Helymeghatározás, térképkezelés

Ekler Péter

BME VIK AUT, AutSoft

peter.ekler@aut.bme.hu

Tematika

1. Android platform bemutatása, Kotlin alapok
2. Alkalmazás komponensek, Kotlin konvenciók
3. Felhasználói felület
4. Fragmentek, haladó UI
5. Listák kezelése hatékonyan
6. Perzisztens adattárolás, adatbázisok, haladó Kotlin
7. Felhő szolgáltatások
8. Hálózati kommunikáció
9. Helymeghatározás, térkép kezelés
10. Architektúra komponensek, JetPack

Tartalom

- Intent-ek típusai
- BroadcastReceiver komponens
- Helymeghatározás, Fused location
- Geocoding/Reverse geocoding
- Térkép megjelenítés
- Markerek kezelése, overlayek
- Eseménykezelés térképen

Komponensek közötti kommunikáció, Intent

Sandbox modell

- A legtöbb platformon az alkalmazások egymástól elkülönítve futnak
  - Minden app a saját „homokozójában” (S*andbox*)
  - Szigorú korlátozások a sandbox-ból kinyúló műveletekre
    - Hardver elérés, pl kamera, szenzorok, stb
    - Rendszerszintű adatok, háttértár
    - Szálak, alk. komponensek közti kommunikáció
  - Cél: adatvédelem, alkalmazások védelme egymástól

Lazán csatolt komponensek és köztük a kommunikáció

Mi a helyzet Androidon?

- Alkalmazások külön VM példányokban (ez is sandbox)
- Kritikus műveletekhez engedély szükséges
- Alkalmazás = komponensek halmaza

A komponensek akár alkalmazások között is kommunikálhatnak egymással (!)

- Két komponens között: ***Intent***
- Egy komponensből mindenki másnak: ***Broadcast Intent***
- Csak adat megosztása (ContentProvider)

Kommunikáció formái 1/3

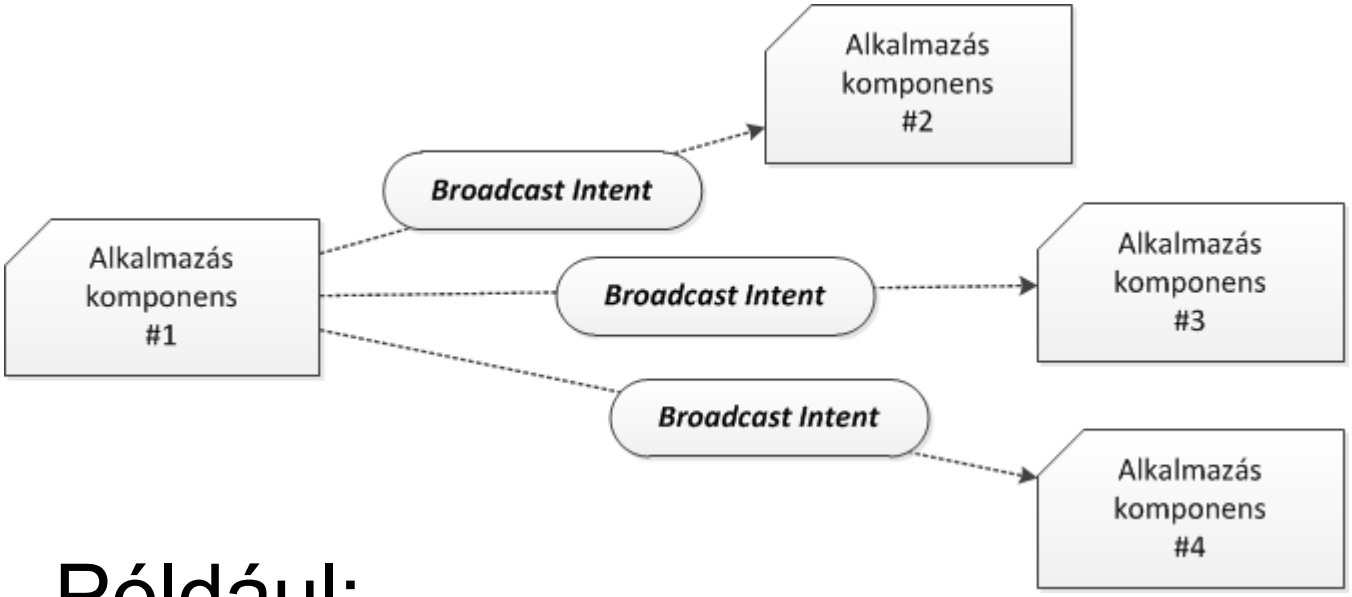
- Egyik komponensből a másikba: ***Intent***

Például:

- Következő képernyőre lépés (új Activity indítása)
- Zenelejátszó service indítása

