Recent Innovations in Mechatronics (RIiM) Vol. 5. (2018). No. SI.
DOI: 10.17667/riim.2018.si/16

Kommunikáció Android okostelefon és Raspberry Pi3-on üzemelő saját MySQL adatbázis között

Molnár Ádám
Informatika Kar
Debreceni Egyetem, Műszaki Kar
Debrecen, Magyarország
molnaradam.dm@gmail.com

Erdei Timotei István
Mechatronikai Tanszék
Debreceni Egyetem, Műszaki Kar
Debrecen, Magyarország
timoteierdei@eng.unideb.hu

***Absztrakt*—Az okostelefonok egyre nagyobb hullámú terjedésének köszönhetően nagyobb igény keletkezik arra, hogy adatainkat saját adatbázisban tároljuk, illetve mindezt mobiltelefonról is elérhetővé tegyük, valamint prezentálható formában tároljuk. Ezen technológia akár nagyvállalati környezetben is könnyen megvalósítható, például adott cégvezetők azonnali, naprakész információt képesek kapni a vállalat jelenlegi állásáról egy Android operációs rendszerrel ellátott okostelefonról. Projektünkben egy egyedi-saját tervezésű adatbázis fut egy Raspberry PI 3-as alaplapon működő webszerveren. Az adatbázis és adatai bármilyen Android alapú mobil eszközről - a megfelelő applikáció birtokában - bármikor, bárhonnan interneten keresztül lekérdezhetőek/frissíthetőek/új rekordok helyezhetőek el.**

Kulcsszavak—*Raspberry, MySQL, PHP, Apache, Android, JSON, Python*

I. BEVEZETŐ

A Debreceni Egyetem Épületmechatronikai Kutatóközpontja ad otthont a különböző fejlesztések számra. A Kutatóközpontban rendelkezésre áll minden fejlesztéshez szükséges infrastruktúra, hogy FPGA alapú komplex rendszerek kerülhessenek kialakításra [7]. Az IoT térnyerését figyelembe véve az új technológiák irányába is van lehetőség nyitni, ennek keretei között került megtervezésre egy IoT alapú FDM típusú 3D nyomtató is, prototípus alkatrészek gyártására [8].

Az egyre globalizálódó világunkban az információ áramlás és annak közlése egyre nagyobb hangsúlyt kap, aminek alappillérévé az Internet vált. Ennek felhasználásával képesek vagyunk adatainkat saját készítésű webszerveren tárolni, és onnan bárhol a világban elérni, mindezt egy alacsony költségű és fogyasztású, hitelkártya méretű számítógépen működtetve [9].

A technológia megvalósításához alapul szolgáló alkalmazás a szerző által szakdolgozatként készített, karakter készítő Android program biztosította a keretet, amelyhez az inspirációt, a M.A.G.U.S. asztali szerepjáték Új Törvény Könyv-e adta, így annak mentén került elkészítésre. Az előnye ennek a koncepciónak, hogy az elkészült project, mivel ellátja az inspiráció tárgyául szolgáló feladatot, így flexibilitása révén más területeken is alkalmazható.

II. TERVEZÉSI SZEMPONTOK

Az Android alapú mobiltelefonok belső memóriájának limitáltsága miatt, valamit a különböző eszközökön futó azonos alkalmazásokból eredő adatok egy helyen való tárolása miatt, fontossá vált egy egyedi webszerver készítése, ahonnan a felhasználó kényelmesen elérheti saját adatait, új adatokat tárolhat, és bizonyos limitált formában az adatok egy részét más felhasználókkal is megoszthassa [10] [11].

Habár a belső memória területe eszközönként változó, illetve mindez SD kártyával bővíthető, ez nem mindenki számára elérhető. Valamint egyes eszközök gyárilag tiltják az SD kártyán való adattárolást, belső memóriájuk pedig elenyésző. Ebből kifolyólag fontos, hogy az alkalmazások minél kevesebb információt tároljanak az eszközön.

A webszerver kialakítására költség, fogyasztás szempontjából, a Raspberry PI 3-as modellje [4], tökéletesen megfelelt a fenti célokra.

1. ábra: Raspberry Pi Jubileumi csomag

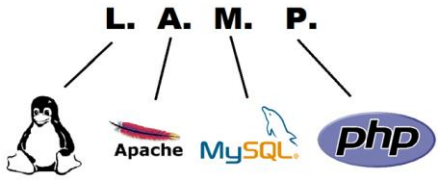
III. RASPBERRY PI MINT WEBSZERVER

Az eredeti elgondolás szerint egy SQLite adatbázis működött volna az eszközön, és mindezt egy dinamikus RESTFUL web service kezelte volna, illetve összekötőként

DOI: 10.17667/riim.2018.si/16

működött volna a mobil és a Raspberry között. Azonban, mivel a Raspberry PI ARM processzorral van felszerelve, ami a RESTFUL web service kialakításához szükséges IDE telepítését nem tette lehetővé, valamint a jubileumi csomag egy mindössze 1GB-os SD kártyával érkezett, így a helytakarékosságot is figyelembe kellett venni.

