EÖTVÖS LORÁND TUDOMÁNYEGYETEM
TERMÉSZETTUDOMÁNYI KAR

Kőbánya térképe, kutyatartók számára

SZAKDOLGOZAT
FÖLDTUDOMÁNYI ALAPSZAK

Készítette:
Rigó Dániel Imre
térképész és geoinformatikus szakirányú hallgató

Témavezető:
Dr. Kovács Béla
adjunktus
ELTE Térképtudományi és Geoinformatikai Tanszék

Budapest, 2016

Tartalom

1.BEVEZETÉS, CÉLOK

Akár szülő, akinek a gyermeke szeretne kutyát tartani, akár egy tapasztaltabb kutyatartó, aki új helyre költözött, mindegyikükben felmerül néhány alapvető kérdés. Hova lehet kivinni a kutyát futtatni? Hol lehet a legközelebb eledelt és egyéb felszerelést venni? Hol van a közelben állatorvos? Ezen kérdések a legalapvetőbb problémákra keresik a választ, ugyanis minden kutyának szüksége van testmozgásra az egészségük érdekében, ha pedig mégis bekövetkezik a baj és megbetegszik vagy baleset történik, jobb mihamarabb állatorvoshoz fordulni. Ha pedig eledelt kell neki venni vagy bármilyen felszerelést, ami nem várathat, akkor jó tudni, hogy hol tud a gazda a leghamarabb hozzájutni.

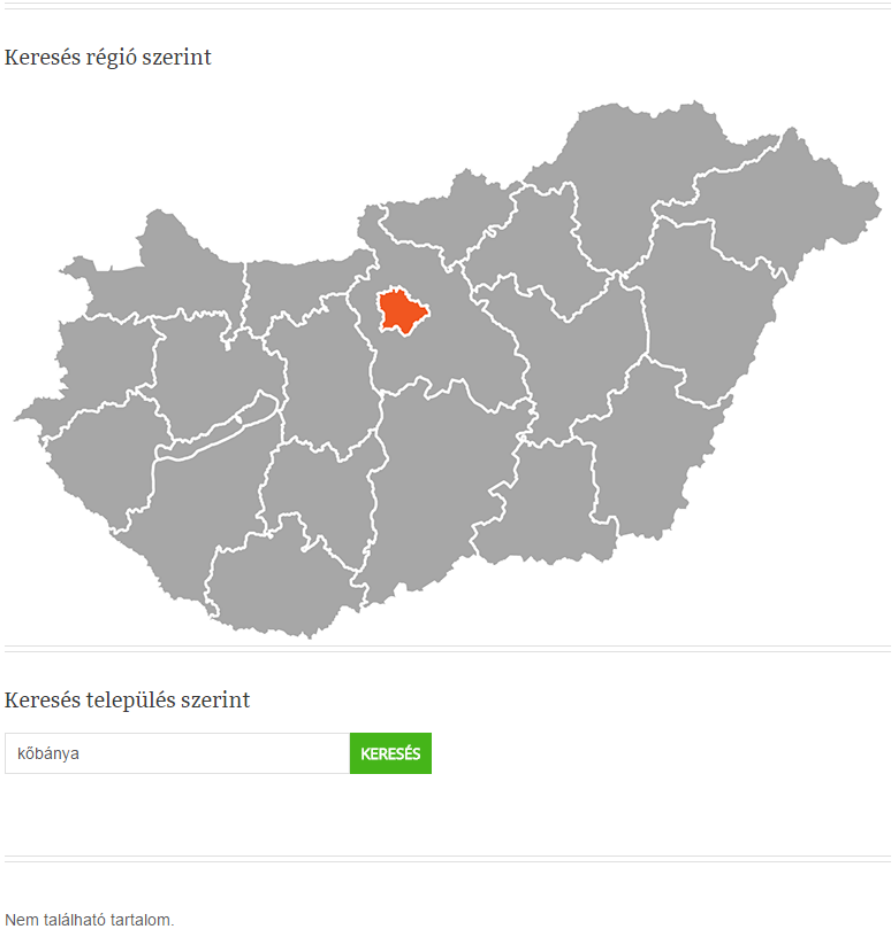
Az általam felmért terület Budapest X. Kerülete, Kőbánya. Itt elég sok lakótelep, illetve egyéb emeletes ház van, ahol a kutyatartóknak fontos ismerniük a környéken lévő kutyabarát létesítményeket. A kutyák jó kedvének és egészségének fenntartásán túl, a kerületben 2016. január 1-étől új önkormányzati rendelet lépett életbe, mely szerint a parkokban is csak az arra kijelölt helyeken lehet elengedni őket.

Diplomamunkám célja, hogy ezekben és egyéb kérdésekben próbáljon választ adni a helyi kutya tartóknak, egy térképes alkalmazás segítségével. Az alkalmazást az emberek által manapság legtöbbet használt eszközökre, az okos telefonokra készítettem. Ezeken belül is a legelterjedtebb operációs rendszerre, az Androidra. Majdnem minden embernek ott van a zsebébe, így bármikor, ha szüksége van rá, tudja is használni.

1.1.KITEKINTÉS

Mielőtt elkezdtem a diplomamunkámat elkészíteni, körbenéztem az interneten olyan weboldalak után, amelyek kutyafuttatók és egyéb kutyabarát helyek keresésére szolgálnak. Találtam is néhányat, de ezen oldalak túlnyomóan nem jelenítenek meg Kőbányán ilyen helyeket.

Ezek közül az egyik ilyen oldal a www.kutyabarat.hu, melynek kutyafuttató keresőjéről lentebb látható a kép, melyen látható, hogy Kőbánya címszóra nem található tartalom.



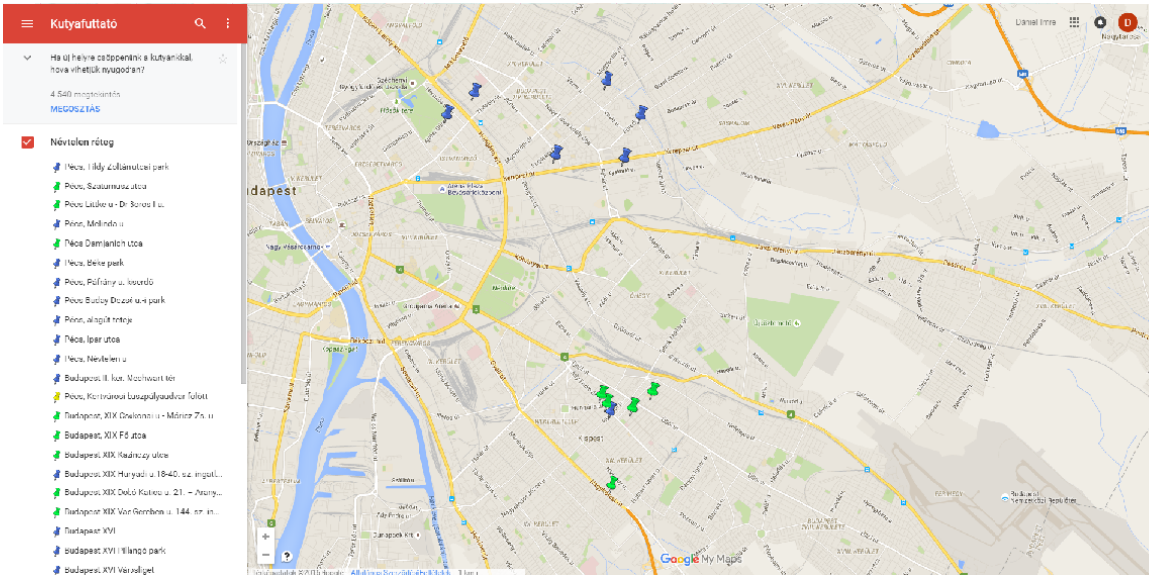
1. ábra kutyabarat.hu keresési eredmény

Egy másik oldal pedig Google Maps-en jeleníti meg a különböző kutyabarát helyeket, rajzszögekkel. De itt is jól látszik a képen, hogy Kőbányán 1 db jel sincs, csak némely szomszédos kerületben. Ez a: https://www.google.com/maps/d/viewer?hl=hu&mid=1YfYUghcZrdyLPeJJ3AywofcG1x8 címen érhető el.

