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

Turisztikai mobilalkalmazás Székelyudvarhely számára

Szerzők:

Buksa Ferencz - Attila
Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar, Informatika szak, III. év
Kiss Alpár
Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar, Informatika szak, III. év

Témavezetők:

dr. Simon Károly, egyetemi adjunktus
Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar
Kiss Kincső, szoftverfejlesztő, Codespring
Szécsi Zsolt, szoftverfejlesztő, Codespring

Kivonat

A dolgozat témáját képező projekt célja egy olyan szoftverrendszer megvalósítása, amely információkkal látja el a Székelyudvarhelyre érkező turistákat a helyi látványosságokról. A szoftver a TourInfo turisztikai információs iroda felkérése alapján készült.

A rendszer tartalmaz egy Android és egy webes kliens alkalmazást, valamint egy központi szervert, amely adatokkal szolgálja ki ezeket az alkalmazásokat. A turisták hozzáférhetnek a nevezetességek leírásaihoz, a hozzájuk tartozó hanganyagokhoz és képekhez. Az Android alkalmazás segítségével útvonaltervet is készíthetnek a látványosságokhoz. A turisztikai iroda munkatársai a webes felületen tudják menedzselni a rendszerben tárolt adatokat.

A dolgozat bemutatja a projekt felépítését, a fejlesztés során felhasznált technológiákat és eszközöket, a komponensek közötti kommunikációt és a fontosabb architekturális döntéseket, valamint ismerteti az alkalmazás működését.

Tartalomjegyzék

Bevezető

A Travel2U elsősorban turisták számára készült informatikai rendszer, amely információval szolgál a székelyudvahelyi nevezetességekről. A TourInfo turisztikai információs iroda felkérése alapján készült, a Codespring szoftverfejlesztő cég támogatásával.

A rendszer egy webes és egy Android kliens alkalmazásból valamint egy, az ezeket adatokkal kiszolgáló központi szerverből áll. A webes felületen lehet felvinni, módosítani vagy törölni a nevezetességek adatait, moderálni a hozzájuk fűzött megjegyzéseket. Az Android kliens biztosít egy lista és egy térkép nézetet, ahol megtekinthető az összes nevezetesség, valamint egy részletes nézetet, amely képeket, leírást, hanganyagokat, referencia linkeket társít az egyes látványosságokhoz. Ezek az információk a webes felületen is megtekinthetőek a turisták által. Mindezek mellett a turisták értékelni tudják a nevezetességeket, és megjegyzéseket fűzhetnek hozzájuk.

A projekt fejlesztése a U-Hub Mentorprogram keretén belül kezdődött 2017 nyarán. A fejlesztő csapat tagjai Buksa Ferencz-Attila és Kiss Alpár voltak, a fejlesztést segítő mentorok Kiss Kincső és Szécsi Zsolt, a témavezető tanár Dr. Simon Károly. A 2017-2018-as tanévben a csoportos projekt tantárgy keretén belül folytatódott a fejlesztés, erre az időre bővült a csapat három taggal: Imets Anna, Joó Nándor, Kelemen Kinga-Orsolya. A mentorokhoz csatlakozott továbbá Hobaj Roland a Codespring részéről. Az egyetemi félév során az Android kliens alkalmazás kibővült egy audio lejátszóval, valamint nemzetköziesítve lett. A webes alkalmazás bővült egy hírfolyammal, ahol a nevezetességekhez írt hozzászólások láthatóak, készültek UI tesztek és a webes felület többnyelvűsítése is megvalósult. A csoportos projekt befejezése után a jelen dolgozat szerzői folytatták a fejlesztést.

A projekt prototípusa 2018 márciusában került ki a Codespring teszt szerverére, jelenleg tesztelés alatt áll a TourInfo alkalmazottai által és hamarosan publikusan is elérhetővé, a turisták által használhatóvá válik.

A dolgozat a projekt funkcionalitásait és működését, a fejlesztése során felhasznált eszközöket és technológiákat, valamint a fejlesztési folyamatot mutatja be.

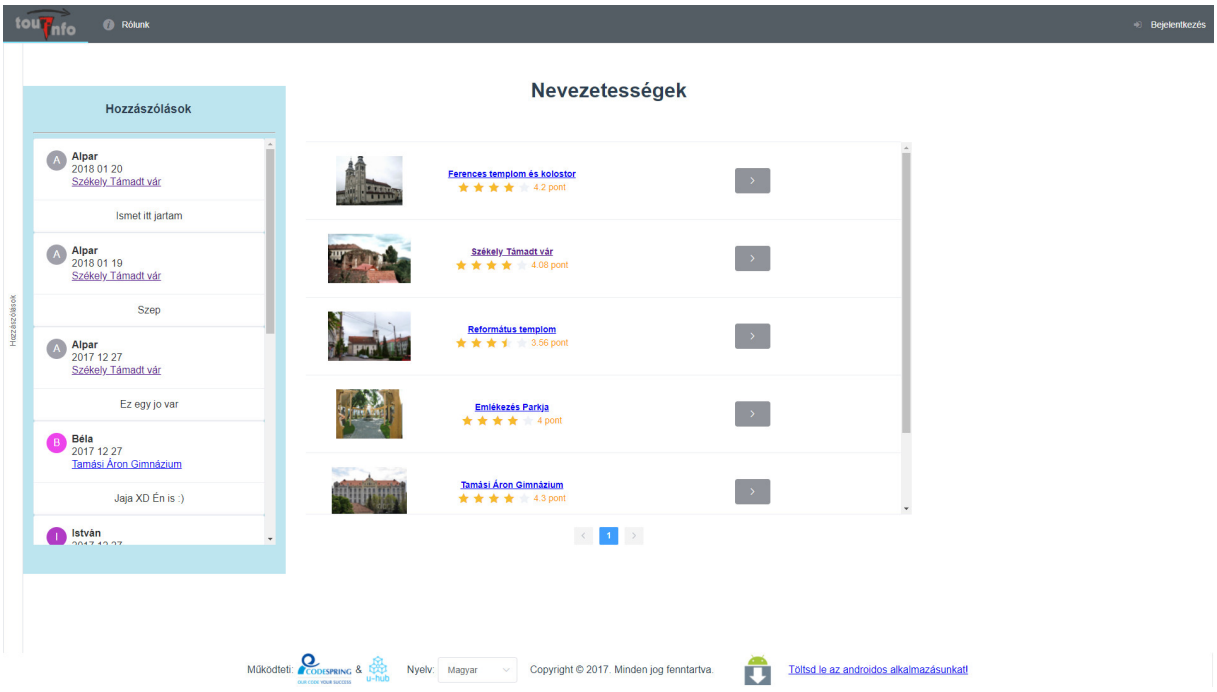
1. A Travel2U projekt

1.1. Funkcionalitások

A Travel2U projekt Android kliens alkalmazása a turistákat látja el információkkal, míg a webes kliens az adatok menedzsmentjét teszi lehetővé a TourInfo alkalmazottak számára. A webes felületen a turisták is megtekinthetik a nevezetességekkel kapcsolatos információkat.

1.1.1. A webes kliens alkalmazás funkcionalitásai

A webalkalmazás a nevezetességek adatainak menedzsmentjére készült. Egy másodlagos szerepe, hogy a főoldalon bejelentkezés nélkül megtekinthetőek a látványosságok egy listában, illetve egy részletesebb nézet is elérhető ezekről, és láthatóak a nevezetességekhez írt hozzászólások. A listában a látványosságok neve mellett található egy kis előnézeti kép, illetve egy értékelés, ez látható a(z) 1. ábrán.



