LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

Sociāla maršruta aplikācija ar karšu integrāciju

KVALIFIKĀCIJAS DARBS

Autors: Reinis Ziediņš
Studenta apliecības Nr.: RZ15007
Darba vadītājs: Dr. dat. Jānis Zuters

Rīga 2017

Anotācija

Kvalifikācijas darbā tiek aprakstīta sociāla rakstura maršrutu lietotne “Social trails”, kas ir paredzēta viedtālruņiem ar Android operētājsistēmu. Lietotne paredzēta, lai tās lietotāji varētu rīkot izklaidējošus pārgājienus un iepazīt citus tās lietotājus. Tam, lai pārgājieni būtu būtiski konkrētajam lietotājam, applikācija tos rāda Google Maps, vadoties pēc ierīces atrašanās vietas.

Atslēgvārdi: Android, Google maps, Java, Location

Abstract

Social routes application with maps integration

This qualification work describes a social routes application “Social trails”, designed for smartphones with Android operating system. This app is designed so that its users could create entertaining treks and get to meet its other users. This app uses Google Maps and user’s current location, so that the treks would be relevant to the user in question.

Keywords: Android, Google Maps, Java, Location

1. Vārdnīca

Termins	Skaidrojums
Android	Operētājsistēma mobilajām iekārtām, tādām kā viedtālruņi un planšetdatori, kas darbojas uz Linux kodola bāzes.
API I (application programming interface)	Lietojumprogrammas ārējā saskarne. Iepriekš definētu metožu kopums, ko var izmantot citās programmās.
Google Maps	Tīmekļa karšu serviss un tehnoloģija
XML (Extensible Markup Language)	Ir iezīmēšanas valoda, spējīga aprakstīt visdažādākā veida datus. Tās galvenā nozīme ir strukturēta teksta un informācijas koplietošanas atvieglošana Internetā.
SQLite	Ir maza izmēra iegultā relāciju datu bāžu pārvaldības sistēma.
Java	Objektorientēta programmēšanas valoda.

2. Ievads

“Social trails” ir android lietotne, kurā lietotājam ir iespējams izveidot gājienu no konkrēta sākuma punkta līdz konkrētam galamērķim, kas uzrādas kartē, ar kājām gājēja maršrutu pa vidu. Sistēma paredzēta, lai atvieglotu gājienu organizēšanu un stimulētu svešiniekus doties līdzi šādos gājienus un tādā veidā iepazīties.

Lai veiktu šo projektu, man ir nepieciešams apgūt Android lietotņu iztrādes programmu Android Studio un Java programmēšanas valodu ar ko esot ērti un salīdzinoši viegli programmēt android lietotnes.

3. Programmatūras prasību specifikācija

3.1 Ievads

3.1.1 Dokumenta nolūks

Šis Programmatūras prasību specifikācijas (PPS) dokuments ir paredzēts mobīlās lietotnes “Social trails” prasību aprakstīšanai, lai varētu pēc tās vadoties izstrādāt dokumentam atbilstošu sistēmu.

3.1.2 Darbības sfēra

Lietotne paradzēta kā pasākumu organizācijas vietne. Atšķirībā no citiem servisiem, kuros ir iespējams izveidot pasākumus, šajā lietotnē pasākums ir gājienu maršruta formā.

3.1.3 Saistība ar citiem dokumentiem

Dokumentu izveidē ievērotas Latvijas valsts standarta LVS 68:1996„Programmatūras prasību specifikācijas ceļvedis” prasības

3.2 Vispārējais apraksts

2.2.1 Produkta perspektīva

Lai veiktu paradzētās funkcijas, izstrādātā sistēma ir pilnīgi atkarīga no Google Maps API kartēm un funkcijām. Lietotne ir domāta viedtālruņiem ar Android sitēmas versijām ne mazākām par 4.0.

2.2.2 Lietotāju rakstura iezīmes

Lietotājam ir jāprot angļu valoda. Lietotājam ir jābūt pieejams viedtālrunis ar Android operētājsistēmu, interneta pieslēgums un jābūt pamata zināšanām Android OS lietošanā.

