



MOBILNE APLIKACIJE

Vežbe 8

Lociranje i mape

2022/2023

Sadržaj

1. Location API	3
1.1 Implementacija	3
2. Google Maps API	5
3. Iscrtavanje rute	15
4. Primer	17

1. Location API

GPS je globalni navigacioni satelitski sistem koji omogućava određivanje lokacije i tačnog vremena. Sastoji se iz svemirskog, kontrolnog i korisničkog segmenta. Svemirski segment čine minimalno 24 satelita koji šalju signal na osnovu koga može da se odredi udaljenost od satelita i tačno vreme. Korisnički segment čine GPS prijemnici, koji primaju taj signal i metodom trilateracije izračunavaju svoju lokaciju i tačno vreme.

LocationManager

Klase *LocationManager* omogućava pristup sistemskom servisu za lociranje. To je glavna klasa Android Location API-a koja omogućava komponentama aplikacije da:

- periodično primaju informacije o lokaciji i azimutu uređaja i
- prime obaveštenje kada se uređaj nađe u blizini zadate lokacije

Lokacija uređaja može da se odredi na 3 načina:

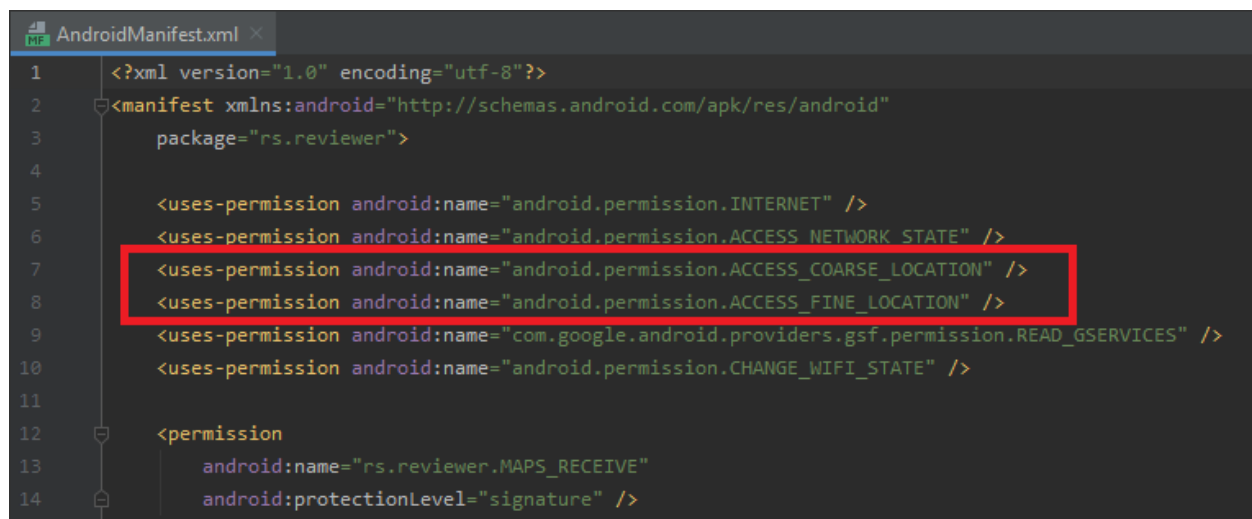
- GPS_PROVIDER (GPS)
- NETWORK_PROVIDER (GSM, Wi-Fi)
- PASSIVE_PROVIDER (informacije o lokaciji koje je dobavila druga aplikacija)

LocationListener

Interfejs *LocationListener* prima obaveštenja od *LocationManager*-a kada se lokacija promeni.

1.1 Implementacija

Prvo smo u *AndroidManifest* datoteci dodali potrebna prava pristupa. Na slici 1 se nalazi primer definisanja statičkih prava pristupa.



```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3    package="rs.viewer">
4
5    <uses-permission android:name="android.permission.INTERNET" />
6    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
7    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
8    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
9    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
10   <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
11
12   <permission
13     android:name="rs.viewer.MAPS_RECEIVE"
14     android:protectionLevel="signature" />
```

Slika 1. Definisanje statičkih prava pristupa

Sledeći korak je da kreiramo klasu *MapFragment* koja implementira *LocationListener* interfejs.

