**Universitatea Alexandru Ioan Cuza Iași**

Facultatea de Informatică



LUCRARE DE LICENŢĂ

iPhotoTag

**propusă de**

Octavian Cristea

Sesiunea: *iulie, 2015*

**Coordonator ştiinţific**

Lect. Dr. Ionuţ Pistol

UNIVERSITATEA ALEXANDRU IOAN CUZA IAŞI

FACULTATEA DE INFORMATICĂ

iPhotoTag

Octavian Cristea

Sesiunea: *iulie,2015*

**Coordonator ştiinţific**

Lect. Dr. Ionuţ Pistol

DECLARAŢIE PRIVIND ORIGINALITATE ŞI RESPECTAREA

DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de licenţă cu titlul „*iPhotoTag*” este scrisă de

mine şi nu a mai fost prezentată niciodată la o altă facultate sau instituţie de învăţământ

superior din ţară sau străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele

preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a

plagiatului:

* toate fragmentele de text reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi deţin referinţa precisă a sursei;
* reformularea în cuvinte proprii a textelor scrise de către alţi autori deţine referinţa

precisă;

* codul sursă, imaginile etc. preluate din proiecte *open*-*source* sau alte surse sunt

utilizate cu respectarea drepturilor de autor şi deţin referinţe precise;

* rezumarea ideilor altor autori precizează referinţa precisă la textul original.

Iaşi, 26.06.2015

Absolvent *Octavian Cristea*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura în original)

DECLARAŢIE DE CONSIMŢĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*iPhotoTag*”, codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date de test

etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” Iași să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale

programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezentei

lucrări de licenţă.

Iaşi, *26.06.2015*

Absolvent *Octavian Cristea*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura în original)

Cuprins

[Introducere 6](#_Toc423390942)

[Motivaţie : 6](#_Toc423390943)

[Pârghii ale tehnologiei în momentul actual 6](#_Toc423390944)

[Obiective generale : 7](#_Toc423390945)

[Descrierea aplicaţiei : 8](#_Toc423390946)

[Contribuţii 9](#_Toc423390947)

1. [Tehnologii existente 10](#_Toc423390948)
   1. [Google Geocoding 10](#_Toc423390949)
      1. [Formatul unui request la API : 10](#_Toc423390950)
      2. [Parametrii necesari unui request : 10](#_Toc423390951)
      3. [Exemplu request : 11](#_Toc423390952)
      4. [Coduri “status” ale raspunuslui : 13](#_Toc423390953)
      5. [Concluzii Google Geocoding API 17](#_Toc423390954)
   2. [Geonames 18](#_Toc423390955)
      1. [Specificaţii : 18](#_Toc423390956)
      2. [Structura unei înregistrări XML : 19](#_Toc423390957)
      3. [Tipurile de clase : featureClass 19](#_Toc423390958)
      4. [Coduri – featureCodes 21](#_Toc423390959)
      5. [Utilizarea API-ului 26](#_Toc423390960)
      6. [Tool-uri : 27](#_Toc423390961)
      7. [Plugins 27](#_Toc423390962)
      8. [Frameworks 27](#_Toc423390963)
      9. [Funcţionalităţi – funcţiile / metodele oferite 27](#_Toc423390964)
      10. [Concluzii Geonames API 29](#_Toc423390965)
2. [Analiza și proiectarea aplicației 30](#_Toc423390966)
   1. [Tipul aplicației 30](#_Toc423390967)
   2. [Componentele unei aplicații Android 30](#_Toc423390968)
      1. [Activities – 31](#_Toc423390969)
      2. [Services 31](#_Toc423390970)
      3. [Broadcast Receivers 31](#_Toc423390971)
      4. [ContentProviders 32](#_Toc423390972)
      5. [Componente adiționale 32](#_Toc423390973)
   3. [Scopul aplicației 33](#_Toc423390974)
   4. [Construcţia aplicaţiei 33](#_Toc423390975)
   5. [MainActivity 34](#_Toc423390976)
   6. [Modulul camerei de fotografiat 35](#_Toc423390977)
   7. [Modulul Galeriei de imagini 37](#_Toc423390978)
   8. [Modulul de afişare al fotografiei 39](#_Toc423390979)
      1. [Funcţionalitatea din background 40](#_Toc423390980)
      2. [Vizualizarea datelor geografice 42](#_Toc423390981)
      3. [Catalogarea înregistrărilor din vectorii punctelor de interes . 45](#_Toc423390982)
      4. [Afişarea rezultatelor obţinute pentru o locaţie 46](#_Toc423390983)

