XVII. reál- és humántudományi Erdélyi Tudományos Diákköri Konferencia (ETDK)
Kolozsvár, 2014. május 16–18.

Online kalandjáték okostelefonokra és webes játékszerkesztő

Szerzők:
Brassai Beáta
Babeş-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, Informatika szak, III. év
Varga Boglárka
Babeş-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, Informatika szak, III. év

Témavezetők:
dr. Simon Károly, egyetemi adjunktus, Babeş-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, Magyar Matematika és Informatika Intézet
Török-Vistai Tamás, projektmenedzser, Codespring Kft.

Kivonat

A dolgozat az ARTHAS GeoQuesting projektet, a hozzátartozó webes játékszerkesztő felületet és mobilalkalmazást, valamint ezek megvalósítási és működési hátterét mutatja be.

A projekt központi eleme egy mobilkészülékkel játszható kalandjáték, amelynek keretein belül a felhasználónak feladványokat kell megoldania és ellenőrző pontokat kell megkeresnie a környezetében, így haladva lépésről lépésre a cél felé. A játékot játszva bárki könnyedén, játékos formában, szórakoztató módon ismerheti meg egy adott régió nevezetességeit. Az alkalmazás segítséget nyújthat az oktatásban, felhasználható turisztikában, reklám- és szórakoztatóiparban.

A projekt a felhasználók rendelkezésére bocsájt egy webes játékszerkesztő felületet, ahol a játékok könnyen létrehozhatóak és menedzselhetőek, változatos feltételeken alapuló küldetések állíthatóak össze (szöveges feladványok megoldása, helyszínek megkeresése, a kiterjesztett valóság részeként megjelenő virtuális tárgyak megtalálása stb.).

Tartalomjegyzék

Bevezető

A dolgozat célja bemutatni az ARTHAS GeoQuesting projektet, amely magába foglal egy GeoQuesting szervert, egy telefonos alkalmazást, és egy webes játékszerkesztő felületet.

A GeoQuesting egy mobilkészülékkel játszható kincskereső játék. A projekt célja egy olyan játék létrehozása volt, amely egyszerre szórakoztató és tanulságos, azáltal, hogy megfelelő játékélményt nyújt a játékosoknak, miközben információkat közöl környezetükről. A játékosnak küldetéseket kell teljesítenie, ezen belül feladványokat kell megoldania, ellenőrző pontokat kell megkeresnie, így haladva lépésről lépésre a cél felé. A játékok mind témájukban, mind összeállításukban változatosak lehetnek, mivel a szoftver lehetőséget ad a játékok testreszabására. Attól függően, hogy hogyan vannak megfogalmazva a feladványok, illetve összeállítva az ellenőrzőpontok, egy játék alkalmazható például turisztikai ismertetőként, használható az oktatásban, vagy lehet egyszerűen szórakoztató.

Egy felhasználási eset lehet például egy történelmi jellegű játék létrehozása, amely során a játékos végigvezethető egy város nevezetességein, elkalauzolható a fontosabb épületekhez, és az egyes helyszíneken híres történelmi személyekről tudhat meg információkat. Az alkalmazás lehetőséget ad arra, hogy a játékost pontosan elvezesse egy adott helyszínre, ehhez biztosítva egy útvonaltervezőt is. Egy helyszínen kérheti a játékost, hogy forduljon egy adott irányba, illetve kérdéseket tehet fel, amelyekre a felhasználónak válaszolnia kell a küldetés teljesítése érdekében. Bizonyos ellenőrző pontokon elhelyezhetőek NFC (Near Field Communication) kártyák, amelyek további utasításokat vagy információkat szolgáltathatnak egy adott helyszínről. Továbbá, a játék lehetőséget biztosít a kiterjesztett valóság részeként megjelenő virtuális tárgyak elhelyezésére is.

A projekt biztosít egy webes felületet játékok szerkesztésére és menedzselésére. A felhasználók létrehozhatnak új játékokat, tetszés szerint határozhatják meg a küldetéseket, feladványokat, ellenőrzőpontokat, illetve összeállíthatják ezek sorrendjét. A szerkesztés befejezése után a játékok elérhetővé tehetőek a felhasználók számára, és ettől kezdve bárki által játszhatóak lesznek.

A GeoQuesting projekt fejlesztői csapata a Codespring Kft.-nél töltött szakmai gyakorlat alatt sajátította el a fejlesztés során felhasznált eszközök és technológiák alapjait, majd az ezt követő tanév során a cég infrastruktúráját felhasználva folytatta az alkalmazás fejlesztését. A projekt létrejöttéért köszönet illeti Török-Vistai Tamás projektmenedzsert a Codespring Kft. részéről, valamint Dr. Simon Károly irányítótanárt, akik a fejlesztési folyamatot irányították.

1 Hasonló projektek és alkalmazások

Több olyan játék létezik, amelyik bizonyos mértékben hasonlít a dolgozatban tárgyalt GeoQuesting mobilalkalmazáshoz, de vannak alapvető különbségek.

A földrajzi koordináták alapján helyszínek megtalálását célzó játékok közül közismert és világszinten elterjedt a főként sport és turisztikai tevékenységeket népszerűsítő Geocaching kincskereső játék, amelynek célja elrejtett tárgyak megtalálása GPS készülékek segítségével. Az ilyen típusú játékok esetében gondot jelenthet a tárgyak kihelyezése, folyamatos ellenőrzése és „karbantartása". Az alapötletet továbbgondolva, születtek olyan kezdeményezések, amelyek mobilkészülékeken futtatható alkalmazásokon alapulnak. Ezeknek esetében a „kincsek" fizikai tárgyak helyett lehetnek például a kiterjesztett valóság részeként megjelenő virtuális objektumok. Vannak játékok, amelyek során a felhasználónak nem egyszerűen helyszíneket és tárgyakat kell megtalálnia, hanem különböző küldetéseket is teljesítenie kell. Ebben a kategóriában megemlíthető a Google és a Niantic Labs által fejlesztett és egyre népszerűbbé váló Ingress nevű játék, amely egy kiterjesztett valóságon alapuló telefonos alkalmazás. Egy összetett és részletesen kidolgozott háttértörténeten alapszik, a játékos karakterének fejlődésére és a csoportos küldetések élményére fektet hangsúlyt. Bár bizonyos szempontból oktatási és turisztikai célt is szolgál, mivel a küldetések helyszínei általában emlékművek, nevezetes helyek környékén találhatóak, elsődleges célja mégis inkább a szórakoztatás. A GeoQuesting esetében nagyobb hangsúlyt kap az ismeretek fejlesztése. Az Ingress-hez viszonyítva a GeoQuesting mind a háttértörténet, mind a küldetéstípusok szempontjából nagyobb szabadságot ad, és kevésbé összetett, de változatosabb játékok készíthetőek a segítségével. A játék élményét nem feltétlenül egy virtuális univerzum, hanem sokkal inkább a játékos valós környezetének megismerése adja.

Tulajdonképpen a bemutatott GeoQuesting alkalmazás elődjének tekinthető a Kolozsváron fejlesztett Augmented Reality-based Treasure Hunt Applications for Smartphones (ARTHAS) keretrendszer (innen ered a projekt kódneve is), amelyhez iOS és Android operációs rendszereken futó kliensalkalmazás prototípusok is készültek [1]. Ezeknek a tesztelése során a rendszer számos hátrányára és hibájára fény derült, így, bár alapelképzelés szempontjából hasonló rendszerről van szó, sem kód szintjén, sem keretrendszerként nem lehetett felhasználni a GeoQuesting fejlesztésénél.

Az említett rendszernek központi részét képezte a képfelismerés és a kiterjesztett valóság, csak képfelismerésen alapuló helymeghatározással kapcsolatos feladattípusok voltak támogatottak. Ebből a szempontból a GeoQuesting sokkal általánosabb, több különböző feladattípust támogat (GPS alapú helymeghatározáson és a mobilkészülék szenzorjaitól kapott

információkon alapuló feladatok, feladványokon alapuló küldetések, NFC alapú helymeghatározáson alapuló küldetések stb.), és könnyen kiterjeszthető új feladattípusokkal (akár képfelismerésen alapuló helymeghatározás alapján ellenőrizhető küldetésekkel is). Ugyanakkor a feladattípusok kombinálhatóak is. Például, lehetséges, hogy egy adott küldetés teljesítéséhez nemcsak arra van szükség, hogy a felhasználó eljusson a megfelelő helyszínre, hanem ott bizonyos irányba fordulva, bizonyos szögben kell tartania a telefonját, ahhoz, hogy így képeket láthasson annak kameráján keresztül és ezekhez kapcsolódó kérdéseket válaszolhasson meg.

A régebbi Arthas projekt csak keretrendszerként szolgált, nem biztosított szerkesztőfelületet, lehetőséget a játékok egyszerű, programozási ismereteket nem igénylő szerkesztésére és menedzselésére. A GeoQuesting webes játékszerkesztő felületet is biztosít, így bárki könnyen létrehozhat új játékokat, egy könnyen kezelhető felületen menedzselheti ezeket és programozási ismeretek hiányában is megteheti mindezt.

A két rendszer között technológiai szempontból is jelentős eltérések vannak. Míg a GeoQuesting szerver Java Enterprise Edition platformra épül, a régi Arthas keretrendszer esetében nem voltak alkalmazva ilyen vállalati fejlesztéseket támogató technológiák. A mobilkliensek és a szerver oldal közti kommunikáció az említett rendszeren belül az Apache Mina keretrendszerrel volt megvalósítva, így folyamatos kapcsolatot igényelt a kliens és a szerver között. Tekintettel arra, hogy a játék jellegéből adódóan egyáltalán nem biztos, hogy minden helyszínen biztosított a megfelelő internetkapcsolat (például két ellenőrzőpont között), ez a megközelítés sem számít megfelelő megoldásnak. A kapcsolat folyamatos fenntartására nincs is szükség, csak az fontos, hogy a feladványok és válaszok küldésének pillanatában a kliens kapcsolatot tudjon létesíteni a szerverrel. A GeoQuesting rendszer ennek megfelelően, RESTful webszolgáltatásokon keresztül valósítja meg a kommunikációt.

