Android table football application development

Tamas Mark Matuz [a]

[a] *OPTEN Kft., 1138 Budapest Dunavirág utca 2-6. Gateway Office Park, 1. torony. 4. em. Hungary, matuz.mark@opten.hu*

Abstract

For quite a few years now, mobile devices have generated more data traffic than desktop devices, which is why I decided to create my own mobile application, which can be used by almost anyone with a phone based on the Android operating system. According to Google, 99% of devices can run within this layer. My goal is to create something unique, useful and new, that's why I decided to create an application related to sports, since it is usually included in people's daily lives. After a little study and research, I decided to make a foosball app, as there are very few similar apps on the market. With the application, I want to encourage young people, or simply people who want to play foosball, to get out of their homes and find the right table for them. After registration, users can already see nearby tables and their ratings with a few clicks. I created the client in Java, and the web application server in PHP. The application relies heavily on the Internet connection, so it cannot store or display data without it. The client communicates with the application server via the HTTP protocol, with JSON objects, as both languages support it. In any case, I set as a primary goal that the data should function and appear in the same way not only on modern devices, but also on older, outdated devices. That's why I set the minimum SDK as low as possible and the maximum as high as possible. My further goals to be achieved include the cleanness and simplicity of the user interface, so I used the official Material Design theme developed by Google. Also, app security was an important consideration for me. That's why I used reliable libraries and safe filters on both the client and server side.

Keywords: *programming; Android; PHP*

Androidos asztalifutball applikáció fejlesztése

Matuz Mark Tamas [a]

[a] *OPTEN Kft., 1138 Budapest Dunavirág utca 2-6. Gateway Office Park, 1. torony. 4. em. Hungary, matuz.mark@opten.hu*

Absztrakt

Jó pár éve a mobil eszközök több adatforgalmat generálnak, mint az asztali eszközök, ezért is határoztam magam el úgy, hogy én is elkészítem a saját mobil applikációmat, amit szinte bárki képes használni, aki rendelkezik Android operációs rendszer alapú telefonnal. Ezen a rétegen belül az eszközök 99% képes futtatni a Google állítása szerint. A célom az, hogy valami egyedit, hasznosat és újat alkossak, ezért döntöttem úgy, hogy egy sporttal kapcsolatos applikációt szeretnék készíteni, mivel ez általában szerepel az emberek mindennapjaiban. Kisebb tanulmányozás és kutatás után úgy határoztam el magam, hogy egy csocsó applikációt készítek, mivel nagyon kevés hozzá hasonló applikáció szerepel a piacon. Az alkalmazással szeretném ösztönözni a fiatalokat, vagy csak szimplán a csocsózni vágyó réteget, hogy kimozduljanak otthonukból és megtalálják a számukra megfelelő asztalt. A felhasználók regisztrálás után pár kattintással már láthatják is a közelben helyezkedő asztalokat és azok minősítését. A klienst Java, míg a webes alkalmazásszervert PHP nyelven készítettem el. Az applikáció erősen támaszkodik az internet kapcsolatra, így enélkül nem képes adatokat tárolni, megjeleníteni. A kliens az alkalmazásszerverrel HTTP protokollon keresztül kommunikál, JSON objektumokkal, mivel mindkét nyelv támogatja. Mindenképp elsődleges célként tűztem ki azt, hogy ne csak a modern eszközökön, hanem a régebbi elavult eszközökön is ugyanúgy működjenek és jelenjenek meg az adatok. Éppen ezért a lehető legalacsonyabbra tettem a minimum

SDK-t és a legnagyobbra a maximumot. A további megvalósítandó céljaim közé a felhasználói interfész letisztultsága és egyszerűsége tartozik, így a hivatalos Google által fejlesztett Material Design témát alkalmaztam. Ezenkívül fontos szempont volt számomra az alkalmazás biztonsága. Pont ezért használtam megbízható könyvtárakat és biztonságos filtereket mind kliens mind szerveroldalon is.

Kulcsszavak: programozás; Android; .PHP

1. Bevezető

Manapság az emberek jelentős része aktív szociális életet él, részt vesz családi kirándulásokon, sportol barátaival vagy szórakozik ismerőseivel. Ugyanakkor van egy csoport, akik az interneten keresik a közösséget, keveset tartózkodnak a szabadban. A fejlesztendő applikáció célja, hogy ösztönözze az embereket az aktívabb életmódra, lehetőséget nyújtva találkozókra és mozgásra. Az app fiatalabbaknak, szórakozni vágyóknak szól, emellett olyanoknak is, akik kedvelik a csocsót. Az Android platformra fejlesztett alkalmazás térképen mutatja be a közelben lévő csocsó asztalokat, melyekhez adatokat is lehet kapcsolni. A térkép segít a felhasználóknak könnyen megtalálni és eldönteni, melyik asztalt érdemes felkeresni.

A cikk bemutatja az alkalmazás fejlesztését mind a szerver, mind a kliens oldalról. A kliens részt az Android Studio segítségével Java nyelven kerül létrehozásra, míg az alkalmazásszervert PHP segíti, MySQL adatbázisban tárolva az információkat. Tervezésre kerül egy webes felületet is, amely lehetővé teszi böngészőből történő hozzáférést. A Google Maps API segítségével kerül megjelenítésre térkép, amely a böngészők mellett más platformokra is könnyen adaptálható.

Az alkalmazás tehát az emberek aktívabb életmódjának ösztönzését tűzte ki célul a csocsót szerető emberek körében. A fejlesztés során a kliens és szerver oldali technológiák, mint a Java, PHP és MySQL, valamint a Google Maps API fontos szerepet játszanak. Az alkalmazás nem csak a mobil platformra korlátozódik, hanem jövőbeli terv a webes és iOS verzió is, annak érdekében, hogy minél szélesebb körben elérhető legyen mindenki számára.

