XXI. reál- és humántudományi Erdélyi Tudományos Diákköri Konferencia (ETDK)
Kolozsvár, 2018. május 24–27.

EKETour

Szoftverplatform túrák szervezéséhez

Szerzők:

Bakó Bencze
Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar, Informatika szak, III. év
Bartha Vivien–Emőke
Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar, Informatika szak, III. év

Témavezetők:

dr. Simon Károly, egyetemi adjunktus
Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar
Kintzel Levente, projektmenedzser,
Codespring
Lőrincz Csaba, szoftverfejlesztő,
Codespring

Kivonat

Az EKETour szoftverrendszer célközönsége minden olyan szervezet, amely túrák és/vagy szabadtéri rendezvények lebonyolításával foglalkozik. A szoftver kialakításához az Erdélyi Kárpát Egyesület (EKE) járult hozzá, az általa megfogalmazott elvárásokat figyelembe véve jött létre a platform első verziója.

A cél egy egységesített regisztrációs felület kialakítása volt a túrák számára. Ezáltal a visszajáró túrázók jelentkezése leegyszerűsödik, mivel az általános információkat csak a regisztráció első szakaszában kell megadniuk. A felület arra is lehetőséget ad, hogy egy felhasználó egyszerre több személy adatait felvezesse egy-egy profilba.

A szervezők a túrák nyomonkövetését jelenleg papíron végzik, így ebből a szempontból is szükségessé vált egy digitális megoldás kialakítása. A túrázók NFC kártyákat kapnak, amelyekkel igazolhatják magukat az ellenőrzőpontoknál. A szervezők egy mobil alkalmazás segítségével tudják ellenőrizni az elhaladókat, validálni és naplózni adataikat.

Tartalomjegyzék

Bevezető

Az Erdélyi Kárpát Egyesület (EKE) 1891-től folytatja munkásságát, rendszeresen szervez-nek túrákat a természet kedvelőinek. Ezek útvonal és táv szerint több altúrára tagolódnak, hogy mindenki megtalálhassa a számára legmegfelelőbbet. Heti rendszerességgel indítanak kisebb túrákat, emellett évente három nevezetes emlék- és teljesítménytúrát szervez az egyesület.

Az EKETour projekt célja egy olyan szoftverrendszer megvalósítása, amely segít az ilyen jellegű túrák lebonyolításában.

A kolozsvári EKE által megfogalmazott elvárásokat szem előtt tartva jött létre a platform első verziója, így a projekt specifikusabb célja az volt, hogy a szervezet által megrendezett emléktúrák számára egy egységesített regisztrációs felületet valósítson meg.

Az alkalmazás funkcionalitásait egy egyszerű példán keresztül lehet a legjobban szemléltetni: egy család részt szeretne venni a májusban megszervezésre kerülő Jókai Mór emlék- és teljesítménytúrán. Az édesapa regisztrál a felületen megadva az e-mail címét és telefonszámát. A sikeresen létrehozott felhasználóval a további szükséges adatokat felvezeti egy profilba, amit a későbbiekben könnyen módosíthat. Az alkalmazás lehetőséget nyújt több profil készítésére, így az édesapa felhasználójával a család összes tagját regisztrálhatja. A létrehozott profilokkal lehet jelentkezni a kívánt túra egy kiválasztott változatára. Nincs megkötve, hogy az előzőleg együtt regisztráló család ugyanazt a távot tegye meg. Egy edzett természetjáró választhatja a leghosszabb útvonalakat, míg a kezdőnek egy néhány kilométeres könnyebb terep is megfelelő kihívásnak számíthat.

A rendszernek van egy Android alkalmazás része, amelyet a szervezők használhatnak. Az alkalmazás révén a túrák menedzselése egyszerűbbé válik: a startpontoknál a résztvevők egy-egy NFC-kártyát kapnak, majd az ellenőrzőpontoknál elhaladva az ott várakozó szervezők az alkalmazás segítségével leolvassák a kártyát. Az azonosítással párhuzamosan így megtörténik az adatok naplózása is.

Az EKETour projekt webes felületén a rendszergazda a túrák menedzselése mellett a naplózott adatok alapján a résztvevőket is követheti, annak érdekében, hogy a felmerülő problémákat gyorsan jelezhesse a terepen dolgozóknak. A rendszer tartalmaz egy kimutatásokat megjelenítő felületet is, amelyen különböző paraméterek megadásával lehet létrehozni jelentéseket.

A projekt fejlesztése a Codespring Mentorprogram szakmai gyakorlatának ideje alatt indult, Kintzel Levente és Lőrincz Csaba mentorálásával és Dr. Simon Károly szakmai irányításával. A csapatot Sulyok Csaba is segítette a tevékenységek koordinációjában. Az egyetemi csoportos projekt tantárgy keretén belül a csapat új tagokkal bővült: Petkes Richárd András, Petkes Zsolt József és Szarvadi Levente András egy egyetemi félév idejére csatlakoztak a fejlesztőkhöz. Ezt követően a jelen dolgozat szerzői folytatták a fejlesztést.

1. Az EKETour projekt

1.1. Terminológia

A dolgozatban az EKETour projektre jellemző kifejezések fognak megjelenni, ezek magyarázata található a következő alfejezetben.

Túra alatt azokat a nagyobb eseményeket kell érteni, amelyeket az EKE minden évben megrendez. Ilyenek az emlék- és teljesítménytúrák, mint például a Jókai Mór emléktúra, amelyek egy hosszabb távot ölelnek fel, és pihenési zónák is fel vannak tüntetve az útvonalaikon, ezek **ellenőrzőpontokként** lesznek említve. Az elhaladókat a *pontbírók* – szervező, aki a túra lebonyolításával foglalkozik – várják az ellenőrzőpontoknál. Az ellenőrzőpontokat a helyszínre utaló nevek alapján azonosítják, így a részvevők könnyebben találnak rájuk. Példaként, a Jókai Mór emléktúrához kapcsolódó pontok elnevezése: Tordai-hasadék, Torockó, Székelykő stb.

Több **túraverzió**, más néven **túraváltozat** kapcsolható egy adott túrához. Ezek egymástól több szempontból is megkülönböztethetők. Típus szempontjából lehetnek gyalogos vagy kerékpáros túrák. A kiindulási és érkezési pontok szempontjából is különböznek, így a hosszúságuk is változó. A túraváltozatokra vonatkozóan is van egy elnevezési konvenció: *típus + útszakaszhossz*, például a Jókai Mór emléktúra gyalogos változatainak esetében *Gyalogos 15* vagy *Gyalogos 55* (utóbbi a leghosszabb és legnehezebb változat, mely egyben a teljes túra útvonalát lefedi).

Ahogyan arról már a bevezetőben is szó esett, önmagában a rendszerbe történő regisztráció nem elégséges ahhoz, hogy jelentkezni lehessen egy túrára. Az *alapfelhasználó* több **profilt** menedzselhet, és a jelentkezéskor ezeket regisztrálhatja a túrákra. A profilrendszer által családtagokat, munkatársakat, barátokat lehet bevezetni a rendszerbe. Amellett, hogy ez a megoldás kényelmesebb, megoldja az olyan problémákat is, mint például a kiskorúak helyzete, akik esetleg nem rendelkeznek e-mail címmel vagy telefonszámmal, illetve nem vehetnek részt kísérő nélkül a túrákon.

A **naplózás** fogalma a túrák lebonyolításánál jelenik meg. Az ellenőrzőpontoknál elhaladó résztvevőket a pontbírók feljegyzik az alkalmazás segítségével, majd a rögzített adatok el lesznek küldve a szervernek, amely ezeket az adatokat összesíti, és túrák szerint csoportosítja. Ezek az információcsomagok rögzítik a résztvevő nevét, az azonosítás időpontját és helyszínét, így a későbbiekben minden résztvevő esetében visszakövethető, hogy betartotta-e a kijelölt útvonalat.

1.2. Funkcionalitások

Az EKETour szoftverrendszer legfontosabb feladata a megrendezésre kerülő túrák egységes menedzselése. Ehhez a rendszer két kliensalkalmazást biztosít, amelyek egy központi szerverrel kommunikálnak.

1.2.1. Az Android alkalmazás funkcionalitásai

A versenyzők naplózása az Android alkalmazáson belül történik. Az alkalmazás nemzetköziesített, a felület a telefon alapértelmezett nyelvén jelenik meg, ammenyiben az illető nyelvre az applikáció talál előre lefordított szöveget. Ezt követően a rendszer szinkronizálni próbálja az adatokat a szerverrel, majd megjeleníti a soron következő túrákat. Az aktuális túra kiválasztása után a versenyzőket egy NFC-kártyával és egy sorszámmal tudja párosítani az alkalmazás (ez a verseny indítása előtt, a kiindulási pontnál történik meg). Ezt követően a pontbírók megjelölhetik, hogy melyik ellenőrzőponton tartózkodnak, majd várják a versenyzőket. Az előzőleg sikeresen regisztrált NFC-kártyával rendelkező versenyzőket az alkalmazás képes azonosítani és rögzíti elhaladásuk időpontját. Amennyiben rendelkezik hálózati hozzáféréssel, a naplózott adatokat szinkronizálja a szerveralkalmazással.

1.2.2. Webes felület funkcionalitásai

A webes felületre a rendszergazdák és a felhasználók jelentkezhetnek be és a jogosultságaik függvényében férhetnek hozzá információkhoz.