A helyhiány és a hardware limitáltsága miatt döntöttünk úgy, hogy a leghatékonyabb megoldás egy L.A.M.P. szerver kialakítása lesz.



2. ábra: L.A.M.P. szerver vizuális reprezentációja

L.A.M.P., ahogy a 2.ábrán is látható, Linux, Apache, MySQL, PHP technológiák rövidítéséből adódik [1]. Jelen projekt esetében a Linux disztribúció a Debian alapú Raspbian operációs rendszert jelenti, azon belül is lite verziót, mely nem tartalmaz grafikus felületet (Pixie), ezzel is jelentősen csökkentve az elfoglalt területet.

A Raspberry L.A.M.P. web szerverré való konfiguráláshoz többlépcsős telepítés sorozatot volt szükséges végrehajtani.
Az említett operációs rendszer telepítése után az Apache2 web szolgáló telepítése következett.
Az Apache HTTP szerver egy UNIX és Windows alapú operációs rendszerekre készített http szerver, mellyel megbízható és biztonságos web szerver készíthető.

Sikeres Apache telepítés után következtek a PHP csomagok, a MySQL szerver telepítése [3], illetve a PHP-MySQL könyvtár, mely lehetővé teszi PHP scripteken keresztül az adatbázisban történő querry-k végrehajtását [1].

IV. L.A.M.P. FELMERÜLŐ PROBLÉMÁK

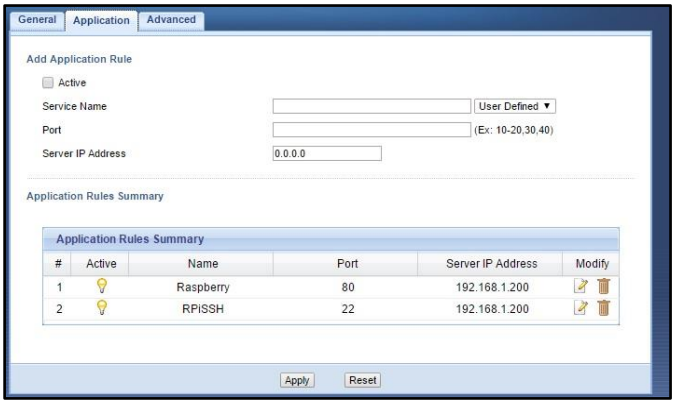
A web szerver sikeres elkészítése után, a web szerver ezen állapotban még csak belső hálózaton belül működött. Ennek a problémának hátterében két fő akadály állt.

Az első az, hogy nem vállalati hálózatról beszélünk, hanem magán hálózaton épült ki a rendszer, ezért dinamikus IP címmel rendelkezik a hálózat. A problémát tovább nehezítette, hogy a szolgáltató hetente változtatja az IP címet, valamint a router minden egyes reset, IP cím változás, vagy esetleges egyéb, előre nem látható hibából kifolyólag minden alkalommal megváltoztatja a hálózathoz kapcsolt összes eszköz IP címét.

A második problémát az alapértelmezett tűzfal okozta, mely meggátolta a külső hálózatról érkező kérések feldolgozását.

A problémák kiküszöbölésére az eszköznek statikus IP címet kellett adni, ezzel gátolva, hogy a router új IP címet adjon neki. A tűzfal probléma megoldására port továbbítást kellett végrehajtani a routeren. HTTP kéréseket tipikus a 80-as porton kezelik, míg az SSH kapcsolódás a 22-es proton történik meg [6].

A megfelelő konfigurációs beállítások után a router WAN címét megcímezve elérhetővé vált az Apache szerver, illetve onnan elérhetővé váltak az eszközön a /var/www/html/ könyvtárban tárolt scriptek is. A 22-es port továbbítás szükségességét az indokolta, hogy bárhonnan, bármikor elérhető kell legyen az eszköz úgy, hogy a rajta történő fejlesztés is biztosított legyen SSH kapcsolaton keresztül.