Kommunikáció formái 2/3

- Egy komponensből mindenki másnak: ***Broadcast Intent***

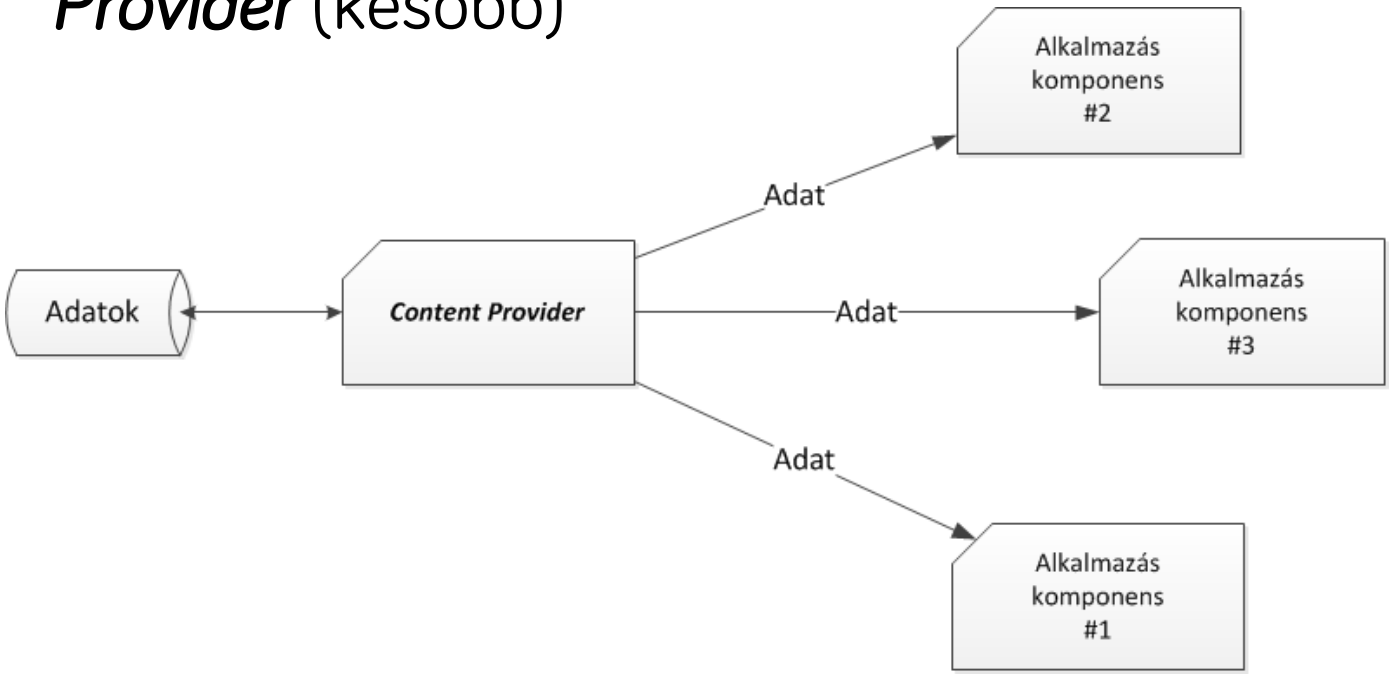


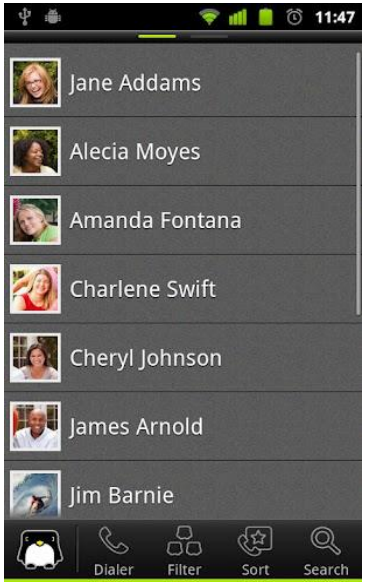
Például:

- „Akkufeszültség alacsony” rendszerüzenet

Kommunikáció formái 3/3

Adatok szolgáltatása komponensek közt: ***Content Provider*** (később)



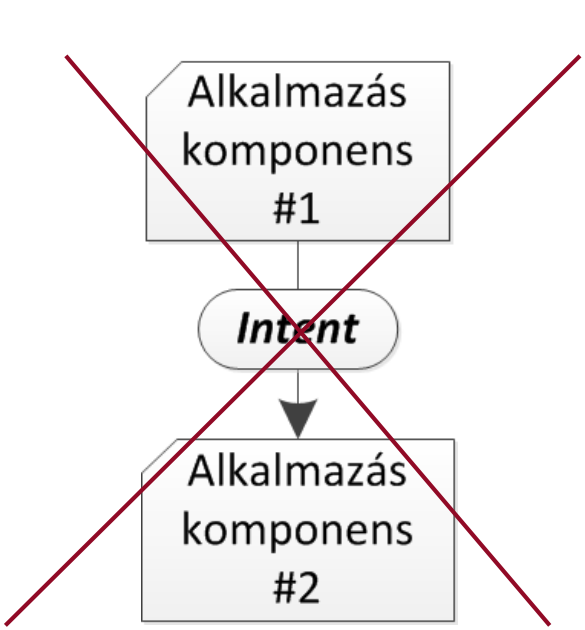


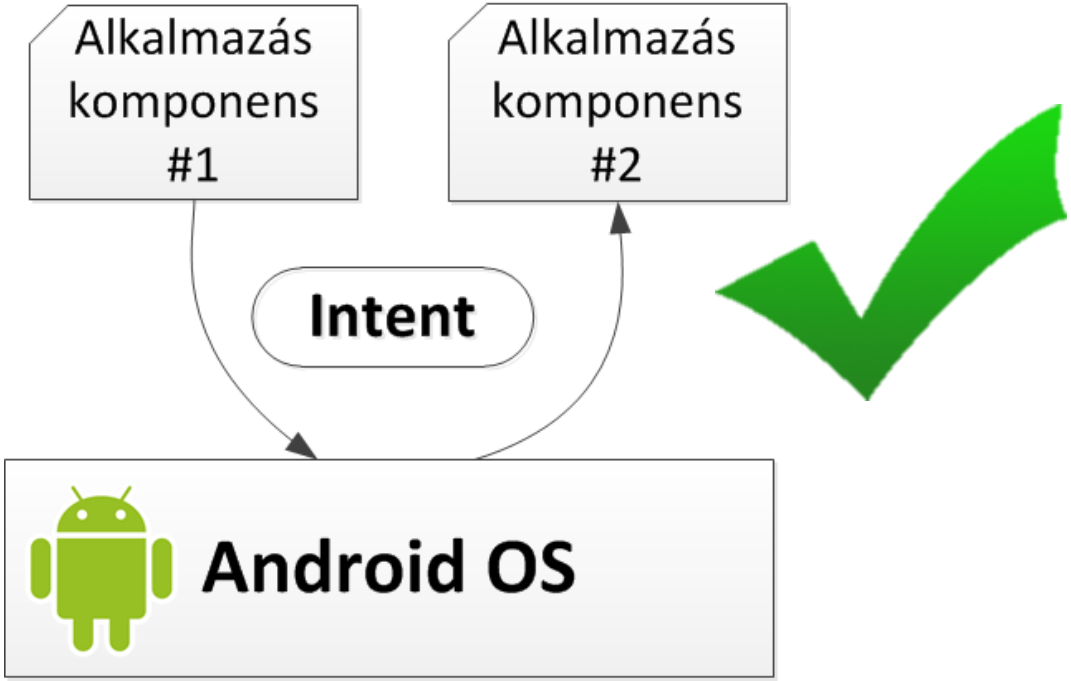
Például:

