МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Інтерактивний додаток для вивчення історії Сумщини»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ-62 Ткаченко Антон Юрійович

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою _____ «__» _____ 2020 р.

Науковий керівник _____ к.т.н., доц., Федотова Н.А.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Голова комісії _____ Шифрін Д. М.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

Суми-2020

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«___»_____2020 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Ткаченко Антон Юрійович

1 Тема роботи *Інтерактивний додаток для вивчення історії Сумщини*

керівник роботи *Баранова Ірина Володимирівна, к.т.н., доцент*

затверджені наказом по університету від « 14 » травня 2020 р. № 0576-III

2 Строк подання студентом роботи «1» червня 2020 р.

3 Вхідні дані до роботи технічне завдання на розробку інтерактивного додатку

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області інтерактивного додатку, функціональне і структурне проектування додатку, розробка інтерактивного додатку.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність роботи, аналіз програмних продуктів-аналогів, мета та задача дипломного проекту, аналіз технологій реалізації, етапи розробки інтерактивного додатку

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання_____01.10.2019_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження технологій створення мобільного додатку	01.01.2020-15.02.2020	
2	Створення макету інтерфейсу додатку	01.03.2020-16.03.2020	
3	Функція «карта»: Застосування google maps для реалізації карти в додатку	17.03.2020-31.03.2020	
4	Створення функції пошуку місцерозташування користувача	01.04.2020-15.04.2020	
5	Створення меню історичних пам’яток.	16.04.2020-30.05.2020	
6	Створення сцени перегляду 3D моделі пам’ятки	01.02.2020-10.02.2020	
7	Заповнення бази даних та підключення файлів моделей	11.04.2020-14.05.2020	
8	Завершення розробки: Проведення виправлень додатку, тестування реалізованого функціоналу	15.05.2020-27.05.2020	

Студент _____ Ткаченко А.Ю..

(підпис)

Керівник роботи _____ к.т.н., доц. Федотова Н.А.

(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Інтерактивний додаток для вивчення історії Сумщини».

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 24 найменувань, додатків. Загальний обсяг роботи – 85 сторінок, у тому числі 41 сторінка основного тексту, 2 сторінка списку використаних джерел, 44 сторінок додатків.

Кваліфікаційну роботу бакалавра присвячено розробці мобільного додатку для вивчення історії видатних пам’яток міста Суми.

У роботі проведено аналіз предметної області та моделювання інтерактивного додатку. Висвітлено дослідження актуальності проблеми, огляд існуючих програмних продуктів та поставленні задачі проекту.

У другому розділі наведено структурно-функціональне моделювання роботи мобільного додатку, проектування моделі бази даних та моделювання варіантів використання.

У третьому розділі показано практичну реалізацію розробки мобільного додатку: розроблено архітектуру, інтерфейс, представлено програмну реалізацію функцій додатку, та представлення кінцевого результату.

Результатом проведеної роботи є мобільний додаток, який дає можливість ознайомитись з історичними пам’ятками міста Суми, подивитися їх розташування на карті і визначити де знаходиться користувач та продемонструвати 3D модель пам’ятки. Практичне значення роботи полягає у підвищенні розвитку туристичної галузі міста та інформуванні туристів та населення про історичні пам’ятки.

Ключові слова: МОБІЛЬНИЙ ДОДАТОК, КАРТА, ІНФОРМАЦІЯ, ІСТОРИЧНИЙ ОПИС, ПАМ’ЯТКА, 3D МОДЕЛЬ, БАЗА ДАНИХ.

ЗМІСТ

ВСТУП

Подорож, завжди була, і є на сьогодні невід'ємною частиною людського життя. Для деяких це просто хобі, а для інших це стиль життя. Але, і тих і тих об'єднує цікавість та бажання пізнання чогось нового для себе. Нових країн, міст, пам'яток архітектури. Та кожен прагне зробити свою подорож максимально зручною. У сучасному світі технології весь час розвиваються, і тепер люди подорожують не з паперовими мапами, а з цифровими, що значно полегшує мандрівку.

Та багато людей, які приїжджають перший раз до незнайомого міста, мають труднощі з пошуком цікавих місць та видатних пам'яток. Було б зручно мати у своєму смартфоні додаток-екскурсовод, який покаже, де знаходяться визначні пам'ятки, розкаже про них, та допоможе до них дібратися. Саме такого інтерактивного додатку не вистачає місту Суми.

Метою дипломного проекту є розробка мобільного додатку-екскурсоводу, завдяки якому користувач зможе знайти певну визначну пам'ятку міста Суми, дізнатися її історію, прокласти маршрут до неї, та переглянути її 3D модель.

Для досягнення поставленої мети було сформульовано перелік задач, які необхідно дослідити:

- Ознайомлення з розробкою мобільних додатків та дослідження літератури за цією темою.
- Пошук подібних мобільних додатків та ознайомлення з ними.
- Створення 3D моделей будівель, які будуть відображатися у додатку.
- Розробка мобільного додатку, який полегшить пошук визначних пам'яток міста Суми.

Результатом дипломної роботи має бути мобільний додаток, у якому користувач зможе відкрити карту міста Сум, визначити місцезнаходження користувача та продивитися на карті визначні пам'ятки; також користувач зможе продивитися історію даної споруди та її 3D модель.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження актуальності проблеми

Предметна область додатку є освітньо-туристичною. Додаток є інтерактивним екскурсоводом. Впродовж користування додатком користувач не тільки дізнається маршрути до певних пам’яток, але й ознайомиться з історією відомих будівель міста Суми. Також, завдяки завантаженим 3D-моделям будівель користувач зможе детальніше роздивитися споруди.

Предметна область додатку має дві проблеми, які додаток вирішує:

1) Освітня: якщо користувач нічого не знає або знає недостатньо про пам’ятку, то він може дізнатися про її історію за допомогою додатку;
2) Туристична: додаток допомагає знайти користувачеві найкоротший маршрут до певної історичної пам’ятки за допомогою карти, що сильно полегшує подорожування по місту.

1.2 Огляд існуючих програмних продуктів

Для виявлення подібних програмних продуктів, необхідно поставити певні вимоги, які будуть відповідати вимогам розроблюваного додатку. В першу чергу програмний продукт має бути мобільним додатком для платформи андроїд. По-друге він має вирішувати наступні задачі:

1) Пошук видатних пам’яток міста та прокладання маршруту до них.
2) Надавати історичні відомості про певну пам’ятку.
3) Демонстрація пам’ятки у вигляді 3D моделі, яку можна обертати.

Серед всіх існуючих додатків найбільш відповідним до вимог є додаток під назвою “Redigo”. Додаток “Redigo” [1] – це компактний путівник по різним країнам світу, який працює в оффлайн режимі та має наступні можливості:

1) Надання загальних відомостей про певну країну або про певне місто.
2) Пошук видатних пам’яток за допомогою карти
3) Розмовник, який включає в себе шість різних мов, та можливість виразиться без слів за допомогою ілюстрацій.

Інтерфейс екрану з відкритим мобільним додатком “Redigo” зображений на рисунку 1.1.

Рисунок 1.1 – Зображення головного меню додатку “Redigo”

Проведемо аналіз недоліків та переваг мобільного додатку “Redigo”.

Переваги:

1) Підтримує більшу частину країн світу.

2) Має розмовник який включає в себе шість різних мов, та можливість виразиться без слів за допомогою ілюстрацій.
3) Може працювати в режимі оффлайн.

Недоліки:

1) Немає історичного опису пам’ятки.
2) Немає демонстрації 3D моделі будівлі.

Також один з найвідоміших подібних додатків є додаток “MomondoPlaces” [2]. Цей додаток пропонує маршрути для екскурсій на основі настрою користувача. Користувач обирає одне з запропонованих місць, та додаток прокладає маршрут до нього.

Рисунок 1.2 – Зображення головного меню додатку “ MomondoPlaces ”

Проведемо аналіз недоліків та переваг мобільного додатку “MomondoPlaces”.

Переваги:

1) Вибір здійснюється на основі настрою користувача.
2) Працює в режимі офлайн.

Недоліки:

1) Працює тільки із закладами типу ресторанів, кафе.

В результаті аналізу аналогічних додатків зроблено висновок про необхідність розробки власного додатку, який не матиме зазначених недоліків.

1.3 Постановка задачі проекту

Мета дипломного проекту полягає в створенні інтерактивного додатку, що полегшить пошук визначних пам’яток по місту Суми та дасть можливість ознайомлення з їх історією.

Для досягнення мети необхідно вирішити такі задачі:

- Провести дослідження літератури за темою створення мобільних додатків на андроїд, а саме вивчення мови програмування Java та ознайомлення з середовищем програмування.
- Дослідити метод інтеграції карти у мобільний додаток та роботи з нею. Застосувати даний метод при розробці.
- застосувати технологію 3D моделювання для відтворення 3D моделей будівель. Змоделювати будівлі та інтегрувати їх у додаток.

Функціональні вимоги до створюваного додатку:

- Інтерактивний додаток має бути реалізованим у вигляді мобільного додатку для ОС Андроїд та бути у відкритому доступі на платформі PlayMarket.
- Повинна бути можливість працювати з картою (на якій відображено місце розташування користувача, місце розташування пам'яток (найвідоміших) зазначених маркерами).

- Повинен бути віртуальний перегляд пам'яток, супроводжуваний описом історії споруди; функція має вигляд списку із пам’яток (також ця функція відкривається через карту)

Більш детальна інформація про розробку мобільного додатку представлена в технічному завданні (Додаток А).

Розробка мобільного додатку буде здійснюватися у середовищі Android Studio [3]. Android Studio– це інтегроване середовище розробки під операційну систему андроїд, яка працює з мовою програмування Java.

Робота з 3D моделями пам’яток буде здійснюватися у програмі 3D max [4] фірми Autodesk. У подальшому моделі будуть інтегруватися у мобільний додаток.

База даних буде керуватися реляційною системою SQLite [5] призначеною для ОС андроїд.

2 МОДЕЛЮВАННЯ ІНТЕРАКТИВНОГО ДОДАТКУ

2.1 Структурно-функціональне моделювання роботи мобільного додатку

Робота мобільного додатку розпочинається з його запуску користувачем. Перед користувач відкривається головне меню, та постає вибір між двома функціями.

Перша функція - це карта. Її задача полягає в тому, щоб користувач мав можливість прокладати маршрут до пам’яткі, яка його цікавить. Для цього необхідно вибрати на карті пам’ятку, позначену маркером. Функція карта включає в себе перехід на функцію історичного опису.

Функція історичного опису представлена у вигляді списку назв пам’яток, з яких користувач обирає необхідну. Після вибору відкривається текст з історичним описом пам’ятки. На цій сторінці є перехід на функцію показу 3D моделей будівель.

Функція показу 3D моделей будівель дає можливість користувачеві переглянути модель з можливостю її обертання.

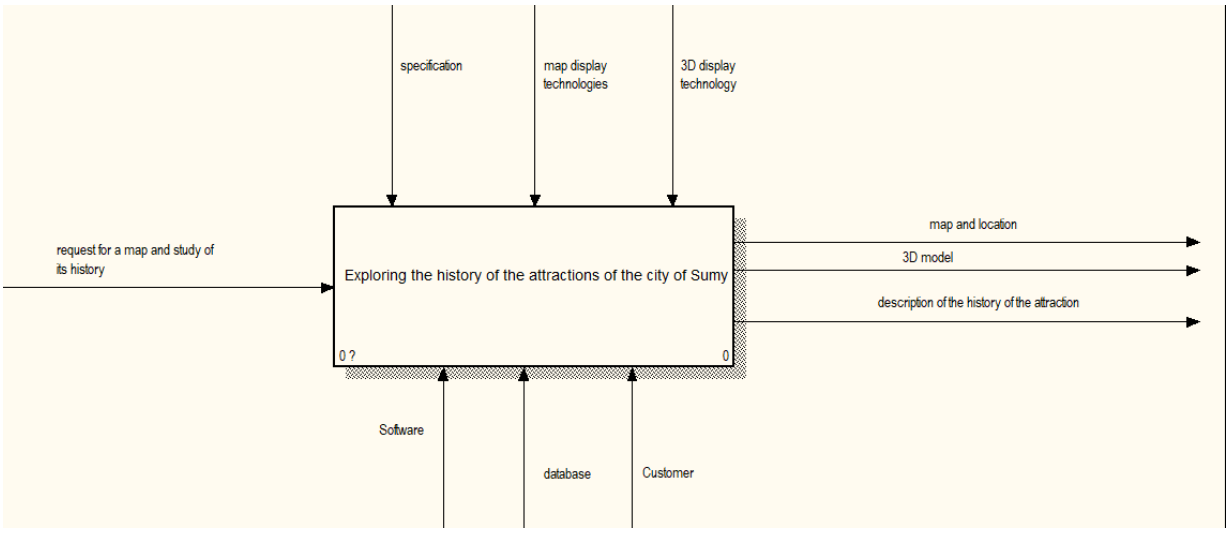

Рисунок 2.1 – Контекстна діаграма

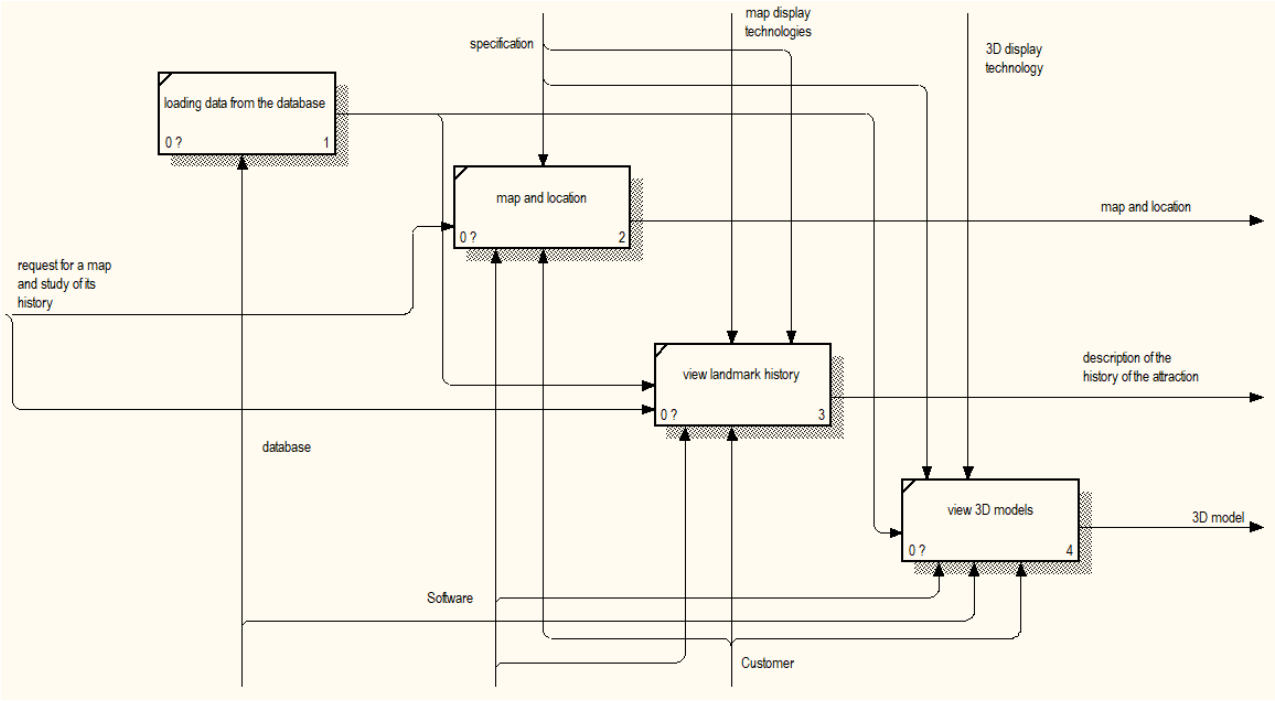

Рисунок 2.2 – Діаграма декомпозиції першого рівня

2.2 Проектування моделі бази даних

База даних додатку складається з двох таблиць та двох сутностей. Перша таблиця складається з файлів 3D моделей . Друга таблиця складається з текстового опису видатних пам’яток. Необхідності в нормалізації діаграм немає через простоту структури бази даних.

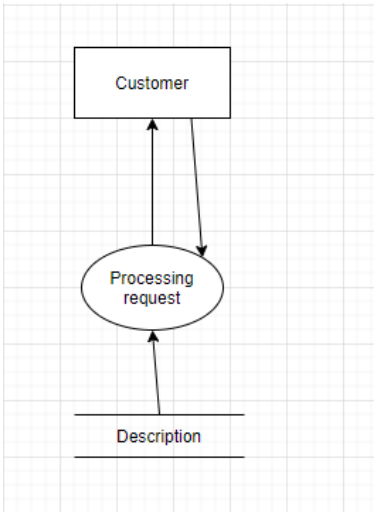

Рисунок 2.3 –DF діаграма

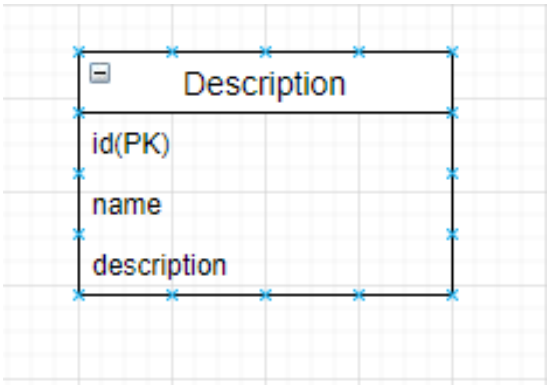

Рисунок 2.4 – ER діаграма

2.3 Моделювання варіантів використання

Для розуміння структури мобільного додатку було розроблено схему варіантів використання, яка представлена на рисунку.

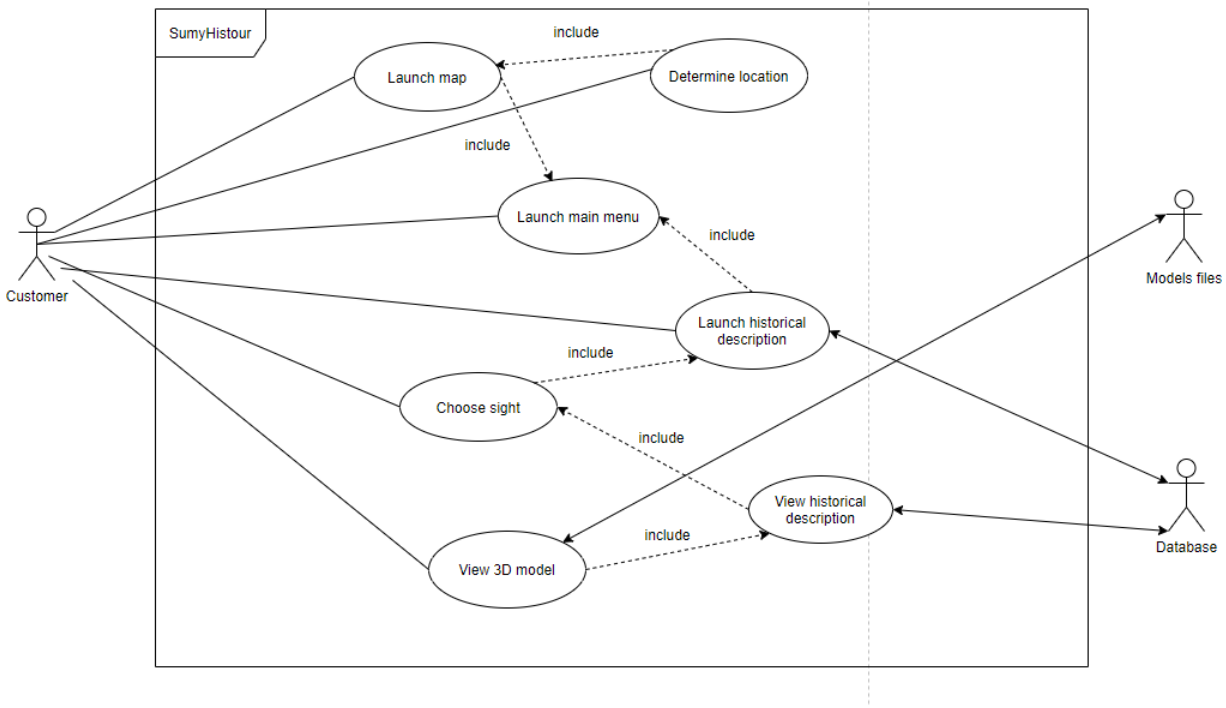

Рисунок 2.4 – Діаграма варіантів використання

На діаграмі представлено взаємодію користувача і додатка. Додаток має три основні функції:

1) Визначення місцеположення.
2) Можливість переглянути історичний опис пам’ятки
3) Можливість переглянути 3D моделі будівель. Ця функція є залежною, її відкрити тільки з функції опису.

