Рисунок 3.1 – Спрощений вигляд макету ListView всередині CardView

Коли ListView розміщується всередині CardView, це зазвичай робиться для досягнення наступних цілей:

- CardView використовується для візуального об'єднання всього списку елементів. Замість того, щоб відображати довгий неперервний список, CardView створює візуальну межу навколо нього, роблячи його більш відокремленим від іншого вмісту на екрані.

- Кожен CardView може містити окремий, логічно пов'язаний список елементів. Це може бути корисно, наприклад, для відображення різних категорій новин, списків друзів або переліків продуктів.

- CardView дозволяє застосовувати загальні стилі (фон, колір, закруглення кутів, тінь) до всього списку елементів, що значно спрощує оформлення.

Хоча розміщення ListView безпосередньо всередині CardView є можливим, іноді більш ефективним може бути використання RecyclerView замість ListView. RecyclerView є більш гнучким та продуктивним компонентом для відображення списків, особливо великих та динамічних. RecyclerView також може бути легко розміщений всередині CardView.

Даний метод часто використовується для відображення списку останніх новин, де кожна новина (заголовок, зображення, короткий опис) обернена в окремий елемент ListView, а весь список новин може бути обернений в CardView для візуального відділення від іншого контенту. Також цей метод застосовується для відображення списку завдань, де кожне завдання є елементом ListView, а група завдань на певний день може бути обернена в CardView.

Підсумовуючи, ListView всередині CardView - це спосіб візуально згрупувати та відокремити список елементів, надаючи йому стилізований контейнер з закругленими кутами та тінню. Це допомагає покращити візуальну організацію та сприйняття інформації користувачем.

На рисунку 3.2 показано візуальне представлення спрощеного алгоритму методу як ListView розміщується та працює всередині CardView в Android-розробці.

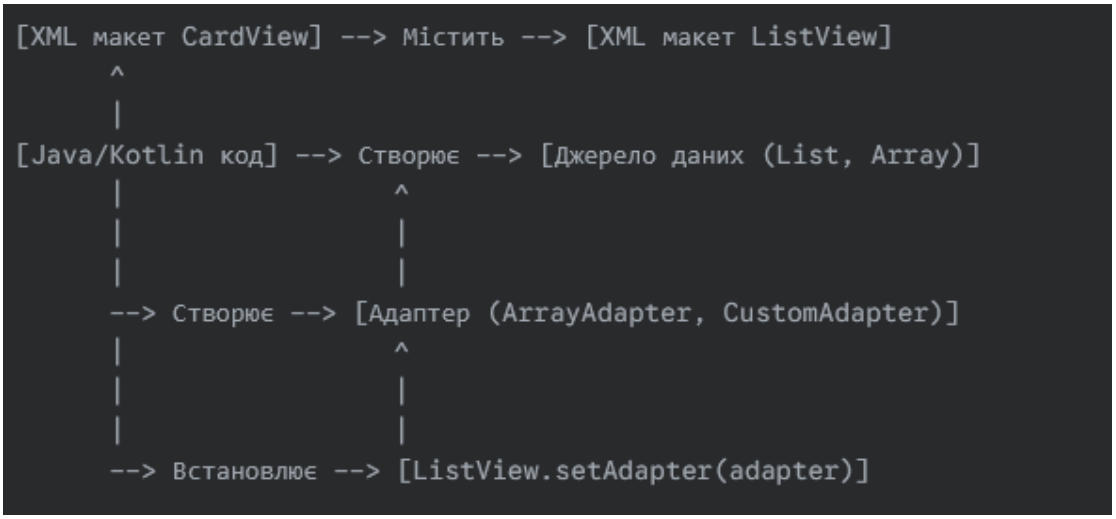

Рисунок 3.2 – Графічне представлення алгоритму роботи способу “ListView всередині CardView”

ListView зазвичай інтегрується в CardView, оскільки потрібен вид, який CardView повинен показати у вигляді картки. На рисунку 3.3 показано приклад демонстрації того, як виглядає CardView разом з ListView.

```
<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/result_card"
    android:layout_width="match_parent"
    android:layout_height="80dp"
    android:elevation="4dp"
    app:cardCornerRadius="4dp"
    android:layout_margin="5dp"
    app:cardBackgroundColor="#fafafa">
    <ListView
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
    </ListView>
</android.support.v7.widget.CardView>
```

Рисунок 3.3 - CardView з фрагментом коду ListView

На рисунку 3.4 показано результат вищезазначеного коду та те, як ми використовували CardView з ListView для покращення нашого інтерфейсу користувача.

Рисунок 3.4 – Приклад реалізації способу “ListView всередині CardView” в пропонованому додатку

3.2. Використання компоненти RecyclerView для відображення даних

RecyclerView в Android - це гнучкий та ефективний компонент для відображення великих наборів даних у вигляді списків або сіток. Він є більш просунутою та рекомендованою заміною для застарілого компонента ListView.

Основна ідея RecyclerView полягає в переробці (recyclying) елементів, які стають невидимими під час прокручування. Замість того, щоб створювати новий View для кожного елемента даних, RecyclerView використовує вже існуючі, але зараз невидимі View, заповнюючи їх новими даними. Це значно економить пам'ять та покращує продуктивність, особливо при роботі з великими списками.

Ключові особливості та переваги RecyclerView:

- Коли елемент списку прокручується за межі екрана, його View не видаляється, а поміщається в пул "перероблених" View. Коли з'являється

новий елемент, замість створення нового View, RecyclerView бере один з перероблених, оновлює його дані та відображає.

- RecyclerView не нав'язує конкретний спосіб відображення елементів. За допомогою LayoutManager можна легко змінювати вигляд списку:

  - LinearLayoutManager: Відображає елементи у вертикальному або горизонтальному лінійному списку (як ListView).
  - GridLayoutManager: Відображає елементи у вигляді сітки з заданою кількістю стовпців або рядків.
  - StaggeredGridLayoutManager: Відображає елементи у вигляді "рваної" сітки, де елементи можуть мати різну висоту або ширину.
  - Можна створювати власні LayoutManager для кастомних розміток.