2. Felhasznált technológiák

A projekt elkészítése során rengeteg különböző szoftver került felhasználásra, ezek közül szeretném kiemelni a legfontosabbakat és néhány olyan információt közölni róluk, ami szükséges a projekt további megismeréséhez. Legtöbbjük ingyenesen letölthető és

használható, habár néhány közülük fizetős, de helyettük lehet más hasonló olcsóbb, vagy akár ingyenes programot is használni.

2.1. *MySQL*

A MySQL a legnépszerűbb nyílt forráskódú adatbázis, amely a web-alapú alkalmazások terén vezető adatbázis nyelvvé vált. Olyan nagyvállalatok, mint a Facebook, Twitter, YouTube és Yahoo!, azáltal hogy megbízhatóságban, teljesítményben és kezelhetőségben már bizonyított, aktívan használják ezt az adatbázisrendszert. Az Oracle vállalat irányítja a MySQL fejlesztéseit, és olyan új lehetőségeket teremt, amelyek támogatják a következő generációs webet, valamint a felhő-alapú, mobil és beágyazott alkalmazásokat. (D. Nichter, 2021)

2.2. *ADOdb könyvtár*

Az ADOdb egy gyors és könnyen használható absztrakciós réteg a PHP programnyelvhez. Ez a réteg lehetővé teszi, hogy ugyanazt a kódot felhasználják az adatbázisok széles köréhez való hozzáférés során. A projektet azóta, hogy 2000-ben létrejött, aktívan karbantartják, és nem csak az alkotója, hanem számos közösségi tag is részt vesz a fejlesztésében. Az ADOdb különböző összetevőket tartalmaz, amelyek megkönnyítik az adatbázisok lekérdezését és frissítését. Emellett tartalmaz egy objektum-orientált Active Record könyvtárat, séma kezelőt és teljesítményellenőrzőt is. (R. Nixon, 2014)

2.3. *Volley*

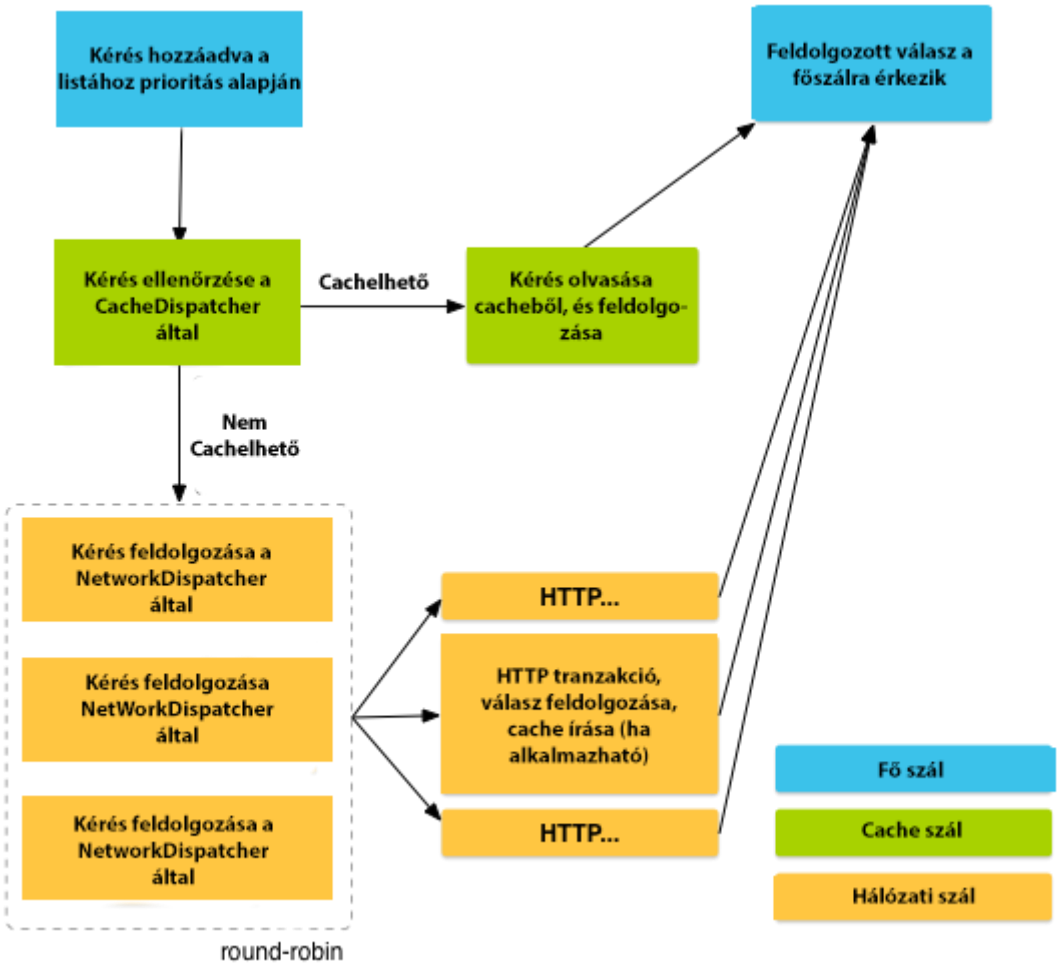
A Volley egy HTTP könyvtár, amely megkönnyíti az Android alkalmazások számára a hálózati események kezelését, és ami ennél is fontosabb, felgyorsítja azokat. (Gerber, A., Craig, C., & Selvaraj, D.,2015)

A könyvtár az alábbi előnyöket nyújtja:

1. Automatikus ütemezés a hálózati kérelmeknek.
2. Párhuzamos hálózati kapcsolatok kezelése.
3. Memória-gyorsítótár használata a szabványos HTTP gyorsítótár-kezeléssel.
4. Lehetőség a kérelmek prioritizálására.
5. Egyszerű vagy több kérelem törlési lehetőség.
6. Hibakeresési eszközök támogatása.

