

TDK Dolgozat

Szerzők:

Bán Róbert

Fazakas Tibor

Merli András

Mezei Zsolt

Szász István

XXXII. Országos Tudományos Diákköri Konferencia (OTDK)
Szeged, 2015. április 16-18.

The Social Jukebox

- A Közösségi Zenedoboz -



Szerzők:

Bán Róbert

Babeş-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, információ mérnöki szak, IV. év

Fazakas Tibor

Babeş-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, információ mérnöki szak, IV. év

Merli András

Babeş-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, információ mérnöki szak, IV. év

Mezei Zsolt

Babeş-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, információ mérnöki szak, IV. év

Szász István

Babeş-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, információ mérnöki szak, IV. év

Témavezető:

dr. Bodó Zsolt egyetemi adjunktus,

Babeş-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, Magyar Matematika és Informatika Intézet

Kivonat

A *The Social Jukebox* alkalmazás csomag egy közösségi zenedoboz szerepét tölti be. Célja, hogy az egy helyiségben tartózkodó emberek ízlésének a lehető legmegfelelőbb zenei szolgáltatást nyújtsa. A csomagban két szerver oldali és egy mobiltelefon alkalmazás található.

A *Social Jukebox* egy telefonos alkalmazás, elérhető Android, iOS és Windows Phone rendszereken egyaránt. Az alkalmazás tárolja a felhasználó kedvenc zenéit, és amikor szükséges, elküldi a listát a szervernek.

A *Social Jukebox Server* egy különálló webalkalmazás, amely képes fogadni a telefonokról érkező lejátszási listákat. Ezután egy intelligens összefésülő algoritmus segítségével olyan lejátszási listát állít elő, amely a legjobban megfelel mindenkinek. Természetesen feladatai között szerepel a zene lejátszása is.

A *Social Jukebox Net* webalkalmazás tárolja a zeneszámok metaadatait és a telefonos felhasználók adatait egy adatbázisban.

Tartalom

1 Bevezető	5
2 Felhasznált technológiák és eszközök	7
3 The Social Jukebox alkalmazás csomag.....	8
3.1 Követelmények	9
3.2 Kommunikáció	12
3.3 Architektúra	14
3.4 Megvalósítás	17
3.5 Az összefésülő algoritmus	22
4 Esettanulmány	23
5 Következtetések és továbbfejlesztési lehetőségek.....	26
6 Köszönetnyilvánítás	28
7 Hivatkozások.....	29

1 Bevezető

Napjainkban ha elmegyünk egy kávézóba vagy vendéglőbe, ott vagy a rádió szól, vagy egy előre meghatározott lejátszási lista van vagy esetleg egy alkalmazott (vagy egy lemezlovas, zenész) felel a zene biztosításáért. De milyen jó lenne, ha a törzshelyünkön a reggeli kávé elfogyasztása közben a kedvenc zenéinket hallgathatnánk. A múltban is voltak hasonló ötletek.

A 19–20. században megjelentek az érmés zenedobozok¹ és gépzongorák². Ezek voltak az első formái az automatikus, pénzérmével működő zenei eszközöknek. Az 1890-es években használtak először felvételeket a hangszerek helyett. A „jukebox”³ (zenedoboz) szó az 1940-es években került be a köztudatba az Amerikai Egyesült Államokban. A klasszikus zenegépen gombok voltak, a gombokra pedig különböző betűk és számok voltak gravírozva vagy nyomtatva, melyek segítségével ki lehetett választani a zeneszámot. Egy számláló eltárolta, hogy melyik szám hányszor volt meghallgatva, így a tulajdonos a legnépszerűbbeket meghagyta, míg a kevésbé népszerűeket lecserélte új zeneszámokra. Sok gyártó készített zenegépeket: az 1890-es években a *Wurlitzer*, 1920-ban a *Seeburg*, 1930-ban pedig a *Rock-Ola*. A zenegépek az '50-es években voltak a legnépszerűbbek, majd a digitális világ terjeszkedésének köszönhetően sajnos teljesen eltűntek.



A projekt célja egy **modern zenedoboz** elkészítése volt, amely rendelkezik ősei jó tulajdonságaival, de mindezek mellett lépést tart a korrallal. Egy klasszikus zenedobozhoz a vendég odalépett, fizetett, majd lejátszhatta kedvenc zeneszámát – feltéve, hogy az megtalálható volt a gépben. A mi alkalmazásunk esetében is hasonló történik, csak ebben az esetben a vendég szempontjából sokkal kényelmesebbnek mondható módszerrel, az okostelefonjával a kliens automatikusan működteti a rendszert. Az alkalmazás *pénztárcabarát*abb elődjénél, a szolgáltatásért már a vendég nem kell fizessen. Az alkalmazás

¹ http://en.wikipedia.org/wiki/Musical_box

² http://en.wikipedia.org/wiki/Player_piano

³ <http://en.wikipedia.org/wiki/Jukebox>

rendelkezik a zenét hallgatni vágyók kedvenc dalainak a listájával, ezért képes egy összefésült lejátszási listát összeállítani.

Az *összefésülő algoritmus* minden felhasználóhoz rendel egy *elégedettségi indexet*, mely egy meghatározott értékről indul, és idővel folyamatosan csökken. Ez az index úgymond méri, hogy ki mennyire van megelégedve a zenei kínálattal. Ha valamelyik felhasználó listájáról lejátszásra kerül egy zeneszám, akkor az ő elégedettsége nőni fog. Az algoritmus figyelembe veszi az aktuális elégedettséget, és megvizsgálja, hogy a lehetséges zeneszámok milyen hatással lennének a vendégek elégedettségére. Megpróbálja növelni az *átlag-elégedettséget*, hogy mindenkinek minél jobban megfeleljen a zenei repertoár. A *szórás* csökkentésével a személyek elégedettségi szintje között kevesebb lesz az eltérés. A *“senkit nem hagyunk hátra”* elvet követve a legelégedetlenebb személy szintjét folyamatosan emelni kell. A sokféle szempont figyelembevételével az alkalmazás kiválasztja és lejátsza a legalkalmasabb zeneszámot. A legalkalmasabb szám megtalálására egy szimulációt hajt végre: a beérkezett zeneszámok halmazában lévő összes zeneszámmra megvizsgálja, hogy ha lejátszásra kerülne, milyen hatással lenne az egyén elégedettségi szintjére.

Jelenleg a piacon két, a közösségi zenedoboz kategóriájába sorolható alkalmazást találtunk: a Mixgar-t⁴ és a Noispot-ot⁵. Az alkalmazásunk szerkezetileg és kinézetileg is különbözik az előző két alkalmazástól. Viszont a legfontosabb amiben eltér: az az összefésülő algoritmus és az az ötlet, hogy a telefonos felhasználó előre összeállíthatja kedvenc zenéinek a listáját, amelyet az alkalmazás automatikusan továbbít az összefésülő algoritmus számára.

⁴ <http://mixgar.com>

⁵ <http://noispot.com>

2 Felhasznált technológiák és eszközök

A projekt elkészítése során több technológia és eszköz került felhasználásra.



Az *Android* telefonos alkalmazás fejlesztése a Google által kiadott *Android SDK*-t [1] használva történt. A fejlesztéshez *Android ADT plug-in*-nal [2] bővült ki az *Eclipse* [3] integrált fejlesztői környezet.



Az *Xcode* [4] egy programfejlesztő csomag a Mac OS X operációs rendszerre, amit az Apple fejlesztett ki. A programkód *Objective C* nyelvben íródott. A fejlesztői csomag része az *iOS SDK* [5], amely a programkódot *iOS 7.1*-es operációs rendszer számára futtatható kódra fordította le.



A *Visual Studio* a Microsoft fejlesztőkörnyezete, támogatja a .NET [6] keretrendszert. A *Windows Phone 8* alkalmazás fejlesztése e rendszer segítségével történt *C#* [7] nyelven. Az alkalmazás a *Windows Phone SDK* [8], a felhasználói felület pedig a *WPF* [9] keretrendszer felhasználásával készült. Kódelellenőrzés és automatikus kódrefaktorálás végett a *JetBrains ReSharper 7.0*-ás verziójú *plug-in*-nal egészült ki a *Visual Studio*.



A Java fejlesztésekhez a programozók *JDK 7*-et használtak, fejlesztői környezetként pedig a *NetBeans IDE 7.4* [10] integrált fejlesztői környezetet, amely tartalmazza például a *HTML5* összes fejlesztői lehetőségét, ezen kívül lehetőséget nyújt vizuális *CSS*-szerkesztésre és lehetővé tesz *JavaScript*-debuggolást is, melyekre a weboldalak fejlesztésénél volt szükség. Az alkalmazás webes felületéhez a *Java Server Pages* (JSP) [11] technológia került alkalmazásra, a szerver által használt adatok tárolása pedig *MySQL* [12] adatbázis-kezelő rendszerrel történt. A build folyamat automatizálására az *Apache Maven 3* [13] került felhasználásra. A szerver oldali naplózás *Logback* [14] segítségével történt.



A szerveralkalmazás *Apache Tomcat*-ben [15] fut. Ez az Apache Software Foundation fejlesztése, platformfüggetlen, tisztán Java nyelvben készült.

A hálózati kommunikáció megvalósítása a *JSON* (JavaScript Object Notation) [16] szabvány segítségével történt.

