XVII. reál- és humántudományi Erdélyi Tudományos Diákköri Konferencia (ETDK)

Kolozsvár, 2014. május 15–18.

Cross-platform borértékelő mobilalkalmazás

Szerző:
Kiss Kincső
Babeş-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar,
Informatika szak, 3. évfolyam

Témavezető:
Dr. Simon Károly
egyetemi adjunktus, BBTE, MIK,
Magyar Matematika és Informatika Intézet
képzési tanácsadó, Codespring Kft.

Kivonat

A dolgozat központi témája a WineHunter cross-platform mobil alkalmazás.

A WineHunter projekt célja egy mobilkészülékeken futó alkalmazás kifejlesztése, amely lehetővé teszi borkedvelők és borszakértők számára az általuk kóstolt borok jellemzését és értékelését, a borkóstolás alapvető lépéseinek és meghatározott kritériumainak figyelembevételével, a vonatkozó nemzetközileg elismert módszerek alapján.

Az alkalmazás biztosít felhasználójának egy módosítható borlap sablont, amelynek alapján kóstolt borairól részletes elemzést készíthet, és több kapcsolódó funkcionalitást elérhet. Legfontosabb tulajdonsága, hogy cross-platform, több mobil platformra is kiadható a kód módosítása nélkül. Egyaránt használható például Android és iOS operációs rendszerrel rendelkező készülékeken.

A dolgozat a projekt megvalósításához felhasznált módszereket, eszközöket és technológiákat ismerteti, majd röviden bemutatja az alkalmazás szerkezetét, megvalósításának fontosabb részleteit és ismerteti működését.

Tartalomjegyzék

Bevezető

A dolgozat központi témája a WineHunter projekt keretein belül megvalósított cross-platform mobilalkalmazás, ennek elméleti és technológiai háttere, felépítése és működése.

A szőlőtermelés és borkészítés több száz éves hagyománnyal rendelkezik, és jelentős részét képezi a mezőgazdaságnak. A borismeretnek fontos szerepe van a gasztronómiában. A különböző íz világokhoz kiválasztott borok fontos szerepet játszanak a vendéglátásban, a képzett borajánlók keresett szakemberek a területen. A megfelelő mennyiségben történő borfogyasztásnak bizonyítottan kedvezőek az élettani hatásai. A borvidékeknek fontos szerepe van a turizmusban, a borturizmus egy feltörekvőben lévő ágazata a területnek.

Az emberek szívesen fogyasztanak bort étkezéskor, egészségügyi okokból, vagy szórakozásként, de ahhoz, hogy valaki a borfogyasztást kulturális élményként élje meg, szüksége van valamilyen szintű szaktudásra.

Számtalan lehetőség van arra, hogy valakiből hozzáértő fogyasztó váljon. Az érdeklődők rendelkezésére állnak könyvek, szaklapok, tanfolyamok, interneten található információk, és a tanulás fontos része maga a borkóstolás is.

A pincészetek és a borkedvelőket tömörítő szervezetek gyakran szerveznek borkóstolókat, ahol borászok, vagy borszakértők mutatnak be borokat az érdeklődőknek. Az ilyen eseményeken a résztvevők egy borlapot kapnak, amelyet a kóstolás folyamán ki kell tölteniük. Mielőtt még elkezdődne a tényleges kóstolás, először az üveg címkéjéről leolvasható adatokat kell leírni (a bor neve, évjárata, borászat neve stb.). Ezután következik a kóstolás három fontos lépése: a szín, az illat és az íz megfigyelése. A borlap előre meghatározott sablon alapján lehetőséget ad a különböző tulajdonságok pontos felvezetésére, kitöltésével a fogyasztó részletes értékelést készíthet az általa kóstolt borról. Az ilyen jellegű borlapok kitöltése elvárás is bizonyos borajánlói szakfokozatok elérésekor, adott fokozatok megszerzéséhez akár több ezer ilyen elemzést is kérhetnek.

A WineHunter projekt alapötlete a fentebb leírtak alapján született. Az emberek jelentős része ma már rendelkezik okostelefonnal vagy tablettel. A borértékelés folyamata kényelmesebbé tehető úgy, hogy a fogyasztó saját mobilkészülékének segítségével értékeli az aktuálisan kóstolt bort, így nincs szükség a borlapok használatára és utólag könnyebben kereshetőek és karbantarthatóak az adatok.

A WineHunter alkalmazás ehhez szolgáltat a felhasználónak egy könnyen kezelhető felületet, amelyen gyorsan, standard borlap sablon alapján értékelhet egy adott bort, majd a

kész jellemzést elmentheti, és utólag visszanézheti, esetleg módosíthatja. A vonatkozó szakértői javaslatok alapján készült alapértelmezett sablonon kívül, készíthető saját, testreszabott borlap is, amelyből kiszűrhetőek például a fogyasztó által soha nem értékelt tulajdonságok. A korábbi kóstolások listája könnyen átlátható, rendezhető saját kritériumok alapján, és lehetőség van gyors keresésre a borkóstolás során megadott bármilyen kritérium alapján.

Az alkalmazás lehetőséget ad képek hozzárendelésére a kóstolt borokhoz, és további járulékos funkcionalitásokat is biztosít, például segítségével a fogyasztó saját borkészletét is menedzselheti.

Az alkalmazás fontos tulajdonsága, hogy cross-platform megoldás, olyan technológiák felhasználásával készült, amelyek lehetővé teszik futtatását különböző platformokon, a forráskód változtatása nélkül. Például, Android és iOS operációs rendszereket használó készülékeken egyaránt használható.

Egy rövid bevezető után a borkóstolás világába, amely az alkalmazás témájának alapja, a dolgozat a fejlesztés során használt eszközöket, módszereket és technológiákat mutatja be. Ezután ismerteti a projekt szerkezetét és megvalósításának fontosabb elemeit, majd az alkalmazás működését. A dolgozat végén, az elért eredmények összefoglalása után a továbbfejlesztési lehetőségekről és tervekről esik szó.

Az alkalmazás létrejöttéért köszönet illeti a Codespring Kft.-t és a kolozsvári Borvadász társaságot. A Codespring biztosította a fejlesztéshez szükséges infrastrukturális hátteret, a Borvadász társaság szakemberei a borértékeléssel kapcsolatos elméleti hátteret szolgáltatták és az alkalmazás tesztelésében is közreműködtek.

1. A borkóstolás fogalomtára

A szőlőtermesztés és borkészítés hagyománya évezredekkel ezelőttre nyúlik vissza, és a Kaukázuson túli területekről származik. Európában a bor elkészítési módjának a megismerése az ókori görög és római kereskedőknek köszönhető, valamint a kereszténység elterjedésének, ahol a bor a szertartások fontos kellékének számított. Később a gyarmatosítások következtében a borkultúra az egész világon elterjedt, és szerves részévé vált a mezőgazdaságnak és az emberiség mindennapjainak. Jelenleg a világon feltehetően 10 millió hektáron történik szőlőtermesztés és borkészítés, aminek a 71%-a Európában található [1]. A termelt bormennyiség egyharmada a három vezető bornagyhatalomban készül, Franciaországban, Olaszországban és Spanyolországban.

A borkultúra tanulmányozása, a borok megismerése és minőségi fogyasztásának megtanulása mára egy népszerű tudományág lett a borkedvelők körében. Számos nemzetközileg elismert intézmény és szervezet foglalkozik a borismeret népszerűsítésével. A tanulni vágyóknak képzések, tanfolyamok, könyvek, szaklapokat, folyóiratokat állnak rendelkezésükre, amelyek segítségével elsajátíthatják a borkészítés technológiáit, és hozzáértő fogyasztóvá válhatnak. A képzések lehetőséget biztosítanak szintek elérésére, amelyek által valakiből igazi borszakértő válhat, akik manapság keresett és értékelt szakemberek.

A borok jellemzése és értékelése számtalan tényező figyelembe vételével történik. A borkészítés legmeghatározóbb tényezője a szőlőfajta, ami lehet fehér vagy kék. A fehér szőlőből fehér borok készülnek, míg a kékből, a vörös borokon kívül, készíthető rosé, siller vagy fehér bor. Ezen kívül az elkészítést számtalan tényező befolyásolja, mint a szőlőtermő vidék klímája, a talaj, valamint a felhasznált eszközök és technológiák.