A rendszergazda feladatkörébe tartozik a túrákhoz tartozó adatok bevezetése az adatbázisba és ezek menedzselése. Létrehozhat túrákat, megadva azok nevét, általános leírását és dátumát. A túrákhoz hozzárendelhet különböző túraváltozatokat, melyeknek van egy egyedi elnevezése, egy típusa (bringa vagy gyalogos), illetve egy kezdési és egy végpontja. Az adminisztrátor regisztrációs díjat határozhat meg a változatozatoknak, amely három pénznemben (euró, lej és forint) van feltüntetve. Ellenőrzőpontokat csatolhat a túrákhoz, ezek az útvonalon elhelyezett pihenésre és a résztvevők azonosítására kijelölt területek. Mindezek mellett a rendszergazda a felhasználókat és a hozzájuk tartozó profilokat is listázhatja.

A webes felület lehetőséget nyújt a vendég felhasználóknak arra, hogy megtekinthessék a részletes leírásokat a túrákról. Az oldalon megjelenő tartalom nyelve módosítható, és ennek megfelelően az oldalon megjelenített árak pénzneme is változik. A vendég felhasználó beregisztrálhat a rendszerbe e-mail cím és telefonszám megadásával.

A bejelentkezett felhasználó profilokat hozhat létre. Ezek kitöltésekor meg kell adnia a teljes nevet, telefonszámot, póló méretet, a képviselt szervezet nevét, az étkezéssel kapcsolatos esetleges különleges elvárásokat (pl. vegetáriánus), illetve jeleznie kell, ha a regisztrált személy

rendelkezik EKE-tagsággal. A profilban feltüntetett adatok a túrákra való jelentkezéshez szükségesek. A kisebb korosztály általában nem rendelkezik e-mail címmel vagy telefonszámmal, de a profilrendszer segítségével szüleik regisztrálhatják őket a túrákra. A létrehozott profilokkal egyenként jelentkezhet a felhasználó bármelyik túrára, kiválasztva az adott profil számára legmegfelelőbb változatot.

1.2.3. Szerver oldali funkcionalitások

A szerver egy adatbázisban tárolja a teljesítménytúrák adatait és a felhasználók által létrehozott profilokat. RESTful API-n keresztül szolgálja ki a két kliensalkalmazás kéréseit, visszatéríti a lekérdezett adatokat, a megfelelő jogosultságok mellett a kért műveleteket végrehajtja, felmerülő hiba esetén megfelelő hibakódot és könnyen értelmezhető hibaüzenetet küld vissza a klienseknek.

2. Szerver oldali technológiák

Az EKETour szerveralkalmazása a Spring Java alapú nyílt forráskódú keretrendszerre épül, amely az Inversion of Control [13] (IoC) tervezési mintán alapszik. Az IoC konténer automatikusan megoldja a komponensek függőségeinek kezelését is, a Dependency Injection mintának megfelelően. Feladatköre egy adott osztályból példányosított objektum létrehozása, életciklusának meghatározása, fenntartása, a szükséges függőségek beinjektálása. Mindezt futási időben teszi meg.

2.1. Spring Boot és konfiguráció

A Spring Boot[12] egy gyors és egyszerű módszer egy Spring alapú alkalmazás létrehozására. Adott projekttípusokra jellemző, előre elkészített konfigurációkat biztosít, és a build eszköz számára meghatározott függőségeket leíró állomány (pl. build.gradle) alapján kapcsolja be a szükséges modulokat az alkalmazásba.

A Spring keretrendszer esetében a legtöbb konfigurációs műveletet Java osztályok segítségével is el lehet végezni, így nem kell feltétlenül XML állományok létrehozásával és menedzselésével foglalkozni. Egyszerűbb szerkezetű állományokat is támogat a keretrendszer, például property fájlban lehet konfigurálni az adatbázishozzáféréssel, karakter kódolással és egyéb általános beállításokkal kapcsolatos paramétereket.

2.2. Spring Data JPA

A Spring Data[15] lehetővé teszi a DAO implementációk megvalósításának a kiküszöbölését. A modellek perzisztálásához egy előre definiált interfészt kell kiterjeszteni, amelynek alapján a rendszer automatikusan biztosítja a CRUD műveleteket. A keretrendszer bonyolultabb lekérdezések megvalósítására is lehetőséget ad: az interfészekben deklarált metódusok neve, paraméterei és visszatérési értéke alapján tudja generálni ezeket. Összetettebb lekérdezések esetében a metóduson elhelyezett annotáció segítségével a konkrét JPQL utasításokat is meg lehet adni.

2.3. Spring MVC Rest

Az EKETour szerveralkalmazás REST kéréseken keresztül szolgálja ki a klienseit. Ebben a Spring MVC segít, ami a projekt esetében JSON formátumban közvetíti az adatokat.

2.4. Spring Security

A felhasználók azonosítására és a kérések hitelesítésére az EKETour a Spring Security modult használja fel. Mintaillesztés alapján dönti el a rendszer, hogy adott kérésekhez milyen jogosultságra van szükség. Ezeknek a beállítása egy Java osztályból is megvalósítható. A Spring Security a jelszavak hash-elésére is biztosít előre implementált módszereket.

2.5. Arquillian

Az EKETour projekt esetében megfogalmazott funkcionalitások minőségének biztosításában fontos szerepet játszanak az előre definiált tesztesetek.

A metódusok és komponensek tesztelése mellett, amelyet a JUnit keretrendszer biztosít, az alkalmazásban létrehozott szolgáltatások helyes működésének vizsgálata is fontos szempont volt. Az automatizált integrációs tesztek az Arquillian keretrendszer segítségével vannak megvalósítva. Ez egy konténer-független technológia, alkalmazható akár Java SE alkalmazások esetében is, de ugyanígy támogatja az OSGI alapú, illetve a servlet konténerekbe (Tomcat , Jetty stb.), vagy Java EE alkalmazásszerverekre (JBoss, Glassfish stb.) kitelepített alkalmazások tesztelését is. Beágyazott (embedded), távolsági (remote), vagy a felhasználó által menedzselt konténereket is támogat.

Az EKETour által használt JBoss alkalmazásszerver esetében, a teszteket megfogalmazó osztályok a *@RunWith(Arquillian.class)* annotációval, míg a tesztek a *@Test* annotációval vannak megjelölve. Az osztályban megjelenik egy *@Deployment()*-tel felannotált metódus, amely a teszt projekt összeállításáért felel. Megadható a végtermék típusa (.jar, .war vagy .ear), illetve meghatározhatóak azok az osztályok, amelyeket telepíteni kell a teszt futtatásához. A konfigurált csomagot a ShrinkWrap modul kitelepíti a kijelölt szerverre.

A szolgáltatások működését az *Assert* osztály metódusai tesztelik: a visszatérített érték és az elvárt érték közötti összefüggés alapján ellenőrizhető egy adott metódus helyes működése. A keretrendszer inicializálja és hozzáférhetővé teszi az integrációs tesztekhez szükséges erőforrásokat.

3. A webes alkalmazás technológiái

Az EKETour szoftverrendszer két kliensalkalmazással rendelkezik: egy webes felületet biztosít adminisztációs és publikus felhasználásra, illetve egy Android alkalmazást a szervezők számára.

3.1. Angular

A webes felhasználói felület az Angular 4 nyílt forráskódú frontend technológiára épül. A 2017 márciusában megjelent keretrendszer architektúráját a hierarchikusan elrendeződő komponens struktúra jellemzi leginkább.

Legfőbb előnyei a modularitás, a dinamikus betöltés, az aszinkron sablon-összeállítás (template compilation) és a reaktív programozás támogatása.

Az Angular 4 [10] a korábbi verziójához hasonlóan támogatja az MVC tervezési mintát, viszont más lehetőségeket is biztosít architekturális szempontból, ilyen a *reaktív programozás*, az *egyirányú adatfolyam* és *állapot-centralizált menedzsment*.

A webalkalmazás esetében többször volt alkalmazva például az aszinkron események kezelésére az RxJS-ben implementált *Observable* minta adatfolyam alapú megvalósítása. Egy egyirányú adatfolyam (unidirectional data flow) alapján, a nézeten bekövetkező események (*action*) új állapotokhoz (*state*) vezetnek, amelyeknek megfelelően változik a nézet.

3.2. TypeScript

A TypeScript[11] a Microsoft által fejlesztett programozási nyelv, az *ECMAScript 6* [6] egy bővített változata, amely a JavaScript által biztosított eseményvezérelt (event-driven), funkcionális, illetve prototípus alapú programozási stílusok mellett az osztály alapú, objektumorientált programozási paradigmát is támogatja. A TypeScript fordító a forráskód lefordítása (transpile) által minden böngésző számára értelmezhető Javascript kódot generál.

4. Az Android alkalmazás technológiái

4.1. Retrofit

A Retrofit [9] egy Java alapú REST kliens, amit az EKETour Android alkalmazás is használ. Viszonylag egyszerűvé teszi a strukturált adatok le- és feltöltését. A Retrofit-et be lehet állítani, hogy az adatokat, típusuktól függően, hogyan szerializálja és deszerializálja. Az EKETour szerveralkalmazás JSON formátumban szolgálja ki a kliensek REST hívásait. Ebben az esetben a Retrofit GSon szerializációját használja az Android alkalmazás. Egy interfészen belül a metódusokat annotálnunk kell a kérés típusának megfelelően. A Retrofit a metódus visszatérítési értéke és az esetlegesen ugyancsak annotált paraméterek alapján tudja kezelni a kérést.

