PAŃSTWOWA WYŻSZA SZKOŁA ZAWODOWA
W NOWYM SĄCZU

Instytut Techniczny
Informatyka Stosowana

DOKUMENTACJA PROJEKTOWA

PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

Projekt turystyczny

Autorzy:
Frączek Bartłomiej
Buliszak Tomasz

Prowadzący:
mgr inż. Dawid Kotlarski

Nowy Sącz 2021

Spis treści

1. Ogólne określenie wymagań

Projekt ma za zadanie stworzenie aplikacji mobilnej która będzie zawierała około 20 punktów na mapie danej przestrzeni. W skład 20 punktów będą wchodziły wybrane znane miejsca jak zabytki , instytucje kultury miasta czy nawet punkty widokowe.

Aplikacja będzie w głównej mierze spełniała rolę przewodnika turystycznego w nowoczesnej formie. Przestrzenią która zostanie wykorzystana będzie cała mapa miasta Nowego Sącza lub pewien wycinek okolicy tego miasta. Ilość wykorzystanej mapy do stworzenia aplikacji jest zależna od umiejscowienia punktów. Sama aplikacja będzie w głównej mierze opierała się na klasycznym GPS. Pobór mapy będzie realizowana przez mapę Google by móc po jednorazowym takim poborze wykorzystywać aplikacje nawet w sytuacji braku dostępu do sieci.

20 punktów będą wpisane zawsze na mapę by ułatwić gdy osoba wykorzystująca aplikację w danym czasie znajdzie się właśnie w sytuacji braku dostępu do Internetu. Na podstawie naszej obecnej pozycji pobranej przy pomocy GPS aplikacja będzie mogła pokazać drogę do danej interesującej nas w danej chwili z podanych punktów.

Aplikacja nie poda nam jedynej możliwej drogi dotarcia do celu tylko parę możliwych z jej wariantów. Oczywiste jest to że zostanie nam ukazana także najkrótsza możliwa drogą przy pomocy wykorzystywania odpowiedniego algorytmu.

Dodatkową funkcją aplikacji jest to że jest w stanie wykorzystać czujnik oświetlenia by pomóc podróżującym korzystać z aplikacji o późnych porach dnia. Aplikacja będzie również posiadała jeszcze jeden dodatek którym będzie podsumowanie szybkości przejścia przebytego danego odcinka.

Aplikacja zawarta w projekcie ma wyglądać następująco. Po jej uruchomieniu zostanie nam pokazana mapa miasta Nowy Sącz pobierana na bieżąco przy pomocy GPSa. Po prawej stronie ma znajdować się wysuwanie menu z przyciskami aby ułatwić osobie korzystającej z aplikacji wybrać punkt opcji z menu niż żeby odszukiwała obecnie interesujący ją punkt szukając go po mapie. Z aplikacji ma da się korzystać kiedy ekran telefonu jest wyłączony.

Osoba która zleca wykonanie projektu chce aby aplikacja nie tylko wyznaczała odpowiedniej trasy dla pieszych czy poruszających się autobusem czy samochodem chce również aby określała trasę dla jego prywatnego odrzutowca dostępnymi dla niego punktami parkowania na terenie Nowego Sącza.

Ograniczeniem aplikacji jest to że będzie istniała tylko w wersji dla telefonów z oprogramowaniem Android. Istnieje szansa że aplikacja w wersji finalnej znajdzie się

w sklepie Google Play. W przyszłości aplikację można rozwijać w sposób dodawania kolejnych punktów lub nie ograniczanie się do jednego miasta tylko wykorzystać okolicę znajdującą się wokół niego. Bądź sprawienie aby Projekt turystyczny pojawił się także dla oprogramowania IOS.

Tab. 1.1. Tablica wybranych punktów.

Lp.	Nazwa miejsca
1	Miasteczko Galicyjskie w Nowym Sączu
2	Rynek w Nowym Sączu
3	Zamek Królewski
4	Bazylika Św. Małgorzaty
5	Muzeum Okręgowe w Nowym Sączu
6	Synagoga w Nowym Sączu
7	Kapliczka Św. Jana Nepomucena
8	Kościół Ducha Świętego w Nowym Sączu
9	Las Falkowski
10	Góra Zabełecka
11	Sądecki Park Etnograficzny
12	Dworzec PKP
13	Planty
14	Park Strzelecki
15	Europa II
16	Galeria trzy korony
17	Stadion Imienia ojca Augustyna
18	Małopolskie Dworce Autobusowe S.A. Nowy Sącz
19	Hotel Beskid
20	Instytut Techniczny PWSZ

2. Określenie wymagań szczegółowych

Głównym celem projektu jest stworzenie aplikacji która będzie pełniła rolę poradnika turystycznego w nowoczesnym wydaniu. Przy obecnych zasobach posiadanej wiedzy zostanie napisana w miarę prosta aplikacja w obsłudze. Po uruchomieniu aplikacji przy odpowiedniej biblioteki która korzysta z czujnika GPS zostanie pobrana mapa miasta Nowy Sącz oraz nasza bieżąca lokalizacja. Owa mapa w będzie główną częścią layouta aplikacji.

Z prawej zaś strony będzie znajdowało się wysuwane menu które będzie zawierają 2 przyciski. Po kliknięciu w pierwszy przycisk wyskoczy nam okienko w którym będzie zawarta tabelka w którą będą wpisane wybrane przez klienta punkty przedstawiające różne zabytki, punkty kultury, ważne miejsca dla klienta lub punkty widokowe.