[Concluzii 48](#_Toc423390984)

[Posibile direcţii viitoare 48](#_Toc423390985)

[Bibliografie 49](#_Toc423390986)

# Introducere

## Motivaţie :

Am ales această temă pentru că mi-a permis să îmi realizez una din ideile pe care le am în legătură cu ce aş putea face eu la nivelul actual cu tehnologia şi să creez totodată şi o aplicaţie de un real folos . Fiecare dintre noi a fost poate la un moment dat într-o excursie şi ca în orice excursie vor fi şi nelipsitele fotografii , iar o fotografie practic marchează un moment de timp , ce-ar fi dacă am avea mai mult de o fotografie , dacă atunci când ne vom uita din nou peste ea , vom avea disponibilă şi locaţia unde a fost facută , ba mai mult , orice informaţie posibilă despre obiectul fotografiat . Dacă am avea o poză la un monument istoric , o statuie , o cladire cu anumită importanţă în acea regiune şi pe lângă însăşi fotografia am avea şi datele relevante despre obiectivul fotografiat ; astfel folosindu-mă de nivelul actual al tehnologiei , de serviciile puse la dispoziţie de furnizorii de baze de date cu aspect de geolocalizare , am reuşit să dezvolt o aplicaţie care realizează acest lucru .

## Pârghii ale tehnologiei în momentul actual

La momentul actual , sunt puse la dispoziţie servicii web , API-uri de către furnizorii de date cu caracter geocoding , printre cei mai importanţi : Google , care pune la dispoziţie serviciile Google Maps , Google Geocoding , Google Places cât şi librării / exemple de cod cu integrări ale servicilor pentru a propulsa dezvoltarea aplicaţiilor de această natură . Geonames este din nou un furnizor de date important , având o bază de date de acoperă un areal global , chiar şi la nivel de mări şi oceane , fiind printre primii cu această inovaţie , cu un tag special pentru elemente topografice de tip undersea .

Fiecare furnizor ofera serviciile şi funcţionalităţile lor , dar nu s-a lucrat încă la un sistem standard de catalogare a elementelor topografice , fapt care reprezintă una din probleme , anume standradizarea tag-urilor de clasificare a punctelor de interes , fiecare dintre aceştia oferind un anumit mod de descriere cu un grad de identificare mai general sau mai specific . În prezent se lucrează la această problemă şi se dezvoltă convenţii de numire şi de clasificare a acestor taguri , iar pentru o rapida dezvoltare a acestor taguri de clasificare mai toţi furnizorii oferă şi posibilitatea utilizatorilor / dezvoltatorilor de aplicţii de a crea noi taguri pentru a le satisface nevoile , în situaţiile în care au nevoie de definirea unui tag cu putere de clasificare , discriminare mai mare .

Mare pare din furnizorii serviciilor de acest tip au pus la dispoziţie dezvoltatorilor librării pentru cele mai populare limbaje de programare la momentul actual , exemple de utilizare ale API-urilor , structura de clasificare a datelor , multi dintre aceştia oferind acest serviciu gratuit , în limita unui anumit număr de requesturi zilnice . Este clar ca se doreşte o creştere în această direcţie a tehnologiei , eliminarea costurilor parctic făcând ca aceste servicii să se adreseze unei clase mult mai largi de dezvoltatori .