4.2. OrmLite

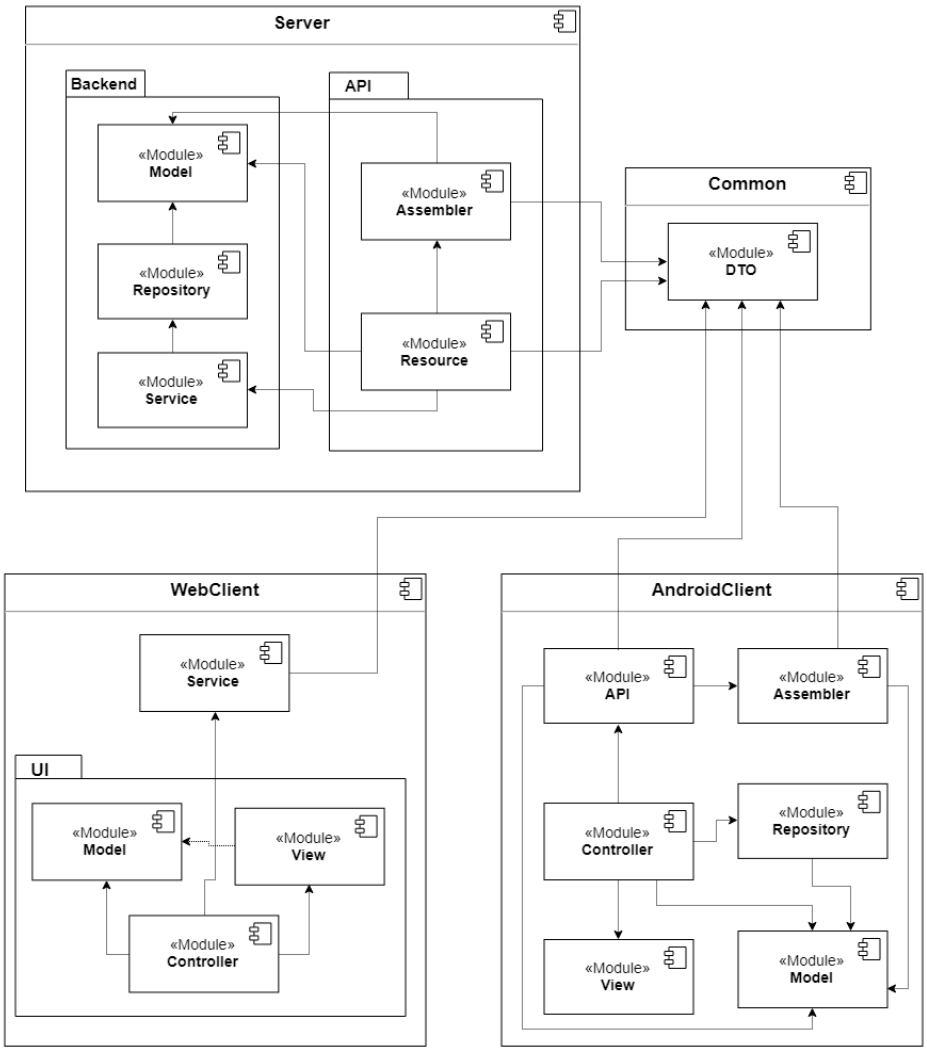
Az Object Relational Mapping Lite (ORM Lite) [17] keretrendszer néhány egyszerű, alapvető funkciót biztosít a Java objektumok relációs adatbázisba való mentésére, lekérésére és törlésére. Az ORM Lite támogatja az Android natív SQLite adatbázisát is. Az objektumoknak az attribútumait @DatabaseField-el kell annotálni. A modellek opcionálisan @DatabaseTable annotációval láthatóak el. Ezek alapján az ORMLite létrehozza az alkalmazás SQL tábláit a megfelelő mezőkkel és leegyszerűsíti a perzisztenciával kapcsolatos műveletek elvégzését.

4.3. NFC

A Near Field Communication (NFC) [1] rövid hatótávolságú vezeték nélküli kommunikációs technológiák egy csoportja, amelyek jellemzően 4 cm, vagy annál kisebb távolságot igényelnek a kapcsolat létrehozásához. Az EKETour esetében lehetővé teszi az adatok megosztását az NFC-kártya és a mobileszköz között.

5. Az EKETour projekt megvalósításának fontosabb lépései

5.1. Architektúra



1. ábra. Az EKETour szoftverrendszer architektúrája.

Architektúra szempontjából a rendszer három fő komponensre tagolható: szerver, Android kliens és webes kliens. Az 1. ábrán megjelenik egy negyedik komponens is, a *Common* modul, amelyben a *Data Transfer Object* tervezési minta megvalósítása található. Ezáltal valósul meg a komponensek közötti kommunikáció, biztosítva az adatok egységes alakját, és szükség esetén bizonyos részinformációk elrejtését.

A szerver komponens két alkomponensre tagolódik: a backend és az API (*Application Programming Interface*). A backend-ben található a *Modell*, amely az alkalmazásban meghatározott entitások gyűjteménye. A *Repository* felelős az adatbázissal való kapcsolattartásért. A rendszer a *MySql* relációs adatbázissal való kommunikációra egy *Java Persistence API (JPA)* implementációt használ. A *Service* a rendszer üzleti logika (*Buisness logic*) rétege, amely az

adathozzáférési réteggel kommunikálva szolgálja ki a kéréseket.

Az API részét képezik az *Assembler*-ek, amelyek átalakítóként működnek a modellek és a megfelelő DTO-k között. A *Controller* feladata a kliensektől érkező kérések továbbítása a szolgáltatási rétegnek. A kapott válaszokat az Assembler-ek DTO-kba alakítják, majd ezek a DTO-k lesznek szerializálva és továbbítva a klienseknek.

A webes kliens esetében a Service komponens felelős a szerverrel való kommunikációért. A Controller vezérli a felületet, illetve ő veheti igénybe a Service komponens szolgáltatásait.

Az *AndroidClient* komponens az MVC tervezési minta alapján van felépítve. A *Model* komponens az entitásokat tartalmazza, a *Controller* dolgozza fel a UI felületen történő eseményeket. Az API által a kérések továbbítva lesznek a szervernek. Itt is megjelenik a Repository modul, amely az SQLite lokális adatbázissal kommunikál. A rendszer ezáltal támogatja a offline működést.

5.2. Szerver oldali megvalósítások

5.2.1. Adatmodell

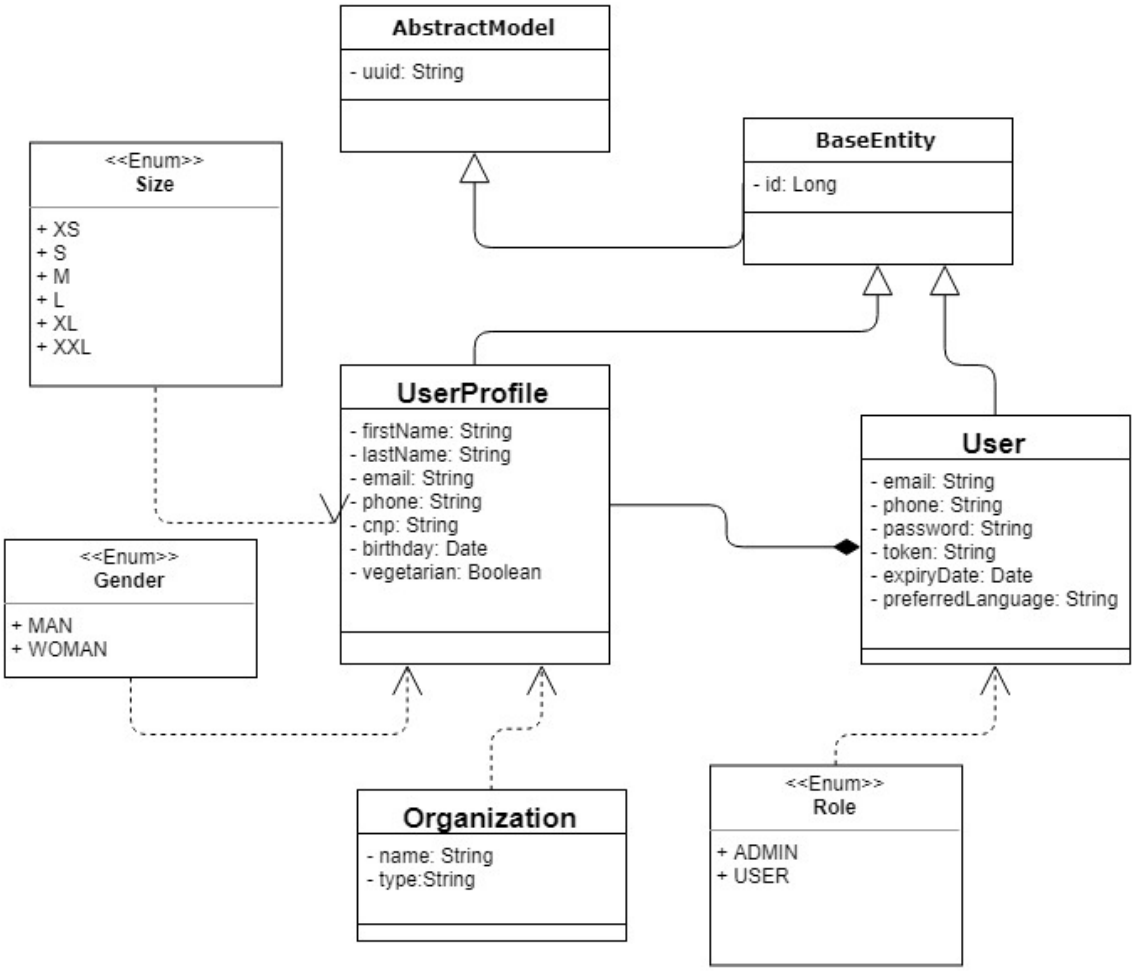
Az EKETour rendszer entitásai *Java Bean*-ként vannak reprezentálva, megfelelő Java Persistence API annotációkkal ellátva. Az entitások privát adattagokkal rendelkeznek, melyeket publikus (*getter* és *setter*) metódusokkal lehet elérni, illetve egy alapértelmezett paraméter nélküli konstruktorral. Ezek a Java Bean osztályok az *edu.codespring.eketour.server.backend.model* csomagban vannak elhelyezve, ősosztályaik az AbstractModel, illetve a BaseEntity osztályok.