- Névjegyzék elérése saját alkalmazásból

Intent átadása

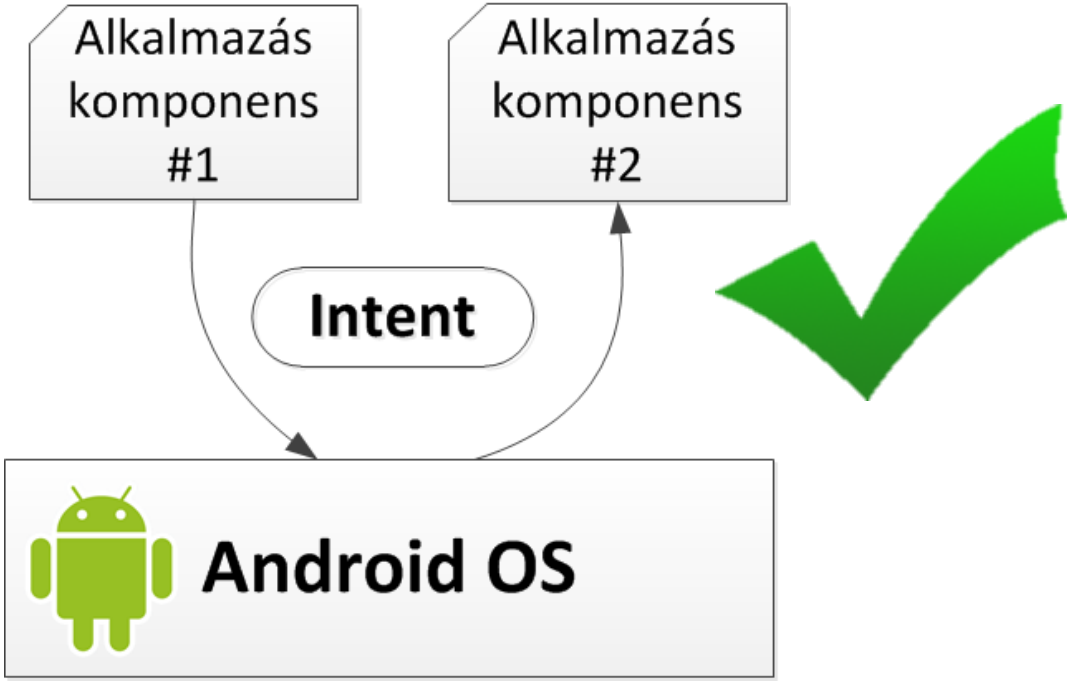
- Mindig az Android runtime-on keresztül!

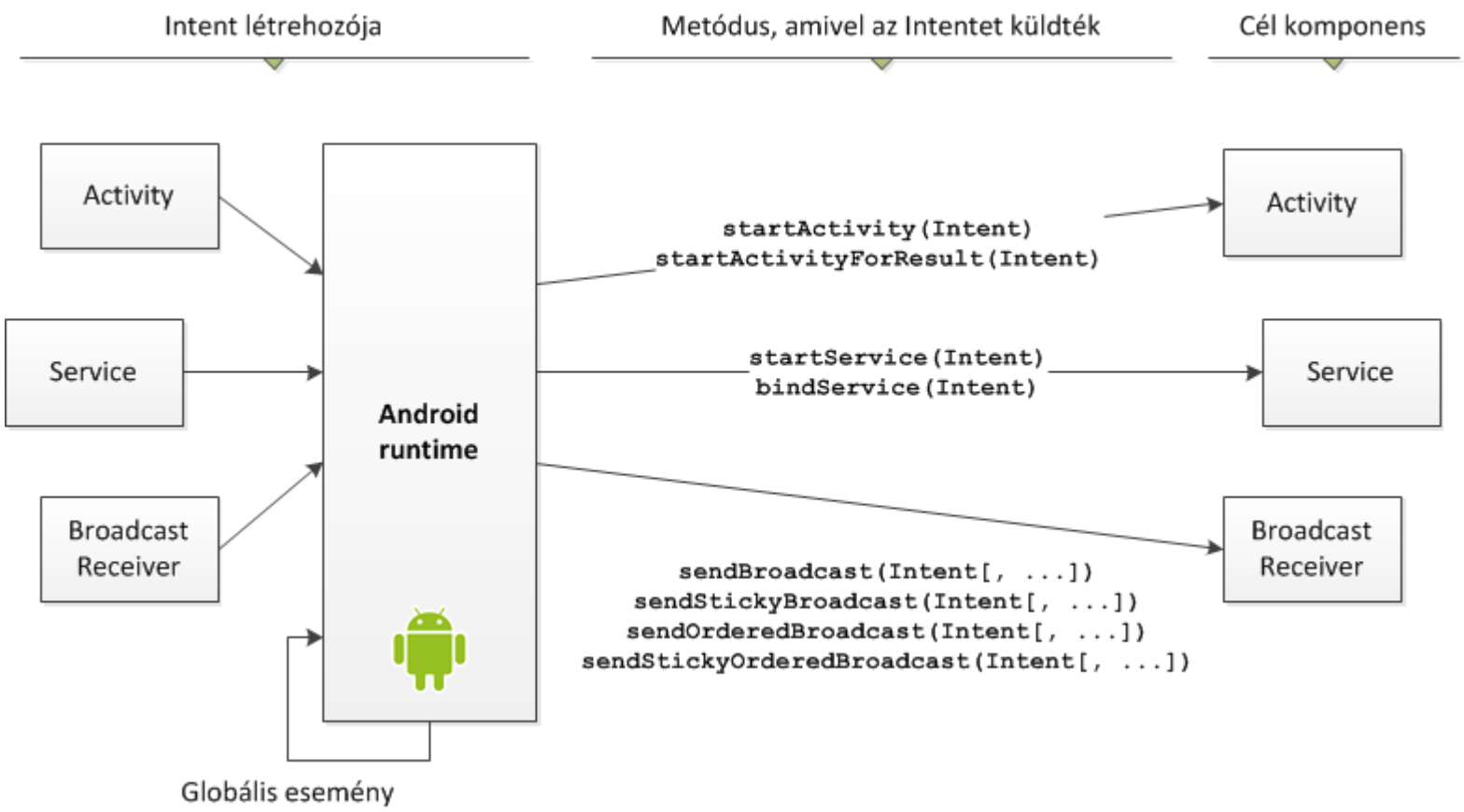




Intent átadása

- Mindig az Android runtime-on keresztül!




Intent létrehozója
Metódus, amivel az Intentet küldték
Cél komponens
Activity
Service
Broadcast Receiver
Android runtime
Globális esemény
startActivity(Intent)
startActivityForResult(Intent)
Activity
startService(Intent)
bindService(Intent)
Service
sendBroadcast(Intent[, ...])
sendStickyBroadcast(Intent[, ...])
sendOrderedBroadcast(Intent[, ...])
sendStickyOrderedBroadcast(Intent[, ...])
Broadcast Receiver

Intent típusai és részei

- **Intent típusok:**
  - Explicit Intent:
    - konkrétan meg van nevezve a cél komponens
  - Implicit Intent:
    - a végrehajtandó feladat kerül leírásra
- **Intent részei:**
  - **Címzett komponens osztályneve** *(Component name)*: ha üres akkor az Android megkeresi a megfelelőt
  - **Akció** *(Action)*: az elvárt vagy megtörtént esemény
  - **Adat** *(Data)*: az adat (URI-ja és MIME típusa), amin az esemény értelmezett
  - **Kategória** *(Category)*: további kritériumok a feldolgozó komponessel kapcsolatban
  - **Extrák** *(Extras)*: saját kulcs-érték párok, amiket át akarunk adni a címzettnek
  - **Kapcsolók** *(Flags)*: Activity indításának lehetőségei