A harmadik oldal amit találtam ebben a témában, a www.kozelben.hu. Ezen az oldalon két darab kutyafuttatót találtam megjelölve a X. kerületben.

A negyedik, amit találtam a www.kutyabarathelyek.hu, bár elég sok találatot jelenített meg, természetesen ez se az összeset, amit én a későbbiekben felmértem. De ez az oldal a

leghasznosabb, amit találtam, habár így is, ha nem csak kutyafuttatót keresek, más oldalakat is meg kell látogatnom, míg az általam készített alkalmazásban minden mást is meg lehet találni.



2. ábra A második weboldal keresési eredménye

2.FELMÉRÉS

A felméréseket kerékpáron vagy gyalog végeztem, Garmin eTrex Legend CX és Garmin GPSmap 62 műszerekkel.

Felmérés közben a műszer által számolt hibahatár, maximum 10 m volt. Mikor ennél az értéknél magasabb volt más irányokból végeztem el a műveletet.

A kerületben minden lakott területen bejártam az utcákat és összegyűjtöttem az összes objektumot, melyre egy kutya gazdájának szüksége van.

Ezen terepi munka során, felmérési jegyzőkönyvet készítettem.

2.1.MŰSZEREK

Garmin eTrex Legend CX	
Méret	2.2" x 4.2" x 1.2" (5.6 x 10.7 x 3.0 cm)
Kijelző méret	1.3" x 1.7" (3.3 x 4.3 cm)
Kijelző felbontás	176 x 220 pixel
Kijelző típus	256 színes TFT
Elem	2 db AA
Elem élettartam	32 óra
Vízállóság	IPX7
Csatlakozás	USB
Letölthető térkép	van
Egyéb térkép	van
Bővíthetőség	64 MB microSD-vel
Útpontok száma	500
Track log	10 000 pont, 20 mentés
Egyéb	Geocaching, Nap/Hold információ, stb.

3. ábra eTrex Legend CX (forrás: https://buy.garmin.com)

Garmin GPS Map 62

Méret	2.4" x 6.3" x 1.4" (6.1 x 16.0 x 3.6 cm)
Kijelző méret	1.43" x 2.15" (3.6 x 5.5 cm)
Kijelző felbontás	160 x 240 pixel
Kijelző típus	transzflektív, 65ezer színű TFT
Elem	2 db AA
Elem élettartam	20 óra
Vízállóság	IPX7
Csatlakozás	USB és NMEA 0183
Alaptérkép	van
Letölthető térkép	van
Memória	1,7 GB (nem bővíthető)
Útpontok száma	2 000
Track log	10 000 pont, 200 mentés
Egyéb	Geocaching, Nap/Hold információ, stb.

(forrás: https://buy.garmin.com)

4. ábra GPS Map 62

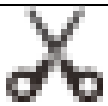
3.FELDOLGOZÁS

Az adatok feldolgozását a GPS Track Maker szoftverrel készítettem. A felmért adatokat a jegyzőkönyv segítségével átszerkesztettem, átláthatóbbá és kezelhetőbbé alakítva. A felmért objektumokat kategorizáltam :

- *Állatorvos*
- *Kutya felszereléseket és eledelt forgalmazó bolt*
- *Kutyakozmetika*
- *Kutyaürülék gyűjtő szemetes*
- *Kutyaiskola*
- *Kutyafuttató*
    - *Kutya játékok / agility akadályok*

Ezen kategorizálás alapján megszerkesztettem GPS Track Maker-ben a végső fájlt, melynek GPX a fájlformátuma. Ezt a fájlt olvassa be és jeleníti meg az alkalmazásom.

A jeleket a http://softicons.com oldalról töltöttem le.

	Állatorvos
	Bolt
	Kutyakozmetika
	Kutyaiskola
	Kutyaürülék gyűjtő
	Kutyafuttató

3.1.GPS TRACK MAKER

A GPS Track Maker egy ingyenesen használható GIS (Geographical Information System) szoftver, melyet a brazíliai Geo Studio Technology fejleszt. GPS adatok feldolgozására hozták létre. Képes adatokat kezelni valamennyi GPS műszerről, akár valós időben is.

Az adatokat nem csak megjelenteni képes, de segítségével bizonyos mértékig szerkeszteni is lehet azokat. A szerkesztést a rajzoló eszközökkel, illetve a kijelölő és cserélő eszközökkel könnyű megvalósítani.

4.AZ ANDROID OPERÁCIÓS RENDSZERRŐL

5. ábra Android

(forrás: https://commons.wikimedia.org/wiki/File:Android_robot.svg)

Az Android operációs rendszer fejlesztését az Android Inc. kezdte el, majd 2005-ben a Google Inc. felvásárolta, ezután pedig az Open Handset Alliance nevű konzorcium folytatta. Fejlesztésének célja egy Linux alapú operációs rendszer elkészítése volt, mely problémák nélkül képes kezelni a mobiltelefonok integrált hardvereit (érintőképernyő, Bluetooth, stb.).

Eleinte nem volt szó a Java nyelvről csak miután a Google felvásárolta a fejlesztő céget. Ekkor került a Linux kernel fölé, a Java nyelvet támogató, úgynevezett Dalvik virtuális gép. Ez a virtuális gép felelős a felhasználói felület kezeléséért és az alkalmazások futtatásáért. A Dalvik virtuális gép fő jellemzői:

- Nem kompatibilis a korábbi Sun virtuális géppel
- Megújult utasításkészlettel dolgozik
- A Java-programok nem egy-egy .class állományba kerülnek fordítás után, hanem egy .dex kiterjesztésű, úgynevezett Dalvik Executable formátumba
- A Java csak, mint nyelv jelenik meg

Ezen a virtuális épen egy Javában írt, de felügyelt kód (managed code) fut. Ennek köszönhetően az Android alkalmazások biztonságosabban futnak le, mert így nem vagy nagyon ritkán tudja egy-egy alkalmazás megbénítani a teljes rendszert.

2008. szeptember 23-ától, az első kereskedelmi forgalomba kiadott verziótól, úgynevezett API (Application Programming Interface) szinteket különböztetünk meg. A legelső az Android 1.0 Alpha volt, melynek 1-es az API szintje. A legújabb pedig 2015. szeptemberétől az API 23. melynek az év decemberében kiadták a 6.0.1.-es verzióját. Ezek

az API szintek felfelé kompatibilisek egymással, aminek köszönhetően egy API 1-esre írt alkalmazás futtatható akár API 23-on is. Visszafelé ez nem működik, ugyanis minden újabb verzióba raknak újításokat, amiket a régebbi verziók nem ismernek. Én a 4.0.3.-as verzióra fejlesztettem (API 15), mely jelenleg a legelterjedtebb az Androidot futtató eszközökön.

