XVII. reál- és humántudományi Erdélyi Tudományos Diákköri Konferencia (ETDK)
Kolozsvár, 2014. május 15–18.

Közlekedési információk bejelentését és követését támogató szoftverrendszer

Szerzők:
Balázsi Beáta
Babeş-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, informatika szak, III. év
Kardos Örs-Tihamér
Babeş-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, informatika szak, III. év
Ráduly Sándor
Babeş-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, informatika szak, III. év

Témavezető:
dr. Simon Károly egyetemi adjunktus,
Babeş-Bolyai Tudományegyetem, Kolozsvár,
Matematika és Informatika Kar,
Magyar Matematika és Informatika Intézet

Kivonat

A dolgozatban bemutatott Sparrow projekt célja egy közeledéssel kapcsolatos információk megosztására és követésére alkalmas szoftverrendszer kifejlesztése. A rendszer részét képezi egy szerveralkalmazás, egy események (sebességmérés, forgalmi dugó, baleset, veszélyes útszakasz stb.) bejelentésére és térképen történő megjelenítésére alkalmas mobilalkalmazás, egy webes adminisztrációs felület, valamint egy adatelemzési célokat szolgáló webes alkalmazás.

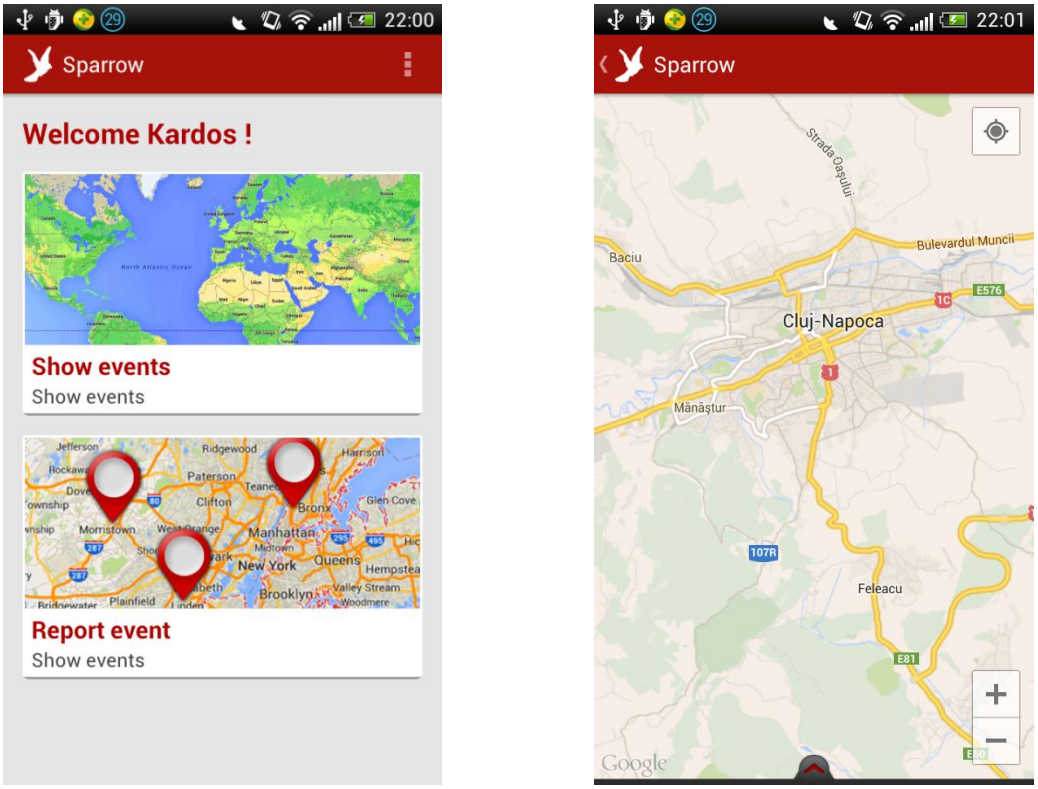
A legtöbb hasonló rendszerrel ellentétben, az alkalmazás az előre megadott kamera adatbázisok mellett nagy hangsúlyt fektet a felhasználói közösség által valós időben bejelentett eseményekre. Számos hasznos járulékos funkcionalitást is biztosít, például vizuális és auditív figyelmeztetést ad közeledő eseményeknél, valamint adatszűrési lehetőségeket támogat. A mobil kliensalkalmazás felülete intuitív, vezetés közben is kényelmesen és biztonságosan használható.

Tartalomjegyzék

Bevezető

A Sparrow projekt célja egy olyan szoftverrendszer megvalósítása, amelynek segítségével a felhasználóknak lehetőségük nyílik a környezetükben lévő forgalmi eseményeket egy mobilalkalmazáson keresztül bejelenteni (például: úttorlasz, baleset, sebességmérés, forgalmi dugó stb.) és a környezetükben lévő hasonló eseményekről értesülni.

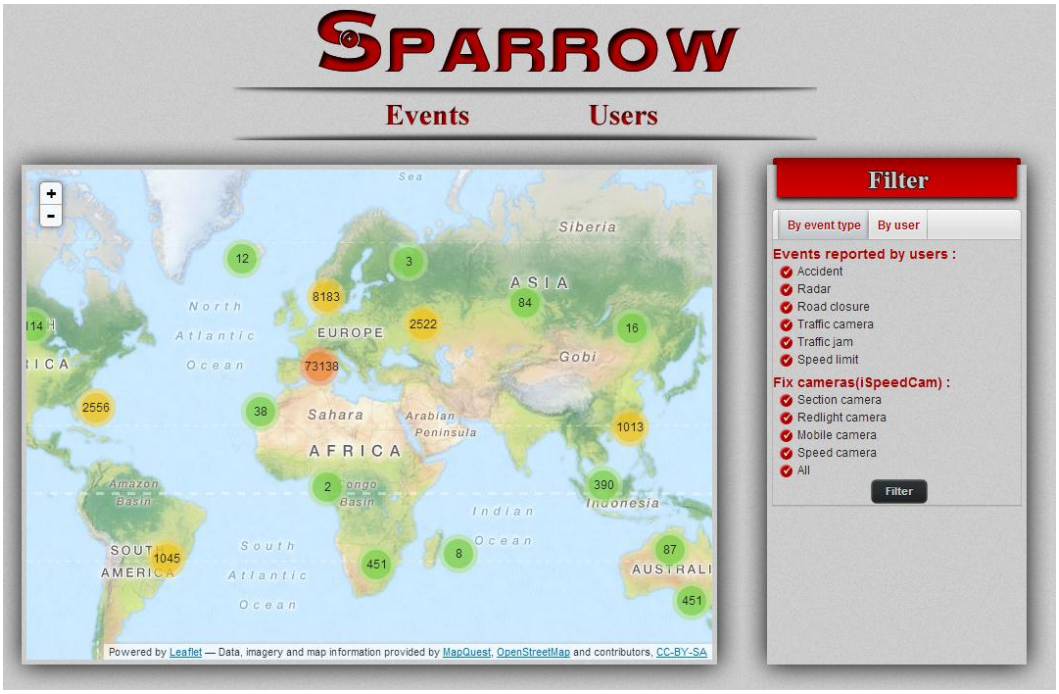
Tekintsünk egy példát: egy autóvezető szeretne zökkenőmentesen átutazni egy városon (például nem szeretne forgalmi dugóba kerülni). Úgy dönt, hogy kikerüli a város főterét. Egy másik autóvezető a város kerülőútján forgalmi dugóba kerül, így az alkalmazást használva jelzi ezt. Az első autóvezető értesítést kap erről, mobilalkalmazásának térképnézetében megjelenik az új esemény. Így előre látja, hogyha a kerülőút zsúfolt (1. ábra, jobb oldali képernyő) és más útvonalat választ.



1. ***ábra:*** *bejelentkezés utáni felület és az eseményeket megjelenítő térképnézet a Sparrow mobilalkalmazáson belül*

A felhasználóknak lehetőségük van visszajelzéseket is küldeni az eseményekkel kapcsolatosan. Megerősíthetnek egy adott eseményt, jelezhetik adott bejelentés valótlanságát, illetve megjegyzéseket fűzhetnek az eseményekhez. A mobilalkalmazás fejlesztése során fontos cél volt felhasználóbarát és szoftver-ergonómia szempontból megfelelő felület, tekintettel arra, hogy az alkalmazást többségben autóvezetők fogják használni, vezetés közben.

A Sparrow szoftver rendelkezik egy adminisztrációs felülettel is (2. ábra), amely segítségével megtekinthetőek a felhasználók adatai, útvonalai, eseményei, illetve menedzselhetőek a rendelkezésre álló adatforrások. Az adatok térképen jeleníthetőek meg és a megjelenítés optimalizálásának (események klaszterezése) érdekében a szoftver map reduce módszert is alkalmaz. Az adminisztrációs felületen is lehetőség van különböző kritériumok szerint szűrni az információkat (pl. felhasználók vagy eseménytípusok szerint).



2. ***ábra:*** *adminisztrációs felület*

A rendszer részét képezi egy adatfeldolgozással kapcsolatos felület is, ahol az adminisztrátorok külső adatforrásokkal tudnak dolgozni, optimalizálási műveleteket tudnak futtatni az adathalmazokon (szűrés, összefésülés), illetve a műveletek eredményeit térképén jeleníthetik meg.

A Sparrow ötletének kidolgozásakor egy kiindulási pontként szolgált a Codespring Kft. által fejlesztett iSpeedCam alkalmazás, amely egy adott kamera adatbázis alapján figyelmeztet felhasználókat. Az ötletet továbbfejlesztve, megszületett egy olyan alkalmazást terve, amelynek segítségével a felhasználók valós időben is bejelenthetik a forgalommal kapcsolatos eseményeket.

A projekt fejlesztésének koordinációját a Codespring Kft. vállalta, Dr. Simon Károly egyetemi adjunktus vezetésével és Török-Vistai Tamás projektmenedzser szakmai segítségével.

A projekt fejlesztésében a Babeş-Bolyai Tudományegyetem három harmadéves hallgatója vett részt: Balázsi Beáta, Kardos Örs-Tihamér és Ráduly Sándor.