Drugi guzik za to ma ułatwić osobą korzystający w zależności od pory dnia ustawić jasność wyświetlacza z samego punktu aplikacji. Więc po kliknięciu w niego wyświetlone zostanie nam okienko z którego będzie mogli skorzystać aby spełnić wymagania wyżej wymienionej funkcji.

Aplikacja przy pomocy poborze naszej obecnej lokalizacji i wyborze interesującego nas punktu wybranego przy pomocy menu albo przy pomocy wybrania owego punktu zaznaczonego na mapie przy skorzystaniu z odpowiedniego wybranego algorytmu jest wstanie wyznaczyć dla nas długość trasy.

Wyżej wymieniony algorytm będzie w stanie wybrać nam odpowiednią trasę w zależności od naszej obecnej w danej czasie możliwości transportu oraz określi nam przybliżony czas dotarcia do celu.

Z algorytmu wyboru alternatywnej trasy będą mogły skorzystać tylko osoby poruszające się następującymi sposobami: pieszo, samochodem i autobusem.

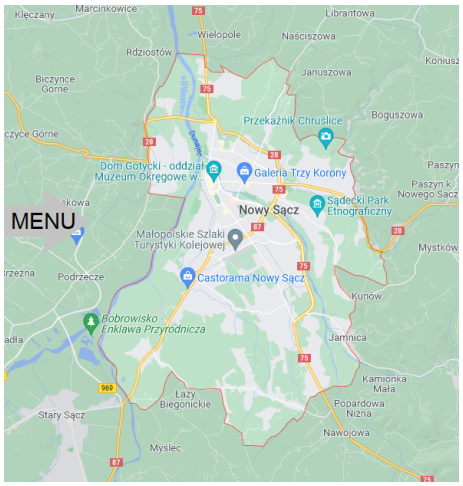
Aplikacja nie jest w stanie spełnić wymagania klienta z punktu obecnej posiadanej przez nas wiedzy określenie dla niego trasy z której mógłby skorzystać swoim prywatnym odrzutowcem a także określić dla niego miejsca do zaparkowania go w bliskiej odległości danego punktu. Wiarygodność do słów klienta iż posiada swój prywatny odrzutowiec jest niska.

Aplikacja także nie jest wstanie po części spełnić wymagania że można z niej korzystać podczas wyłączonego ekranu telefonu. Gdyż nie jest możliwe korzystanie z aplikacji która opiera się na ukazywaniu mapy gdy ekran nie jest włączony. Występuje tu błąd logiczny ponieważ przy obecnej dostępnej dla nas technologii tylko nie wielka część aplikacji jest wstanie działać przy wyłączonym ekranie typu latarka.

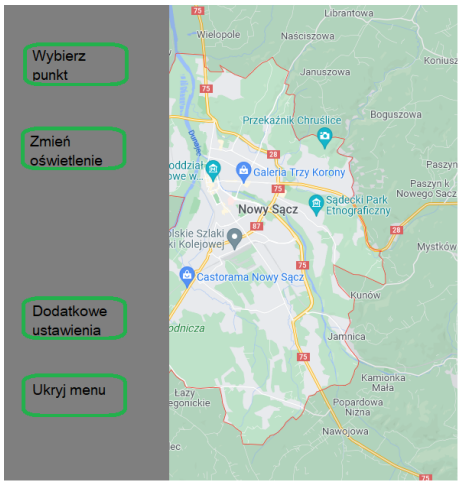
Aplikacja będzie obsługiwała tylko telefony z systemem Android oraz będzie znajdowała się w sklepie Google Play.

W przyszłości aplikację można rozwinąć aplikacje w następujący sposoby jak dodanie większej ilości punktów, umożliwienie działania aplikacji na systemie IOS albo rozszerzenie aplikacji w sposób dodania okolic miasta Nowy Sącz albo dodania większej ilości miast czy przestrzeni.

Aplikacja jeżeli wykryje jakąś niepożądaną sytuację ze strony osoby korzystającej po prostu się wyłączy oraz nie będzie w stanie włączyć się do czasu zresetowania telefonu.



Rys. 2.1. Layout1



Rys. 2.2. Layout2

3. Projektowanie

Pierwszym narzędziem jaki zostanie przez nas wykorzystanym jest repozytorium jakie oferuje środowkisko git. Środowisko owe będzie wykorzystywane w sposób lokalny jak zarazem poprzez publiczne repozytorium na stronie github.com. Na owej stronie powstanie nasze wspólne repozytorium które będzie zawierało wszystkie pliki wykorzystywane przez nas w trakcie projektu. Git jest przyjemnym narzędziem w użytkowaniu gdyż daje dostęp do poprzednich wersji jeżeli zostanie popełniony błąd w obecnej pracy zawsze można przywrócić obraz z działającą wersję. Pozatym w gicie każdy nasz ruch jest rejestrowany przez to na bieżąco można zobaczyć ile pracy wykonałą dana osoba.

Drugi narzędziem którym będziemy się posługiwać jest program Andorid Studio.Android Studio jest to oficjalne środowisko programistyczne dla systemu operacjyjnego od firmy Google, zbudowane na oprogramowaniu JetBrains InteliJ IDEA i zaprojektowane na specjalnie na potrzeby rozwoju Androida.Środowisko oferuje możliwość pracowania w następujących językach programowaniu: Java, Kotlin,C++.Nasza aplikacja powstanie główniej operacjąc się o sam język Java.