4.1.AZ ALKALMAZÁSOK FELÉPÍTÉSE

Az Android alkalmazások több különböző típusú komponensből állhatnak, melyek közül legalább egynek lennie kell a programban. Az egyes komponens típusok különböző célt szolgálnak. Legfontosabb jellemzőjük, hogy egyedi felületet biztosítanak, melyen keresztül a rendszer számára elérhetőek. Fontos megjegyezni, hogy nem minden komponens érhető el a felhasználók számára. Az Android négy fő alkalmazáskomponens-típust támogat, ezek:

- Activity
- Service
- ContentProvider
- BroadcastReciever

Egy alkalmazás, ugyanabból az alkalmazáskomponensből többet is tartalmazhat.

4.1.1.ACTIVITY

Az Activity a leggyakrabban használt alkalmazáskomponens. Saját felülettel rendelkezik, melyet gyakorlatilag egy ablaknak nevezhetnénk. Egy alkalmazás állhat több Activity-ből, melyek különböző, jól elkülönülő célokat szolgálnak, de együtt adják az alkalmazás teljes funkcionalitását.

Az Android lehetőséget biztosít, hogy más alkalmazásokból hívjunk meg Activityt, így a platform kellően rugalmas és átjárható rendszert biztosít. Erre példa az én

alkalmazásom is, ahol a Google Maps alkalmazás Activity-jét hívtam meg és azon helyeztem el a saját felmérésem eredményét.

4.1.2.SERVICE

A Service komponens, a hosszabb ideig, a háttérben futó feladatokat jelenti. Legfőbb jellemzője, hogy nincs saját felhasználói felülete, viszont indíthat Activity-ket vagy egyéb felugró ablakokat (pl. Toast). Az operációs rend szer alapértelmezetten is futtat különböző Service-eket, melyek szükségesek a működéshez. Az éppen futó szolgáltatásokat meg is tekinthetjük a rendszer beállításainál. Tipikus funkciók, amiket szolgáltatásokkal szoktak megoldani:

- Háttértartalom szinkronizálás
- Pozíció / használat nyomon követése
- Zenelejátszás
- Letöltések

4.1.3.CONTENTPROVIDER

A ContentProvider egy tartalomszolgáltató komponens, melynek feladata egy adatforrás kezelése és az adatforrásra vonatkozó kérések kiszolgálása. A komponens összetettsége abból ered, hogy az adatforrás amivel dolgozik bármi lehet, akár a készülék memóriájában lévő adatbázis, akár weben található RSS. A rendszerben minden ContentProvider-nek saját egyedi, úgynevezett URI-azonosítója van. Ezen keresztül, akár más alkalmazásból is elérhető egy-egy ContentProvider.

Erre példa az alkalmazásomban, ahogy a Google szervereiről lekéri a térképes adatokat, melyeket egy Service feldolgoz és átadja az Activity-nek.

4.1.4.BROADCASTRECIEVER

A BroadcastReciever-nek az a feladata, hogy egy esemény hatására valamilyen feladatot végrehajtson. Az Android sok eseményt jelez broadcast-ok formájában, melyekre feliratkozhatunk az alkalmazásunkkal. Ilyen események például az alacsony akkumulátor szint, telefonhívás vagy üzenet érkezése. A BroadcastReciever-nek nincs saját felhasználói felülete, de meg tud jeleníteni Activity-ket, felugró ablakokat vagy akár hangjelzést is.

Amikor bekövetkezik egy broadcast esemény az Android rendszer megvizsgálja, hogy melyik alkalmazások vannak feliratkozva és azoknak a szükséges komponenseit elindítja. Ezen kívül a rendszer lehetőséget biztosít saját broadcast esemény indítására, melyek ugyanúgy viselkednek, mint a beépített broadcast-ok.

5.A GOOGLE MAPS-RŐL

A Google Inc. által fejlesztett, ingyenesen használható internetes térképszolgáltató rendszer. 2005. október 6-a óta érhető el, bár akkoriban Google Local néven. 2006-óta Magyarország is megtalálható rajta a többi európai országgal együtt.

A többi Google szolgáltatáshoz hasonlóan ez is nagy mennyiségű JavaScript-tel készült, a nagy interaktivitás elérése miatt. Ám nem tisztán JavaScript, ugyanis nagymértékben XML hálózati kereséseket is használ. Ezek együttes használatával jobb minőséget érnek el. Ennek a fejlesztésnek a neve AJAX.

A térképek Mercator vetületűek, melyeket különböző helyekről szerzik be, eleinte a Navteq-től később pedig a Tele Atlas-tól. Azonban a fejlődő országokban a közösség fejleszti a Google Térképkészítővel. Később mindenhol lecserélték a Tele Atlas-t a saját adatbázisukra, de vannak országok ahol egyedi partnere van a Google-nek (pl. Németország-Geobasis).

A térképnek két nézete van, egy normál térkép nézet és a műholdas nézet, mely műhold képek mozaikjából áll és erre vannak ráillesztve az utak és a pontszerű objektumok.

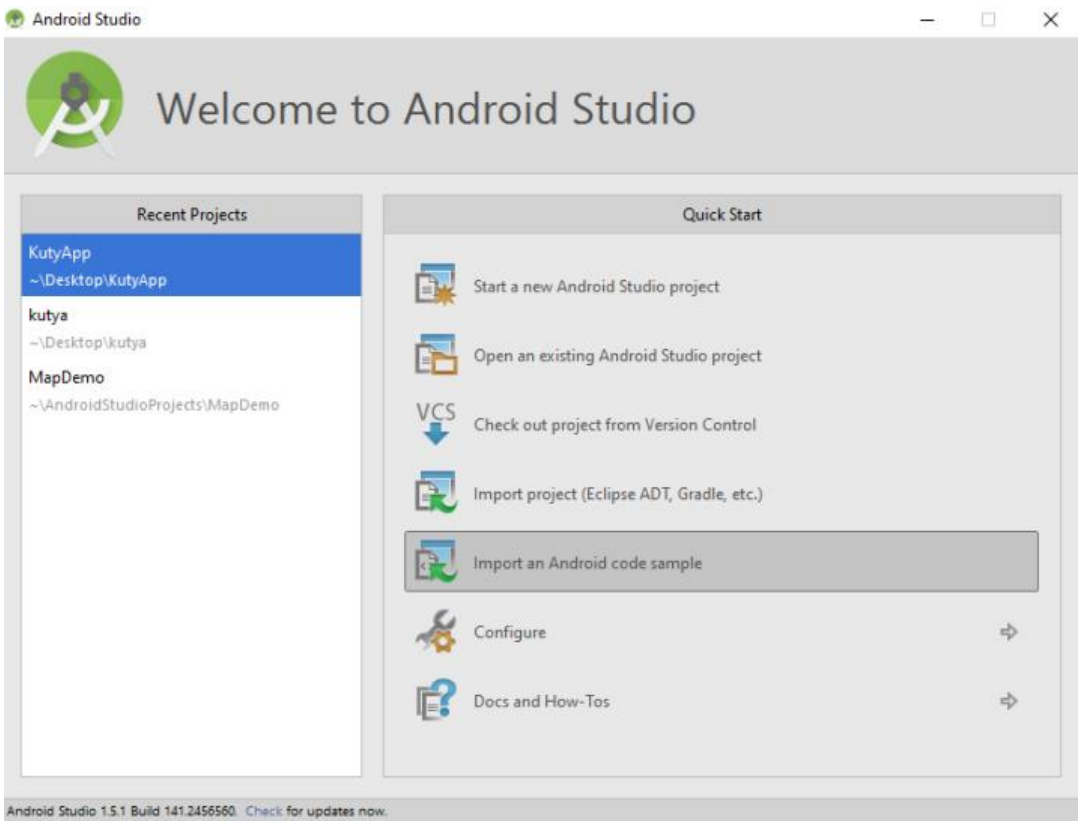
2007-től pedig a kiegészítették az „Utcakép” nézettel a Google Streetview-val, mely 2013-tól Magyarországot is lefedi.

