ANDROID VI.

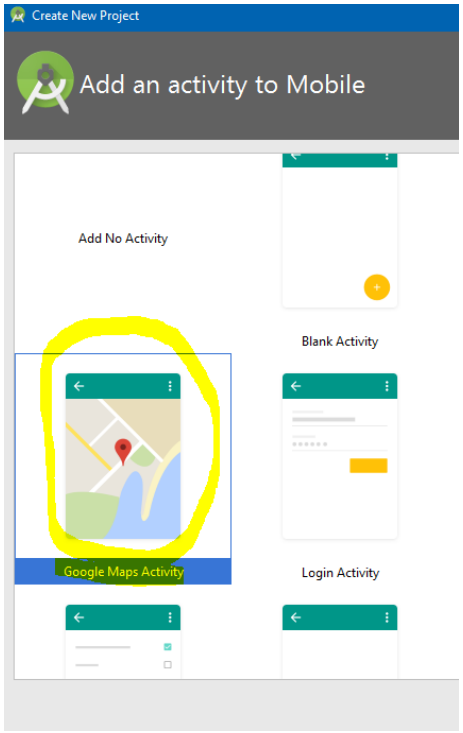
7. týden, KIV/MKZ 2016

L. Pešička

OBSAH

- Mapy
- Widgety

Napřed stručné HOW-TO na zprovoznění mapy
Potom další podrobnosti



MAPY NA ANDROIDU – I.

- Google Map Android API
- Mapu lze vložit do aktivity jako fragment
- activity_maps.xml:

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.pesic.mkz_mapypokus1.MapsActivity" />
```

MAPY NA ANDROIDU - II.

- Použití v aktivitě:

```
private GoogleMap mMap;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);
    // Obtain the SupportMapFragment and get notified when the map is ready to be used.
    SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map);
    mapFragment.getMapAsync(this);
}


public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;

    // Add a marker in Sydney and move the camera
    LatLng sydney = new LatLng(-34, 151);
    mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));
    mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
}
```

MAPY NA ANDROIDU - III.

- Práva v manifestu
- Google Play Services
- Emulátor s Google API

- Je potřeba získat map key (zdarma)
  a uvést jej např. do google_maps_api.xml

```
    Once you have your key (it starts with "AIza"), replace the "google_maps_key"
    string in this file.
    -->
    <string name="google_maps_key" templateMergeStrategy="preserve"
        translatable="false"> AIza[redacted]</string>
</resources>
```

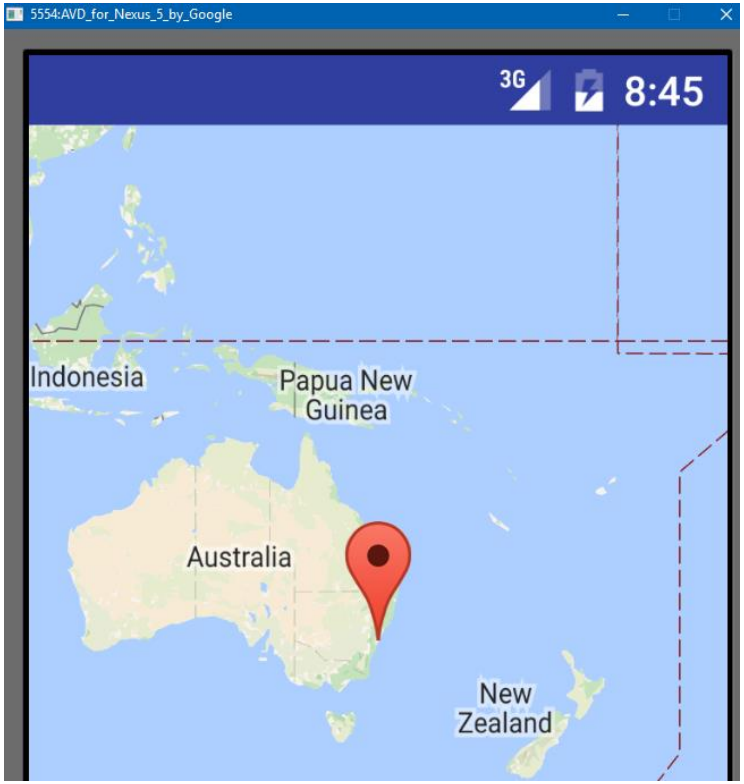
MAPY NA ANDROIDU – IV.

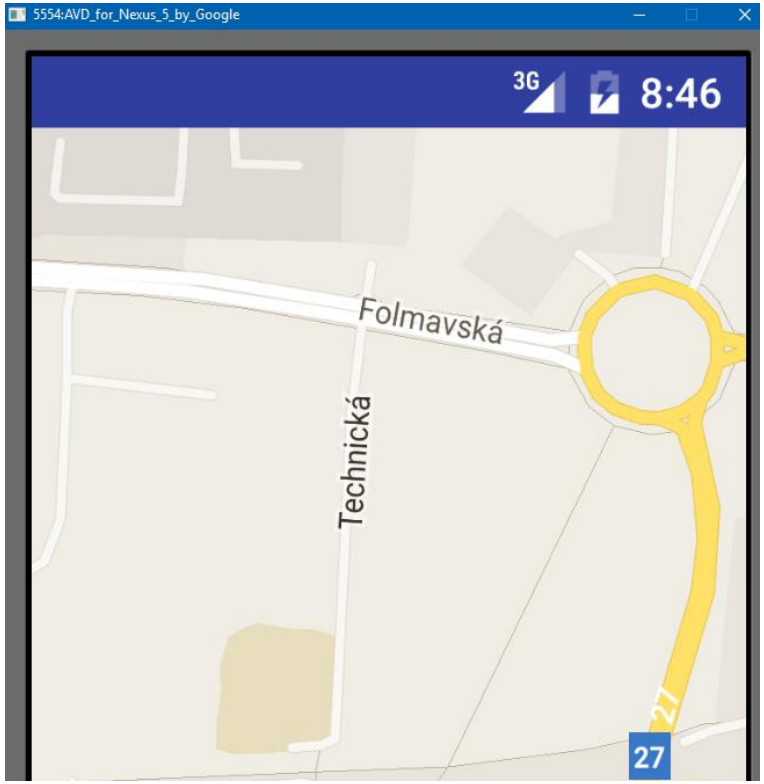
- Lze použít SupportMapFragment
  nebo MapFragment od API12 (3.1)