Explicit Intent

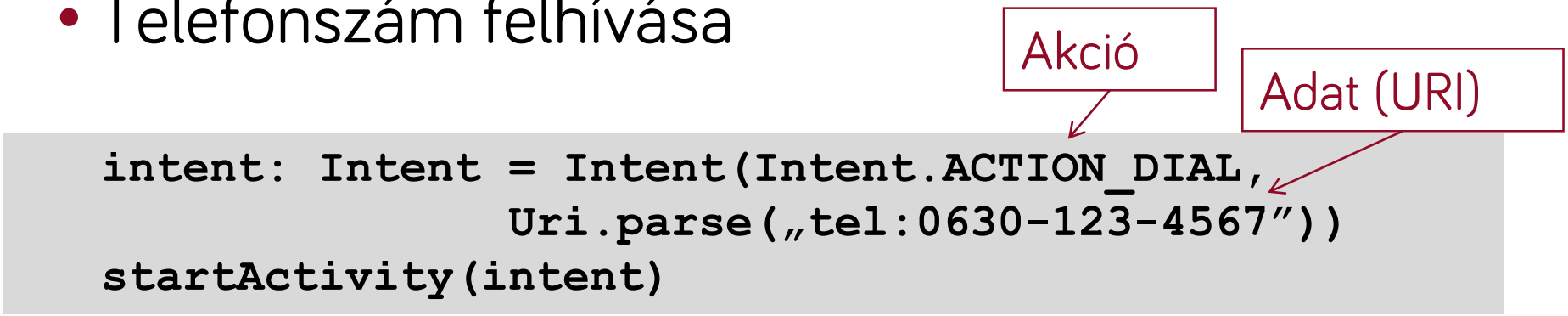
- Mindkét esetben a *startActivity()* függvényt használjuk:
  - `startActivity(Intent)`
- **Explicit hívás**: az Intent-ben kitöltjük a címzett komponens nevét (konstruktorból vagy setterrel)

```
intent: Intent = Intent(getApplicationContext(),
                        ListProductsActivity::class.java)
startActivity(intent)
```

- Ha a **`ListProductsActivity`**-ből már van példány a memóriában akkor folytatódik, ha nincs akkor az Android példányosítja és elindítja

Implicit Intent - Példa

- Telefonszám felhívása



- Névjegy kiválasztása

Intent képességek

- Android alkalmazás komponensek:
  - Activity, Service, Broadcast Receiver
- Kommunikáció köztük: Intentekkel
- Nem csak alkalmazáson belül, hanem azok között is lehetséges
  - Használhatunk más alkalmazásban lévő komponenst
  - Kiajánlhatjuk a sajátunkat
- Rendszerszintű eseményeket kezelhetünk

Intent Filter

- Lehetséges a saját alkalmazásunk funkcióinak kiajánlása mások számára
  - Az Androidban beépítve vannak ilyenek, ld. Intent Action (pl. ACTION_CALL, ACTION_IMAGE_CAPTURE)
- Az AndroidManifest-ben kell deklarálni (miért?)
- Ha nincs Intent filter beállítva, akkor a komponens kizárólag explicit intentet képes fogadni
- Ha van Intent filter, akkor explicit és implicit intenteket is ki tud szolgálni

BroadcastReceiver komponens

Broadcast események

- Rendszerszintű eseményekre fel lehet iratkozni – Broadcast üzenet
- Az Intent alkalmas arra hogy leírja az eseményt
- Sok beépített Broadcast Intent, lehet egyedi is

```
ACTION_TIME_TICK
ACTION_TIME_CHANGED
ACTION_TIMEZONE_CHANGED
ACTION_BOOT_COMPLETED
ACTION_PACKAGE_ADDED
ACTION_PACKAGE_CHANGED
ACTION_PACKAGE_REMOVED
ACTION_PACKAGE_RESTARTED
ACTION_PACKAGE_DATA_CLEARED
```

```
ACTION_UID_REMOVED
ACTION_BATTERY_CHANGED
ACTION_POWER_CONNECTED
ACTION_POWER_DISCONNECTED
ACTION_SHUTDOWN
```

Broadcast események

- Nem csak az Android, hanem alkalmazások (Activity-k és Service-ek) is dobhatnak Broadcast Intentet
  - Telephony service küldi az ACTION_PHONE_STATE_CHANGED Broadcast Intentet, ha a mobilhálózat csatlakozás megváltozott
  - android.provider.Telephony.SMS_RECEIVED
  - Sok más, érdemes tájékozódni ha valamit szeretnénk lekezelni
  - Saját alkalmazásunkból is dobhatunk a **`sendBroadcast(String action)`** metódussal
  - Ez is Intent, lehet Extra és Data része

Broadcast intentek elkapása

- Broadcast Receiver nevű komponens segítségével
  - Kódból vagy manifestben kell regisztrálni
    - (bizonyos Action-ök esetén nem mindegy, tájékozódni!, pl. TIME_TICK)
  - Intent filterrel állíthatjuk be hogy milyen Intent esetén aktivizálódjon
- Nem Activity, nincs felhasználói felülete
- Azonban képes Activity-t indítani
- Használata: BroadcastReceiver osztályból származtatunk, és felüldefiniáljuk az onReceive() metódust, majd intent-filter

Broadcast intentek kezelése

```
class OutgoingCallReceiver : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
        val outNumber = intent.getStringExtra(Intent.EXTRA_PHONE_NUMBER)
        Toast.makeText(context, outNumber, Toast.LENGTH_LONG).show()

    }
}
```

AndroidManifest.xml:

```
<receiver android:name=".OutgoingCallReceiver">
    <intent-filter>
        <action android:name=
          "android.intent.action.NEW_OUTGOING_CALL"/>
    </intent-filter>
</receiver>
```

Broadcast intentek kezelése

Broadcast továbbdobásának megakadályozása:

- abortBroadcast()

Például ha a fülhallgató média gombjait kell kezelni és nem akarjuk, hogy a zenelejátszó is megkapja a Broadcast-ot ☺

Gyakoroljunk!

- Készítsünk egy AirPlane mód változásra figyelő BroadcastReceivert!
- Készítsünk egy kimenő hívásra figyelő BroadcastReceivert!
  - Változtassuk meg a hívott számot!
  - Egészítsük ki SMS figyeléssel!
    - Valósítsuk meg, hogy ne kerüljön be inbox-ba a bejövő SMS

SMS Receiver 1/2

- Manifest:

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

...

