



MATEMATIKAI ÉS INFORMATIKAI INTÉZET

Kliens-szerver kommunikáció Android platformon

Készítette

Balajti-Tóth Kristóf

Programtervező Informatikus Bsc

Témavezető

Tajti Tibor

Egyetemi adjunktus

EGER, 2019

Tartalomjegyzék

1. Fejlesztői eszközök	4
1.1. Fejlesztői környezetek	4
1.1.1. Android Studio	4
1.1.2. Pycharm Professional Edition	4
1.2. Postman	4
2. Platformok	5
2.1. A szerver kiválasztása és felépítése	5
2.2. Mobil platform választása	6
3. Felhasznált technológiák	7
3.1. Folyamatos integrálás	7
3.2. Verzió kezelés	7
3.3. Szerveren használt technológiák	8
3.3.1. Flask	8
3.3.2. SQLite	8
3.3.3. Docker	8
3.3.4. Tmux	9
3.4. Androidon használt technológiák	9
3.4.1. OkHttp3	9
3.4.2. Okio	9
3.4.3. Pusher	9
3.4.4. Firebase Messaging	9
3.4.5. Room	9
3.4.6. CodeView	9
3.4.7. Material Design 2.0	9
4. Megvalósított funkciók	10
5. Funkciók a jövőre nézve	11
6. Tapasztalatok	12

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 ¹ 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.

¹ 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.

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.

The screenshot shows the Digital Ocean admin interface for a project named 'InfoSec Adventures'. The 'Resources' tab is active, displaying a list of 'DROPLETS (1)'. A single droplet named 'reversedroid' is shown with the following details:

Image	Size	Region	IPv4	IPv6	Private IP
Ubuntu 18.10 x64	1 vCPUs 1GB / 25GB Disk (\$5/mo) Resize	FRA1	207.154.198.244	Enable	Enable

Below the droplets, the 'DOMAINS (1)' section shows a domain 'reversedroid.infosecadventures.com' with '1 A / 3 NS / 1 SOA' records.

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

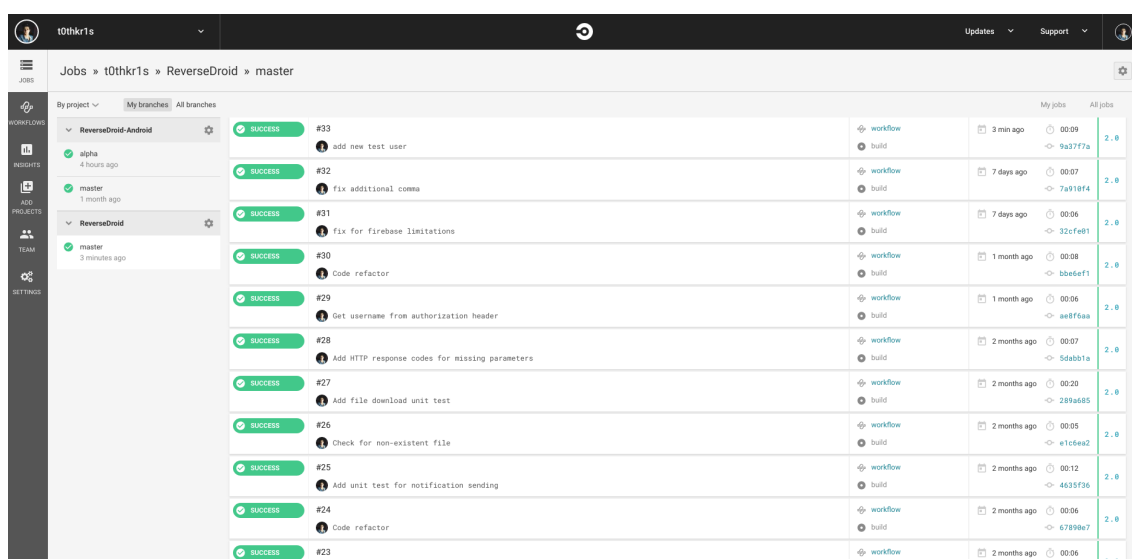
A mobilos operációs rendszerek közül az Androidot választottam. Már korábban sikerült megismerkednem az Android nyújtotta lehetőségekkel és előnyökkel. A többi mobilos operációs rendszerrel ellentétben az Android nyílt forráskódú és a piac több mint felét uralja. Ez annak is köszönhető, hogy 2005-ben 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	
By project	My jobs
ReverseDroid-Android	
alpha	
master	
ReverseDroid	
master	
#33	add new test user
#32	fix additional comma
#31	fix for firebase limitations
#30	Code refactor
#29	Get username from authorization header
#28	Add HTTP response codes for missing parameters
#27	Add file download unit test
#26	Check for non-existent file
#25	Add unit test for notification sending
#24	Code refactor
#23	