```
<fragment    class="com.google.android.gms.maps.MapFragment"
   android:layout_width="match_parent"
   android:layout_height="match_parent"/>
```

- Mapu získáme voláním
  getMapAsync(OnMapReadyCallback)
  zavolá callback když je mapa připravena

MAPY NA ANDROIDU - VÝSLEDEK





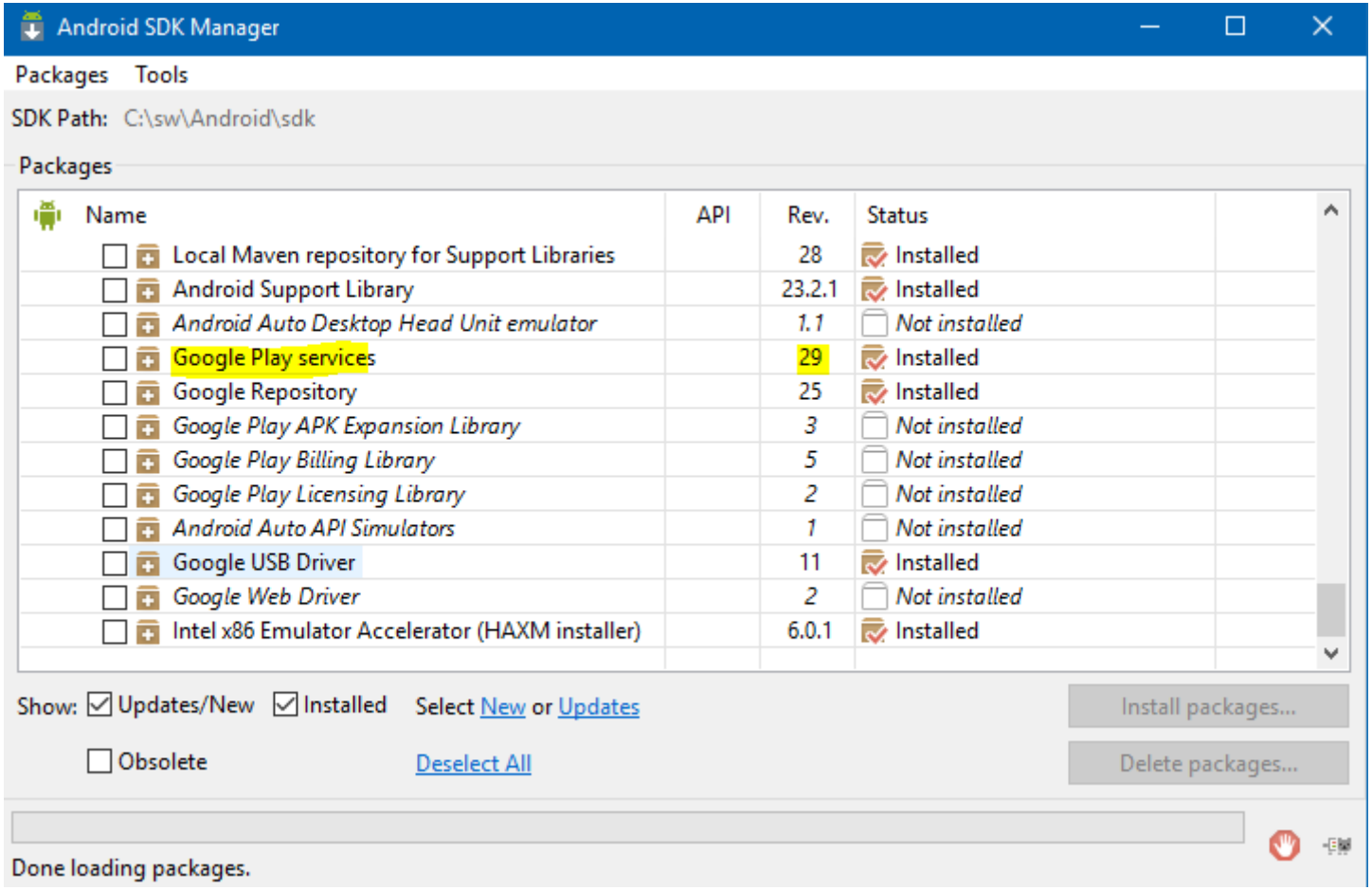
MAPY

- Google Maps API v2 - Android
  - Nativní implementace pro Android
  - Návod: https://developers.google.com/maps/documentation/android/
- Google Maps API v3
  - JavaScript, použití ve WebView, prohlížeči
  - Návod: https://developers.google.com/maps/documentation/javascript/tutorial

Zprovoznění mapy (API v2)

- Google Play Services
- Emulátor s Google API (ne Android API)
- Vydaný API key
  - Pro náš certifikát: debug x relase
  - Potřebujeme SHA-1 fingerprint certifikátu
  - Přes google consoli - získání API klíče
  - Přidání klíče do aplikace (manifest, …)
- Práva v manifestu
- Mapový fragment v aplikaci

GOOGLE PLAY SERVICES



ANDROID STUDIO

- build.gradle module

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

    dependencies {
        compile 'com.google.android.gms:play-services:8.4.0'
    }
```

- Počet metod v aplikaci musí být pod 65536 je-li potřeba šetřit, jen vybrané:

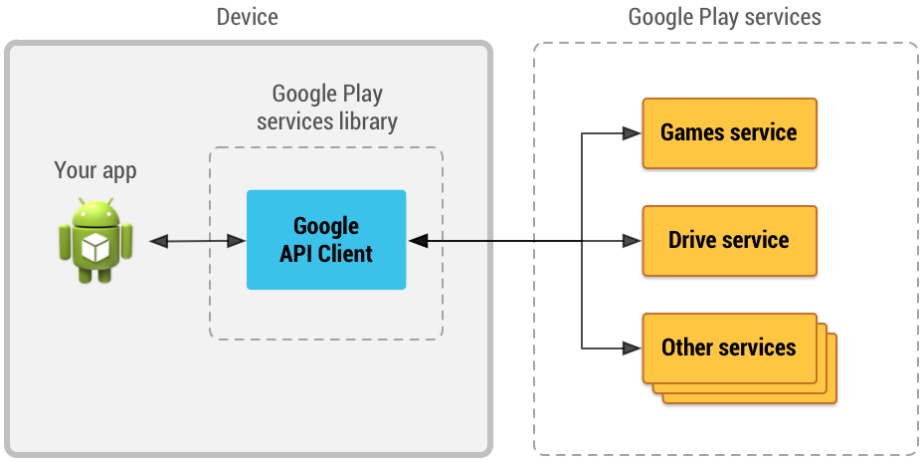
Google Maps: com.google.android.gms:play-services-maps:8.4.0

GOOGLE PLAY SERVICES

- Běží jako služba v Android OS
- Interakce přes klientskou knihovnu
- Aktualizováno přes Google Play store
- Od Android 2.3 (Gingerbread)
- Můžeme používat nejnovější API
- Nezávislé na výrobci

Telefon:

Nastavení
Aplikace
Služby
Google Play
(verze)



GOOGLE PLAY SERVICES

Vybrané služby:

- Places
  - location-aware aplikace
- Location settings
  - Více senzorů pro určení polohy
  - GPS, wifi, airplain mode
  - Dnes místo android.location
  - Dialog pro povolení senzorů jedním dotykem
- Fit
  - Senzory, recording, history, …
  - Výpočet kalorií, vzdáleností

GOOGLE PLAY SERVICES - SLUŽBY

- Mobile Ads
  - Reklamy ☺
- Play Game Services
  - Nearby connections – najít další zařízení na lokální síti a vyměňovat si zprávy (hrát hry)
  - Použití telefonu jako ovladače hry
- Google API client
- Drive
- SafetyNET API
  - Zda běží na zařízení, které prošlo Android Compatibility testem