A régebbi prototípusok tesztelése során a rendszer által használt Vuforia keretrendszeren alapuló képfelismerő és AR modul nem bizonyult megfelelőnek. A GeoQuesting esetében kipróbált OpenCV keretrendszer a Vuforia felhasznált verziójával ellentétben lehetőséget ad a felismerést végző modul érzékenységének beállítására. Ez azért indokolt, mert a játéknak folyamatosan igazodnia kell olyan környezeti tényezőkhöz, mint például napszakok, mozgó emberek, árnyékok stb. A régebbi rendszer az említett beállítások hiánya miatt legtöbbször túlérzékenynek bizonyult, nem ismerte fel megfelelően a helyszínek beazonosításához használt képeket. A GeoQuesting tesztelése alatt az OpenCV keretrendszer rugalmassága ellenére szintén nem tűnt megfelelőnek. A beállítások ellenére a képfelismerés nem történt meg elég hatékonyan, így nem érte meg függővé tenni az alkalmazást a keretrendszertől. A jelenlegi verzióban a képfelismerésen alapuló helymeghatározás, mint

funkcionalitás, nem szerepel. Mivel a GeoQuesting küldetéstípusok szempontjából jóval általánosabb a régebbi Arthas rendszernél, ez nem is számított annyira központi funkcionalitásnak. Ettől függetlenül, folyamatban van további képfelismerésen alapuló AR keretrendszerek kipróbálása, és amennyiben sikerül megfelelő technológiát találni, az integrálva lesz a rendszerbe.

Összefoglalva, bár komoly kezdeményezések léteznek a területen belül és több projekt esetében hasonlóak az alapelképzelések, a bemutatott GeoQuesting alkalmazásnak vannak olyan tulajdonságai, amelyek egyedivé teszik a kategóriában.

2 Fejlesztési módszerek és eszközök

Az Arthas GeoQuesting fejlesztése során az agilis szoftverfejlesztési szemléletmód érvényesült, a csapat Scrum módszereket használt [2]. A körülbelül egy évig tartó fejlesztés kéthetes ciklusokban, úgynevezett sprintekben valósult meg, a Scrum módszer által javasolt rendszeres találkozókkal. A felhasználói történetek prioritási sorrendbe voltak rendezve, az alkalmazás folyamatosan egészült ki új funkcionalitásokkal, és minden sprint végén működő funkcionalitások voltak bemutatva a terméktulajdonos szerepét is betöltő szakmai irányítóknak.

A forráskód menedzsmentjére és változásainak követésére a csapat a Mercurial osztott verziókövető rendszert [3] használta a fejlesztés során. A központi tároló menedzsmentje a RhodeCode web-alkalmazás segítségével történt, kliensalkalmazásként TortoiseHg-t és a fejlesztői környezetbe beépülő modulokat használtak a fejlesztők.

Összetettebb szoftverek esetében a rendszer felépítése és a függőségek kezelése nem manuálisan történik, hanem különböző automatizált build eszközök (pl. Ant, Maven, Gradle) használatával. A GeoQuesting esetében az Apache Maven build és függőségmenedzsment eszköz [4] biztosította ezt a szolgáltatást. A Maven esetében a Project Object Model (POM) határozza meg a projekt szerkezetét, függőségeit és a felépítéséhez szükséges modulokat. A modellt leíró XML állományok alapján az eszköz megoldja a függőségeket (amennyiben szükséges, távoli tárolókból letölti a megfelelő csomagokat), felépíti a projektet (a fordításon és csomagoláson kívül beleértve az automatizált tesztek futtatását, a megfelelő modulok kitelepítését az alkalmazásszerverre, különböző további műveletek – pl. kódformázás – elvégzését stb.). Intézményeken belül hasznos lehet saját belső tároló és ennek menedzsmentjét biztosító rendszer használata. Az ide feltöltött modulok szervezeten belül más fejlesztők által is használhatóak, függőségként hozzákapcsolhatóak a projektekhez. Ezen kívül a belső rendszer a távoli publikus tárolók tükrözésére is használható. Erre a célra a GeoQuesting esetében az Artifactory repository management rendszer volt használva.

A fejlesztés során a szoftvert lehetőleg mindig helyes, fordítható és működő állapotban kell tartani. Ilyen szempontból jó stratégia a folytonos integráció (Continuous Integration - CI) elvének követése. Jó módszer, ha a fejlesztők gyakran (pl. naponta) megosztják egymás között változtatásaikat és minden változtatás után (a központi tárolóba történő commit/push műveletek), vagy legalább adott időközönként automatikusan megtörténik a projekt felépítése (build). Ezt a célt szolgálják a különböző CI eszközök, amelyek build szervereket biztosítanak, és ezek összekapcsolhatóak tesztszerverekkel (a build folyamat után a szoftver automatikusan kitelepíthető ezekre) és további (például

minőségbiztosítási) rendszerekkel. A fejlesztőknek így mindig van egy rálátásuk a projekt aktuális állapotára. A GeoQuesting fejlesztésénél a Jenkins támogatta a CI módszer alkalmazását.

A szoftvertermékek minőségének biztosítását szolgálják a különböző automatizált kódelemző eszközök, mint például a Yasca, a Squale, a CodePro Analytix és a SonarQube. A dolgozatban tárgyalt projekt esetében a csapat a SonarQube eszközt használta [5]. A szoftver előre meghatározott szabályok alapján végzi a kód elemzését, ellenőrzi a duplikátumokat, kommenteket, a kód komplexitását és a tesztlefedettséget is. Az elemzések eredményeit meg lehet tekinteni egy webes felületen, ahol a fejlesztők részletes jelentések által kapnak képet a kód minőségéről. Együtt használható CI eszközökkel, így a GeoQuesting esetében a Jenkins-el együtt biztosította, hogy mindig aktuális adatokkal szolgáljon a kódról. JavaEE alkalmazásszerverként a GlassFish szolgált. A webszolgáltatásokra vonatkozó API dokumentáció publikálása Swagger UI segítségével történt. A fejlesztés első szakaszában a TestFlight platform tette kényelmesebbé és hatékonyabbá a mobilalkalmazás tesztelését. A csapaton belüli kommunikációt, a dokumentumok menedzsmentjét és megosztását, valamint közös szerkesztését az XWiki rendszer biztosította. Fejlesztői környezetként NetBeans-t és Android Studio-t használtak a fejlesztők, kiegészítve további (pl. az adatbázis menedzsmentjét, tervezést, felhasználói felület vázlatok elkészítését) támogató eszközökkel.

3 Felhasznált technológiák

A GeoQuesting rendszer fejlesztésénél a csapat nagy hangsúlyt fektetett arra, hogy mind szerver-, mind kliensoldalon megfelelő, a szoftverfejlesztési iparban aktuálisnak számító technológiákat alkalmazzon. Architektúrájának és a felhasznált vállalati fejlesztéseket támogató keretrendszereknek köszönhetően a szoftver biztonságosan és hatékonyan használható sok felhasználó esetében is, könnyen skálázható, konfigurálható, bővíthető és karbantartható.

3.1 A Java Enterprise Edition platform

A Java Enterprise Edition (JavaEE)[6] komponens alapú alkalmazások tervezését, fejlesztését, összeállítását és telepítését elősegítő platform, amely támogatja a többrétegű, osztott vállalati alkalmazások készítését. Tulajdonképpen egy szabványcsalád (pontosabban minden verziójának megfelel egy-egy JSR[1]), amelyet a különböző Java alkalmazásszerverek implementálnak (referencia implementációja az Oracle által fejlesztett GlassFish).

A Java Standard Edition (JavaSE) platformhoz képest több API-t és keretrendszert biztosít (amelyeket a JavaEE szabványcsalád részét képező további szabványok specifikálnak és az alkalmazásszerverek ezekhez különböző implementációkat biztosítanak). Megemlíthető például a szerver oldali komponensek létrehozását és menedzsmentjét biztosító EJB (Enterprise JavaBeans), a dependency injection tervezési mintát támogató CDI (Context and Dependency Injection), az adathozzáférési réteg megvalósítását egyszerűsítő és egységesítő Java Persistence API (JPA), az üzenetorientált rendszereket támogató Java Message Service (JMS), a tranzakció-menedzsment megvalósítására szolgáló Java Transaction API (JTA), a biztonságot garantáló Java Authentication and Authorization Service (JAAS), a webes fejlesztéseknél alkalmazható Servlet API, JavaServer Pages (JSP) technológia és JavaServer Faces (JSF) keretrendszer, a webszolgáltatások implementálását elősegítő JAX-WS és JAX-RS stb.

A JavaEE szabványt implementáló alkalmazásszerverek az említett keretrendszereken keresztül biztosítják a skálázhatóságot, az erőforrások hatékony menedzsmentjét, a perzisztencia réteget, a biztonságot, a tranzakciók menedzsmentjét a központosított konfigurálás lehetőségét, lehetővé téve így a fejlesztőknek, hogy munkájuk során az alkalmazás logikájára koncentrálhassanak. A felsoroltakon kívül a szerverek általában további szolgáltatásokat is biztosítanak (például clustering, load balancing és failover támogatást

[1] Java Specification Request (lásd: https://jcp.org/en/jsr/all)

nyújthatnak stb.). A Java esetében több választási lehetőség van alkalmazásszerverek szempontjából, rendelkezésre áll például a GlassFish (Oracle, GlassFish Community), JBoss (JBoss/RedHat), WebLogic (Oracle), Apache Geronimo, IBM WebSphere stb.

