



MATEMATIKAI ÉS INFORMATIKAI INTÉZET

# Kliens-szerver kommunikáció Android platformon

**Készítette**

Balajti-Tóth Kristóf

Programtervező Informatikus BSc

**Témavezető**

Tajti Tibor

Egyetemi adjunktus

EGER, 2019

# Tartalomjegyzék

<b>1. Fejlesztői eszközök</b>	<b>5</b>
1.1. Fejlesztői környezetek . . . . .	5
1.1.1. Android Studio . . . . .	5
1.1.2. Pycharm Professional Edition . . . . .	5
1.2. Postman . . . . .	5
<b>2. Platformok</b>	<b>6</b>
2.1. A szerver kiválasztása és felépítése . . . . .	6
2.2. Mobil platform választása . . . . .	7
<b>3. Felhasznált technológiák</b>	<b>8</b>
3.1. Folyamatos integrálás . . . . .	8
3.2. Verzió kezelés . . . . .	8
3.3. Szerveren használt technológiák . . . . .	9
3.3.1. Flask . . . . .	9
3.3.2. SQLite . . . . .	9
3.3.3. Docker . . . . .	9
3.3.4. Tmux . . . . .	10
3.3.5. jd-cmd . . . . .	10
3.3.6. Apktool . . . . .	10
3.3.7. dex2jar . . . . .	10
3.3.8. JSON . . . . .	10
3.4. Androidon használt technológiák . . . . .	10
3.4.1. OkHttp3 . . . . .	10
3.4.2. Okio . . . . .	10
3.4.3. Pusher . . . . .	10
3.4.4. Firebase Messaging . . . . .	10
3.4.5. Room . . . . .	10
3.4.6. CodeView . . . . .	10
3.4.7. Espresso . . . . .	11
3.4.8. Material Design 2.0 . . . . .	11

<b>4. Teszteléshez felhasznált eszközök és források</b>	<b>12</b>
4.0.1. Sérülékeny Android alkalmazások . . . . .	12
<b>5. Megvalósított funkciók</b>	<b>13</b>
<b>6. Továbbfejlesztési lehetőségek</b>	<b>14</b>
<b>7. Tapasztalatok</b>	<b>15</b>

# Bevezetés

Az szoftver fejlesztés egy nagyon komplex folyamat és rengeteg részletre oda kell figyelni. Az elkészült programnak hatékonynak, hibamentesnek és gyorsnak kell lennie. Természetesen, mindezt határidőn belül kell teljesíteni. Sajnos a biztonság nem egy első számú szempont egy megrendelő szemében, csak akkor ha már valami baj történt. Inkább a gyorsaságon és a folyamatok automatizálásán van a hangsúly, ezért nem meglepő, hogy a fejlesztés életciklusának tervezési szakaszában kevés figyelem fordul a szoftver biztonságossá tételére.

A statista.com [1] kutatása szerint 2020-ra több mint 4.78 billió telefon lesz használatban. Ezzel a cégek is tisztában vannak és tudják, hogy ha még több emberhez szeretnék eljuttatni a szolgáltatásukat, akkor rendelkezniük kell saját mobilos alkalmazással.

A mobilos eszközöket célzó támadások száma hatalmas ütemben nő. Mindez azért lehetséges, mert figyelmen kívül marad a „secure coding”-nak nevezett gyakorlat. Egy alkalmazásnak a sebezhetőségét különböző támadási vektoron is ki lehet aknázni. Az elején, bennem többek között az a kérdés merült fel, hogy honnan tudható hogy ez alkalmazás ebezhető-e vagy sem. egy kérdés merült fel. Honnann tudhatom, hogy egy adott alkalmazás sebezhető-e vagy sem. A leghatékonyabb módszer ha visszafejtjük a fájl forráskódra. Ezt angolul „reverse engineering”-nek nevezik. A visszaállított fájlok olvashatósága nem lesz tökéletes, főleg ha obfuszkált <sup>1</sup> kóddal állunk szemben, de egy tapasztalt szem így is kitudja szűrni a gyakori hibákat.

