МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.
«____»_____ 20__ р.

ДИПЛОМНИЙ ПРОЕКТ (РОБОТА) БАКАЛАВРА

ПОЯСНЮВАЛЬНА ЗАПИСКА

НА ТЕМУ:

Додаток для ОС Android з використанням мов XML, Java

Освітньо-кваліфікаційний рівень "бакалавр"
Спеціальність 122 – "комп'ютерні науки"

Керівник проекту:	_____ (підпис)	Щербаков Є.В. (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Критська Я.О. (ініціали, прізвище)
Студент:	_____ (підпис)	Федоряченко О.І. (ініціали, прізвище)
Група:		КН-14д

Сєвєродонецьк 2018

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки

Кафедра Комп'ютерних наук та інженерії

Освітньо-кваліфікаційний рівень Бакалавр

Напрям підготовки _____
(шифр і назва)

Спеціальність 6.050101 – "комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри _____
_____ І.С. Скарга-Бандурова
«_____» _____20____р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) БАКАЛАВРА

Федоряченку Олексію Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Додаток для ОС Android з використанням мов XML, Java

керівник проекту (роботи) доц. Щербаков Є.В.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від " " 201_ р. № ____

2. Термін подання студентом роботи

3. Вихідні дані до роботи матеріали переддипломної практики

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз та постановка задачі; огляд існуючих засобів розробки додатків, реалізація додатку.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Комп'ютерна презентація

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Критська Я. О.		

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання, збір матеріалів	14.05.2018 – 17.05.2018	
2	Огляд літератури й обгрунтування необхідності розробки	17.05.2018 – 20.05.2018	
3	Розробка технічного завдання	20.05.2018 – 22.05.2018	
4	Проектування структури додатка	22.05.2018 – 24.05.2018	
5	Розробка додатку на ОС Android	24.05.2018 – 30.05.2018	
6	Інформаційне наповнення додатку	30.05.2018 – 03.06.2018	
7	Перевірка реалізованого фунціоналу	03.06.2018 – 05.06.2018	
8	Охорона праці та безпека в надзвичайних ситуаціях	05.06.2018 – 07.06.2018	
9	Оформлення пояснювальної записки	07.06.2018 – 09.06.2018	

Студент _____ Федоряченко О.І.
(підпис) (прізвище та ініціали)

Керівник _____ Щербаков Є.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту (роботи) бакалавра: 82 с., 27 рис., 5 табл., 28 бібліографічних джерел посилань, 2 додатка.

Об'єкт розробки: Додаток для ОС Android з використанням мов XML, Java.

Мета роботи: розробка додатку для ОС Android з використанням мов XML, Java.

В проекті виконано:

1. У розділі «Аналіз засобів розробки додатків для ОС Android з використанням мов XML, Java» було виконано зрівняння аналогічних додатків, був проведений огляд предметної області, були поставлені задачі щодо розробки системи.

2. У розділі «Проектування додатків для ОС Android з використанням мов XML, Java» були розглянуті інструменти, за допомогою яких розроблявся додаток.

3. У розділі «Реалізація додатку навігатора та гіда по місту для ОС Android з використанням мов XML, Java» була описана архітектура, функціональність, структура і активності розробленого додатку.

4. У розділі «Охорона праці та безпека в надзвичайних ситуаціях. Екологія» був проведений аналіз шкідливих виробничих факторів. І на основі цього аналізу були запропоновані заходи усунення цих факторів.

Отримано наступні результати: додаток навігатор та гід по місту для ОС Android.

Ключові слова: Android, XML, Java, додаток, Google.

Умови одержання дипломного проекту: СНУ ім. В. Даля, пр. Центральний 59-А,. м. Сєвєродонецьк, 93400.

ЗМІСТ

ВСТУП

В наш час люди дуже часто користуються додатками на своїх мобільних пристроях з ОС Android та IOS, дуже зручно мати все необхідне в одному пристрої. Додатки є найбільш відповідним рішенням проблеми з орієнтуванням. З додатком навігатором у смартфоні значно простіше знайти необхідне місцезнаходження крамниці, ресторану, готелю і т. д. Особливо корисно мати такий додаток у дорозі, подорожі, та просто в незнайомому місці. Користувач може обрати з основних трьох категорій, або отримати перелік закладів, які його оточують, та прокласти маршрут до одного з них, що значно заощадить час на пошуки.

Завдяки додатку навігатору можна поділитися своїм місцезнаходженням з друзями або родичами, щоб вони мали можливість дізнатися, де ви зараз знаходитесь.

Створення додатку навігатора та гіда по місту для ОС Android забезпечить їх більш детальною інформацію про місто, місце знаходження, оточуючі об’єкти, та допоможе прокласти маршрут до потрібного їм закладу самостійно. Легкий інтерфейс дозволить швидко ознайомитись і одразу розпочати використовувати на практиці.

В дипломному проекті була поставлена задача розробки додатку навігатора та гіду по місту для ОС Android з використанням мов XML, Java. Додаток для ОС Android з простим інтерфейсом для зручного користування являє собою помічник, який зможе показати місцезнаходження користувача, показати об’єкти, які знаходяться поряд, вибрати місце з основних категорій, а після вибору потрібного закладу дасть по ньому інформацію та прокладе до нього маршрут.

Мета дипломного проекту – проектування та реалізація додатку навігатора та гіда по місту для ОС Android з використанням мов XML, Java.

1. АНАЛІЗ ЗАСОБІВ РОЗРОБКИ ДОДАТКІВ ДЛЯ ОС ANDROID З ВИКОРИСТАННЯМ МОВ XML, JAVA

1.1. Огляд предметної області

З 2002 року розпочалася історія Android, коли корпорація Google зацікавилася напрацюваннями Енді Рубіна і вирішила зробити з цього великий проект. В 2007 році Google вирішив організувати великий альянс розробників мобільних пристроїв з метою просувати Android, як операційну систему для телефонів, і у них це відмінно вийшло. Android займає лідируючу позицію на ринку операційних систем. Прогнози щодо даної платформи позитивні. Вибір розробки програми для Android є дуже актуальним. Важливими аспектами зростання динаміки з'явилися такі характеристики як, відкритість системи, можливість вносити зміни в основні програми, та звичайно можливість швидкої і легкої розробки.

Розробнику який працює з платформою Android дається можливість писати код на Java абстрагуючись від ядра. У цієї операційної системи є такі переваги як: фреймворк, який має широкий набір API для створінь різноманітних видів додатків, а також дає можливість повторного використання і заміни компонентів, що пропонуються платформою і іншими додатками. Інструментами Android виступає: комунікації, база даних SQLite, Media Player, 2D і 3D графіка, а також протоколи обміну і різні бібліотеки. Також наявність віртуальної машини Dalvik, що забезпечує запуск додатків. Для створення додатку на Java для відкритої платформи Android перше, що знадобиться, це Java Development Kit від Oracle.

Java Development Kit - це багатоплатформний інтерактивний пакет створений для розробників, працюючих на мові Java. Комплект складається з декількох компонентів таких як: приклади та шаблони, компілятор Java, стандартні бібліотеки, а також утиліти які знадобляться для роботи.

Також необхідно встановити IDE додаток, який допомагає програмістам в написанні коду. Ця програма надає стислий набір інструментів на кшталт відладчиків, компіляторів та багато чого іншого. Такі інтегровані середовища розробки використовуються сезонними розробниками і новачками, які мають бажання створити додаток.

У наш час людям хочеться досягти максимального комфорту в кожній зі сфер життя, в тому числі і в використанні міжнародної мережі Інтернет. З розвитком високих технологій, які дозволяють створювати персональні мобільні пристрої і різні гаджети, корпоративний ринок отримав потужний стимул до розвитку. Користувач, бажаючи завжди залишатися в мережі, використовує в якості комунікатора телефон. Це зумовило появу мобільного Інтернету. Телефони грають важливу роль в повсякденному житті: з їх допомогою читають файли, заходять на пошту, дивляться відео, слехають музику, друкують документи за допомогою мережевого принтера, та багато іншого. Смартфони люди носять з собою завжди і скрізь. При перебуванні поза домом, або під час подорожей і відряджень можна замість ноутбука підключатися за допомогою планшета або смартфону. Портативні пристрої значно перевершують рідкокристалічні громадини з суперпотужним процесором. Головні переваги – це звичайно розмір, тривалість роботи без підзарядки, і багатофункціональність.

Варто визнати, що майбутнє ПК – у портативній, легкої та функціональної техніці – планшетах, електронних книжках, нетбуках і смартфонах – і все це в основному працює саме на операційній системі Android. Згідно зі статистикою, саме Android лідирує зараз на ринку смартфонів, займаючи на ньому більше половини всього обсягу продажів. Для сучасної людини важливо постійно мати доступ до Інтернету, електронної пошти, соціальних мереж. Ефективність та функціональність «міні-комп'ютерів» звичайно не була б доведена до такого високого рівня без спеціалізованих додатків.

Додатки на сьогодняшній день створюються з різними цілями. Якщо раніше основною метою додатків була розвага користувача, то зараз є програми і по доставці піци, і за викликом таксі, і ті що вказують маршрут, ті що надають допомогу в пошуку магазину або необхідного товару, і т.д. Що звичайно говорить нам про те, що ринок додатків значно виріс. Утиліти лягли в основу повсюдного обміну даними та інформацією, що дозволяє економити дорогоцінний час кожного.

Основні програми діляться на ті, які необхідні для приємного проведення часу, і ті, які використовуються виключно в робочих цілях. Перша група включає іграшки та розважальні програми, софт для відтворення аудіо та відео, засоби для комунікації і т.д. Інший напрямок розрахований на комплексне рішення певного завдання. Зокрема деякі утиліти здатні контролювати перебіг бізнес процесів та складати аналітичні звіти.

Створення мобільних додатків другого типу більш поширене. Вони вже щільно увійшли в такі життєві напрямки як медицина, державні організації і навіть виробничі компанії. Розважальні утиліти можуть відіграти роль інструментів маркетингу для більшості підприємства, але навіть це не дозволяє їм скласти конкуренцію за сферою застосування діловому напрямку.

Сьогодні багато компаній, які надають ті чи інші послуги створюють свої додатки для того щоб спростити життя для своїх клієнтів, які будуть користуватися їх послугами. І це означає, що тепер клієнтові не потрібно буде дзвонити та домовлятися про все, а потрібно буде просто відкрити програму та натиснути на кнопку. Така простота дуже приваблює компанії, які хочуть вийти на новий рівень обслуговування. Тому багато компаній зараз замовляють додатки для смартфонів. Актуальність мобільних додатків очевидна. Головне, щоб напередодні розробки були чітко поставлені цілі софта і його застосування.

Розробляти додатки повинно з урахуванням особливостей мобільних пристроїв, такими як: відмінності інтерфейсу, інші розміри екрану, сенсорне управління. Якщо інтерфейс розрахований на тактильне управління, необхідно збільшити розміри відображуваних елементів, адже якщо вони будуть занадто малими, потрапити в них пальцем буде значно важче.

Також варто згадати про категорію користувачів які активно використовують корпоративні додатки. Зокрема, категорія осіб, які займають керівні посади, визначає мобільне робоче місце як єдино оптимальний варіант, так як саме за допомогою додатків вдається перенаправляти потоки інформації і працювати з великими обсягами.

Android – це платформа з відкритим кодом, заснована на ядрі Linux. Встановлена вона на тисячах девайсів широкого кола виробників. З її допомогою можна отримати доступ до всіх компонентів пристрою, на якому виконується ця ОС, починаючи від низькорівневого програмування графіки і закінчуючи використанням вбудованої камери. XML є поширеним форматом для обміну інформацією в Інтернеті, тому велика ймовірність, що він знадобиться для доступу до даних в Web.

Ринок пропонує величезне різноманіття версій мобільних пристроїв, кожне з них вимагає певного виду платформи для роботи з додатками. На сьогодні найбільшою популярністю користуються платформи Android і iOS, а також BlackBerry і Linux - вони актуальні для роботи з корпоративними додатками. А не менш популярна Windows Phone більшою мірою орієнтована на ринок споживачів, ніж на корпоративну сферу.

Безкоштовні інструменти розробки для Android дозволяють швидко почати створення безкоштовних або майже безкоштовних додатків. Коли ви готові показати світу вашу програму, ви можете опублікувати її за допомогою Android Market. Публікація в Android Market вимагає одноразового реєстраційного внеску і, на відміну від App Store Apple, робить ваш додаток доступним для скачування і покупки після швидкого огляду - якщо програма не порушує правила і закон.

Відмінності Android SDK, які пропонує Вам переваги як розробнику:

1) Пакет Android SDK доступний для Windows, Mac і Linux, тому вам не потрібно платити за нове "залізо" для створення програмного забезпечення.
2) SDK вбудований в Java.
3) З урахуванням поширення програми через Android Market, воно буде доступно відразу для сотень тисяч користувачів. Ви не обмежуєтеся тільки офіційним Market'ом, оскільки є альтернативи. Наприклад, ви можете опублікувати додаток на своєму блозі.
4) Так само як технічна документація SDK, для розробників Android створюються нові ресурси. Платформа набирає все більшої популярності серед користувачів і розробників.

Однією з найбільш привабливих рис платформи Android є використання мови програмування Java. SDK Android підтримує не всю, але досить велику частину можливостей стандартного середовища виконання Java (Java Runtime Environment - JRE). Сама платформа Java вже довгий час підтримує безліч різних способів використання XML, причому більшість API для Java, орієнтованих на XML, доступні в Android. Прикладами таких API можуть служити об'єктна модель документів (Document Object Model - DOM) і простий Java API для XML (Java's Simple API for XML - SAX), які вже багато років є частиною технології Java. Зворотним прикладом є новіший потоковий API (Streaming API for XML - StAX), який не підтримується в Android (при цьому до складу Android входить еквівалентна за своїми можливостями бібліотека).

У додатку Android всі зображення, ярлики і файли компонування користувальницького інтерфейсу розташовуються в каталозі (директорії) res В ньому міститься не менше чотирьох підкаталогів:

1) layout – містить XML-файли компонування користувальницького інтерфейсу Android;

2) drawable – включає в себе вимальовуючи обєкти, зокрема ярлик програми;
3) raw – містить файли, які можуть зчитуватися в потоковому режимі під час виконання програми. Такі необроблені файли відмінно підходять для того, щоб повідомляти додаткам інформацію для налагодження, так як знімають необхідність виходити в мережу і отримувати звідти дані;
4) values - включає в себе значення, які додаток буде зчитувати під час виконання, або статичні дані, які додаток буде використовувати для таких цілей.

Для доступу до даних каталогу res розробник Java, що віддає перевагу працювати в традиційному стилі, може писати такий код, в якому будуть будуватися відносні шляхи до файлів, а потім для відкриття ресурсів використовуватиметься файловий API.

Додатки отримують доступ до ресурсів цих каталогів, користуючись методом ContextgetResources і класом R.

В даному дипломному проекті поставлено задачу розробити додаток навігатор та гід по місту для ОС Android з використанням мов XML, Java, який орієнтований на туристів та людей не місцевих.

В наш час дуже актуально користуватися додатком навігатором для ОС Android, для цього достатньо мати смартфон або планшет з підключенням до Інтернету, це значно допоможе зорієнтуватись у дорозі, подорожі, та просто у незнайомому місці та заощадить час.

1.2. Мова програмування Java

Існує три основні версії мови Java: SE (Standard Edition) для індивідуальних користувачів, EE (Enterprise Edition) для великих колективів користувачів і ME (Micro Edition) для застарілих мобільних фліп-телефонів.

Більшість сучасних смартфонів Android використовують Java SE, а не Java ME.

Цікавим для ОС Android є те, що в ній в повній мірі використовується стандартна версія Java (Java SE 6 або 7), так само, як це робиться на PC. Коренем ієрархії служить клас Object. Необ'єкти в Java - це тільки прості числові, символьні і булеві типи. Основні елементи мови - об'єкти класів. Клас - це сукупність даних і методів (функцій, процедур) обробки цих даних. Класи організовані в ієрархію у вигляді "строго" дерева (множинного спадкоємства в дусі C ++ в Java немає). Будь-яка програма в Java - це клас (з методом main для додатка). Одиницею компіляції та виконання також є клас.

Класи в Java об'єднані в пакети (package), що полегшують розробку програм тієї чи іншої функціональності. Зазвичай в пакет потрапляють класи з різних гілок дерева ієрархії класів. Приклади пакетів: java.net - класи для організації мережевої взаємодії; java.io - класи введення-виведення; java.awt - класи компонентів графічного інтерфейсу користувача.

Оригінальний текст програми (класу) обробляється компілятором Java, проте, на відміну від традиційних компіляторів (таких, як C і C ++), він створює не об'єктний файл (наприклад, формату .EXE), що містить машинні команди, а файл з послідовністю байт-кодів . На етапі виконання програми байт-коди зчитуються і інтерпретуються віртуальною Java-машиною (JVM - Java Virtual Machine). Тобто, відкомпільована Java-програма може бути виконана на тих обчислювальних архітектурах, для яких реалізована JVM.

Необхідно відзначити, що динамічне зв'язування - це, в даний час, стандартний атрибут будь-якого обчислювального середовища (згадаємо ".so" в UNIX і ".dll" в Windows).

При підготовці та запуску Java-програм на виконання відсутній традиційний етап компонування - об'єднання коду програми з кодом всіх використовуваних її функцій (в C) і класів (в C ++). JVM здійснює завантаження необхідних класів динамічно в момент виникнення потреби в них.

З метою спрощення в Java немає конструкцій struct і union (з C), тому що вона об'єктно-орієнтована мова, і в ньому немає множинного спадкоємства і перевантаження операцій (з C ++). Найголовнішим є те, що в Java немає простих вказівників. Виключення з мови вказівників і арифметичних операцій над ними усунуло і цілий клас помилок, пов'язаних з неправильним використанням оперативної пам'яті.

Програми на Java, як на інтерпретованій мові, не можуть бути настільки економічні, як на компільованих C і C ++. Однак компіляція в байт-коди робить Java-програми більш швидкими, ніж програми на інших інтерпретованих мовах (JavaScript, PHP). Більш того, сучасні JVM мають здатність компілювати байт-коди в машинні команди цільової ЕОМ "на льоту", а це вже робить швидкість виконання Java-програм порівнянної зі швидкістю програм, написаних на C і C ++. Крім цього Java забезпечує можливість для критичних ділянок програми створювати код на C або C ++. Java-програма складається з одного або декількох класів, розміщених в одному або декількох файлах з розширенням .java. Для кожного класу з вихідного файлу створюється байт-код класу.

Автоматичний збір сміття (garbage collection) запобігає "витоку" пам'яті. Підвищенню стійкості програм на Java сприяє також використання конструкції try / catch / finally, що дозволяє згрупувати код, відповідальний за обробку винятків (exceptions), в тому числі помилок виконання, в одному місці програми.