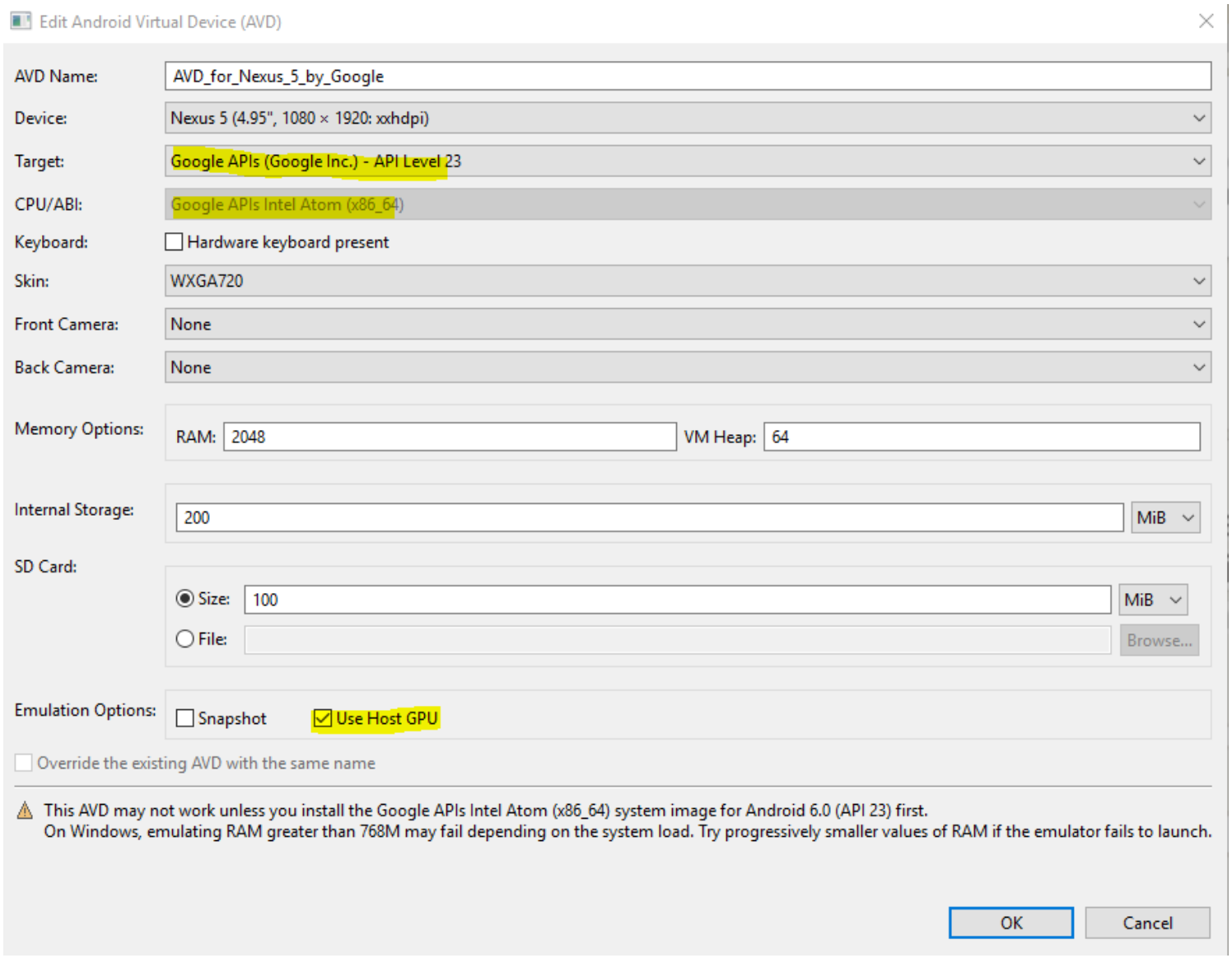
PŘEHLED VŠECH API

- https://developers.google.com/android/guides/setup#add_google_play_services_to_your_project

Google+
Google Account Login
Google Actions, Base Client Library
Google Address API
Google App Indexing
Google App Invites
Google Analytics
Google Cast
Google Cloud Messaging
Google Drive
Google Fit
Google Location, Activity Recognition, and Places

Google Maps
Google Mobile Ads
Mobile Vision
Google Nearby
Google Panorama Viewer
Google Play Game services
SafetyNet
Google Wallet
Android Wear

EMULÁTOR S GOOGLE API



MAPY
ZÍSKÁNÍ API KLÍČE - FINGERPRINT

FINGERPRINT SHA-1

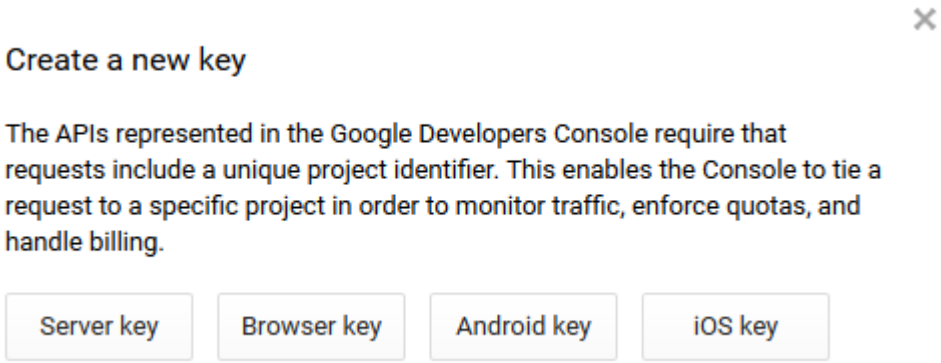
```
C:\Program
Files\Java\jdk1.8.0_51\bin>keytool
-list -v -alias androiddebugkey
-keystore
c:\Users\pesic\.android\debug.keystore
-storepass android -keypass android
```

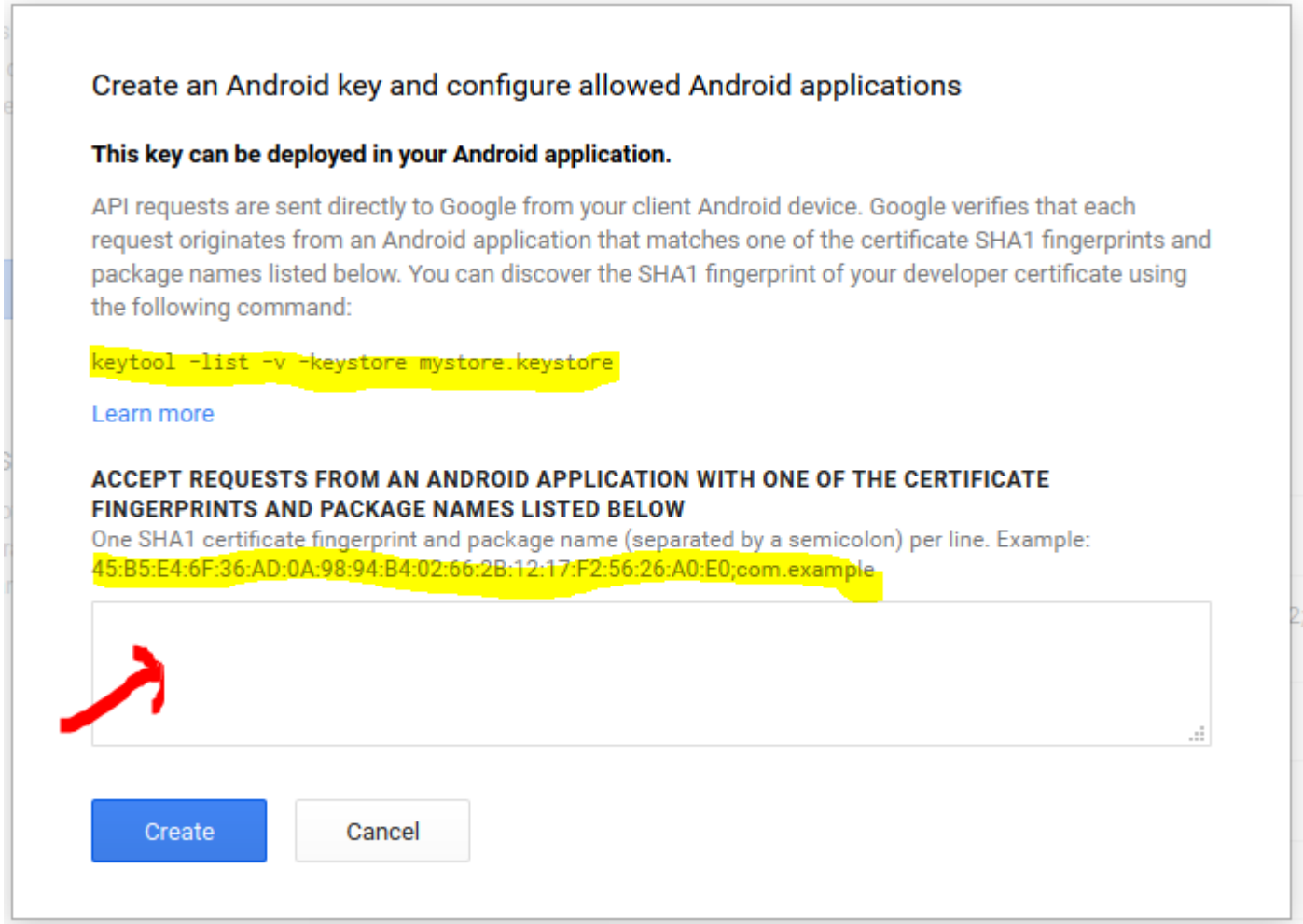
PŘÍKLAD VÝSTUPU

Alias name: androiddebugkey
Creation date: 2.3.2012
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: XXXXXXX
Valid from: Fri Mar 02 10:58:44 CET 2012 until: Sun Feb 23 10:58:44 CET 2042
Certificate fingerprints:
MD5: XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX
SHA1: XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX
Signature algorithm name: SHA1withRSA
Version: 3

MAPY - ZÍSKÁNÍ API KEY

- https://console.developers.google.com
- API Auth – Credentials




Create an Android key and configure allowed Android applications
This key can be deployed in your Android application.
API requests are sent directly to Google from your client Android device. Google verifies that each request originates from an Android application that matches one of the certificate SHA1 fingerprints and package names listed below. You can discover the SHA1 fingerprint of your developer certificate using the following command:
keytool -list -v -keystore mystore.keystore
Learn more
ACCEPT REQUESTS FROM AN ANDROID APPLICATION WITH ONE OF THE CERTIFICATE FINGERPRINTS AND PACKAGE NAMES LISTED BELOW
One SHA1 certificate fingerprint and package name (separated by a semicolon) per line. Example:
45:B5:E4:6F:36:AD:0A:98:94:B4:02:66:2B:12:17:F2:56:26:A0:E0;com.example
Create
Cancel