Metode koje implementiramo su:

- *onLocationChanged*
- *onStatusChanged*
- *onProviderEnabled*
- *onProviderDisabled*

onLocationChanged

Ova metoda se poziva svaki put kada uređaj dobije novu informaciju o lokaciji (slika 2). Ova metoda prima nove informacije o koordinatama.

```
@Override
public void onLocationChanged(Location location) {
    Toast.makeText(getActivity(), text: "NEW LOCATION", Toast.LENGTH_SHORT).show();
    if (map != null) {
        addMarker(location);
    }
}
```

Slika 2. Metoda *onLocationChanged*

onStatusChanged

Ova metoda se poziva kada se promeni status provajdera.

onProviderEnabled

Ova metoda se poziva kada je provajder dozvoljen (*enabled*) od strane korisnika.

onProviderDisabled

Ova metoda se poziva kada je provajder isključen (*disabled*) od strane korisnika.

2. Google Maps API

Mapa je umanjena slika zemljine površine na kojoj su prikazani raspored i uzajamna povezanost prirodnih, veštačkih i društvenih pojava.

Google Maps API omogućava korišćenje Google Maps servisa na Android platformi:

- prikaz mape
- rukovanje kamerom (pomeranje, zumiranje mape, itd.)
- postavljanje oznaka, linija, poligona, itd.

Da bi se koristile Google mape potrebno je izvršiti sledeće korake:

1. Preuzeti Google Play Service SDK
2. Preuzeti ključ za korišćenje Google Maps API-a
3. Uneti ključ u *AndroidManifest* datoteku
4. Zatražiti odgovarajuća prava pristupa u *AndroidManifest* datoteci
5. Deklarisati mapu kao pogled u odgovarajućem rasporedu
6. Definisati fragment/aktivnost koja prikazuje raspored

Prvi korak smo već izvršili prilikom inicijalnog konfigurisanja Android Studio-a.

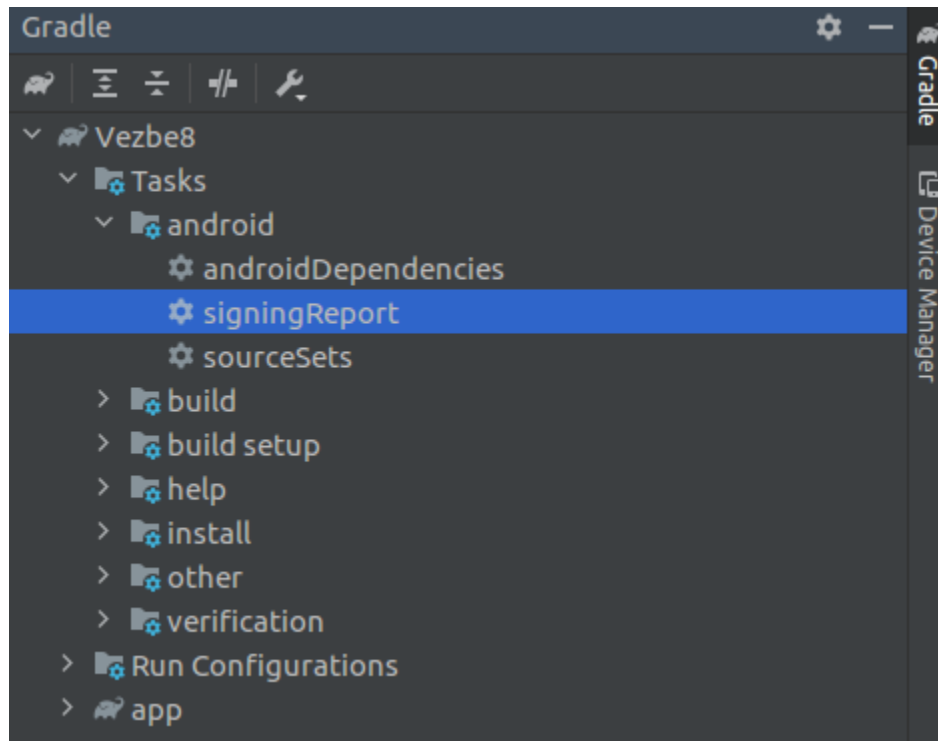
Da biste dobili ključ potrebno je da se registrujete na *Google Cloud Platform Console* i da ostavite kreditnu karticu, ali se sam API ključ ne plaća. Za potrebe primera, omogućiti *Maps SDK for Android*, *Routes API* i *Directions API*. Detalje oko preuzimanja ključa možete pronaći na sledećem linku: <https://developers.google.com/maps/documentation/android-sdk/get-api-key>