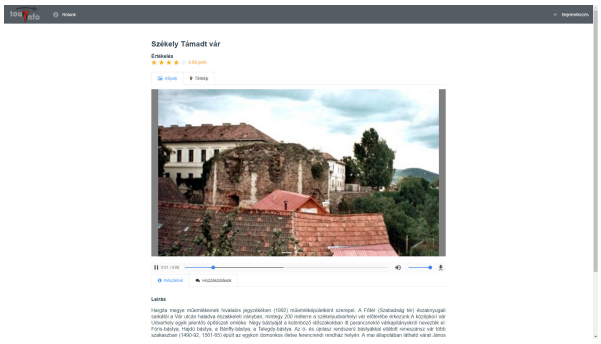
1. ábra. A nevezetességek listája, amely megtekinthető a főoldalon, bejelentkezés nélkül

Egy látványosság nevére kattintva egy új nézet nyílik meg a következő tartalmakkal:

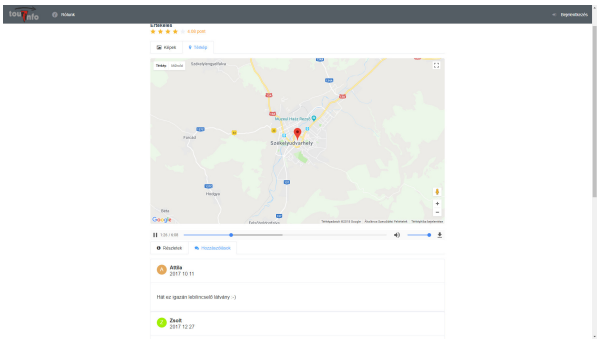
- Értékelés
- Képek
- Egy térkép a nevezetesség helyzetével

- Hanganyag
- Leírás
- Hozzászólások a nevezetességhez

Ezek láthatóak a(z) 2. ábrán.



(a) Részletes nézet a képekkel és a leírással

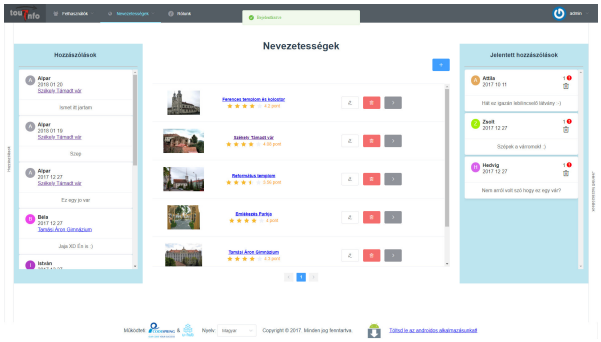


(b) Részletes nézet a helyzettel és a hozzászólásokkal

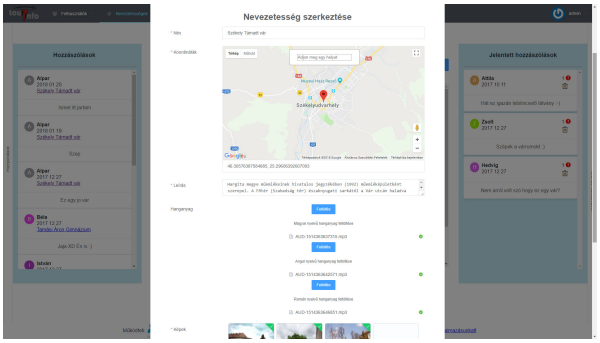
2. ábra. Egy nevezetesség részletes nézete képekkel, térképpel, leírással és hozzászólásokkal

A *Rólunk* menüpont a TourInfo irodáról tartalmaz információkat. Az oldalra két típusú felhasználóval lehet belépni: TourInfo felhasználóval, illetve rendszergazdával.

A TourInfo felhasználók számára belépés után a *Nevezetességek listája* menüpont jelenik meg, ahol egy hasonló listát látnak, mint a főoldalon, azonban itt lehetőség van a látványosságok törlésére, valamint az adatok előnézetére és módosítására egy-egy modális ablakban. Jobb oldalon a bejelentett hozzászólások láthatóak, amelyeket indokolt esetben törölni lehet. Ezeket lehet látni a(z) 3. ábrán.



(a) A nevezetességek listája az előnézeti, módosítási és törlési opciókkal, illetve a jelentett hozzászólások a jobb oldalon

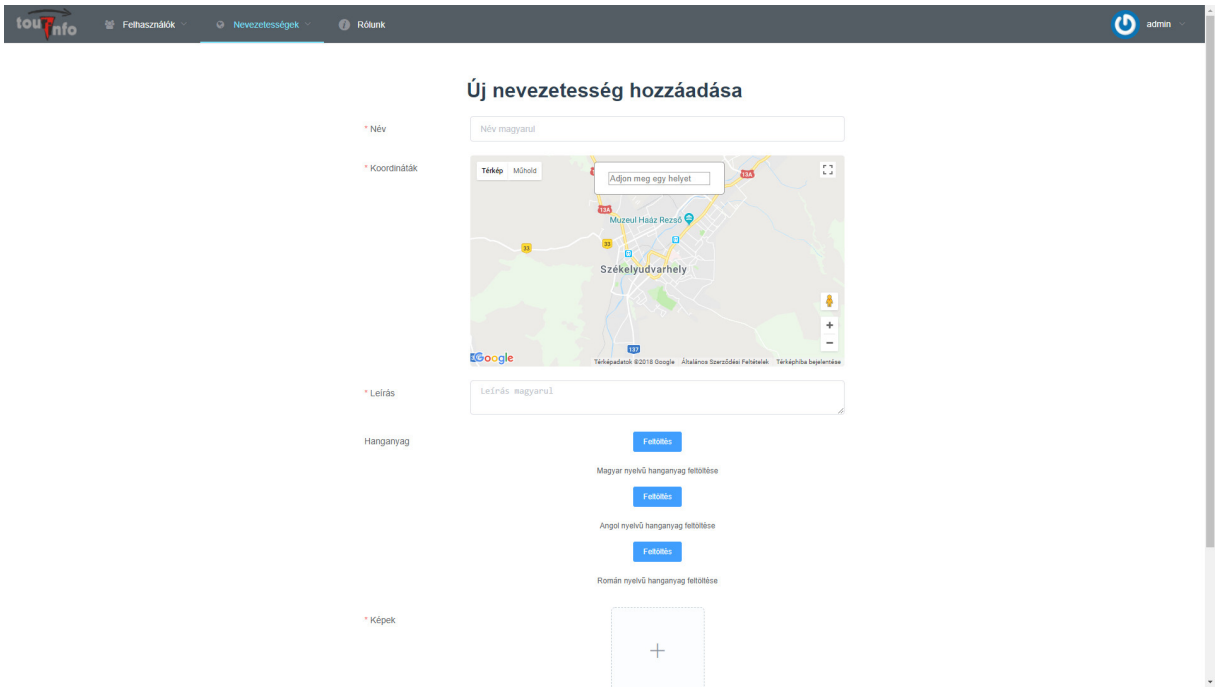


(b) Egy nevezetesség adatainak módosítása a modális ablakban

3. ábra. A nevezetességek bejelentkezett felhasználók számára megjelenített listája és egy létező elem módosítása

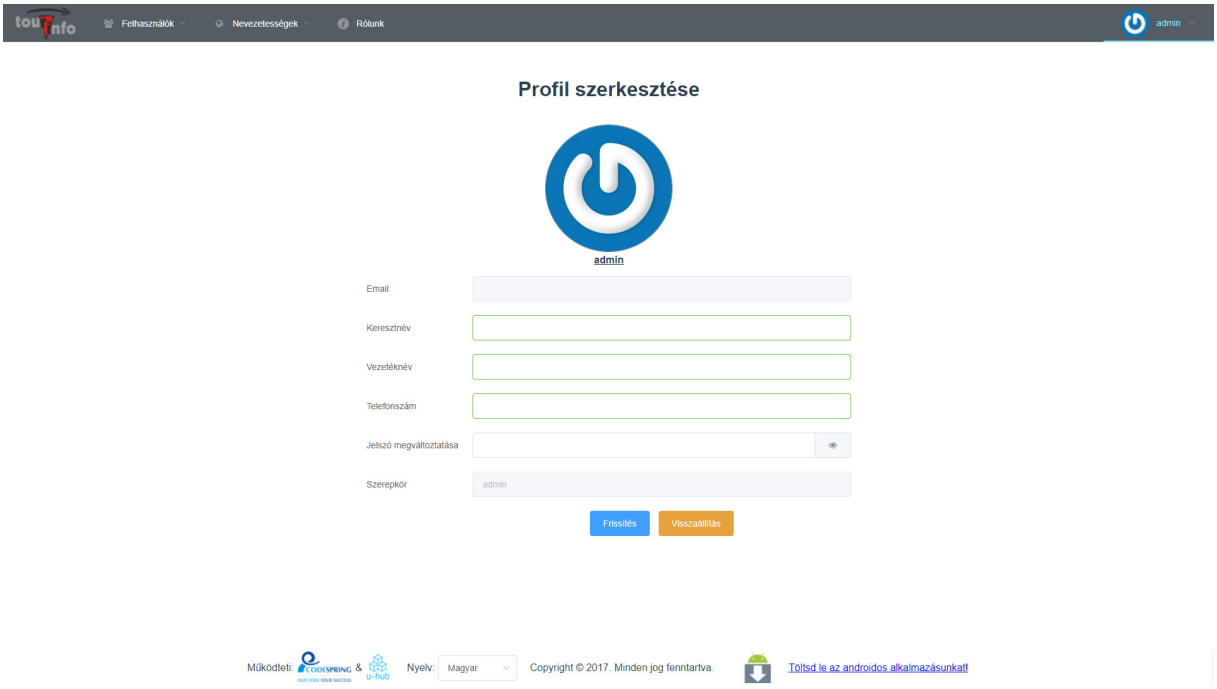
Az *Új nevezetesség hozzáadása* menüpont alatt lehet felvinni egy új látványosság adatait. A lista nézetből ez szintén megvalósítható a megfelelő gomb segítségével. Itt meg kell adni a