<receiver android:name=".SMSReceiver"  android:enabled="true">

    <intent-filter android:priority="1000">

        <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
</intent-filter>

</receiver>
```

SMS Receiver 2/2

```
class SMSReceiver : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
        val extras = intent.extras ?: return
        val pdus = extras.get("pdus") as Array<ByteArray>
        for (pdu in pdus) {
            val msg = SmsMessage.createFromPdu(pdu)
            val origin = msg.originatingAddress
            val body = msg.messageBody
            Toast.makeText(context,
                  "SMS caught, number: $origin body: $body", Toast.LENGTH_LONG)
                  .show()
        }
    }
}
```

Alkalmazáskomponens indítása Boot után

- Néha olyan szolgáltatásokra van szükség, amelyek mindig futnak a készüléken
- Ilyen esetben fontos, hogy a készülék indítása esetén ezek automatikusan is el tudjanak indulni
- Az Android lehetőséget biztosít arra, hogy feliratkozzunk a „*Boot befejeződött*” eseményre és valamilyen alkalmazás komponenst elindítsunk:
  - *BroadcastReceiver* definiálása Manifest-ben
  - *android.intent.action.BOOT_COMPLETED*
- *A BroadcastReceiver onReceive()* függvényében elindíthatjuk a megfelelő komponenst

Helymeghatározás

Helymeghatározás

- Android által támogatott módok:
  - GPS
  - Hálózat: Android's Network Location Provider
- GPS pontos, de csak kültéren működik és magas az energiaigénye
- Hálózat alapú:
  - Cella alapú helymeghatározás
  - WiFi alapú helymeghatározás
  - Gyors, de gyakran pontatlan
- Egy időben akár mindkét módszert is alkalmazhatjuk

Nem triviális a pozíció meghatározása

- GPS, mobil hálózat és WiFi alapú metódus is adhat egyidőben információt, de figyelembe kell venni a következőket:
  - Pontosság
  - Sebesség
- A felhasználó folyamatos mozgásban lehet, sokszor szükség van becslésre
- Előfordulhat, hogy egy frissen kapott pozíció pontatlanabb, mint a 10 másodperccel ezelőtti!

Location engedély beállítása

- Manifest állományba új permission

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

- `ACCESS_COARSE_LOCATION`: csak hálózat alapú helymeghatározás engedély
- `ACCESS_FINE_LOCATION`: GPS és hálózat alapú helymeghatározás engedély

Android Location API 1/2

- Callback függvényekben érkezik az információ
- `LocationManager` rendszerszolgáltatás lekérése:

```
var locationManager: LocationManager =
getSystemService(Context.LOCATION_SERVICE) as LocationManager
```

- Folyamatos frissítés kérése egy `LocationListener` implementáció átadásával

```
// provider, mintime, mindistance, listener
locationManager.requestLocationUpdates(Locati
onManager.NETWORK_PROVIDER, 0, 0,
locationListener)
```

Android Location API 2/2

- requestLocationUpdates() paraméterei:
  - Provider típus (GPS vagy hálózat alapú)
    - Kétszer is meghívható más-más providerrel (GPS és hálzózat)
  - Minimum idő két frissítés között (0: lehető leggyakrabban)
  - Minimum távolság két frissítés között (0: lehető leggyakrabban)
  - LocationListener implementáció
- LocationListener callback függvényei:
  - Új pozíció:

```
onLocationChanged(Location location)
```

  - Provider állapotváltozás:

```
onStatusChanged(String provider, int status, Bundle extras)
```

  - Provider használhatóvá vált:

```
onProviderEnabled(String provider)
```

  - Provider nem használható

```
onProviderDisabled(String provider)
```

LocationListener példa

```
var locationListener: LocationListener = object  : LocationListener {
    override fun onLocationChanged(location: Location?) {
    }

    override fun onStatusChanged(provider: String?,
      status: Int, extras: Bundle?) {
    }

    override fun onProviderEnabled(provider: String?) {
    }

    override fun onProviderDisabled(provider: String?) {
    }

}
```

Leiratkozás

- `removeUpdates(...)` függvény hívás minden létrehozott `LocationListener`-re
- Nagyon fontos, hogy **ne felejtsük el**!
- `locationManager.removeUpdates(locationListener)`

FusedLocationProvider

- Alkalmazások képesek megosztani a hely információkat egymással
- Google Play Services függőség:

```
implementation 'com.google.android.gms:play-services-location:11.6.0'
```

- FusedLocationProviderClient

```
private val fusedLocationClient: FusedLocationProviderClient =
        LocationServices.getFusedLocationProviderClient(context)
```

Indítás és leállítás

```
fun startLocationMonitoring() {
    val locationRequest = LocationRequest()
    locationRequest.interval = 1000
    locationRequest.fastestInterval = 500
    locationRequest.priority = LocationRequest.PRIORITY_HIGH_ACCURACY


    fusedLocationClient.requestLocationUpdates(locationRequest,
            locationCallback, Looper.myLooper())

}

fun stopLocationMonitoring() {
    fusedLocationClient.removeLocationUpdates(locationCallback)
}
```