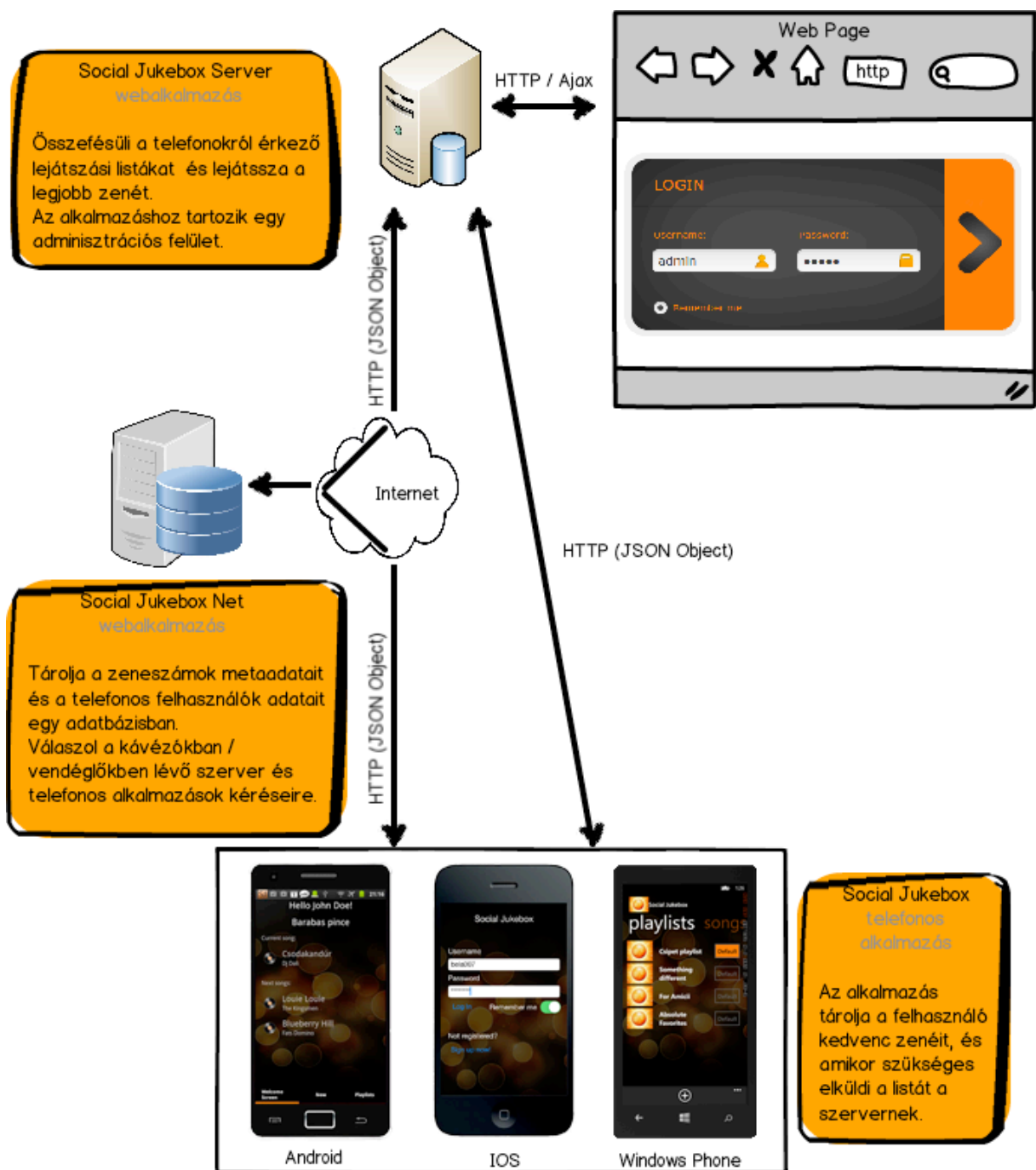
A projekt menedzselése a *Redmine* [17] web alapú alkalmazás segítségével valósult meg.



A forráskód verziókövetéséhez a *Git* [18] rendszerre esett a választás. Ez egy osztott verziókövető rendszer, amely peer-to-peer megközelítést használ.

3 The Social Jukebox alkalmazás csomag

A *The Social Jukebox* csomag három nagyobb alkalmazásból tevődik össze (1. ábra): egy okostelefonokra telepíthető szoftver és két szerveralkalmazás tartozik a csomaghoz. Az egyiket a vendéglátóhelyek telepítik a gépeikre, míg a másik szerver interneten keresztül lesz elérhető.



1. ábra: *The Social Jukebox* alkalmazás csomagot ábrázoló diagram

3.1 Követelmények

A *The Social Jukebox* projekt több különálló egységből áll, ezért a legfontosabb követelmény a projekt fejlesztése szempontjából az egységek közötti megfelelő kommunikáció. Az alábbiakban a dolgozat kitér a komponensek vázlatos specifikációjára.

3.1.1 Social Jukebox Net

A *Social Jukebox Net* feladata a telefonos alkalmazások és *Social Jukebox Server*-ek kéréseinek, valamint a központi adatbázisnak a kezelése.

A fontosabb funkciók közé tartozik az új telefonos felhasználók regisztrációja és a felhasználók zeneszám validációs kéréseinek fogadása, majd azonosító információk küldése. Ehhez tartozik a zeneszámok azonosítása részleges információk alapján. Emellett az alkalmazás elvégzi a zeneszámok metaadatait tartalmazó adatbázis karbantartását és frissítését, ami jelenleg pár millió zeneszámról és több százezer előadóról tartalmaz információt.⁶

A *Social Jukebox Net* a *Social Jukebox Server*-től érkező statisztikai adatokat fogadja, mint például az összes lejátszási lista, lejátszott zeneszámok, helyiség és a jelenlevő felhasználók száma, majd ezeket feldolgozza.

3.1.2 Social Jukebox Server

A *Social Jukebox Server* feladata a regisztrált telefonos felhasználók kéréseinek fogadása, a zenelejátszás irányítása és az alkalmazás vezérléséhez szükséges adminisztrációs felület biztosítása.

A szerver egyik funkciója a telefonos alkalmazások bejelentkező kérésének fogadása a felhasználóknak a helyiségbe való belépésekor, majd a helyiséget azonosító információk elküldése. Egy másik feladata a telefonos alkalmazások lejátszási listáinak fogadása és mentése.

A *Social Jukebox Server* végzi a telefonos felhasználók lejátszási listáinak összefésülését, ami alapján elkészül a szerveren futó zenelejátszó lejátszási listája.

Az adminisztrációs felület bejelentkezés után lehetőséget ad a zenelejátszó kezelésére: indítás, szüneteltetés, megállítás, a hangerő szabályozása.

⁶ A zenei adatbázis a *MusicBrainz* zeneenciklopédia adataival volt feltöltve. (<http://musicbrainz.org>)

Továbbá lehetőséget biztosít a lejátszott zenékkal kapcsolatos beállítások megadására: zeneszámok, előadók, stílusok tiltása és a zenefájlokat kezelő adatbázis beállítása. Statisztikákat jelenít meg a lejátszott zenékről, a helyiségben tartózkodó mobil felhasználók adatairól; kik azok, akik küldtek lejátszási listákat, ki mikor jelentkezett be és jelentkezett ki.

3.1.3 Social Jukebox telefonos alkalmazás

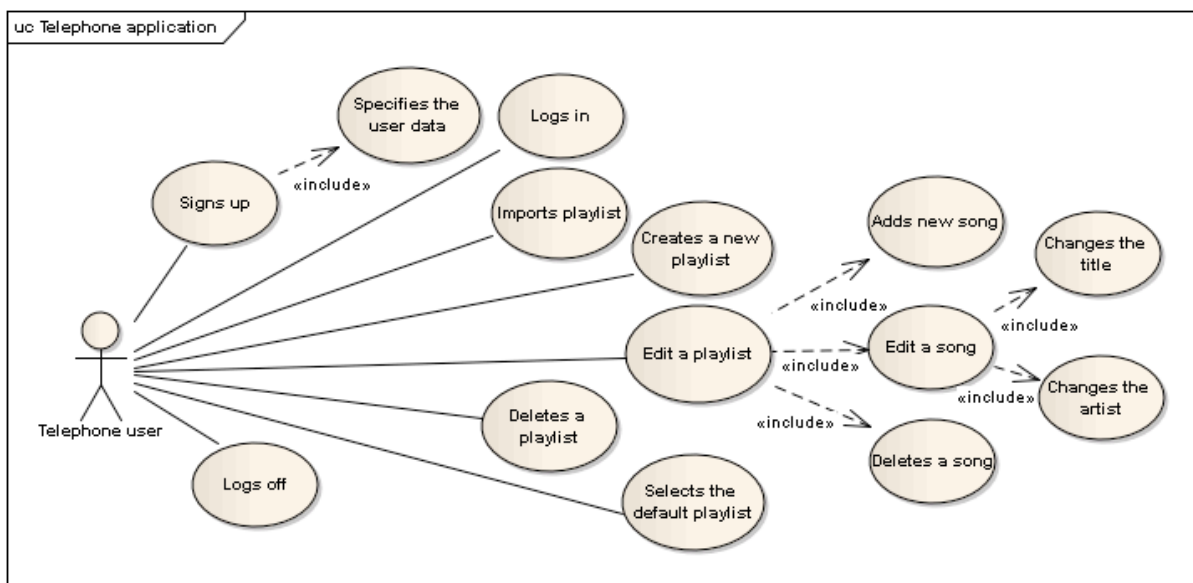
A telefonos alkalmazás feladata a felhasználó azonosítása, az általa meghatározott lejátszási listák kezelése, illetve a kapcsolat tartása a közösségi térrel.

A szoftver alapfunktionalitása, hogy kommunikáljon a kávézóban levő szerverekkel. Biztosítja a lejátszási listák küldését, illetve fogadja a szerver által küldött adatokat. Ezen adatok alapján informálja a felhasználót, megjeleníti az éppen lejátszott, illetve a közeljövőben lejátszásra kerülő zeneszámok információit, valamint a hely információit és reklámjait.

A felhasználó szempontjából lehetőséget nyújt regisztrálásra, bejelentkezés után új lejátszási listák létrehozására, módosítására, illetve elküldésére a kávézó számára.