Key for Android applications

API KEY	AIzaS[redacted]
ANDROID APPLICATIONS	C0:82:FE:89:D3:3C:28:FB:84:F5:CB:0E:54:88:01:4B:F4:DD:94:12;com.example.pesicka.myapplication
ACTIVATION DATE	Mar 25, 2015, 9:12:00 AM
ACTIVATED BY	pesicka@gmail.com (you)
OBSOLETE KEY	AIzaS[redacted]
STATUS	Active until Mar 26, 2015, 9:12:00 AM

Edit allowed Android applications | Regenerate key | Revert to obsolete key | Delete

ANDROID STUDIO

- Nový projekt - Google Maps Activity
- Práva v manifestu:

```
<uses-permission android:name="android.permission.INTERNET" />
<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" />
<!--
The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
     Google Maps Android API v2, but are recommended.
-->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

ODKAZ NA API KEY V MANIFESTU

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <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="@string/google_maps_key" />

    <activity
        android:name=".MapsActivity"
        android:label="@string/title_activity_maps" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

Může být i v google_maps_api.xml

XML LAYOUT

```
<fragment
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"
android:id="@+id/map"
tools:context=".MapsActivity"

android:name="com.google.android.gms.maps.Sup
portMapFragment" />
```

MARKER

Přidání markeru, přesun kamery,…

Klikatelné markery

```
public void onMapReady(GoogleMap map) {
    map.addMarker(new MarkerOptions()
        .position(new LatLng(0, 0))
        .title("Marker"));
}
```

DRAGGABLE MARKER

- Je možné jej udělat přesunutelným
- Není defaultně

```
static final LatLng PERTH = new LatLng(-31.90, 115.86);
Marker perth = mMap.addMarker(new MarkerOptions()
                          .position(PERTH)
                          .draggable(true));
```

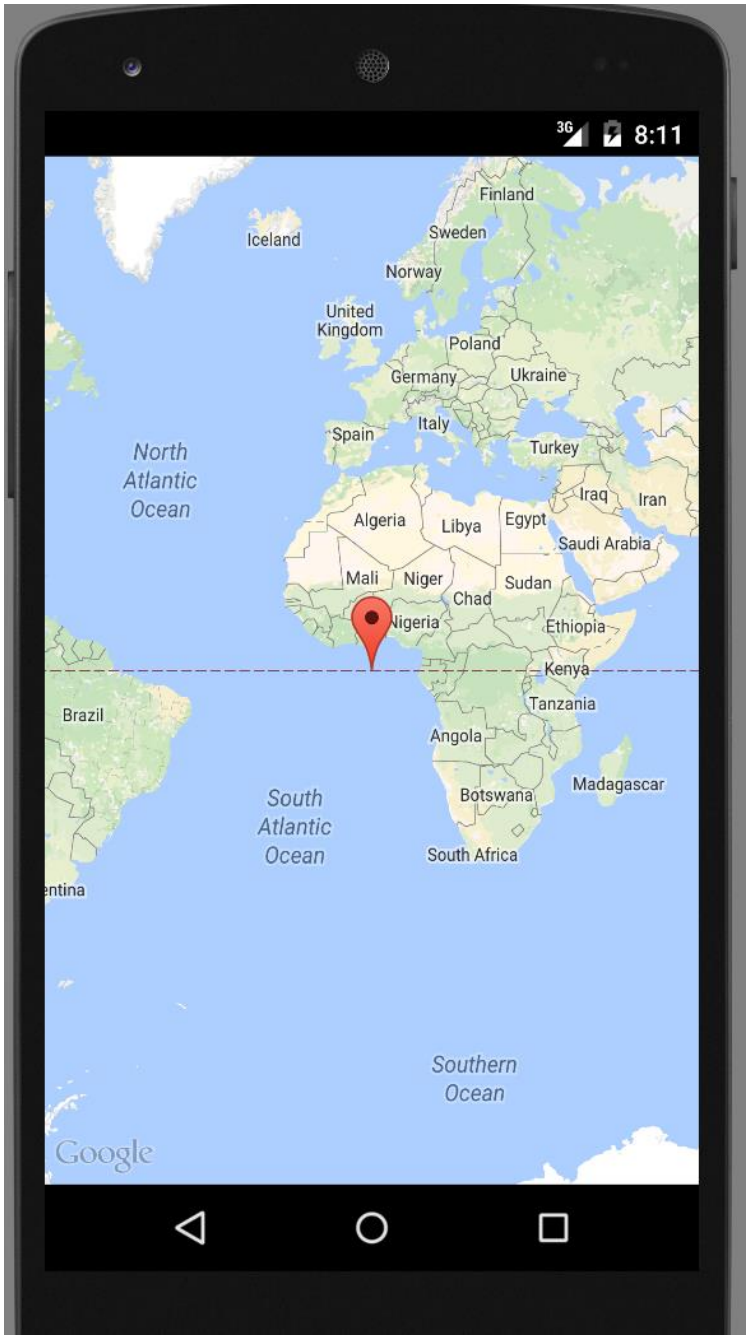
Vlastnosti markeru

- Pozice
  - Vyžadovaná
- Anchor
  - Defaultně prostředek spodní části obrázku ukazuju na souřadnici na mapě
- Alpha
  - Průhlednost
- Title
- Snippet
  - Text pod titulkem

VLASTNOSTI MARKERU

- Icon
  - Obrázek místo defaultního
- Draggable
  - Možnost přesunout marker
- Visible
- Flat or Billboard orientation
- Rotation

Celý přehled:
https://developers.google.com/maps/documentation/android-api/marker#add_a_marker



VÝSLEDEK BĚHU V EMULÁTORU

POZICE MĚSTA, ZNAČKA

- static final LatLng *HAMBURG* =
  new LatLng(53.558, 9.927);
- kontrola: do mapy.cz lze zadat přímo do vyhledávacího okna 53.558, 9.927

- Marker hamburg = map.addMarker(
  new MarkerOptions()
  .position(HAMBURG).title("Hamburg"));
- přidá na mapu špendlík s nápisem Hamburg

INTERAKTIVITA MAPY

- lze zaregistrovat posluchače, který bude reagovat při kliknutí na značku

posluchač:
setOnMarkerClickListener(OnMarkerClickListener)

metoda:
onMarkerClicked(Marker)

INTERAKTIVITA

```
public class MarkerDemoActivity extends
  android.support.v4.app.FragmentActivity
    implements OnMarkerClickListener
{
    private Marker myMarker;
...

public boolean onMarkerClick(final Marker marker) {

        if (marker.equals(myMarker))
        {
            //handle click here
        }
    }
}
```

ZMĚNA POHLEDU

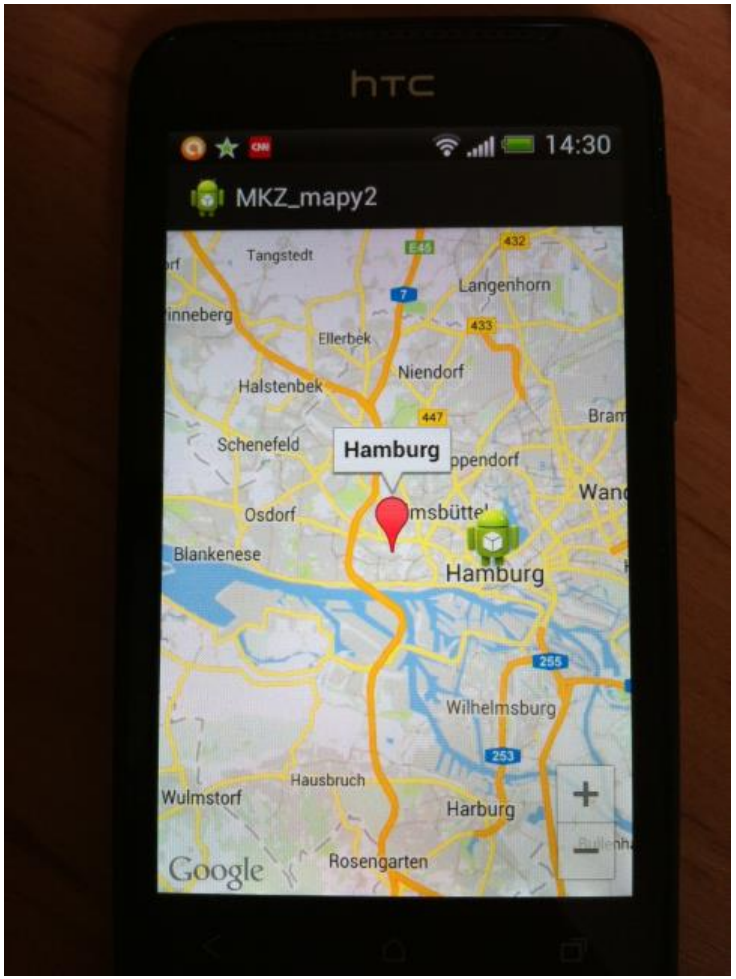
```
// Move the camera instantly to Hamburg with a zoom of 15.

