Eötvös Loránd Tudományegyetem
Informatikai Kar
Térképtudományi és Geoinformatikai Tanszék

Térfigyelő kamera nyilvántartó alkalmazás

Nagyprogram dokumentáció

Hallgató:
Nguyen Thai Binh

Témavezető:
Dr. Gede Mátyás

2012. május 30.

Tartalomjegyzék

1

Bevezetés

A térfigyelő kamerák egyre nagyobb figyelmet igényelnek a mindennapi életünkben. Kamerák nyomon követnek és rögzítenek eseményeket, amelyeket megfigyelhetünk, esetleg később megtekinthetjük.

Azonban nem mindig tudjuk, hogy az adott kamera a térben pontosan hol helyezkedik, milyen állapotban van, és mekkora a látómezője. Ha térkép alapokra tudnánk helyezni egy kamerával kapcsolatos térbeli adatait, akkor sokkal könnyebben lehetne nyomonkövetni állapotukat karbantartási szempontból, helyzet változtatásukról, ezáltal könnyebben tudnánk tájékoztatni a felhasználóit.

2

Követelmény leírás

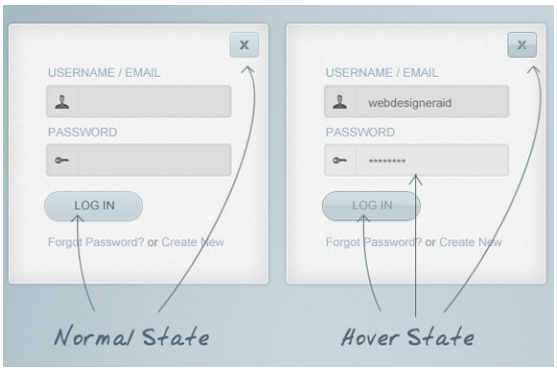
2.1. Funkcionális követelmények

2.1.1. Felhasználói kezelés

Azonosítás. A renszer használatához, felhasználói azonosítás szükséges. Az azonosítás módja a megszokott felhasználói név, vagy email cím valamint jelszó megadásával történik. Megadott számú sikertelen belejelentkezés esetén a felhasználói név vagy email cím letiltása fél vagy több óráig.

Session. Sikeres azonosítás után, böngészőben tárolt session ideje alatt lehet a rendszert használni. Session timeout után, újra bekell jelentkeznie.

Regisztráció. Új felhasználó esetén felvétele regisztrációval történik, ahol megkell adnia az email címét valamint kívánt jelszavát.



2.1. ábra. Bejelentkezés form terv

2.1.2. Kamera kezelés

Felvitel. Egy kamera felviteléhez szükséges adatok:

1. Pozíció (Latitude, Longitude) adat.
2. Egyedi azonosító, amely egyértelműen meghatározza az felvinni kívánt kamerát.
3. Pontos cím.
4. Állapot.
5. Mobilitás.
6. Video Stream cím.

Állapot. Kamera állapota tájékoztatja felhasználót és a karbantartó egységet, hogy javításra szorul-e vagy sem. Állapotai lehetnek:

- Inaktív: Nem elérhető, javításra szorul.
- Aktív: Működik.
- Hibás: Hardver vagy szoftveres hiba, javításra szorul.
- Javítás alatt: Karbantartó egység éppen dolgozik rajta.

Moblitás. Kamera egy tulajdonsága, amely egyértelműen leírja, hogy ideiglenesen van az adott helyen például egy rendezvény ideje alatt, vagy állandó a pozíciója. Mobilitás tulajdonságából tudhatjuk, hogy milyen célból lett kitelepítve egy kamera.

- Mobil: Ideiglenesen vagy megadott időtartamig van a meghatározott helyen.
- Fix: Állandó, objektumok, forgalom megfigyelés szempontjából van.

Video stream. Minden IP alapú kamera multimédia tartalmat továbbít a felhasználó számára, esetünkben legyen ez video alapú hang nélkül. Kamera felvitele során lekell tudnunk ellenőrízni, hogy a video tartalom továbbítás valóban működik-e vagy sem.

Módosítás. Egy kamera módosítása esetén csak a helyzetét, látómezőjét, állapotát, mobilitását és video stream címét lehet megváltoztatni. Azonosítóját nem lehet megváltoztatni, mivel ez alapján tudjuk egyértelműen beazonosítani a rendszerben.

2.1.3. Esemény kezelés

Egy esemény megfigyeléséhez több kamera szükséges. Fontos, hogy az adott esemény megtervezésekor előre tudni, hogy mely kamerák szorulnak javításra vagy pozíció változatatásra, esetleges mobil kamerák elhelyezésére. Egy eseményben résztvevő kamerák megtekintési sorrendje is nagyon fontos szerepet játszik, azonban a sorrendet változtatni lehet, felmerülő igény esetén.