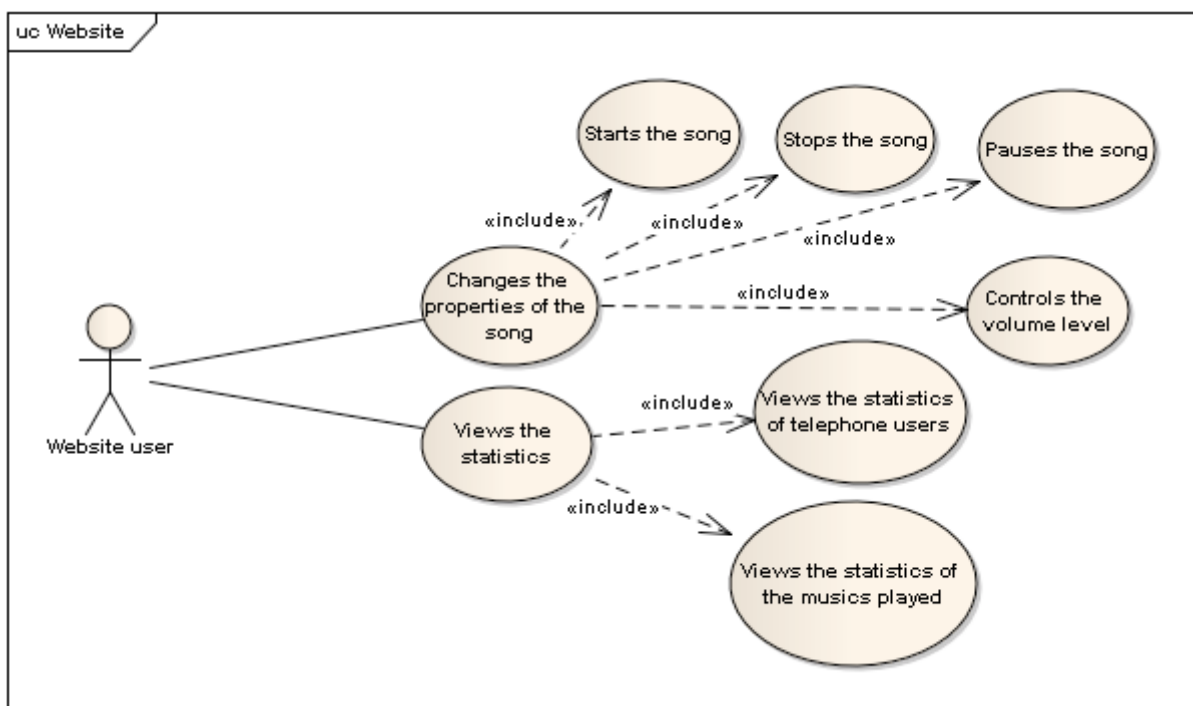
3.1.4 Használati esetek

Az alábbi *use-case diagram* (2. ábra) a felhasználó szemszögéből szemlélteti a telefonos alkalmazás fontosabb használati eseteit.



2. ábra A telefonos alkalmazás főbb funkcionálisai ábrázolva egy használati eset diagramon, a telefont használó személy szemszögéből

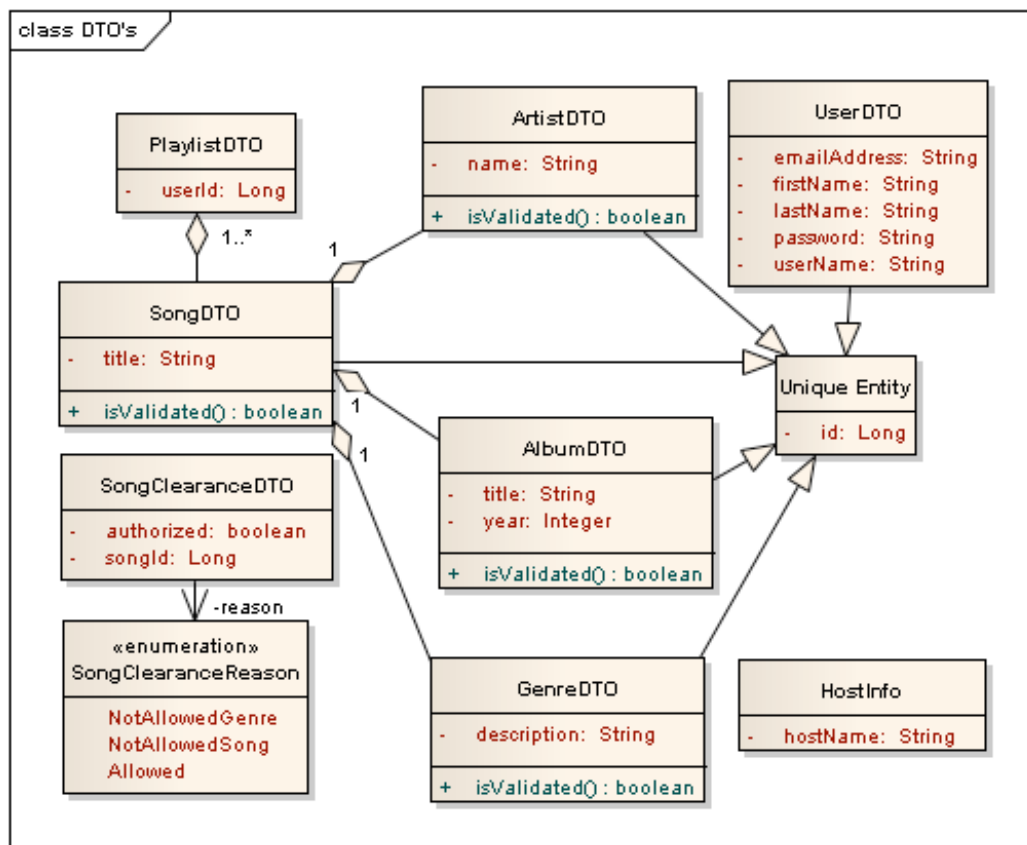
A kávézóba, a vendéglőbe elhelyezett szerveren futó webalkalmazás adminisztrációs felületéhez az ott dolgozók férhetnek hozzá. Az alábbi *use-case diagram* (3. ábra) a fontosabb használati eseteket mutatja be.



3. ábra A weboldal főbb funkcionálisai ábrázolva egy használati eset diagramon, a vendéglátóhely személyzete szemszögéből

3.2 Kommunikáció

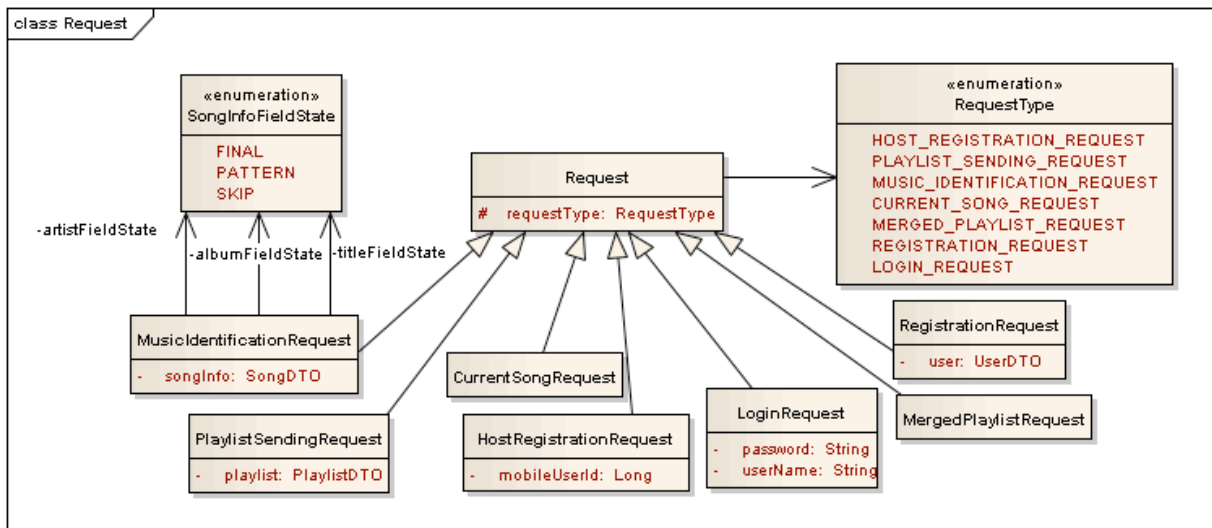
Az alábbi *osztály diagram* (4. ábra) a kommunikációban résztvevő *DTO*-k (Data Transfer Object) láthatóak, amelyek segítségével adatokat osztanak meg egymás között az alkalmazások.



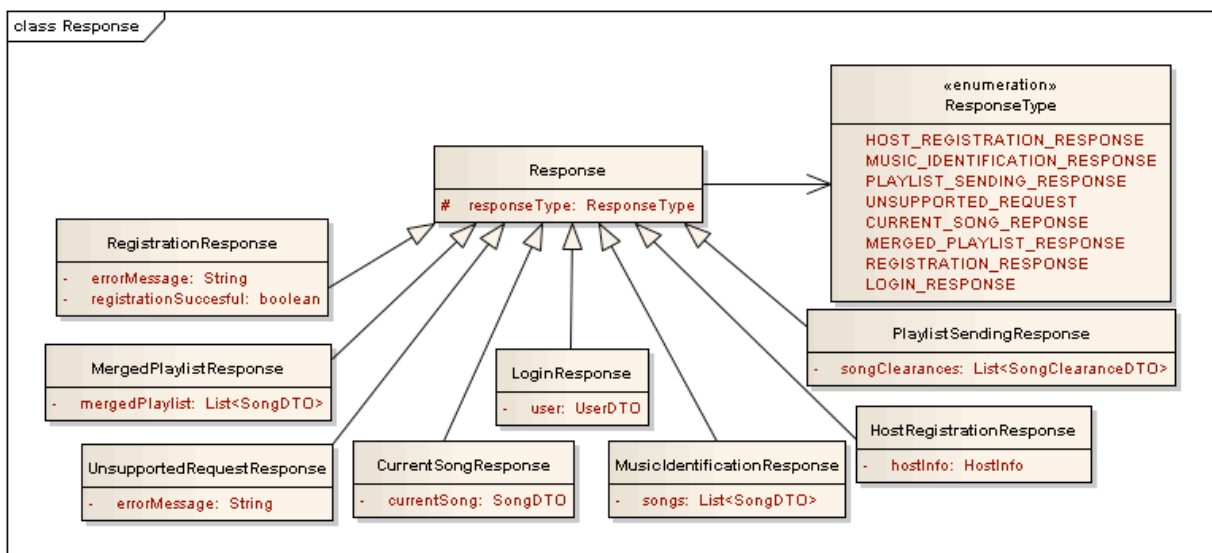
4. ábra kommunikációban résztvevő DTO-k vázlatos osztály diagramja

Az alkalmazás témájából adódóan a DTO-k többsége zenei entitásokat reprezentál. Mivel ezeket az adatokat a felhasználók is létrehozhatják, nagy a hiba (elírás, részleges kitöltés) lehetősége. Ezért minden zenei entitás osztálya rendelkezik egy egyedi azonosítóval, amelyet akkor kap, ha a központi adatbázisban már szerepel ez a bejegyzés. Ellenkező esetben az adatok nem tekinthetők érvényesnek.

A *SongClearanceDTO* arról tartalmaz információt, hogy egy lejátszási listán szereplő zeneszámot le lehet-e játszani vagy sem, illetve, ha nem, akkor annak okát (nincs meg a zeneszám, tiltva van a stílus, előadó, zeneszám stb.). Ezenkívül a kommunikációban részt vesznek a mobil felhasználókról és a helyiségekről információkat tároló osztályok.