map.moveCamera(CameraUpdateFactory.newLatLngZoom(HAMBURG, 15));

// Zoom in, animating the camera.

map.animateCamera(CameraUpdateFactory.zoomTo(10), 2000, null);
```

UKÁZKA VÝSLEDNÉHO ZOBRAZENÍ



ukázka běhu aplikace na reálném zařízení

Hamburg
- defaultní marker
- popisek

Druhý marker ve formě ikony robota, po kliknutí zobrazí titulek a podrobný text

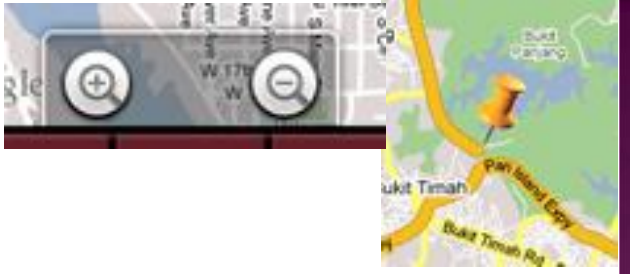
PRÁCE S MARKERY

tutoriál

markery více událostí

zobrazení toastu s podrobnou informací

http://bon-app-etit.blogspot.be/2012/12/add-informationobject-to-marker-in.html



DALŠÍ MOŽNOSTI

- změna pohledu na mapu
  - MAP_TYPE_TERRAIN, HYBRID, NORMAL
  - map.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
- Indoor mapy
  - Přesunout kameru na místo objektu s indoor mapou
  - map.moveCamera(CameraUpdateFactory.newLatLng Zoom(new LatLng(-33.86997, 151.2089), 18));
- Markery
- Flat markery
  - Šipka se točí podél cesty…
- Polylines
  - Cesty a routy na mapě

Interaktivní přehled

- https://developers.google.com/maps/documentation/android/

LOKAČNÍ SLUŽBY

1. API v Androidu: android.location

2. GPlayServices: Google Location Services
   - V současné době spíše doporučené
   - Vybere vhodného poskytovatele
   - Automaticky řeší řadu věcí (spotřeba)

Otázky lokačních služeb

- Více zdrojů polohy
  - GPS, Cell-ID,Wi-Fi
  - Přesnost, rychlost, battery-efficiency
- Pohyb uživatele
  - Uživatel se může pohybovat
  - Je třeba se ptát na polohu opakovaně
- Proměnlivá přesnost
  - Poloha před 10s z jednoho zdroje může být přesnější než novější údaj z jiného zdroje

LOKAČNÍ SLUŽBY

- systémová služba LocationManager
  - API pro určení polohy
- vyžádání instance od systému getSystemService(Context.LOCATION_SERVICE)
  - vrací handler nové instance
- dotazovat se LocationProviders na polohu
- registrovat se pro periodické updaty polohy z *location providera* (dle kritéria nebo jména)
- registrovat se pro spuštění Intentu, když bude zařízení **blízko** určité lokace (v metrech)

MOŽNOSTI

- získání pozice v bodě dotyku mapy
  - latitude, longtitude
  - onTouchEvent(…) {
  - GeoPoint p =
    mapView.getProjection().fromPixels(
    (int) event.getX(), (int) event.getY()); ..
- geocoding, reverzní geocoding
  - známe lat,long => určit adresu (geocoding)
  - známe adresu => určit lat, long (reverse geoc.)

GEOCODER

```
Geocoder myLocation = new Geocoder(getApplicationContext(),
Locale.getDefault());

List<Address> myList = myLocation.getFromLocation(latPoint,
lngPoint, 1);
```

GeoCoder používá Internet pro vyhledávání:

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

INFORMACE O ADRESE

```
double latitude = location.getLatitude();
double longitude = location.getLongitude();
Geocoder gc = new Geocoder(this, Locale.getDefault());
try {
List<Address> addresses = gc.getFromLocation(lat, lng, 1);
StringBuilder sb = new StringBuilder();
if (addresses.size() > 0) {
Address address = addresses.get(0);
for (int i = 0; i < address.getMaxAddressLineIndex(); i++)
sb.append(address.getAddressLine(i)).append("\n");
sb.append(address.getLocality()).append("\n");
sb.append(address.getPostalCode()).append("\n");
sb.append(address.getCountryName());
}
```

UKÁZKA - POZICE EMPIRE STATE BUILDING

```
Geocoder geoCoder = new Geocoder(this, Locale.getDefault());
        try {
            List<Address> addresses =
geoCoder.getFromLocationName( "empire state building", 5);
            String add = "";
            if (addresses.size() > 0) {
                p = new GeoPoint(
                        (int) (addresses.get(0).getLatitude() * 1E6),
                        (int) (addresses.get(0).getLongitude() * 1E6));
                mc.animateTo(p);
                mapView.invalidate();
            }
        } catch (IOException e) {  e.printStackTrace(); }
```

max. počet výsledků

Získání polohy

- GPS
  - nejpřesnější
  - funguje outdoor
  - velká spotřeba energie (baterie)
  - pomalé
- Android Network Location Provider
  - využití cell tower (BTSky)
  - WiFi s lokační informací
  - rychlejší, menší spotřeba

Možné zdroje chyb

- různé zdroje lokace
  - GPS, cell-id, WiFi
  - různá přesnost, rychlost, energetická náročnost
- pohyb uživatele
  - jak často znovu vyžádat polohu
- proměnlivá přesnost
  - lokace před 10s z jednoho zdroje může být přesnější než nejnovější lokace z jiného (i dokonce stejného) zdroje

VYŽÁDÁNÍ UPDATU POLOHY

LocationListener implementuje callbacky když se změní poloha uživatele nebo stav služby

```
LocationManager lm = (LocationManager)
  this.getSystemService (Context.LOCATION_SERVICE);

LocationListener locLis = new LocationListener() {
   public void onLocationChanged(Location loc) { … }
   public void onStatusChanged(String provider, int status,
       Bundle extra ) {}
   public void onProviderEnabled(String provider) {}
   public void onProviderDisabled(String provider) {}
}
lm.requestLocationUpdates
    (LocationManager.NETWORK_PROVIDER, 0, 0, locLis);
```

posluchač

jaký typ poskytovatele, jak často, minimální vzdálenost

REQUESTLOCATIONUPDATES()

- typ poskytovatele
  - NETWORK_PROVIDER = cell-id, wifi
  - GPS_PROVIDER
- frekvence updatů
  - počet milisekund (jen poradní, kvůli baterii)
  - =0 .. co nejčastěji
- minimální změna vzdálenosti
  - v metrech
  - =0 .. co nejčastěji
- náš posluchač
  - přijímá callbacky

VYŽÁDÁNÍ PRÁV V MANIFESTU

- jen NETWORK_PROVIDER:
  - ACCESS_COARSE_LOCATION
- GPS_PROVIDER i NETWORK_PROVIDER:
  - ACCESS_FINE_LOCATION

```
<manifest ... >
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    ...
</manifest>
```

JE DOSTUPNÁ GPS?