3 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ВИВЧЕННЯ ІСТОРІЇ СУМЩИНИ

3.1 Розробка бази даних

Для створення бази даних у нашому мобільному додатку було використано бібліотеку SQLite. Для керування базою даних було створено інтерфейс, що програмно підключається до мобільного додатку, і який містить певні функції, що являють собою запрограмовані запити. Створений інтерфейс зображений на рисунку 3.1.

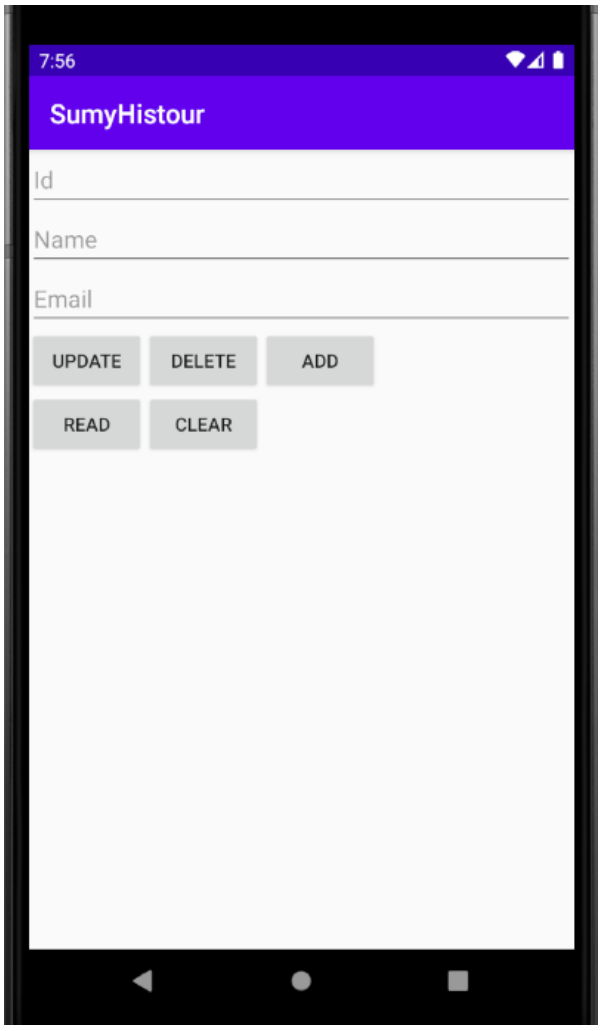

Рисунок 3.1 – Інтерфейс керування бази даних

Було створено клас DBHelper, який створює базу даних під назвою description.

База даних складається з однієї таблиці description. Таблиця description має три колонки. Перша колонка під назвою id являє собою первинний ключ та має тип даних INT.

Друга і третя колонки мають назви name і description та мають тип даних VARCHAR.

```
package com.example.sumyhistour;
import ...

public class DBHelper  extends SQLiteOpenHelper{

    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "description ";
    public static final String TABLE_CONTACTS = "description ";

    public static final String KEY_ID = "id";
    public static final String KEY_NAME = "name";
    public static final String KEY_MAIL = "description ";

    public DBHelper(Context context) { super(context, DATABASE_NAME,  factory: null, DATABASE_VERSION); }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create table " + TABLE_CONTACTS + "(" + KEY_ID
                + " integer primary key," + KEY_NAME + " text," + KEY_MAIL + " varchar" + ")");


    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("drop table if exists " + TABLE_CONTACTS);

        onCreate(db);

    }


}
```

Рисунок 3.2 – Скріншот коду класу DBHelper

Для взаємодії з базою даних були розроблені наступні запити та функції:

1) Створення таблиці: create table description (id integer primary key, name text, description varchar);

2) Видалення змісту таблиці: drop table if exists description;
3) Видалення певного рядку: database.delete();
4) Оновлення таблиці: database.update();
5) Зчитування з таблиці: database.query();
6) Додавання в таблицю: database.insert();

3.2 Програмна реалізація

Проєкт реалізований у вигляді мобільного додатку під платформу Android.

Структурно додаток має п’ять вікон – головне, вікно відображення карти, вікно з переліком пам’яток, вікно з інформацією про пам’ятку та вікно відображення триивимірної моделі. Переходи між вікнами показано на рисунку 3.3.

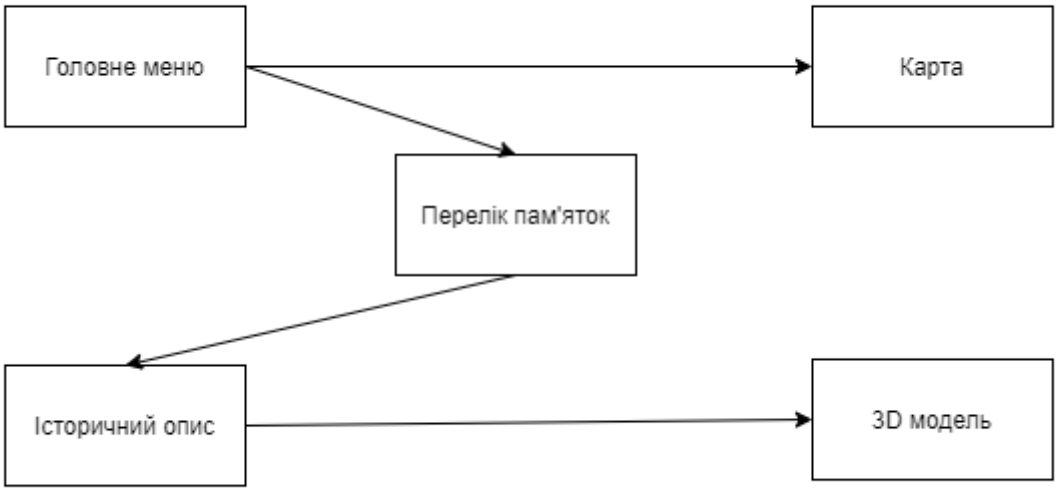

Рисунок 3.3 – Переходи між вікнами додатку

Створено головне меню, яке складається з кнопок (button) «Карта» і «Опис» та кнопок-зображень (image-button), завдяки яким здійснюється перехід на україномовний або англомовний інтерфейс, рисунок 3.4.

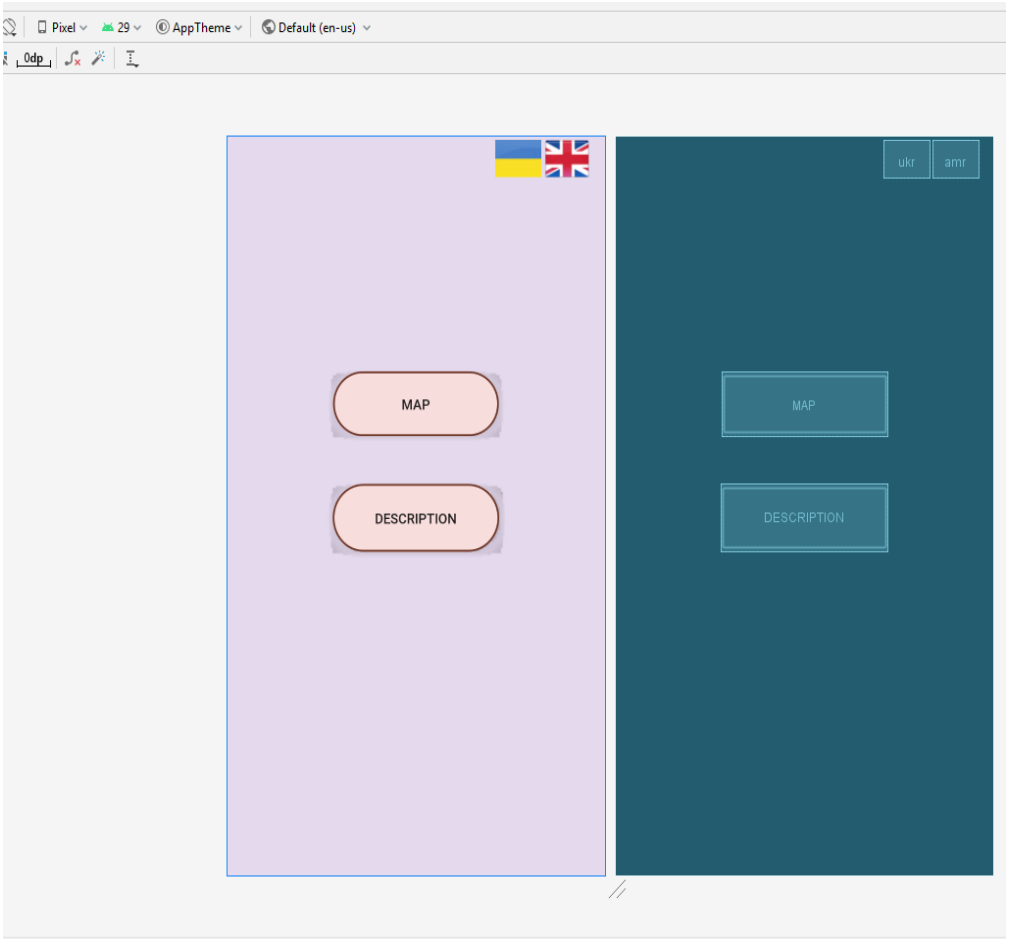

Рисунок 3.4 – Скріншот розробки вікна головного меню

Для переходу на інше вікно для кнопок були створені обробники подій в класі MainActivity.java.

Для переходу на іншомовний інтерфейс, було розроблено дві кнопки-зображення, новий клас і вікно. В клас MainActivity.java було додано обробник подій для цих кнопок, та умову, виконання якої дає змогу перейти на іншомовний інтерфейс. Скріншот розробки представлений на 3.5.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.andr
    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"
    android:background="#E5D9ED"

    tools:context=".MainActivity">

    <Button
        android:id="@+id/descrip2"
        android:layout_width="181dp"
        android:layout_height="67dp"
        android:layout_marginStart="162dp"
        android:layout_marginTop="354dp"
        android:layout_marginEnd="162dp"
        android:layout_marginBottom="330dp"
        android:background="@drawable/button_states"

        android:text="Опис"

        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/route2"
        android:layout_width="180dp"
        android:layout_height="64dp"
        android:layout_marginStart="162dp"
        android:layout_marginTop="231dp"
        android:layout_marginEnd="162dp"
        android:layout_marginBottom="432dp"
        android:background="@drawable/button_states"
        android:text="Карта"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageButton
```

Рисунок 3.5 – Розробка іншомовного інтерфейсу

Елемент меню «Карта» відкриває вікно з гугл-картою та кнопкою пошуку місцерозташування пристрою. Після натискання на кнопку, з'являється маркер на карті, який позначає місцерозташування користувача. Дане вікно було створено за допомогою вбудованого шаблону Google Maps Activity. Шаблон включає в себе

клас підтримки карти і xml документ, що відображає карту. Додавання шаблону представленно на рисунку 3.6.

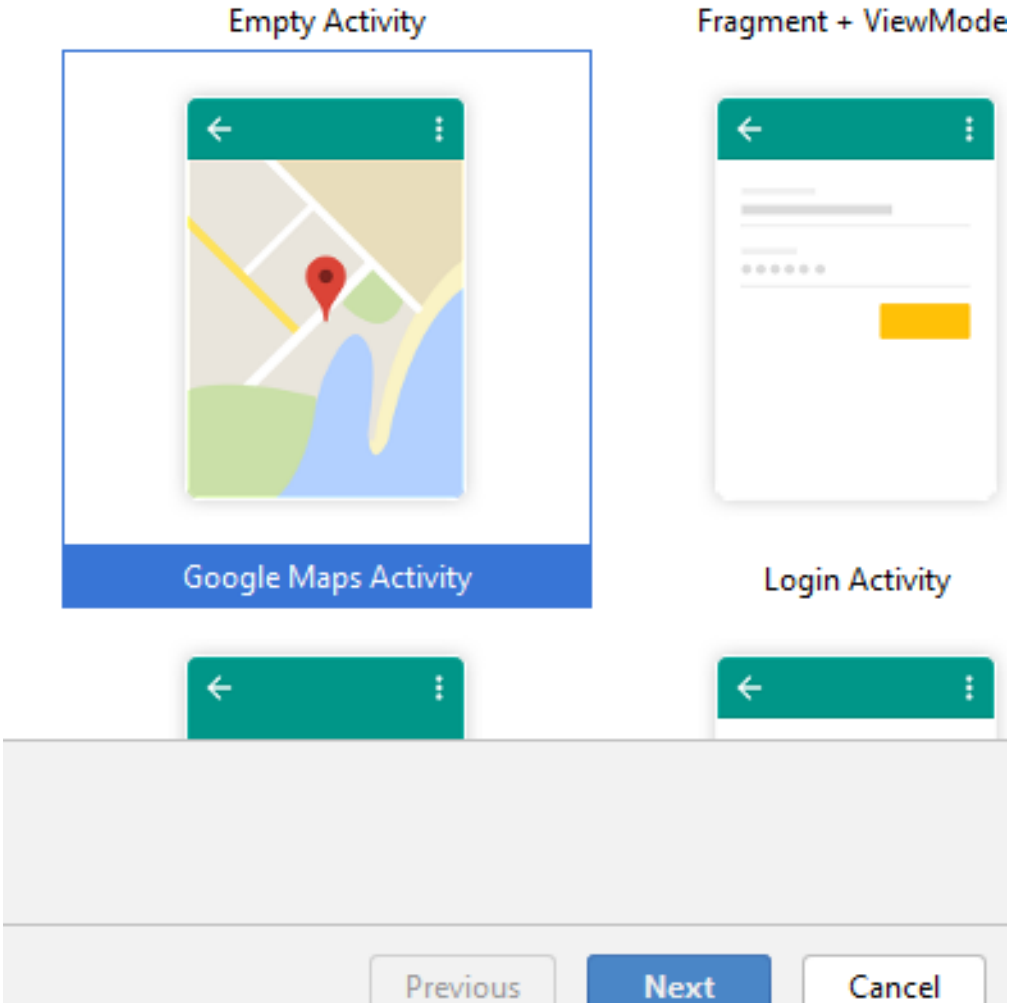

Рисунок 3.6 – Створення вікна активності з картою

Для правильної роботи карт необхідно було підключити необхідну бібліотеку «com.google.android.gms: play-services-maps: 17.0.0» у файлі build.gradle в методі залежностей. Ця бібліотека дозволяє користуватися фунціями, необхідними для створення карти та роботи з нею. Необхідні залежності показані на рисунку 3.7.

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 29
    buildToolsVersion "29.0.3"

    defaultConfig {
        applicationId "com.example.sumyhistour"
        minSdkVersion 21
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility = 1.8
        targetCompatibility = 1.8
    }

}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])

    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'com.google.android.gms:play-services-maps:17.0.0'
    testImplementation 'junit:junit:4.12'
    play-services-measurement-base versions="15.0.0,15.0.2"
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
    implementation project(path: ':app:engine')
}
```

Рисунок 3.7 – Створення вікна активності з картою

За замовчуванням було створено клас MapActivity.java, що забезпечує роботу с картою та xml файл activity_maps.xml, завдяки якому карта відображається у вікні.

Клас MapActivity.java передає інформацію про карту у вікно activity_maps.xml, де карта відображається через елемент вікна fragment.

Для роботи карти було використано такі бібліотеки, як: GoogleMap, SupportMapFragment.

Для завантаження карти, за замовчуванням, було створено функцію OnCreate(), де була створена змінна mapFragment типу даних SupportMapFragment [11]. З допомогою цієї змінної клас пов’язується з вікном інтерфейсу.

Також в класі було створено функцію getLocation() та OnGPS(). Функція getLocation() визначає координати місцезнаходження користувача. Для її роботи було підключено бібліотеку LatLng [12], що визначає довготу і широту.

Функція OnGPS() перевіряє, чи є включеною система навігації у смартфоні, якщо ні, то вмикає її.

У класі було створено наступні змінні: mMap, locationManager, latitude, longitude. Змінна mMap є об’єктом класу GoogleMap [13], в якому зберігається інформація про саму карту.

Змінна locationManager – це об’єкт класу LocationManager [14], який дозволяє дізнатися геолокацію пристрою. Змінні latitude і longitude мають тип даних double, та зберігають в собі координати широти і довготи.

У вікні карти передбачена кнопка визначення місцезнаходження, яка обробляється методом OnClick(). Метод OnClick() знаходиться у методі OnMapReady і включає в себе функції getLocation() та OnGPS().

У вікні activity_maps.xml було створено два елемента: фрагмент і кнопку, рисунок 3.8-3.9.

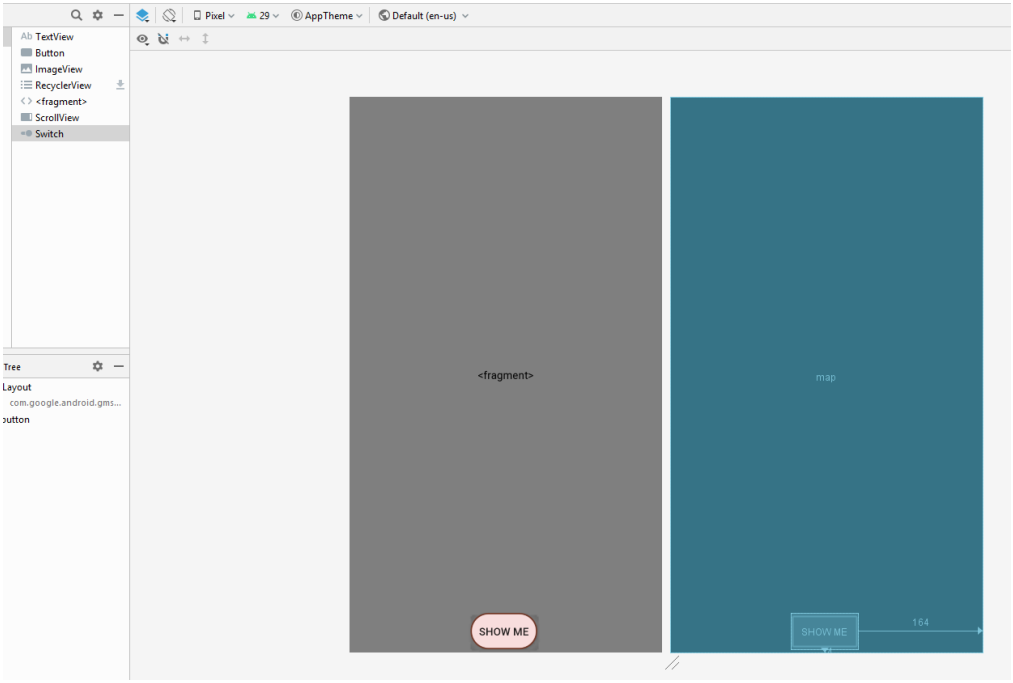

Рисунок 3.8 – Розробка вікна інтерфейсу «Карта»

```xml
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <fragment
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MapsActivity" />

    <Button
        android:id="@+id/loc_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginEnd="164dp"
        android:layout_marginBottom="4dp"
        android:background="@drawable/button_states"
        tools:text="Show me" />

</RelativeLayout>
```

Рисунок 3.9 – Розробка вікна інтерфейсу «Карта»

Елемент меню «Опис» відкриває вікно зі списком пам'яток, який зчитується з бази даних, де кожен елемент списку (ListView) представлений у вигляді кнопки. Після натискання на певну пам'ятку, відкривається нове вікно з історичним описом цієї пам'ятки. Створення елементу списку зображенно на рисунку 3.10.

