BABEŞ-BOLYAI UNIVERSITY CLUJ–NAPOCA
FACULTY OF MATHEMATICS AND INFORMATICS
SPECIALIZATION: COMPUTER SCIENCE

License Thesis

Chain-Guide, the cyclists webapplication

Abstract

The topic of dissertation is the self-developed, Java-based Web application called Chain-Guide. The app is designed to help cyclists to find suitable routes, services(shop, rental, service) for them.

As part of the application , users can plan cycle-friendly routes avoiding the busy streets of the city and can discover new hiking trails as well. Besides, there is possibility to choose from a variety of services based on different searching criterias, and also to evaluate them. The maintainability of the application is provided by an administrative interface. The information about cycling routes is served by the OpenCycleMap through MapQuest API. The web-based user interface uses the Vaadin framework, while the communication with the database was based on the Hibernate framework.

Its strength lies on the fact that is unique in the market among the cyclist web applications wich are available in our country and provide additional information as well.

This work is the result of my own activity. I have neither given nor received unauthorized assistance on this work.

JULY 2015 KÁTAY CSILLA

ADVISOR:
RUFF LAURA, ASSISTANT PROFESSOR

BABEŞ-BOLYAI UNIVERSITY CLUJ–NAPOCA
FACULTY OF MATHEMATICS AND INFORMATICS
SPECIALIZATION: COMPUTER SCIENCE

License Thesis

Chain-Guide, the cyclists webapplication

SCIENTIFIC SUPERVISOR:
RUFF LAURA, ASSISTANT PROFESSOR

STUDENT:
KÁTAY CSILLA

JULY 2015

UNIVERSITATEA BABEŞ-BOLYAI, CLUJ–NAPOCA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ

Lucrare de licenţă

Chain-Guide

CONDUCĂTOR ŞTIINŢIFIC:
LECTOR DR. RUFF LAURA

ABSOLVENT:
KÁTAY CSILLA

IULIE 2015

BABEŞ-BOLYAI TUDOMÁNYEGYETEM KOLOZSVÁR
MATEMATIKA ÉS INFORMATIKA KAR
INFORMATIKA SZAK

Licensz-dolgozat

Chain-Guide, a biciklibarát webalkalmazás

TÉMAVEZETŐ:
DR. RUFF LAURA, EGYETEMI ADJUNKTUS

SZERZŐ:
KÁTAY CSILLA

2015 JÚLIUS

Tartalomjegyzék

1. fejezet

Bevezető

A dolgozat témája a saját fejlesztésű, Chain-Guide nevű, *Java* alapú webalkalmazás megvalósítása. Az alkalmazás célja, hogy segítse a biciklisek közlekedését, azon városokban is ahol a *Google Maps* ezen része (cycling direction and bike routes[1]) még nem elérhető, pedig a kerékpár utak száma növekvőben van. Az alkalmazás az útvonalválasztás mellett, a bicikli orientált szolgáltatások (kölcsönzés, szervízelés stb.) terén is segítséget nyújt felhasználóinak.

Az alkalmazás keretén belül, a felhasználók kerékpárbarát útvonalakat tervezhetnek elkerülve ezáltal a város forgalmas utcáit vagy új túraösvényeket, esetleg extrém parkokat is felfedezhetnek a kalandra vágyók. A szolgáltatások terén sem marad alul, hiszen lehetőséget nyújt, hogy a felhasználó megtalálja a neki megfelelő üzletet, szervízt vagy kölcsönzőt. A keresési feltételek listájában mind a közvélemény, mind a nyitvatartás és közelség is helyet foglal. Mindemellett véleményezésre is lehetőségük nyílik a felhasználóknak.