```
manager = (LocationManager)
   getSystemService(Context.LOCATION_SERVICE );

boolean statusOfGPS =
   manager.isProviderEnabled
            (LocationManager.GPS_PROVIDER);
```

VÝBĚR POSKYTOVATELE

- můžeme nastavit vlastnosti poskytovatele, a systém vybere, který nejvíce vyhovuje daným vlastnostem
- třída Criteria
  - lze nastavit požadované vlastnosti
- getBestProvider()
  - vybere poskytovatele, který nejvíce vyhovuje daným kritériím

- aplikace by se měla vyrovnat i se situací, když není žádný provider k dispozici (null)

KRITÉRIA

funkce	popis
setAccuracy()	Požadovaná přesnost (ACCURACY_FINE, ACCURACY_COARSE)
setAltitudeRequired()	Je třeba nadmořská výška?
setBearingRequired()	Informace o orientaci (natočení)
setCostAllowed()	Za získání polohy se musí platit…
setPowerRequirement()	Je pro nás důležitá spotřeba CRITERIA.POWER_LOW CRITERIA.POWER_MEDIUM CRITERIA.POWER_HIGH
setSpeedRequired()	Potřebujeme získat info o rychlosti

POUŽITÍ

```
locationManager = (LocationManager) this.getSystemService(LOCATION_SERVICE);
    geocoder = new Geocoder(this);

    Criteria criteria = new Criteria();
    criteria.setAccuracy(Criteria.ACCURACY_COARSE);
    List<String> lProviders = locationManager.getProviders(false);
    for(int i=0;i<lProviders.size();i++){
        Log.d("LocationActivity", lProviders.get(i));
    }
    String provider = locationManager.getBestProvider(criteria, true); // null

    long minTime = 60000;
    float minDistance = 5;

    locationManager.requestLocationUpdates(provider, minTime, minDistance, this);
```

zdroj: http://stackoverflow.com/questions/10687409/locationmanagers-getbestprovider-returning-null

JAK DLOUHO POSLOUCHAT UPDATY?

- jakmile se aplikace spustí
- až po aktivaci určité funkce

- dlouhá doba přijímání updatů
  - vyčerpává baterii
- krátká
  - nemusíme dosáhnout požadované přesnosti

začínáme poslouchat zavoláním **requestLocationUpdates**(…)

POSLEDNÍ ZNÁMÁ LOKACE

- příjem prvního lokačního fixu trvá často příliš dlouho (z hlediska uživatele)
- využití nakešované polohy

```
Location lastKnownLocation =
    getLastKnownLocation(locationProvider);
```

Ukončení poslechu updatů

odstraní posluchače:

locationManager.removeUpdates (locationListener)

UKÁZKA - STACK OVERFLOW

- získání nejlepšího provideru:

```
lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);

locationListener = new MyLocationListener();
  Criteria criteria = new Criteria();
  criteria.setAccuracy(Criteria.ACCURACY_FINE);
  String provider = lm.getBestProvider(criteria, true);
  Location mostRecentLocation = lm.getLastKnownLocation(provider);
  if(mostRecentLocation!=null){
  latid=mostRecentLocation.getLatitude();
  longid=mostRecentLocation.getLongitude();
  }
  lm.requestLocationUpdates(provider, 1, 0, locationListener);
```

UKÁZKA - STACK OVERFLOW

- posluchač:

```
private class MyLocationListener implements LocationListener {

@Override
 public void onLocationChanged(Location loc) {
   if (loc != null) {
    latid = loc.getLatitude();
    longid = loc.getLongitude();
    // if(loc.hasAccuracy()==true){
    accuracyd = loc.getAccuracy();
    String providershown = loc.getProvider();
    accuracy.setText("Location Acquired. Accuracy:"
      + Double.toString(accuracyd) + "m\nProvider: "+providershown);
    accuracy.setBackgroundColor(Color.GREEN);
    // }

    userinfo=usernamevalue+"&"+Double.toString(latid)+"&"+Double.toString(longid);
    submituserlocation(userinfo);
   }

  }
```

POSTUP ZÍSKÁNÍ POLOHY

1. spuštění aplikace
2. o něco později, začít poslouchat updaty od vybraného poskytovatele
3. udržovat „aktuální nejlepší odhad“ polohy
4. přestat poslouchat updaty
5. využít nejlepší odhad

AKTUÁLNÍ NEJLEPŠÍ ODHAD

- viz příklad v SDK
  http://developer.android.com/guide/topics/location/obtaining-user-location.html

- zkontrolovat, zda update je výrazně novější
  - použít je, uživatel se mohl přemístit
- zkontrolovat zda je přesnost udaná poskytovatelem lepší nebo horší než u předchozího odhadu
  - může být lepší o něco starší update s menší chybou
- od jakého poskytovatele je update
  - kterému důvěřujeme více

location.getAccuracy() – poskytovatel nám spolu s polohou vrátí odhad chyby, který se může měnit

TIPY NA ZLEPŠENÍ VÝDRŽE

- zmenšit velikost okna, kdy se poslouchají updaty
- dostávat updaty méně často
- omezit množinu poskytovatelů

S využitím Google Play Services

GOOG PLAY SERVICES - POSLEDNÍ POLOHA

- Fused location provider
- Nastavit google play services (jedna řádka v build.gradle)
- Práva aplikace, např. ACCESS_FINE_LOCATION
- Připojení ke Google Play Services, v onCreate():

```
// Create an instance of GoogleAPIClient.
if (mGoogleApiClient == null) {
    mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API)
        .build();
}
```

POKRAČOVÁNÍ

```
protected void onStart() {
    mGoogleApiClient.connect();
    super.onStart();
}

protected void onStop() {
    mGoogleApiClient.disconnect();
    super.onStop();
}
```

```
public class MainActivity extends ActionBarActivity implements
        ConnectionCallbacks, OnConnectionFailedListener {
    ...
    @Override
    public void onConnected(Bundle connectionHint) {
        mLastLocation = LocationServices.FusedLocationApi.getLastLocation(
                mGoogleApiClient);
        if (mLastLocation != null) {
            mLatitudeText.setText(String.valueOf(mLastLocation.getLatitude()));
            mLongitudeText.setText(String.valueOf(mLastLocation.getLongitude()));
        }
    }
}
```

PŘÍJEM LOKAČNÍCH UPDATŮ

- Vyžádáme si lokační updaty

```
@Override
public void onConnected(Bundle connectionHint) {
    ...
    if (mRequestingLocationUpdates) {
        startLocationUpdates();
    }
}

protected void startLocationUpdates() {
    LocationServices.FusedLocationApi.requestLocationUpdates(
            mGoogleApiClient, mLocationRequest, this);
}
```

CALLBACK PRO LOKAČNÍ UPDATY

- Callback onLocationChanged()

```
public class MainActivity extends ActionBarActivity implements
        ConnectionCallbacks, OnConnectionFailedListener, LocationListener {
    ...
    @Override
    public void onLocationChanged(Location location) {
        mCurrentLocation = location;
        mLastUpdateTime = DateFormat.getTimeInstance().format(new Date());
        updateUI();
    }

    private void updateUI() {
        mLatitudeTextView.setText(String.valueOf(mCurrentLocation.getLatitude()));
        mLongitudeTextView.setText(String.valueOf(mCurrentLocation.getLongitude()));
        mLastUpdateTimeTextView.setText(mLastUpdateTime);
    }
}
```

ZASTAVENÍ A ZNOVUSPUŠTĚNÍ LOKAČNÍCH UPDATŮ

```
@Override
protected void onPause() {
    super.onPause();
    stopLocationUpdates();
}

protected void stopLocationUpdates() {
    LocationServices.FusedLocationApi.removeLocationUpdates(
            mGoogleApiClient, this);
}
```