- RecyclerView чітко розділяє відповідальність між різними класами:

  - RecyclerView: Відповідає за загальне керування списком, переробку View та обробку прокручування.
  - ViewHolder: Клас, який містить посилання на елементи View кожного елемента списку (наприклад, TextView, ImageView). Це допомагає уникнути повторного виклику findViewById() під час прокручування.
  - Adapter: Клас, який відповідає за зв'язування даних з джерела з ViewHolder та відображення їх у View. Він також обробляє створення нових ViewHolder при необхідності.
  - LayoutManager: Визначає, як елементи будуть розташовані на екрані.
  - ItemAnimator: Відповідає за анімацію елементів при їх додаванні, видаленні або зміні.

Основні компоненти RecyclerView наступні:

- RecyclerView. Сам компонент View, який розміщується у макеті.

- ViewHolder. Клас, який розширює RecyclerView.ViewHolder і містить посилання на дочірні View елемента списку. Для кожного типу елемента списку зазвичай створюється свій ViewHolder.

- Adapter: Клас, який розширює RecyclerView.Adapter.

- LayoutManager: Клас, який розширює RecyclerView.LayoutManager і визначає, як елементи будуть розташовані на екрані. Android надає стандартні реалізації (LinearLayoutManager, GridLayoutManager, StaggeredGridLayoutManager).

- ItemAnimator: Клас, який розширює RecyclerView.ItemAnimator і визначає анімацію елементів. За замовчуванням використовується DefaultItemAnimator.

- ItemDecoration: Клас, який дозволяє додавати візуальні роздільники, відступи або інші декорації між елементами списку.

- OnItemTouchListener: Інтерфейс для обробки подій дотику до елементів списку.

Отже, RecyclerView - це потужний та гнучкий компонент для відображення списків у Android, який забезпечує значно кращу продуктивність та більші можливості налаштування порівняно з ListView завдяки механізму переробки View та розділенню відповідальності між різними класами. Він є стандартним і рекомендованим способом відображення спискових даних в сучасній Android-розробці.

Додаток широко використовує Recycler View для заповнення даних у розділі залізниць. Recycler View дозволяє легко подавати великі обсяги даних. Ви просто надаєте дані та визначаєте, як повинні виглядати кожні елементи, а бібліотека Recycler View створює елементи за потребою [3]. Це дозволяє створювати будь-які списки за допомогою XML-макетів як елементи, які можна подальше налаштовувати. Ця функціональність особливо корисна, коли в Recycler View є багато об'єктів. Покращення

досягається завдяки повторному використанню видів, які більше не видимі користувачеві.

У Recycler View є три компоненти:

1. Card Layout. Для макету картки використовується XML-макет. CardView — це вид, який дозволяє відображати багато видів один на одному. Основна мета CardView — допомогти дизайну інтерфейсу користувача мати багатший вигляд. Найкраща особливість CardView полягає в тому, що вона розширює Framelayout і може відображатися на всіх системах Android.

2. View Holder. Це основна відмінність між ListView і RecyclerView. Рекомендується використовувати шаблон проектування View Holder, але це необов'язково. Однак це обов'язково при використанні RecyclerView. ViewHolder [2]. View Holder — це Java-клас, який містить посилання на види макету картки, які потрібно динамічно оновлювати під час виконання програми за допомогою списку даних, зібраних з електронних баз даних або наданих іншим способом.

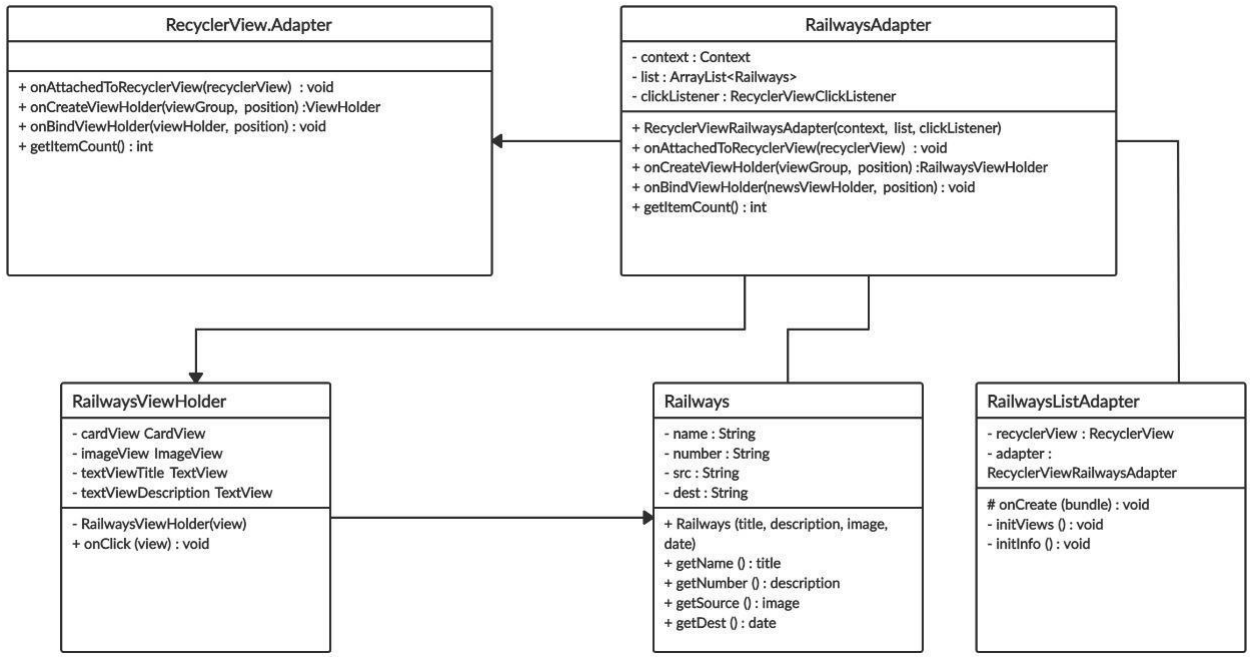

Рисунок 3.5 - Діаграма класу (секція залізниці)

3. Data Class - це тип інформації. Це спеціальний Java-клас, який слугує структурою для зберігання даних для кожного елемента Recycler View. На рисунку 3.5 показано роботу розділу залізниць за допомогою Recycler Adapter. У розділі залізниць, коли користувач вводить початкову та кінцеву зупинки, запускається Railways Adapter. Потім клас даних залізниць (клас POJO) надсилає дані до класу Railways List Adapter, і кожен View Holder прикріплюється до певного елемента виду.