5. ábra A szerverhez intézett kérésben szereplő Request osztályok vázlatos osztály diagramja

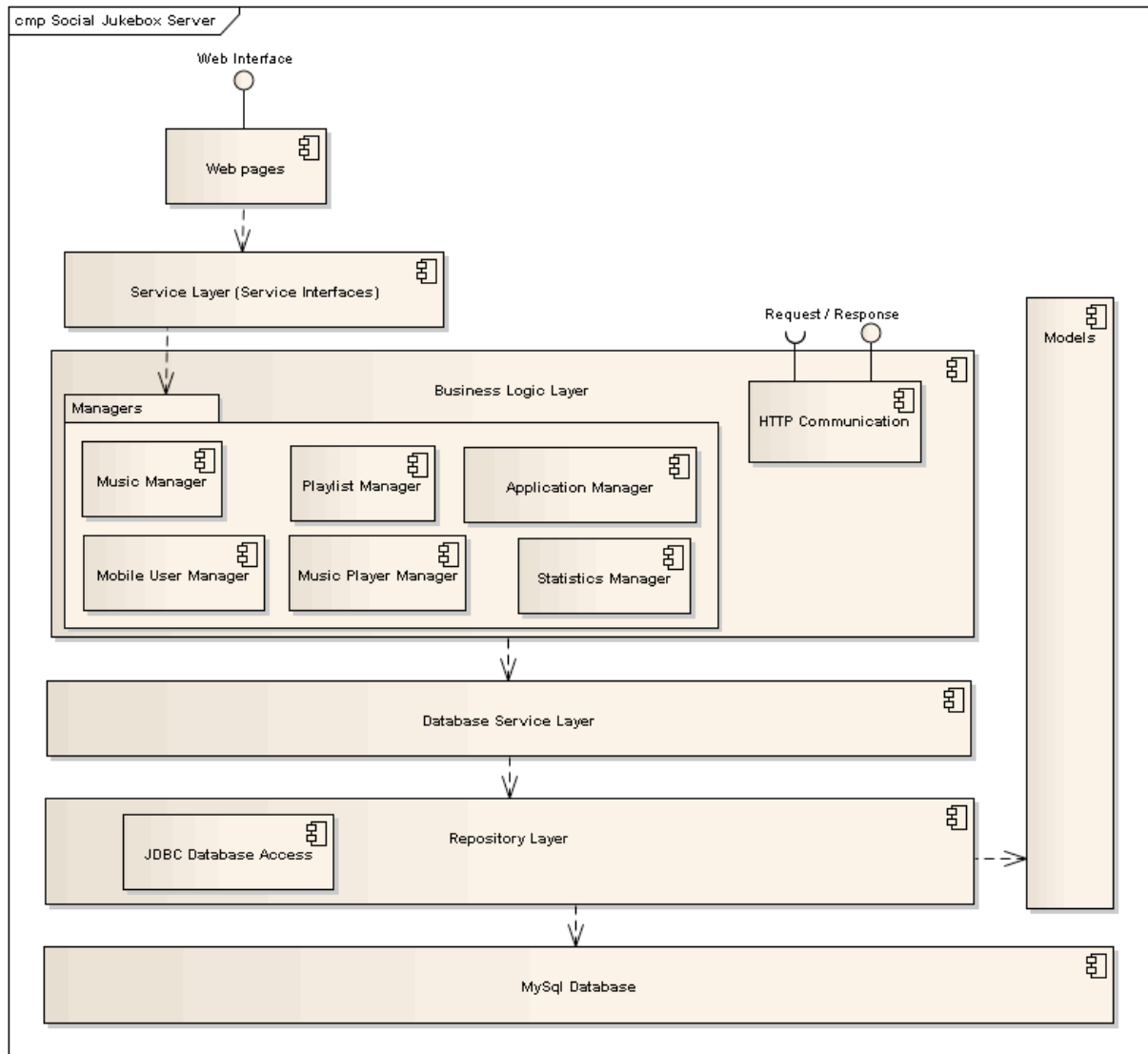


6. ábra A szervertől kapott válaszban szereplő Response osztályok vázlatos osztály diagramja

Az alkalmazások közötti adatcsere a *Request* (5. ábra) és *Response* (6. ábra) ősosztályok és az ezekből származtatott osztályok segítségével történik. A *Request* (*Response*) osztály tartalmaz egy *requestType* (*responseType*) nevű, protected hozzáférésmódosítójú, enum típusú változót. Ez a változó a származtatott osztályok konstruktoraiban van beállítva a származtatott osztálynak megfelelő értékre. Így a *JSON* objektumokat mindig át lehet alakítani *Request/Response* osztályból származtatott osztályokra. A kommunikációnál használt **saját protokoll** egyértelműen meghatározza, hogy mikor milyen típusú kérést lehet küldeni és erre milyen típusú válasz fog érkezni.

3.3 Architektúra

A *Social Jukebox Server* az adatokat egy *MySQL* adatbázisban tárolja (7. ábra).

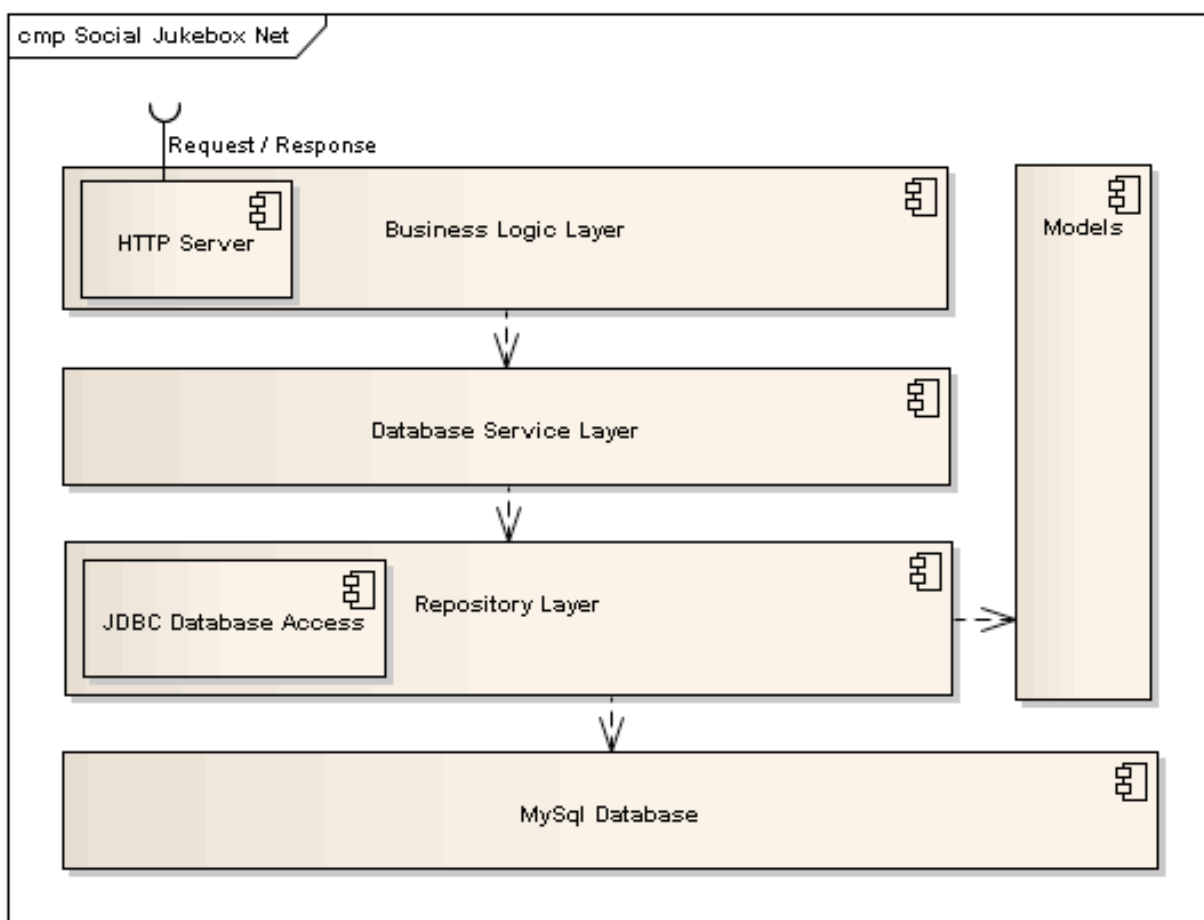


7. ábra *Social Jukebox Server* architektúra diagramja

Az adatok adatbázisba való mentéséért, frissítéséért és az adatbázisból való törléséért a *Repository Layer* felelős. Az adatbázisból kinyert adatokat a layer a *Model* osztályok valamelyikében adja vissza, és az összes magasabb szinten levő komponens ezekkel a *Model* osztályokkal végez műveleteket. A *Database Service Layer* az összetett lekérdezések, és más adatbázis műveleteket végez el. Ezt a réteget a *Business Logic Layer* használja, mely két nagyobb részből áll. Az első egy *HTTP Communication* modul, melynek része egy szerver, ami a mobilalkalmazásoktól érkező kéréseket fogadja, és ezekre válaszol. Az adatcsere

a *Model* komponens *DTO* (Data Transfer Object) osztályait használja. A *Business Logic Layer* másik fő komponense a *Managers*, mely az alkalmazás fő követelményeinek megfelelő műveleteket végzi el: lejátszási listák fogadása és összefésülése, telefonos felhasználók kezelése, zene lejátszása, statisztikák készítése. Az alkalmazásnak a vezérlését és az adatok megjelenítését az *UI*-ról a *Service Layer*-ben megadott interfészek teszik lehetővé.