látványosság nevét és a leírását magyar, angol és román nyelven. A látványosság helyét egy térképen lehet megjelölni. Továbbá kötelezően fel kell tölteni legalább egy képet, amely a lista nézetben az előnézeti kép lesz. Több kép feltöltése esetén az elsőként feltöltött kép lesz az előnézeti kép. Opcionálisan fel lehet tölteni hanganyagot, mind a három nyelven, valamint meg lehet adni referencia linkeket, amelyeken az adott nevezetességről információkkal szolgáló oldalak érhetőek el. A nevezetesség hozzáadása oldal látható a(z) 4. ábrán.



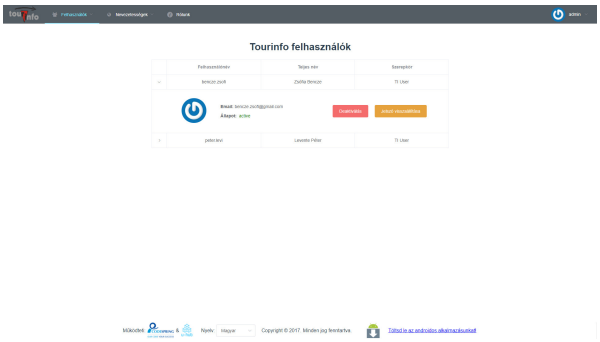
4. ábra. Az oldal, ahol fel lehet vinni egy új nevezetesség adatait

A *Profil szerkesztése* menüpont alatt a felhasználónak lehetősége van saját adatainak megadására és szerkesztésére, ez látható a(z) 5. ábrán.

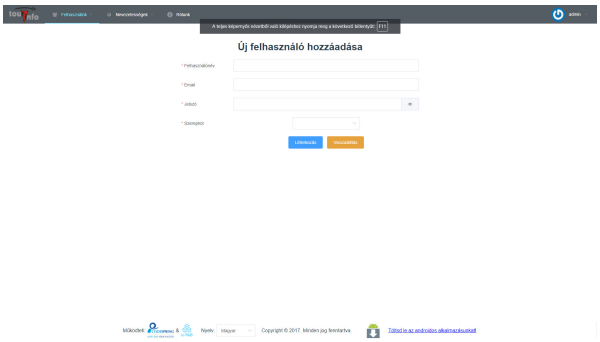


5. ábra. Felhasználói profil szerkesztése

A rendszergazda számára minden említett funkcionalitás elérhető, ugyanúgy, mint a TourInfo felhasználók számára. Továbbá a rendszergazda hozzá tud adni új felhasználókat a rendszerhez, látja a felhasználók adatait és menedzselni tudja őket: fel tudja függeszteni, vagy újra tudja aktiválni a felhasználókat, jelszó visszaállítást tud végezni. Az említett funkcionalitásokat biztosító felületek láthatóak a(z) 6. ábrán.



(a) Felhasználók listája a felfüggesztési/aktiválási és a jelszó visszaállítási opciókkal



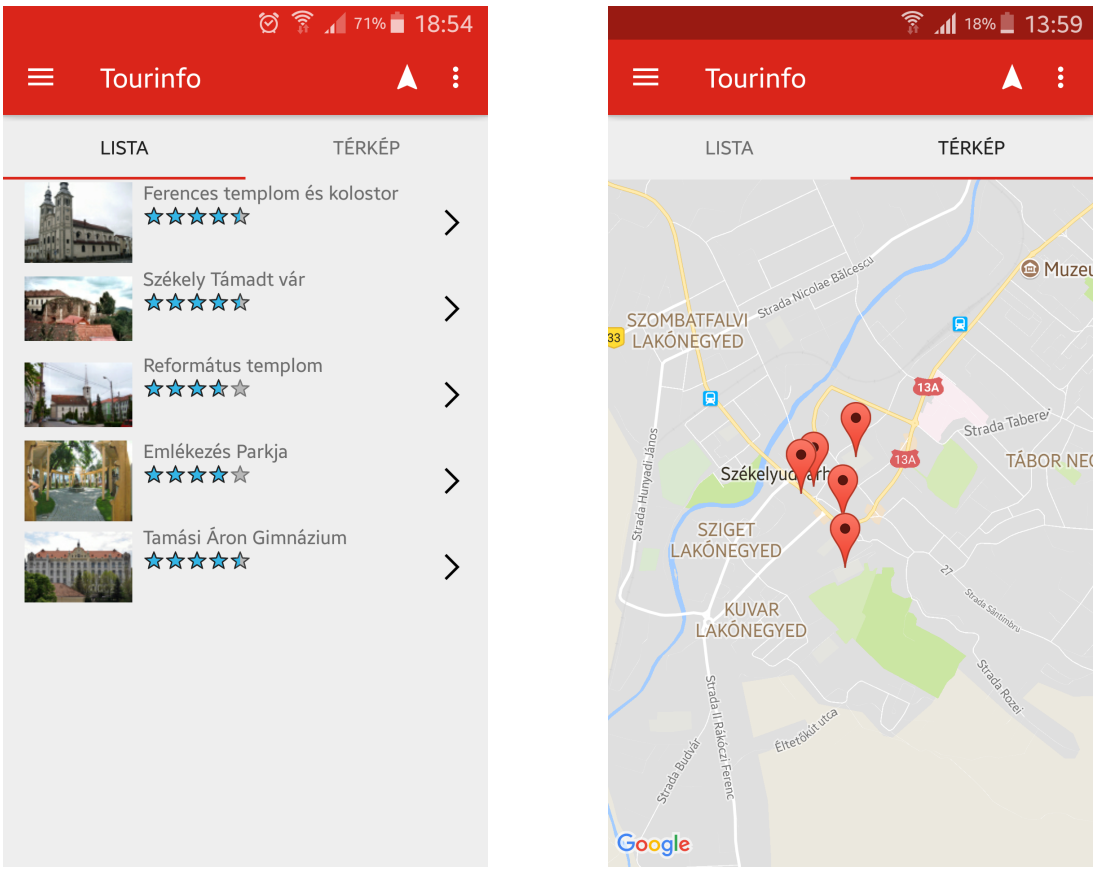
(b) Új felhasználó hozzáadása oldal

6. ábra. Felhasználók menedzsmentje

1.1.2. Az Android kliens funkcionalitásai

Az alkalmazás indításakor egy nyitókép jelenik meg a képernyőn, majd egy lista nézet a nevezetességekkel, ahogy a(z) 7. ábra mutatja.