Az alkalmazás karbantarthatóságát egy adminisztrációs felület biztosítja, mely elengedhetetlen ahhoz, hogy ez naprakész információkat használjon a különböző szolgáltatásokhoz. A biciklizés szempontjából fontosabb út-információkat az *OpenCycleMap*[Allan, 2007] (a legelterjedtebb biciklis réteggel rendelkező térkép-szolgáltatás mely világszerte elérhető, beleértve Romániát is és szabadon aktualizálható, kiegészíthető) adja, melyet a *MapQuest API*[szerzo, ev]-n keresztül ér el az alkalmazás. A webes felületet a *Vaadin*[Grönroos, 2013] keretrendszer segítségével valósítottuk meg, míg az adatbázissal történő komunikáció a *Hibernate*[szerzo, 2015] programkönyvtár segítségével lett kivitelezve.

A dolgozat szerkezetét illetően .... főbb részre bontható..... (itt akkor ezt utólag fogom hozzáírni)

.

.

.

1. http://googlepolicyeurope.blogspot.ro/2013/05/bringing-biking-directions-to-more-of.html

Erőssége abban rejlik, hogy egyedi a piacon kínálkozó biciklis-alkalmazások közt, amelyek hazánkban, környékünkön is elérhetőek. A szolgáltatásokkal járó extra információk is az alkalmazás előnyeiként említhetőek meg, ugyanúgy mint a felhasználóbarát megjelenítés vagy az egyszerű használat.

2. fejezet

Felhasznált technológiák

Összefoglaló: *Ebben a fejezetben a felhasznált tecnológiák kerülnek a középpontba. Szó lesz a Vaadin[Grönroos, 2013] és Hibernate[szerzo, 2015] keretrendszerekről, a MapQuest[szerzo, ev] és OpenCycleMap[Allan, 2007] API-król illetve az Abstract Factory tervezési mintáról.*

2.1. Vaadin

Az alkalmazás webes felületét a *Vaadin* keretrendszer segítségével valósítottuk meg. Ez egy nyílt forráskódú webalkalmazás-keretrendszer, amellyel interaktív web tartalom készíthető *Java* nyelven és a hagyományos *GUI* (grafikus felhasználói felület) fejlesztéshez hasonlítható.

A *Vaadin* megjelenítésre a *Google Web Toolkit*-et (AJAX fejlesztői eszköztár) használja, míg a szerver oldal alapját a *Java Servlet* (Java objektum, mely *HTTP*[1] kérést dolgoz fel és *HTTP* választ generál) technológia képezi. Maga a kódolás *Java* nyelven történik, a *GWT*(Google Web Toolkit) ezt *Javascript* forráskódra alakítja át, ami a böngészőben kerül majd futtatásra. A *GWT* csupán egy vékony megjelenítési réteg a Vaadin esetében, mivel az alkalmazás logika a szerver oldalon helyezkedik el teljes mértékben. A kommunikációra *AJAX* (Asynchronous JavaScript and XML) technológiát használ, míg az adatok *JSON* (Javascript Object Notation) szabvány szerint vannak kódolva.

A *Vaadin* előnyére szolgál az, hogy komponensei kiegészíthetők *GWT*-Widgetekkel illetve lehetőség van *CSS*-el (Cascading Style Sheets) való formázásra is, ami elengedhetetlen egy felhasználóbarát felület megalkotásában. Mivel az *Eclipse* rendelkezik megfelelő beépített *Vaadin* modulokkal, ez is a fejlesztés javára szolgált. A *GWT* fordítónak köszönhetően a legtöbb modern böngészővel kompatibilis. Mivel ezen keretrendszer esetében szükség esetén összekapcsolható a *Java* a *Javascript* kóddal, a *Vaadin* ebből a szemszögből is egy helyes választásnak bizonyult. A Chain-Guide alkalmazás keretén belül így könnyebben hozzá tudtunk férni a térképszolgáltató függvényeihez a *MapQuest Javascipt API*-ján keresztül.