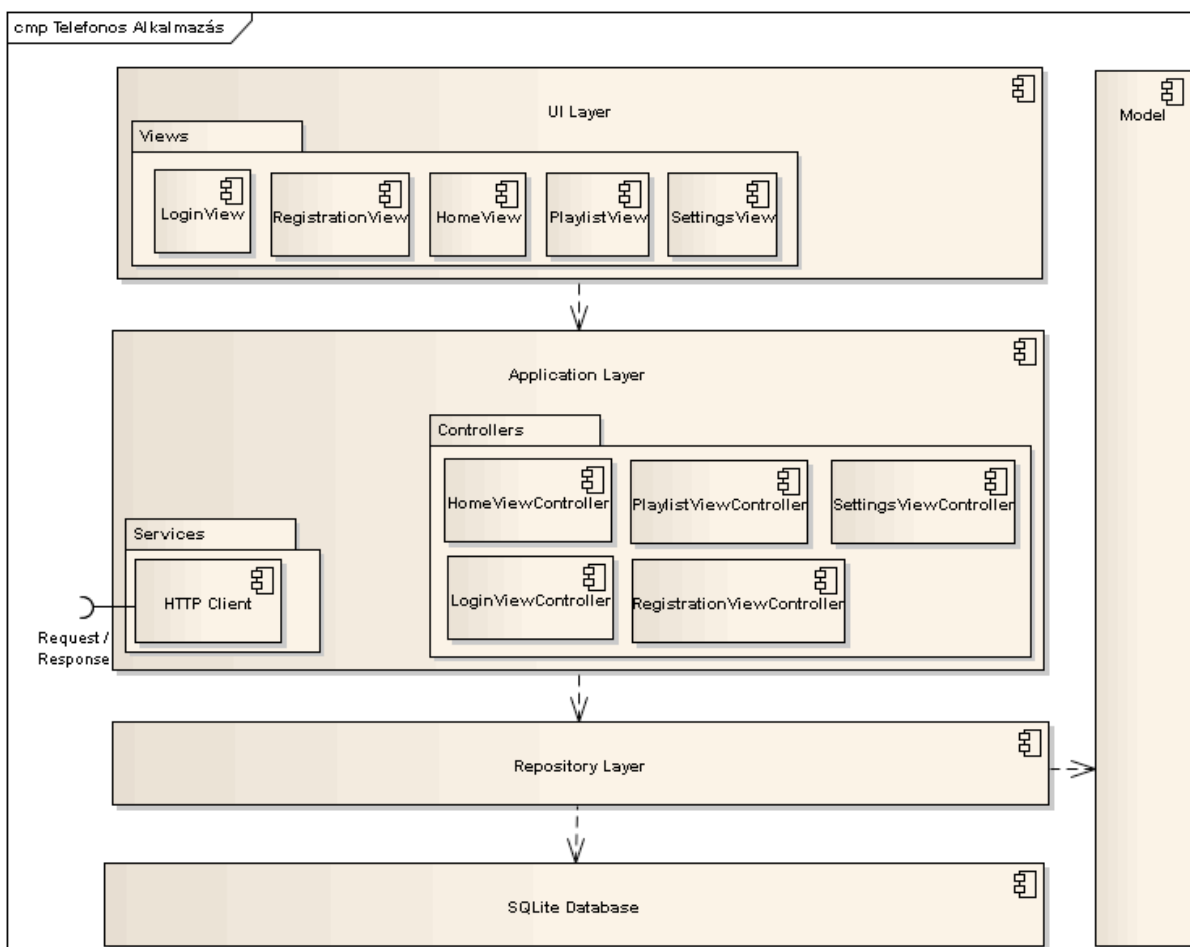
A *Social Jukebox Net* felépítése hasonló a *Social Jukebox Server*-hez (8. ábra): szintén *MySQL* adatbázist használ, az adatok olvasásáért és írásáért a *Repository Layer* felel, míg a bonyolultabb lekérdezéseket a *Data Service Layer* valósítja meg.



8. ábra *Social Jukebox Net* architektúra diagramja

A *Business Logic Layer* a mobilalkalmazások és *Social Jukebox Server*-ek kéréseit fogadja és válaszol rájuk, illetve a zeneszámok metaadatait tartja karban és frissíti.

Ahogy azt a következő architektúra diagram (9. ábra) is szemlélteti, a telefonos alkalmazás az MVC (modell–nézet–vezérlő) szerkezeti mintára épül.



9. ábra Social Jukebox telefonos alkalmazás architektúra diagramja

A telefonos alkalmazás az adatokat az eszközök által biztosított *SQLite* relációs adatbázisokban tárolja, így biztosítva a gyors elérést és az elfogadható erőforrásigényt. A *Model* csomag tartalmazza a kommunikációt egységesítő DTO osztályokat és az adatbázist leíró entitásokat egyaránt. A *Repository* réteg felelős a közvetlen adatbázis műveletekért, az egyes entitások lekérdezéséért, elmentéséért, módosításáért, illetve törléséért.

A *Social Jukebox* alkalmazás rétegében a *Controller*-ek feladata, hogy kezeljék a felhasználói felületről érkező kéréseket kapcsolatba lépve a *Repository* réteggel és a *HTTP kliens*et használó *Service* réteggel – ez utóbbi a szerverekkel történő kommunikációt biztosítja.

A felhasználói interakció működését az *UI* réteg teszi lehetővé. Létrehozza a nézeteket, majd az itt kiváltott eseményeket a megfelelő kontrollerekhez továbbítja.

3.4 Megvalósítás

Az alábbiakban az alkalmazáscsomaghoz tartozó projektek megvalósítása olvasható.



3.4.1 Social Jukebox telefonos alkalmazás

Az alkalmazás telefonos része három operációs rendszert támogat, ezek az *Android*, *iOS*, illetve a *Windows Phone 8*. Az alkalmazás felépítésénél nagy figyelmet kapott, hogy a felhasználó elé egyszerűen kezelhető, ergonomikus felület kerüljön, könnyen tudjon tájékozódni az egyes oldalak és lehetőségek között. Ugyanakkor a felhasználói felület kialakításánál figyelembe vettük az egyes platformokon már bevett megoldásokat, így a *Windows Phone* esetében *Pivot Control*-ra épült az alkalmazás főoldala, míg az *Android*, illetve *iOS* alatt *Tab Control*-okra.

A mobilalkalmazások az operációs rendszer által rendelkezésre bocsátott pehelysúlyú adatbázis-kezelő rendszereket használták, ez az *Android* esetében az *SQLite*. A *Windows Phone 8* esetében a *LINQ to SQL* ORM keretrendszert, míg az *iOS* a *Core Data* ORM keretrendszert használja.

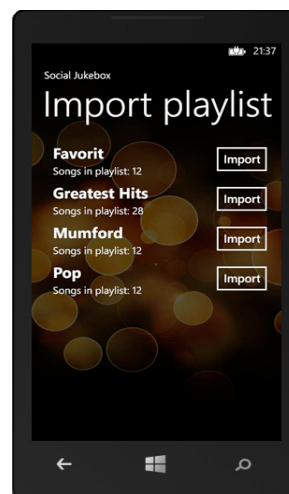
A szerveroldali komponensekkel történő kommunikáció *HTTP* kéréseken keresztül, *JSON* objektumok küldésével-fogadásával történik.⁷

A felhasználó kényelme miatt fontos szempont volt, hogy háttérben is tudjon működni az alkalmazás, ezáltal elősegítve, hogy a felhasználó lejátszási listája már a helyiségbe való belépésnél, vagyis a WiFi-re való csatlakozásnál el legyen küldve a szerverhez, így ő nem kell tegyen semmit, hogy a kedvenc dalait a rendszer figyelembe vegye. A feladat megoldása platformonként eltérő volt. *Android* és *iOS* rendszeren gond nélkül futhat a háttérben egy folyamat. A *Windows Phone* operációs rendszer esetében a háttérben olyan alkalmazások futhatnak, amelyek zenelejátszást, illetve térképes szolgáltatásokat is biztosítanak.⁸

⁷ Az *iOS* SDK *JSON* osztálya kevés szabadságot ad a fejlesztőnek saját típusokra való átalakításhoz. *NSArray* vagy *NSDictionary* típusokat tud átalakítani, ezért minden modellhez egy *toDictionary* és egy *initWithDictionary* metódus hozzáadása szükséges.

⁸ *Windows Phone*-on az automatikus csatlakozáshoz és egyéb háttérműveletek elvégzésére úgynevezett *Agent*-eket lehet használni. Egy *Agent* 30 percenként lefuthat, és futási ideje nem haladhatja meg a 10 másodpercet.

A felhasználó számára lejátszási listák létrehozását megkönnyíti, hogy a telefonon már meglévőket használhat fel (10. ábra). Ha már rendelkezünk egy lejátszási listát reprezentáló állománnyal (pl. M3U formátumban), akkor ezt bevezethetjük az alkalmazás lejátszási listái közé. Androidon az erre vonatkozó adatok elérhetőek a beépített *MediaStore* osztály segítségével. Windows Phone 8 esetén ezeket a szolgáltatásokat a *MediaLibrary* osztály biztosítja, míg iOS esetén a *Media Player keretrendszer*.



10. ábra A telefonos alkalmazás Windows Phone operációs rendszeren

3.4.2 Social Jukebox Server

A webalkalmazás felépítése az *MVC* elvet követi, az adminisztrációs felületet biztosító weboldalak megjelenítésére a megfelelő *JSP* fájlok szolgálnak, a vezérlést *HttpServlet*-ek végzik, ezek fogadják a kéréseket és válaszolnak ezekre, a modell részt pedig a backend egyes Java osztályai alkotják. A weboldalak külalakját stíluslapok írják le.

A weboldalon mindig a szerveren futó lejátszó aktuális állapota és az aktuális statisztikák jelennek meg, ezért az adatcsere megvalósításához az *AJAX* technika alkalmazására volt szükség.

Az alkalmazás indulásakor a *StartupBean* hozza létre az alkalmazás futásához szükséges objektumokat. Ezek közül a legfontosabbak a *backend.managers* csomagban található osztályok. Időszakosan ismétlődő feladatokat látnak el, ezért a *Thread* osztályt öröklík. Mivel csak egy darabra van szükség belőlük, a *Singleton* tervezési mintát követik. Az *ApplicationManager* indítja el a többi *Manager*-t, és ez biztosítja, hogy helyesen álljanak le. A *MobileUserManager* tárolja a jelenlevő telefonos felhasználók adatait, ez regisztrálja be az új felhasználókat és távolítja el őket (ha jelenlétüket azok egy ideje nem jelezték). A *PlaylistManager* fogadja a *MobileUserManager*-nél regisztrált felhasználók lejátszási listáit, törli a már nem jelenlevő felhasználókét, és fenntart egy összefésült listát, melyre új zeneszámokat generál, ha a régiek már lejátszásra kerültek vagy az egyéni listák megváltoztak. A listák összefésülésénél a

PlaylistManager a *MobileUserManager*-től kéri el a felhasználók és az aktuális elégedettségük listáját. Ezek után minden felhasználóra és minden zeneszámmra kiszámolja, hogy mi lenne az új elégedettség értéke. Az új elégedettség kiszámolását egy, a *SatisfactionUpgrader* interfészt megvalósító osztály végzi. Az új elégedettségi szinteket figyelembe véve a *SongChooser* interfészt megvalósító osztály választja ki a lejátszási lista következő elemét. A zeneszámok elérési útvonalát, legutolsó lejátszási dátumát és ebből következően azt, hogy lejátszhatóak-e vagy sem, a *MusicManager* tartja nyilván. A *PlayerManager* kezeli a zenelejátszót, folyamatosan figyeli, hogy volt-e valamilyen változtatás a weblapon, vagy, hogy lejárt-e már az éppen lejátszott szám. A *StatisticsManager* osztály időszakosan menti az adatbázisba a jelenlevő felhasználók számát, ezek elégedettségét, zeneszámok lejátszásának gyakoriságát, és kérésre ábrát készít ezekről.