A lista nézet mellett a felhasználónak lehetősége van a térkép nézetre váltani, ahol a neve-



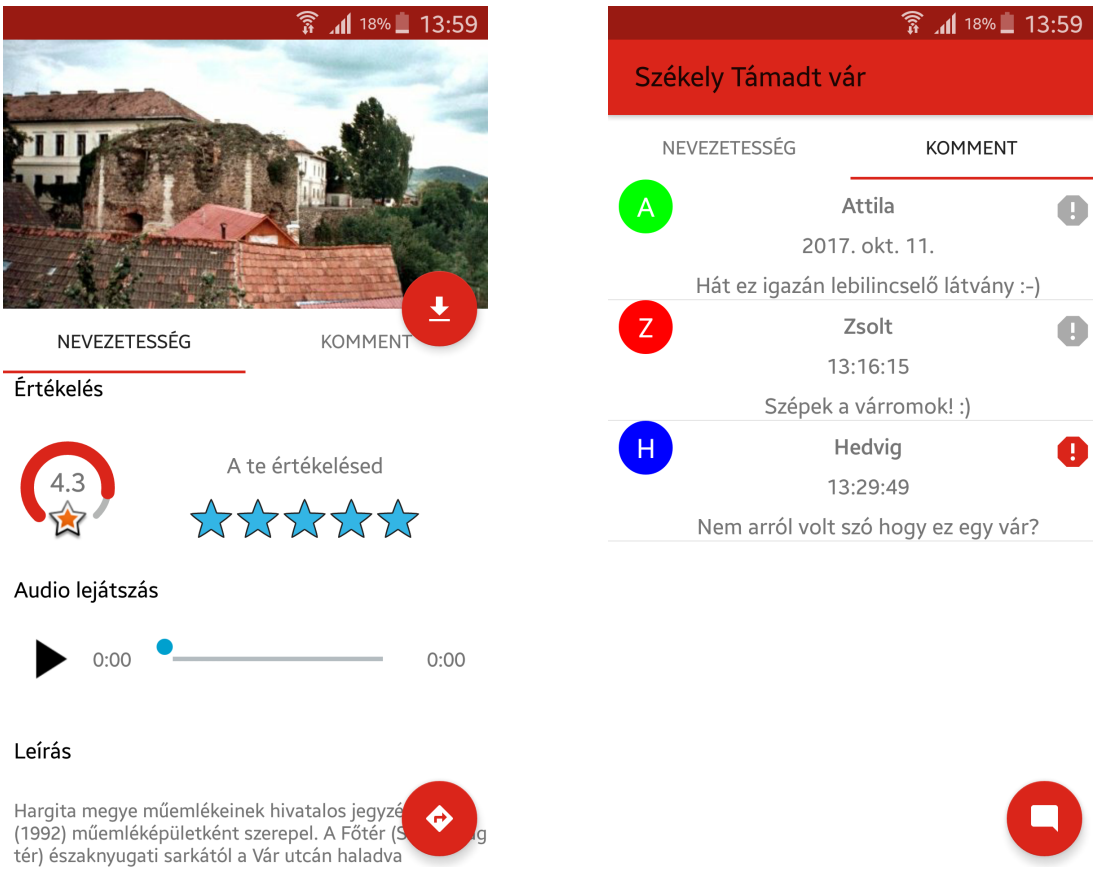
(a) Nevezetességek lista nézetben

(b) Nevezetességek térkép nézetben

7. ábra. A nevezetességek listán és térképen

zetességek egy térképre vannak helyezve. A listából vagy a térképről kiválasztható egy adott nevezetesség, amelynek így megtekinthetőek az adatai.

A nevezetesség részletes adatlapja tartalmazza a leírást, a képeket, valamint az értékeléseket. Továbbá erről a képernyőről érhető el a "Vigyél oda!" funkció és a lejátszó, amellyel elindítható a nevezetességhez csatolt hanganyag lejátszása. A képek az oldal tetején találhatóak, és jobbra vagy balra görgetéssel lehet megtekinteni ezeket. Az értékelés az oldal betöltésekor frissül, ha a felhasználó megadta a saját értékelését, akkor az alkalmazás ezzel együtt határoz meg egy új átlagot, és frissíti az értékelést. A lejátszó abban az esetben használható, ha a nevezetességhez töltöttek fel hanganyagot. Ha több nyelven is adtak meg hanganyagokat, az alkalmazás az aktuálisan használt nyelvet fogja választani, és ha ezen a nyelven nem elérhető hanganyag, akkor választási lehetőséget kínál az elérhetőek közül. A "Vigyél oda!" funkció kiválasztása után az alkalmazás megnyitja a Google Maps-et, amely megtervezi a legrövidebb útvonalat a felhasználó aktuális pozíciójától az adott nevezetességig. Egy nevezetesség részletes nézete a(z) 8a. ábrán látható.

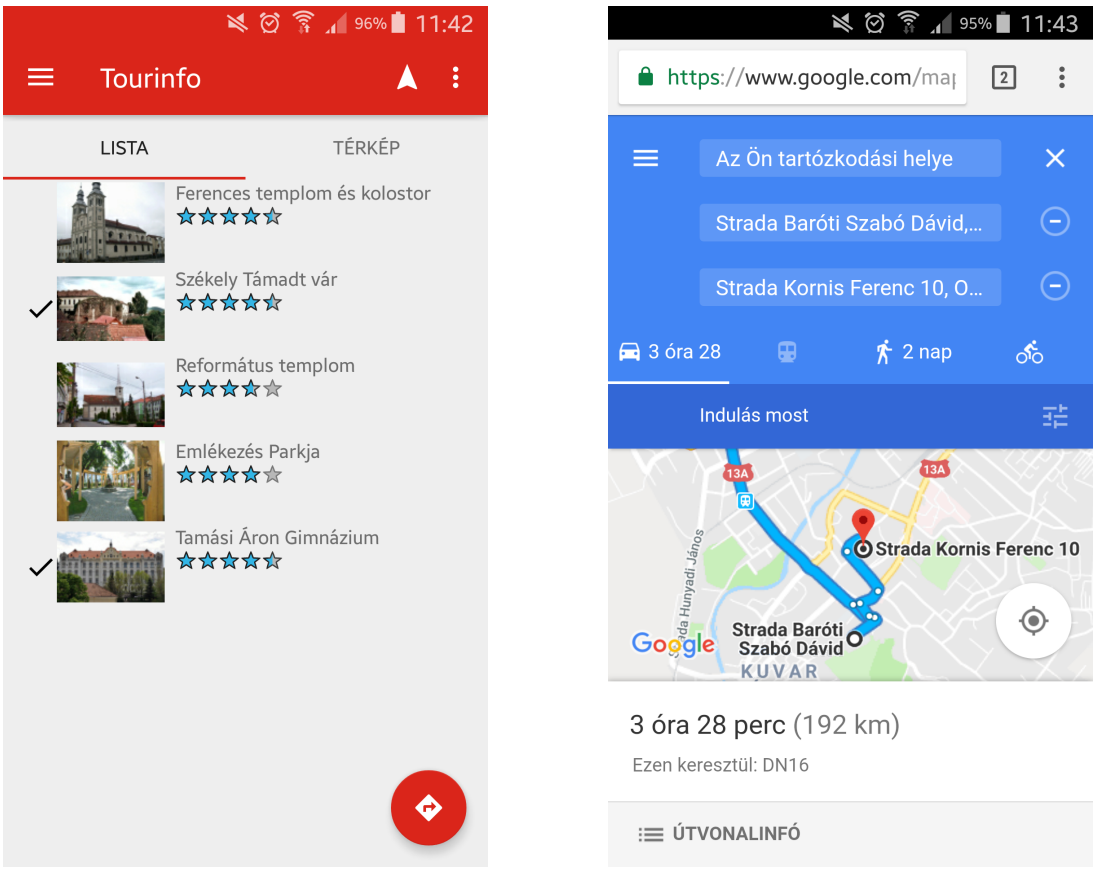


(a) Egy nevezetesség részletes nézete (b) Egy nevezetességhez tartozó hozzászólások

8. ábra. Egy nevezetesség részletes adatai és a hozzászólások listája

A részletes nézet mellett van egy komment fül is, amelyen a többi felhasználó hozzászólásai olvashatóak egy adott nevezetességről. Bárki tud hozzászólást írni, és lehetőség van ezek bejelentésére is, abban az esetben, ha a felhasználó zavarónak ítéli meg az adott hozzászólást. A jelentett kommenteket az adminisztrátorok figyelik, és ha indokolt, akkor törölhetik ezeket. Egy nevezetességhez tartozó hosszászólások láthatóak a(z) 8b. ábrán.

A lista nézetben van egy körúttervezési funkció, amelynek szerepe, hogy a listából kiválasztott nevezetességek alapján tervezzen a Google Maps-en egy útvonalat, amelyen sorban be tudja járni a felhasználó a kiválasztott nevezetességeket. A körút kiindulópontja a felhasználó aktuális pozíciója. Ez a funkcionalitás a nevezetesség menüpont alá tartozik, ahogy a 9. ábra mutatja.