Obecne bilbioteki przez nas używane :

```
import android.Manifest
import android.content.pm.PackageManager
import android.location.Address
import android.location.Geocoder
import android.os.Bundle
import android.widget.SearchView
import androidx.annotation.NonNull
import androidx.core.app.ActivityCompat
import androidx.fragment.app.FragmentActivity
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
import java.io.IOException
```

```
import java.util.List
import android.content.Intent
import android.os.Handler
import android.view.Window
import android.widget.ProgressBar
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
```

Obecny cel naszej aplikacji to stworzenie w pełni funkcjonalnego menu który za pomocą przycisków otwiera kolejne okna z zawartością.

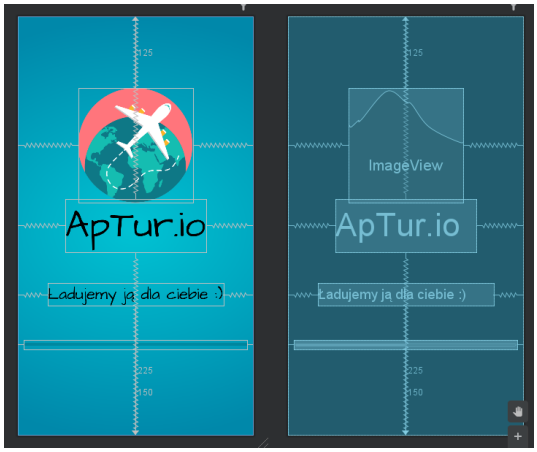
Nasza aplikacja po uruchomeiniu ma pasek ładowania menu który ładuje się około 4. Po załadowania menu mamy dostęp do 2 guzików, pierwszy z nich pełni rolę mapy gps zaś drugi to ustawienia naszej aplikacji.

Zawartość okno głównego :

Guzik numer 1 który pełni po kliknięciu rolę GPS.

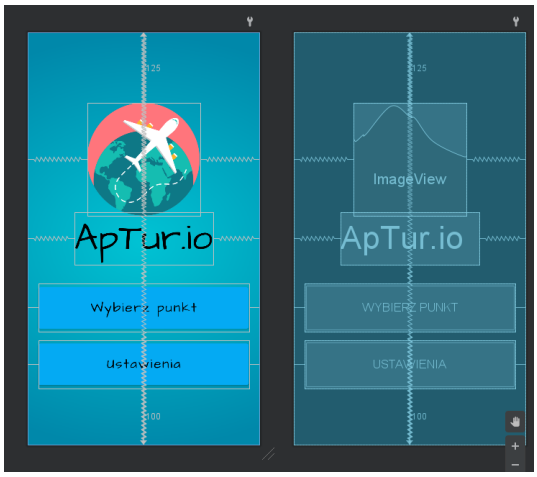
Guzik numer 2 który jest odpowiedzialny za ustawienia

4. Implementacja



Rys. 4.1. Layout podczas ładowania

Na rysunku 4.1 widzimy projekt naszego pierwszego layouta który pojawia się podczas włączenia aplikacji. Layout owy składy się z ikony naszej aplikacji, napisu który zawiera naszą nazwę aplikacji, napis powitający użytkownika oraz pasek ładowania. Pasek ładowania pełni najważniejszą rolę ponieważ od niego zależy czy dostaniemy dostęp do funkcji aplikacji.



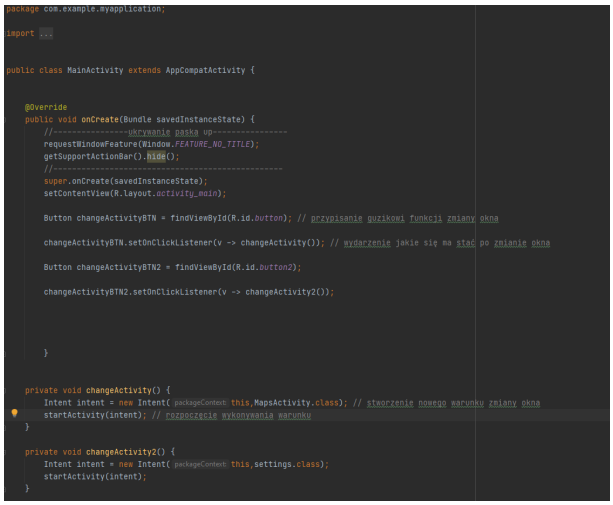
Rys. 4.2. Layout Menu

Na rysunku 4.2 widzimy co się dzieje gdy pasek ładowania zakończy swoją funkcjonalność oraz uzyskamy dostęp do wszystkich funkcji. Layout menu skałada się z ikony naszej aplikacji, napisu który zawiera naszą nazwę aplikacji oraz 2 przycisków. Pierwszy przycisk ma za zadania przekierwoać nasz do mapy. Zaś drugi przycisk mam pełnić rolę podstawowych ustawień.

```
public class SplashActivity extends AppCompatActivity {

    private ProgressBar mProgressBar;
    private int mProgressStatus = 0;

    @Deprecated
    private Handler mHandler = new Handler();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //----------------ukrywanie paska up----------------
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getSupportActionBar().hide();
        //--------------------------------------------------
        setContentView(R.layout.activity_splash);
        @Deprecated
        Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                startActivity(new Intent( packageContext: SplashActivity.this,MainActivity.class));
                finish();
            }
        }, delayMillis: 4000);

        mProgressBar = (ProgressBar) findViewById(R.id.progressbar);

        new Thread(new Runnable() {
            @Override
            public void run() {
                while (mProgressStatus < 100){
                    mProgressStatus++;
                    android.os.SystemClock.sleep( ms: 25);
                    mHandler.post(new Runnable() {
                        @Override
                        public void run() { mProgressBar.setProgress(mProgressStatus); }
                    });
                }
            }
        }).start();
```