A **backend.satisfaction** csomag tartalmazza a *SatisfactionUpgrader*, *SongChooser* interfészeket és az ezeket megvalósító osztályokat. Az interfészekre azért volt szükség, hogy könnyen és gyorsan lehessen ezeket az osztályokat cserélni, mivel több módszer is kipróbálásra került a lejátszási listák összefésülésénél.

A *SatisfactionUpgrader*-t a *ConstantBonusSatisfactionUpgrader* valósítja meg. Az *upgradeSatisfaction* függvény megkapja paraméterként a mobil felhasználó jelenlegi elégedettségére vonatkozó adatokat, a lejátszási listáját és a zeneszámot, aminek a hatását ki akarjuk számolni. Ez az osztály konstans értékkel növeli a felhasználó elégedettségét, ha a zeneszám megtalálható a felhasználó lejátszási listáján. Az érték kisebb lesz, ha csak egy másik zeneszám szerepel a listán ugyanattól az előadótól, illetve még kisebb, ha csak műfajbeli egyezés van. Ha semmilyen egyezés sincs, akkor az elégedettség csökkenni fog. A csökkenés mértéke mértani sorozatot követ annak függvényében, hogy a felhasználó milyen rég nem hallott olyan zeneszámot, ami szerepel a listáján, ezzel biztosítva, hogy az elégedettség drasztikus mértékben csökkenjen, ha nagyon rég nem játszottuk le valakinek a zenéjét. A *SongChooser* interfész egy *chooseSong* metódussal rendelkezik. Ez a metódus megkapja minden felhasználó elégedettségét minden zeneszám után, a megvalósító osztályok ezekből egy „jósági indexet” számolnak és kiválasztják a legjobbat. A különböző megvalósítások között az eltérés a jósági

index számolásának függvényében tér el. A *MaxMinimumSongChooser* növekvő sorrendbe rendezi a zeneszámok utáni elégedettségeket. Úgy választja ki a következő zeneszámot, hogy a sorozat első eleme a legnagyobb legyen. A *MaxAverageSongChooser* minden zeneszámmra kiszámolja az átlagelégedettséget és a legnagyobbnak megfelelő zeneszámot választja. A *MinStdSongChooser* osztály zeneszámok utáni elégedettségek szórásai közül a legkisebbet választja. Az előzőnek a továbbfejlesztése a *MinCoefficientOfVariationSongChooser*, ami a szórásokat elosztja az átlagokkal, a kiválasztott zeneszám a legkisebbnek az argumentuma.

A **WebServer** osztály szintén induláskor példányosítódik és egy *HttpServer*-t hoz létre, hozzárendeli a kommunikációs kontextust és a *MobileRequestHandler* osztályt. A *MobileRequestHandler* a *HttpHandler* osztályból öröklődik és felülírja a *handle* metódust. Ez a metódus egy *HttpExchange* típusú paramétert kap, ennek a törzséből kiveszi a *JSON* objektumot, átalakítja saját *Request* típusra, majd a származtatott *Request* típusú objektumra, elvégzi a szükséges műveleteket, felépíti a megfelelő *Response* osztályból származtatott osztály objektumát és visszaírja a *HttpExchange* objektumba.

Az **adatbázis-hozzáférési réteg** interfészei a *backend.repository* csomagban találhatóak. A réteg megvalósítása az *Abstract Factory* mintán alapszik, a *DAOFactory* osztály szolgáltatja a DAO interfészek kiválasztott megvalósításait. A *repository.jdbc* tartalmazza a *MySQL* adatbázishoz *jdbc driver* segítségével kapcsolódó DAO megvalósításokat.

A **backend.model** csomag tartalmazza a modell osztályokat, amelyek az alkalmazás minden szintjén használva vannak. A *DTO* osztályok, a *Request/Response* osztályok, illetve az ezekből származtatott osztályok egy külön projekt részei (Common). A *Social Jukebox Server* projektbe ez *Maven* dependenciaként van beillesztve.

3.4.3 Social Jukebox Net

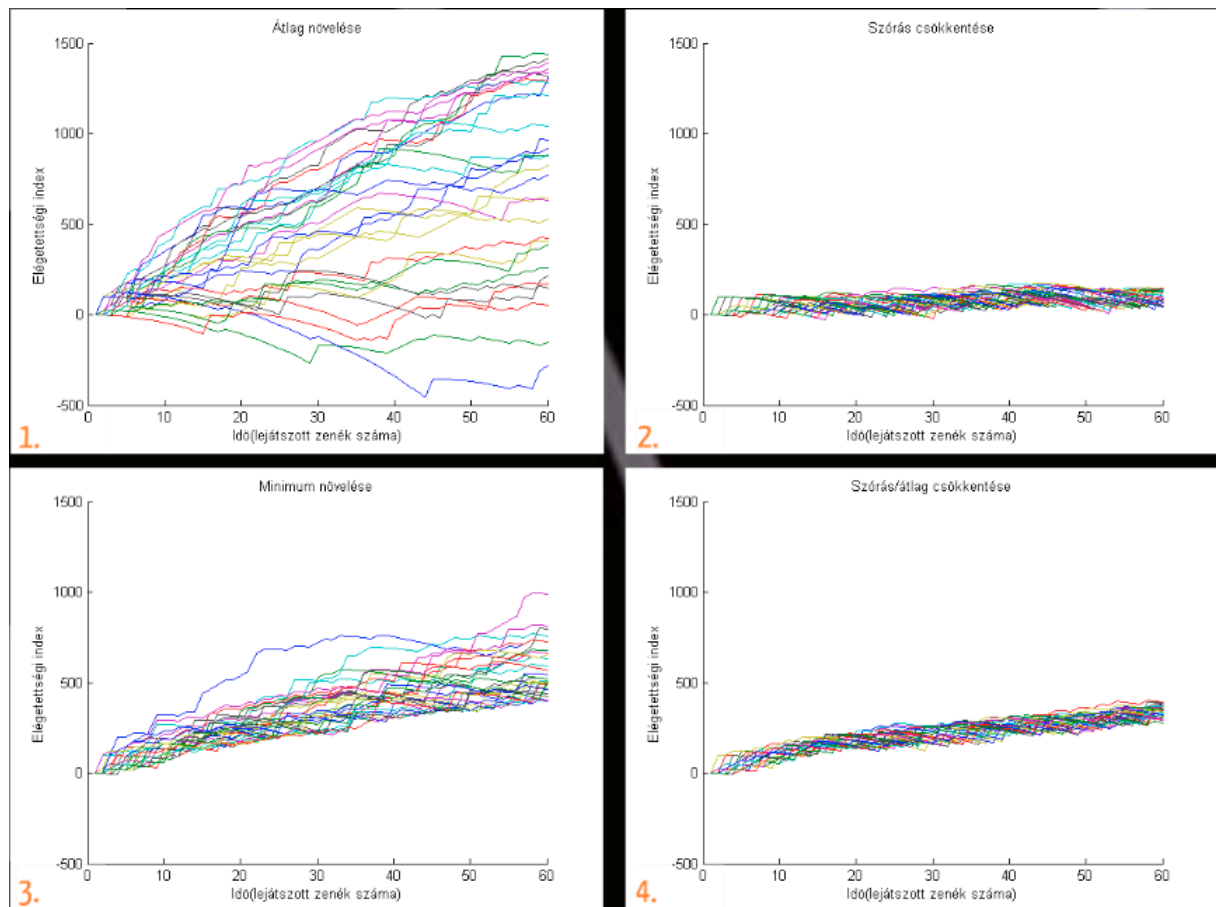
A *Social Jukebox Net* alkalmazás adatbázis-hozzáférés rétege hasonlít a *Social Jukebox Server*-éhez, ez már az architektúrák közötti hasonlóságból is következik. A különbség az, hogy más entitásokat kezel és emiatt más modell osztályokra van szüksége. Ez az adatbázis tárolja a metainformációkat az összes zenei entitásról

(zeneszámok, szerzők, albumok, stílusok stb), melyet a rendszerünk ismer. A már az előző fejezetből ismert *HTTP* *szerver* és a *Request/Response*-okra alapuló kommunikáció valósítja meg a mobilalkalmazások és a *Social Jukebox Server*-ek adatcseréjét a *Social Jukebox Net*-tel. A mobilalkalmazások részleges zeneszám metainformációt is küldhetnek, amit a szerver megpróbál azonosítani. Ha teljes az egyezés, akkor visszaküldi az egyedi azonosítót, ellenkező esetben több hasonló adattal rendelkező zeneszám adatait küldi vissza. Jelenleg a szerver egyszerű mintafelismerést használ: a részleges metainformáció bárhol szerepelhet a teljes leírásban (az elején, a közepén, a végén).

A mobilalkalmazások ezen kívül elküldik a lementett lejátszási listáikat. A listák érvényesített zeneszámaikat statisztikai adatok generálására, a nem érvényesített pedig a zenei adatbázis bővítésére használja fel: minden ismeretlen entitás megjelenési gyakoriságát elmenti, majd ha ez elér egy adott küszöbértéket, bevezeti az érvényes adatok közé.

A *Social Jukebox Server*-ek adatokat küldenek a jelen levő telefonos felhasználókról és a lejátszott zeneszámokról, melyeket szintén az adatbázisban tárolunk el – ezeket statisztikák készítésére lehet felhasználni.

3.5 Az összefésülő algoritmus



11. ábra Négy különböző összefésülési algoritmus futási eredménye (a vendégek elégedettségi indexe idő függvényében). 50 vendég, mindegyiknek 20-20 zeneszáma volt, és 60 számot játszott le a szerver.