(a) Nevezetességek kiválasztása egy körúthoz (b) Körút a kiválasztott nevezetességek között

9. ábra. Körúttervezés

A beállítások menüpont alatt a felhasználónak lehetősége van kiválasztani az alkalmazás nyelvét, amely meghatározza a már fentebb említett hanganyag lejátszást.

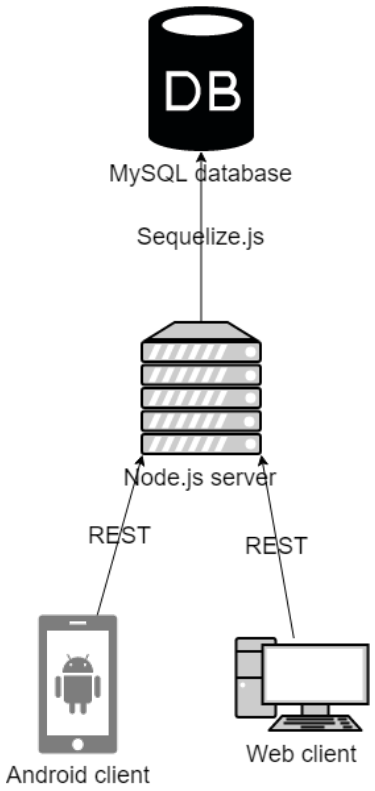
1.1.3. A szerver funkcionalitásai

A szerver felelős a nevezetességek és felhasználók adatainak adatbázisba való mentéséért, módosításáért és lekérdezéséért, valamint az alkalmazáslogika megvalósításáért. Egy RESTful API-t biztosít a kliensek számára, amelyen keresztül elérhetik és módosíthatják az erőforrásokat.

Némelyik erőforrás eléréséhez vagy módosításához megfelelő jogokkal kell rendelkezni. A TourInfo felhasználók adatait csak adminisztrátorok érhetik el, viszont a saját adatait bármelyik TourInfo felhasználó módosíthatja. A nevezetesség adatait bárki megtekintheti, hiszen az Android kliens alkalmazásba nem kell bejelentkezni, és a webes felületen is meg lehet tekinteni az adatokat bejelentkezés nélkül. Ezeknek az adatoknak a módosítására vagy törlésére csak a bejelentkezett TourInfo felhasználóknak vagy rendszergazdáknak van lehetőségük. A biztonság érdekében a kérések hitelesítésen mennek át.

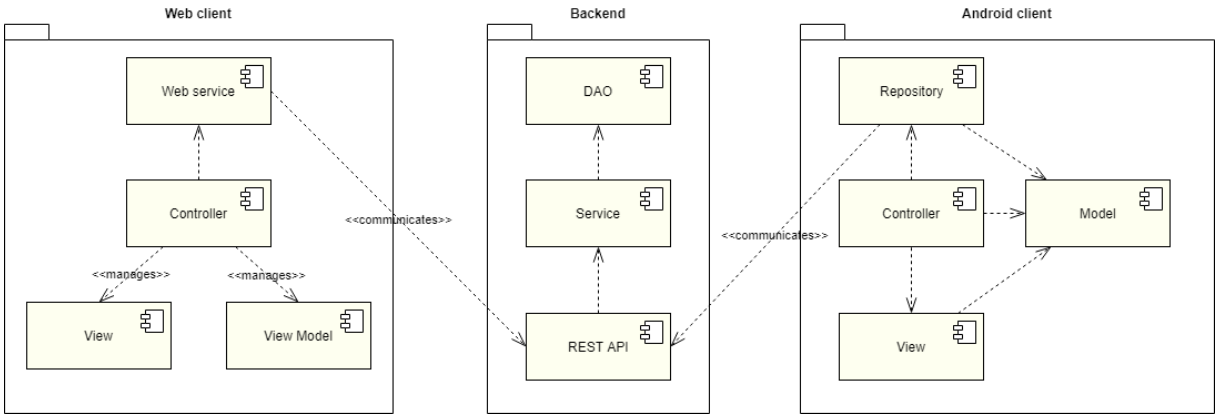
1.2. Architektúra

A projektet egy szerver és két kliens alkotja. A szerver implementációja Node.js-en alapszik, RESTful API-n keresztül kommunikál a webes és Android kliensekkel, Sequelize.js segítségével kommunikál egy MySQL adatbázissal, ahol az adatokat tárolja. Ezt a 10. ábra szemlélteti.



10. ábra. Architektúra diagram

A webes és az Android kliens alkalmazás esetében is a nézetet egy-egy Controller kezeli, amely a web modul esetében a Web service, az Android esetében a Repository komponensekkel kommunikál. Ez a két elem valósítja meg a kommunikációt a szerverrel, a RESTful API-n keresztül. A szerveren belül a RESTful API-t egy Service menedzseli. Ezeknek a komponenseknek a felépítése a(z) 11. ábrán látható.



11. ábra. Komponens diagram

2. Kliens oldali webes technológiák

A webes felület komponenseinek kezeléséért a Vue.js keretrendszer felelős. A felhasználói felület az Element UI eszköztár segítségével van megvalósítva. A szerverrel való aszinkron kommunikáció Axios-al történik.

2.1. Vue.js

A Vue [1] egy nyílt forráskódú JavaScript keretrendszer felhasználói felületek készítéséhez. Főként a nézetre koncentrál, de vannak más jellegű moduljai is, mint például a Vue-Router [2], és könnyen bővíthető külső könyvtárakkal is. Használható Single-Page Application (SPA) web alkalmazások fejlesztéséhez, ilyen a Travel2U webes modulja is. Az oldalon navigálva nem töltődik újra az oldal, hanem más komponenseket jelenít meg a keretrendszer. A tartalom betöltése utólagosan történik JavaScript segítségével.

2.2. Element

Az Element [3] egy Vue-ra épülő eszköztár, amely különböző UI komponenseket tesz elérhetővé. Használata felgyorsítja a fejlesztési folyamatot, mivel sok előre megírt komponenst szolgáltat, amelyek testre is szabhatóak. Használata egyszerű, ezen felül minden komponenshez tartozik dokumentáció és több példakód is. A webalkalmazás fejlesztése során használt UI komponensek nagyrészt ebből az eszköztárból származnak, beleértve a felugró értesítéseket, képnézegetőt, gombokat, navigációs menüt stb. Kivétel a térkép, amelyet egy másik könyvtár szolgáltat.

2.3. Axios

Az Axios [4] egy HTTP kliens böngészőhöz, segítségével GET, POST, PUT és DELETE kéréseket lehet intézni JavaScript-ből. A Vue rendelkezett egy HTTP klienssel, ez volt a vue-resource, de ezt a 2.0-ás verziótól eltávolították és inkább külső (third party) modulokat ajánlanak, főként az Axios-t, ezért döntött a fejlesztőcsapat ennek a használata mellet. Egy példakód látható a(z) 1. kódrészletben, a hozzászólások lekérése a szerverről.

```
axios({
  method: 'get',
  url: `${process.env.SERVER_URL}/comments`,
  withCredentials: true
})
.then(response => {
        this.commentList = response.data
})
.catch(error => {
  var msg
  if (error.response) {
          msg = error.response.data || error.response.statusText
  } else {
          msg = error.message
  }
  this.$message({
    type: 'error',
    message: msg
  })
})
```

1. kódrészlet. Példa kódrészlet a hozzászólások lekérésére a szerverről az Axios segítségével

2.4. Nightwatch.js

A Nightwatch [5] egy automatizált JavaScript tesztelési keretrendszer, segítségével End-To-End UI teszteket lehet írni. A projekt fejlesztése során a webalkalmazás felhasználói felületének tesztelésére volt használva. A dokumentáció alapján könnyen bekonfigurálható, használata egyszerű, a HTML elemeket és azok tartalmát a CSS szelektorokon keresztül lehet elérni, és ellenőrizni. Egy szintaxiskiemelt példakód látható a(z) 2. kódrészletben.