Létrehozás. Egy esemény a legtöbb esetben poligonnal ábrázolható. Adott poligon területébe eső kamerákat megkell tudnunk határozni, még mielőtt elmentenénk az eseményt egyébb adataival, amelyek a következőek:

1. Egyedi azonosító.
2. Megnevezés: kiegészítő információ.
3. Esemény kezdetének dátuma.
4. Esemény végének dátuma.

Módosítás. Egy eseményt csak geometriai tulajdonságát, leírását, kezdeti és vég dátumát lehetséges megváltoztatni.

Megtekintés. Megtekintésekor a kamerákat egymás mellett, megadott sorrendben kell megmutatni. Sorrendiségüket bármikor meglehessen változatni, ez a művelet azonban nem számít módosításnak, mivel csak a kamerák megtekintési sorrendiségét változtattuk.

2.1.4. Súgó

Geometriai objektumok felvitele és módosítása nem minden felhasználó számára egyértelmű, ezért egy olyan felületet kell biztosítani, ahol ikon-funkciók levannak írva valamint egy "homokozó" felületet szükséges, ahol kilehet próbálni.

Az alkalmazás két fő részből áll:

1. Kamera kezelés
2. Esemény kezelés

Tájékoztató jellegű információval kell biztosítani a felhasználót, hogy melyik lépés után, mit kell végrehajtani, független a felhasználói dokumentációtól.

2.1.5. Főoldal

Az alkalmazás belépési pontja, ahonnan kiindulva eltudja érni a kamera kezelés és esemény kezelés funkciókat. Ezenkívül a belépés és regisztráció is ezen a felületen keresztül történik.

2.2. Nem funkcionális követelmény

2.2.1. Termék követelmény

Az alkalmazás webes alapúnak kell lennie. Firefox, Chrome, Opera, Safari valamint Internet Explorer alatt is működnie kell.

2.2.2. Szervezeti követelmény

Felhasznált technológiák csak is kizárólag nyíltforráskódúak legyenek. Kulcs fontosságú az adatbázisrendszer, webes keretrendszer valamint térkép szolgáltató rendszerek nyíltforráskódú vagy ingyenes tulajdonsága.

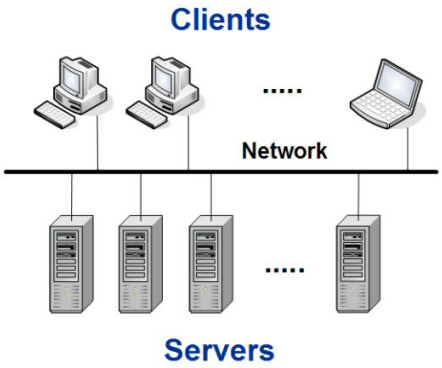
3

Rendszerterv

3.1. Kliens Szerver architektúra

A kliens-szerver architektúra üzenet alapú és moduláris felépítésű, ezáltal támogatják a az újrafelhasználhatóságot, rugalmasságot, együttműködési és bővíthetőséget. Jellemzői:

- Kéréseket küldünk a szerver felé.
- A választ a szervertől kapjuk.
- Szerepek: kérő (client) és kiszolgáló (server).



3.1. ábra. Kliens szerver architektúra

Kliensek. általában alkalmazások vagy eszközök, amelyek szolgáltatásokat kérnek a szervertől, a kapott választ feldolgozva vagy anélkül prezentálják a

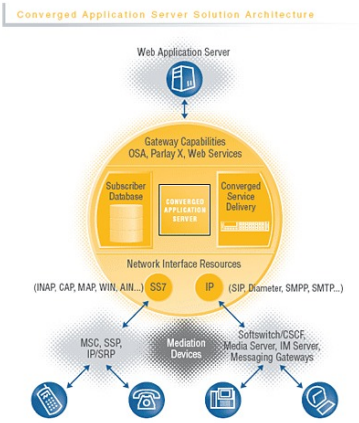
felhasználó felé, vagy továbbítják a kapott választ egy másik kliens-vagy szerver felé. Három csoportba sorolhatjuk a klienseket, hogy milyen bonyolultságú és erőforrás igényű egy kliens.

Vékony kliens (Thin client) minimális eszközökkel rendelkező kliens. A kliensnek szükséges erőforrásokat is a távoli (host) gépen veszi igénybe.

Vastag kliens (Fat client) Jellemzője, hogy önmaga hajtson végre nagyobb adatmennyiségekkel feldolgozásokat, amikor a szerver inkább elsődleges tárolóként viselkedik. 2003-as évek elejétől a gazdag kliens (rich client) kifejezést eltérő értelemben kezdték el használni, mint a vastag klienst. Ennél a kliens architektúránál a szerver központi egységének a kihasználtsága sokkal kiegyenlítettebb, mint a többiek esetében. Maga a kliens tulajdonképpen vastag kliens, de jobban kihasználja a hálózati lehetőségeket a vékony klienshez hasonlóan, így tulajdonképpen egy vastag-vékony kliens hibrid. Lásd gazdag Internet alkalmazás és gazdag kliens platform.