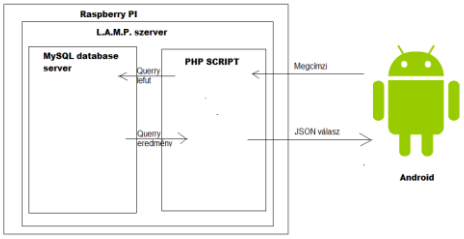


2. ábra: Port továbbítások, Zyxel router-en

A fenti konfigurációkkal a problémák jelentős részét ki is lehetett küszöbölni, viszont még így is gondot okozott a szolgáltató által heti rendszerességgel ütemezett dinamikus IP cím kiosztás. A megoldást a dinamikus DNS jelentette. Egy online dinamikus DNS szolgáltatót, a noIP-t vettük igénybe, mely egy ingyenes szolgáltatás, ami egy domain nevet generál, és egy letölthető program segítségével automatikus frissíti a domain névhez kapcsolt WAN címet. Ennek köszönhetően a quventin.hopto.org cím segítségével elérhetőek lettek a szerver könyvtárában tárolt scriptek. Így folytatódhatott a fejlesztési munka.

V. MŰKÖDÉSI ALAPELV-STRUKTÚRA

A Technológia alapvető működése egy stabil struktúra mentén épül fel. Az Android telefon egy, az AsyncTask osztályt kiterjesztő osztályban egy, a háttérben dolgozó szálon, megcímzi az Apache szerveren található php scriptet. Ennek köszönhetően a PHP scriptben megírt SQL querry-k lefutnak a MySQL szerveren tárolt adatbázisban. A querry-nek amennyiben van produkálható eredménye (tehát select querry-ről van szó és nem insert-ről), akkor a PHP script ezt JSON formájában közzé teszi az Apache szerver adott html webfelületén. Ezt a valid JSON adattömeget az Android telefon eltárolja, feldolgozza és megjeleníti [5].



3. ábra quventin.hopto.org/magus/database/get_hatterek.php script JSON eredménye

VI. MYSQL ADATBÁZIS KÉSZÍTÉSE ÉS FELTÖLTÉSE

A Raspberry-re az 5.5.54 MySQL szerver került telepítésre. Maga a tényleges adatbázis elkészítése és feltöltése terminálon keresztül történt. Lehetséges lett volna a PHPMyAdmin telepítése és konfigurálása a könnyebb adatkezelés végett, de a fentebb említett helytakarékossági okokból ezt a lehetőséget elvetettük.

Adatokkal való feltöltése rendkívül időigényes feladatnak bizonyult. Jelenleg 25 különböző tábla létezik, egyenként több mint 6 különböző tulajdonsággal, helyenként 50-120 rekordot tartalmaz.

```
mysql> select table_name, table_rows from informa
+---------------------------------+------------+
| table_name                      | table_rows |
+---------------------------------+------------+
| fajok                           |          6 |
| fegyverek                       |         69 |
| felhasznalo                     |          2 |
| hasznalati_targyak              |         51 |
| hatterek                        |         24 |
| iskola_alap_kepzettsegek        |        686 |
| iskola_alap_szkepzettsegek      |        134 |
| iskola_hatterek                 |        113 |
| iskola_oktatott_kepzettsegek    |        626 |
| iskola_oktatott_szkepzettsegek  |        181 |
| iskolak                         |         79 |
| istenek                         |         24 |
| jellemek                        |         13 |
| karakter                        |          0 |
| karakter_kepzettsegek           |          0 |
| kasztok                         |         45 |
| kepzettseg_specifikacio         |        218 |
| kepzettsegek                    |         74 |
| korkatergoriak                  |         36 |
| pancelok                        |         16 |
| pszi                            |         48 |
| szakralis_magia                 |        152 |
| szazalekos_kepzettsegek         |         10 |
| szulofoldek                     |         39 |
| tapasztalati_magia              |        262 |
+---------------------------------+------------+
25 rows in set (0.03 sec)
```

4. ábra: Adatbázis táblák és rekord számok

Maga az adatbázis egy relációs adatbázis sémán keresztül lett elkészítve. Az alapvető elv a készítés során az volt, hogy minden egyes alkalmazás használó egy regisztrációs folyamaton megy át az első használatkor. Ekkor, az adatbázis *felhasznalo* táblájában, a regisztráció során megadott információk elhelyeződnek, generálodik számára egy tulajdonos ID, és 0-ra állítódik be a kész karakterek száma. Amennyiben a felhasználó új karaktert készített, a karakter adatbázisban helyeződnek el az adatok, illetve egyedi azonosító generálódik számára és a tulajdonos azonosítója is mellé kerül. Így egy adott karaktert azonosítója alapján lehet a többi táblában azonosítani, és karakter táblából megtudható, ki is a tulajdonosa.