A *Vaadin 7* és az annál újabb verziók (a megvalósított webalkalmazás a 7.2.4-es verziót hasz-

1. Hypertext Transfer Protocol

nálja) a következő webböngészőkkel kompatibilisek:

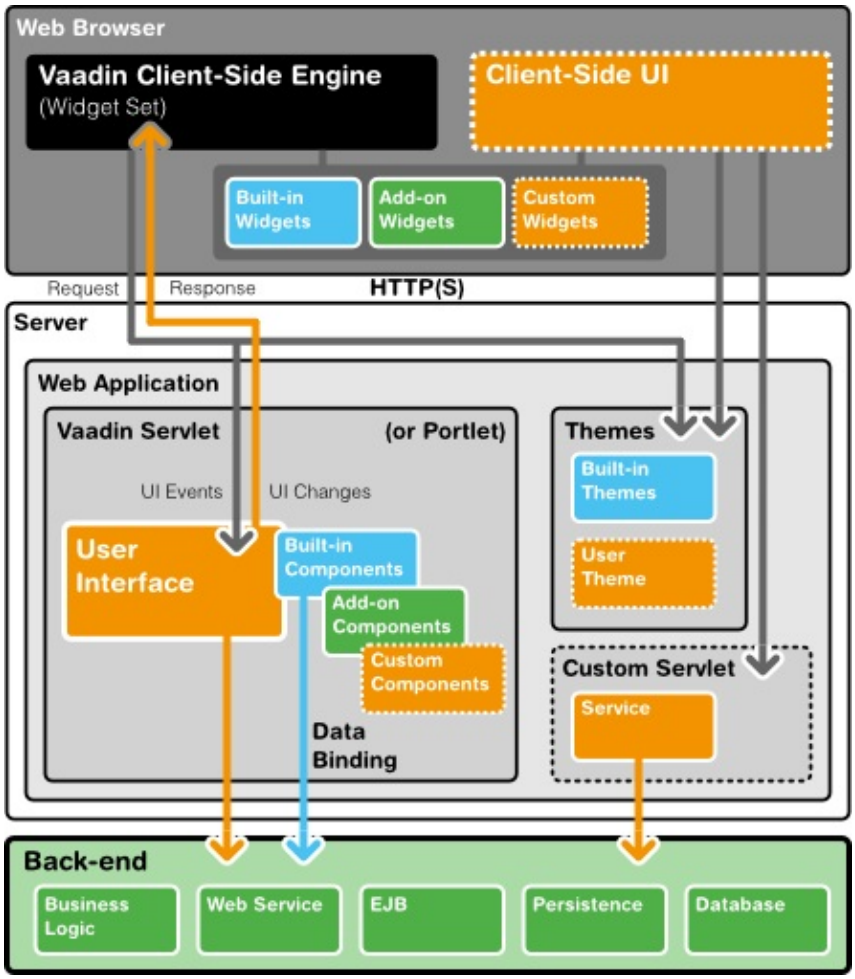
- Google Chrome 23 vagy újabb
- Internet Explorer 8 vagy újabb
- Mozilla Firefox 17 vagy újabb
- Opera 12 vagy újabb
- Safari 6 vagy újabb

A *Vaadin* felépítését illetően a szerver oldali architektúrára kerül a hangsúly, mivel az alkalmazások logikájának nagy része a szerveren fut. Csak a szükséges elemeket helyezi a keretrendszer kliens oldalra, ahol *AJAX* tecnológiát és *Google Web Toolkit*-et használ. A fejlesztés megkönnyítése céljából, elrejti a kliens-szerver architektúrát a fejlesztők elől, e kettő között lévő kommunikációs problémákat is megoldva. A *Vaadin* keretrendszer maga dönt arról, hogy mi kerül majd kliens és mi szerver oldalra(önmaga oldja meg a kód szétválasztását), ezért hasonlít maga a kódolás egy sima desktop alkalmazáséhoz. A *Vaadin* szerver oldali validációt alkalmaz minden egyes műveletnél, így a biztonsági problémák megoldása is rá van bízva.