Na istom linku nalazi se i uputstvo postavljanja restrikcija nad ključem. Prilikom postavljanja restrikcije, traži se unos applicationId i SHA-1 sertifikata. Ime paketa predstavlja *applicationId* koji se nalazi u *build.gradle* datoteci na nivou modula (Slika 3).

```
5
6  android {
7      namespace 'com.example.vezbe8'
8      compileSdk 32
9
10     defaultConfig {
11         applicationId "com.example.vezbe8"
12         minSdk 27
13         targetSdk 32
14         versionCode 1
15         versionName "1.0"
16     }
```

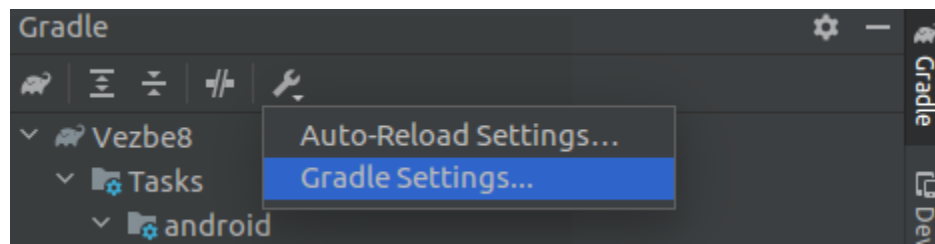
Slika 3. ApplicationId

Klikom na *gradle* paletu u gornjem desnom uglu Android Studio-a otvoriće se prozor sa strukturom projekta kao na slici 4. Dvoklikom na *signingReport* dobićemo ispis SHA-1 sertifikata u konzoli.



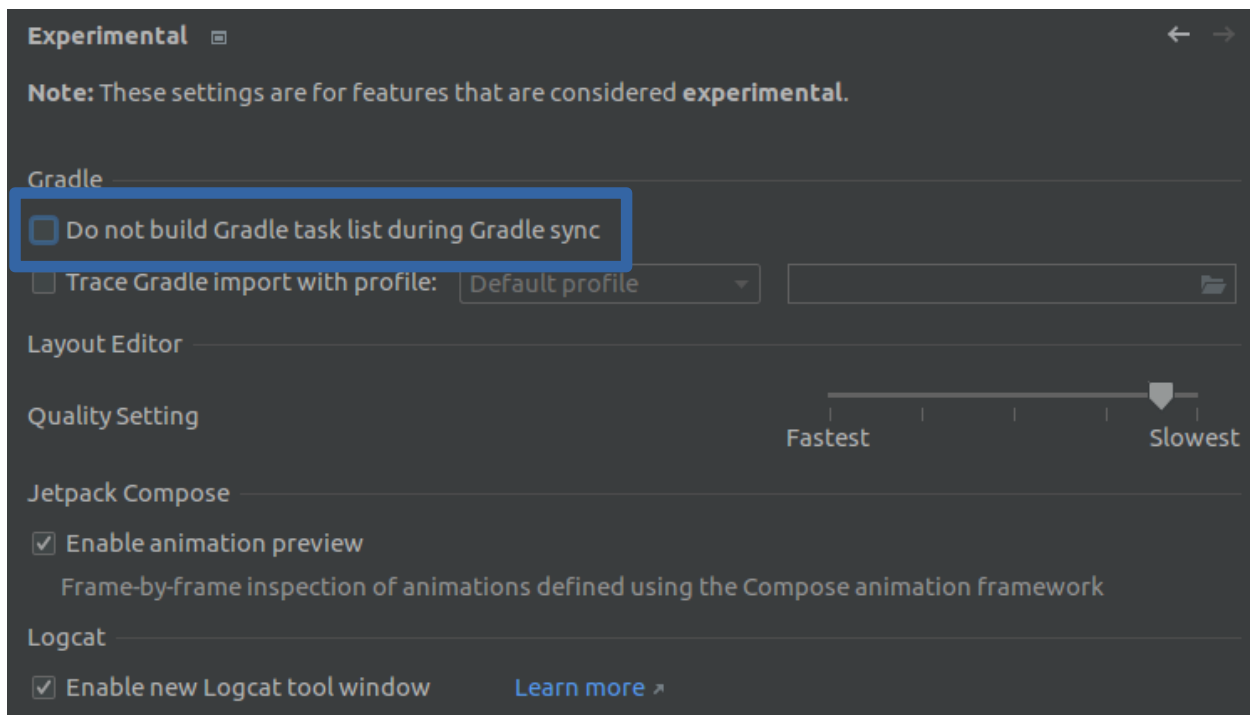
Slika 4. Pokretanje *signingReport*-a za dobijanje sertifikata

