

Android alapú szoftverfejlesztés



Helymeghatározás és térképkezelés Android platformon

Tartalom

1	Felkészülés a mérésre	2
2	Laborfeladatok	2
2.1	Kezdő alkalmazás váz elkészítése	2
2.2	Helymeghatározás	4
2.3	Fragment-ek elkészítése	6
2.3.1	LocationDashboardFragment elkészítése	7
2.3.2	FriendsListFragment elkészítése	12
2.4	Új elem felvitele a névjegyzékből	19
2.5	Térkép megjelenítése	21
2.5.1	Kulcs igénylése	22
2.5.2	Manifest paraméterek	24
2.5.3	Térkép megjelenítése barátokkal	25
2.6	Barát törlésének megvalósítása	28
2.7	További adat megjelenítése a pozíció információk nézetén	28

1 Felkészülés a mérésre

A mérés célja a helymeghatározás és térképkezelés bemutatása. A mérés során egy összetett alkalmazást készítünk el, mely több lapból áll, laponként külön funkcióval. Az egyes lapokat különböző *Fragment*-ekkel, míg a lapozást *ViewPager* komponens segítségével oldjuk meg.

Az alkalmazás megjeleníti a felhasználó pozíció adatait, egy listát biztosít, melyhez fel tudjuk venni a barátainkat és megtekinthetjük a tőlünk való távolságukat, valamint egy térkép nézetén is megtekinthetjük ismerősünk tartózkodási helyét.

A mérés az alábbi témákat érinti:

- Helymeghatározás
- ContentProvider
- Maps V2 API

A labor elején fontos kiemelni, hogy a Maps V2 API sajnos jelenleg Emulátoron még nem működik, nem jeleníthető meg térkép, de a labor ettől függetlenül elvégezhető, mivel a térkép nézet csak kis részét képezi a feladatnak.

A labor során az 5ös érdemjegyhez a térkép megjelenítése nem kötelező tekintettel az API emulátoron való korlátaira.

2 Laborfeladatok

A következőkben egy többlapos, helymeghatározás alapú alkalmazást készítünk el. Az alkalmazás három lapból áll, melyet laponként egy-egy *Fragment* valósít meg. A *Fragment*-ek közti lapozást *ViewPager* komponens segítségével fogjuk megoldani.

Az első lapon egy egyedi nézet megoldás segítségével a pozíció információinkat jelenítjük meg. A második lapon egy listát jelenítünk meg, melyhez a menü segítségével a barátainkat tudjuk felvenni és ha a névjegyzékbe van cím beállítva hozzájuk, a listában láthatjuk a tőlünk való távolságukat. A harmadik nézetén egy térképen jelenítjük meg a barátaink tartózkodási helyét.

2.1 Kezdő alkalmazás váz elkészítése

Első lépésként készítsük el az alkalmazás vázát.

A Manifest állományba vegyük fel a következő engedélyeket:

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```

Vegyük fel a *strings.xml*-be az alábbi szöveges konstansokat:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">FindFriendsLabor</string>
  <string name="hello_world">Hello world!</string>
  <string name="menuAddContact">Személy hozzáadása</string>
  <string name="txtContactPlace">Személy tartózkodási helye:</string>
  <string name="txt_provider">Technológia:</string>
  <string name="txt_latitude">Szélesség:</string>
  <string name="txt_longitude">Hosszúság:</string>
  <string name="txt_speed">Sebesség:</string>
  <string name="txt_alt">Magasság:</string>
  <string name="txt_position_time">Pozíció idő:</string>
  <string name="txt_empty">Üres. - Kérem vegyen fel új elemet a menü
segítségével.</string>
</resources>
```

Az alkalmazás *MainActivity*-jének felülete legyen az alábbi a *res/layout/activity_main.xml* állományban:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity" >

  <android.support.v4.view.ViewPager
    android:id="@+id/mainViewPager"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <!-- PagerTitleStrip is lehet -->

    <android.support.v4.view.PagerTabStrip
      android:id="@+id/PagerTabStrip"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="top"
      android:textSize="22sp" />
  </android.support.v4.view.ViewPager>

</RelativeLayout>
```

Ezzel lehetővé tettük, hogy az *Activity* több *Fragment*-et tudjon megjeleníteni egy lapozható nézetben.

A fő *MainActivity* forrás az alábbi, de mielőtt beillesztenénk, vegyük figyelembe, hogy több dolog is hibás lesz még, mivel hiányoznak későbbi osztályokat, amelyeket hamarosan megvalósítunk.

```
public class MainActivity extends FragmentActivity {

    private static final int REQUEST_PICK_CONTACT = 100;
    private ViewPager pager;
    private MyLocationManager myLocMan;
    private FriendsFragmentPagerAdapter fragmentAdapter;

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

        pager = (ViewPager) findViewById(R.id.mainViewPager);
        fragmentAdapter = new FriendsFragmentPagerAdapter(
            getSupportFragmentManager());
        pager.setAdapter(fragmentAdapter);

        myLocMan = new MyLocationManager(this);
    }

    @Override
    protected void onResume() {
        super.onResume();
        if (myLocMan != null) {
            myLocMan.startLocationMonitoring();
        }
    }

    @Override
    protected void onPause() {
        super.onPause();
        if (myLocMan != null) {
            myLocMan.stopLocationMonitoring();
        }
    }
}
```

Végül a *res/drawable* mappába másoljuk a labor csomagban található képi erőforrásokat.

2.2 Helymeghatározás

Készítsünk egy *MyLocationManager* osztályt például egy külön „location” package-be, mely a helymeghatározásért felelős és új pozíció érkezés esetén *LocalBroadcast*-et küldd az új pozíció értékkel.