A borismereti tanfolyamok és képzések során, az elméleti tananyag elsajátítása mellet, a legfontosabb szerepe a borkóstolásnak van. A könyvekből megtanult értékelési szempontok és folyamatok gyakorlatban történő alkalmazása teszi lehetővé, hogy valakiből hozzáértő fogyasztó váljon.

A borkóstolások alkalmával a környezeti tényezők nagyban befolyásolják az élményt, fontos szerepet játszanak a kóstolások sikerében. Ilyenek a megválasztott helyszín hőmérséklete, a kóstoláshoz használt poharak, a borok hőmérséklete stb.

A borok kóstolása során először a szemmel látható tulajdonságok megfigyelése történik, a bor színének és intenzitásának, valamint tisztaságának meghatározása. Ezután következik az illatával kapcsolatos tulajdonságok meghatározása, végül maga az ízlelés.

A részletes jellemzés elkészítéséhez a kóstolók egy standardizált borlapot kapnak, amelyre először a borok címkéjéről leolvasható adatokat kell felvezetni, még a tényleges kóstolás előtt, majd a megfigyelendő tulajdonságok sablonja segítségével elkészíthető a részletes értékelés. Annak ellenére, hogy egy helyes értékelés igényel némi szaktudást, egy bor jellemzése során nem mindig egyeznek meg az értékelések, mivel a borkóstolás alapjában szubjektív, és a kóstolóban keltett hatások egyénenként változnak.

Ahhoz, hogy valakiből elismert borszakértő váljon, különböző szinteket kell elérnie borismereti tanfolyamok, képzések keretein belül, aminek alapfeltétele az elméleti ismeret és ennek gyakorlatban történő helyes használata. Így akár több ezer értékelés (kitöltött értékelési lapokkal alátámasztva) is szükséges lehet egy fokozat megszerzéséhez.

Az értékelések elkészíthetőek különböző borpincészetek látogatása során, éttermekben, vagy bármikor borfogyasztás során. A borlapok kitöltése viszont kényelmetlen lehet egy kirándulás vagy vacsora közben. Ezen kívül nagymennyiségű kitöltött értékelési lap kezelése, az előző értékelések közötti keresés is nehézkessé válhat. Az említett okokból merült fel egy borlapot helyettesítő szoftver elkészítésének lehetősége, amely könnyen használható mobileszközökön, általa értékelhető és egy helyen tárolható az értékelt borok listája. Több ilyen jellegű alkalmazás elkészült már, de szakértői vélemények alapján, mindegyiknek vannak hiányosságai. Ezeknek a hiányosságoknak a kiküszöbölés érdekében indult el a WineHunter projekt, amelynek célja egy cross-platform mobilalkalmazás kifejlesztése, amely a nemzetközi standardoknak megfelelő borlap sablon alapján nyújt lehetőséget borok részletes jellemzésére és értékelésére bármilyen környezetben.

2. Felhasznált eszközök és módszerek

Egy összetett projekt elkészítésének folyamata során, szükség van különböző fejlesztői eszközök és módszerek használatára, amelyek elősegítik a munkafolyamat gyorsaságát és sikerességét. A felhasznált eszközök és folyamatok a projekt típusától függetlenek, és jelentős szerepet töltenek be a projekt tervezésében, kivitelezésében, valamint karbantartásában. Fontos a forráskód változásainak nyomon követhetősége és archiválása, amelyhez verziókövető rendszer használata szükséges, a feladatok rendszerezése, amely projektmenedzsment rendszerek segítségével megvalósítható, a rendszer automatizált felépítése és függőségeinek megoldása, amely build rendszerek segítségével történik.

A verziókövetők archiválják a programkód összes változatát, lehetőséget biztosítanak korábbi verziókra való visszatérésre, kiadások és jogosultság kezelésre, valamint különböző fejlesztési ágak menedzselésére. Ezzel elősegítik a párhuzamos munkát, valamint a fejlesztési folyamat könnyű áttekintését. A Mercurial [2] egy ingyenes, GNU GPL v2 licensz alatt kiadott, Pythonban írt, osztott verziókövető rendszer. A kliens-szerver modellre épülő verziókövetőkkel ellentétben, amelyek egy központi szerveren tárolják a módosításokat, a Mercurial biztosít egy lokális másolatot, ami tartalmazza a teljes fejlesztéstörténetet, és lehetővé teszi a tároló használatát hálózati hozzáférés hiányában is. Először szükséges egy másolat készítése (*clone*) a lokális tárolóba a projekt meglévő verziójáról, majd a változtatásokat is ide kell először feltölteni (*commit*). A saját tároló tartalma megosztható minden fejlesztővel, a központi tárolóba történő feltöltéssel, de adható hozzáférési jog csak bizonyos fejlesztőknek a saját tárolónkhoz, akár letöltési vagy feltöltési lehetőséggel egyaránt. A TortoiseHg biztosít a Mercurialhoz egy intuitív felhasználói felületet, amely által könnyen nyomon követhetőek és kezelhetőek a projekt lépései és verziói. A WineHunter esetében a Mercurial lokális tároló biztosította a feladatok teljes befejezése előtti verziókövetést, anélkül, hogy a központi tárolóba fel kellett volna tölteni egy félkész vagy hibás változatot.

A Readmine [3] egy ingyenes, GPL v2 licensz alatt kiadott, nyílt forráskódú, Rubyban írt webes projektmenedzsment és hibakövető eszköz. Lehetőséget biztosít projektek fejlesztése során munkamenetek létrehozására, feladatok felvezetésére és szétosztására a fejlesztők között, a projekttel kapcsolatos dokumentumok tárolására, a határidők és a fejlesztés haladásának nyomon követésére alkalmas diagramok készítésére, a felmerülő problémák megosztására és megtárgyalására, a projektben történő változások figyelésére stb. A WineHunter projekt fejlesztése során a Redmine eszközzel történt a feladatok kezelése és a

munkafolyamaton belüli határidők beosztása, valamint a felmerülő hibák dokumentálása és megoldásainak nyomon követése.

A NodeJS [4] egy olyan C/C++ - ban írt szerver oldali JavaScript környezet, amely lehetővé teszi skálázható hálózati alkalmazások írását, valamint a Google V8 Javascript motorja felett aszinkron, esemény alapú I/O rendszer használatát [5]. A telepítés után elérhetővé válik az *npm*, amely a Node.js hivatalos csomagkezelője, és amelynek használatával könnyen telepíthető, listázható és törölhető a Node minden modulja.

A Yeoman [6] egy olyan nyílt forráskódú, kliens-oldali alkalmazások fejlesztését elősegítő eszköztár, amely magába foglalja a *yo, grunt* és *bower* eszközöket, amelyek különálló, de együttműködő NodeJS modulok. A Yeoman széleskörű funkcionalitásai és könnyen konfigurálhatósága nagyban megkönnyíti a web alkalmazások készítését.

A Yo biztosít különböző sablongenerátorokat, amelyek segítségével generálható a projekt váza, és ezzel felgyorsítható a fejlesztési folyamat. Ez a váz lehet konfigurációs állomány, fejlesztést elősegítő szkript, vagy akár a projekt teljes sablonja. Több hivatalos generátort biztosít, amelyek telepítése után létrehozhatóak AngularJS [7], RevealJS [8], ExpressJS [9] stb. alapú alkalmazás vázak, de lehetőséget biztosít saját generátorok írására, amelyekkel teljesen testreszabható ez a folyamat.

A GruntJS [10] egy olyan JavaScript alapú alkalmazások fejlesztése során használt feladatfuttató eszköz, amely lehetővé teszi a buildelés és telepítés során ismétlődően elvégzendő műveletek meghatározását, testreszabását és automatizálását. Számos beépített funkcióval rendelkezik, mint például a JavaScript és CSS fájlok összekötése, tömörítése, minimalizálása, képek tömörítése, előfordítók használata, szintaktikai ellenőrző eszközök és különböző tesztek futtatása, valamint livereload web szerver elindítása az alkalmazás teszteléséhez stb. Használatához a package.json leíró állományban kell megadni függőségként, miután a NodeJS csomagkezelőjének segítségével globálisan telepítve lett. Az elvégzendő feladatokat egy Gruntfile.js állományban kell meghatározni. A feladatok elvégzése a *grunt* parancs futtatásával történik.