Ukoliko ne postoji folder *Tasks* kao na slici 4, potrebno je kliknuti na poslednju stavku menija u *gradle* paleti i izabrati *Gradle Settings...* (Slika 5).



Slika 5. *Gradle Settings*

U *Experimental* odeljku isključiti *Do not build Gradle task list during Gradle sync* opciju (Slika 6). Nakon bildovanja projekta, pojaviće se struktura kao sa slike 4.



Slika 6. *Experimental* tab *Gradle* podešavanja

U *AndroidManifest* datoteci naveli smo dva element *<meta-data>* (slika 7). Prvi element definiše verziju *Google Play* servisa, a u drugi element se unosi ključ.

```
<meta-data
    android:name="com.google.android.gms.version"
    android:value="@integer/google_play_services_version" />
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="YOUR_KEY_GOES_HERE" />
```

Slika 7. Definisanje ključa u *AndroidManifest* datoteci

Umesto unošenja ključa direktno u *AndroidManifest* datoteku, preporučuje se čuvanje ključa u *local.properties* datoteci, nakon čega je potrebno iz *AndroidManifest* datoteke referencirati sačuvani ključ. Za podešavanje ovog dela potrebno je ispratiti korake 3 i 4 sa sledećeg linka: <https://developers.google.com/maps/documentation/android-sdk/config>

Preostali koraci, koji podrazumevaju kreiranje aktivnosti i omogućavanja odgovarajućih prava pristupa, se nalaze u nastavku ovog poglavlja.

Kreirana klasa *MapFragment*, osim što nasleđuje *Fragment* i implementira interfejs *LocationListener*, sad implementira i interfejs *OnMapReadyCallback*.

U metodi *onCreate*, prilikom kreiranja fragmenta, preuzimamo sistemski servis za rad sa lokacijama (slika 8).

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    locationManager = (LocationManager) getActivity().getSystemService(Context.LOCATION_SERVICE);
}

```

Slika 8. Preuzimanje sistemskog servisa za rad sa lokacijama

U metodi *onCreateView* (slika 9) povezali smo naš layout (slika 10) sa *MapFragment*.

```

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup vg, Bundle data) {
    setHasOptionsMenu(true);
    View view = inflater.inflate(R.layout.map_layout, vg, attachToRoot: false);

    return view;
}

```

Slika 9. Metoda *onCreateView*

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">

    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:id="@+id/map_container" />

</LinearLayout>

