Android

Helyfüggő szolgáltatások, Térkép kezelés, HTTP kommunikáció

Dr. Ekler Péter
peter.ekler@aut.bme.hu

BME AUT
Department of Automation and Applied Informatics

Tartalom

- Helyfüggő szolgáltatások
  - Geocoding/reverse geocoding
  - ProximityAlert
  - Geofences
  - Activity recoginition
- Google Maps képességek
  - Térkép beállítások
  - Markerek kezelése
  - Maps Utility Library
- Hálózati kommunikáció alapok
  - HTTP kapcsolatok kezelése

Helyfüggő szolgáltatások

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)
```

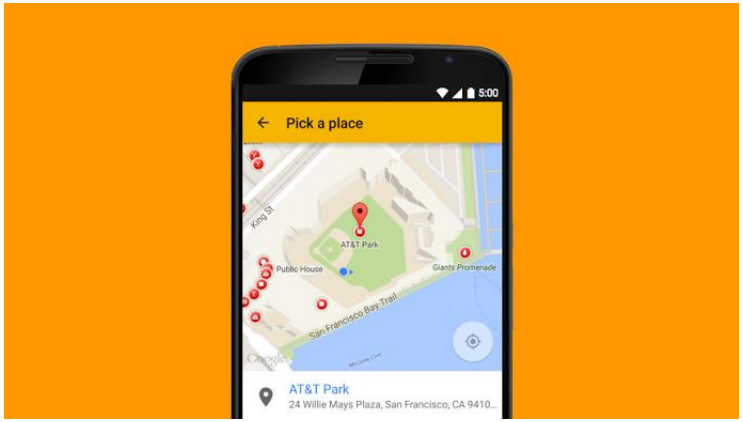
Reverse Geocoding

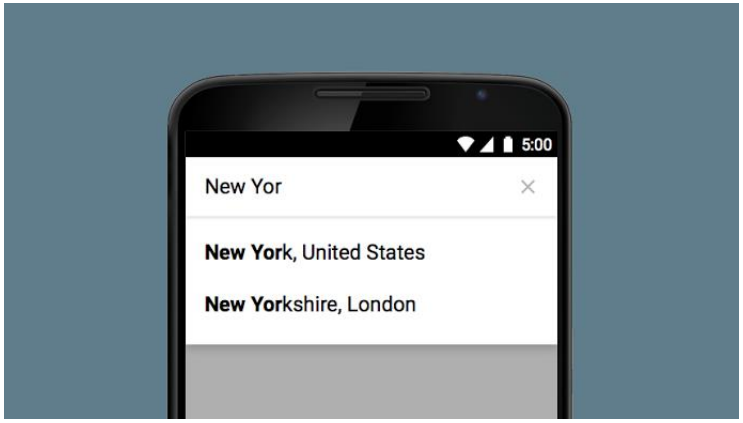
- Cím GPS koordinátából
- Internet engedély szükséges

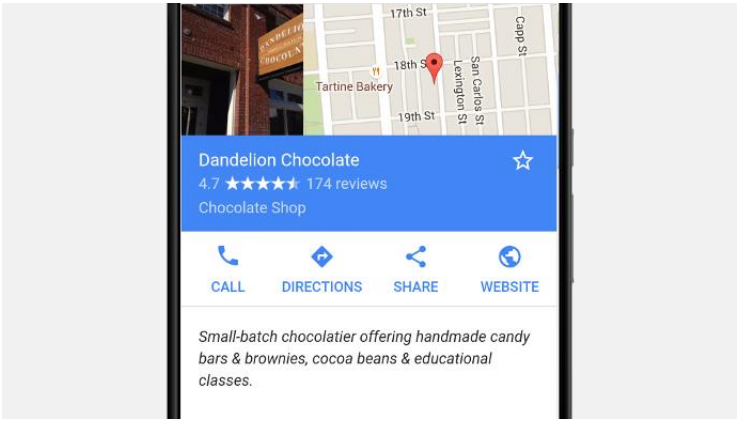
```
val location = …
val latitude = location.getLatitude()
val longitude = location.getLongitude()
val gc = Geocoder(this, Locale.getDefault())
var addrs: List<Address>? =
            gc.getFromLocation(latitude, longitude, 3)
```

Places API

- PlacePicker
  - Hely kiválasztó dialógus
- GeoDataApi
  - Google hely adatbázisához hozzáférés
- PlaceDetectionApi
  - Hozzáférés az az aktuális helyhez és jelentések készítése
- AutoComplete
  - Hely kiegészítő – beviteli mező
- További részletek:
  - https://developers.google.com/places/android-api/







ProximityAlert

- Értesítések egy adott környék megközelítésekor
  - Koordináta és sugár

```
val intent = Intent(ACTION_PROXIMITY_ALERT)

val pendInt: PendingIntent = PendingIntent.getBroadcast(this, requestcode,
intent, flags)

locationManager.addProximityAlert(lat, long, radius, timeout, pendInt)

...

class ProximityIntentReceiver: BroadcastReceiver() {

    @Override

    override fun onReceive(context: Context, intent: Intent) {

        val key = LocationManager.KEY_PROXIMITY_ENTERING

        val entering = intent.getBooleanExtra(key, false)

        ...