A Bower [11] egy olyan függőségkezelő eszköz, amely segítségével egyszerűen letölthetőek és kezelhetőek a fejlesztés során használni kívánt csomagok és függőségek. Lehetőséget biztosít csomagok keresésére, telepítésére és frissítésére parancssorból, a CSS és JavaScript függőségeket egy lokális *bower_components* mappába telepíti. A használatának egyik nagy előnye abból származik, hogy egy bower.json konfigurációs állományban megadva a kívánt csomagokat és azok verzióját, a függőségek megoldása elvégezhető egyetlen parancs lefuttatásával (*bower install*). A "~" prefix használatával a verziószámok előtt a legfrissebb változatok lesznek letöltve abban az esetben, ha visszafele kompatibilisek.

A WineHunter fejlesztése során a *Yo* által biztosított AngularJS alapú generátor segítségével történt az alkalmazás vázának generálása, a *Grunt* biztosította a projekt felépítését (build) és a feladatkezelést, a *Bower* volt használva a különböző függőségek kezelésére.

A WineHunter alkalmazás készítése során a kódelemzés és javítás a JSHint eszköz segítségével történt. A JSHint [12] egy JavaScriptben írt, nyílt forráskódú statikus kódelemző szoftver, amely a NodeJS csomagkezelőjének segítségével telepíthető. Lefuttatásával kiszűri a szintaktikai hibákat, valamint az esetleges problémaforrásokat (pl. változók esetén a definiálás hiánya stb.) a JavaScript kódban. A JSHint rugalmas és konfigurálható egy .jshintrc állomány létrehozásával, amelyben megadhatóak a figyelni kívánt hibák (pl. indentálás, kódblokkok elején és végén megadott kapcsos zárójelek hiánya stb.). Az alapértelmezett szabályokon kívül lehetőség van adott fájlokhoz külön ellenőrző állományokat hozzárendelni. A Grunt-nak megadható feladatként, hogy minden build esetében végrehajtódjon a kódellenőrzés.

A WineHunter fejlesztése a SublimeText [13] szerkesztőben történt, amely könnyen konfigurálható és kiterjeszthető. Egyaránt használható Mac OS X, Windows és Linux operációs rendszereken.

Az Xcode az Apple által, Mac OS X operációs rendszerre kifejlesztett programfejlesztő csomag. Legfontosabb része az Xcode IDE, amely többek között lehetővé teszi iPhone alkalmazások készítését és futtatását emulátorokon, valamint iPhone, iPad és iPod készülékeken. A WiheHunter alkalmazás iOS operációs rendszert használó készülékekre való telepítése és futtatása az Xcode segítségével történt.

3. Felhasznált technológiák

3.1 Apache Cordova és PhoneGap

A PhoneGap [14] mobilalkalmazások fejlesztésére alkalmas nyílt forráskódú keretrendszer, amely lehetővé teszi platform-specifikus nyelvek helyett webes technológiák használatát natív mobilalkalmazások fejlesztésénél. HTML, CSS3 és JavaScript segítségével készíthetőek olyan alkalmazások, amelyek nem egyszerűen böngészőben futnak, hanem telepíthetőek adott platformokra és feltölthetőek a platformoknak megfelelő alkalmazásboltokba.

A PhoneGap projektet a Nitobi vállalat kezdte el fejleszteni, amelyet 2011 októberében felvásárolt az Adobe. Az Apache Software Foundation-nel együttműködve, a projekt alapkódját felhasználva közzétették Apache Licence 2.0 alatt, és létrehozták ezzel az Apache Callback projektet. Ezt a nevet rövid időn belül Apache Cordová-ra változtatták, mivel túl általánosnak találták. A PhoneGap márkanév megmaradt, a Cordova zárt kódokat is tartalmazó kiadványának neveként. Mivel a PhoneGap nyílt forráskódú része teljesen megegyezik a Cordovával, a dolgozat további részeiben a két név szinonimaként is előfordul.

A PhoneGap jelenleg az Apache Cordova egy disztribúciója, amely kiegészíti a nyílt forráskódú szoftvert több extra funkcióval. A PhoneGap első verziói támogatták a Google Android, Blacberry, Apple iOS, Nokia Symbian, LG WebOS, Microsoft Windows Phone és Bada operációs rendszereket, majd a 2.0-ás verzióba integrálták a Windows Phone 7 és 8, valamint a Windows 8 platformtámogatást. Később az Intel és Adobe együttműködésével a Tizen nyílt forráskódú operációs rendszer támogatását is megoldották. A jelenlegi legújabb verzió (3.4.0) segítségével Amazon FireOS, BlackBerry 10, Firefox OS és Ubuntu operációs rendszerekre is készíthetőek applikációk, viszont a BlackBerry, Bada, Symbian és WebOS már nem támogatott.

Különböző JavaScript keretrendszerek használatával (például jQuery mobile) natív megjelenítést lehet kölcsönözni az alkalmazásoknak. A PhoneGap különböző plugin-okat biztosít a fejlesztők számára, amelyek hozzáadásával a projekthez használhatóvá válnak a készülék hardverelemei és szenzorjai. Lehetőség van a kamera használatára, hang és videofelvétel készítésére és lejátszására, internethasználatra, földrajzi pozíció meghatározására, gyorsulásmérő szenzor használatára, saját állományok tárolására és

megnyitására, valamint a készülék és a rendszer adatainak, illetve időzónával és nyelvhasználattal kapcsolatos információknak a lekérésére.

A PhoneGap a 3.0-ás verziójától kezdve két alapfolyamatot biztosít mobilalkalmazások létrehozására. Bármelyik megoldást választja a fejlesztő, a projektnek lesz egy *.cordova* mappája, egy *config.xml* leíró állománya, amely tartalmazni fogja az applikáció általános adatait (azonosító, verziószám, név, amely megjelenik majd a mobiltelefonon és a platformnak megfelelő alkalmazásboltban, rövid leírás és a szerző azonosítója). Itt lesz megadva a tartalom elérési útvonala, a külső hozzáférések engedélyezése, valamint a specifikus paraméterek, amelyek meghatározzák, hogy hogyan fog működni az alkalmazás a készüléken (orientáció, teljes képernyő használata stb.). Lesz egy *index.html*, amely az applikáció kezdőoldala. Ebben kell megadni a hivatkozást a phonegap.js-re vagy cordova.js-re ahhoz, hogy a PhoneGap által biztosított szolgáltatások elérhetőek legyenek. Ide kerülnek a CSS-re, a logika megvalósítására használt JavaScript fájlokra, valamint a fejlesztés során használt erőforrás állományokra való hivatkozások. A projektszerkezet tartalmazni fog egy-egy *merges*, *plugins*, *platforms* és *www* mappát, amelyekbe a projekthez hozzáadott PhoneGap pluginok, támogatott platformoknak megfelelő állományok és statikus erőforrásfájlok kerülnek.

Az alap munkafolyamat a *Web Project Dev*, amely a Cordova Command-Line Interface (CLI) parancsainak hívásával történik. Az eszköz segítségével létrehozható egy Cordova applikáció, és minimális platform-specifikus változtatással telepíthető bármely támogatott mobil operációs rendszerre.

Ahhoz, hogy a Cordova CLI parancsait használni lehessen terminálból, először telepíteni kell a NodeJS-t. A Cordova és a PhoneGap a NodeJS modulok közé tartozik, így Linuxon a *sudo npm install -g cordova* és *sudo npm install -g phonegap* parancsok lefuttatásával az npm globálisan telepíti a Cordova-t és a Phonegap-et.

A hozzáadott modulok használhatóak a fejlesztő által választott bármelyik mappában, így a *cordova create* parancs hívásával létrehozható egy Cordova projekt. A parancsnak paraméterként meg kell adni egy mappa nevét, amelybe generálni fogja a projektet, opcionálisként megadható az alkalmazásnak egy azonosító, valamint a projekt neve. Ezeket később módosítani lehet a *config.xml* projektleíró állományban.

A különböző támogatni kívánt platformok hozzáadása, eltávolítása és listázása szintén a CLI segítségével történik. A hozzáadott platformok a projekt *platforms* mappájában kerülnek egy, a platform nevével ellátott alkönyvtárba. Az alkalmazás buildelhető és futtatható, a támogatni kívánt platformnak megfelelő SDK telepítése után.