## Obiective generale :

* Aplicaţia trebuie să fie capabilă sa realizeze fotografii .
* Să determine coordonatele geografice ale fotografiei în momentul realizării acesteia .
* Pe baza coordonatelor geografice înregistrate , să surpindă un areal formulând un request la un serviciu web cu date de tip geocoding .
* Interpretarea raspunsului , primit în urma requestului .
* Catalogarea după importanţă şi relevanţă în context a datelor obţinute
* Stocarea infromaţiilor obţinute în baza de date internă a aplicaţiei

## Descrierea aplicaţiei :

iPhotoTag este o aplicaţie android destinată asocierii de date de tip geocoding fotografiilor realizate de dispozitivul pe care este instalată , parctic orice fotografie realizată prin intermediul aplicaţiei va avea ca şi date asociate coordonatele geografice – latitudine şi longitudine a locaţiei unde a fost realizată fotografia . Folosindu-se de aceste coordonate , aplicaţia formulează un request către un serviciu web , mapând astfel un areal de puncte de interes în perimetrul coordonatelor geografice determinate la momentul realizării fotografiei .

# Contribuţii

Aplicaţia iPhotoTag este o aplicaţie destinată realizării de fotografii şi de asociere a fotografiei cu locaţia acesteia , punctul de unde a fost facută poza , sub formă de coordonate , latitudine şi longitudine . Aplicaţia , pe baza coordonatelor geografice , va face o serie de requesturi la servicii web ale bazelor de date cu caracter de geocoding , realizează o catalogare a arealului înconjurător , în primă instanţă una în puncte de interes şi monumente , landmark-uri găsite în imediata proximiate a coordonatelor geografice , cea de-a doua catalogare va împărţi cele două categorii în grupări de obiective aparţinând aceleaşi clase , exemple de clase ( restaurant , monument , forme de relief naturale) .

Dacă în arealul surprins avem landmark-uri sau monumente cu o anumită semnificaţie pentru regiunea unde a fost realizată fotografia , este trimis un al doilea request către un api pus la dispoziţie de baza de date enciclopedică wikipedia , care va returna informaţii relevante dacă va gasi un match despre acea locaţie , împreună cu adresa web asociată articolului . Requesturile se vor realiza o singură data , răspunsurile fiind înregistrate în baza de date internă a aplicaţiei , pentru a mări eficienţa şi a reduce costurile .

# Tehnologii existente

## Google Geocoding[[1]](#footnote-1)

*Este un API adresat în mare parte dezvoltatorilor de website-uri şi a aplicaţiilor*

*mobile care doresc să folosească date de tip geocoding în cadrul hărţilor puse la dispoziţie de API-urile Google Maps* .

*Pentru a utiliza API-ul este necearsă o cheie API dat fiind faptul că fiecare request la serviciu va avea ca şi parametru inclus o asemenea cheie . În mare parte , cheia este folosită pentru managementul aplicaţiei , permite monitorizarea utilizării API-ului prin intermediul consolei pentru developeri de la Google , limitările de acces făcându-se pe baza cheii şi nu a adresei IP . API-ul are însă câteva limitări impuse pentru aplicaţiile ce ţin de server-side Geocoding , anume pentru dezvoltatorii care folosesc varianta free apelul la serviciu este limitat la 2.500 de requesturi pe o perioadă de 24 de ore şi maxim 5 requesturi pe secundă* .

### Formatul unui request la API :

https://maps.googleapis.com/maps/api/geocode/*output*?*parameters*

**Output :** ( la latitudinea developerului , fie un obiect de tip JSON sau XML )

### Parametrii necesari unui request :

* **Address** –*adresa străzii pentru care se doreşte conversia geocode în formatul utilizat de serviciul poştal naţional al ţării în cauză , eventual alte elemente adiţionale*

Sau

* **Components** – *un filtru de tip componentă pentru care se doreşte conversia , elemente de tip „components” pot fi acceptate şi ca parametru suplimentar dacă parametrul „address” este furnizat* .

