МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА ПРОГРАМУВАННЯ ТА МАТЕМАТИКИ

Пояснювальна записка

до дипломної роботи

_____бакалавр _____

(освітньо-кваліфікаційний рівень)

на тему «Розробка мобільного додатку«Путівник по місту» »

Виконав: студент 4 курсу, групи ІПЗ-17д
напряму підготовки 121 „Інженерія програмного забезпечення”
_____ Мироненко М.В.
(підпис)

Керівник,
доцент, д.т.н._____ Лифар В.О.
(підпис)

Рецензент,
доцент каф. ПМ,
ст. викл. _____ Батурін О.І.
(підпис)

СЄВЄРОДОНЕЦЬК – 2021

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ
бакалаврської роботи студента гр. ІПЗ-17д Мироненка М.В.

Науковий керівник
доц, д.т.н._____Лифар В.О.
ПІБ, посада

Оцінка наукового керівника: _____

Рецензент Батурін О. І., ст.,викл., каф. ПМ СНУ ім. В. Даля
ПІБ, місто роботи, посада

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту: _____

Голова ЕК,
проф. кафедри ПМ
к.т.н., доцент_____Захожай О.І.
підпис

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет інформаційних технологій та електроніки

Кафедра програмування та математики

Освітньо-кваліфікаційний рівень бакалавр

Напрям підготовки 121 „Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ
Завідувач кафедри ПМ,
д.т.н., доцент
_____ Лифар В.О.
«___» _____ 2021 р.

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТА
Мироненка Миколи Володимировича

1. Тема роботи Розробка мобільного додатку «Путівник по місту».
керівник роботи д.т.н., доц. Лифар Володимир Олексійович

затверджені наказом вищого навчального закладу від «12» квітня 2021 року № 68/15.16

2. Строк подання студентом роботи 11 червня 2021 р.

3. Вихідні дані до роботи

Об'єктом досліджень технології на розробки мобільних

3.1 Літературні джерела

Дизайн Android [Електронний ресурс] - Режим доступу:

https://medium.com/swlh/material-design-an-in-depth-look-d4f4055a5ecf 15

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

4.1 Вступ

4.2 Аналіз предметної галузі (огляд літератури), з висвітленням наступних питань:

Поняття платформ розробки.

Аналіз існуючих платформ розробки.

Вимоги до платформ розробки.

4.3 Основна частина, в якої висвітлити:

Визначення мобільного додатку.

Архітектура мобільних додатків.

Вимоги до мобільного додатку.

Розробка мобільного додатку.

4.4 Висновки

4.4 Перелік використаних джерел

5. Перелік графічного матеріалу немає

6. Дата видачі завдання 12 квітня 2021 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Одержання завдання на виконання роботи	12.04.21	
2	Укладання і погодження з керівником плану і етапів виконання роботи	23.04.21	
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	26.04.21	
4	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху	03.05.21	
5	Проектування інфологічної моделі задачі що реалізується.	11.05.21	
6	Укладання програмного продукту	12.05.21	
7	Укладання, оформлення та погодження пояснювальної записки з керівником	25.05.21	
8	Здача готової пояснювальної записки на кафедру	11.06.21	
9	Укладання доповіді і презентації	14.06.21	

Студент _____ Мироненко М.В.
(підпис)

Керівник роботи _____ Лифар В.О.
(підпис)

РЕФЕРАТ

Робота містить: 58 сторінки основного тексту, 7 сторінок додатків, 34 рисунка, 17 використаних джерел.

Об'єктом дослідження є можливість розробки і створення мобільного додатку.

Метою дипломної роботи є розробка мобільного додатку «Путівник по місту».

Дипломна робота складається з двох основних частин.

У першій частині розглядаються сучасні технології, що використовуються при розробці мобільних додатків, та використовувані методи. Розглянуто теоретичні аспекти розробки мобільних додатків та їх використання в інформаційних цілях. Вивчаються основні типи мобільних додатків.

Також розглядаються основні етапи розробки мобільних додатків та особливості, які необхідно враховувати при розробці.

Друга частина показує модель проекту та структуру мобільного додатку.

У третій частині статті описується практична частина створення мобільного додатку, а саме XML-макета мобільного додатку, та його поєднання з відповідним Java-кодом.

Результатом цієї роботи є повноцінний мобільний додаток компанії «Путівник по місту», наповнений необхідним контентом.

ПЕРЕЛІК ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД – база даних.

ПО –програмне забезпечення.

XML – Extensible Markup Language, розширювана мова розмітки.

API – Application Programming Interface, інтерфейс прикладного програмування.

Java - строго типізований об'єктно-орієнтована мова програмування загального призначення, розроблений компанією Sun Microsystems.

SDK - software development kit набір засобів розробки, що дозволяє фахівцям з програмного забезпечення створювати додатки для певного пакету програм, програмного забезпечення базових засобів розробки, апаратної платформи, комп'ютерної системи, ігрових консолей, операційних систем і інших платформ.

OSM – OpenStreetMap некомерційний картографічний проект по створенню силами спільноти учасників - користувачів Інтернету докладної, вільної і безкоштовної географічної карти світу.

JDK - Java Development Kit, безкоштовно розповсюджуваний компанією Oracle Corporation комплект розробника додатків на мові Java, що включає в себе компілятор Java, стандартні бібліотеки класів Java, приклади, документацію, різні утиліти і виконавчу систему Java.

Зміст

Вступ 9

1. Аналітичний огляд 11

1.1 Поняття мобільного додатку 13

1.2 Класифікація мобільних додатків 13

1.3 Проектування додатків 17

1.4 Вибір способу розробки 18

1.5 Вибір програмного засобу для розробки 20

2. Модель та структура проекту 27

2.1 Організація проекту 27

2.1.1. Організація Activity 29

2.1.2 Організація Fragment 34

2.2 Структурна модель проекту 38

2.3 OpenStreetMap 41

2.4 Варіанти використання запропонованого додатку 43

3.Розробка додатку 44

3.1 Розробка структури додатку 44

3.1.1 Використання RecyclerView 46

3.1.2 Використання NavigationUI 50

3.2 Організація переходів між компонентами додатку 62

3.2.1 Взаємодія між Activity 62

3.2.2 Взаємодія між Fragment 65

Висновок: 67

ДОДАТКИ 70

Додаток А Частина лістингу програмного коду 70

Вступ

Актуальність.Мобільні пристрої не тільки лише міцно ввійшли в повсякденного життя кожної людини, вони певною мірою стали для нас додатковими частинами тіла. Завдяки їх використанню ми маємо змогу не лише спілкуємося між собою, але і замовляємо товари,отримувати інформацію,використовуємо для розваг тощо. Тож розробка програмного забезпечення є дуже необхідною та в майбутньому потреба в ній буде лише рости.

На відміну від минулих поколінь, сучасне поколінням користується туристичними послугами,які орієнтовані в першу чергу на мобільні пристрої, вони широко використовують мобільний телефон під час подорожі - не тільки для бронірування, а також для вивчення та пошуку рішень будь-яких питань.

Активний ріст ринку мобільних пристроїв сприяє зростанню іншого молодого ринку - мобільних додатків. У найближчих годах цей сегмент надає статтю одній з найбільш прибуткових та привабливих для інвесторів.

Вданий момент мобільні застосунки, це новітній засіб просування різноманітних проектів, внутрішнього туризму. У залежності від конкретних компаній та їх пріоритетів, мобільні додатки можуть бути ефективним маркетинговим інструментом для залучення клієнтів або сервісом для роботи з ними.

З вищевикладеного випливає, що мобільний додаток є необхідним атрибутом для сучасної туризму і визначає тему дипломної роботи - "Розробка мобільного додатку «Путівник по місту»".

Об’єкт дослідження: процес розробки мобільного додатку.

Предмет дослідження: мобільний туристичний додаток.

Мета дослідження: Створення повністю функціонуючого туристичного мобільного додатку.

Завдання дослідження:

- розглянути теоретичні основи використання мобільних додатків у туризмі;
- розглянити види мобільних додатків та їх аналоги;
- дослідити та проаналізувати платфрми розробки мобільних додатків.
- дослідити та проаналізувати процес створення мобільних додатків;
- зробити вибір мов програмування та технологій для програмної реалізації описаного продукту;
- розробити структуру додатку;
- розробити мобільний додаток.

Методи дослідження: Технології які використовуються для створення мобільних додатків.

1. Аналітичний огляд

Обгрунтування потреби в мобільному додатку

Туризм представляє себе торговими послугами.

По-перше, це комплексна послуга, з точки зору розробників, так і користувачів. По -друге, це невидима, змінна та інтегрована послуга. По-третє, інформаційно-насищенна послуга. Технічні характеристики туризму мають на увазі його як ідеальну відповідь для застосування інформаційних технологій

Структура туристичної галузі схожа на устрій будь-якої іншої економічної діяльності. Виробники туристичних послуг діють у повному обсязі визначеному керіваними структури, що працють у державних та комерційних організаціях, торгових асоціаціях (наприклад, готельних, транспортних, туристичних агентів тощо).

Виробники туристичних послуг класифікуються на повноцінних визначених категоріях постачальників: (авіакомпанії, готелі, орендатори автомобілів, «туроператори» та «турагенти». Користувачі (туристи) є останніми складовими всієї туристичної системи.

Тенденції розвитку туристичних та готельних послуг також, що зараз хорошій туристичній компанії вже недостатньо пропонувати інтернет-сайти, оптимізовані для мобільних пристроїв. Необхідно також створити спеціальні мобільні додатки.

Подорожі та туризм стали невід'ємною частиною життя величезного числа населення, адже при бажанні і наявності певної суми грошових коштів можна потрапити практично в будь-яку точку світу. Багато людей подорожують в пошуках нових емоцій, відчуттів, щоб побачити щось нове,

поспілкуватися з людьми. Число туристів величезне, і ця величезна кількість людей приваблює розробників мобільних додатків.

Функції мобільних додатків мало відрізняються від функцій сайтів, їх завдань та техніки використання,тобто мета додатка - представити інформацію про пропоновані компанією або декількома компаніями, товари та послуги, в межаї певного міста або країни. Головна відмінність мобільних додатків в тому, що вони надають набагато більше можливостей для отримання початкової інформації користувачів, які ними користуються. Мобільний телефон завжди знаходиться у користувача, і якщо він підключений до Інтернету, то його власнику завжди можна зацікавити конкретне комерційне пропозицію. І користувачелю не потрібно переходити на сайт, відкривати браузер, вводити адресу, замість цього досить просто натиснути на ярлик мобільного додатка. Несомненним достоїнством мобільного додатку є те, що воно може працювати без підключення до мережі[16].

На сьогоднішній день практично у кожного встановлено якийсь картографічний сервіс, який, по суті, теж може бути туристичним додатком. Але крім простих карт з функцією визначення місця розташування за допомогою датчиків GPS і ГЛОНАС, існують і більш просунуті рішення, які пропонують користувачеві не тільки маршрут до будь-якого пам'ятника, але і короткий опис, фотографії, рейтинги від інших користувачів і багато іншого.

Але подібні додатки можуть служити на благо не тільки мандрівникам, а й тим людям, які ніколи не виїжджали зі свого рідного міста.

Путівники по місту допоможуть дізнатися про цікаве в містах, про історію, про події, що відбуваються вмісті, про різні заклади. Також вони можуть допомогти з навігацією в місті, дізнатися номери автобусів і тролейбусів.

1.1 Поняття мобільного додатку

Мобільні програми - це програми, призначені для роботи на смартфонах, планшетах та інших мобільних пристроях. Ці програми зазвичай виконують певне завдання, наприклад, повідомляють прогноз погоди або відображають карти для навігації. Мобільні пристрої мають особливі вимоги через свої маленькі екрани та обмежені можливості введення. Крім того, сенсорний екран, як правило, є єдиним способом введення інформації в мобільний пристрій. Програмістам потрібні спеціальні знання, щоб зрозуміти мобільну платформу, для якої вони хочуть створювати програми[16].

Додатки можуть бути предустановлені виробником, або завантажені користувачем через різні платформи для поширення ПО або існувати в форматі веб-додатків

Канали поширення:

- спеціалізовані портали: AppStore, Android Market

-через sms с порталів стільникових операторів

-самостійний пошук і скачування в Інтернеті.

1.2 Класифікація мобільних додатків

Функціональність мобільних додатків є дуже різноманітною: від ігор та до офісних програм .Однак всі додатки можна поділити на 3 категорії: нативні, гібридні та веб:

- Власні програми створюються для однієї конкретної платформи або операційної системи.

- Веб-програми - це адаптивні версії веб-сайтів, які можуть працювати на будь-якому мобільному пристрої або ОС, оскільки вони доставляються за допомогою мобільного браузера.

- Гібридні додатки - це комбінації як власних, так і веб-програм, однак вони загорнуті в рідну програму, що дає їй можливість мати власну піктограму або завантажуватись із магазину програм.

Нативні

Власні програми створені спеціально для операційної системи мобільного пристрою (ОС). Таким чином, ви можете мати власні мобільні додатки для Android або власні програми для iOS, не кажучи вже про всі інші платформи та пристрої. Оскільки вони створені лише для однієї платформи, ви не можете поєднувати їх - скажімо, використовуйте додаток Blackberry на телефоні Android або використовуйте додаток iOS на телефоні Windows[16].

Використовувана технологія: Власні програми кодуються за допомогою різних мов програмування. Деякі приклади включають: Java, Kotlin, Python, Swift, Objective-C, C ++ та React.

Плюси: Завдяки своїй особливій спрямованості нативні програми мають перевагу в тому, що вони швидші та надійніші з точки зору продуктивності. Як правило, вони ефективніше використовують ресурси пристрою, ніж інші типи мобільних додатків. Власні програми використовують інтерфейс власного пристрою, надаючи користувачам більш оптимізований досвід роботи з клієнтами .

Оскільки власні програми безпосередньо підключаються до апаратного забезпечення пристрою, вони мають доступ до широкого вибору функцій пристрою, таких як Bluetooth, контакти телефонної книги, рулон камери, NFC тощо.

Мінуси: Однак проблема з власними програмами полягає в тому, що якщо ви почнете їх розробляти, вам доведеться дублювати зусилля для кожної з різних платформ. Код, який ви створюєте для однієї платформи, не можна використовувати повторно для іншої. Це збільшує витрати. Не кажучи вже про зусилля, необхідні для підтримання та оновлення кодової бази для кожної версії.

А потім, щоразу, коли відбувається оновлення програми, користувач повинен завантажувати новий файл та перевстановлювати його. Це також означає, що власні програми займають дорогоцінний простір у сховищі пристрою.

Веб-програми

Веб-програми поводяться подібно до власних програм, але доступ до них здійснюється через веб-браузер на вашому мобільному пристрої. Вони не є самостійними програмами в тому сенсі, що потрібно завантажувати та встановлювати код на пристрій. Це фактично чуйні веб-сайти, які адаптують свій інтерфейс користувача до пристрою, на якому користувач працює. Насправді, коли ви стикаєтесь із можливістю «встановити» веб-програму, вона часто просто додає URL-адресу веб-сайту на ваш пристрій.

Використовувана технологія: веб-програми розробляються з використанням HTML5, CSS, JavaScript, Ruby та подібних мов програмування, що використовуються для роботи в Інтернеті.

Плюси: Оскільки це веб-інтерфейс, немає необхідності налаштовувати платформу чи ОС. Це зменшує витрати на розробку.

Крім того, нема чого завантажувати. Вони не займуть місця в пам’яті вашого пристрою, як власний додаток, що полегшить технічне обслуговування - достатньо оновити оновлення через Інтернет. Користувачам не потрібно завантажувати оновлення з магазину програм.

Мінуси: Але це також доречно: веб-програми повністю залежать від браузера, що використовується на пристрої. Функції, доступні в одному браузері, недоступні в іншому, можливо, надаючи користувачам різний досвід.

І оскільки це оболонки для веб-сайтів, вони не будуть повністю працювати в автономному режимі. Навіть якщо у них режим офлайн, пристрій все одно потребуватиме з'єднання з Інтернетом, щоб створити резервну копію даних на вашому пристрої, запропонувати нові дані або оновити те, що на екрані.