```

Slika 10. *map_layout*

Kada je fragment vidljiv korisniku, poziva se metoda *onResume* (slika 11) u ovoj metodi prvo pozivamo pomoćnu metodu *createMapFragmentAndInflate* (slika 12).


```

@Override
public void onResume() {
    super.onResume();

    createMapFragmentAndInflate();

    boolean gps = locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
    boolean wifi = locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);

    if (!gps && !wifi) {
        showLocatonDialog();
    } else {
        if (checkLocationPermission()) {
            if (ContextCompat.checkSelfPermission(getContext(),
                Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {

                //Request location updates:
                locationManager.requestLocationUpdates(provider, minTime: 0, minDistance: 0, listener: this);
                Toast.makeText(getContext(), text: "ACCESS_FINE_LOCATION", Toast.LENGTH_SHORT).show();
            } else if (ContextCompat.checkSelfPermission(getContext(),
                Manifest.permission.ACCESS_COARSE_LOCATION) == PackageManager.PERMISSION_GRANTED){

                //Request location updates:
                locationManager.requestLocationUpdates(provider, minTime: 0, minDistance: 0, listener: this);
                Toast.makeText(getContext(), text: "ACCESS_COARSE_LOCATION", Toast.LENGTH_SHORT).show();
            }
        }
    }
}

```

Slika 11. Metoda *onResume*

U pomoćnoj metodi *createMapFragmentAndInflate* specificiramo po kom kriterijumu želimo da dobijemo informacije (GPS, Wifi, mobilni internet).

Klasa *Criteria* govori koji kriterijum za selektovanje *Location* provajdera koristi aplikacija. Pozivanjem konstruktora ove klase, kreiramo novi *Criteria* objekat (linija 71). Novi objekat dobija informacije sa svih izvora, ako korisnik to dopusti.

Na liniji 75 sistemskom servisu prosleđujemo taj kreirani objekat klase *Criteria*, da bismo mogli da dobijamo informacije sa tog izvora.

Naredna linija kreira novu instancu fragmenta, a linije nakon toga vrše zamenu trenutnog prikaza sa prikazom mape.

Na samom kraju pozivamo učitavanje mape. Tu treba voditi računa da ako je asinhrona operacija, lokacije mogu da se dobiju pre mape ili obrnuto.

```

68     private void createMapFragmentAndInflate() {
69         //...
71         Criteria criteria = new Criteria();
72
73         //...
75         provider = locationManager.getBestProvider(criteria, enabledOnly: true);
76
77
78         mMapFragment = SupportMapFragment.newInstance();
79
80         FragmentTransaction transaction = getChildFragmentManager().beginTransaction();
81         transaction.replace(R.id.map_container, mMapFragment).commit();
82
83         //...
86         mMapFragment.getMapAsync( onMapReadyCallback: this);
87     }
88

```

Slika 12. Pomoćna metoda *createMapFragmentAndInflate*

Sledeći korak u metodi *onResume* je da proverimo da li su provajderi odobreni. U slučaju da nisu, pozivamo metodu pomoćnu *showLocationDialog* koja prikazuje dijalog korisniku (slika 13). Taj dijalog traži od korisnika da odobri rad sa lokacijama i ovo je primer dinamičkih prava pristupa (slika 14).

```

public class LocationDialog extends AlertDialog.Builder{
    public LocationDialog(Context context) {

        super(context);

        setUpDialog();
    }

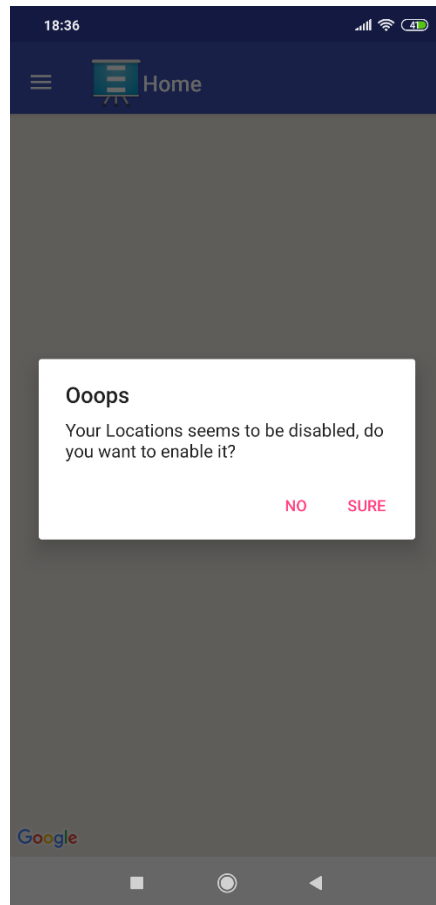
    private void setUpDialog(){
        setTitle(R.string.oops);
        setMessage(R.string.location_disabled_message);
        setCancelable(false);

        setPositiveButton(R.string.sure, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                getContext().startActivity(new Intent(android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS));
                dialog.dismiss();
            }
        });

        setNegativeButton(R.string.no, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                dialog.cancel();
            }
        });
    }
}

```

Slika 13. Klasa *LocationDialog*



Slika 14. Primer kreiranog dijaloga

Ako aplikacija nema prava pristupa koja su joj potrebna, metoda *onResume* poziva metodu *checkLocationPermission*.