На рисунку 3.6 показано демонстрацію роботи View Holder Recycler та RecyclerView Adapter. Коли користувач вводить кінцеву зупинку на першому зображенні рисунка 3.6, запускається Railways Adapter, який запускає Recycler View Holder і Recycler View Adapter, тим самим показуючи дані, заповнені моделлю (базою даних SQLite). Оскільки кількість поїздів, які повертаються, може бути дуже великою, Recycler View постійно переробляє старі дані та замінює їх новими, коли користувач прокручує Recycler View.

Рисунок 3.6 - Демонстрація ViewHolder для RecyclerView

3.3. Реалізація та опис функціональності інтерфейсу користувача

На рисунку 3.7 представлено простий та лаконічний інтерфейс, мобільного застосунку, який відображає ресурси міста, а саме інформацію

про потяги. Заголовок у верхній частині екрана чітко вказує: "Express Trains" (Швидкісні потяги).

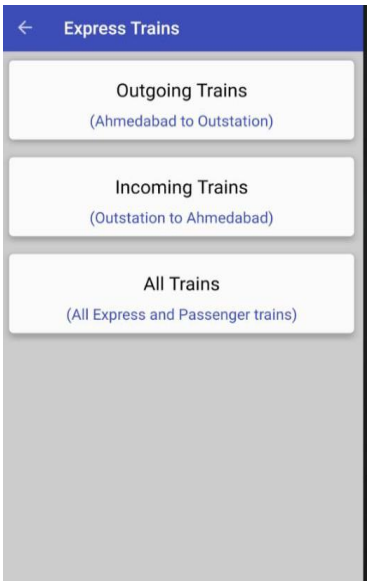

Рисунок 3.7 – Інформація про потяги

Основну частину екрана займають три великі інтерактивні кнопки (картки), розташовані вертикально одна під одною. Кожна кнопка має текстовий опис:

1. Outgoing Trains (Потяги, що відправляються)

Під основним текстом меншим шрифтом вказано уточнення: "(Ahmedabad to Outstation)" (З Ахмедабада до іншого міста). Це вказує на те, що при натисканні на цю кнопку користувач, побачить список швидкісних потягів, які відправляються з міста Ахмедабад в інші населені пункти.

2. Incoming Trains (Потяги, що прибувають)

Під основним текстом меншим шрифтом вказано уточнення: "(Outstation to Ahmedabad)" (З іншого міста до Ахмедабада). Це вказує на те, що при натисканні на цю кнопку користувач, ймовірно, побачить список швидкісних потягів, які прибувають до міста Ахмедабад з інших населених пунктів.

3. All Trains (Усі потяги)

Під основним текстом меншим шрифтом вказано уточнення: "(All Express and Passenger trains)" (Усі швидкісні та пасажирські потяги). Це вказує на те, що при натисканні на цю кнопку користувач, ймовірно, побачить повний список усіх типів потягів (не лише швидкісних), які відправляються або прибувають до міста, або ж просто курсують у межах системи.

При натисканні на кожну з кнопок, призведе до переходу на новий екран зі списком відповідних потягів, де буде детальніша інформація про час відправлення/прибуття, станції, можливі затримки тощо.

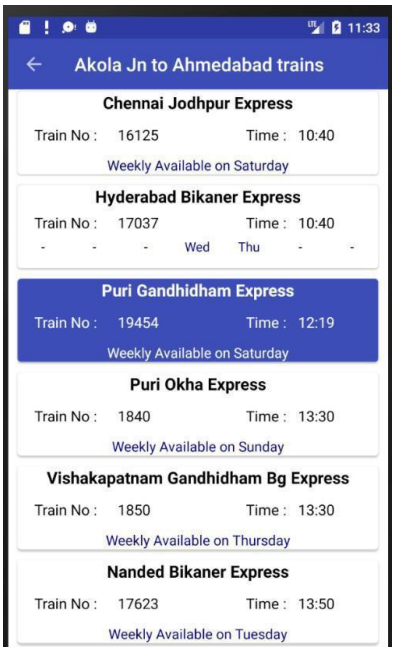

Рисунок 3.7 – Вигляд екрану із деталізацією руху поїздів

На рисунку 3.7 представлено інтерфейс, що відображає список потягів, які курсують за маршрутом від станції "Akola Jn" до станції "Ahmedabad". Заголовок у верхній частині екрана чітко це вказує: "Akola Jn to Ahmedabad trains".

Основну частину екрана займає вертикальний прокручуваний список карток, кожна з яких містить інформацію про окремий потяг. Кожна картка має схожий формат, що полегшує порівняння різних потягів.

Розглянемо структуру інформації в межах однієї картки (на прикладі першого потяга):

- Назва потяга: "Chennai Jodhpur Express" - великим жирним шрифтом вгорі картки.

- Номер потяга: "Train No : 16125" - меншим шрифтом ліворуч.

- Час відправлення (або прибуття): "Time : 10:40" - меншим шрифтом праворуч, вирівняно до правого краю.

- Дні тижня курсування: "Weekly Available on Saturday" - розташовано під номером та часом потяга, вказує, в які дні тижня курсує цей конкретний потяг.

Інші картки в списку відображають аналогічну інформацію для інших потягів, що курсують за вказаним маршрутом.

Натискання на картку потяга, призводить до переходу на екран з більш детальною інформацією про цей конкретний потяг (наприклад, маршрут з усіма зупинками, доступні класи вагонів, вартість квитків тощо).