```
private var locationCallback: LocationCallback = object : LocationCallback() {
    override fun onLocationResult(locationResult: LocationResult) {
        super.onLocationResult(locationResult)
    }
}
```

Fontos tippek & javaslatok

- Ellenőrizzük, hogy a kapott pozíció jelentősen újabb-e
- Ellenőrizzük a kapott pozíció pontosságát (accuracy)
- Az előző pozícióból és a hozzá tartozó sebességből ellenőrizhető az új pozíció realitása
- Ellenőrizzük melyik provider-től származik az új pozíció
- 60 másodpercnél gyakoribb pozíciókérés sokszor felesleges
- Nem mindig kell GPS és hálózati providert egyszerre használni

Térkép nézet

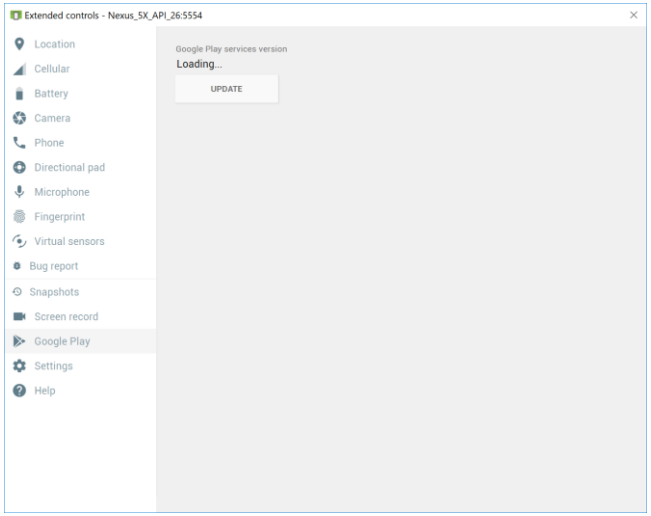
- Földrajzi pozíciók megjelenítése térkép-szerűen
- Teljes kontroll a megjelenítés felett
  - Helyszín, nagyítási szint
  - Térkép, műhold, traffic
- Ráhelyezhetőek overlay-ek
- Megjeleníthetőek rajta tetszőleges POIk

Google Maps API V2

- Google Play Services SDK része
- MapFragment-be helyezett map nézet
  - Kis kijelzőkön jól használható
  - Nagy kijelzőkön könnyebben lehet komplex nézeteket létrehozni
- Nincs szükség MapActivity-re, mint V1-ben
- Vektoros térkép csempék:
  - Gyorsabb megjelenítés
  - Kevesebb adatforgalom
- Fejlettebb térkép cache
- 3D-s nézet támogatás, perspektívikus megjelenítése

Térkép nézet készítése

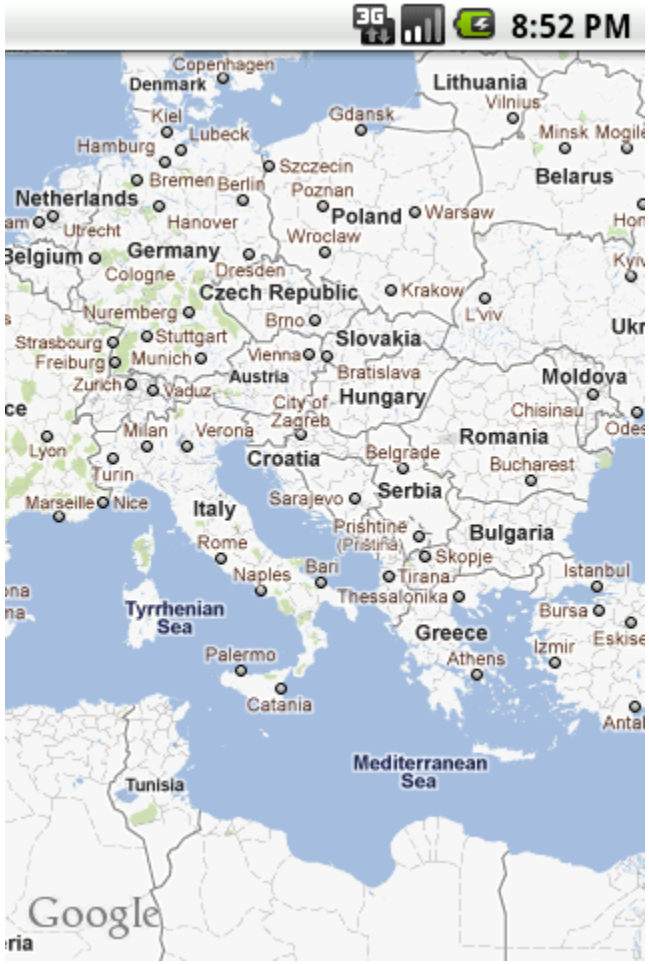
- MapFragment alapú térkép megjelenítés
- A térkép letöltése darabonként, on-demand történik, tehát Internet permission-re szükség van
- Manifest engedélyek beállítása
- OpenGL ES jelzése
- Projekt regisztrálása és API key igénylés
  - Google APIs console
  - https://code.google.com/apis/console/
- Ne importoljuk a teljes play servicest, csak ami kell:
  - https://developers.google.com/android/guides/setup
  - *Emulátoron érdemes frissíteni a Play Services-t*



MapFragment paraméterei

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:map="http://schemas.android.com/apk/res-auto"
  android:id="@+id/map"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  class="com.google.android.gms.maps.SupportMapFragment"
  map:cameraBearing="112.5"
  map:cameraTargetLat="-33.796923"
  map:cameraTargetLng="150.922433"
  map:cameraTilt="30"
  map:cameraZoom="13"
  map:mapType="normal"
  map:uiCompass="false"
  map:uiRotateGestures="true"
  map:uiScrollGestures="false"
  map:uiTiltGestures="true"
  map:uiZoomControls="false"
  map:uiZoomGestures="true"/>
```

Demo – MapFragment felület



MapFragment megjelenítés 1/3

- Google APIs Target kiválasztása projekt létrehozáskor
- Szükséges OpenGLES osztálykönyvtár használat megjelölése a manifest állomány <application> tag-jében:
  - `<uses-feature android:glEsVersion="0x00020000“ android:required="true" />`
- Szükséges engedélyek:
  - `android.permission.INTERNET`
  - `android.permission.ACCESS_NETWORK_STATE`
  - `android.permission.WRITE_EXTERNAL_STORAGE`
  - `com.google.android.providers.gsf.permission.READ_GSERVICES`
  - `<permission android:name= "[package].permission.MAPS_RECEIVE“ android:protectionLevel="signature" />`
- Opcionális: címsor elrejtése nagyobb terület érdekében
  - `<activity android:name= ".HelloGoogleMaps" android:label= "@string/app_name" android:theme= "@android:style/Theme.NoTitleBar">`

MapFragment megjelenítés 2/3

- Map API kulcs Manifest *<application>* tagjén belül

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

- MapFragment XML-ben:

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android=
      "http://schemas.android.com/apk/res/android"
    android:id="@+id/map"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:name=
      "com.google.android.gms.maps.SupportMapFragment"/>
```

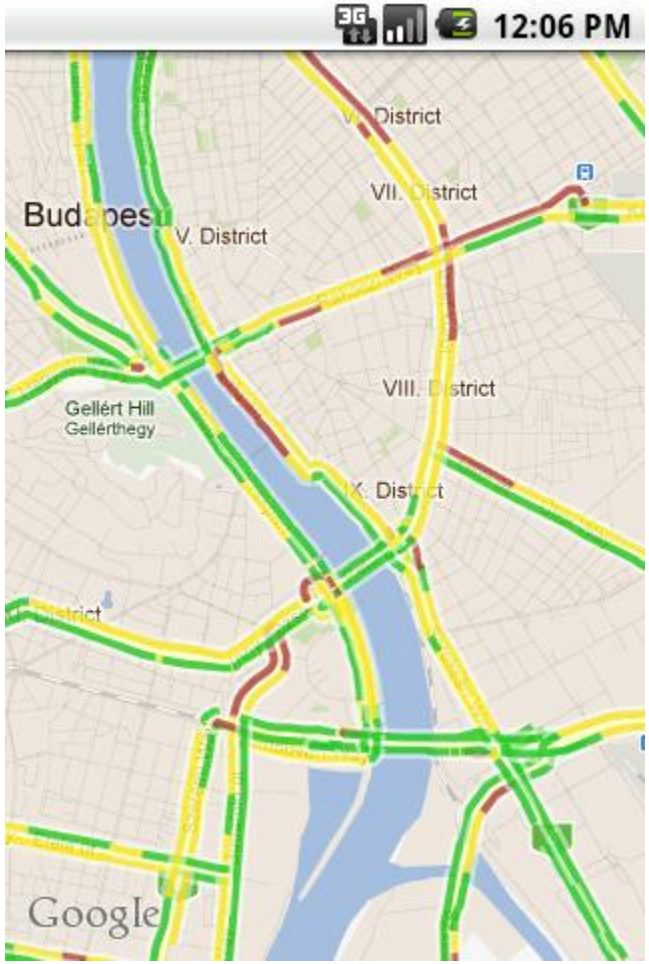
MapFragment megjelenítés 3/3

- `Activity` leszármaztatása elegendő
- `SupportMapFragment` használata, ha szükséges a visszafele kompatibilitás
- `MapFragment` egy `MapView`-ban
- `GoogleMap` elkérése és vezérlése
- Google Maps API AVD létrehozása, Play Services verzió ellenőrzése

Map kezelése - példa