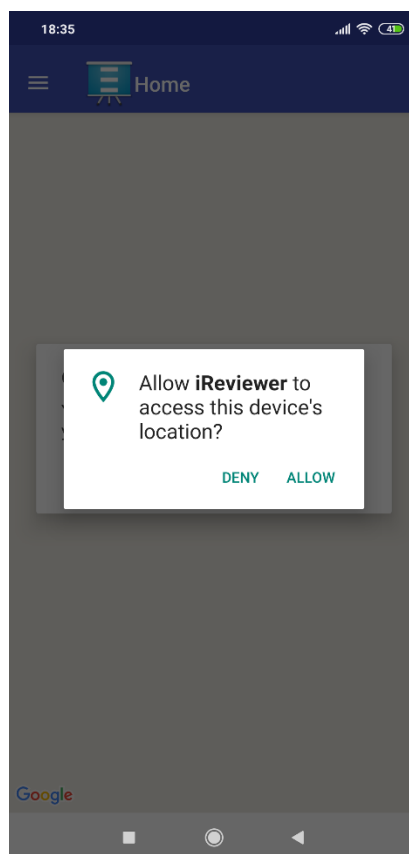
Kod na slici 15 se izvršava ako nije odobreno pristupanje lokaciji (*ACCESS_FINE_LOCATION*), tj. ako je prvi *if* zadovoljen. U drugom *if*-u se pitamo da li treba prikazati objašnjenje iz kog razloga se traži pristup, ako treba ono se prikazuje, ako ne treba samo se traži odobravanje prava pristupa (slika 16).

```

public boolean checkLocationPermission() {
    if (ContextCompat.checkSelfPermission(getActivity(),
        Manifest.permission.ACCESS_FINE_LOCATION)
        != PackageManager.PERMISSION_GRANTED) {
        // Should we show an explanation?
        if (ActivityCompat.shouldShowRequestPermissionRationale(getActivity(),
            Manifest.permission.ACCESS_FINE_LOCATION)) {
            new AlertDialog.Builder(getActivity())
                .setTitle("Allow user location")
                .setMessage("To continue working we need your locations....Allow now?")
                .setPositiveButton( text: "OK", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        //Prompt the user once explanation has been shown
                        ActivityCompat.requestPermissions(getActivity(),
                            new String[]{
                                Manifest.permission.ACCESS_FINE_LOCATION,
                                Manifest.permission.ACCESS_COARSE_LOCATION},
                                MY_PERMISSIONS_REQUEST_LOCATION);
                    }
                })
                .create()
                .show();
        } else {
            ActivityCompat.requestPermissions(getActivity(),
                new String[]{Manifest.permission.ACCESS_FINE_LOCATION,
                    Manifest.permission.ACCESS_COARSE_LOCATION},
                    MY_PERMISSIONS_REQUEST_LOCATION);
        }
    }
}