Az *AbstractModel* egy absztrakt osztály, amely a Serializable interfészt implementálja és egy karakterlánc típusú *uuid* elnevezesű attribútummal rendelkezik. Ez a rendszer által generált 32 hexadecimális számjegyből álló karaktersor, egyedi értékének köszönhetően általa megkülönböztethetők a rendszerben megjelenő objektumok.

A *BaseEntity* osztály egy Long típusú id mezővel bővíti ki az előbb említett osztályt, amely egyedi azonosítóként működik az azonos típusú példányok között. Adatbázis szinten az elsődleges kulcsnak felel meg.

A rendszer működéséhez hozzájáruló főbb entitások a User, UserProfile, Trip, TripVersion, valamint a Checkpoint. A **User** adatmodell a regisztrált fehasználókat reprezentálja. Ez egy *prefferedLanguage* mezőt is tartalmaz, amelyben eltárolódik a felhasználó által a webes felületen legutoljára kiválasztott nyelv. Emellett a *token* és *expiryDate* páros lehetővé teszi az elfelejtett jelszó kicserélését, melynek ellenőrzése mintaillesztéssel történik. A **UserProfile** entitás reprezentálja a profilokat, amelyek esetében megadható, hogy az illető milyen szervezet tagja (*OrganizationType*), illetve meghatározható az esetlegesen igényelt póló mérete (*Size*) és típusa (*Gender*) is. Ezek mellett a túrák és a felhasználók esetében is megjelenik a típus fogal-

ma (*TripType* és *UserType*). Az említett mezők előredefiniált Enum értékek által határozhatóak meg.



2. ábra. Felhasználó és „felhasználóprofilok” származtatása, az attribútumaik és a köztük lévő *egy-a-többhöz* kapcsolat

5.2.2. Adathozzáférési réteg

Az adathozzáférési réteg feladata az adatbázissal való kommunikáció, elvégzi a projekt és az adatbázis közötti adatátvitelt. A Spring Data lehetőséget nyújt többfajta adattároló használatára. A projekten belül a keretrendszer Spring Data JPA [15] modulja van alkalmazva, amely a Hibernate ORM (Object Relational Mapping) keretrendszer feletti absztrakciós szintet képez. A projekt esetében az adathozzáférési réteget a *edu.codespring.eketour.server.backend.repository* csomagban elhelyezett, CRUDRepository-t kiterjesztő interfészek alkotják.

Ahogy az a példa kódrészletben is látszik, az összetettebb lekérdezések is tömören megvalósíthatók. Lehetőség van a **@Query** annotáció használatára, melyben egy megfelelő JPQL utasítást lehet megadni, ahogyan az 5. sorban is látható. A lekérdezésbe illesztendő értékek a hozzá-

```
public interface TripRegistrationRepository extends
                          CrudRepository<TripRegistration, Long> {

    List<TripRegistration> findAllByModifiedDateGreaterThan(Date
            modifiedDate);

    @Query("SELECT t FROM TripRegistration t where userProfile =
                      :userProfile and tripVersion = :tripVersion")
    TripRegistration findByUserProfileAndTripVersion
                    (@Param("userProfile") UserProfile userProfile,
                      @Param("tripVersion") TripVersion tripVersion);

}
```

1. kódrészlet. A TripRegistration entitáshoz tartozó adathozzáférési interfész

tartozó metódus paraméterei lesznek, melyek sorra fel vannak annotálva a **@Param**(„elnevezés”)-el, melynek kötelező attribútuma a query-ben helyet foglaló változó neve.

5.2.3. Kommunikáció

Az EKETour projekten belül a szerverrel való kommunikáció a REST szoftverarchitektúrán alapszik, RESTful szolgáltatásokon keresztül valósul meg. A szerver által biztosított erőforrásokat a kliensalkalmazások közvetett módon elérhetik vagy módosíthatják, annak függvényében, hogy milyen HTTP művelet (POST, GET, PUT, DELETE) és egységes erőforrás-azonosító (URI) van meghatározva a kéréskor. A kliens és szerver között az adatok JSON objektumok formájában vannak továbbítva.

Az EKETour szerver modul *edu.codespring.eketour.server.api.resources* csomagjában vannak a kéréseket feldolgozó osztályok, amelyek a *@RestController* annotációval vannak megjelölve. Az erőforráskezelő osztályban a szükséges service-ek és assembler-ek az *@Autowired*-del vannak annotálva, amely a komponensek közötti függőségeket jelzi.

```
@RequestMapping(method = RequestMethod.GET,
                value = "/api/tripByTripVersion/{tripVersionId}")
public TripDTO findTripByTripVersion
                (@PathVariable("tripVersionId")Long tripVersionId) {

   TripVersion tripVersion = tripVersionService.findOne(tripVersionId);
   return tripAssembler.createDto(tripVersion.getMainTrip());

}
```

2. kódrészlet. A túra keresése egy változat szerint

5.2.4. Biztonság

Az alkalmazás biztonságáért a Spring Security keretrendszer felel, az URL-ek jogkörök szerinti elérhetőségének korlátozása által. Például, azok az elérési utak, amelyek az *admin/* előtaggal kezdődnek, csak rendszergazda jogosultságú felhasználók számára hozzáférhetőek. Ezt a *SecurityConfig*-ot kiterjesztő osztály biztosítja, a *configure* metódus felülírása által.

A jelszavak titkosítására *BCryptPasswordEncoder*-t alkalmaz a rendszer. Egy új felhasználó létrehozásakor, ha minden adat helyesen volt feltüntetve, az adatbázisba való beszúrás előtt hash-eli a megadott jelszót.

5.3. Webes felület megvalósítása

5.3.1. Kommunikáció a szerverrel

A szerverrel való kommunikációhoz szükség volt egy RESTful web alkalmazás felépítésére, az Angular keretrendszer segítségével. A *HttpClient* [8] technológia által valósult meg a kapcsolat a szerverrel.

Egy *ApiService* osztályban generikusan vannak implementálva az általános metódusok, a *http* property igénybevételével, és ezen a szinten van megoldva az érkező JSON formátumú válaszok objektumokba történő átalakítása is.

```
import {Observable} from 'rxjs/Observable';

@Injectable()
export class UserDataService {

  constructor(private api: ApiService) {}

  getUserById(userId: number): Observable<User> {
    return this.api.getById(User, 'api/users', userId);
  }

}
```

3. kódrészlet. A User adatmodellel kapcsoltos műveleteket tartalmazó osztály

Az *Observable* objektum lehetőséget nyújt az üzenetek továbbítására a feladó és a feliratkozók (subscribers) között.

5.3.2. Nemzetköziesítés

A webes alkalmazás felhasználóinak lehetőségükben áll a felületen megjelenő szöveg nyelvének megváltoztatása. Ezt az Angular által támogatott ngx-translate könyvtár biztosítja. Az

ebben található *http-loader* tölti be a fordításokat tartalmazó fájlokat. Tekintve a következő példát *assets/i18n/hu.json*, a fájlokra vonatkozó konvenció a szigorú mappa-struktúra, amelyben az elhelyezett fájlok egy-egy nyelvet reprezentálnak. A JSON kiterjesztésű fájlok *kulcs-érték* párokat tartalmaznak.

```
<h3> {{'trips.NAME' | translate}}: {{trip.name}} </h3>
```

4. kódrészlet. Pipe mechanizmus használata

A 4. HTML kódrészletben megfigyelhető a kulcsra alkalmazott *translate pipe* („csővezeték") mehanizmus használata, amely futási időben cseréli a megjelenítendő szöveget, a weboldalon beállított nyelv függvényében.

5.4. Android alkalmazás megvalósítása

5.4.1. Adatmodell

Az EKETour Android alkalmazás entitásai *Java Bean*-ként vannak reprezentálva, a megfelelő OrmLite annotációkkal ellátva, mind felépítésük, mind hierarchiájuk szempontjából a szerver-oldali modellhez hasonló módon.

5.4.2. Kommunikáció a szerverrel

Az EKETour telefonos alkalmazásban a szerverrel való kommunikáció a RESTful szolgáltatásokon keresztül valósul meg, a Retrofit csomag segítségével.