2.2.3 Vispārējie ierobežojumi

- Mobīlai lietotnei jāspēj strādāt uz Android 4.0 versijas vai jaunākas.
- Gājienu maršrutu attēlošana un izveide tiek veikta izmantojot Google Maps API.

3.3 Funkcionālās prasības

3.3.1 Gājienu maršrutu izveide

Mērķis:

Funkcija paredzēta, lai lietotāja ievadītais gājiena sākuma un beigu punkts tiktu saglabāts un attēlots uz kartes, atbilstoši kājām gājēja maršrutam.

Ievade:

- Gājiena sākumpunkts
- Gājiena beigu punkts
- Gājiena norises datums un laiks

Apstrāde:

Dati tiek saglabāti un nosūtīti Google Maps servisiem, kas tos apstrādā un atsūta aprēķinātu kājām gājēja īsāko maršrutu, starp šiem diviem punktiem.

Izvade:

Starp saņemtajiem diviem punktiem tiek attēlots gājēja maršruts kartē.

3.3.2 Pašreizējā atrašanās vieta

Mērķis:

Funkcija parāda lietotāja atrašanos vietu kartē, lai pēc tās vadoties varētu izvēlēties sev tīkamu gājienu.

Ievade:

Nav.

Apstrāde:

Android mobīlās ierīces lokācijas dati tiek apstrādāti ar Google Maps API.

Izvade:

Lietotāja atrašanās vieta uz kartes.

Mērķis:

Funkcija paradzēta, lai lietotājs vadoties pēc savas atrašanās vietas kartē redzētu sev būtiskus gājienus (laika posms).

3.3.3 Gājienu apskate

Ievade:

Vēlamais gājiena norises laiks (nav obligāts)

Apstrāde:

Atkarībā no izvēles, tiek atlasīti gājiena dati ar atbilstošu norises laiku. Dati tiek apstrādāti ar Google Maps API un tiek izveidots maršruts.

Izvade:

Atlasītie gājieni un to maršruti attēloti kartē.

3.4 Ārējā saskarne

3.4.1 Lietotāja saskarne

Lietotāja saskarnei jābūt ērti lietojamai, tādai, lai lietotājs varētu ar to strādāt bez īpašas sistēmas lietošanas apmācības, tāpēc lietotāja saskarne tiek veidota saskaņā ar Android iztradātāju vadlīnijām. Iztrādājot lietotāja saskarni, jāņem vērā, ka Android ierīces ir ar dažādu izmēru ekrāniem.

3.4.2 Aparatūras saskarne

Lietotne darbojas uz Android operētājsistēmas, minimālā paradzētā versija ir 4.0, Ice Cream Sandwich, maksimālā 7.1.2, Nougat.

4. Programmatūras projektējuma apraksts:

4.1 Ievads

4.1.1 Dokumenta nolūks

Programmas projektējuma apraksta nolūks ir aprakstīt vēlamo sistēmas uzbūvi atbilstoši prasību specifikācijā izvirzītajām prasībām. Dokuments paradzēts programmatūras iztrādātājiem un to uzturētājiem.

4.1.2 Darbības sfēra

Konkrētais projekts ir Android platformas lietotne, veidota Android Studio iztrādes vidē, kas paradzēta gājienu izveidošanai un attēlošanai uz kartes, izmantojot Google Maps API.

Lietotnes pamatfunkcionalitāte ir īstenota Java programmēšanas valodā. Ekrāna un izvelņu izkārtojums un stils ir aprakstīts XML datnē. Lietotnei ir datubāze, ko pārvalda ar SQLite datu pārvaldības sistēmu.