### Exemplu request :

https://maps.googleapis.com/maps/api/geocode/json?address=1600+Amphitheatre+Parkway,+Mountain+View,+CA&key=***API\_KEY***

Output-ul returnat un obiect JSON :

{  
   "results" : [  
      {  
         "address\_components" : [  
            {  
               "long\_name" : "1600",  
               "short\_name" : "1600",  
               "types" : [ "street\_number" ]  
            },  
            {  
               "long\_name" : "Amphitheatre Pkwy",  
               "short\_name" : "Amphitheatre Pkwy",  
               "types" : [ "route" ]  
            },  
            {  
               "long\_name" : "Mountain View",  
               "short\_name" : "Mountain View",  
               "types" : [ "locality", "political" ]  
            },  
            {  
               "long\_name" : "Santa Clara County",  
               "short\_name" : "Santa Clara County",  
               "types" : [ "administrative\_area\_level\_2", "political" ]  
            },  
            {  
               "long\_name" : "California",  
               "short\_name" : "CA",  
               "types" : [ "administrative\_area\_level\_1", "political" ]  
            },  
            {  
               "long\_name" : "United States",  
               "short\_name" : "US",  
               "types" : [ "country", "political" ]  
            },  
            {  
               "long\_name" : "94043",  
               "short\_name" : "94043",  
               "types" : [ "postal\_code" ]  
            }  
         ],  
         "formatted\_address" : "1600 Amphitheatre Parkway, Mountain View, CA 94043, USA",  
         "geometry" : {  
            "location" : {  
               "lat" : 37.4224764,  
               "lng" : -122.0842499  
            },  
            "location\_type" : "ROOFTOP",  
            "viewport" : {  
               "northeast" : {  
                  "lat" : 37.4238253802915,  
                  "lng" : -122.0829009197085  
               },  
               "southwest" : {  
                  "lat" : 37.4211274197085,  
                  "lng" : -122.0855988802915  
               }  
            }  
         },  
         "place\_id" : "ChIJ2eUgeAK6j4ARbn5u\_wAGqWA",  
         "types" : [ "street\_address" ]  
      }  
   ],  
   "status" : "OK"  
}

Observație – răspunsul JSON conține două elemente rădăcină :

* “status” – conține metadate ale request-ului
* “results” – conține informațiile din vectorul geocode și informații geometrice

### Coduri “status” ale raspunuslui :

* “OK” – indică faptul ca nu a avut loc nici o eroare şi adresa a fost parsată cu succes cu cel putin un rezultat de tip geocode returnat
* “ZERO\_RESULTS” – indică faptul că parsarea adresei s-a realizat cu success , dar nu a fost găsit nici un rezultat
* “OVER\_QUERY\_LIMIT” – indică faptul ca s-a depășit numărul de cereri alocate pe secundă sau depăşirea numărului maxim de cereri alocate pe zi .
* “INVALID\_REQUEST” – indică faptul că unul din parametrii request-ului lipsește
* “REQUEST\_DENIED” – indică faptul că request-ul a fost respins
* “UNKNOWN\_ERROR” – indică faptul ca request-ul nu a putut fi procesat din cauza unei erori ale serverului . Este indicat a reface apelul .