Hibrid kliens (Hybrid client) Hasonlít a vastag klienshez, mivel lokálisan dolgozik, de számít a szerverre adattárolás miatt. Ez a megközelítés sajátosságokat kínál mind a vastag kliensből (multimédia támogatás, nagy teljesítmény), mind a vékony kliensből(erőteljes menedzselhetőség, rugalmasság).

Kiszolgáló - Szerver. Kliens-szerver architektúrában, szerver program kéréseket szolgál ki. Háttér számításokat végez, amelyeket a kliens kért, a választ vissza küldi a kérőnek. Kliensek futhatnak ugyanazon a gépen ahol a szerver, vagy hálozaton keresztül kommunikál a szerverrel.



3.2. ábra. Kliens szerver architektúra

3.2. Adattárolás

Jelenleg három választási lehetőségünk van a nyílt forráskódú adatbázisrendszerek közül, amelyek elterjedtek, jól dokumentált, megbízható és nem utolsósorban rendelkeznek térbeli függvények támogatásával:

1. MySQL
2. Spatialite
3. PostgreSQL

3.2.1. MySQL

Egy nagyon fontos kritérium, a helyes működéshez MyISAM szerkezetűnek kell lennie minden táblának. MyISAM támogatást ad egyaránt spatial és nem spatial adatokra.

Spatial Extension funkciók:

- Geometria model támogatás: polygon, multipoint, linestring, curve, ...
- Adatok formátumok WKT (Well known text), WKB (Well known binary)
- Geometria függvények.

Úgy éreztem használati szempontjából egy kicsit körülményes volt.

3.2.2. Spatialite

Gyors, egyszerű, kicsi, hordozható. Egy hátránya van, nem képes konkurensen működni. Úgy gondolom fontos megemlíteni, mert nagyon hatékonyan lehet adatokat tárolni, rengeteg nyelven írtak API-t SQLite adatbázishoz. Spatialite extension-je pedig szinte minden geometria függvényt lefed, amit MySQL adatbázis motor szolgáltat.

3.2.3. PostgreSQL - Postgis

Postgres adatázishoz külön telepíthető komponens. Hatékony, nyíltforráskódú adatbázisrendszer.

Spatial extension funkciók:

- Geometria model támogatás: teljes.
- Adat formátumok: WKT, WBT.
- Geometria függvények.
- Adatkezelő felület: ERSI, SVG adatok import és export funkciók.
- Bulk Loader: tömeges adatok feltöltése.

- Third party szoftver támogatás: QGis, Grass, ...

Három adatbázis motor közül PostreSQL-re esett a választásom.

A következő címeken linkeken lehet elérni a fent említett adatbázisrendszereket:

Adatbázisrendszer Neve	Leírás
MySQL	http://www.mysql.com/
Spatialite	http://www.gaia-gis.it/spatialite/
PostgreSQL	http://www.postgresql.org/
Postgis	http://postgis.refractions.net/

Elsőre a MySQL megfelelőnek tűnik, azonban az igényeket felmérve, a következő feltételeket nem tudja teljesíteni:

- Spatial kiegészítője még nagyon kezdetleges funkcionalítással rendelkezik.
- Nem tudja lefedni a spatial műveletek többségét.
- Nagy adathalmaz esetén már lassúvá válik.

Mivel a PostgreSQL adatbázisrendszerhez PostGIS kiegészítés tartozik, amely a térbeli adattípusok , különböző vetületi rendszerek , térbeli műveletek , adatformátumok közti konverzió szolgáltatásokkal rendelkezik, így PostgreSQL adatbázisrendszerre esett a választás.

Ezenkívül a PostgreSQL nagyon jól dokumentált és rengeteg példa kóddal rendelkezik. Több programozási nyelvhez van Application Programming Interface (API).

3.3. Kliens oldali vektoros megjelenítő és szerkesztő

Az alkalmazáshoz olyan megjelenítő eszközt kell találni, amelynek használatához kevés gyakorlatot igényel vagy előképzettség szükségeltetik. Erre jelenleg:

1. Google Maps API [1]
2. Yahoo Maps API [2]
3. Bing Maps API [3]
4. OpenLayers [4]

Kiválasztásuknál a következő szempontokat kellett vizsgálni:

[1] https://developers.google.com/maps/
[2] http://developer.yahoo.com/maps/
[3] http://www.microsoft.com/maps/developers/web.aspx
[4] http://openlayers.org/

- API támogatás.
- Jól dokumentált.
- Ingyenesen elérhető.
- Geometriai objektumok szerkeszthetősége.
- Adatformátumok támogatása import és export esetekben.

Ezek közül az OpenLayers bizonyult a legjobbnak, így ráesett a választás. A következő lépések szükségesek a térkép megjelenítéséhez.

1. OpenLayers inicializálás.
2. Térkép réteg megadás, amely lehet WMS (Web Map Service) vagy szolgáltatók által megadott API-n keresztül.
3. Vektoros réteg inicializálás.
4. Szerkesztési kontrollok inicializálása és callback függvények implementálása.
5. Adatformátumok definiálása.