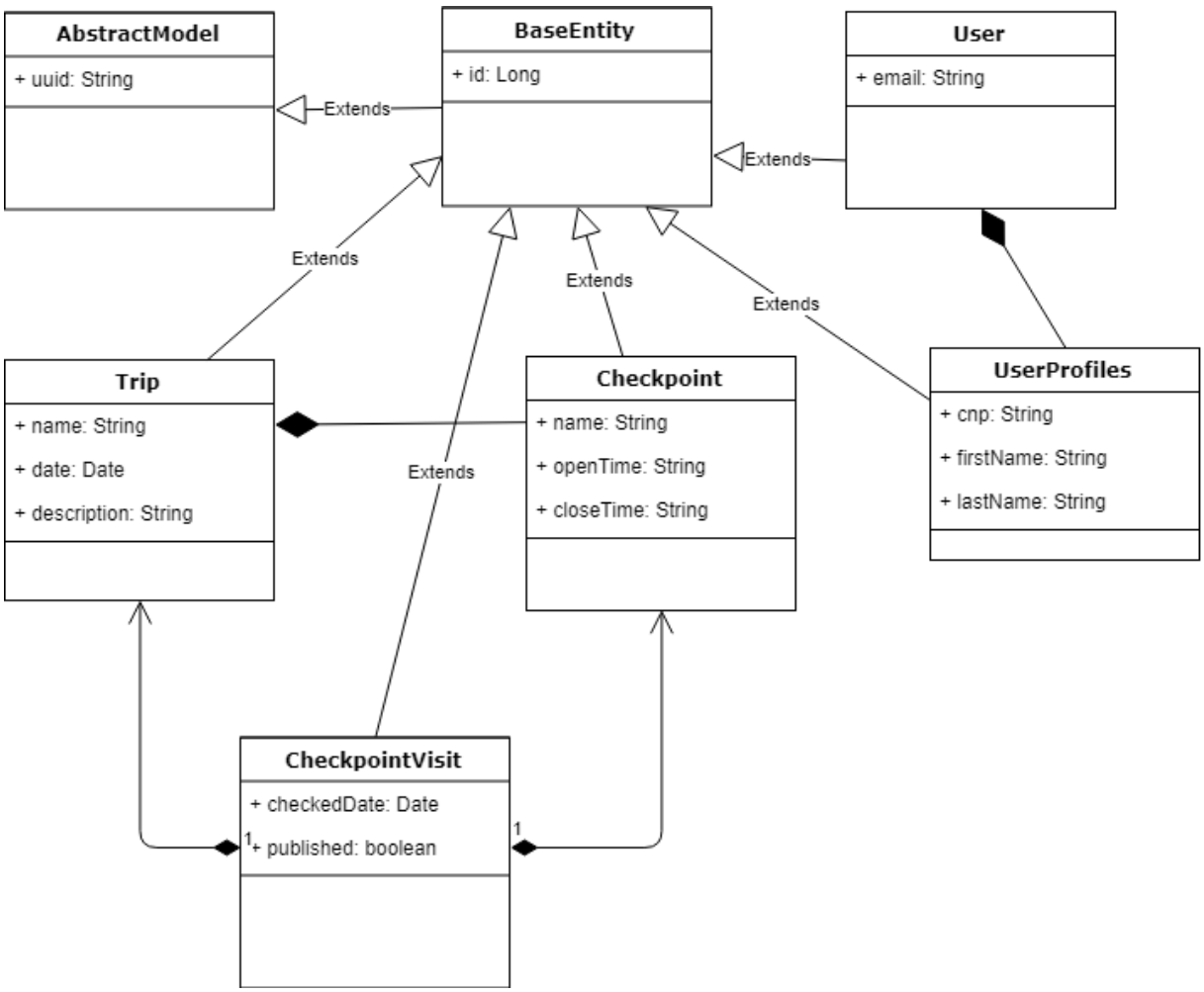
```
OkHttpClient okHttpClient = new OkHttpClient.Builder()
        .addInterceptor(httpLoggingInterceptor).build();

GsonBuilder gsonBuilder = new GsonBuilder();
gsonBuilder.registerTypeAdapter(Date.class, new DateDeserializer())
        .registerTypeAdapter(Date.class, new DateSerializer());

GsonConverterFactory factory = GsonConverterFactory
                                .create(gsonBuilder.create());

return new Retrofit.Builder().baseUrl(BASE_URL)
                .addConverterFactory(factory).client(okHttpClient).build();
```

5. kódrészlet. Retrofit objektum felépítése



3. ábra. Android alkalmazás adatmodellje. A *CheckpoitVisit* attribútumai a naplózás dátuma és egy állapot jelző flag (szinkronizáció sikeressége)

5.4.3. Szinkronizáció

A hálózati hozzáférést igénylő műveletek esetében az adatok szinkronizálódnak: a szervertől lekért adatok a telefon adatbázisába is el lesznek mentve. Az adatmodellek tartalmaznak egy a módosítás időpontját jelző mezőt, így az alkalmazás a szervertől csak az utolsó módosítás utáni adatokat kéri el.

5.4.4. NFC-kártya olvasása

Az android.nfc csomagban található NfcAdapter osztály segítségével kapcsolatot létesíthetünk az NFC szenzorral.

```
@Override
protected void onResume() {
    super.onResume();
    PendingIntent pendingIntent = PendingIntent
                    .getActivity(this, 0, new Intent(this, getClass())
                        .addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);
    IntentFilter filter = new IntentFilter();
    filter.addAction(NfcAdapter.ACTION_TAG_DISCOVERED);
    filter.addAction(NfcAdapter.ACTION_NDEF_DISCOVERED);
    filter.addAction(NfcAdapter.ACTION_TECH_DISCOVERED);

    NfcAdapter nfcAdapter = NfcAdapter.getDefaultAdapter(this);
    if (nfcAdapter != null) {
        nfcAdapter.enableForegroundDispatch(this, pendingIntent,
                                new IntentFilter[]{filter}, this.techList);
    }
}
```

6. kódrészlet. NFC-kártya kapcsolat létesítése

```
@Override
protected void onNewIntent(Intent intent) {
    if (Objects.equals(intent.getAction(), NfcAdapter
                                                .ACTION_TAG_DISCOVERED)) {
        nfcId = ByteArrayToHexString(intent
                                .getByteArrayExtra(NfcAdapter.EXTRA_ID));
        ...
    }
}
```

7. kódrészlet. NFC-kártya olvasása

6. Az EKETour működése

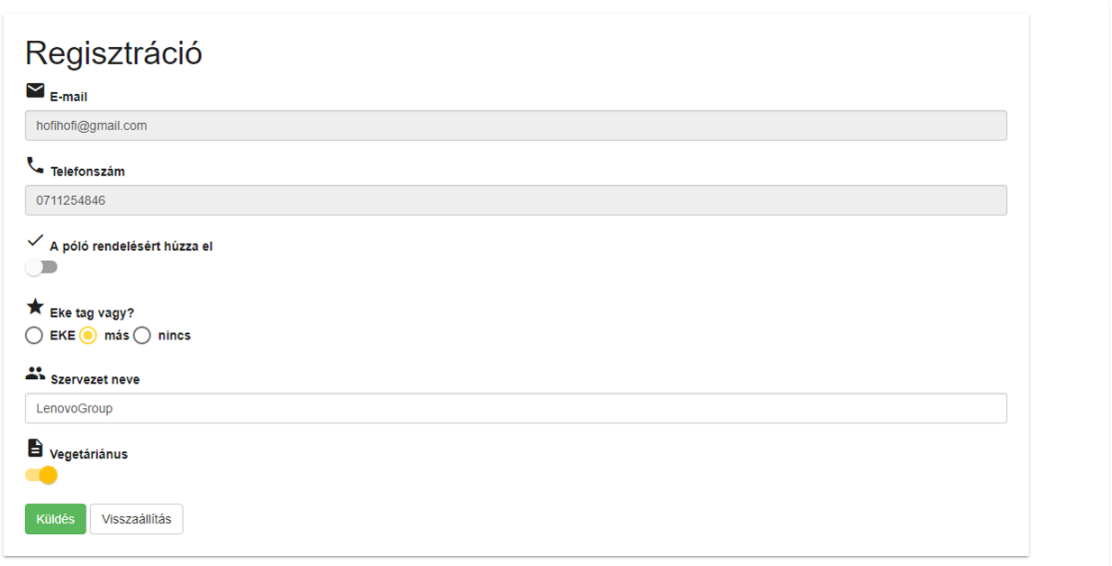
A következő fejezet részletesebben bemutatja az EKETour alkalmazás fontosabb funkcionalitásait. Az alkalmazás többnyelvűsítése az ábrákon is látható, ellenben az egyes műveletekre való hivatkozás a magyar kifejezések alapján történik majd.

6.1. A webes felhasználói felület használata

A webes felület megnyitásakor a felület alapértelmezett nyelve a magyar, de a felhasználónak lehetősége van ezt módosítani a fejlécben található zászlóra kattintva, amint a 5. ábrán is látható. Ez a művelet a használat során bármikor elérhető.

A menüben megjelenik az a három funkcionalitás, amely egy vendégfelhasználó számára is elérhető. Meg lehet tekinteni a túrákat és a hozzájuk tartozó fontosabb információkat: leírás, dátum, a túra változatai, a változatok regisztrációs díjai, illetve az elindulás és érkezés helyszíne.