```
class MapsActivity : AppCompatActivity(), OnMapReadyCallback {
    private lateinit var myMap: GoogleMap

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_maps)
        val mapFragment = supportFragmentManager
                .findFragmentById(R.id.map) as SupportMapFragment
        mapFragment.getMapAsync(this)
    }

    override fun onMapReady(googleMap: GoogleMap) {
       myMap = googleMap

       myMap.isTrafficEnabled = true
       myMap.mapType = GoogleMap.MAP_TYPE_SATELLITE

       val budapest = LatLng(47.0, 19.0)
       myMap.addMarker(MarkerOptions()
                       .position(budapest)
                       .title("Marker in Hungary"))
       myMap.moveCamera(CameraUpdateFactory.newLatLng(budapest))
    }
}
```

Demo - Forgalmi nézet példa



Térkép nézet vezérlése

- Érintés esemény kezelése
  - `GoogleMap.setOnMapClickListener(OnMapClickListener)`
- UiSettings objektum:

```
myMap.uiSettings.isRotateGesturesEnabled = true
myMap.uiSettings.isCompassEnabled = true
myMap.uiSettings.isZoomControlsEnabled = true
```

MapView

- `View` osztály leszármazottja
- Térkép megjelenítése
- Konténer szerep `GoogleMap` objektumon keresztül
- `Activity` életciklus függvényeit továbbítani kell a `MapView` fele
- Egy *Activity* egyszerre jelenleg *leginkább* csak egy *MapView*-t támogat

Marker megjelenítése 1/2

```
val hungary = LatLng(47.0, 19.0)
myMap.addMarker(MarkerOptions().
  position(hungary).
  title("Marker in Hungary"))
myMap.moveCamera(CameraUpdateFactory.newLatLng(hungary))
```

Marker megjelenítése 2/2

```
val markerHU = myMap.addMarker(
        MarkerOptions()
                .position(hungary)
                .title("Magyarország")
                .snippet("Lakosság: 9.700.000")
                .icon(BitmapDescriptorFactory.fromResource(
                        R.mipmap.ic_launcher_round)))
markerHU.isDraggable = true
```

Marker kezelés

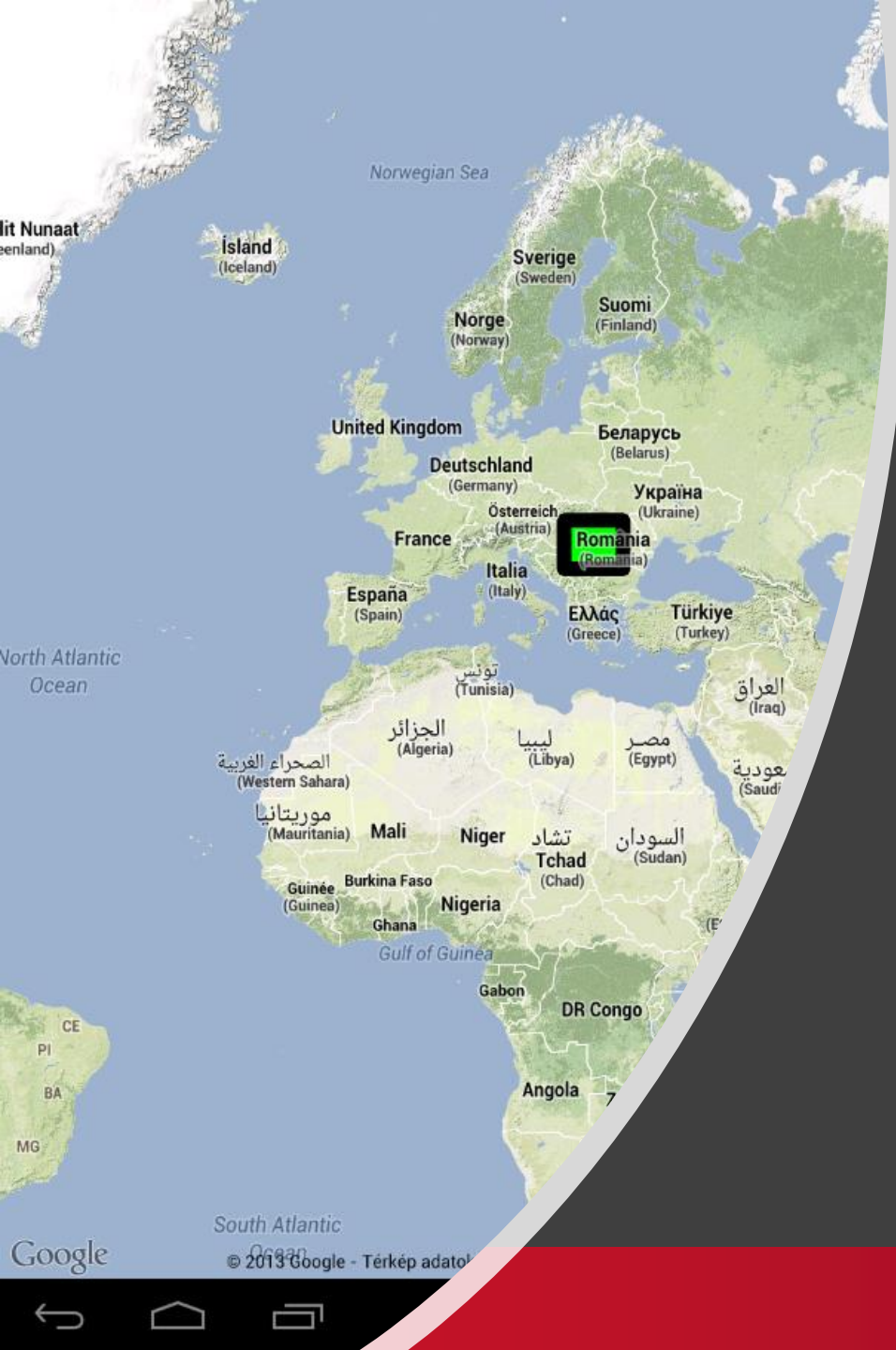
- `InfoWindow`:
  - Testreszabható `InfoWindow` felület
  - `InfoWindowAdapter`
  - Megjelenítés/eltüntetés progamozottan
  - Eseménykezelés: `OnInfoWindowClickListener`
- Marker eseménykezelők:
  - `OnMarkerClickListener`
  - `OnMarkerDragListener`
  - Stb.

Rajzolás térképre

- Támogatott elemek:
  - Polygon
  - Polyline
  - Circle
- Testre szabható megjelenítés
  - Vonal szín
  - Kitöltés szín
  - Z-index
  - Láthatóság
  - Stb.

Téglalap rajzolása térképre