A GeoQuesting esetében a GlassFish alkalmazásszervert használták a fejlesztők. Az üzleti logikát megvalósító szerver oldali komponensek EJB-k, az ezek közötti függőségek a dependency injection minta alapján, CDI alkalmazásával menedzseltek, az adathozzáférési réteg JPA alapú. A web alkalmazás használja továbbá a Servlet API-t, a mobilalkalmazás webszolgáltatásokon keresztül kommunikál a szerverrel, JAX-RS alkalmazásával. Az aktuális fejlesztések és továbbfejlesztési tervek további Java EE technológiák integrációját is igényelik (pl. JAAS alapú biztonsági mechanizmus stb.).

3.2 Enterprise JavaBeans

Az Enterprise JavaBeans (EJB) [7][8] tulajdonképpen a Java EE alkalmazások szerver oldali komponensmodellje, az alkalmazások üzleti logikáját EJB komponensek valósítják meg. Ezek lehetnek session bean-ek vagy message-driven bean-ek.

A session bean-ek lehetnek állapot nélküliek (stateless) vagy állapottal rendelkezőek (stateful). A stateless bean szolgáltatások (metódusok) összessége, az állapotát nem őrzi meg két metódushívás között. A stateful bean egy egyedi kliens munkamenetén belül használható, több metódushívás között megőrizhet állapotinformációkat. A session bean-ekkel interfészeiken keresztül kommunikálhatunk. Megvalósíthatnak lokális (az adott konténeren belül menedzselt komponensek által hívható műveletek), remote (távoli komponensek által, RMI-n keresztül hívható metódusok) és endpoint (webszolgáltatásokon keresztül publikált műveletek) interfészeket, illetve no-interface nézetet is biztosíthatnak (a komponens minden metódusa látható lesz lokálisan).

A message-driven bean-ek (MDB) aszinkron üzenetalapú kommunikációt tesznek lehetővé, fogadhatnak például JMS üzeneteket, amelyekre reagálva végrehajthatnak üzleti logikával kapcsolatos műveleteket.

A komponenseket az alkalmazásszerver az EJB konténeren belül menedzseli, hatékonyságnövelő módszereket alkalmazva (instance pooling az állapot nélküli session bean-ek és MDB-k esetében, eviction mechanizmus a stateful session bean-ek esetében).

Az EJB 3.0-ás specifikációja előtt létezett egy harmadik kategória is, a rendszer központi entitásait reprezentáló entity bean-ek kategóriája. Ezeket váltották le a JPA entitások, amelyek már nem tekinthetőek a szó szoros értelmében EJB komponenseknek (ezeket közvetlen módon, nem interfészeiken keresztül használjuk). Tulajdonképpen meta adatokkal felannotált POJO-k (Plain Old Java Object). A JPA az EJB-től különállóan, például

JavaSE alkalmazásokban is alkalmazható és a JPA 2.1-től kezdve külön szabvány által specifikált, tehát már nem része az EJB 3.1-es szabványnak.

A GeoQuesting szerver esetében állapot nélküli session bean-ek segítségével van megvalósítva mind az adathozzáférési, mind a szolgáltatási réteg.

A komponensek az EJB konténer által menedzseltek, a komponensek közötti függőségek a Dependency Injection (DI) tervezési minta alapján vannak megoldva, amely az Inversion of Control (IoC) egy formája.

A klasszikus programozási modellben a vezérlés folyamatát a programozó határozza meg, ellentétben az IoC modellt alkalmazó keretrendszerekkel, amikor a vezérlésért maga a keretrendszer felelős. A DI, azaz a függőségek befecskendezése, az IoC minta alkalmazása, amely lehetővé teszi, hogy az alkalmazást különálló komponensek összességének tekintsük, melyek összekapcsolását a konténer végzi. A GeoQuesting projekt esetében a minta az adathozzáférési, szolgáltatási, valamint a webes rétegekben van alkalmazva. A projekt használja mind az EJB specifikáció szerinti DI mechanizmust, mind a Context and Dependency Injection (CDI) [9] specifikáció szerinti módszert. Ahhoz, hogy a mechanizmus a Vaadin keretrendszer segítségével felépített webes komponensek esetében is működhessen a Vaadin CDI kiegészítése volt felhasználva.

3.3 JPA

A Java Persistence API (JPA) [10] meghatároz egy egységesített eljárást a rendszer központi entitásainak, perzisztens adatainak kezelésére. Relációs adatbázisban történő tárolás szempontjából egy JDBC feletti absztrakciós szintet képez. A szabványnak több implementációja ismert, például a Hibernate, az OpenJPA, a Toplink stb. A GeoQuesting a szabvány referencia implementációját, az EclipseLink-et használja.

A JPA meghatároz egy teljes objektum-relációs leképezést, amely kétféleképpen konfigurálható: az entitásokon belül megadott meta-adatok segítségével (annotációs mechanizmus), vagy XML leíróállományok alkalmazásával. A JPA annotációkkal meghatározható az objektumok leképezése a relációs sémának megfelelően, beleértve azok kapcsolatait és több tulajdonság beállítható (pl. adatbázis szintű megkötések az adatokra vonatkozóan, egyes adatokra vonatkozó generálási stratégiák, az adatok betöltési stratégiája (lusta/mohó), entitás hierarchiák adatbázisba történő leképzése stb.).

A JPA specifikál egy lekérdező nyelvet (JPQL), amely funkcionalitásaiban hasonló az SQL nyelvekhez, de jobban megfelel az objektumorientált szemléletmódnak. Biztosít továbbá egy Criteria Query API-t, amely különösen hasznos lekérdezések dinamikus, futási időben történő felépítésekor.

A perzisztenciával kapcsolatos műveletek egy Entity Manager szolgáltatáson keresztül valósíthatóak meg. Ha egy adott entitással kapcsolatban kérjük valamilyen perzisztenciával kapcsolatos művelet elvégzését, az entitás menedzseltté (managed/attached) válik, és a szolgáltatás szinkronizálja az állapotát az adatbázisba mentett állapottal, mindaddig, amíg egy „lecsatolást” követően az illető entitás vissza nem kerül nem menedzselt (unmanaged/detached) állapotba. A menedzselt entitások alkotják a perzisztencia környezetet (Persistence Context), amely Java EE alkalmazások esetében általában tranzakció hatókörű. Az entity managert használó EJB komponensek esetében annak befűzése történhet a dependency injection mechanizmus alapján (pl. a @PersistenceContext annotációval).

A GeoQuestion szerver oldali entitásai JPA meta adatokkal annotáltak, a perzisztencia réteg EclipseLink-en keresztül biztosított, a lekérdezések JPQL, illetve Criteria Query API segítségével vannak felépítve. A rendszer kihasználja a JPA számos előnyös tulajdonságát, például az entitás hierarchiák kezelését, rendezett listák elmentésének lehetőségét stb. Az adathozzáférési réteg komponensei állapotnélküli session bean-ek, amelyekkel a szolgáltatás réteg komponensei lokális interfészeken keresztül kommunikálnak. Az entity manager-ek a konténer által menedzseltek, dependency injection által vannak befűzve a komponensekbe, a perzisztencia kontextusok tranzakció hatókörűek. Az adathozzáférésért felelős repository komponensek hierarchiába szervezettek, a generikus típusok alkalmazásának lehetőségét kihasználva az általános műveletek az alaposztályoktól öröklődnek.

3.4 RESTful alkalmazások

A REST (Representational State Transfer) [11] osztott rendszereken belüli kommunikációt meghatározó architektúra modell, programozási módszer. A modellnek megfelelően az adatok és műveletek erőforrásokként vannak kezelve, amelyeket négy alapművelettel lehet manipulálni (http analógia):

- PUT – erőforrás létrehozása, vagy létező erőforrás módosítása
- POST – új erőforrás létrehozása
- GET – erőforrás lekérése
- DELETE – erőforrás törlése

A REST modellnek megfelelő API-kat nevezzük RESTful API-knak, ilyen kontextusban beszélhetünk RESTful webszolgáltatásokról is. A Java alapú RESTful webszolgáltatásokra vonatkozó szabvány a JAX-RS (Java API for RESTful Web Services), ennek referencia implementációja a Jersey. Az erőforrások JAX-RS meta-adatokkal

annotálhatóak és ezek alapján a keretrendszerek továbbítani tudják ezekhez a kéréseket, a megfelelő formátumban szolgáltatva a paramétereket és visszatérítve a szolgáltatások válaszait.

A konkrét adatátvitel szempontjából az adatok többféle formátumban lehetnek (egyszerű szöveg, HTML, XML, JSON stb.). A Java objektumok konkrét formátumnak megfelelő szerializációját és deszerializációját különböző keretrendszerek támogatják, az erre vonatkozó szabványoknak megfelelőn. Például, a Jackson keretrendszeren keresztül, az alacsonyabb szinten történő JSON feldolgozás mellett, a Jersey lehetőséget ad egyszerű POJO-JSON megfeleltetésre és támogatja a JAX-B szabvány alapján történő megfeleltetést is.