Figyeljük meg, hogy egyszerre GPS és hálózat alapú helymeghatározásra is feliratkozunk, valamint hogy a lehető leggyakoribb helymeghatározást állítjuk be. Ez fejlesztés során elfogadható, éles alkalmazásnál azonban legtöbbször nincs rá szükség.

```
public class MyLocationManager implements LocationListener {

    public static final String BR_LOCATION_INFO = "BR_LOCATION_INFO";
    public static final String KEY_LOCATION = "KEY_LOCATION";
    private Context context;
```

```
private LocationManager locMan;

public MyLocationManager(Context aContext) {
    context = aContext;
    locMan = (LocationManager)
context.getSystemService(Context.LOCATION_SERVICE);
}

public void startLocationMonitoring() {
    locMan.requestLocationUpdates(
        LocationManager.GPS_PROVIDER,
        0, 0, this);
    locMan.requestLocationUpdates(
        LocationManager.NETWORK_PROVIDER,
        0, 0, this);
}

public void stopLocationMonitoring() {
    if (locMan != null) {
        locMan.removeUpdates(this);
    }
}

@Override
public void onLocationChanged(Location location) {
    Intent intent = new Intent(BR_LOCATION_INFO);
    intent.putExtra(KEY_LOCATION, location);
    LocalBroadcastManager.getInstance(context).sendBroadcast(intent);
}

@Override
public void onProviderDisabled(String provider) {
    // TODO Auto-generated method stub
}

@Override
public void onProviderEnabled(String provider) {
    // TODO Auto-generated method stub
}

@Override
public void onStatusChanged(String provider, int status, Bundle extras)
{
    // TODO Auto-generated method stub
}
}
```

Figyeljük meg, hogy a *MainActivity onResume()* függvényében feliratkozunk a helymeghatározásra, de az *onPause()* függvényben leiratkozunk róla.

2.3 *Fragment-ek elkészítése*

Első lépésként valósítsunk meg két *Fragment*-et, hogy mielőbb ki tudjuk próbálni a felületet. A *MainActivity*-ben jelenleg a *FriendsFragmentPagerAdapter* osztály megvalósítása hiányzik még, amely a lapozáshoz szükséges. Az osztály feladata, hogy megadja melyik lapon melyik *Fragment* látszódjon, illetve mi a *Fragment*-ekhez tartozó cím.

Elsőként tekintsük a *FriendsFragmentPagerAdapter* forrását, majd utána az egyes *Fragment*-eket. A *FriendsFragmentPagerAdapter*-t egy külön „adapter” package-be valósítsuk meg

```
public class FriendsFragmentPagerAdapter extends FragmentPagerAdapter {

    private LocationDashboardFragment locationDashboardFragment;
    private FriendsListFragment friendsListFragment;

    public FriendsFragmentPagerAdapter(FragmentManager fragmentManager) {
        super(fragmentManager);
        locationDashboardFragment = new LocationDashboardFragment();
        friendsListFragment = new FriendsListFragment();
    }

    @Override
    public Fragment getItem(int pos) {
        switch (pos) {
            case 0:
                return locationDashboardFragment;
            case 1:
                return friendsListFragment;
            default:
                return null;
        }
    }

    @Override
    public CharSequence getPageTitle(int pos) {
        switch (pos) {
            case 0:
                return LocationDashboardFragment.TITLE;
            case 1:
                return FriendsListFragment.TITLE;
            default:
                return null;
        }
    }

    @Override
    public int getCount() {
        return 2;
    }

    public FriendsListFragment getFriendsListFragment() {
        return friendsListFragment;
    }
}
```

2.3.1 LocationDashboardFragment elkészítése

A *Fragment*-ek közül elsőként készítsük el a *LocationDashboardFragment*-et, melynek forrása a következő:

```
public class LocationDashboardFragment extends Fragment {

    public static final String TAG = "LocationDashboardFragment";
    public static final String TITLE = "Pozíció adatok";

    private TextView tvProviderValue;
    private TextView tvLatValue;
    private TextView tvLngValue;
    private TextView tvSpeedValue;
    private TextView tvAltValue;
    private TextView tvPosTimeValue;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        View root = inflater.inflate(R.layout.location_dashboard,
            container, false);
        return root;
    }

    @Override
    public void onViewCreated(View view, Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        initField(R.id.fieldProvider, getActivity().getString(R.string.txt_provider));

        initField(R.id.fieldLat, getActivity().getString(R.string.txt_latitude));
        ;

        initField(R.id.fieldLng, getActivity().getString(R.string.txt_longitude));
        );

        initField(R.id.fieldSpeed, getActivity().getString(R.string.txt_speed));

        initField(R.id.fieldAlt, getActivity().getString(R.string.txt_alt));

        initField(R.id.fieldPosTime, getActivity().getString(R.string.txt_position_time));
    }

    private void initField(int fieldId, String headText) {
        View viewField = getView().findViewById(fieldId);
        TextView tvHead = (TextView) viewField.findViewById(R.id.tvHead);
        tvHead.setText(headText);

        switch (fieldId) {
            case R.id.fieldProvider:
```

```
        tvProviderValue = (TextView)
viewField.findViewById(R.id.tvValue);
        break;
        case R.id.fieldLat:
            tvLatValue = (TextView)
viewField.findViewById(R.id.tvValue);
            break;
        case R.id.fieldLng:
            tvLngValue = (TextView)
viewField.findViewById(R.id.tvValue);
            break;
        case R.id.fieldSpeed:
            tvSpeedValue = (TextView)
viewField.findViewById(R.id.tvValue);
            break;
        case R.id.fieldAlt:
            tvAltValue = (TextView)
viewField.findViewById(R.id.tvValue);
            break;
        case R.id.fieldPosTime:
            tvPosTimeValue = (TextView)
viewField.findViewById(R.id.tvValue);
            break;
        default:
            break;
    }
}

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

    LocalBroadcastManager.getInstance(getActivity()).registerReceiver(
        mMessageReceiver, new
        IntentFilter(MyLocationManager.BR_LOCATION_INFO));
}

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

    LocalBroadcastManager.getInstance(getActivity()).unregisterReceiver(mMe
ssageReceiver);
}

private BroadcastReceiver mMessageReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        Location myLocation =
intent.getParcelableExtra(MyLocationManager.KEY_LOCATION);

        tvProviderValue.setText(myLocation.getProvider());

        tvLatValue.setText(String.valueOf(myLocation.getLatitude()));

        tvLngValue.setText(String.valueOf(myLocation.getLongitude()));

        tvSpeedValue.setText(String.valueOf(myLocation.getSpeed()));

        tvAltValue.setText(String.valueOf(myLocation.getAltitude()));
    }
}
```



```
tvPostTimeValue.setText(new  
Date(myLocation.getTime()).toString());  
}  
};  
}
```

Ebben a *Fragment*-ben a felület beállítása után inicializáljuk a felületi elemeket és referenciát szerzünk azokra a *TextView* komponensekre, melyek a pozíció adatokat fogják megjeleníteni az *initField(...)* függvényben, melyet rendre az *onViewCreated(...)* életciklus függvény végén hívunk meg.