```

Slika 15. Metoda *checkLocationPermission*



Slika 16. Primer zahteva

Kada korisnik odgovori na zahtev aplikacije za odobravanje pristupa lokaciji, poziva se metoda *onRequestPermissionsResult* (slika 17). Ovoj metodi sistem prosleđuje šta je odgovor korisnika i dalje se pozivaju metode da ažuriraju lokaciju, ako je to omogućeno.

```
@Override
public void onRequestPermissionsResult(int requestCode,
                                     String permissions[], int[] grantResults) {
    switch (requestCode) {
        case MY_PERMISSIONS_REQUEST_LOCATION: {
            // If request is cancelled, the result arrays are empty.
            if (grantResults.length > 0
                && grantResults[0] == PackageManager.PERMISSION_GRANTED) {

                // permission was granted, yay! Do the
                // location-related task you need to do.
                if (ContextCompat.checkSelfPermission(getActivity(),
                    Manifest.permission.ACCESS_FINE_LOCATION)
                    == PackageManager.PERMISSION_GRANTED) {

                    //Request location updates:
                    locationManager.requestLocationUpdates(provider, minTime: 0, minDistance: 0, listener: this);
                }
            } else if (grantResults.length > 0
                && grantResults[1] == PackageManager.PERMISSION_GRANTED){

                // permission denied, boo! Disable the
                // functionality that depends on this permission.
                if (ContextCompat.checkSelfPermission(getActivity(),
                    Manifest.permission.ACCESS_FINE_LOCATION)
                    == PackageManager.PERMISSION_GRANTED) {
                    // ...
                }
            }
        }
    }
}
```

Slika 17. Metoda *onRequestPermissionsResult*

Kada su u pitanju grupe prava pristupa, sistem može automatski da odobrava sva prava pristupa, ako je korisnik prihvatio jedno iz te grupe. Ako tražimo od korisnika da odobri npr. *READ_CONTACTS* i on prihvati, sledi da kada aplikacija bude tražila da se odobri *WRITE_CONTACTS* neće se zahtevati interakcija sa korisnikom, već će se zahtev automatski odobriti, jer je korisnik već prihvatio *READ_CONTACT*, koji je u istoj grupi.

Google Maps API omogućava podešavanja početnog stanja mape deklarativno (u XML dokumentu) ili proceduralno (u Java klasi):

- pozicija kamere (lokaciju, zum, azimut i nagib)
- vrsta mape
- vidljivost kontrola (zum, kompas)
- omogućeni gestovi (za rukovanje kamerom)

Klasa *MapFragment* je implementira metodu *onMapReady* (slike 18 i 19), koju obezbeđuje interfejs *OnMapReadyCallback*.

Tek kada je mapa spremna, možemo da radimo sa njom tj. da reagujemo na razne događaje (dodavanje markera, pomeranje markera, klik na mapu itd).

```

@Override
public void onMapReady(GoogleMap googleMap) {
    map = googleMap;
    Location location = null;

    if (checkLocationPermission()) {
        if (ContextCompat.checkSelfPermission(getContext(),
            Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {

            //Request location updates:
            location = locationManager.getLastKnownLocation(provider);
        } else if (ContextCompat.checkSelfPermission(getContext(),
            Manifest.permission.ACCESS_COARSE_LOCATION) == PackageManager.PERMISSION_GRANTED) {

            //Request location updates:
            location = locationManager.getLastKnownLocation(provider);
        }
    }
}

```

Slika 18. Metoda *onMapReady* 1

```

//ako zelimo da rucno postavljamo markere to radimo
//dodavajuci click listener
map.setOnMapClickListener((latLng) -> {
    map.addMarker(new MarkerOptions()
        .title("YOUR_POSITION")
        .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_RED))
        .position(latLng));
    home.setFlat(true);

    CameraPosition cameraPosition = new CameraPosition.Builder()
        .target(latLng).zoom(14).build();

    map.animateCamera(CameraUpdateFactory.newCameraPosition(cameraPosition));
});

//ako zelimo da reagujemo na klik markera koristimo marker click listener
map.setOnMarkerClickListener((marker) -> {
    Toast.makeText(getActivity(), marker.getTitle(), Toast.LENGTH_SHORT).show();
    return true;
});

```

Slika 19. Metoda *onMapReady* 2

Na samom kraju, definišemo metodu fragmenta *onPause*, jer rad sa lokacijama dosta troši bateriju, pa čim završimo rad sa mapama, to treba i da kažemo.

```

@Override
public void onPause() {
    super.onPause();

    locationManager.removeUpdates(listener: this);
}

```

Slika 20. Metoda *onPause*

3. Iscrtavanje rute

Polyline je niz povezanih segmenata linije koji se koriste za iscrtavanje ruta. Pomoću *PolylineOptions* objekta, *iscrtana linija* se može stilizovati (boja, debljina linije...). Takođe, može se omogućiti reakcija na klik događaj. Primeri i uputstva mogu se pronaći na sledećem linku: <https://developers.google.com/maps/documentation/android-sdk/polygon-tutorial>.