Az 1. ábrán a kliens valamilyen kérést indítványoz a szerver felé, amely egy úgy nevezett kéréssorba kerül prioritás alapján. Ezután, ha a kérés szerepel a gyorsítótárban akkor az adatokat ebből olvassa ki, majd dolgozza fel, ellenkező esetben tovább küldi a NetworkDispatcher-nek. A szerverre megérkeznek a kérések, ezeket feldolgozva visszaküldi a kliensnek, amely a fő szálra érkezik. A gyorsítótár szálból is a fő szálba érkezik az adatfeldolgozás után.

A projektben kevés helyen használható a gyorsítótárazás, mivel minimális a statikus adatok száma. Tehát, ha az egyik felhasználó megnéz egy adott asztalt akkor egy másik felhasználó ugyanazt fogja látja, amit az előző. Ez tökéletes lenne, ha nem lennének dinamikus adatok. Ha módosítanak például a hely nevén, akkor a cache miatt ez nem lenne látható. Az ilyen eseteket még egyszerűen lehet kezelni, ugyanis asztal módosítása során kitöröljük a cache-t és minden működni fog tovább, azonban a szavazás miatt mégsem tökéletes. Ezek az adatok nagyon gyakran módosulnak, nem érdemes gyorsítótárba elhelyezni.



1. ábra: Kérések feldolgozási folyamata

2.4. *WAMP*

A WampServer egy Windows hálózati webfejlesztői környezet, amely lehetővé teszi az Apache, PHP és MySQL alkalmazások létrehozását. Emellett tartalmazza a PhpMyAdmin plugint, ami nagyban megkönnyíti az adatbázishoz való hozzáférést. A WAMP-ot előnyben részesítem, mert ingyenesen elérhető és könnyen kezelhető felületet kínál. Ráadásul széles körű konfigurációs lehetőségeket kínál, amelyek nélkülözhetetlenek a projekt szempontjából. Természetesen más hasonló szoftvereket is használhatnánk, mint például a LAMP vagy a XAMPP (az előbbi csak Linux rendszeren, az utóbbi pedig platformfüggetlenül elérhető). A WAMP telepítése során az Apache, PHP és MySQL komponensek külön telepítése nem szükséges, mivel ezt a WampServer automatikusan elvégzi. Gyakorlatilag ez a program megkönnyíti a hálózati részt, nélküle (vagy egy másik hasonló program nélkül) nem lennénk képesek kommunikálni a klienssel.

2.5. *PhpMyAdmin*

A PhpMyAdmin egy ingyenes szoftver, amely a PHP programozási nyelven készült annak érdekében, hogy megkönnyítse a MySQL adatbázis adminisztrálását webes felületen keresztül. Képes számos MySQL művelet támogatására, különösen az adatbázisok, táblák, oszlopok, relációk, indexek, felhasználók és engedélyek kezelésére. Ezeket a műveleteket a felhasználói felületen keresztül is végrehajthatjuk, vagy közvetlenül SQL parancsok formájában futtathatjuk le.

2.6. *Postman*

Az elérni kívánt API végpontokkal rendelkező szerverrel való kommunikáció egy API kérés segítségével valósul meg, majd ezek a kérések különböző műveleteket végeznek. Ezeket a műveleteket a HTTP metódusok végzik. A leggyakoribb metódusok a GET, POST, PUT és DELETE. A metódusok nevei önmagukért beszélnek: a GET segítségével adatokat lekérdezünk a szerverről, a POST-tal hozzáadunk adatot egy meglévő fájlhoz a szerveren, a PUT-tal cseréljük ki egy meglévő fájlt a szerveren, a DELETE pedig adatokat töröl a szerverről. A Postman szoftver leegyszerűsíti az API kérések küldését. Nem szükséges parancssor vagy terminál használata a kérések teszteléséhez, mivel egy intuitív grafikus felületet biztosít, ami könnyen használható.

2.7. *TortoiseSVN*

A TortoiseSVN egy könnyen kezelhető, nyílt forráskódú verziókezelő és forrásellenőrző szoftver, amely az Apache Subversion alapján működik és kifejezetten a Windows operációs rendszerre lett fejlesztve. A TortoiseSVN által nyújtott felhasználói felület könnyen érthető és használható. A szoftvert a GPL (General Public License) alapján fejlesztették, ami azt jelenti, hogy teljesen ingyenesen használható mindenki számára, beleértve a vállalatokat is, és nincsenek korlátozások a használatára. Mivel ez nem egy beépített funkciója semmilyen IDE-nek, mint például a Visual Studio vagy az Eclipse, a TortoiseSVN-t bármelyik fejlesztői környezetben és bármilyen típusú fájlokhoz használhatjuk.

2.8. *Photoshop*

Az Adobe Photoshop egy rasztergrafikus képszerkesztő és fotófeldolgozó szoftver, amelyet az Adobe Systems fejlesztett. A rasztergrafikus képek képpontokból (pixelekből) állnak, és az állományok minden egyes képpontjának színét rögzítik. A rasztergrafika kihasználja azt a sajátosságot, hogy bizonyos méret alatt az emberi szem nem észleli a pontokat, hanem folytonos színes felületeket lát. A rasztergrafika legfőbb előnye abban rejlik, hogy valósághű képek készítésére alkalmas módon működik, amelyhez nem szükséges olyan hosszú idő, mint egy festmény elkészítéséhez.

2.9. *NetBeans*

A NetBeans IDE gyors és egyszerű fejlesztési platformot nyújt Java asztali, mobil és hálózati alkalmazásokhoz. Támogatást nyújt továbbá JavaScript, HTML és CSS nyelvekkel írt HTML5 alkalmazásokhoz is. Ezen felül kiváló eszközökkel szolgál a PHP és C/C++ fejlesztők számára is. Az IDE ingyenes és nyílt forráskódú, és széles körű felhasználói és fejlesztői közösség támogatja világszerte.