Továbbá figyeljük meg, hogy a *Fragment* *onResume()* és *onPause()* életciklus függvényeiben fel- illetve leiratkozunk arról a **LocalBroadcast** eseményről, melyet a *MyLocationManager* küldd új pozíció érkezésekor.

Amennyiben az *initField(...)* függvényben hibát tapasztalunk az R osztályból jövő konstansok switch-case szerkezetében, akkor ellenőrizzük, hogy a projektet nem-e library projektként hoztuk létre. ha igen, akkor ezt állítsuk vissza normál projektre a *project.properties* file-ban (vagy törléssel, vagy **false**-ra állítással):

```
10 # To enable ProGuard to shrink  
11 #proguard.config=${sdk.dir}/  
12  
13 # Project target.  
14 target=android-17  
15 android.library=true  
16
```

A jelenség oka, hogy a 14-es ADT verzió óta a library projektekben nem garantált hogy az *R.java* tartalma konstans marad (újrafelhasználhatósági okok miatt), így case-be nem rakhatók az itt lévő értékek.

A *LocationDashboardFragment* felhasználói felülete a *res/layout/location_dashboard.xml* állományban az alábbi:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" >  
  
    <ScrollView  
        android:id="@+id/scroller"  
        android:layout_width="fill_parent"  
        android:layout_height="fill_parent"  
        android:fillViewport="true" >  
  
        <LinearLayout  
            android:layout_width="match_parent"  
            android:layout_height="wrap_content"  
            android:orientation="vertical" >  
  
            <include  
                android:id="@+id/fieldProvider"  
                layout="@layout/tile_info" />  

```

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:baselineAligned="false"
    android:weightSum="2" >

    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1" >

        <include
            android:id="@+id/fieldLat"
            layout="@layout/tile_info" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1" >

        <include
            android:id="@+id/fieldLng"
            layout="@layout/tile_info" />
    </LinearLayout>
</LinearLayout>

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:baselineAligned="false"
    android:weightSum="2" >

    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1" >

        <include
            android:id="@+id/fieldSpeed"
            layout="@layout/tile_info" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1" >

        <include
            android:id="@+id/fieldAlt"
            layout="@layout/tile_info" />
    </LinearLayout>
</LinearLayout>

<include
    android:id="@+id/fieldPosTime"
    layout="@layout/tile_info" />

</LinearLayout>
</ScrollView>
```

```
</LinearLayout>
```

A felhasználói felületnél **figyeljük meg a *ScrollView* használatát**, fontos jellemzője, hogy csak egy belső elemet tartalmazhat.

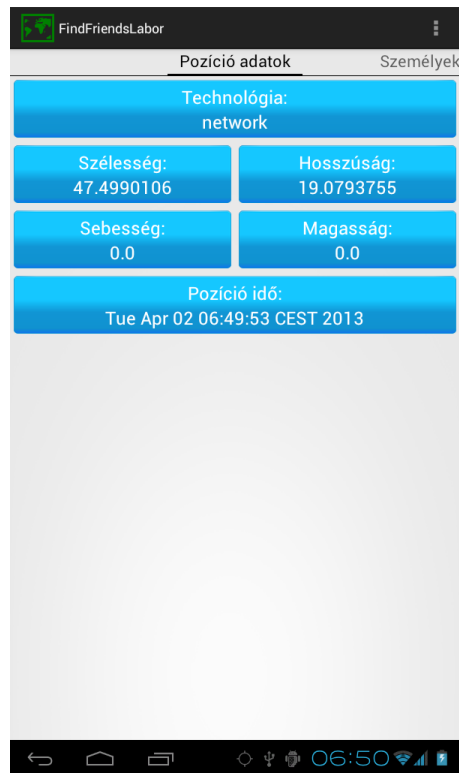
Továbbá **figyeljük meg, hogy *include* paranccsal** hogyan hivatkozunk be egy Dashboard téglalap elemet, amit külön XML-ben írunk le a *res/layout/tile_info.xml*-ben, melynek kódja a következő:

```
<?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="wrap_content"
    android:gravity="center"
    android:orientation="vertical"
    android:background="@drawable/tile_bg"
    android:layout_margin="5dp">

    <TextView
        android:id="@+id/tvHead"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@android:color/white"
        android:textSize="24sp"
        android:text="Data:" />

    <TextView
        android:id="@+id/tvValue"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@android:color/white"
        android:textSize="24sp"
        android:text="0" />

</LinearLayout>
```



Figyeljük meg, hogy a `tile_info` háttérként használt `drawable/tile_bg` valójában a `tile_bg.9.png` állományra hivatkozik ami egy úgynevezett **ninepatch kép**. A ninepatch kép jellemzője, hogy a kép köré rajzolható fekete pontok/vonalak segítségével megadható, hogy az Android milyen irányban méretezze át a képet. Ninepatch képek rajzolásához az Android SDK következő helyéről indítható programmal van lehetőségünk: `AndroidSDK/tools/draw9patch.bat`. **Vizsgáljuk meg** a laborvezető segítségével a `tile_bg.9.png` állományt evvel a programmal.

2.3.2 FriendsListFragment elkészítése

Készítsük el a `FriendsListFragment` `Fragment`-et, melynek feladata a kiválasztott barátok megjelenítése, és annak jelzése, hogy milyen távol vannak tőlünk.

Ennek megvalósításához szükségünk lesz két segéd osztályra, melyeket az adatkezeléshez fogunk használni. Ezeket az osztályokat helyezhetjük külön `data` package-ba.