Гібридні програми

А ще є гібридні програми. Це веб-програми, які виглядають і відчувають себе як рідні програми. Вони можуть мати піктограму головного екрану, адаптивний дизайн, швидку продуктивність, навіть функціонувати в автономному режимі, але це справді веб-програми, які виглядають рідними.

Використовувана технологія: гібридні програми використовують суміш веб-технологій та власних API. Вони розроблені з використанням: Ionic, Objective C, Swift, HTML5 та інших.

Плюси: Створення гібридного додатка набагато швидше та економічніше, ніж власне додаток. Таким чином, гібридний додаток може бути мінімально життєздатним продуктом - способом довести життєздатність побудови власного додатка. Вони також швидко завантажуються, ідеально підходять для використання в країнах з більш повільним з'єднанням з Інтернетом і надають користувачам стабільний досвід роботи. Нарешті, оскільки вони використовують єдину базу коду, коду підтримувати набагато менше.

Мінуси: гібридним програмам може не вистачати потужності та швидкості, що є характерними рисами рідних програм.

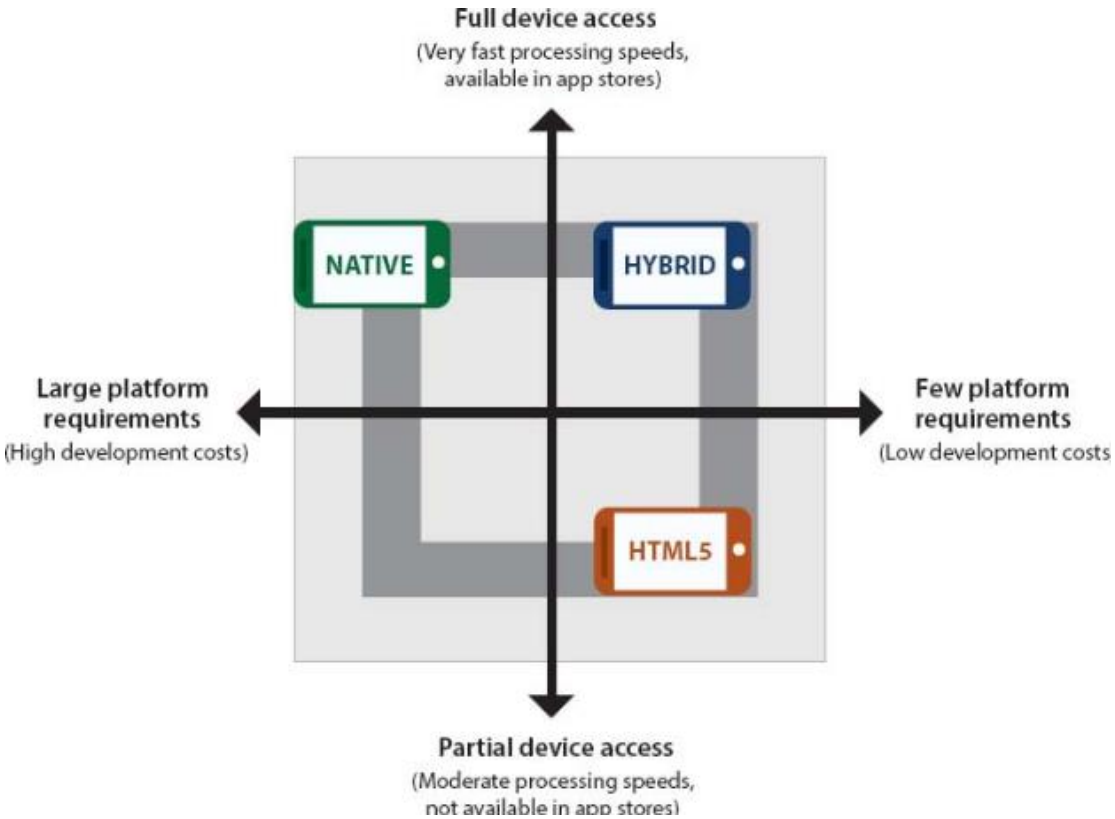

Рисунок 1.1 - Загальна схема видів додатків

1.3 Проектування додатків

Проектування і розробка мобільних додатків включає:

-Затвердження початкового технічного завдання на розробку додатку.

-Визначення структурної схеми додатку-розташування розділів, контента і навігації.

-Мобільний дизайн -створення дизайну, елементів навігації.

-Розробка коду програми, її модулів, баз даних та інших елементів необхідних в проекті.

1.4 Вибір способу розробки

Мобільний сайт

Працює тільки через браузер. На відміну від звичайних веб-сайтів, він розроблений спеціально для мобільних пристроїв.

Поряд з мобільними, існують сайти з адаптивним дизайном. Адаптивний сайт містить HTML- сторінки, пов'язані разом, які проглядаються в браузерах через інтернет. Адаптивна (гумова) верстка призначена для правильного відображення на всіх розмірах екранів[10].

Переваги мобільного / адаптивного сайту:

Сумісність. Зручний під час роботи з різними типами смартфонів / планшетів. Не потребує розробки окремої версії з урахуванням різних операційних систем. Може підтримувати легку інтеграцію з такими функціями, як QR-коди, текстові повідомлення.

Більш широке охоплення цільової аудиторії. Завдяки підтримці декількох пристроїв, яку забезпечує адаптивний веб-дизайн на різних платформах, задіяна більш широка аудиторія користувачів.

Підтримка, обслуговування. У порівнянні з нативними додатками, які вимагають завантаження кожного оновлення, адаптивні веб-сайти дозволяють гнучко вносити зміни. Внесені зміни стають активними і видимими відразу на всіх типах пристроїв[16].

Обмеження:

Зручність. На відміну від програми, адаптивний веб-сайт не може ефективно використовувати всі функції смартфона. Камери, GPS, телефонний набір і інші функції, інтегровані в пристрої, не завжди добре розроблені для адаптивних веб-сайтів.

Розмір екрану пристрою. Через невеликі розміри екрани смартфонів, планшетів відображають набагато менше контенту в порівнянні з монітором настільного ПК або екраном ноутбука. Незважаючи на те, що адаптивний веб-дизайн динамічно підлаштовується під потрібний розмір екрану, він фактично зменшує і перебудовує контент, доступний на робочому столі.

Автономний доступ. Автономний режим функціонування мобільного сайту можливий тільки при використанні кешованих сторінок. Повноцінна робота вимагає гарне підключення до інтернету.

Мобільний додаток

На відміну від адаптивних / мобільних веб-сайтів, що працюють через браузери, нативні додатки повинні бути завантажені з певних порталів, таких як Google Play Market, App Store або інших.

Мобільні програми створюється окремо для кожної операційної системи, вимагають установки, забезпечують більш швидкий доступ до вмісту.

Плюси:

Зручність. Аналіз показує, що додатки більш популярні, ніж аналогічні веб-сайти, оскільки більш зручні. Вони забезпечують кращу взаємодію з користувачем, швидше завантажують контент, простіше у використанні. Крім цього, мають push-повідомлення та дизайн, який більш гнучко сумісний з різними розмірами екрану.

Персоналізація. Мобільні програми є відмінним рішенням для служб, які вимагають регулярного використання. Вони дозволяють користувачам створювати особисті облікові записи, а також зберігати важливу інформацію під рукою.

Робота в автономному режимі. Оскільки додатки вимагають установки, вони можуть надавати доступ до своїх функцій і контенту навіть без підключення до Інтернету.

Мінуси:

Сумісність. Забезпечення належного функціонування нативного додатки залежить від вимог конкретної операційної системи. Це означає, що для кожної платформи (iOS, Android) потрібна окрема робоча версія програми.

Підтримка. Коли додаток створюють для кількох платформ, його підтримка вимагає більше часу та затрат. Ротрібно регулярно робити оновлення, виправляти проблеми.

Тож незважаючи на плюси адаптивного веб-дизайну, мобільні додатки є більш популярними і саме тому буде розроблено мобільний додаток , який в майбутньому можна буде вдосконалити в залежності до потреб користувачів.

1.5 Вибір програмного засобу для розробки

Перше, що потрібно зробити для вирішення дилеми «додатки для Android або додатка для iOS», - це визначити цільовий ринок застосування додатка.

Android - це операційна система, яка в використовується для моібльних, сенсорних пристроїв. Її конструкція дозволяє користувачам інтуїтивно викоритовувати мобільні пристрояї за допомогою рухів пальців, такі як проведення та натискання. Також використовується як програмне забезпечення Android для телевізорів, автомобілів та наручних годинників - кожен із яких має унікальний користувальницький інтерфейс[12].

Платформа Android об'єднує операційну систему, побудовану на основі ядра ОС Linux, проміжне програмне забезпечення і вбудовані мобільні додатки.

Розробка і розвиток мобільної платформи Android виконується в рамках проекту AOSP (Android OpenSource Project) під керуванням OHA (Open Handset Alliance), керує всім процесом пошуковий гігант Google. Android підтримує фонове виконання завдань; надає багату бібліотеку елементів призначеного для користувача інтерфейсу; підтримує 2D і 3D графіку, використовуючи OpenGL стандарт; підтримує доступ до файлової системи і вбудованій базі даних SQLite.

З точки зору архітектури, система Android представляє собою повний програмний стек, в якому можна виділити такі рівні:

• Базовий рівень (Linux Kernel) - рівень абстракції між апаратним рівнем і програмним стеком;

• Набір бібліотек і середовище виконання (Libraries & AndroidRuntime) забезпечує найважливіший базовий функціонал для додатків, містить віртуальну машину Dalvik і базові бібліо-теки Java необхідні для запуску Android додатків;

•Рівень каркаса додатків (Application Framework) забезпечує розробникам доступ до API, наданих компонентами системи рівня бібліотек;

• Рівень додатків (Applications) - набір встановлено-них базових додатків.

Розглянемо компоненти платформи більш докладно. У основі компонентної ієрархії лежить ядро ОС Linux 2.6 (кілька урізане), воно служить проміжним рівнем між апаратним та програмним забезпеченням, забезпечує функціонування системи, надає системні служби ядра: управління пам'яттю, енергосистемою і процесами, забезпечення безпеки, роботи з мережею і

драйверами. Рівнем вище розташовується набір бібліотек і середовище виконання. Бібліотеки реалізують такі функції:

• надають реалізовані алгоритми для вищого рівня;

• забезпечує підтримку файлових форматів;

• здійснює кодування і декодування інформації (наприклад, мультимедійні кодеки);

• виконує отрисовку графіки і т.д.

Бібліотекі реалізовані на C / C ++ і скомпільовані під конкретне апаратне забезпечення пристрою, разом з яким вони і поставляються виробником в передбаченому вигляді. .

iOS (раніше iPhone OS) - це мобільна операційна система, створена та розроблена Apple Inc. виключно для свого обладнання. Саме операційна система керує багатьма мобільними пристроями компанії, включаючи iPhone та iPod Touch; цей термін також включав версії, що працюють на iPad, доки назва iPadOS не була введена з версією 13 у 2019 році. Це друга за частотою встановлення мобільна операційна система у світі після Android. Це основа для трьох інших операційних систем Apple: iPadOS, tvOS і watchOS. Це запатентоване програмне забезпечення, хоча деякі його частини є відкритими за ліцензією Apple Public Source License та іншими ліцензіями [12].

Розробка Android-проектів

Для створення додатків для Android розробники використовують мови програмування Java, C++, Kotlin, C# Dart, Python. Крім того, розробники Android використовують такі передові інструменти розробки Google:

Android Jetpack, набір попередньо зібраних компонентів Android

Firebase відома як комплексна платформа для розробки мобільних додатків.

Комплект розробника Android SDK, пов'язаний з Android Studio, інтегрованим середовищем розробки

Розробка iOS-проектів

У той час як операційна система Android має відкритий вихідний код, iOS має закритий вихідний код. Закритий вихідний код означає, що iOS працює тільки на пристроях Apple. Команда розробників може використовувати Swift або Objective-C. Також інструменти розробки для iOS включають в себе: [12].

iOS SDK або Software Development Kit інтегрований з платформою для користувача інтерфейсу Cocoa Touch. Платформа надає графічні елементи, елементи управління призначеним для користувача інтерфейсом і інші елементи.

XCode - офіційна інтегроване середовище розробки (IDE) для розробки під iOS.

Swift Playgrounds - це середовище розробки для Swift.

TestFlight - це онлайн-сервіс для бездротової установки і тестування. Цей онлайн-сервіс дозволяє розробникам тестувати програми і збирати цінні відгуки перед випуском програми[12].

Переваги розробки під Android

Відкрита система. Розробники Android отримують доступ до більшої кількості функцій, обмежених у додатках iOS.

Фрагментація. З одного боку, можно розглядати фрагментацію як недолік, але ви можете розробляти програми не тільки для смартфона Android,

але і для більш широкого спектру пристроїв, включаючи носяться пристрої, телевізори, автомобільні системи та інші.

Реліз. У порівнянні з iOS, додатки для Android легше опублікувати в Google Play. Весь процес може зайняти всього декілька годин.

Недоліки Android

Фрагментація. Як вже говорилося, фрагментація також може бути недоліком Android. В даний час пристрої Android мають різні розміри екрану, дозволу і т.д. Команді розробників може знадобитися більше часу, щоб налаштувати функції програми для певних пристроїв.

Тестування. Версії Android і пристрої можуть відрізнятися. Таким чином, QA-фахівцям може знадобитися більше часу для тестування вашої програми.

Ваптість росробки. Чим більше часу займе етап розробки і тестування, тим вищою буде ціна Android-додатки. Але ціна також залежить від можливостей і складності програми.

Переваги iOS

Дохід. Як ви, можливо, вже знаєте, користувачі Apple витрачають більше грошей на покупки додатків в порівнянні з користувачами Android.

Кількість пристроїв. Як ми вже говорили, iOS підтримує тільки продукти Apple. З цієї причини ваш додаток має відповідати обмеженій кількості екранів і пристроїв.

Дизайн користувальницького інтерфейсу. Apple надає розробникам докладний посібник з стилю користувальницького інтерфейсу додатку. Команді потрібно менше часу, що робить її більш доступною на етапі розробки програми.

Недоліки iOS

Випуск програми. Майте на увазі, що в App Store діють досить суворі правила перевірки. Торговий майданчик може відхилити ваше додаток через проблеми з безпекою, відсутність цінного контенту або низьку продуктивність. Крім того, розробник повинен відправити додаток на реальне тестування, яке зазвичай займає кілька днів.

Гнучкість. Додатки для iOS зазвичай складно налаштувати, тому що платформа має безліч обмежень.

Демографія

Демографічні дані багато говорять про користувачів Android і iOS.

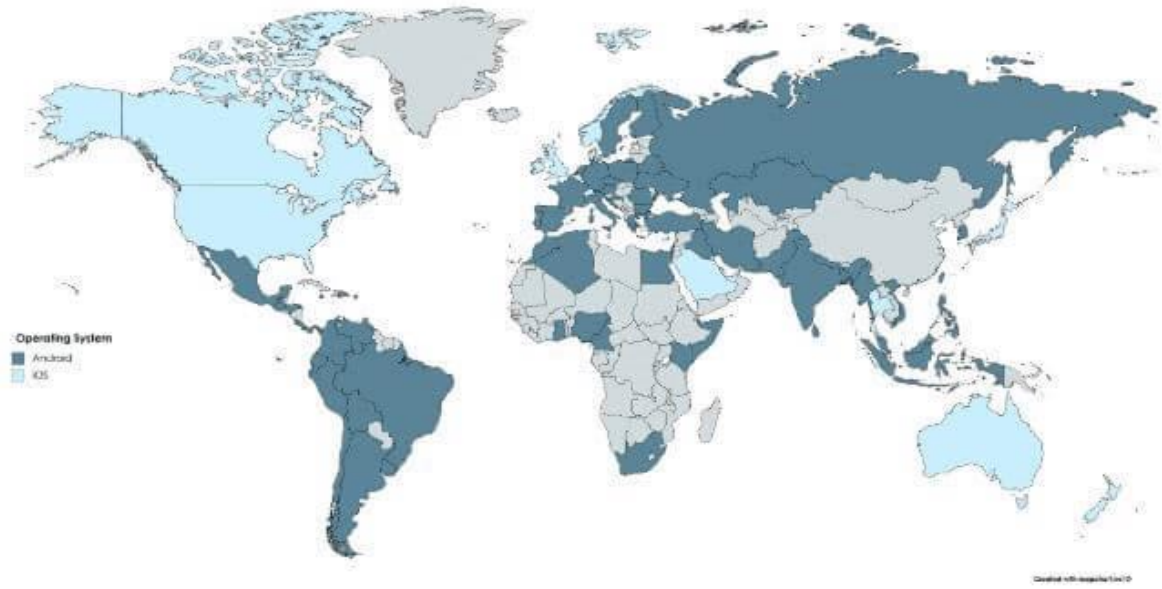

Рисунок.1.2 - Карта використання мабільних платформ в світі

Android в даний момент займає найбільш значну частку в світі серед платформ. Ця ринкова частка припадає на країни, що розвиваються і регіони з низькими доходами.