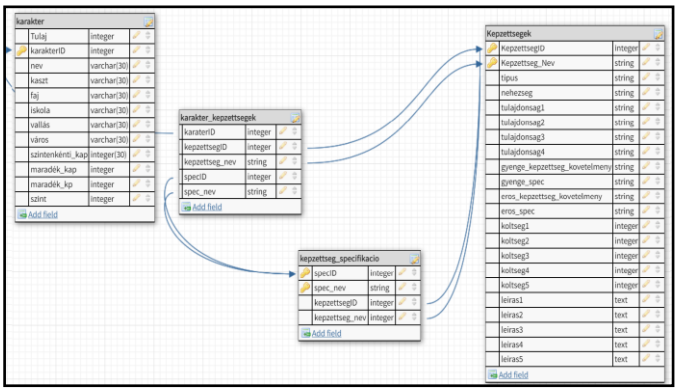
A relációs adatbázis elve szerint több nagy tábla létezik, mely az adott információ tömeg részét tartalmazza. Ezeket a főbb táblákat kevesebb tulajdonsággal, ám jövőben annál nagyobb méretű kapcsolótáblák kötik össze, melyek mindig az az adott karakter azonosítóját, illetve az adott információs táblában elhelyezkedő rekord azonosítóját és nevét tartalmazza.

```
mysql> show tables from karakter_kepzettsegek;
ERROR 1049 (42000): Unknown database 'karakter_kepzettsegek'
mysql> show columns from karakter_kepzettsegek;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| karakterID    | int(11)     | YES  | MUL | NULL    |       |
| kepzettsegID  | int(11)     | YES  | MUL | NULL    |       |
| kepzettseg_nev| varchar(30) | YES  |     | NULL    |       |
| specID        | int(11)     | YES  | MUL | NULL    |       |
| spec_nev      | varchar(30) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)

mysql>
```

5. ábra: karakter_kepzettsegek tábla

Példaként megemlítendő a *karakter_kepzettsegek (6. ábra)* tábla, melyben azok az információk találhatóak, amelyek egy adott karakter birtokolt képzettségeit tartalmazzák. Három táblát köt össze, a karakterek (14 adat), kepzettsegek (22 adat) illetve kepzettseg_specifikaciok (4 adat). Ezt a terjedelmes adatmennyiséget mindössze 5 jellemző alapján tárolja, melyek a másik 3 táblában, mint elsődleges kulcs szerepelnek, és ehhez úgy nevezett foreign key (külső kulcs) segítségével kapcsolódnak (7.ábra). Ennek a kialakításnak köszönhetően a querry-k (lekérdezések) igen egyszerűen megoldhatók táblák összekapcsolásának segítségével (JOIN).



6. ábra karakter_kepzettsegek tábla kapcsolódásai.

Az Android alkalmazás jelenlegi formájában a M.A.G.U.S szerepjáték szabályai alapján történő karakter készítést teszi lehetővé. A karakter készítés során a program az adatbázisból nyeri ki az adatokat, és ezek segítségével készíthető el egy bizonyos karakter. Amennyiben új karakter kerül elhelyezésre, az adatok a megfelelő kapcsolótáblákat töltik fel. Lekérdezés esetén pedig ezekből kinyerhetők a releváns információk.

Az adatbázis keretének elkészítése után a tényleges adatok feltöltése igen időigényes feladatnak bizonyult. A felhasználandó adatok M.A.G.U.S. Új Törvénykönyv című szabálykönyvből lettek importálva. Mivel digitális verzió, ebből a könyvből csak scannelt formátumban létezik, a szövegek és adatok rögzítése több lépcsős folyamattá változott. Először a scannelt verziót feldolgozható adattömeggé kellett konvertálni. Ehhez az ABBY PDF

DOI: 10.17667/riim.2018.si/16

Transformer szoftvercsomagot használtuk, mely kisebb – nagyobb hibákkal sikeresen feldolgozhatóvá tette az anyagot. Probléma adódott viszont, hogy helyenként kosz, por vagy esetleges tintapaca miatt egy oldalon problémát okozott a konvertálás. Ennek köszönhetően minden szöveget ellenőrizni, és az apróbb hibákat javítani kellett. Legtöbb nehézséget az úgy nevezett Pszi diszciplínák fejezet feldolgozása okozta, ugyanis a Ψ karaktert, mely igen gyakran fordult elő a szövegben, nem tudta a program értelmezni. Ennek köszönhetően F, T, Y és egyéb érdekes kombinációk születtek ebből a karakterből. Ezen felül kisebb-nagyobb problémák adódtak ebből, de nem megoldhatatlanok.