A telefonokról érkező lejátszási listákat a vendéglátó hely szervere össze kell fésülje. Az alapötlet, hogy elégedettségi indexeket társít minden jelenlévőhöz. Ez az index 0-ról indul, és idővel folyamatosan csökken. Ha valaki kedvenc zeneszámát hallja akkor 100, ha kedvenc előadójától hall számot akkor 25, ha kedvenc stílusú zenéjét hallja akkor plusz 10 pontot kap. Több algoritmus is kipróbálásra és tesztelésre került (11. ábra). Az ábrákon a vendégek elégedettségi indexei láthatóak az idő függvényében. Az első algoritmus az átlagot próbálja maximalizálni, míg a második a szórást próbálja csökkenteni. Látható, hogy az első esetben magasabbak az indexek, viszont sokkal szétszórtabbak a második esethez képest. A harmadik esetben a minimumot próbálja maximalizálni, a negyedik algoritmus pedig a szórás/átlag arányát minimalizálja, azaz egyidőben próbálja csökkenteni a szórást és növelni az átlagot. Az alkalmazás ezt a módszert használja jelenleg.

4 Esettanulmány

A *The Social Jukebox* alkalmazáscsomag működésének szemléltetésére tekintünk az alábbi példát:

Egy helyiségben 4 ember tartózkodik, legyen a nevük Mr. Pink, Mr. Brown, Mr. Orange és Mr. Blue. Mr. Blonde majd később fog csatlakozni hozzájuk. A négy személy zenei ízlése sokban hasonlít egymásra. Tekintsük az egyes felhasználók aktív lejátszási listáit (12. ábra):

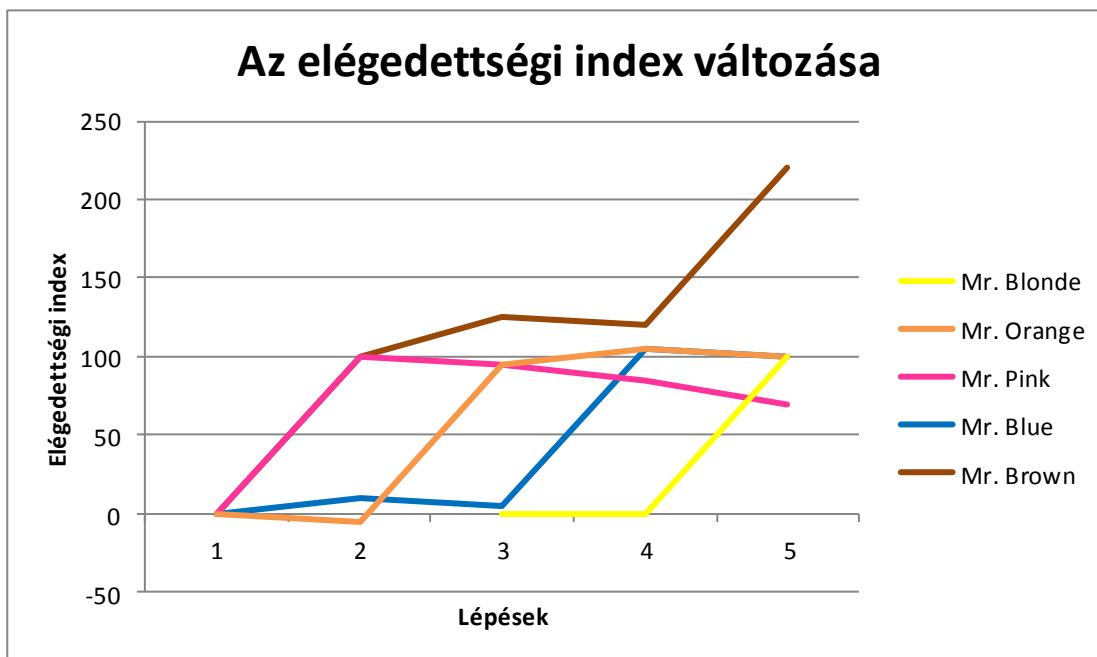
Mr. Pink	Mr. Brown	Mr. Orange	Mr. Blue
Foreigner	Bob Marley	Bob Marley	Queen
Juke Box Hero (Hard rock)	Mr. Brown (Reggae)	Judge Not (Ska)	We Are the Champions (Hard rock)
P!nk Give Me A Reason (Pop)	Stealers Wheel Late Again (Folkrock)	Led Zeppelin Stairway to Heaven (Rock)	George Baker Little Green Bag (Rock)
Pink Floyd Wish You Were Here (Acoustic rock)	Foreigner Juke Box Hero (Hard rock)	Non Phixion The Future Is Now (Hip hop)	Eiffel 65 Blue (Da Ba Dee) (Dance-Pop)

12. ábra A vendégek kedvenc zenéinek a listája

Jól látható, hogy egyetlen közös zeneszám, egy közös előadó, és két közös stílus jelenik meg a lejátszási listákon.

A lejátszási listák fogadása utána a szerveralkalmazás elkészíti az összefésült lejátszási listát. A meglevő algoritmusok közül a szerver azt használja, amelynél az egyes felhasználók elégedettségi indexeinek átlagát növeli, illetve csökkenti a szórást. Kezdetben az összes felhasználó elégedettségi indexe nulla, ezt úgy lehet a legjobban növelni, ha kiválasztjuk a listákon szereplő közös zeneszámot, jelen esetben a Foreigner - *Juke Box Hero* című dalát.

A 13. ábra szemlélteti a szerver által lejátszott művek függvényében az elégedettségi indexet:



13. ábra Az 5 vendég elégedettségi indexének a változása az idő (lépések) függvényében

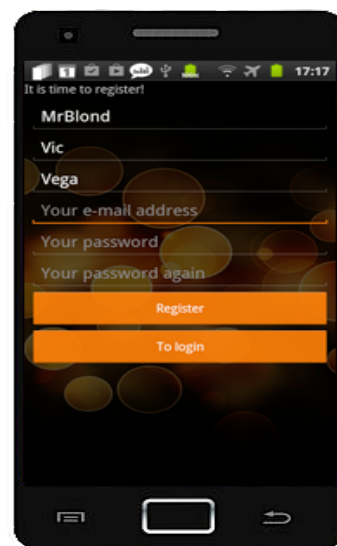
Az első lépés után megfigyelhető, hogy Mr Brown, és Mr. Pink elégedettségi indexe 100 ponttal emelkedett, mert lejátszási listáikon megtalálható a lejátszott mű. Észrevehetjük, hogy Mr. Blue elégedettségi indexe is növekedett 10-el, mivel az ő lejátszási listáján azonos stílusú zenemű is megtalálható, mint az első lépésben lejátszott zene. Mivel a Foreigner - *Juke Box Hero* dalához hasonló szerzemény nem található Mr. Orange listáján, ezért az ő elégedettségi indexe csökken az első lépésben 5 ponttal.

Második lépésnél az algoritmus a szórás csökkentése és az elégedettségi indexek átlagának növelése érdekében Mr. Orange elégedettségi indexét próbálja növelni. Mivel Bob Marley előadó megtalálható Mr. Brown listáján is, ezért Mr. Orange Bob Marley - *Judge Not* zeneszáma lesz kiválasztva.

Hasonló a helyzet a harmadik lépésnél is, azzal a különbséggel, hogy itt Mr. Blue listájáról azért lesz kiválasztva George Baker - *Little Green Bag* című dala, mert Mr. Orange listáján is van Rock műfajú zeneszám.

A harmadik lépés után új felhasználó érkezik a helyiségbe. Az ő neve Mr. Blonde.

Mr. Blonde nem rendelkezik még az alkalmazás mobilos változatával, de ebben a pillanatban ő is telepíti azt. Telepítés után regisztrál (14. ábra) és gyorsan létre tud hozni egy lejátszási listát az által, hogy importálja a telefonjában lementett lejátszási listát. A létrehozott lista a következő: 1. Lil' Kim - *No Time* (Hip-hop); 2. Stealers Wheel - *Late Again* (Folkrock).



14. ábra A telefonos alkalmazás Android operációs rendszeren

A következő lépésben a Mr. Blonde által feltöltött listából lesz kiválasztva egy zenemű, mivel az ő elégedettségi indexe a legalacsonyabb (neki még nem volt alkalma meghallgatni a saját zenéjét, mert most érkezett). A szerver ugyanakkor figyelembe veszi a többi ottlévő felhasználó lejátszási listáiról az információkat, hogy ezt a kiválasztást a legjobban tesse meg. Ezért Stealers Wheel - *Late Again* című dalát választja ki Mr. Blonde lejátszási listájáról (ez a zeneszám jobban tetszik a többieknek, mint Lil' Kim zenéje).

A szerver weboldalról felügyelhető és irányítható. A négy vendég érkezése után elindul Madonna egyik zeneszáma. A helyiségben dolgozó személyzet a szám lejátszása közben a képernyőn ezt látja (15. ábra; a Playlist automatikusan frissülni fog Mr. Blond megérkezésekor):

Kutyaszorító Csapszék is ready for the best music service

Kutyaszorító Csapszék Connected users: 4 Activ Playlists: 4

Player
Madonna - Like a Virgin
01:16 / 04:31

Music visualizer

Events
Mr. Pink logged in.
Mr. Pink sent a new playlist.
Mr. Brown logged in.
Mr. Orange logged in.
Mr. Brown sent a new playlist.
Mr. Blue logged in.
Mr. Blue sent a new playlist.
Mr. Orange sent a new playlist.