2.10. *PHP*

A PHP egy elterjedt, nyílt forráskódú szerveroldali scriptnyelv, amely leginkább webfejlesztéshez alkalmazzák. PHP kódot lehet beágyazni közvetlenül HTML kódba is. A PHP-t megkülönbözteti más kliensoldali scriptnyelvektől (például JavaScript), mivel a kód a szerveren fut, kigenerálva a HTML kódot, amelyet csak ezután küld a kliensnek. A kliens csak a script eredményét kapja meg, de nem látja az alapul szolgáló kódot. A PHP használatában a legkiemelkedőbb tulajdonság az egyszerűség, ami még a kezdők számára is

könnyen kezelhetővé teszi. Emellett számos korszerű funkciót is tartalmaz. Bár a PHP fejlesztés főként a szerveroldali scriptekre fókuszál, sok más célra is felhasználható. Például alkalmas parancssoros scriptek létrehozására és asztali alkalmazások fejlesztésére is. (R. Nixon, 2014)

2.11. *Android platform*

Az egyre összetettebb és globalizált világban kiemelten fontos a kooperatív eszközhasználat szerepe. A mobil eszközök növekvő jelentősége komoly versenyt gerjesztett a technológiai óriások között, mint például a Symbian, Google, Microsoft, Apple és Nokia, akik a mobil platform piaci részesedéséért versengtek. Az Android, mint nyílt forráskódú mobil platform, radikális változásokat hozott, hiszen nem jár előfizetési költséggel, emellett pedig új lehetőségeket teremtett a mobilfejlesztés terén, számos előnyt nyújtva a versenytárs platformokkal szemben. Napról napra új funkciókkal, alkalmazásokkal és felületekkel gazdagodik, amelyekkel a felhasználók interakciózhatnak. Az Android továbbra is globálisan a legelterjedtebb operációs rendszer marad, amelyet világszerte több százmillió ügyfél használ, ezt támasztják alá a piaci és felhasználási részesedési adatok. A komplex szoftvercsomag magába foglalja az operációs rendszert, köztes rétegű szoftvereket és kulcsfontosságú alkalmazásokat is.

2.12. *Android Studio*

Az Android Studio hivatalos integrált fejlesztői környezet, amelyet az Android alkalmazások fejlesztésére használnak, és az alapjai az IntelliJ IDEA-n alapulnak. Az IntelliJ IDEA erőteljes kód szerkesztőjére és fejlesztői eszközeire építve az Android Studio kibővíti a termelékenységet az Android alkalmazások fejlesztése során. Néhány ilyen funkció a következők: (Gerber, A., Craig, C., & Selvaraj, D.,2015)

- Rugalmas Gradle-alapú rendszer, amely lehetővé teszi a hatékony építési folyamatokat.
- Gyors és gazdag funkciókkal ellátott emulátor, amely valós készülékek szimulálását teszi lehetővé.
- Egységes fejlesztői környezet, amelyben az összes Android verziót könnyedén fejleszthetjük.
- Azonnali futtatás lehetősége, ami kisebb változtatások esetén újraépítés nélkül teszi lehetővé az APK futtatását.

- Kód sablonok és GitHub integráció, amelyek segítik a közös alkalmazásfunkciók és mintakód importálását.
- Széles körű tesztelési eszközök és keretrendszerek a minőségi alkalmazások fejlesztéséhez.
- C++ és NDK támogatás a natív fejlesztési lehetőségekért.

2.13. *Java*

A Java egy programozási nyelv, amelyet különböző platformokra készített szoftverek létrehozásához használhatunk. Amikor egy programozó Java alkalmazást fejleszt, a fordított kód (bájtkód) a legtöbb operációs rendszeren futtatható, ideértve a Windows-t, a Linuxot és a Mac OS-t is. A Java szintaxisa C és C++ programozási nyelvekből eredeztethető. A Java-t a 90-es évek közepén James A. Gosling fejlesztette ki, aki a Sun Microsystems egykori szakembere volt. Később az Oracle vállalat felvásárolta a Java-t. (B. Eckel, B., 2003)

3. Program tervezése

A projekt előkészületei során elengedhetetlen a projekt alapkövetelményeinek, határidejének és a program feladatainak, függőségeinek tisztázása.

A projekt egyben szakdolgozat is volt, így teljes egészében az én tervezésem és fejlesztésem alatt állt. Emiatt a feladatokat úgy kellett megterveznem, hogy azokat egyedül végezhessem el. Ezért csak az időt lehetett módosítani a feladatok tervezésekor.

Elméletben a projekt végső határideje megegyezett a szakdolgozat leadásának időpontjával, de a gyakorlatban a programot jóval előbb kellett befejezni, hogy elegendő idő maradjon a dokumentáció elkészítésére. Próbáltam egyenletesen osztani a feladatokat hetente, így folyamatos előrehaladást értem el a projekten. De a valóságban a helyzet másként alakult, mivel az Android platform teljesen új volt számomra. Emiatt néhány kliens oldali feladatnál késések történtek, de ezt az időt sikerült korrigálni a szerver oldalon. A tervezési fázisban eldöntöttem, hogy a kliens és a szerver részt párhuzamosan fejlesztem, nem külön-külön. Így sok hiba azonnal kiderült.

A tervezés elején, miután meghatároztam az alkalmazás általános keretét, kijelöltem a feladatokat, figyelembe véve, hogy azokat egyedül fogom elvégezni. Olyan munkát választottam, amelyet egyedül meg tudok valósítani, hogy a többi feladat ne szenvedjen és a projekt egésze is zavartalan maradjon.

A szoftver logikájának tervezésekor gondoltam a használható technológiákra, könyvtárakra és eszközökre. Nagyon fontos volt, hogy megbízható, fejleszthető és könnyen integrálható könyvtárakat válasszak. Például, ha a projektet más platformra is tervezzük, akkor a kiválasztott könyvtárnak kompatibilisnek kell lennie a másik platformmal is. Emellett olyan könyvtárakat választottam, amelyeket más projektekben is könnyen lehet felhasználni és amelyeket a támogatásuk nem szűnt meg.