Un răspuns al unui request este format din următoarele componente :

* types[] - este un vector care indică tipul rezultatului returnat
* formatted\_address – este un String ce conține denumirea locației , de obicei echivalentul adresei poştale în formatul impus de țara de proveniență
* address\_components[] – vector cu următoarele componente
  + types[] – vector ce indică tipul locației
  + long\_name – reprezintă o descriere textuală a adresei
  + short\_name – reprezintă abrevierea locației ex Alaska – AK
* postcode\_localities[] – este un vector ce denotă localitățile reunite de un cod poștal , acest câmp este prezent atunci când un cod poștal cuprinde mai multe localități .
* geometry – cu următoarele câmpuri :
  + location – ce conține valorile de latitudine și longitudine
  + location\_types – înregistrează informații suplimentare asupra locației căutate cu următoarele valori suportate :
    - „ROOFTOP” – *indică faptul ca rezultatul returnat este un geocode foarte precis cu acuratețe la nivel de adresă de stradă*
    - „RANGE\_INTERPOLATED” – reflectă o aproximare ( de obicei pe un drum) interpolată între două puncte precise (intersecții) , acest câmp este returnat cand nu sunt disponibilie coduri de tip „ROOFTOP” pentru o adresă a unei străzi
    - „GEOMETRIC\_CENTER” – indică faptul că rezultatul este o arie geometrică
    - „APPROXIMATE” – indică faptul ca rezultatul este aproximativ
  + viewport – conține o recomandare în ceea ce privește viewport-ul de afișare a rezultatului returnat specificând două valori pentru latitudine și longitudine , reprezentând colțul de sud-vest și cel de nord-est al chenarului viewport-ului .
  + bounds – reține limitele viewport-ului , arealului căutat .
* partial\_match – indică faptul ca rezultatul returnat nu este un match exact al cererii pentru o locație
* place\_id – este un identificator unic al unei locații de obicei folosit pentru a marca centre comerciale , restaurante , business centers , etc . care pune la dispoziție informații precum : număr de telefon , program , review-uri ale clienților și altele .

Obs. Address Types și Address Component Types – ambele conțin un vector denumit „types[]”. Pentru address type , vectorul types[] conține inormații despre adresa stăzii , țara sau entitatea politică , în timp ce „types[]” din address\_components[] indică tipul fiecărei părți din adresă , anume numărul străzii sau al țării și altele , de exemplu multe orașe pot avea tag-uri atat „political” cât și „locality” .

Următoarele sunt câmpuri compune prezente atât în address types cât și în address component types :

* street\_address
* route
* intersection
* political
* country
* administrative\_area\_level\_1 – tag administrativ ca regiuni , judete , municipii , orase .
* administrative\_area\_level\_2
* administrative\_area\_level\_3
* administrative\_area\_level\_4
* administrative\_area\_level\_5
* colloquial\_area – indică numele colocvial al entității
* sublocality
* neighborhood
* premise
* subpremise
* postal\_code
* natural\_feature
* airtport
* park
* point\_of\_interest

\*Notă – lista nu este exhaustivă și este subiect al schimbării . Alți posibili parametrii

* floor
* establishment
* parking
* post\_box
* postal\_town
* sublocality
* street\_number
* room
* bus\_station
* train\_station
* transit\_station

Google Geocoding API – are și posibilitatea de filtrare a rezultatelor după componente . Se pot folosi ca și filtre următoarele atribute :

* route
* locality
* administrative\_area
* postal\_code
* country

Accesul la serviciul de geocoding este realizat asincron , necesită realizarea unei funcții de callback de executat la terminarea requestului , care va realiza procesarea rezultatelor returnate .

Accesul la Google Maps API este realizat via obiectul de tip google.maps.Geocoder , apelul de inițializare este realizat de metoda Geocoder.geocode() care pasează un obiect GeocodeRequest ce va cuprinde datele de căutare a poziției geografice dorite :

GeocodeRequest – câmpuri :

{  
 address: string,  
 location: LatLng,  
 bounds: LatLngBounds,  
 region: string  
}

* address - adresa la care se dorește conversia în geocode
* latLang – latitudinea și longitudinea pentru care se dorește obținerea celui mai apropiat rezultat de tip adresa în limbaj natural
* bounds (optional)– atribut ce trasează si mai mult limitele impuse de câmpul latLang
* region (optional) – codul regiuni
* componentRestrictions (opţional) – utilizat pentru a restricționa aria de căutare la o regiune specifică .