```
@Override
public void onResume() {
    super.onResume();
    if (mGoogleApiClient.isConnected() && !mRequestingLocationUpdates) {
        startLocationUpdates();
    }
}
```

LITERATURA

- Kompletní popis v dokumentaci:

http://developer.android.com/training/location/receive-location-updates.html

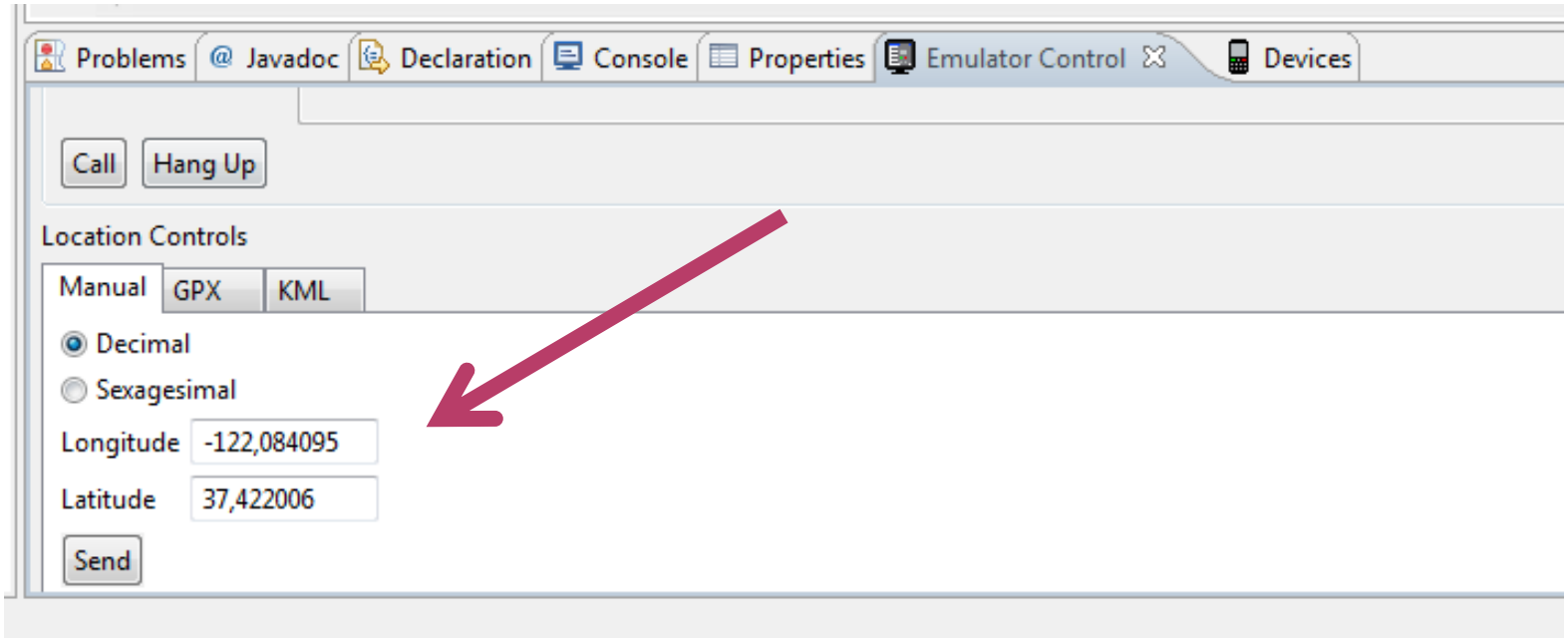
http://developer.android.com/training/location/index.html

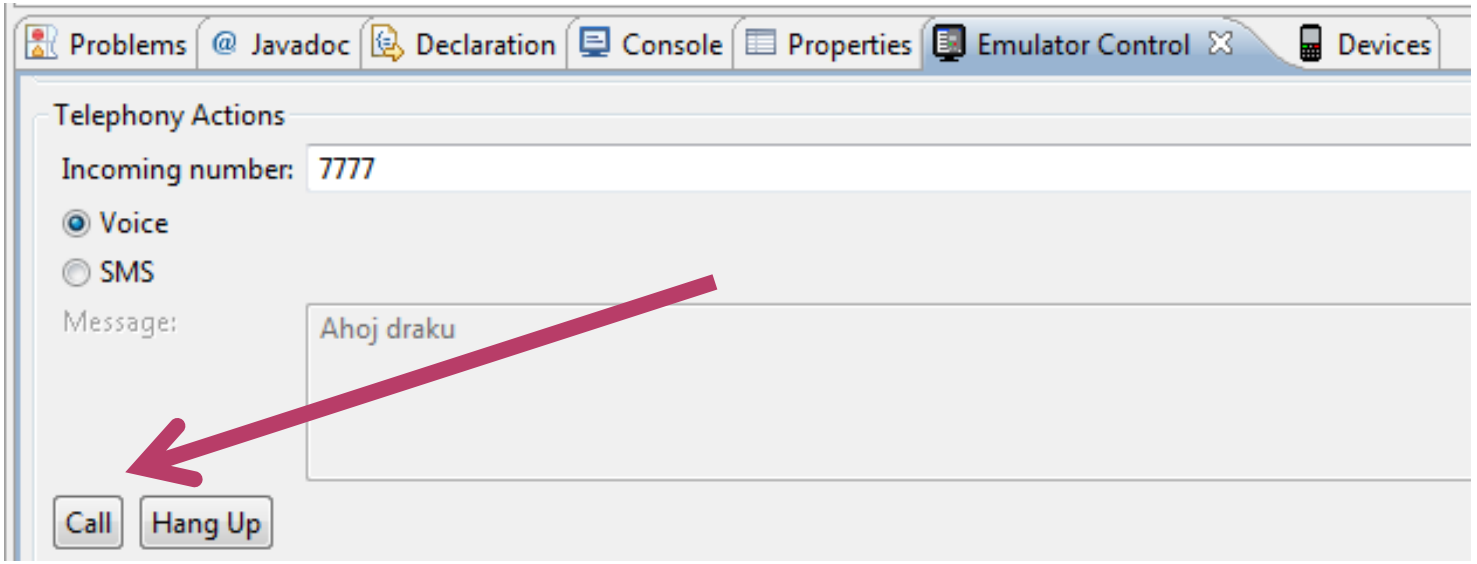
EMULÁTOR

můžeme emulovat různé události:

- příchozí telefonní hovor
- příchozí SMS
- lokace
  - manuální, GPX, KML
  - přehrávání souboru se změnou polohy..

Eclipse – Windows – Show View – Other – Emulator control


Problems
Javadoc
Declaration
Console
Properties
Emulator Control
Devices
Call
Hang Up
Location Controls
Manual
GPX
KML
Decimal
Sexagesimal
Longitude
-122,084095
Latitude
37,422006
Send


Problems
Javadoc
Declaration
Console
Properties
Emulator Control
Devices
Telephony Actions
Incoming number:
7777
Voice
SMS
Message:
Ahoj draku
Call
Hang Up

WIDGETY

- miniaplikace (na domovské obrazovce, lock screen)
  - aplikace zapouzdřená v jiné aplikaci
- obsah pravidelně aktualizován
  - přijímá updaty
  - min. interval 30 min (lze častěji přes AlarmManager)
  - probudí zařízení i ze spánku!! spotřeba !!

- Příklady widgetů:
  - předpověď počasí
  - nové zprávy
  - aktuální rozvrh hodin

POJMY Z OBLASTI WIDGETŮ

- **Appwidgethost**
  - aplikace, která hostí widgety
  - home screen nebo lock screen

- **RemoteView**
  - view, která lze použít ve widgetech
  - omezená podmnožina
  - FrameLayout, LinearLayout, RelativeLayout
  - AnalogClock, Button, Chronometer, ImageButton,
  - ImageView, ProgressBar, TextView
  - od Android 3.0 navíc:
    ListView, GridView, StackView, AdapterViewFlipper

POJMY Z OBLASTI WIDGETŮ

- **AppWidgetProvider**
  - widget představuje broadcast receiver
  - v manifestu přidán *<receiver …>*
    s akcí *APPWIDGET_UPDATE*
  - naše třída oddděná od AppWidgetProvider

  - implementujeme metodu **onUpdate()**
    vlastní funkcionalita widgetu
    do 5s volání hotovo,
    jinak zvážit spuštění služby,
    která na konci aktualizuje stav widgetu

TUTORIÁL

- widget
- widget s využitím služby

tutorial:

http://www.vogella.com/articles/AndroidWidgets/article.html

WIDGET

- objekt **AppWidgetProviderInfo**
  - XML soubor
  - layout
  - frekvence updatů (min 30minut, 0 nikdy)
  - AppWidgetProvider třída

- **AppWidgetProvider**
  - vlastní odvozená třída
  - základní metody pro interakci s widgetem, založené na broadcastech
  - broadcast při:
    updated, enabled, disabled, deleted

WIDGET

- layout je definován v XML
- lze vytvořit Activitu, která se spustí, když uživatel přidá widget a umožní jeho konfiguraci

I. MANIFEST - BROADCAST RECEIVER

```
<receiver android:name="ExampleAppWidgetProvider" >

    <intent-filter>
        <action
android:name="android.appwidget.action.APPWIDGET_UPDATE" />
    </intent-filter>

    <meta-data android:name="android.appwidget.provider"
            android:resource="@xml/example_appwidget_info" />
</receiver>
```

v manifestu uvedeme receiver, případně službu service
reaguje na událost APPWIDGET_UPDATE
dále cesta na XML soubor, popisující vlastnosti widgetu

II. KONFIGURACE WIDGETU (/RES/XML)