A cross-platform alkalmazások alaptulajdonsága, hogy egységesen kezelik a különböző funkciókat, a felületet, valamint a hozzáadott pluginokat. Ennek ellenére néhány esetben nem lehet elkerülni a saját komponensek, képek, ikonok megjelenítésével kapcsolatos beállítások használatát ahhoz, hogy platformtól függően natív megjelenést kölcsönözzünk az alkalmazásnak. Például, Android készülékeken van dedikált „vissza” (back) gomb, Apple mobiltelefonokon nincs. Az ilyen különbségeket külön kell kezelni az alkalmazás fejlesztése során és a PhoneGap lehetőséget ad erre a *merges* mappa segítségével, amelyben elhelyezhetőek a platformfüggő megvalósítások. Az alkalmazás telepítése után az adott platformnak megadott stílust fogja használni.

A másik típusú munkafolyamat a *Native Platform Dev*, amely egy alacsonyabb szintű parancssor felület, a *Plugman* használatával történik. Ezt a folyamatot akkor ajánlott használni, ha az alkalmazás csak egy adott platformot fog támogatni, és hozzá szeretne férni az adott platform alacsonyabb szintű tulajdonságaihoz.

A Cordova Command-Line Interface előnye cross-platform alkalmazások készítésénél, hogy létrehoz egy alap szerkezetet a projektnek, amelyet később tetszés szerint lehet módosítani, valamint minimális platform-specifikus beállítás segítségével hozzáadhatóak a különböző PhoneGap által támogatott pluginok, megkönnyítve a fejlesztők munkáját.

A WineHunter fejlesztéséhez az alkalmazás vázának kigenerálása a Yeoman segítségével történt, de mivel a szerkezet módosítható, lehetővé vált a Cordova CLI által létrehozott projektszerkezet és a kigenerált váz összevonása, ami biztosította a továbbiakban a WineHuner Cordova projektként történő kezelését, valamint az ezzel járó előnyök felhasználását.

A PhoneGap keretrendszer egyik legnagyobb előnye a webes alkalmazásokkal szemben, hogy rendelkezik egy olyan eszköztárral, amely natív eszközökhöz és szenzorokhoz való hozzáférést támogató API-kat biztosít a fejlesztők számára. Ezek tulajdonképpen pluginok, amelyek tartalmazzák a támogatott platformoknak megfelelő natív kódkönyvtárakat, és ezek segítségével biztosítják a kommunikációt az alkalmazás és a platform eszközei között. A keretrendszer modularitásának köszönhetően, elegendő csak a használni kívánt pluginok hozzáadása az alkalmazáshoz, nem szükséges a teljes eszköztár injektálása.

A pluginok hozzáadása az alkalmazáshoz a Cordova CLI vagy a Plugman segítségével történik. A Cordova CLI *cordova plugin add* parancs hívásával adhatóak hozzá az alkalmazáshoz a használni kívánt eszközök és szenzorok API-jai, valamint a *cordova plugin remove* parancssal törölhetőek azok.

Az alkalmazáshoz hozzáadott pluginoknak szerepelniük kell a legmagasabb szintű *plugin.xml* leíró állományban, amelynek tartalmaznia kell az adott plugin azonosítóját,

valamint a közös JavaScript interfész elérését. Ez utóbbi fogja biztosítani a kommunikációt a készülék natív eszközei és a felület között. A leíró állományban meg kell határozni a támogatni kívánt platformot és a neki megfelelő natív kódot tartalmazó fájlokhoz tartozó elérési útvonalakat.

A WineHunter alkalmazás fejlesztésében jelentős szerepe van PhoneGap által biztosított eszköztárnak. A különböző platform specifikus eltérésekből adódó problémák megoldásához szükséges a készülék specifikus tulajdonságaihoz való hozzáférés, amit a *Device* plugin tesz lehetővé, a kóstolásokhoz hozzárendelt képek készítése a készülék beépített kamerájával történik, amelyhez a *Camera* plugin biztosít hozzáférést. Ezek mellett az alkalmazás egyik legjelentősebb funkcióját, a kóstolások exportálását és importálását a PhoneGap *File* pluginja teszi lehetővé, amely segítségével elérhetővé válik a készülék fájlrendszere, annak módosítása és előzőleg exportált saját állomány importálása.

3.2 JavaScript keretrendszerek

A jQuery [15] egy olyan nyílt forráskódú, cross-platform JavaScript keretrendszer, amely funkcionalitásokban gazdag, szintaxisa miatt egyszerűen használható, és lehetővé teszi a HTML dokumentumok bejárását és manipulálását, valamint események és animációk kezelését web alkalmazások fejlesztése során. Napjaink egyik leghasználtabb JavaScript keretrendszere, amelyet részletes dokumentációja, egyszerűsége és könnyen használhatósága tett népszerűvé a fejlesztők körében.

A jQuery egyik fontos újítása, hogy lehetőséget biztosít a HTML dokumentum elemeinek kijelölésére, anélkül, hogy egyenként be kellene járni és megkeresni az adott elemet. A kijelölés a “$” jQuery definiáló karakter után zárójelben megadott elem segítségével történik. A kijelölés csak akkor eredményes, ha az adott kódrészlet csak a HTML dokumentum létrejötte után hívódik meg, és a keresett elem már létezik a DOM-ban (Document Object Model). Ennek ellenőrzésére is biztosít lehetőséget a keretrendszer egy, a “$” után megadott, üres függvény segítségével, valamint a $(document).ready() hívásával, amely a JavaScript *onload()* metódusának felel meg.

A keretrendszer lehetőséget biztosít a HTML dokumentum szerkezetének vagy tartalmának lekérésére és módosítására. A módosítások köthetőek kijelölt elemekhez és eseményekhez. Új csomópont hozzáadása az *add()*, eltávolítása az elemhez hozzáfűzött *remove()* eljárással történik.

A jQuery másik nagy előnye, hogy lehetőség van a dokumentum css-ben megadott tulajdonságainak visszatérítésére és manipulálására, a *css()* függvény segítségével. A

metódusnak paraméterként a változtatni kívánt tulajdonság nevét és értékét kell megadni, majd egy kiválasztott elemhez hozzáfűzve hajtható végre a változtatás.

A jQuery keretrendszer segítségével dinamikusan kiválthatóak és kezelhetőek lesznek a dokumentummal kapcsolatos olyan események, mint az egérkattintás, a billentyűk lenyomása, az oldal betöltése vagy elhagyása, űrlapok elküldése stb. Mivel nagyon sok eseménytípus van, és előfordul, hogy ezek ugyanannak a logikának az alapján lesznek kezelve, a keretrendszer biztosít egy *on()* és egy *off()* függvényt, amelyek megoldást nyújtanak az események egységes kezelésére. Paraméterként meg kell adni a kezelni kívánt egy vagy több esemény nevét és egy függvényt, amely akkor fut le, ha az adott esemény kiváltódik. Az *on()* függvény visszatéríti a kezelt esemény objektumát, amely további információkat ad az eseményről (például kattintás esemény dokumentumon belüli pozícióját stb.). A függvényen belül megadott kódrészlet abban az esetben lesz végrehajtva, ha az adott esemény kiváltódott a dokumentumnak azon az elemén, amelyhez az *on()* függvény hozzá lett rendelve. Az *off()* függvény megszünteti a paraméterként megadott események figyelését.

A jQuery társprojektje a jQuery UI [16] számos animációt és widget-et biztosít a fejlesztők számára, amelyek a vizuális megjelenítést teszik interaktívabbá és színesebbé. A legfontosabb animációk közé tartoznak a különböző effektek, az ablak átméretezése, a drag and drop és a kiválasztás. A jQuery UI widget-jei biztosítanak gombokat, slider-eket, tooltip-eket, naptárat, párbeszéd ablakokat, automatikus kitöltést szövegbevitel esetén, valamint sok más olyan eszközt, amelyekkel biztosítható a megfelelő felhasználói felület cross-platform alkalmazásokhoz.

A jQuery által szolgáltatott lehetőségek teszik lehetővé a WineHunter projektben az események hatására történő felületmódosítások dinamikus kezelését, a DOM futási időben történő változtatását. A jQueryUI widgetjei növelik a felhasználói élményt az alkalmazás használata során, biztosítják például a sliderrel történő tulajdonság értékének meghatározását és a datepicker-ből történő dátumválasztást.

Az AngularJS [17] egy nyílt forráskódú, a Google által kiadott model-view-controller (MVC) JavaScript keretrendszer, amely lehetővé teszi a web alapú alkalmazások fejlesztését szerver oldal használata nélkül. A keretrendszer segítségével a projekt modulokra osztható, az adatkezelést és a modell felülethez való kötését teljesen kliens oldalra helyezi át.