Один з класів програми повинен бути відкритим (public) класом і містити метод main (), з якого починається виконання програми.

Java-програма завершує свою роботу інструкцією return в методі main або після виконання останньої інструкції методу main. Однак, багатопотокова програма закінчується із завершенням останнього потоку.

У Java підтримуються коментарі трьох видів:

1) У стилі мови C (від "/ *" до "* /").

2) У стилі мови C ++ (від "//" до "\ n").

3) Спеціальні коментарі "для документування" (від "/ **" до "* /").

В Java відсутній препроцесор, подібний препроцесору C ++. А значить і немає директив таких, як #define, #include, #ifdef. В Java немає необхідності в них.

Еквівалентом константи, оголошеної в C / C ++ через #define, служить змінна, оголошена в класі Java як static final, наприклад java.lang.Math.PI. Переваги такого підходу: сувора типізація констант і унікальність ієрархічних імен, що виключає конфлікти (також PI неможна перевизначити).

Макроси (інше використання #define) замінені inline-підстановкою, що здійснюється компілятором Java для "коротких" методів автоматично. Непотрібність #include пояснюється двома факторами. По-перше, файл класу (з розширенням .class) одночасно є і оголошенням класу, і його визначенням. Стандартизованість в розміщенні файлів класів по папках дає можливість інтепретатору Java однозначно визначати місце розташування завантаженого класу, значить відпадає необхідність директивного включення файлів вихідного коду.

Умовна компіляція (# ifdef / # endif в C ++) в Java виконується неявно. Справа в тому, що нормальний Java-компілятор (наприклад, javac) визначає ділянки вихідного тексту, які ніколи не будуть виконуватися, і ігнорує їх, не генеруючи для них байт-коду. Це означає, що конструкція C / C ++ у вигляді:

```
#ifdef DEBUG
... оцінний код ...
#endif
може бути смодельована в Java конструкцією:
private static final boolean DEBUG = true / false;
if (DEBUG) {
  ... оцінний код ...
  };
```

Так як DEBUG - константа, компілятор ще до етапу інтерпретації знає, буде код налагодження коли-небудь виконуватися чи ні.

Символи, рядки і ідентифікатори (тобто імена класів, змінних методів) в Java формуються 16-бітовими символами Unicode. Це забезпечує, наприклад, можливість давати класам і та їх членам російські імена. Для

подання у вихідних текстах Java символів, які не мають графічного представлення (наприклад, через відсутність відповідних шрифтів) використовується esc-послідовність \ uxxxx, де x - шістнадцяткова цифра. Подання в Unicode символів латинського алфавіту збігається з їх поданням до ASCII і ISO8859-1 (Latin-1). Але, на жаль, Unicode включає в себе кодування кирилиці, яка збігається зі стандартом ISO8859-5, який не користуються у нас популярністю (хоча всі UNIX-системи його підтримують).

1.3. Мова розмітки XML

XML - це розширювана мова розмітки (eXtensible Markup Language), це означає, що його можна використовувати для всього, що потрібно налаштовувати; можна створювати власні набори тегів для будь-яких виправданих цілей. У цієї мови є багато спільного з мовою HTML5, використовуваної для розробки сайтів, а в даний час і для дизайну додатків. Це мова розмітки, оскільки вона використовує "теги" для визначення того, що потрібно зробити. Мови розмітки відрізняються від мов програмування тим, що вони використовують теги і атрибути в цих тегах, а також вкладені структури для реалізації завдань, які мови програмування високого рівня, такі як Java, реалізують, використовуючи більш складні програмні структури, такі як масиви, цикли і методи. Використання XML для того, що орієнтовано на дизайн, особливо, інтерфейс користувача, досвід користувача, стилі і теми, графіку, тощо, звільняє членів групи розробки додатків, які розробляють дизайн додатки від того, щоб знати, чи навіть розуміти, як працює мова програмування Java. Тим більше, що вивчення розмітки XML значно легше, ніж освоєння програмних структур і концепцій мови.

XML розмітка міститься в текстових файлах з розширенням .xml. XML-файли можна створювати в текстовому редакторі, наприклад, в Блокноті; однак, більшість програмістів використовують інструменти

розробки програмного забезпечення з можливістю дизайну на мові розмітки, такі як Eclipse, IntelliJ, або NetBeans. Створені XML- файли потім зчитуються або аналізуються (parsed) ОС Android або кодом програми Java і перетворюються в структури об'єктів Java з використанням "визначень" XML в кожному файлі.

XML складається з тегів і їх атрибутів. XML-теги і атрибути, які можна використовувати з конкретним фреймворком, наприклад, при розробці для Android, задаються за допомогою схеми іменування XML. Така схема для Android міститься в централізованому репозитарії на серверах Android компанії Google. Атрибути є частиною тегів і використовуються для конфігурації і тонкого налаштування функцій тегів, а також для посилань до ресурсів медіа, шрифтів, стилям, значенням кольорів, тем, іншим XML-визначенням і аналогічним ресурсам, які можуть знадобитися, щоб відформатувати або визначити, як елемент додатку відображається користувачу на його екрані.

XML повинна мати схеми іменування, тому, що ця мова спочатку проектувалася як "розширювана". Це означає, що немає "стандартної" версії XML; кожна версія налаштовується на деяку необхідну кінцевим користувачем реалізацію. У разі XML для Android, він був спеціально налаштований для розробки Android-додатків.

Нижче наводиться лістинг файлу activity_main.xml простого додатка:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

tools:context="com.example.gkvar.myapplication.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
```

```
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

На схему іменування XML є посилання зверху кожного файлу визначення XML, так що розмітка XML всередині цього файлу може перевірятися по її схемі іменування XML (налаштувальної специфікації), щоб переконатися, що все теги і атрибути є дійсними і «правильними (valid)» . Eclipse - ADT робить це не в "реальному часі", тому, щоб розробляти XML-розмітку, не потрібно активне підключення до Інтернету. Процес перевірки коректності тегів і атрибутів (на відповідність з визначеннями в схемі іменування XML) називається валідацією XML.

У будь-якій розширюваній мові розмітки URL-адреса схеми іменування повинна перебувати в першому (зовнішньому) "батьківському" тезі. Цей батьківський тег зазвичай містить "дочірні" теги, які "вкладені" всередині батьківського тега. Для більшої наочності вкладеність тегів зазвичай відзначається відступами відносно один одного.

ОС Android реально має дві схеми іменування XML, розміщених в двох різних репозитаріях. Перша схема - схема іменування XML для пакетів Android (apk) була спроектована для високорівневого XML, орієнтованого на дизайн, і знаходиться в schemas.android.com/apk. 95% часу розробки програми використовується саме ця схема. Друга схема - схема іменування для інструментарію Android (tools) спроектована для низькорівневого XML, орієнтованого на використання ОС, і розміщується в schemas.android.com/tools. Цей XML можна використовувати, наприклад, для оголошення об'єктів класу Context.

1.4. Компоненти додатку

Основними складовими Android програми є компоненти додатка. Не всі компоненти є фактичними точками входу для користувача, а деякі

компоненти залежать один від одного, але кожен з них існує в якості власного вікна – кожний з компонентів є унікальною складовою частиною, яка допомагає визначити загальну поведінку програми.

Існує чотири типи компонентів програми. Кожен тип слугує для визначеної мети та має яскраво виражений життєвий цикл, який визначає, як компонент створюється й завершується.

Типи компонентів додатків:

1) Activity

Являє собою один екран з призначеним для користувача інтерфейсом. Activity працюють разом і формують цілісний користувальницький інтерфейс в додатку, але кожен з них працює незалежно від інших. Таким чином, інша програма може почати будь-який з цих видів «Activity».

Діяльність реалізована як підклас «Activity».

2) Services (Служби).

Представляє собою компонент, працюючий у фоновому режимі, щоб виконувати тривалі операції або роботи для віддалених процесів. Користувальницький інтерфейс служби не забезпечують. Як приклад, служба може відтворювати музику у фоновому режимі, поки користувач виконує дії в іншому додатку, також може отримати дані по мережі, не блокуючи взаємодію користувача з активністю. Ще один компонент, наприклад, Activity, може запустити службу і працювати в звязці.

Сервіс реалізований як підклас служби, і ви можете дізнатися більше про це в керівництві Послуги розробників.

3) Content providers (Контент-провайдери).

Управляє загальним набором даних додатку. Можна зберігати дані у файловій системі, Інтернеті, базі даних «SQLite», або будь-якому місці, до якого додаток може отримати доступ.

Якщо контент-провайдер дозволяє, то через нього інші додатки можуть запитувати або навіть змінювати дані. Уявимо, що Android система забезпечує контент-провайдер, управляючій інформацією про контакти

користувача. Тобто, будь-яка програма з відповідними правами може запитувати частину контент-провайдера.

Також контент-провайдери дуже корисні для читання і запису даних, яка не розділяє, а є приватною для вашої програми.

Контент-провайдер реалізований як підклас «ContentProvider» і повинен реалізувати стандартні API-інтерфейси, вони дозволяють виконувати транзакції іншим програмам.

4) Broadcast receivers.

Широкомовний приймач це компонент, відповідаючій за загальносистемні широкомовні анонси. Додатки можуть ініціювати трансляції, наприклад, щоб інші програми знали про дані, які були завантажені в пристрій для використання. Кожен широкомовний приймач являє собою доручення у вигляді Intent-у. Широкомовні приймачі не відображають для користувача інтерфейс, але вони можуть створити повідомлення для попередження користувача, коли відбувається широкомовна подія. Зазвичай широкомовний приймач виступає в якості "шлюза" для інших компонентів і має за мету виконати найменше роботи. Широкомовний приймач виконаний у вигляді підкласу «BroadcastReceiver».

1.5. Ресурси додатку

Ресурси це одні з найважливіших компонентів додатку. Екстерналізація ресурсів - це завжди хороша практика, так як забезпечує узгодженість і запобігає дублюванню ресурсів, на які можна посилатися з декількох місць в межах додатку. Відділення коду і ресурсів також має інші переваги:

Вид і сприйняття пристроїв: так як платформа Android сильно фрагментована, не існує магічної конфігурації інтерфейсу користувача (UI), яка однаково б виглядала і сприймалася на всіх пристроях з Android. Зазвичай, макети екранів, а також зображення потрібно налаштовувати на

основі розміру екрану і його щільності. Екстерналізація ресурсів дозволяє додатку використовувати відповідний набір ресурсів, ґрунтуючись на специфікації пристрою.

Фреймворк Android надає повний набір інтерфейсів прикладного програмування (API) і структуру для організації ресурсів з тим, щоб було легко підібрати відповідні ресурси для додатка, грунтуючись на характеристиках середовища виконання.

Локалізація програми: додаток може мати альтернативні набори ресурсів для різних мов і він може перемикатися між ними під час виконання, грунтуючись на мові користувача. Це дає можливість підтримки в одному двійковому додатку декількох мов і регіональних параметрів.

Android групує ресурси по дев'ятьом основними категоріями на основі їх типу. Кожна з цих груп має окремі папки в папці ресурсів src / res. Після компіляції і побудови додатку, Android автоматично створює файл з класом Java ім'я_пакету_додатку. R (наприклад, com \ example \ alex \ addressbook.R), за допомогою якого здійснюється доступ до ресурсів з коду програми. У (табл. 1.1) перераховані ці дев'ять груп ресурсів, відповідні їм папки ресурсів і їх групи всередині класу констант R.

Таблиця 1.1 Групи ресурсів, їх папки, їх R-константи і префікси XML

Група ресурсів	**Підпапка**	**Посилання в коді Java**	**Посилання в XML**
Анімації властивостей (Property Animations)	animator	R.animator	@animator/<file>
Анімації перетворень (Tween animation)	anim	R.anim	@anim/<file>
Список кольорів станів	color	R.color	@color/<file>
Графіка(Drawables)	drawable	R.drawable	@drawable/<file>
Контейнери(Layouts)	layout	R.layout	@layout/<file>
Меню	menu	R.menu	@menu/<file>
Raw	raw	R.raw	
Прості значення	values	R.arrays	
		R.bool	@bool/<id>
		R.color	@color/<id>
		R.dimen	@dimen/<id>
		R.integer	@integer/<id>

Продовження таблиці 1.1

		R.string	@string/<id>
		R.style	@style/<id>
XML	xml	R.xml	

Клас R містить вкладені класи, які в свою чергу містять всі ідентифікатори (ID) ресурсів для всіх ресурсів проекту:

```
package com.example.addressbook;
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
        public static final int textview_border=0x7f020001;
    }
    public static final class id {
        public static final int addContactItem=0x7f070019;
        public static final int addressTextView=0x7f070006;
           …
    }
    public static final class layout {
        public static final int activity_main=0x7f030000;
        public static final int add_contact=0x7f030001;
        public static final int view_contact=0x7f030002;
    }
    public static final class menu {
        public static final int main_activity_menu=0x7f060000;
        public static final int view_contact_menu=0x7f060001;
    }
    public static final class string {
        public static final int address=0x7f040000;
        public static final int app_name=0x7f040001;
           …
    }
    public static final class style {
        public static final int ContactLabelTextView=0x7f050000;
        public static final int ContactTextView=0x7f050001;
    }
}
```

Перед використанням XML-сумісні структури, які задані кореневим і дочірніми тегами, а також усіма їх атрибутами в файлі XML, повинні трансформуються в Java-сумісні структури даних. В ОС Android це робиться за допомогою процесу, який в програмі на Java викликається за допомогою методу .inflate () або .findViewById () і який називається "inflation

(надуванням)". Java-сумісні структури даних, які створюються цим процесом з визначень XML, відомі як об'єкти.

Кожного разу, коли додається, змінюється або видаляється ресурс, перегенеріруется і клас R. Наприклад якщо додати файл зображення з ім'ям logo.png в папку res / drawable, Android Studio створює статичне поле з ім'ям logo в класі drawable, вкладеному в клас R. Мета класу R в проекті - це можливість звернення до ресурсу в коді Java додатку. Наприклад, можна звернутися до файлу зображення logo.png за допомогою R.drawable.logo.

1.6. Аналіз існуючих аналогів

Додаток «Интересный Киев» був взятий за основний аналог та прототип. Проведемо аналіз додатку для виявлення недоліків.

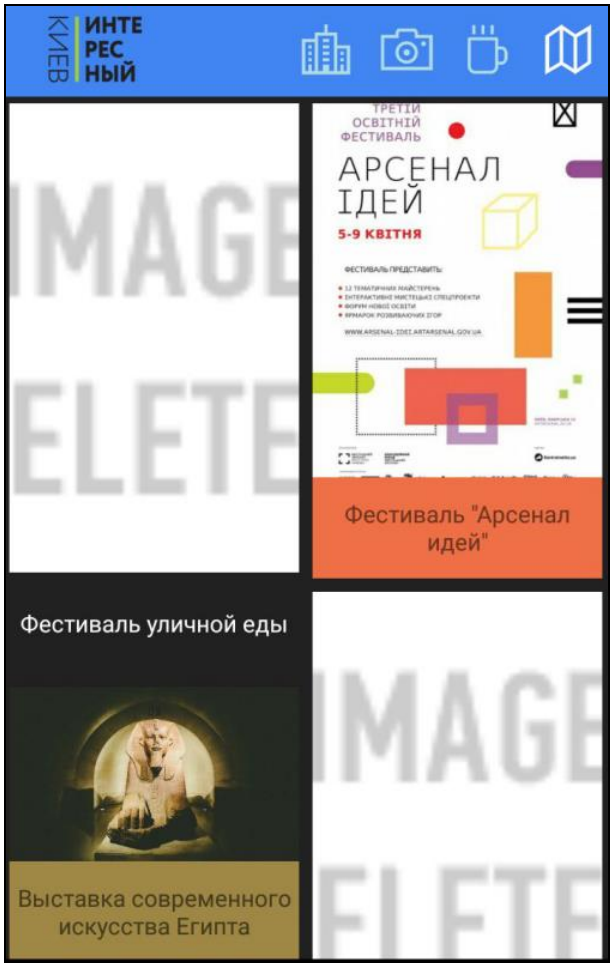

Рисунок 1.1 – Приклад головного меню додатка «Интересный Киев»

У цьому додатку можна подивитись на Google map позначки цікавих місць для туристів, також є карта 1913 року для того щоб подивитись як змінилось місто за роки. У іншому ж зроблено не дуже зручно, є вибір з трьох категорій: «Місто» - де можна натиснувши побачити список фестивалів, виставок, галерей, концертів, майстер класів та іншого, але там знаходиться дуже багато застарілих новин , було б краще якщо вже не актуальна інформація не відображалася, а відображалися тільки діючі заходи; «Пам’ятки» - музеї, парки, кіно, театр, фестивалі, виставки , тут також можна зустріти дуже багато новин по заходи, які вже відбулися, це тільки ускладнює пошуки діючого заходу який би зацікавив користувача; «Заклади де можна поїсти» - кафе, бари, ресторани… До кожного місця можна прокласти маршрут за допомогою Google maps.

Рисунок 1.2 – Приклад одного з багатьох застарілих заходів у додатку «Интересный Киев»

Також був розглянутий такий додаток як «Путеводитель по Одессе». В ньому можна подивитись на мапі Google позначки цікавих місць для туристів і прокласти маршрут до потрібного місця. Також є інформація о місті (історія міста та сучасність), ще є фотографії. У додатку зроблений вибір по категоріям: «О городе» - інформація про місто; «Места» - переводить на сторінку з категоріями (Достопримечательности, Памятники, Театры, Музеи), де можна обрати одну з категорій та подивитися список місць відповідних цій категорії, але неможливо звідси натиснувши по місцю на мапі прокласти до нього маршрут, бо не знаходить поточне місцезнаходження користувача, постійно йде загрузка (рис.1.4), а якщо натиснути на кнопку «Войти в Google» і увійти під своїм логіном переводить просто у браузер, або видає білий екран. Прокласти маршрут до місця виходить тільки з пошуку цікавих місць на мапі, за допомогою Google maps; «Share» - виводить на сторінку де пропонують поділитися з друзями додатком; «Календарь» - виводить на сторінку де показаний календар всіх діючих подій у місті; «Еще» - виводить сторінку з категоріями (Интересные места, Отдых, Досуг, Размещение, Еда…), де можна отримати ще більше інформації яка може знадобитися користувачу.