1. Alkalmazott módszerek és felhasznált eszközök

A szoftverfejlesztés során a fejlesztők munkáját segítik a verziókövető rendszerek, a build és függőségmenedzsment eszközök, a fejlesztői környezetek, a web- és alkalmazásszerverek, a projektmenedzsment és hibakövető rendszerek, a statikus verifikációt és kódelemzést végző szoftverek és további eszközök.

A verziókövetők lehetőséget adnak a forráskód változásainak archiválására és követésére, különböző verziók kezelésére, így erőteljesen támogatják a csapatmunkát. Segítségükkel pontosan nyomon követhető a fejlesztési folyamat. A Sparrow szoftverrendszer fejlesztése során a csapat a Mercurial [1] rendszert használta. A választás azért esett a Mercurialra, mert egy osztott verziókövető rendszer, számos előnye van a hagyományosabbnak számító központosított rendszerekkel szemben (a változtatások többször beküldhetőek a lokális tárolóba, hatékonyabban támogatottak az összefésülési műveletek, nem szükséges minden változtatáshoz internet kapcsolat és biztonsági szempontból is előnyt jelent a több tároló). Asztali kliensalkalmazásként a csapat a TortoiseHg-t használta, a központi tároló menedzsmentjét a RhodeCode rendszer biztosította.

A szoftverfejlesztési folyamat során nagyon fontos a jó időbeosztás, illetve a feladatok megfelelő elosztása és nyomon követése. A projektmenedzsment rendszerek lehetőséget adnak feladatok meghatározására, ezek fontossági sorrendjének megadására, és a feladatok hozzárendelhetőek fejlesztőkhöz. Hasonló módon a tesztelések és ellenőrzések során talált hibák is követhetőek, javításuk fejlesztőkhöz rendelhető. A Sparrow fejlesztése során használt Redmine ingyenes, nyílt forráskódú projektmenedzsment és hibakövető rendszer, amely egy webes felületet biztosít a fejlesztők számára. A legtöbb verziókövető rendszert támogatja, lehetőséget ad feladatok és hibák követésére és menedzselésére, valamint beépített Gantt diagramot, naptárat, wikit biztosít.

Összetett, több modulból álló projektek esetében nagy segítséget jelentenek a fejlesztők számára a különböző build és függőségkezelő eszközök. A programozók számára nehézséget okozna külön lefordítani a projekt egyes részeit, illetve külön kezelni a függőségeket, ezért ajánlott egy ilyen eszköz használata a fejlesztés során. A Java programozási nyelvben írt alkalmazások számára több build eszköz is rendelkezésünkre áll. A Sparrow projekt esetében az

Apache Maven-re [2] esett a választás, mivel az egyik legnépszerűbb build eszköz és függőség menedzsmentet is biztosít. A Sparrow saját repository-t is használt, melynek menedzsmentjét az Artifactory biztosította.

A szoftverek minőségének biztosítását kódelemző szoftverek segíthetik. A fejlesztés során hangsúlyt kell fektetni a kód helyességére, illetve bizonyos konvenciók és minták betartására az optimális, egységes és könnyen átlátható kód érdekében. Egy kódelemző szoftver feladata a kódban szereplő esetleges hibák kiemelése, illetve a fejlesztők figyelmeztetése és segítése a kód javításában. A Sparrow szoftverrendszer fejlesztése során használt kódelemző eszköz a SonarQube volt. A SonarQube egy Javaban fejlesztett rendszer, amely jelenleg több mint 25 programozási nyelvet támogat. Az ellenőrzés több szempont szerint zajlik: duplikátumok ellenőrzése, tesztlefedettség, komplexitás és kódolási konvenciók ellenőrzése, hibák kiszűrése adott szabályrendszer alapján. A hibák esetében a rendszer javaslatot is tesz a javításra.

A fejlesztés során használva volt a Jenkins nevű folytonos integráció szerver, amely támogatja az Apache Ant és Apache Maven alapú projekteket.

A projekt iteratív fejlődését a Scrum fejlesztési módszer alkalmazása segítette elő.

A fejlesztés során a NetBeans fejlesztői környezetet használták a fejlesztők, illetve az Android Studio-t az Android kliensalkalmazás fejlesztésére. Web szerverként az Apache Tomcat [3] szolgált. Az adatbázisban levő adatok megtekintésére és kezelésére a Robomongo eszköz volt használva. A felhasználói felületek megtervezésére, vázlatok elkészítésére a Balsamiq mockup eszköz szolgált.

2. Felhasznált technológiák

A Sparrow szerver oldala a Spring keretrendszerre épül, a projekt MongoDB adatbázist használ, a webes felület Vaadin keretrendszer segítségével van felépítve, az Android kliensalkalmazás RESTful webszolgáltatásokon keresztül kommunikál a szerver oldallal.

2.1. A Spring keretrendszer

A Spring [4][5] egy platformfüggetlen, nyílt forráskódú, Java programozási nyelven alapuló alkalmazás keretrendszer, amely egy Inversion of Control (IoC) konténeren keresztül biztosítja a komponensek menedzsmentjét és a Dependency Injection (DI) minta alkalmazásának lehetőségét. Rod Johnson ausztrál származású számítógép-szakértő kezdte meg fejlesztését a kétezres évek elején, aki később a SpringSource vállalat egyik társ-alapító tagja lett. A keretrendszer 2003 júniusában volt először kiadva Apache 2.0 licenc alatt, de csak 2004

márciusában volt kiadva az 1.0-ás verzió. A keretrendszer legfrissebb stabil kiadása a 4.0.3-as verzió, amelyet 2014. márciusában adtak ki.

A Spring keretrendszer használata által megszűntethetőek a projekten belüli modulok közötti szoros függőségek, az alkalmazás könnyen karbantarthatóvá válik, így a keretrendszer megkönnyíti a programozók munkáját.

Tulajdonképpen a Spring keretrendszerek gyűjteménye, számos könnyen konfigurálható és integrálható modult biztosít. A teljes Spring projekt hozzávetőlegesen húsz keretrendszerből tevődik össze, ezek a core konténer, adathozzáférés/integráció, web, aspektusorientált programozás (AOP), alkalmazás menedzsment (instrumentation) és teszt kategóriákba vannak csoportosítva.

A *Core* és *Beans* modulok, amelyek a core konténer részei, biztosítják a keretrendszer legalapvetőbb funkcióit, az IoC konténert és a működéséhez elengedhetetlen DI funkciót. A klasszikus programozási modellekben a vezérlés és végrehajtás folyamata (control flow) a kód által van irányítva. Például, az egyszerű objektumorientált rendszerek esetében mind az objektumok létrehozása, mind az objektumok közötti kapcsolatok kialakítása a programozók feladata. Az IoC elv alapján működő keretrendszerek átveszik a folyamat irányítását. Az IoC egyik formája a DI, amelynek megfelelően az objektumok menedzsmentje a konténert biztosító keretrendszer feladata és a komponensek közötti függőségeket is a keretrendszer kezeli.

A Spring core modulon kívül a Sparrow a Spring Data keretrendszert is használja, amely egy absztrakciós szintet biztosít az alkalmazás adathozzáférési rétege számára, így téve lehetővé az adatbázis könnyű cserélhetőségét anélkül, hogy a felette levő rétegeket változtatni kelljen. A Sparrow alkalmazásszerver a Spring DATA – MongoDB verzióját használja adathozzáférési rétegén belül.

2.2. MongoDB

Nagy, hierarchikus adathalmazok használata, vagy változó séma esetén a relációs adatbázisok már nem biztos, hogy biztosítani tudják a megfelelő gyorsaságot és hatékonyságot, illetve flexibilitást. Az ilyen esetekben jó alternatívát jelenthetnek a NoSQL adatbáziskezelő-rendszerek.

A MongoDB [6] egy platformfüggetlen, nyílt forráskódú dokumentumorientált NoSQL adatbázis, amely nagy teljesítményű, képes hatalmas adatmennyiségekkel gyorsan és hatékonyan dolgozni. Mint NoSQL adatbázis, a MongoDB nem a hagyományos relációs adatbázis szerkezetet támogatja, hanem JSON formátumon alapuló BSON formátumú objektumokkal és

dinamikus sémákkal dolgozik. Ennek köszönhetően a Java modell objektumok könnyen leképezhetőek az adatbázisba, beleértve a kapcsolatokat és hierarchiákat is, nem szükségesek a relációs adatbázisoknál megszokott JOIN műveletek. Az objektumok közötti függőségek esetében referenciák használatára is lehetőséget ad, így kiküszöbölhető a duplikátumok tárolása az adatbázisban. Számos további hasznos funkcionalitást biztosít, például támogatja a 2D indexelést, a térinformatikai lekérdezéseket és földrajzi koordinátákon alapuló távolsági számítások elvégzését.

A Sparrow projekt adatbáziskezelő-rendszereként a MongoDB tűnt a legjobb választásnak, mivel képes kezelni a folyamatosan felgyülemlő adatmennyiséget, illetve rendelkezik a 2D-s indexelési lehetőséggel, valamint támogatja a földrajzi pontokkal kapcsolatos lekérdezéseket és számításokat.

A Sparrow projekt webes felhasználói felületének implementálásához nagy mennyiségű adat feldolgozására és megjelenítésre volt szükség. A MongoDB lehetőséget biztosít az adatok kötegelt feldolgozására, a MapReduce módszer használatára, így a Sparrow projekt is ezt alkalmazza. A MapReduce egy megadott szempont szerint csoportosítja az adatokat és a hasonló adatokat egy csomóponttá vonja össze, ezzel jelentősen csökkentve az adatmennyiséget.

A módszer két függvényen alapszik, a *map* és a *reduce* függvényeken, de ha szükség van az adatok szerkezetének módosítására, egy *finalize* függvény is alkalmazható. A *map* nevű függvény kulcs-érték párokat fogad, amiből új kulcs-érték párokat gyárt. Annak függvényében adja az új kulcsokat, hogy milyen típusú adatokat szeretnénk egy csoportba összevonni. A *reduce* függvény a *map* függvény kimenetét kapja bemenetként és az azonos kulcsú értékeket összevonja egy halmazba. A *finalize* függvényt abban az esetben érdemes használni, ha a *reduce* függvény kimenetén további átalakításokat kell elvégezni a helyes eredmény biztosításához.