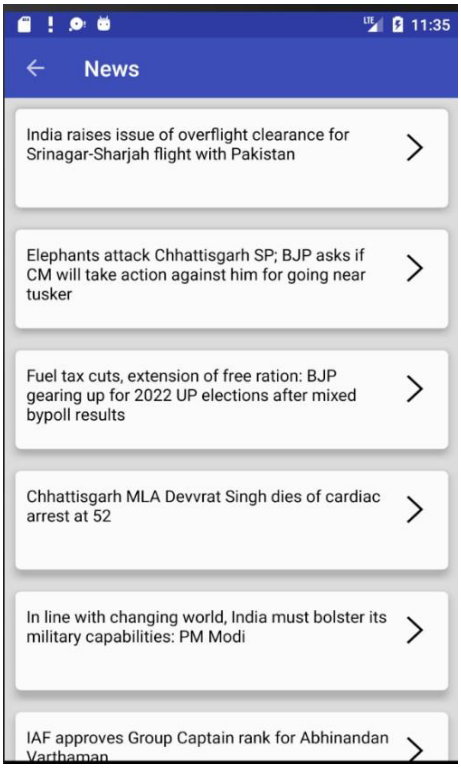

Рисунок 3.8 – Перелік новин міста

На рисунку 3.8 представлено інтерфейс мобільного застосунку, що відображає список новин. Заголовок у верхній частині екрана чітко це вказує "News".

Основну частину екрана займає вертикальний прокручуваний список карток, кожна з яких містить короткий заголовок новини. Кожна картка має схожий формат, що полегшує перегляд.

Розглянемо структуру інформації в межах однієї картки (на прикладі першої новини).

Заголовок новини: "India raises issue of overflight clearance for Srinagar-Sharjah flight with Pakistan" - розміщено в основній частині картки, достатньо великим шрифтом для зручного читання. Індикатор переходу: Праворуч у кожній картці розміщена стрілка "вправо" (>). Цей елемент вказує на те, що натискання на картку призведе до переходу на інший екран з більш детальною інформацією про цю конкретну новину.

Інші картки в списку відображають аналогічну структуру для інших новин.

У верхньому лівому куті присутня стандартна стрілка "назад", що дозволяє повернутися на попередній екран. Використано синій колір для заголовка та білий для тексту на світло-сірому фоні карток, що створює контраст та є візуально приємним.

Натискання на будь-яку з карток новин, призводить до відкриття нового екрана з повним текстом відповідної новини, деякі з додатковими деталями, зображеннями або відео.

На рисунку 3.9 представлено інтерфейс для пошуку закладів харчування або місць, пов'язаних з їжею.

Основну частину екрана займає вертикальний прокручуваний список карток, кожна з яких містить інформацію про окремий заклад. Кожна картка має схожий формат, що полегшує порівняння різних місць.

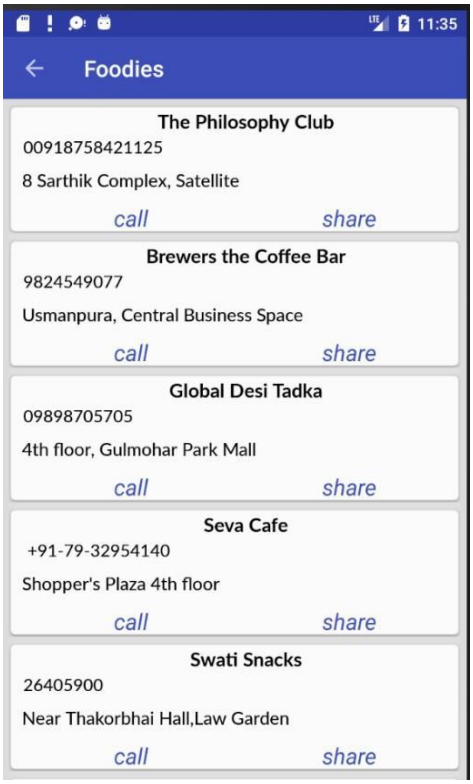

Рисунок 3.9 – Вигляд екрану з переліком закладів харчування міста

Розглянемо структуру інформації в межах однієї картки (на прикладі першого закладу).

- Назва закладу. "The Philosophy Club" - розміщено вгорі картки, достатньо великим та жирним шрифтом.

- Номер телефону. "00918758421125" - розміщено під назвою закладу.

- Адреса. "8 Sarthik Complex, Satellite" - розміщено під номером телефону.

- Дві інтерактивні кнопки. У нижній частині картки розташовані дві кнопки з текстом "call" (зателефонувати) та "share" (поділитися). Ці кнопки, ймовірно, дозволяють користувачеві зателефонувати до закладу або поділитися інформацією про нього.

Інші картки в списку відображають аналогічну структуру для інших закладів. Заклади організовані у вертикальний список, що дозволяє легко прокручувати та переглядати доступні варіанти. Кожен заклад представлений у вигляді окремої картки на світло-сірому фоні, що візуально відокремлює

інформацію та робить її більш структурованою. У верхньому лівому куті присутня стандартна стрілка "назад", що дозволяє повернутися на попередній екран.

При натисканні на кнопку "call" буду ініційовано виклик на вказаний номер телефону закладу. Натискання на кнопку "share", відкриває меню для обміну інформацією про заклад через різні канали (месенджери, соціальні мережі тощо). Натискання на саму картку закладу призводить до переходу на екран з більш детальною інформацією про цей заклад (години роботи, меню, відгуки, фотографії тощо).

3.4. Використання компоненту інтерфейсу Web View

WebView в Android — це компонент інтерфейсу користувача (UI), який дозволяє вбудовувати веб-контент (веб-сторінки, HTML, JavaScript, CSS) у нативний Android-додаток. По суті, WebView є вбудованим браузером, який відображає веб-сторінки або локальний HTML-контент у межах програми, не відкриваючи зовнішній браузер. Це дає змогу створювати гібридні додатки, які поєднують нативні елементи (наприклад, меню, кнопки) із веб-контентом.

WebView є класом у пакеті android.webkit, що успадковується від View, тому його можна додавати до макетів (layouts) як будь-який інший UI-елемент (наприклад, Button або TextView).

Основні характеристики WebView:

- WebView є класом у пакеті android.webkit, що успадковується від View, тому його можна додавати до макетів (layouts) як будь-який інший UI-елемент (наприклад, Button або TextView).

- Відображення веб-сторінок із URL (наприклад, https://example.com).

- Відображення локального HTML-контенту (з ресурсів програми або створеного динамічно).

- Підтримка JavaScript, CSS, мультимедії та інших веб-технологій.

- Взаємодія між веб-контентом і нативним кодом (Java/Kotlin) через JavaScript-інтерфейси.

- WebView підтримує обробку навігації, помилок і взаємодії з користувачем через класи WebViewClient і WebChromeClient;

Переваги WebView

- Легко додати веб-контент без створення складних нативних інтерфейсів.