Вартість розробки

При виборі між розробкою для iOS і Android враховуйте, що орієнтовні ціни на додатки для розробки для Android можуть бути вище. Як згадувалося раніше, ОС Android працює на багатьох пристроях з екранами різного розміру. Це збільшує час розробки, а також витрати.

З іншого боку, iOS має обмежену кількість пристроїв, що також прискорює процес розробки. Однак витрати на розробку для iOS і Android залежать від кількості функцій і складності програми.

Тестування

У iOS і Android є свої симулятори для тестування додатків. Однак між ними є деякі відмінності. TestFlight, середа тестування iOS, працює швидше, ніж емулятор Android. Але віртуальна машина емулятора Android більш ефективна і має більш реалістичне уявлення.

Оновлення додатків

Обидві операційні системи випускають нові оновлення ОС один раз на рік. Таким чином, візьміть до уваги, що оновлення додатків iOS для нової версії ОС може зайняти до двох тижнів і вплинути на план вашого продукту і бізнес-стратегію. Що стосується Android, то публікація з'явилася нова версія програми займе кілька годин.

Моделі доходів

Ще один фактор, який необхідно враховувати при виборі платформи, - це те, де знаходиться велика частина вашої аудиторії. Android має більш високий відсоток додатків, підтримуваних рекламою, в той час як платформа розробки iOS покладається в основному на покупки.

2. Модель та структура проекту

Врахувавши наявні дані було вибрано платформою розробки Android. В якості середовища розробки обрано Android studio тому що наразі воно є офіційним інтегрованим середовищем розробки (IDE) додатків для Android, в основі якого лежить середовище IntelliJ IDEA. Варто зазаначити що розробка під Android неможлива без використання Android SDK. В основному, Android-додаток складається з: Java-класів, тобто підкласів Android SDK і Java-класів, без батьківських елементів в Android SDK; файлів програми; маніфесту додатку; ресурсів.

2.1 Організація проекту

Для розробки додатку будуть використані основні компоненти Android такі як: активності (Activity), фрагменти (Fragment) та xml layout-и, які відповідають за дизайн проекту.

Архітектура додатків основана на двох типах файлів: активностях в форматі java і макетах екрану з xml разміткою. Layout-файли містять в собі всі елементи дизайну, які можуть бути побачені користувачем. Вони подані у вигляді xml-коду, завдяки цьому при правильному налаштуванні вони одинакоо відобразяться на різних пристроях. Кожному макету екранів відповідає свій окремий файл з кодом званий Activity.

Весь програмний код додатку міститься в файлах Activity або Java класах. Кожен клас відповідає лише за ті елементи, які розташовані на його layout файлі [10]. Винятком є лише додаткові класи де описують процеси які користувач не може побачити , наприклад, створення БД. Кожна з активностей

оголошується в файлі маніфесту. Він визначає глобальні значення для вашого пакета, в ньому ви описуєте, що знаходиться всередині вашого застосування - діяльності, сервіси і тд. Ви так само визначаєте, як всі ці елементи взаємодіють з Андроїд [9].

Крім цього додаток містить у собі і різні ресурси: картинки, текст, аудіо, навігацію, меню, що зберігаються в папці res. Насправді класів набагато більше, але це основні. Виділені жовтим - ті, з якими розробник працює, теж важливі важливі, але безпосередньо вони рідше вастосовуються [9].

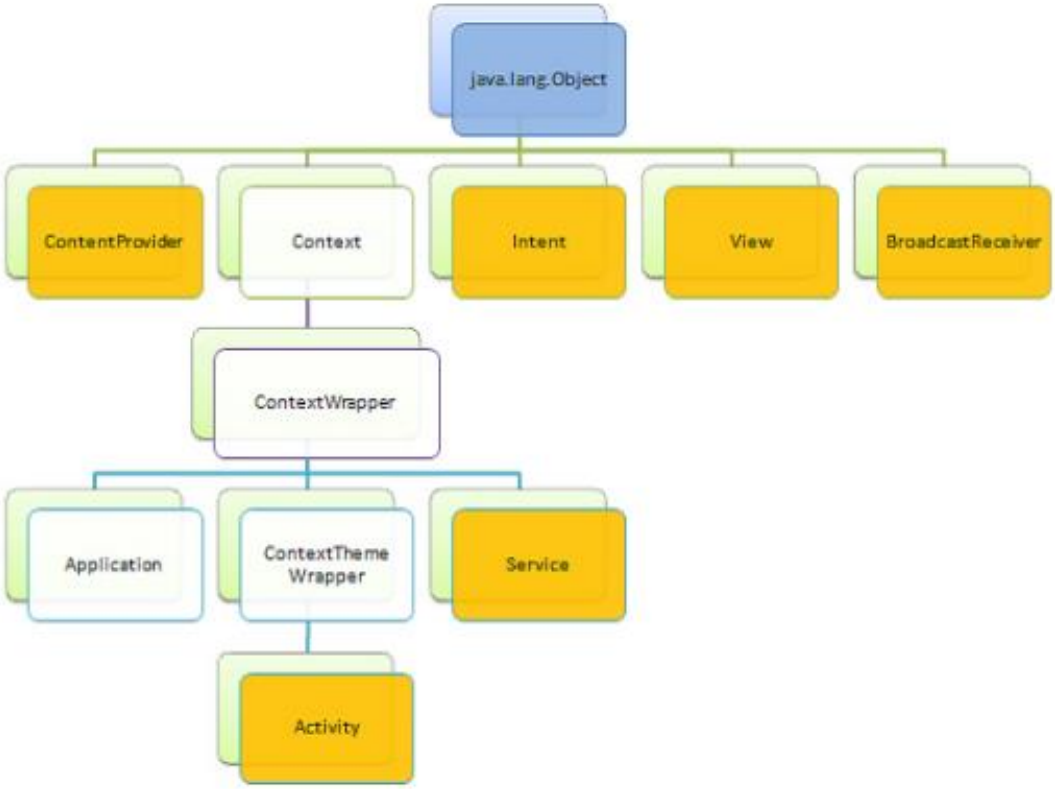

Рисунок 2.1 - Загальна структура Android проектів

View – це клас, що представляє основу для компонентів користувацького інтерфейсу. Він займає область на екрані і відповідає за відрисовку та обробку того що відбувається на екрані. View - це базовий клас для віджетів, які застосовують для в якості компонентів користувацького інтерфейсу. Підклас ViewGroup - це базовий клас, що застосовується для відображення макетів, які

є невидимими контейнерами, що містять інші подання (або ViewGroups) і визначають їх властивості макета [1].

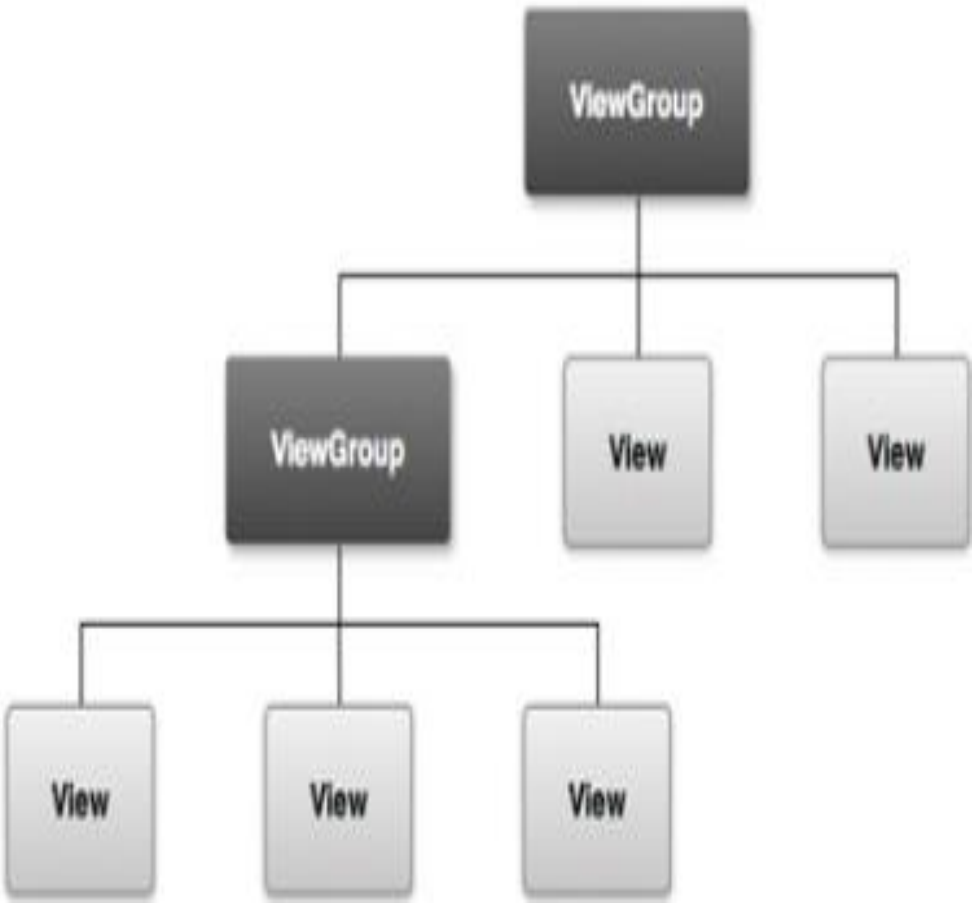

Рисунок 2.2 - Загальній ієрархії компонентів View

2.1.1. Організація Activity

Activity - це окремий екран в Android. Це як вікно в додатку для робочого столу, або фрейм в програмі на Java [1].Activity дозволяє вам розмістити всі ваші компоненти користувальницького інтерфейсу або віджети на цьому екрані [11].

Життєвий цикл Activity

Всі Activity Android мають життєвий цикл. При запуску користувачем додатка система дає цьому додатку високий пріоритет. Кожна програма запускається у вигляді окремого процесу, що дозволяє системі давати одним процесам вищий пріоритет, на відміну від інших.

Завдяки цьому, наприклад, при роботі з одними додатками Android дозволяє не блокувати вхідні дзвінки. Після припинення роботи з додатком, система звільняє все пов'язані ресурси і переводить додаток у розряд низькопріоритетного і закриває його.

Всі об'єкти activity, які є знаходяться в додатку, керуються системою, що має вигляд стека, який називається back stack.

При старті нової activity вона поміщається поверх стека і виводиться на екран, доки не з'явиться нова activity. Коли поточна activity закінчує свою роботу, то вона видаляється з стека, і відновлює роботу та activity, яка раніше була другою в стеці [11].

Методи життєвого цикла:

- onCreate ()

onCreate - перший метод, з якого починається виконання activity. У цьому методі activity переходить в стан Created. Цей метод обов'язково повинен бути визначений в класі activity. У ньому проводиться первісна настройка activity. Зокрема, створюються об'єкти візуального інтерфейсу. Цей метод отримує об'єкт Bundle, який містить попереднього стану activity, якщо воно було збережено. Якщо activity заново створюється, то даний об'єкт матиме значення null. Якщо ж activity раніше була створена, але перебувала в загальмованому стані, то bundle містить пов'язану з activity інформацію.

Після того, як метод onCreate () завершив виконання, activity переходить в стан Started, і та система викликає метод onStart()

- onStart

У методі onStart () відбувається підготовка до висновку activity на екран пристрою. Як правило, цей метод не вимагає перевизначення, а всю роботу робить вбудований код. Після завершення роботи методу activity відображається на екрані, викликається метод onResume , а activity переходить в стан Resumed.

- onResume

При виклику методу onResume activity переходить в стан Resumed і відображається на екрані пристрою, і користувач може з нею взаємодіяти. І власне activity залишається в цьому стані, поки вона не втратить фокус, наприклад, внаслідок перемикання на іншу activity або просто через виключення екрану пристрою.

- onPause

Якщо користувач вирішить перейти до іншої activity, то система викликає метод onPause , а activity переходить в стан Paused. У цьому методі можна звільняти використовувані ресурси, припиняти процеси, наприклад, відтворення аудіо, анімацій, зупиняти роботу камери (якщо вона використовується) і т.д., щоб вони менше позначалися на продуктивність системи.

Але треба враховувати, що в цей стан activity як і раніше залишається видимої на екрані, і на роботу даного методу відводиться дуже мало часу, тому не варто тут зберігати якісь дані, особливо якщо при цьому потрібно звернення до мережі, наприклад, відправка даних по інтернету, або звернення до бази даних - подібні дії краще виконувати в методі onStop().

Після виконання цього методу activity стає невидимою, не відображається на екрані, але вона все ще активна. І якщо користувач

вирішить повернутися до цієї activity, то система викличе знову метод onResume, і activity знову з'явиться на екрані.

Інший варіант роботи може виникнути, якщо раптом система бачить, що для роботи активних додатків необхідно більше пам'яті. І система може сама завершити повністю роботу activity, яка невидима і знаходиться в тлі. Або користувач може натиснути на кнопку Back (Назад). В цьому випадку у activity викликається метод onStop .

- onStop

У цьому методі activity переходить в стан Stopped. У цьому стані activity повністю невидима.

У методі onStop слід визволили використовувані ресурси, які не потрібні користувачеві, коли він не взаємодіє з activity. Тут також можна зберігати дані, наприклад, в базу даних.

При цьому під час стану Stopped activity залишається в пам'яті пристрою, зберігається стан всіх елементів інтерфейсу. Наприклад, якщо в текстове поле EditText був введений якийсь текст, то після відновлення роботи activity і переходу її в стан Resumed ми знову побачимо в текстовому полі раніше введений текст.

Якщо після виклику методу onStop користувач вирішить повернутися до колишньої activity, тоді система викличе метод onRestart . Якщо ж activity зовсім завершила свою роботу, наприклад, через закриття програми, то викликається метод onDestroy () .

- onDestroy

Ну і завершується робота activity викликом методу onDestroy , який виникає або, якщо система вирішить вбити activity в силу конфігураційних

причин (наприклад, поворот екрану або при многоконном режимі), або при виклику методу finish () .

Також слід зазначити, що при зміні орієнтації екрану система завершує activity і потім створює її заново, викликаючи метод onCreate .

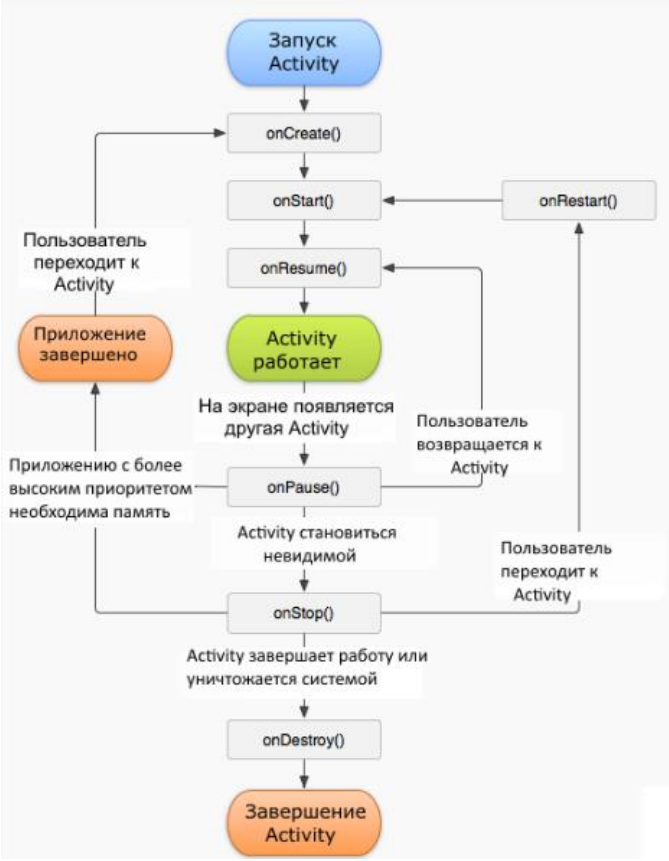

Рисунок 2.3 - Життєвий цикл активності

В цілому перехід між станами activity можна виразити наступною схемою.