2.3. Az Android platform

Az Android [7][8] egy Linux kernelre épülő operációs rendszer, amelyet a Google, az Open Handset Alliance és az Android Open Source Project közösen fejleszt. Az alkalmazások Java-ban, az operációs rendszer magja és a különböző könyvtárak C-ben és C++-ban vannak megírva. Elsősorban okostelefonokra és tabletekre van tervezve, legfrissebb stabil verziója a 4.4.2 KitKat, amelyet 2013 decemberében adtak ki.

Ahhoz, hogy az Android platformra fejlesszünk, szükségünk van az Android szoftverfejlesztői csomagra (Android SDK), amely számos fejlesztési eszközt tartalmaz (debugger, könyvtárak, emulátor stb.).

Mivel az operációs rendszer nagyon sok készüléktípust támogat, amelyek API szintje, képernyőmérete, felbontása és pixel-sűrűsége nagyban különbözik egymástól, külön figyelmet igényel, hogy az alkalmazás egységes módon működjön bármilyen készüléken. Ebben segítenek az Android Support Library könyvtárcsomagok, amelyek lehetővé teszik új design módszerek használatát régebbi készülékeken, illetve kiegészítő csomagokat tartalmaznak, amelyek nem képezik részét az alap platformnak. Fontos továbbá a tervezési alapelvek követése, amelyekre a Sparrow is hangsúlyt fektet.

Az Android alkalmazások felhasználói felületének alapját Activity-k képezik, amelyek a 3.0-ás verziótól fragmentek-kel (nagyobb képernyők támogatása, illetve dinamikus UI) egészültek ki. Habár kódból is beállíthatóak a különböző nézetek, ajánlott ezeket XML állományokban meghatározni, ezáltal is jobban szétválasztva a nézetet az üzleti logikától.

Annak érdekében, hogy a felhasználói felület használata gördülékeny maradjon, a Sparrow projekt mobil alkalmazásában Service-k vannak használva. A Service-k olyan komponensek, amelyek lehetővé teszik hosszas műveletek elvégzését külön szálakon. Ezek a műveletek akkor is el lesznek végezve, ha már nem a szolgáltatást elindító alkalmazás van előtérben. Ugyancsak a felhasználói felület gördülékeny működése érdekében úgynevezett AsyncTask-eken keresztül van megoldva a Sparrow projektben a szerverrel való kommunikáció is, így szintén egy háttér szálon fut. A Servicek-kel ellentétben az AsyncTask-eket ideális esetben csak rövid, néhány másodperces műveletek elvégzésére ajánlott használni.

Egy Android alkalmazásban az adatok SQLite adatbázisban tárolhatóak, ennek érdekében az alap csomag tartalmaz bizonyos osztályokat, amelyek lehetővé teszik a saját adatbázisok menedzselését.

2.4. A Vaadin keretrendszer

A Vaadin [9][10] egy web-alkalmazás keretrendszer, amelynek célja minőségi webes felhasználói felületek létrehozása és karbantartása hatékony és gyors módszerekkel. A Vaadin szerver oldalon Servlet technológiát alkalmaz, míg a kliens oldal a Google Web Toolkit (GWT) fejlesztői eszköztárra épül.

Lehetőséget nyújt interaktív felhasználói felületek készítésére, a programozó Java nyelvben építheti fel felületet, majd a háttérben HTML/JavaScript kód generálódik, amely később a böngészőben kerül futtatásra.

A keretrendszer egy átfogó komponens készletet biztosít a felületek felépítéséhez és lehetőség van egyéni komponensek létrehozására és használatára is. A komponensek elhelyezkedését a tárolókon belül elrendezés menedzserek szabályozzák.

A Sparrow Admin és DataMining modulok webes felületei Vaadin keretrendszer segítségével készültek.

2.5. RESTful webszolgáltatások

A Sparrow szoftverrendszeren belül a mobil kliensalkalmazás szerverrel való kommunikációja a REST architektúrán alapszik, RESTful webszolgáltatásokon keresztül valósul meg. A kommunikáció megvalósításának érdekében a projekt a JAX-RS (Java API for RESTful Services) szabvány referencia implementációját, a Jersey keretrendszert alkalmazza. A Jersey ugyanakkor több, mint JAX-RS implementáció, számos további hasznos funkcionalitást biztosít.

Az adatok küldése a kliens és a szerver között JSON formátumú objektumok segítségével valósul meg. A JSON szerializáció és deszerializáció a JAX-B (Java Architecture for XML Binding) szabványt implementáló Jackson keretrendszer segítségével történik.

2.6.Térképszolgáltatások

Térképszolgáltatások szempontjából több API közül választhatunk, amelyek egy része ingyenes. A legelterjedtebb a GoogleMaps API, de mára már számos alternatívája létezik, mint például a Leaflet API, az OpenLayers, a MapQuest. A Sparrow projektben két térképszolgáltatás is használva van: a GoogleMaps API az Android kliens-alkalmazás térképnézetének biztosításához, a Leaflet API a webes-alkalmazás térképnézetének megvalósításához.

A GoogleMaps API a Google által fejlesztett, ingyen használható térképszolgáltatás, amelyet 2005 nyarán adtak ki. Számos lehetőséget biztosít a fejlesztők számára és nagy előnye, hogy kevés erőfeszítéssel a térképnézet teljes felülete személyre szabható. Az alap funkciókon kívül (pl. görgetés, közelítés stb.) lehetőség van a térkép feletti rétegre rajzolni (bizonyos információk megjelenítése, markerek). Mind webes alkalmazásokba, mind mobil alkalmazásokba (Android, iOS) integrálható.

A Leaflet egy nyílt forráskódú JavaScript könyvtár, amelyet több ismert alkalmazás használ (pl. FourSquare, Flickr stb.). A Sparrow webes megjelenítő felülete is ezt használja, mivel egy könnyen bővíthető könyvtárról van szó és a mobil böngészők is támogatják. A Sparrow esetében szükséges volt a könyvtár klaszterezési moduljának kibővítése a helyes számadatok megjelenítésének érdekében.

2.7. További technológiák

A naplózási keretrendszerek biztosítják az alkalmazáson belüli üzenetek rendszerezését, illetve szabályozható azok kimeneti helye, továbbá beállítható az üzenetek különböző szinteken való viselkedése.

A Simple Logging Facade for Java (SLF4J) [11] absztrakciós szintet képez különböző naplózási keretrendszerek fölött. Lehetőséget biztosít a felhasználók számára, hogy a kívánt naplózási keretrendszert telepítési időben kössék be. A Log4j [12] az Apache által fejlesztett, Java-alapú keretrendszer, amely naplózási szinteket (info, debug stb.) biztosít az üzenetek elhatárolásához.

A szoftverek tesztelése a verifikációs és validációs folyamat része, amely által biztosítjuk a termék minőségét, a követelmények és funkcionalitások betartását, illetve bizonyítjuk, hogy a program az elvártak alapján működik. A hiányosságtesztelés célja a szoftverhibák megtalálása. A fejlesztés során minél hamarább felfedjük ezeket a hibákat, annál olcsóbb azok javítása.

A JUnit [13] egy platformfüggetlen unit teszt keretrendszer, amely lehetővé teszi Java alapú szoftverek automatizált tesztelését. A Mockito [14] egy Java-alapú, nyílt forráskódú teszt keretrendszer, amely lehetővé teszi mock objektumok használatát. A mock objektumok az eredeti objektumok makettjei, helyettesítik a vizsgált objektum külső függőségeit, illetve képesek szimulálni azok viselkedését.

A Sparrow projekt fejlesztése során a JUnit és Mockito tesztelési keretrendszerek által készültek az automatizált tesztek, illetve az SLF4J és Log4j keretrendszerek biztosították a naplózást.

3. A Sparrow projekt

A következő részek ismertetik a Sparrow projekttel kapcsolatos fontosabb követelményeket, a rendszer alapvető funkcionalitásait, modelljét és architektúráját, illetve ezek megvalósításának fontosabb részleteit.

3.1. Követelmények, fontosabb funkcionalitások

A Sparrow négy fontosabb alrendszerből tevődik össze, a Sparrow szerverből, az adminisztrációs felületet biztosító webes alkalmazásból, az Android mobil kliensalkalmazásból és az adatelemzésekre használható webes alkalmazásból.

A szerver biztosítja az adathozzáférési réteget, az üzleti logikát megvalósító komponenseket, illetve az ezek elérését lehetővé tevő szolgáltatási réteget, valamint a kommunikációt a kliensalkalmazásokkal. A további alrendszerek különböző felhasználói szerepköröknek megfelelő, specifikus funkcionalitásokat biztosítanak.

3.1.1. Az Android kliensalkalmazás funkcionalitásai

Az Android kliensalkalmazás lehetővé teszi, hogy a felhasználók a Sparrow rendszerben lévő forgalmi eseményeket térképen követhessék, illetve a környezetükben észlelt eseményeket bejelenthessék mobileszközük segítségével.

A Sparrow Android kliensalkalmazás fontosabb funkcionalitásai:

- bejelentkezési és regisztrációs felületek biztosítása;
- különböző szociális hálózatokon keresztüli bejelentkezés (Google+, Facebook, LinkedIn);
- elfelejtett jelszó esetén új jelszó kérése;
- forgalmi események regisztrálásának lehetősége (pl. balesetek, forgalmi dugók, térfigyelő kamerák bejelentése);
- az adatbázisban levő forgalmi események és kamerák megjelenítése Google Maps térképen (a felhasználó környezetében lévő események);
- az egyes eseményekkel kapcsolatos visszajelzések küldése (megjegyzések, megerősítés, törlési kérés);
- az adatbázisban levő forgalmi események változásának valós idejű frissítése a térképen (pl. inaktív esemény, új esemény);
- felhasználók vizuális és auditív értesítése a környezetükben levő eseményekről (abban az esetben is, ha az alkalmazás nincs az előtérben);
- események kategória szerinti szűrésének biztosítása.

Fontos nem funkcionális követelmény a felhasználóbarát és szoftver-ergonómiai szempontból megfelelő felület, mivel az alkalmazást a felhasználók legtöbbször vezetés közben használják.

3.1.2. A webes adminisztrációs felület funkcionalitásai

A webes adminisztrációs felület, az adminisztrátorok számára menedzselhetővé teszi az adatbázisban szereplő felhasználókról és eseményekről tárolt adatokat.