    }

}
```

További GEO API-k

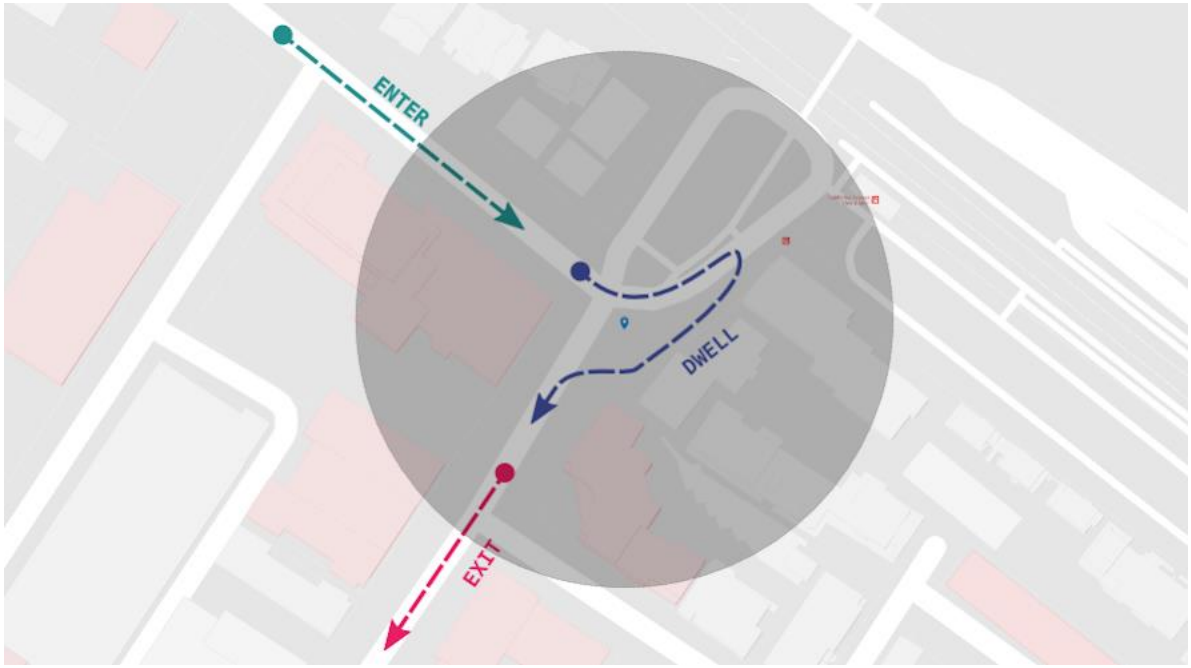
- Google Play Services resze
- Új API-k:
  - Fused location provider (erről már volt szó):
    - https://developer.android.com/training/location/retrieve-current.html
  - Geofencing API
  - Activity Recognition

Geofencing API

Mostoha Roland (mostoha.roland@autsoft.hu)

Geofencing API

- Földrajzi határok beállítása és feliratkozás belépés vagy elhagyás események bekövetkezésekor



Geofencing API

- Google Play Services része
- Egyszerű, de gazdag API
  - *Geofence* lista gyors és kötegelt megadása vagy eltávolítása
  - Több *Geofence* egyidejű kezelése
  - Riasztások szűrése
  - Hatékony helymeghatározás *Fused Location Provider* használatával
  - Alacsony energiafogyasztás: a helymeghatározási technológiák dinamikus használata a *Geofence* határától függően

Geofencing API

- Szükséges engedély (dangerous, el kell kérni):
  android.permission.ACCESS_FINE_LOCATION
- Szükséges függőség:
  com.google.android.gms:play-services-location:X
- API használata
  - Kliens inicializálása: *LocationServices.getGeofencingClient(this)*
  - *IntentService* létrehozása, amely feldolgozza és reagál az *Geofence API* eseményeire
  - *IntentService* és a számunkra érdekes események regisztrációja a kliensnél
  - *GeofenceTransition* alapján lekezelni az eseményeket
  - Szükség esetén kliens lecsatolása és *Geofence* lista törlése

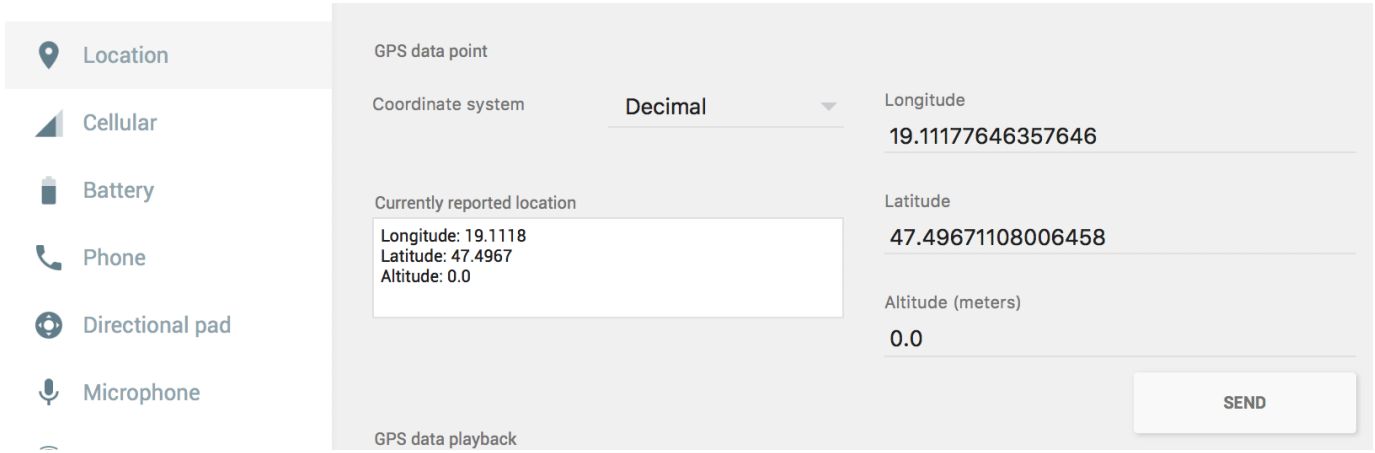
Geofence API demo alkalmazás

- A hős utcát megközelítve dobjunk egy figyelmeztető értesítést, elhagyva pedig egy megnyugtatót

47.49671, 19.11177

Tippek a teszteléshez

- Google Play Services legújabb verziója szükséges a teszteléshez
- Emulátoron teszteljünk, használjuk a Location tool-t a pozíciók beküldéséhez



Tippek a teszteléshez

- Ne használjunk túl pontos pozíciókat vagy túl kis távolságot
- *Settings- Location mode – High accuracy* módot használjunk, különben az API 1000-es error code-al tér vissza
- Wi-Fi legyen bekapcsolva
- Az esemény kiváltáshoz akár 2-3 perc szükséges lehet

Kotlin kiegészítések

Pair, Triple beépített osztályok Kotlinban:

```
// Hős street (latitude, longitude, circular distance in meters)
private val region = Triple(47.49671, 19.11177, 5000f)
```

by lazy delegate. Első híváskor inicializálódik, utána ugyanazt a példányt használja:

```
private val geofencePendingIntent: PendingIntent by lazy {
    val intent = Intent(this, GeofenceIntentService::class.java)
    PendingIntent.getService(this, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT)
}
```

Számok tagolása kódban. Növeli az olvashatóságot:

```
.setExpirationDuration(120_000L)
```

Kotlin kiegészítések

Java || helyett or, javítja az olvashatóságot

(Geofence.*GEOFENCE_TRANSITION_ENTER* or Geofence.*GEOFENCE_TRANSITION_EXIT*)

Függvényhívások ugyanazon az objektumon: apply