A szövegjavítás után a releváns adatokat egyszerű .txt kiterjesztésű szöveges fájlokba mentetettük át, külön a fontosabb értékeket, és egy másik fájlba a hozzá tartozó szöveg halmazokat, melyeket véletlenszerű elrendezésük miatt kézzel kellett alapvető formára hozni. Így minden egyes táblához tartozott kettő, vagy ennek többszöröse számú szöveges dokumentum, melyek *.txt vagy *_text.txt nevet viselték.

A szöveges fájlok UTF-8-as karakterkódolással elmentve, átkerültek a Raspberryre, ahol Linux parancsokat alkalmazva, az automatizálható további formázásokat el lehetett egyszerűen végezni.

*sed -i -e 's/ő/ö/g' -e 's/ű/ü/g' -e 's/Ő/ö/g' -e 's/Ű/Ü/g' *.txt*

A fenti parancsot alkalmazva a szöveges fájlokban lecserélődtek a megfelelő karakterek, nevezetesen minden ő-ből ö, illetve ű-ből ü lett. Ennek a döntésnek a hátterében az állt, hogy az adatbázisból a mobilra küldött adat JSON fájlként jelenik meg, mely csak és kizárólag az UTF-8 karakter kódolást alap esetét ismeri, mely nem tartalmazza a tradicionális latin 2-es betűket. Tesztelések során arra jutottunk, hogy bárhogy konvertáljuk bit szinten a karaktereket, JSON kódolás esetén egy olyan anomália jön létre, mely ő és ű esetén karakterkód változást vált ki, így teljesen más karaktert jelenít meg az Android. Így jutottunk arra az igen egyszerű megoldásra, hogy szimplán kimaradnak az ű és ő betűk a szövegből, mely nem okoz akkora problémát a felhasználónak.

*1. sed -i 's/;;/;/g' *.txt*

*2. sed -i 's/;//g' *_text.txt*

*3. sed -i 's/;/\t;/g' *t.txt*

Az előbb felsorolt három utasítás rendkívül fontos volt a formázás során. Az első minden .txt kiterjesztésű fájlban a pontos vessző duplikációkat veszi ki. A második minden szöveges leírást tartalmazó fájlból eltávolítja a pontos vesszőket. A pontos vesszők adatelhatárolási egységként voltak használva, így könnyebben vizsgálhatók és javíthatók voltak az az adatok. A harmadik utasítás a formázás utolsó parancsa volt, mely minden egyes pontos vesszőt egy tabulátor egységgel helyettesítette, könnyebbé téve ezzel az adatbázisba való beolvasást.

Ezek előtt szükségessé vált az *.txt és *_text.txt fájlok egy fájllá való egyesítése. A formázások során már biztosra vehető volt, hogy a *.txt fájl n-edik sorához a *_text.txt fájl n-edik sora tartalmazza a leírást. Ennek összeolvasására egy Python script megírása volt a megoldás. A Raspbian jessie lite verzója alapértelmezetten tartalmazza a Python 2 fordító csomagjait, így ez külön telepítést nem igényelt.

A Python egy magas szintű nyelv [2], mely támogatja az imperatív, procedurális, objektum orientált, illetve funkcionális paradigmákat. A 8. ábrán található osszerak.py elnevezésű Python script volt a leggyakrabban használt formázó script.

```
#!/usr/bin/python
import sys
import subprocess
p = subprocess.Popen(["ls | grep .txt"], stdout=subprocess.PIPE, shell=T
x, err = p.communicate()
a=0
fname1=""
fname2=""
for betu in x:
        if betu=="\n" and a==1:
                with open(fname1) as f:
                        content=f.readlines()
                content=[x.strip() for x in content]
                f.close()
                with open(fname2) as f:
                        content2=f.readlines()
                f.close()
                kimenet=[a+b for a, b in zip(content, content2)]
                subprocess.call(["rm", fname1])
                subprocess.call(["rm", fname2])
                subprocess.call(["touch", fname1])

                cel=fname1
                file = open(cel,"w")
                for line in kimenet:
                        file.write(line)
                file.close()
                fname1=""
                fname2=""
                a=0
        elif a==0 and betu!="\n":
                fname1=fname1+betu
        elif a==1 and betu!="\n":
                fname2=fname2+betu
        else:
                a=a+1
```

7. ábra osszerak.py forráskódja

A program Linux parancsokat alkalmazva készít új fájlokat, melyek már a kívánt formát veszik fel. Az „ls | grep .txt", subprocess lefuttatása a futtatott mappa összes létező .txt kiterjesztésű fájl nevét kapja válaszul a program. Ebből a név sorozatból kettesével válogatva *.txt és *_text.txt fájlok párosaival dolgozik tovább. Összefoglalva, a script beolvassa mindkettő szöveges állomány tartalmát, majd egy listába mentve ezeket összefűzi soronként.