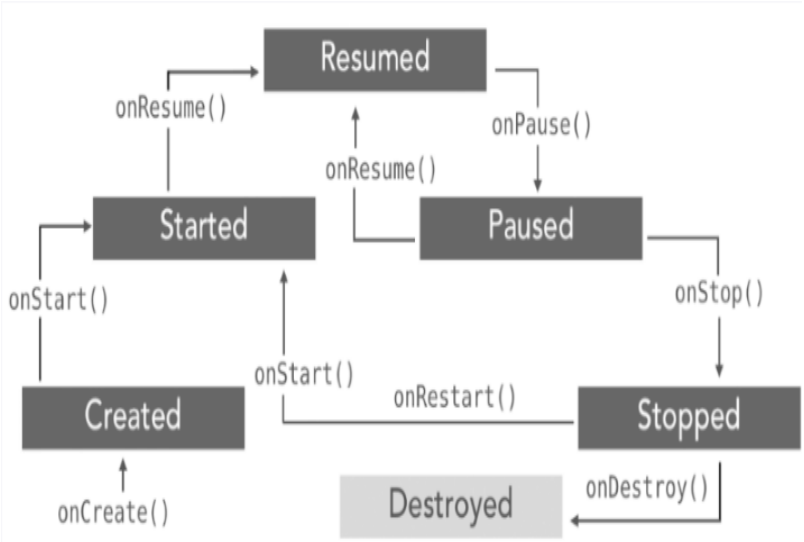

Рисунок 2.4 - Схема переходів між станами activity

2.1.2 Організація Fragment

Fragment представляє шматочок візуального інтерфейсу програми, який може використовуватися повторно і багаторазово. У фрагмента може бути власний файл layout, у фрагментів є свій власний життєвий цикл. Фрагмент існує в контексті activity і має свій життєвий цикл, поза activity осібно він існувати не може. Кожна activity може мати кілька фрагментів.

Фактично фрагмент - це звичайний клас Java, який успадковується від класу Fragment. Однак як і клас Activity, фрагмент може використовувати xml-файли layout для визначення графічного інтерфейсу. І таким чином, ми можемо додати окремо клас Java, який представляє фрагмент, і файл xml для зберігання в ньому розмітки інтерфейсу, який буде використовувати фрагмент [13].

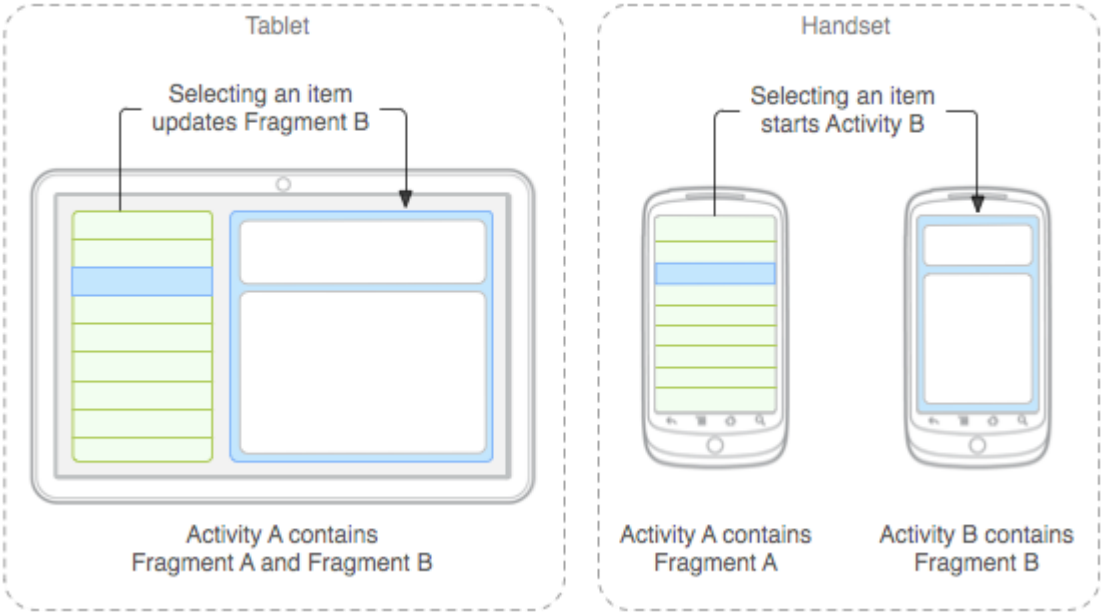

Рисунок.2.5 - Використання фрагментів

Життєвий цикл Fragment:

- onCreate ():

onCreate (): створюється фрагмент. У цьому методі ми можемо отримати раніше збережений стан фрагмента через параметр методу Bundle savedInstanceState. (Якщо фрагмент створюється перший раз, то цей об'єкт має значення null) Цей метод буде викликано після виклику методу onCreate () у activity.

- onCreateView ():

onCreateView (): фрагмент створює уявлення (View або візуальний інтерфейс). У цьому методі ми можемо встановити, який саме ізуальний інтерфейс буде використовувати фрагмент. При виконанні цього методу уявлення фрагмента переходить в стан INITIALIZED. А сам фрагмент все ще перебуває в стані CREATED

- onViewCreated ():

onViewCreated (): викликається після створення уявлення фрагмента.

- onViewStateRestored ():

onViewStateRestored (): отримує стан уявлення фрагмента. Після виконання цього методу уявлення фрагмента переходить в стан CREATED

- onStart ():

onStart (): викликається, коли фрагмент стає видимим і разом з поданням переходить в стан STARTED

- onResume ():

onResume (): викликається, коли фрагмент стає активним, і користувач може з ним взаємодіяти. При цьому фрагмент і його уявлення переходять в стан RESUMED

- onPause ():

onPause (): фрагмент продовжує залишатися видимим, але вже не активний і разом з поданням переходить в стан STARTED onStop (): фрагмент більше не є видимим і разом з поданням переходить в стан CREATED

- onDestroyView ():

onDestroyView (): знищується уявлення фрагмента. Подання переходить в стан DESTROYED

- onDestroy ():

onDestroy (): остаточно знищення фрагмента - він також переходить в стан DESTROYED

- onDetach () :

Метод onDetach () викликається, коли фрагмент видаляється з FragmentManager і відкріплюється від класу Activity. Цей метод викликається після всіх інших методів життєвого циклу

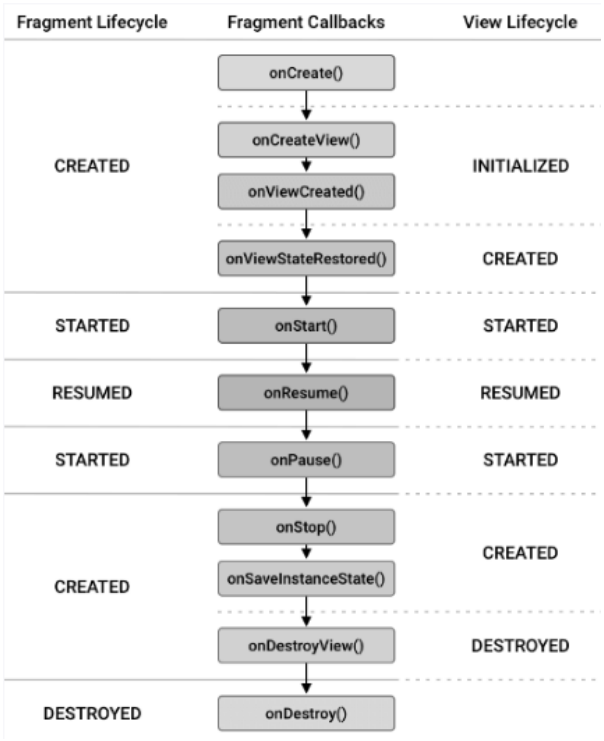

Рисунок.2.6 Життєвий цикл Fragment

Одна activity може використовувати кілька фрагментів, наприклад, з одного боку список, а з іншого - детальний опис вибраного елементу списку. У такій конфігурації activity використовує два фрагмента, які між собою повинні взаємодіяти[13].

Model-View-ViewModel (MVVM) - це структурний шаблон дизайну, який розділяє об'єкти на три окремі групи:

- Model містять дані програми. Зазвичай це конструкції або прості класи[10].
- View відображають візуальні елементи та елементи керування на екрані. Вони, як правило, підкласи UIView.
- ViewModel перетворюють інформацію про модель у значення, які можуть відображатися у поданні. Зазвичай це класи, тому їх можна передавати як посилання.

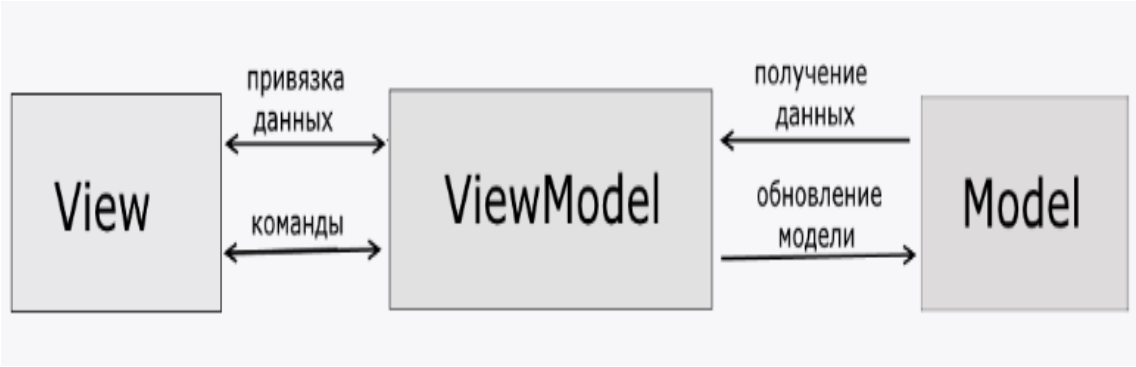

Рисунок 2.7 - Model-View-ViewModel

2.2 Структурна модель проекту

В основі додатку лежатимуть активності, які міститимуть в собі фрагменти і будуть відповідати за певний фукціонал програми.

В програмі буде чотири активності, і майже козна з них нестиме в собі певний набір фрагментів необхідних для виконня різноманітних завдань :

- SplashActivity - відповідатиме за відображення екрану завантаження програми.
- MainActivity - відповідатиме за відображення основної інформації додатку.
- DetailActivity- відповідатиме за відображення детальної інформації, щодо певного елементу MainActivity.
- MapActivity - відповідатиме за відображення і роботу з картами.

При запуску програми відбудеться поява екрану завантаження після чтого користувач отримає доступ до головної сторінки додатку та його основної інформації і в залежності від свої планів зможе отримати більш детальну інформацію про необхідне йому місце тощо.

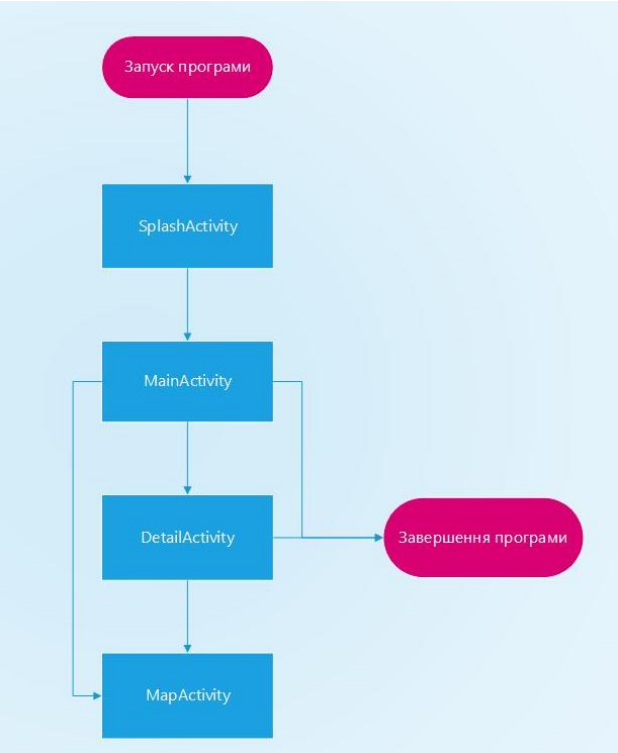

Рисунок.2.8 - Структурна схема додатку

MainActivity міститиме в собі розбиті на категорії списки з інформацією, при натисканні елементів яких відбудеться перенаправленя(за допомогою елементів навігації) в DetailActivity, також надаватиме доступ до карт, та додаткової інформації типу номерів екстрених служб або таксі, за допомогою навігаційних елементів.

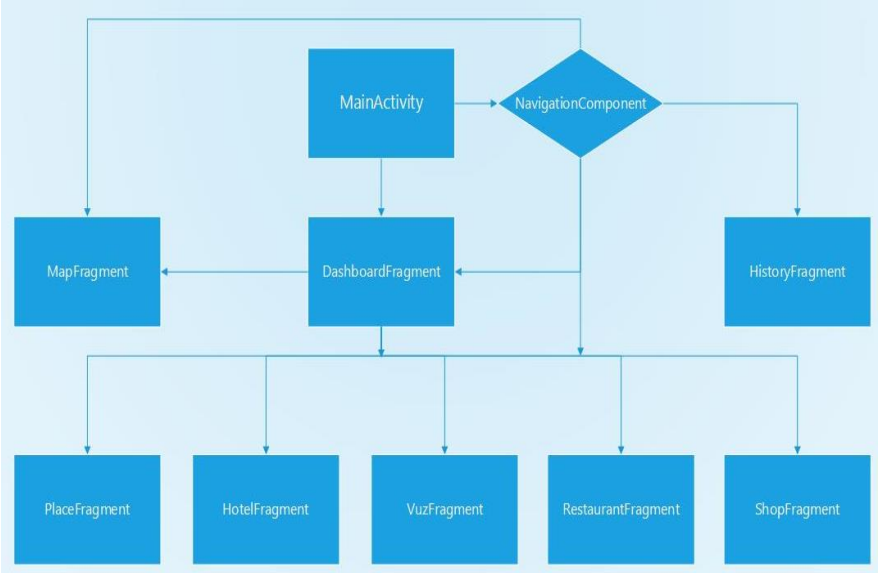

Рисунок.2.9 Схема роботи MainActivity

DetailActivity міститиме в собі детальну інформацію про пеаний елемент списку з MainActivity, надаватиме його опис, основні відомості, за можливосты відкриватиме сайт в браузері, зможе здійснити виклик за номером зазначеним номер, зазначеним в цьому елементі, а зможе також відкрити карту з відображенням місцезнаходження необхідного елементу на карті у вигляді маркера, який при натисканні виведе назву елементу та його знаходження тобно назву вулиці, будинку тощо. Навігацію між елесентами DetailActivity та їх зв'язок з MainActivity та MapActivity будуть забезпечути компоненти навігацію системи андроід.