Fejlesztői oldalról pedig elérhető a Google Maps API kulcs, melyet szintén a Google fejleszt és ingyenesen felhasználható, regisztráció és igénylés után. Ez, és a teljes Google Maps elérhető Androidon is, ezért ezt választottam fejlesztési iránynak.

6.ANDROID FEJLESZTŐKÖRNYEZET

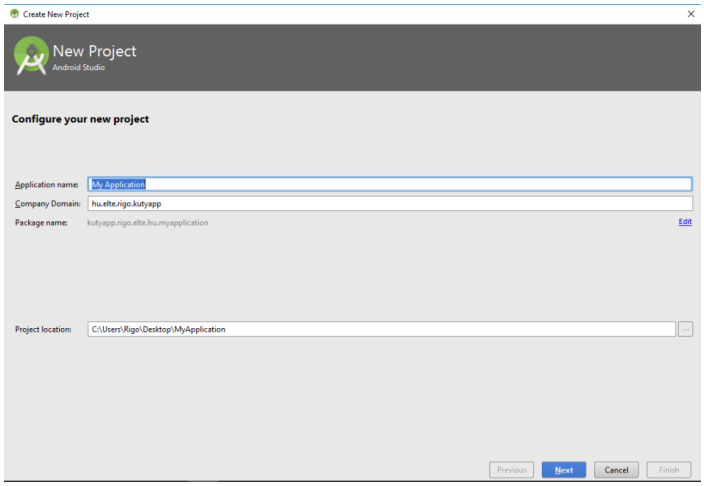
Én az Android Studio-t használtam fejlesztésre, ugyanis ez a Google saját fejlesztő-környezete 2014-óta, és minden benne van ami kell egy alkalmazás elkészítéséhez. Az egészet úgy kezdtem, hogy letöltöttem a programot a https://developer.android.com/sdk/index.html oldalról. Az Android Studio egyik nagy előnye, a régebbe használt Eclipse-hez képest, már itt feltűnik, hogy az SDK (Software Development Kit) hozzá tartozik a fejlesztőkörnyezethez és nem kell külön csatolni hozzá.

Feltelepítés után azonnal el lehet kezdeni a fejlesztést. Az első ablakban lehet kiválasztani, hogy új vagy régebbi projekt induljon el.



6. ábra Android Studio - kezdő ablak

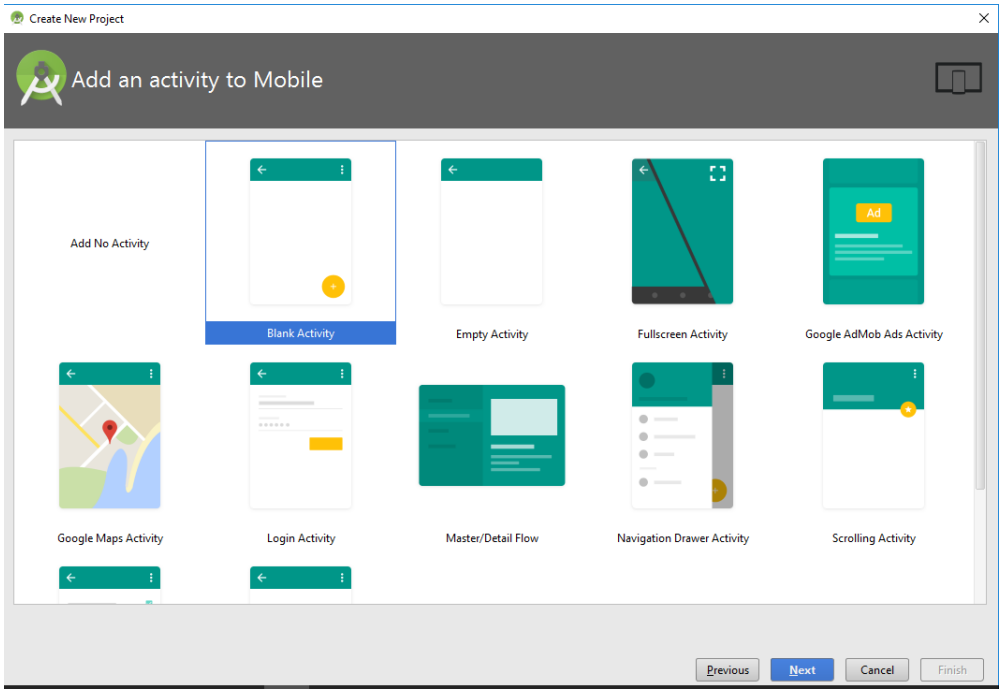
Ezután meg kell adni az alkalmazás nevét, domain nevét és, hogy hova mentse.



7. ábra Android Studio - Név megadása

Ezen tovább haladva, a következő ablakban kell kiválasztani, hogy melyik Android verzióra készül a program. Nem csak telefonra és tabletre lehet fejleszteni, de lehet Wear-re (okos óra), TV-re, Android Auto-ra és Google Glass-ra is.

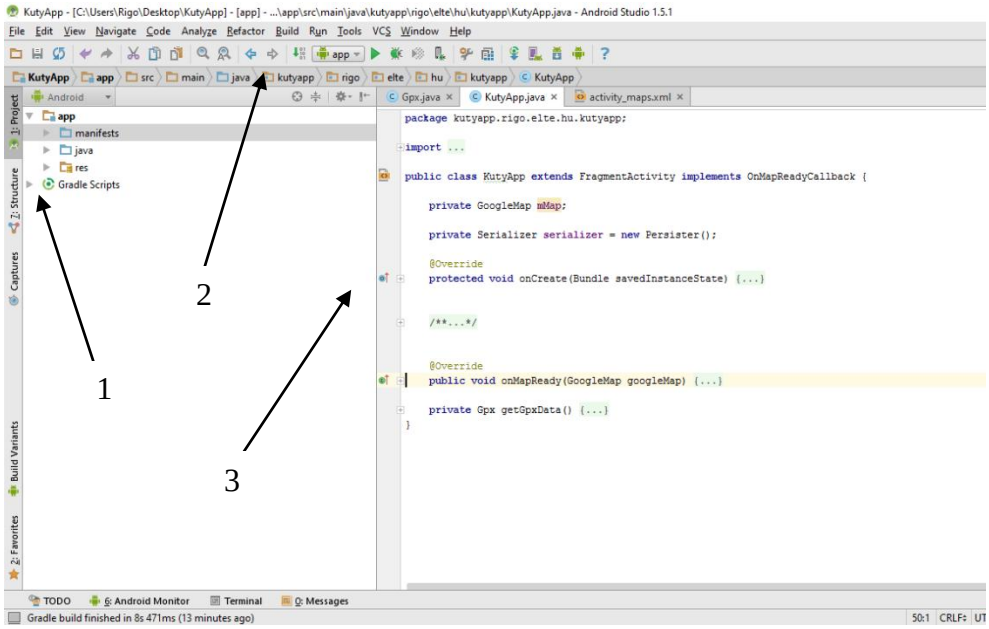
Én ugyebár a telefont választottam és a 4.0.3. (Ice Cream Sandwich) verziót. Ezután történik az alkalmazás fő képernyőjének kiválasztása, ez az úgy nevezett „Main Activity”.