Ezt követően a két szöveges fájlt eltávolítja a script a tartalmazó mappából, és létrehozz a rövidebb fájlnévvel egy üres szöveges állományt. Végezetül pedig beleírja az immáron konkatenált sorokat. Az így keletkezett adatokra pedig már kiadható a fentebb említett 3. parancs, és ezzel a szöveges fájl formázása lezárult, megfelelő alakot öltve. Természetesen nem ez volt az egyetlen Python script, hiszen sokszor egyedivé kellett változtatni a forráskód bizonyos részeit, hogy megfelelően működjön.

LOAD DATA LOCAL INFILE 'abszolút elérési út' INTO TABLE tábla_név (értékek);

A fenti kód részlet egy SQL querry. Ennek segítségével a szöveges állomány abszolút elérési útjának megadásával, illetve érték beillesztési sorrend megadásával hatékonyan beilleszthető volt az adat az adatbázisba.

DOI: 10.17667/riim.2018.si/16

VII. PHP SCRIPTEK

A PHP scriptek látják el a köztes kommunikációs elemet az Android alapú eszköz és a MySQL adtabázis szerver között. Az adatbázis hozzáféréshez egy külön összekötő elemet kellett telepíteni a telepítés során. Gyakorlatban ez egy külön PHP library volt, melynek segítségével néhány egyszerűbb paranccsal komplex SQL lekérdezéseket hajthattunk végre.

Alapvetően háromfajta PHP script készült az eszközre, melyek különböző formában jelennek meg. Ezek sorrendben a POST, GET illetve a POST és a GET típusú scriptek keveréke a POST-GET. A POST típusú PHP scriptek lényege, hogy egy bizonyos - általában - adat rögzítési lekérdezést hajtanak végre, ám a szükséges lekérdező módosító adatokat külső forrásból nyerik, mely jelen esetben az Android eszközt jelenti, így egy dinamikus SQL querry jön létre, melyek nem rendelkeznek kimenettel. A GET típusú scriptek, a POST-al ellentétben, egy előre rögzített statikus lekérdezést jelentenek, melyek nem igényelnek módosító adatokat külső forrásból. Általánosságban egy tábla összes, vagy előre meghatározott részét, dinamikus specifikáció nélkül kérdezik le.

```php
<?php
$db_name="MAGUS";
$mysql_user="root";
$mysql_pass="Kincses1991";
$server_name="localhost";

$param=$_POST["param"];


$sql = "select kaszt_nev from kasztok where faj_nev='$param';";
$con=mysqli_connect($server_name,$mysql_user,$mysql_pass,$db_name);
mysqli_query($con,'SET CHARACTER SET utf8');
$result = mysqli_query($con,$sql);

$response=array();
while($row = mysqli_fetch_array($result)){
array_push($response,array("elem"=>$row[0]));
}
echo json_encode(array("server_response"=>$response));

mysqli_close($con);
?>
```

8.ábra get_kasztok.php script forrás kódja

A harmadik típusú, vagyis a POST-GET típus (9. ábra), a POST és GET típust keveri. Dinamikusan paraméterezhető SQL querry-t futtat és ennek válaszát JSON formátumban adja vissza az Android okostelefonnak. A fenti PHP fájl futtatásához szükséges egy String típusú változót küldeni http kapcsolaton keresztül. Ezt a küldött paramétert a script értelmezi, eltárolja a $param változóban. Az értelmezett, érkezett paraméteren felül, még 4 konstans szükséges, mely az adatbázishoz való kapcsolódás feltétele. Szükséges az adatbázis neve: MAGUS, a felhasználó: root (alapértelmezett admin, mert külön felhasználókat az adatbázis nem definiál), az adatbázis szerver jelszava, illetve maga a szerver neve: localhost. Az $SQL változóval tárolásra kerül a futtatandó lekérdezés, a megfelelő paraméterrel frissítve, míg a $con változó a mysqli_connect paranccsal, illetve a megadott konstansokkal kapcsolódás hozható létre az adatbázishoz.

Érdemes a script-be megjegyezni kettő fontos parancsot:

1. mysqli_query($con,'SET CHARACTER SET utf8');
2. mysqli_query ($con, $sql);

Az első parancs során kapcsolat jön létre a script és a szerver között, valamint a válasz karakter kódolását utf-8 típusúra konvertálja. Ennek szükségességét az indokolja, hogy a JSON alapvetően csak a sima UTF-8 típusú karaktereket képes értelmezni, ellenkező esetben furcsa anomáliák léphetnek fel a latin2-es karakterkészletben elhelyezkedő karakterek körében.