A szakdolgozatomban Android platformra készült telepítő fájlok forrás fájlokká való visszaállításáról írok, valamint bemutatom hogyan valósítható meg a kliens-szerver kommunikáció egy REST API és egy Androidos alkalmazás segítségével. A projectet „Reverse Droid”-nak neveztem el.

---

<sup>1</sup> Az obfuszkáció célja röviden, hogy megnehezítse a visszafejtett kód olvashatóságát.

# 1. fejezet

## Fejlesztői eszközök

### 1.1. Fejlesztői környezetek

#### 1.1.1. Android Studio

Az Android Studio jelenleg az egyetlen jól támogatott és minőségi fejlesztői környezet Android fejlesztéshez. Régebben sok panaszt hallottam az emulátorára, hogy nagyon lassú és körülményes a használata. Mára már egy pillanat alatt lehet futtatni a programunk és abszolút kényelmes lett a használata. Rendelkezik APK elemzővel, vizuális felhasználó felület szerkesztővel és intelligens kód szerkesztővel is. Az egyik kedvenc funkcióm a valós idejű profilozó, ami segítségével megtudjuk nézni valós időben, milyen erőforrásokat használ az alkalmazásuk. Ez különösen hasznos, ha megakarunk találni egy memória szivárgást vagy egy olyan részt, ami a kelleténél jobban meríti az akkumulátorunk. Említésre méltó még a flexibilis build rendszere is, a Gradle. Használatával megtehetjük, hogy külön build típusokat hozunk létre a különböző eszközökre. [2]

#### 1.1.2. Pycharm Professional Edition

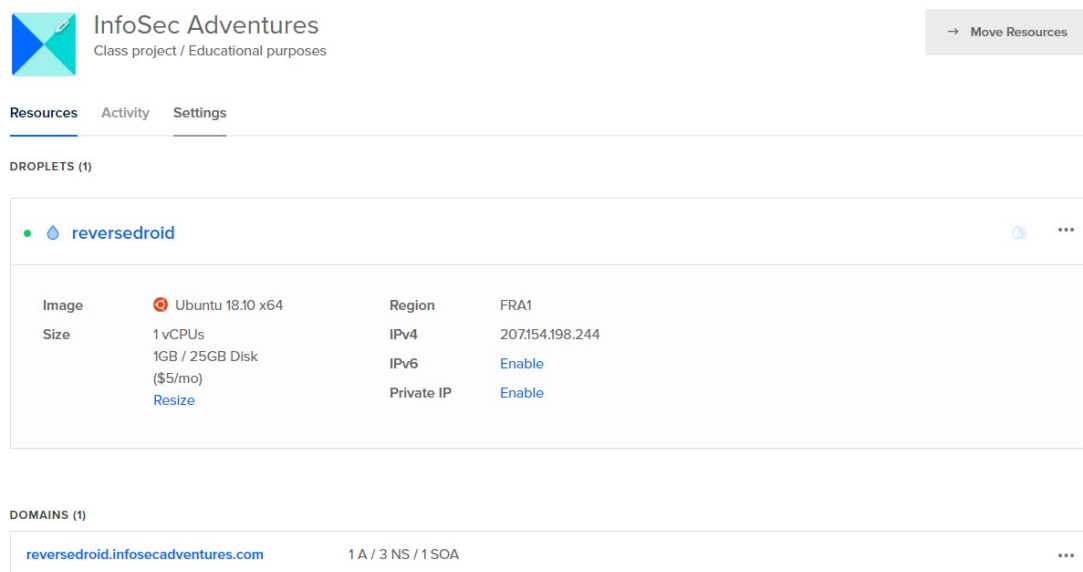
### 1.2. Postman

## 2. fejezet

# Platformok

### 2.1. A szerver kiválasztása és felépítése

Olyan szerverre volt szükségem, ami nem túl költséges, de mégis megfelelően testreszabható és gyors tárhelyet biztosít. A választásom a Digital Ocean felhő szolgáltatására esett. Az oldal felületén lehetőségünk van több, úgynevezett *droplet*-et létrehozni, amik nem mások mint virtuális szerverek. Megadhatjuk milyen disztribúciót szeretnénk telepíteni, jelen esetben én az Ubuntu Linux 18.10-es verzióját telepítettem.