A GeoQuesting szerver a JAX-RS referencia implementációját, a Jersey-t használja. Az adatcsere a DTO (Data Transfer Object) [12] tervezési mintára alapszik. Mind a szerver-, mind a kliensoldalon a megfelelő assembler komponensek biztosítják a modell objektumok és DTO-k közötti átalakítást. A DTO-k szerializálása és deszerializálása a legtöbb esetben egyszerű JSON-POJO megfeleltetéssel történik. Mivel bizonyos DTO-k között öröklődési viszony van és a megfelelő assembler kiválasztásának a konkrét típusnak megfelelően, dinamikusan kell megtörténnie, ezeknek a DTO-knak az esetében az alacsonyabb szintű API használatával történik a JSON adatok feldolgozása. Az Android kliensalkalmazás oldalán a Jackson keretrendszeren keresztül biztosított a szerializálási és deszerializálási mechanizmus, az ezzel kapcsolatos műveletek a háttérben történnek, AsyncTask-ok segítségével.

3.5 Vaadin

A Vaadin [13] egy olyan nyílt forráskódú webalkalmazás keretrendszer, amely lehetőséget biztosít mind szerver oldali, mind kliens oldali fejlesztésre. Szerver oldalon Java servlet technológiát használ, míg kliens oldalon JavaScript, HTML és GWT [14] technológiákat. A kliens és szerver közötti kommunikáció AJAX alapú, a felhasználói felület állapotának a továbbítása JSON formátumban történik.

A keretrendszer előnye, hogy böngészőtől független, így a fejlesztőnek nem kell kompatibilitási problémákra figyelnie és az alkalmazás fejlesztésére koncentrálhat. Továbbá a GWT és JavaScript technológiáknak köszönhetően nem igényli a végfelhasználótól semmilyen böngésző kiegészítés telepítését. Ugyanakkor, mivel a Vaadin nem rendelkezik külső függőségekkel, a legtöbb esetben kompatibilis más Java technológiákkal, könyvtárakkal és eszközökkel. Követi a JavaEE standardokat, de könnyen használható együtt a Spring keretrendszerrel is.

A Vaadin egy eseményvezérelt keretrendszer, nem csupán statikus tartalmak megjelenítésére (weboldalak), hanem főként web alkalmazások (dinamikus tartalmak) fejlesztésére szolgál. Gazdag, interaktív UI komponens gyűjteménnyel rendelkezik. A felhasználói felületek tervezéséhez biztosít a fejlesztőnek alapvető komponenseket, mint a nyomógombok, táblázatok, elhelyezések stb., ezekhez pedig események, figyelők rendelhetők hozzá. Támogatja a „drag and drop" mechanizmust, és lehetőség van a data binding egyszerű használatára, például egy űrlapon található komponensekhez egyetlen művelettel hozzárendelhetőek és megjeleníthetőek egy központi entitásra vonatkozó adatok. Mindez az MVC (model-view-controller) vagy MVP (model-view-presenter) elv támogatásával történik.

Az alap komponensek kiegészíthetők GWT widgetekkel, és testre szabhatók CSS segítségével. Mindemellett rendelkezésünkre állnak úgynevezett Add-onok, azaz extra komponensek, témák és eszköztárak. Ezek legtöbb esetben ingyenesen elérhetők a Vaadin Directory-ban, ugyanakkor a fejlesztők itt közzé tehetik a saját komponenseiket is.

A GeoQuesting alkalmazás webes játékszerkesztő felülete a Vaadin keretrendszer segítségével készült. Az említett add-on komponensek több ízben is hasznosnak bizonyultak az alkalmazás fejlesztésekor. Jelen van például egy küldetéshez kötött úgynevezett irány feltétel beállításánál, a pozíció meghatározásakor térkép formájában stb. Ezeknek a komponenseknek a megjelenítése felhasználóbarát, használatuk egyszerű, beintegrálásuk a projektbe egyetlen maven függőség megadásával lehetséges. Kiemelendő még a már említett CDI, amely szintén Vaadin add-on komponensként van bekötve a rendszerbe. A CDI használata annotációkkal történik, biztosít egy ViewProvider-t, mely a nézetek közötti navigációért felelős. Egy másik példa az add-on komponensek használatára, a Leaflet térkép beintegrálása, amely a GPS pozíció egyszerűbb megadása érdekében volt bevezetve.

A Vaadin további előnye, hogy a munkamenet követése és a biztonsági mechanizmusok automatikusan biztosítottak, a szerver oldalon történő validáció pedig kiszűri az adatmanipulációt.

3.6 Az Android platform

Az Android platform egy Linux kernelre épülő, főként mobileszközökön használt operációs rendszer. 2005-ben került a Google tulajdonába, napjainkban az Android Nyílt Forrás Kódú Projekt fejlesztését a Google vezeti.

A platformra történő fejlesztés szempontjából az elsődleges programozási nyelv a Java. A fordítás során a Java forráskódból fordított bájtkód át lesz alakítva az Android operációs rendszeren futó Dalvik virtuális gép formátumába, Dalvik Executable (.dex) állományokba. Ezek az állományok az erőforrás állományokkal, az alkalmazás leíróval és az

erőforrásadatokkal együtt egy .apk állományba tömörítődnek, amely felkerül a felhasználó eszközére.

Az Android platform többrétegű architektúrával rendelkezik. A legalsó rétegen található a Linux kernel, a következő rétegen találhatóak a programkönyvtárak és szolgáltatások (pl. SqLite, OpenGL, FreeType, SSL, WebKit stb.). Ezekre a könyvtárakra épül az Android futtatókörnyezet, amelynek alapja a Dalvik virtuális gép. Fölötte levő rétegen találjuk az Andorid alkalmazás keretrendszert, mely segítségével lehetővé válik az új felhasználói alkalmazások fejlesztése.

Az Android platformra történő fejlesztéshez a Google biztosít egy SDK-t (Software Development Kit), amely egy összetett fejlesztési eszköz, tartalmaz hibakeresőt, könyvtárakat, dokumentumokat, példakódot, oktatóanyagot és emulátort.

A dolgozatban leírt Android alkalmazás esetén a nézetek fragmentek segítségével vannak megoldva. A fő nézet egy activity, amely tartalmaz egy konténert. Ebbe a konténerbe kerülnek be a nézeteknek megfelelő fragmentek. Azért ezt a megoldást választották a fejlesztők, mert lehetőség van hatékonyan, futási időben a fragmentek cseréjére.

A GeoQuesting játékok feladványainak megoldásához szükséges a telefonkészülék különböző szenzorjainak használata. Egy helyszín megkeresése esetén az alkalmazás vizsgálja a telefonkészülék GPS koordinátáit és összehasonlítja a játékszerkesztő által megadott koordinátákkal. A pozíció szenzorok közül a mágneses mező értékét lekérdezve ellenőrizni lehet, hogy a játékos milyen irányba van fordulva a telefonkészülékkel, például milyen épületet néz.

A GPS jel elérhetősége épületeken belül korlátozott, ezért a játékban lehetőség van olyan feladatok létrehozására, amelyek a készülékek NFC (Near Field Communication) moduljával NFC illetve RFID (Radio-Frequency Identification) kártyák olvasásának segítségével határozza meg a játékos helyzetét. Az NFC egy kommunikációs szabvány, amely rádióhullámokat használ az adatcserére, kis hatótávolságú, az eszközök egymáshoz közel helyezésével valósul meg a kommunikáció. A kapcsolat felépítéséhez leegyszerűsített beállítások szükségesek, ezért a használata egyszerű és gördülékeny, a Bluetooth-al ellentétben ahol az eszközök párosítása szükséges az adatcsere előtt. Mivel rádióhullámok segítségével valósul meg a kommunikáció, rossz fényviszonyok esetén is használható, a vonalkóddal szemben.

A GeoQuesting alkalmazás mobil része mögött egy SQLite adatbázis van, amelyben csak minimális adatok tárolódnak, például a bejelentkezett felhasználó adatai. A további adatokat (játékok, küldetések, feladatok) a felhasználó a szerver oldalon levő MySQL adatbázisból, REST API híváson keresztül kapja.

3.7 AR és képfelismerő keretrendszerek

A kiterjesztett valóság (Augmented Reality) egy dinamikusan fejlődő, viszonylag új kutatási és fejlesztési ágazat. Nem a virtuális valóságot jelenti, ahol a felhasználó teljesen mesterséges világot lát, hanem a valós környezet van kiegészítve a számítógép által generált mesterséges elemekkel. Felhasználási területek: reklám, marketing, navigáció, orvostudomány, szórakoztatóipar stb.

A kiterjesztett valóság alapú rendszereket a virtuális tárgyak megjelenítésének módszere alapján két csoportba sorolhatjuk: markeres, illetve pozíció és irányalapú. A marker alapú AR egy speciális képrészletet keres, amely kiemelkedik a környezetből (általában fehér alapon fekete keret, mely egy egyedi fekete-fehér alakzatot tartalmaz). A kamera segítségével meghatározható a marker helye és pozíciója, így ráhelyezhető a virtuális objektum. A pozíció és irányalapú AR-t általában mobiltelefonok esetében alkalmazzák. Ahhoz, hogy a valós kép helyét meghatározzuk, szükség van a felhasználó helyzetére (GPS koordináta), arra, hogy milyen irányba van fordulva (az iránytűjének helyzete), milyen dőlésszöge van a készüléknek (gyorsulásmerő adatai) és esetleg a kamera képén megjelenő egyedi képrészletre.

A GeoQuesting elsősorban a szenzorok adataira támaszkodik AR elemek megjelenítésénél, de fejlesztése során történtek próbálkozások képfelismerésen alapuló módszer integrációjára is. Egy olyan funkcionalitás bevezetése volt a cél, amely egy aktuálisan látott képet összehasonlít egy már régebben elkészített, a játékszerkesztő által feltöltött képpel. Az egyezés bizonyíthatná, hogy a játékos éppen a megfelelő helyszínen tartózkodik. Ennek a funkcionalitásnak a megvalósítására az OpenCV [15] nyílt forráskódú könyvtár volt felhasználva, amely segítséget nyújt a valós idejű számítógépes képfeldolgozásban. Ebben a könyvtárban találhatók előre megírt algoritmusok, melyeket alkalmazni lehet a képfelismerésben. Az OpenCV több különböző módszer segítségével tudja a képek jellemzőit azonosítani és leírni. Ilyenek például a BRIEF (Binary Robust Independent Elementary Features), SIFT (Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features) és ORB (Oriented FAST and Rotated BRIEF).