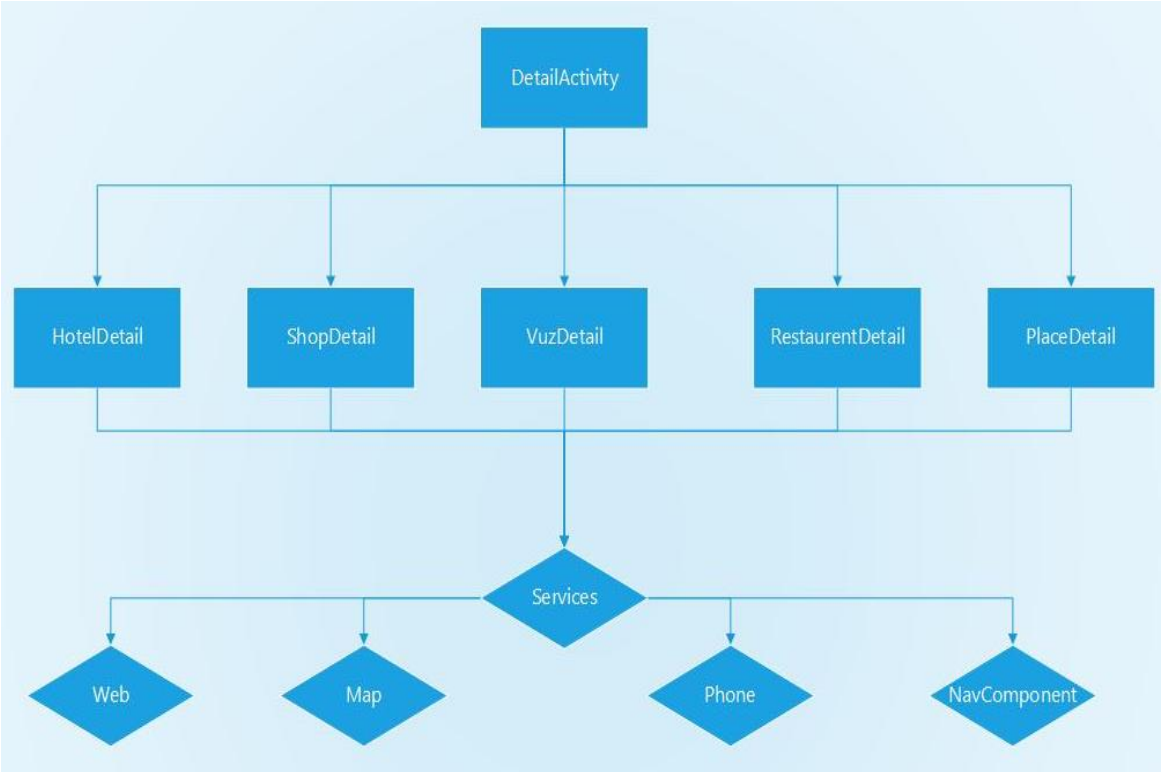

Рис.2.10 - Схема роботи DetailActivity

MapActivity - буде розділена на декілька категорій, в вигляді таблиці з елементами навігації в горі, відповідних до категорій наявних в MainActivity, а також відображатими маркери відповідні до елементів наявних в списках , крім цьогу саме тут на одній з вкладок таблиці відображатиметься повна інформація про елементи доступні в програмі в вигляді маркерів, з інформаційними вікнами, які виступатимуть для показу додаткової інформації, щодо місцезнаходжень елементів. Для реалізації карти та відображення маркерів буде використано андроід бібліотеку osmdroid, яка використовує для роботи з картами картографічний сервіс OpenStreetMap.

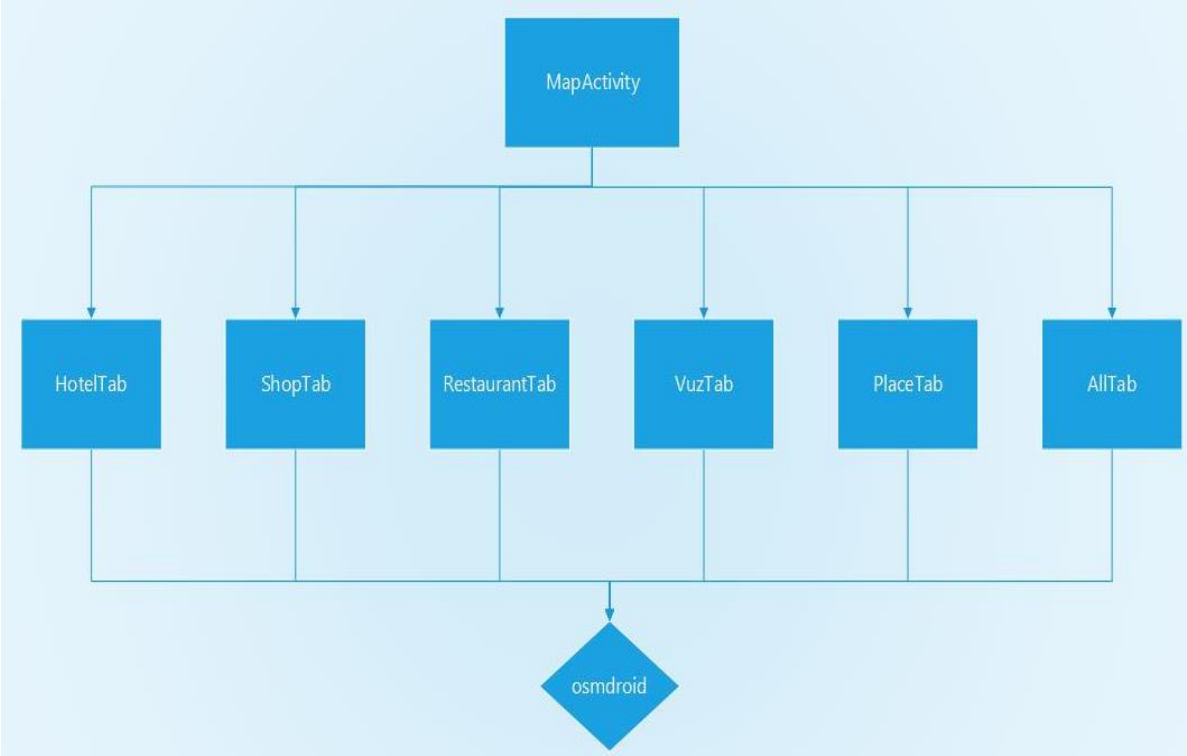

Рис.2.11 - Схема роботи MapActivity

2.3 OpenStreetMap

В якості картографічного сервісу було обрано OpenStreetMap - проект з побудови безкоштовної географічної бази даних світу.Його мета - записати всі географічні об’єкти на планеті. Хоча це почалося з картографування вулиць, воно вже вийшло далеко за рамки, включаючи пішохідні доріжки, будівлі, водні шляхи, трубопроводи, ліси, пляжі, поштові скриньки та навіть окремі дерева. Поряд з фізичною географією, проект також включає адміністративні межі, деталі використання земель, автобусні маршрути та інші абстрактні ідеї, які не видно з самого ландшафту [15].

База даних використовує вікіподібну систему, де будь-який картограф може додавати або редагувати будь-яку функцію в будь-якій області, а повна історія редагування зберігається для кожного об’єкта. Це означає, що будь-які помилки або навмисний вандалізм можуть бути відкореновані, зберігаючи дані точними. OpenStreetMap не використовує існуючийгеоінформаційна система (ГІС) для зберігання своїх даних, але замість цього використовує власне програмне забезпечення та модель даних, щоб максимально спростити процес краудсорсингу та забезпечити максимальний рівень гнучкості у тому, що і як відображається.

Дані OpenStreetMap можуть безкоштовно використовувати будь-хто для будь-яких цілей. Вони випускаються за ліцензією, яка дозволяє копіювати, змінювати та розповсюджувати дані. В Інтернеті доступно безліч карт, усі з яких можна використовувати безкоштовно, а деякі можна вбудувати у ваші власні веб-сторінки або використовувати в змішаннях, але вони також мають обмеження щодо того, що ви можете робити з цими службами та даними вони забезпечують.

На відміну від більшості джерел географічних даних, практично немає обмежень щодо того, що можна робити з даними OpenStreetMap. Ви можете використовувати його для будь-яких цілей, включаючи комерційну діяльність, без сплати збору за ліцензію. Ви можете будь-яким способом редагувати

інформацію та публікувати результати. Ви можете передавати дані комусь іншому, не потребуючи дозволу, а вони, в свою чергу, можуть їх передавати. Твоє єдине зобов'язання - «ділитися однаково»; тобто дозволити кожному, кому ви надаєте дані OpenStreetMap, перерозподілити їх самостійно [15].

2.4 Варіанти використання запропонованого додатку

Користувачами поданої програми є туристи, гості міста, студенти які вирішили вступати в університети даного міста. В залежності від їх потреб вони зможуть отримати необхідну інформацію, дізнатись нове, знайти щось необхідне їм. Отримати інформацію місцезнаходження готелів, ресторанів, магазинів, розважальних закладів та освітніх закладів, а також їх опис.

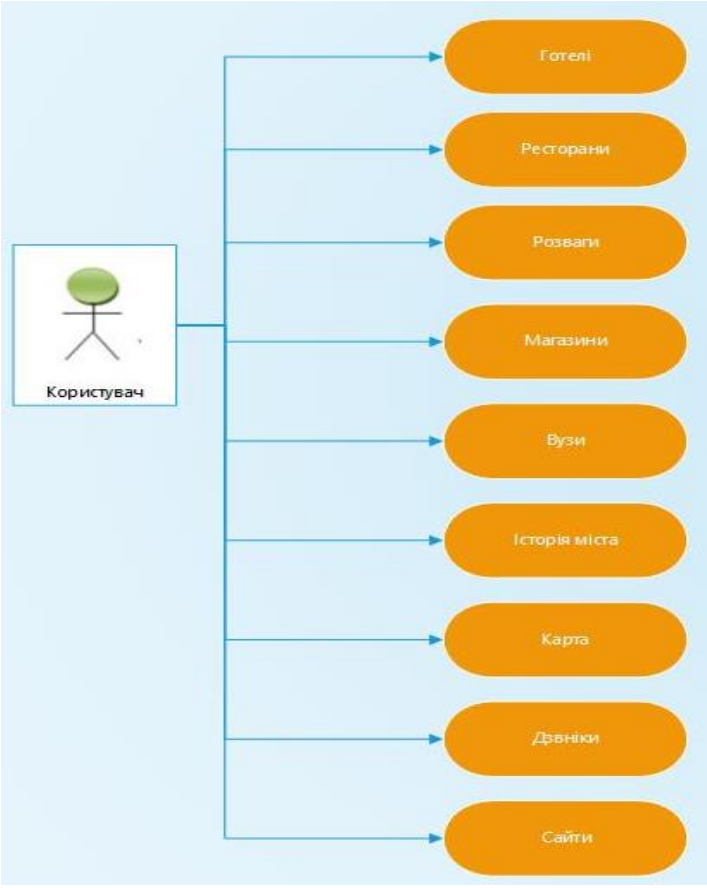

Рисунок 2.12 - Схема використання додатку користувачем

3.Розробка додатку

3.1 Розробка структури додатку

На початковому етапі розробки додатку, визначається його структура, те, як він буде працювати в подальшому. На початку роботи користувача з даним додатком, користувач потрапить на головну активність(MainActivity) де отримає змогу перейти в цікаву йому категорію.

В якості головної сторінки використовується GridLayout з елементами CardView в середені. GridLayout представляє собою,ще один контейнер, який створює табличні уявлення. GridLayout створює колекції рядків, кожна з яких складається з окремих центрів.

За допомогою атрибутів android: rowCount і android: columnCount встановлюється число рядків і стовпців відповідно. При цьому ширина стовпців встановлюється автоматично по ширині самого широкого елемента [2].

Однак ми можемо явно задати номер стовпця і рядка для певного елемента, а при необхідності розтягнути на декілька стовпців або рядків. Для цього ми можемо застосовувати такі атрибути:

- android: layout_column : номер стовпця (відлік йде від нуля)
- android: layout_row : номер рядка
- android: layout_columnSpan : кількість стовпців, на які розтягується елемент

- android: layout_rowSpan : кількість рядків, на які розтягується елемент

CardView - це віджет в Android, який можна використовувати для відображення будь-якого типу даних, надаючи макет із закругленим кутом разом із певною висотою. Основне використання CardView полягає в тому, що він допомагає надати насичене відчуття та вигляд дизайну інтерфейсу [6].

```
<androidx.cardview.widget.CardView
    android:layout_marginTop="10dp"
    android:id="@+id/cardView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:elevation="8dp"
    app:cardCornerRadius="8dp"
    app:cardPreventCornerOverlap="true"
    app:cardUseCompatPadding="true">
</androidx.cardview.widget.CardView>
```

Рисунок 3.1 Задання CardView

Цей віджет можна легко побачити в багатьох різних програмах для Android. CardView може використовуватися для створення елементів у списку або всередині подання Recycler.

Найкраща частина CardView полягає в тому, що він розширює Framelayout і може відображатися на всіх платформах Android.

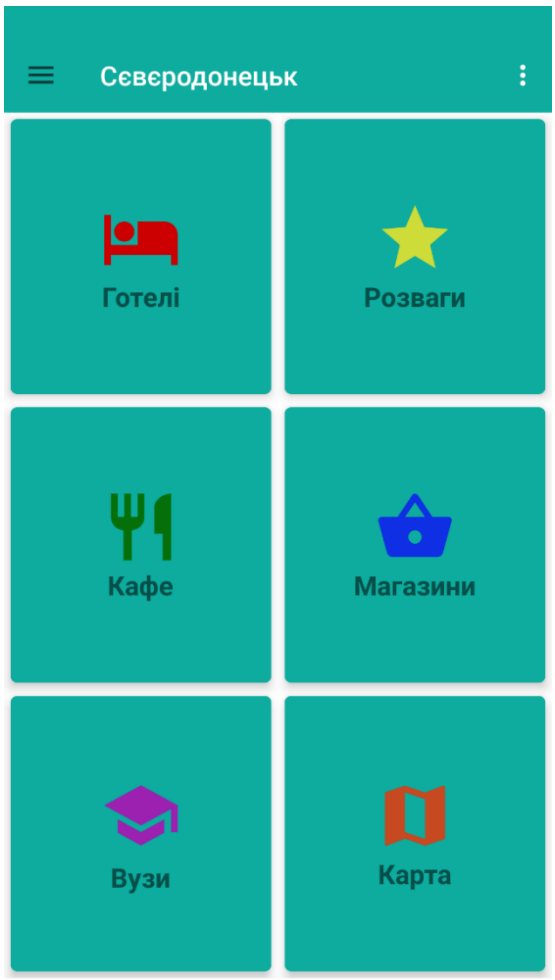

Рисунок 3.2-Використання CardView в GridLayout

3.1.1 Використання RecyclerView

Варто зазначити, що категорії в даному додатку – це фрагменти які містять в собі RecyclerView, а для відображення інформації в них використовується CardView.

Клас RecyclerView підтримує показ колекції даних.

Це модернізована версія ListView та GridView класів, передбачених фреймворком Android. Перегляд Recycler вирішує кілька проблем, які виникають у існуючих віджетів. Він запровадив стиль програмування, що

призводить до хорошої продуктивності. Він також постачається із анімацією за замовчуванням для видалення та додавання елементів [5].

RecyclerView дозволяє використовувати різні менеджери компонування для позиціонування елементів.

```
@Override
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    RecyclerView recyclerView = view.findViewById(R.id.recyclerView);
    recyclerView.setHasFixedSize(true);
    recyclerView.setNestedScrollingEnabled(false);
    recyclerView.setItemViewCacheSize(10);

    if (getContext() != null) {
        recyclerView.setLayoutManager(new LinearLayoutManager(getContext(),
                LinearLayoutManager.VERTICAL, reverseLayout: false));
        recyclerView.setAdapter(new HotelAdapter(HotelData.fetchHotelData(getContext()),
                getContext()));
    }
}
```

Рисунок 3.2 Підключення Recyclerview

Recyclerview використовує ViewHolder для зберігання посилань на подання для одного запису.

Клас ViewHolder - це статичний внутрішній клас у адаптері, який містить посилання на відповідні подання. За допомогою цих посилань код може уникнути трудомісткого методу findViewById () для оновлення віджетів новими даними.