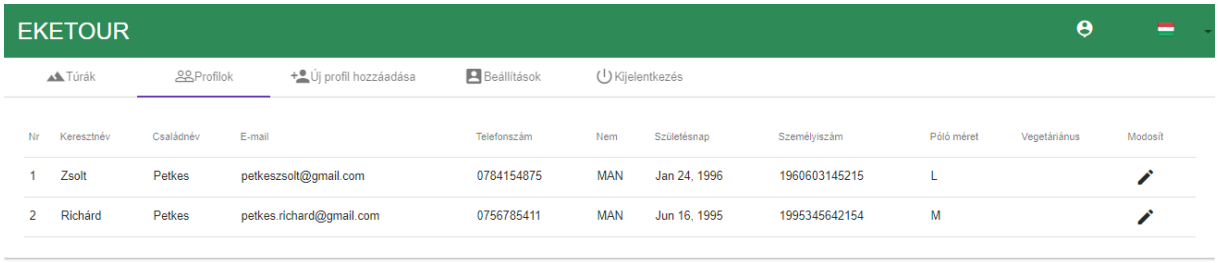
A másik két művelet, ami rendelkezésére áll, az a regisztráció, illetve a bejelentkezés. Regisztrációkor szükséges az e-mail cím és a telefonszám megadása. Sikeres regisztráció esetén a vendég egy üdvözlő e-mailt kap a rendszertől. Belépéskor a helyes e-mail és jelszó pár megadása után jelenik meg a felhasználói felület.



4. ábra. Túrára való regisztráció

A bejelentkezett felhasználó, az *Új profil hozzáadása* menüpontot kiválasztva, létrehozhatja az első profilját, amelyet a saját adataival kitölthet. A feltüntetett adatok között megjelennek olyan plusz információk, mint a póló méret és az étkezésekkel kapcsolatos elvárások (vegetáriánus-e). Ezek alapján a túrákra történő jelentkezéskor a rendszer automatikusan kitölti az adatlapok vonatkozó részeit, de a felhasználó még itt is módosíthatja az adatokat. Több profilt is felvezethet, például megadhatja családtagok, barátok, munkatársak adatait. Ezeket később

megtekintheti, illetve a hibás vagy megváltozott adatokat módosíthatja.

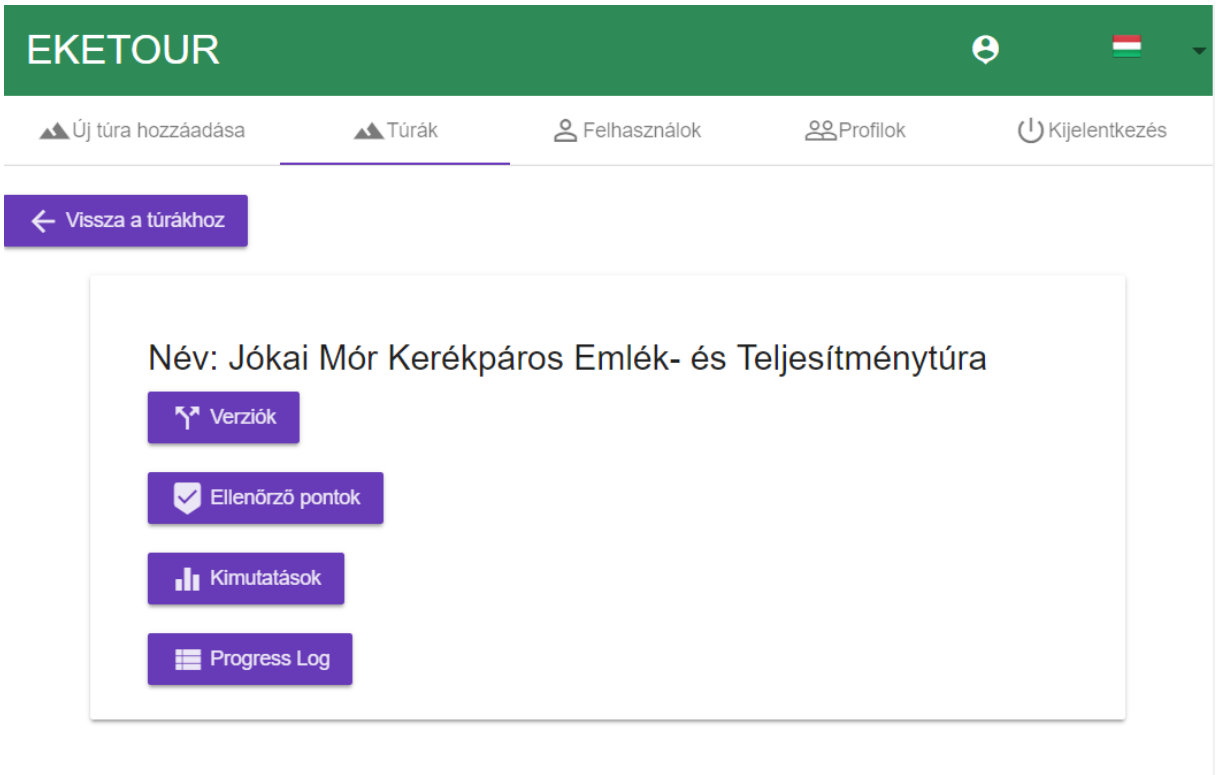


5. ábra. Felhasználóhoz tartozó profilok megtekintése

A *Beállítások* menü alatt lehetőség van az alapfelhasználóhoz tartozó telefonszám és jelszó cseréjére.

A *Túrák*ra kattintva a bejelentkezett felhasználónak lehetősége van kiválasztani a számára legmegfelelőbb túrát, majd az ahhoz tartozó túraváltozatok közül is választ. Végül a profil kiválasztását követően véglegesítheti a regisztrációt, ezt megelőzően adott az adatok módosításának lehetősége is (lásd 4. ábra). Így az alapfelhasználóhoz tartozó profilokkal megkötés nélkül lehet jelentkezni, akár eltérő túraváltozatokra is.

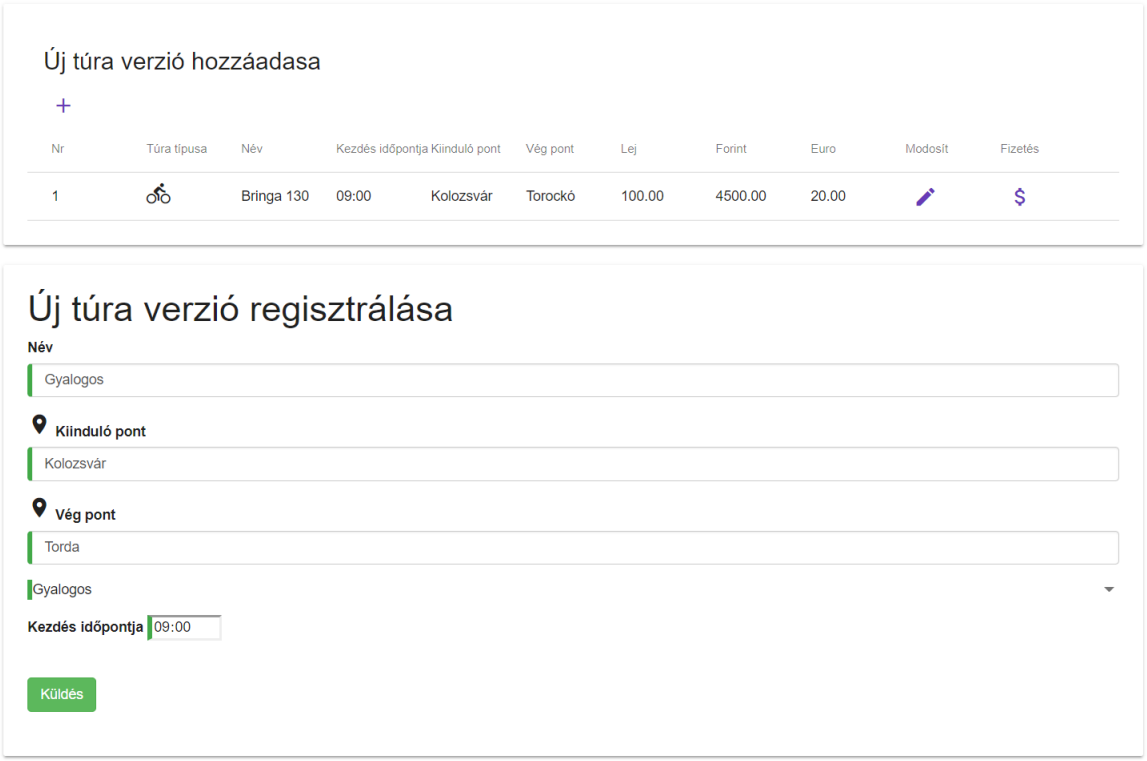
Az adminisztrátor jogosultsággal rendelkező felhasználónak további funkcionalitásokat nyújt a felület. Számára kilistázhatóak a regisztrált felhasználók, illetve a felvezetett profilok.