A tartalmat, vagyis a modell réteget, a JavaScript változók szolgáltatják, a kontroller réteget a JavaScript függvények, a view réteget a HTML kód. Az utóbbit az AngularJS kiegészíti saját attribútumokkal, amelyek segítségével megadható, hogy milyen adatok jelenjenek meg, valamint, hogy azok milyen eseményekre hogyan reagáljanak. Legnagyobb újítása a kétirányú adatkötés, amely a *$rootscope* vagy *$scope* attribútumokhoz hozzárendelt

változókon keresztül megvalósított kommunikáció a felület és a kontroller között. Eredményeként, ha a modell változik, akkor a felületen megjelenített érték is változik, és fordítva. Minden kontroller saját *scope*-pal rendelkezik, a *rootscope* bárhol egységesen elérhető.

Az AngularJS biztosít különböző direktívákat, valamint lehetőség van saját direktíva írására is. Segítségével viselkedési formákat, irányelveket, vagy a DOM módosítását adhatjuk meg. Ezek a futtatás során lesznek összevetve a már meglévő HTML kóddal majd végrehajtva azon. A direktívák használhatóak több helyen, ezáltal kiszűrve a kód ismétlődését.

Az keretrendszer által biztosított vagy saját filterek segítségével egységesen formázhatóak az adatok a felületen vagy a kontrollerben.

A service modulok lehetővé teszik különböző funkciók megvalósítását egy helyen, amelyet később a kontrollerben elkérve, *scope*-okhoz is adhatunk. Ilyenek például az adatbázis műveletek, a fájlkezelés, a telefon kamera funkciójának kezelése stb.

Az AngularJS támogatja a különböző HTML oldalak és kontrollerek URL-lekhez való kötését, amely egy ngView direktíva segítségével valósítható meg. A HTML oldalakon belül a direktíva által megadhatóak, hogy a kívánt URL-hez kötött oldal hova töltődjön be.

A WineHunter alkalmazás megvalósítása az AngularJS elveit követi, felhasználva a fent említett modulok rendszerét és a keretrendszer által biztosított szolgáltatások előnyeit.

A PersistenceJS [17] egy aszinkron JavaScript objektumrelációs leképező (ORM – Object-Relational Mapping) könyvtár, amely használható böngészőben vagy szerver oldalon egyaránt. Segítségével létrehozhatóak WebSQL típusú adatbázisok, amelyek támogatva vannak többek között Android és iOS mobil operációs rendszereken is. Ez az ORM keretrendszer teszi lehetővé a WineHunter alkalmazáson belül kitöltött kóstolási sablonok, borokhoz rendelt képek elérési útvonalának, valamint a raktáron levő borok számának mentését, listázását, szűrését, és törlését az adatbázisból.

4. PhoneGap build-ek különböző eszközökre

A támogatni kívánt platformok hozzáadása az alkalmazáshoz, és eltávolítása a *cordova platform add* valamint *cordova platform remove* Cordova CLI parancsainak hívásával történik, amelyeknek paraméterként meg kell adni a kívánt platform nevét. A hozzáadás után, a projekt *platforms* mappájában létrejön egy alkönyvtár a hozzáadott platform nevével és benne a platform-specifikus beállításokkal. Ahhoz, hogy fel lehessen építeni és futtatni lehessen az alkalmazást, először szükséges a támogatni kívánt platformnak megfelelő SDK telepítése a megfelelő eszközre. Az iOS platformra történő telepítés csak Mac-en lehetséges, az Android támogatva van Windowson, Linuxon és Mac-en egyaránt.

4.1 Android

Az alkalmazás Androidra történő telepítéséhez és futtatásához először telepítenünk kell az Android SDK-t [18] arra az eszközre, amelyen ezt végre szeretnénk hajtani, figyelembe véve az ehhez szükséges rendszerkövetelményeket. A Cordova a 2.2, 2.3 és 4.x Android verziókat támogatja. A platform hozzáadása után beállítható, hogy melyik a legkisebb verzió, amelyet az alkalmazás támogat. A WineHunter alkalmazást a 4.x Android-dot használó készülékeken lehet használni.

Az SDK telepítése után a *tools* és *platform-tools* könyvtárakat hozzá kell adni a *PATH* környezeti változóhoz, függetlenül attól, hogy melyik operációs rendszeren dolgozunk. Ennek következtében elérhetővé válnak a Cordova Command-Line tool parancsai, amelyek segítségével felépíthető és telepíthető alkalmazás.

A projekt felépítése a *cordova build android* parancs hívásával történik, amely a *platforms/android* alkönyvtárban létrehozza az Android specifikus projektet, felhasználva a HTML, CSS3 és JavaScriptben megírt kódot. A kigenerált projekt megnyitható és szerkeszthető Eclipse-ben, NetBeans-ben, SublimeText-ben, vagy bármilyen Android projekt szerkesztésére alkalmas editorban, viszont minden buildelés során a projekt újragenerálódik, így a megtartandó módosításokat a web alapú állományokban kell elvégezni.

Az alkalmazást mobiltelefonra a *cordova run android* parancs futtatásával lehet telepíteni. Az alkalmazás telepítése történhet emulátorra a *cordova emulate android* parancs futtatásával, amelynek előfeltétele egy Android emulátor létrehozása.

4.2 iOS

Az iOS-re történő telepítés és futtatás csak OS X operációs rendszert használó Apple számítógépeken tehető meg. Az Xcode telepítése után a parancssori eszközök használatát engedélyezni kell a *Preferences* menüpont alatt, ahhoz, hogy a Cordova futtatni tudja azokat. Az iOS SDK és Xcode telepítésével biztosítva lesznek szimulátorok, amelyek segítségével tesztelhetőek lesznek az alkalmazásban használt Cordova funkcionalitások, de a szimulátorok a Macintosh-t használják fő processzorként, így előfordulhat, hogy ami tökéletesen működik az szimulátorban, az a készüléken nagyon lassan vagy egyáltalán nem fog.

A projekt felépítése a *cordova build ios* parancs futtatásával történik, amely a *platforms/ios* alkönyvtárba létrehoz egy iOS specifikus projektet. Az itt létrejött *.xcodeproj* állomány megnyitható az Xcode segítségével, ezután módosítható és telepíthető a kigenerált alkalmazás a kiválasztott szimulátorra vagy mobiltelefonra. Az Xcode nem ad lehetőség egyszerre több szimulátoron vagy telefonon való futtatásra ugyanabban az időben. A módosítások az újrabuildelés során ebben az esetben is felülíródnak.

5. A WineHunter alkalmazás

A WineHunter borkedvelők és borszakértők számára készült mobilalkalmazás, amely kényelmesebbé teszi a borkóstolások folyamatát azáltal, hogy a borok értékelése saját mobilkészülék segítségével megoldható. Ennek megvalósításához biztosít egy részletes, nemzetközileg elismert szempontok alapján összeállított borértékelő sablont, valamint felületet az értékelések utólagos karbantartására, lehetőséget az értékelések exportálására és importálására a telefon külső memóriájából, valamint további járulékos funkcionalitásokat.

A WineHunter alkalmazás egyik legfontosabb és legelőnyösebb tulajdonsága, hogy cross-platform, így a közösségeken belül használt okostelefonok operációs rendszerbeli eltérései nem jelentenek problémát.

5.1 Fontosabb követelmények és funkcionalitások

A WineHunter alkalmazás biztosít a felhasználók számára egy felületet a borértékelő sablonnal, amely lehetővé teszi a kóstolt bor általános tulajdonságainak felvezetését (például neve, kóstolás dátuma, borászat neve, alkoholtartalom, szőlőfajták stb.), képek készítését (maximum négy kép), továbbá a bor színének, illatának és ízének részletes jellemzését. Néhány tulajdonság (például szőlőfajták, illat és íz aromái stb.) esetében a sablon biztosít, egy előre elkészített listát, amelyből a jellemző értékek kiválaszthatóak, valamint bizonyos tulajdonságok esetében egy előre megadott alapértéket (kóstolás dátuma, évjárat, a bor színének tisztasága stb.), ezáltal is megkönnyítve és felgyorsítva az értékelés folyamatát.

Az alkalmazás biztosít egy felületet saját borértékelő sablonok készítésére, ahol testreszabható az eredetileg rendelkezésre álló sablon, ezáltal törölhetőek a soha ki nem töltött tulajdonságok. Az alkalmazás lehetőséget biztosít az új értékelés készítésekor a használni kívánt sablon kiválasztására.