Адаптер керує моделлю даних і адаптує його до окремих записів в віджеті. Він розширює RecyclerView.Adapter клас і призначається RecyclerView за допомогою RecyclerView.setAdapter методу.

Вхідними даними для адаптера можуть бути будь-які довільні об'єкти Java. На основі цього входу адаптер повинен повернути загальну кількість елементів за допомогою свого getItemCount() методу.

Адаптер готує макет елементів, роздуваючи правильний макет для окремих елементів даних. Ця робота виконана в onCreateViewHolder методі. Він повертає об’єкт типу ViewHolder за кожний візуальний запис.

Цей екземпляр використовується для доступу до видів у завищеному макеті. Метод onCreateViewHolder викликається тільки тоді коли новий вид повинен бути створений.

```
@NonNull
@Override
public HotelViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int viewType) {
    return new HotelViewHolder(LayoutInflater
            .from(context)
            .inflate(R.layout.fragment_hotel, viewGroup,  attachToRoot: false));
}

@Override
public void onBindViewHolder(@NonNull final HotelAdapter.HotelViewHolder hotelViewHolder, int position) {

    final Hotel hotel = hotels.get(position);

    hotelViewHolder.hotelImage.setImageResource(hotel.getHotelImageId());
    hotelViewHolder.hotelTitle.setText(hotel.getHotelTitle());
    hotelViewHolder.hotelRating.setText(String.valueOf(hotel.getHotelRating()));
    hotelViewHolder.hotelRatingBar.setRating(hotel.getHotelRating());
    hotelViewHolder.hotelType.setText(hotel.getHotelType());

    hotelViewHolder.cardView.setOnClickListener(view -> Utils.detailIntent(context, hotel, hotelViewHolder.hotelImage));
}
```

Рисунок 3.3 Використання адаптеру

Кожен видимий запис у RecyclerView адаптером заповнюється правильним елементом моделі даних. Як тільки елемент даних стає видимим, адаптер призначає ці дані окремим віджетам, які він надував раніше. Ця робота виконується в onBindViewHolder [5].

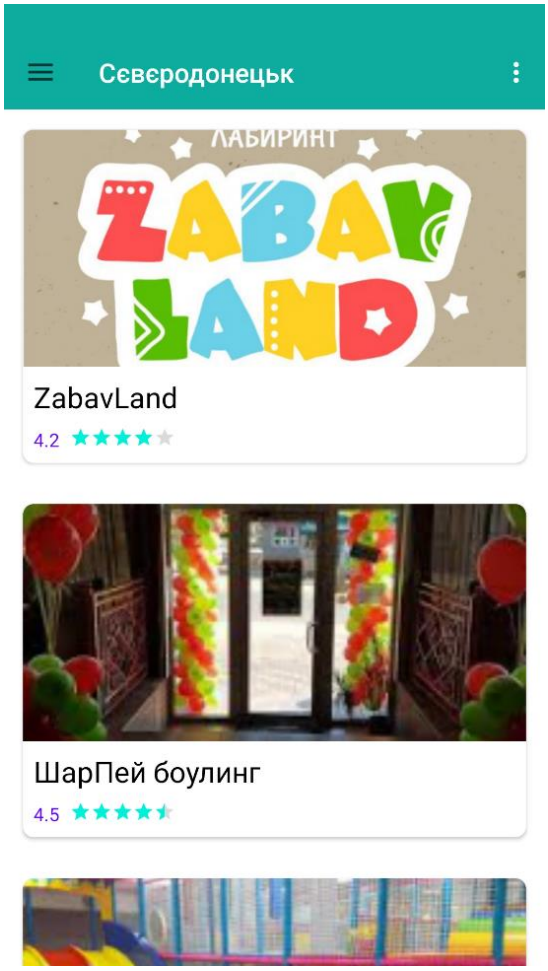

Рисунок 3.4 - Використання Recycleviw

Категорія “Готелі” надає інформацію про перелік готелів міста,в якості списка елементів(RecyclerView) та дає змогу користувачу перейти в розділ з детальною інформацією щодо кожного елемента категорії.

Категорія “Розваги” надає інформацію про перелік цікавих місць міста,в якості списка елементів,та дає змогу користувачу перейти в розділ з детальною інформацією щодо кожного елемента цієї категорії.

Категорія “Кафе” надає інформацію про перелік ресторанів/кафе міста,в якості списка елементів,та дає змогу користувачу перейти в розділ з детальною інформацією щодо кожного елемента цієї категорії.

Категорія “Магазини” надає інформацію про перелік ресторанів/кафе міста,в якості списка елементів,та дає змогу користувачу перейти в розділ з детальною інформацією щодо кожного елемента цієї категорії.

Категорія “Вузи” надає інформацію про перелік вищих навчальних закладів міста,в якості списка елементів,та дає змогу користувачу перейти в розділ з детальною інформацією щодо кожного елемента цієї категорії.

Категорія “Історія” надає інформацію стосовно історії міста, його заснування, історичних подій тощо.

Категорія “Карта” відображає повну карту міста,а також має поділ на вкладки відповідно до категорій додатку з відображенням відповідних їм елементів.Для створення вкладок використовується TabLayout. Він представлений в бібліотеці підтримки проектування для реалізації вкладок.

Вкладки створюються методом newTab () класу TabLayout. Заголовок та піктограма вкладок встановлюються за допомогою методів setText (int) та setIcon (int) інтерфейсу TabListener відповідно. Вкладки макета прикріплюються до TabLayout за допомогою методу addTab (Tab).

Варто зазначити, що для навігації між категоріями використовується NavigationUI/Navigation drawer.

3.1.2 Використання NavigationUI

NavigationUI – це, клас який містить статичні методи, які керують навігацією за допомогою верхньої панелі програми, панелі навігації та нижньої навігації [1].

NavigationUI містить методи, які автоматично оновлюють вміст на верхній панелі додатків, коли користувачі переходять через категорії в додатку.

Наприклад, NavigationUI використовує мітки призначення з створеного навігаційного графіку, щоб підтримувати актуальну назву верхнього рядка програми [3].

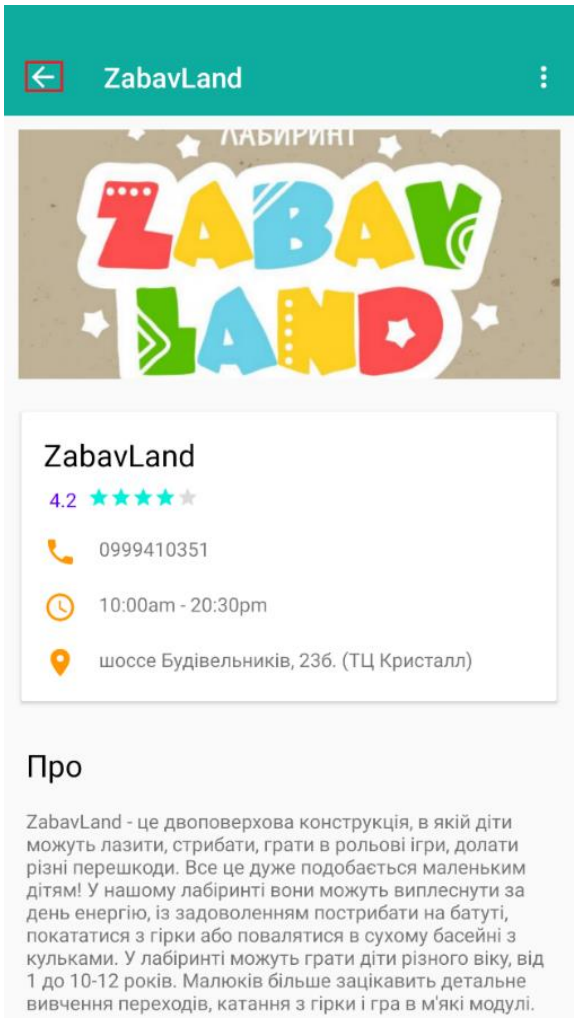

Рисунок 3.5 Мітки призначення NavigationUI

При використанні NavigationUI з реалізаціями верхньої панелі додатків, про яке йдеться нижче, мітка, яка прикріплюється до пунктів призначення,

може бути автоматично заповнена з аргументів, наданих цільовому пункту, використовуючи формат {argName} мітки.

NavigationUI використовує AppBarConfiguration об’єкт для управління поведінкою кнопки навігації у верхньому лівому куті області відображення додатка. Поведінка кнопки навігації змінюється залежно від того, перебуває користувач у пункті призначення верхнього рівня [3].

Пункт призначення верхнього рівня - це корінь або пункт призначення найвищого рівня у наборі ієрархічно пов’язаних пунктів призначення.

Пункти призначення верхнього рівня не відображають кнопку назад (стрілка вліво) на верхній панелі програми, оскільки немає пункту призначення вищого рівня.

За замовчуванням початковий пункт призначення вашої програми є єдиним пунктом призначення верхнього рівня.

Для цього необхідно додати лишень декілька строки коду:

```
actionBar.setDisplayHomeAsUpEnabled(true);

 actionBar.setDisplayShowHomeEnabled(true);
```

Коли користувач перебуває у пункті призначення верхнього рівня, кнопка навігації стає видимою і має вигляд трьох паралельних горизонтальних рисок, якщо в додатку використовується DrawerLayout.

Якщо DrawerLayout не використовується, кнопка навігації приховується.

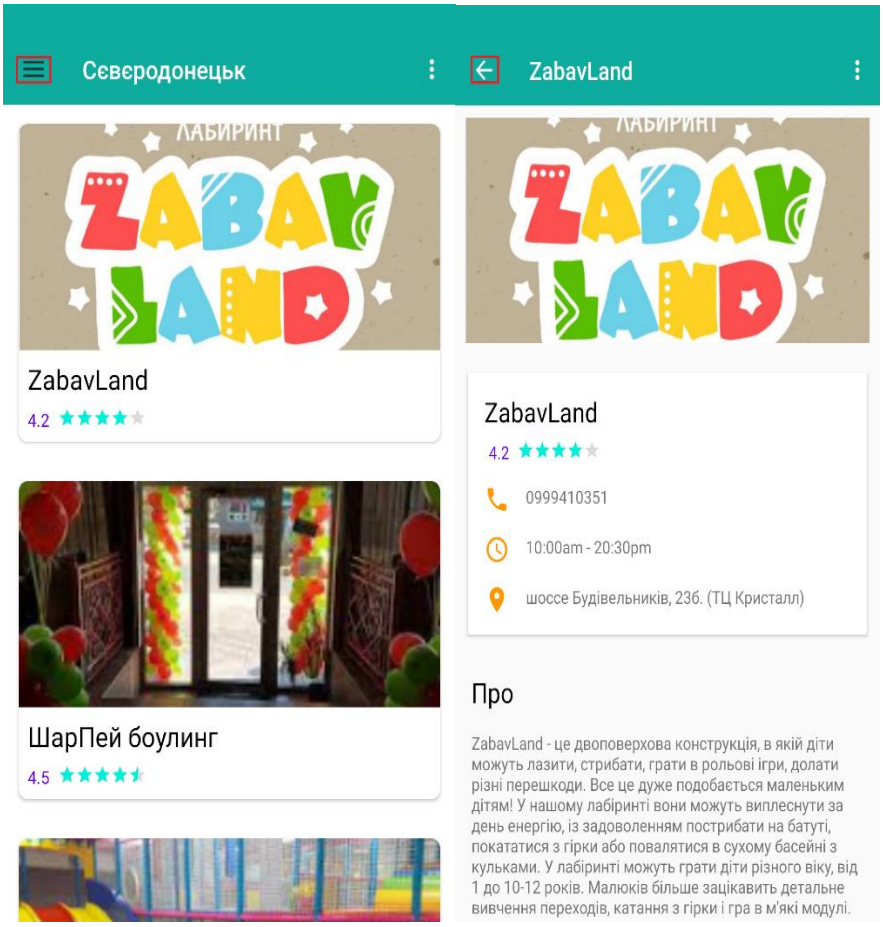

Рисунок 3.6 – Відображення місця призначення користувача, зліва показано,що користувач перебуває у пункті призначення верхнього рівня.

NavigationDrawer - це панель інтерфейсу, яка відображає головне меню навігації вашого додатка[1]. Для його підключення необхідно додати наступні строки в build.gradle

```
dependencies {
        implementation"androidx.navigation:navigation-fragment:$nav_version"
        implementation "androidx.navigation:navigation-ui:$nav_version"
}
```

або повністю підключити бібліотеки material design

```
dependencies {
```

```
implementation "com.google.android.material:material:$nav_version"
}
```

Для його використання необхідно додати його в відповідний файл xml в якому задати всі необхідні для відображення елементи.

NavigationDrawer складається з двох частин верхньої де відображаються дані завдяки звичайному файлу xml, та файлу типу menu, яке використовується для створення та відображення необхідних категорій та підкатегорій, які в свою чергу використовуються для перенаправлень до відповідних їм розділам[4].

```
<item android:title="@string/dodatok">
    <menu>
    <item
        android:id="@+id/map"
        android:icon="@drawable/ic_baseline_map_24"
        android:title="@string/mapa" />

    <item
        android:id="@+id/history"
        android:icon="@drawable/ic_baseline_history_24"
        android:title="@string/story" />

        <item
            android:id="@+id/univer"
            android:icon="@drawable/study"
            android:title="@string/vuz" />
    </menu>
</item>
```

Рисунок 3.7 – Xml код додатку NavigationDrawer

Він з’являється, коли користувач торкається значка навігації на панелі програм, має вигляд трьох горизонтальних ліній і знаходиться вгорі в крайній лівій частині панелі, або коли користувач проводить пальцем від лівого краю екрана. В цей момент злів з’являється в якості спливаючого меню NavigationDrawer і надає користувачу можливість використовувати елементи

навігації зазначені в додатку, вони мають вигляд списка можуть бути позначені зображеннями та інколи виділені кольором.Також іноді представляються не єдиним списком, а з підгрупами.

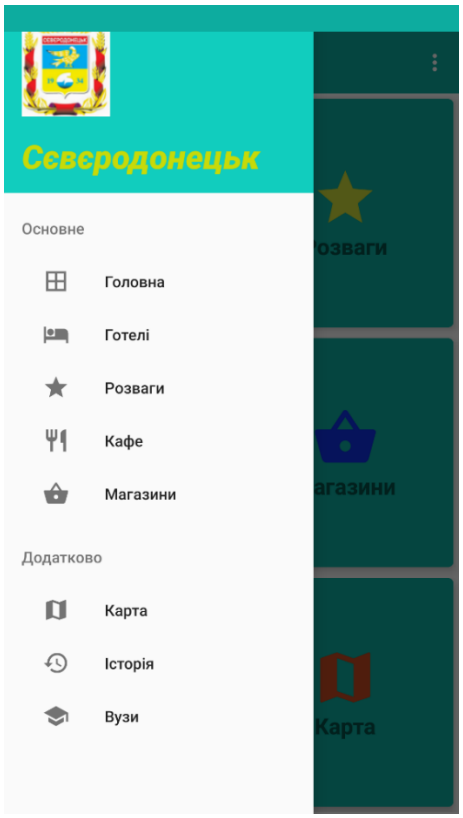

Рисунок 3.8- Використання NavigationDrawer

Коли користувач перебуває в іншому пункті призначення, кнопка навігації відображається як кнопка назад .

Щоб налаштувати кнопку навігації, використовуючи лише пункт початку як пункт призначення верхнього рівня, створюється AppBarConfiguration об’єкт і передається відповідний йому графік навігації.