Egy *Vaadin* alkalmazás futás közbeni architektúrája a 2.1 ábrán látható.

Az ábra a kliens és szerver oldalak közti kommunikációt szemlélteti egy futás közbeni állapotban, amikor a kliens oldali kód már be van töltődve a böngészőbe.A szerver oldali *Vaadin* alkalmazás `servlet`-ként fut egy *Java* web szerveren, kiszolgálva a *HTTP* kéréseket. A *Vaadin Servlet* osztály fogadja a kliens kéréseit és nekik megfelelő eseményekként továbbítja őket az alkalmazásban megadott esemény-figyelőkhöz(listener) a *UI*-on belül. A kliens oldali motor(engine) mely a böngészőben fut, fogadja a kéréseket és végrehajta a kért változtatásokat a weboldalon.

A Chain-Guide alkalmazás keretén belül a *GUI* felépítése a `com.vaadin.ui.UI` absztrakt osztály kiterjesztésével lett megvalósítva, ez képviseli az alkalmazás belépési pontját amikor az alkalmazás url-jét beírják a böngészőbe. A különböző webtartalom megjelenítéséhez különböző nézetek(view) lettek felhasználva, a `com.vaadin.navigator.View` megvalósítása által. A nézetek közti navigáció a `com.vaadin.navigator.Navigator` osztály segítségével lett megvalósítva, amihez szükséges volt az adott nézet regisztrációja is. Az alkalmazáson belül a navigátor bárhonnan elérhető a `UI.getCurrent().getNavigator()` metódus meghívásával.



2.1. ábra. Egy *Vaadin* alkalmazás futás közbeni architektúrája.

2.2. Hibernate

Az alkalmazás backend része a *Hibernate* programkönyvtár segítségével lett kivitelezve. A *Hibernate* egy *ORM* (objektum-relációs leképezést megvalósító) keretrendszer *Java* platformra, melynek legfőbb célja az adatbázissal történő kommunikáció leegyszerűsítése. Segítségével az adatbázisban lévő rekordokat objektumként kezelhetjük és állapotmegörző módon adattáblákban tárolhatjuk. Legfőbb jellemzője ezek mellett, hogy adatbázis függetlenséget biztosít.

A *HQL* (Hibernate Query Language) a *Hibernate* saját adatlekérdező nyelve, mely lehetőséget teremt lekérdezések írására és futtatására (*SQL* tudás nélkül). A keretrendszer ezen *HQL* lekérdezésekből generálja az adatbáziskezelő rendszer számára megfelelő *SQL* (Structured Query Language) lekérdezéseket. Így, a fejlesztők előnyére, megkíméli őket az eredményhalmazok objektumokra történő konverziójától.

Az adattáblák és osztályok közti leképezéseket vagy mappinget *XML* (Extensible Markup Lan-

guage), esetleg *Java* annotációk segítségével valósítja meg.

```
<hibernate-mapping>
    <class name="edu.ubbcluj.backend.model.Rating" table="rating" catalog="bike" optimistic-
        lock="version">
        <id name="id" type="int">
            <column name="id" />
            <generator class="identity" />
        </id>
        <many-to-one name="services" class="edu.ubbcluj.backend.model.Services" fetch="select"
            >
            <column name="serviceId" />
        </many-to-one>
        <many-to-one name="users" class="edu.ubbcluj.backend.model.Users" fetch="select">
            <column name="userId" />
        </many-to-one>
        <property name="rate" type="java.lang.Integer">
            <column name="rate" />
        </property> }
    </class>
</hibernate-mapping>
```

Az fenti példában a Rating.hbm (Hibernate Mapping File) állomány tartalma látható. Ez az XML file az Értékelés(Rating) adattábala és a neki megfelelő modell osztály között teremti meg a kapcsolatot. A `<generator class="identity" />` tag az egyedi azonosító generálására szolgál, amely az `id` nevezetű, elsődleges kulcs típusú adattagot jellemzi. A `many-to-one` tag név az egy-a-többhöz kapcsolat leírására szolgál míg a `property` név alatt az olyan tábla adattagokat adjuk meg, melyek nem állnak kapcsolatban más táblák mezőivel.