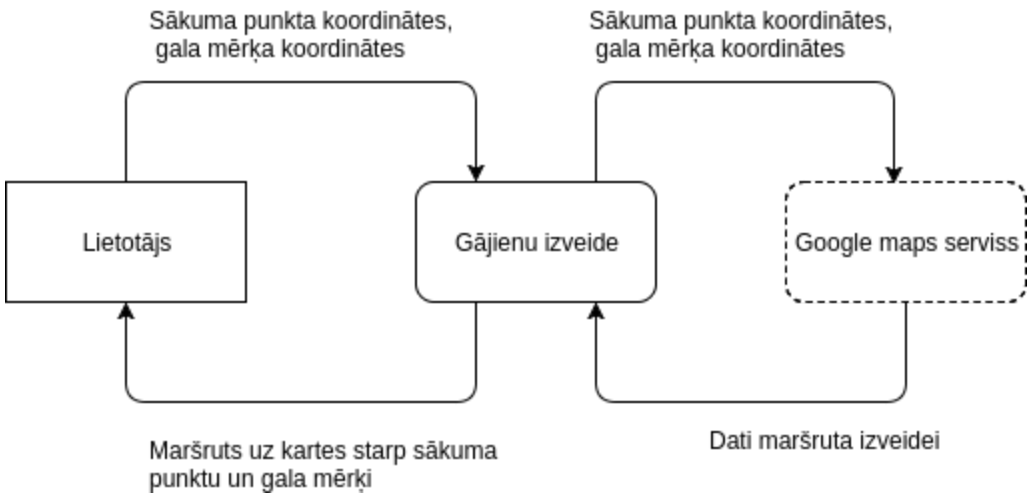
4.1.3 Saistība ar citiem Dokumentiem

Dokumenta noformēšanā ir ievērotas Latvijas valsts standarta LVS 72:1996 „Ieteicamā prakse programmatūras projektējuma aprakstīšanai” prasības un dotā projekta programmatūras prasību specifikācija.