Mivel ezt a műveletet mindig végre kell hajatni, ha egy térképet szeretnék használni, kód megismétlődés, újrafelhasználhatóság szempontokból is ajánlott volt kitenni egy külön osztályba, amely a térképi funkciókért felelős. Ez így is történt, létrejött az *OpenLayersMap* osztály.

```
  /**
   * @name OpenLayersMap
   * @param String Div container.
   * @param Array Layers identifier (at least one layer required).
   * @param Array Controls names (optional).
   * @return OpenLayersMap.
   * @author Nguyen Thai Binh.
   */
OpenLayersMap(String container, Array layers, Array Controls)
    returns OpenLayersMap

    addCamMap = new OpenLayersMap(
        'map',
        ['osm', 'seccam', 'vect'],
        ['drag', 'vect', 'polygon', 'modify']);
    init = new initializeListeners(addCamMap);
```

3.4. Webes keretrendszer

A keretrendszer kiválasztásakor a következő szempontok kerültek előtérbe:

- Web alapú: Webes keretrendszer.
- Moduláris: Egy egyszerű keretrendszer, amelyhez később igény szerint más komponensek csatlakoztathatóak.
- Rugalmas: Bővíthetőség valamit külső nem keretrendszerbeli komponensekkel való együttműködés.
- Grafikus elemek kezelése.

Ezen szempontokat figyelembe véve a következő keretrendszereket vizsgáltam meg:

1. Akelos
2. JEE
3. Flash CS4
4. Ruby on Rails

3.4.1. Akelos

MVC (Model, Controller, View) keretrendszer PHP-ban.
Magas szintű API-k. Ruby On Rails alapján lett klónozva, feleslegesnek éreztem, hogy egy másik szkript nyelvet tanuljak meg, ahhoz, hogy ugyanazt elérjem Ruby On Rails-ben, amit már ismerek, így nem esett rá a választás.

3.4.2. JEE

Java Enterprise termékek vizsgálati céljából JBoss rendszer választottam, mivel kiforrott, és megbízható.

Nagyon igéretesnek tűnt, főleg Hibernate rugalmassága és Java nyelv miatt. A következő komponenseit néztem meg:

1. Hibernate: rugalmas adat generálási és elérési réteg.
2. Open Faces: felhasználói komponensek könyvtára.
3. JBoss Application Server: alkalmazás szerver.
4. Google Maps plugin: kiegészítő modul google maps-hez.

Azonban a Faces engine kiszámíthatatlan viselkedése javascript-ekkel, le kellett mondanom. Mivel Google Maps API-ja javascriptben íródott , használata javascript megbízható működését igényel.

3.4.3. Flash CS4

Grafikai szempontokat teljesen jól teljesítette, azonban egy nagyon nagy problémája van. Az adatok elérése adatbázison keresztül nagyon körülményes, számomra használhatatlan. Adatbázis kezelés php file-okon keresztül lehet megvalósítani POST/GET metódusokon keresztül.

3.4.4. Ruby on Rails

Ruby on Rails webes keretrendszert választottam. A Ruby on Rails (röviden Rails) a Ruby programozási nyelvre épülő, nyílt forrású (MIT licenc alatti) webalkalmazás-keretrendszer. David Heinemeier Hansson írta 2004-ben, a Basecamp program kódjának felhasználásával.

Technikai háttér. Alapelvei a Don't repeat yourself (ne ismételd magad) és a Convention over Configuration (konvenciók a beállítások előtt): minden információ csak egy helyen szerepel (például egy adatbáziskezelő osztályban nem kell az oszlopokat definiálni, a Rails közvetlenül kiolvassa a nevüket az adatbázisból), és a konvenciókat követő elnevezésekhez automatikusan kódot generál a rendszer (például az adatbázis sales táblája automatikusan hozzárendelődik a Sale osztályhoz). AJAX-támogatása miatt a web 2.0 alkalmazások egyik népszerű keretrendszere.

Az alkalmazás futtatása. Noha a WEBrick, a Rubyban írt webszerver nagyon jó tesztelésre, kész alkalmazások futtatására, különösen nagy terhelés alatt nem alkalmas. A kész alkalmazások deploymentjéhez több megoldás kínálkozik. A Mongrel mellett lehetőség van lighttpd-n vagy IIS-en futtatni az alkalmazásokat. Ugyanakkor a Mongrel parserére, a Rack és az Event Machine-re épített Thin sebessége miatt kedvelt választás. Azonban a deployment könnyedsége miatt a Phusion Passenger lett a hivatalosan ajánlott platform. Ezzel Apache vagy Nginx szerveren futtathatjuk a Rails keretrendszerben írt alkalmazásunkat.

Keretrendszer struktúrája. A Ruby on Rails keretrendszer különböző csomagokat tartalmaz, mint az

ActiveRecord Objektum relációs leképezésért felelős csomag.