2.5. Vue-Gettext

A Vue-Gettext [6] egy plugin (beépülő modul) Vue alkalmazások nemzetköziesítéséhez, amely felhasználja a GNU gettext [7] több segédprogramját. A fordítani kívánt szövegeket több

```
module.exports = {
  'default e2e tests': function (browser) {
    // automatically uses dev Server port from /config.index.js
    // default: http://localhost:8080
    // see nightwatch.conf.js
    const devServer = browser.globals.devServerURL

    browser
      .url(devServer)
      .waitForElementVisible('#app', 5000)
      .assert.elementPresent('#monument-list')
      .assert.elementPresent('#monument-table')
      .assert.elementPresent('.monument-table-column')
      .end()
  }
}
```

2. kódrészlet. Példa kódrészlet a nevezetességek listájának tesztelésére a Nightwatch segítségével

módon is meg lehet jelölni, a HTML sablonban egy *translate* elemben megadjuk a szöveget, más HTML elemben, ahol szerepel a *v-translate* attribútum és JavaScript-ből a *vm.$gettext(msgid)* direktíva segítségével, ahol az *msgid* a fordítani kívánt szöveg. Egy makefile segítségével lehet kezelni a fordítási folyamatot. A *make makemessages* paranccsal összegyűjti a forrásállományokból (.html, .js, .vue) az annotációkkal ellátott szövegeket és elkészíti a .po kiterjesztésű fájlokat, amelyek a kulcsokat tartalmazzák. A projekt esetén a .po fájlok szerkesztéséhez a Poedit [8] volt használva. A szövegek lefordítása után a *make translations* paranccsal generálja a translations.json-t, amely a fordításokat tartalmazza. Innen kerülnek be a lefordított szövegek a forráskódokban annotált helyekre.

2.6. Vue2-Google-Maps

A Vue2-Google-Maps [9] egy JavaScript modul, amelynek komponensei lehetővé teszik a Google Maps API alkalmazását VueJS-ben. A projektben a nevezetesség helyének kiválasztására és megtekintésére szolgál. Használatához szükség volt egy Google Maps API Key-re.

3. Kliens oldali Android technológiák

Az Android kliens alkalmazás fejlesztése az Android SDK segítségével történt, a UI elemek elérése a Butterknife segítségével van megvalósítva. A képek betöltéséért a Picasso a felelős, míg a szerverrel való kommunikáció a Retrofit felhasználásával történik.

3.1. Android SDK

Az Android szoftverfejlesztő készlet [10] egy átfogó fejlesztési eszközcsomagot tartalmaz, ide tartozik például a debugger, a könyvtárak, a QEMU-n alapuló virtuális eszköz stb. A projekt esetében a készlet felhasználásával történt az Android kliens alkalmazás fejlesztése.

3.2. Retrofit

A Retrofit [11] egy nyílt forráskódú HTTP kliens Android és Java rendszerekhez, amelyet a Square fejlesztett ki. Ez a technológia valósítja meg a projekten belül a hálózati kommunikációt a szerver és az Android kliens alkalmazás között. A Retrofit annotációkat szolgáltat, amelyekkel GET, PUT, POST vagy DELETE kérések hozhatóak létre, mint ahogy a 3. kódrészlet is mutatja. A projektben ezek a kérések aszinkron függvényhívásokkal vannak megvalósítva, amelyekre a szerver JSON formátumban válaszol. Az JSON válaszokat a Retrofit automatikusan konvertálja és létrehoz egy-egy példányt a megfelelő modell osztályokból.

```
@GET("/api/monuments")
    Call<ArrayList<Monument>> getMonumentList(@Query("lang") String lang);
```

3. kódrészlet. Példa egy GET annotációra

3.3. Butterknife

A Butterknife [12] egy nyílt forráskódú könyvtár, amely összeköti a UI elemeit a Java osztályokban deklarált változókkal. A rendszer lehetőséget biztosít az Activity-k és Fragment-ek esetében is a hozzájuk rendelt nézetek attribútumainak eléréséhez, ahogy a 4. kódrészlet is mutatja.

```
@BindView(R.id.imageViewPager) ViewPager imageViewPager;
```

4. kódrészlet. Példa egy deklarációra Butterknife annotációval

3.4. Picasso

A Picasso [13] egy nyílt forráskódú könyvtár Android alkalmazásokhoz, amelyet a Square fejlesztett ki. Ez a technológia valósítja meg az alkalmazásban szereplő képek betöltését egy adott tárolóba. A szerverrel való kommunikáció során egy linket kap az alkalmazás, ezt használva a Picasso automatikusan elvégzi a betöltést. Hiba esetén egy az alkalmazás részét képező képet (placeholder-t) jelenít meg a tárolóban.

4. Szerver oldali technológiák

4.1. Node.js

A Node.js [14] egy JavaScript futtatási környezet, amelynek segítségével JavaScript kódokat lehet futtatni a böngésző nélkül. A Chrome V8 JavaScript motrára épül. Rengeteg modullal és hozzájuk tartózó részletes dokumentációval rendelkezik, ezeken kívül további külső modulok is használhatóak. Ezeket a modulokat többek közt az npmjs.com oldalon lehet megtalálni, és általában tartozik hozzájuk egy kisebb dokumentáció is. Telepítésük a npm (Node Package Manager) segítségével történik, amely a Node.js beépített függőségmenedzsment eszköze. A telepített modulok a projekt könyvtárában egy *node_modules* nevű alkönyvtárba kerülnek.

4.2. Sequelize.js

A Sequelize [15] egy ORM (Object Relational Mapping) keretrendszer Node.js-hez. A keretrendszer több dialektust támogat, többek között a MySQL-t is. A szoftverrendszer adatainak tárolására is MySQL adatbázis van használva. Nagyban megkönnyíti a fejlesztés folyamatát, mivel nem kell az adatbázis lekérdező nyelvét használni és segít a felhasználói bemenet megtisztításában. Egy szintaxiskiemelt példakód látható a(z) 5. kódrészletben a Sequalize konfigurálására és a(z) 6. kódrészletben a használatára egy előre definiált modell alapján.

4.3. Bcrypt

A Bcrypt [16] egy titkosító program, amely a Blowfish rejtjelező algoritmust használja. Több nyelven is van implementációja, többek között JavaScript-ben is. A szervernél ezt az implementációt biztosító könyvtár van használva a felhasználók jelszavainak titkosítására és ellenőrzésére. Egy példakód látható a(z) 6. kódrészletben jelszó titkosításra a Bcrypt segítségével.

4.4. Mysql-Wait

A Mysql-Wait [17] egy Node.js modul, amely MySQL adatbázis kapcsolat állapotának ellenőrzésére szolgál. Nagyon fontos a projekt számára, mivel az adatbázis és a szerver külön konténerben fut és a szerver csak akkor tud kapcsolódni az adatbázishoz, ha az már elindult.

```
const config = require('../config');
const sequelize = require('sequelize');
const dbConnection = new sequelize(
 config.database.dbName,
 config.database.username,
 config.database.password,
 {
  host: config.database.host,
  port: config.database.port,
  dialect: config.database.dialect,
  define: {
    charset: 'utf8',
    dialectOptions: {
      collate: 'utf8_general_ci'
    }
  }
 });

module.exports = dbConnection;
```

5. kódrészlet. Példa kódrészlet a Sequalize bekonfigurálására az adatbázis elérhetősége, típusa, neve és a felhasználó alapján

4.5. Basic-auth

A Basic-Auth [18] egy Node.js modul a HTTP kérések fejlécében levő autorizációs mező tartalmának kinyerésére. A webes alkalmazásba történő bejelentkezés folyamatának szerver oldali részében vesz részt. A beérkező kérésben be van állítva az autorizációs mező a felhasználó nevével és jelszavával, hash-elt formában. Ezeknek a hitelesítő adatoknak a kinyerésére van használva az említett modul.

4.6. Multer

A Multer [19] egy JavaScript modul Node.js-hez, amellyel fájlfeltöltést lehet kezelni szerver oldalon. A segítségével van megvalósítva a feltöltött képek és hanganyagok szerver oldali lementése.

4.7. Node-Thumbnail

A Node-Thumbnail [20] egy JavaScript modul, amelynek segítségével képeknek lehet létrehozni a kicsinyített változatát. A web és Android kliens alkalmazásban megjelenített lista

elemeiben elhelyezett előnézet képek generálásához van használva.

4.8. Winston

A Winston [21] egy személyre szabható JavaScript logger, a szerver hibaüzeneteinek és debug információinak loggolására van használva. A konfigurációjában megadható, hogy a különböző szintű információkat külön fájlba írja, például error.log, info.log és debug.log.