A labor példában az adatokat csak a memóriában fogjuk tárolni, nem valósítjuk meg a perzisztencia funkciókat, így az alkalmazásból való kilépés után a barátok listája törlődni fog. Ennek megvalósításához szükség esetén használhatnánk a korábbi laborokban megtanult módszereket (pl. SQLite).

A barátok leírását jellemző osztály (`LocationFriend`) feladata a név, a lakcím és a földrajzi koordináta tárolása. Az osztály konstruktorában csak a szöveges címet kapja meg és Geocoding segítségével azt alakítja át földrajzi koordinátákra.

A *LocationFriend* osztály forrása a következő:

```
public class LocationFriend {

    private String name;
    private String address;
    private boolean hasCoords = false;
    private double lat = 0;
    private double lng = 0;

    public LocationFriend(String aName, String aAddress, Context aContext)
    {
        name = aName;
        address = aAddress;
        setCoords(aContext);
    }

    private void setCoords(final Context aContext) {
        new Thread() {
            public void run() {
                try {
                    Geocoder geocoder = new Geocoder(aContext);
                    List<Address> locations = null;
                    locations
geocoder.getFromLocationName(address, 1);
                    lat = locations.get(0).getLatitude();
                    lng = locations.get(0).getLongitude();
                    hasCoords = true;
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }.start();
    }

    public String getName() {
        return name;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public double getLat() {
        return lat;
    }

    public double getLng() {
        return lng;
    }

    public boolean hasCoords() {
        return hasCoords;
    }
}
```

Illetve szükségünk lesz egy *LocationFriendsManager* osztályra, mely *LocationFriend* objektumokat tárol a memóriában, illetve lehetővé teszi azok felvételét, lekérdezését. Az egyszerűség kedvéért a *LocationFriendsManager*-t Singleton mintával valósítjuk meg, hogy könnyen elérhető legyen a kódban. SQLite adatbázissal való összekötés esetén ezt a mintát azonban nem javasoljuk, mivel könnyebben előfordulhat, hogy elfeledkezünk az adatbázis bezárásáról és a Garbage Collector nem tudja felszabadítani. Android platformon mindig óvatosan kezeljük a Singleton alapú megoldásokat.

A *LocationFriendsManager* osztály forrása a következő:

```
public class LocationFriendsManager {

    private static LocationFriendsManager instance = null;

    public static LocationFriendsManager getInstance() {
        if (instance == null) {
            instance = new LocationFriendsManager();
        }
        return instance;
    }

    private ArrayList<LocationFriend> locationFriends;

    public LocationFriendsManager() {
        locationFriends = new ArrayList<LocationFriend>();
    }

    public LocationFriend getLocationFriend(int aIndex) {
        return locationFriends.get(aIndex);
    }

    public int getLocationFriendsNum() {
        return locationFriends.size();
    }

    public void addFriend(LocationFriend locFriend) {
        locationFriends.add(locFriend);
    }

    public ArrayList<LocationFriend> getLocationFriends() {
        return locationFriends;
    }

    public void clear() {
        if (locationFriends != null) {
            locationFriends.clear();
        }
    }

}
```

Végül a barátok lista (*FriendListFragment* – lásd később) kiszolgálásához szükségünk van egy Adapter osztályra, mely a lista tartalommal való feltöltésért és a listaelemek megjelenítéséért felelős. Hozzunk létre egy *LocationFriendsAdapter* osztályt az *adapter* package-ban, melynek forrása:

```
public class LocationFriendsAdapter extends BaseAdapter {

    private Location myLocation = null;
    private Context context;

    public LocationFriendsAdapter(Context aContext) {
        context = aContext;
        LocationFriendsManager.getInstance().clear();
        LocationFriendsManager.getInstance().addFriend(
            new LocationFriend("John Doe", "Zalaegerszeg",
context));
        LocationFriendsManager.getInstance().addFriend(
            new LocationFriend("Jeff Johnson", "Miskolc",
context));
        LocationFriendsManager.getInstance().addFriend(
            new LocationFriend("Test Elek", "Debrecen",
context));
        LocationFriendsManager.getInstance().addFriend(
            new LocationFriend("James Demo", "Budapest",
context));
    }

    public void setMyPosition(Location aMyLocation) {
        myLocation = aMyLocation;
        notifyDataSetChanged();
    }

    static class ViewHolder {
        TextView tvName;
        TextView tvAddress;
        TextView tvDistance;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View v = convertView;
        if (v == null) {
            LayoutInflater inflater = LayoutInflater.from(context);
            v = inflater.inflate(R.layout.friendrow, null);
            ViewHolder holder = new ViewHolder();
            holder.tvName = (TextView)
v.findViewById(R.id.textViewName);
            holder.tvAddress = (TextView)
v.findViewById(R.id.textViewAddress);
            holder.tvDistance = (TextView) v
                .findViewById(R.id.textViewDistance);
            v.setTag(holder);
        }

        final LocationFriend lFriend =
LocationFriendsManager.getInstance()
                .getLocationFriend(position);
        if (lFriend != null) {
            ViewHolder holder = (ViewHolder) v.getTag();
            holder.tvName.setText(lFriend.getName());
            holder.tvAddress.setText(lFriend.getAddress());
            if (myLocation != null && lFriend.hasCoords()) {
                float[] results = new float[1];
```

```
        Location.distanceBetween(myLocation.getLatitude(),
                                myLocation.getLongitude(),
lFriend.getLat(),
                                lFriend.getLng(), results);
        holder.tvDistance.setText("" + results[0] + " m");
    }
}

return v;
}

@Override
public int getCount() {
    return
LocationFriendsManager.getInstance().getLocationFriendsNum();
}

@Override
public Object getItem(int position) {
    return
LocationFriendsManager.getInstance().getLocationFriend(position);
}

@Override
public long getItemId(int position) {
    return position;
}
}
```

Az *LocationFriendsAdapter* osztályba, az egyszerűsítés kedvéért, a konstruktorba felvettünk néhány kezdő adatot.