A Sparrow Web UI fontosabb funkcionalitásai:

- bejelentkezési felület biztosítása;

- események és kamerák megjelenítése Leaflet térképen;
- szűrési lehetőségek biztosítása az események kategóriái szerint, illetve felhasználók szerint;
- felhasználók adatainak részletes megjelenítése (belépési adatok, általuk bejelentett események, aktivitás stb.), módosítási lehetőségek biztosítása;
- felhasználók útvonalainak megjelenítése a térképen.

3.1.3. Az adatelemző modul funkcionalitásai

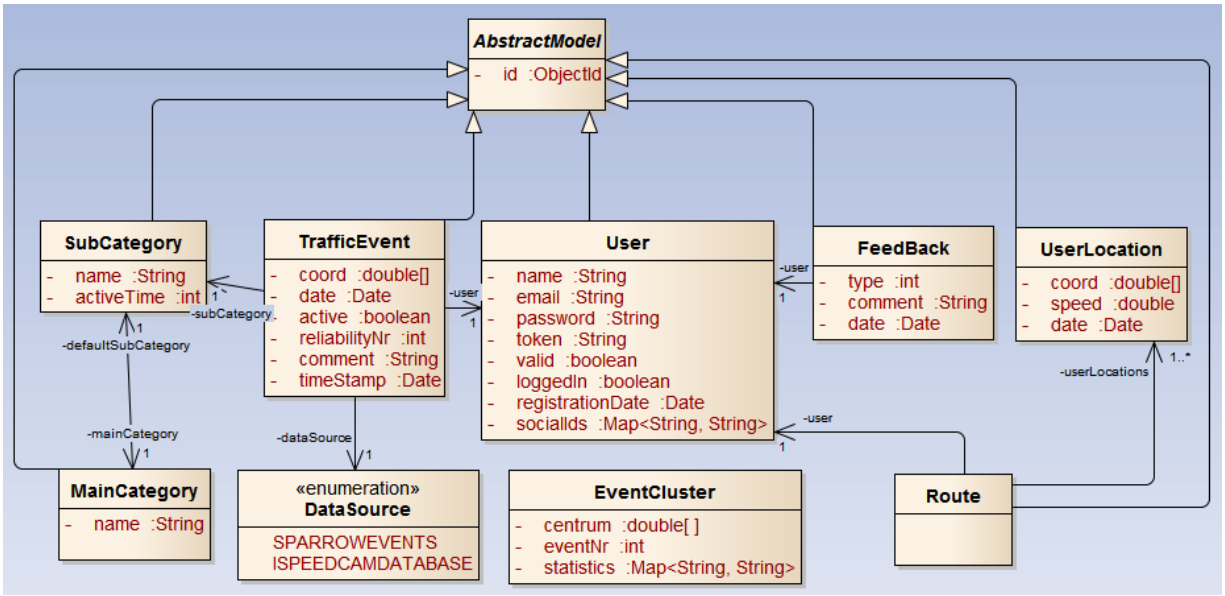
A Sparrow DataMining modul egy felületet biztosít, ahol az adminisztrátorok vizuálisan menedzselni tudják az adatforrásokat, illetve optimalizálási műveleteket futtathatnak le azokon.

A DataMining modul fontosabb funkcionalitásai:

- optimalizálási műveletek (duplikátumok szűrése, azonos kategóriájú események összesítése adott hatókörön belül stb.) futtatása a rendelkezésre álló adatforrásokon
- a futtatott műveletek eredményei Leaflet térképen való megjelenítése
- aktuális állapotok mentésének biztosítása

3.2. Környezeti elemzés

A Sparrow szoftverrendszer központi entitásait reprezentáló modell osztályok az edu.codespring.sparrow.backend.model csomagban találhatóak, több szerver oldali komponens között megosztottak.



*3. **ábra:** A szerver oldali modellt szemléltető részleges osztálydiagram*

(a metódusokat és kevésbé fontos attribútumokat mellőzve)

Az osztályok az EventCluster osztály kivételével rendelkeznek egy közös ősosztállyal, amely implementálja a Persistable interfészt. A Persistable a Serializable kiterjesztése, tartalmazza az *isNew()* és a *getId()* metódusokat. Az *isNew()* segítségével lekérdezhető, hogy az adott objektum rendelkezik-e azonosítóval, az azonosítót az adatbázisba való mentés után kapja. Az AbstractModel ősosztály egy *ObjectId* típusú azonosítót tartalmaz, amelyet a MongoDB biztosít számára.

Az EventCluster osztály azért képez kivételt, mert a MapReduce módszer futásának következtében lesz lementve az adatbázisba és emiatt nem *ObjectId* típusú az azonosítója.

A TrafficEvent osztály példányai tárolják a forgalmi eseményekkel kapcsolatos információkat. Egy esemény rendelkezik koordinátákkal (coord), bejelentési dátummal (date), egy állapotot jelző attribútummal (active), megbízhatósági számmal (reliabilityNr), hozzászólásokkal (comment), időbélyeggel (timestamp), rögzíti a felhasználót, aki bejelentette (user), valamilyen altípusba tartozik (subCategory) és valamilyen adatforrásból (dataSource) származik, amely lehet a felhasználói közösség, vagy valamilyen előre megadott adatbázis (pl. kamera adatbázis). A DataSource felsorolás típus (enum) határozza meg a lehetséges adatforrásokat.

A User osztályok példányai reprezentálják a Sparrow rendszerbe regisztrált felhasználókat. Tartalmazzák a felhasználótól bekért információkat (e-mail cím, név, jelszó), illetve egy token-t, amelyet az Android kliensalkalmazás bejelentkezés után megkap a szervertől, így a felhasználónak a legközelebbi bejelentkezésnél már nem feltétlenül kell beírnia a bejelentkezési adatokat, a token segítségével a szerver azonosítani tudja.

A Coordinates osztály földrajzi koordinátákat reprezentál. A UserLocation osztály a felhasználó pontos koordinátáin kívül tartalmazza az aktuális sebességét és a pozíció rögzítésének időpontját. A Route osztály felhasználó egy útvonalát jelképezi, UserLocation példányokat tartalmazó listát és egy felhasználó példányt foglal magába.

A FeedBack osztály tartalmazza egy felhasználónak a visszajelzését egy adott eseményről, amely három típusú lehet: hozzászólás, visszaigazolás és törlési kérés.

A MainCategory és SubCategory az esemény típusára vonatkozik, minden főkategóriának (pl. útlezárás) van egy alapértelmezett alkategóriája, így ha a felhasználó nem szeretné, vagy nem tudja pontosan megadni az esemény típusát (pl. nem tudja az útlezárás okát), elegendő csak a főkategóriát meghatároznia.

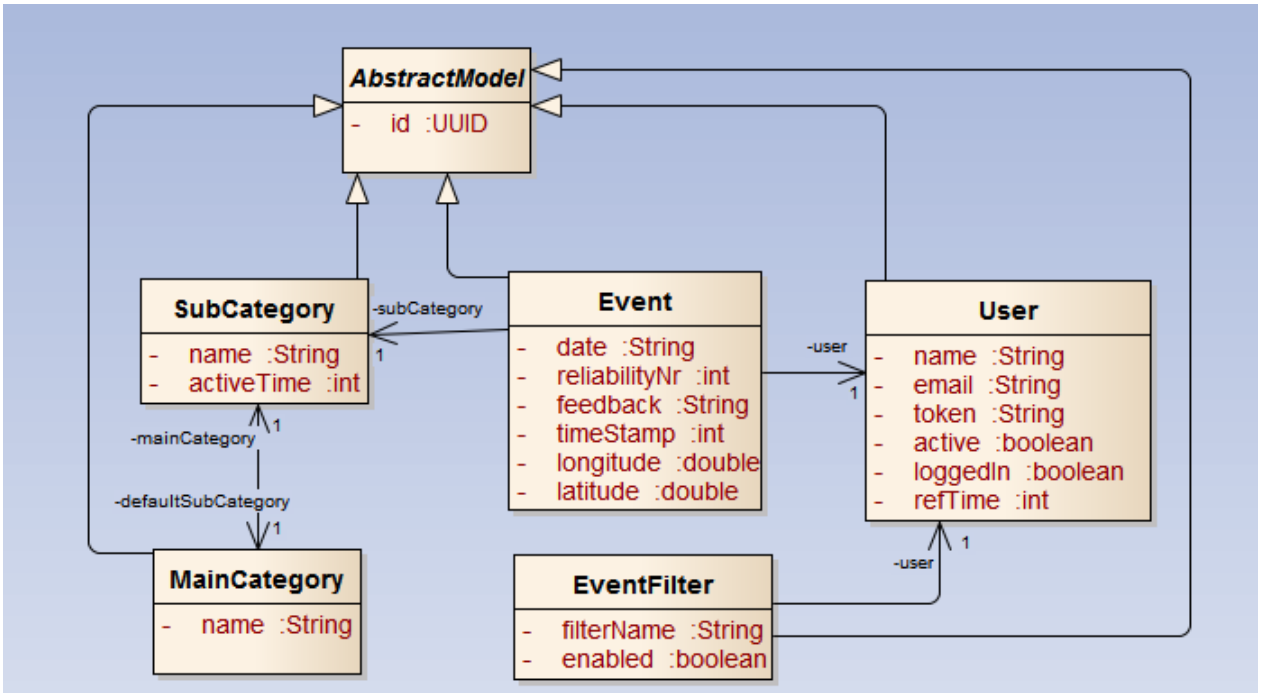
Az EventCluster osztály a *mapreduce* módszer használatánál fontos, az események egy csoportját jelképezi. Az azonosítója egy koordináta, mivel egy adott helyszín körüli események

vannak összevonva egy EventCluster példányba. További jellemzői az összes tartalmazott esemény száma, illetve a tartalmazott események száma eseménytípusokra bontva.

Az entitások adatbázisba lesznek mentve, így rendelkeznek Spring Data MongoDB specifikus meta adatokkal, @Document annotációval vannak ellátva és a megfelelő helyeken használva vannak a @DBRef és @Id annotációk.

A Sparrow projekt Android modulja külön modellel dolgozik, amely a szerver oldali modell leegyszerűsített változata. Kliens oldalon nem volt szükség minden adatra és a szerver oldali keretrendszerek meta adataira sem. Az Android oldalon használt modell osztályok az edu.codespring.sparrow.android.model csomagban találhatóak.