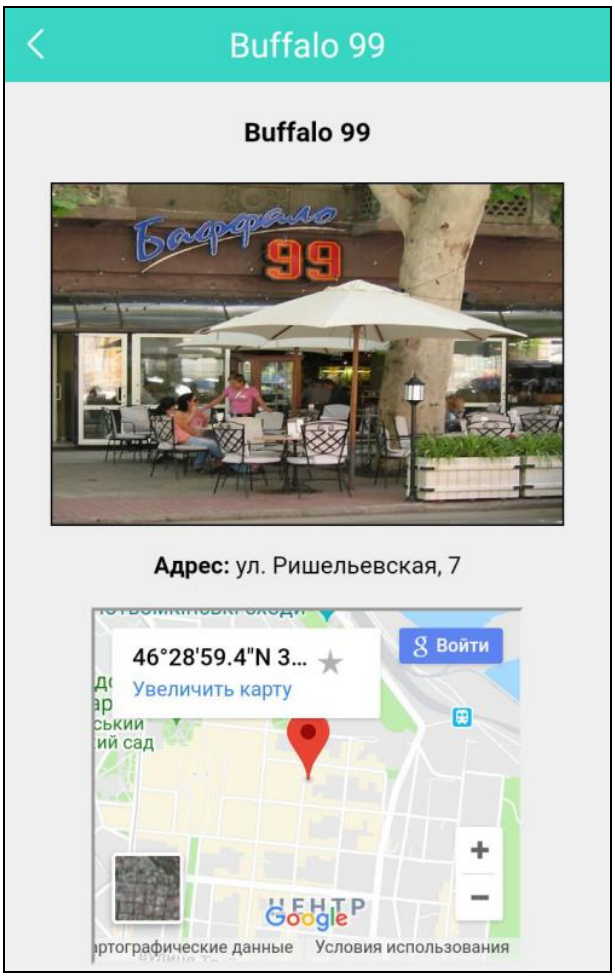

Рисунок 1.3 – Приклад одного з місць у додатку «Путеводитель по Одессе»

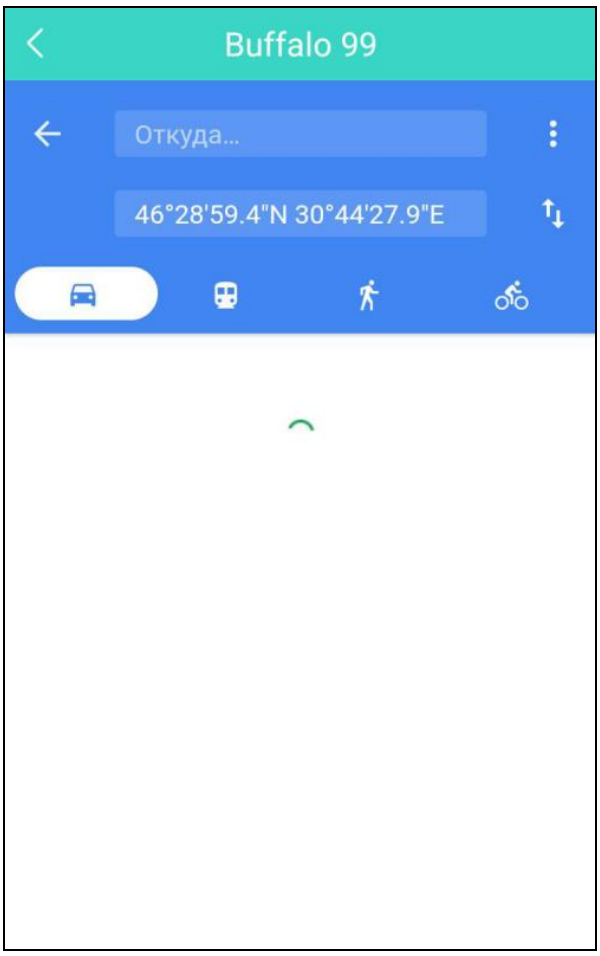

Рисунок 1.4 – Приклад прокладання маршруту до обраного закладу у додатку «Путеводитель по Одессе»

Проаналізувавши усі переваги та недоліки, було складено технічне завдання на розробку додатка на ОС Android з використанням мов XML, Java.

1.7. Технічне завдання на розробку додатка

ТЕХНІЧНЕ ЗАВДАННЯ

При розробці додатка для ОС Android з використанням мов XML, Java потрібно вирішити наступні завдання:

- проаналізувати предметну область та існуючі програмні засоби, які її описують;
- спроектувати додаток-навігатор для ОС Android з використанням мов XML, Java;

– розробити додаток-навігатор для ОС Android з використанням мов XML, Java;

Об'єкт проектування – додаток навігатор та гід по місту для ОС Android.

Предмет проектування – методи реалізації додатку навігатора та гіда по місту для ОС Android.

Додаток на ОС Android призначений для надання інформації людям для їх кращого орієнтування на вулицях міста, більше орієнтований на туристів і людей не місцевих:

– невеличка інформація про місто, фото, та погода у місті;
– поточне місцезнаходження користувача;
– інформація про оточуючі заклади;
– знаходження потрібного закладу за трьома основними категоріями;
– побудова маршруту.

У підрозділі "Цілі створення системи" приводять найменування і необхідні значення технічних, технологічних, виробничо-економічних або інших показників об'єкту автоматизації, які мають бути досягнуті в результаті створення додатку на ОС Android, і вказують критерії оцінки досягнення цілей створення системи.

Підвищення рівня орієнтування у місті з наявністю корисної детальної інформації для туристів і людей не місцевих.

Більш детальне побудування маршруту, зручне управління.

Значне скорочення часу пошуку потрібного закладу.

Для реалізації поставлених цілей система має вирішувати такі задачі:

– надання інформації про місто;
– надання інформації про поточне місцезнаходження користувача;
– надання інформації про оточення;
– знаходження потрібного заклади за трьома основними категоріями;
– побудова маршруту до потрібного закладу.

2. ПРОЕКТУВАННЯ ДОДАТКІВ ДЛЯ ОС ANDROID З ВИКОРИСТАННЯМ МОВ XML, JAVA

2.1. Вибір та обґрунтування засобів реалізації

Платформа розробки Android є багатошаровим набором програмних компонентів, які включають в себе ОС Linux з тисячами бібліотек функцій на Java, функціонуючі зверху ядра Linux. Java-бібліотеки значно спрощують задачу спілкування додатків з ядром Linux, яке є досить складним за своєю суттю.

Нагорі цього шару з коду Java-бібліотек, є шар вищого рівня на мові розмітки XML. XML розмітка забезпечує опис елементів більш високого рівня, таких як елементи дизайну користувальницького інтерфейсу (UI) додатку, тем, стилів, а також нових медіа-засобів. Ці засоби орієнтовані на дизайн дозволяють управляти тим, як Android-додаток виглядає, звучить і функціонує, та фіксувати свій користувальницький досвід. Розмітка XML дозволяє легко визначати структури об'єктів Java додатків, а також всі константи, які використовуються в програмній логіці Java.

Фундамент, на якому будується Android, - це акуратно закодована і ретельно протестована ОС Linux. Linux і її основні сервіси керують технічними засобами смартфонів, планшетів, читалок електронних книг, розумних годинників, інтерактивного TV (iTV), а також надають Android-додаткам повний доступ до функцій кожного пристрою споживчої електроніки, в тому числі до GPS, сенсорного екрану, камери, пам'яті, сховищ даних, Wi-Fi, Bluetooth, гіроскопа, компасу, акселерометру, NFC, порту USB і т.д.

Однак, Linux робить це все не сама. Android має широкий спектр бібліотек Java-API, які забезпечують функції і послуги більш високого рівня для таких речей, як управління базою даних SQLite, 2D-зображеннями, 3D-

рендерингом, анімацією, цифровим потоковим відео, відтворенням цифрового аудіо, Bluetooth, і багатьом іншим.

Також Android підтримує всі популярні медіа-формати з відкритим вихідним кодом, які можна використовувати в додатках, в тому числі потужний відеокодек VP8 компанії ON2, який зараз є в Android 4. ON2 була придбана Google, а VP8 був випущений під назвою WebM, який можна знайти в Android, а також в браузерах, таких як Firefox і Chrome.

Android платформа включає в себе велику кількість додатків "кінцевого споживача", які користувачі звикли бачити на Android-пристроях. Прикладами можуть служити утиліта управління телефоном, клієнт електронної пошти, клієнти різних платформ соціальних медіа, менеджер контактів, веб-браузер, Google Search, Google Maps, будильник, календар, медіа-плеєр, основні гри і т.д.

В розробці додатків використовується комбінація XML і Java, які виконуються на рівнях поверх ОС, слід зауважити, що забезпечується і прямий доступ до ОС з використанням низькорівневих мов, таких як C або C ++. В розробці таких програм використовується Android Native Development Kit (NDK), а не Software Development Kit (Android SDK), який використовується при звичайній розробці.

У середовищі розробки Android всі програми, створюються за допомогою комбінації мови Java і XML-розмітки. Eclipse і ADT компілюють їх в "байт-код", який призначений для виконання за допомогою спеціальної утиліти, яка входить до складу ОС Android. Eclipse (за допомогою SDK) розміщує цей код у файлі формату DEX (аналогу EXE-файлу).

Таким чином, схема компіляції і виконання додатків виглядає так:

XML (.xml) → Java (.java) → Байт-код (.dex) → Додаток (.apk) →

Середовище виконання (DVM) → Екран

Середовище виконання DVM (до Android 4.4 вона була єдиною середовищем виконання; в 4.4 можна вибирати між DVM і ART, а Android 5.0 і вище використовують тільки ART), надає Java-коду і XML-розмітки

додатків основний набір бібліотек операційної системи, які забезпечують доступ до технічних засобів пристроїв, а також низькорівневим функціям ОС Android.

Значними перевагами XML є інтегровані середовища розробки, які допомагають у будуванні XML-дерева. Є цілий ряд технологій перевірки коректності XML-коду. XSLT - спеціальна мова перетворення XML. Підтримка XML вбудована навіть безпосередньо в синтаксис деяких мов (як, наприклад, E4X в ActionScript).

2.2. Віртуальна машина Dalvik

ОС Android завжди використовувала для виконання Java-коду ОС Android машину часу виконання Dalvik VM (або DVM). DVM була створена спеціально для ОС Android і забезпечувала агресивну оптимізацію Java-коду, XML-розмітки і активних ресурсів нових медіа так, що вони могли бути використані в малопотужних, мобільних, однопроцесорних та інших подібних вбудованих середовищах. Вбудоване середовище - це середовище, в якому малопотужний комп'ютер (який може працювати на батареї) "вбудований" всередині виробу побутової електроніки. Це є звичайною справою в смартфонах, планшетах, розумному годиннику, розумних окулярах, читалках електронних книг, телевізорах iTV, плеєрах iDVD.

Коли користувач вперше запускає додаток Android, натиснувши на значок запуску, DVM, використовуючи файл DEX, створює виконувану двійкову версію програми. Це називається JIT-виконанням програмного забезпечення. Виконуваний двійковий код (нулі і одиниці) будуть сумісними з конкретним типом процесора, який використовується в конкретному пристрої з Android. Потім Android створює процес, якому потім виділяється пам'ять і ресурси CPU (кванти часу процесора), так що він має необхідні ресурси для виконання своїх різноманітних функцій. Кожен раз, коли додаток запускається і породжується процес для нього, створюється

екземпляр DVM для цього процесу, в рамках якого для додатка встановлюється або резервується простір пам'яті Android-пристрою.

Використання DVM дає можливість виконувати більше додатків в обмеженому просторі пам'яті (від 1 до 2 Гб) і з більш повільним (від 1 до 2 ГГц) процесором, які є в сьогоднішніх пристроях споживчої електроніки. DVM також захищає всі запущені процеси від втручань один в одного. Таким чином, неправильна робота одного додатка не порушить роботу всієї операційної системи Android.

DVM реалізує інструкції мови Java разом з директивами дизайну користувальницького інтерфейсу додатку (у вигляді об'єктів Java), які визначаються за допомогою XML-розмітки, комбінує їх з зовнішніми медіа-активами (відео, аудіо, зображеннями, 3D і т.д.), і переводить цю інфраструктуру в оптимізований, низькорівневий, двійковий код. Ці низькорівневі інструкції поміщаються в пам'ять пристрою Android і представляються процесу ОС або менеджеру потоків, який потім витягує ці низькорівневі інструкції з пам'яті і виконує їх за допомогою CPU (процесора), тим самим виконуючи Android-додаток.

Ще однією важливою перевагою середовища виконання DVM є те, що не потрібно створювати іншу версію програми для кожного типу процесора і виробника на ринку. Як відомо, Intel, AMD, і nVidia змагаються в мобільному просторі, також як процесор ARM і процесор Cortex серед інших. Маючи єдину версію байт-коду Java в DEX-файлі і середу виконання всередині ОС Android, яка перетворює його в двійковий код на машинній мові, сумісному з архітектурою процесора даного Android-пристрою, процес розробки для Android спрощується на порядок.

2.3. Віртуальна машина ART (Android RunTime)

До цих пір в ОС Android під час виконання за замовчуванням використовується DVM, і що ART, введена, починаючи з Android 4.4,

автоматично не активується в якості первинної утиліти диспетчеризації виконання. Це тому, що ART є новою і до кінця неперевіреної, в той час як DVM витримала випробування часом і працює адекватно, особливо на пристроях з швидкими процесорами і великий пам'яттю. Починаючи з Android 5.0, ART буде основним засобом виконання за замовчуванням. ART - це також віртуальна машина і також утиліта виконання для ОС, як і DVM. Головна відмінність полягає в тому, що DVM використовує модель JIT, а ART використовує модель AOT (Ahead of Time - завчасно). У моделі AOT двійкове подання на машинній мові створюється, коли користувач встановлює Android-додаток, а не при кожному запуску програми.

Перевагою цього підходу, є, звичайно, малий час запуску, що люблять всі користувачі, а недоліком є більший час завантаження і установки. Також недоліком є те, що потрібно місце для зберігання цього двійкового файлу на машинній мові. Хоча під час установки додатків, щоб створити виконавчий модуль на двійковій машинній мові для кожного встановлюваного додатку, буде використано більше часу процесора і пам'яті (буде потрібно більше енергії акумулятора), коли користувач встановить всі додатки, то під час фактичного використання Android пристрою день у день буде використовуватися менше енергії акумулятора, пам'яті і часу процесора.

2.4. Теми і стилі

Для того щоб додаток мав єдиний стиль оформлення, можна скористатися спеціальною темою (колекція стилів, які забезпечують професійний вид додатку).

Для завдання теми досить вказати ім'я теми в маніфесті:

```
<Activity android: name = ". About"
     android: label = "@ string / about_title"
     android: theme = "@ android: style / Theme.Dialog">
</ Activity>
```

Префікс @android: перед ім'ям теми або стилю означає, що використовується вбудована тема або стиль системи Android. Стиль - це один або кілька згрупованих атрибутів форматування, які відповідають за зовнішній вигляд і поведінку візуальних елементів або вікон. Стиль може задавати такі властивості, як ширину, відступи, колір тексту, розмір шрифту, колір фону і так далі. Самі стилі зберігаються в XML-файлах, окремо від файлів розмітки.

Уявимо, що в файлі розмітки активності є такі рядки:

```
<TextView
    android: layout_width = "fill_parent"
    android: layout_height = "wrap_content"
    android: textColor = "# 00FF00"
    android: typeface = "monospace"
    android: textSize = "18sp"
    android: text = "@ string / hello" />
```

Можна винести деякі властивості елемента TextView в файл стилів res / values / styles.xml наступним чином:

```
<? Xml version = "1.0" encoding = "utf-8"?>
<Resources>
    <Style name = "MyStyle" parent = "@ style / Text">
        <Item name = "android: textSize"> 18sp </ item>
        <Item name = "android: textColor"> # 00FF00 </ item>
        <Item name = "android: typeface"> monospace </ item>
    </ Style>
</ Resources>
```

Тоді в файлі розмітки активності опис TextView буде виглядати так:

```
<TextView
    style = "@ style / MyStyle"
...
    android: text = "@ string / hello" />
```

Стильовий XML-файл завжди розміщується в папці res / values / проекту. Файл не має значення, головне, щоб розширення було .xml, а сам файл знаходився у зазначеній папці. У новий проект, який створюється Android Studio, автоматично включається готовий файл стилів res / values / styles.xml, в який можна додавати нові стилі, редагувати або видаляти вже наявні. Також можна створювати свої файли стилів. Кореневим вузлом файлу повинен бути елемент <resources>. Для кожного стилю, що включається в файл, потрібно додати елемент <style> з унікальним ім'ям. Далі для кожної

властивості створюються елементи <item> з іменами відповідних властивостей. Значенням елемента <item> має бути ключове слово, колір у шістнадцятковому представленні, посилання на інший тип ресурсів або інше значення в залежності від властивості стилю.

Приклад стилю:

```
<? Xml version = "1.0" encoding = "utf-8"?>
<Resources>
    <Style name = "MyStyle"
parent = "@ android: style / TextAppearance.Medium">
        <Item name = "android: layout_width"> fill_parent </
item>
        <Item name = "android: layout_height"> wrap_content
</ item>
        <Item name = "android: textColor"> # 00FF00 </ item>
        <Item name = "android: typeface"> monospace </ item>
    </ Style>
</ Resources>
```

Під час компіляції всі властивості з файлу стилів витягуються і застосовуються до візуальних елементів. Атрибут parent для елемента style є необов'язковим і дозволяє задавати ідентифікатор ресурсу іншого стилю, з якого успадковуються властивості.

Тема - це більш ємне поняття, ніж стиль. По суті, тема – це стиль, який відноситься до всього екрану активності або додатку, а не до окремого компоненту додатка. Теми схожі на визначення стилів. Точно так, як стилі, теми оголошуються в XML-файлі елементами <style>, і посилаються на них тим же самим способом. Різниця полягає в тому, що тема додається до всього додатку або до окремої активності через елементи <application> і <activity> в файлі маніфесту додатка, так як теми не можуть бути застосовані до окремих компонентів.

Щоб встановити тему, потрібно відкрити файл AndroidManifest.xml і відредагувати тег <application>, щоб він включав в себе атрибут android: theme із зазначенням імені стилю:

```
<Application android: theme = "@ style / CustomTheme">
```

Якщо потрібно, щоб тема відносилася не до всього додатку, а до окремої активності, то атрибут android: theme потрібно додати в тег <activity>.

У багатьох випадках немає необхідності придумувати свої стилі і теми, так як Android містить безліч власних вбудованих тем. Наприклад, можна використовувати тему Dialog, щоб вікно керування виглядало як діалогове вікно:

```
    <Activity android: theme = "@ android: style / Theme.Dialog">
    Для прозорого екрану можна використовувати тему Translucent:
    <Activity android: theme = "@ android: style / Theme.Translucent">
```

Якщо тема подобається, але кілька властивостей все-таки хочеться змінити, то потрібно просто додати тему як батьківську тему до своєї теми. Наприклад, хочеться модифікувати стандартну тему Theme_Light, щоб використовувати свої кольори:

```
    <Color name = "custom_theme_color"> # b0b0ff </ color>
    <Style name = "CustomTheme" parent = "android: Theme.Light">
        <item
    name = "android: windowBackground"> @ color / custom_theme_color
    </ Item>
        <item
    name = "android: colorBackground"> @ color / custom_theme_color
    </ Item>
    </ Style>
```

Тепер в маніфесті можна використовувати свій стиль замість Theme.Light:

```
    <Activity android: theme = "@ style / CustomTheme">
```

Список властивостей, які можуть використовуватися для налаштування власних тем:

android: windowNoTitle - використовується значення true, щоб приховати заголовок.

android: windowFullscreen - використовується значення true, щоб приховати рядок стану і звільнити місце для докладання.

android: windowBackground - ресурс кольору або drawable для фону.

android: windowContentOverlay - Drawable, який малюється поверх вмісту вікна. За замовчуванням, це тінь від рядка стану. Можна використовувати null (@null в XML-файлі) для видалення ресурсу.