Playlist
1. Stealers Wheel - Late Again
2. George Baker - Little Green Bag
3. Pink - Give Me A Reason
4. Led Zeppelin - Stairway to Heaven
5. Eiffel 65 - Blue (Da Ba Dee)

Happy Guests
1. Mr. Pink 100 3
2. Mr. Brown 100 3
3. Mr. Blue 10 3
4. Mr. Orange -5 3

Sad Guests
1. Mr. Orange -5 3
2. Mr. Blue 10 3
3. Mr. Brown 100 3
4. Mr. Pink 100 3

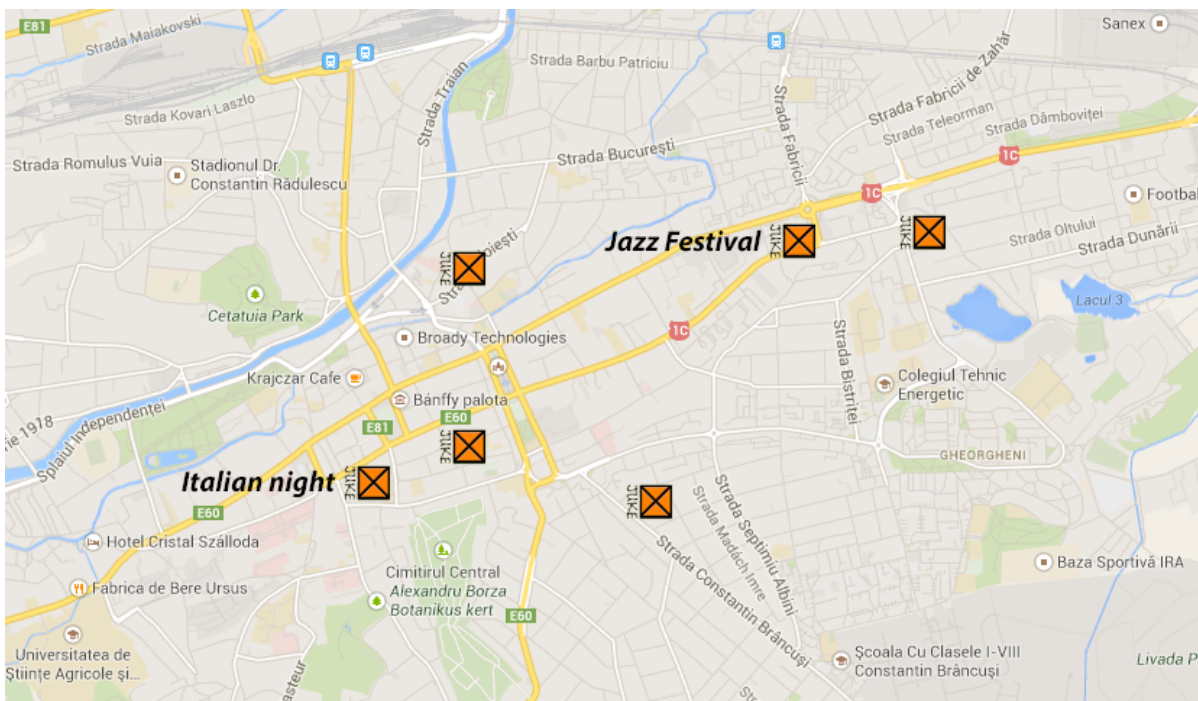
© 2013-2014 Social Jukebox. All Rights Reserved. Rendered in 141 ms.

15. ábra Social Jukebox Server weboldala

5 Következtetések és továbbfejlesztési lehetőségek

A projekt jelen állapotában emeli egy helyiség zenei szolgáltatásának színvonalát. Bevonja a vendégeket is a helyiség hangulatának kialakításába, elősegíti a tulajdonos számára a hely népszerűsítését, a felhasználóknak pedig biztosítja azt az élményt, hogy ők is formálhatják a helyiség zenei repertoárját.

A tervek között szerepel egy térképmodul és egy Events modul hozzáadása az alkalmazáshoz. A térkép célja, hogy a mobilos felhasználó számára megkönnyítse az olyan helyiségek (éttermek, kávézók) keresését, amelyek rendelkeznek a *Social Jukebox* rendszerrel. Így a felhasználók az alkalmazásba integrált térképen láthatnák az egyes helyiségek információit, hogy milyen zenestílust preferálnak, illetve, hogy milyen eseményekre kerül sor a közeljövőben. Az eseményeket az Events modul tartalmazza, és lehetővé tenné, hogy a helyiségek információkat tehessenek közzé a náluk zajló eseményekről. Például, hogy szombat este *Olasz Est* lesz egy bizonyos kávézóban, máshol éppen akkor *Jazz Fesztivál* (16. ábra). Ez azt jelentené, hogy az oda betérő vendégek lejátszási listáiról csak a megfelelő stílusú zeneműveket venné figyelembe a rendszer. A modul segítségével a helyiségek reklámozhatják saját eseményeiket.



16. ábra A térkép és események modul működését illusztráló Google térkép

A szerver felhasználható egy olyan webrádió típusú alkalmazáshoz, amihez bárki böngészőn keresztül csatlakozhat. A felhasználó regisztrálás után saját lejátszási listákat tölthet fel. Ezek alapján a szerver egy audióstreamet sugározna. A felhasználónak lehetősége lenne visszajelzéseket adni a zene minőségéről. A visszajelzések alapján és a lejátszási listák segítségével az összefésülő algoritmus finomhangolható.

Social Jukebox Net megvalósításánál szó volt arról, hogy a felhasználók részleges zeneszám metainformáció küldése esetén javaslatokat kapnak hasonló adatokkal rendelkező zeneszámokról. A javaslatok segítségével kiválasztják, hogy melyik zeneszámmra gondoltak, amikor begépték a részleges vagy hibás szöveget. A jelenlegi mintafelismerésen sokat lehetne javítani, ha Apache Solr⁹ (vagy hozzá hasonló) keresési platformot felhasználnánk.

A *gamification*¹⁰ stratégiát alkalmazva lehetne tovább fejleszteni az alkalmazást, a zenei adatbázist bővíteni és növelni a márkahűséget.

Jelenleg az *Social Jukebox Net*-re új adatokat úgy lehet hozzáadni, hogy megfelelő számú felhasználó felveszi a lejátszási listába ugyanazt a még szerveren nem létező *Előadó*, *Cím* párost. Ezek között lehetnek eltérések, mint például helyesírási elírások, hiányzó második szerző stb. Jövőbeli tervek között szerepel egy olyan felhasználói felület hozzáadása a rendszerhez, melynek segítségével új adatok vihetők be a *Social Jukebox Net* metaadatai közé. Itt a felhasználó tesztek, feladatokat, kvizeket oldana meg, amelyekért pontokat, jelvényeket, kitüntetések és különböző jutalmakat kaphatna, amely mellett, hogy növeli a felhasználói élményt, motiválhatja is a felhasználót.

Egy másik felhasználási mód egy márkahűséget növelő rendszer kidolgozása lenne. A felhasználó jelvényeket, pontokat, és akár valós jutalmakat is kapna feladatok teljesítéséért. Például, ha *Törzsvendég* lenne egy bizonyos helyen, akkor kedvezményt kapna bizonyos termékek vásárlására, vagy az ő lejátszási listáján szereplő számok előnyben részesülnének.

⁹ <https://lucene.apache.org/solr/>

¹⁰ „A legegyszerűbb definíció szerint akkor beszélhetünk gamificationról, ha játékelemeket alkalmazunk az élet játékon kívüli területein, hogy a folyamatokat még érdekesebbé, és lebilincselőbbé tegyük.” - <http://www.gamesforbusiness.hu/blog/mi-az-a-gamification>

6 Köszönetnyilvánítás

A dolgozat ösztöndíjat nyert a Márton Áron Tehetséggondozó Szakkollégiumi Programban és felvételt nyert a Farkas Gyula Szakkollégiumba. Köszönettel tartozunk támogatóinknak: a Balassi Intézet-Márton Áron Szakkollégiumnak, a Farkas Gyula Egyesületnek, a Kolozsvári Magyar Egyetemi Intézetnek, és a Bethlen Gábor Alapkezelő Zrt.-nek. Továbbá köszönet illeti szakmai irányítónkat, dr. Bodó Zalánt.

A dolgozat bemutatásra került a Márton Áron Tehetséggondozó Szakkollégiumi program I. módszertani konferenciáján, a Márton Áron Tehetséggondozó Szakkollégiumi Program beszámoló konferenciáján, a XVII. Erdélyi Tudományos Diákköri Konferencián és az Első kárpát-medencei szakkollégiumi konferencián. A zsűri és a résztvevők tanácsa és javaslatai sokat segítettek a változatos felhasználói igények feltárásában.

Előre is köszönjük a fiatfalvi Alpha Bárnak és a Kaplonyban lévő kocsmakávézó-fagyizóknak, hogy megengedik, hogy alkalmazásunkat éles helyzetben is tesztelni tudjuk helyiségükben.

7 Hivatkozások

- [1] James Steele, Nelson To, *The Android Developer's Cookbook: Building Applications with the Android SDK*, Pearson Education, 2010.
- [2] ***, Android ADT plug-in Eclipsehez, utolsó megtekintés dátuma: 2014.04,
<http://developer.android.com/tools/sdk/eclipse-adt.html>
- [3] ***, Eclipse, utolsó megtekintés dátuma: 2014.04, <http://www.eclipse.org/ide/>
- [4] ***, Xcode, utolsó megtekintés dátuma: 2014.04 <http://en.wikipedia.org/wiki/Xcode>
- [5] ***, iOS Dev Center, utolsó megtekintés dátuma: 2014.04
<https://developer.apple.com/devcenter/iOS/index.action>
- [6] Andrew Troelsen: *Pro C# 2010 and the .NET 4 Platform*, 2010,
- [7] Joseph Albahari, Ben Albahari: *C# 5.0 in a Nutshell, 5th Edition*, 2012,
- [8] ***, Windows Phone Dev Center, utolsó megtekintés dátuma: 2014.04,
<https://dev.windowsphone.com/en-us>,
- [9] Adam Nathan: *WPF 4 Unleashed*, 2010
- [10] ***, NetBeans, utolsó megtekintés dátuma: 2014.04, <http://en.wikipedia.org/wiki/NetBeans>
- [11] Hans Bergsten: *JavaServer Pages, 3rd Edition*, 2003
- [12] ***, MySQL, utolsó megtekintés dátuma: 2014.04, <http://www.mysql.com/>
- [13] ***, Apache Maven, utolsó megtekintés dátuma: 2014.04, <http://maven.apache.org/>
- [14] ***, Logback, utolsó megtekintés dátuma: 2014.04, <http://logback.qos.ch/>
- [15] ***, Apache Tomcat, utolsó megtekintés dátuma: 2014.04, <http://tomcat.apache.org/>
- [16] ***, JSON, utolsó megtekintés dátuma: 2014.04, <http://en.wikipedia.org/wiki/JSON>
- [17] ***, Redmine, utolsó megtekintés dátuma: 2014.04, <http://en.wikipedia.org/wiki/Redmine>
- [18] ***, Git, utolsó megtekintés dátuma: 2014.04, [http://en.wikipedia.org/wiki/Git_\(software\)](http://en.wikipedia.org/wiki/Git_(software))