- Підтримка сучасних веб-технологій (HTML5, CSS3, JavaScript).

- Веб-контент можна використовувати на різних платформах із мінімальними змінами.

- Веб-сторінки можна оновлювати на сервері без оновлення самого додатка.

WebView в Android — це потужний інструмент для вбудовування веб-контенту в нативні додатки, який ідеально підходить для гібридних рішень. Він простий у використанні, але вимагає обережного налаштування для забезпечення безпеки та продуктивності.

Оскільки мобільні пристрої стають все більш поширеними, інтеграція веб-сервісів у мобільні додатки стає необхідністю [7]. WebView — це вид у додатку Android, який відображає веб-сторінки. Об'єкти WebView дозволяють включати веб-контент у макет діяльності, але вони не мають усіх функцій повністю розвинених браузерів. WebView — це чудове рішення, коли вам потрібні більш точні можливості управління інтерфейсом користувача та розширені конфігураційні опції [3]. Він дозволяє інтегрувати веб-сторінки в спеціально створене середовище для вашого проекту. Ми використовуємо WebView у додатку для отримання списку місць для пікніків, отриманих з існуючого API (Application Programming Interface). Щоб додати додаткову безпеку до веб-сторінки, ми використовуємо об'єкт класу WebSettings.

```
String url = "https://www.holidify.com/weekend-getaways/from-ahmedabad.html
@RequiresApi(api = Build.VERSION_CODES.CUPCAKE)
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_bicnic);
    toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    getSupportActionBar().setTitle("Місця для пікніків поблизу Ахмедабада")
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    getSupportActionBar().setDisplayShowHomeEnabled(true);
    webview = (WebView)findViewById(R.id.webView1);
    pbar = (ProgressBar)findViewById(R.id.progressBar1);
    webview.setWebViewClient(new Picnic.WebViewClient());
    webview.loadUrl(url);
}
```

Рисунок 3.9 - Web view в Android

Пояснення методів, оголошених у класі WebSettings(), наведено нижче.

- setJavaScriptEnabled(): Цей метод класу WebSettings повідомляє WebView увімкнути виконання JavaScript.

- setBuiltInZoomControls(): Встановлює, чи повинен WebView використовувати свої вбудовані механізми зуму.

- setDisplayZoomControls(): Встановлює, чи повинен WebView відображати екранні елементи управління зумом при використанні вбудованих механізмів зуму.

- setAppCacheEnabled(): Встановлює, чи повинні бути увімкнені API кешування додатків.

- setDatabaseEnabled(): Встановлює, чи увімкнено API зберігання бази даних.

- setDomStorageEnabled(): Встановлює, чи увімкнено API зберігання DOM. Значення за замовчуванням — false.

- setUseWideViewPort(): Встановлює, чи повинен WebView увімкнути підтримку мета-тегу HTML "viewport" або чи повинен використовувати широкий вид порту.

- setLoadWithOverviewMode(): Встановлює, чи WebView завантажує сторінки в режимі огляду, тобто збільшує вміст, щоб він помістився на екрані за шириною [3].

```
final WebSettings webSettings = webview.getSettings();
webSettings.setJavaScriptEnabled(true);
webSettings.setBuiltInZoomControls(true);
webSettings.setDisplayZoomControls(false);
webSettings.setAppCacheEnabled(true);
webSettings.setDatabaseEnabled(true);
webSettings.setDomStorageEnabled(true);
webSettings.setUseWideViewPort(true);
webSettings.setLoadWithOverviewMode(true);
webview.getSettings().setAllowFileAccess(true);
webview.getSettings().setAllowContentAccess(true);
webview.setScrollbarFadingEnabled(false);
webSettings.setPluginState(WebSettings.PluginState.ON);
```

Рисунок 3.10 – Додавання JavaScript to WebView

Цей код (рис. 3.10) призначений для налаштування компонента WebView, який використовується для відображення веб-контенту в додатку. Код конфігурує об’єкт WebSettings (налаштування WebView) та сам WebView, щоб забезпечити підтримку певних функцій, таких як JavaScript, масштабування, кешування, доступ до файлів тощо.

3.5. Опис процесу тестування

Увесь прогрес розробки цього додатку було успішно здійснено за допомогою інструменту Android Linting. Lint — це інструмент сканування коду, наданий Android Studio, який може допомогти вам знайти та вирішити проблеми з структурною якістю коду.

Наприклад, Lint може допомогти в наступних проблемах:

- XML-файли ресурсів, що містять невикористані простори імен.

- Використання застарілих елементів.

- Виклики API, які не підтримуються цільовими версіями API, що може призвести до помилок у роботі коду [3].

Android Lint — це інструмент статичного аналізу, який входить до складу Android SDK і вбудований в Android Studio. Він аналізує код (Java, Kotlin), XML-файли (макети, маніфест, ресурси) і Gradle-скрипти, щоб виявити:

- Помилки програмування (наприклад, неправильне використання API).

- Проблеми з продуктивністю (наприклад, неефективне використання пам’яті).

- Проблеми безпеки (наприклад, незахищені дозволи).

- Порушення стилю коду (наприклад, невідповідність стандартам Google).

- Проблеми з доступністю (наприклад, відсутність описів для елементів UI).

- Застаріле використання API або бібліотек.

Основні характеристики Android Lint

- Lint перевіряє код без його виконання, аналізуючи структуру, синтаксис і семантику. Це дозволяє виявляти проблеми на ранніх етапах розробки.

- Lint автоматично запускається під час компіляції або вручну через меню Android Studio (Analyze > Inspect Code). Результати відображаються у вікні Inspection Results із детальними описами проблем і пропозиціями щодо їх виправлення.

- Lint має сотні вбудованих правил (наприклад, MissingTranslation, HardcodedText, UnusedResources). Розробники можуть увімкнути/вимкнути

певні перевірки або створювати власні правила через конфігураційний файл (lint.xml).

- Lint можна запускати через командний рядок (наприклад, ./gradlew lint) для інтеграції в системи безперервної інтеграції (CI), такі як Jenkins або GitHub Actions.

- Категорії перевірок:

- Правильність (Correctness): Виявлення логічних помилок, таких як неправильне використання API чи відсутність дозволів у маніфесті.