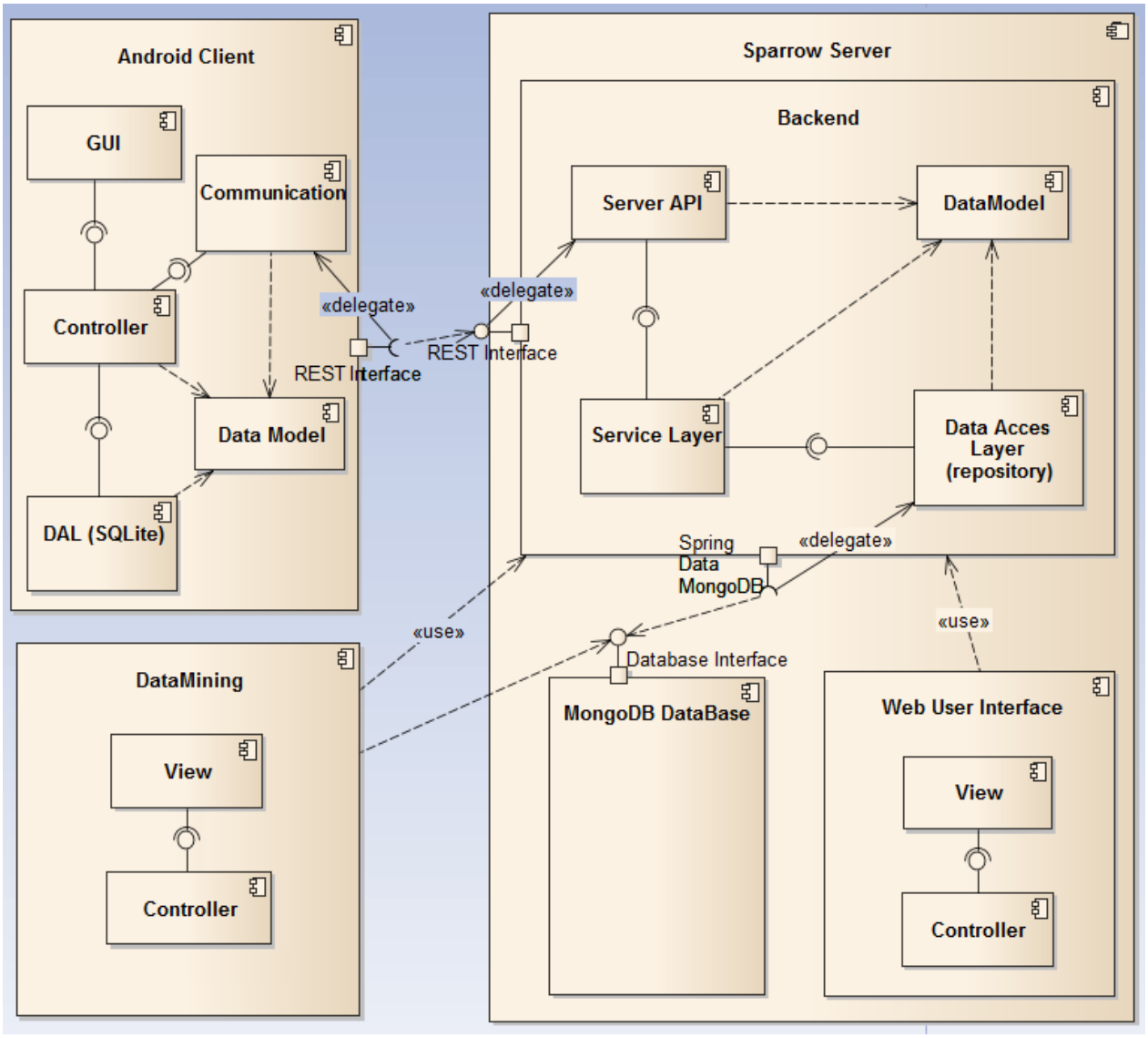
Az Android kliens modulban be van vezetve egy új entitás is, az EventFilter, amely az események szűrésének megkönnyítésére és adatbázisba való mentésére szolgál. Egy másik különbség, hogy a User osztályban egy új attribútum is megjelenik, a *refTime*, amely megadja, hogy a kliens milyen időközönként kezdeményezzen kommunikációt a szerverrel.



*4. **ábra**: A kliens oldali modellt szemléltető részleges osztálydiagram*

3.3. Architektúra

A Sparrow szoftverrendszer architektúra szempontjából két nagyobb komponensre bontható, a Sparrow szerver oldalra és a Sparrow Android kliensre.



*5. **ábra:** A Sparrow architektúrája*

A szerver oldal újabb három komponensre bontható: itt található a Backend, a webes adminisztrációs felület és a rendszer központi adatbázisa.

Adatbázisként a MongoDB NoSQL dokumentumorientált adatbázis-kezelő rendszer szolgál. A rendszer adatbázisába az egyes forgalmi események koordinátáit el kell menteni, majd a földrajzi koordináták között biztosítani kell a gyors keresést nagy adatmennyiség esetén is. A MongoDB támogatja a 2d indexelést, így eleget téve ezeknek az elvárásoknak.

A backend komponensen belül található adathozzáférési réteg (Data Access Layer) felel az adatbázis műveletek elvégzéséért, az entitások elmentéséért, törléséért illetve betöltéséért felhasználva a szerver oldali modell csomagot. Mindez a Spring Data-MongoDB keretrendszer használatával van megvalósítva.

A Service réteg teszi lehetővé a kommunikációt a rendszert alkotó különböző alrendszerek között. Itt találhatóak az üzleti logikáért felelős komponensek. Ezek közé tartoznak a háttérműveleteket megvalósító modulok is, amelyek bizonyos időközönként lefutva menedzselik a felhasználókat és az eseményeket (pl. felhasználók kijelentkeztetése, régi események inaktívvá tétele).

A szerver oldali API réteg felel a különböző kérések és válaszok megfelelő továbbításáért. Kérés esetén továbbítja ezt a Service rétegnek, amely elvégzi a megfelelő műveleteket, majd a választ visszaadja az API rétegnek, amely a maga során ezt továbbítja a megfelelő kliensnek.

A Sparrow webes adminisztrációs felülete direkt módon kommunikál a backendel, illetve a szolgáltatási rétegen keresztül az adathozzáférési réteggel. Ezen modul két komponensre bontható: egyrészt megtalálhatóak a nézetért felelős osztályok a View csomagban, másrészt rendelkezik egy Controller csomaggal, ami vezérlésért felelős.

Az Android kliens oldalon az adathozzáférési réteg egy SQLite adatbázishoz kapcsolódik. Ez a réteg felel az adatbázis műveletek végrehajtásáért, az események és a felhasználók menedzseléséért.

A kontroller csomag több részből áll és mindegyiknek jól meghatározott funkcionalitása van. A kontrollereknek egyrészt összekötő szerepe van a modult alkotó különböző komponensek között (kommunikáció a többi réteggel), másrészt ezek felelnek a különböző műveletek megvalósításáért is. Ugyancsak itt található egy service is, amelynek köszönhetően a felhasználói felület válaszideje nem romlik, mivel az egyes műveletek külön szálon futhatnak. A service feladata a kérések küldése a szerver felé, illetve a GPS koordináták pontos meghatározása. A szerverrel való RESTful kommunikáció AsyncTask-ek segítségével van megvalósítva.

A DataMining modul felhasználja a backend egyes szolgáltatásait, illetve a Spring Data-MongoDB keretrendszer segítésével direkt módon kommunikál az adatbázissal. A komponens két csomagból tevődik össze: a View a nézet kialakításáért felelős osztályokat tartalmazza, míg a Controller a vezérlést valósítja meg.

4. A rendszer megvalósításának rövid összefoglalója

A következőkben a dolgozat röviden összefoglalja a Sparrow projekt alrendszereinek megvalósításával kapcsolatos fontosabb részleteket.

4.1. A szerveroldal megvalósítása

A Spring keretrendszer használatához szükség van a keretrendszer konfigurálására, amely xml alapú kontextus állományok segítségével történik. A Sparrow esetében két kontextus fájlról beszélhetünk: a *backendContext.xml* és *databaseMiningContext.xml* állományokról, amelyek abban különböznek egymástól, hogy más adatbázis konfigurációt tartalmaznak. A *backend* kontextusában konfigurálva van a Spring keretrendszer futásához szükséges AspectJ, a MongoDB adatbázis, a konverziót végző EventClusterReadConverter bean, a Spring e-mailküldő szolgáltatása, illetve a Sparrow szerveren futó időzített szolgáltatások.

A Sparrow repository rétegének megvalósítása a Spring Data MongoDB keretrendszer segítésével történt, amely leegyszerűsíti a programozó dolgát, mivel konkrét implementációt nem szükséges megírni az egyes modellobjektumok kezeléséhez, hanem csak egy interfészt kell létrehozni, amely már tartalmazza a legelemibb metódusokat (*save, delete, deleteAll, findAll, findOne*) és könnyen bővíthető újabb metódusokkal. A metódusok nevei alapján a keretrendszer generálja a lekérdezéseket és elvégzi a kért adatbázis műveleteket. A bonyolultabb lekérdezések esetében, ha nem akarunk túl hosszú metódusneveket adni, lehetőség van megadni a MongoDB lekérdezést egy *@Query* annotáción belül. A Sparrow repository rétegén belül minden interfész a ModelRepository interfészt terjeszti ki, amely csak a Sparrow modell objektumaival kapcsolatos adathozzáférési műveleteket támogat és kötelezővé teszi az *ObjectId* típusú azonosító használatát.

A Sparrow szolgáltatási rétege vigyáz arra, hogy a felhasználók ne maradjanak bejelentkezett állapotban, abban az esetben, ha kliens oldalon hiba történik vagy az internetkapcsolat megszakad, illetve arról is gondoskodik, hogy az események lejárati idejük után inaktívvá váljanak. Ezek a funkciók időzített szolgáltatások formájában vannak implementálva, a a Spring keretrendszer Task Scheduling szolgáltatását igénybe véve, amely egyszerűen konfigurálható a *@Scheduled* annotáció segítségével.