3. РЕАЛІЗАЦІЯ ДОДАТКУ НАВІГАТОРА ТА ГІДА ПО МІСТУ ДЛЯ ОС ANDROID З ВИКОРИСТАННЯМ МОВ XML, JAVA

3.1. Проектування архітектури програмного продукту

Додаток для ОС Android складається з набору активностей, кожній з яких відповідає один екран додатка. Кожна активність представлена в проекті класом, реалізованому на мові Java, що зберігається в однойменному файлі з розширенням .java. Кожній активності відповідає xml-файл з її описом. В xml-файлі описано у вигляді xml-коду розташування об'єктів візуалізації. При запуску активності система Android автоматично розпізнає розмір екрану мобільного пристрою і призводить виведений контент у відповідність з розміткою, описаною в xml-файлі. Таким чином, одна і та ж активність буде виглядати однаково незалежно від діагоналі використовуваного пристрою. Також, для кожного додатка Android повинен існувати xml-файл, в якому у вигляді xml-коду будуть прописані мінімальні вимоги до системи, а також активність, яка викликається при запуску додатка.

Додаток працює з базою даних, до якої дає доступ ключ аутентифікації Google API key, що дає змогу отримати інформацію напряму з Google за допомогою API та сервісів таких як: Google SDK Android API, Google+ API, Places API, Places SDK for Android. Завдяки Google API key буде підключена вся необхідна інформація для мапи, і буде відображена інформація про місця на мапі. Такий підхід зменшує час відгуку і значно спрощує роботу розробникам.

Дизайн має бути зручним та простим і в першу чергу орієнтуватися на користувача, який без особливих зусиль зможе знайти і скористатися будь якою інформацією, що міститься у додатку.

Хороша зручна навігація додатку означає, що користувачі точно знатимуть, де знаходяться, і де розташовані елементи додатка і як

використовувати ці елементи. Правильна структура інформації дозволяє користувачам без проблем продовжувати користування додатком, і бути впевненими в тому, що вони завжди зможуть без зусиль повернутися до раніше переглянутої інформації. Якщо навігація буде не зручною, то значний процент користувачів це може відштовхнути, вони або заплутаються у функціях, або просто не зможуть знайти потрібну їм інформацію навіть якщо вона там була.

3.2. Розробка коду розмітки strings.xml

```
    <resources>
        <string name="app_name">Sev Guide</string>
        <string name="action_settings">Settings</string>
        <string name="rating_count"> (%s REVIEWS)</string>
        <string   formatted="false"   name="image_count">%s   of
%s</string>
        <string name="search_in">Search in</string>
        <string                  name="weather_forecast">Weather
Forecast</string>
        <string name="about">About</string>
        <string name="home_sleep">1</string>
        <string name="home_eat">2</string>
        <string name="home_enjoy">3</string>
        <string name="around_me">AROUND ME ?</string>
        <string name="home_menu_favs">Favorite Places</string>
        <string name="home_menu_rate">Leave Feedback</string>
        <string              name="list_label_price_level">Price
level</string>
        <string name="list_label_rating">Rating</string>
        <string     name="place_detail_photos_counter">1     of
4</string>
        <string
name="place_detail_label_address">ADDRESS</string>
        <string
name="place_detail_label_phone">PHONE(s)</string>
        <string name="place_detail_label_tag">TAG(s)</string>
        <string      name="place_detail_label_price_level">PRICE
LEVEL</string>
        <string name="place_detail_label_no_value">-/-</string>
        <string name="place_detail_label_rating">RATING</string>
        <string name="no_internet_connection">Please connect to
the network</string>
        <string name="no_weather">Weather Unavailable</string>
        <string   name="town_range">Your   are   outside   of
%s</string>
        <string    name="current_location_issue">Cannot    find
current location</string>
```

```
        <string
name="google_api_key">AIzaSyARNr__qjtOm7F_2jOavQkZwIi-
Ztcn804</string>
        <string name="your_are_here">You are here</string>
        <string   name="turn_on_gps">Please   turn   on   your
GPS</string>
    </resources>
```

У коді розмітки strings.xml прописані ресурси як для головного меню додатка, такі як: «Search in», «About», «Weather», «AROUND ME»; так і ресурси для сторінок з інформацією про заклади, такі як «ADDRESS», «PHONE(s)», «TAG(s)», «PRICE LEVEL», «RATING» і т.д.

Також тут прописаний «google_api_key» який включає в себе такі API та сервіси як: Google SDK Android API, Google+ API, Places API, Places SDK for Android. Завдяки цьому API ключу буде працювати мапа, і буде відображена вся необхідна нам інформація.

3.3. Розробка коду розмітки AndroidManifest.xml

```
    <?xml version="1.0" encoding="utf-8"?>
    <manifest
        package="com.aleksey.navigator"

xmlns:android="http://schemas.android.com/apk/res/android">
        <uses-permission
android:name="android.permission.INTERNET"/>
        <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>
        <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
        <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
        <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
        <uses-feature
            android:glEsVersion="0x00020000"
            android:required="true"/>
        <application
            android:name=".SevGuide"
            android:allowBackup="true"
            android:icon="@drawable/ic_launcher"
            android:label="@string/app_name"
            android:theme="@style/AppTheme">
            <meta-data
                android:name="com.google.android.gms.version"
```

```
android:value="@integer/google_play_services_version"/>
            <meta-data
                android:name="com.google.android.geo.API_KEY"
                android:value="@string/google_api_key"/>
            <activity
                android:name=".MainActivity"
                android:label="@string/app_name"
                android:screenOrientation="portrait"
                >
                <intent-filter>
                    <action
android:name="android.intent.action.MAIN"/>
                    <category
android:name="android.intent.category.LAUNCHER"/>
                </intent-filter>
            </activity>
        </application>
    </manifest>
```

«xmlns:android=="http://schemas.android.com/apk/res/android"> визначає простір імен Android. Після цього йде перелік дозволів користувача «uses-permission» на які користувачу буде запропоновано погодитись. Далі йде «uses-feature», дозволяє додатку описувати функції змінних пристроїв, які вона використовує. Нижче йде «application» де прописано назву додатка, ярлик додатку, мітку, та тему. Далі йде «meta-data», це вкаже версію служб Google Play, з якої скомпільовано додаток. Наступним йде активність «activity android:name=".MainActivity"». Далі йде фільтр «intent-filter», за допомогою якого система передасть неявний об'єкт Intent додатку, тільки якщо він може пройти через один з ваших фільтрів Intent.

3.4. Розробка програмного коду AppSettings.java

```
package com.aleksey.navigator.settings;
/**
 * Створив alekseyfedoryachenko
 */
public class AppSettings {
    /**
     * Широта вашого місцезнаходження
     */
    public static double LATITUDE = 48.94763373;
    /**
     * Довгота вашого місцезнаходження
```

```
         */
        public static double LONGITUDE = 38.48982682;
        /**
         * Масштаб мапи
         */
        public static final int MAP_INITIAL_ZOOM = 14;
        /**
         *  Радіус в метрах для відображення місцезнаходження
об'єктів
         */
        public static final int GOOGLE_PLACES_LOCATION_RADIUS =
1000;
        /**
         * Радіус в метрах для пошуку об'єктів
         */
        public static final int GOOGLE_PLACES_SEARCH_RADIUS =
10000;
        /**
         * Місто для додатку
         */
        public static final String TOWN = "Severodonetsk";
        /**
         * Країна для додатку
         */
        public static final String COUNTRY = "Ukraine";
        /**
         * Ключ для відображення погоди
         */
        public static final String OPEN_WEATHER_MAP_KEY =
"341950b2f7df14440f419e5c2d9e9b25";
        /**
         * Шлях до ресурсу розмітки
         */
        public static final String XMLResourcePath =
"about.xml";
    }
```

У програмному коді AppSettings.java для відображення на карті міста Сєвєродонецьк була задана його широта «LATITUDE» та довгота «LONGITUDE». Також був заданий масштаб мапи «MAP_INITIAL_ZOOM» для зручного користування.

Далі був заданий радіус в метрах для відображення об'єктів – 1000м «GOOGLE_PLACES_LOCATION_RADIUS».

Також був задано радіус в метрах для пошуку об'єктів – 10000м «GOOGLE_PLACES_SEARCH_RADIUS».

Наступним шагом було введено назву міста та країни для відображення (public static final String «TOWN», «COUNTRY»). Та був введений ключ OPEN_WEATHER_MAP_KEY для відображення погоди у місті. Також вказаний шлях до ресурсу розмітки XMLResourcePath = "about.xml", у якому міститься невелика інформація про місто, та декілька фотографій міста.

3.5. Опис функціональності

Додаток для ОС Andriod являє собою помічника та навігатор гід по місту Сєвєродонецьк, який допоможе знайти потрібний заклад користувачу. Додаток зорієнтований на туристів і людей не місцевих, їм будуть представлені корисні для них функції, які допоможуть ознайомитись з основними закладами які їм знадобляться. У додатку запропонований вибір з трьох категорій: «HOTEL» - побачити готелі де можна заночувати і інформацію про них, «EAT» - побачити закусочні, кафе та ресторани і інформацію про них, «ENJOY» - побачити заклади де можна розважитись та гарно провести час і інформацію про них. «AROUND ME?» - подивитися що знаходиться навколо користувача. Ще можна отримати невеличку інформацію про місто та подивитись фотографії – «About», і погоду у місті – «Weather». Також можна увести у пошук конкретний заклад і знайти його та отримати по ньому інформацію. При виборі потрібного закладу можна прокласти до нього маршрут за допомогою Google maps.

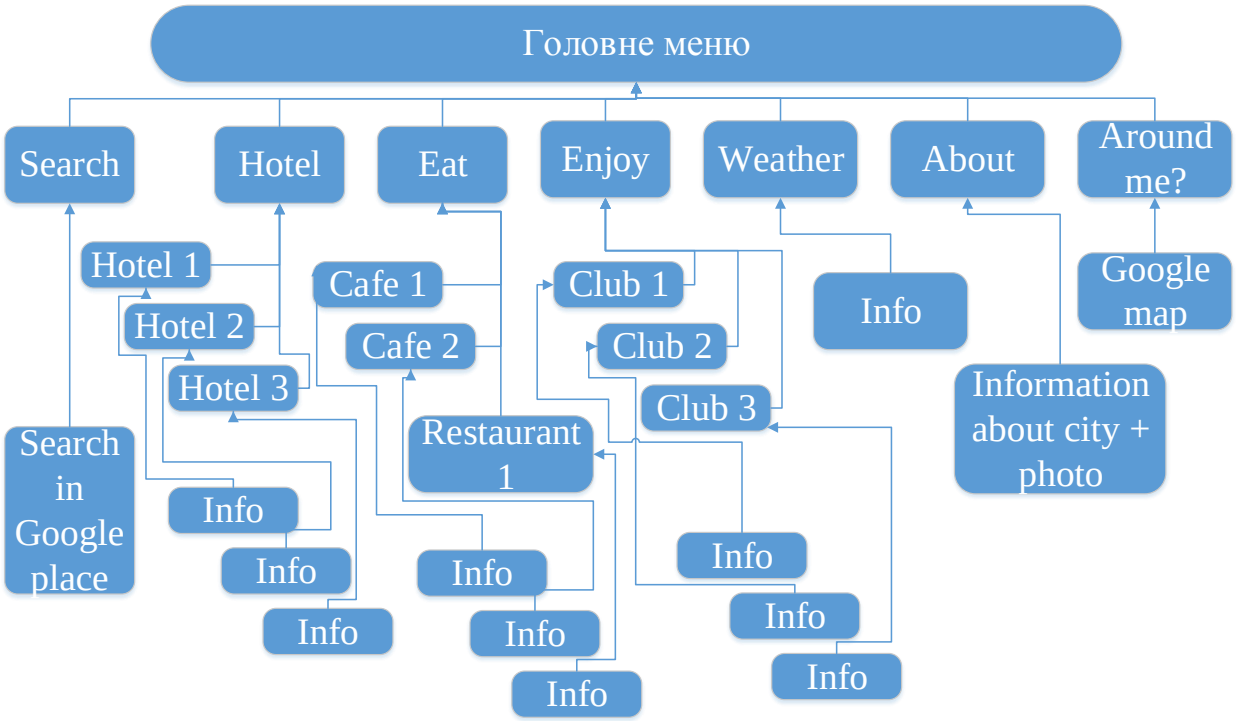

Рисунок 3.1 – Структура додатку на ОС Android

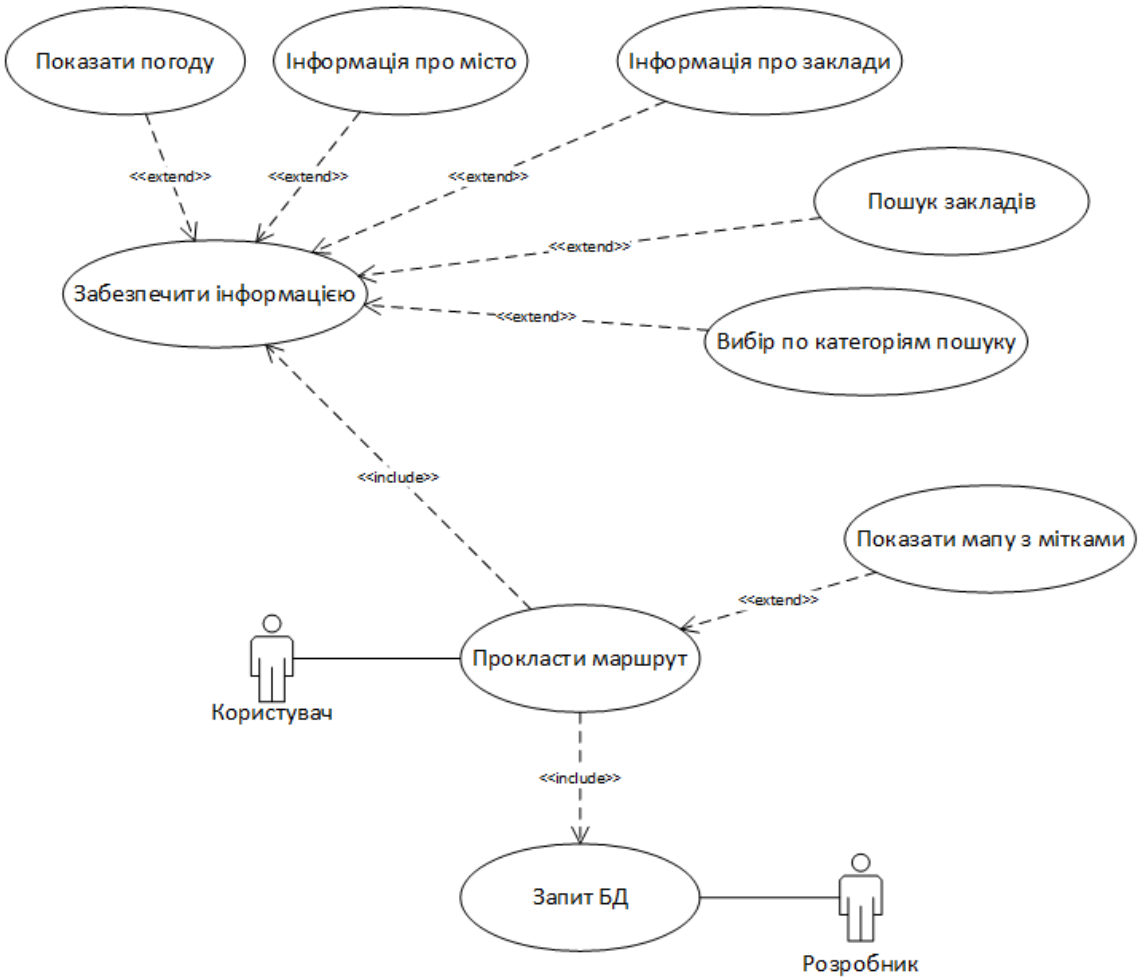

Рисунок 3.2 – Схема обслуговування користувача додатка

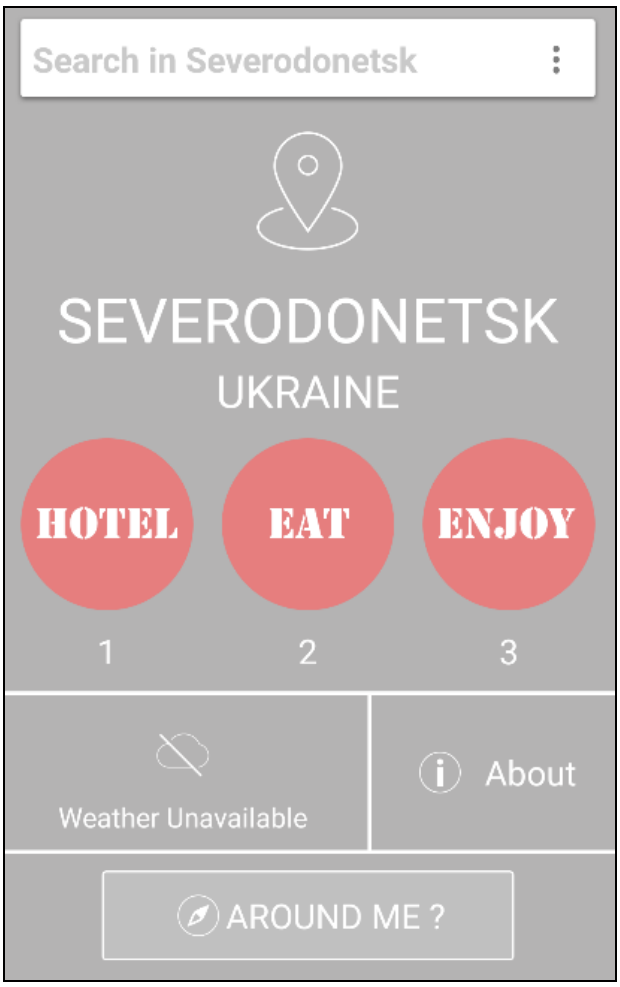

Рисунок 3.3 – Головне меню додатка

Зверху знаходиться пошук, якщо увести туди назву закладу що хочете знайти, будуть показані заклади що відповідають уведеному. Також є три кнопки категорії: «HOTEL», «EAT», «ENJOY». При натисканні на «HOTEL» - буде переведено на сторінку зі списком закладів де можна заночувати і інформацію про кожний з них. Якщо натиснути на «EAT» - буде переведено на сторінку зі списком закусочних, кафе та ресторанів і інформацію про кожне з них. Якщо натиснути на «ENJOY» - буде переведено на сторінку зі списком закладів де можна розважитись та гарно провести час і інформацію про кожне з них.