8. ábra Android Studio - Activity kiválasztása

Itt a kép bal alsó sarkában látható Google Maps Activity-t választottam. Ezután meg kell adni az Activity osztálynak és a layout osztálynak a nevét, valamint, hogy milyen néven legyen látható majd az alkalmazás. Ez a név fog megjelenni a fejlécben mikor fut az alkalmazás és az alkalmazás ikonja mellet. A layout határozza meg, hogy az Activity milyen módon jelenik meg. Az Activity és layout-jának leírása XML fájlba történik.

Amint ezeket a lépéseket elvégeztem, kezdhettem is a fejlesztést. Mielőtt azonban belekezdek, érdemes bemutatni az ablakok és a projekt struktúrájának elrendezését.



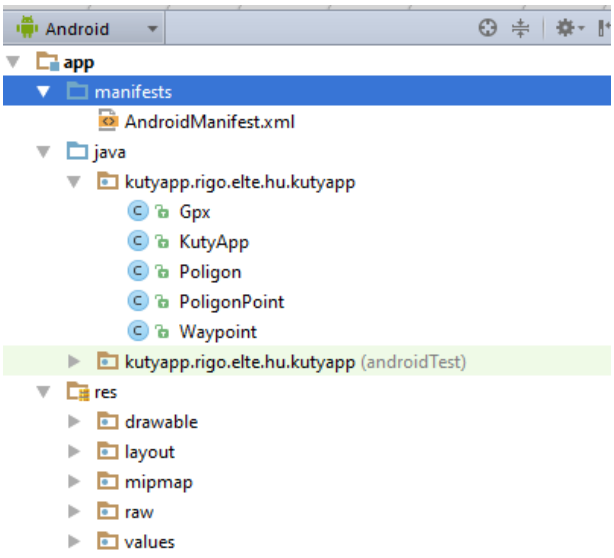
9. ábra Android Studio - Fejlesztői felület

1) Projekt könyvtár
2) Eszköztár
3) Munkaterület

6.1.PROJEKT KÖNYVTÁR

A jobb oldali képen jól látszik az alkalmazás logikus és átlátható felépítése. A legelső mappa a manifest, mely az AndroidManifest.xml fájlt tartalmazza. Ebben az állományban vannak felsorolva az applikáció alap tulajdonságai:

- Melyik a minimális Android verzió, amelyiken elfut
- Milyen hozzáféréseket kér a felhasználótól (pl.: internet hozzáférés, GPS használat, stb.)
- Az ikon hozzárendelése erőforrásból
- API kulcs a Google Maps használatához



10. ábra Android Studio - Projekt könyvtár

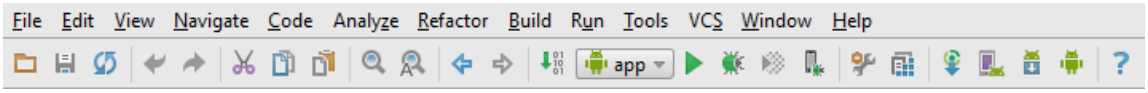
A gyakorlatban ez a legelső rész, amit az Android lefuttat. A manifest mappa alatt látható a java könyvtár. Mint a neve is mutatja, itt találhatók a java osztályok, melyek az alkalmazás gerincét adják, illetve egy teszt mappa, ami lefuttatásokkor íródik automatikusan.

A következő pedig a res könyvtár. Ezt érdemesebb jobban kifejteni, mint az előzőket, ugyanis ebben van minden erőforrás, amit a java osztályok használnak. A képen látható mappák közül a raw az egyetlen, amit én hoztam létre, a többi automatikusan legenerálódik.

- Drawable: Ebbe kerülnek bele azok a képek vagy alakzatok, amik az alkalmazás futása közben megjelennek.
- Layout: Itt található a, már az Activity választásnál is szóba került, layout leíró XML fájl.
- Mipmap: Ezen belül van még egy mappa, melybe a launcher ikon található. Ennek mindig .png kiterjesztésűnek kell lennie és ajánlott több felbontásban is belerakni, hogy különböző kijelzőméretű eszközökön is jól látszódjon.

- Values: Itt vannak azok az erőforrások, melyek az Android rugalmasságához nagyban hozzájárulnak. Itt adjuk meg külön XML fájlokban például a színeket amiket használni akarunk vagy a string-eket amik megjelennek az alkalmazásban, de a teljes engedély a Google Maps-hez is ebben a mappában van. Ezek közül a strings.xml szerintem a legérdekesebb, ugyanis ez teszi lehetővé, hogy különböző nyelveken megjelenjenek a szövegek. Létre lehet hozni több strings fájlt is és ilyenkor a neve után kell írni a lokációt, ami két betűből áll és az országra utal, ahol az a nyelv is van (pl. strings-HU).

6.2.ESZKÖZTÁR



11. ábra Android Studio - Eszköztár

A felső eszközsor elején a File menüben, lehet elindítani új projektet, ha nem akarunk kilépni a programból vagy betölteni egy régebbi projektünket. De innen lehet elmenteni is más néven, mint az eredeti neve. Az Edite-ben különböző szerkesztő módok vannak. A View-ban pedig testre szabható bizonyos mértékig az Android Studio kinézete.

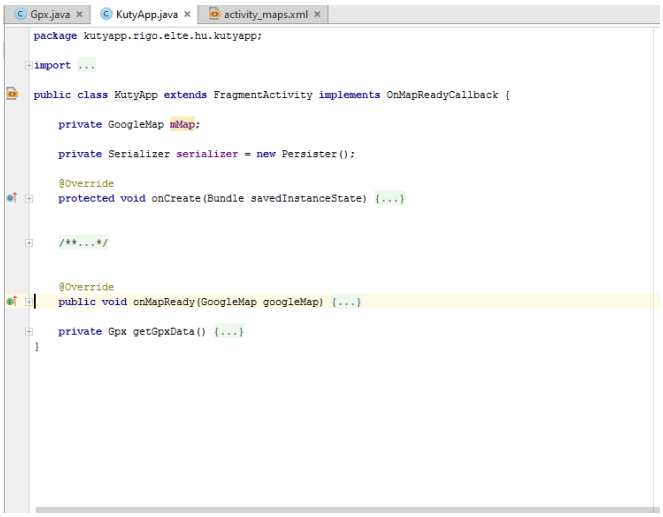
A felső sorban, ami még érdekes most, az a Build és Run fülek. A Build-ben lehet legenerálni a végső alkalmazás kiterjesztést, az APK-t. Ez a fájl kerül fel az eszközre és erről települ fel maga az alkalmazás. A Run-ban pedig megtalálhatóak a futtatáshoz szükséges feladatok. Ezek megtalálhatók az alsó sorban is. Közvetlenül alatta lehet kiválasztani, hogy miként futtassa (app), mellette jobbra található a futtatás elindítása ( ), amellett pedig a hibaszűrő van ( ).