A *mapreduce* szolgáltatás is egy időzített szolgáltatásként működik a Sparrow szoftverrendszeren belül, amely naponta egyszer fut le és összegzi az addigi eseményeket. A Sparrow számára szükséges volt nagy mennyiségű adatot megjeleníteni a térképen, amelyet a térkép nagyon lassan tudott feldolgozni, és a megjelenítés átláthatósága is romlott. A *mapreduce* módszer segítségével a rendszer egy adott koordináta körül egy adott körzetben az összes eseményt egy csomóponttá vonja össze, ennek köszönhetően az adathalmaz mérete lecsökken. A létrejött csomópontokban típus szerint csoportosítva vannak az események, így a felhasználó számára továbbra is látható lesz, hogy egy adott csomópont milyen típusú eseményeket tartalmaz.

A csomópontok adatbázisba mentődnek a *mapreduce* által, mint EventCluster objektumok és egy adott nagyítási szint alatt ilyen EventCluster objektumok jelennek meg a térképen. Ha eléggé ráközelít a felhasználó egy területre, adott nagyítási szint fölött láthatja a konkrét eseményeket is.

A Sparrow igénybe vette a Spring keretrendszer e-mail küldési szolgáltatását is, amely a szintén a kontextus állományban van konfigurálva (megadva a mail szerver konfigurációját, illetve az e-mail címet és jelszót, amelyről el lesznek küldve az üzenetek). A Spring által biztosított JavaMailSender osztály segítségével történik a konkrét e-mailküldés.

Szükséges volt a fejlesztés során egy MongoDB konverter használata az előzőleg említett EventCluster típus eltérése miatt, ezért konfigurálva van egy EventClusterReadConverter, amely egy MongoDB specifikus DBObject típust átalakít EventCluster típusba. A konverziót végző osztály függvénye meghívódik mindig, amikor EventCluster típus olvasódik ki az adatbázisból.

A Sparrow kommunikációs rétege a JAX-RS, illetve JAX-B szabványokra és ezeknek megfelelő Jersey és Jackson implementációkra épül.

A REST erőforrásokat JAX-RS annotációk segítségével tudjuk konfigurálni. @Path annotáció segítségével kell megadni a REST szolgáltatás elérési útját. A megfelelő annotáció használatával meg kell adni, hogy milyen HTTP parancs hatására hívódjon meg egy adott metódus, illetve a @Produces annotáció paramétere jelzi, hogy az adatok milyen formátumban lesznek elküldve (a Sparrow JSON formátumot alkalmaz). A klienshez egy http válasz jön, ez rendelkezik státuszkóddal, amely segítségével megtudhatjuk, hogy történt-e hiba. Amennyiben a szolgáltatás metóduson belül hiba történik, kivétel keletkezik, amelyet az ApiExceptionMapper kezel. Ez @Provider annotációval van ellátva és implementálja a Jersey által biztosított ExceptionMapper interfészt, megfelelő választ küldve a felhasználónak a kapott kivétel függvényében.

A szerver és a kliens különböző modelleket használ és a kommunikáció szempontjából az is fontos, hogy ne terheljük a hálózatot, csak a szükséges adatok közlekedjenek rajta. Ennek érdekében a Sparrow a DTO (Data Transfer Object) tervezési mintát használja. A kliens és a szerver oldalon is assembler komponensek felelősek a DTO-k modell objektumokba és a modell objektumok DTO-kba történő átalakításáért.

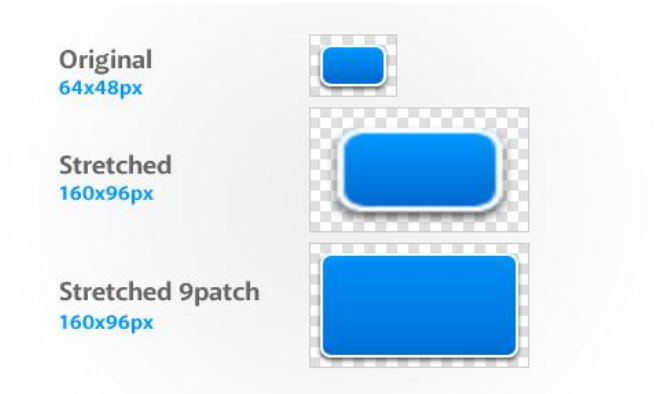
4.2. Az Android kliensalkalmazás megvalósítása

Android nézeteket kétféleképpen tudunk meghatározni: az egyik megoldás, hogy a programból adjuk meg a megfelelő kinézeti elemeket és ezek elhelyezkedését a képernyőn, a másik megoldás, hogy xml állományokban határozzuk meg ezeket a komponenseket. Az MVC

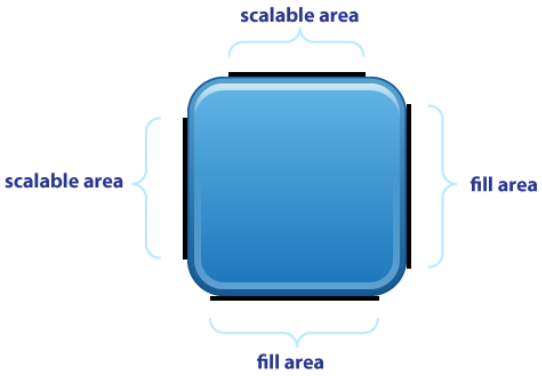
elv szempontjából a jobb megoldás az xml állományok használata. Az xml állományokban van megadva a komponensek elrendezése és mérete (pl. szövegdobozok, gombok, címkék stb.), ezek a megfelelő Activity-khez rendelhetőek. Ahol lehet, a Sparrow rendszer Android alkalmazása ezt a mintát követi.

Az alkalmazás olyan módon van megtervezve és kivitelezve, hogy az egyes kinézeti elemek könnyen változtathatóak legyen. Ennek érdekében az egyes szín, méret és szöveg elemek a nekik megfelelő xml állományokban vannak meghatározva (colors.xml, dimens.xml és strings.xml). Mivel nincsenek az adatok beleégetve a kódba, a kód könnyen módosítható, a beállítások több helyen is felhasználhatóak, csökkentve egyben a kódmennyiséget és ezáltal az alkalmazás méretét is. Abban az esetben, ha tablet készülékeket is akar támogatni a rendszer, nincs szükség a teljes xml állomány (layout) újradefiniálására, elég, ha a megfelelő vizuális elemeket (komponensek közötti térköz, komponensek mérete stb.) írjuk át tabletnek megfelelően, amelyek a dimens.xml állományban találhatóak (ebben az esetben külön lesz egy dimens.xml okostelefonok részére és külön a tabletek részére). A Sparrow alkalmazás alkalmazza az Android fejlesztők által ajánlott responsive design tervezést az úgynevezett Multi-pane layoutok használata által.

Ugyancsak a különböző képernyőméreteken és felbontásokon való egységes kinézet érdekében a Sparrow kliensalkalmazásban az Android platform által felismert 9-patch típusú képek vannak használva (9.png kiterjesztés). Ez lehetővé teszi a képek átméretezését (skálázását) anélkül, hogy a kép minősége romoljon, ahogy ezt a 6. ábra is mutatja.



*6. **ábra**: 9patch képek skálázása [15]*



*7. **ábra:** 9patch kép szerkezete*

A 7. ábrán illusztrálva van egy ilyen 9.png kiterjesztésű kép kinézete. Fontos itt megemlíteni, hogy a kék gomb szélén látható egy pixeles fekete csíkok is a kép részei, ezek határozzák meg a kép skálázható és kitölthető részeit.

Mivel az alkalmazást a felhasználók többsége vezetés közben használja, fontos volt egy ennek megfelelő, könnyen használható felhasználói felület megalkotása, amely nem vonja el túlságosan a figyelmet a vezetésről. Ennek érdekében az alkalmazás navigációja egyszerű, jól látható megfelelő méretű gombokkal és érintés visszajelzési funkcióval (touch feedback). Az alkalmazás egy saját témával rendelkezik, egyénileg definiált komponensekkel, amelyek meghatároznak egy Sparrow brand-et (pl. piros, fehér, szürke alapszínek stb.).

A regisztráció kikerülésének érdekében a Sparrow alkalmazás lehetőséget biztosít szociális hálózatok segítségével történő bejelentkezésre. A funkcionalitás egy MIT (Massachusetts Institute of Technology) licenc alatt szabadalmazott projekt felhasználásával van kivitelezve. Jelenleg három szociális hálót (Google+, Facebook, LinkedIn) támogat a rendszer, azonban a felhasználók igényeihez mérten további szociális hálók támogatásával könnyen kibővíthető.

A navigáció „lelkét” a fejlesztők körében népszerű ActionBar képezi. Pontosabban, mivel az ActionBar csak a 11-es API szinttől lett bevezetve és az alkalmazás a 8-as API szintet is támogatja, ennek egy alternatívája van felhasználva az alkalmazásban, amely az Android Support Library v7 része. A projektben két másik hivatalos kiegészítő csomag is használva van annak érdekében, hogy olyan funkciókat is el lehessen érni, amelyek a régebbi API verziókban még nem voltak benne, vagy amelyek nem képezik részét az újabb API-knak sem. Ezek a csomagok a Google Play Services (térkép funkciók), illetve az Android Support Library v4.

A Sparrow szoftverrendszer architektúrája úgy van tervezve, hogy a támogatott forgalmi esemény típusok aránylag kevés erőfeszítéssel kibővíthetőek legyenek. Ennek érdekében a kliensalkalmazás egy forgalmi típusokat tartalmazó listát kap a szervertől akkor, amikor a felhasználó bejelentkezik a rendszerbe, és ennek a listának az alapján dinamikusan építi fel az alkalmazás esemény bejelentő felületét egy adapter osztály használatával.

Annak érdekében, hogy a különböző hosszabb ideig tartó folyamatok ne rontsák a felhasználói felület válaszidejét a Sparrow alkalmazás service komponenst használ az ilyen műveletek elvégzésére. A service felel a felhasználó pontos GPS koordinátáinak meghatározásáért, illetve bizonyos időközönként kéréseket küld az AsyncTask-eknek, amelyek továbbítják a kérést a szervernek, majd várja a választ, és annak függvényében, ha kell, értesíti a megfelelő activity-t egy service provider segítségével. Az activity regisztrálja magát erre a providerre, így valahányszor értesítést kap, frissíti a térképet és a rajta levő eseményeket. Ugyancsak a service dönti el, hogy melyik eseményről kell értesítenie a felhasználót.