Rys. 4.3. Skrypt ładowanie menu

Na rysunku 4.3 mamy deklaracja tworzenia paska ładowania który składa sie z funckjci pętli while która ma zadanie załadować nasz pasek i przypisać do niego odpowiedni czas trwania.Możemy też zauważyć która wywołuję dostęp do głównego ekranu aplikacji. Oraz funkcję która uktywa nasz pasek ładowania kiedy proces dobiegnia końca.



```
package com.example.myapplication;

import ...


public class MainActivity extends AppCompatActivity {


    @Override
    public void onCreate(Bundle savedInstanceState) {
        //----------------ukrywanie paska up----------------
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getSupportActionBar().hide();
        //--------------------------------------------------
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button changeActivityBTN = findViewById(R.id.button); // przypisanie guzikowi funkcji zmiany okna

        changeActivityBTN.setOnClickListener(v -> changeActivity()); // wydarzenie jakie się ma stać po zmianie okna

        Button changeActivityBTN2 = findViewById(R.id.button2);

        changeActivityBTN2.setOnClickListener(v -> changeActivity2());




    }


    private void changeActivity() {
        Intent intent = new Intent( packageContext: this,MapsActivity.class); // stworzenie nowego warunku zmiany okna
        startActivity(intent); // rozpoczęcie wykonywania warunku
    }

    private void changeActivity2() {
        Intent intent = new Intent( packageContext: this,settings.class);
        startActivity(intent);
    }
```

Rys. 4.4. Skrypt znajdujący sie w głównym menu

Na rysunku 4.4 mamy ukazane główne menu w formie kodu. Same menu zawiera funckję guzików które po kliknięciu odpalają okienka z odpowiednią zawartością. Za główny proces tego etapu odpowiada funkcja changeActivity.

```
dependencies {

    implementation 'androidx.appcompat:appcompat:1.4.0'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.2'
    implementation 'com.google.android.gms:play-services-maps:18.0.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
    implementation 'com.google.android.gms:play-services-location:18.0.0'
}
```

Rys. 4.5. Implementacje z build gradle

Na rysunku 4.5 możemy zauważyć gotowe biblioteki zasobu build gradle.Na zdjęciu znajdują się wszystkie bilioteki obecnie przez nasz używane do pełnej funckonalności aplikacji.

```
<string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">AIzaSyAZlA4H2PfLuygHhJuBFntwzRfCqFRnR5I</string>
```

Rys. 4.6. Klucz API Google Maps

Na rysunku 4.6 znajduje się nasz klucz API od google.Klucz ten pełni najważnjeszą rolę w całej aplikacji ponieważ główną częścią naszej aplikacji jest mapa oaprta na API google maps.Taki klucz można zdobyć rejestrując swój projekt na stronie console.cloud.google.com. Również na tej stronie możemy korzystać z innych przydatnych dla nas API.

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

Rys. 4.7. Permisje

Na rysunku 4.7 przedstawione zostały permisję którę również jak wyżej wymieniony klucz pełną bardzo wielką rolę a już nie do konkretnej rzeczy jak mapa tylko odpowiadają już za czynośći całej aplikacji. Między innymi my korzystamy z opcji kożysatnia z zasobów telefonu, internetu, aparatu oraz pobraniu naszej lokalizacji.

```
verride
otected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);

  setContentView(R.layout.activity_maps); // ustawienie głównej zawartości na fragment oraz mapę
  searchView = findViewById (R.id.sv_location); //ustawienie wyszukiwania lokalizacji za pomocą wyszukiwarki
  supportMapFragment=(SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.google_map); //ustawienie mapy przy pomocy fragmentu
  searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() { //możliwość wpisywania danych do wyszukiwarki
      @Override
      public boolean onQueryTextSubmit(String query) {
          String location=searchView.getQuery().toString(); //wpisanie podanej lokazlizacji do stringa
          List<Address> addressList=null; // adres wpisywany do listy nie może być pusty
          if(location != null || !location.equals("")){
              Geocoder geocoder=new Geocoder( context: MapsActivity.this);
              try{
                  addressList=geocoder.getFromLocationName(location, maxResults: 1); // wpisanie do listy wyszukiwanej lokalizacji
              }catch(IOException e){
                  e.printStackTrace();
              }
              Address address= addressList.get(0); //adres przybiera 1 lokacje z listy
              LatLng latLng= new LatLng(address.getLatitude(),address.getLongitude()); // ustawienie współrzędnych geograficznych podanego punktu
              googleMap.addMarker(new MarkerOptions().position(latLng).title(location)); // ustawienie zanznika na podany punkt
              googleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng, zoom: 10)); //ustawienie przybliżenia na wyszukaną lokalizację
          }
          return false;
      }

      @Override
      public boolean onQueryTextChange(String newText) {
          return false;
      }
  });
```