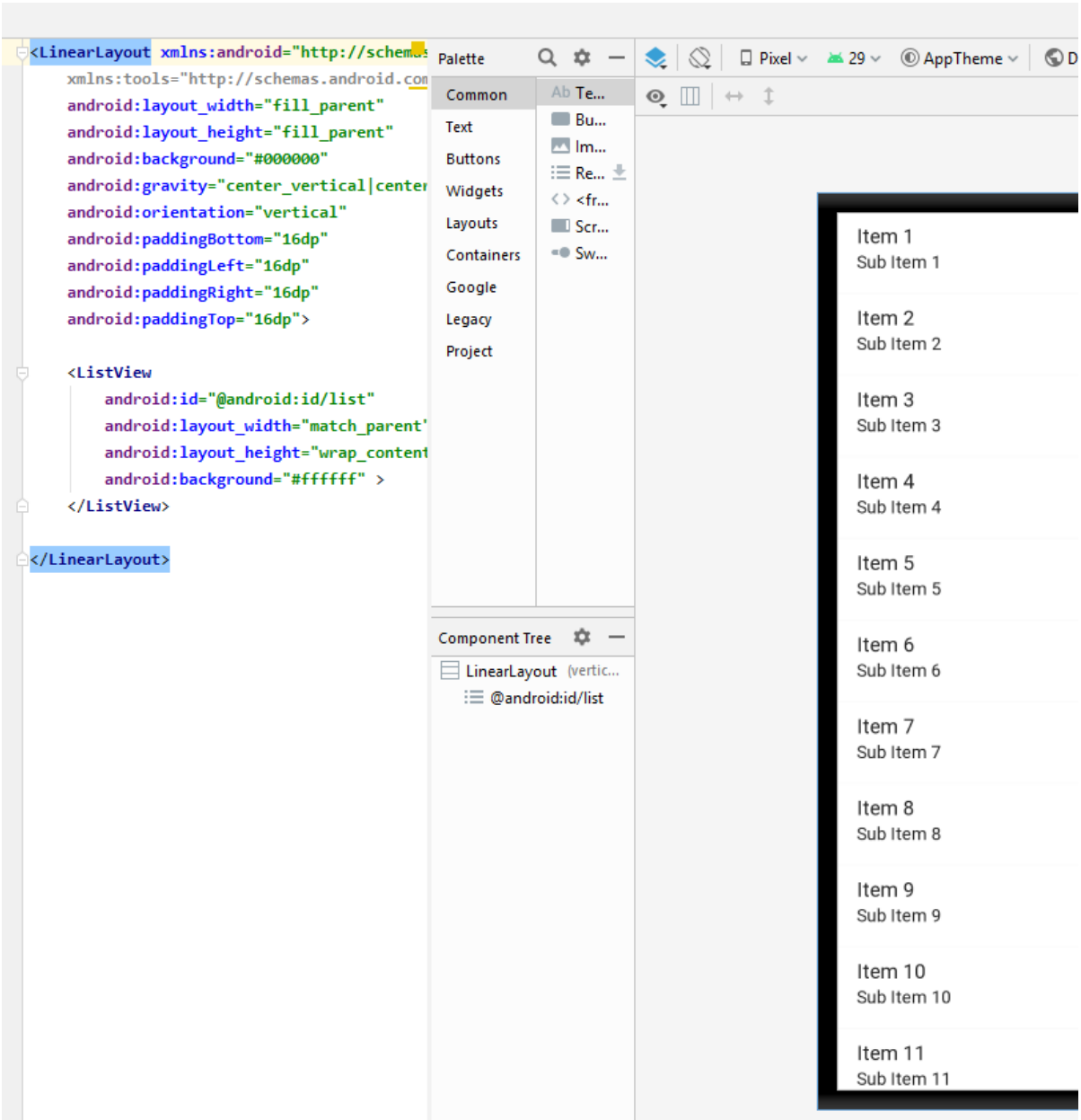

Рисунок 3.10 – Розробка вікна інтерфейсу «Карта»

Для створення бази даних було додано клас DBHelper.java, у якому прописані запити на створення файлу з базою даних.

Для роботи бази даних була підключена бібліотека SQLiteDatabase.

Для вікна «Опис» було створено клас DescrU.java та xml файл activity_descr_u2.xml.

У класі DescrU.java було програмно реалізовано запит виводу назв пам’яток з бази даних у список, який передається у вікно activity_descr_u2.xml.

Для виводу списку на екран було створено елемент ListView, який зображено на рисунках 3.11-3.12.

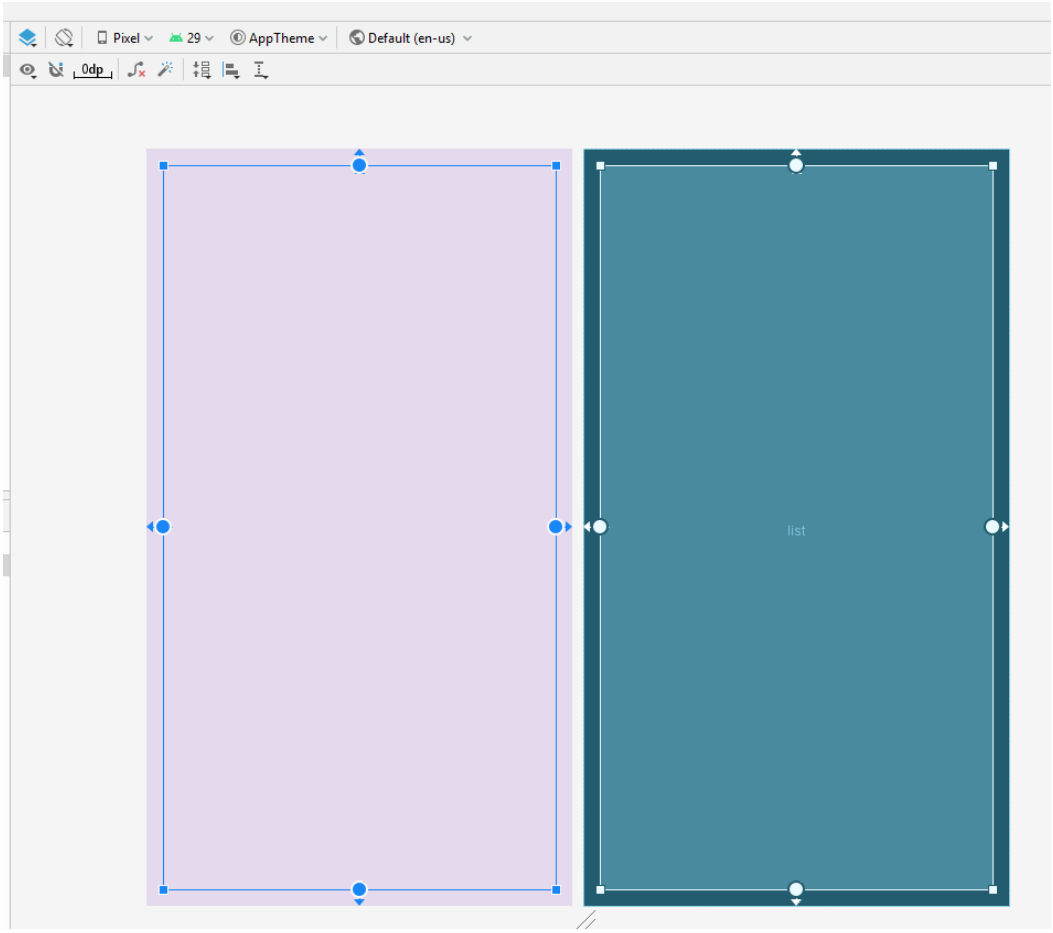

Рисунок 3.11 – Розробки вікна перегляду списку пам’яток

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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"
    android:background="#E5D9ED"
    tools:context=".MainActivity">

    <ListView
        android:id="@+id/list"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="16dp"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:layout_marginRight="16dp"
        android:layout_marginBottom="16dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Рисунок 3.12 – Скріншот розробки вікна перегляду списку пам’яток

Вікно з історичним описом пам'ятки містить сам історичний опис (який реалізований елементом textview) та кнопку «View 3D». При розробці даного вікна було створено клас DescrU2.java і xml файл activity_descr_u2.xml. У класі DescrU2.java був програмно реалізований запит на вивід ім’я і опису обраної пам’ятки.

Для відображення тексту з бази даних, у вікні activity_descr_u2.xml було використано елемент textview. Для textview було обрано шрифт «textAppearanceListItemSmall», стиль шрифта «bold», вирівнювання по центру. У стилях textview в якості параметра висоти було обрано wrap_content, це означає що висота елементу регулюється довжиною тексту. У вікні була додано кнопку запуску перегляду 3D моделі і фонове зображення. Дане вікно зображено на рисунках 3.13-3.14.

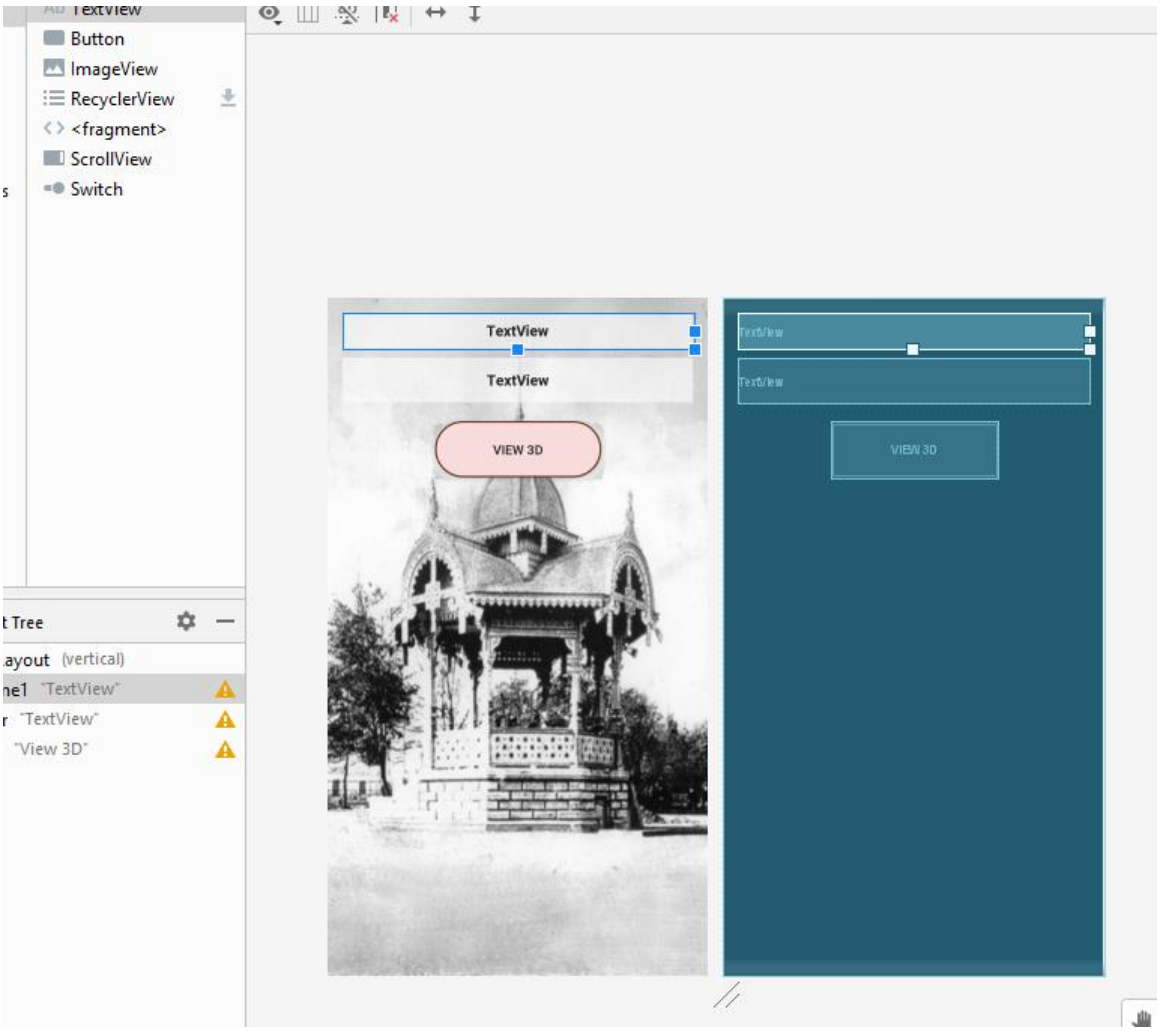

Рисунок 3.13 – Розробка вікна опису пам’ятки

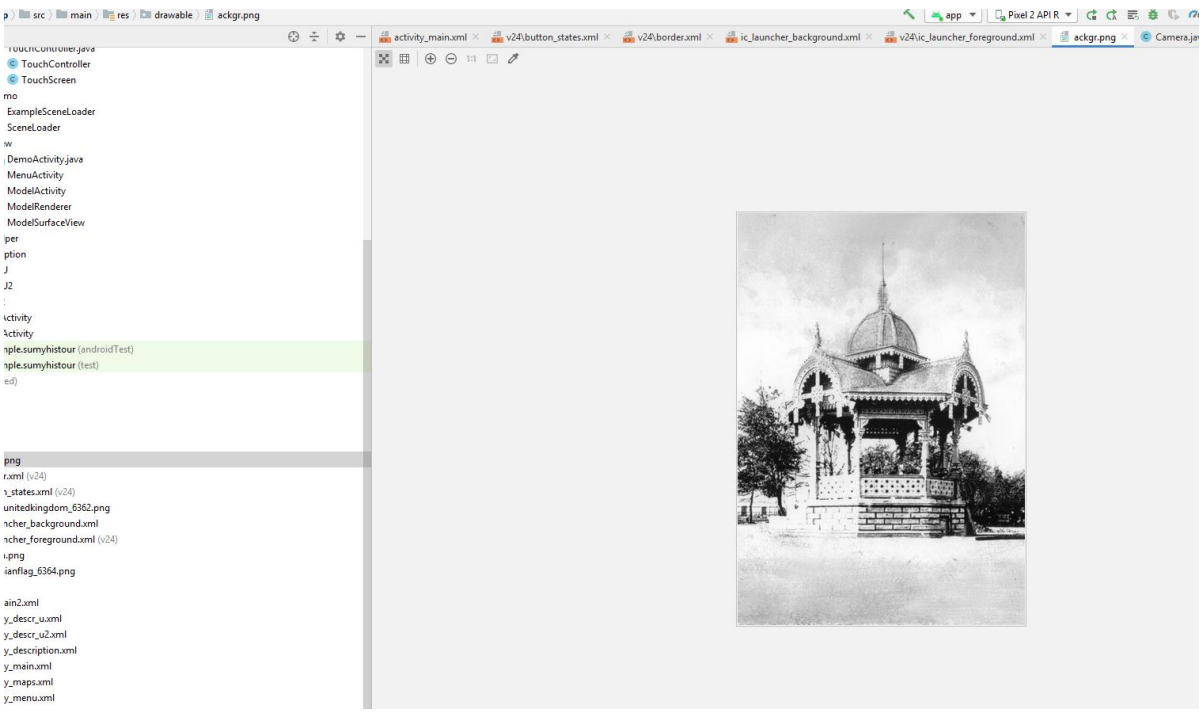

Рисунок 3.14 – Розробка вікна опису пам’ятки

Вікно перегляду 3D моделі було реалізовно за допомогою OpenGl. OpenGl завантажує файл моделі, аналізує та будує цю модель. Для його роботи була підключена бібліотека GLU [15] та створено відповідні класи, які представленні на рисунку 3.15.

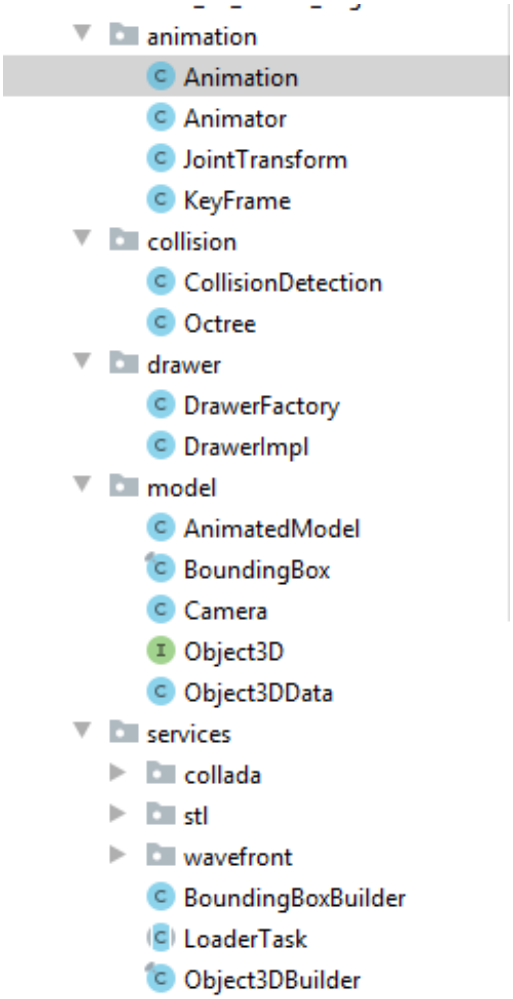

Рисунок 3.15 – Дерево класів

Для завантаження 3D моделі було створено клас MenuActivity.java, у якому відбувається пошук файлу за ідентифікатором пам’ятки з бази даних. Для моделі обрано формат файлу obj, тому що він є універсальним і підтримується більшістю програм, орієнтованих на роботу з 3D графікою.

Оскільки спочатку 3D моделі мали файли формату *.max, їх необхідно було екпортувати у потрібний формат obj для можливості роботи з додатком. Для цього

моделі пам’яток [16-19] завантажені у програму 3d Max та за допомогою функції експорту приведені до потрібного формату obj.