Az Android platform nem engedi meg, hogy a kommunikáció a fő szálon legyen. Ennek érdekében ahhoz, hogy a kliensalkalmazás a szerverrel kommunikálni tudjon, szükség van az AsyncTask-ek használatára. A Sparrow Android alkalmazásában négy AsyncTask teszi lehetővé a kommunikációt a szerverrel: egy ősosztály és a kérés típusának (GET, PUT és POST) megfelelő három származtatott osztály.

A kommunikáció megvalósításának érdekében szükségessé vált a JAX-B használata, de ez az Android platformon nehézkes. Az Android platform nem engedélyezi, hogy jar kiterjesztésű csomagban olyan osztály szerepeljen, amely a javax.xml névtér alatt van, nem megengedett, hogy a csomag .java kiterjesztésű állományokat tartalmazzon, a platform nem támogatja a csomag szintű annotációk használatát, nem biztosítja a javax.xml.bind csomagot és nem is támogatja annak használatát. Továbbá a JAX-B használ olyan osztályokat, amelyeket az Android platform nem biztosít. Ezek miatt a megszorítások miatt a standard JAX-B implementációk eredeti formájukban a platformon belül nem használhatóak. Ennek következtében a Sparrow rendszerben egy módosított JAX-B csomag van használva.

4.3. Az adatelemző modul megvalósítása

Ahhoz, hogy az adatforrásokkal biztonságosan tudjunk dolgozni, szükséges volt egy második kontextus fájl létrehozása, a databaseMining.xml, amely más adatbázis konfigurációt tartalmaz, mint az eredeti kontextus fájl, és egyúttal konfigurálja a Spring keretrendszert is a DataMining modul számára.

Ellentétben a Backend-el, a DataMining modul nem a Spring Data-MongoDB által biztosított interfész alapú adatbázis hozzáférést használja, hanem az egyes műveletek elvégzését a MongoTemplate osztály segítségével végzi el, amely direkt kommunikációt biztosít az adatbázissal. A bonyolultabb műveletek elérését egy Factory tervezési mintán alapuló osztály biztosítja, a műveleteket implementáló osztályok működése két fázisra van felosztva: először inicializálják a szükséges adattagokat, majd futtatják a kívánt műveleteket.

A BSON típusú objektumok előállítását a DBObject interfész teszi lehetővé, implementációja a BasicDBObject osztály *put* metódusa által biztosítja a kulcs-érték párok meghatározását, amelyet a MongoDB adatbázis értelmezni tud.

4.4. Tesztelés

A Sparrow API tesztelésének konfigurációját egy külön kontextus fájl végzi el, amelyben definiálva vannak a tesztelendő osztályok külső függőségeinek makettjei (mock objektumok).

Ez az állomány konfigurálja a Spring keretrendszer használatát is a tesztelés folyamán. A bean-ek definiálása során megadható a mock keretrendszer osztálya, illetve metódusa, amelynek segítségével a makettek létrehozása történik.

Az API tesztelése során a tesztesetek futásáért a Jersey által biztosított Grizzly teszt-konténer felel. A keretrendszer minden egyes teszteset futtatását elszigetelten, egy külön szerver indításával oldja meg.

A REST-assured keretrendszer metódusokat biztosít, amelyek segítségével implementálhatóak a tesztesetek, megadhatóak a hívási paraméterek, a HTTP kérés típusa, a kérés törzse, a visszatérített adatok stb.

A Sparrow Service tesztelése teljes mértékben a Mockito és JUnit keretrendszerek segítségével történt. A JUnit @RunWith annotációja a paraméterként kapott osztály alapján dönti el, hogy melyik osztály legyen használva a tesztek futtatásához (a Sparrow esetében a MockitoJUnitRunner osztály). A tesztelendő osztály külső függősségeinek mock-olását a @Mock annotáció oldja meg, a függősségek dinamikus bekötését az @InjectMocks annotáció.

5. A Sparrow működése

A következőkben a Sparrow rendszer kliensalkalmazásai és felhasználói felületei által biztosított fontosabb funkcionalitások kerülnek bemutatásra.

5.1. Az Android kliensalkalmazás használata

Ahhoz, hogy a Sparrow Android kliensalkalmazását használni tudják, a felhasználóknak előbb regisztrálniuk kell a Sparrow szoftverrendszerbe megadva nevüket, egy érvényes email címet, illetve egy jelszót. Ezt követően a rendszer egy megerősítő e-mailt küld az általuk megadott címre, amelyben egy link található. A felhasználó a link-re történő rákattintás után tud bejelentkezni a rendszerbe. Abban az esetben, ha valaki nem szeretné kitölteni a regisztrációs form-ot, lehetősége van szociális hálózat segítségével történő bejelentkezésre. Azok a felhasználók, akik elfelejtették jelszavukat egy új jelszót igényelhetnek, amelyet az általuk megadott email címre küld a rendszer.

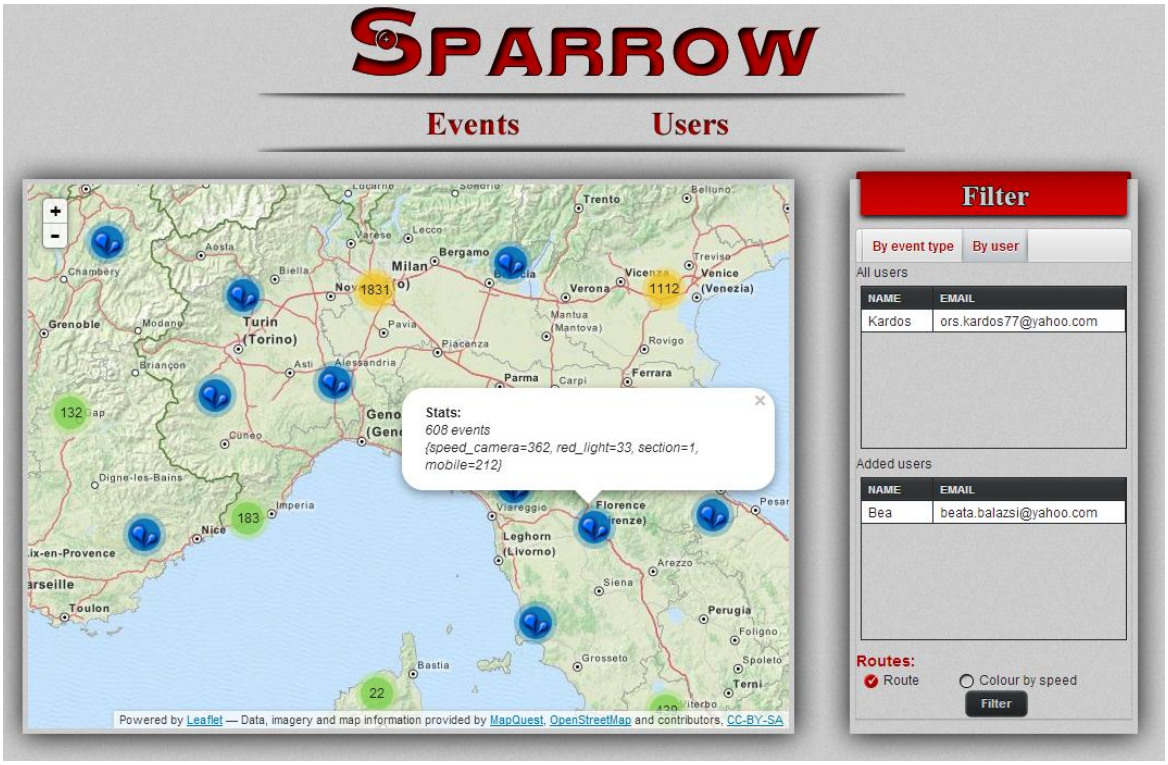
Sikeres bejelentkezés után a felhasználók egy Google Maps térképen láthatják a körülöttük levő már bejelentett eseményeket (amelyeket egy-egy marker jelképez), illetve az általuk észlelt forgalmi eseményeket egy másik felületen bejelenthetik. Egy markerre való kattintás után az adott marker által jelölt eseményről kapnak részletesebb információt (pl. mikor

volt bejelentve az adott esemény, hány felhasználó erősítette meg stb.), illetve lehetőségük van megerősíteni az adott eseményt, érvénytelenítését kérni vagy hozzászólást fűzni hozzá.

A profil nézet lehetőséget ad a felhasználóknak a regisztrációnál megadott adataik megváltoztatására. A beállítás nézetben konfigurálhatják az alkalmazást, például különböző kategóriák szerinti szűrést kérhetnek (milyen típusú eseményeket szeretnének látni a térképen), megadhatják, hogy milyen gyakran küldje a kliensalkalmazás a kéréseket a szervernek.

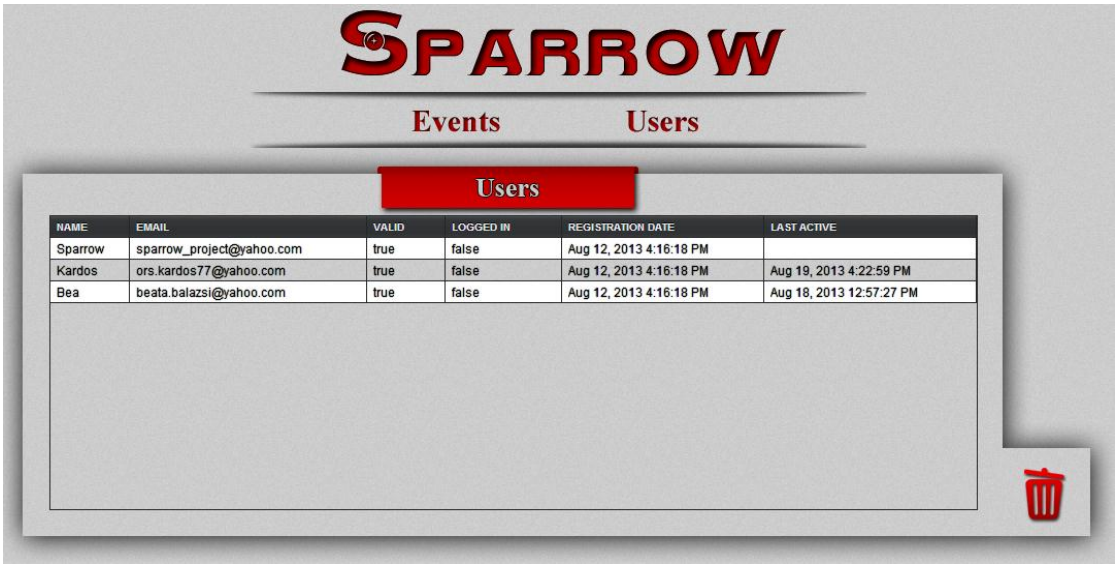
5.2. A webes adminisztrációs felület használata

A rendszer adminisztrátora bejelentkezés után alapértelmezetten az összes forgalmi eseményt és kamerát látni fogja a térképen. Amennyiben csak bizonyos típusú eseményekben érdekelt, szűrési paramétereket határozhat meg. Felhasználók szerint is végezhet szűrést, és megtekintheti a térképen a felhasználók útvonalait is.