A beépített „dirty check”[2] is pozitívumként emelhető ki, hiszen megakadályozza a felesleges beszúrásokat az adatbázisba.A *Hibernate* esetében két féle betöltési módról beszélhetünk: lusta betöltés[3] és mohó betöltés. Lusta betöltés esetén csak akkor fut le a lekérdezés, amikor először hivatkozunk az objektumra, míg a mohó esetén az már az objektum betöltésekor. Átlátható módon biztosítja a *Plain Old Java Object*-ek (POJO) perzisztenciáját a felhasználók számára (az egyetlen követelmény, hogy az osztálynak legyen egy argumentum nélküli konstruktora).

2.3. Apache Maven

A projekt moduljainak egyszerű menedzselését a *Maven* szoftver biztosította, melynek legfőbb célja az összeállítási (build) folyamatok automatizálása. Előnyére szolgál, hogy dinamikusan is le tud tölteni komponenseket, szoftver-csomagokat, ha szükséges. Egy *XML* file (POM) segítségével adhatjuk meg, hogyan legyen a projekt felépítve, milyen sorrendben legyenek buildelve a különböző modulok, illetve, hogy milyen külső függőségeket, pluginokat, komponenseket használjon. A buildelés szabványosítása által a tervezési minták terjesztése a célja.

Az alábbi példában egy részlet látható az alkalmazás backend[4] részének a pom.xml állomá-

2. *Hibernate* jellemzője, a keretrendszer leellenőrzi, hogy egy adott objektumon történt-e változás vagy sem, és ha igen, csak akkor hajtja végre a frissítést(update)
3. lazy loading
4. adat elérési réteg

nyából. A részletben függőségként a textitHibernate és textitMySQL konnektorok láthatóak, amiket az adatbázissal történő kommunikációra használ a rendszer.

```
<dependencies>
        <dependency>
                <groupId>org.hibernate</groupId>
                <artifactId>hibernate-core</artifactId>
                <version>4.3.8.Final</version>
        </dependency>
                <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-java</artifactId>
                        <version>5.1.34</version>
                </dependency>
  </dependencies>
```

Az összeállítási (build) folyamat automatizálására az alábbi példa emelhető ki. A példában a `<build> tag`-ek közé a *CSS* állományok automatikus lefordítását és frissítését kérjük a rendszertől a projekt build-elésével együtt.

```
        <build>
                <finalName>bike-web</finalName>
                <plugins>
                        <plugin>
                <groupId>com.vaadin</groupId>
                <artifactId>vaadin-maven-plugin</artifactId>
                <version>7.2.4</version>
                <executions>
                    <execution>
                        <goals>
                            <goal>clean</goal>
                            <goal>resources</goal>
                            <goal>update-theme</goal>
                            <goal>compile-theme</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
                </plugins>
        </build>
```

2.4. MapQuest

Az alkalmazás esetében a térképpel kapcsolatos információkat és függvényeket a *MapQuest* szolgáltatta. Ez egy amerikai ingyenes online térkép szolgáltatás, mely hazánkban is elérhető, és a webes desktop és mobil alkalmazásokat is egyaránt támogatja. A különböző API-k és szolgáltatásai révén egyszerűen integrálható. A fejlesztők számára szükséges egy *AppKey* (Aplication Key), egy egyedi kulcs, mely által a *MapQuest* szerverei azonosítani tudják az alkalmazásunkat, annak érdekében, hogy helyes válaszokat térítsenek vissza kéréseinkre. Ez ingyenesen igényelhető regisztráció[5] által. A *MapQuest* út-, közlekedés- és forgalommal kapcsolatos információit alapértelmezetten az *OpenStreetMap* (szabadon szerkeszthető és felhasználható térkép) szolgáltatja. E térkép kerékpár rétege az *OpenCycleMap*, amely biciklis szempontból hasznos információkat szolgáltat világszerte, beleértve Romániát is.