Нижче знаходяться «Weather» та «About». Якщо натиснути «Weather», то буде переведено на сторінку на якій показаний прогноз погоди у місті, а при натисканні на «About» буде переведено на сторінку на якій показана невелика інформація про місто та фотографії міста.

В самому низу знаходиться кнопка «AROUND ME?» при натисканні на яку відкривається мапа і показує оточуючі користувача об'єкти.

Рисунок 3.4 – Сторінка з інформацією про місто та фотографіями, яка з'являється після натискання кнопки «About»

На цій сторінці користувач може ознайомитися з невеликою інформацією про місто та подивитися фотографії, кожну з яких можна розгорнути натисканням на неї.

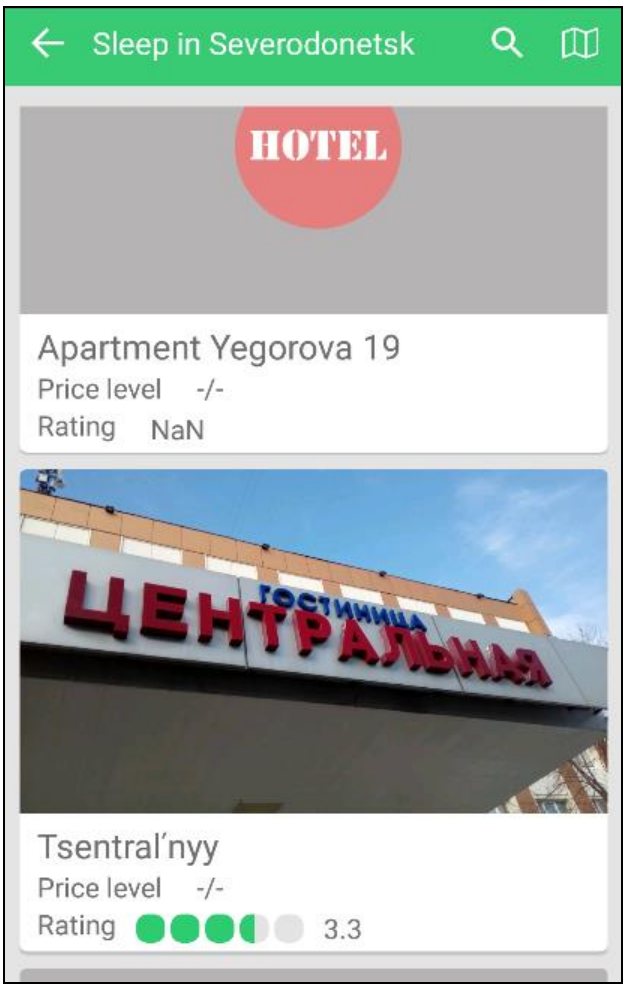

Рисунок 3.5 – Сторінка зі списком закладів де можна заночувати, яка з'являється після натискання кнопки «HOTEL»

На цій сторінці користувачу буде показаний список готелів, з рейтингами. Зверху знаходиться пошук де можна одразу увести і знайти заклад який вас цікавить, а не шукати у списку. Поряд знаходиться кнопка мапи, після натискання на неї користувача буде переведено на мапу Google з мітками на ній всіх готелів зі списку, та пошуком зверху.

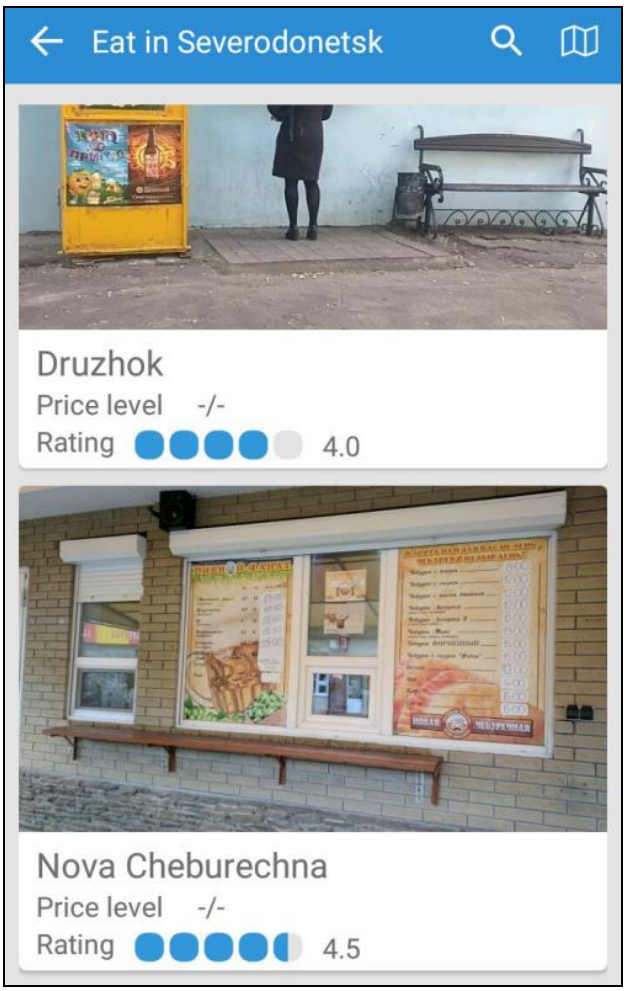

Рисунок 3.6 – Сторінка зі списком закладів де можна поїсти, яка з'являється після натискання кнопки «EAT»

На цій сторінці користувачу буде показаний список закладів де можна поїсти, з рейтингами. Зверху знаходиться пошук де можна одразу увести і знайти заклад який вас цікавить, а не шукати у списку. Поряд знаходиться кнопка мапи, після натискання на неї користувача буде переведено на мапу Google з мітками на ній всіх закладів де можна поїсти зі списку, та пошуком зверху.

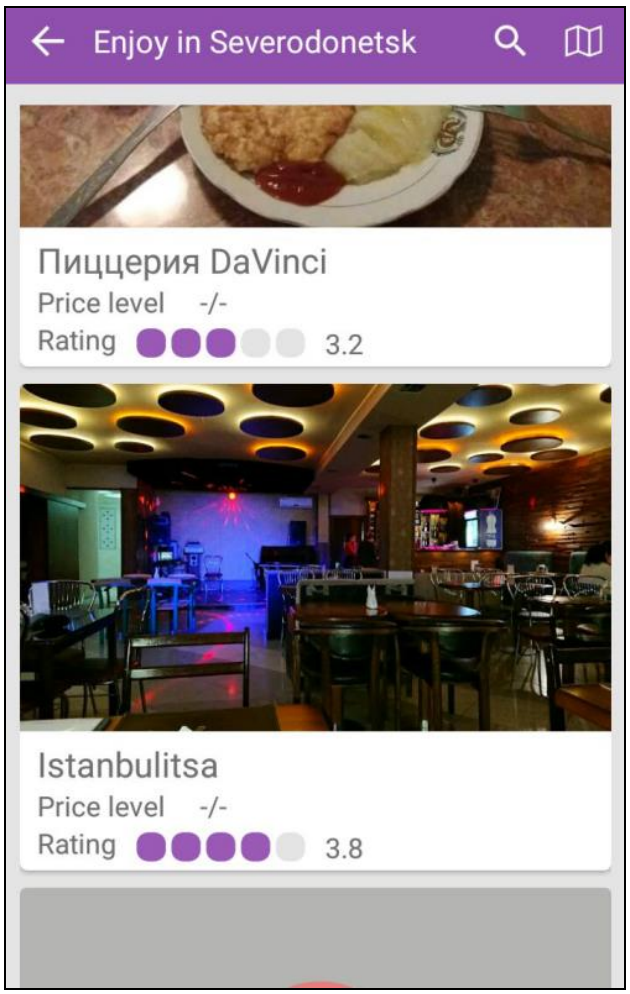

Рисунок 3.7 – Сторінка зі списком закладів де можна розважитись, яка з'являється після натискання кнопки «ENJOY»

На цій сторінці користувачу буде показаний список закладів де можна розважитись, з рейтингами. Зверху знаходиться пошук де можна одразу увести і знайти заклад який вас цікавить, а не шукати у списку. Поряд знаходиться кнопка мапи, після натискання на неї користувача буде переведено на мапу Google з мітками на ній всіх закладів де можна розважитись зі списку, та пошуком зверху.

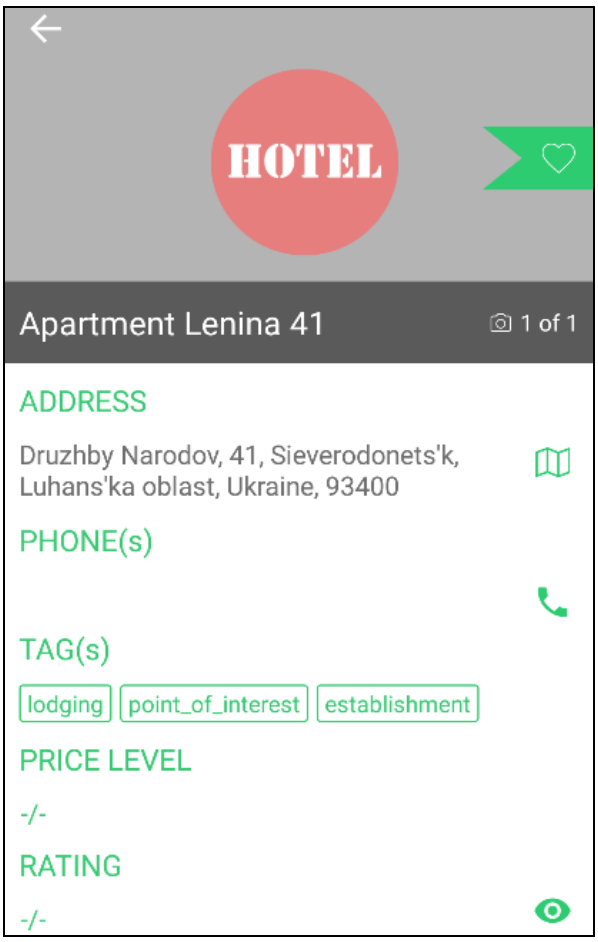

Рисунок 3.8 – Сторінка з інформацією про готель відкритою зі списку готелів

Тут можна подивитися фотографії якщо вони є, також можна зберігати заклади у обране. Нижче є точна адреса закладу поряд з якою кнопка мапи, при натисканні на яку користувача одразу буде направлено на мапу Google де розташований саме цей заклад. Ще нижче знаходиться телефон, та кнопка при натисканні на яку користувач зможе обрати яким способом зв'язатися з цим закладом: по телефону, Viber, WhatsApp, Skype та ін. Також є мітки, рівень цін, та рейтинг.

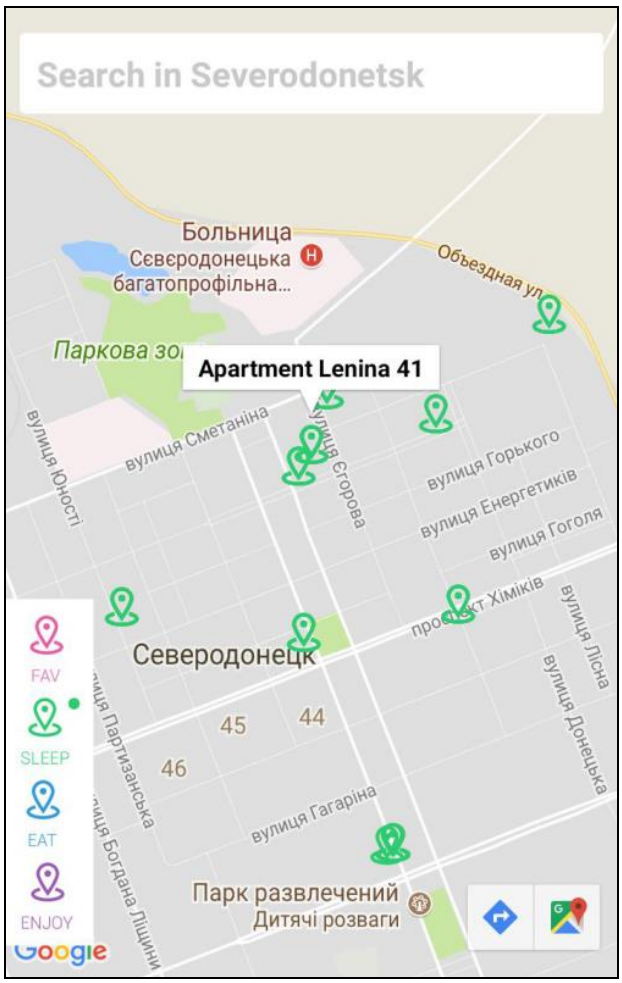

Рисунок 3.9 – Сторінка з мапою Google з мітками на ній всіх готелів зі списку

На мапі Google можна вибрати що відобразити або навпаки приховати на карті натиснувши «FAV», «SLEEP», «EAT», «ENJOY», всі категорії закладів відображаються на мапі різними кольорами для зручності. До потрібного закладу можна прокласти маршрут за допомогою Google maps, для цього потрібно обрати заклад, та натиснути на відповідну кнопку знизу справа, і маршрут буде побудовано.

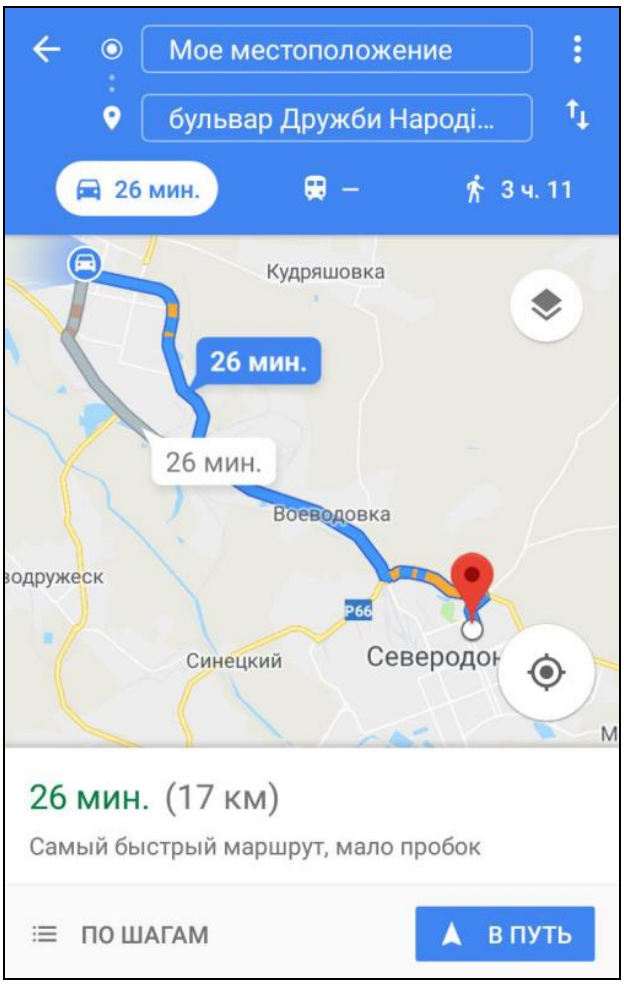

Рисунок 3.10 – Прокладений маршрут до обраного готелю

Після вибору потрібного закладу, по натисканню кнопки знизу справа для складання маршруту до нього, за допомогою Google maps маршрут був побудований.

4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

Завданням даної роботи бакалавра було розробка додатку для ОС Android з використанням мов XML та Java, і як результат був створений додаток навігатор та гід по місту. За цим додатком в подальшому розроблятиметься реальна система, яка значно полегшить процес пошуку потрібного місця туристам та людям не місцевим. Так як в процесі проектування використовувався ПК, то аналіз потенційно небезпечних і шкідливих виробничих чинників виконується для персонального комп'ютера на якому буде розроблений додаток.

4.1. Загальні питання з охорони праці

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. В законі України «Про охорону праці» визначається, що охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

При роботі з обчислювальною технікою змінюються фізичні і хімічні фактори

навколишнього середовища: виникає статична електрика, електромагнітне випромінювання, змінюється температура і вологість, рівень вміст кисню і озону в повітрі.

4.1.1. Правові та організаційні основи охорони праці

Основним організаційним напрямом у здійсненні управління в сфері охорони праці є усвідомлення пріоритету безпеки праці і підвищення соціальної відповідальності держави, і особистої відповідальності працівників.

Обов'язки працівників щодо додержання вимог нормативно-правових актів з охорони праці, відповідальність робітників всіх категорій за порушення вимог щодо охорони праці та структура організації/виробництв системи управління охорони праці визначені безпосередньо у [9].

4.1.2. Організаційно-технічні заходи з безпеки праці

В організації/підприємстві проводиться навчання і перевірка знань з питань охорони праці відповідно до вимог Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці, затвердженого наказом Держнаглядохоронпраці України від 26.01.2005 N 15, зареєстрованого в Міністерстві юстиції України 15.02.2005 за N 231/10511 [10].

Також впроваджені організаційні заходи з пожежної безпеки - навчання і перевірку знань відповідно до вимог Типового положення про інструктажі, спеціальне навчання та перевірку знань з питань пожежної безпеки на підприємствах, в установах та організаціях України, затвердженого наказом Міністерства України з питань надзвичайних ситуацій та у справах захисту населення від 46 наслідків Чорнобильської катастрофи від 29.09.2003 N 368, зареєстрованого в Міністерстві юстиції України 11.12.2003 за N 1148/8469 [11].

4.2. Аналіз стану умов праці

Робота над створенням додатку для ОС Android проходитиме в приміщенні відповідної установи. Для даної роботи достатньо однієї людини, для якої надано робоче місце зі стаціонарним комп'ютером.

4.2.1. Вимоги до приміщень

Геометричні розміри приміщення зазначені в табл. 4.1. ті відповідають нормам згідно з [12].

Таблиця 4.1 – Розміри приміщення

Найменування	Значення
Довжина, м	5
Ширина, м	5
Висота, м	3
Площа, $м^2$	25
Об'єм, $м^3$	75

4.2.2. Вимоги до організації місця праці

При порівнянні відповідності характеристик робочого місця нормативним основні вимоги до організації робочого місця за [13] (табл. 4.2) і відповідними фактичними значеннями для робочого місця, констатуємо повну відповідність.