ActionController Kérések feldolgozása és válasz küldés kliensnek.

Html ERB Megjelenítés és Elrendezésért felelős csomag.

ActionMailer Email szolgáltatásokért felelős csomag.

Ezeken kívül bárki készíthet kiegészítéseket az alapcsomagok kibővítésére. Ahhoz, hogy a térbeli adatokat könnyen tudjam használni valamint térbeli kereséseket tudjak végezni, felhasználtam a következő kiegészítő csomagokat:

1. *gem 'pg'* PostgreSQL adatbázis driver.

2. *gem 'rgeo'* Geometriai objektumok kezeléséért felelős csomag [5].

3. *gem 'rgeo-geojson'* Geometriai objektumok geojson adatformátumba való konvertálásáért felelős csomag.

4. *gem 'rgeo-activerecord'* Geometriai objektumok relálciós leképezéséért felelős.

5. *gem 'activerecord-postgis-adapter'* PostGIS kiegészítőhöz objektum relációs leképező driver.

Felhasználók kezelését pedig *gem 'devise'* [6] csomagot használtam fel.

Az MVC architektúra. *Model*: Active Record osztály a model réteg magja. Ezen keresztül szinte bármilyen adatbázissal tudunk dolgozni. Alkalmazás írása közben is meg van a lehetőség, hogy adatbázis motort változtassunk mivel a kapcsolatot egy yml fileban van tárolva és Migration modul segítségével bármikor vissza tudjuk állítani a sémát egy másik adatbázison.

View: Active View osztály felel a megjelenítésért. html.erb kiterjesztés megadja a lehetőséget, hogy egyszerre natív html, javascript valamint ruby kódot tudjunk futtatni.

Controller Moduláris komponensekre lehet osztani az alkalmazást. Minden egyes kontroller saját funkcionalitásáért felelős, igy struktúráltabb és átláthatóbb a kód.

A következő címeken linkeken lehet elérni a fent említett keretrendszereket:

Keretrendszer Neve	Leírás
Akelos	http://www.akelos.org/
JEE	Oracle JEE
Flash CS4	http://www.adobe.com/products/flash/
Ruby on Rails	http://rubyonrails.org/

3.5. Projekt struktúra

Egy rails projekt három fontos könyvtárból áll:

- *app*
- *db*
- *public*

[5] https://github.com/dazuma/rgeo
[6] https://github.com/plataformatec/devise

3.5.1. db könyvtár

Ebben vannak az adatbázissal kapcsolatos definíciós file-ok találhatóak, nevezetesen:

1. *schema.rb*: Séma leíró file.
2. *migrate*: ebben a könyvtárban található az adatbázis séma változtatásának verziói. Lehetőséget ad arra, hogy tetszőleges séma verzióra álljunk vissza.

Szintén adatbázissal kapcsolatos definíciós file a *config/database.yml*, amely adatbázis kapcsolatot definiál mind három fejlesztési fázisra:

1. Development
2. Test
3. Production

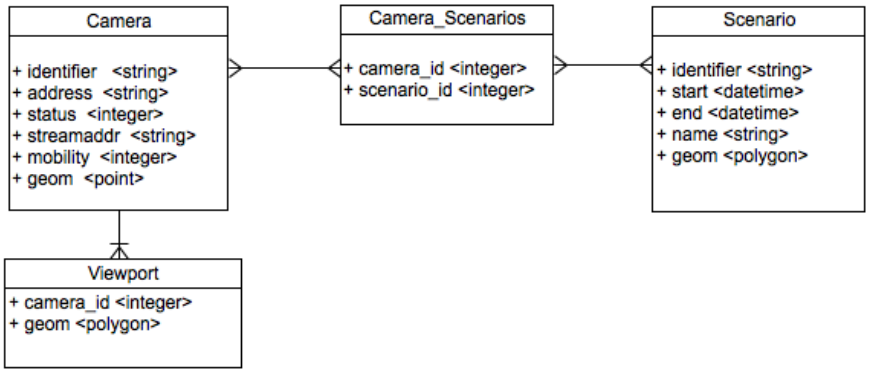
```
development:
  adapter: postgresql
  host: 5432
  port: <portnumber>
  username: <username>
  password: <password>
  database: geowiki_dev

production:
  adapter: postgresql
  host: 5432
  port: <portnumber>
  username: <username>
  password: <password>
  database: geowiki_prod
```

Három fő táblára és két kapcsoló van szükségünk:

1. Camera: Kamerák helyzetének és leíró adatainak tárolására.
2. Viewport: Egy adott kamerához tartozó látómezőinek geometriai adatai.
3. Scenario: Egy eseményhez tartozó leíró adatok.

3.5.2. app könyvtár

Controller, View, Helper, Model osztályok kódja található. A controller adja a logikai megvalósítást. A modelben található az adatbázis objektumokra vonatkozó leképezési objektumok. A helperben olyan osztályok találhatóak, amelyek a megjelenítésben adnak segítséget, itt olyan objektumokra, vagy kód szegmensekre gondolok, amelyeket gyakran használunk. A viewben találhatjuk meg az rhtml file-okat.