```
val polyRect: PolygonOptions = PolygonOptions().add(
        LatLng(44.0, 19.0),
        LatLng(44.0, 26.0),
        LatLng(48.0, 26.0),
        LatLng(48.0, 19.0))
val polygon: Polygon = myMap.addPolygon(polyRect)
polygon.fillColor = Color.GREEN
```

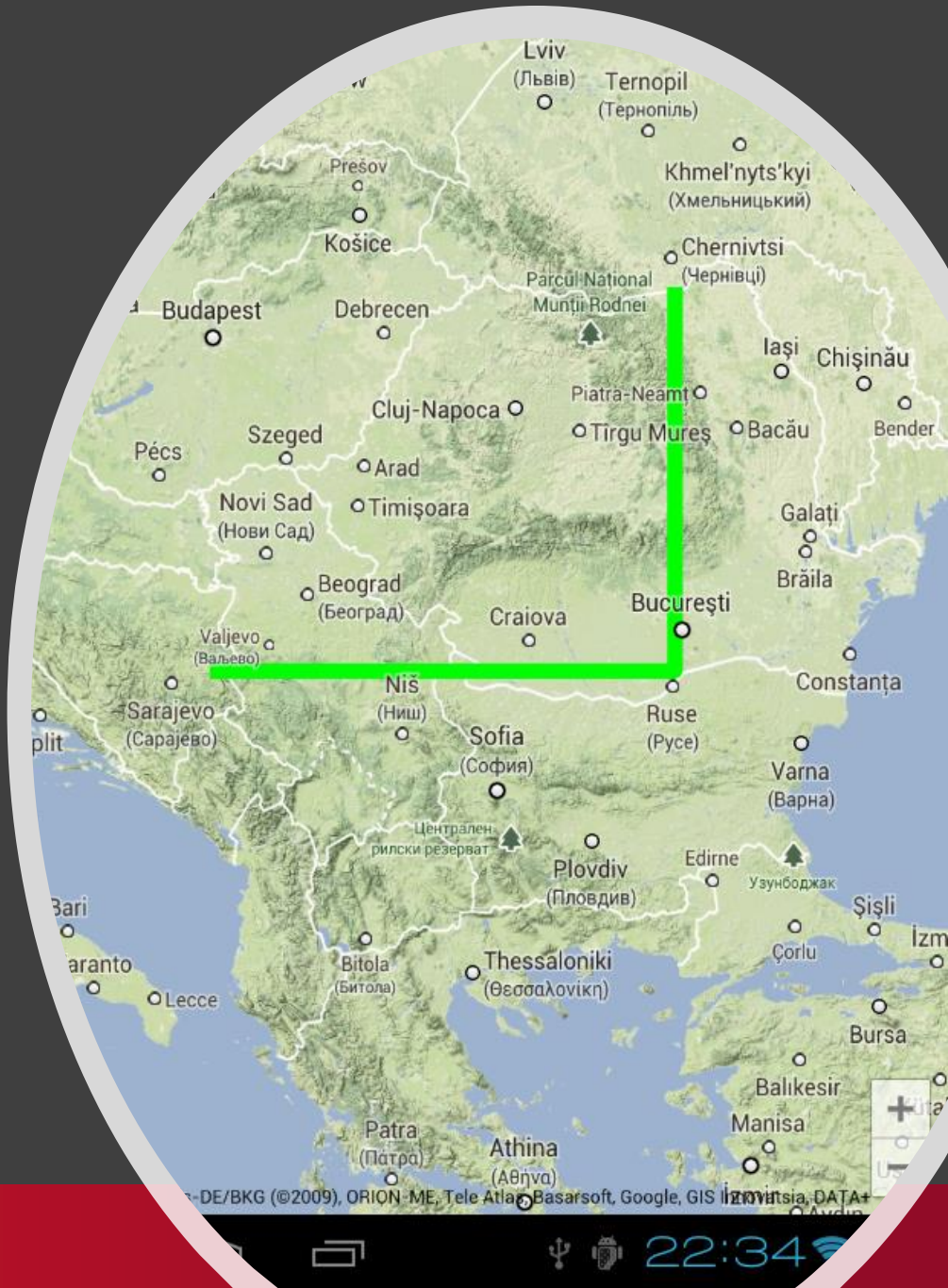


Demo – Téglalap rajzolás

Vonal rajzolás példa

```
val polyLineOpts = PolylineOptions().add(
        LatLng(44.0, 19.0),
        LatLng(44.0, 26.0),
        LatLng(48.0, 26.0))
val polyline = myMap.addPolyline(polyLineOpts)

polyline.color = Color.GREEN
```

Demo – Vonal rajzolás



További térkép funkciók

- `moveCamera()`: kamera mozgatása
- `animateCamera()`: animált mozgatás
- CameraPosition: kamera állítása factory-val
  - Target
  - Zoom
  - Bearing: orientáció
  - Tilt: döntés

```
val cameraPosition = CameraPosition.Builder()
        .target(LatLng(47.0, 19.0))
        .zoom(17f)
        .bearing(90f)
        .tilt(30f)
        .build()
myMap.animateCamera(CameraUpdateFactory.newCameraPosition(
  cameraPosition))
```

Geocoding

- GPS koordináta postacímből
- Internet engedély szükséges

```
val geocoder = Geocoder(this, Locale.ENGLISH)
val streetAddress = "Blaha Lujza tér 1, Budapest"
var locations: List<Address>? = null
geocoder.getFromLocationName(streetAddress, 3)
```

Reverse Geocoding

- Cím GPS koordinátából
- Internet engedély szükséges

```
val location = …
val latitude = location.getLatitude()
val longitude = location.getLongitude()
val gc = Geocoder(this, Locale.getDefault())
var addrs: List<Address>? =
            gc.getFromLocation(latitude, longitude, 3)
```

Összefoglalás

- Intent
- BroadcastReceiver
- Helymeghatározás, Fused location
- Geocoding/Reverse geocoding
- Térkép megjelenítés
- Markerek kezelése, overlayek
- Eseménykezelés térképen

A következő alkalommal...

- Android JetPack
- Architektúra komponensek
- LifeCycleObserver, LiveData, ViewModel, Paging, Data Binding
- Navigation, WorkManager
- Gyakran használt külső osztálykönyvtárak
- Legjobb gyakorlatok és elvek
- Összegzés

Köszönöm a figyelmet!

peter.ekler@aut.bme.hu

AutSoft