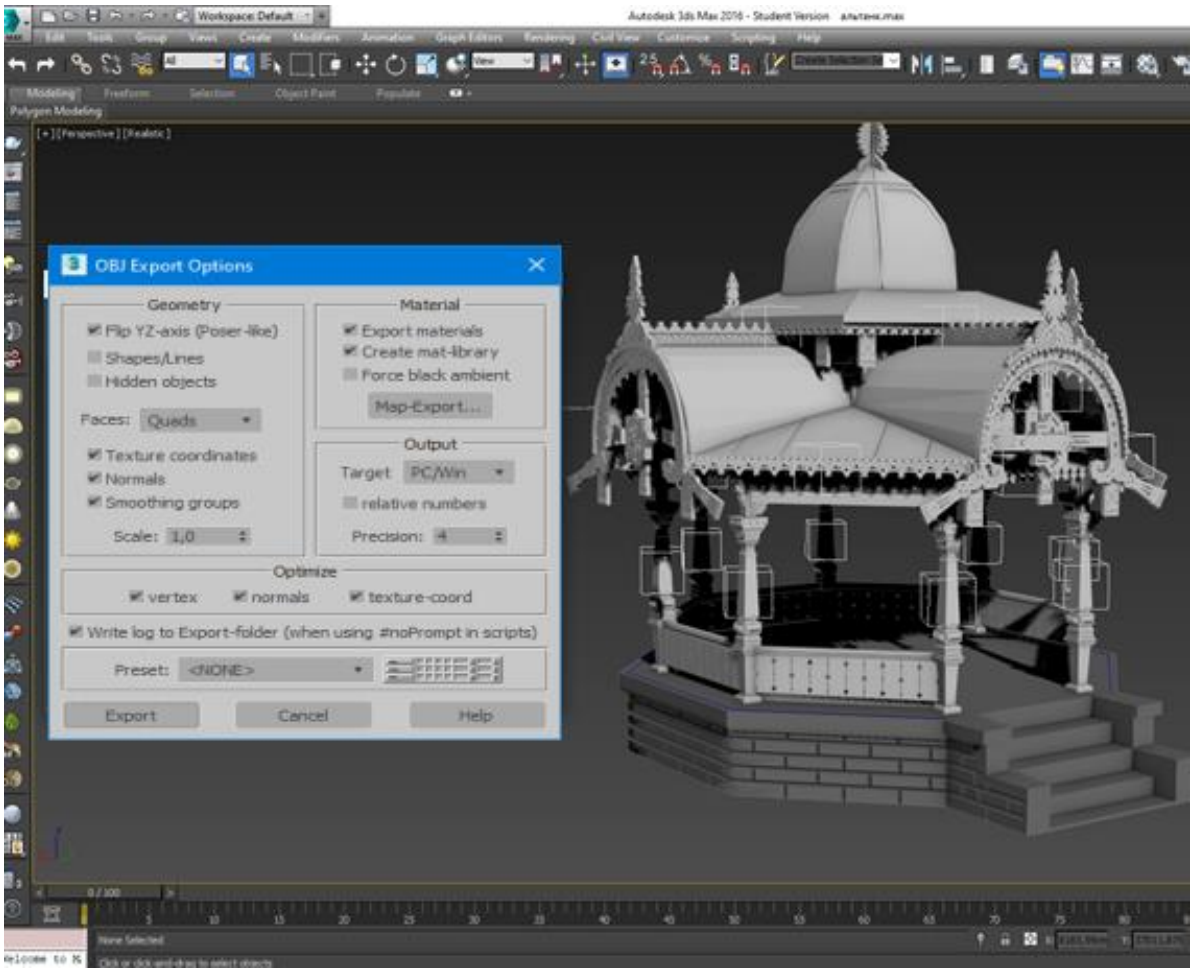

Рисунок 3.16 – Експорт моделі у формат obj

Для відображення 3D моделі було створено класи ModelActivity.java, ModelRenderer, ModelSurfaceView, у яких прораховується як буде виглядати модель, та потім будується.

Було створено класс сцени SceneLoader.java, у якому було підключено класи специфікації OpenGl.

У OpenGl реалізовані класи: освітлення, камери, обертання моделі. Обертання моделі реалізовано завдяки бібліотеці android.opengl.Matrix [20].

У моделі є координати місцерозташування в сцені. Вони змінюються залежно від вектору та інтенсивності обертання моделі користувачем і

записуються в матрицю, яка відтворює в онлайн режимі нове розташування моделі.

Для зчитування взаємодіїї користувача з моделлю, було створено клас-контроллер TouchController.java, у якому застосовано такі бібліотеки як: android.util.Log [21] – бібліотека для взаємодіїї користувача з програмою, android.view.View [22] – бібліотека об’єктів контейнерів які реагують на діїї користувача, android.widget.ImageView [23] – бібліотека контейнерів для перегляду зображень, android.opengl.Matrix – бібліотека об’єктів для відтворення віртуального тривимірного простору.

У класі Camera.java були застосовані бібліотеки android.util.Log та android.opengl.Matrix. У класі освітлення були застосовані бібліотеки android.util.Log та java.lang.reflect.Field [24]. Бібліотека java.lang.reflect.Field надає можливість дослідження даних програми.

Для дизайну додатку було створено xml файл-стиль.

Файл button_states.xml - це стиль для кнопок, у якому створена рамка і заданий колір кнопки. Даний файл представлений на рисунку 3.17.

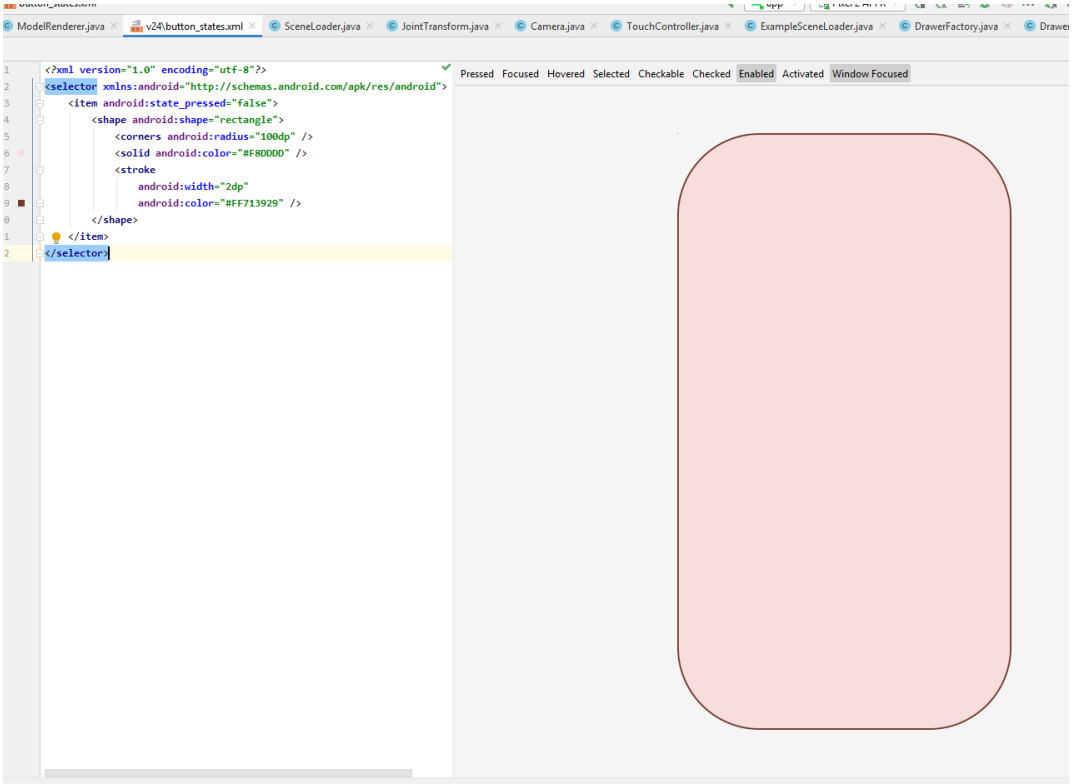

Рисунок 3.17 – Файл стилів кнопок

Коди всіх класів проєкту представлені у додатку В.

3.3 Використання мобільного додатку

Демонстрація роботи мобільного додатку проведена на емуляторі Pixel 2 API R у середовищі Android Studio:

1) Запуск мобільного додатку. Демонстрація головного меню, на якому зображено кнопки «Опис» і «Карта», та кнопки-зображення, які дають можливість перейти на україномовний або англомовний інтерфейс:

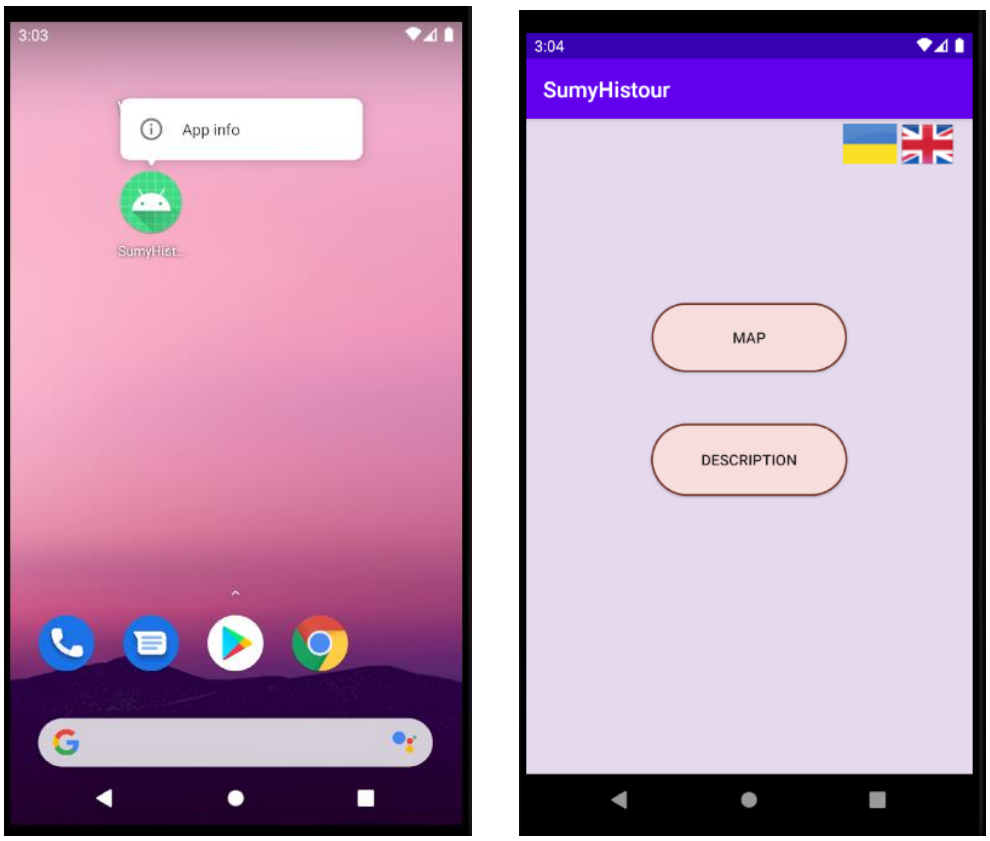

Рисунок 3.18 – Запуск мобільного додатку з емулятору та головне меню

2) Далі натиснемо на кнопку «Map», для демонстрації роботи вікна з картою та функцією пошуку місцерозташування:

Рисунок 3.19 – Демонстрація функції «Карта»

3) Натиснемо кнопку на карті, що визначить місцерозташування користувача та збільшить масштаб карти. На карті бачимо де знаходиться користувач та історичні пам’ятки:

Рисунок 3.20 – Визначення місцерозташування користувача

4) Далі повернемося на головне меню та відкриємо функцію «Опис». Відкриється список пам’яток, зчитаний з бази даних, і реалізований у вигляді кнопок:

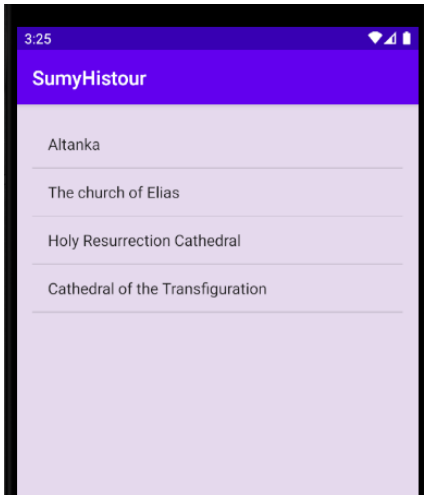

Рисунок 3.21 – Список пам’яток

5) Далі виберемо певну пам’ятку та натиснемо на відповідну кнопку. Відкриється історичний опис з можливістю перегляду 3D моделі. У якості прикладу візьмемо Альтанку:

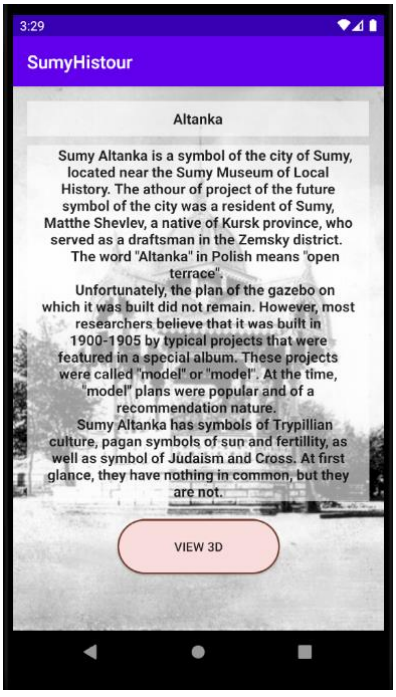

Рисунок 3.22 – Демонстрація функції опису

6) Наступним натиснемо на кнопку «View 3D», щоб продивитися 3D модель Альтанки. Модель можна обертати:

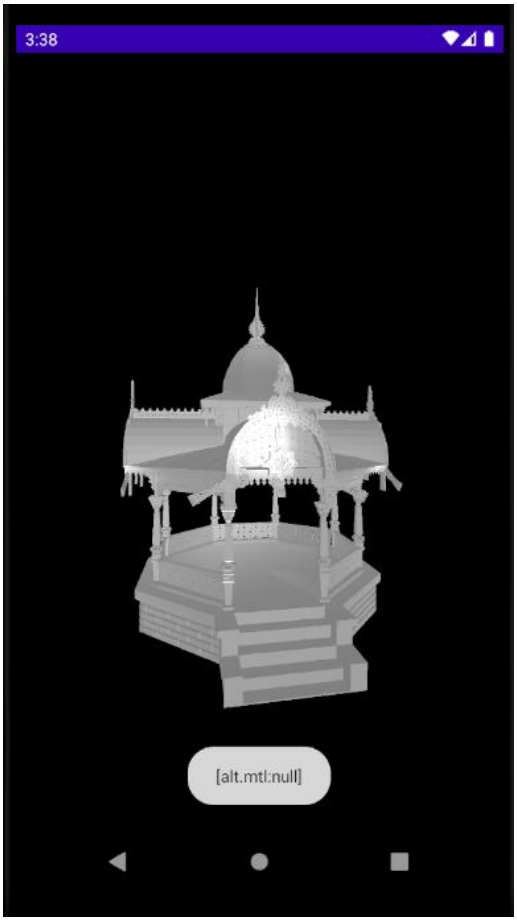

Рисунок 3.23 – Показ 3D моделі

7) Повернемося у головне меню, для демонстрації переходу інтерфейсу на українську мову. Натиснемо на флаг України, щоб змінити мову інтерфейсу:

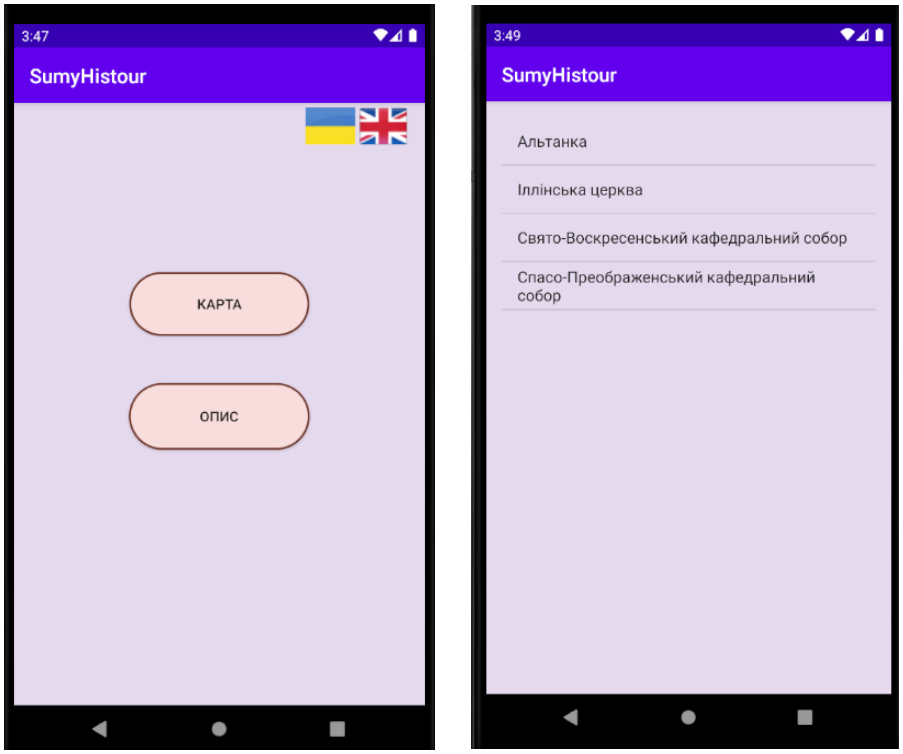

Рисунок 3.24 – Демонстрація інтерфейсу на українській мові

Рисунок 3.25 – Демонстрація інтерфейсу на українській мові

При емуляції роботи мобільного додатку не було виявлено ніяких помилок.

ВИСНОВКИ

У результаті виконання дипломного проекту було створено мобільний додаток для вивчення історії пам’яток Сумщини.

Було досліджено предметну область розроблюваного мобільного додатку, поставлені чіткі задачі для його виконання. Проведено аналіз існуючих аналогів, що показало, що конкретно такого мобільного додатку на ринку немає.

Було розроблено структурно-функціональну модель додатку, по якій успішно була проведена розробка.

Було вирішено наступні задачі:

- Формулювання вимог для розробки мобільного додатку.
- Було проведено планування робіт, побудова діаграми Ганта. Також було проведено дослідження на можливі ризики у ході виконання проектної роботи. Було створено матрицю ризиків та план їх усунення у разі виникнення.

Були створені всі необхідні діаграми, що показують структуру і архітектуру мобільного додатку, по яким була проведена успішна реалізація.

Створений мобільний додаток дає можливість ознайомитись з історичними пам’ятками міста Суми, подивитися їх розташування на карті і визначити де знаходиться користувач та продемонструвати 3D модель пам’ятки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інформація про додаток “Regido”. URL: https://apps.apple.com/ru/app/redigo/id439329970 (дата звернення: 20.04.2020).

2. Інформація про додаток “Momondo Places”. URL: https://www.momondo.ua/ (дата звернення: 20.04.2020).

3. Середовище розробки мобільних додатків. “Android Studio”. URL: https://developer.android.com/studio (дата звернення: 20.04.2020).

4. Програмний продукт для розробки 3D моделей. “3D max Autodesk”. URL: https://www.autodesk.ru/products/3ds-max (дата звернення: 20.04.2020).

5. Реляційна система для мобільних додатків. “SQLite”. URL: https://www.sqlite.org/index.html (дата звернення: 20.04.2020).

6. Використання баз даних на платформи андроїд. URL: https://www.internet-technologies.ru/articles/ispolzovanie-prostoy-bazy-dannyh-sqlite-v-android-prilozhenii.html (дата звернення: 20.04.2020).

7. Використання баз даних на платформи андроїд. URL: https://habr.com/ru/company/microsoft/blog/229041 (дата звернення: 20.04.2020).

8. Ресурс для створення діаграм. “Draw.io”. URL: https://app.diagrams.net/ (дата звернення: 20.04.2020).

9. Методичні вказівки. “Загальні положення та правила складання. ДСТУ 8302:2015r”. URL: https://drive.google.com/drive/u/0/folders/1GV (дата звернення: 20.04.2020).

10. UML для диплома. “Використання UML моделей для проектування ІС”. URL: https://drive.google.com/drive/u/0/folders/1GV (дата звернення: 20.04.2020).

11. Опис бібліотеки SupportMapFragment. URL: https://developers.google.com/android/reference/com/google/android/gms/maps/SupportMapFragment (дата звернення: 20.04.2020).

12. Опис бібліотеки LatLng. URL: https://developers.google.com/maps/documentation/android-sdk/reference/com/google/android/libraries/maps/model/LatLng (дата звернення: 20.04.2020).

13. Опис бібліотеки GoogleMap. URL: https://developers.google.com/maps/documentation/android-sdk/map?hl=ru (дата звернення: 20.04.2020).

14. Опис бібліотеки LocationManager. URL: https://developer.android.com/reference/android/location/LocationManager (дата звернення: 20.04.2020).

15. Опис бібліотеки GLU. URL: https://coderlessons.com/tutorials/java-tekhnologii/vyuchit-jogl/jogl-kratkoe-rukovodstvo (дата звернення: 20.04.2020).

16. Любивий Ю.О. Візуалізація 3D-моделі Спасо-Преображенського кафедрального собору. [Текст]: кваліфікаційна робота бакалавра; напрям підготовки 6.050101 – Комп'ютерні науки / Ю.О.Любивий; наук. керівник І.В. Баранова. - Суми: СумДУ, 2018. - 91 с.