2.1. ábra. Droplet a Digital Ocean admin felületén.

A projecthez készítettem egy subdomain-t és telepítés után a droplet IP címét hozzárendeltem ehhez a subdomain-hez. Ezzel biztosítottam, hogy domain név alapján is elérhető legyen a szerver. Ez a 2.2 képen jól látható.

A kész projectben nem ezt a folyamatot választottam, hanem a Digital Ocean által nyújtott „one-click apps” menüben egyszerűen kiválasztottam a Docker alkalmazást és

Type	Host	Value	TTL	
A Record	@	185.199.108.153	Automatic	
A Record	@	185.199.109.153	Automatic	
A Record	@	185.199.110.153	Automatic	
A Record	link	52.72.49.79	Automatic	
A Record	reversedroid	207.154.198.244	Automatic	

2.2. ábra. DNS rekordok a domain beállításában.

az elkészült képfájlt ezen futattam. Így automatizálva a szerver telepítésének folyamatát és megspórolva magának a Docker-nek a telepítését és konfigurálását. Erről még a Szerveren használt technológiák fejezet Docker alfejezetében bővebben írok.

## 2.2. Mobil platform választása

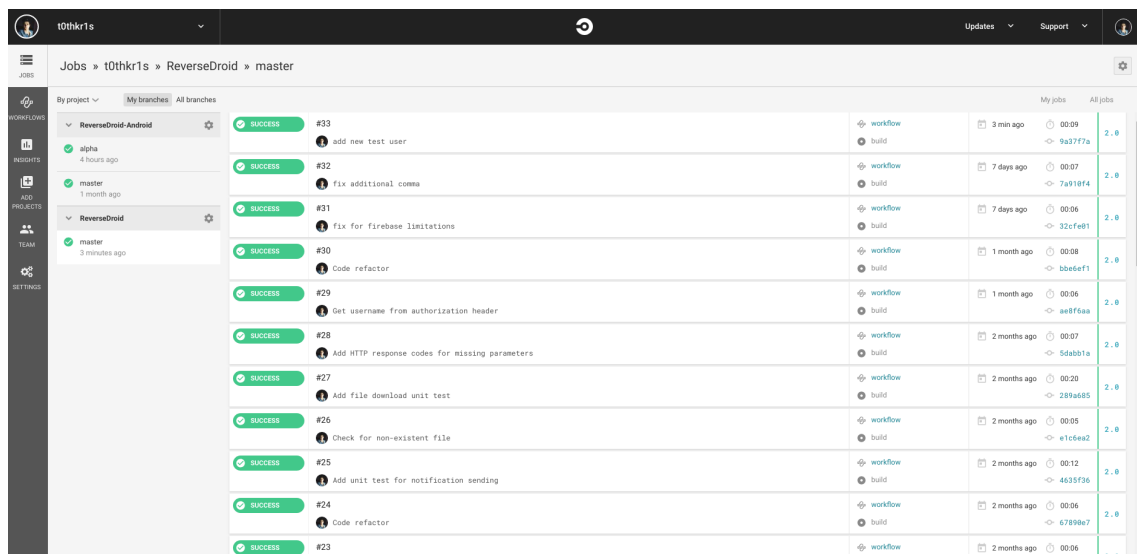
A mobilos operációs rendszerek közül az Androidot választottam. Már korábban sikerült megismerkednem az Android nyújtotta lehetőségekkel és előnyökkel. A többi mobilos operációs rendszerrel ellentétben az Android nyílt forráskódú és a piac több mint felét uralja. Ez annak is köszönhető, hogy 2005-ben a Google felvásárolta az Android projectet és azóta ők tartják karban. A fejlesztő környezete elérhető mind a három fő operációs rendszerre (Linux, macOS, Windows). Számomra ezek voltak a legnyomósabb érvek a rendszer kiválasztásában.