Az elmentett értékelések listája utólag megtekinthető, lehetőség van keresésre a bor neve vagy a borászat szerint. A listából kiválasztott értékelés bármely tulajdonsága módosítható, eltávolítható az értékelésből, vagy hozzáadhatóak újabb, addig nem értékelt tulajdonságok, valamint a teljes értékelés is törölhető.

Az értékelések listája exportálható egy állományba, amely a telefon külső memóriájába kerül, amit bármikor a későbbiekben importálni lehet onnan. Importálás esetén csak azok az értékelések kerülnek vissza az alkalmazás adatbázisába, amelyek akkor nem szerepelnek a listában. Ha egy értékelés szerepel az exportált állományban és az alkalmazás adatbázisában egyaránt, de a tartalmuk nem egyezik meg, az utóbbi nem íródik felül, hanem az aktuálisan szereplő értékelés marad az adatbázisban.

Az alkalmazás lokalizálva van magyar és angol nyelvre.

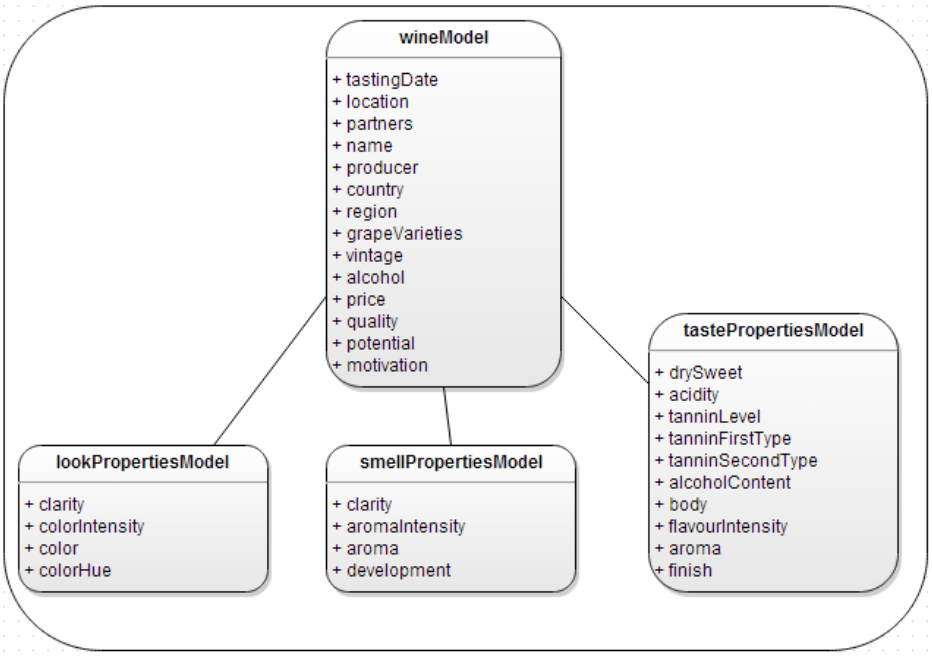
5.2 Környezeti elemzés

A *wineModel* tartalmazza a borokra és a kóstolásra vonatkozó alaptulajdonságokat, mint a kóstolás dátuma és helye, a kóstoló partnerek listája, a bor neve, a származási helye (ország és régió), a szőlőfajták listája, amelyből a bor készült, az évjárat, alkoholtartalom, a bor ára, minősége és potenciális felhasználásának meghatározása.

A *lookPropertiesModel* a bor színét jellemző tulajdonságok listáját tartalmazza, mint a tisztasága, színének intenzitása, a bor színe és árnyalata.

A *smellPropertiesModel* tartalmazza a bor illatával kapcsolatos jellemzőket, mint az illat tisztasága, intenzitása, az aromák listája, amelyek az illatában érezhetőek, valamint a fejlettsége.

A *tastePropertiesModel* reprezentálja a bor ízével kapcsolatos tulajdonságokat. Ide tartozik a bor édessége vagy szárazsága, savassága, a tartalmazott tanninok listája, az alkoholtartalom erőssége, a bor testessége, ízének intenzitása, a benne érezhető aromák listája, valamint a bor ízének lecsengése.

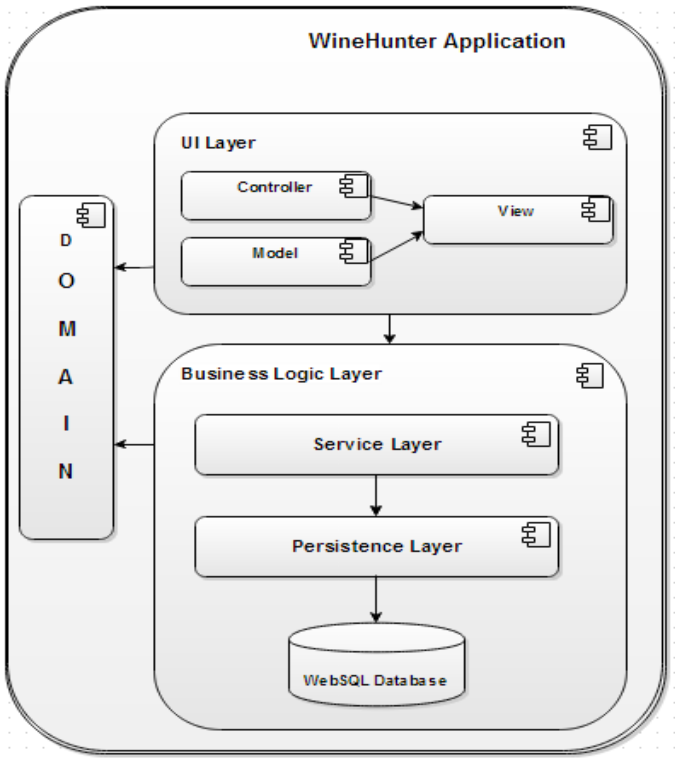


1.ábra A központi entitások szerkezete

5.3 Architektúra

Az alkalmazás adatainak mentése a PersistenceJS aszinkron JavaScript ORM keretrendszeren keresztül történik. Egyik fontos előnye, amely miatt az alkalmazás során erre

került a választás, hogy aszinkron és független más JavaScript keretrendszerektől, könnyen integrálható a projektbe, valamint kliens oldali, JavaScript-ből elérhető WebSQL adatbázist hoz létre, amely egy böngészőbe ágyazott SQLite adatbázis. Támogatva van többek között az Android és Safari böngészőkben, ezáltal lehetővé téve a WineHunter mobilalkalmazás adatainak lokális tárolását.



2.ábra: A WineHunter alkalmazás architektúrája

A *Domain* magába foglalja a mobilalkalmazás központi entitásait reprezentáló modelleket, amelyet az alkalmazás összes alrendszere használ.

A *UI Layer* magába foglalja a *Model*, *View* és *Controller* komponenseket. Ezek felelősek a felhasználói felületen történő tevékenységek értelmezéséért, események kiváltásáért, azok továbbításáért az alsóbb rétegekhez, illetve a *Service Layer* által biztosított szolgáltatások kezeléséért, és a feldolgozott adatok megjelenítéséért a felületen.

A *Business Logic Layer* tartalmazza a *Service Layer*-t illetve a *Persistence Layer*-t, amelyek feladata adathozzáférési és üzleti logikával kapcsolatos műveletek megvalósítása. A *Service Layer* meghatározza az alkalmazásban felhasznált PhoneGap API által biztosított szolgáltatásokat. A *Persistence Layer* biztosítja a kommunikációt az adatbázis és az alkalmazás között, magába foglalja a PersistenceJS keretrendszer által megvalósított adathozzáférési műveleteket.

5.4 A megvalósítás fontosabb részleteinek összefoglalása

A WineHunter fejlesztéséhez az alkalmazás vázának kigenerálása a *Yeoman* munkafolyamat *Yo* sablongenerátor alkalmazásának segítségével történt, de mivel a szerkezet módosítható, lehetővé vált a Cordova Command-Line Interface által létrehozott projektszerkezet és a kigenerált váz összevonása, ami biztosította a továbbiakban a WineHuner Cordova projektként történő kezelését, valamint az ezzel járó előnyök kihasználását.

A Cordova CLI *cordova platform add* parancsának segítségével történt az Android és iOS platformok hozzáadása az alkalmazáshoz, amely következtében az alkalmazás telepíthető és használható Andorid 4.x valamint iOS operációs rendszereket használó készülékeken.