Таблиця 4.2 - Характеристики робочого місця

Найменування параметра	Фактичне значення	Нормативне значення
Висота робочої поверхні, мм	750	680 ÷ 800
Висота простору для ніг, мм	730	не менше 600
Ширина простору для ніг, мм	660	не менше 500
Глибина простору для ніг, мм	700	не менше 650
Висота поверхні сидіння, мм	470	400 ÷ 500
Ширина сидіння, мм	400	не менше 400
Глибина сидіння, мм	400	не менше 400
Висота поверхні спинки, мм	600	не менше 300
Ширина опорної поверхні спинки, мм	500	не менше 380
Радіус кривини спинки в горизонтальній площині, мм	400	400
Відстань від очей до екрану дисплея, мм	800	700 ÷ 800

4.2.3. Навантаження та напруженість процесу праці

Під час виконання робіт використовують ПК та периферійні пристрої (лазерні та струменеві), що призводить до навантаження на окремі системи організму. Такі перекоси у напруженні різних систем організму, що трапляються під час роботи з ПК, зокрема, значна напруженість зорового аналізатора і довготривале малорухоме положення перед екраном, не тільки не зменшують загального напруження, а навпаки, призводять до його посилення і появи стресових реакцій.

Роботу за дипломним проектом визнано, таку, що займає 50% часу робочого дня та за восьмигодинної робочої зміни рекомендовано встановити додаткові регламентовані перерви: (потрібне вибрати):

- для розробників програм тривалістю 15 хв через кожну годину роботи;

- для операторів персональних комп’ютерів тривалістю 15 хв через дві години роботи;

- для операторів комп’ютерного набору тривалістю 10 хв через кожну годину роботи.

4.3. Виробнича санітарія

На підставі аналізу небезпечних та шкідливих факторів при виробництві (експлуатації), пожежної безпеки можуть бути надалі вирішені питання необхідності забезпечення працюючих достатньою кількістю освітлення, вентиляції повітря, організації заземлення, тощо.

4.3.1. Аналіз небезпечних та шкідливих факторів при виробництві (експлуатації) виробу

Аналіз небезпечних та шкідливих виробничих факторів виконується у табличній формі (табл. 4.3). Роботу, пов'язану з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із

забезпеченням виконання НПАОП 0.00.-1.28-10 «Правил охорони праці під час експлуатації електронно-обчислювальних машин»

Таблиця 4.3 – Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Кількісна оцінка	Нормативні документи
1	2	3	4
фізичні			
- підвищена температура поверхонь обладнання	експлуатація ЕОМ	2	[12]
- підвищений рівень шуму на робочому місці	-//-	2	[18]
- підвищений рівень вібрації	-//-	2	[19] [20]
- підвищена або знижена вологість повітря	-//-	2	[15]
- підвищений рівень електромагнітного випромінення	-//-	2	[21]
- підвищений рівень напруги електричної мережі, замикання якої може відбутися через тіло людини	-//-	4	[22] [23]
- підвищений рівень статичної електрики	-//-	2	[22]
- недостатність природного світла	порушення умов праці (вимог до приміщень)	2	[16]
хімічні:			
- загазованість повітря робочої зони, яка впливає на організм людини через органи дихання та надає токсичну і канцерогенну дію	оплавлення електричних і комутаційних кабелів, резисторів, конденсаторів, напівпровідникових діодів, транзисторів й інше в ЕОМ та системах кондиціонування повітря - CO, CO_2, SO_2, P_2O_5, H_2S, HCl, H, NH_3, , інш.	3	[24] [25] [26] [27]
психофізіологічні:			
- нервово-психічна перевантаження (розумове, перенапруження аналізаторів-зорових)	- пошук інформації для постановки теми; - пошук та аналіз аналогів і літератури; - пошук наявних технологій, моделювання та аналіз алгоритмів; - виконання роботи за темою	4	[28] [13]

Продовження таблиці 4.3

	диплома, тестування; - оформлення роботи		
- фізичні (статичне – сидіння)	порушення умов праці (організації місця праці-сидіння користувача,) та організації робочого часу - безперервна робота)	2	[28] [13]

4.3.2. Пожежна безпека

Для гасіння пожеж в квартирі приміщенні пропонується використовувати порошкові або вуглекислотні вогнегасники, так як вони є універсальними.

Згідно [14] таке приміщення, площею 25 м^2, відноситься до категорії "В" (пожежонебезпечної). Відповідно до норм первинних засобів пожежогасінні пропонується використовувати:

- ручний вуглекислий вогнегасник ОУ-5 в кількості 1 шт. або хімічний пінний ОХП-10 – 1 шт;

- повсть 1 1 м^2, кошму 2×1,5 м2 або азбестове полотно 2×2 м^2 в кількості 1 шт.

4.3.3. Електробезпека

На робочому місці виконуються наступні вимоги електробезпеки: ПК, периферійні пристрої та устаткування для обслуговування, електропроводи і кабелі за виконанням та ступенем захисту відповідають класу зони за ПУЕ (правила улаштування електроустановок), мають апаратуру захисту від струму короткого замикання та інших аварійних режимів. Лінія електромережі для живлення ПК, периферійних пристроїв і устаткування для обслуговування, виконана як окрема групова три- провідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників мають спеціальні

контакти для підключення нульового захисного провідника. Електромережа штепсельних розеток для живлення персональних ПК, укладено по підлозі поруч зі стінами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання. Металеві труби та гнучкі металеві рукави заземлені. Захисне заземлення включає в себе заземлюючих пристроїв і провідник, який з'єднує заземлюючий пристрій з обладнанням, яке заземлюється - заземлюючий провідник.

4.4. Гігієнічні вимоги до параметрів виробничого середовища

4.4.1. Мікроклімат

Мікроклімат робочих приміщень – це клімат внутрішнього середовища цих приміщень, що визначається діючої на організм людини з'єднанням температури, вологості, швидкості переміщення повітря. Отже оптимальні значення для температури, відносної вологості й рухливості повітря для зазначеного робочого місця відповідають [15] і наведені в табл. 4.4:

Таблиця 4.4 – Норми мікроклімату робочої зони об’єкту

Період року	**Категорія робіт**	**Температура C^0**	**Відносна вологість %**	**Швидкість руху повітря, м/с**
Холодна	легка-1 а	22 - 24	40 – 60	0,1
Тепла	легка-1 а	23 - 25	40 – 60	0,1

4.4.2. Освітлення

У приміщенні, де розташовані ЕОМ передбачається природне бічне освітлення, рівень якого відповідає [16]. Джерелом природного освітлення є сонячне світло. Регулярно повинен проводитися контроль освітленості, який підтверджує, що рівень освітленості задовольняє ДБН і для даного приміщення в світлий час доби достатньо природного освітлення.

Розрахунок освітлення.

Для виробничих та адміністративних приміщень світловий коефіцієнт приймається не менше -1/8, в побутових – 1/10:

$$S_b = \left(\frac{1}{5} \div \frac{1}{10}\right) \cdot S_n, \tag{4.1}$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

$S_n = a \cdot b = 5 \cdot 5 = 25$ м^2,

$S = 1/8 \cdot 25 = 3{,}125$ м^2.

Приймаємо 2 вікна площею S=1,6 м^2 кожне.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників n виробляється по формулі (4.2):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M}, \tag{4.2}$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2; S = 25 м^2;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400лм (для ЛБ-80).

Підставивши числові значення у формулу (4.2), отримуємо:

$$n = \frac{300 \cdot 25 \cdot 1{,}1 \cdot 1{,}5}{5400 \cdot 0{,}575 \cdot 2} \approx 2{,}0$$

Приймаємо освітлювальну установку, яка складається з 2-х світильників, які складаються з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

4.5. Вентилювання

У приміщенні, де знаходяться ЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції (вентиляційні шахти), тобто при V приміщення > 40 м^3 на одного працюючого допускається природна вентиляція. Цей метод забезпечує приток потрібної кількості свіжого повітря, що визначається в СНіП.

4.6. Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Відповідно до санітарно-гігієнічних нормативів та правил експлуатації обладнання наводимо приклади деяких заходів безпеки.

Розрахунок захисного заземлення (забезпечення електробезпеки будівлі).

Згідно з класифікацією приміщень за ступенем небезпеки ураження електричним струмом [17], приміщення в якому проводяться всі роботи відноситься до першого класу (без підвищеної небезпеки). Під час роботи використовуються електроустановки з напругою живлення 36 В, 220 В, та 360 В. Опір контура заземлення повинен мати не більше 4 Ом.

Розрахунок проводять за допомогою методу коефіцієнта використання (екранування) електродів. Коефіцієнт використання групового заземлювача η – це відношення діючої провідності цього заземлювача до найбільш можливої його провідності за нескінченно великих відстаней між його електродами. Коефіцієнт використання вертикальних заземлювачів ηв в

залежності від розміщення заземлювачів та їх кількості знаходиться в межах 0,4…0,99. Взаємну екрануючу дію горизонтального заземлювача (з'єднувальної смуги) враховують за допомогою коефіцієнта використання горизонтального заземлювача η_c.

Послідовність розрахунку.

1) Визначається необхідний опір штучних заземлювачів $R_{шт.з.}$:

$$R_{шт.з.} = \frac{R_д \cdot R_{пр.з.}}{R_{пр.з.} - R_д}, \quad (4.3)$$

де $R_{пр.з.}$ – опір природних заземлювачів; $R_д$ – допустимий опір заземлення. Якщо природні заземлювачі відсутні, то $R_{шт.з.}=R_д$.

Підставивши числові значення у формулу (рис.4.3), отримуємо:

$$R_{шт.з.} = \frac{4 \cdot 40}{40 - 4} \approx 4 \text{ Ом}$$

2) Опір заземлення в значній мірі залежить від питомого опору ґрунту ρ, Ом·м. Приблизне значення питомого опору глини приймаємо ρ=40 Ом•м (табличне значення).

3) Розрахунковий питомий опір ґрунту, $\rho_{розр}$, Ом·м, визначається відповідно для вертикальних заземлювачів $\rho_{розр.в}$, і горизонтальних $\rho_{розр.г}$, Ом·м за формулою:

$$\rho_{розр..} = \psi \cdot \rho, \quad (4.4)$$

де ψ – коефіцієнт сезонності для вертикальних заземлювачів I кліматичної зони з нормальною вологістю землі, приймається для вертикальних заземлювачів $\rho_{розр.в}$=1,7 і горизонтальних $\rho_{розр.г}$=5,5 Ом·м.

$$\rho_{розр.в} = 1,7 \cdot 40 = 68 \text{ Ом•м}$$

$$\rho_{розр.г} = 5,5 \cdot 40 = 220 \text{ Ом•м}$$

4) Розраховується опір розтікання струму вертикального заземлювача $R_в$, Ом, за (4.5).

$$R_в = \frac{\rho_{розр.в}}{2 \cdot \pi \cdot l_в} \cdot \left(\ln \frac{2 \cdot l_в}{d_{ст}} + \frac{1}{2} \cdot \ln \frac{4 \cdot t + l_в}{4 \cdot t - l_в} \right), \quad (4.5)$$

де $l_в$ – довжина вертикального заземлювача (для труб - 2−3 м; $l_в$=3 м);

$d_{ст}$ – діаметр стержня (для труб - 0,03−0,05 м; $d_{ст}$=0,05 м);

t – відстань від поверхні землі до середини заземлювача, яка визначається за ф. (4.6):

$$t = h_в + \frac{l_в}{2}, \quad (4.6)$$

де $h_в$ – глибина закладання вертикальних заземлювачів (0,8 м); тоді $t = 0,8 + \frac{3}{2} = 2,3$ м

$$R_в = \frac{68}{2 \cdot \pi \cdot 3} \cdot \left(\ln \frac{2 \cdot 3}{0,05} + \frac{1}{2} \cdot \ln \frac{4 \cdot 2,3 + 3}{4 \cdot 2,3 - 3} \right) = 18,5 \text{ Ом}$$

5) Визначається теоретична кількість вертикальних заземлювачів n штук, без урахування коефіцієнта використання $\eta_в$:

$$n = \frac{2 \cdot R_в}{R_д} = \frac{2 \cdot 18,5}{4} = 9,25 \quad (4.7)$$

І визначається коефіцієнт використання вертикальних електродів групового заземлювача без врахування впливу з’єднувальної стрічки $\eta_в$ =0,57 (табличне значення).

6) Визначається необхідна кількість вертикальних заземлювачів з урахуванням коефіцієнта використання $n_в$, шт:

$$n_в = \frac{2 \cdot R_в}{R_д \cdot \eta_в} = \frac{2 \cdot 18,5}{4 \cdot 0,57} = 16,2 \approx 16 \quad (4.8)$$

7) Визначається довжина з’єднувальної стрічки горизонтального заземлювача $l_с$, м:

$$l_с = 1,05 \cdot L_в \cdot (n_в - 1), \quad (4.9)$$

де $L_в$ – відстань між вертикальними заземлювачами, (прийняти за $L_в$ = 3м);

$n_в$ – необхідна кількість вертикальних заземлювачів.

$$l_с = 1,05 \cdot 3 \cdot (16-1) \approx 48_{м}$$

8) Визначається опір розтіканню струму горизонтального заземлювача (з'єднувальної стрічки) $R_г$, Ом:

$$R_г = \frac{\rho_{розр.г}}{2 \cdot \pi \cdot l_с} \cdot \ln \frac{2 \cdot l_с^{\,2}}{d_{см} \cdot h_г}, \qquad (4.10)$$

де $d_{см}$ – еквівалентний діаметр смуги шириною b, $d_{см}$ =0,95b, b = 0,15 м;

$h_г$ – глибина закладання горизонтальних заземлювачів (0,5 м);

$l_с$ - довжина з'єднувальної стрічки горизонтального заземлювача $l_с$, м

$$R_г = \frac{220}{2 \cdot \pi \cdot 48} \cdot \ln \frac{2 \cdot 48^2}{0,95 \cdot 0,15 \cdot 0,5} = 8,1 \text{ Ом}$$

9) Визначається коефіцієнт використання горизонтального заземлювача $\eta_{с.}$ відповідно до необхідної кількості вертикальних заземлювачів $n_в$.

Коефіцієнт використання з'єднувальної смуги $\eta_с$=0,3 (табличне значення).

10) Розраховується результуючий опір заземлювального електроду з урахуванням з'єднувальної смуги:

$$R_{заг} = \frac{R_в \cdot R_г}{R_в \cdot \eta_с + R_г \cdot n_в \cdot \eta_в} \le R_д. \qquad (4.11)$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: $R_{заг}$ <4 Ом, а саме:

$$R_{заг} = \frac{18,5 \cdot 8,1}{18,5 \cdot 0,3 + 8,1 \cdot 16 \cdot 0,57} = 1,9 \le R_д$$

Висновки до розділу 4

В результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом написаному в кваліфікаційній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника.

Приведені рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електробезпеки. Були наведені розміри приміщення та наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, а також – наведені інструкції з охорони праці, техніки безпеки при роботі на комп’ютері.

ВИСНОВКИ

В даній роботі було проведено огляд предметної області, що стосується теми дипломного проекту «Додаток для ОС Android з використанням мов XML, Java». Було виконано порівняльний аналіз аналогічних додатків. Розглянута та проаналізована мова розмітки XML, та мова програмування Java. Обґрунтована актуальність створення додатка на ОС Android. Розробка додатку навігатора та гіда по місту є актуальним тому що це дуже заощаджує час користувача на пошуки потрібного закладу, до потрібного закладу можна легко прокласти маршрут, також дозволяє краще орієнтуватися у місті за рахунок отримання інформації про саме місто, також представлені три основні категорії по яким можна знайти потрібний заклад та отримати інформацію по ньому, також можна дізнатися про оточуючі користувача заклади. Додаток більше зорієнтований на туристів та людей не місцевих.

Для вирішення задачі потрібно проаналізувати предметну область та існуючі програмні засоби, які її описують, спроектувати та розробити додаток навігатор для ОС Android з використанням мов XML, Java.

Були сформульовані основні задачі та описані методи та засоби, які були обрані для реалізації додатку. Також були враховані недоліки аналогічних додатків які були розглянені. Додаток навігатор та гід по місту був створений відповідно до технічного завдання на розробку та протестований.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ НА ВИКОРИСТАНУ ЛІТЕРАТУРУ ТА ІНШУ НОРМАТИВНО-ТЕХНІЧНУ ДОКУМЕНТАЦІЮ

1. Кєй Хорстманн, Гарі Корнелл «Java. Библиотека профессионала. Том 1».10-е видання 2016, PDF.
2. Г. Шілдт. «Java 8. Полное руководство 9-е издание» 2015.
3. Г. Шілдт. «Java 8. Руководство для начинающих»(6-е видання) 2015.
4. Developers [електронний ресурс] / Розбір XML-даних – Режим доступу: www. URL: http://developer-android.unlimited-translate.org/training/basics/network-ops/xml.html
5. EASYCODE [електронний ресурс] / Ресурси – Android додатки – Режим доступу: www. URL: http://easy-code.com.ua/2014/09/resursi-android-dodatki/
6. IBM [електронний ресурс] / Робота з XML в Android – Режим доступу: www. URL: https://www.ibm.com/developerworks/ru/library/x-android/index.html
7. «Международный научно-исследовательский журнал» [електронний ресурс] / Розробка програми для Android на Java – Режим доступу: www. URL: https://research-journal.org/technical/razrabotka-prilozheniya-dlya-android-na-java/
8. stfalcon.com [електронний ресурс] / Особливості Java з точки зору Android-розробника – Режим доступу: www. URL: https://stfalcon.com/ru/blog/post/android-developer-java-review

До розділу 4:

9. НПАОП 0.00-6.03-93 «Порядок опрацювання та затвердження власником нормативних актів про охорону праці, що діють на підприємстві».
10. НПАОП 0.00-4.12- 05 «Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці».

11. НАПБ Б.02.005-2003 «Типове положення про інструктажі, спеціальне навчання та перевірку знань з питань пожежної безпеки на підприємствах, в установах та організаціях України».

12. ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень».

13. ДСанПіН 3.3.2.007-98 «Правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

14. НАПБ Б.03.002-2007 «Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою».

15. ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень».

16. ДБН В.2.5-28:2015 «Природне і штучне освітлення».

17. НПАОП 40.1-1.01-97 «Правила безпечної експлуатації електроустановок».

18. ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку».

19. ДСН 3.3.6.039-99 «Державні санітарні норми виробничої загальної та локальної вібрації».

20. ДСТУ ГОСТ 12.1.012-90 «Система стандартів безпеки праці вібраційна безпеку загальні вимоги».

21. ГОСТ 12.1.006-84 «Електромагнітні поля радіочастот. Допустимі рівні на робочих місцях і вимоги до проведення контролю».

22. ГОСТ 12.1.030-81 «Електробезпека. Захисне заземлення. Занулення».

23. ГОСТ 13109-97 «Норми якості електричної енергії в системах електропостачання загального призначення».

24. НПАОП 40.1-1.21-98 «Правила безпечної експлуатації електроустановок споживачів».

25. ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування».

26. ГОСТ 12.1.005-88 «Загальні санітарно-гігієнічні вимоги до повітря робочої зони».