A dolgozatban leírt alkalmazás esetén az ORB módszert alkalmazták a fejlesztők, amely elsősorban olyan domináns pontokat keres a képen, melyek kiemelkednek, majd ezeket a pontokat összehasonlítja a tudásbázisban levő kép pontjaival. A fejlesztőnek lehetősége van meghatározni az egyezés megállapításához szükséges megegyező pontok számát. Második lépésben a módszer a megegyező pontok között vizsgálja a távolságot, ezzel kiküszöbölve, hogy ne egymás mellett helyezkedjenek el a képen. Ennek a távolságnak az esetében szintén beállítható egy küszöbérték.

Sajnos a tesztelések során nem bizonyult elég hatékonynak ez a módszer, a környezeti tényezők, napszakok, időjárás zavarták a képfelismerést. Nem sikerült olyan eredményeket elérni, amelyek indokolták volna függővé tenni a kliensalkalmazást az OpenCV keretrendszertől, ezért a jelenlegi rendszer nem tartalmazza a képfelismerésen alapuló funkcionalitást. Folyamatban van viszont további képfelismerésen alapuló AR keretrendszerek keresése és kipróbálása.

3.8 További technológiák

A Swagger egy nyelvfüggetlen keretrendszer, amely segítséget nyújt RESTful webszolgáltatások leírására, előállítására, megjelenítésére és tesztelésére. A specifikáció elkészítését és fejlesztését a Reverb cég Wordnik[2] online értelmező szótár fejlesztése során kezdeményezte. A GeoQuesting esetén az REST API hívások dokumentálásra és tesztelésére a Swagger által biztosított eszközöket használták a fejlesztők.

A GeoQuesting rendszeren belül a naplózást a log4j keretrendszer biztosítja. A log4j fölötti absztrakciós szint az slf4j, amely lehetőséget biztosít a naplózási keretrendszer egyszerű cseréjére, a kód módosítása nélkül.

A rendszer több részénél (pl. webes felület komponenseivel kapcsolatos események továbbítása a megfelelő vezérlőkhöz, vagy a mobilalkalmazásban a felület frissítésére egy feladat befejezésekor) cél volt a komponensek közötti szoros függőségek felszámolása. Erre az egyik alkalmazott megoldás a „feliratkozás-értesítés” mechanizmus, amelynek megvalósításához a GeoQuesting Google Guava EventBus-t használ.

Az adatbázisba való mentés előtt mindig szükséges az adatok helyességének ellenőrzése. Az adatbázis szintű validáció mellett (a GeoQuesting esetében JPA meta adatokkal vannak megadva az adatbázis szintű megszorítások) jó stratégia már a felsőbb szinteken elvégezni a validációt. Erre a célra bizonyos esetekben alkalmazhatóak az entitások szintjén leírt megkötések, más esetekben a szolgáltatási rétegen belüli komponensek végezhetnek ellenőrzéseket. A GeoQuesting projekt esetében a validáció a JSR 303: Bean Validation specifikációra épülő Hibernate Validator keretrendszert használja, amely a következő három alapvető lehetőséget biztosítja a helyesség ellenőrzésére: megszorítások alkalmazása, ezek megszegésének kezelése, valamint saját, testreszabott validátorok létrehozása. A GeoQuesting projektben több helyen is találhatók az entitásokra vonatkozó megszorítások, melyek Bean Validation annotációk segítségével vannak leírva. Ilyen például a felhasználók email címére vonatkozó megszorítás, a dátumok helyességének ellenőrzése stb.

[2] https://www.wordnik.com/

A megszorítások alapján a Vaadin keretrendszer a webes felület szintjén is biztosítani tudja a bevitt adatok helyességét.

Mivel a GeoQuesting egy összetett rendszer, a módosításokat követő felépítés és kitelepítés (build és deploy) viszonylag sok időt vesz igénybe. Szerver oldalon ennek a problémának a kiküszöbölését a JRebel [16] fejlesztési eszköz oldotta meg. A JRebel lehetővé teszi a kód újratöltését anélkül, hogy a konténert újra kellene indítani, vagy az alkalmazást újra ki kellene telepíteni a szerverre. Ha csak egy apró CSS módosítás eredményét szeretné megnézni a fejlesztő, elég a böngészőben frissítenie az alkalmazást, hogy a változások azonnal láthatóak legyenek. A JRebel kiterjeszti a használt class loader-t olyan módon, hogy az menedzselni tudja az újratöltött osztályokat. Egy osztály betöltésekor a JRebel a classpath-ben megkeresi a neki megfelelő .class állományt, majd összekapcsolja ezeket. Ezután a betöltött osztályban bekövetkezett változások a kiterjesztett class loader-en keresztül automatikusan propagálva lesznek az alkalmazás felé.

A GeoQuesting mobilalkalmazáson belül lehetőség van egy térkép megjelenítésére, amely egy útvonalat kínál a felhasználónak az aktuális pozíciójától a következő ellenőrzőpontig, ezzel is segítve a felhasználót a célpont megtalálásában. Ehhez a projekt a Google Maps API-t használja, amely az egész Földről biztosít térképeket és műholdfelvételeket, lehetőséget ad utcatérkép nézetre, útvonaltervezésre, stb.

Az alkalmazás játékszerkesztői felületén szintén nagy segítség, ha az ellenőrzőpontok meghatározásánál a felhasználónak nem kell fejből tudnia a pontos koordinátákat, vagy külső program segítségével megkeresnie azokat, hanem helyben biztosítva van egy térkép nézet, amellyel könnyen meg tudja jelölni a pontos helyszíneket. Ez a funkcionalitás a Leaflet keretrendszer segítségével biztosított.

4 Az Arthas GeoQuesting projekt

A következő rész ismerteti az Arthas GeoQuesting projekt követelményeit, bemutatja az alkalmazás architektúráját.

4.1 Alapvető követelmények és funkcionalitások

A következőkben a projekt funkcionális követelményei lesznek összefoglalva. A leírás nem tér ki a részletekre és nem tartalmazza a nem funkcionális követelményeket.

4.1.1 Szerver oldal

A szerveroldal felelős az adatbázissal való kommunikációért, illetve az alkalmazáslogika megvalósításáért, magába foglalja a webes játékszerkesztő felületet, illetve az API-t, amely RESTful webszolgáltatások segítségével szolgálja ki a telefonos kliensalkalmazással kéréseit. A szerver fontosabb funkcionalitásai:

- Biztosítja az adathozzáférési réteget, felelős a projekt központi entitásainak adatbázisba való mentéséért, módosításáért, törléséért, illetve a lekérdezésekért.
- Biztosítja az alkalmazáslogikát megvalósító komponenseket tartalmazó, illetve a komponensek elérhető szolgáltatásait meghatározó szolgáltatási réteget.
- A webes játékszerkesztő felület biztosít:
  - Egy bejelentkező felületet.
  - Egy felületet játék létrehozására, adatainak beállítására, küldetések hozzáadására és küldetések szerkesztésére.
    - Helyszín megtalálásával kapcsolatos feladatoknál térképnézet biztosítása.
    - Kérdésen alapuló feladványoknál több helyes válasz bevitelének lehetősége.
  - Egy felületet a különböző játékok (folyamatban levő, nyilvánossá tett, befejezett) listázására és menedzsmentjére.
- A GeoQuesting API lehetőséget biztosít az adatok küldésére a kliens és szerver között. Az adatokat DTO formájában küldi a két modul között, felelős ezen objektumok átalakításáért a megfelelő adatmodellbe, amelyet assemblerek segítségével hajt végre.

4.1.2 Kliens oldal

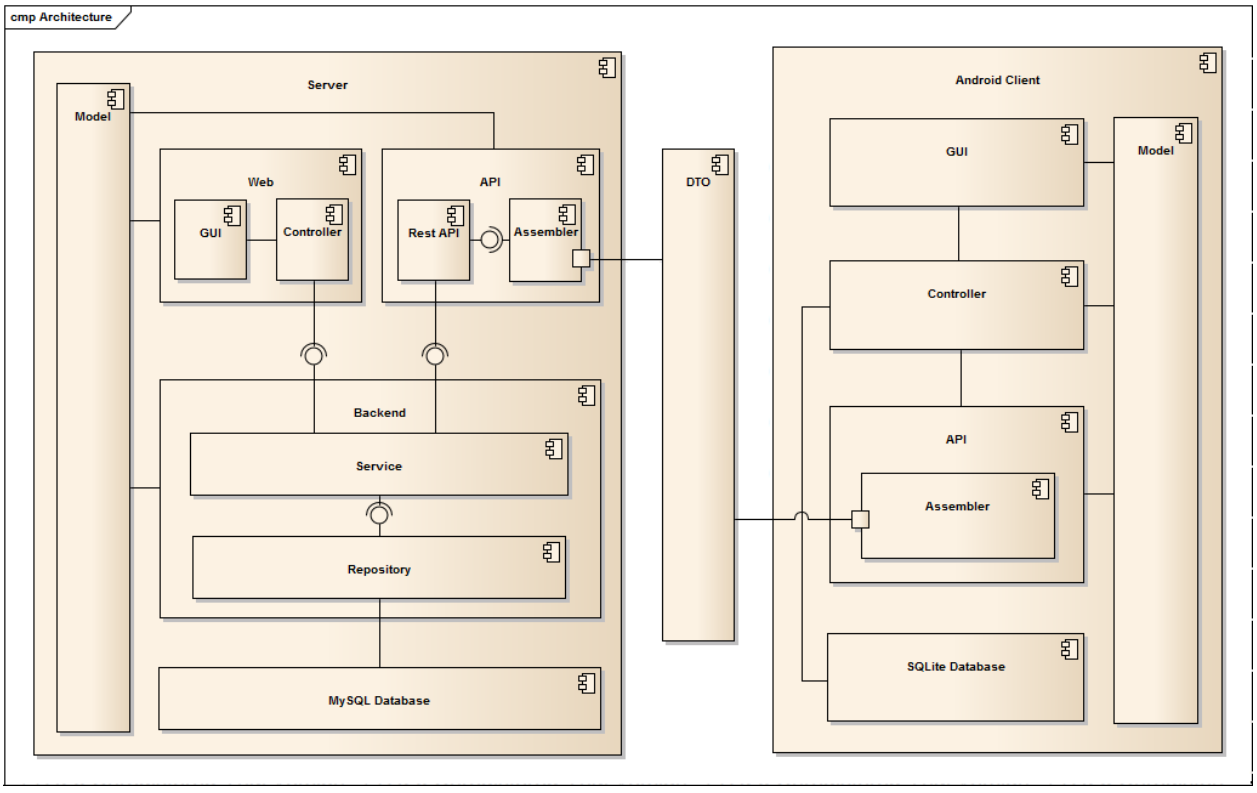
A kliens oldal valósítja meg a tulajdonképpeni játékot. Az adatokat a szerver oldalon levő adatbázisból kapja a GeoQuesting API-n keresztül, és itt küldi vissza az általa létrehozott információkat is.