4.9. Nodemon

A Nodemon [22] egy JavaScript modul, amely megkönnyíti a fejlesztési folyamatot azzal, hogy figyeli a forrásfájlokat és újraindítja a szervert, ha változtatás történik. Nagy segítség a fejlesztők számára, mivel nem kell minden változtatás után manuálisan újraindítaniuk a szervert.

4.10. Mocha

A Mocha [23] egy JavaScript tesztelési keretrendszer Node.js-hez, a projektben unit tesztelésre van használva.

```
const bcrypt = require('bcrypt');
const config = require('../config');
const dbConnection = require('../db');
const logger = require('../log/logger');
const user = dbConnection.import('./model')
user.sync().then(() => {
  logger.info('The user table is synchronized');
  bcrypt.hash(config.admin.password, config.saltRounds, (err, hash) => {
    user
    .findOrCreate({
      where: { username: config.admin.username },
      defaults: {
        role: config.admin.role,
        password: hash,
        defaultPassword: hash
      }
    })
    .spread((user, created) => {
      if (created) {
        logger.info("The ADMIN user is created");
      } else {
        logger.info("The ADMIN user already exists in the database");
      }
    });
  });
});

module.exports = user;
```

6. kódrészlet. Példa kódrészlet a Sequelize ORM keretrendszer, a Bcrypt titkosító és a Winston logger használatára

5. Fejlesztési eszközök és módszerek

A projekt fejlesztése során felhasznált fejlesztői eszközök és módszerek leegyszerűsítették és felgyorsították a munkafolyamat egyes fázisait. Ilyen eszköz például a Git, amely lehetővé tette a verziókövetést, a Gradle, amely az Android oldali build- és függőségmenedzsmentet segítette elő, az NPM, amely a web és szerver oldali függőségmenedzsmentet valósítja meg. Az alkalmazott fejlesztési módszer a Scrum volt.

5.1. Scrum

A Scrum [24] egy Agile módszer, amely segíti a fejlesztői csapat tagjai közötti kommunikációt.

A projekt során megvalósítandó funkcionalitásokat a csapat két hetes sprint-ekbe osztotta be. Minden sprint végén volt egy kiértékelő, illetve egy demo, ahol be voltak mutatva a futam ideje alatt megvalósított feladatok. A sprint-eket indító tervezési fázisban voltak meghatározva és prioritizálva a következő futam során megvalósítandó feladatok. A szakmai gyakorlat ideje alatt a csapat minden nap tartott egy rövid megbeszélést, amelyen a tagok sorra beszámoltak arról, hogy mivel haladtak az azelőtti nap, melyek a célkitűzéseik aznapra, valamint, hogy ütköztek-e valamilyen problémába. Az egyetemi év kezdete után a napi megbeszélések helyét heti rendszerességű gyűlések vették át.

5.2. NPM

Az NPM [25] (node package manager) egy függőségmenedzsment eszköz, amely JavaScript könyvtárakat és modulokat tesz elérhetővé. Három különálló részből áll: az npmjs.com weboldal, ahol böngészni lehet a csomagok között; a registry, amely egy publikus adatbázis a JavaScript csomagokkal; és a CLI (command line interface), amelyet parancssorból lehet használni a csomagok telepítéséhez. Az npm lehetőséget nyújt saját csomagok publikálásához, lehet privát csomagokat is feltölteni és cégek számára van lehetőség a registry tükrözésére saját szerverre. A projekt esetében az eszköz a web és szerver oldali függőségmenedzsmentet valósítja meg.

5.3. ESLint

Az ESLint [26] egy nyílt forráskódú statikus kódelemző eszköz JavaScript programok forráskódjának ellenőrzésére. Segítségével kiküszöbölhetőek egyes hibák, és kialakítható egy egységes programozási stílus a csapatban résztvevő fejlesztők között. Segítségével észlelhetőek például a nem deklarált vagy nem inicializált változók, és megszorításokat lehet megszabni a stílussal kapcsolatban, mint például a konzisztens indentálás, a pontosvesszők használata, Unix-os vagy Windows-os sorvégek stb. A projekt esetében a Node.js szerver és a webes felület JavaScript kódjainak ellenőrzésére van használva.

5.4. Webpack

A Webpack [27] egy modul-csomagoló eszköz, amely képes összecsomagolni JavaScript modulokat vagy webalkalmazásokat képekkel, betűtípusokkal és CSS fájlokkal együtt, és rekurzívan felépít hozzájuk egy gráfot a függőségekkel és azok függőségeivel. A webes kliens alkalmazás build-elésére van felhasználva.

5.5. Gravatar

A Gravatar [28] egy szolgáltatás, amelyet Tom Preston-Werner fejlesztett ki, és célja, hogy a regisztráció után a felhasználók által megadott e-mail címhez hozzárendel egy, szintén a felhasználó által megadott képet, és azután ha a felhasználó máshol is használja az e-mail címet, akkor az ott regisztrált profilhoz automatikusan hozzáadja a Gravatar-on megadott képet. A projektben a Gravatar valósítja meg a webes oldalon az adminisztrátorok profiljainak esetében a kép hozzárendelését, ha azzal az e-mail címmel már regisztráltak a Gravatar-on.

5.6. Android Studio

Az Android Studio [29] a Google hivatalos fejlesztői környezete az Android alkalmazásokhoz, amely IntelliJ-re épül. Az Android kliens alkalmazás esetében ebben voltak megírva az xml állományok, amelyek a UI elemeket és azok hierarchiáját tartalmazzák, valamint a Java állományok, amelyek a kommunikációt illetve a UI elemek viselkedését határozzák meg.

5.7. Gradle

A Gradle [30] egy build- és függőségmenedzsmentet megvalósító rendszer, amely Groovy-n és Kotlin-on alapszik. A projekt esetében ez az eszköz valósítja meg az Android oldalon a függőségek automatikus letöltését és konfigurálását, valamint a rendszer automatikus build-elését.

5.8. Fabric.io

A Fabric.io [31] egy felület, ahol a fejlesztőknek lehetőségük van regisztrálás után közzétenni az alkalmazásukat, és e-mailes meghívások által ezt megosztani további felhasználókkal. Egy alkalmazás esetében a Fabric különböző statisztikákat is készít, például arról, hogy milyen gyakran használják a meghívottak az alkalmazást, és rögzíti az alkalmazás használata során fellépő hibákat is. A projekt fejlesztése során, mikor egy új funkcionalitás valósult meg, akkor, a develop branch-hez hozzáadva az új részt, az alkalmazás frissült a Fabric-on is. Amikor a felhasználók kipróbálták az új részt, a Fabric e-mailen keresztül értesítette a fejlesztőket, ha az alkalmazás leállt valamilyen hiba következtében, pontosan megadva azt is, hogy a felhasználó az alkalmazás melyik részén ütközött hibába.

5.9. Git

A Git [32] egy osztott verziókövető rendszer. A projekt fejlesztése során Git repository-ban voltak tárolva a projekt egyes verziói, külön a web, a szerver és az Android projektek. Kliens alkalmazásként a GitKraken volt használva.

5.10. GitLab

A GitLab egy repository-menedzsment rendszer, amely a Git tárhely menedzselése mellett lehetőséget nyújt a feladatok nyomon követésére is, valamint a dokumentáció tárolására, és a folyamatos integráció megvalósítására, illetve felületet biztosít a kód felülvizsgálatára.

A fejlesztés során a feladatok kezelésére, a kód felülvizsgálatára és a folyamatos integrációra használta a fejlesztőcsapat. A rendszer négy projektre van szétbontva: az Android kliens, a webes kliens, a szerver és a Docker projekt, amely alprojektként hivatkozik az előző három projektre és a kitelepítésért felelős.

5.11. Docker

A Docker [33] egy konténer platform, amely megkönnyíti az alkalmazások telepítését. Egy konfigurációs fájlban (Dockerfile) lehet megadni, hogy mire van szüksége az alkalmazásnak, például milyen futtatási környezet kell hozzá, be lehet állítani környezeti változókat stb. A Dockerfile-ból egyetlen paranccsal létrehozható a Docker Image. Egy Image futó példányát konténernek hívják. A Docker Image-eket lehet mozgatni, és elindítható belőlük akárhány konténer, ez által válik az alkalmazás könnyen skálázhatóvá. A Docker Compose megkönnyíti a konténerek indítását és több konténer kapcsolatának leírását.