Ezektől jobbra a szélén található az AVD (Android Virtual Device) Manager ( ), ebben lehet Android virtuális gépet létrehozni, melyben ugyanúgy lefuttathatók az alkalmazások. Tőle jobbra az SDK (Software Development Kit) Manager ( ). Innen kezelhetjük, hogy melyik Android verzióknak a program könyvtárait szeretnénk letölteni. És egyéb beépülő modulokat is innen lehet letölteni. A következő ikon pedig az Android

Device Monitor (), melyen keresztül parancsokat adhatunk az emulátorunknak, hogy úgy viselkedjen, mintha valódi eszköz lenne. Ilyen események például: telefonhívás, üzenet érkezése vagy új koordinátát érzékel a GPS.

6.3.MUNKATERÜLET

A munkaterület két fő részből áll, a fülekből, amelyekkel válthatunk az egyes projekt fájlok között, és a szövegszerkesztő felületből, ahol magát a programkódot írhatjuk, szerkeszthetjük. Mivel az Android Studio egy teljes értékű fejlesztő környezet, itt is sok gyorsbillentyű, segítő feliratok és automata szövegjavító áll rendelkezésünkre, a gyors és hatékony fejlesztés érdekében.



12. ábra Android Studio - Munkaterület

7.AZ ALKALMAZÁS FEJLESZTÉSE

Az alkalmazás fejlesztéséhez a Java tanulmányaim adták az alapot, melyhez három fő segédanyagot használtam, hogy Androidra elkészülhessen az alkalmazás. Az egyik Ekler Péter, Fehér Marcell, Forstner Bertalan és Kelényi Imre által írt Android alapú szoftverfejlesztés című könyv, a másik pedig egy 29 részes YouTube video sorozat, mely Ekler Péter előadásairól szól. A harmadik pedig természetesen az Android fejlesztői oldala.

7.1.ELSŐ LÉPÉSEK

A kezdőlépések elvégzése után (alkalmazás neve, Google Maps Activity kiválasztása, stb.), el is kezdtem elkészíteni az applikációt. Első lépésként az Android Studio ajánlotta weboldalt kellett meglátogatnom, ahol igényelhettem API kulcsot, hogy használhassam a Google Maps Activity-t az alkalmazásomban. Miután megkaptam a kódot, bemásoltam a res/values/google_maps_api.xml fájlba a szükséges helyre.

Amikor ebben a formában elindítjuk az alkalmazást, egy térképen kívül csak egy piros tűt látunk Sydney-be szúrva, mert ez az alapértelmezett marker. Egyébként lehet bárhova belenagyítani, elhúzni a térképet ugyanúgy, mint az alap Google Maps alkalmazásban.

Következőnek létrehoztam a res könyvtárban a raw mappát, amibe beillesztettem az általam felmért és megszerkesztett adatokat tartalmazó GPX fájlt.

Ezután a Gradle fájlból meghívtam az XML olvasásra alkalmas modulokat, amiket könnyedén meg lehet találni fejlesztői fórumokon.

```
compile('org.simpleframework:simple-xml:2.7.1')
{
    exclude group: 'xpp3', module: 'xpp3'
    exclude group: 'stax', module: 'stax-api'
    exclude group: 'stax', module: 'stax'
}
```

7.2.JAVA

A Java egy általános célú, magas szintű, objektum orientált programozási nyelv, melyet 2009-ig a Sun Microsystems fejlesztett, azóta az Oracle. Létrehozásának célja a platformfüggetlenség volt. Ezért van szükség a Java nyelven írt programok futtatásához virtuális gépre (JVM – Java Virual Machine), ugyanis ez fordítja le az adott platform számára használható szintaxisra.

7.2.1.GPX.JAVA

Ez a Java osztály felelős a GPX fájlformátum leírására, hogy milyen értékeket adhat az alkalmazás számára. A program kisebb mérete érdekében a fájl beolvasására annotációkat használtam.

Első lépésként meg kellett határozni, hogy a fájl amiből az adatokat kiolvassa gpx formátumban van.

```
@Root (name = "gpx", strict = false)
public class Gpx {
```

Ezután megadtam, hogy milyen objektum típusokat keressen a fájlban. Ezek az útpontok (waypoint) és poligonok (polygon). Ezeket listaként kell meghatározni, melyek különböző elemeket tárolnak. Ezeket az @ElementList annotációval oldottam meg.

```
@ElementList (inline = true, required = false)
private List<Waypoint> wayPoint;

@ElementList (inline = true, required = false)
private List<Poligon> polygons;
```

Ezek alapján, könnyedén elkészíthetőek, az ezeket meghívó függvények. A kijelölésük után jobb klikk/generate/getter and setter. Itt kiválasztom mindkét elemet és legenerálja az alábbi függvényeket.

```
public List<Poligon> getPolygons() {
    return polygons;
}

public void setPolygons(List<Poligon> polygons) {
    this.polygons = polygons;
}

public List<Waypoint> getWayPoint() {
    return wayPoint;
}

public void setWayPoint(List<Waypoint> wayPoint) {
    this.wayPoint = wayPoint;
}
```

Ezzel el is készült a GPX beolvasásához szükséges alap osztály.

7.2.2.WAYPOINT.JAVA

A következő osztály az útpontok osztálya. Először meg kell adni, hogy a fájlban milyen néven (wpt) találja meg ezeket az objektumokat.

```
@Root(strict = false, name = "wpt")
public class Waypoint {
```

A továbbiakban a lista egyes objektumainak attribútumait (@Attribute) és elemeit (@Element) határoztam meg változók formájában, beleértve a változók típusait is.

```
@Attribute(name = "lat")
double lat;

@Attribute(name = "lon")
double lon;

@Element(name = "name")
String name;

@Element (name = "desc", required = false)
String desc;

@Element (name = "sym")
String sym;
```

A két attribútum a „lat” és „lon” értelemszerűen az egyes pontok szélességi és hosszúsági koordinátákat tartalmazzák, ezért double típusú változókként vannak definiálva. Az @Element-ként definiált további tulajdonságok, pedig a további jellemzőket

tárolják. A „name” az objektum nevét, a „desc” pedig leírást tartalmaz. A „sym” pedig meghatározza, hogy milyen térképi jel tartozik az adott objektumhoz.

Erről az öt adatról is, az előző fejezetben leírtak alapján elkészítettem a gettereket és settereket.

7.2.3.POLIGON.JAVA / POLIGONPOINT.JAVA

A Poligon.java osztályban határoztam meg, hogy milyen elemei vannak a poligon objektumoknak. Először is van egy objektumneve a fájlban (trk) és van egy neve, ami a valódi neve.

```
@Root(strict = false, name = "trk")
public class Poligon {

    @Element(name = "name")
    String name;
```

Ezen kívül tartalmaz még egy @ElementList-et, mely a poligon határoló pontjait tartalmazza. Ez a PoligonPoint.

```
@ElementList(name = "trkseg")
List<PoligonPoint> pontok;
```

Ezekből szintén létrehoztam a gettereket és a settereket.

Hogy működjön ez az osztály is, létrehoztam a PoligonPoint.java osztályt, mely a sarokpontok koordinátáit adja meg. Ezek az @Attribute-ok is double értékkel térnek vissza az alkalmazásba.

```
@Root(name = "trkpt", strict = false)
public class PoligonPoint {

    @Attribute(name = "lat")
    double lat;

    @Attribute(name = "lon")
    double lon;
```

Miután a getterek és setterek itt is kész lettek, bele is kezdtem az Activity elkészítésébe.

7.2.4.MAPACTIVITY.JAVA

Ez az osztály három fő részből tevődik össze. Az első rész az onCreate függvény, mely elindítja az Activity-t és meghív minden szükséges erőforrást a térkép megjelenítéséhez. Ehhez a programrészhez gyakorlatilag hozzá se kellett nyúlnom, ugyanis az Android Studio automatikusan legenerálta nekem ezt a programrészt.