17. Войцеховський Я.С. Візуалізація 3D-моделі культової споруди. [Текст]: Комплексний курсовий проект; напрям підготовки 6.050101 – Комп'ютерні науки / Я.С.Войцеховський; наук. керівник І.В. Баранова. - Суми: СумДУ, 2015. - 51 с.

18. Сурган О.О. Візуалізація 3D-моделі Пантелеймонівської церкви [Текст]: Дипломний проект; напрям підготовки 6.050101 – Комп'ютерні науки / О.О.Сурган; наук. керівник І.В. Баранова. - Суми: СумДУ, 2018. - 63 с.

19. Прахов К.О. Візуалізація 3D-моделі туристичної пам'ятки «Альтанка» [Текст]: Дипломний проект; напрям підготовки 6.050101 – Комп'ютерні науки / К.О.Прахов; наук. керівник І.В. Баранова. - Суми: СумДУ, 2017. - 53 с.

20. Опис бібліотеки Matrix. URL: https://developer.android.com/reference/android/opengl/Matrix (дата звернення: 20.04.2020).

21. Опис бібліотеки Log. URL: https://developer.android.com/reference/android/util/Log (дата звернення: 20.04.2020).

22. Опис бібліотеки View. URL: https://developer.android.com/reference/android/view/View (дата звернення: 20.04.2020).

23. Опис бібліотеки ImageView. URL: http://developer.alexanderklimov.ru/android/views/imageview (дата звернення: 20.04.2020).

24. Опис бібліотеки JavaReflect. URL: https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/Field.html (дата звернення: 20.04.2020).

ДОДАТОК А.
ТЕХНІЧНЕ ЗАВДАННЯ

на розробку інтерактивного додатку для вивчення історії Сумщини

Суми 2020

1 Призначення й мета створення інтерактивного додатку

1.1 Призначення інтерактивного додатку

Інтерактивний додаток призначений для вивчення історії Сумщини.

1.2Мета створення інтерактивного додатку

Підтримка екскурсійної привабливості Сумщини та полегшення роботи екскурсовода. Забезпечення можливості зручного пошуку визначних пам’яток та ознайомлення з ними.

1.3 Цільова аудиторія

У цільовій аудиторії інтерактивного додатку можна виділити наступні групи:

- Туристи міста Суми.
- Мешканці міста Суми.
- Органи керування міста Суми.
- Інші зацікавлені відвідувачі.

2 Вимоги до інформаційної системи

2.1 Вимоги до інформаційної системи в цілому

Проект повинен бути реалізований у вигляді мобільного додатку під назвою “SumyHistour”. Мобільний додаток повинен складатися із взаємозалежних розділів із чітко розділеними функціями.

Для користуванням мобільним додатком ніяких спеціальних навичок не потрібно, користування інтерфейсом буде інтуїтивно зрозумілим.

У додатку не буде системи облікових записів, ним можна буде користуватись без реєстрації. Тому нема потреби у резервному копіювання даних. Позиція користувача визначається за допомогою технології GPS.

2.2 Вимоги до функцій, виконуваних мобільним додатком

Структура додатку

Головне меню мобільного додатку складається з наступних функцій:

- карта (на якій відображено місце розташування користувача, місце розташування пам'яток (найвідоміших) зазначених маркерами).

- віртуальний перегляд пам'яток, супроводжуваний описом історії споруди; функція має вигляд списку із пам’яток (також ця функція відкривається через карту).

Навігація

Користувацький інтерфейс додатку повинен забезпечувати наочне, інтуїтивно зрозуміле представлення структури розміщеної на ньому інформації, швидкий і логічний перехід з однієї функції на іншу. Навігаційні елементи повинні забезпечувати однозначне розуміння користувачем їх змісту.

Наповнення додатку (контент)

Сторінки всіх функцій мобільного додатку повинні бути реалізовані програмним шляхом. Інформація про пам’ятки та файли моделей повинні знаходитись у базі даних. Карта повинна бути підключена за допомогою бібліотеки.

Система навігації (карта додатку)

Взаємозв'язок між розділами додатку представлено на рисунку 1.

Рисунок 1– Карта додатку

2.3 Вимоги до функціональних можливостей

Функціональні можливості розділів

Карта: на карті можна прокласти маршрут від поточного місцезнаходження до вибраної пам’ятки; з карти можна відкрити функцію опису.

Опис: в функції опису буде знаходитися список видатних пам’яток міста Суми; при натисканні на одну з пам’яток відкриється її історичний опис та в ньому функція перегляду 3d-моделі, яку можна обертати для детального перегляду.

Загальні вимоги

Стиль мобільного додатку повинен бути простим та сучасним. Інтерфейс має бути гармонійних кольорів, щоб користувачеві було зручно користуватися ним. Розташування функцій у головному меню схематично показано на рис. 2, 3.

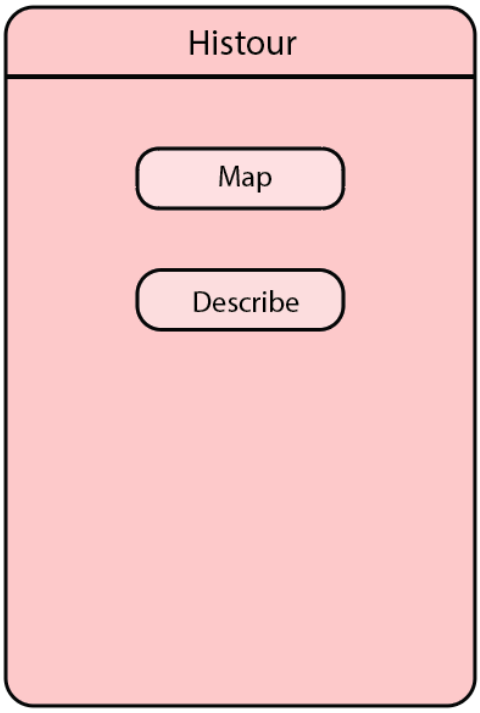

Рисунок 2 – Головне меню

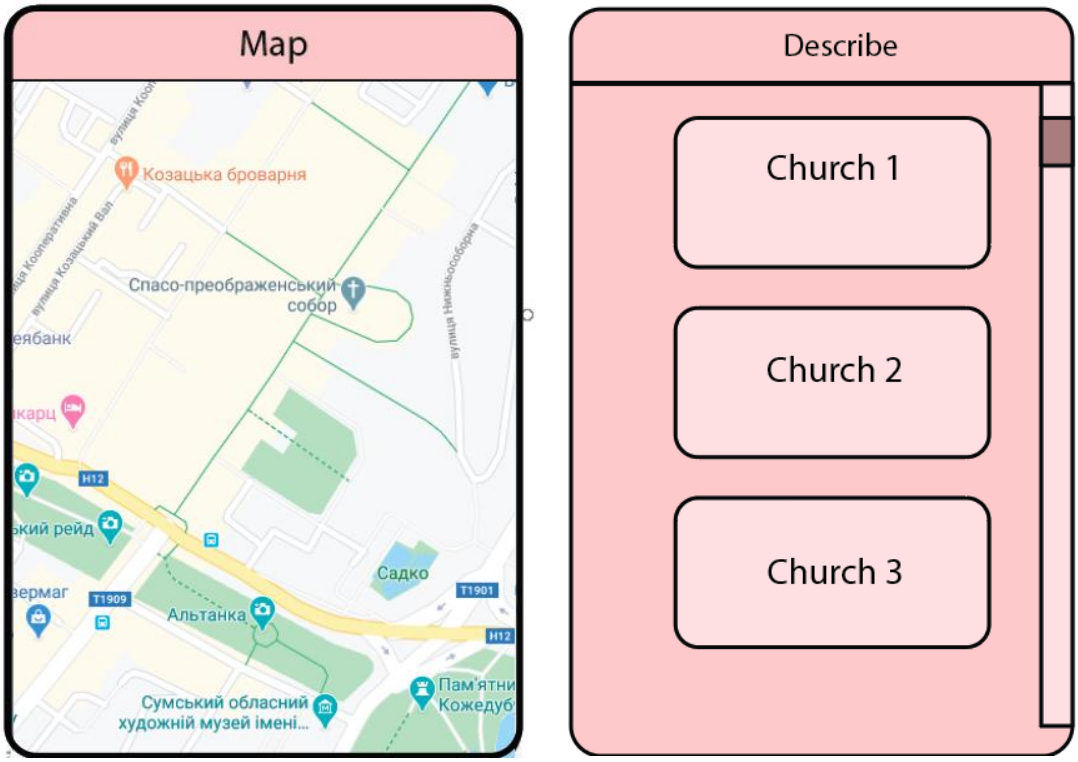

Рисунок 3 – Функція карти та опису

2.4 Вимоги до видів забезпечення

Вимоги до інформаційного забезпечення

Реалізація додатку відбувається з використанням:

- Java
- Google Maps
- SQLite

Вимоги до лінгвістичного забезпечення

Інтерфейс мобільного додатку повинен бути англійською та українською мовами.

Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

– Android 6.0 та вище;
– GPS.

Вимоги до апаратного забезпечення

Апаратне забезпечення на пристрої клієнта повинне забезпечувати підтримку програмного забезпечення, зазначеного в п. 2.4.

3 Склад і зміст робіт зі створення додатку

Докладний опис етапів роботи зі створення мобільного додатку наведено в табл. 1.

Таблиця 1 – Етапи створення додатку

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Проектування інтерфейсу за допомогою xml	5 день
2	Функція «карта»: Застосування google maps для реалізації карти в додатку	6 днів
3	Створення функції пошуку місцерозташування користувача	20 днів
4	Створення меню історичних пам’яток.	5 дні
5	Створення сцени перегляду 3D моделі пам’ятки	3 дні
6	Заповнення бази даних та підключення файлів моделей	3 дні
7	Завершення розробки: Проведення виправлень додатку, тестування реалізованого функціоналу	4 дні
8	Демонстрування роботи мобільного додатку	1 день
	Загальна тривалість робіт (з урахуванням резервного строку на налагодження й виправлення помилок) і строк закінчення проекту	47

4 Вимоги до складу й змісту робіт із введення додатку в експлуатацію

Для створення функціоналу додатку та його експлуатації необхідно дотримуватися вимог в вищенаведених розділах ТЗ.

ДОДАТОК Б.
ПЛАНУВАННЯ РОБІТ

Б.1 Ідентифікація ідеї проекту

Метою інтерактивного додатку дипломного проекту є полегшення пошуку по місту історичних пам’яток Сумщини та ознайомлення з ними.

Дипломний проект призначений для створення інтерактивного додатку для вивчення історії пам’яток .

Інтерактивний додаток має бути представлений у вигляді мобільного додатку з певним набором взаємопов’язаних функцій.

Б.2 Деталізація мети методом SMART

Конкретна (Specific). Створення мобільного додатку з певним набором функцій для полегшення пошуку визначних пам’яток та ознайомлення з ними.

Вимірювана (Measurable). Залучити, якомога більше туристів та мешканців міста Суми користуватися мобільним додатком, вивчаючи історію пам’яток.

Досяжна (Achievable). Для досягнення поставленої мети є всі необхідні навички та ресурси.

Реалістична (Relevant). Створений додаток допоможе туристам з пошуком історичних пам’яток та ознайомлення з ними, також він буде корисним для мешканців які не знають історію про ці пам’ятки.

Обмежена у часі (Time—framed). Ціль має часове обмеження, яке визначається замовником. В даному випадку умовним замовником є університет.

Б.3 Описання фази розробки ІТ—проекту

Б.3.1 Планування змісту структури робіт ІТ—проекту (WBS)

Ієрархічна структура робіт (Work Breakdown Structure) — це розбиття проекту на послідовні ієрархічні етапи, які мають бути виконані за фіксований термін для досягнення поставленої мети .

Головний принцип побудови якісної WBS — декомпозиція. Декомпозиція — це розбиття основних етапів проекту на дрібніші. Чим краще WBS декомпозована, тим зрозумілішим буде проект для виконання. WBS може бути представлена графічно, у вигляді ієрархічного дерева послідовних етапів проекту або у вигляді текстового переліку. Виконаємо побудову WBS структури, у якій зазначимо всі виконувані роботи залежно від головних етапів:

1. Створення документації – етап, який включає в себе створення технічного завдання та планування робіт. Розробка ТЗ складається з мети, постановки задачі, визначення етапів розробки, створеного прототипу інтерфейсу, опису функціоналу мобільного додатка.

2. Вивчення мови програмування та технології розробки під ОП андроїд

2.1. Поглиблене вивчення мови програмування Java.

2.2. Інсталяція та ознайомлення з Android Studio IDE.

2.3. Вивчення технології підвантаження віртуальної реальності для відображення 3D будівль.

3. Розробка мобільного додатку.

3.1. Написання програмного коду.

3.2. Створення інтерфейсу.

3.3 Занесення карти з маркерами до функціоналу мобільного додатку.

3.4 Міграця 3D моделей у додаток.

3.5 Розробка документації

4. Тестування мобільного додатку.

4.1. Проведення тестування навантаження.

4.2. Проведення функціонального тестування.

4.3 Проведення юзабіліті тестування

WBS-структура для даного проекту представлена на рисунку Б.1

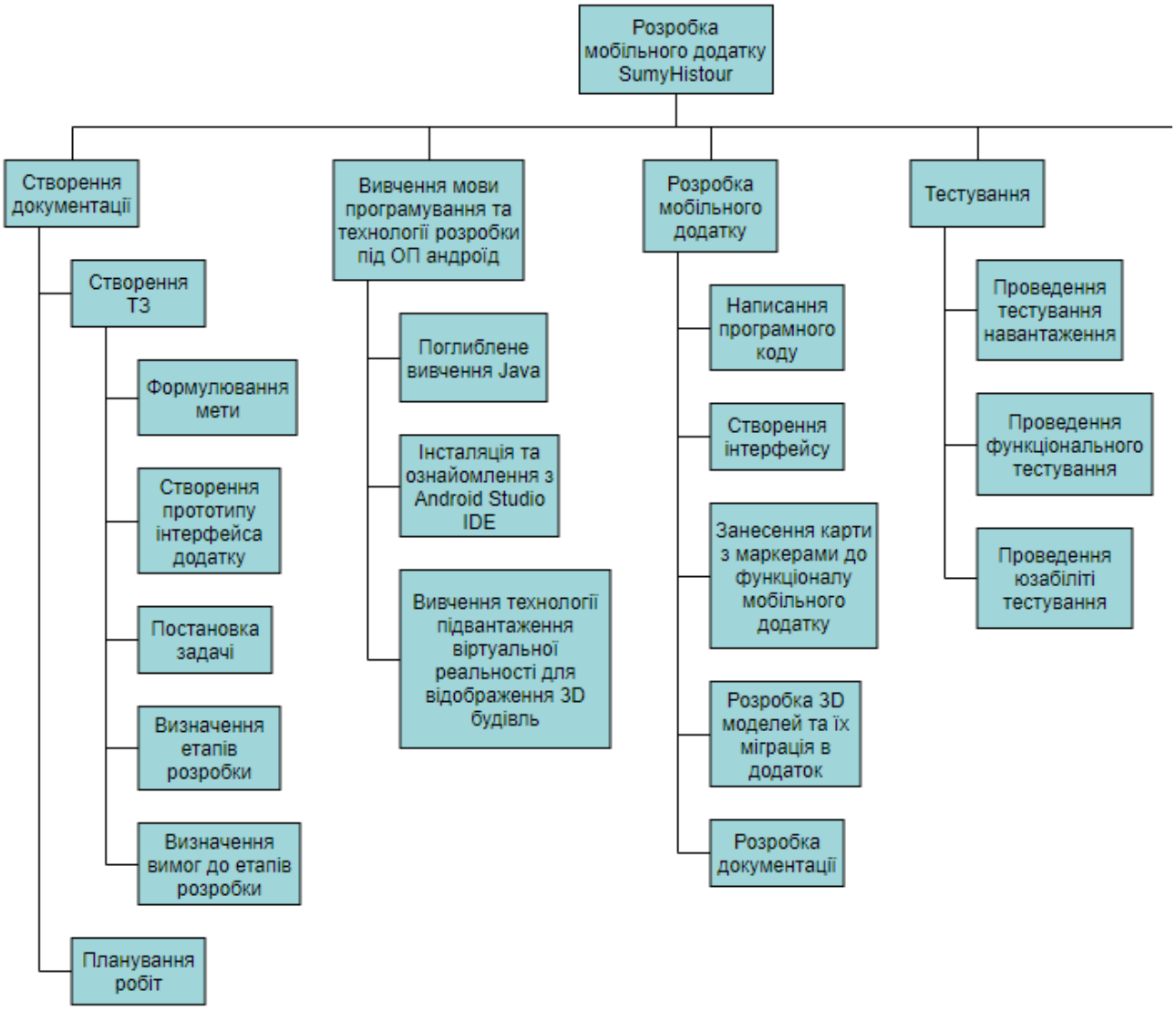

Рисунок Б.1 - WBS-структура мобільного додатку SumyHistour

Б.3.2 Планування структури організації, для впровадження готового проекту (OBS)

OBS-структура проекту – організаційна структура, яка включає всіх учасників проекту та їх ролі в проекті. Вона показує взаємовідношення між учасниками проекту.

Структура має графічне представлення у вигляді ієрархічного дерева.

На верхньому рівні структури знаходиться команда проекту. На другому рівні вказуються ті, хто буде виконувати проект. Наступний рівень включає в себе конкретних осіб які відповідають певному етапу розробки WBS.

OBS-структура для даного проекту представлена на рисунку Б.2

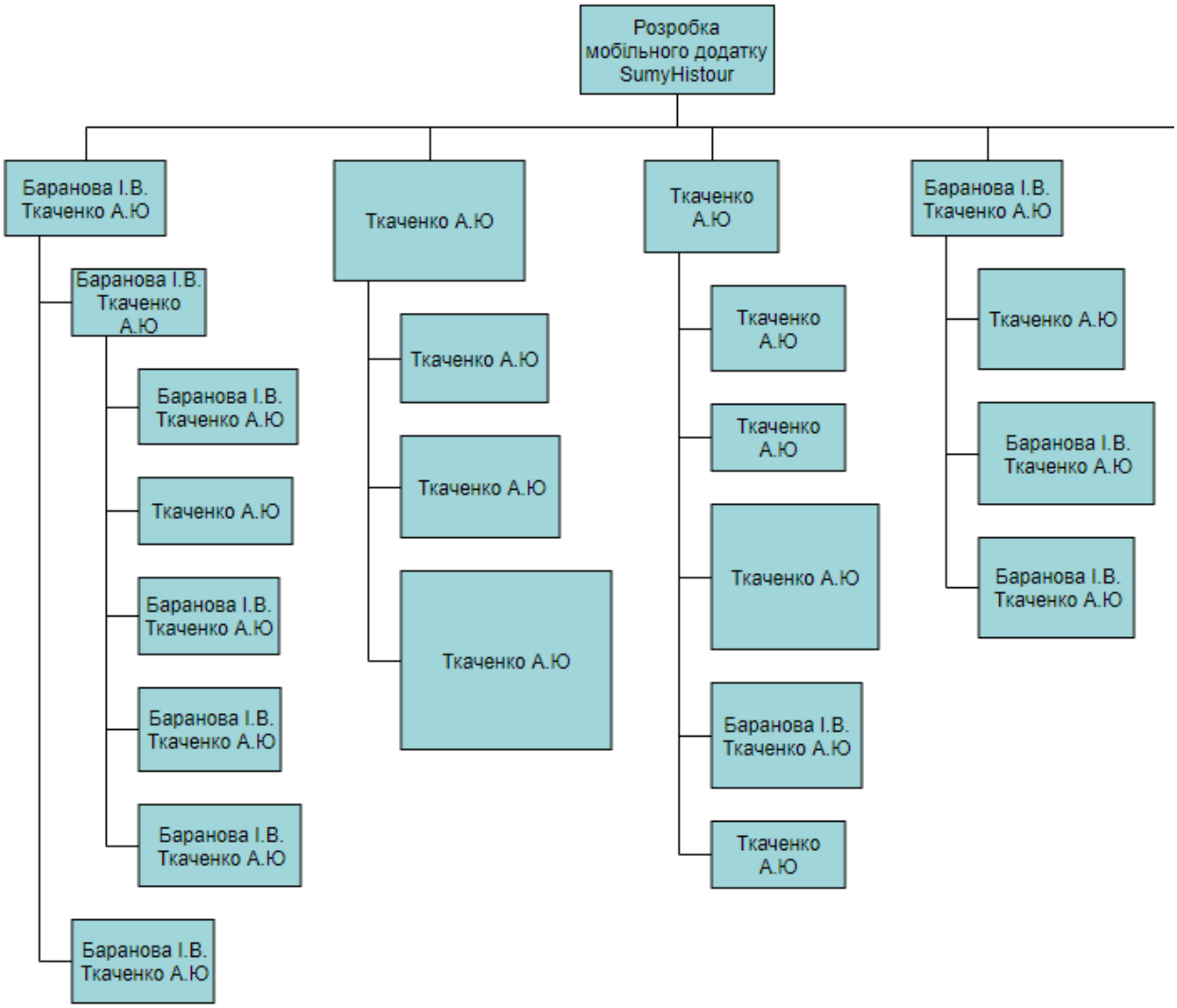

Рисунок Б.2 - Організаційна структура виконавців

Б.3.3 Побудова матриці відповідальності (виконавців пакетів робіт)

Матрицю відповідальності було створено на основі WBS та OBS, де кожен учасник має свою роль:

– відповідальний (В) – той хто відповідає за виконання роботи та вирішує спосіб її реалізації;

– консультант (К) – той хто наглядає за виконанням роботи та дає поради по способу реалізації, має відповідальність за виявлення недоліків.