- Продуктивність (Performance): Виявлення неефективного коду, наприклад, витоків пам’яті чи надмірного використання ресурсів.

- Безпека (Security): Виявлення вразливостей, таких як незахищені мережеві запити чи жорстко закодовані паролі.

- Доступність (Accessibility): Перевірка UI на відповідність стандартам доступності (наприклад, наявність contentDescription для зображень).

- Стиль і форматування (Style): Перевірка відповідності стандартам кодування

Проект постійно перевірявся за допомогою команди ./gradlew lint, щоб перевірити, чи показує lint помилки. Усі проблеми lint зберігаються у файлі lint_baseline в Android Studio. Після запуску програми, якщо проблеми lint є відомими, то Studio їх ігнорує. Якщо проблеми lint невідомі, Studio викидає виняток. Інструмент lint пропонує шляхи до HTML і XML версій звіту lint після завершення перевірок. Потім HTML-звіт можна відкрити в браузері, перейшовши до нього.

3.6. Висновки до розділу

У третьому розділі було здійснено практичну реалізацію мобільного додатку для представлення міських ресурсів, зосереджену на побудові

інтерфейсу користувача та забезпеченні його функціональності. Зокрема, реалізовано макет додатку з використанням методу “ListView всередині CardView”, що забезпечує зручне та структуроване представлення інформаційних блоків.

Для ефективного відображення динамічних списків даних було застосовано компонент RecyclerView, який дозволяє гнучко управляти виведенням елементів та оптимізує продуктивність застосунку.

Реалізація інтерфейсу користувача охопила розробку логіки взаємодії з елементами управління, організацію навігації між екранами та забезпечення інтуїтивної доступності функцій додатку. Додатково використано компонент WebView, що дало змогу інтегрувати зовнішні веб-ресурси безпосередньо в середовище мобільного застосунку.

Завершальним етапом стала реалізація процесу тестування, в рамках якого перевірено коректність відображення даних, працездатність функціоналу та стабільність взаємодії з користувачем. Результати тестування підтвердили відповідність реалізації поставленим вимогам та забезпечили готовність додатку до подальшого розгортання.

ВИСНОВКИ

В дипломній роботі було повністю реалізовано цикл створення мобільного додатку, спрямованого на представлення міських ресурсів у зручному, візуально привабливому форматі. На початковому етапі проведено аналіз актуальності проблеми та вивчено існуючі рішення, що дозволило чітко окреслити функціональні та технічні вимоги до майбутнього застосунку.

На основі обґрунтованого вибору технологій розробки, до яких увійшли Android SDK, мова розмітки XML, база даних SQLite, було реалізовано архітектуру застосунку з урахуванням принципів ефективної організації даних і взаємодії між модулями. Значну увагу приділено інтерфейсу користувача — реалізовано сучасний візуальний стиль із використанням компонентів CardView, ListView та RecyclerView, що забезпечує комфортне сприйняття інформації та її структуроване подання.

Для інтеграції зовнішніх веб-ресурсів застосовано компонент WebView, що розширює можливості додатку без втрати цілісності дизайну. Реалізація функціональності базувалася на принципах адаптивності, інтуїтивності та швидкодії, а етап тестування підтвердив стабільну роботу системи за різних сценаріїв використання.

У підсумку, створено ефективний інструмент, здатний сприяти цифровій трансформації міського середовища, покращенню сервісів для жителів і туристів та забезпеченню доступності важливої інформації в зручному форматі. Робота демонструє можливості інтеграції сучасних програмних технологій у сфері міського планування, інформування населення та підвищення якості урбаністичного середовища.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Zhiqiang Wei, Dandan Zhang, and Yongquan Yang. "Android Developers." MVC Framework Based on Spring MVC and Mybatis, https://ieeexplore.ieee.org/abstract/document/6805007.
2. Weerasekara, Supun Ishara. Android Recycler View, 8 Feb. 2020, https://levelup.gitconnected.com/android-recycler-viewb1bc493cac53.
3. Бронюйте варіанти дозвілля, атракції та тури | GetYourGuide - https://www.getyourguide.com/uk-ua/?visitor-id=51X6QA28BWOIGRY2IKAI131XSOY6Y38C&locale_autoredirect_optout=true
4. Viator: Travel Tours, Activities, and Things to Do | 2025 - https://www.viator.com/
5. Враження Airbnb – Airbnb - https://www.airbnb.com.ua/s/experiences?redirectSource=slash_experiences&_set_bev_on_new_domain=1745002336_EAZDExNTU3MjY0MT
6. Eventbrite - Discover the Best Local Events & Things to Do - https://www.eventbrite.com/
7. Time Out | Best Things To Do and Events In Cities Worldwide - https://www.timeout.com/
8. BEST of San Francisco, CA Burgers – Yelp - https://www.yelp.com/search?find_desc=Burgers&find_loc=San+Francisco%2C+CA
9. RATP : transports à Paris et en Ile-de-France : bus, métro, tramway, RER | RATP - https://www.ratp.fr/
10. hvv - Verbindungen suchen - https://geofox.hvv.de/web/de/connections?clear=true&language=de
11. Herzlich Willkommen! - ÖPNV-Info - https://www.oepnv-info.de/
12. Improve your code with lint checks | Android Studio | Android Developers - https://developer.android.com/studio/write/lint

13. Developer Guide, Jan. 2025, https://developer.android.com/guide.
14. Krasner, G E, and Stephen Pope. A Cookbook for Using the Model - View Controller User Interface Paradigm in Smalltalk 80, Jan. 1998, https://www.researchgate.net/publication/248825145_A_cook book_for_ using_the_model_- view_controller_user_interface_paradigm_in_Smalltalk-_80.
15. Huina, Xie, and Stephen T Pope. Application of JAVA Programming Language in Computer Software Development, Electronic Technology and Software Engineering, 2017.
16. Basole, R. C., & Patel, P. C. (2018). Mobile applications and the sharing economy: A conceptual framework. Journal of Services Marketing, 32(6), 603-615.
17. SQLite Developers. SQLite Tutorials Point, https://www.tutorialspoint.com/ sqlite/index.htm.
18. DB Browser for SQLite. – https://sqlitebrowser.org/
19. Boulos, M. N. K., Resch, B., Crowley, D. N., Breslin, J. G., Sohn, G., Burtner, R., ... & P. (2011). Crowdsourcing, citizen sensing and mobile GIS for public health and environmental monitoring: Overview of the methods and their roles in data collection, analysis and interpretation. International Journal of Health Geographics, 10(1), 39.
20. Carmona, M. (2018). Public places–urban spaces: the dimensions of urban design. Routledge.
21. Choi, J., Lee, H., & Kim, J. (2019). The impact of mobile application quality on user satisfaction and loyalty. Information & Management, 56(6), 785-798.
22. Dey, A. K. (2001). Understanding and using context. Personal and ubiquitous computing, 5(1), 4-7.
23. Fogg, B. J. (2003). Persuasive technology: Using computers to change what we think and do. Morgan Kaufmann.