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.

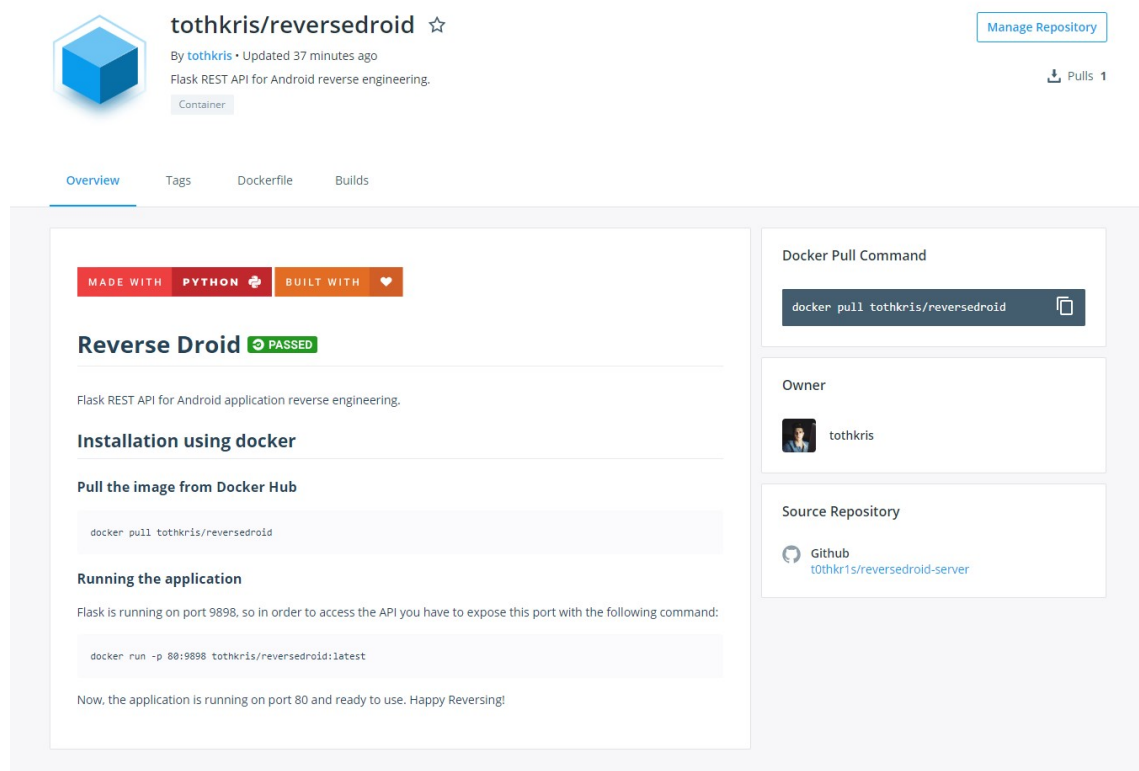
3.3. Szerveren használt technológiák

3.3.1. Flask

3.3.2. SQLite

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

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

3.4.7. Material Design 2.0

4. fejezet

Megvalósított függvények

5. fejezet

Funkciók a jövőre nézve

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.

6. fejezet

Tapasztalatok

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>