A második része az osztálynak, jeleníti meg magát a térképet, beleértve az általam gyűjtött adatokat is.

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

    Gpx gpx = getGpxData();
```

A saját adatok megjelenítéséért felelős részt két kisebb blokk alkotja. Az első a pontszerű objektumokat jeleníti meg.

```
for (int i=0; i<gpx.getWayPoint().size(); i++) {
 Waypoint getWpt = gpx.getWayPoint().get(i);

 BitmapDescriptor bitmapDescriptor;

 switch (getWpt.getSym()) {
   case "kuka" :
    bitmapDescriptor =
BitmapDescriptorFactory.fromResource(R.drawable.kuka);
   break;
   ……….
   case "iskola" :
    bitmapDescriptor =
BitmapDescriptorFactory.fromResource(R.drawable.iskola);
   break;
   default:
    bitmapDescriptor =
BitmapDescriptorFactory.fromResource(R.drawable.def);
    }

 LatLng wpt = new LatLng(getWpt.getLat(), getWpt.getLon());

mMap.addMarker(new
MarkerOptions().position(wpt).title(getWpt.getName())
.snippet(getWpt.getDesc()).icon(bitmapDescriptor));
```

A for ciklus végig halad a GPX fájlban egyesével minden objektumon, és először is eldönti, hogy melyik objektum, milyen térképi kategóriába tartozik. Ez alapján hozzárendeli a megfelelő térképi jelet. Ez után kiolvassa az adott pont koordinátáit és oda rak a térképen egy Markert, majd a feliratba beleilleszti a nevét és leírását. Végül a hozzárendelés alapján a megfelelő térképi jellel behelyettesíti az alap Marker jelet ( ).

A második blokk pedig a poligonok kirajzolását végzi, ugyanúgy két lépésben, mint a GPX kiolvasásáért felelős Poligon és PoligonPoint osztályok.

```
for (int i=0; i<gpx.getPolygons().size(); i++) {
    PolygonOptions polygonOptions = new PolygonOptions();
    Poligon poligon = gpx.getPolygons().get(i);

    for (int j=0; j<poligon.getPontok().size(); j++) {
        polygonOptions.add(new
LatLng(poligon.getPontok().get(j).getLat(),
poligon.getPontok().get(j).getLon()));
    }

    int color = R.color.color;
    polygonOptions.fillColor(color);
    mMap.addPolygon(polygonOptions);
}
```

Az külső for ciklus a poligonokon meg végig, miközben a belső az adott poligon sarok pontjait határozza meg. A belső ciklus lejárta után határoztam meg a poligon színét és kitöltésének módját. A színt XML erőforrásból hívtam meg. Majd a ciklus legvégén az mMap.addPolygon() függvény rajzolja ki a kész poligont.

A MapActivity osztály harmadik része adja meg az alkalmazásnak, hogy hol találja meg a szükséges GPX fájlt és melyik osztály segítségével tudja felhasználni.

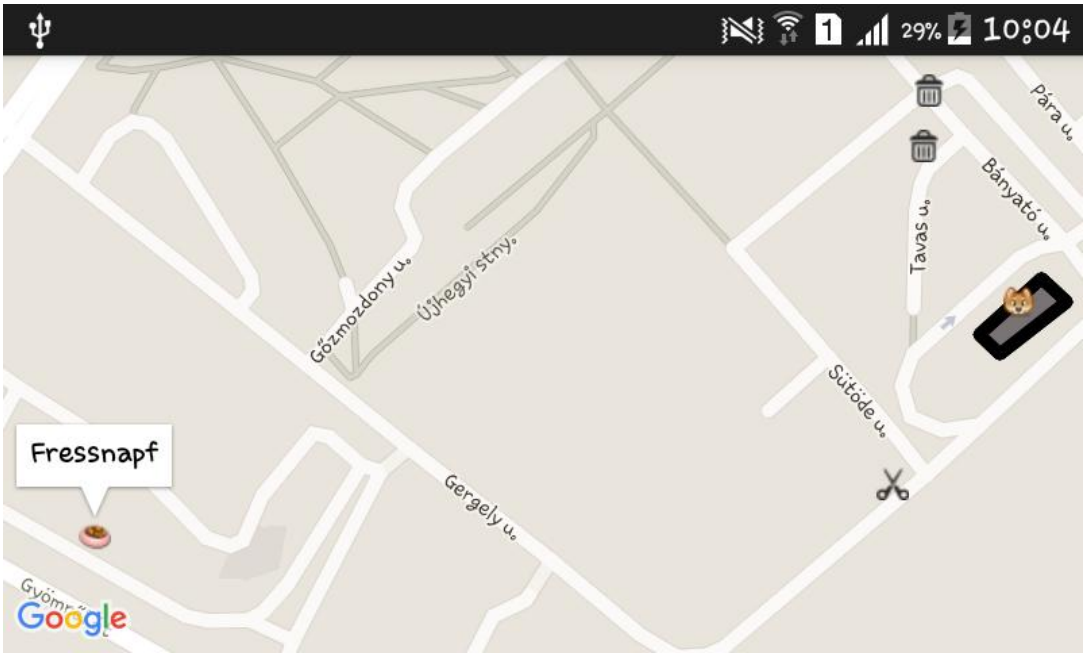
```
private Gpx getGpxData() {
    InputStream source =
getResources().openRawResource(R.raw.kutya);