A hivatkozott *res/layout/friendrow.xml* forrása:

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

    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:layout_alignParentLeft="true"
        android:gravity="center_vertical" >

        <ImageView
            android:id="@+id/imageViewAvatar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="left"
            android:paddingLeft="10dp"
            android:src="@drawable/avatar" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
```



```
        android:layout_height="fill_parent"
        android:layout_alignParentTop="true"
        android:layout_toLeftOf="@+id/linearLayout2"
        android:layout_toRightOf="@+id/linearLayout1"
        android:gravity="center"
        android:orientation="vertical"
        android:paddingLeft="10dp" >

        <TextView
            android:id="@+id/textViewName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="left"
            android:textColor="@android:color/black"
            android:text="Name" />

        <TextView
            android:id="@+id/textViewAddress"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="left"
            android:textColor="@android:color/black"
            android:text="Address" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/linearLayout2"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:layout_alignParentRight="true"
        android:gravity="center"
        android:paddingLeft="10dp" >

        <TextView
            android:id="@+id/textViewDistance"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:gravity="right"
            android:paddingRight="10dp"
            android:text="0 m"
            android:textColor="@android:color/black"
            android:textSize="20sp" />
    </LinearLayout>
</RelativeLayout>
```

Az előzőek alapján a *FriendsListFragment* forrása:

```
public class FriendsListFragment extends ListFragment {

    public static final String TAG = "FriendsListFragment";
    public static final String TITLE = "Személyek";

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

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                          Bundle savedInstanceState) {
    View v = inflater.inflate(R.layout.friendslist, container,
false);
    return v;
}

@Override
public void onActivityCreated(Bundle savedInstanceState) {
    super.onActivityCreated(savedInstanceState);
    setListAdapter(new
LocationFriendsAdapter(getActivity().getApplicationContext()));
}

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

    LocalBroadcastManager.getInstance(getActivity()).registerReceiver(
        mMessageReceiver,                                new
        IntentFilter(MyLocationManager.BR_LOCATION_INFO));
}

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

    LocalBroadcastManager.getInstance(getActivity()).unregisterReceiver(mMe
ssageReceiver);
}

private BroadcastReceiver mMessageReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        Location myLocation =
intent.getParcelableExtra(MyLocationManager.KEY_LOCATION);

        ((LocationFriendsAdapter)getListAdapter()).setMyPosition(myLocation);
    }
};
}
```

A *FriendsListFragment* felületét leíró *res/layout/friendslist.xml* tartalma pedig:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

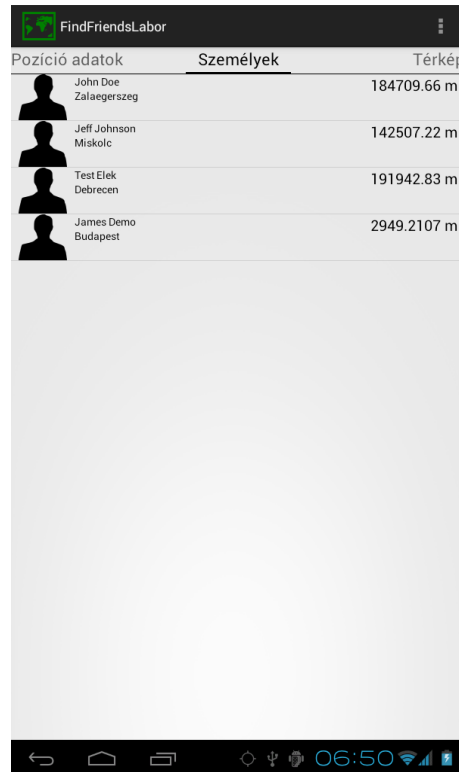
    <ListView
        android:id="@android:id/list"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:drawSelectorOnTop="false" />

    <!-- Üres lista esetén ez jelenik meg -->
```

```
<TextView
    android:id="@android:id/empty"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:text="@string/txt_empty"/>

</FrameLayout>
```

Figyeljük meg milyen módon van megoldva az üres lista esetén a jelzés.



2.4 Új elem felvitele a névjegyzékből

Következő feladatunk, hogy a barátok listájába a menüből tudjunk új elemet felvenni. Ehhez első lépésként egy menü erőforrásra lesz szükségünk, a `res/menu/activity_main.xml`-ben:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/itemAddContact"
        android:title="@string/menuAddContact"/>
</menu>
```

Ezt követően már csak a menü kezelést kell a *MainActivity*-ben megvalósítani helyesen. A személy kiválasztása menüpont esetén a névjegyzéket fogjuk megjeleníteni, melyből választhat a felhasználó és visszatéréskor kiolvassuk a kiválasztott személy nevét,

valamint lakcímét. Az alkalmazás helyes működéséhez olyan nevet válasszunk, akihez van felvéve lakcím, ha nincs ilyen, előtte egy személyhez állítsunk be például egy várost. Egyes emulátorokon nem biztos, hogy működik a névjegy kiválasztás és a cím kiolvasás, ebben az esetben használjunk „demo” adatokat és azokat illesszük be a listába.

Ehhez a *MainActivity* kódját egészítsük ki a következő függvényekkel:

```
@Override
protected void onActivityResult(int requestCode,
    int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == RESULT_OK) {
        switch (requestCode) {
            case REQUEST_PICK_CONTACT:
                Uri contactUri = data.getData();
                if (contactUri != null) {
                    Cursor c = null;
                    Cursor addrCur = null;
                    try {
                        ContentResolver cr =
getContentResolver();
                        c = cr.query(contactUri, null, null,
null, null);
                        if (c != null && c.moveToFirst()) {
                            String id =
c.getString(c.getColumnIndex(BaseColumns._ID));
                            String name =
c.getString(c.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));
                            String address = "";

                            addrCur = cr.query(
ContactsContract.CommonDataKinds.StructuredPostal.CONTENT_URI,
null,
ContactsContract.CommonDataKinds.StructuredPostal.CONTACT_ID + " = ?",
new String[]{id},
null);

                            if (addrCur != null &&
addrCur.moveToFirst()) {
                                address =
addrCur.getString(addrCur.getColumnIndex(ContactsContract.CommonDataKinds.S
tructuredPostal.DATA));
                            }

                            LocationFriendsManager.getInstance().addFriend(
                                new
LocationFriend(name, address, getApplicationContext()));
                                FriendsListFragment
friendsListFragment = fragmentAdapter
                                .getFriendsListFragment();
                                if (friendsListFragment !=
null) {
                                    LocationFriendsAdapter
adapter = (LocationFriendsAdapter) friendsListFragment
```

```
.getListAdapter();

adapter.notifyDataSetChanged();

        if (adapter != null) {

        }

    } finally {
        if (c != null) {
            c.close();
        }
        if (addrCur != null) {
            addrCur.close();
        }
    }

    break;
default:
    break;
}

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.activity_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {

    if (item.getItemId() == R.id.itemAddContact)
    {
        Intent intent = new Intent(
            Intent.ACTION_PICK,
            ContactsContract.Contacts.CONTENT_URI);
        startActivityForResult(intent, REQUEST_PICK_CONTACT);
    }

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

2.5 Térkép megjelenítése

Utolsó nagyobb feladatunk, hogy hozzunk létre egy új lapot (Fragment-et), amely az ismerősöket jeleníti meg térképen.