27. ГОСТ 12.1.044-89 «Пожежевибухонебезпечність речовин і матеріалів».

28. НПАОП 0.00-1.28-10 «Про затвердження правил охорони праці під час експлуатації електронно-обчислювальних машин».

ДОДАТОК А

Лістинг коду Constants.java :

```
package com.aleksey.navigator.googleplaces;
import com.aleksey.navigator.R;
import com.aleksey.navigator.settings.AppSettings;
import java.util.ArrayList;
import java.util.List;
/**
 * Створив alekseyfedoryachenko.
 */
public class Constants {

    public static final String PLACE_PHOTO_URL =
"https://maps.googleapis.com/maps/api/place/photo?maxwidth=%d&ph
otoreference=%s&key=%s";

    public static final String NEARBY_PLACES_URL =
"https://maps.googleapis.com/maps/api/place/search/json";

    public static final String GOOGLE_PLUS_URL =
"https://www.googleapis.com/plus/v1/people/";

    /**
     * Значення за замовчуванням
     */
    public static int DEFAULT_PHOTO_WIDTH = 540;

    public enum PLACE_TYPES {

        GOOGLE_PLACES_SLEEP("Sleep in", R.color.ab_color_green,
"lodging"),

        GOOGLE_PLACES_EAT("Eat in", R.color.ab_color_blue,
"bakery|meal_delivery|meal_takeaway|food|restaurant"),

        GOOGLE_PLACES_ENJOY("Enjoy in", R.color.ab_color_purple,
"cafe|bar|night_club"),

        GOOGLE_PLACES_FAV("Favourite in", R.color.ab_color_fav,
"");

        /**
         * Назва типу. Встановити панелі дій на екран зі списком
місць
         */
        private String title;
        /**
         * Колір поточного типу. Використовується для
встановлення панелі дій на екрані зі списком та інших
переглядів, якщо це потрібно
```

```
         */
        int resourceId;
        /**
         * Google тип. Використовуємо це для створення URL-
адреси для отримання місць
         */
        String type;

        PLACE_TYPES(String title, int resourceId, String value)
{
            this.title = title;
            this.resourceId = resourceId;
            this.type = value;
        }

        public static List<String> getAllTypesAsList() {

            List<String> result = new ArrayList<String>();

            for (int i = 0; i < PLACE_TYPES.values().length;
i++) {
                result.add(PLACE_TYPES.values()[i].getType());
            }

            return result;
        }

        public static PLACE_TYPES isValueInAnyType(String value)
{

            for (int i = 0; i < PLACE_TYPES.values().length;
i++) {
                if
(values()[i].getType().toString().indexOf(value) > -1) {
                    return values()[i];
                }
            }

            return null;
        }

public static int getFullRatingImageForType(PLACE_TYPES type) {
            if (type == GOOGLE_PLACES_SLEEP) {

                return R.drawable.ic_rate_sleep_full;

            } else if (type == GOOGLE_PLACES_EAT) {
                return R.drawable.ic_rate_eat_full;

            } else if (type == GOOGLE_PLACES_ENJOY) {
                return R.drawable.ic_rate_enjoy_full;
            } else if (type == GOOGLE_PLACES_FAV) {
                return R.drawable.ic_rate_fav_full;
```

```
            }

            return 0;
        }

        public static int getHalfRatingImageForType(PLACE_TYPES
type) {
            if (type == GOOGLE_PLACES_SLEEP) {

                return R.drawable.ic_rate_sleep_half;

            } else if (type == GOOGLE_PLACES_EAT) {
                return R.drawable.ic_rate_eat_half;

            } else if (type == GOOGLE_PLACES_ENJOY) {
                return R.drawable.ic_rate_enjoy_half;
            } else if (type == GOOGLE_PLACES_FAV) {
                return R.drawable.ic_rate_fav_half;
            }

            return 0;
        }

        public static int getFavIconEmptyId(PLACE_TYPES type) {
            if (type == GOOGLE_PLACES_SLEEP) {

                return R.drawable.selector_fav_sleep;

            } else if (type == GOOGLE_PLACES_EAT) {
                return R.drawable.selector_fav_eat;

            } else if (type == GOOGLE_PLACES_ENJOY) {
                return R.drawable.selector_fav_enjoy;
            } else if (type == GOOGLE_PLACES_FAV) {
                return R.drawable.selector_favourite;
            }

            return 0;
        }

        public static int getIconMap(PLACE_TYPES type) {
            if (type == GOOGLE_PLACES_SLEEP) {

                return R.drawable.ic_sleep_map;
            } else if (type == GOOGLE_PLACES_EAT) {

                return R.drawable.ic_eat_map;
            } else if (type == GOOGLE_PLACES_ENJOY) {

                return R.drawable.ic_enjoy_map;
            }else if(type == GOOGLE_PLACES_FAV){

                return R.drawable.ic_fav_map;
```

```
            }

            return 0;
        }

        public static int getIconPhone(PLACE_TYPES type) {
            if (type == GOOGLE_PLACES_SLEEP) {

                return R.drawable.ic_sleep_call;

            } else if (type == GOOGLE_PLACES_EAT) {
                return R.drawable.ic_eat_call;

            } else if (type == GOOGLE_PLACES_ENJOY) {
                return R.drawable.ic_enjoy_call;
            } else if (type == GOOGLE_PLACES_FAV) {
                return R.drawable.ic_fav_call;
            }

            return 0;
        }

public static int getIconEye(PLACE_TYPES type) {
            if (type == GOOGLE_PLACES_SLEEP) {

                return R.drawable.ic_sleep_view;

            } else if (type == GOOGLE_PLACES_EAT) {
                return R.drawable.ic_eat_view;

            } else if (type == GOOGLE_PLACES_ENJOY) {
                return R.drawable.ic_enjoy_view;
            } else if (type == GOOGLE_PLACES_FAV) {
                return R.drawable.ic_fav_view;
            }

            return 0;
        }

        public static int getEmptyReviewImage(PLACE_TYPES type)
{

            if (type == GOOGLE_PLACES_SLEEP) {

                return R.drawable.ic_sleep_ratings_user_pic;

            } else if (type == GOOGLE_PLACES_EAT) {
                return R.drawable.ic_eat_ratings_user_pic;

            } else if (type == GOOGLE_PLACES_ENJOY) {
                return R.drawable.ic_enjoy_ratings_user_pic;
            }
```

```
        return 0;
    }

    public static int getTagsBg(PLACE_TYPES type) {

        if (type == GOOGLE_PLACES_SLEEP) {

            return R.drawable.tags_bg_green;

        } else if (type == GOOGLE_PLACES_EAT) {
            return R.drawable.tags_bg_blue;

        } else if (type == GOOGLE_PLACES_ENJOY) {
            return R.drawable.tags_bg_purple;
        } else if (type == GOOGLE_PLACES_FAV) {
            return R.drawable.tags_bg_fav;
        }

        return 0;
    }

    public static int getSearchIcon(PLACE_TYPES type) {

        if (type == GOOGLE_PLACES_SLEEP) {

            return R.drawable.ic_search_sleep;

        } else if (type == GOOGLE_PLACES_EAT) {
            return R.drawable.ic_search_eat;

        } else if (type == GOOGLE_PLACES_ENJOY) {
            return R.drawable.ic_search_enjoy;
        }

        return 0;
    }

    public static int getSearchAutoTextBg(PLACE_TYPES type)
{

        if (type == GOOGLE_PLACES_SLEEP) {

            return R.drawable.et_half_border_green;

        } else if (type == GOOGLE_PLACES_EAT) {
            return R.drawable.et_half_border_blue;

        } else if (type == GOOGLE_PLACES_ENJOY) {
            return R.drawable.et_half_border_purple;
        }

        return 0;
    }
```

```
        public static int getEmptyImagePlaceHolder(PLACE_TYPES
type) {

            if (type == GOOGLE_PLACES_SLEEP) {

                return R.drawable.sleep_placeholder;
            } else if (type == GOOGLE_PLACES_EAT) {

                return R.drawable.eat_placeholder;
            } else if (type == GOOGLE_PLACES_ENJOY) {

                return R.drawable.enjoy_placeholder;
            } else if (type == GOOGLE_PLACES_FAV) {

                return R.drawable.fav_placeholder;
            }

            return 0;
        }

        public String getTitle() {
            return title + " " + AppSettings.TOWN;
        }

        public int getColourId() {
            return resourceId;
        }

        public String getType() {
            return type;
        }
    }
}
```

ДОДАТОК Б

Комп’ютерна презентація:

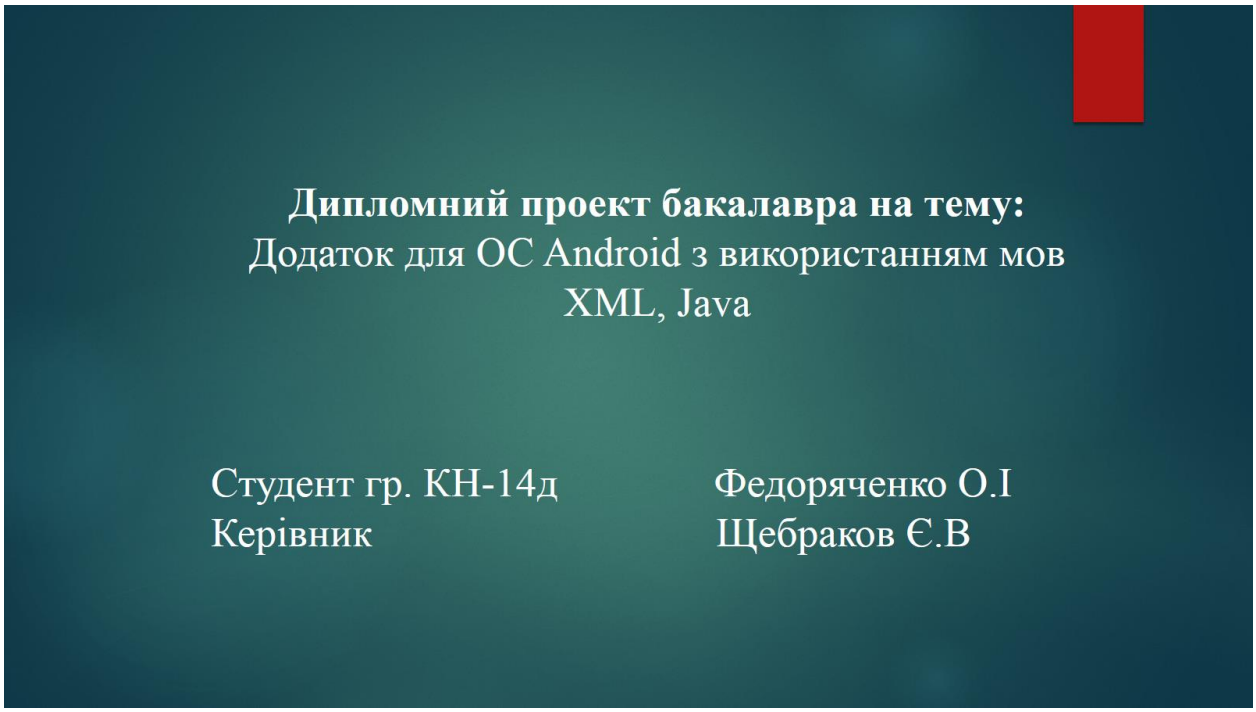

Рисунок Б.1 – Титульний лист

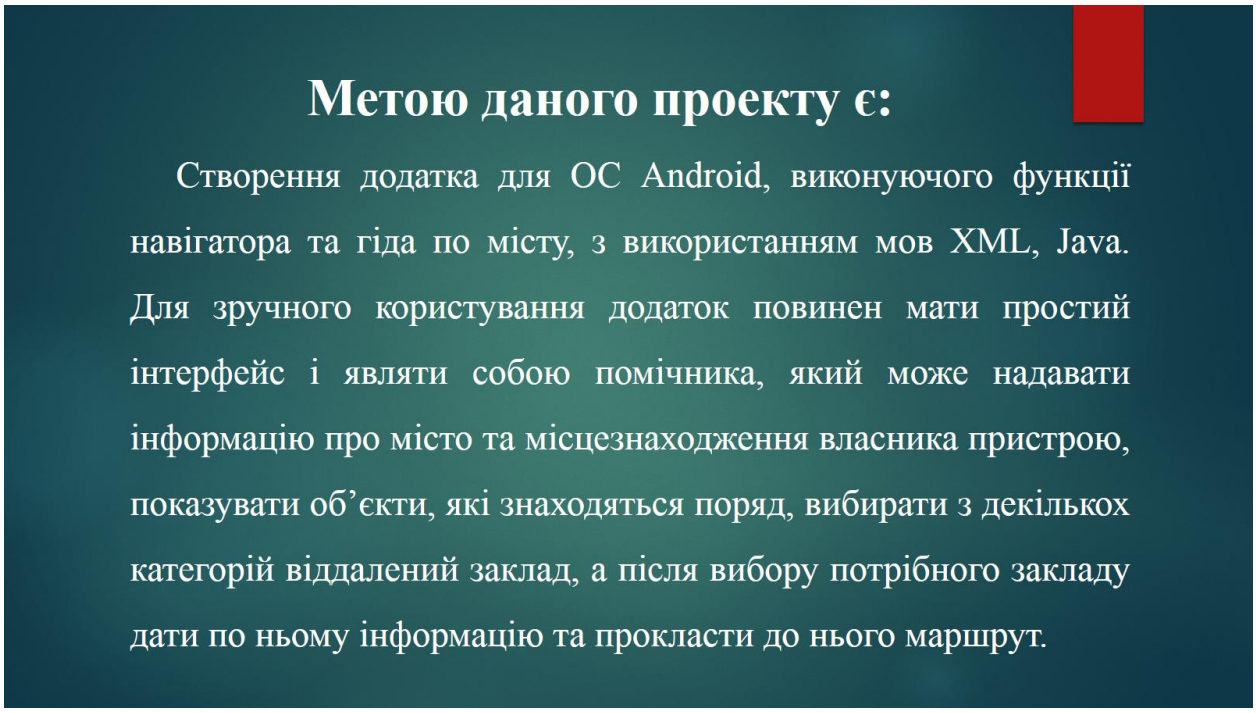

Рисунок Б.2 – Мета проекту

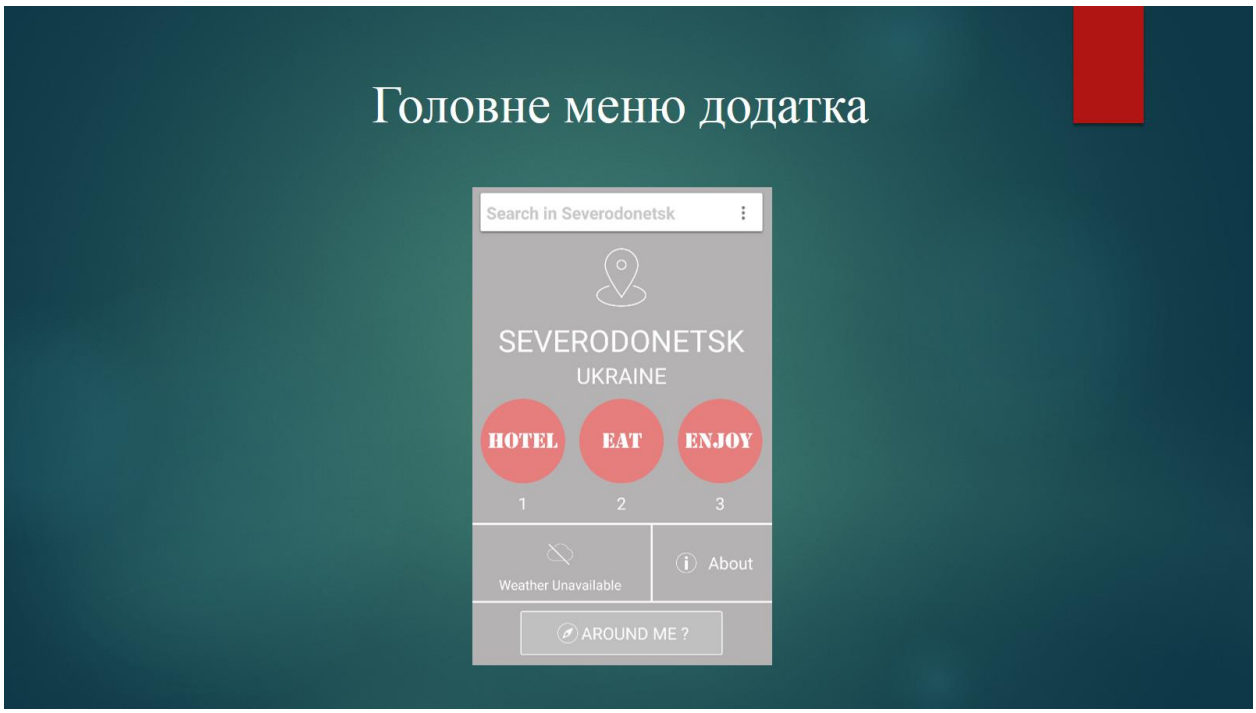

Рисунок Б.3 – Головне меню додатка

Рисунок Б.4 – Сторінка «About»

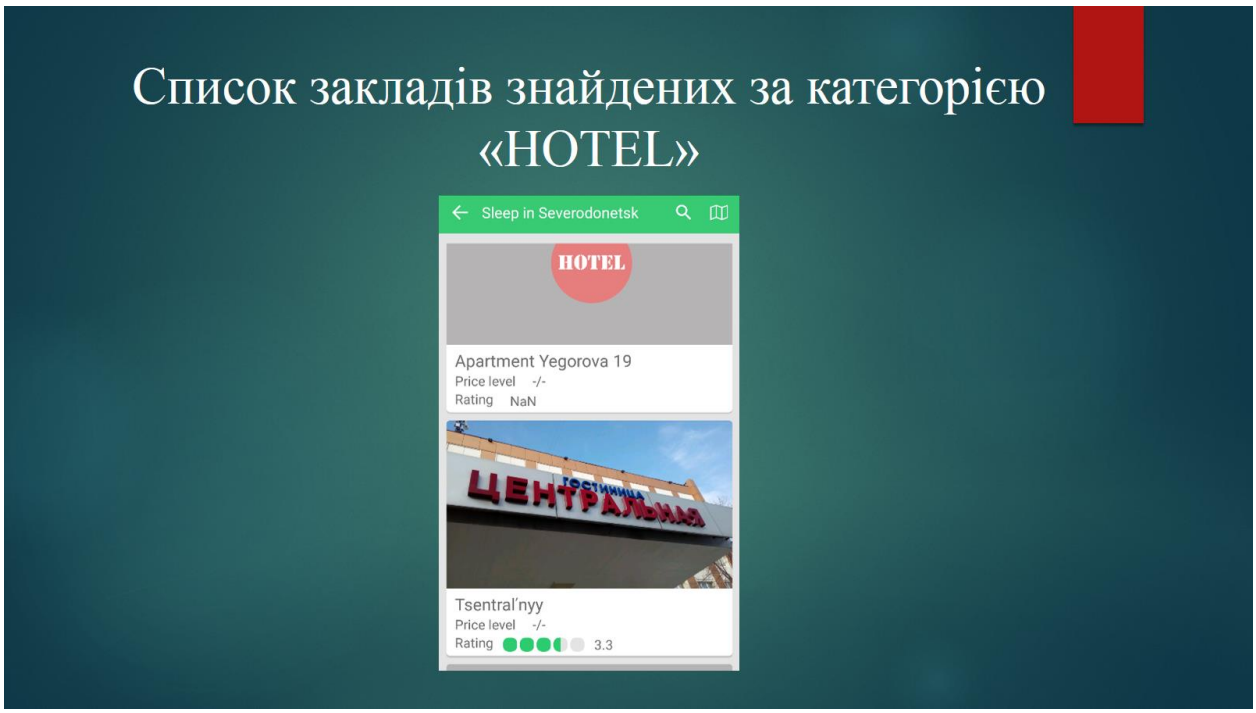

Рисунок Б.5 – Список закладів знайдених за категорією «HOTEL»