Fontosabb funkcionális követelmények:

- A nyilvánossá tett játékok megjelenítése.
- Játék kiválasztása, adatainak megjelenítése, és indítása.
- A küldetések megjelenítése (feladvány, adott helyszín megtalálása stb.).
    - GPS feltétel (helyszín megtalálása) alapú feladvány esetében térképnézet és útvonal javaslat megjelenítése.
    - Iránytű megjelenítése orientációs feladványhoz
- A megoldás helyességének validációjához szükséges információk átküldése, a megoldások ellenőrzése: helyszín megtalálására vonatkozó feladat esetén a koordináták és/vagy készülék irányának ellenőrzése, kérdés esetén a válasz helyességének ellenőrzése (adott pontosságú egyezés) stb.

4.2 Architektúra

A rendszer komponenseit és az azok között lévő kapcsolatokat az 1. ábra szemlélteti.



1. ábra: Az ARTHAS GeoQuesting projekt architektúra diagramja

Szerver oldalon az alkalmazás központi entitásai a model csomagban lévő osztályok által vannak reprezentálva. Ezeket az entitásokat közvetlenül használja az alkalmazás másik három alrendszere: a Backend, a Web valamint az API. Az adatok tárolására MySQL relációs adatbázist használ a rendszer.

A Backend alrendszer tartalmazza az adathozzáférés és adatszolgáltatás rétegeket. Az adatok lekérdezése, beszúrása és törlése a Repository rétegben van megvalósítva a már említett Entity Manager szolgáltatáson keresztül. A szolgáltatás réteg (Service) interfészeken keresztül továbbítja az adatokat a webes komponensek, illetve az API felé. Az adatok eléréséhez a repository réteg interfészein keresztül kommunikál a komponensekkel.

A web komponensek azokat a felhasználókat hivatottak kiszolgálni, akik más felhasználók számára új játékokat szeretnének létrehozni és ezeket szerkeszteni. Az MVC elvnek megfelelően különválaszthatóak az irányításért felelős komponensek, valamint a megjelenítést szolgálók. A Controller a vezérlést valósítja meg, a GUI biztosítja a szerkesztőfelületet.

A szerverhez hasonlóan a kliens oldalon is megtalálható egy model csomag, amely a kliens oldali modell osztályokat tartalmazza. Nagyrészük a szerver oldali entitások egyszerűsített változata. Ez azért indokolt, mert a kliensalkalmazás felhasználói bizonyos információkban nem érdekeltek, és fölösleges lenne a telefon memóriáját nem használt adatokkal terhelni.

A két rendszer közötti kommunikációt a szerver oldalon elhelyezkedő API biztosítja, ezen belül is az egyes REST API hívások. Az API biztosítja, hogy megfelelő kérésekre minden esetben megfelelő, helyes válaszok érkezzenek. Az alkalmazás az API hívások során DTO objektumokat küld, tulajdonképpen ezek valósítják meg a kapcsolatot a két rendszer között. Mind szerver, mind kliens oldalon megtalálhatóak az API-n belül az Assembler-ek, amelyek a DTO komponenseket átalakítják a megfelelő modell osztályokba.

A mobil kliens az adatokat SQLite adatbázisban tárolja. A Controller felelős az adatokhoz való hozzáférésért, valamint a kliens oldali üzleti logikával kapcsolatos műveletek megvalósításáért. Itt van implementálva többek között az egyes küldetésekhez tartozó feltételek kiértékelése is.

A kliens oldali alkalmazás egy könnyen használható felhasználói felületet biztosít. Ide tartozik az egyes feladványtípusok megjelenítése (térkép kirajzolása, iránytű megjelenítése), a játék menetének nyomon követése (állapotjelző megjelenítése), stb.

5 A GeoQuesting használata

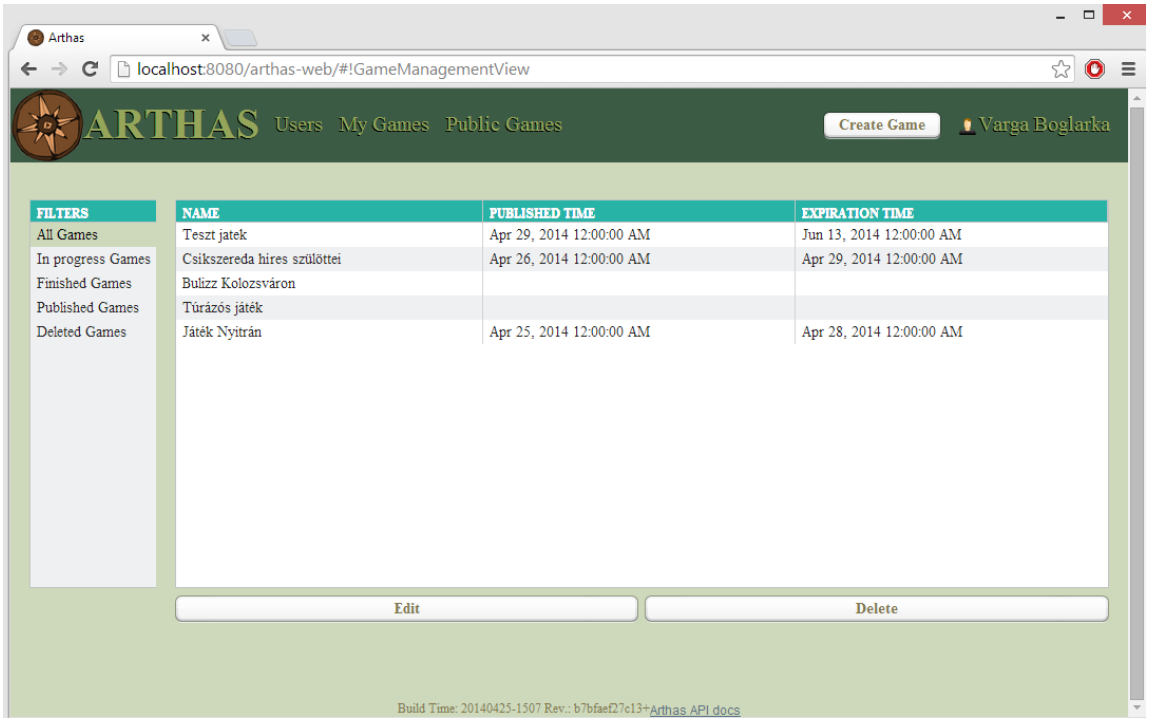
Felhasználói szempontból a GeoQuesting két alkalmazást biztosít: a webes játékszerkesztő felületet, illetve a mobil kliensalkalmazást.

5.1 A webes játékszerkesztő felület

A webes játékszerkesztő felület segítségével a felhasználók játékokat tervezhetnek meg, szerkeszthetik ezeket, valamint a későbbiekben nyomon követhetik a játékaikkal kapcsolatos adatokat. A web alkalmazás biztosít egy adminisztrációs felületet is, ahol a felhasználók menedzselhetőek.