U klasi *DrawRouteFragment* kreirane su dve lokacije koje će biti predstavljene na mapi (Slika 21). Ruta se iscrtava od Madrida do Barselone.

```
LatLng barcelona = new LatLng( latitude: 41.385064, longitude: 2.173403);
mMap.addMarker(new MarkerOptions().position(Barcelona).title("Marker in Barcelona"));

LatLng madrid = new LatLng( latitude: 40.416775, longitude: -3.70379);
mMap.addMarker(new MarkerOptions().position(Madrid).title("Marker in Madrid"));
```

Slika 21. Kreiranje lokacija

Da bismo dobili neophodne putanje, tj. delove putanja koje želimo da iscrtamo, kreiramo *DirectionsAPI* zahtev. Metodi *getDirections* prosleđujemo početnu i krajnju lokaciju (Slika 22).

```
//Execute Directions API request
GeoApiContext context = new GeoApiContext.Builder()
    .apiKey(BuildConfig.MAPS_API_KEY)
    .build();
DirectionsApiRequest req = DirectionsApi.getDirections(context, origin: barcelona.latitude + "," + barcelona.longitude,
    destination: madrid.latitude + "," + madrid.longitude);
```

Slika 22. Kreiranje *Directions API* zahteva

Kao odgovor, između ostalog, dobijamo niz *legs[]*. Jedan *legs* objekat sadrži informacije o ruti između dve definisane lokacije. Ukoliko navodimo samo početnu i krajnju lokaciju, kao u primeru, postojaće samo jedan *legs* objekat. Unutar ovog objekta, nalazi se niz koraka *steps[]* koji sadrži informacije o pojedinačnim delovima rute. Svaki *step* sadrži i enkodovani *polyline* čijim dekodiranjem se dobija *points* objekat koji predstavlja jedan korak putanje. Na slici 23 dat je primer iteracije kroz *legs* i *steps* niz, kao i dekodiranja tačaka putanje.

```

DirectionsResult res = req.await();
//Loop through legs and steps to get encoded polylines of each step
if (res.routes != null && res.routes.length > 0) {
    DirectionsRoute route = res.routes[0];

    if (route.legs != null) {
        for(int i=0; i<route.legs.length; i++) {
            DirectionsLeg leg = route.legs[i];
            if (leg.steps != null) {
                for (int j=0; j<leg.steps.length;j++){
                    DirectionsStep step = leg.steps[j];
                    if (step.steps != null && step.steps.length >0) {
                        for (int k=0; k<step.steps.length;k++){
                            DirectionsStep step1 = step.steps[k];
                            EncodedPolyline points1 = step1.polyline;
                            if (points1 != null) {
                                //Decode polyline and add points to list of route coordinates
                                List<com.google.maps.model.LatLng> coords1 = points1.decodePath();
                                for (com.google.maps.model.LatLng coord1 : coords1) {
                                    path.add(new LatLng(coord1.lat, coord1.lng));
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```

Slika 23. Iteracija kroz segmente putanje

Za iscrtavanje putanje, sve tačke dobijene dekodiranjem *points* objekta, smeštamo u niz i iscrtavamo pomoću metode *addPolyline()* (Slika 24).

```

//Draw the polyline
if (path.size() > 0) {
    PolylineOptions opts = new PolylineOptions().addAll(path).color(Color.BLUE).width(5);
    mMap.addPolyline(opts);
}

```

Slika 24. Iscrtavanje rute na mapi

4. Primer

Domaći se nalazi na *Canvas*-u (*canvas.ftn.uns.ac.rs*) na putanji *Вежбе/08 Вежбе/08 Задачаκ.pdf*.

Primer možete preuzeti na sledećem linku: <https://gitlab.com/antesevicceca/mobilne-aplikacije>.

Za dodatna pitanja možete se obratiti asistentima:

- Svetlana Antešević (svetlanaantesevic@uns.ac.rs)
- Jelena Matković (matkovic.jelena@uns.ac.rs)