Obs. Se poate pasa numai câmpul “address” sau doar “location” anume latitudine și longitudine în cazul acesta realizându-se reverse geocoding .

Răspunsul obținut este stocat într-un obiect literal “GeocoderResults” care are următoarea formă :

results[]: {  
 types[]: string,  
 formatted\_address: string,  
 address\_components[]: {  
   short\_name: string,  
   long\_name: string,  
   postcode\_localities[]: string,  
   types[]: string  
 },  
 partial\_match: boolean,  
 geometry: {  
   location: LatLng,  
   location\_type: GeocoderLocationType  
   viewport: LatLngBounds,  
   bounds: LatLngBounds  
 }  
}

### Concluzii Google Geocoding API

Google Geocoding API oferă instrumente puternice de dezvoltare , complexitatea unei aplicaţii şi cât de multe poate realiza o aplicaţie rămânând la latitudinea developerului , la limitele imaginaţiei acestuia , dar cu toate acestea , baza de date a punctelor de interes acoperă doar câteva regiuni , în ciuda puterilor de manipulare numeroase ale informaţiei şi capabilităţilor acestui API de a realiza o mapare cu acurateţe maximă a elementelor surprinse într-un areal.

## Geonames[[2]](#footnote-2)

### Specificaţii :[[3]](#footnote-3)

*API Provider - <http://www.geonames.org/>*

*API Homepage - <http://www.geonames.org/export/>*

*API Kits -* Java, Ruby, Python, PHP, PERL, Obj-C

*API forum -* [http://groups.google.com/group/geonames](http://groups.google.com/group/geonames" \t "_blank)

Developer support : [http://forum.geonames.org/](http://forum.geonames.org/" \t "_blank)

Protocol / Formats : [XML](http://www.programmableweb.com/data-formats/xml), [JSON](http://www.programmableweb.com/data-formats/json), [JSONP](http://www.programmableweb.com/data-formats/jsonp), [RDF](http://www.programmableweb.com/data-formats/rdf), [CSV](http://www.programmableweb.com/data-formats/csv), [RSS](http://www.programmableweb.com/data-formats/rss), [KML](http://www.programmableweb.com/data-formats/kml), [REST](http://www.programmableweb.com/data-formats/rest)

Authentication Mode : Security Token

*Geonames este o bază de date geografică ce acoperă arealul tuturor ţărilor conţinând peste opt milioane de nume de locaţii şi care este disponibilă de descărcat sau utilizat în regim gratuit .*

Geonames pune la dispoziţie pentru facilitarea accesului la baza de date un *API - REST* . Pentru a utiliza API-ul este necesară crearea unui cont , deoarece comparativ cu Google Geocoding unde accesul se face pe bază de cheie , *accesul la API-ul geonames se face pe bază de username , astfel în componenţa fiecărui request la API va fi inclus şi username-ul în lista de parametrii* .[[4]](#footnote-4)

Ca şi structură de înregistrare a unei locaţii , altfel spus structura tag-ului unei locaţii ca şi atribute relavante fiecare locaţie va avea un ID unic , o clasă de apartenenţă

featureClass , un cod de clasificare din cadrul clasei – featureCode , *codul ISO3166-2[[5]](#footnote-5),* anume codul de judeţ (Ex : IS , BC, TM , BH , etc.) , ţara de apartenenţă , coordonata latitudine şi coordonata longitudine a punctului , distanţa de la coordonatle precizate la punctul de interes din tag şi locaţia de unde este privit obiectul .