A második parancs során lefut a lekérdezés, a válasz értelmezésre kerül, és egy tömbben tárolódig el, úgy, hogy minden egyes adatot az „elem:" kifejezés előzi majd meg (amennyiben a lekérdezés válaszának 1 sora több elemből is áll, úgy minden egyes elemet speciális címkével kell ellátni a könnyebb értelmezhetőség érdekében.). A megfelelő formai tárolást követően JSON formátummá kellet kódolni, így a megadott String-et követően az Android eszköz értelmezni tudja a kapott adatokat.

VIII. JSON FELDOLGOZÁSA ANDROID OLDALON

A PHP script válaszát valamilyen módon Android oldalon kellett feldolgozni. Ehhez több megoldás is létezett, például külső könyvtárak importálása, ám a legegyszerűbb megoldásnak egy saját osztály készítését találtuk, melynek szülőosztálya az AsyncTask osztály volt [5].

```
@Override
protected String doInBackground(String... params) {
    String metodus=params[0];
    String faj=params[1];
    switch (metodus){
        case "MAX":
            jsonURL="http://quventin.hopto.org/magus/database/get_maxok.php";

            try {
                URL url=new URL(jsonURL);
                HttpURLConnection kapcsolat=(HttpURLConnection)url.openConnection();
                kapcsolat.setRequestMethod("POST");
                kapcsolat.setDoOutput(true);
                OutputStream os = kapcsolat.getOutputStream();
                BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(os,"UTF-8"));
                String data = URLEncoder.encode("param","UTF-8")+"="+URLEncoder.encode(faj,"UTF-8");
                bw.write(data);
                bw.flush();bw.close();os.close();
                InputStream is = kapcsolat.getInputStream();
                BufferedReader br=new BufferedReader(new InputStreamReader(is));
                StringBuilder json = new StringBuilder();
                jsonString="";
                while((jsonString=br.readLine())!=null){
                    json.append(jsonString+"\n");
                }
                br.close();is.close();kapcsolat.disconnect();
                return json.toString().trim();
            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
```

9.ábra doInBackground forráskód

A 9. ábrán látható forráskód a feladathoz készült alkalmazás forráskódjának egy részét tartalmazza, mely egy POST-GET típusú PHP scriptet címez meg. Alapvetően az AsyncTask-ot öröklő osztálynak 4 metódust kell implementálnia, melyből egyet kötelezően módosítani kell. Ezek a onPreExecute, onPostExecut, onProgressUpdate illetve a doInBackground, az utóbbi a kötelező, a másik három csupán a futtatás előtti, utáni és közbeni esetleges extra működést szolgálják kielégíteni. Az újra felhasználás elvét kihasználva, általánosságban egy activityhez, egy Asynctask osztály tartozott, így az egy activity-n belül történő lekéréseket

egy switch utasítással választottuk el egymástól. A doInBackground metódus úgy nevezett Variadic paraméter sort vár, mely akárhány, akármennyi típusú paramétert tartalmazhat. Általában az első mindig a futtatandó verziója volt a metódusnak, majd ezt követték az adott php scriptben megkövetelt paraméterek száma.

Az utasítás sorozatot kivétel kezelnünk kellett, MalformedURLException illetve a szokásos IOException kivételekre. Az előbbi a hibás URL cím esetén jelez, míg az utóbbi a hibás INPUT/OUTPUT hibák esetén. Egy String változóban megadásra került a megfelelő URL cím, mellyel a PHP scriptet címeztük meg. Ezt használva létesítettünk a HttpURLConnection osztályt alkalmazva kapcsolatot a LAMP szerveren operáló scripttel. A megfelelő paraméterek átadása után, a telefon várakozik a szerver válaszára (PHP lefut, query lefut, PHP formáz, JSON kódol). Habár jelenleg az információk nagysága a másodperc töredék része alatt jelenik meg, elképzelhető, hogy ez hosszasabb időt fog igénybe venni a későbbiekben. Mivel az AsyncTask osztályt öröklő osztály utasításai egy külön szálon, a háttérben futnak, az eredeti szálon futó parancsok szintén haladnak a megfelelő sorrendben. Ezt a problémát igen egyszerűen ki lehetett küszöbölni azzal, hogy a lekérést elindító parancs várjon a doInBackground visszatérő értékére, ám ez hosszabb folyamat esetén felhasználói felület fagyáshoz vezet. Erre a problémára lesz használatos az onProgressUpdate metódus, amiben egy AlertDialog osztályt alkalmazva jelezzük a felhasználónak, hogy az információ lekérés folyamatban van.

```
public void jsonBont(String s,String metodus) throws JSONException {

    JSONObject jo=new JSONObject(s);
    JSONArray jsonArray=jo.getJSONArray("server_response");
    JSONObject jsonObject=jsonArray.getJSONObject(0);
    if(metodus.equals("MAX")){
        maxok[0].setText(jsonObject.getString("ero"));
        maxok[1].setText(jsonObject.getString("gyors"));
        maxok[2].setText(jsonObject.getString("ugy"));
        maxok[3].setText(jsonObject.getString("allo"));
        maxok[5].setText(jsonObject.getString("egs"));
        maxok[4].setText(jsonObject.getString("kar"));
        maxok[6].setText(jsonObject.getString("int"));
        maxok[7].setText(jsonObject.getString("akr"));
        maxok[8].setText(jsonObject.getString("asszt"));
        maxok[9].setText(jsonObject.getString("erz"));
```