```
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="294dp"
    android:minHeight="72dp"
    android:updatePeriodMillis="86400000"
    android:previewImage="@drawable/preview"
    android:initialLayout="@layout/example_appwidget"
    android:configure="com.example.android.ExampleAppWidgetConfigure"
    android:resizeMode="horizontal|vertical"
    android:widgetCategory="home_screen|keyguard"
    android:initialKeyguardLayout="@layout/example_keyguard">
</appwidget-provider>
```

min…	- minimální velikost widgetu
interval updatu	– v ms, min. 30 min, co největší vhodný když zařízení spí, vzbudí se na update
initialLayout	- XML soubor se vzhledem widgetu
configure	- aktivita při přidání widgetu
widgetCategory	- home screen , lock screen (keyguard) XML layout při zamknuté obrazovce lze odlišný

FREKVENCE UPDATU

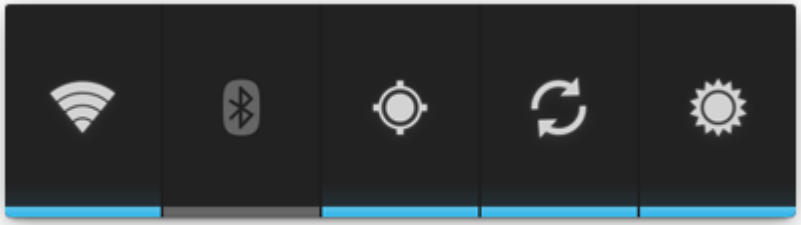
- minimálně 30 minut
- častější
  - nastavíme na 0 – nebude se aktualizovat
  - využít AlarmManager, služba
  - AĺarmManager.RTC – neprobudí zařízení ze spánku, k vyvolání dojde až po vzbuzení zařízení

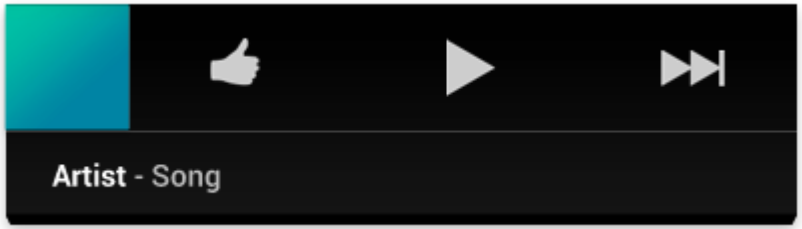
LAYOUT WIDGETU

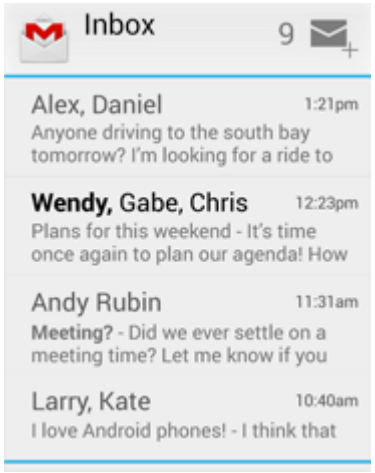
- úvodní layout v /res/layout
- formát widgetu (4x1 – celý řádek)
- layouty založené na RemoteViews (omezená množina)
- povolené layouty:
  Frame, Linear, Relative, Grid
- třídy widgetů:
  viz slide o RemoteView

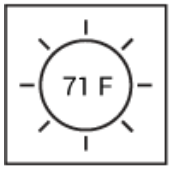
VZHLED WIDGETU

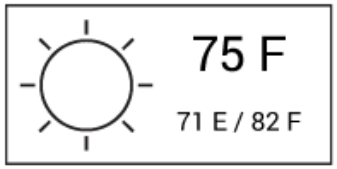
- informační
- kolekce – více informací stejného typu
- control widgety – ovládání něčeho
- hybridní – ovládání hudby + info o skladbě

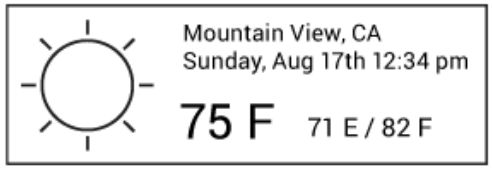


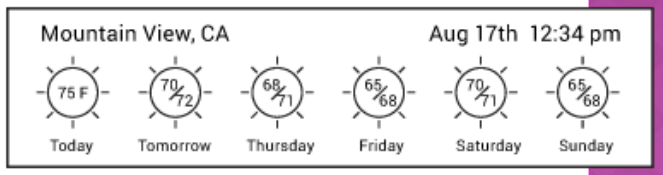












APPWIDGET PROVIDER

- rozšiřuje BroadcastReceiver
- přijímá pouze relevantní události
- volání metod při události:
- **onUpdate()**
  - voláno při dosažení updatovacího intervalu
  - při přidání widgetu (když není konfig. aktivita)
- **onAppWidgetOptionsChanged()**
  - při prvním umístění
  - při změně velikosti
  - zobrazit, skrýt obsah na základě velikosti
  - velikost zjistíme viz další slide

VELIKOST WIDGETU

- **getAppWidgetOptions()**
  - vrací Bundle
    - mapování ze Stringů do Parcelable typů
  - OPTION_APPWIDGET_MIN_WIDTH (dp)
  - OPTION_APPWIDGET_MIN_HEIGHT
  - OPTION_APPWIDGET_MAX_WIDTH
  - OPTION_APPWIDGET_MAX_HEIGHT

AppWidgetProvider - pokr.

další callbacky:

- onDeleted
  - při smazání z App Widget Hostu (z domovské obr.)
- onEnabled
  - když je instance vytvořena poprvé
  - když uživatel přidá druhou instanci, už se nevolá
- onDisabled
  - když je poslední instance odebrána
- onReceive
  - pro každý broadcast
  - před každým výše uvedeným callbackem
  - stačí nám využít onUpdate()

BROADCAST RECEIVER

- Komponenta
- Registruje se na systémovou nebo aplikační událost
- Všechny registrované receivery jsou informovány o dané události

- Dědí od BroadcastReceiver
- Je definovaný jako receiver v manifestu
- V programu lze: Context.registerReceiver()
- Při události se volá **onReceive**(context, intent)

Aktivace broadcast receiveru

- Vytvoří se Intent s nějakou akcí
- Předá se metodě sendBroadcast()

SPUŠTĚNÍ SLUŽBY BROADCAST RECEIVEREM

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class MyReceiver extends BroadcastReceiver {
@Override
public void onReceive(Context context, Intent intent)  {

   Intent service = new Intent(context, WordService.class);
    context.startService(service);
}
}
```

PENDING INTENT

- Token, který dáme jiné aplikaci (notifikační manager, alarm manager,…)
- Dovolí jiné aplikaci použít práva naší aplikace k vykonání předdefinované části kódu

- PendingIntent.getActivity()
- PendingIntent.getBroadcast()

```
Intent intent = new Intent(this, UpdateReceiver.class);
intent.putExtra("someExtra", true);
PendingIntent pi = PendingIntent.getBroadcast(this, 0, intent, 0);
```

SPUŠTĚNÍ SLUŽBY PŘES ALARM

```
Intent intent = new Intent(this, MyService.class);

PendingIntent pintent = PendingIntent.getService(this, 0, intent, 0);

AlarmManager alarm =
    (AlarmManager)getSystemService(Context.ALARM_SERVICE);

// Start every 30 seconds
    alarm.setRepeating(AlarmManager.RTC_WAKEUP,
    cal.getTimeInMillis(), 30*1000, pintent);
```

NAHRÁVÁNÍ DAT NA POZADÍ

- nějaký čas zabere ContentProvideru poskytnout nám data => zpoždění, ANR, …
- iniciovat dotaz na samostatném vláknu, čekat na výsledek, pak zobrazit
- lze zařídit objektem, který spustí dotaz asynchronně na pozadí a připojí se k Aktivitě, až bude výsledek => CursorLoader
- CursorLoader navíc automaticky znovu spustí dotaz, když se data změní

CURSOR LOADER

- asynchronní nahrávání dat u aktivit a fragmentů
- od Android 3.0 (API 11)
- CursorLoader
- obecný Loader, rozšířit AsyncTaskLoader
- LoaderCallbacks<Cursor> rozhraní

- příklad: http://www.compiletimeerror.com/2013/12/how-to-use-android-cursorloader.html#.UzLDtIXqJVA

CURSOR LOADER

- inicializace dotazu
  - LoaderManager.initLoader()
- spuštění dotazu
  - onCreateLoader()
- dokončení dotazu
  - onLoadFinished()
- dojde ke změně dat
  - onLoaderReset()

použití např.
http://developer.android.com/training/load-data-background/handle-results.html