A design tervezésére nem fordítottam túl sok időt, mivel úgy gondoltam, hogy ez nem a projekt legfontosabb része. Az egyszerűség és a letisztultság mellett azonban ügyeltem arra, hogy a Material Design irányelveit kövessem. A dizájnt a Photoshop segítségével készítettem el, majd az Android Studio automatikusan generálta a különböző rétegeket. Döntöttem úgy, hogy nem támogatom a döntött képernyőt, mivel ez komoly tervezési és adatvesztési problémákat okozott volna a kliens oldalon.

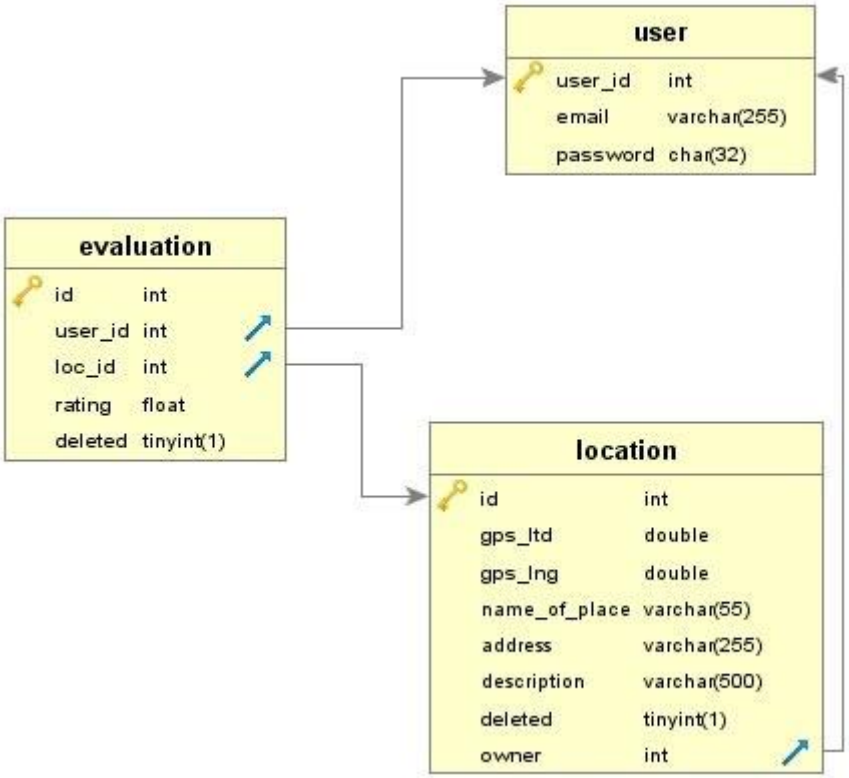
Az adatbázis tervezésénél a fejleszthetőségre is ügyeltem, minden táblának létrehoztam egy elsődleges azonosítót, hogy szükség esetén könnyen hivatkozni lehessen más táblákra.

3.1. *Adatbázis szerkezete*

Az alkalmazás jelenleg erősen támaszkodik a hálózati kapcsolatra, tehát csak internetkapcsolat használatával használható. Ennek következtében elegendő a szerveroldali adatbázis, és nem szükséges lokális adatbázis, mivel az alkalmazás jelenleg nem képes szinkronizálásra.

Bár más megközelítés is lehetséges lett volna, például az adatok tárolása fájlokban, azonban ez kevésbé hatékony és megbízhatatlan megoldás lenne, emellett pedig felhasználóbarát sem. Ennek eredményeképpen inkább a relációs adatbázis mellett döntöttem, mivel jelenleg ez a legalkalmasabb forma az adatok tárolására.

Az adatbázis három összekapcsolt táblából áll, amelyek egymással kapcsolatban vannak. Az adatbázis szerkezete a 2. ábrán látható.



2. ábra: Az adatbázis szerkezetét illusztráló ábra

4. Implementáció

4.1. *Szerver*

A szerver mindig PHP kódot alkalmaz az adatok feldolgozására, majd néhány adatbázis művelet (nem minden esetben történik adatbázis módosítás) után válaszüzenetet küld vissza, amelyet a kliensnek kell feldolgoznia. Célom az volt, hogy minimalizáljam a kliens oldalon végzendő műveleteket, és a legnagyobb részt a szerver végezze, majd csak az eredményt továbbítsa. Néhány műveletet azonban ésszerű okokból a kliens oldalon is elvégzünk. Ilyen például a validáció, amit logikus itt elvégezni, mivel így csökkenthetjük a szerver terhelését és a felhasználó adatforgalmát. Azonban a bejövő adatok érvényességét természetesen a szerveren is ellenőrizzük, hiszen továbbra is lehet nem kívánt adat támadóktól.

A szerveren tárolt kódokat logikailag négy csoportba sorolhatjuk:

- Az autentikációt végző kódok.
- Az alap adatbázis műveleteket végző kódok.
- Az adatok kimutatását kezelő kódok.
- Gyakran használt függvények.

4.1.1. Autentikáció feldolgozása

Az autentikáció folyamán csupán egy típusú kérés metódust használtam. A kliens minden autentikációval kapcsolatos műveletnél POST metódust küld. Az autentikáció fogalmába a bejelentkezés, regisztráció és az automatikus bejelentkezés is beleértendő. A bejelentkezés és regisztráció műveletek kizárólag gombnyomásra hajtódnak végre. A jelszót md5() algoritmussal hasheljük, majd az emailt és a jelszót ellenőrizzük az adatbázisban. A regisztráció során új rekord kerül hozzáadásra, amennyiben az email cím még nem foglalt, egyébként pedig a felhasználó hibaüzenetet kap.

4.1.2. Alap adatbázisműveletek elvégzése

Ebbe a részbe tartozik minden olyan logika, amely a felhasználó asztalának módosítására, beszúrására és törlésére szolgál. Ezen felül a szavazatok kezelését is ezen szakasz végzi. Az alkalmazott HTTP metódusok a kérések típusától függnek.