Rys. 4.8. Fragment kodu mapy 1

Na rysunku 4.8 widzimy skryp mapy który jest główną częścią aplikacji.Składa się on z fragmentu google maps API oraz naszej wyżej zaprojektowanej wyszukiwarki.Jeżeli zostanie wpisane coś przez użytkownika w miejsce wyszukiwarki nastąpi przypisanie podanej lokalizacji do listy adresów. Następnie adres ten jest pobierany z owej listy a następnie nadawane są mu odpowiednie kordynaty geograficzne by następnie ustawić go już jako odpowiedni marker na mapie z nazwą lokalizacji wpisanej w wyszukiwarkę. Na samym końcu jest ustawione odpowiednie przybliżenie na naszą wyszkukaną lokalizację.

```
verride
blic void onMapReady(@NonNull GoogleMap Map) {
  googleMap=Map;
  if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED && ActivityCompat.
      return; // nadanie możliwości sprawdzenia czy urządzenie ma włączoną usługę lokalizacji oraz czy zezwala na wykorzystanie przez aplikację
  }
  googleMap.setMyLocationEnabled(true); // ustawienie naszej obecnej lokalizacji
```

Rys. 4.9. Fragment kodu mapy 2

Na rysunku 4.9 Jest to kontunuacja skryptu powyżej która pełni funkcję uzupełniającą jaką jest sprawdzenie dostępu do permisji lokalizacji oraz zezwolenie na udostępnianie naszej obecnej lokalizacji.

```
<fragment
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/google_map"
    android:name="com.google.android.gms.maps.SupportMapFragment">

</fragment>


<SearchView
    android:id="@+id/sv_location"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:layout_marginStart="10dp"
    android:layout_marginTop="10dp"
    android:layout_marginEnd="10dp"
    android:layout_marginBottom="10dp"
    android:background="@drawable/bg_round"
    android:elevation="5dp"
    android:iconifiedByDefault="false"
    android:queryHint="Search....">

</SearchView>
```