3.3. ábra. Adatbázis terv - Entity diagram

Camera ActiveRecord. Camera objektum reláció leképező model. A szokványos SQL lekérdezéseket implementálja alapértelmezésben, valamint kiegészítve:

1. wkt: Geometria objektum Well-known-text adatformátum lekérés.
2. geom_collection: Geometria objektum és hozzátartozó összes viewport geometria objektum lekérése WKT formátumban.

Scenario ActiveRecord. Scenario objektum reláció leképező model. A szokványos SQL lekérdezéseket implementálja alapértelmezésben, valamint kiegészítve a *wkt* geometria objektum WKT formátum lekérése.

Camera Controller. Kamerával kapcsolatos kérések feldolgozása mint egy adott kamera objektum WKT (Well-known-text-format) formátum lekérése az adatbázisból. Funkciói:

1. Kamera létrehozása.
2. Kamerák lekérése.
3. Kamerák keresése.
4. Kamera módosítása.
5. *ajax* Kamerák aszinkron dinamikus lekérése.

Scenario Controller. Eseménnyel kapcsolatos kérések feldolgozása és kiszolgálása. Funkciói:

1. Esemény létrehozása.
2. Esemény módosítása.

3. Események keresése: dátum szerint, név szerint, azonosító szerint.
4. Adott területen levő összes kamerák lekérése.
5. Adott eseményben résztvevő kamerák sorrendjének beállítása és elmentése.

Application Helper. Megjelenítést segéd függvényeket tartalmazó osztály. Funkciói:

1. Dátumok parse-olása.
2. Kamerák állapotától függően a megfelelő ikon hozzárendelése.
3. Kamerák mobilitásától függően a megfelelő ikon hozzárendelése.

3.5.3. public könyvtár

Publikus tartalom, mint javascript, grafikai file-ok, html file-ok. Itt található olyan segéd könyvtárak mint:

1. OpenLayers - térkép megjelenítése és szerkesztésért felelős javascript könyvtár.
2. JQuery - javascript könyvtár.
3. JQuery UI - user interface-el kapcsolatos javascript könyvtár.
4. Icons - ikonok könyvtára.

JQuery. JQuery egy gyors, robusztus, könnyen használható javascript könyvtár. DOM fa szerkezetet könnyedén lehet kezelni elem id-re vagy class id-re hivatkozva. Például:

```
$("#element_id")
$(".class_id")
```

A DOM elem lekérése után, már meghívhatjuk a natív html függvényeket vagy külső javascript library-ket a kiválasztott elemre.

3.6. Térkép szerver

GeoServer. egy nyíltforráskódú szerver, Java nyelven íródott. Digitális térképek szolgáltatására alkalmas WMS (Web Map Service) [7] és WFS (Web Feature Service) [8] szolgáltatásokat keresztül.

Célja. Térinformatikai adatok szolgáltatása Webes felületen keresztül.

[7] http://en.wikipedia.org/wiki/Web_Map_Service
[8] http://en.wikipedia.org/wiki/Web_Feature_Service

Tulajdonságai. Különböző adatformátumokat támogat mint:

1. PostGIS
2. Oracle Spatial
3. ArcSDE
4. DB2
5. MySQL Spatial
6. ESRI Shape file.
7. GeoTIFF

Exportálási adatformátumai lehetnek: KML, GML, ESRI Shapefile, GeoRSS, PDF, GeoJSON, JPEG, GIF, SVG, PNG és még sok más multiméediás tartalom.

4

Felhasználói leírás

4.1. Hardver követelmények

A szoftver használatához a következő hardverekre van szükség:

Processzor Javasolt egy legalább 1GHz Pentium CPU.

Memória (RAM) Minimum: 1024 MB.

Merevlemez Ruby interpreter. Rails 3.0.9. Szükséges Gem csomagok. Összesen kb. 100MB.

Monitor

Billentyűzet

Egér

Mivel ez egy webes alkalmazás, ezért platform független, azonban a következő böngészők alatt támogatott:

1. Mozilla Firefox
2. Google Chrome
3. Openra
4. Safari

4.2. Alkalmazás telepítése és indítása

Sajnos a telepítés nem automatizálható teljes mértékben, ha mégis lehetséges akkor nagyon bonyolult lenne a script és felkell készíteni, hogy a következő operációs rendszereken működjön:

1. Linux disztribúciók.
2. Macintosh OSX.
3. Microsoft Windows.

Ezért a következő részekben tárgyalom az alkalmazás működéséhez szükséges szoftver komponenseket.