```
return GeofencingRequest.Builder().apply {
    setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER)
    addGeofences(geofenceList)
}.build()
```

Activity Recognition

Mostoha Roland (mostoha.roland@autsoft.hu)

Activity Recognition API

- Google Play Services része
- A felhasználók cselekvéseit detektálhatjuk és reagálhatunk rá
  - IN_VEHICLE
  - ON_BICYCLE
  - ON_FOOT
  - STILL
  - WALKING
  - RUNNING

Activity Recognition API

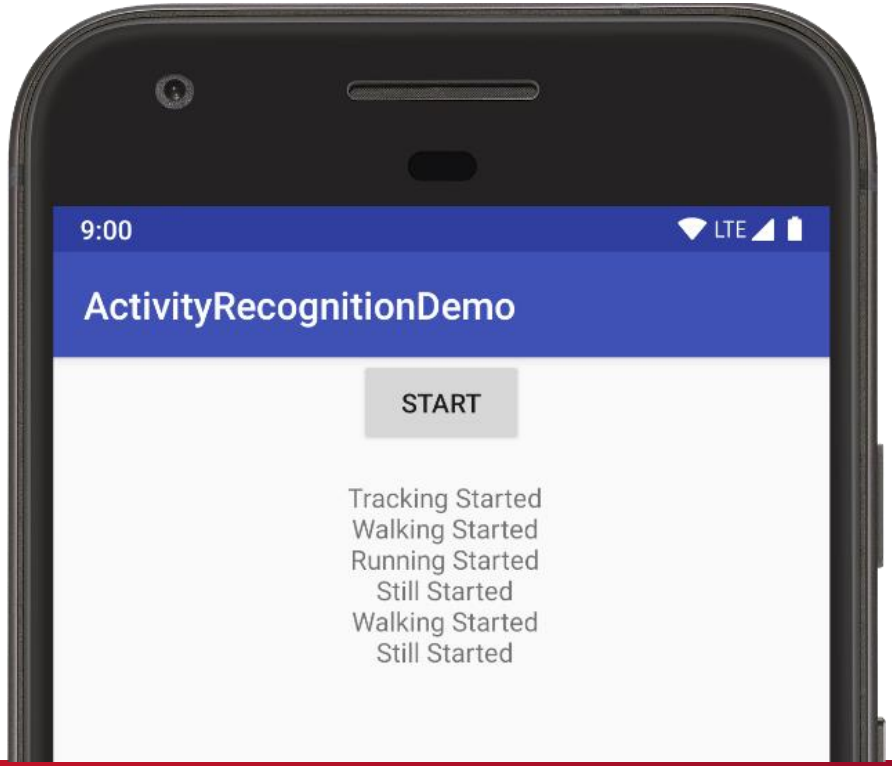
- Feliratkozhatunk a cselekvés megváltozását jelző ENTER és EXIT eseményekre
- Pl. Kalória számláló alkalmazás, amely automatikusan változtatja az elégetett kalória mennyiséget aszerint hogy gyaloglunk, futunk vagy állunk
- Pl. Egy chat alkalmazás, amely vezetés közben letiltja az értesítéseket

Activity Recognition API

- Szükséges engedély:
  com.google.android.gms.permission.ACTIVITY_RECOGNITION
- Szükséges függőség:
  com.google.android.gms:play-services-location:X
- API használata
  - Kliens inicializálása: *ActivityRecognition.getClient(this@MyActivity)*
  - *IntentService* létrehozása, amely feldolgozza és reagál az *ActivityRecognition* eseményeire
  - *IntentService* és a számunkra érdekes események regisztrációja a kliensnél
  - *ActivityType* és *TransitionType* alapján lekezelni az eseményeket
  - Kliens lecsatolása

Activity Recognition demo alkalmazás

- Activity Log gyűjtése az események alapján



Tippek a teszteléshez

- Valós eszközön teszteljünk (az emulátor szenzorai csak ritkán triggerelik a szolgáltatást)
- Tiltsuk le az elforgatást
- Az újabb verziókban már automatikusan történik a mintavételezés, nem tudjuk befolyásolni a gyakoriságot
- Legyünk türelmesek és kitartóak ☺

Kotlin kiegészítések

Előtte:

```
LocalBroadcastManager.getInstance(this).registerReceiver(
        object : BroadcastReceiver() {
            override fun onReceive(context: Context, intent: Intent) {
                ...
            }
        },
        statusIntentFilter)
```

Utána:

```
LocalBroadcastManager.getInstance(this).registerReceiver(statusIntentFilter) {
    ...
}
```

Térkép kezelése

Térkép nézet

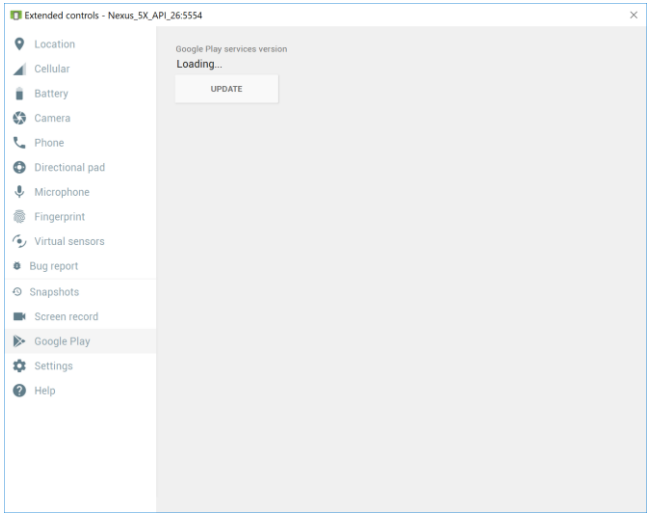
- Földrajzi pozíciók megjelenítése térkép-szerűen
- Teljes kontroll a megjelenítés felett
  - Helyszín, nagyítási szint
  - Térkép, műhold, traffic
- Ráhelyezhetőek overlay-ek
- Megjeleníthetőek rajta tetszőleges POIk

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

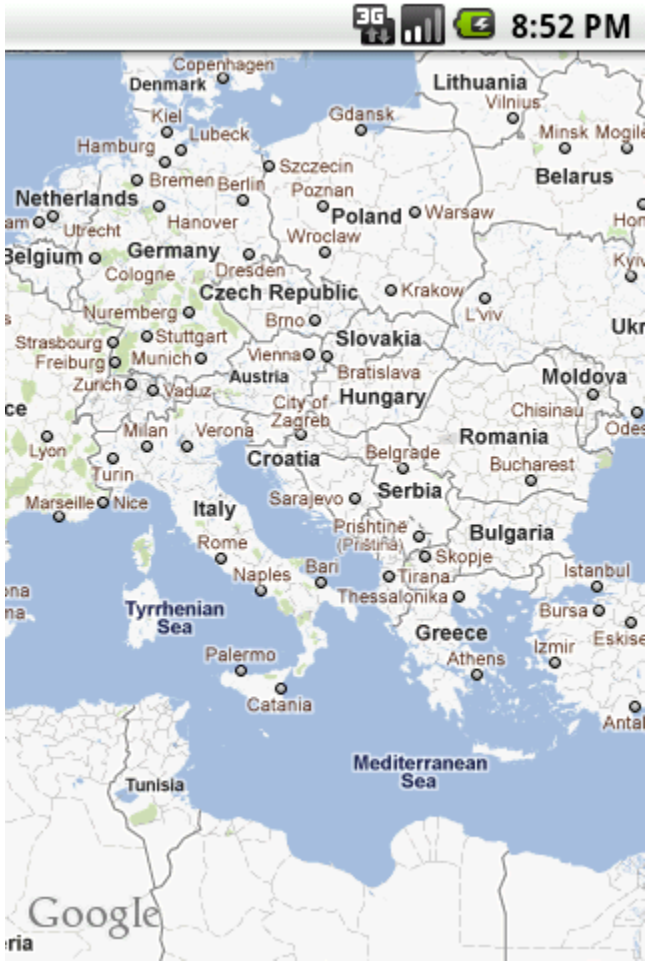
- MapFragment alapú térkép megjelenítés
- A térkép letöltése darabonként, on-demand történik, tehát Internet permission-re szükség van
- Manifest engedélyek beállítása
- OpenGL ES jelzése
- Projekt regisztrálása és API key igénylés
  - Google APIs console
  - https://code.google.com/apis/console/
- Ne 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

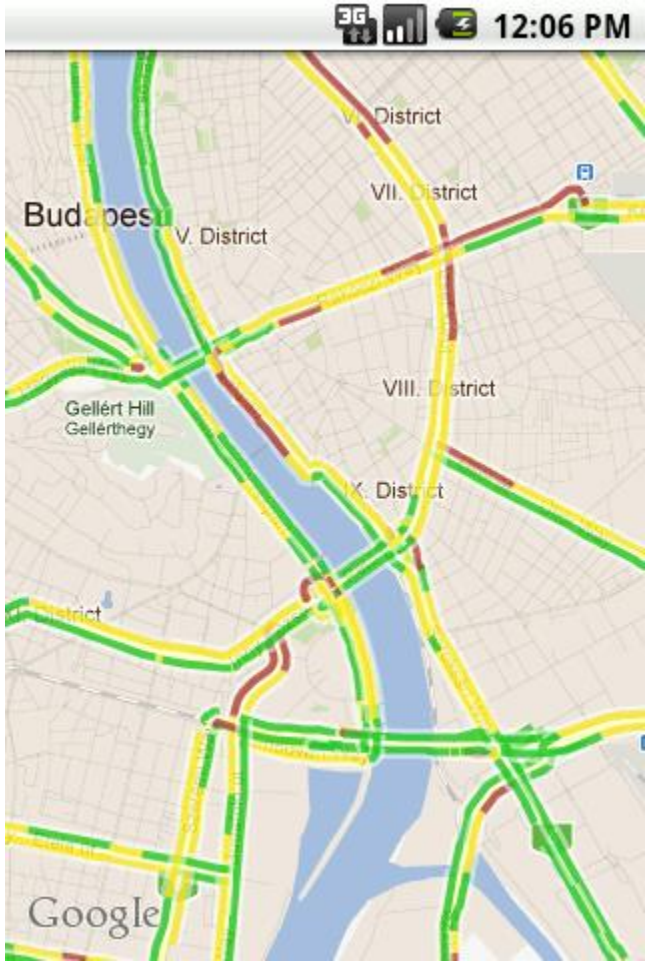
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_maps)
        val mapFragment = supportFragmentManager
                .findFragmentById(R.id.map) as SupportMapFragment
        mapFragment.getMapAsync(this)
    }

    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
```