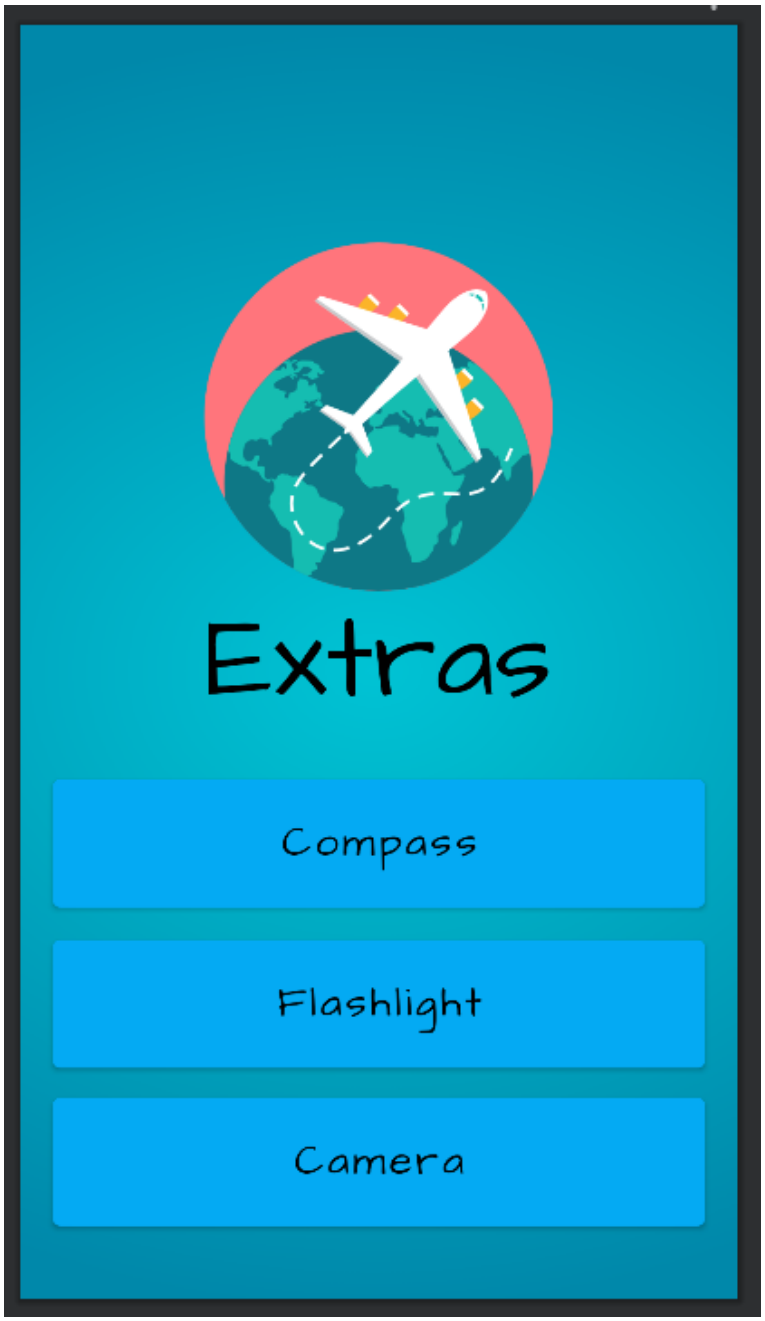
Rys. 4.10. Wyszukiwarka w formie kodu

Na rysynku 4.10 przedstawiana jest wyszukiwarka w formie kodu.Jak można zauważyć wyszukiwarka składa się z gotowego obrazu o nazwie bg round oraz napisu Search a także odpowiednia ustawionych marginesów. Na rysunku możemy też zauważyć fragment mapy od google maps API który jest obierany z ich własnej biblioteki oraz nic nie zmieniany przez nas.

```
<Button
    android:id="@+id/button"
    android:layout_width="374dp"
    android:layout_height="85dp"
    android:layout_marginStart="16dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="16dp"
    android:text="@string/button1"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Rys. 4.11. Guzik w formie kodu

Na rysunku 4.11 jest widoczny jeden z guzików którego używamy jak widać ma przypisane odpowienie swoje miejsce znajdowania na layout, swoje marginesy a także swój tekst znajdujący się w string xml oraz swoją długość i szerokość.



Rys. 4.12. Dodatki

Na rysyunku 4.12 widzimy zawartość wybrania wcześniej guzika dodatki z menu który skałada się z zdjęcia naszej aplikacji z napisu dodatki.Główną częścia są guziki które odpowiadają za funkcję takie jak kompas, latarka i aparat.

```
private void runFlashlight() {

    imageButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (!state)
            {
                CameraManager cameraManager = (CameraManager) getSystemService(Context.CAMERA_SERVICE);

                try {
                    String cameraId = cameraManager.getCameraIdList()[0];
                    cameraManager.setTorchMode(cameraId,  enabled: true);
                    state = true;
                    imageButton.setImageResource(R.drawable.torch_on);
                }
                catch (CameraAccessException e)