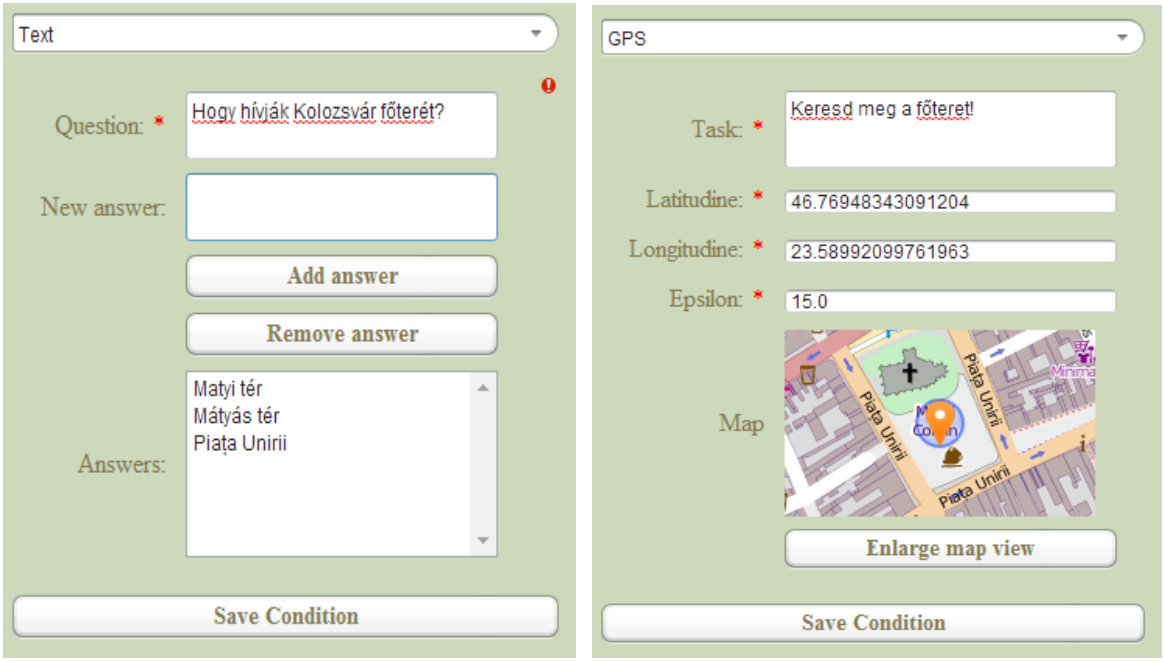
A könnyebb megértés érdekében a dolgozat egy példán keresztül mutatja be a szerkesztőfelület fontosabb funkcionalitásait.

A GeoQuesting oldalát látogató felhasználókat egy bejelentkezési felület várja, melyen a felhasználó email címének és jelszavának megadására van szükség. Sikeres bejelentkezés után a felületen a felhasználó által létrehozott játékok láthatóak, néhány fontosabb információt kiemelve (2. ábra). Ugyanitt van lehetőség játékok szűrésére is, listázhatóak például a már elkezdett játékok, a befejezett játékok, a felhasználó által létrehozott és publikált játékok stb. A még nem publikált játékok esetében lehetőség van ezek szerkesztésére, egyébként megtekinthető a teljes játék, de módosításra már nincs lehetőség.



2. ábra: Játékok listázása és szűrése a webes szerkesztő felületen

Játék létrehozásakor a felhasználónak meg kell adnia a játék nevét (jelen példában „Ismerjük meg Kolozsvár nevezetességeit”), valamint leírását. Opcionálisan megadható a régió, ahol a játék zajlik (esetünkben Kolozsvár), illetve nyilvánossá tehető a játék (erre a későbbiekben is lehetőség van). A játék csak akkor lesz más felhasználók számára is publikus és elérhető, ha a játékkészítő ezt engedélyezi. A publikáláshoz be kell állítani a játék elkezdésének időpontját, illetve lejárási határidejét. Kizárólag ebben a periódusban lesz játszható az adott játék.



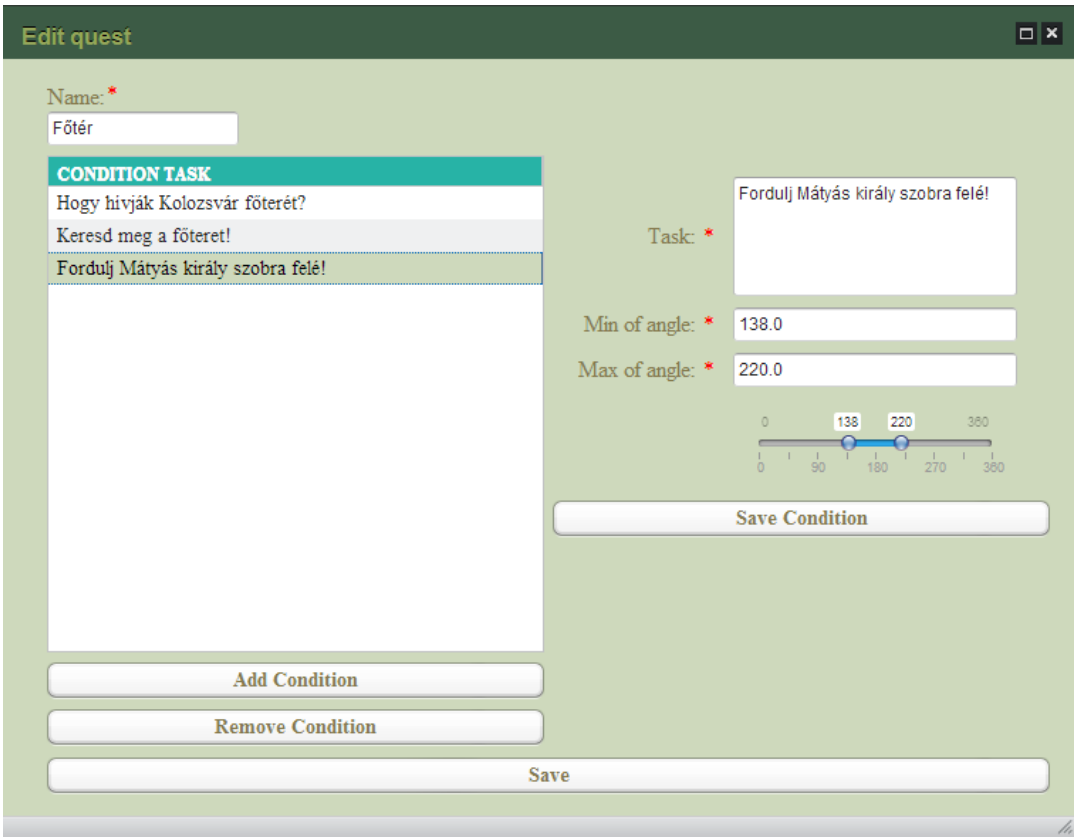
3. ábra: Szöveg (kérdésre adott válasz) és GPS (helyszín megtalálása) feltételek beállításai

A következő lépésekben össze kell állítani a játékhoz tartozó küldetéseket. A szerkesztő egy küldetés esetében több különböző feltétel típust kombinálhat, ezek teljesítése jelenti egy küldetés teljesítését. A példát követve elsőként egy szöveges (3. ábra), azaz kérdés-válasz alapú feladatot kell majd a játékosnak teljesítenie. A kérdésre („Hogy hívják Kolozsvár főterét?”) a játékosok többféleképpen adhatnak választ (pl. magyarul, románul, ékezetekkel vagy ékezetek nélkül), ezért a szerkesztőnek is lehetősége van megadni több elfogadható választ. Ezen kívül a validációnál csak adott pontosságú egyezés lesz ellenőrizve, például egy-két karakterelütés nem fog számítani.

A példa esetében a játékszerkesztő szeretné, ha a játékos el is menne a főtérre, így tovább bővíti a küldetést egy GPS alapú feladattal (3. ábra), ahol pontosan meghatározza a főtér koordinátáit. Ebben egy térkép nézet nyújt segítséget, egyetlen kattintással bejelölheti a helyszínt. Megadható még egy eltérési küszöbérték (méterben), amelyen belül teljesítettnek tekinti az alkalmazás az adott feltételt.

A küldetés következő pontja Mátyás király szobra felé irányul, ezért a játékosnak abba az irányba kellene fordulnia. Ezt a szerkesztő úgy érheti el, hogy egy irány feltétellel (4. ábra)

bővíti a feladatok listáját. A játékost arra utasítja, hogy forduljon a szobor felé, és az égtájaknak megfelelően megad egy szög intervallumot, amelyet majd a telefonon egy mágneses szenzor segítségével ellenőrzi a rendszer. Egy küldetésen belül több szöveg alapú feladat is megadható, azonban irány, NFC vagy GPS alapú feladatok hozzáadásához új küldetés létrehozására van szükség.



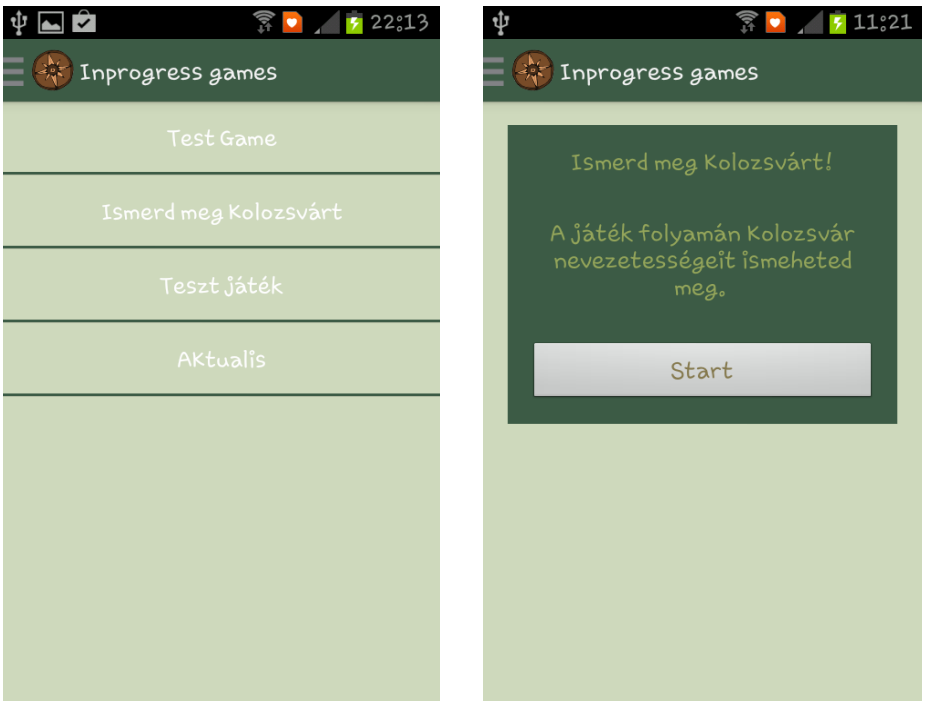
4. ábra: A küldetésekhez tartozó feltételek listája, illetve az irány feltétel beállításai

A játék szerkesztését bármikor lezárhatjuk a mentés gombbal, és bármikor folytathatjuk a szerkesztését mindaddig, míg a játékosok számára publikálatlan marad.