## 3. fejezet

# Felhasznált technológiák

### 3.1. Folyamatos integrálás

A folyamatos integrálás egy extrém programozási gyakorlat. A folyamatos integrálás arról szól, hogy ha egy feladat elkészült akkor azt egyből beintegráljuk a rendszerbe. A beintegrálás után természetesen minden egység tesztnek sikeresen le kell futnia. Több nagy cég is a *CircleCi*-t használja a folyamatos integráláshoz. Ilyen például a *Facebook*, *Spotify*, *Kickstarter* és a *GoPro*.



Jobs » t0thkr1s » ReverseDroid » master		My jobs		All jobs	
ReverseDroid-Android	success #33	workflow	3 min ago	00:09	2.0
alpha	add new test user	build		9a37f7a	
master	fix additional comma	workflow	7 days ago	00:07	2.0
	fix for firebase limitations	build		7a918f4	
ReverseDroid	fix for firebase limitations	workflow	7 days ago	00:06	2.0
master	Code refactor	build		32cfe81	
	Get username from authorization header	workflow	1 month ago	00:08	2.0
	Add HTTP response codes for missing parameters	build		bbeeef1	
	Add file download unit test	workflow	2 months ago	00:20	2.0
	Check for non-existent file	build		289a655	
	Add unit test for notification sending	workflow	2 months ago	00:05	2.0
	Code refactor	build		e1c6ea2	
	Code refactor	workflow	2 months ago	00:12	2.0
		build		4635f36	
		workflow	2 months ago	00:06	2.0
		build		6789ae7	
		workflow	2 months ago	00:06	2.0

3.1. ábra. Sikeres project buildek a CircleCi felületén.

### 3.2. Verzió kezelés

Már a project kezdetekor készítettem egy privát Github repository-t, hogy nyomon tudjam követni a változtatásaimat és esetleges hiba esetén visszaállítani egy korábbi verzióra.



## 3.3. Szerveren használt technológiák

### 3.3.1. Flask

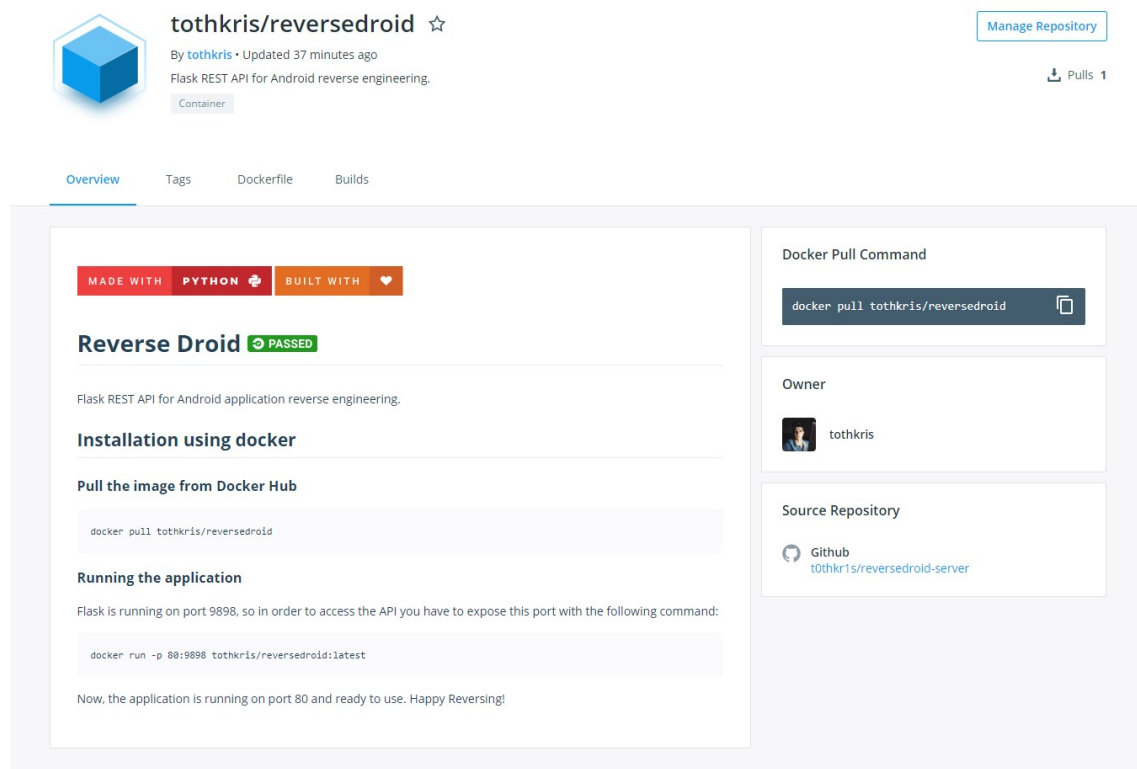
A szerveret a Flask webes mikro keretrendszer felhasználásával készítettem el. A *LinkedIn* és *Pinteres* is említésre méltó alkalmazások, amik felhasználták ezt a keretrendszert.