A `javacript` állományok integrációjáról (*Vaadinba* való beágyazásáról) a referencia oda!! részben található egy részletesebb leírás. Röviden összefoglalva a *MapQuestJavascript API*-t nem

5. http://developer.mapquest.com/fr/web/info/account/app-keys

szükséges letölteni mint különálló `javascript` állomány, csupán az elérési útvonalat kell megadni `javasciptes annotáció` segítségével (az `AppKey` -el együtt), hasonlóan a saját `javascipt` állományok betöltéséhez.

```
import com.vaadin.annotations.JavaScript;

@JavaScript({"http://open.mapquestapi.com/sdk/js/v7.2.s/mqa.toolkit.js?key=APPKEY","mylib.js"
    })
```

A *Javascript Maps API* egyike a legelterjedtebb *MapQuest*-es szolgáltatásoknak. Funkcionalitásait illetően lehetőséget nyújt térképes felületek létrehozásához különböző extra opciókkal (live traffic, self-localization stb.), vannak beépített útkereső függvényei, melyeknek paraméterként a `bicycle` kulcsszót megadva biciklibarát útvonalak rajzoltathatóak ki a térképre. A `geocoding` modul átjárhatóságot biztosít a koordinátákat tartalmazó `LatLng` objektumok és a direkt módon megadott címek közt, melyek ugyanazt a pontot határozzák meg a térképen. A különböző eseménykezelő függvényeivel interaktívabbá varázsolhatók az alapműveletek, illetve a térkép objektumok is felülírhatók, személyre szabhatóak, egy felhasználóbarát felület kialakításának érdekében.

Az alkalmazás legtöbbet használt moduljaként a `Geocoding` modul emelhető ki. Konkrétabban a `geocodeAndAddLocation` és a `reverseGeocodeAndAddLocation` függvények hangsúlyozhatók ki, melyek segítségével a `LatLng` objetumokból valós címek nyerhetők és jeleníthetők meg a térképen illetve fordítva. Ezek keretén belül a `POI`[6] objektumok is személyre szabhatóak, változtatható az ikonjuk, info-ablakuk, illetve felülirhatók a rájuk értelmezett események is(kattintás, mozgatás stb.). Az alkalmazás elengedhetetlen része a minden oldalon megjelenő térkép objektum amelyet az `MQA` modul `TileMap` függvényének meghívásával rajzolhatunk ki a paraméterként megadott opciókkal. Ezen paraméter egy olyan adat struktúra, melyben megadható, hogy hova töltődjön be a térkép, mi legyen a középpontja, mekkora legyen az alapértelmezett közelítés, stb. A `Routing` modul is az alkalmazás alapjait képezi, hiszen elengedhetetlen két pont közötti útvonal megjelenítéséhez. Az `AddRoute` függvénynek megadhatóak úgy `LatLng` objektumok mint címek. Az `options` struktúrában meghatározható a keresés típusa, például `bicycle`, amely egy olyan útvonalat jelenít meg A és B pontok között mely a legbiciklibarátabbnak nevezhető (ahol lehet a kerékpár utatakat veszi, ha pedig nincs igyekszik találni olyan kisebb utcákat, amely elkerüli a forgalmas útszakaszokat, figyelembe véve a közlekedési szabályokat). A `shortest` opcióval a fizikai értelemben vett legrövidebb útszakaszt kapjuk válaszként. Az útkeresés típusa mellett megadhatók még a megjelnítésre vonatkozó extra opciók is, amellyel rövid útmutatót is megjeleníthetünk az adott útvonalra.