10. ábra JSON parser forráskód

A visszatérő érték egy külön metódusban kezelhető (10.ábra), mely a kapott String-et JSON objektumként kezeli, majd ezt egy JSONArray tömbbé konvertálja, és ebből nyeri ki a kapott információkat, a php script által megadott címke alapján, amelyeket az alkalmazás listába, tömbbe vagy egyéb tárolásra alkalmas formátumba elmenti, és később a felhasználó számára megjelenít.

IX. ÖSSZEGZÉS

A projekt végeredménye egy okostelefonról elérhető, dinamikusan feltölthető, illetve adatait lekérhető relációs adatbázis, mely php scripteken keresztül kommunikál a külvilággal. Mindez egy viszonylag olcsó, alacsony fogyasztású, kártya méretű eszközön, mely bárki számára könnyen elérhető technológiává alakult. Ugyanezen technológiát KKV-k kötelékében is elképzelhetővé válik, hiszen alacsony költségvetésnek köszönhetően a kis- és közép vállalkozások is megengedhetik maguknak.

KÖSZÖNETNYILVÁNÍTÁS

Szeretném köszönetem kifejezni a projekt elkészítése alatt adott hasznos tanácsaiért és észrevételeiért Erdei Timotei Istvánnak, a Debreceni Egyetem oktatójának, illetve a Stack Overflow lelkes fórum használóknak, akiknek köszönhetően ismételten bebizonyosodott az a csodálatos IT közmondás, miszerint: „Valakinek, valahol volt már közel hasonló problémája, és megkérdezte Stackoverflow-n”.

A publikáció elkészítését az EFOP-3.6.1-16-2016-00022 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

HIVATKOZÁSOK

[1] P. Kamal, „Definitive Guide to Setting up a LAMP Server using Open Source Software”. Technical Report, October 2015

[2] J. Andress, R. Linn, „Chapter 2 – Introduction to Python”, Coding for Penetration Testers (Second Edition) , 2017, pp: 43-79.

[3] Jan L. Harrington, „Installing and Running MySQL”, SQL Clearly Explained (Second Edition), 2003, pp:3-8.

[4] A. Pajankar, „Introduction to Single Board Cumputers and Raspberry Pi”, Raspberry Pi Image Processing Programming, January 2017, pp 1-24.

[5] Murat Yener, Onur Dundar, „Android Application Development With Android Studio”, Expert Android Studio,,pp 45-79

[6] „How to Forward Ports on Your Router”, (2017, May 14) [Online]Available: https://www.howtogeek.com/66214/how-to-forward-ports-on-your-router/

[7] Szász C, Chindriş V, Husi G, „Embryonic systems implementation with FPGA- based artificial cell network hardware architectures,” ASIAN JOURNAL OF CONTROL 12:(2) pp. 208-215. (2010)

[8] T. I. Erdei, ZS. Molnár, G. Husi, „Selecting Equipment and Supplies for Self–Replicating 3D Printer,” Acta Technica Corviniensis - Bulletin of Engineering, Hunedoara9.1, Jan-Mar 2016, pp. 59-62.

[9] T. I. Erdei, Zs. Molnár, N. C. Obinna, G. Husi, „Cyber physical systems in mechatronic research centre,” MATEC Web Conf. Volume 126, 2017.

[10] T. I. Erdei, Zs. Molnár, N. C. Obinna, G. Husi, „A Novel Design of an Augmented Reality Based Navigation System & its Industrial Applications,” 15th IMEKO TC10 – Technical Diagnostics in Cyber-Physical Era Budapest, Hungary, 6 – 7 June, 2017 - Organised by: MTA SZTAKI – Hungarian Academy of Sciences - Institute for Computer Science and Control.

[11] T. I. Erdei, Zs. Molnár, N. C. Obinna, G. Husi, „Surveillance and Security System in the Building Mechatronics Research Center,” Electrical Engineering and Mechatronics Conference EEMC’16 - 7th-18th-19th March, 2016.