                {}
            }
            else
            {
                CameraManager cameraManager = (CameraManager) getSystemService(Context.CAMERA_SERVICE);

                try {
                    String cameraId = cameraManager.getCameraIdList()[0];
                    cameraManager.setTorchMode(cameraId,  enabled: false);
                    state = false;
                    imageButton.setImageResource(R.drawable.torch_off);
                }
                catch (CameraAccessException e)
                {}
            }
        }
    });
```

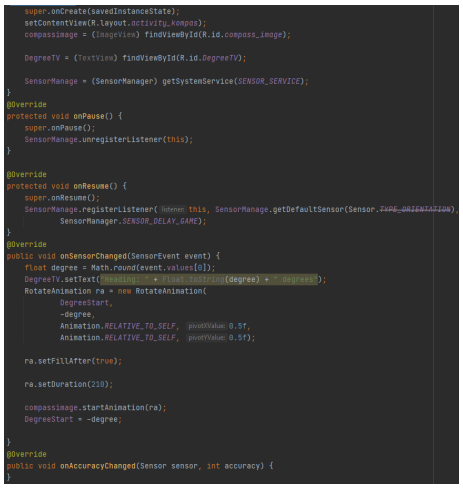
Rys. 4.13. Skrypt latarki

Na rysunku 4.13 przedstawiony jest skrypt na funkcję latarki który jest podzielony na dwie części. Pierwsza część odpowiada za pierwsze kliknięcie jeżli uzyskany został dostęp do aparatu oraz wciśnięty guzik znajdujący się na obrazie latraki zostanie włączona. Druga część skryptu odpowiada co się ma stać jeżeli klikniemy drugi raz guzik wtedy światło laterki pozostaje wyłączone.

```
camera = (Button)findViewById(R.id.buttonó);
camera.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        try {
            Intent intent = new Intent();
            intent.setAction(MediaStore.ACTION_IMAGE_CAPTURE);
            startActivity(intent);
        }
        catch (Exception e){
            e.printStackTrace();
        }
    }
});
```

Rys. 4.14. Skrypt aparatu

Na rysunku 4.14 jest skrypt odpowiedzialny za funckję aparatu jeżeli zostaje uzyskany dostęp do aparatu, po wciśnięciu guzika zostanie uruchomiony nasz aparat znajdujący się w telefonie.



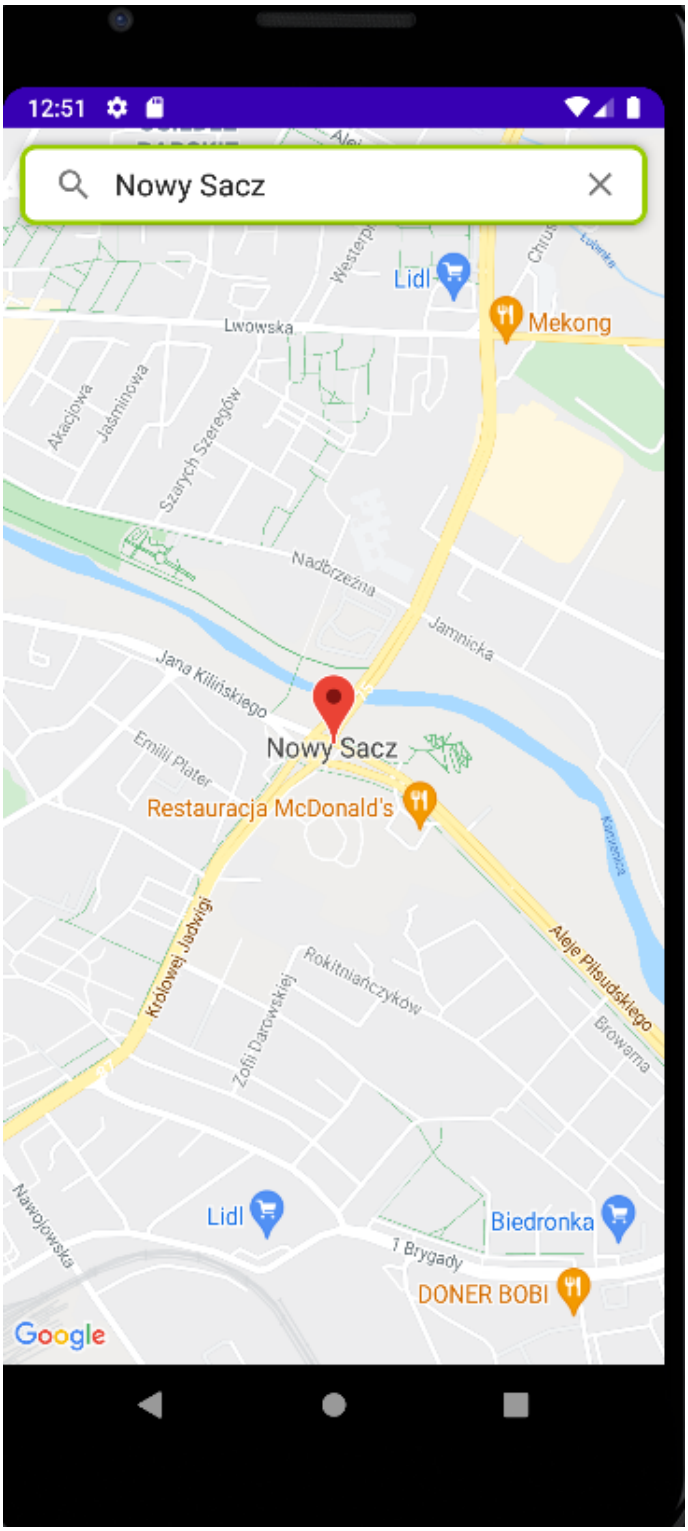
```
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_kompas);
    compassimage = (ImageView) findViewById(R.id.compass_image);