6. ábra. Túrához kapcsolódó menüpontok

Az adminisztrátor végzi a túrák menedzselését. Az *Új túra hozzáadása* menüpontra kat-

tintva megkezdheti a túrák bevezetését, illetve a már létrehozottakat megtekintheti a *Túrák*-ra kattintva. Ezután a túrákhoz kapcsolódó adatok meghatározása következik. A táblázatból kiválasztva a megfelelő túrát a 6. ábrán látható lehetőségek fogadják az adminisztrátort. A *Verziók*-at kiválasztva új túraváltozatokat hozhat létre, megadva az elnevezést, kiinduló- és végpontot, a túra típusát (bringás vagy gyalogos) és időpontját. Ezt követően a frissen felvezetett túraváltozat megjelenik az elérhető túraverziók között, amint a 7. ábrán is látható. Regisztrációs díj hozzárendeléséhez a túraváltozattal megeggyező sorban levő *dollárjel ikonra* kell kattintani. A regisztrációs díj értékét három pénznemben kell feltüntetni (lejben, forintban és euróban), lévén, hogy külföldi résztvevői is lehetnek az emléktúráknak.



7. ábra. Túraverzió létrehozása

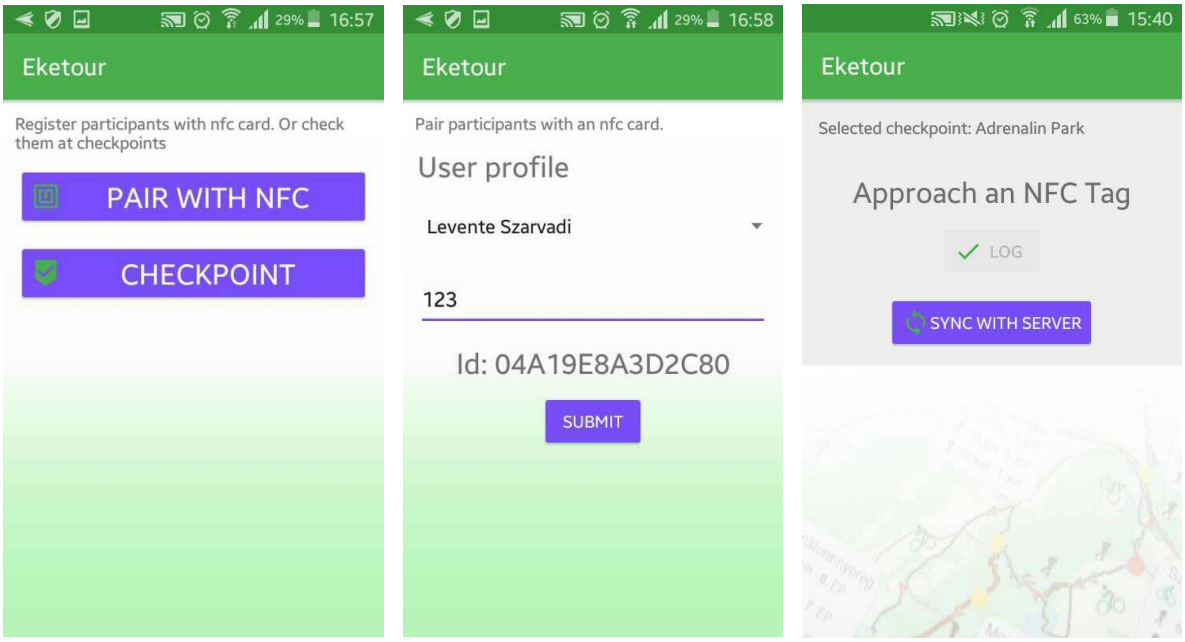
Visszatérve az előző oldalra (6. ábra) kiválaszthatóak az *Ellenőrzőpontok*. Új ellenőrzőpont létrehozásakor egy helymeghatározó elnevezést és egy „nyitvatartási” időintervallomot – meddig tartózkodik a helyszínen a pontbíró– kell meghatározni. A létrehozást követően az új helyszín megjelenik az ellenőrzőpontok listájában.

A *Kimutatások* gomb egy kimutatás felületre navigál. Itt az adminisztrátor kilistázhatja például, hogy egy adott túrán kik azok a résztvevők, akik pólót igényeltek a regisztráláskor. A *Progress Log* a túra napján válik hasznossá, mivel itt lehet majd követni, hogy ki melyik ellenőrzőpontnál haladt el. A résztvevők neve szerint szűrni is lehet az adatokat.

6.2. Az Android kliensalkalmazás használata

Az Android alkalmazás a szervezők és pontbírók számára lett kifejlesztve. Használatához az okostelefonnak támogatnia kell az NFC-kártya olvasását és írását.

A pontbíró az alkalmazást megnyitva automatikusan szinkronizálja a számára szükséges adatokat: a megrendezésre kerülő túrákat és a résztvevők listáját. A szinkronizálást manuálisan is kérheti, ha a menü ikont megérintve a megjelenő listából a *Sync* opciót kiválasztja. Ezt követően a túrák listájából kiválaszthatja azt a túrát, amelyiken jelenleg szervezői szerepet tölt be. A megjelenő felületen két lehetőség áll rendelkezésre, ahogy a 8a. ábra is szemlélteti.



(a) Túra kiválasztása után (b) NFC párosítása résztvevővel (c) NFC azonosítás

8. ábra. Az EKETour Android alkalmazás.

Az első, *Pair with NFC* opciót akkor választja ki a szervező (8b. ábra), amikor ő egy kiindulópontban tartózkodik. Ekkor a feladata az, hogy az érkező résztvevőket regisztrálja az adott túrára. A szervező kikeresi a listából a résztvevő nevét, az előzőleg hozzárendelt sorszámát beírja az üres mezőbe, majd az NFC-kártyát hozzáérintve a telefonhoz, párosítja azt a résztvevővel. A *Submit* gombra kattintva a sikeres párosítás(oka)t elküldi a szervernek.

A köztes ellenőrzőpontoknál várakozó szervezők feladata az elhaladó résztvevők azonosítása. Erre ad lehetőséget a 8c. ábrán látható második opció, a *Checkpoint* gomb, amely a túra útvonalán elhelyezkedő összes ellenőrzőpontot megjeleníti. A szervező innen kiválasztja az aktuális tartózkodási helyét, majd megkeződhet az elhaladó túrázók naplózása. A kártya hozzáérintésével megjelenik a képernyőn a kártya tulajdonosának a neve és a regisztrációkor kapott sorszám. A művelet érvényesítése a *Log* gombra kattintva történik meg. Ha van internetkapcsolat, akkor a *Sync with server* gomb érintésével a lokálisan mentett naplózási adatok elküldhetőek a szervernek.

7. Eszközök és módszerek

Az EKETour projekt fejlesztése Scrum [14] módszer alapján valósult meg, közismert eszközök alkalmazásával.

A Scrum agilis szoftverfejlesztési stratégia, iteratív és inkrementáló tulajdonsággal rendelkezik. A fejlesztés két hetes sprintekre tagolódott a nyári gyakorlat folyamán, napi rendszerességű megbeszélésekkel (Scrum meeting). A csoportos projekt tantárgy esetében három hetes sprinteket alkalmazott a csapat, heti megbeszélésekkel. Egy iteráció a tervezési fázissal kezdődött (planning), ekkor döntötte el a csapat, hogy a backlog-ban előre meghatározott feladatok — úgynevezett *user story-k* -– közül, melyek kerüljenek át a *sprint backlog*-ba. Ezek voltak ezután megvalósítva a következő iterációban. A sprint lezárásakor az elfogadási feltételeknek megfelelő user story-k voltak bemutatva egy *demó* keretein belül. A sprint lezajlásának kiértékelése a *retrospektív* („visszatekintés") gyűlések keretein belül történt.

Elsődleges cél volt a hatékony csapatmunka, ebben segített a Git[3] verziókövető rendszer is. Ez egy osztott forráskód-kezelő rendszer, segítségével a projekt korábbi verziói is elérhetőek és szükség esetén visszaállíthatóak. Minden funkcionalitás esetében egy új fejlesztési ág volt létrehozva, így csökkenthető az esetleges konfliktusok száma a fejlesztés során. A befejezettnek tekintett ágak változtatásai egy átvizsgálást követően kerültek be a stabil fejlesztési ágba.

A fejlesztés során szükség volt egy rendszerre, amelynek segítségével megvalósulhatott a feladatok nyilvántartása, prioritizálása, követése. A *GitLab* szolgált projektmenedzsment eszközként, biztosítva egy Kanban Board-ot, amelyen nyomon követhetővé vált a projekt fejlődése, a user story-k állapota, naplózhatóvá vált, hogy ki mennyi időt szánt egy adott feladatra. Emellett a GitLab repository menedzsment rendszerként is szolgált, megoldva a forráskód tárolását, illetve a biztosítja a *pipeline* rendszert, amely által a folytonos integráció (*Continuous Integration*) folyamata is megvalósításra került.

Egy *pipeline* különböző munkák (*job*) sokasága, amelyeket szakaszokba (*stage*) lehet sorolni. Ezeket a munkákat a Gitlab által biztosított YAML fájlban kell definiálni, különböző beállítások eszközölhetőek a job-ok esetén. Kötelező paramétere a *script* amelyben a lefutásra kerülő parancsokat tartalmazza, emellett megadható a stage neve, amelyhez majd tartozni fog, ez opcionális, viszont követhetőbbé teszi a lefutott pipeline kimenetelét. Az EKETour esetén a *.gitlab.ci.yml* -ben definiált munkák a kódellemzést, a modulok külön felépítését és a tesztek lefuttatását fogalmazzák meg.