5.2 A mobil kliensalkalmazás

A mobilalkalmazás lehetőséget nyújt a felhasználónak a tulajdonképpeni játékra, vagyis egy előre elkészített játék feladványainak megoldására. Ha a felhasználó megnyitja az alkalmazást, egy bejelentkező felület jelenik meg. Sikeres bejelentkezés után megjelenítheti az adatbázisban szereplő összes nyilvánossá tett játékot (5. ábra). Ebből a listából kiválaszthatja a neki tetsző játékot, több információt tudhat meg róla, majd elindíthatja azt.

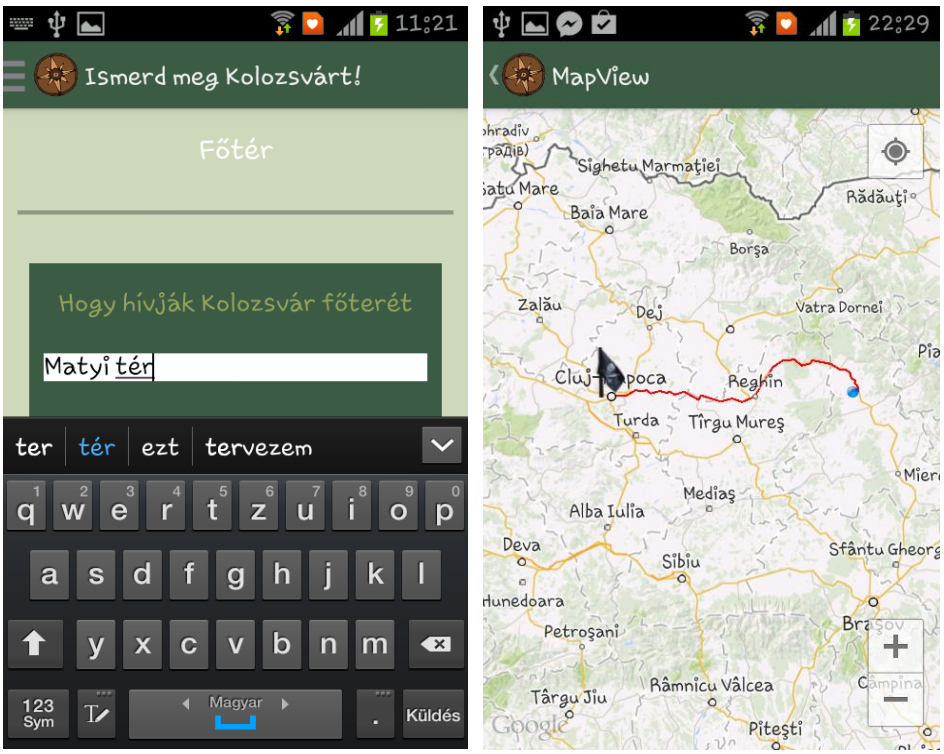
Az „Ismerd meg Kolozsvárt!” játék példában az első küldetés megnevezése: „Főtér”. Ehhez tartozik több különböző feladat, amelyeket a játékosnak egymás után kell teljesítenie.



5. ábra: Játékok listája és kiválasztott játék leírása

Elsőként egy kérdésre kell válaszolnia Kolozsvár főterével kapcsolatosan. (6. ábra). A válasz ellenőrzésére a GeoQuesting a Jaro-Winkler távolság[3] kiszámítására használt algoritmust implementál, amely a megadott választ összehasonlítja a játékkészítő által megadott összes válaszlehetőséggel. Az algoritmus az összehasonlítás után visszatérít egy 0 és 1 közötti értéket, és ha ez az érték 0,8 feletti, akkor a válasz helyesnek lesz tekintve a rendszer által. A következő feladat egy GPS koordinátákkal megadott helyszínre küldi el a felhasználót, Kolozsvár főterére. Az alkalmazás segítségképpen biztosít egy térképnézetet, amelyen egy útvonal javaslatot mutat a játékosnak a jelenlegi helyzetétől a feladat által mutatott helyig. Ezt az útvonalat az alkalmazás a Google API-tól kéri le, és a Google térképszolgáltatás segítségével jeleníti meg. Ha a játékos eljut a megfelelő helyszínre, a következő feladat arra utasítja a játékost, hogy forduljon a Mátyás szobor irányába (egy DirectionCondition típusú feltételen alapuló feladat). A telefon mágneses szenzorjának segítségével az alkalmazás megállapítja, hogy milyen irányba van fordulva a játékos. Miután a játékos teljesítette a küldetésen belüli összes feladatot, megkapja a következő küldetést. Ha a játékot nem fejezi be egyszerre, lehetősége van folytatni egy későbbi időpontban az utoljára elkezdett küldetéstől. A játék akkor ér véget, ha a játékos az összes küldetést teljesítette.

[3] http://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance



6. ábra: kérdés alapú feladvány és útvonal javaslat adott helyszín megtalálásához

Következtetések és továbbfejlesztési lehetőségek

A GeoQuesting keretein belül sikerült egy olyan játékalkalmazást létrehozni, amelyet játszva bárki könnyen, játékos formában, szórakoztató módon ismerheti meg egy adott régió nevezetességeit. Az alkalmazás segítséget nyújthat az oktatásban, felhasználható turisztikában, reklám- és szórakoztatóiparban is. A webes játékszerkesztő felület segítségével a játékok könnyen létrehozhatóak, szerkeszthetőek és menedzselhetőek. Architektúrájának és a felhasznált technológiáknak köszönhetően a rendszer könnyen bővíthető, skálázható és karbantartható.

A fejlesztés során számos további lehetséges funkcionalitás, továbbfejlesztési lehetőség felmerült. Néhány fontosabb terv:

- Mivel az OpenCV alapú képfelismerő modul integrálása nem vezetett megfelelő eredményhez, folyamatban van más képfelismerésen alapuló AR keretrendszerek kipróbálása.
- A mobilkészülék szenzorjain alapuló pontosabb helymeghatározás AR tartalmak és kapcsolódó feladványok megjelenítéséhez. Felhasználható a játékos pozíciója, az, hogy milyen irányba van fordulva (iránytű), milyen dőlésszöge van a készüléknek (gyorsulásmérő).
- Új feladványtípusok bevezetése (pl. vonalkód alapján azonosított helyszínek NFC nélküli készülékek számára, stb.)
- Belső értékelési- (játékok értékelése), pontozási- (ranglisták) és jutalomrendszer bevezetése
- Játékok automatikus szűrése (pl. a felhasználó pozíciója, vagy a készülékek adottságai alapján)
- Virtuális tárgyak megosztása a játékosok között (pl. az NFC kártyák segítségével)
- Felhasználók menedzsmentjével és biztonsággal kapcsolatos megoldások bevezetése

Tervbe van véve továbbá egy publikus tesztjáték elkészítése, amelyet követően publikus népszerűsítő játékokat lehetne publikálni.

Hivatkozások

1. Zsolt Bálint, Botond Kiss, Beáta Magyari, Károly Simon, (2012), Augmented Reality and Image Recognition Based Framework for Treasure Hunt Games, Pre-proceedings of the 2012 IEEE 10th International Symposium on Intellinget Systems and Informatics (included into the IEEE Xplore database), Subotica, Serbia, 147-152.
2. Srum Hivatalos Weboldal, https://www.scrum.org/, utolsó megtekintés dátuma, 2014.04.20.
3. Mercurial Hivatalos Weboldal, http://mercurial.selenic.com/, utolsó megtekintés dátuma: 2014.03.28.
4. Apache Maven Hivatalos Weboldal, http://maven.apache.org/, utolsó megtekintés dátuma: 2014.03.28.
5. SonarSource Hivatalos Weboldal, http://www.sonarsource.org/features/, utolsó megtekintés dátuma: 2014.03.28.
6. Arun Gupta, Java EE 7 Essentials, O'Reilly Media, 2013.
7. Enterprise JavaBeans hivatalos dokumentáció, http://www.oracle.com/technetwork/-java/javaee/ejb/index.html, utolsó megtekintés dátuma: 2014.03.17.
8. Richard Monson-Haefel, Bill Burke, Enterprise JavaBeans 3.1, O'Reilly Media, 2006.
9. CDI specifikáció, http://www.cdi-spec.org/, utolsó megtekintés dátuma 2014.04.20.
10. Java Persistence API, http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html, utolsó megtekintés dátuma: 2014.03.12.
11. RESTFul webszolgáltatásról szóló dokumentáció, http://www.oracle.com/technetwork/articles/javase/restful-142517.html, utolsó megtekintés dátuma 2014.04.24.
12. Martin Fowler, Patterns of Enterprise Application Architecture, Addison-Wesley, 2003
13. Marko Grönroos, Book of Vaadin, Vaadin Ltd, 2012.
14. Google Web Toolkit hivatalos oldala, http://www.gwtproject.org/, utolsó megtekintés: 2014.04.29.
15. Az OpenCV hivatalos oldala, http://www.opencv.org/, utolsó megtekintés dátuma 2014.03.30.
16. A JRebel hivatalos oldala, http://zeroturnaround.com/software/jrebel/, utolsó megtekintés dátuma: 2014.03.17