A WineHunter alkalmazás fejlesztésében jelentős szerepe van a PhoneGap által biztosított eszköztárnak. A pluginok hozzáadása az alkalmazáshoz a *cordova plugin add* Cordova CLI parancs segítségével történt, amely biztosította a készülék natív információihoz és eszközeihez való hozzáféréshez szükséges natív implementációkat a támogatott két platform esetében. A készülék specifikus tulajdonságaihoz való hozzáférést a *Device* plugin tette lehetővé, amely elengedhetetlen a különböző platform specifikus eltérésekből adódó problémák megoldásához. A *Camera* plugin lehetővé tette a kóstolásokhoz rendelhető képek elkészítését a mobiltelefon natív kamerájával, valamint a képek hozzárendelését az aktuálisan kóstolt bor értékelő sablonjához. Ezek mellett a PhoneGap *File* pluginja teszi lehetővé az alkalmazás egyik legfontosabb funkciójának megvalósítását, az előzőleg elmentett kóstolások listájának exportálását és későbbi importálását, mivel a plugin biztosítja a készülék fájlrendszeréhez való hozzáférést, valamint annak módosítását.

Az alkalmazás megvalósítása az AngularJS model-view-controller keretrendszerének elveit követi. A projekt modulokra van osztva (service, controller, directive, filter), amelyek biztosítják a view réteget megvalósító HTML oldalak kiszolgálását. A *Service* modul magába foglalja a készülék eszközeihez való hozzáférést biztosító metódusokat. Az adathozzáférési modul biztosítja az entitásokkal végezhető adatbázis műveleteket. A Controller modulhoz tartoznak az alkalmazás vezérléséért felelős állományok. Az AngularJS legfontosabb tulajdonsága, a kétoldali adatkötés biztosítja a kontrollerek és a felület közti kommunikációt, amelynek megfelelően az alkalmazás feldolgozott adatai dinamikusan jelennek meg a felületen, valamint változás esetén mindkét oldalon változnak az értékek.

A WineHunter fejlesztése során az AngularJS saját direktívái biztosították a viselkedési formák megvalósítását a felületen (ngModel, ngController, ngClick, ngShow stb.),

valamint saját direktívákon keresztül valósult meg a saját komponensek készítése (whTastingList, whAutocompleteList stb.), amelyek akár több helyen is használhatóak. Ezek a direktívák módosíthatják a DOM szerkezetét és magukba foglalhatják a komponensek működésének logikáját is.

Az AngularJS által biztosított filterek lehetővé tették az alkalmazáson belül megjelenített adatok egységes formázását (uppercase, date stb.). A filterek használata a HTML fájlokban történik a filter nevének megadásával.

A keretrendszer *Resource Localization Service* különálló moduljának integrálásával valósult meg az alkalmazás lokalizálása magyar és angol nyelvre, amely meghatározott nevű és szerkezetű (kulcs-érték párokat tartalmazó) *resource* állományokat felhasználva végzi el a lokalizálást.

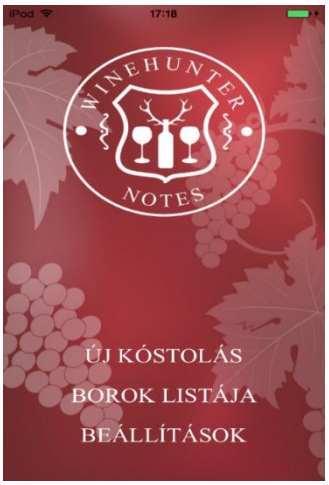
A WineHunter alkalmazás megvalósításában nagy szerepe van a jQuery JavaScript keretrendszernek, amely lehetővé tette a DOM elemeinek kiválasztását az elemekhez rendelt ID-k vagy osztályok (*class*) által, és azok szerkezetének dinamikus módosítását különböző, a kontrollerekben figyelt események hatására.

A felhasználóbarát felület a jQueryUI által biztosított widgetek felhasználásával történt, amely lehetővé teszi többek között a kóstolás dátumának datepicker-ből történő kiválasztását, a tulajdonságok erősségének meghatározását slider segítségével, valamint néhány tulajdonság kitöltése esetén automatikus kitöltési funkciót biztosít stb.

A PersistenceJS JavaScript ORM keretrendszer teszi lehetővé a WineHunter alkalmazáson belül kitöltött kóstolási sablonok, borokhoz rendelt képek elérési útvonalának, valamint a raktáron levő borok számának mentését, listázását, szűrését, és törlését az adatbázisból.

6. A WineHunter működése

A WineHunter telepítése és elindítása után az alkalmazás egy főmenüvel indul (3.ábra), amely lehetőséget ad a három fő funkcionalitás kiválasztására: az új kóstolás felvételére, az előzőleg kóstolt borok értékelési listájának megtekintésére és menedzselésére, valamint a beállítások megadására.



3.ábra: A WineHunter főmenüje

Az *Új kóstolás* menüpont kiválasztásával az alkalmazás által biztosított borértékelő sablonhoz navigálhatunk, amelynek első oldalán lehetőség van a bor általános tulajdonságainak felvezetésére (név, borászat, szőlőfajták, évjárat stb.), amelyek leolvashatóak a kóstolt bor üvegének címkéjéről. Az oldal tetején megjelenő kamera ikonra kattintva készíthető maximum négy kép a kóstolt borról a készülék beépített kamerájának segítségével, amelyek később teljes képernyő módban is megtekinthetőek a készített képre kattintva. A dátum mező alapból az aktuális dátummal van kitöltve.

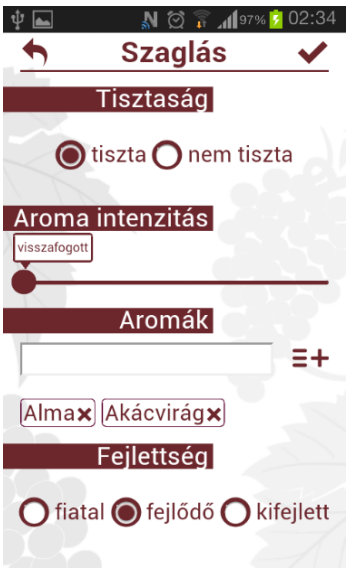
4.ábra Általános tulajdonságok

A partnerek és szőlőfajták esetében, a szövegmezők kitöltése után a mezők melletti ikonra kattintva a mező tartalma egy listához adódik hozzá. Lehetőség van az értékek törlésére is a listákból. A mezők kitöltése során biztosítva van egy, az előzőleg mentett értékekből összeállított lista, amely gépeléskor egyezés esetén megjelenik. A javaslatlistából kiválasztott elem automatikusan hozzáadódik a listához, ezáltal könnyítve és gyorsítva a kitöltés folyamatát.

A második oldalon (5. ábra) a bor megjelenésével kapcsolatos tulajdonságok felvezetése lehetséges (színének tisztasága, erőssége, színárnyalat). Alapértelmezetten ki vannak választva az általánosan jellemző értékek, amelyek radio button-ok segítségével változtathatóak.

5.ábra: A megjelenéssel kapcsolatos tulajdonságok

A harmadik oldalon (6. ábra) az illattal kapcsolatos tulajdonságok vannak feltüntetve. Szintén biztosítottak alapértelmezett értékek és a módosítás az előző oldalhoz hasonlóan történik. Az aromák meghatározása az első oldalon levő partnerek és szőlőfajták felvezetésének mintáját követi, biztosítva egy autocomplete funkcionalitást a kitöltéshez, amely ugyancsak a már korábban mentett értékeket veszi alapul.



6.ábra A bor illatával kapcsolatos tulajdonságok

A negyedik oldal (7. ábra) az ízleléssel kapcsolatos tulajdonságokat foglalja magába, ahol az előző oldalaktól eltérően, néhány tulajdonság erősségének meghatározása sliderek segítségével történik (savasság, tanninok, alkoholtartalom stb.).

Az ötödik és egyben az értékelés utolsó oldala (8. ábra) a kóstolás utáni következtetések felvezetésére ad lehetőséget.

7.ábra Ízleléssel kapcsolatos tulajdonságok

8.ábra A sablon befejező oldala

Az értékelő sablon kitöltése során minden oldalon megjelenik a jobb felső sarokban egy mentés ikon, amelynek érintésével az addig kitöltött adatok mentődnek az alkalmazás adatbázisába. Az értékelés készítése során csak a bor nevének kitöltése kötelező, amely ha nem történt meg, mentés esetén egy figyelmeztető üzenetet követően az alkalmazás az értékelés első oldalához navigál, ahol ez megtehető.