(majd utólag beírom a hivatkozást a gyakorlati rész megfelelő fejezetére, ahol a konkrét példák lesznek).

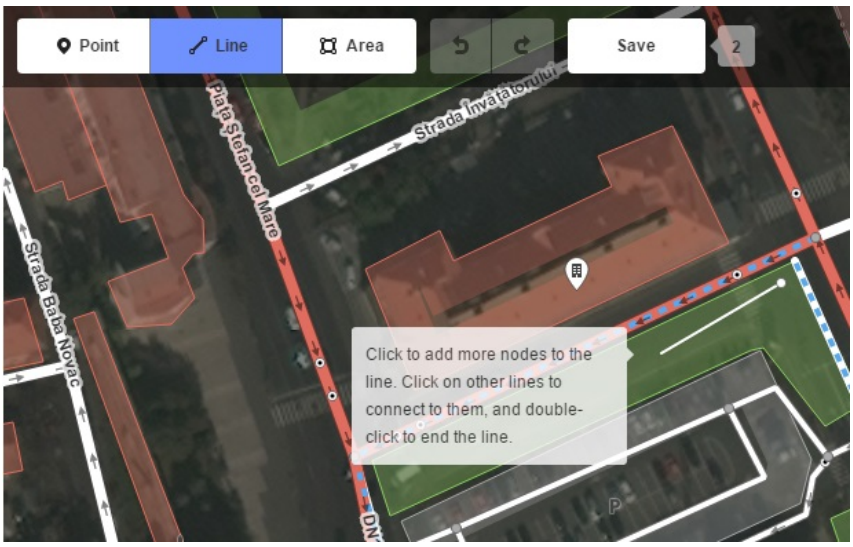
6. a térképen egy pontot megjelenítő objektum

2.5. OpenCycleMap

Az *OpenCycleMap* az *OpenStreetMap* térképes szolgáltatás egy rétege (layer). Az *OpenStreetMap* (OSM) egy szabadon szerkeszthető és felhasználható térképfejlesztés. A térképek egyszerű helyismeretből vagy hordozható `GPS` eszközökből, légifotókból származó adatokra épülnek, amelyeket az `Open Database License` (nyílt adatbázis) tárol. A regisztrált felhasználóknak lehetőségük van szerkeszteni a vektor alapú adatokat illetve `GPS` nyomvonalakat is feltölthetnek. A romániai adatok szempontjából a legjobban aktualizált térképszolgáltató, illetve biciklis információk terén a legjelentősebb.

A biciklis réteg tartalmazza az összes nemzetközileg elismert bicikliutat illetve a lokális és regionális kerékpár utak javát, ezen kívül megjeleníthetők túraútvonalak, bicikli üzletek és parkolók is egyaránt, ahol azokat a felhasználók hozzáadták a térképhez. Előnyére szolgál, hogy a változtatások (pl. ha egy új bicikli utat szeretnénk hozzáadni), 24 órán belül bekerülnek az adatbázisba, és egy-két napon belül láthatóvá válik mindenki számára (a 2.2 és a 2.3 képek, egy általam hozzáadott bicikliutat és szerkesztési folyamatát ábrázolják).

2.2. ábra. Egy általam hozzáadott bicikliút.



2.3. ábra. A bicikliút szerkesztési folyamata.

Irodalomjegyzék

Allan, A. Open cycle map, 2007. URL http://www.thunderforest.com/opencyclemap/.

Grönroos, M. Book of vaadin, 2013. URL https://vaadin.com/book.

szerzo, . Hibernate - relational persistence for idiomatic java, 2015. URL http://docs.jboss.org/hibernate/orm/4.3/manual/en-US/html/.

szerzo, . Mapquest javascript api 7.2, ev. URL http://developer.mapquest.com/web/documentation/sdk/javascript/v7.2/api/index.html.