### Structura unei înregistrări XML :

<geoname>

<toponymName>Palatul Culturii Iași</toponymName>

<name>Palatul Culturii Iași</name>

<lat>47.15737</lat>

<lng>27.5869</lng>

<geonameId>6949053</geonameId>

<countryCode>RO</countryCode>

<countryName>Romania</countryName>

<fcl>S</fcl>

<fcode>PAL</fcode>

<fclName>spot, building, farm</fclName>

<fcodeName>palace</fcodeName>

<population/>

<asciiName>Palatul Culturii Iasi</asciiName>

<alternateNames/>

<elevation>0</elevation>

<continentCode>EU</continentCode>

<adminCode1 ISO3166-2="IS">23</adminCode1>

<adminName1>Iaşi</adminName1>

<adminCode2/>

<adminName2/>

<timezone dstOffset="3.0" gmtOffset="2.0">Europe/Bucharest</timezone>

<distance>0.26348</distance>

</geoname>

### Tipurile de clase : featureClass [[6]](#footnote-6)

1. Administrative Boundary Features - reuneşte şi desemnează regiuni administrative ale unei ţări , împarte în subdiviziuni şi realizează catalogarea de la nivel de ţară , la judeţ , reşedinţă de judeţ , municipiu , oraş , comună , sat , areal .

**H.** Hydrographic Features – sumează mulţimea codurilor ce descriu obiective hidrografice din areal

**L.** Area Features – desemnează tipuri de regiuni / zone , Ex : zonă industrială , business center , zonă agricolă , zonă minieră etc.

**P.** Populated Place Features – reuneşte aglomerări de clădiri şi populaţii , deseori cu referinţă la oraşe , sate , aglomerări de mase de persoane

**R.** Road / Railroad Features – mulţimea căilor de transport , străzi , cai de linie ferata , poteci , tuneluri , autostrăzi

**S.** Spot Features – mulţime a punctelor de interes – magazin , aeroport , instituţie , bancă etc.

**T**. Hypsographic Features – reuneşte elementele de relief – zone muntoase , depresiuni câmpii , lagune , limane , delte , insule etc.

**U**. Undersea Feature – mulţime a elementelor subacvatice

**V.** Vegetation Features – mulţime de descriere a arealelor de păduri , câmpii , plantaţii de arbori şi arbuşti .

### Coduri – featureCodes [[7]](#footnote-7)

Lista completă a codurilor cuprinde 675 de coduri care realizează discretizarea elementelor claselor , dintre cele mai importante pentru fiecare clasă precizăm :

**A.**

|  |  |  |
| --- | --- | --- |
| ADM1 | first-order administrative division | a primary administrative division of a country, such as a state in the United States |
| ADM1H | historical first-order administrative division | a former first-order administrative division |
|  |
| LTER | leased area | a tract of land leased to another country, usually for military installations |
| PCL | political entity |  |
| PCLD | dependent political entity |  |
| PCLF | freely associated state |  |

**H.**

|  |  |  |
| --- | --- | --- |
| LK | lake | a large inland body of standing water |
| AIRS | seaplane landing area | a place on a waterbody where floatplanes land and take off |
| ANCH | anchorage | an area where vessels may anchor |

|  |  |  |
| --- | --- | --- |
| BAY | bay | a coastal indentation between two capes or headlands, larger than a cove but smaller than a gulf |
| BNKR | stream bank | a sloping margin of a stream channel which normally confines the stream to its channel on land |