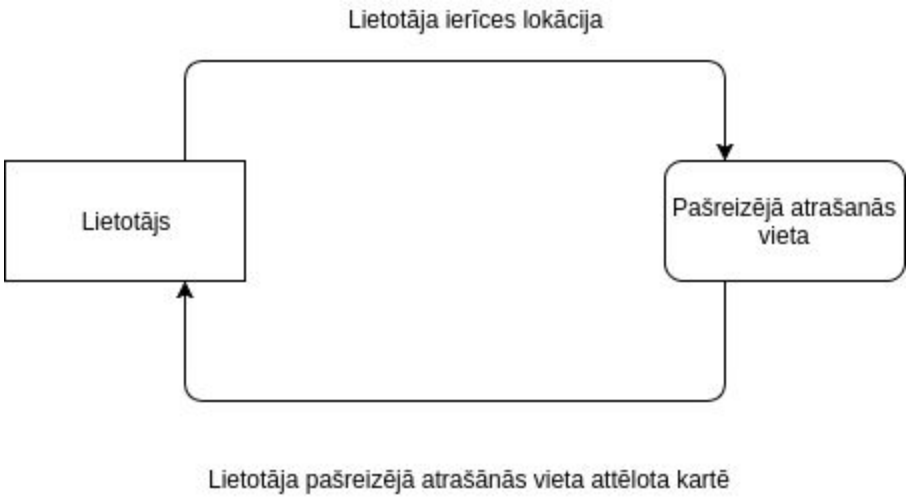
4.2 Datu plūsmu diagramma

4.2.1 0.Līmenis

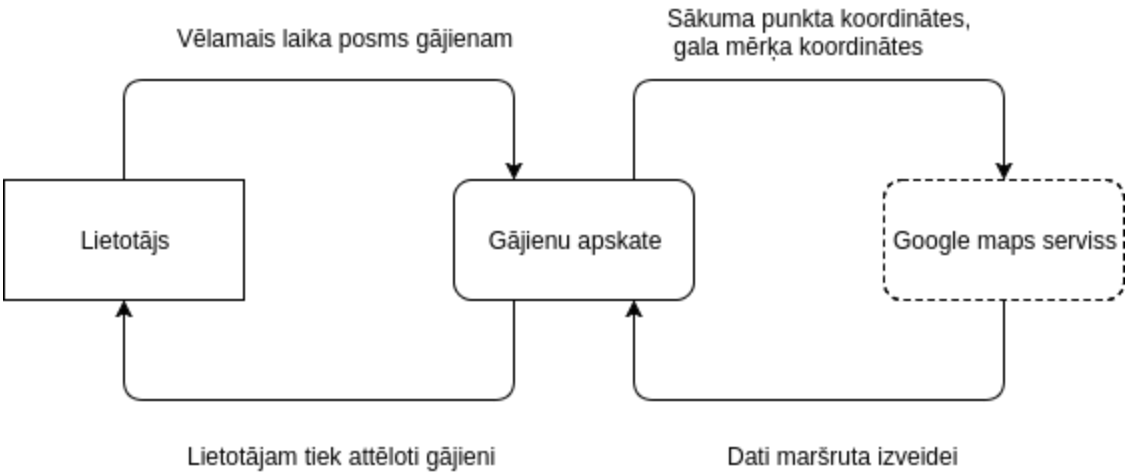
4.2.2 1.Līmenis - Gājiena izveide



4.2.3 1.Līmenis - Lietotāja pašreizējā atrašanās vieta



4.2.4 1.Līmenis - Gājienu apskate



4.4 Modeļu dekompozīcija

4.4.1 Klase MapsActivity

Funkcija : Klase ar lielāko daļu funkciju strādājot ar karti. Atspoguļo lietotājam karti, sevī ietver Android dzīvescikla funkcijas, kā arī Google Maps funkcijas un klausītājus. Šajā klasē tiek inicializēta liela daļa globālo mainīgo skatu un adapteru.

Datne : MapsActivity.java

4.4.2 Klase DBHandler

Funkcija : Klase, kas atļauj piekļūt datubāzei, tādejādi lasot un rakstot datus. Šeit ir definētas gājienu pievienošanas datubāzei un apskatīšanas funkcijas un datubāzes versiju kontrole.

Datne : DBHandler.java

4.4.3 Klase DirectionsJSONParser

Funkcija: Klase, kas veic gājienu maršrutu izveidi un atgriež kartē attēlojumu maršrutu (polyline).

Datne : DirectionsJSONParser.java

4.4.4 Klase Route

Funkcija : Izveido gājiena objektu, ko var padot klasei DBHandler

Datne : Route.java

4.4.5 Klase Splash

Funkcija: Klase, kas pievieno lietotnes ielādes ekrānu (pirms lietotnes palaišanas ekrāns)

Datne: Splash.java

5. Testēšanas dokumentācija

5.1 Skatu testēšana

Visi testi tika veikti uz Nexus 6 viedtālruņa ar Android versiju 5.1.1

5.2 Main screen (Karte)

Paredzētā darbība	Faktiskie rezultāti
Atrodoties kartes ekrānā nospiežot telefona BACK pogu, lietotne nolaižas lejā neislēdzoties.	Lietotne turpina strādāt fonā.
Uzpiežot uz pogas “+” ir iespējams uz kartes izveidot gājienu.	Nākamie klikšķi uz kartes ir novietu gājiena sākuma un beigu punktu.
Spiežot bultiņas ikonu bez gājiena nosaukuma ievadīšanas, izlec kļūdas teksts.	Kļūdas teksts neparādās, labots.
Uzpiežot uz pulksteņa ikonas atveras attiecīgā aktivitāte.	Atveras gājienu filtrēšana pēc norises laika sadaļa.

5.3 Add trail screen

Paredzētā darbība	Faktiskie rezultāti
Spiežot “add” pogu bez datuma vai laika ievades, izlec kļūdas teksts	Kļūdas teksts neparādās, gājiens tiek izveidots, labots.
Veiksmīgi aizpildot formu un spiežot “add” pogu, izlec kartes ekrāns un gājiens tiek saglabāts.	Gājiens tiek saglabāts, bet neizlec kartes ekrāns, labots.
Nospiežot telefona BACK pogu, izlec kartes ekrāns.	Lietotne tiek nolaista lejā.