24. Goodhue, D. L., & Thompson, R. L. (1995). Task-technology fit and individual performance. MIS quarterly, 213-236.
25. Guttman, R. H., Moukas, A., & Maes, P. (1998). Agent-mediated electronic commerce. The Knowledge Engineering Review, 13(2), 131-149.
26. Höllerer, T., & Feiner, S. (2004). Mobile augmented reality. Telematic and informatics, 21(3-4), 309-329.
27. Kaplan, A. M., & Haenlein, M. (2010). Users of the world, unite! The challenges and opportunities of Social Media. Business horizons, 53(1), 59-68.
28. Lumsden, L., Skinner, G., Coyle, J., Coffey, D., & муніципалітет, Г. (2012). Mobile apps for health: what do they offer?. British journal of general practice, 62(605), e745-e748.
29. O'Brien, H. L., & Toms, E. G. (2008). What is user engagement? A conceptual framework for defining user engagement with technology. Journal of the American Society for Information Science and Technology, 59(6), 938-955.
30. Palen, L., & Salzman, C. (2002). Voice-mail graffiti and community questions: Exploring the uses of a community telephone network. Proceedings of the 2002 ACM conference on Computer supported cooperative work, 274-283.
31. Rettig, J. (1994). Prototyping for tiny fingers. Communications of the ACM, 37(4), 21-27.
32. Rogers, Y., Sharp, H., & Preece, J. (2011). Interaction design: beyond human-computer interaction. John Wiley & Sons.
33. Shneiderman, B., & Plaisant, C. (2010). Designing the user interface: strategies for effective human-computer interaction. Pearson Education.
34. A. Moukas, K. Chandrinos, P. Maes. Trafficopter: A Distributed Collection System Traffic Information. Proceedings of the Cooperative Information

Agents-98. In Lecture Notes in Artificial Intelligence. Kluwer Academic Publishers, 1998

35. Van Dijk, J. A. G. M. (2012). The network society (3rd ed.). Sage.
36. Zomerdijk, L. G., & Voss, C. A. (2010). Service design for experience-centric services. Journal of service research, 13(1), 67-82.
37. Naik, V.M. Deshmukh, and Harshad M Deshmukh. "A Redundant Structure Based Tokenization Model for XML Parsing on Android Mobile Devices", 24 Nov. 2016, https://ieeexplore.ieee.org/abstract/document/7755329.

ДОДАТКИ

Додаток А

Фрагменти розроблених класів

Лістинг А.1. Клас Express Adapter

```
package com.rushabh.nasikapp.ahmedabadexplorer.Express;
import android.content.Context;
import android.content.Intent;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import com.rushabh.nasikapp.ahmedabadexplorer.R;
import java.util.ArrayList;

class ExpressAdapter extends RecyclerView.Adapter<ExpressAdapter.MyViewHolder> {
    ArrayList<Express> express_arraylist;
    Context context;

    public ExpressAdapter(Context context, ArrayList<Express> express_arraylist) {
        this.express_arraylist = express_arraylist;
        this.context = context;
    }

    public class MyViewHolder extends RecyclerView.ViewHolder {
        public TextView heading, subheading;

        public MyViewHolder(View itemView) {
            super(itemView);
            heading = (TextView) itemView.findViewById(R.id.heading);
            subheading = (TextView) itemView.findViewById(R.id.subheading);
        }
    }

    @Override
    public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View itemView = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.list_of_express_main, parent, false);
        return new MyViewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(MyViewHolder holder, final int position) {
        Express express = express_arraylist.get(position);
        holder.heading.setText(express.getTitle());
        holder.subheading.setText(express.getDesc());

        holder.heading.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (((Express) ExpressAdapter.this.express_arraylist.get(position)).getType().e
                    Intent i = new Intent(context, Express_Station_List.class);
                    i.putExtra("destination", "Ahmedabad");
                    i.putExtra("type", "i");
                    context.startActivity(i);
                } else if (((Express) ExpressAdapter.this.express_arraylist.get(position)).getT
                    Intent i = new Intent(context, ListofExpress.class);
                    i.putExtra("source", "All");
                    i.putExtra("destination", "Trains");
                    i.putExtra("type", "a");
                    context.startActivity(i);
                } else if (((Express) ExpressAdapter.this.express_arraylist.get(position)).getT
                    Intent i = new Intent(context, Express_Station_List.class);
                    i.putExtra("source", "Ahmedabad");
                    i.putExtra("type", "o");
                    context.startActivity(i);
```

```
                }
            }
        });

        holder.subheading.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (((Express) ExpressAdapter.this.express_arraylist.get(position)).getType().e
                    Intent i = new Intent(context, Express_Station_List.class);
                    i.putExtra("destination", "Ahmedabad");
                    i.putExtra("type", "i");
                    context.startActivity(i);
                } else if (((Express) ExpressAdapter.this.express_arraylist.get(position)).getT
                    Intent i = new Intent(context, ListofExpress.class);
                    i.putExtra("source", "All");
                    i.putExtra("destination", "Trains");
                    i.putExtra("type", "a");
                    context.startActivity(i);
                } else if (((Express) ExpressAdapter.this.express_arraylist.get(position)).getT
                    Intent i = new Intent(context, Express_Station_List.class);
                    i.putExtra("source", "Ahmedabad");
                    i.putExtra("type", "o");
                    context.startActivity(i);
                }
            }
        });
    }

    @Override
    public int getItemCount() {
        return express_arraylist.size();
    }
```