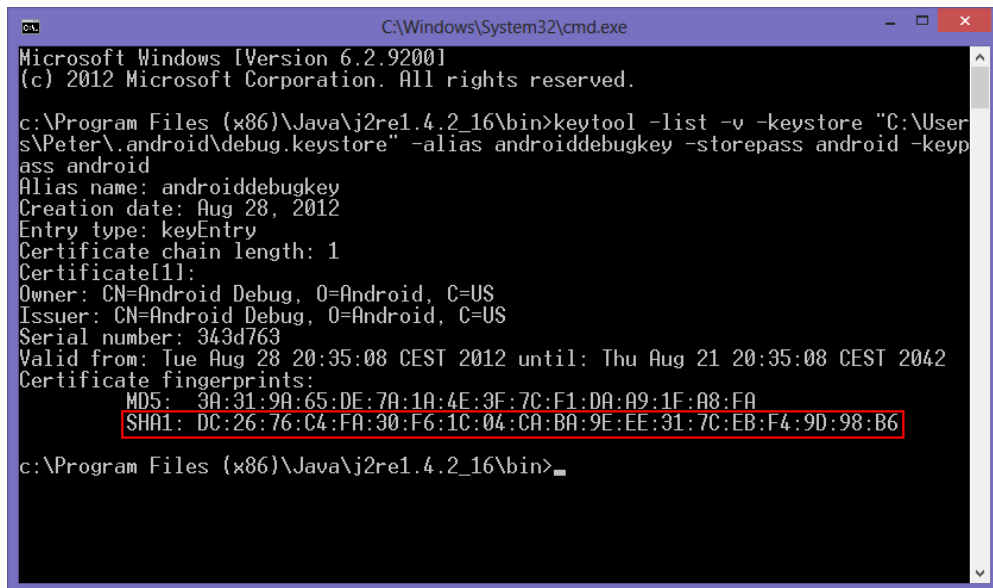
Ennek megvalósításához az Android Maps API V2-vel kell megismerkednünk, regisztrálnunk kell a projektünket a *Google APIs Console*-on és igényelnünk kell egy Maps API kulcsot. Részletesen:

<https://developers.google.com/maps/documentation/android/>

2.5.1 Kulcs igénylése

Elsőként olvassuk ki a debug kulcsból az SHA1 lenyomatot konzolból a következő paranccsal (a keytool a JavaSDK/JRE-ben található):

```
keytool -list -v -keystore "C:\Users\your_user_name\.android\debug.keystore" -alias androiddebugkey -storepass android -keypass android
```



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

c:\Program Files (x86)\Java\j2re1.4.2_16\bin>keytool -list -v -keystore "C:\User
s\Peter\.android\debug.keystore" -alias androiddebugkey -storepass android -keyp
ass android
Alias name: androiddebugkey
Creation date: Aug 28, 2012
Entry type: keyEntry
Certificate chain length: 1
Certificate11:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 343d763
Valid from: Tue Aug 28 20:35:08 CEST 2012 until: Thu Aug 21 20:35:08 CEST 2042
Certificate fingerprints:
MD5: 3A:31:9A:65:DE:7A:1A:4E:3F:7C:F1:DA:A9:1F:A8:FA
SHA1: DC:26:76:C4:FA:30:F6:1C:04:CA:BA:9E:EE:31:7C:EB:F4:9D:98:B6