5.4 Filter trail screen

Paredzētā darbība	Faktiskie rezultāti
Veiksmīgi izvēloties datumu un nospiežot pogu “filter” gājieni kartes ekrānā tiek filtrēti.	Izlec kartes ekrāns un gājieni tiek filtrēti pēc datuma.
Izvēloties pagātnes datumu, parādās kļūdas teksts.	Kļūdas teksts neparādās, labots (nav iespējams izvēlēties pagātnes datumu).

6. Projekta organizācija

Kvalifikācijas darba tēmu izvēlējos patstāvīgi, jo vēlējos izaicināt sevi uz jaunas valodas un OS apguvi. Izaicināju sevi izveidot savu lietotni, kas izmanto Google Maps API, kurai piemīt savs īpašais sociālais raksturs, un tā būtu piemērota cilvēkam pieaugošo komforta vajadzību apmierināšanai. Ņemot vēra to, ka šīs tehnoloģijas man bija pavisam jauns izaicinājums, samērā ilgs laika termiņš tika veltīts tehnoloģiju pamatu un labās prakses apgūšanai.

Tā kā lietotnei nav konkrēta pasūtītāja, programmatūras prasību specifikāciju un programmatūras projektējuma aprakstu izstādāju patstāvīgi.

Projekta organizācija, sakarā ar tēmas patstāvīgu izvēlēšanos un darba veikšanu vienā piegājienā ir ūdenskrituma modelis.

7. Kvalitātes nodrošināšana

Strādājot ar projektu, tika ievēroti programmatūras iztrādes standarti, kuri tika pieminēti PPS un PPA sadaļās. Rakstot kodu, centos ievērot labas programmēšanas praksi t.i. strukturēju kodu saprotamā un viegli lasāmā veidā, koda rezerves glabāju izmantojot GIT, kā arī veicu funkcionālo testēšanu.

8. Darba ietilpības novērtējums

8.1 Darba ietilpības aprēķināšana

Darba ietilpības novērtējumam, tika izmantots COCOMO modelis. COCOMO metodei izmantoju COCOMO II kalkolatoru un koda rindiņu skaitu.

Nr.	Parametrs	Vērtība
Software Scale Drivers		
1	Koda rindiņu skaits	1000
2	Precedentedness	Nominal
3	Development Flexibility	Nominal
4	Architecture / Risk Resolution	Low
5	Team Cohesion	Nominal
6	Process Maturity	Nominal
Software Scale Drivers		
7	Required Software Reliability	Nominal
8	Data Base Size	Low
9	Product Complexity	Nominal
10	Developed for Reusability	Nominal
11	Documentation Match to Lifecycle Needs	Nominal
Personnel		
12	Analyst Capability	Low
13	Programmer Capability	Nominal
14	Personnel Continuity	Low
15	Application Experience	Low

16	Platform Experience	Low
17	Language and Toolset Experience	Low
Platform		
18	Time Constraint	Nominal
19	Storage Constraint	Nominal
20	Platform Volatility	Nominal
Project		
21	Use of Software Tools	Low
22	Multisite Development	Nominal
23	Required Development Schedule	Nominal

Rezultātā - darbietilpības novērtējums 2.6 personmēneši. Programma arī iespējams aprēķināt iespējamo grafiku laiku, kas ir 5 mēneši. Ņemot vērā dokumentācijas rakstīšanas laiku (0.25 personmēneši), COCOMO II modeļā aprēķinātais laiks ir samērā precīzs (2.85 personmēneši).

8.2 Secinājumi

Kvalifikācijas darba rakstīšanas gaitā, ieguvu daudz jaunu zināšanu, konkrēti JAVA programmēšanas valodā un Android studio lietošanā, kā arī paplašināju savas SQL zināšanas. Darba veikšanas sākumā pieļautās pierakstu kļūdas un kopējais laika ierobežojums, manī radīja izpratni par veicamā darba plānošanas jēgu.

Lietotne ir izstrādāta atbilstoši aprakstītajām prasībām, darbietilpīguma novērtējums sakrita ar reālo projekta izstrādes laiku. Vadījos pēc PPA, PPS un varu droši teikt, ka to esamība ļauj izplānot darbam nepieciešamo laiku un ir efektīvs uzskates meteriāls datubāžu veidošanā.