*8. **ábra:** A Sparrow webes adminisztrációs felület szűrése felhasználók szerint*

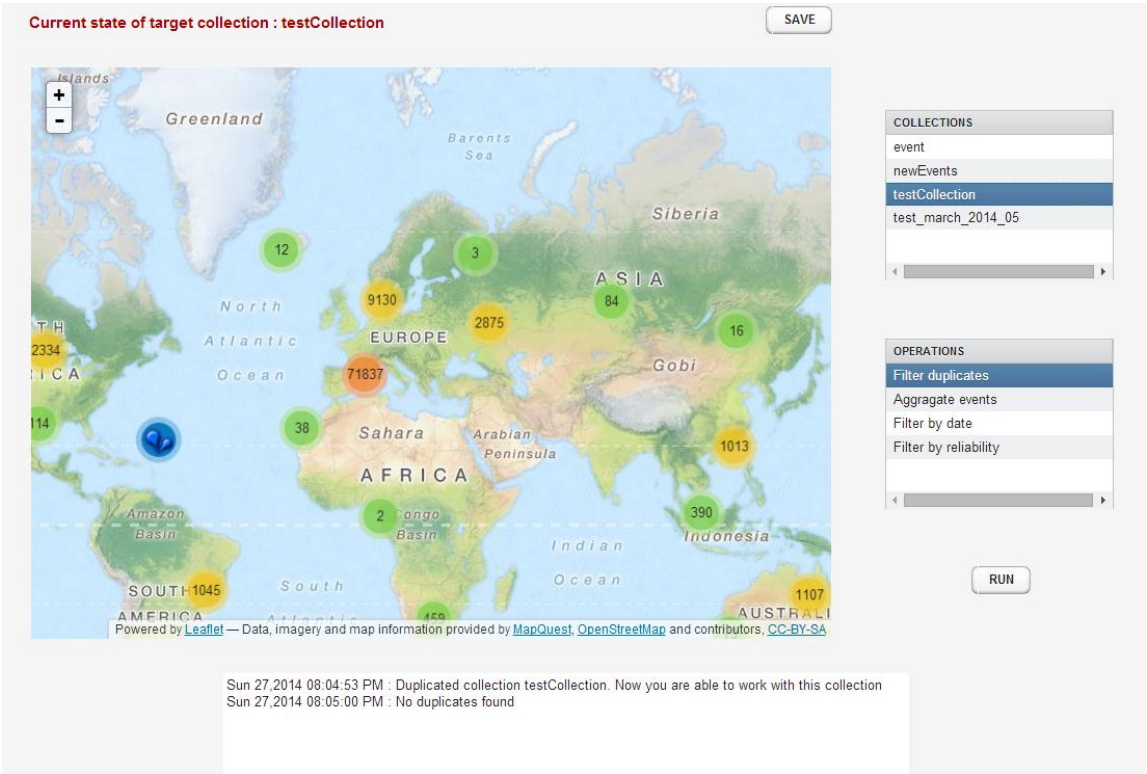
Lehetőség van felhasználókkal kapcsolatos információkat (profil adatok, bejelentett eseményekkel kapcsolatos statisztikák stb.) megtekinteni és módosítani.



*9. **ábra:** A Sparrow webes adminisztrációs felület felhasználókkal kapcsolatos nézete*

5.3. Az adatelemző felület használata

A DataMining modul lehetőséget ad az adminisztrátorok számára az adatforrások vizuális kezelésére, illetve az irreleváns adatok szűrésére. Az adatbázisban tárolt adatokon optimalizálási műveletek futtathatóak le (például duplikátumok törlése, azonos kategóriájú események összesítése egy bizonyos térségen belül stb.).



*10. **ábra**: A DataMining modul felülete*

Első lépésként ki kell választani az adatforrást, amelyen szeretnénk elvégezni az optimalizálási műveleteket, majd a művelet típusát kell megadni. Indítás után a műveletek adatbázis oldalon végződnek el, majd az eredmény azonnal megtekinthetővé válik a térképen.

A térkép fölött megtekinthető, hogy melyik adatforráson dolgozunk, illetve menthetjük a térképen látható állapotot. A rendszer naplózza az elvégzett műveleteket, illetve azok eredményét, ezeket az információkat is megtekinthetjük a felületen.

Az adminisztrátor az optimalizálási műveletek futtatása után, a kapott eredményt csatolhatja a rendszer által használt adatbázishoz, így az eredményként kapott gyűjtemény elemeit (események) megoszthatja a felhasználókkal.

Következtetések és továbbfejlesztési lehetőségek

A Sparrow projekt fejlesztése során sikerült egy olyan alkalmazás prototípusát megvalósítani, amely főleg az autóvezetői közösséget célozza meg. Használatával a felhasználók közlekedési információkat kaphatnak és oszthatnak meg, ezáltal előreláthatóan tervezhetik meg útvonalaikat, lehetséges akadályokat és veszélyeket kerülhetnek ki.

Kliensalkalmazásként sikerült létrehozni egy Android platformra tervezett mobil-alkalmazást, amely lehetőséget ad különböző szociális hálózatokkal való bejelentkezésre (Facebook, LinkedIn, Google+), használata által a felhasználók előre meghatározott, kategóriákba sorolt eseményeket jelenthetnek be, illetve jeleníthetnek meg Google Maps térképen.

Megvalósításra került egy adminisztrációs webes felület is, amely segítségével az adminisztrátorok megtekinthetik az adatbázisban tárolt adatokat, feltételek alapján szűrhetik azokat. A felület lehetőséget ad a felhasználók tevékenységeinek (például bejelentett események, bejárt útvonalak stb.) megtekintésére és menedzselésére is.

Sor került egy adatfeldolgozásra alkalmas felület megalkotására is, ahol lehetőség nyílik az adatbázisban tárolt eseményeken különböző szűrési és összefésülési műveleteket elvégezni, illetve ellenőrizni az adatok hitelességét.

A Sparrow projekt továbbfejlesztésével kapcsolatban számos lehetőség felmerült. Néhány funkcionalitás, amellyel bővíteni lehetne az alkalmazást:

- egy webes felület létrehozása, ahol a regisztrált felhasználók menedzselhetik adataikat, bejelentett eseményeiket, útvonalaikat, illetve megtervezhetik útvonalaikat a bejelentett közlekedési információk alapján;
- egy webes felület létrehozása média partnerek számára (például rádió), amelyről bejelenthetnének eseményeket (pl. hallgatók által telefonon bejelentett események);
- egy plug-in készítése, amely eseményeket jelenít meg, integrálható webes felületekbe, illetve konfigurálható, hogy melyik terület eseményeit „közvetítse” (partnerek – pl. médiapartnerek, on-line hírportálok – weboldalaiba történő beágyazási lehetőség);
- az alkalmazás szorosabb összekapcsolása szociális hálózatokkal (pl. események megosztása szociális hálózatok segítségével);
- a felhasználók felé közvetített adatok forrásainak hatékonyabb menedzselése az adminisztrációs felületről, a kliensalkalmazásokon belül lehetőség a felhasznált források kiválasztására.

Hivatkozások

[1] Mercurial hivatalos weboldal, http://mercurial.selenic.com/, utolsó megtekintés dátuma: 2014.04.12

[2] Apache Maven hivatalos weboldal, http://maven.apache.org/, utolsó megtekintés dátuma: 2014.04.12

[3] Apache Tomcat hivatalos weboldal, http://tomcat.apache.org/, utolsó megtekintés dátuma: 2014.04.12

[4] Spring keretrendszer hivatalos weboldal, http://spring.io/, utolsó megtekintés dátuma: 2014.03.28

[5] Clarence Ho, Rob Harrop, Pro Spring 3, New York: Springer Science+Business Media, Apress Media LLC, 17 April 2012

[6] MongoDB hivatalos weboldal, https://www.mongodb.org/, utolsó megtekintés dátuma: 2014.03.28

[7] Android hivatalos weboldal, http://www.android.com/, utolsó megtekintés dátuma: 2014.03.28

[8] Zigurd Mednieks, Laird Dornin, G. Blake Meike, Masumi Nakamura, Programming Android, 2nd ed., Sebastopol, California: O'Reilly Media, September 2012

[9] Vaadin keretrendszer hivatalos weboldal, https://vaadin.com/, utolsó megtekintés dátuma: 2014.03.28

[10] Marko Grönroos, Book of Vaadin: Vaadin 7 Edition, revision 2nd ed., Vaadin Ltd, 2014

[11] SLF4J hivatalos Weboldal, http://www.slf4j.org/, utolsó megtekintés dátuma: 2014.04.10

[12] Log4j hivatalos Weboldal, http://logging.apache.org/log4j/2.x/, utolsó megtekintés dátuma: 2014.04.10

[13] JUnit hivatalos Weboldal, http://junit.org/, utolsó megtekintés dátuma: 2014.04.08

[14] Mockito hivatalos Weboldal, https://code.google.com/p/mockito/, utolsó megtekintés dátuma: 2014.04.08

[15] Android: 9patch erőforrások, http://martincohen.tumblr.com/post/11060125232/android-9-patch-resources, utolsó megtekintés dátuma: 2014.04.22