Лістинг А.2. Клас SQLite List Adapter

```
package com.rushabh.nasikapp.ahmedabadexplorer.Express;
import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;
import com.rushabh.nasikapp.ahmedabadexplorer.BuildConfig;
import com.rushabh.nasikapp.ahmedabadexplorer.R;
import java.util.ArrayList;

class SQLiteListAdapter_express extends ArrayAdapter<Array_Express> {
    ArrayList<Array_Express> data = new ArrayList();
    int layoutResourceId;
    Context mContext;
    String name;

    public SQLiteListAdapter_express(Context context, int layoutResourceId, ArrayList<Array_Exp
        super(context, layoutResourceId, data);
        this.layoutResourceId = layoutResourceId;
        this.mContext = context;
        this.data = data;
    }
```

```
    @Override
    public long getItemId(int position) {
        return 0;
    }

    @Override
    public View getView(int position, View child, ViewGroup parent) {
        final Holder holder;
        LayoutInflater layoutInflater;
        if (child == null) {
            layoutInflater = (LayoutInflater) mContext.getSystemService(Context.LAYOUT_INFLATER_
            child = layoutInflater.inflate(R.layout.layout_of_list, null);
            holder = new Holder();
            holder.name = (TextView) child.findViewById(R.id.train_name);
            holder.number = (TextView) child.findViewById(R.id.train_no);
            holder.journey = (TextView) child.findViewById(R.id.journey);
            child.setTag(holder);
        } else {
            holder = (Holder) child.getTag();
        }

        final Array_Express user = (Array_Express) this.data.get(position);
        holder.name.setText(user.getname());
        holder.number.setText(user.getnumber());
        holder.journey.setText(BuildConfig.FLAVOR + user.getSource() + "-" + user.getDestinatio

        holder.name.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent full_schedule = new Intent(mContext, Express_schedule.class);
                full_schedule.putExtra("name", holder.name.getText().toString());
                full_schedule.putExtra("number", holder.number.getText().toString());
                full_schedule.putExtra("source", user.getSource());
        holder.number.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent full_schedule = new Intent(mContext, Express_schedule.class);
                full_schedule.putExtra("name", holder.name.getText().toString());
                full_schedule.putExtra("number", holder.number.getText().toString());
                full_schedule.putExtra("source", user.getSource());
                full_schedule.putExtra("destination", user.getDestination());
                mContext.startActivity(full_schedule);
            }
        });

        holder.journey.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent full_schedule = new Intent(mContext, Express_schedule.class);
                full_schedule.putExtra("name", holder.name.getText().toString());
                full_schedule.putExtra("number", holder.number.getText().toString());
                full_schedule.putExtra("source", user.getSource());
                full_schedule.putExtra("destination", user.getDestination());
                mContext.startActivity(full_schedule);
            }
        });

        return child;
    }

    public class Holder {
        TextView name;
        TextView number;
        TextView journey;
    }
}
```

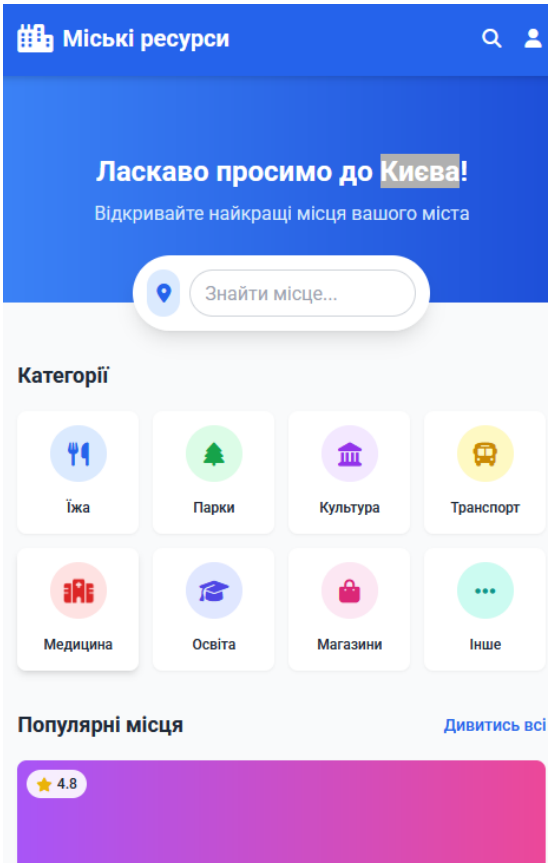

Рисунок А.1. – Інтерфейс додатку

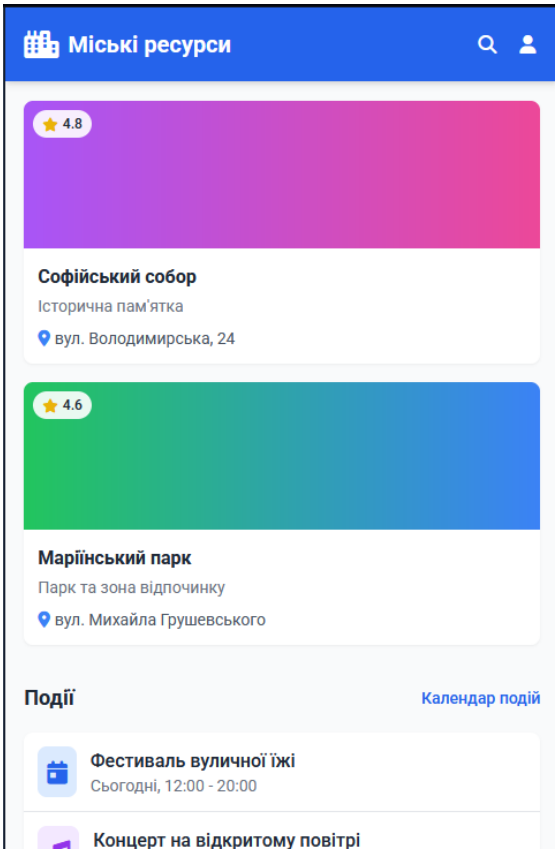

Рисунок А.2 – Визначні місця

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи: “Розробка мобільного додатку представлення ресурсів міста ”

Обсяг пояснювальної записки: __75__ аркушів.

Дата закінчення роботи: 10 червня 2025 р.

Підпис студента _____