Általánosan elmondható, hogy új asztal beszúrásakor és a szavazatok leadásakor POST, míg asztal szerkesztése vagy törlése esetén a PUT metódust alkalmaztam. Érdemes megjegyezni, hogy bár a törlés logikailag a DELETE metódusba tartozna, nálunk csupán logikai törlést végzünk, amely az adatbázis frissítését jelenti, így a törlést a PUT metódusba helyeztem.

Ezenkívül ebben a szakaszban gondoskodunk a szerkesztés és beszúrás során kapott BASE64 alapú kódok képké alakításáról és feltöltéséről. Az összes kép a szerveren tárolódik, ahol a neve az adott hely azonosítójából származik, és a fájlformátuma JPEG. Ugyan lehetséges lenne a képeket szövegként tárolni az adatbázisban a "location" táblában, azonban így nőne az adatbázis mérete fölöslegesen. Kép módosítása esetén a régi kép törlésre kerül a szerverről, majd a frissített kép kerül feltöltésre ugyanazzal a névvel.

A szavazás lebonyolításához kihasználjuk az adatbázis és az ADOdb adta lehetőségeket. Fontos kiemelni, hogy minden felhasználó csak egyszer szavazhat, és a szavazat később már nem változtatható meg vagy törölhető. Amennyiben egy felhasználó, aki már korábban szavazott, újra próbálna szavazni, hibaüzenetet kap, mivel nem lehet ugyanazzal a felhasználói azonosítóval új rekordot létrehozni az értékelés táblában

4.1.3. Adatok kimutatása

Ez a szekció csupán adatokat lekérdező műveleteket tartalmaz, nem végzi módosítást az adatbázison. A kimutatásokat három kategóriába sorolhatjuk:

- Egyetlen hely lekérése
- Összes saját hely lekérése
- Összes hely lekérése

Minden három esetben csak a publikus (nem törölt) adatokat szolgáltatjuk a kliensnek JSON objektumok formájában.

Az utolsó esetben nincs szükség a felhasználói azonosítóra, mivel nem szűrünk a felhasználók alapján; az összes felhasználó által hozzáadott helyeket szeretnénk lekérdezni. A lekérdezés során lehetőség van szűrni értékelés, távolság és hely neve alapján. Különösen a távolság kiemelendő, a többi paraméter egyszerűen használható paraméterek a lekérdezésben. A távolságot matematikai számításokkal határozzuk meg, amihez szögfüggvényeket használunk. Az asztalok helyzetét az adatbázisból szerezzük be, a felhasználó helyzetét pedig a kliens adja meg. Az adatbázis képes kezelni a szögfüggvényeket, így ez alapján vissza tudjuk adni a kért adatokat a kliensnek. Az adatokat két különböző módon jeleníthetjük meg, amelyekről a kliens fejezetben található egy ábra.

A második esetben egyértelműen szükségünk van a felhasználó azonosítójára (nem a felhasználó által küldött azonosítóra, hanem a korábban említett munkamenetből származóra), mivel csak az ő adatait szeretnénk lekérdezni.

Az első eset egy különleges helyzet. Általában egy adott hely adatainak lekérdezésére használjuk, ahol a tulajdonos pontszáma nem jelenik meg (csak az átlagos pontszám), de nem csak erre a célra használjuk. Például a szerkesztés folyamata során is alkalmazható, ahol szükség van a saját értékelésre. Ennek lekérdezésére segédlekérdezést használhatunk, amely visszaadja az értékelést, vagy rosszabb esetben NULL-t, ami nem okoz problémát, mivel a szerkesztésen kívüli funkciók továbbra is megfelelően működnek.

4.2. *Kliens*

A kliens oldalt a következő csoportokra lehet felosztani szerepük alapján:

- Activity
- Fragment
- Segéd- és főosztályok, adapterek
- Manifest
- Építő fájlok

Az utolsó két csoport, a Manifest és az Építő fájlok automatikusan létrejönnek, amikor az IDE új projektet hoz létre, és ezeket semmiképp sem szabad törölni, mivel nélkülük az alkalmazás fordítása nem lehetséges. Azonban később ezeket a fájlokat is módosíthatjuk szükség esetén. A többi csoportot, az Activity-t, Fragment-et és a Segéd- és főosztályokat, adaptereket a projekt céljának megfelelően mi magunk készítjük el.