### 3.3.2. SQLite

Az SQLite a legtöbbet használt adatbázis motor a világon. Számptalan alkalmazás használja és Android-on is ez az alapértelmezett adatbázis. Az SQLite nyílt forráskódú, így mindenki nyugodtan használhatja.

### 3.3.3. Docker

A szerverhez készítettem egy Dockerfile-t és csatoltam a project Github-os repository-ját. Ezzel elérve, hogy minden egyes változtatásnál a Docker Hub újra buildelje a képfájlt. A folyamat nagyon hasonlít a folyamatos integrálásra.



3.2. ábra. A szerver Docker Hub-on is elérhető.

### **3.3.4. Tmux**

A Tmux vagy másnéven terminál multiplexer használatával több shell-t lehet futtatni. Ez még nem is lenne túl nagy segítség, viszont ezt a folyamatot le lehet csatolni majd vissza anélkül, hogy elveszne a tartalom. Ezen felül biztosítja, hogy a parancsok és azok kimenete kereshetők legyenek. Az alkalmazás alapvetően a parancssorba írja a bejövő kéréseket és ez egy idő után nehezen követhető. A Tmux nagy segítségemre volt, mivel lehetőséget adott hogy keressek a kérések között.

### **3.3.5. jd-cmd**

Szükségem volt egy parancs sorból használható Java Decompiler-re is.