– спостерігач (С) – той хто наглядає за виконанням роботи та дає поради по способу реалізації.

Матриця відповідальності представлена в табл. Б.1

Таблиця Б.1 – Матриця відповідальності

WBS\OBS	Ткаченко	Баранова	Шендрик
1 Створення документації	В	К	
1.1 Створення ТЗ	В	К	
1.1.1 Формулювання мети	В	К	
1.1.2 Постановка задачі	В	К	
1.1.3 Створення прототипу інтерфейсу додатку	В	К	
1.1.4 Визначення етапів розробки	В	К	
1.1.5 Визначення вимог до етапів розробки	В	К	
1.2 Планування робіт	В	К	
2 Вивчення мови програмування та технології розробки під ОП андроїд	В	С	
2.1. Поглиблене вивчення Java	В	С	
2.4. Інсталяція та ознайомлення з Android Studio IDE	В	С	
2.5. Вивчення технології завантаження віртуальної реальності для відображення 3D будівлю	В	С	
3 Розробка мобільного додатку	В	С	
3.1 Написання програмного коду	В	С	
3.2 Створення інтерфейсу	В	С	
3.3 Занесення карти з маркерами до функціоналу мобільного додатку	В	С	
3.4 Розробка 3D моделей та їх міграція в додаток	В	К	
3.5 Розробка документації	В	К	
4 Тестування	В	К	С
4.1 Проведення тестування навантаженням	В	К	С
4.2 Проведення функціонального тестування	В	К	С
4.3 Проведення юзабіліті тестування	В	К	С

Б.4 Побудова календарного графіку виконання ІТ-проекту

Діаграма Ганта – це графічне відображення плану робіт проекту з певним часовим обмеженням. На діаграмі представлений час початку та закінчення певного етапу роботи проекту, можливі часові затримки. Список етапів відображений вертикально. Діаграму Ганта для проекту було зроблено у програмі MS Excel.

Графік виконання дипломного проекту представлено у вигляді діаграми Ганта на рисунку Б.3 і Б.4.

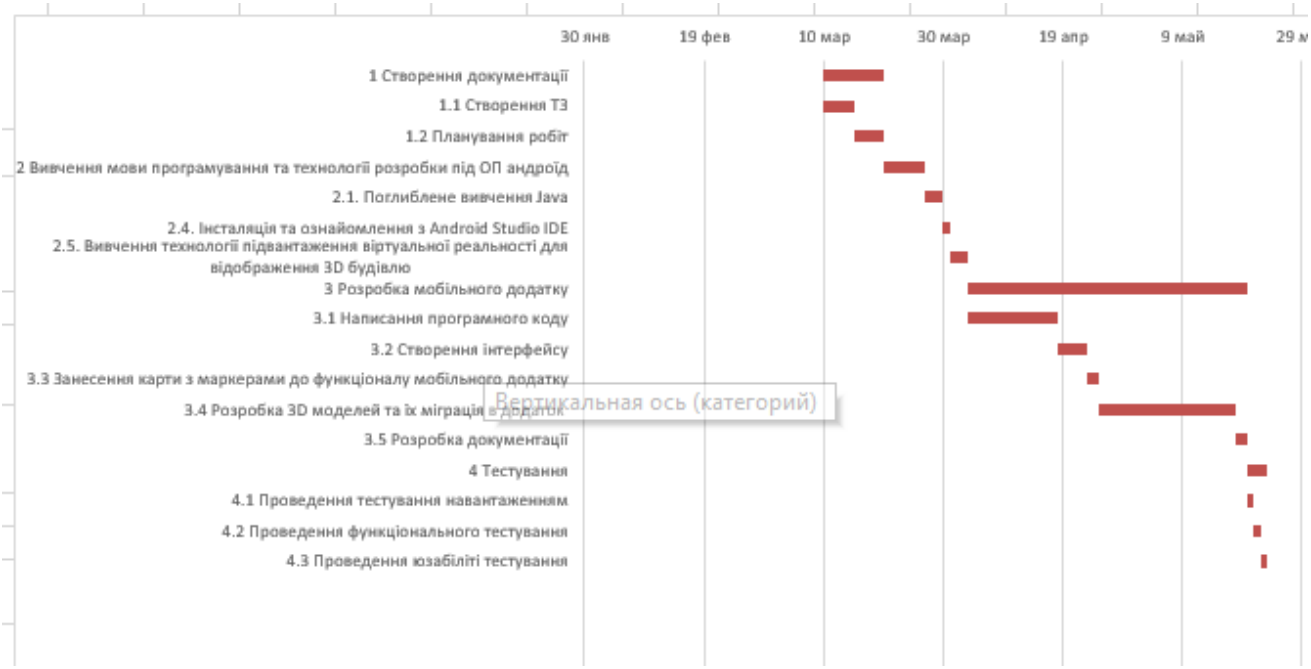

Рисунок Б.3 - Діаграма Ганта

Этап проекта	Начало	Длительность		Конец
1 Створення документації	10.03.2020	10		26.04.2020
1.1 Створення ТЗ	10.03.2020	5		14.03.2020
1.2 Планування робіт	15.03.2020	5		19.03.2020
2 Вивчення мови програмування та технології розробки під ОП андроїд	20.03.2020	7		26.03.2020
2.1. Поглиблене вивчення Java	27.03.2020	3		29.03.2020
2.4. Інсталяція та ознайомлення з Android Studio IDE	30.03.2020	1		30.03.2020
2.5. Вивчення технології підвантаження віртуальної реальності для відображення 3D будівлю	31.03.2020	3		02.04.2020
3 Розробка мобільного додатку	03.04.2020	47		19.05.2020
3.1 Написання програмного коду	03.04.2020	15		17.04.2020
3.2 Створення інтерфейсу	18.04.2020	5		22.04.2020
3.3 Занесення карти з маркерами до функціоналу мобільного додатку	23.04.2020	2		24.04.2020
3.4 Розробка 3D моделей та їх міграція в додаток	25.04.2020	23		17.05.2020
3.5 Розробка документації	18.05.2020	2		19.05.2020
4 Тестування	20.05.2020	3		22.05.2020
4.1 Проведення тестування навантаженням	20.05.2020	1		20.05.2020
4.2 Проведення функціонального тестування	21.05.2020	1		21.05.2020
4.3 Проведення юзабіліті тестування	22.05.2020	1		22.05.2020

Рисунок Б.4 - Діаграма Ганта

Б.5 Ідентифікація ризиків

Ідентифікація ризиків – це процес виявлення ризиків, які можуть негативно вплинути на проект. Це процес, якій постійно повторюється за життєвий цикл проекту.

Для того, щоб ідентифікувати ризики, потрібно створити дерево ризиків. Існує декілька типів ризиків. Нашому проекту відповідають такі типи ризиків як: зовнішні, технічні та організаційні. На рисунку Б.5 наведена класифікація ризиків.

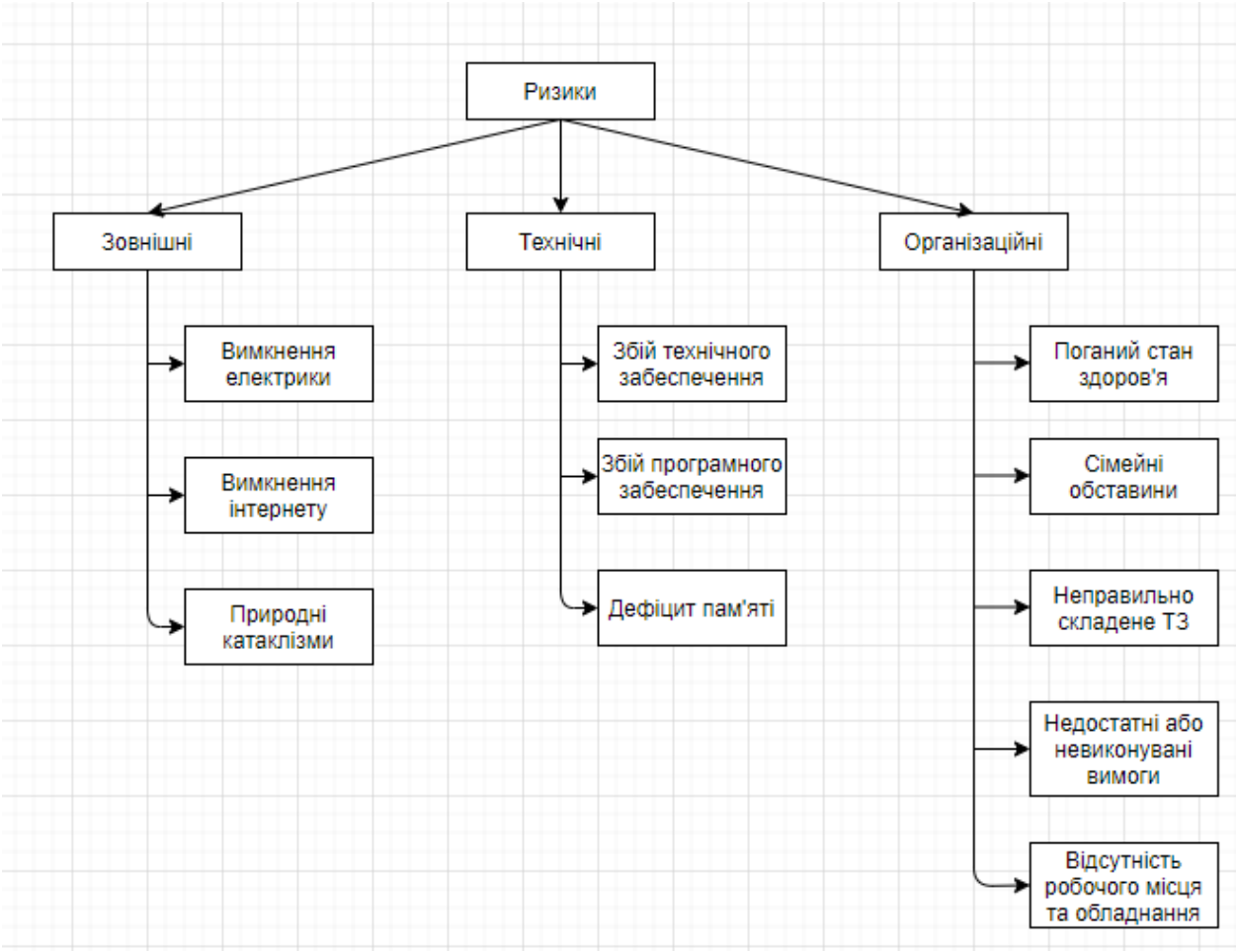

Рисунок Б.5 – Ризики

Матриця ризиків

Ризики представлені за допомогою RBM матриці (Risk Breakdown Matrix) на рисунку Б.6.

Класифікація ризиків за імовірністю виникнення:

– слабоімовірнісні;

– малоімовірнісні;

– імовірні;

– досить імовірні;

– майже імовірні.

Класифікація ризиків за імовірністю виникнення за величиною втрат:

– мінімальна;

– низька

– середня

– висока

– максимальна

Виконаємо класифікацію ризиків даного проекту. Для цього складемо таблицю Б.2.

Таблиця Б.2 – Класифікація ризиків дипломного проекту

Ризик		ймовірність виникнення	величина втрат
Вимкнення електрики	R1	2	5
Вимкнення інтернету	R2	3	3
Природні катаклізми	R3	1	4
Збій технічного забеспечення	R4	2	4
Збій програмного забеспечення	R5	2	3
Дефіцит пам'яті	R6	2	2
Поганий стан здоров'я	R7	3	4
Сімейні обставини	R8	3	4
Неправильно складене ТЗ	R9	3	3
Недостатні або невиконувані вимоги	R10	2	3
Відсутність робочого місця та обладнання	R11	1	4

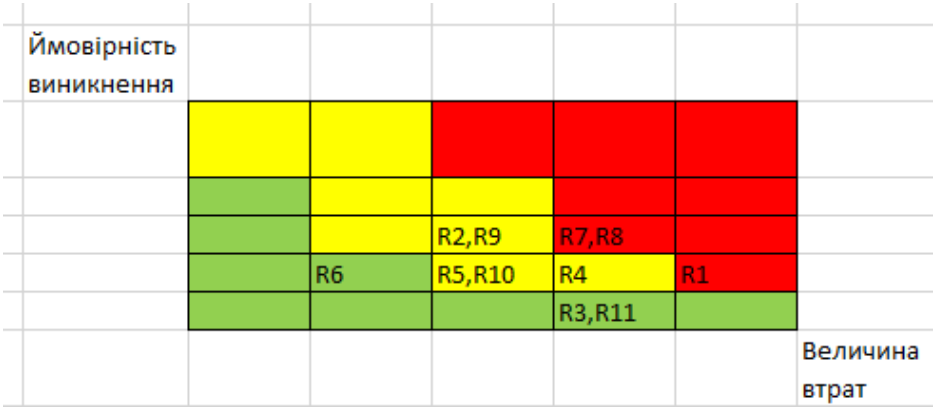

Рисунок Б.6 – Матриця імовірності втрат

Класифікація за ступенем впливу та за рівнем ризику (табл. Б.3)

Таблиця Б.3 – Класифікація за ступенем впливу та за рівнем ризику

Ризик		Ступінь впливу	Рівень ризику
Вимкнення електрики	R1	10	виправданні ризики
Вимкнення інтернету	R2	6	виправданні ризики
Природні катаклізми	R3	8	виправданні ризики
Збій технічного забезпечення	R4	9	виправданні ризики
Збій програмного забезпечення	R5	7	виправданні ризики
Дефіцит пам'яті	R6	3	прийнятні ризики
Поганий стан здоров'я	R7	12	недопустимі ризики
Сімейні обставини	R8	12	недопустимі ризики
Неправильно складене ТЗ	R9	6	виправданні ризики
Недостатні або невиконувані вимоги	R10	6	виправданні ризики
Відсутність робочого місця та обладнання	R11	10	недопустимі ризики

Класифікація за ступенем впливу:

- ігноровані ($1 \leq R \leq 4$);
- незначні ($5 \leq R \leq 8$);
- помірні ($9 \leq R \leq 11$);
- вагомі ($12 \leq R \leq 19$);
- критичні ($20 \leq R \leq 25$).

Класифікація за рівнем ризику:

- прийнятні ризики;
- виправданні ризики;
- недопустимі ризики;

План по усуненню ризиків:

- мати при собі запасне обладнання;
- перевірити якість програмного забезпечення, з котрим необхідно працювати;
- періодично робити резервні копії роботи;
- безперервна взаємодія з замовником;
- враховувати досвід проектів-аналогів;
- резервувати час на випадок помилок планування та виникнення непередбачених обставин;

ДОДАТОК В.
ЛІСТИНГ МОБІЛЬНОГО ДОДАТКУ

MainActivity.java

```
package com.example.sumyhistour;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    Button descrip;
    Button route;
    ImageButton ukr;
    ImageButton amr;
    static Integer a = 0;
    public int a() {

        return a;

    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ukr = (ImageButton) findViewById(R.id.ukr);
        ukr.setOnClickListener(this);
        route = (Button) findViewById(R.id.route);
        route.setOnClickListener(this);
        descrip = (Button) findViewById(R.id.descrip);
        descrip.setOnClickListener(this);
        amr = (ImageButton) findViewById(R.id.amr);
        amr.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.descrip:
                Intent intent = new Intent(this, DescrU.class);
                startActivity(intent);
                break;
            case R.id.route:
                Intent intent2 = new Intent(this, MapsActivity.class);
                startActivity(intent2);
                break;
            case R.id.ukr:
                a = 1;
                Intent intent3 = new Intent(this, Main2.class);
                startActivity(intent3);
```

```
                finish();
                break;
            case R.id.amr:
                a = 0;
                Intent intent4 = new Intent(this, MainActivity.class);
                startActivity(intent4);
                finish();
                break;
            default:
                break;
        }
    }

}
```

MapsActivity.java

```
package com.example.sumyhistour;

import androidx.appcompat.app.AlertDialog;
import androidx.core.app.ActivityCompat;
import androidx.fragment.app.FragmentActivity;

import android.Manifest;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.location.Location;
import android.location.LocationManager;
import android.os.Bundle;
import android.provider.Settings;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;


public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;
    LocationManager locationManager;
    private static  final int REQUEST_LOCATION=1;
    Button getlocationBtn;
    double latitude = 50.908453,longitude = 34.798952;
    int bool = 0;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
```

```
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
        ActivityCompat.requestPermissions(this,new String[]
                {Manifest.permission.ACCESS_FINE_LOCATION}, REQUEST_LOCATION);
        getlocationBtn=findViewById(R.id.loc_button);



    }

    private void getLocation() {

        //Check Permissions again

        if
(ActivityCompat.checkSelfPermission(MapsActivity.this,Manifest.permission.ACCESS_FINE_LO
CATION) != PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(MapsActivity.this,

                Manifest.permission.ACCESS_COARSE_LOCATION)
!=PackageManager.PERMISSION_GRANTED)
        {
            ActivityCompat.requestPermissions(this,new String[]
                    {Manifest.permission.ACCESS_FINE_LOCATION}, REQUEST_LOCATION);
        }
        else
        {
            Location LocationGps=
locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
            Location
LocationNetwork=locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
            Location
LocationPassive=locationManager.getLastKnownLocation(LocationManager.PASSIVE_PROVIDER);

            if (LocationGps !=null)
            {
                double lat=LocationGps.getLatitude();
                double longi=LocationGps.getLongitude();

                latitude=lat;
                longitude=longi;


            }
            else if (LocationNetwork !=null)
            {
                double lat=LocationNetwork.getLatitude();
                double longi=LocationNetwork.getLongitude();

                latitude=lat;
                longitude=longi;


            }
            else if (LocationPassive !=null)
            {
                double lat=LocationPassive.getLatitude();
                double longi=LocationPassive.getLongitude();

                latitude=lat;
                longitude=longi;
```

```
            }


            //Thats All Run Your App
        }

    }

    private void OnGPS() {

        final AlertDialog.Builder builder= new AlertDialog.Builder(this);

        builder.setMessage("Enable GPS").setCancelable(false).setPositiveButton("YES",
new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                startActivity(new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));
            }
        }).setNegativeButton("NO", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

                dialog.cancel();
            }
        });
        final AlertDialog alertDialog=builder.create();
        alertDialog.show();
    }



    @Override
    public void onMapReady(GoogleMap googleMap) {

        mMap = googleMap;
        Location location;
        getlocationBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                locationManager=(LocationManager)
getSystemService(Context.LOCATION_SERVICE);

                //Check gps is enable or not

                if (!locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER))
                {
                    //Write Function To enable gps

                    OnGPS();
                }
                else
                {
                    //GPS is already On then

                    getLocation();
                }
                LatLng Sumy = new LatLng(latitude, longitude);
                mMap.addMarker(new MarkerOptions().position(Sumy).title("Sumy"));
                mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(Sumy, 15F));
            }
```

```
        });


    }
}
```