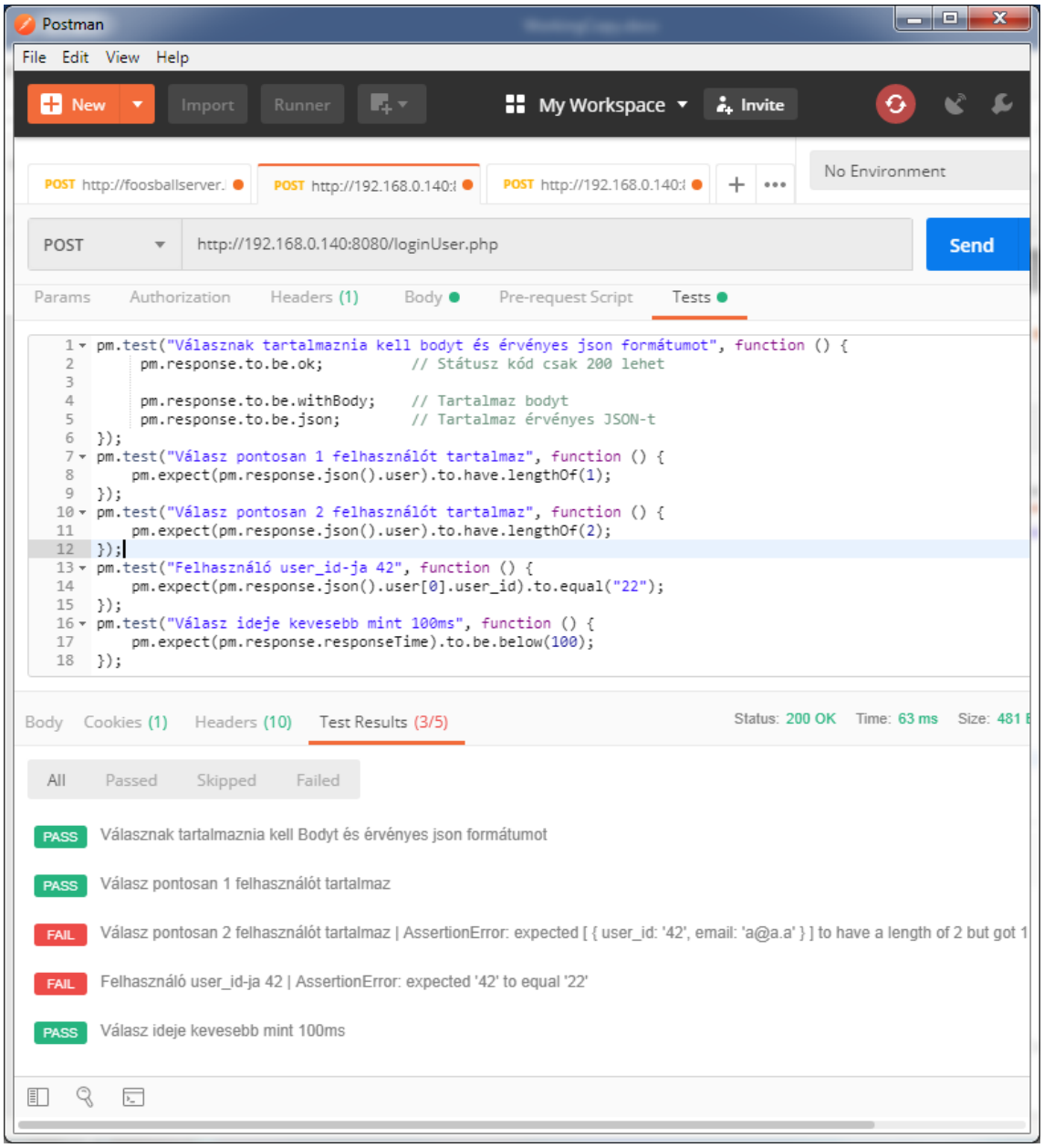
Az korábban már említettem, hogy a kliens minden esetben olyan eszköz lehet, amely Android operációs rendszerrel rendelkezik. Azonban ehhez számos előfeltétel szükséges, köztük az engedélyek megléte is.

5. Tesztelés

Szoftver fejlesztés közben és után is rendkívül fontos a tesztelés. Különösen Android alkalmazások esetében számos kimenetelt kell ellenőriznünk, főleg, ha nem csupán offline alkalmazást hozunk létre. Vegyük figyelembe, hogy nem csupán egyetlen Android verzióra tervezünk, hanem szinte az összes elterjedtre. Érdemes különválasztani a kliens és a szerveroldali tesztelést. Lényeges kiemelni, hogy a tesztelés nem a hibamentességet mutatja be, hanem a hibák jelenlétét felfedi. A teszteket nem automatizáltam, minden lépést manuálisan végeztem el.

5.1. *Kérések tesztelése*

Ehhez a célhoz a korábban már említett Postman funkcióit használtam. A tesztek elkészítéséhez némi JavaScript ismeret is szükséges, mivel ezen a kliensoldali scriptnyelven írhatjuk meg saját tesztjeinket. Egy egyszerű példán keresztül szeretném bemutatni, hogy hogyan tesztelhetünk kéréseket a Postman alkalmazás segítségével (3. ábra).



3. ábra: Kérések tesztelése Postman segítségével

Az ábrán szemléltethető öt teszt kódja és az ezek eredményei láthatók, amelyek közül három sikeresen teljesítette a tesztelést, míg kettő megbukott.

Az első teszt csak akkor tekinthető sikeresnek, ha a válasz tartalmaz testet és érvényes JSON formátumban van. Ehhez a teszthez külön megszorításokat is rögzítettem, ami azt jelenti, hogy a teszt kudarcra is vezethet. Ha például az HTTP státuszkód nem 200, akkor a teszt is meghiúsul.

A következő két teszt közül csak egy képes sikeresen teljesülni, mivel mindkettő azonos logikát használ, csak különböző paraméterekkel (felhasználók száma) hajtja végre. Emellett

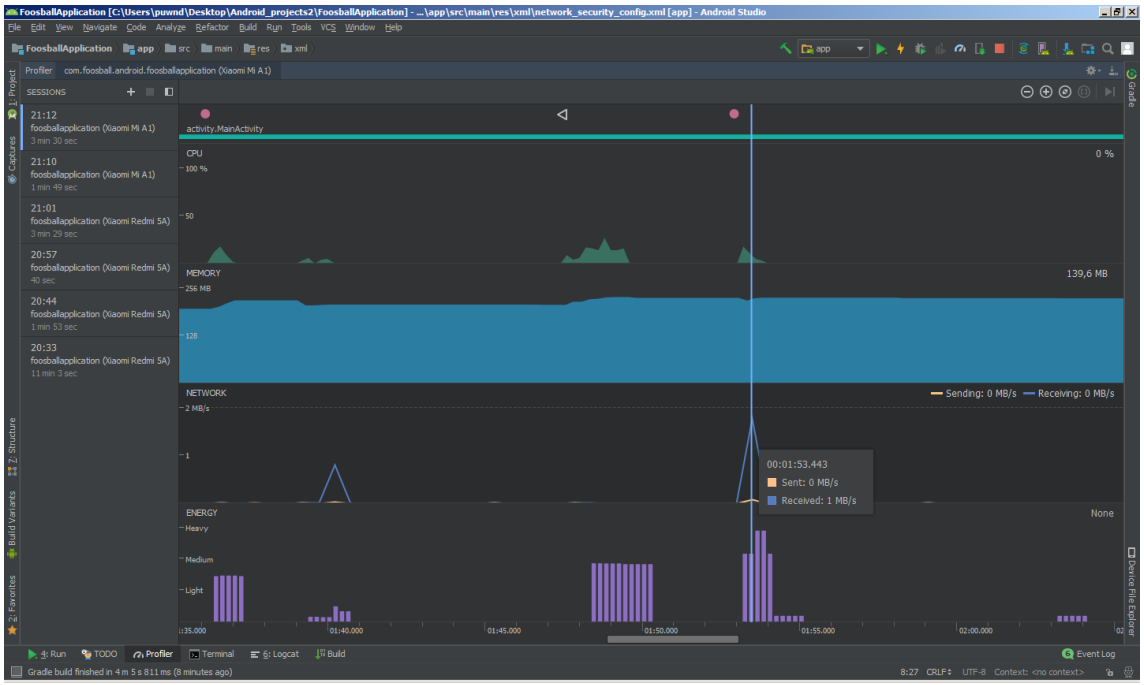
minden esetben csak egy vagy nulla felhasználóval térhet vissza, így a harmadik teszt mindig megbukik.