Esmu drošs un parliecināts, ka kvalifkācijas darba izstrādē iegūta pieredze un zināšanas palīdzēs man programmu izveidē nakotnē.

9. Konfigurāciju pārvaldība

Koda uzturēšanai tiek izmantota GIT versiju kontroles sistēma. Datu glabātuves ziņai tiek izmantots serviss „Github", kas šo GIT tehnoloģiju izmanto.

10. Izmantotā literatūra

1. Google Maps Android API resursi - https://developers.google.com/maps/documentation/android-api/
2. Android reference - https://developer.android.com/reference/packages.html
3. Youtube android pamācības - https://www.youtube.com/watch?v=QAbQgLGKd3Y&list=PL6gx4Cwl9DGBsvRxJJOzG4r4k_zLKrnxl
4. LVS 68:1996 „Programmatūras prasību specifikācijas ceļvedis”
5. LVS 72:1996 – „Ieteicamā prakse programmatūras projektējuma aprakstīšanai”
6. COCOMO II darbietilpības aprēķināšanas modulis - http://csse.usc.edu/tools/cocomoii.php

11. Pielikumi

11.1 Lietotnes koda piemēri

11.1.1 Klases DBHandler būtiskās funkcijas (atbild par interakciju ar datubāzi)

```
  //Creation  the route table
@Override
public void onCreate(SQLiteDatabase db) {
  String query = "CREATE TABLE " + TABLE_ROUTES +  "(" +
        COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," +
        COLUMN_ROUTENAME + " TEXT, " +
        COLUMN_STARTLAT + " double, " +
        COLUMN_STARTLNG + " double, " +
        COLUMN_FINISHLAT + " double, " +
        COLUMN_FINISHLNG + " double" +
        ");";
  db.execSQL(query);
}


//Function to add routes to DB
public void addRoute(Route route) {
  ContentValues values = new ContentValues();
  values.put(COLUMN_ROUTENAME, route.get_name());
  values.put(COLUMN_STARTLAT, route.get_startLat());
  values.put(COLUMN_STARTLNG, route.get_startLng());
  values.put(COLUMN_FINISHLAT, route.get_finishLat());
  values.put(COLUMN_FINISHLNG, route.get_finishLng());
  SQLiteDatabase db = getWritableDatabase();
  db.insert(TABLE_ROUTES, null, values);
  db.close();

}

//Functions to get coordinates from DB
public double[] getLatStart() {

  SQLiteDatabase db = getWritableDatabase();
  String query = "SELECT " + COLUMN_STARTLAT + " FROM " + TABLE_ROUTES + " WHERE 1";

  Cursor c = db.rawQuery(query, null);

  c.moveToFirst();
  int i = 0;
```

```
    while(!c.isAfterLast()) {
        i++;
        c.moveToNext();
    }
    double[] List = new double[i];
    c.moveToFirst();
    int j = 0;
    while(!c.isAfterLast()) {
        if(c.getString(c.getColumnIndex(COLUMN_STARTLAT))!= null) {
            List[j] = c.getDouble(c.getColumnIndex(COLUMN_STARTLAT));
        }
        j++;
        c.moveToNext();
    }
    db.close();
    return List;
}
public double[] getLngStart() {

    SQLiteDatabase db = getWritableDatabase();
    String query = "SELECT " + COLUMN_STARTLNG + " FROM " + TABLE_ROUTES + " WHERE 1";

    Cursor c = db.rawQuery(query, null);

    c.moveToFirst();
    int i = 0;
    while(!c.isAfterLast()) {
        i++;
        c.moveToNext();
    }
    double[] List = new double[i];
    c.moveToFirst();
    int j = 0;
    while(!c.isAfterLast()) {
        if(c.getString(c.getColumnIndex(COLUMN_STARTLNG))!= null) {
            List[j] = c.getDouble(c.getColumnIndex(COLUMN_STARTLNG));
        }
        j++;
        c.moveToNext();
    }
    db.close();
    return List;
}
```