Map esemény kezelés

```
mMap.setOnMapClickListener {
    val markerHU = mMap.addMarker(
        MarkerOptions()
            .position(it)
            .title("Hello")
            .snippet("Lakosság: 9.700.000"))
    markerHU.isDraggable = true

    mMap.animateCamera(CameraUpdateFactory.newLatLng(it))
}
```

Marker kezelés

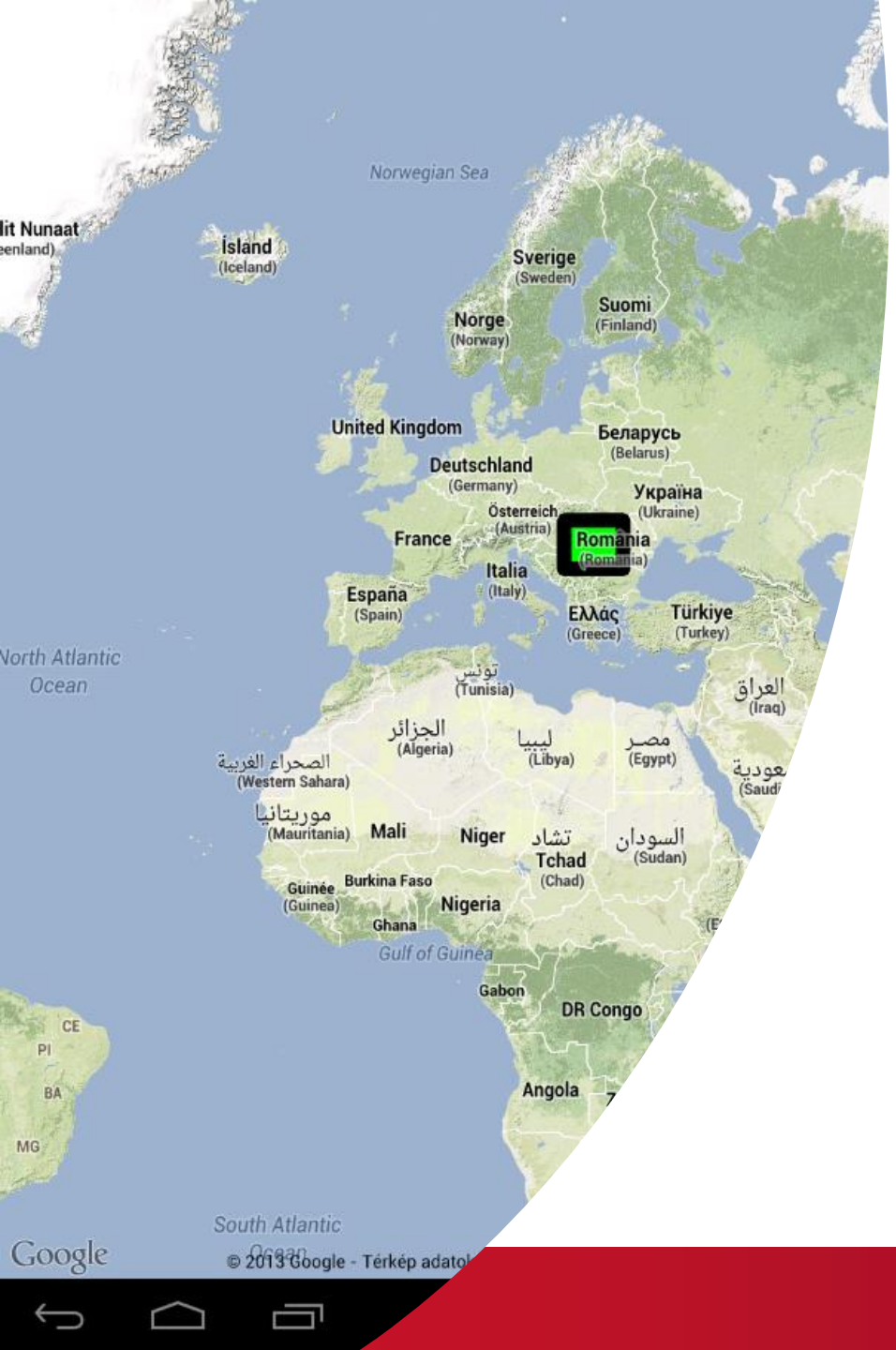
- `InfoWindow`:
  - Testreszabható `InfoWindow` felület
  - `InfoWindowAdapter`
  - Megjelenítés/eltüntetés progamozottan
  - Eseménykezelés: `OnInfoWindowClickListener`
- Marker eseménykezelők:
  - `OnMarkerClickListener`
  - `OnMarkerDragListener`
  - Stb.