<https://github.com/kwart/jd-cmd>

### **3.3.6. Apktool**

<https://github.com/iBotPeaches/Apktool>

### **3.3.7. dex2jar**

<https://bitbucket.org/pxb1988/dex2jar>

### **3.3.8. JSON**

## **3.4. Androidon használt technológiák**

### **3.4.1. OkHttp3**

### **3.4.2. Okio**

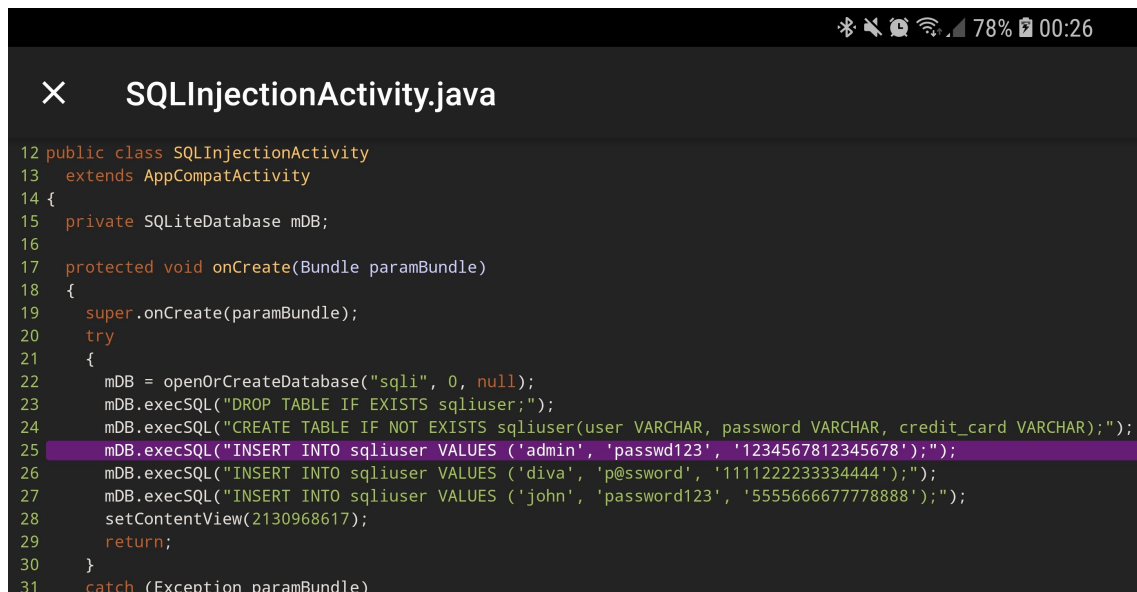
### **3.4.3. Pusher**

### **3.4.4. Firebase Messaging**

### **3.4.5. Room**

### **3.4.6. CodeView**

A forrás kód megjelenítését nem lett volna célszerű nulláról felépíteni, ezért inkább kész megoldások után néztem. Több megfelelő forrás kód megjelenítő könyvtárat találtam, így funkciók alapján kellett döntenem, hogy melyiket válasszam.



```
12 public class SQLInjectionActivity
13     extends AppCompatActivity
14 {
15     private SQLiteDatabase mDB;
16
17     protected void onCreate(Bundle paramBundle)
18     {
19         super.onCreate(paramBundle);
20         try
21         {
22             mDB = openOrCreateDatabase("sqli", 0, null);
23             mDB.execSQL("DROP TABLE IF EXISTS sqliuser;");
24             mDB.execSQL("CREATE TABLE IF NOT EXISTS sqliuser(user VARCHAR, password VARCHAR, credit_card VARCHAR);");
25             mDB.execSQL("INSERT INTO sqliuser VALUES ('admin', 'passwd123', '1234567812345678');");
26             mDB.execSQL("INSERT INTO sqliuser VALUES ('diva', 'p@ssword', '1111222233334444');");
27             mDB.execSQL("INSERT INTO sqliuser VALUES ('john', 'password123', '5555666677778888');");
28             setContentView(2130968617);
29             return;
30         }
31         catch (Exception paramBundle)
```

3.3. ábra. A vissza fejtett forrás kód megjelenítése.

### 3.4.7. Espresso

### 3.4.8. Material Design 2.0

## 4. fejezet

# Teszteléshez felhasznált eszközök és források

### 4.0.1. Sérülékeny Android alkalmazások

A teszteléshez keresnem kellett olyan sérülékeny alkalmazásokat, amelyeken jól lehet demonstrálni a visszafejtést. Minden alkalmazás, amit itt megemlítek nyílt forrás kódú és kimondottan erre a célra készítették őket. Ez azért is jó, mert a visszafejtett kódot össze lehet hasonlítani az eredetivel és megnézni mennyire volt hatékony a visszafejtési folyamat. A következő lista tartalmazza a teszteléshez felhasznált szándékosan sérülékeny alkalmazásokat.

- Diva
- Sieve
- Pivaa
- Frida

## 5. fejezet

### Megvalósított függvények

## 6. fejezet

# Továbbfejlesztési lehetőségek

Úgy gondolom, hogy sokkal nagyobb piaci érték rejlik ebben az alkalmazásban. A jövőben is szeretném folytatni a fejlesztést. Szeretnék több figyelmet fordítani az biztonságra és hatékonyságra. Gondolok itt a biztonságos kommunikációra TLS-es keresztül és a harmadik féltől származó könyvtárak csökkentésére. A kód visszafejtése végén egy összegző report is hasznos lehet a felhasználó számára, ami tartalmazhatja a feldolgozott fájlok számát.

## 7. fejezet

# Tapasztalatok

Úgy érzem elértem a célom ezzel a projecttel és sokat sikerült tanulnom a folyamatban. Új technológiákat ismertem meg és használtam. Természetesen nem volt minden magától értetődő és jó pár nehézséggel is találkoztam.

# Irodalomjegyzék

- [1] ONLINE: Number of mobile phone users worldwide from 2015 to 2020, <https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide>
- [2] ONLINE: Everything you need to build on Android <https://developer.android.com/studio/features.html>