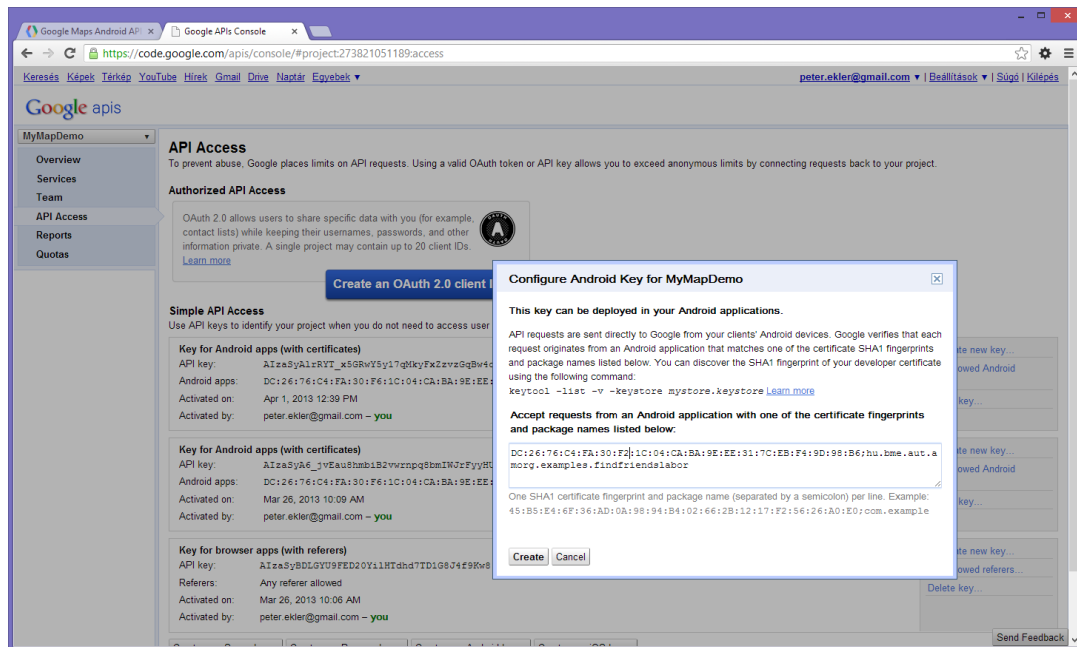
c:\Program Files (x86)\Java\j2re1.4.2_16\bin>
```

Ezt követően regisztráljuk a projektünket a Google APIs Console-on:

<https://code.google.com/apis/console/>

A Console-on bal oldalt a Services menüpontot választva kapcsoljuk be a „Google Maps Android API v2” szolgáltatást.

Ezt követően a Console-on szintén bal oldalt az API Access menüt választva, az új oldalon alul válasszuk a „Create new Android key..” menüt és adjuk meg az SHA1 kulcsot, majd utána „,” jellel az Android alkalmazásunk fő package azonosítóját ([SHA1];package).



Sikeres generálás után másoljuk ki az API key-t, mivel szükségünk lesz rá a Manifest állományba.

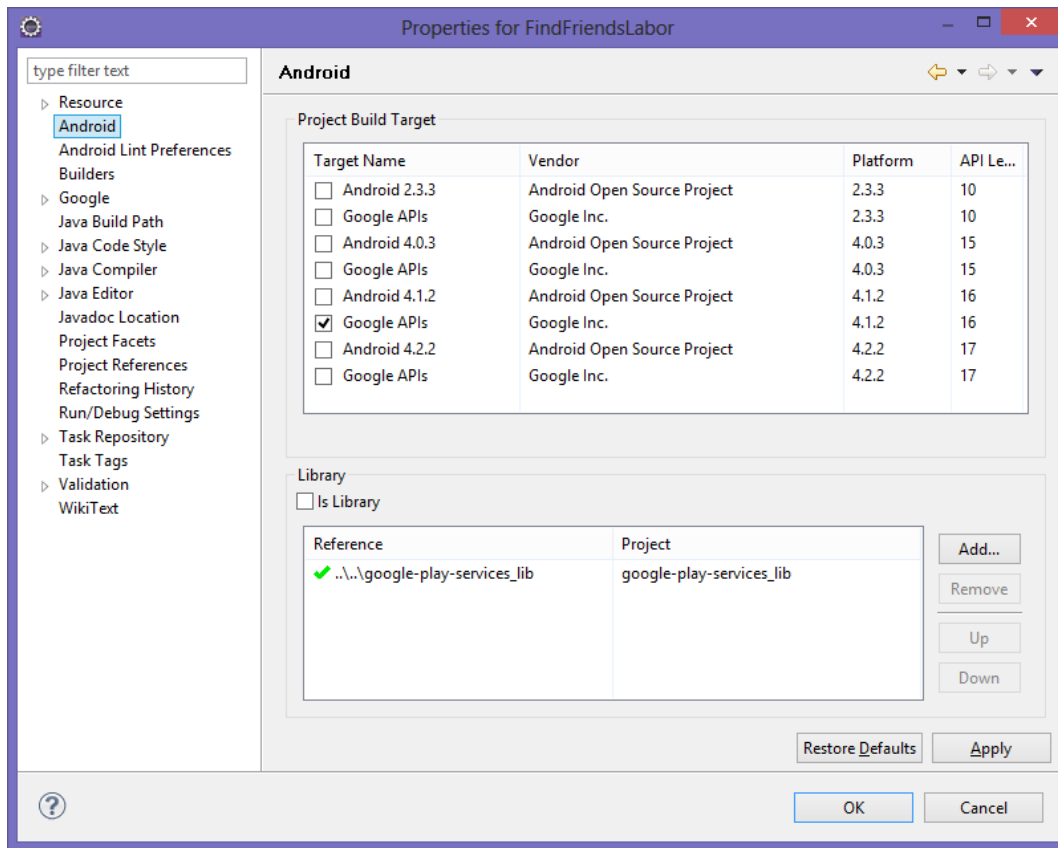
A Maps API V2 használatához a Google Services library betöltése szükséges, ez azonban nem egy egyszerű .jar betöltését jelenti, hanem az SDK-ban található google-play-services_lib projektet kell importálnunk, mivel a projekt több erőforrást is tartalmaz, amit nem lehetne .jar-ba csomagolni.

Eclipse-be importáljuk be ezt a library projektet: Jobbegér->import->”Existing Android Code into Workspace”.

A library projektet a labor anyagában is elhelyeztük, de az SDK-ban elvileg az alábbi helyen érhető el:

[Android-sdk]\extras\google\google_play_services\libproject\google-play-services_lib

Ezt követően a saját projektünk beállításában (jobbeger->properties) vegyük fel a hivatkozást a library projektre:



2.5.2 Manifest paraméterek

A Manifest állományba vegyük fel a következő engedélyeket:

```
<uses-feature
    android:glEsVersion="0x00020000"
    android:required="true" />

<permission
    android:name="hu.bme.aut.amorg.examples.findfriendslabor.permission.MAPS_RECEIVE"
    android:protectionLevel="signature" />

<uses-permission
    android:name="hu.bme.aut.amorg.examples.findfriendslabor.permission.MAPS_RECEIVE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission
    android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
```


Fontos, hogy az engedélynél a saját alkalmazásunk csomag azonosítóját írjuk be a megjelölt helyen!

Illetve a Manifest állományban még az `<application>` tag-en belül be kell állítani az API kulcsot, amit korábban generáltunk:

```
<application
    ... >
    <meta-data
        android:name="com.google.android.maps.v2.API_KEY"
        android:value="AIzaSyAlrRYT_x5GRwY5y17qMkyFxZzvzGqBw4c" />
```

2.5.3 Térkép megjelenítése barátokkal

Végül a térkép megjelenítéséhez készítsünk egy új *Fragment*-et, *FriendsMapFragment* néven:

```
public class FriendsMapFragment extends SupportMapFragment {

    public static final String TAG = "FriendsMapFragment";
    public static final String TITLE = "Térkép";