Main2.java

```
package com.example.sumyhistour;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;

import androidx.appcompat.app.AppCompatActivity;

public class Main2 extends AppCompatActivity implements View.OnClickListener{
    Button descrip;
    Button route;
    ImageButton ukr;
    ImageButton amr;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.act_main2);

        ukr = (ImageButton) findViewById(R.id.ukr2);
        ukr.setOnClickListener(this);
        route = (Button) findViewById(R.id.route2);
        route.setOnClickListener(this);
        descrip = (Button) findViewById(R.id.descrip2);
        descrip.setOnClickListener(this);
        amr = (ImageButton) findViewById(R.id.amr2);
        amr.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.descrip2:
                Intent intent = new Intent(this, DescrU.class);
                startActivity(intent);
                break;
            case R.id.route2:
                Intent intent2 = new Intent(this, MapsActivity.class);
                startActivity(intent2);
                break;
            case R.id.ukr2:
                Intent intent3 = new Intent(this, Main2.class);
                startActivity(intent3);
                finish();
                break;
            case R.id.amr2:
                Intent intent4 = new Intent(this, MainActivity.class);
                startActivity(intent4);
                finish();
                break;
            default:
```

```
                break;
        }
    }
}
```

DescrU.java

```
package com.example.sumyhistour;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;


public class DescrU extends AppCompatActivity {

    ListView userList;
    DBHelper databaseHelper;
    SQLiteDatabase db;
    Cursor userCursor;
    SimpleCursorAdapter userAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_descr_u);
        // находим список

        userList = (ListView)findViewById(R.id.list);
        userList.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long
id) {
                Intent intent = new Intent(getApplicationContext(), DescrU2.class);
                intent.putExtra("id", id);
                startActivity(intent);
            }
        });

        databaseHelper = new DBHelper(getApplicationContext());
    }
    public void onResume() {
        super.onResume();
        // открываем подключение
        db = databaseHelper.getReadableDatabase();
       MainActivity buff = new MainActivity();
        if(buff.a() == 1)
        userCursor =  db.rawQuery("select * from "+ DBHelper.TABLE_CONTACTS + " where _id
> 4", null);
        else
            userCursor =  db.rawQuery("select * from "+ DBHelper.TABLE_CONTACTS + "
where _id < 5", null);
        // определяем, какие столбцы из курсора будут выводиться в ListView
        String[] headers = new String[] {DBHelper.KEY_NAME, DBHelper.KEY_MAIL};
```

```
        // создаем адаптер, передаем в него курсор
        userAdapter = new SimpleCursorAdapter(this, android.R.layout.simple_list_item_1,
                userCursor, headers, new int[]{android.R.id.text1}, 0);
        userList.setAdapter(userAdapter);
    }
    // по нажатию на кнопку запускаем UserActivity для добавления данных

    @Override
    public void onDestroy(){
        super.onDestroy();
        // Закрываем подключение и курсор
        db.close();
        userCursor.close();
    }
}
```

DescrU2.java

```
package com.example.sumyhistour;

import android.content.ContentValues;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.View;

import android.widget.Button;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

import com.example.sumyhistour.model3D.view.MenuActivity;

//import com.example.sumyhistour.model3D.view.MenuActivity;

public class DescrU2 extends AppCompatActivity implements View.OnClickListener{
    TextView yearBox;
    TextView name1;
    DBHelper sqlHelper;
    SQLiteDatabase db;
    Cursor userCursor;
    long userId=0;
    Button but;
    static Integer buff;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_descr_u2);
        but = (Button) findViewById(R.id.but);
        but.setOnClickListener(this);

        yearBox = (TextView) findViewById(R.id.year);
        name1 = (TextView) findViewById(R.id.name1);

        sqlHelper = new DBHelper(this);
        db = sqlHelper.getWritableDatabase();

        Bundle extras = getIntent().getExtras();
        if (extras != null) {
            userId = extras.getLong("id");
```

```
        }
        // если 0, то добавление
        if (userId > 0) {
            // получаем элемент по id из бд
            userCursor = db.rawQuery("select * from " + DBHelper.TABLE_CONTACTS + "
where " +
                    DBHelper.KEY_ID + "=?", new String[]{String.valueOf(userId)});
            userCursor.moveToFirst();
            yearBox.setText(String.valueOf(userCursor.getString(2)));
            name1.setText(String.valueOf(userCursor.getString(1)));
            buff = Integer.valueOf(userCursor.getString(0));
            userCursor.close();
            db.close();
        }
    }

    public int buff() {
        return buff;
    }
    @Override
    public void onClick(View v) {

        Intent intent = new Intent(this, MenuActivity.class);
        startActivity(intent);

    }
}
```

DBHelper.java

```
package com.example.sumyhistour;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DBHelper  extends SQLiteOpenHelper{

    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "contactDb";
    public static final String TABLE_CONTACTS = "contacts";

    public static final String KEY_ID = "_id";
    public static final String KEY_NAME = "name";
    public static final String KEY_MAIL = "mail";

    public DBHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create table " + TABLE_CONTACTS + "(" + KEY_ID
                + " integer primary key," + KEY_NAME + " text," + KEY_MAIL + " varchar"
+ ")");


    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
```

```
        db.execSQL("drop table if exists " + TABLE_CONTACTS);

        onCreate(db);

    }


}
```

MenuActivity.java

```
package com.example.sumyhistour.model3D.view;


import android.app.ListActivity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;

import com.example.sumyhistour.DescrU2;
import com.example.sumyhistour.R;


import java.util.HashMap;
import java.util.Map;

public class MenuActivity extends ListActivity {

    DescrU2 id = new DescrU2();

    private Map<String, Object> loadModelParameters = new HashMap<>();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu);
        launchModelRendererActivity(Uri.parse("assets:///models/" +
String.valueOf(id.buff()) + ".obj"));
    }


    private void launchModelRendererActivity(Uri uri) {
        Log.i("Menu", "Launching renderer for '" + uri + "'");
        Intent intent = new Intent(getApplicationContext(), ModelActivity.class);
        intent.putExtra("uri", uri.toString());
        intent.putExtra("immersiveMode", "true");

        // content provider case
        if (!loadModelParameters.isEmpty()) {
            intent.putExtra("type", loadModelParameters.get("type").toString());
            loadModelParameters.clear();
        }

        startActivity(intent);
        finish();
    }
}
```

ModelActivity.java

```
package com.example.sumyhistour.model3D.view;

import android.annotation.TargetApi;
import android.app.Activity;
import android.content.ActivityNotFoundException;
import android.content.Intent;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Toast;

import com.example.sumyhistour.R;
import com.example.sumyhistour.model3D.demo.ExampleSceneLoader;
import com.example.sumyhistour.model3D.demo.SceneLoader;

import org.andresoviedo.util.android.ContentUtils;

import java.io.IOException;

/**
 * This activity represents the container for our 3D viewer.
 *
 * @author andresoviedo
 */
public class ModelActivity extends Activity {

    private static final int REQUEST_CODE_LOAD_TEXTURE = 1000;
    private static final int FULLSCREEN_DELAY = 10000;

    /**
     * Type of model if file name has no extension (provided though content provider)
     */
    private int paramType;
    /**
     * The file to load. Passed as input parameter
     */
    private Uri paramUri;
    /**
     * Enter into Android Immersive mode so the renderer is full screen or not
     */
    private boolean immersiveMode = true;
    /**
     * Background GL clear color. Default is light gray
     */
    private float[] backgroundColor = new float[]{0f, 0f, 0f, 1.0f};

    private ModelSurfaceView gLView;

    private SceneLoader scene;

    private Handler handler;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        // Try to get input parameters
        Bundle b = getIntent().getExtras();
        if (b != null) {
            if (b.getString("uri") != null) {
                this.paramUri = Uri.parse(b.getString("uri"));
            }
            this.paramType = b.getString("type") != null ?
Integer.parseInt(b.getString("type")) : -1;
            this.immersiveMode = "true".equalsIgnoreCase(b.getString("immersiveMode"));
            try {
                String[] backgroundColors = b.getString("backgroundColor").split(" ");
                backgroundColor[0] = Float.parseFloat(backgroundColors[0]);
                backgroundColor[1] = Float.parseFloat(backgroundColors[1]);
                backgroundColor[2] = Float.parseFloat(backgroundColors[2]);
                backgroundColor[3] = Float.parseFloat(backgroundColors[3]);
            } catch (Exception ex) {
                // Assuming default background color
            }
        }
        Log.i("Renderer", "Params: uri '" + paramUri + "'");

        handler = new Handler(getMainLooper());

        // Create our 3D sceneario
        if (paramUri == null) {
            scene = new ExampleSceneLoader(this);
        } else {
            scene = new SceneLoader(this);
        }
        scene.init();

        // Create a GLSurfaceView instance and set it
        // as the ContentView for this Activity.
        try {
            gLView = new ModelSurfaceView(this);
            setContentView(gLView);
        } catch (Exception e) {
            Toast.makeText(this, "Error loading OpenGL view:\n" +e.getMessage(),
Toast.LENGTH_LONG).show();
        }

        // Show the Up button in the action bar.
        setupActionBar();

        // TODO: Alert user when there is no multitouch support (2 fingers). He won't be
able to rotate or zoom
        ContentUtils.printTouchCapabilities(getPackageManager());

        setupOnSystemVisibilityChangeListener();
    }

    /**
     * Set up the {@link android.app.ActionBar}, if the API is available.
     */
    @TargetApi(Build.VERSION_CODES.HONEYCOMB)
    private void setupActionBar() {
        // if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
        // getActionBar().setDisplayHomeAsUpEnabled(true);
        // }
    }
```

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.model, menu);
        return true;
    }

    @TargetApi(Build.VERSION_CODES.JELLY_BEAN)
    private void setupOnSystemVisibilityChangeListener() {
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.JELLY_BEAN) {
            return;
        }
        getWindow().getDecorView().setOnSystemUiVisibilityChangeListener(visibility -> {
            // Note that system bars will only be "visible" if none of the
            // LOW_PROFILE, HIDE_NAVIGATION, or FULLSCREEN flags are set.
            if ((visibility & View.SYSTEM_UI_FLAG_FULLSCREEN) == 0) {
                // The system bars are visible. Make any desired
                hideSystemUIDelayed();
            }
        });
    }

    @Override
    public void onWindowFocusChanged(boolean hasFocus) {
        super.onWindowFocusChanged(hasFocus);
        if (hasFocus) {
            hideSystemUIDelayed();
        }
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {

            case R.id.model_load_texture:
                Intent target = ContentUtils.createGetContentIntent("image/*");
                Intent intent = Intent.createChooser(target, "Select a file");
                try {
                    startActivityForResult(intent, REQUEST_CODE_LOAD_TEXTURE);
                } catch (ActivityNotFoundException e) {
                    // The reason for the existence of aFileChooser
                }
                break;
        }

        hideSystemUIDelayed();
        return super.onOptionsItemSelected(item);
    }

    private void toggleImmersive() {
        this.immersiveMode = !this.immersiveMode;
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.JELLY_BEAN) {
            return;
        }
        if (this.immersiveMode) {
            hideSystemUI();
        } else {
            showSystemUI();
        }
        Toast.makeText(this, "Fullscreen " +this.immersiveMode,
Toast.LENGTH_SHORT).show();
    }
```

```
    private void hideSystemUIDelayed() {
        if (!this.immersiveMode) {
            return;
        }
        handler.removeCallbacksAndMessages(null);
        handler.postDelayed(this::hideSystemUI, FULLSCREEN_DELAY);

    }

    private void hideSystemUI() {
        if (!this.immersiveMode) {
            return;
        }
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
            hideSystemUIKitKat();
        } else if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN) {
            hideSystemUIJellyBean();
        }
    }

    // This snippet hides the system bars.
    @TargetApi(Build.VERSION_CODES.KITKAT)
    private void hideSystemUIKitKat() {
        // Set the IMMERSIVE flag.
        // Set the content to appear under the system bars so that the content
        // doesn't resize when the system bars hide and show.
        final View decorView = getWindow().getDecorView();
        decorView.setSystemUiVisibility(View.SYSTEM_UI_FLAG_LAYOUT_STABLE |
View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
                | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN |
View.SYSTEM_UI_FLAG_HIDE_NAVIGATION // hide nav bar
                | View.SYSTEM_UI_FLAG_FULLSCREEN // hide status bar
                | View.SYSTEM_UI_FLAG_IMMERSIVE);
    }

    @TargetApi(Build.VERSION_CODES.JELLY_BEAN)
    private void hideSystemUIJellyBean() {
        final View decorView = getWindow().getDecorView();
        decorView.setSystemUiVisibility(View.SYSTEM_UI_FLAG_LAYOUT_STABLE |
View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
                | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN |
View.SYSTEM_UI_FLAG_HIDE_NAVIGATION
                | View.SYSTEM_UI_FLAG_FULLSCREEN | View.SYSTEM_UI_FLAG_LOW_PROFILE);
    }

    // This snippet shows the system bars. It does this by removing all the flags
    // except for the ones that make the content appear under the system bars.
    @TargetApi(Build.VERSION_CODES.JELLY_BEAN)
    private void showSystemUI() {
        handler.removeCallbacksAndMessages(null);
        final View decorView = getWindow().getDecorView();
        decorView.setSystemUiVisibility(View.SYSTEM_UI_FLAG_VISIBLE);
    }

    public Uri getParamUri() {
        return paramUri;
    }

    public int getParamType() {
        return paramType;
    }
```

```
    public float[] getBackgroundColor() {
        return backgroundColor;
    }

    public SceneLoader getScene() {
        return scene;
    }

    public ModelSurfaceView getGLView() {
        return gLView;
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode != RESULT_OK) {
            return;
        }
        switch (requestCode) {
            case REQUEST_CODE_LOAD_TEXTURE:
                // The URI of the selected file
                final Uri uri = data.getData();
                if (uri != null) {
                    Log.i("ModelActivity", "Loading texture '" + uri + "'");
                    try {
                        ContentUtils.setThreadActivity(this);
                        scene.loadTexture(null, uri);
                    } catch (IOException ex) {
                        Log.e("ModelActivity", "Error loading texture: " +
ex.getMessage(), ex);
                        Toast.makeText(this, "Error loading texture '" + uri + "'. " +
ex
                                .getMessage(), Toast.LENGTH_LONG).show();
                    } finally {
                        ContentUtils.setThreadActivity(null);
                    }
                }
        }
    }
}
```

ModelRenderer.java

```
package com.example.sumyhistour.model3D.view;

import android.opengl.GLES20;
import android.opengl.GLSurfaceView;
import android.opengl.Matrix;
import android.util.Log;

import com.example.sumyhistour.model3D.demo.SceneLoader;

import org.andresoviedo.android_3d_model_engine.animation.Animator;
import org.andresoviedo.android_3d_model_engine.drawer.DrawerFactory;
import org.andresoviedo.android_3d_model_engine.model.AnimatedModel;
import org.andresoviedo.android_3d_model_engine.model.Camera;
import org.andresoviedo.android_3d_model_engine.model.Object3D;
import org.andresoviedo.android_3d_model_engine.model.Object3DData;
import org.andresoviedo.android_3d_model_engine.services.Object3DBuilder;
import org.andresoviedo.util.android.GLUtil;
```

```
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;

public class ModelRenderer implements GLSurfaceView.Renderer {

   private final static String TAG = ModelRenderer.class.getName();
    /**
     * Add 0.5f to the alpha component to the global shader so we can see through the
skin
     */
   private static final float[] BLENDING_FORCED_MASK_COLOR = {1.0f, 1.0f, 1.0f, 0.5f};
   // frustrum - nearest pixel
   private static final float near = 1f;
   // frustrum - fartest pixel
   private static final float far = 100f;
   // stereoscopic variables
   private static float EYE_DISTANCE = 0.64f;
   private static final float[] COLOR_RED = {1.0f, 0.0f, 0.0f, 1f};
   private static final float[] COLOR_BLUE = {0.0f, 1.0f, 0.0f, 1f};

   // 3D window (parent component)
   private ModelSurfaceView main;
   // width of the screen
   private int width;
   // height of the screen
   private int height;

    /**
     * Drawer factory to get right renderer/shader based on object attributes
     */
   private DrawerFactory drawer;
    /**
     * 3D Axis (to show if needed)
     */
    private final Object3DData axis = Object3DBuilder.buildAxis().setId("axis");

   // The wireframe associated shape (it should be made of lines only)
   private Map<Object3DData, Object3DData> wireframes = new HashMap<>();
   // The loaded textures
   private Map<Object, Integer> textures = new HashMap<>();
   // The corresponding opengl bounding boxes and drawer
   private Map<Object3DData, Object3DData> boundingBoxes = new HashMap<>();
   // The corresponding opengl bounding boxes
   private Map<Object3DData, Object3DData> normals = new HashMap<>();
   private Map<Object3DData, Object3DData> skeleton = new HashMap<>();

   // 3D matrices to project our 3D world
   private final float[] viewMatrix = new float[16];
   private final float[] modelViewMatrix = new float[16];
   private final float[] projectionMatrix = new float[16];
   private final float[] viewProjectionMatrix = new float[16];
   private final float[] lightPosInWorldSpace = new float[4];
    private final float[] cameraPosInWorldSpace = new float[3];

   // 3D stereoscopic matrix (left & right camera)
    private final float[] viewMatrixLeft = new float[16];
```

```
    private final float[] projectionMatrixLeft = new float[16];
    private final float[] viewProjectionMatrixLeft = new float[16];
    private final float[] viewMatrixRight = new float[16];
    private final float[] projectionMatrixRight = new float[16];
    private final float[] viewProjectionMatrixRight = new float[16];



    /**
     * Whether the info of the model has been written to console log
     */
    private Map<Object3DData, Boolean> infoLogged = new HashMap<>();
    /**
     * Switch to akternate drawing of right and left image
     */
    private boolean anaglyphSwitch = false;

    /**
     * Skeleton Animator
     */
    private Animator animator = new Animator();
    /**
     * Did the application explode?
     */
    private boolean fatalException = false;

    /**
     * Construct a new renderer for the specified surface view
     *
     * @param modelSurfaceView
     *            the 3D window
     */
    public ModelRenderer(ModelSurfaceView modelSurfaceView) throws
IllegalAccessException, IOException {
        this.main = modelSurfaceView;
        // This component will draw the actual models using OpenGL
        drawer = new DrawerFactory(modelSurfaceView.getContext());
    }

    public float getNear() {
        return near;
    }

    public float getFar() {
        return far;
    }

    @Override
    public void onSurfaceCreated(GL10 unused, EGLConfig config) {
        // Set the background frame color
        float[] backgroundColor = main.getModelActivity().getBackgroundColor();
        GLES20.glClearColor(backgroundColor[0], backgroundColor[1], backgroundColor[2],
backgroundColor[3]);

        // Use culling to remove back faces.
        // Don't remove back faces so we can see them
        // GLES20.glEnable(GLES20.GL_CULL_FACE);

        // Enable depth testing for hidden-surface elimination.
        GLES20.glEnable(GLES20.GL_DEPTH_TEST);

        // Enable not drawing out of view port
```

```
        GLES20.glEnable(GLES20.GL_SCISSOR_TEST);
    }

    @Override
    public void onSurfaceChanged(GL10 unused, int width, int height) {
        this.width = width;
        this.height = height;