A negyedik teszt egyszerűen a felhasználó azonosítóját ellenőrzi. A diagramból egyértelműen látható, hogy a sikertelen tesztek milyen hibaüzenetekkel társulnak, mivel a sikertelenség több okból is fakadhat.

Az utolsó teszt rendkívül egyszerű, az időválaszidőt ellenőrzi, ami nem lehet 0.1 másodpercnél kevesebb. Ezen teszt esetében nem szükséges JSON válasz, mivel erre nincs megszorítás. Egyértelműen érthető, hogy a különféle kérésekhez számtalan tesztesetet állíthatunk fel. A tesztelés folyamán logikus sorrendet megállapítani fontos, főleg nagyobb méretű tesztek esetén. Fontos megemlíteni, hogy a diagramon megjelenő érvényes URL minden esetben érvényes JSON választ ad (vagy üres választ, vagy olyat, amely tartalmaz felhasználót). Ha a URL-t például érvénytelenre változtatnánk, az nem csak az első tesztet, hanem logikailag az aztól függő összes tesztet meghiúsíthatja, mivel a tesztek feltétele, hogy érvényes JSON válasz érkezzen.

5.2. *Teljesítmény mérés*

Unit tesztet nem hajtottam végre, ugyanis a fejlesztés során folyamatosan teszteltem a különböző eseteket, így különálló unit tesztek készítését nem végeztem el. A teljesítményt a szoftver saját, erre kifejlesztett eszközeivel vizsgáltam meg. Sajnos a részletes eredményeket csak Android 8.0 verzió felett lehetett pontosan elemezni. A szoftver képes volt a processzor- és memóriahasználatot, valamint a hálózati adatforgalmat és az energiafogyasztást is rendkívül pontos és hasznos információkkal szolgálni.



4. ábra: Teljesítmény tesztelése futási időben

A 4. ábráról leolvasható, hogy több eszközzel és többször lett tesztelve különböző esetekben. A különböző eseményeket (érintés, visszalépés stb.) fent jelzi kis ikonokkal, emellett az időt is jelzi legalul, így pontosan nyomon követhető minden eset. Jelenleg egy asztal 'profil' oldalát nyitottam meg, amely használja a legtöbb erőforrást. 1 MB adatot töltött le 1 másodperc alatt (kép miatt lett nagyobb, bejelentkezésnél néhány byte-ot tölt csak le), minimálisan terheli a CPU-t és az akkumulátort. Általánosan elmondható, hogy az alkalmazás háttérben nem generál adatforgalmat, és csak minimálisan meríti a telefont, viszont, ha olyan funkciót használunk, amihez GPS is szükséges (és előtérbe helyezzük az appot), akkor rövid idő alatt is sokat meríthet. Az eredményeket munkafolyamatokban tárolja az Andorid Studio, tehát az adat elvész, ha bezárjuk az alkalmazást, így érdemes exportálni a fontosabb adatokat munka közben.

6. Konklúzió

Az cikk átfogóan bemutatta egy alkalmazás fejlesztési folyamatát, kiemelve mind a szerver, mind a kliens oldali megoldásokat. A kliens oldal Android Studio és Java segítségével készült, míg a szerver oldal PHP-t és MySQL adatbázist alkalmazva tárolja az adatokat. A tervezés során számos szempontot figyelembe véve készül egy webes felület is, mely a böngészőn keresztül is elérhető lesz. Az alkalmazásba integrált Google Maps API lehetővé teszi a térképes megjelenítést, és ennek köszönhetően az alkalmazás könnyen adaptálható más

platformokra is. Az alkalmazás fő célja az emberek egészségesebb és aktívabb életmódra ösztönzése, különös tekintettel a csocsót szerető közösségre. A fejlesztés során a kliens és szerver oldalon alkalmazott technológiák, mint a Java, PHP és MySQL, illetve a Google Maps API, kulcsfontosságú szerepet játszanak a funkcionalitás és felhasználói élmény biztosításában. Az alkalmazás fejlesztése nem merül ki a mobilplatformon, hiszen a jövőben tervezik a webes és iOS verziók elkészítését is. Ennek az a célja, hogy minél szélesebb körben elérhető legyen, és lehetővé tegye az alkalmazás használatát mindenki számára, függetlenül az eszközválasztásától. Az így elért sokszínűség és elérhetőség hozzájárul az alkalmazás sikeréhez és hatékonyságához az emberek életmódjának pozitív irányú befolyásolásában.

Irodalomjegyzék

Eckel, B. (2003). *Thinking in JAVA*. Prentice Hall Professional.

Nichter, D. (2021). *Efficient MySQL Performance*. " O'Reilly Media, Inc.".

Nixon, R. (2014). Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. " O'Reilly Media, Inc.".

Gerber, A., Craig, C., & Selvaraj, D. (2015). Learn Android Studio: Build Android Apps Quickly and Effectively. Apress.

Rövid szakmai életrajz

2018 óta dolgozom webfejlesztőként, backend illetve full stack webfejlesztői szerepeket is töltöttem be. Az Opten Kft.-nél 2021 óta dolgozom, a szakterületem PHP alkalmazások és API-ok tervezése és kivitelezése.