Minden oldalon elvégezhető a mentés, de annak következtében, hogy visszamenőleg minden adat mentésre kerül, elegendő az utolsó oldalon megtenni azt a teljes értékelés mentéséhez. A sikeres mentés után az alkalmazás a főmenüre navigál vissza.

A bal felső sarokban megjelenő *vissza* ikon ad lehetőséget az értékelési sablonból a főmenüre való visszatérésre, miután a megjelenő párbeszédablakban kérjük a sablon mentését vagy elvetését.

A főmenü második menüpontja (*Borok listája*) az előzőleg elmentett kóstolások listájához navigál (9. ábra). A lista minden elemének esetében megjelenik a kóstolt bor neve, valamint a kóstolás dátuma, a borászat neve, amelytől származik a kóstolt bor, valamint annak évjárata (ha ezek a mezők az értékelés során ki lettek töltve).

9. ábra Kóstolások listája

Minden elem mellett jobb felől megjelenik egy boros üveget ábrázoló ikon és rajta egy szám, amely jelzi, hogy az adott borból hány palack van a felhasználó saját borkészletében, valamint rákattintva az érték módosítható. A palack mellett megjelenő ikon kiválasztásával törölhető az adott kóstolás az alkalmazás adatbázisából.

Az oldal jobb felső sarkában megjelenő állománykezelő ikon kiválasztásával megjelennek az adatok exportálására és importálására alkalmas funkciók. A bal oldali ikon kiválasztásával a mentett kóstolások listája egy *exportedWHData.json* nevű állományban lesz exportálva a készülék saját fájlrendszerébe, ahol publikusan elérhetővé válik. A jobb oldali ikon kiválasztásával az előzőleg exportált kóstolások listája importálható az alkalmazásba, ahová csak azok a listaelemek kerülnek vissza, amelyek aktuálisan nem szerepelnek az alkalmazás adatbázisában.

A listában lehetőség van keresésre a kóstolás során megadott bármilyen tulajdonság alapján, a képernyő alján levő *search* ikon érintése után megjelenő szövegmező segítségével.

A lista egy elemét kiválasztva, az alkalmazás a borértékelő sablonhoz navigál, amelynek szerkezete teljesen megegyezik a főmenü *Új kóstolás* menüpontjával elérhető kóstolási sablonnal, azzal a különbséggel, hogy a kiválasztott kóstolás során megadott tulajdonságok értékeivel lesz kitöltve a sablon. Ezáltal a teljes értékelés megtekinthető, valamint módosítható. Mentés után a kóstolások listájára navigál vissza az alkalmazás.

A főmenü *Beállítások* menüpontja, tartalmazza az alkalmazás használatának testreszabását elősegítő beállításokat (10. ábra). Itt van lehetőség az alkalmazás nyelvének

beállítására (angol, magyar), kiválasztható a használni kívánt borlap típusa (alap sablon, saját), valamint saját borértékelő sablon készíthető az *Új sablon* menüpont kiválasztásával.

10. ábra Beállítások nézet

Az Új sablon menüpont egy új oldalra navigál, ahol megjelenik az alkalmazás által biztosított borértékelő sablon. Minden tulajdonság mellett megjelenik egy törlés ikon, amely lehetővé teszi az adott tulajdonság eltávolítását a saját értékelési sablonból. Ezáltal létrehozható egy olyan borlap, amely csak a felhasználó által kitölteni kívánt tulajdonságokat tartalmazza. Az oldal jobb felső sarkában megjelenő *mentés* ikon segítségével menthető a készített borértékelő lap, megadva annak egy nevet. A bal felső sarokban levő *vissza* gomb kiválasztásával elvethetőek a módosítások.

Következtetések és továbbfejlesztési lehetőségek

Az alkalmazás megvalósításával lehetőség nyílik a borkóstolások során készítendő értékelések készítésére és utólagos menedzselésére, kényelmesebbé téve ezáltal a bórkóstolás folyamatát. Az alkalmazás egy könnyen használható, intuitív felületet biztosít, amellyel elősegíti a gyors és egyszerű értékelést, valamint az értékelés folyamatának testreszabását.

A felhasznált eszközök, technológiák és módszerek által lehetővé vált az alkalmazás különböző platformokon történő egységes használata. Mivel egy alkalmazás elkészítése külön Android és iOS rendszerekre, valamint utólagos karbantartása időigényesebb, a cross-platform megoldás egy életképes alternatívának bizonyult.

A WineHunter alkalmazás jelen állapotában egy bemutató verzió, amelynek működését és használatát számos funkcionalitással lehet javítani és bővíteni:

- Jelenleg csak Android és iOS operációs rendszerekre építhető fel az alkalmazás. A későbbiekben a build rendszert konfigurálni lehetne más PhoneGap által támogatott platformoknak megfelelően is.
- A kóstolás aktuális helyének meghatározása a GPS koordináták segítségével, valamint ezekhez saját nevek hozzárendelése, a helyszín adatainak későbbi felhasználása.
- A kóstolt borok pontozásának beépítése az aktuálisan biztosított sablonba (pontszám szerint, vagy csillagok segítségével), valamint a különböző pontozási rendszerek közötti megfeleltetés megvalósítása.
- Az alkalmazás lokalizálása több nyelvre.
- Egy szerver fejlesztése, amely lehetőséget biztosítana online borkatalógusokhoz való hozzáféréshez. Ezáltal a kóstolt borok értékeléskor kiválaszthatóak lennének a katalógusból, kitöltve ezzel néhány általános tulajdonságot a borlapon. A szerver oldal lehetőséget biztosíthatna a saját értékelések megosztására a megfelelő oldalakon, borkóstolási események létrehozására, oktatási anyagok publikálására stb.

Hivatkozások

[1] Bormédia on-line interaktív bormagazin,
http://www.bormedia.hu/oldal.php?menu=111&ceg_id=111&lang=hun,
utolsó megtekintés dátuma: 2014.04.30
[2] Mercurial Hivatalos Weboldal, http://mercurial.selenic.com/ utolsó megtekintés dátuma: 2014.04.30
[3] Redmine Hivatalos Weboldal, http://www.redmine.org/
utolsó megtekintés dátuma: 2014.03.29
[4] NodeJS Hivatalos Weboldal, http://nodejs.org/
utolsó megtekintés dátuma: 2014.04.10
[5] Weblabor online szakmai lap, http://weblabor.hu/cikkek/nodejs-alapok
utolsó megtekintés dátuma: 2014.04.25
[6] Yeoman Hivatalos Weboldal , http://yeoman.io/
utolsó megtekintés dátuma: 2014.04.15
[7] AngularJS Hivatalos Weboldal, http://angularjs.org/
utolsó megtekintés dátuma: 2014.04.28
[8] RevealJS Hivatalos Weboldal, http://lab.hakim.se/reveal-js/#/
utolsó megtekintés dátuma: 2014.04.02
[9] ExpressJS Hivatalos Weboldal, http://expressjs.com/
utolsó megtekintés dátuma: 2014.04.02
[10] Grunt Hivatalos Weboldal, http://gruntjs.com/
utolsó megtekintés dátuma: 2014.04.12
[11] Bower Hivatalos Weboldal, http://bower.io/
utolsó megtekintés dátuma: 2014.04.12
[12] JSHint Hivatalos Weboldal, http://www.jshint.com/
utolsó megtekintés dátuma: 2014.04.13
[13] Sublime Text Hivatalos Weboldal, http://www.sublimetext.com/
utolsó megtekintés dátuma: 2014.04.13
[14] PhoheGap Hivatalos Weboldal, http://phonegap.com/
utolsó megtekintés dátuma: 2014.04.27
[15] jQuery Hivatalos Weboldal, http://jquery.com/
utolsó megtekintés dátuma: 2014.04.26
[16] jQueryUI Hivatalos Weboldal, https://jqueryui.com/
utolsó megtekintés dátuma: 2014.04.09
[17] PersistenceJS Hivatalos Weboldal, https://github.com/zefhemel/persistencejs
utolsó megtekintés dátuma: 2014.04.11
[18] Android Developer Hivatalos Weboldal,
http://developer.android.com/sdk/index.html
utolsó megtekintés dátuma: 2014.04.12