        // Adjust the viewport based on geometry changes, such as screen rotation
        GLES20.glViewport(0, 0, width, height);

        // the projection matrix is the 3D virtual space (cube) that we want to project
        float ratio = (float) width / height;
        Log.d(TAG, "projection: [" + -ratio + "," + ratio + ",-1,1]-near/far[1,10]");
        Matrix.frustumM(projectionMatrix, 0, -ratio, ratio, -1, 1, getNear(), getFar());
        Matrix.frustumM(projectionMatrixRight, 0, -ratio, ratio, -1, 1, getNear(),
getFar());
        Matrix.frustumM(projectionMatrixLeft, 0, -ratio, ratio, -1, 1, getNear(),
getFar());
    }

    @Override
    public void onDrawFrame(GL10 unused) {
        if(fatalException){
            return;
        }
        try {

            GLES20.glViewport(0, 0, width, height);
            GLES20.glScissor(0, 0, width, height);

            // Draw background color
            GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT | GLES20.GL_DEPTH_BUFFER_BIT);

            SceneLoader scene = main.getModelActivity().getScene();
            if (scene == null) {
                // scene not ready
                return;
            }

            float[] colorMask = null;
            if (scene.isBlendingEnabled()) {
                // Enable blending for combining colors when there is transparency
                GLES20.glEnable(GLES20.GL_BLEND);
                GLES20.glBlendFunc(GLES20.GL_ONE, GLES20.GL_ONE_MINUS_SRC_ALPHA);
                if (scene.isBlendingForced()){
                    colorMask = BLENDING_FORCED_MASK_COLOR;
                }
            } else {
                GLES20.glDisable(GLES20.GL_BLEND);
            }

            // animate scene
            scene.onDrawFrame();

            // recalculate mvp matrix according to where we are looking at now
            Camera camera = scene.getCamera();
                cameraPosInWorldSpace[0]=camera.xPos;
                cameraPosInWorldSpace[1]=camera.yPos;
                cameraPosInWorldSpace[2]=camera.zPos;
            if (camera.hasChanged()) {
                // INFO: Set the camera position (View matrix)
```

```
            // The camera has 3 vectors (the position, the vector where we are looking
at, and the up position (sky)

            // the projection matrix is the 3D virtual space (cube) that we want to
project
            float ratio = (float) width / height;
            // Log.v(TAG, "Camera changed: projection: [" + -ratio + "," + ratio + ",-
1,1]-near/far[1,10], ");

            if (!scene.isStereoscopic()) {
               Matrix.setLookAtM(viewMatrix, 0, camera.xPos, camera.yPos, camera.zPos,
camera.xView, camera.yView,
                     camera.zView, camera.xUp, camera.yUp, camera.zUp);
               Matrix.multiplyMM(viewProjectionMatrix, 0, projectionMatrix, 0,
viewMatrix, 0);
            } else {
               Camera[] stereoCamera = camera.toStereo(EYE_DISTANCE);
               Camera leftCamera = stereoCamera[0];
               Camera rightCamera = stereoCamera[1];

               // camera on the left for the left eye
               Matrix.setLookAtM(viewMatrixLeft, 0, leftCamera.xPos, leftCamera.yPos,
leftCamera.zPos, leftCamera
                           .xView,
                     leftCamera.yView, leftCamera.zView, leftCamera.xUp, leftCamera.yUp,
leftCamera.zUp);
               // camera on the right for the right eye
               Matrix.setLookAtM(viewMatrixRight, 0, rightCamera.xPos, rightCamera.yPos,
rightCamera.zPos, rightCamera
                           .xView,
                     rightCamera.yView, rightCamera.zView, rightCamera.xUp,
rightCamera.yUp, rightCamera.zUp);

               if (scene.isAnaglyph()) {
                  Matrix.frustumM(projectionMatrixRight, 0, -ratio, ratio, -1, 1,
getNear(), getFar());
                  Matrix.frustumM(projectionMatrixLeft, 0, -ratio, ratio, -1, 1,
getNear(), getFar());
               } else if (scene.isVRGlasses()) {
                  float ratio2 = (float) width / 2 / height;
                  Matrix.frustumM(projectionMatrixRight, 0, -ratio2, ratio2, -1, 1,
getNear(), getFar());
                  Matrix.frustumM(projectionMatrixLeft, 0, -ratio2, ratio2, -1, 1,
getNear(), getFar());
               }
               // Calculate the projection and view transformation
               Matrix.multiplyMM(viewProjectionMatrixLeft, 0, projectionMatrixLeft, 0,
viewMatrixLeft, 0);
               Matrix.multiplyMM(viewProjectionMatrixRight, 0, projectionMatrixRight, 0,
viewMatrixRight, 0);

            }

            camera.setChanged(false);

         }


         if (!scene.isStereoscopic()) {
            this.onDrawFrame(viewMatrix, projectionMatrix, viewProjectionMatrix,
lightPosInWorldSpace, colorMask, cameraPosInWorldSpace);
            return;
```

```
        }


        if (scene.isAnaglyph()) {
           // INFO: switch because blending algorithm doesn't mix colors
           if (anaglyphSwitch) {
              this.onDrawFrame(viewMatrixLeft, projectionMatrixLeft,
viewProjectionMatrixLeft, lightPosInWorldSpace,
                    COLOR_RED, cameraPosInWorldSpace);
           } else {
              this.onDrawFrame(viewMatrixRight, projectionMatrixRight,
viewProjectionMatrixRight, lightPosInWorldSpace,
                    COLOR_BLUE, cameraPosInWorldSpace);
           }
           anaglyphSwitch = !anaglyphSwitch;
           return;
        }

        if (scene.isVRGlasses()) {

           // draw left eye image
           GLES20.glViewport(0, 0, width / 2, height);
           GLES20.glScissor(0, 0, width / 2, height);
           this.onDrawFrame(viewMatrixLeft, projectionMatrixLeft,
viewProjectionMatrixLeft, lightPosInWorldSpace,
                 null, cameraPosInWorldSpace);

           // draw right eye image
           GLES20.glViewport(width / 2, 0, width / 2, height);
           GLES20.glScissor(width / 2, 0, width / 2, height);
           this.onDrawFrame(viewMatrixRight, projectionMatrixRight,
viewProjectionMatrixRight, lightPosInWorldSpace,
                 null, cameraPosInWorldSpace);
        }
      }catch (Exception ex){
         Log.e("ModelRenderer", "Fatal exception: "+ex.getMessage(), ex);
         fatalException = true;
      }
   }

   private void onDrawFrame(float[] viewMatrix, float[] projectionMatrix, float[]
viewProjectionMatrix,
                    float[] lightPosInWorldSpace, float[] colorMask, float[]
cameraPosInWorldSpace) {


      SceneLoader scene = main.getModelActivity().getScene();

      // draw light
      if (scene.isDrawLighting()) {

         Object3D lightBulbDrawer = drawer.getPointDrawer();

         // Calculate position of the light in world space to support lighting
         if (scene.isRotatingLight()) {
            Matrix.multiplyMV(lightPosInWorldSpace, 0,
scene.getLightBulb().getModelMatrix(), 0, scene.getLightPosition(), 0);
            // Draw a point that represents the light bulb
            lightBulbDrawer.draw(scene.getLightBulb(), projectionMatrix, viewMatrix, -1,
lightPosInWorldSpace,
                  colorMask, cameraPosInWorldSpace);
         } else {
```

```
            lightPosInWorldSpace[0] = scene.getCamera().xPos;
            lightPosInWorldSpace[1] = scene.getCamera().yPos;
            lightPosInWorldSpace[2] = scene.getCamera().zPos;
                lightPosInWorldSpace[3] = 0;
         }

         // FIXME: memory leak
         if (scene.isDrawNormals()) {
            lightBulbDrawer.draw(Object3DBuilder.buildLine(new
float[]{lightPosInWorldSpace[0],
                        lightPosInWorldSpace[1], lightPosInWorldSpace[2], 0, 0, 0}),
projectionMatrix,
                  viewMatrix, -1,
                        lightPosInWorldSpace,
                  colorMask, cameraPosInWorldSpace);
         }
      }

      // draw axis
        if (scene.isDrawAxis()){
         Object3D basicDrawer = drawer.getPointDrawer();
         basicDrawer.draw(axis, projectionMatrix, viewMatrix, axis.getDrawMode(), axis
               .getDrawSize(),-1, lightPosInWorldSpace, colorMask,
cameraPosInWorldSpace);
        }


        // is there any object?
      if (scene.getObjects().isEmpty()){
         return;
      }

      // draw all available objects
      List<Object3DData> objects = scene.getObjects();
      for (int i=0; i<objects.size(); i++) {
         Object3DData objData = null;
         try {
            objData = objects.get(i);
            if (!objData.isVisible()) continue;

            Object3D drawerObject = drawer.getDrawer(objData, scene.isDrawTextures(),
scene.isDrawLighting(),
                        scene.isDoAnimation(), scene.isDrawColors());

            if (drawerObject == null){
               continue;
            }

            if (!infoLogged.containsKey(objData)) {
               Log.v("ModelRenderer","Drawing model: "+objData.getId());
               infoLogged.put(objData, true);
            }

            boolean changed = objData.isChanged();

            // load model texture
            Integer textureId = textures.get(objData.getTextureData());
            if (textureId == null && objData.getTextureData() != null) {
               Log.i("ModelRenderer","Loading texture
'"+objData.getTextureFile()+"'...");
               ByteArrayInputStream textureIs = new
ByteArrayInputStream(objData.getTextureData());
```

```
                textureId = GLUtil.loadTexture(textureIs);
                textureIs.close();
                textures.put(objData.getTextureData(), textureId);
                Log.i("GLUtil", "Loaded texture ok. id: "+textureId);
            }
            if (textureId == null){
                textureId = -1;
            }

            // draw points
            if (objData.getDrawMode() == GLES20.GL_POINTS){
                Object3D basicDrawer = drawer.getPointDrawer();
                basicDrawer.draw(objData, projectionMatrix, viewMatrix, GLES20.GL_POINTS,
lightPosInWorldSpace, cameraPosInWorldSpace);
            }

            // draw wireframe
            else if (scene.isDrawWireframe() && objData.getDrawMode() !=
GLES20.GL_POINTS
                    && objData.getDrawMode() != GLES20.GL_LINES && objData.getDrawMode()
!= GLES20.GL_LINE_STRIP
                    && objData.getDrawMode() != GLES20.GL_LINE_LOOP) {
                // Log.d("ModelRenderer","Drawing wireframe model...");
                try{
                    // Only draw wireframes for objects having faces (triangles)
                    Object3DData wireframe = wireframes.get(objData);
                    if (wireframe == null || changed) {
                        Log.i("ModelRenderer","Generating wireframe model...");
                        wireframe = Object3DBuilder.buildWireframe(objData);
                        wireframes.put(objData, wireframe);
                    }
                    drawerObject.draw(wireframe, projectionMatrix, viewMatrix,
wireframe.getDrawMode(),
                            wireframe.getDrawSize(), textureId, lightPosInWorldSpace,
                            colorMask, cameraPosInWorldSpace);
                    animator.update(wireframe, scene.isShowBindPose());
                }catch(Error e){
                    Log.e("ModelRenderer",e.getMessage(),e);
                }
            }

            // draw points
            else if (scene.isDrawPoints() || objData.getFaces() == null ||
!objData.getFaces().loaded()){
                    drawerObject.draw(objData, projectionMatrix, viewMatrix
                            , GLES20.GL_POINTS, objData.getDrawSize(),
                            textureId, lightPosInWorldSpace, colorMask,
cameraPosInWorldSpace);
            }

            // draw skeleton
            else if (scene.isDrawSkeleton() && objData instanceof AnimatedModel &&
((AnimatedModel) objData)
                    .getAnimation() != null){
                Object3DData skeleton = this.skeleton.get(objData);
                if (skeleton == null){
                    skeleton = Object3DBuilder.buildSkeleton((AnimatedModel) objData);
                    this.skeleton.put(objData, skeleton);
                }
                animator.update(skeleton, scene.isShowBindPose());
                drawerObject = drawer.getDrawer(skeleton, false, scene.isDrawLighting(),
scene
```

```
                              .isDoAnimation(), scene.isDrawColors());
                  drawerObject.draw(skeleton, projectionMatrix, viewMatrix,-1,
lightPosInWorldSpace, colorMask, cameraPosInWorldSpace);
               }

               // draw solids
               else {
                     drawerObject.draw(objData, projectionMatrix, viewMatrix,
                           textureId, lightPosInWorldSpace, colorMask,
cameraPosInWorldSpace);
               }

               // Draw bounding box
               if (scene.isDrawBoundingBox() || scene.getSelectedObject() == objData) {
                  Object3DData boundingBoxData = boundingBoxes.get(objData);
                        if (boundingBoxData == null || changed) {
                            boundingBoxData = Object3DBuilder.buildBoundingBox(objData);
                            boundingBoxes.put(objData, boundingBoxData);
                        }
                        Object3D boundingBoxDrawer = drawer.getBoundingBoxDrawer();
                  boundingBoxDrawer.draw(boundingBoxData, projectionMatrix, viewMatrix, -1,
                              lightPosInWorldSpace, colorMask, cameraPosInWorldSpace);
               }

               // Draw normals
               if (scene.isDrawNormals()) {
                  Object3DData normalData = normals.get(objData);
                  if (normalData == null || changed) {
                     normalData = Object3DBuilder.buildFaceNormals(objData);
                     if (normalData != null) {
                        // it can be null if object isnt made of triangles
                        normals.put(objData, normalData);
                     }
                  }
                  if (normalData != null) {
                     Object3D normalsDrawer =
drawer.getDrawer(normalData,false,false,scene.isDoAnimation(),
                           false);
                     animator.update(normalData, scene.isShowBindPose());
                     normalsDrawer.draw(normalData, projectionMatrix, viewMatrix, -1, null,
                                 lightPosInWorldSpace, cameraPosInWorldSpace);
                  }
               }

               // TODO: enable this only when user wants it
               // obj3D.drawVectorNormals(result, viewMatrix);
            } catch (Exception ex) {
               Log.e("ModelRenderer","There was a problem rendering the object
'"+objData.getId()+"':"+ex.getMessage(),ex);
            }
         }
      }

      public int getWidth() {
         return width;
      }

      public int getHeight() {
         return height;
      }

      public float[] getModelProjectionMatrix() {
```

```
        return projectionMatrix;
    }

    public float[] getModelViewMatrix() {
        return viewMatrix;
    }
}
```

ModelSurfaceView.java

```
package com.example.sumyhistour.model3D.view;

import android.opengl.GLSurfaceView;
import android.view.MotionEvent;

import com.example.sumyhistour.model3D.controller.TouchController;


import java.io.IOException;

/**
 * This is the actual opengl view. From here we can detect touch gestures for example
 *
 * @author andresoviedo
 *
 */
public class ModelSurfaceView extends GLSurfaceView {

    private ModelActivity parent;
    private ModelRenderer mRenderer;
    private TouchController touchHandler;

    public ModelSurfaceView(ModelActivity parent) throws IllegalAccessException,
IOException {
        super(parent);

        // parent component
        this.parent = parent;

        // Create an OpenGL ES 2.0 context.
        setEGLContextClientVersion(2);

        // This is the actual renderer of the 3D space
        mRenderer = new ModelRenderer(this);
        setRenderer(mRenderer);

        // Render the view only when there is a change in the drawing data
        // TODO: enable this?
        // setRenderMode(GLSurfaceView.RENDERMODE_WHEN_DIRTY);

        touchHandler = new TouchController(this, mRenderer);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        return touchHandler.onTouchEvent(event);
    }

    public ModelActivity getModelActivity() {
        return parent;
    }
```

```
    public ModelRenderer getModelRenderer(){
        return mRenderer;
    }

}
```

MainActivity. xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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"
    android:background="#E5D9ED"

    tools:context=".MainActivity">

    <Button
        android:id="@+id/descrip"
        android:layout_width="181dp"
        android:layout_height="67dp"
        android:layout_marginStart="162dp"
        android:layout_marginTop="354dp"
        android:layout_marginEnd="162dp"
        android:layout_marginBottom="330dp"
        android:background="@drawable/button_states"

        android:text="Description"

        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/route"
        android:layout_width="180dp"
        android:layout_height="64dp"
        android:layout_marginStart="162dp"
        android:layout_marginTop="231dp"
        android:layout_marginEnd="162dp"
        android:layout_marginBottom="432dp"
        android:background="@drawable/button_states"
        android:text="Map"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageButton
        android:id="@+id/amr"
        android:layout_width="50dp"
        android:layout_height="37dp"
        android:layout_marginTop="4dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.955"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
```

```
        app:srcCompat="@drawable/flagofunitedkingdom_6362"
        tools:ignore="MissingConstraints" />

    <ImageButton
        android:id="@+id/ukr"
        android:layout_width="50dp"
        android:layout_height="37dp"
        android:layout_marginStart="292dp"
        android:layout_marginTop="4dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/ukrainianflag_6364" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

MapsActivity. Xml

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <fragment
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MapsActivity" />

    <Button
        android:id="@+id/loc_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginEnd="164dp"
        android:layout_marginBottom="4dp"
        android:background="@drawable/button_states"
        tools:text="Show me" />

</RelativeLayout>
```

Main2. xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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"
    android:background="#E5D9ED"

    tools:context=".MainActivity">

    <Button
```

```
        android:id="@+id/descrip2"
        android:layout_width="181dp"
        android:layout_height="67dp"
        android:layout_marginStart="162dp"
        android:layout_marginTop="354dp"
        android:layout_marginEnd="162dp"
        android:layout_marginBottom="330dp"
        android:background="@drawable/button_states"

        android:text="Опис"

        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/route2"
        android:layout_width="180dp"
        android:layout_height="64dp"
        android:layout_marginStart="162dp"
        android:layout_marginTop="231dp"
        android:layout_marginEnd="162dp"
        android:layout_marginBottom="432dp"
        android:background="@drawable/button_states"
        android:text="Карта"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageButton
        android:id="@+id/amr2"
        android:layout_width="50dp"
        android:layout_height="37dp"
        android:layout_marginTop="4dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.955"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/flagofunitedkingdom_6362"
        tools:ignore="MissingConstraints" />

    <ImageButton
        android:id="@+id/ukr2"
        android:layout_width="50dp"
        android:layout_height="37dp"
        android:layout_marginStart="292dp"
        android:layout_marginTop="4dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/ukrainianflag_6364" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

DescrU. xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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"
    android:background="#E5D9ED"
    tools:context=".MainActivity">

    <ListView
        android:id="@+id/list"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="16dp"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:layout_marginRight="16dp"
        android:layout_marginBottom="16dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

DescrU2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/ackgr"
    android:orientation="vertical"
    android:padding="16dp">


    <TextView
        android:id="@+id/name1"
        android:layout_width="379dp"
        android:layout_height="39dp"
        android:background="#99FFFFFF"
        android:textStyle="bold"

        android:textAppearance="?android:attr/textAppearanceListItemSmall"
        android:gravity="center"
        android:paddingStart="?android:attr/listPreferredItemPaddingStart"
        android:paddingEnd="?android:attr/listPreferredItemPaddingEnd"
        android:minHeight="?android:attr/listPreferredItemHeightSmall"
        android:text="TextView" />

    <TextView
        android:id="@+id/year"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:background="#99FFFFFF"
        android:textStyle="bold"
```

```
        android:textAppearance="?android:attr/textAppearanceListItemSmall"
        android:gravity="center"
        android:paddingStart="?android:attr/listPreferredItemPaddingStart"
        android:paddingEnd="?android:attr/listPreferredItemPaddingEnd"
        android:minHeight="?android:attr/listPreferredItemHeightSmall"
        android:text="TextView" />

    <Button
        android:id="@+id/but"
        android:layout_width="180dp"

        android:layout_height="61dp"
        android:layout_marginStart="100dp"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="162dp"
        android:layout_marginBottom="432dp"

        android:background="@drawable/button_states"
        android:text="View 3D" />


</LinearLayout>
```