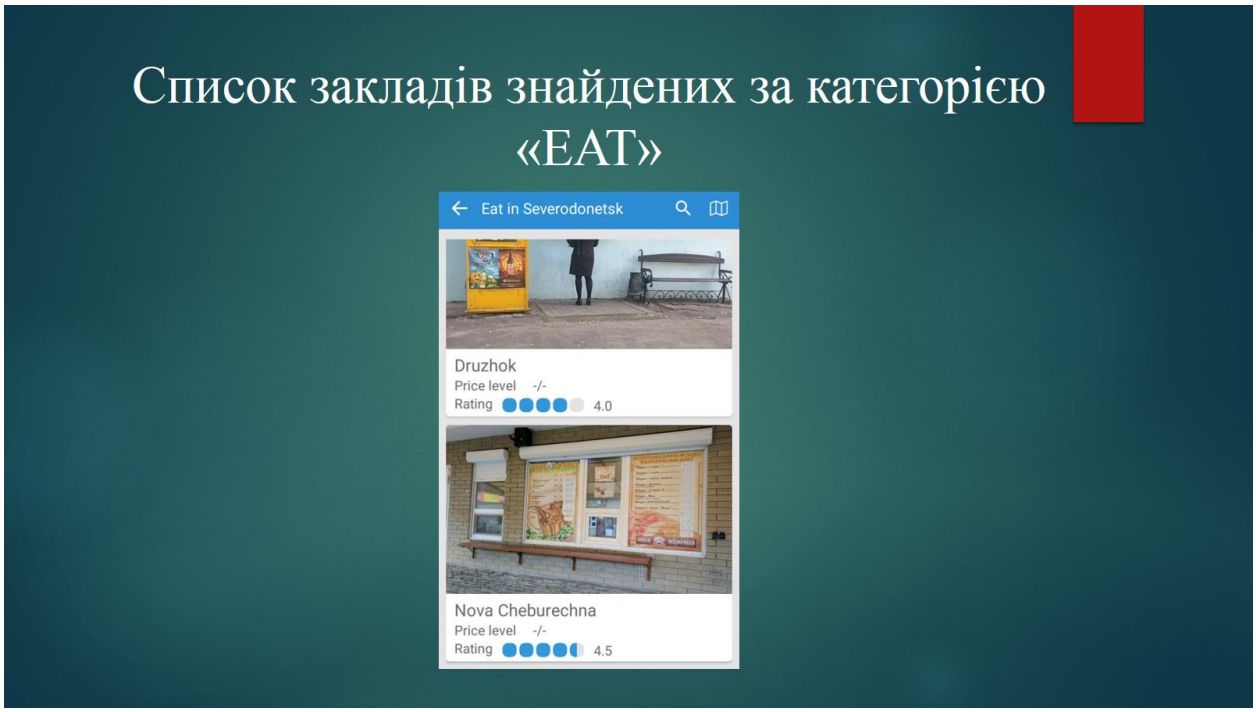

Рисунок Б.6 – Список закладів знайдених за категорією «EAT»

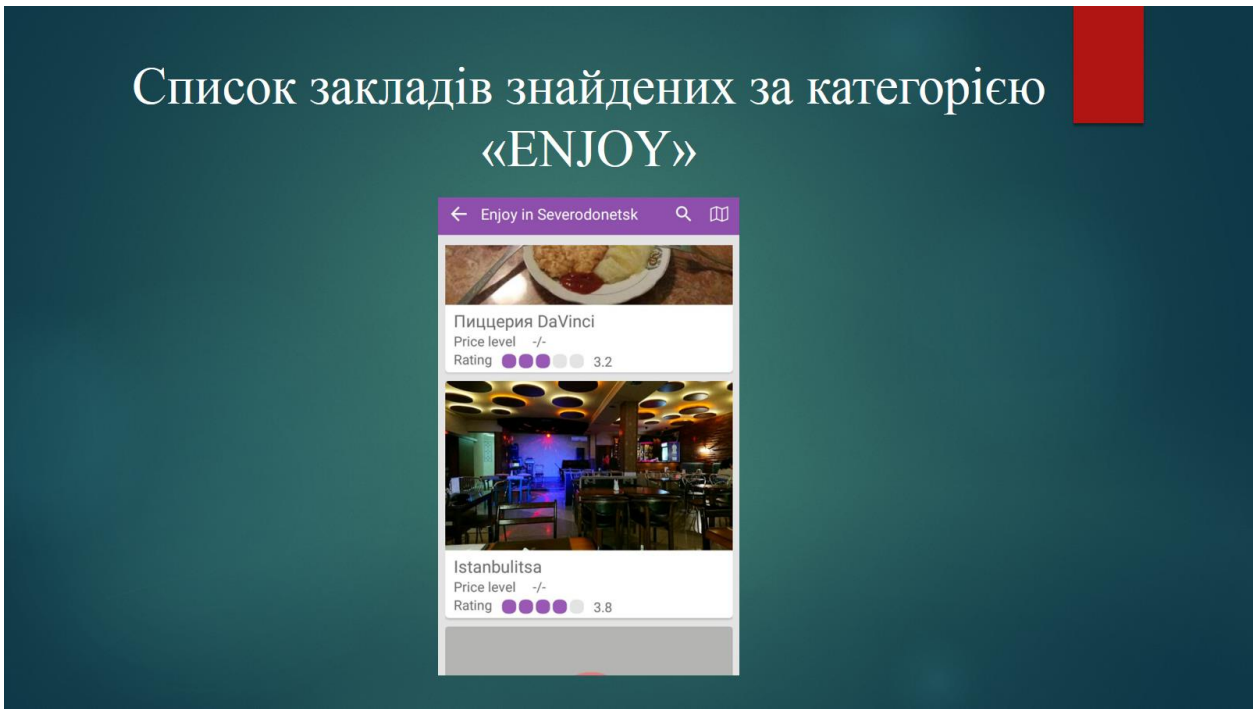

Рисунок Б.7 – Список закладів знайдених за категорією «ENJOY»

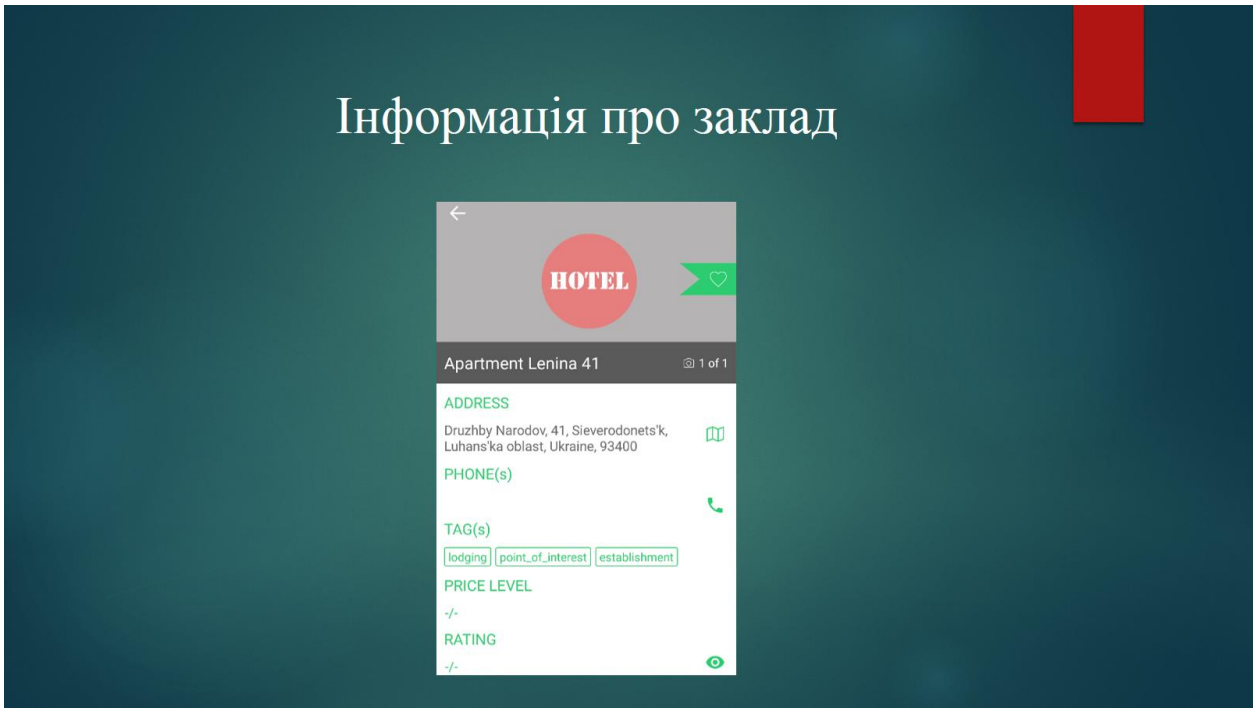

Рисунок Б.8 – Інформація про заклад

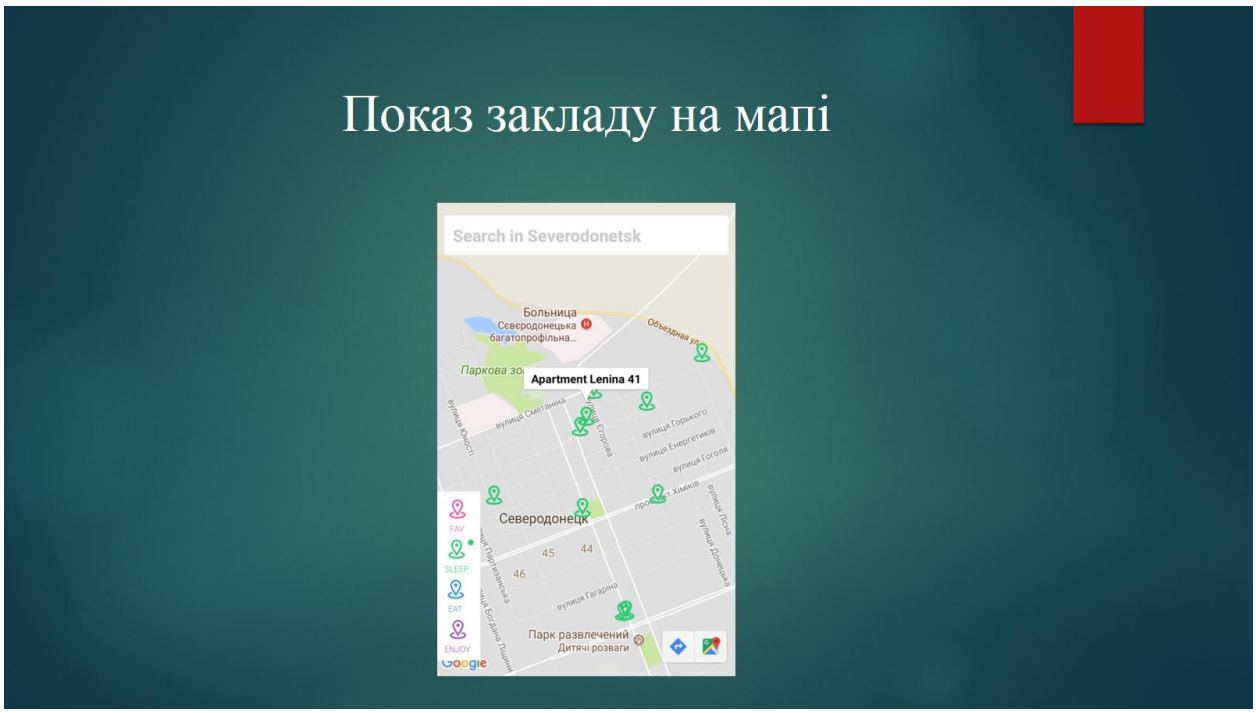

Рисунок Б.9 – Показ закладу на мапі

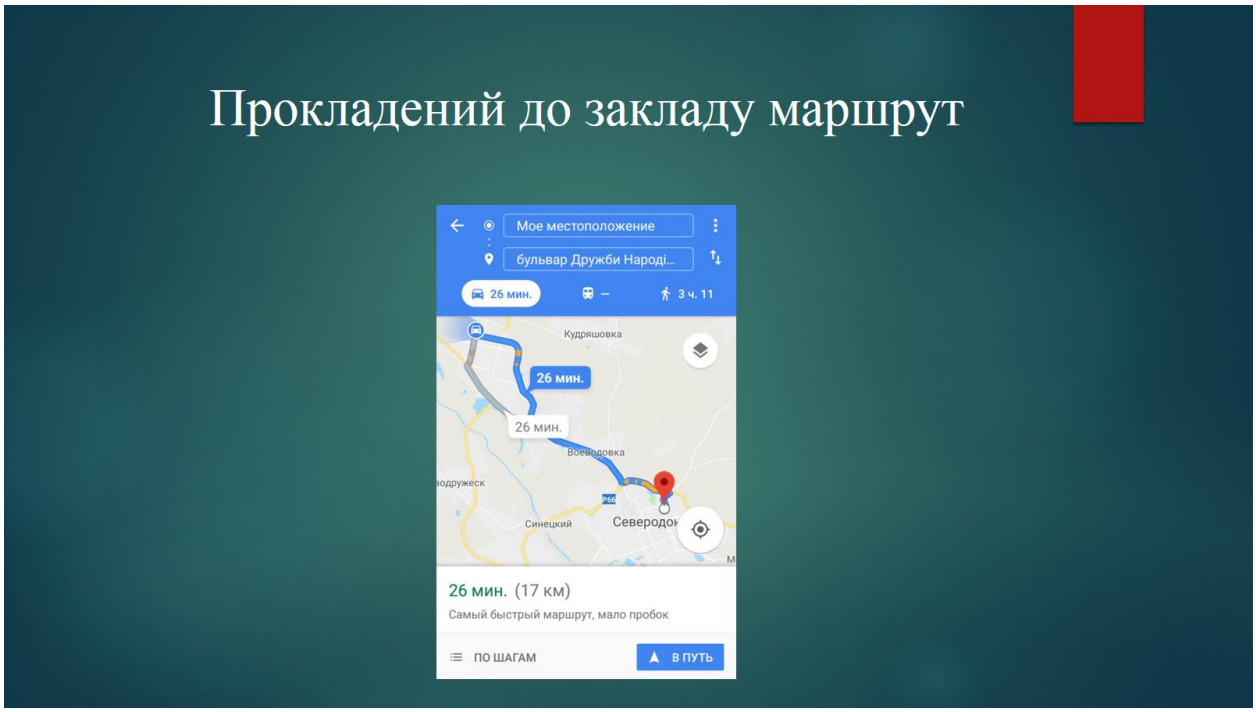

Рисунок Б.10 – Прокладений до закладу маршрут

Рисунок Б.11 – Структура додатка

Рисунок Б.12 – Схема обслуговування користувача додатка

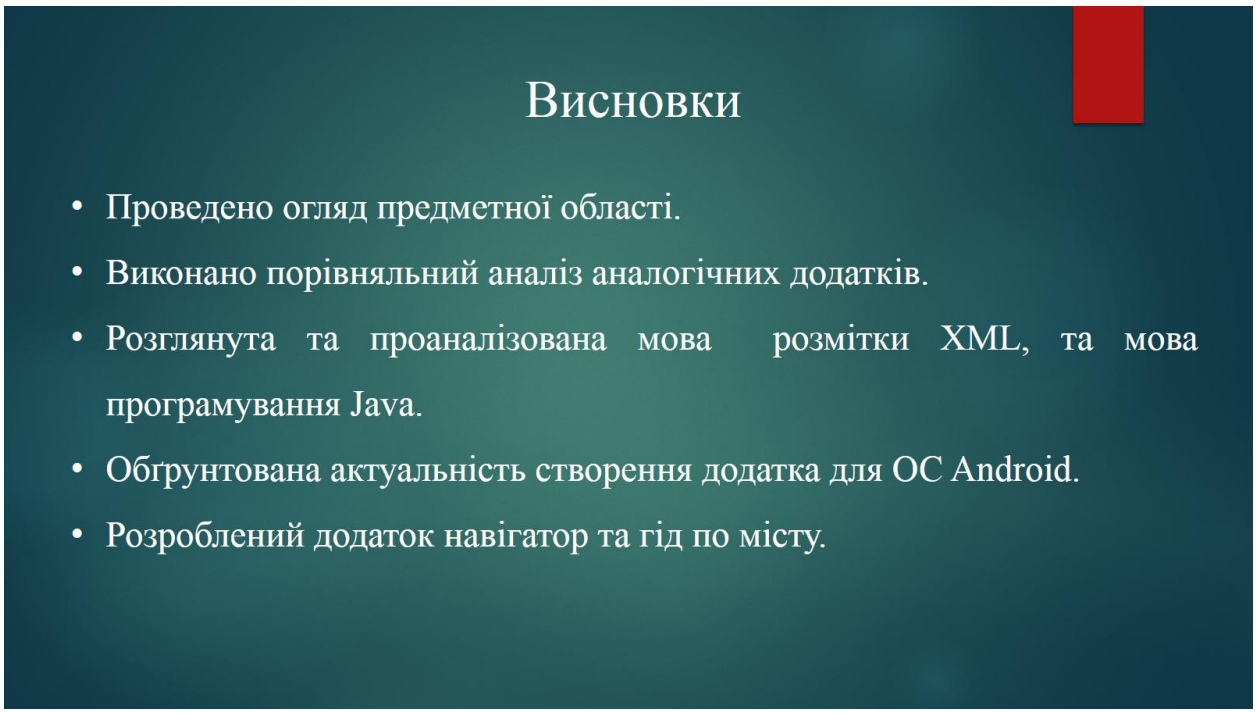

Рисунок Б.13 – Висновки