4.2.1. Ruby interpreter telepítése

Az alkalmazás magja, mivel a webes keretrendszer Ruby interpreteren fut. Letölthető: http://www.ruby-lang.org/en/ Telepítési útmutatók:

1. Microsoft Windows
   http://ruby.about.com/od/tutorials/a/installruby.htm
2. Linux
   http://ruby.about.com/od/tutorials/a/installruby.htm
3. Mac OSX
   http://ruby.about.com/od/tutorials/a/installruby.htm

4.2.2. Rails telepítése

Miután feltelepítettük a Ruby interpretert, Ruby on Rails webes keretrendszert kell feltelepíteni. Ezekután a Ruby csomagkezelőt kell használni, hogy feltelepítsük a Rails-t:

```
gem install rails
```

Ez a művelet eltarthat körülbelül 5-10 percig. Addig tovább mehetünk az adatbázisrendszer telepítéséez.

4.2.3. PostgreSQL és Postgis telepítése

Telepítési útmutatók:

1. Microsoft Windows
   http://wiki.postgresql.org/wiki/Running_%26_Installing_PostgreSQL_On_Native_Windows
2. Linux
   http://hocuspokus.net/2008/05/install-postgresql-on-ubuntu-804/
3. Mac OSX
   http://macdevcenter.com/pub/a/mac/2002/06/07/postgresql.html

Miután felkerült a PostgreSQL adatbázisrendszer, PostGIS spatial kiegészítőt kell feltelepíteni az útmutatók alapján:

1. Windows
   http://postgis.refractions.net/download/windows/

2. Linux
   Csomag kezelő segítségével könnyen feltelepíthető.

3. Mac OSX
   http://www.kyngchaos.com/software:postgres

4.2.4. GeoServer telepítése

Miután feltelepült a webes keretrendszer és az adatbázis rendszer. GeoServer-t, térkép szervert kell feltelepíteni. Letölthető: http://geoserver.org/display/GEOS/Download Azonban még mielőt nekifutnánk, JAVA JRE-t felkell telepíteni, mert a GeoServer futtatásához Java VM futtató környezet szükséges. Letölthető: http://java.com/en/download/index.jsp Telepítés előtt kitudjuk választani Microsoft Windows és Mac OSX-ra, hogy telepítőt használjunk vagy előre lefordított könyvtárat másoljuk be a megfelelő helyre. Linux esetében a apache web szerverre kell feltelepíteni az Geoserver-t. Útmutatások:

1. Microsoft Windows:
   http://docs.geoserver.org/stable/en/user/installation/windows/index.html

2. Linux
   http://docs.geoserver.org/stable/en/user/installation/linux/index.html

3. Mac OSX
   http://docs.geoserver.org/stable/en/user/installation/osx/index.html

4. WAR (Web Archive) telepítés Java Application Container-be
   http://docs.geoserver.org/stable/en/user/installation/war.html

4.2.5. Forráskód klónozása

Mivel az alkalmazás forráskódját verziókövető rendszerben tárolom, így elérhető bárki számára a https://github.com/posseidon/camapp címen. Letöltésére két módszer lehetséges:

1. Tömörített állományban letölthető az oldalról.

2. Git verziókövető rendszerrel klónozható a saját gépre (feltétel a GIT alkalmazás megléte).

4.2.6. Kiegészítő Gem csomagok telepítése

Miután az alkalmazás forráskódját sikeresen elhelyeztük a saját gépre, bekell lépni az alkalmazás könyvtárába és a következő parancsokat kell kiadni:

```
bunlde install
rake db:migrate
```

bundle. parancs hatására a szükséges kiegészítő csomagokat fogja feltelepíti a Ruby csomagkezelő.

db:migrate. Rake parancs hatására a szükséges adatbázis szerkezetet létrehozza az adatbázisrendszerben, a *database.yml* file-nak megfelelő adatkapcsolati paraméterekkel.

4.2.7. Alkalmazás indítása

A következő paranccsal elindítható az alkalmazás:

```
rails server -p 8000
```

rails server. parancs hatására elindul a webszerver.

-p 8000. kapcsoló definiálja, hogy melyik porton fog hallgatni a webszerver.

4.3. Alkalmazás használata

Az alkalmazás három fő részből áll:

1. Kamera kezelés
2. Esemény kezelés
3. Help

Használatához regisztráció szükséges.

4.3.1. Főoldal

Főodalon három link található kamera, esemény kezelésre és help oldal elérésére. Minden linkhez tartozik funkcionális linkek, amelyekre rákattintva azonnal elérhetőek a alfunkciók.

Kamerához tartozó alfunkciók:

- Kamerák keresés és megtekintés.
- Kamera felvitel.
- Kamera leíró adatainak módosítása.

4.1. ábra. Főoldal

- Help oldalra mutató link.

Eseményhez tartozó alfunkciók:

- Események keresése és megtekintése.
- Esemény felvitel.
- Esemény leíró adatainak és geometriájának módosítása.
- Help oldalra muatató link.