11.1.2 Klases MapsActivity būtiskās funkcijas (atbild par interakciju ar Google Maps API)

```
// Placing route marker functionality
 mMap.setOnMapClickListener(new GoogleMap.OnMapClickListener() {
    @Override
    public void onMapClick(LatLng latLng) {
        if (isPlacing) {

            //Clearing markers
            if (markerPoints.size() > 1) {
                markerPoints.clear();
               // mMap.clear();
            }

            markerPoints.add(latLng);

            MarkerOptions options = new MarkerOptions();

            options.position(latLng);

            if (markerPoints.size() == 1) {
                startMarkerLat = latLng.latitude;
                startMarkerLng = latLng.longitude;
                options.icon(BitmapDescriptorFactory.fromResource(R.drawable.start));
            } else if (markerPoints.size() == 2) {
                finishMarkerLat = latLng.latitude;
                finishMarkerLng = latLng.longitude;
                isRoute = true;
                options.icon(BitmapDescriptorFactory.fromResource(R.drawable.flag));
            }
            // Add the new marker
            mMap.addMarker(options);

            if (markerPoints.size() >= 2) {
                //Get first click location, starting point
                LatLng origin = (LatLng) markerPoints.get(0);
                LatLng dest = (LatLng) markerPoints.get(1);

                String url = getDirectionsUrl(origin, dest);

                DownloadTask downloadTask = new DownloadTask();

                downloadTask.execute(url);
            }
        }
    }
});

 //Adding zoom out/in functionality
 mMap.getUiSettings().setZoomControlsEnabled(true);
```

```
  //Adding show current location functionality
  if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED || ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
      mMap.setMyLocationEnabled(true);
  }




}
//Add route to database
public void _addRoute(View view) {
  if (isRoute) {
      Route route = new Route(_inputRoute.getText().toString(), startMarkerLat, startMarkerLng, finishMarkerLat,
finishMarkerLng);
      dbHandler.addRoute(route);
      _createTrail(view);
      printDatabase();

  }
}
//Enter/Cancel create a trail mode
public void _createTrail(View view) {
  Button createTrailButton = (Button)findViewById(R.id.createTrailButton);
  if (isPlacing) {
      isPlacing = false;
      createTrailButton.setText("Create Trail");
      if (markerPoints.size() > 1) {
          markerPoints.clear();
       /*   mMap.clear();*/
      }

  }
  else  {
      isPlacing = true;
      createTrailButton.setText("Cancel");
  }

}
//Set current location marker
locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED || ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
  //Check if network is enabled
  if (locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER)) {
```

```
    locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 60, 0, new
LocationListener() {
        @Override
        public void onLocationChanged(Location location) {
            //get current latitude
            double latitude = location.getLatitude();
            //get current longitude
            double longitude = location.getLongitude();
            LatLng latLng = new LatLng(latitude, longitude);
            Geocoder geocoder = new Geocoder(getApplicationContext());
            try {
                List<Address> addressList = geocoder.getFromLocation(latitude, longitude, 1);
                String str = addressList.get(0).getLocality() + ",";
                str += addressList.get(0).getCountryName();
                mMap.addMarker(new MarkerOptions()
                        .position(latLng)
                        .title(str)
                        .icon(BitmapDescriptorFactory.fromResource(R.drawable.current))
                );
                mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    });
  }
...
```

Kvalifikācijas darbs „Sociāla maršruta aplikācija ar karšu integrāciju” izstrādāts Latvijas Universitātes Datorikas fakultātē.

Ar savu parakstu apliecinu, ka darbs izstrādāts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: *Reinis Ziediņš* _____ ___.05.2017.

Rekomendēju darbu aizstāvēšanai
Darba vadītājs: *Dr. dat. Jānis Zuters* _____ ___.05.2017.

Recenzents: *M.dat., Aleksandrs, Zeļenkovs*

Darbs iesniegts 29.05.2017.
Kvalifikācijas darbu pārbaudījumu komisijas sekretārs: *Darja Solodovņikova*_____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē
___.06.2017. prot. Nr. _____

Komisijas sekretārs(-e): _____