A fejlesztők folyamatosan arra törekedtek, hogy a módosításokat követően az alkalmazás konzisztens állapotban maradjon: a forráskód szintaktikailag és szemantikailag helyes, illetve fordítható legyen, az alkalmazás futtatható legyen, és a régi funkcionalitásai is helyesen működjenek. Ezzel kapcsolatos problémák elsősorban akkor merülhetnek fel nagyobb valószí-

nűséggel, amikor különböző módosításokat tartalmazó ágak kerülnek összefésülésre (*merge*). Amint felkerül valamilyen változás a központi *repository*-ba (tárolóba), a CI rendszer első lépésként felépíti (*build*) az alkalmazást, beleértve az automatizált tesztek lefuttatását. Abban az esetben, ha sikertelen a build, vagy valamelyik teszt, a rendszer azonnal értesíti a fejlesztőket és visszakereshetőek az érintett részek, megtalálható a hibát kiváltó rész és javítható.

A projekt fejlesztése során a Java forráskód ellenőrzésére két statikus kódelemző is használva volt: a *FindBugs*[7], amely a lehetséges futás közbeni hibákat (bug-okat) detektálja, és a *CheckStyle*[4], amely a standardizált konvencióknak nem megfelelő kódrészeket észleli. A TypeScript forráskódot a *TSLint* elemezte olvashatóság, karbantarthatóság és lehetséges funkcionalitási hibák szempontjából.

Az alkalmazás különböző részeinek tesztelésére elsősorban unit tesztek vannak alkalmazva, melyek a kód írásával párhuzamosan voltak létrehozva *JUnit* keretrendszer segítségével. Feladatuk egy egység helyes működésének a felülvizsgálata.

A komponensek mellett a szolgáltatások tesztelése is szükséges volt, az integrációs tesztek, a 2.5. alfejezetben leírtaknak megfelelően, az Arquillian keretrendszer segítségével voltak létrehozva.

Az Android alkalmazás felületét *Espresso* [5] UI tesztek megírásával lehetett ellenőrizni. A keretrendszer lehetőséget nyújt elemek jelenlétének, létrejöttének vizsgálatára, ezek helyes működésének, interakcióinak megfigyelésére, illetve a komponenseknek megfelelő értékek kiértékelésére. Főleg feketedoboz típusú tesztelésre ad lehetőséget.

Az EKETour esetében a *Gradle*[2] végzi az Android alkalmazás és a szerver fordítását és függőségeik automatikus megoldását. A webes modul kezelése az *npm (node package manager)* [16] által valósult meg.

A Java projekt fejlesztése *IntelliJ IDEA*, a web alkalmazás fejlesztése *WebStorm IDEA*, a mobil alkalmazás fejlesztése *Android Studio* fejlesztői környezetekben volt megvalósítva. Ezek mellett több fejlesztői eszközt igénybe vett a csapat, mint például az adatbázis kezelését egyszerűsítő *MySql Workbench*-et, vagy a HTTP-kérésekre visszakapott válaszok vizsgálatára alkalmas *Postman*-t.

8. Következtetések

Az EKETour projekt fejlesztése során sikerült egy olyan szoftverrenszert megalkotni, amely a túrák menedzselésére és az ezekre történő regisztrálásra biztosít egy egységes felületet, megkönnyítve mind a szervezők, mind a résztvevők feladatait.

Az Android alkalmazás lehetőséget nyújt a túrázók regisztrálására (NFC-kártyával való párosítás), azonosítására és ezeknek a műveleteknek a naplózását is biztosítja.

A webes felhasználói felületen, a megfelelő adatokat megadva, az adminisztrátor létrehozhatja a túrákat, a hozzájuk tartozó túraváltozatokat és ellenőrzőpontokat, illetve a részvételi díjakat is meghatározhatja. Hozzáférése van a rendszerben szereplő felhasználói profilokhoz, megtekinthet adott túrákkal kapcsolatos kimutatásokat (pl. megrendelt pólók adatai) és a naplózás által nyomon is követheti a túrákat.

A bejelentkezett felhasználó létrehozhat több profilt, amelyekkel a későbbiekben regisztrálni tud adott túrákra, kiválasztva a legmegfelelőbb túraverziókat. Mielőtt véglegesítené a műveletet, további adatokat is megadhat, például rendelhet pólót, meghatározva annak méretét és típusát.

9. Továbbfejlesztési lehetőségek

Profilkép készítése minden versenyző számára a rajtvonalnál segíthet a résztvevők azonosításában. Az ellenőrzőpontoknál a pontbírok az NFC-kártya leolvasásakor a fényképeket is ellenőrizhetnék.

Mivel a túrák hosszabb ideig tarthatnak, a legtöbb versenyzőnek gondot okozhat a nap végén egy szállás keresése, vagy a visszautazás megoldása. Továbbfejlesztésként a szállítási és szállási opciók is bevezethetőek a rendszerbe, amelyeket regisztrációkor a versenyzők igénybe vehetnek, hasonlóan mint ahogy a pólók rendelése történik.

A túra nevezési díja egy jelképes összeg, ezért a fentebb említett szolgáltatások igénybevétele plusz költségnek számíthat, ezért ezeket az árakat is kezelnie kellene a rendszernek. A felületen, egy jól látható helyen, meg kell jeleníteni a felhasználó számára az összes költség alapján kiszámított végösszeget is.

A túra díjának az online befizetése leegyszerűsítené a szervezők munkáját. Bizonyos határidők elteltével, a korosztályok figyelembe vételével, vagy egyéb kedvezményekkel a túráknak a díja változhat. Mindezeket a szabályokat a rendszerbe beépítve, már a túrára való regisztrációkor fizethetnének a résztvevők online banki átutalással.

Az ellenőrzőpontokat egy térképen feltüntetve kirajzolódna a túra útvonala. Több túraváltozat esetében a felhasználók könnyen láthatnák az útvonalak közti különbséget. Az ellenőrzőpontokat is hitelesíteni lehet a pontbírók számára a telefon helymeghatározása által.

Hivatkozások

[1] *Android NFC*. URL: https://developer.android.com/reference/android/nfc/package-summary (utolsó elérés dátuma: 2018. ápr. 11.)

[2] *Building an Application with Spring Boot*. URL: https://spring.io/guides/gs/spring-boot/ (utolsó elérés dátuma: 2018. ápr. 17.)

[3] Scott Chacon és Ben Straub. *Pro Git, 2nd Edition*. Apress, 2014.

[4] *checkstyle – Checkstyle 8.9*. URL: http://checkstyle.sourceforge.net (utolsó elérés dátuma: 2018. ápr. 11.)

[5] *Create UI Tests with Espresso Test Recorder*. URL: https://developer.android.com/studio/test/espresso-test-recorder.html (utolsó elérés dátuma: 2018. ápr. 11.)

[6] Ralf S. Engelschall. *ECMAScript 6 - New Features: Overview & Comparison*. URL: http://es6-features.org/#Constants (utolsó elérés dátuma: 2018. ápr. 20.)

[7] *FindBugs™ - Find Bugs in Java Programs*. URL: http://es6-features.org/#Constants (utolsó elérés dátuma: 2018. ápr. 11.)

[8] Sergey Kryvets. *Simply about new HttpClient in Angular*. URL: https://sergeome.com/blog/2017/11/26/simply-about-new-httpclient-in-angular/ (utolsó elérés dátuma: 2018. ápr. 11.)

[9] David Weiser Lars Vogel Simon Scholz. *Using Retrofit 2.x as REST client - Tutorial*. URL: http://www.vogella.com/tutorials/Retrofit/article.html (utolsó elérés dátuma: 2018. ápr. 11.)

[10] Yohan Lasorsa. *The Missing Introduction to Angular and Modern Design Patterns*. URL: https://medium.com/ngx-rocket/the-missing-introduction-to-angular-and-modern-design-patterns-43e8815c2801 (utolsó elérés dátuma: 2018. ápr. 11.)

[11] Microsoft. *Typescript hivatalos oldala*. URL: http://www.typescriptlang.org (utolsó elérés dátuma: 2018. ápr. 20.)

[12] Josh Long Phillip Webb Dave Syer. *Spring Boot Reference Guide*. URL: https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/ (utolsó elérés dátuma: 2018. ápr. 11.)

[13] Keith Donald Rod Johnson Juergen Hoeller. *Spring Framework Reference Documentation, Part III. Core Technologies*. URL: https://docs.spring.io/spring/docs/3.0.x/spring-framework-reference/html/beans.html (utolsó elérés dátuma: 2018. ápr. 11.)

[14] *Scrum hivatalos oldala*. URL: http://www.scrumguides.org/scrum-guide.html (utolsó elérés dátuma: 2018. ápr. 11.)

[15] *Spring Data JPA - Reference Documentation*. URL: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/ (utolsó elérés dátuma: 2018. ápr. 11.)

[16] Michael Wanyoike és Peter Dierx. *A Beginner's Guide to npm — the Node Package Manager*. URL: https://www.sitepoint.com/beginners-guide-node-package-manager/ (utolsó elérés dátuma: 2018. ápr. 11.)

[17] Gray Watson. *OrmLite - Lightweight Object Relational Mapping (ORM) Java Package*. URL: hhttp://ormlite.com (utolsó elérés dátuma: 2018. ápr. 11.)