4.3.2. Bejelentkezés és regisztráció

Jobb felső sarokban található két link regisztráció és bejelentkezésre. Regisztráció esetén a következő adatokat kell megadni:

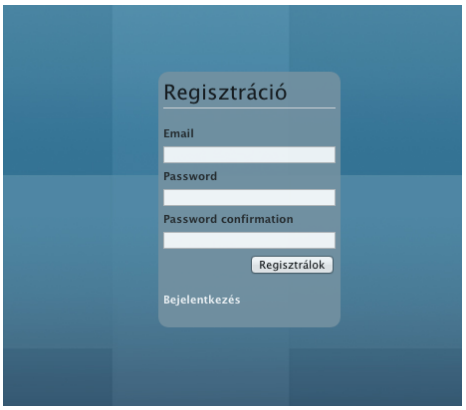
- Email cím
- Jelszó

Bejelentkezés esetén email cím és jelszó alapján történik a felhasználó azonosítás.

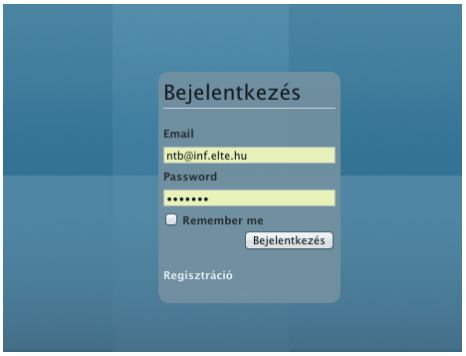
4.3.3. Kamera nézet

Kamerák főoldalon a következő funkciókat tartalmaz:

1. Cím kereső: egy cím megadása után, az alkalmazás betölti a keresett címhez tartozó térképet.
2. Aktív kamerák listája.
3. Visszalépés a főoldalra.



4.2. ábra. Felhasználó regisztráció



4.3. ábra. Felhasználó bejelentkezés

4. Kamera felvitel.

5. Aktív kamerák módosítása.

6. Aktív kamera video stream-jének megtekintése.

Egy felvett kamera módosításakor a hozzátartozó geometria és leíró adatokat tudjuk módosítani.

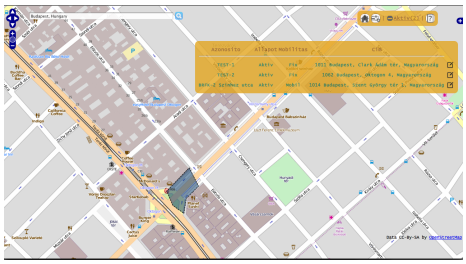
Geometria módosító vagy elhelyezés funkcióira vonatkozó segítség a Help oldalon található.

4.3.4. Esemény nézet

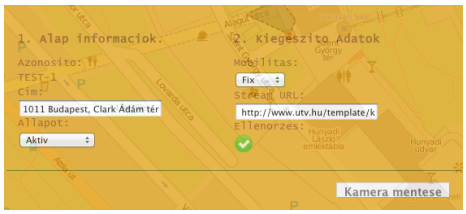
Esemény főoldalon a következő funkciókat tartalmaz:

1. Aktív események listája.

2. Aktív eseményben résztvevő kamerák megtekintése.

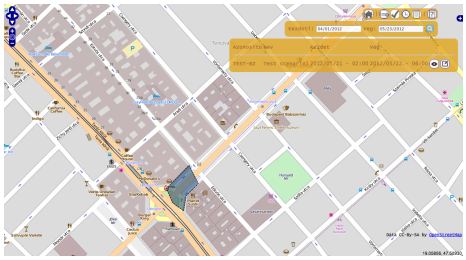


4.4. ábra. Kamera kezelés főoldal



4.5. ábra. Kamera leíró adatainak módosítási form-ja.

3. Aktív esemény módosítása.
4. Események keresése dátum vagy azonosító alapján.
5. Főoldalra navigálás.
6. Help oldalra navigálás.

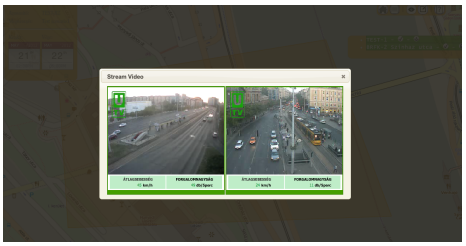


4.6. ábra. Esemény főoldal.

Egy esemény megtekintése során résztvevő kamerák listája, kezdeti és vég dátuma, valamint a kamerákhoz tartozó státusz, mobilitás ikonok találhatóak. Kamerák megtekintés linkre kattintva a résztvevő kamerák video stream-je jelenik meg megadott sorrendben.

Geometria módosító vagy elhelyezés funkcióira vonatkozó segítség a Help oldalon található.

4.7. ábra. Esemény megtekintése.



4.8. ábra. Eseményhez tartozó kamerák video streamje.

4.3.5. Help nézet

Help oldalon:

- Navigációs linkek.
- Geometria módosításához tartozó funkciók.
- Kamera és eseményekre vonatkozó műveletek leírása.
- Homokozó, ahol a felhasználó begyakorolja a térképi elemek használatát és módosítását.

4.9. ábra. Navigációs ikonokhoz tartozó leírás.