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
```



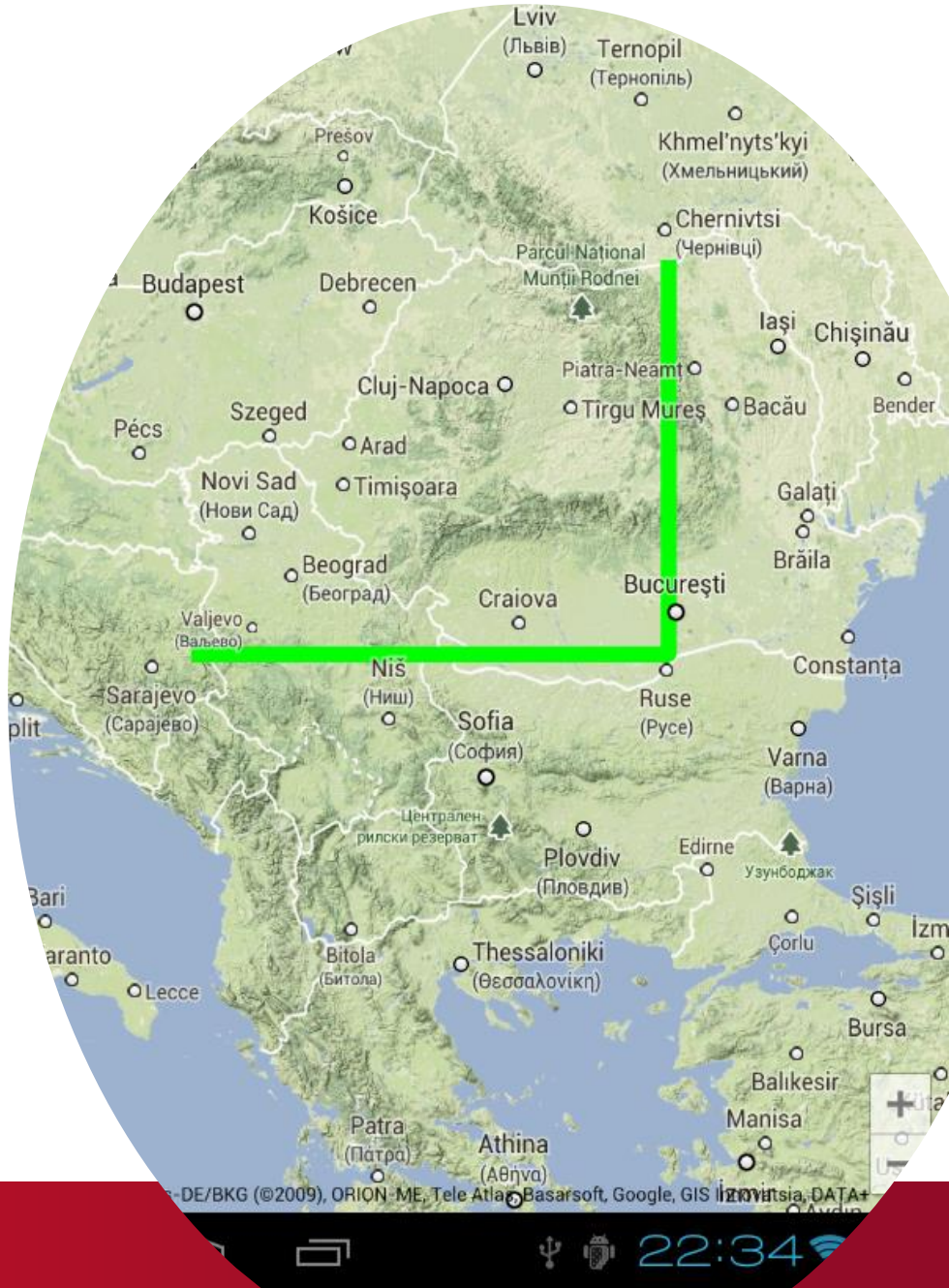
Demo – Téglalap rajzolás

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
```

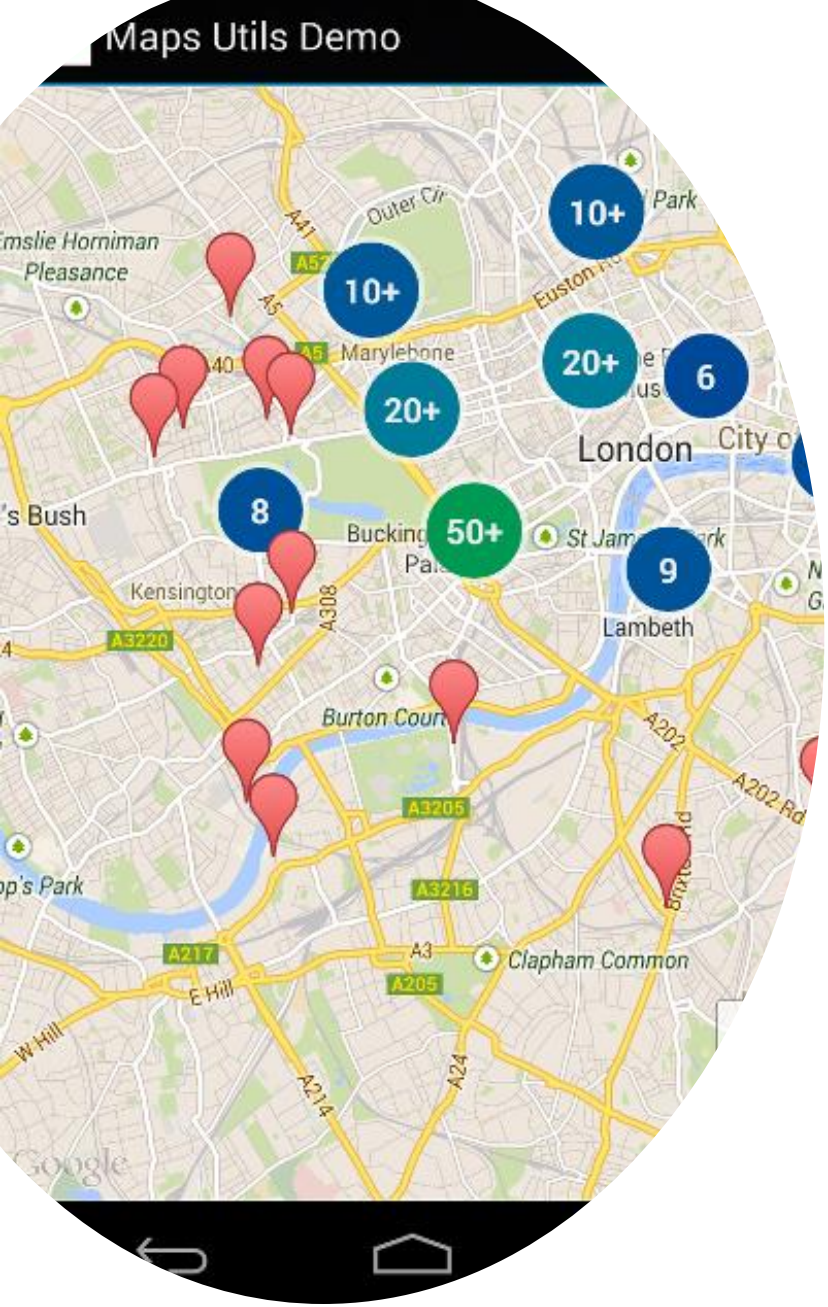
Demo – Vonal rajzolás



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))
```



https://developers.google.com/maps/documentation/android-api/utility/?hl=en

Map Utility Library

Hálózati kapcsolatok kezelése

Hálózati kapcsolatok

- Short-range
  - NFC
  - Bluetooth
  - Nearby API
  - Wifi-Direct
- Long-range/Internet
  - HTTP
  - TCP/IP-UDP

HTTP Kapcsolatok kezelése

HTTP kommunikáció Android platformon

- Egyik leggyakrabban használt kommunikációs technológia
- HTTP metódusok
  - GET, POST, PUT, DELETE
- Teljes körű HTTPS támogatás és certificate import lehetőség
- REST kommunikáció támogatása (Representational State Transfer)

HTTP kapcsolatok kezelése

- Szükséges engedély:
  <uses-permission android:name=
  "android.permission.INTERNET"/>
- Új szálban kell megvalósítani a hálózati kommunikáció hívást!
- Ellenőrizzük a HTTP válasz kódot:
  - http://www.restpatterns.org/HTTP_Status_Codes
  - http://www.w3.org/Protocols/HTTP/HTRESP.html
- HTTP REST
  - http://en.wikipedia.org/wiki/Representational_state_transfer
- Ügyeljünk az alapos hibakezelésre
- HTTP GET példa:
  - http://numbersapi.com/10/math

HTTP könyvtárak Android-on

- A rendszer két megvalósítás is tartalmaz:
  - Standard Java HTTP implementáció (***HttpURLConnection***)
  - **Apache** HTTP implementáció (*HttpClient*)
- Apache Depricated – Ne használjuk, ki is vették
- Igazán egyik sem jó
  - 3rd party megoldás – Square OkHttp
  - http://square.github.io/okhttp/

HTTP GET - HttpURLConnection