    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
    }

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

        getMap().setMyLocationEnabled(true);
        getMap().moveCamera(CameraUpdateFactory.newLatLngZoom(new
        LatLng(47, 19), 5));
        getMap().getUiSettings().setAllGesturesEnabled(false);

        for (int i=0; i<LocationFriendsManager.getInstance().getLocationFriendsNum(); i++) {
            LocationFriend tmpFriend =
            LocationFriendsManager.getInstance().getLocationFriend(i);
            if (tmpFriend.hasCoords()) {
                getMap().addMarker (
                    new MarkerOptions().
                    position(new
                    LatLng(tmpFriend.getLat(), tmpFriend.getLng())).
                    title(tmpFriend.getName()));
            }
        }
    }
}
```

Illetve módosítsuk a *FriendsFragmentPagerAdapter*-t, hogy a térképet is megjelenítse egy harmadik lapon:

```
public class FriendsFragmentPagerAdapter extends FragmentPagerAdapter {

    private LocationDashboardFragment locationDashboardFragment;
    private FriendsListFragment friendsListFragment;
    private FriendsMapFragment friendsMapFragment;

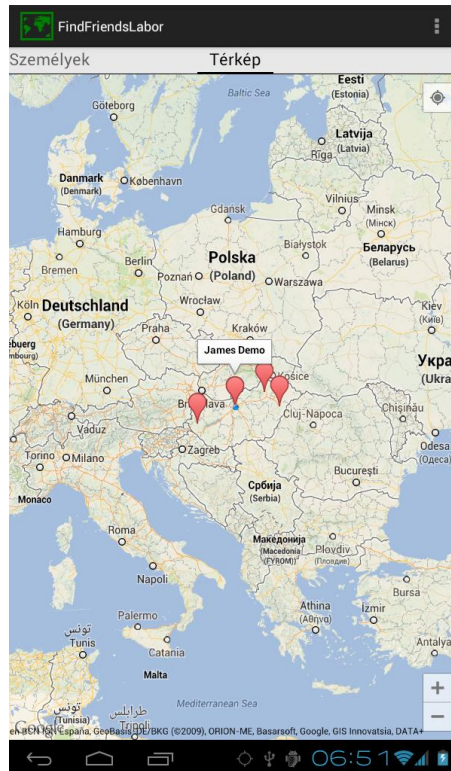
    public FriendsFragmentPagerAdapter(FragmentManager fragmentManager) {
        super(fragmentManager);
        locationDashboardFragment = new LocationDashboardFragment();
        friendsListFragment = new FriendsListFragment();
        friendsMapFragment = new FriendsMapFragment();
    }

    @Override
    public Fragment getItem(int pos) {
        switch (pos) {
            case 0:
                return locationDashboardFragment;
            case 1:
                return friendsListFragment;
            case 2:
                return friendsMapFragment;
            default:
                return null;
        }
    }

    @Override
    public CharSequence getPageTitle(int pos) {
        switch (pos) {
            case 0:
                return LocationDashboardFragment.TITLE;
            case 1:
                return FriendsListFragment.TITLE;
            case 2:
                return FriendsMapFragment.TITLE;
            default:
                return null;
        }
    }

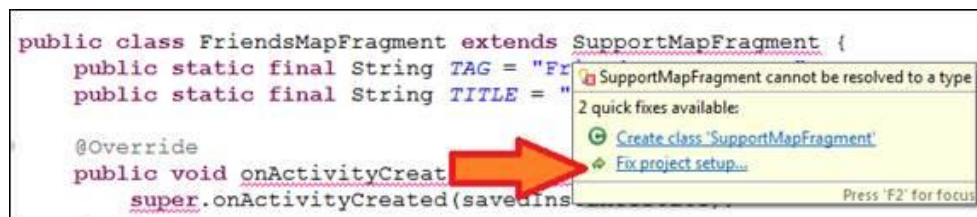
    @Override
    public int getCount() {
        return 3;
    }

    public FriendsListFragment getFriendsListFragment() {
        return friendsListFragment;
    }
}
```

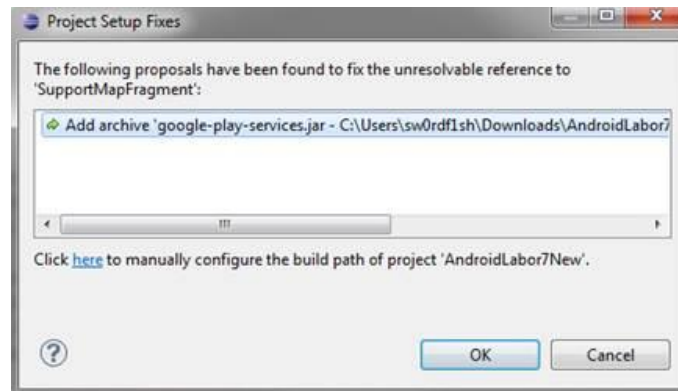


Amennyiben a labor gépeken a Google Play Services lib referencia nem működik, ennek valószínűleg az az oka, mivel az Eclipse abszolút elérési úttal importálja a libraryt. Ebben az esetben amikor elsőként hozzáadjuk a library projektet, még jónak tűnik a hozzáadás, de az ablak bezárása majd újrainvitása után látszik hogy nincs rendben a hozzáadás, ezt egy, piros X is jelzi nem is használhatók az osztályok amiket tartalmaz a csomag (pl. SupportMapFragment).

Ennek megoldására, hogy a megfelelő térképkezelő kódrésznel, amit pirossal jelöl az Eclipse, a hibás elemen „Fix project setup” menüpontot kell választani.



A felugró dialogust pedig Ok-val végül be kell zárnunk.



2.6 Barát törlésének megvalósítása

Tegye lehetővé, hogy a barátok listából lehessen törölni. Például a listaelemen hosszú lenyomás hatására jelenjen meg egy menü (korábbi laborban tanult módon), ahol el lehessen érni a törlés funkciót.

2.7 További adat megjelenítése a pozíció információk nézetén

Valósítsa meg, hogy az első oldalon további információk is megjelenjenek a pozícióval kapcsolatban, például pontosság és irány (*accuracy*, *bearing*).