A Travel2U szoftverrendszer kitelepítéséhez és futtatásához Docker Compose van használva, mivel külön konténerben fut az adatbázis szerver és az alkalmazás szerver. Az alkalmazás szerver számára van írva egy saját Docker Image, amelyben a futtatási környezet Node.js, be vannak állítva a szükséges környezeti változók, és le van build-elve az Android és webes kliens alkalmazás.

Következtetések és továbbfejlesztési lehetőségek

Következtetések

A Travel2U projekt egy teszt verziója már működésben van, jelenleg a TourInfo alkalmazottai tesztelik a rendszert. A tesztelési fázis lezárása után a publikus verziója is hamarosan elérhetővé válik, használható lesz a Székelyudvarhelyre érkező turisták által.

A fejlesztés során a csapat több helyen bemutatta az alkalmazást, mint például a U-Hub Mentorprogram nyári gyakorlatát lezáró táborban, az IT Plus klaszter által szervezett karácsonyi IT meetup-on, valamint a Babeş-Bolyai Tudományegyetem Bittologatók előadássorozatának keretein belül.

Továbbfejlesztési lehetőségek

A projekt fejlesztése során több továbbfejlesztési lehetőség felmerült. Mind a webes, mind az Android kliens alkalmazás esetében még megoldásra vár a bejelentkezés szociális hálók használatával a turisták számára. Ha a turistáknak is lesz profiljuk, akkor a nevezetességekhez fűzött hozzászólásaikat is tudják majd szerkeszteni, illetve törölni, valamint lehetőségük lesz megosztani nevezetességeket és hosszászólásokat a saját felhasználójukkal.

További lehetőség, hogy minden nevezetesség esetében az alkalmazás tudja ajánlani a következő legközelebbi látványosságot. És felmerült, hogy az alkalmazásba ne csak nevezetességeket lehessen feltölteni, hanem külön éttermeket, kávézókat is, és ezek is kerüljenek bele az ajánlási rendszerbe. A média anyagokat tematika alapján is lehetne csoportosítani, például lehetnének aktuális és történelmi tartalmak stb. A web alkalmazás esetében felmerült, hogy ne csak a nevezetességeket tekinthessék meg a turisták, hanem székelyudvarhelyi eseményeket, rendezvényeket is lehessen feltölteni.

Hivatkozások

[1] Evan You. *Introduction — Vue.js*. URL: https://vuejs.org/v2/guide/ (utolsó elérés dátuma: 2018. márc. 17.)

[2] *Introduction · vue-router*. URL: https://router.vuejs.org/en/%20https://github.com/vuejs/vue-router (utolsó elérés dátuma: 2018. márc. 21.)

[3] *Element - A Desktop UI Toolkit for Web*. URL: http://element.eleme.io/#/en-US (utolsó elérés dátuma: 2018. márc. 17.)

[4] Rubén Norte Matt Zabriskie, Nick Uraltsev. *GitHub - axios/axios: Promise based HTTP client for the browser and node.js*. URL: https://github.com/axios/axios (utolsó elérés dátuma: 2018. márc. 19.)

[5] Nightwatch. *Nightwatch.js | Node.js powered End-to-End testing framework*. 2016. URL: http://nightwatchjs.org/ (utolsó elérés dátuma: 2018. márc. 18.)

[6] Marc Hertzog. *GitHub - Polyconseil/vue-gettext: Translate your Vue.js applications with gettext.* URL: https://github.com/Polyconseil/vue-gettext (utolsó elérés dátuma: 2018. márc. 19.)

[7] *gettext*. URL: https://www.gnu.org/software/gettext/ (utolsó elérés dátuma: 2018. márc. 22.)

[8] Václav Slavík. *Poedit - Gettext Translations Editor*. URL: https://poedit.net/ (utolsó elérés dátuma: 2018. márc. 19.)

[9] Daniel Sim. *GitHub - xkjyeah/vue-google-maps: Google maps component for vue with 2-way data binding*. URL: https://github.com/xkjyeah/vue-google-maps (utolsó elérés dátuma: 2018. márc. 19.)

[10] *Download an Archived Android SDK | Android Developers*. URL: https://developer.android.com/sdk/download.html (utolsó elérés dátuma: 2018. márc. 20.)

[11] UK-Green Building Council. *Introduction to Retrofit*. 2013. URL: http://www.baeldung.com/retrofit%20http://www.ukgbc.org/content/retrofit (utolsó elérés dátuma: 2018. márc. 19.)

[12] Jake Wharton. *Butter Knife*. 2013. URL: http://jakewharton.github.io/butterknife/%20https://android-arsenal.com/details/1/131.

[13] *Picasso*. URL: http://square.github.io/picasso/ (utolsó elérés dátuma: 2018. márc. 19.)

[14] *Node.js*. URL: https://nodejs.org/en/ (utolsó elérés dátuma: 2018. márc. 20.)

[15] *Manual Sequelize The node.js ORM for PostgreSQL, MySQL, SQLite and MSSQL*. URL: http://docs.sequelizejs.com/.

[16] *GitHub - kelektiv/node.bcrypt.js: bcrypt for NodeJs*. URL: https://github.com/kelektiv/node.bcrypt.js (utolsó elérés dátuma: 2018. márc. 20.)

[17] Josh Vanderwillik. *GitHub - JoshWillik/mysql-wait: Mysql wait will wait until your mysql database is up, and then trigger*. URL: https://github.com/JoshWillik/mysql-wait (utolsó elérés dátuma: 2018. márc. 20.)

[18] *GitHub - jshttp/basic-auth: Generic basic auth Authorization header field parser*. URL: https://github.com/jshttp/basic-auth (utolsó elérés dátuma: 2018. márc. 20.)

[19] *GitHub - expressjs/multer: Node.js middleware for handling 'multipart/form-data'*. URL: https://github.com/expressjs/multer (utolsó elérés dátuma: 2018. márc. 20.)

[20] Honza Pokorny. *GitHub - honza/node-thumbnail: Thumbnail worker queue for node.js*. URL: https://github.com/honza/node-thumbnail (utolsó elérés dátuma: 2018. márc. 20.)

[21] *GitHub - winstonjs/winston: A logger for just about everything*. URL: https://github.com/winstonjs/winston (utolsó elérés dátuma: 2018. márc. 20.)

[22] Remy Sharp. *nodemon*. URL: https://github.com/remy/nodemon%20https://nodemon.io/ (utolsó elérés dátuma: 2018. márc. 20.)

[23] *Mocha - the fun, simple, flexible JavaScript test framework*. URL: https://mochajs.org/ (utolsó elérés dátuma: 2018. márc. 24.)

[24] Margaret Rouse. *What is Scrum?* 2017. URL: https://www.scrum.org/resources/what-is-scrum%20http://searchsoftwarequality.techtarget.com/definition/Scrum (utolsó elérés dátuma: 2018. ápr. 14.)

[25] *What is npm?* URL: https://docs.npmjs.com/getting-started/what-is-npm (utolsó elérés dátuma: 2018. ápr. 14.)

[26] *Getting Started with ESLint - ESLint - Pluggable JavaScript linter*. URL: https://eslint.org/docs/user-guide/getting-started (utolsó elérés dátuma: 2018. ápr. 14.)

[27] *webpack*. URL: https://webpack.js.org/ (utolsó elérés dátuma: 2018. ápr. 14.)

[28] *Gravatar - Globálisan felismerhető Avatarok*. URL: http://hu.gravatar.com/ (utolsó elérés dátuma: 2018. márc. 20.)

[29] „Meet Android Studio | Android Studio". (). URL: https://developer.android.com/studio/intro/index.html.

[30] *Gradle Build Tool*. URL: https://gradle.org/ (utolsó elérés dátuma: 2018. ápr. 14.)

[31] *Crashlytics — Fabric for Android documentation*. URL: https://docs.fabric.io/android/fabric/overview.html%20https://docs.fabric.io/android/crashlytics/overview.html (utolsó elérés dátuma: 2018. ápr. 14.)

[32] *About - Git*. URL: https://git-scm.com/about (utolsó elérés dátuma: 2018. ápr. 14.)

[33] Solomon Hykes. *Docker - Build, Ship, and Run Any App, Anywhere*. 2013. URL: https://www.docker.com/ (utolsó elérés dátuma: 2018. ápr. 14.)