```
fun httpGet(urlAddr: String) {
    var reader: BufferedReader? = null
    try {
        val url = URL("http://mysrver.com/api/getitems")
        val conn = url.openConnection() as HttpURLConnection
        reader = BufferedReader(InputStreamReader(conn.inputStream))
        var line: String?
        do {
            line = reader.readLine()
            System.out.println(line)
        } while (line != null)
    } catch (e: IOException) {
        e.printStackTrace()
    } finally {
        if (reader != null) {
            try {
                reader.close()
            } catch (e: IOException) {
                e.printStackTrace()
            }
        }
    }
}
```

HTTP POST- HttpURLConnection

```
fun httpPost(urlAddr: String, content: ByteArray) {
    // ...
    var os: OutputStream? = null
    try {
        val url = URL("http://mysrver.com/api/refreshitems")
        val conn = url.openConnection() as HttpURLConnection
        conn.requestMethod = "POST"
        conn.doOutput = true
        conn.useCaches = false
        os = conn.outputStream
        os.write(content)
        os.flush()
        // …
    } catch (e: IOException) {
        e.printStackTrace()
    } finally {
        // …
        if (os != null) {
            try {
                os.close()
            } catch (e: IOException) {
                e.printStackTrace()
            }
        }
    }
}
```

Timeout értékek beállítása

- Fontos, hogy minden hálózati kommunikáció megfelelő módon kezelje a timeout-ot
- Timeout a kapcsolat megnyitásra
- Timeout az eredmény kiolvasására
- Példa:

```
…
val conn = url.openConnection() as HttpURLConnection
…
conn.setConnectTimeout(10000)
conn.setReadTimeout(10000)
…
```

Header paraméterek beállítása

- Egyszerű Header beállítása:

```
val conn = url.openConnection() as HttpURLConnection
conn.setRequestProperty(”[KEY]”,”[VALUE]”)
```

- Cookie beállítása:

```
val conn = url.openConnection() as HttpURLConnection
conn.setRequestProperty(”Cookie”,”sessionid=abc;age=15”)
```

- Összetett példa:

```
val conn = url.openConnection() as HttpURLConnection
conn.setRequestProperty(„Content-Type”,”application/json”)
conn.setRequestProperty(”Cookie”,”sessionid=abc;age=15”)
```

URL Encoding 1/2

- URL GET paraméterekben nem lehetnek „extra karakterek”
- Ilyen karakterek esetén URL encode-olásra van szükség
- Példa:
  - http://avalon.aut.bme.hu/~tyrael/phpget.php?name=John Doe
  - Vs.
  - http://avalon.aut.bme.hu/~tyrael/phpget.php?name=John%20Doe
- Szóköz esetén a kapott hiba:
  - 11-26 18:39:24.417: ERROR/AndroidRuntime(17232): java.lang.IllegalArgumentException: Illegal character in query at index 53: http://avalon.aut.bme.hu/~tyrael/phpget.php?name=John Doe
- Megoldás:
  - String name = URLEncoder.encode("John Doe");
  - connect("http://avalon.aut.bme.hu/~tyrael/phpget.php?name="+name)

URL Encoding 2/2

- *URLEncoder*:
  - Az ('a'..'z', 'A'..'Z') –n, számokon ('0'..'9') és '.', '-', '*', '_' karaktereken kívül minden karakter a hexadecimális megfelelőjévé kerül konvertálásra, például: '#' -> %23
  - `encode(String s)`
  - `encode(String s, String charsetName)`
- *URLDecoder*
  - *URLEncoder* fordítottja, az *application/x-www-form-urlencoded* MIME típusú szövegeket tudja visszalakítani
  - `decode(String s)`
  - `decode(String s, String encoding)`

Aszinkron kommunikáció

UI módosítása más szálból

- Az alkalmazás indításakor a rendszer létrehoz egy úgynevezett *main* szálat (UI szál)
- Sokáig tartó műveletek blokkolhatják a felhasználói felületet, ezért új szálba kell indítani őket
- Az ilyen műveletek a végén az eredményt a UI-on jelenítik meg, azonban az Android a UI-t csak a fő szálból engedi módosítani!
- Több megoldás is szóba jöhet:
  - *Activity.runOnUiThread(Runnable)*
  - *View.post(Runnable)*
  - *View.postDelayed(Runnable, long)*
  - *Handler*
  - *AsyncTask és LocalBroadcast (példa: laboron szerepelni fog)*
  - *Külső libek, pl. EventBus, Otto*
  - *REST külső lib: RetroFit*

AsyncTask példa

```
class AsyncTaskUploadVote : AsyncTask<String, Void, String>() {

    override fun onPreExecute() {
        // ...
    }

    override fun doInBackground(vararg params: String): String? {
        val result = null
        // Hálózati kommunikáció, válasz mentése result-ba
        return result
    }

    override fun onPostExecute(result: String?) {
        // Eredmény használata UI szálon
        Log.d("RESULT", result)
    }

}

AsyncTaskUploadVote("Yes").execute()
```

Tipikus adatformátumok

Adatok küldése, válaszok feldolgozása

- Sokszor egy előre definiált formátumban/protokollon történik a kommunikáció kliens és szerver között
- Legtöbb esetben egy harmadik fél szerverétől kapott válasz is valamilyen jól strukturált formátumban érkezik
- Tipikus formátumok:
  - CSV (Comma Separated Value(s))
  - JSON (JavaScript Object Notation)
  - XML (Extensible Markup Language)
- Természetesen lehet saját protokoll is

JSON formátum

- Szintaktikai elemek: ’{’, ’]’ ,’[’, ’]’ ,’:’ ,’;’
- Példa:

```
{
   "keresztnev" : "Elek",
   "vezeteknev" : "Teszt",
   "kor" : 23,
   "cim" :
   {
      "utca" : "Baross tér",
      "varos" : "Budapest",
      "iranyitoszam" : "1087"
   },
   "telefon":
   [
      {
        "tipus"  : "otthoni",
        "szam": "123 322 1234"
      },
      {
        "tipus"  : "mobil",
        "szam": "626 515 1567"
      }
   ]
}
```

JSON feldolgozás

- *JSONObject*:
  - JSON objektumok parse-olása
  - Elemek elérhetősége a kulcs megadásával:
    - `getString(String name)`
    - `getJSONObject(String name)`
    - `getJSONArray(String name)`
  - JSON objektum létrehozása *String*-ből vagy *Map*-ból
- *JSONArray*:
  - *JSONObject*-hez hasonló működés JSON tömbökkel
  - Parse-olás, elemek lekérdezése index alapján, hossz
  - Létrehozás például *Collection*-ból

JSON API minták

- *Currency Exchange:*
  - *https://api.exchangeratesapi.io/latest?base=HUF*
- OpenWeather
  - http://api.openweathermap.org/data/2.5/weather?q=Budapest,hu&units=metric&appid=f3d694bc3e1d44c1ed5a97bd1120e8fe
- TV Show Data API:
  - http://api.tvmaze.com/search/shows?q=stargate

REST API gyűjtemények

- https://github.com/toddmotto/public-apis
- https://github.com/abhishekbanthia/Public-APIs
- https://github.com/Kikobeats/awesome-api

XML formátum