    Gpx example;
    try {
        example = serializer.read(Gpx.class, source);
        return example;
    } catch (Exception e) {
        throw new RuntimeException("cannot read gpx file", e);
    }
}
```

7.3.TESZT

Miután elkészült az alkalmazás, el is indítottam. Két féle módon lehet letesztelni, hogy fut-e a program. Az egyik, hogy az AVD(Android Virtual Device) Manager-ben létrehozunk egy emulátort. Ezzel az a baj, hogy lassú és a ráközelítés után nem tudunk kijönni belőle, ugyanis azt csak két ujjal lehet megoldani telefonon is.

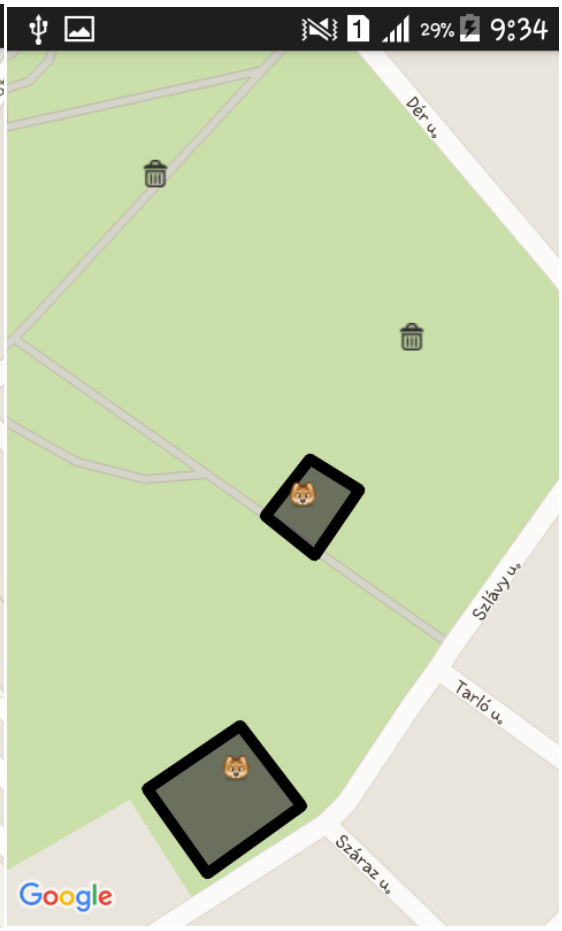
A másik megoldás, amit én is használtam, hogy az SDK(Software Development Kit) Manager-ben letöltöttem az USB Drivert, a telefonomon pedig beállítottam a fejlesztői módot és így ha USB kábellel összekötöttem a számítógépet és a telefont, az alkalmazás azonnal a telefonomon futott le. Ez Windows operációs rendszeren macerásabb, mint Linuxon, ugyanis a Linux automatikusan érzékeli és nem kell a driver. Windowson USB driverrel se mindig működik ez a módszer. Fórumokon utána olvastam, de eddig szinte senkinek nem sikerült megoldania a problémát, ezért érdemes Linuxon fejleszteni Android alkalazásokat.

A tesztelés alatt vettem észre egy problémát. A probléma az, hogy a poligonokat a térkép csak szürke kitöltéssel jeleníti meg és nem zölden, ahogy megadtam a colors erőforrást. Erre a problémára nem találtam magyarázatot, ezért nem is tudtam kijavítani.

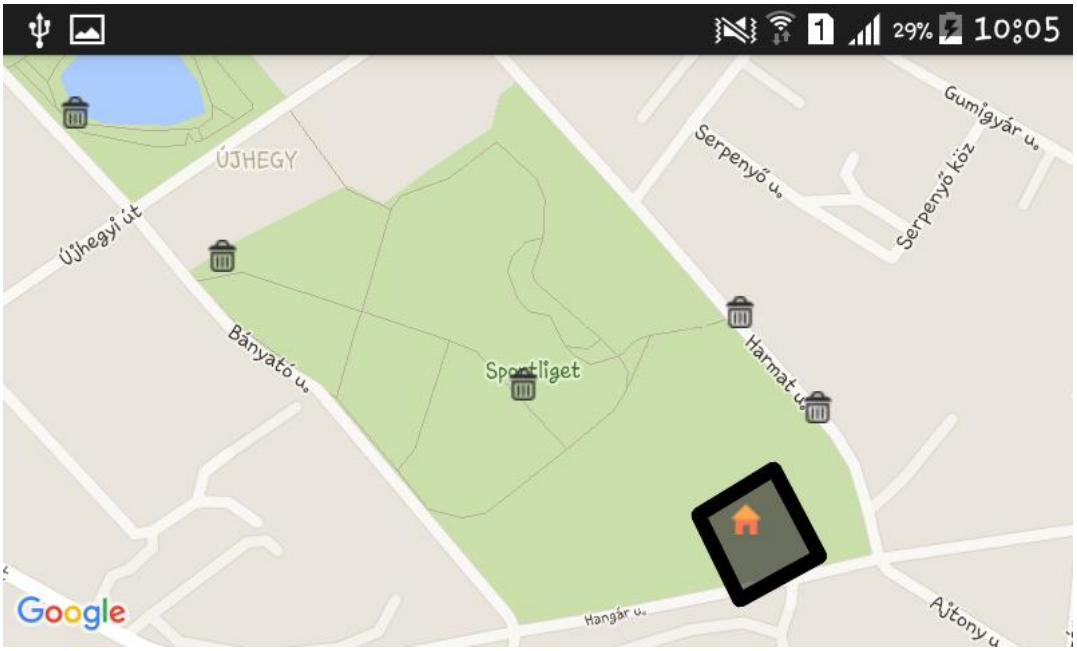


13. ábra Képernyő fotó - Újhegy

14. ábra Képernyő fotó – Óhegy park



15. ábra Képernyő fotó – Gergely bánya



16. ábra Képernyőfotó – Sportliget

8.JÖVŐ

Mivel ez az alkalmazás, csak Kőbányát fedi le, és Kőbányán is most kezdtek el újabb kutyafuttatókat létesíteni, a későbbiekben is fejlesztésre szorul a térképi tartalom. Akár aktualizálásról, akár az alkalmazás által lefedett terület méretének növeléséről legyen szó.

Nem csak a térképi tartalom fejlesztése a cél, hanem maga az alkalmazásé is. Mind a felhasználói felület, mind a megjelenítés finomításokra szorulnak. A felhasználói felületre szeretnék keresőt beilleszteni, hogy a felhasználók céltudatosan is tudják használni az alkalmazást. Illetve, hogy a pozíciójukhoz legközelebb eső objektumokat ki tudják szűrni. Amennyiben ezek a terveim sikerülnek, mindenki számára elérhetővé szeretném tenni a Google Play-en is.

9.ÖSSZEGZÉS

Dolgozatom célja egy Kőbánya területét lefedő digitális térkép volt, melyet Android platformra készítettem, alaptérképként a Google Maps-et használtam.

A felméréseket 2015 nyarán kezdtem, melyeket 2016 májusában fejeztem be. E munkafolyamat elvégzése alatt körvonalazódott meg bennem, hogy a későbbiekben hogyan kategorizáljam az adataimat. A felmérések közben folyamatosan dolgoztam fel az adatokat.

Az alkalmazásomat 2016. április közepére fejeztem be. A térképi kategóriák alapján, ekkor állítottam össze a jelkulcsot, a http://softicons.com oldalon található ikonokból. Itt először egy kisebb problémával találkoztam, ugyanis az első ikonok, amiket használtam 48x48 pixelből álló PNG képek voltak, melyek túl nagynak bizonyultak. A jelenleg használt ikonok viszont 16x16-os méretűek, ezek ideálisak a térképi jeleknek.

Összességében úgy gondolom, hogy az eredeti céljaim nehezét megoldottam és sikerült a munkám alatt új ismereteket szereznem.

TÁBLÁZATOK

IRODALOM JEGYZÉK

Ekler Péter – Fehér Marcell – Forstner Bertalan – Kelényi Imre (2012):
Android-alapú szoftverfejlesztés – Az Android rendszer programozásának bemutatása

Nagy Gusztáv (2007): Java programozás

INTERNETES HIVATKOZÁSOK

http://developer.android.com/index.html
https://commons.wikimedia.org/wiki/File:Android_robot.svg
http://softicons.com
https://buy.garmin.com
https://hu.wikipedia.org/wiki/Google_T%C3%A9rk%C3%A9p

KÖSZÖNET NYILVÁNÍTÁS

Ezúton köszönetet szeretnék mondani Dr. Kovács Bélának, amiért kérésemre elvállalta, hogy a témavezetőm legyen és segített, mikor problémába ütköztem.

Valamint hálás vagyok minden barátomnak és családtagomnak, akik mellettem álltak egész végig.

Nyilatkozat

Alulírott Rigó Dániel Imre (Neptun kód: HRZFYN) nyilatkozom, hogy jelen szakdolgozatom teljes egészében saját, önálló szellemi termékem. A szakdolgozatot sem részben, sem egészében semmilyen más felsőfokú oktatási vagy egyéb intézménybe nem nyújtottam be. A szakdolgozatomban felhasznált, szerzői joggal védett anyagokra vonatkozó engedély a mellékletben megtalálható.

A témavezető által benyújtásra elfogadott szakdolgozat PDF formátumban való elektronikus
publikálásához a tanszéki honlapon

HOZZÁJÁRULOK NEM JÁRULOK HOZZÁ

Budapest, 2016. május 13.

…….
a hallgató aláírása