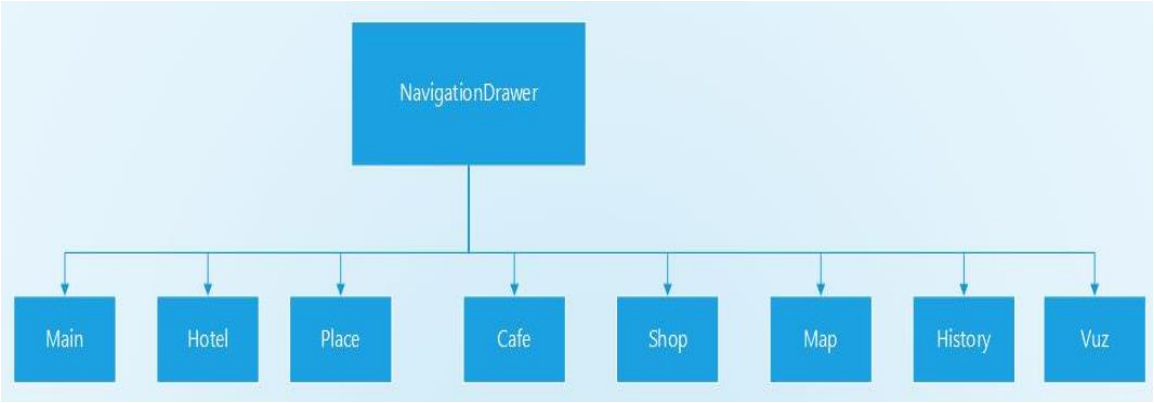

Рисунок 3.9 Структурна схема NavigationDrawer

При обранні користувачем певного елементу, певної категорії,для перегляду його додаткової інформації, відбувається перехід в DetailActivity в якій в залежності від обраного до цього елемента відображаються основні відомості: адреси, номери телефонів, сайти. Також тут користувач отримує змогу перейти на карту де відображено необхідний йому елемент.

Для відображення карт використовується безкоштовна бібліотека osmdroid, в основі якої лежить OpenStreetMap,що забезпечує високу точніть геоданних, та повноцінну роботу карти без необхідності підключення інтернету. Такаж для поліпшення використання було ствреноtablayout для навігації між категоріями на карті.

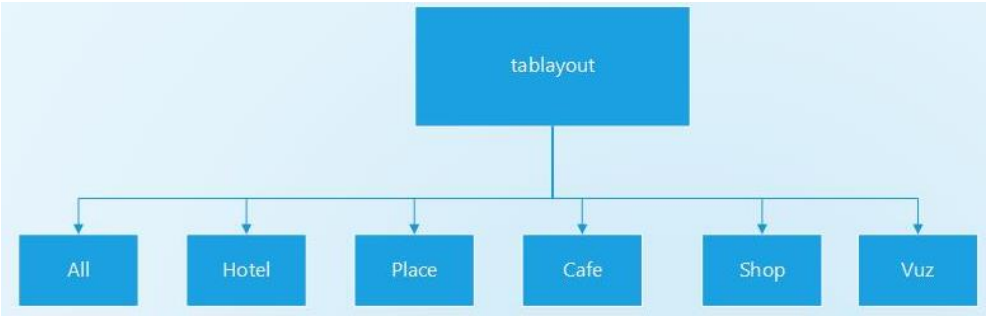

Рисунок 3.10 Структура tablayout

TabLayout забезпечує горизонтальний макет для відображення вкладок. Заповнення відображуваних вкладок здійснюється через екземпляри TabLayout.Tab. Ви створюєте вкладки за допомогою newTab (). Звідти ви можете змінити мітку або піктограму вкладки за допомогою TabLayout.Tab.setText (int) та TabLayout.Tab.setIcon (int) відповідно. Щоб відобразити вкладку, вам потрібно додати її до макета за допомогою одного з методів addTab (Tab) [14].

Також слід додати слухача через addOnTabSelectedListener (OnTabSelectedListener), щоб отримати сповіщення про зміну стану вибору будь-якої вкладки.

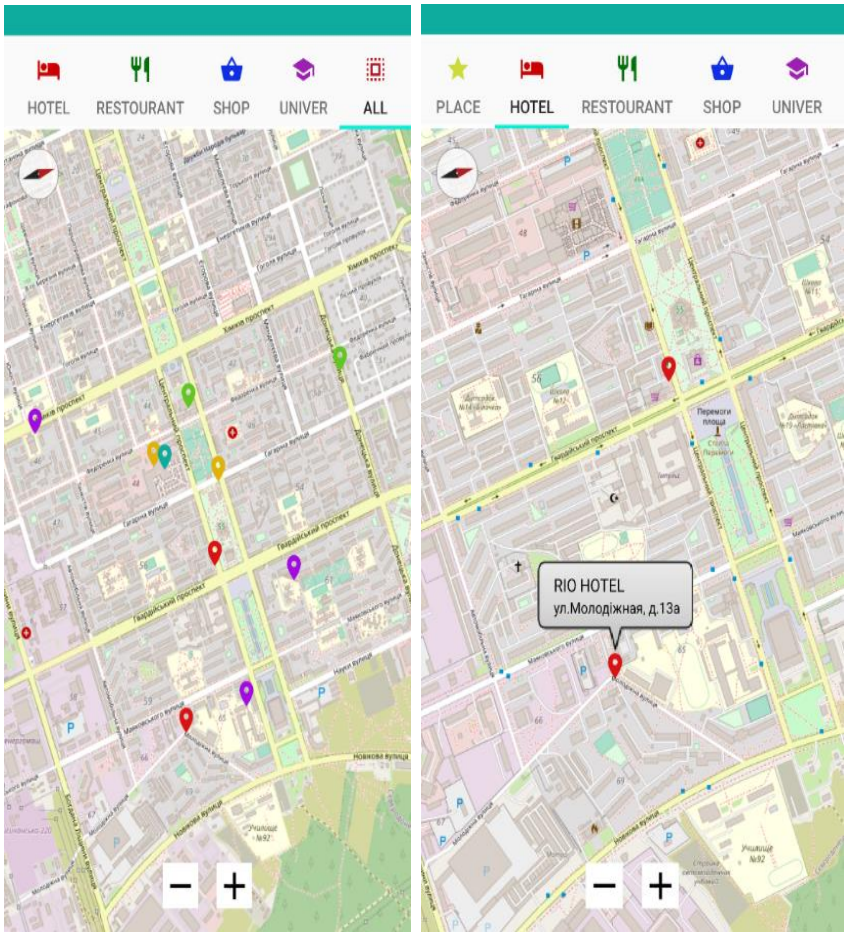

Рисунок 3.11 - Відображення маркерів та додаткової інформації при натисканні на карті

Для відображення маркерів через використано компонент osmdroid під назвою Overlay.

Overlay - клас, що представляє накладення, яке може відображатися поверх MapView. Щоб додати накладення, слід створити підклас, екземпляр та додати його до списку, отриманого з getOverlays() MapView.

Цей клас реалізує форму обробки жестів, подібну до GestureDetector.SimpleOnGestureListener та GestureDetector.OnGestureListener. Різниця полягає в тому, що є додатковий аргумент для предметів [15].

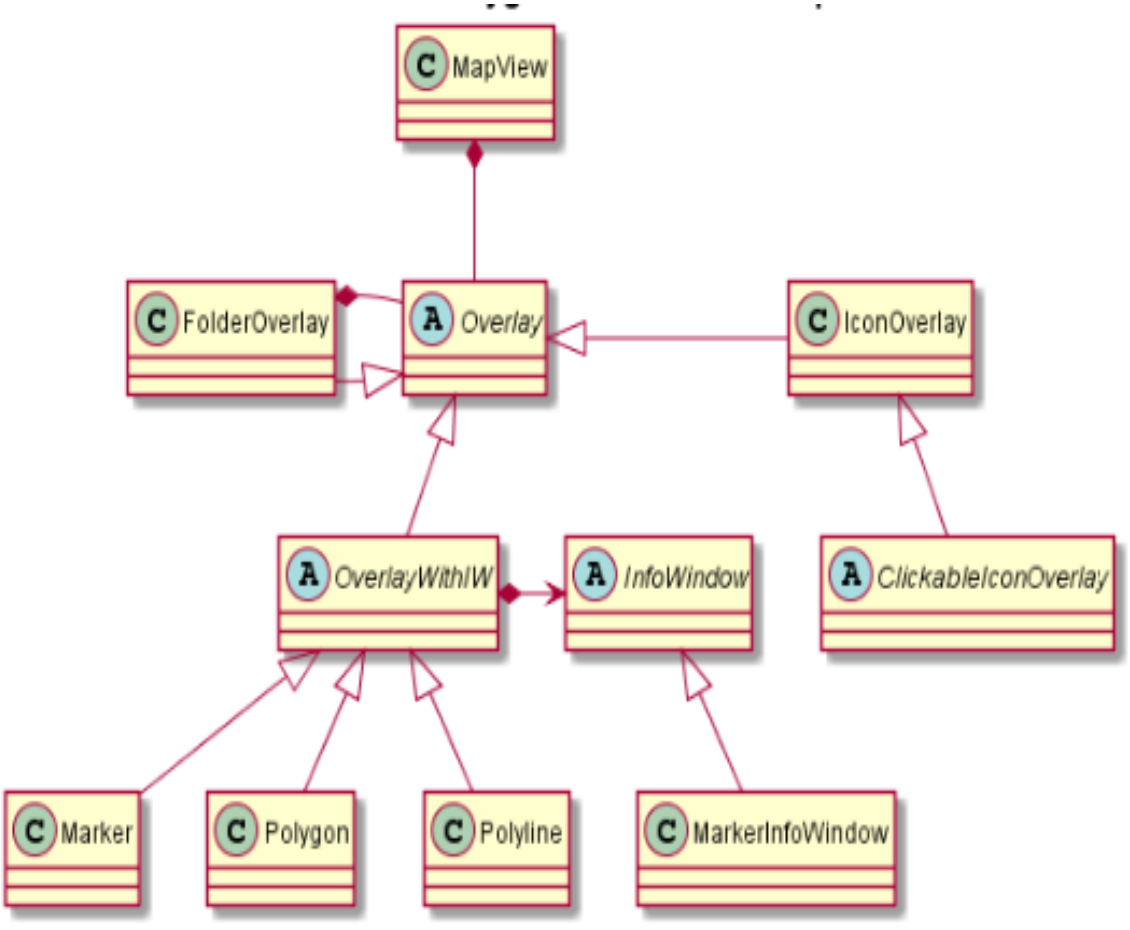

Рисунок.3.12 - Схема роботи карти з Overlay в додатку

Для показу інформації при натисканні на меркер використовується InfoWindow

InfoWindow- абстрактний клас, спливаюче подання MapView, яке може відображатися на , пов'язаному з ним IGeoPoint. Типове використання: бульбашкоподібні вікна, що відображаються при клацанні накладеного елемента (тобто Marker). InfoWindow імітує клас InfoWindow API Google Maps JavaScript V3 в Google Maps[15].Основні відмінності:

- Структура та зміст подання несуть відповідальність абонента.
- Те саме Інформаційне вікно може бути пов'язане з багатьма елементами.

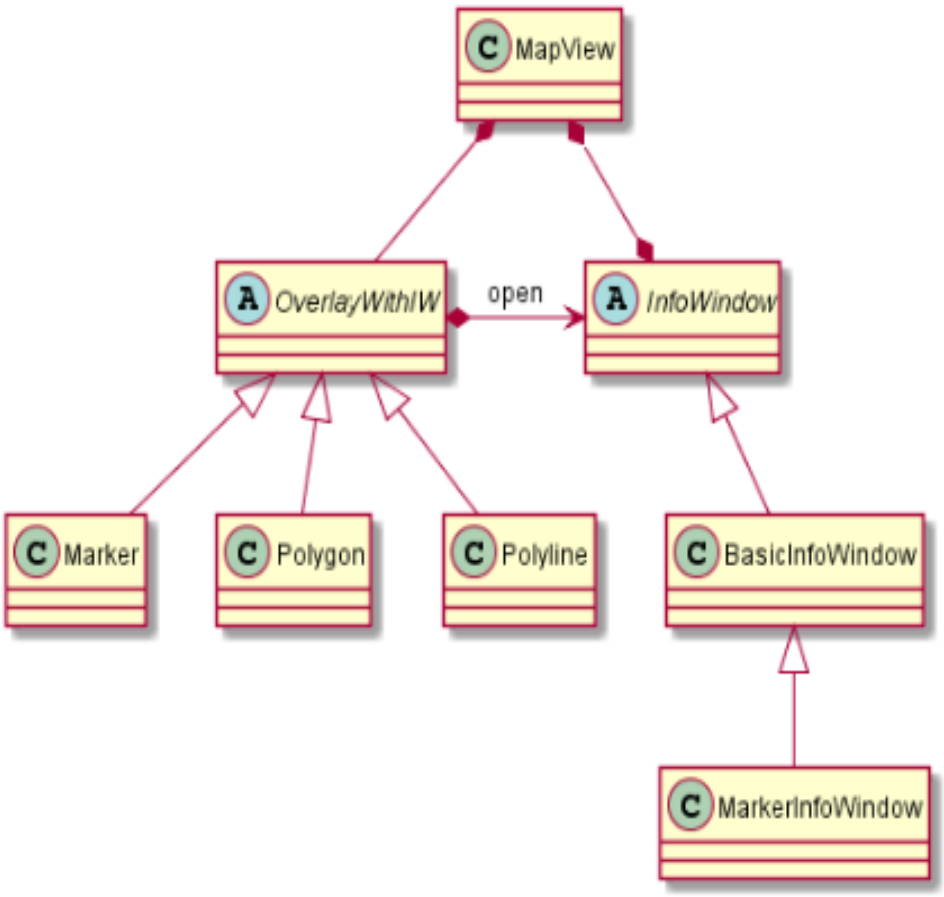

Рисунок 3.13 - Схема роботи карти з InfoWindow в додатку

Також варто зазначити що для використання osmdroid необхідно в файл build.gradle і Android Manifest необхідно додати дозволи та залежності.

В build.gradle:

```
dependencies {
    implementation 'org.osmdroid:osmdroid-android:6.1.6 '
}
```

В Android Manifest:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Рисунок 3.14 Android Manifest для osmdroid

Для відображення додаткових елементів керування та інформації було використано плаваюче меню, в ньому відображена інформація про автора додатку, занесені номери екстрених служб та таксі, також саме тут користувач матиме змогу відкрити простий браузер, якщо не зможе знайти необхідну йому інформацію в додатку.

Меню є загальним компонентом інтерфейсу користувача у багатьох типах програм. Щоб забезпечити знайомий та послідовний досвід користувача, використовується Menu API [7].

У спливаючому меню відображається список елементів у вертикальному списку, який прикріплений до подання, яке викликало меню. Це добре для забезпечення переповнення дій, що стосуються конкретного вмісту, або для надання опцій для другої частини команди.

Дії у спливаючому меню не повинні безпосередньо впливати на відповідний вміст - для цього призначені контекстні дії. Швидше, спливаюче меню призначене для розширених дій, які стосуються регіонів вмісту у вашій діяльності.

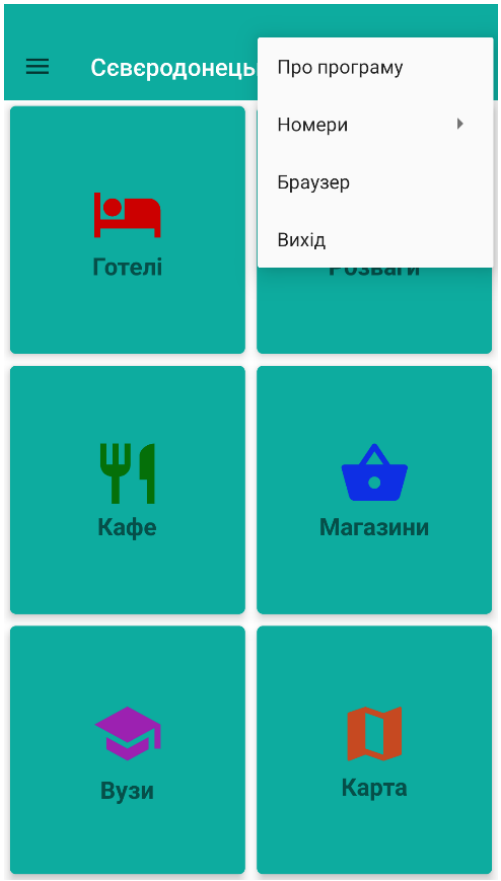

Рисунок 3.15 – Використання спливаючого меню

Для створення спливаючого меню меню використовують такі компоненти як:

<item> - Елемент підтримує кілька атрибутів , які можна використовувати , щоб визначити зовнішній вигляд і поведінку по елементу. Елементи у наведеному меню включають такі атрибути:

android:id - Ідентифікатор ресурсу, унікальний для елемента, який дозволяє програмі розпізнавати елемент, коли користувач його вибирає.

android:icon - Посилання на малюнок, який можна використовувати як піктограму елемента.

android:title - Посилання на рядок, який слід використовувати як заголовок елемента.

android:showAsAction - Вказує, коли і як цей елемент повинен відображатися як елемент дії на панелі програми.

```
<item android:id="@+id/item2"
    android:title="@string/num"
    app:showAsAction="never">
    <menu>
        <item android:id="@+id/police"
            android:title="@string/police"
            android:icon="@drawable/ic_phone"/>
```

Рисунок 3.16– Xml-код спливаючого меню

3.2 Організація переходів між компонентами додатку

Для переходів між компонентами додатку використовуються наміри(Intent) для активностей та fragmentTransaction.replace для фрагментів.