    DegreeTV = (TextView) findViewById(R.id.DegreeTV);

    SensorManage = (SensorManager) getSystemService(SENSOR_SERVICE);
}
@Override
protected void onPause() {
    super.onPause();
    SensorManage.unregisterListener(this);
}

@Override
protected void onResume() {
    super.onResume();
    SensorManage.registerListener( listener: this, SensorManage.getDefaultSensor(Sensor.TYPE_ORIENTATION),
            SensorManager.SENSOR_DELAY_GAME);
}
@Override
public void onSensorChanged(SensorEvent event) {
    float degree = Math.round(event.values[0]);
    DegreeTV.setText("Heading: " + Float.toString(degree) + " degrees");
    RotateAnimation ra = new RotateAnimation(
            DegreeStart,
            -degree,
            Animation.RELATIVE_TO_SELF,  pivotXValue: 0.5f,
            Animation.RELATIVE_TO_SELF,  pivotYValue: 0.5f);

    ra.setFillAfter(true);

    ra.setDuration(210);

    compassimage.startAnimation(ra);
    DegreeStart = -degree;

}
@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
}
```

Rys. 4.15. Skrypt kompasu

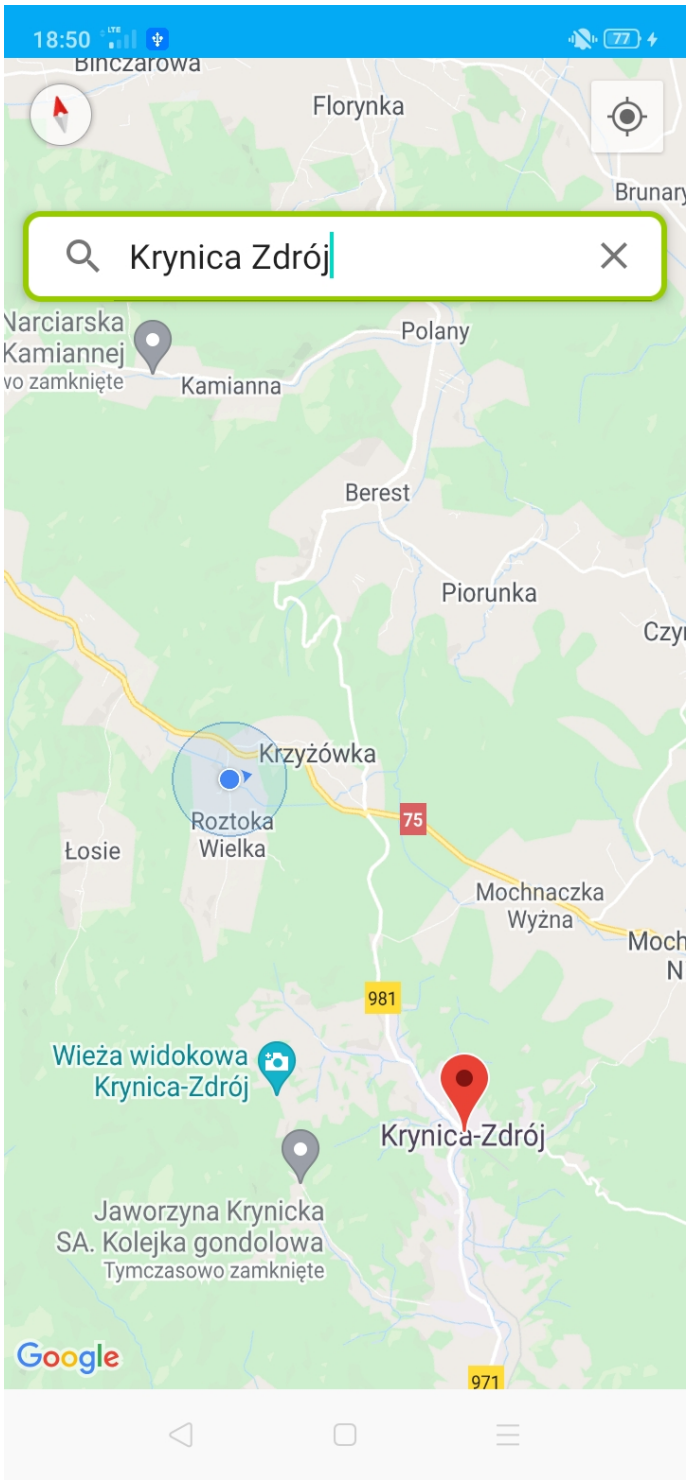
Na rysuknu 4.15 skrypt kompasu wygląda następująco najpierws jest sprawdzane czy zdjęcie kompasu się załadowało następnie jest sprawdzane czy mamy dostęp do sensora ruchu.Podczas sparawdzania jest wywyoływana funckja która zatrzymuje kompas w bezruchu oraz zatrzymuje go także kiedy nie ruchamy naszym telefonem.Następnie przy pomocy danych z naszego sensoru pobieramy ilość stopni oraz animujemy kompas na tej podstawie w odpowiednią stronę świata.

5. Testowanie

Rys. 5.1. Widok aplikacji po wybraniu opcji wybierz punkt

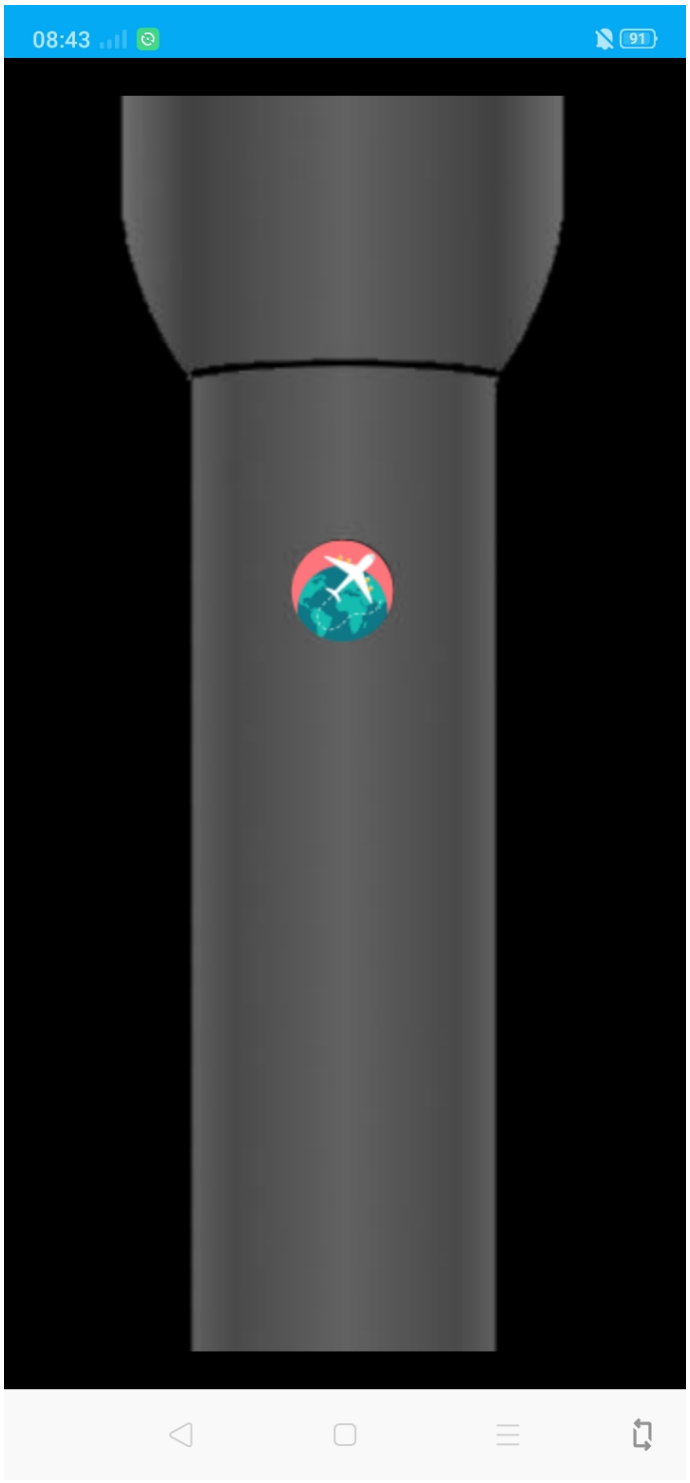


Rys. 5.2. Widok aplikacji po wpisaniu lokalizacji



Rys. 5.3. Widok aplikacji po wykryciu naszej obecnej lokalizacji

Rys. 5.4. Widok aplikacji po wybraniu opcji kompasu



Rys. 5.5. Widok aplikacji po wybraniu opcji latarki

6. Podręcznik użytkownika

Spis rysunków

Spis tabel