```
<?xml version="1.0"?>
<employees>
  <person>
    <name>Big Joe</name>
    <address>Beach Street 12.</address>
    <phone>111-222</phone>
  </person>
  <person>
    <name>Small Joe</name>
    <address>Hill Street 13.</address>
    <phone>222-333</phone>
  </person>
</employees>
```

XML feldolgozás

- Az Android gazdag eszközkészletet biztosít XML-ek feldolgozására
- SAX alapú feldolgozás
  - *javax.xml.parsers.SAXParser*
  - Különféle függvényekkel dolgozhatjuk fel az értelmező által generált eseményeket
  - Az eseményeket akkor generálja az értelmező, amikor a jelölő nyelv meghatározott részeihez ér
- DOM alapú feldolgozás
  - *javax.xml.parsers.DocumentBuilder*
  - *javax.xml.parsers.DocumentBuilderFactory*
  - Memóriába kerül beolvasásra az XML mint egy „fa”
  - Lekérdezhetők az elemek

Külső osztálykönyvtárak XML és JSON feldolgozásra

- XML:
  - SimpleXML
- JSON:
  - GSON
- REST API tesztelésére:
  - Postman Chrome Client

GSON POJO példa (Kotlin)

```
class PhoneInfo(
        @SerializedName("DeviceID")
        val deviceId: String,

        @SerializedName("OperatingSystem")
        val operatingSystem: String
)
```

GSON POJO példa (Java)

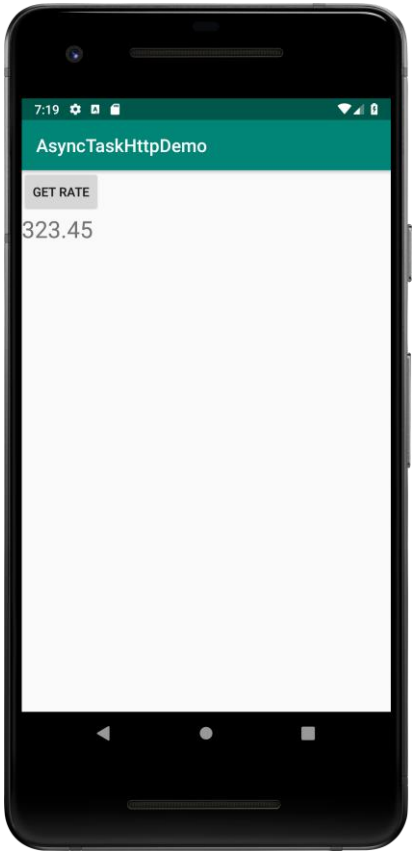
```
public class PhoneInfo {
  @SerializedName("DeviceID")
  private String deviceId;
  @SerializedName("OperatingSystem")
  private String operatingSystem;

  public PhoneInfo(String deviceId, String operatingSystem) {
    this.deviceId = deviceId;
    this.operatingSystem = operatingSystem;
  }
  public String getDeviceId() {
    return deviceId;
  }
  public String getOperatingSystem() {
    return operatingSystem;
  }
}
```

REST API-k kezelése

Példa

- https://api.exchangeratesapi.io/latest?base=EUR
- HttpURLConnection + AsyncTask



AsyncTask saját Callback-el

AsyncTask konstruktor paraméterében vár *Callback*-et:

```
class HttpGetTaskWithCalback(val callback: (String) -> Unit) :
  AsyncTask<String, Void, String>() {

    override fun doInBackground(vararg params: String): String {

        // … http lekérdezés és eredmény betöltése result String-be
        return result
    }

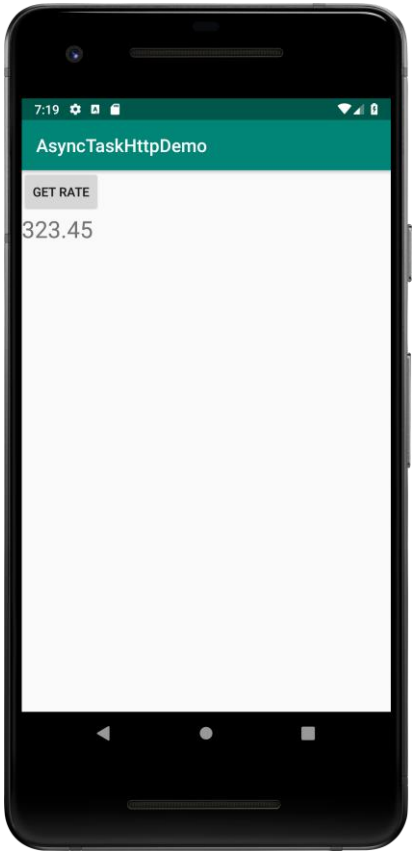
    override fun onPostExecute(result: String) {
        callback.invoke(result)
    }
}
```

Használat például Activity-ben:

```
btnGetRate.setOnClickListener{
    HttpGetTaskWithCalback { result ->
        Toast.makeText(this@ManActivity, result, Toast.LENGTH_LONG).show()
    }
}
```

Példa

- https://api.exchangeratesapi.io/latest?base=EUR
- OkHttp – Get példa



HTTP GET - OkHttp

```
object OkHttpHelper{
    fun getRates():String{
        val client=OkHttpClient.Builder()
            .connectTimeout(5000,TimeUnit.MILLISECONDS)
            .build()

        val request=Request.Builder()
            .url(
"https://api.exchangeratesapi.io/latest?base=EUR")
            .get()
            .build()

        val call=client.newCall(request)
        val response=call.execute()

        val responseStr=response.body()!!.string()
        return  responseStr
    }
}
```

Használat például Activity-ben:

```
Thread {
    var data = OkHttpHelper.getRates()
    runOnUiThread{
        Toast.makeText(this@MainActivity, data, Toast.LENGTH_LONG).show()
    }
}.start()
```

Retrofit

- HTTP API megjelenítése Java interface formában

```
interface ItemsService {
    @GET("/items/{item}/details")
    fun listItems(@Path("item") item: String): Call<List<Item>>
}
```

- Retrofit osztály a konkrét implementáció generálására

```
val retrofit = Retrofit.Builder()
        .baseUrl("https://api.myshop.com")
        .build()
val service = retrofit.create(ItemsService::class.java)
```

- Mindenhívás az *ItemsService* mehet szinkron és aszinkron módon:

```
val items: Call<List<Item>> = service.listItems("myItem")
```

Retrofit

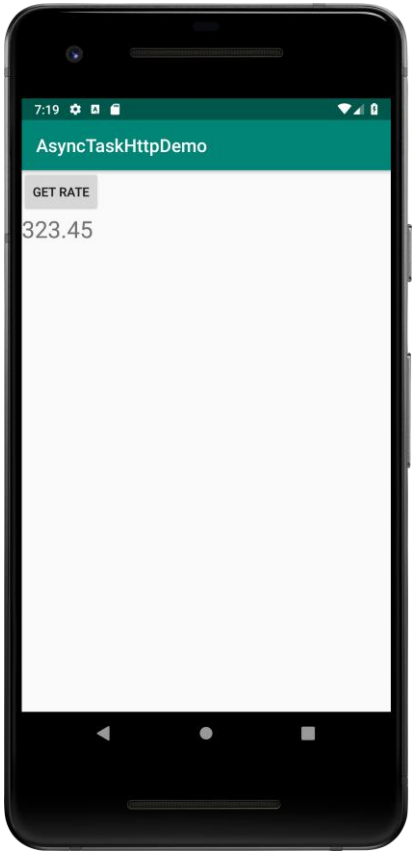
- HTTP kérések leírása annotációkkal:
  - URL és query paraméterek
  - Body – objektum konverzió (JSON, protocol buffers)
  - Multipart request és file feltöltés
- Gradle:
  - implementation 'com.squareup.retrofit2:retrofit:2.4.0'
- További információk:
  - http://square.github.io/retrofit/

Retrofit – GSON támogatás

- Automatikus konverzió a háttérben
  - Be kell állítani a Retrofitnak hogy mit használjon a konverzióhoz.
  - ```
    val retrofit=Retrofit.Builder()
            .baseUrl("http://api.myserver.com/")
            .addConverterFactory(GsonConverterFactory.create())
            .build()
    ```
- Gradle:
  - implementation 'com.google.code.gson:gson:2.8.5'
    implementation 'com.squareup.retrofit2:converter-gson:2.4.0'
- További információk:
  - http://square.github.io/retrofit/
    ```

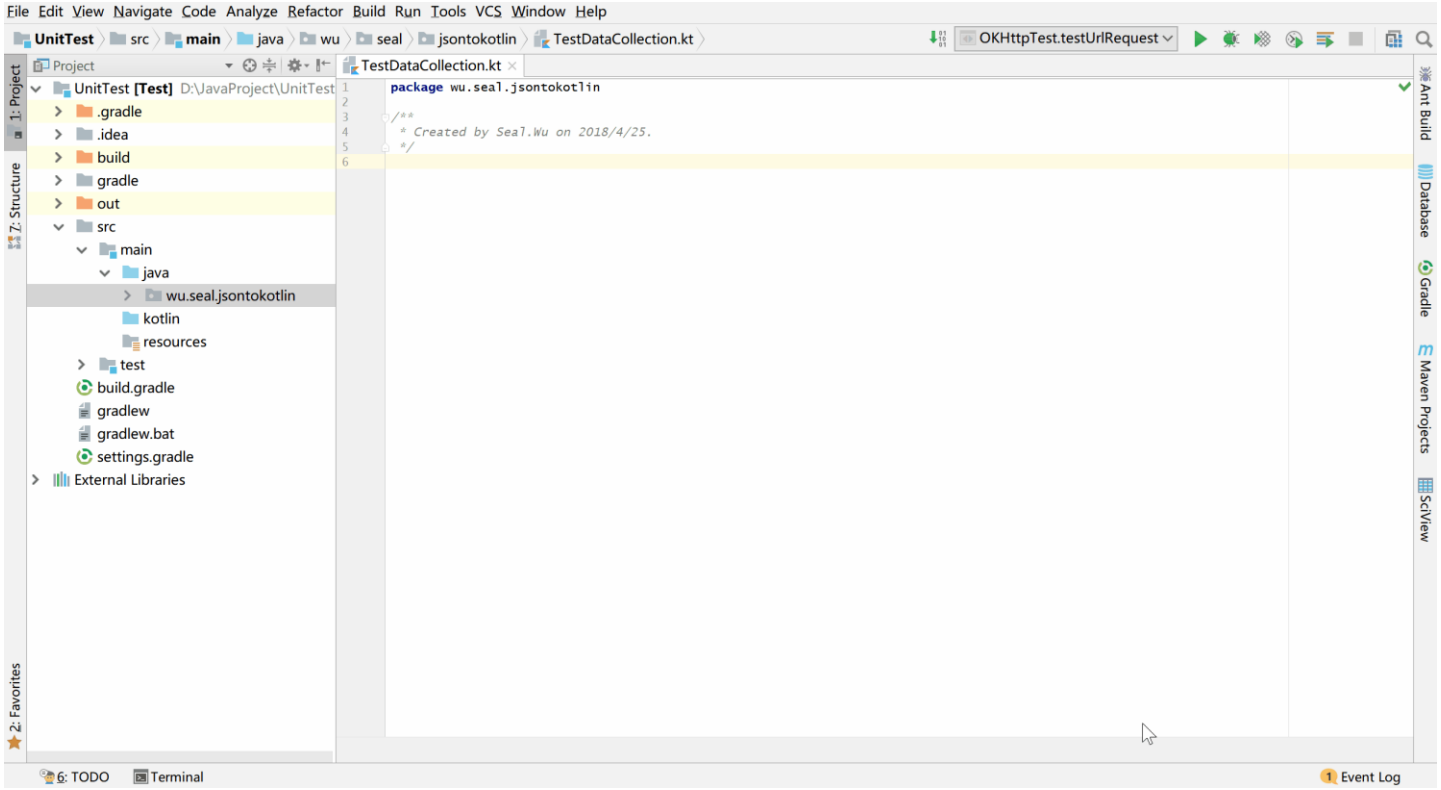
Példa - Retrofit

- https://api.exchangeratesapi.io/latest?base=EUR
- Retrofit 2 + GSON



Entitás vagy *data* class generálás JSON-ból

- *data* class, csak ha kellenek a *componentN* és egyéb függvények
- https://http4k-data-class-gen.herokuapp.com/
- https://github.com/wuseal/JsonToKotlinClass



Retrofit - Entitások / data class

```
data class MoneyResult(
    var date: String,
    var rates: Rates,
    var base: String
)

data class Rates(
    var BGN: Double,
    var CAD: Double,
…
}
```

Retrofit – API interface

```
import retrofit2.Call
import retrofit2.Callback
import retrofit2.http.GET
import retrofit2.http.Query


interface CurrencyExchangeAPI {
    @GET("/latest")
    fun getRates(@Query("base") base: String): Call<MoneyResult>
}
```

Retrofit - használat

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    val retrofit = Retrofit.Builder()
        .baseUrl("https://api.exchangeratesapi.io/")
        .addConverterFactory(GsonConverterFactory.create())
        .build()

    val currencyAPI = retrofit.create(CurrencyExchangeAPI::class.java)

    btnGetRate.setOnClickListener {
        val ratesCall = currencyAPI.getRates("EUR")
        ratesCall.enqueue(object: Callback<MoneyResult> {
            override fun onFailure(call: Call<MoneyResult>, t: Throwable) {
                tvResult.text = t.message
            }

            override fun onResponse(call: Call<MoneyResult>,
              response: Response<MoneyResult>) {
                tvResult.text = response.body()?.rates?.HUF.toString()
            }
        })
    }
}
```

Összefoglalás

- Helyfüggő szolgáltatások
  - Geocoding/reverse geocoding
  - ProximityAlert
  - Geofences
  - Activity recoginition
- Google Maps képességek
  - Térkép beállítások
  - Markerek kezelése
  - Maps Utility Library
- Hálózati kommunikáció alapok
  - HTTP kapcsolatok kezelése

Kérdések