3.2.1 Взаємодія між Activity

Intent - це асинхронні повідомлення, які дозволяють компонентам програми вимагати функціональності від інших компонентів Android.

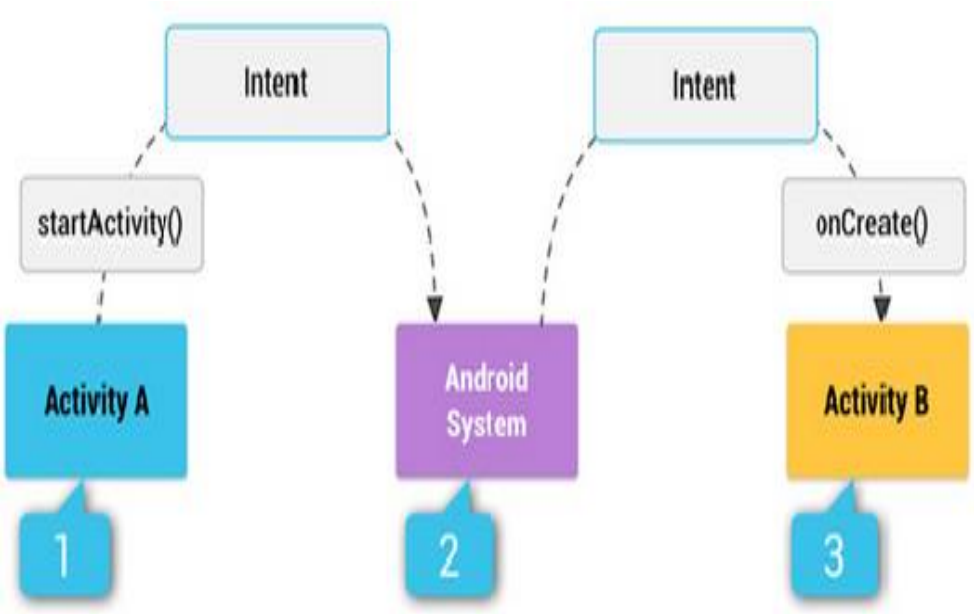

Рисунок 3.17 – Схема використання intent

Наміри дозволяють взаємодіяти з компонентами самих програм, а також з компонентами, внесеними іншими програмами.Вони часто також включають дані, пов’язані з дією, такі як адреса, яку ви хочете переглянути, або повідомлення електронної пошти, яке потрібно надіслати. Залежно від наміру, який ви хочете створити, дані моють бути Uri, певного типу даних, або задум може взагалі не потребувати даних.

Якщо ваші дані є Uri, існує простий Intent() конструктор, за допомогою якого ви можете визначити дію та дані.

Після створення Intent та встановлення додаткової інформації використовують, startActivity() щоб надіслати її до системи.

Наприклад, може розпочати зовнішню діяльність для відкриття браузера або іншої активності.

```
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.com"));
startActivity(intent);
```

Рисунок 3.18 -. Відкриття браузеру

```
Intent intent = new Intent( packageContext: this,MapTabActivity.class);
startActivity(intent);
```

Рисунок 3.19 - Відкритя іншої активності

Android підтримує явні та неявні наміри. Додаток може визначити цільовий компонент безпосередньо у намірі (явний намір) або попросити систему Android оцінити зареєстровані компоненти використовуючи доступні дані (неявні наміри).

Явні наміри явно визначають компонент, який повинен викликати система Android, використовуючи клас Java як ідентифікатор.

Явні наміри зазвичай використовуються в програмі, оскільки класи в програмі контролюються розробником програми.

Неявні наміри вказують дію, яку слід виконати, і необов'язково дані, що надають вміст для дії. Коли неявний намір надіслано в систему, він шукає тільки ті компоненти, які призначені для певних дій та типів даних.

В випадку коли знайдено тільки один компонент, система запускає цей компонент відразу. Якщо система Android ідентифікує кілька компонентів, користувач отримає діалогове вікно вибору та може вирішити, який компонент слід використовувати для наміру.

3.2.2 Взаємодія між Fragment

Фрагменти не можуть безпосередньо взаємодіяти між собою. Для цього треба звертатися до контексту, в якості якого виступає клас Activity. Для звернення до activity, як правило, створюється вкладений інтерфейс.

Під час виконання операцій FragmentManager може додавати, видаляти, замінювати та виконувати інші дії з фрагментами у відповідь на взаємодію користувача.

Кожен набір змін фрагмента, який ви фіксуєте, називається транзакцією, і ви можете вказати, що робити всередині транзакції, використовуючи API, надані FragmentTransaction класом.

Можно згрупувати кілька дій в одну транзакцію - наприклад, транзакція може додавати або замінювати кілька фрагментів. Це групування може бути корисним, коли на одному екрані відображаються кілька фрагментів наприклад, з розділеними поданнями.

Можно зберегти кожну транзакцію у зворотному стеку, яким керує FragmentManager, дозволяючи користувачеві переходити назад через зміни фрагментів - подібно до переходу назад за діями.

Щоб додати фрагмент до FragmentManager, необхідно використовувати add() за транзакцією. Цей метод отримує ідентифікатор контейнера для фрагмента, а також ім'я класу фрагмента, який ви хочете додати. Доданий фрагмент переміщується до RESUMED стану.

Щоб видалити фрагмент із хосту, використовується remove(), передавши екземпляр фрагмента, який було отримано з менеджера фрагментів через findFragmentById() або findFragmentByTag().

Якщо представлення фрагмента раніше було додано до контейнера, представлення видаляється з контейнера в цей момент. Вилучений фрагмент переміщується до DESTROYED стану.

Використовуючи replace() для заміни існуючого фрагмента в контейнері екземпляра нового класу фрагментів, який надається.

Виклик replace() еквівалентний виклику remove() з фрагментом у контейнері та додаванням нового фрагмента до цього самого контейнера.

Виклик commit() не виконує операцію відразу. Навпаки, транзакція планується запустити в основному потоці інтерфейсу користувача, як тільки вона зможе це зробити. Однак при необхідності ви можете зателефонувати, commitNow()щоб негайно запустити транзакцію фрагментів у вашому потоці інтерфейсу.

```
FragmentManager fragmentManager=getSupportFragmentManager();
FragmentTransaction fragmentTransaction=fragmentManager.beginTransaction()
fragmentTransaction.replace(R.id.frame,fragment).commit();
drawerLayout.closeDrawer(GravityCompat.START);
fragmentTransaction.addToBackStack(null);
```

Рисунок 3.20 - Використання FragmentManager для преходів

Висновок:

Цей дипломний проект полягав у розробці моіблього додатку.

У наш час мобільні міцно ввійшли в повсякденного життя кожної людини. Тому розробка додатків є дуже важливою адже в майбутньому потреба в ній буде лише рости.

У цій роботі ми розглянули основні теоретичні та практичні питання створення мобільниї додатків - мовні та програмні засоби, що використовуються при розробці. Також були проаналізовані сучасні технології мобільної розробки, що дозволяють створювати повноцінні мобільні додатки. При розробці додатку були використані готові xml-модулі які є частиною однієї з основних бібліотек предустановлених в Android studio - Material Design. Дані модулі були обрані з урахуванням специфіки додатку та успішно запущенні в його структуру.

В результаті цих дій програмний продукт відповідає поставленим вимогам, є повністю функціонуючим, готовим до використання.

СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Glossary of Android - [Електронний ресурс] - Режим доступу:

https://developers.google.com/android/for-all/vocab-words/ - Дата звернення: 12.05.2021

2. Дизайн Android [Електронний ресурс] - Режим доступу:

https://medium.com/swlh/material-design-an-in-depth-look-d4f4055a5ecf 15 - Дата звернення: 12.05.2021

3. NavigationUI - [Електронний ресурс] - Режим доступу: https://developer.android.com/guide/navigation/navigation-ui - Дата звернення: 09.05.2021

4. Navigation drawer - [Електронний ресурс] - Режим доступу: https://material.io/components/navigation-drawer - Дата звернення: 12.05.2021

5. Recycleview - [Електронний ресурс] - Режим доступу: https://metanit.com/java/android/5.11.php - Дата звернення: 10.05.2021

6. Android CardView - [Електронний ресурс] - Режим доступу: https://betacode.net/12701/android-cardview - Дата звернення: 12.05.2021

7. Андроид Menu - [Електронний ресурс] - Режим доступу: http://developer.alexanderklimov.ru/android/menu.php - Дата звернення: 12.05.2021

8. Navigation drawer - [Електронний ресурс] - Режим доступу: https://material.io/components/navigation-drawer - Дата звернення: 12.05.2021

9. Архитектура Android-приложений. Часть III - [Електронний ресурс] - Режим доступу: https://habr.com/ru/post/141201/ - Дата звернення: 12.05.2021

10. Виды приложений и их структура - [Електронний ресурс] - Режим доступу: https://intuit.ru/studies/courses/12643/1191/lecture/21983?page=2

11. Activity (Активность, Деятельность) - [Електронний ресурс] - Режим доступу: http://developer.alexanderklimov.ru/android/theory/activity-theory.php - Дата звернення: 12.05.2021

12. Android vs IOS - [Електронний ресурс] - Режим доступу: https://theappsolutions.com/blog/development/ios-vs-android/ - Дата звернення: 11.05.2021

13. Что такое Android Fragment - [Електронний ресурс] - Режим доступу: https://javadevblog.com/chto-takoe-android-fragment-zhiznenny-j-tsikl-i-primer-ispol-zovaniya-fragmentov.html - Дата звернення: 12.05.2021

14. Делаем вкладки с помощью TabLayout - [Електронний ресурс] - Режим доступу: https://android-tools.ru/coding/delaem-vkladki-s-pomoshhyu-tablayout/ - Дата звернення: 11.05.2021

15. Osmdroid API - [Електронний ресурс] - Режим доступу: https://github.com/osmdroid/osmdroid - Дата звернення: 12.05.2021

16. Mobile Apps - [Електронний ресурс] - Режим доступу: https://science.jrank.org/programming/Mobile_Apps.html- Дата звернення: 12.05.2021

17. Mobile Apps - [Електронний ресурс] - Режим доступу: https://science.jrank.org/programming/Mobile_Apps.html - Дата звернення: 12.05.2021

18. Филлипс Б., Стюарт К., Марсикано К. «Android. Программирование для профессионалов» Питер, 2017 год, 688 стр., 3-е изд., ISBN: 978-5-4461-0413-0

19. П. Дейтел, Х. Дейтел, Э. Дейтел, М. Моргано Д27 Android для программистов: создаём приложения. — СПб.: Питер, 2013. — 560 с.: ил.

ДОДАТКИ

Додаток А Частина лістингу програмного коду

```
public class MainActivity extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener {

    DrawerLayout drawerLayout;

    ActionBarDrawerToggle toggle;

    Toolbar toolbar;

    NavigationView navigationView;

    Fragment fragment;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        drawerLayout = findViewById(R.id.drawer);

        navigationView = findViewById(R.id.nav_view);

        toolbar = findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);

        toggle = new ActionBarDrawerToggle(this, drawerLayout, toolbar, R.string.open, R.string.close);

        drawerLayout.addDrawerListener(toggle);

        toggle.syncState();



        navigationView.bringToFront();
```

```
    navigationView.setNavigationItemSelectedListener(this);
    onNavigationItemSelected(navigationView.getMenu().getItem(0).setChecked(true));
}

@SuppressLint("NonConstantResourceId")
@Override
public boolean onNavigationItemSelected(@NonNull MenuItem item) {
    int id=item.getItemId();
    switch (id)
    {
        case R.id.categoty:
            fragment = new CategoriesFragment();
            loadFragment(fragment);
            break;
        case R.id.hotel:
            fragment=new HotelFragment();
            loadFragment(fragment);
            ActionBar actionBar = getSupportActionBar();
            assert actionBar != null;
            actionBar.setTitle(getString(R.string.gotel));
            break;
        case R.id.place:
            fragment=new PlaceFragment();
```

```
            loadFragment(fragment);
            actionBar = getSupportActionBar();
            assert actionBar != null;
            actionBar.setTitle(getString(R.string.activ));
            break;
        case R.id.restaurant:
            fragment=new RestaurantFragment();
            loadFragment(fragment);
            actionBar = getSupportActionBar();
            assert actionBar != null;
            actionBar.setTitle(getString(R.string.cafe));
            break;
        case R.id.shop:
            fragment=new ShopFragment();
            loadFragment(fragment);
            actionBar = getSupportActionBar();
            assert actionBar != null;
            actionBar.setTitle(getString(R.string.shopp));
            break;
        case R.id.univer:
            fragment=new UniverFragment();
            loadFragment(fragment);
            actionBar = getSupportActionBar();
```

```
            assert actionBar != null;
            actionBar.setTitle("Вузи");
            break;
        case R.id.history:
            fragment=new HistoryFragment();
            loadFragment(fragment);
            actionBar = getSupportActionBar();
            assert actionBar != null;
            actionBar.setTitle("Історія");
            break;
        case R.id.map:
            Intent intent = new Intent(this,MapTabActivity.class);
            startActivity(intent);
            break;
        default:
            fragment = new CategoriesFragment();
            loadFragment(fragment);
            return true;
    }
    return true;
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
```

```
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu_th,menu);
    return true;
}
@SuppressLint("NonConstantResourceId")
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    switch (item.getItemId()){
        case R.id.item1:
            fragment=new FaQFragment();
            loadFragment(fragment);
            return true;
        case R.id.item2:
            return true;
        case R.id.police:
            Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:" +"102"));
            startActivity(intent);
            return true;
        case R.id.doc:
            Intent intent2 = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:" +"103"));
            startActivity(intent2);
            return true;
        case R.id.mns:
```

```
                Intent intent3= new Intent(Intent.ACTION_DIAL, Uri.parse("tel:" +"101"));
                startActivity(intent3);
                return true;
            case R.id.taxi:
                Intent intent4 = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:" +"+380955522555"));
                startActivity(intent4);
                return true;
            case R.id.item3:
                browser();
                return true;
            case R.id.item4:
                exit();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
    private void browser() {
        Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.com"));
        startActivity(intent);
    }
    public void exit() {
```

```
        AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
        alertDialogBuilder.setTitle("Ви дійсно бажаєте вийти");
        alertDialogBuilder
                .setMessage("Для виходу натисніть Так!")
                .setCancelable(false)
                .setPositiveButton("Так",
                        (dialog, id) -> {
                            moveTaskToBack(true);
                            android.os.Process.killProcess(android.os.Process.myPid());
                            System.exit(1);
                        })
                .setNegativeButton("Hi", (dialog, id) -> dialog.cancel());
        AlertDialog alertDialog = alertDialogBuilder.create();
        alertDialog.show();
    }
    private void loadFragment(Fragment fragment) {
        FragmentManager fragmentManager=getSupportFragmentManager();
        FragmentTransaction fragmentTransaction=fragmentManager.beginTransaction();
        fragmentTransaction.replace(R.id.frame,fragment).commit();
        drawerLayout.closeDrawer(GravityCompat.START);
        fragmentTransaction.addToBackStack(null);
    }
}
```