|  |  |  |
| --- | --- | --- |
| CHN | channel | the deepest part of a stream, bay, lagoon, or strait, through which the main current flows |
| CHNL | lake channel(s) | that part of a lake having water deep enough for navigation between islands, shoals, etc. |
| CHNM | marine channel | that part of a body of water deep enough for navigation through an area otherwise not suitable |
| CHNN | navigation channel | a buoyed channel of sufficient depth for the safe navigation of vessels |
| CNFL | confluence | a place where two or more streams or intermittent streams flow together |
| CNL | canal | an artificial watercourse |
| CNLA | aqueduct | a conduit used to carry water |
| CRNT | current | a horizontal flow of water in a given direction with uniform velocity |
| ESTY | estuary | a funnel-shaped stream mouth or embayment where fresh water mixes with sea water under tidal influences |
| FISH | fishing area | a fishing ground, bank or area where fishermen go to catch fish |
| FJD | fjord | a long, narrow, steep-walled, deep-water arm of the sea at high latitudes, usually along mountainous coasts |
| GLCR | glacier(s) | a mass of ice, usually at high latitudes or high elevations, with sufficient thickness to flow away from the source area in lobes, tongues, or masses |
|  |
| LK | lake | a large inland body of standing water |
| LKC | crater lake | a lake in a crater or caldera |
| LKI | intermittent lake |  |
| LKN | salt lake | an inland body of salt water with no outlet |
| LKNI | intermittent salt lake |  |
| PND | pond | a small standing waterbody |
| PNDI | intermittent pond |  |
| PNDN | salt pond | a small standing body of salt water often in a marsh or swamp, usually along a seacoast |
| RF | reef(s) | a surface-navigation hazard composed of consolidated material |
| RFC | coral reef(s) | a surface-navigation hazard composed of coral |
| RPDS | rapids | a turbulent section of a stream associated with a steep, irregular stream bed |
| SEA | sea | a large body of salt water more or less confined by continuous land or chains of islands forming a subdivision of an ocean |
| SPNT | hot spring(s) | a place where hot ground water flows naturally out of the ground |
| STM | stream | a body of running water moving to a lower level in a channel on land |
| SWMP | swamp | a wetland dominated by tree vegetation |
| SYSI | irrigation system | a network of ditches and one or more of the following elements: water supply, reservoir, canal, pump, well, drain, etc. |
| WTRH | waterhole(s) | a natural hole, hollow, or small depression that contains water, used by man and animals, especially in arid areas |

**L.**

|  |  |  |
| --- | --- | --- |
| AMUS | amusement park | Amusement Park are theme parks, adventure parks offering entertainment, similar to funfairs but with a fix location |
| AGRC | agricultural colony | a tract of land set aside for agricultural settlement |
| BTL | battlefield | a site of a land battle of historical importance |
| FLDI | irrigated field(s) | a tract of level or terraced land which is irrigated |
| MILB | military base | a place used by an army or other armed service for storing arms and supplies, and for accommodating and training troops, a base from which operations can be initiated |
| MNA | mining area | an area of mine sites where minerals and ores are extracted |
| NVB | naval base | an area used to store supplies, provide barracks for troops and naval personnel, a port for naval vessels, and from which operations are initiated |
| OAS | oasis(-es) | an area in a desert made productive by the availability of water |
| OILF | oilfield | an area containing a subterranean store of petroleum of economic value |
| PRT | port | a place provided with terminal and transfer facilities for loading and discharging waterborne cargo or passengers, usually located in a harbor |
| QCKS | quicksand | an area where loose sand with water moving through it may become unstable when heavy objects are placed at the surface, causing them to sink |
| RES | reserve | a tract of public land reserved for future use or restricted as to use |
| RESA | agricultural reserve | a tract of land reserved for agricultural reclamation and/or development |
| RESF | forest reserve | a forested area set aside for preservation or controlled use |
| RESH | hunting reserve | a tract of land used primarily for hunting |
| RESN | nature reserve | an area reserved for the maintenance of a natural habitat |
| RESP | palm tree reserve | an area of palm trees where use is controlled |
| RESV | reservation | a tract of land set aside for aboriginal, tribal, or native populations |
| RESW | wildlife reserve | a tract of public land reserved for the preservation of wildlife |
| RGN | region | an area distinguished by one or more observable physical or cultural characteristics |
| RGNE | economic region | a region of a country established for economic development or for statistical purposes |
| RGNH | historical region | a former historic area distinguished by one or more observable physical or cultural characteristics |
| RGNL | lake region | a tract of land distinguished by numerous lakes |
| RNGA | artillery range | a tract of land used for artillery firing practice |
| SALT | salt area | a shallow basin or flat where salt accumulates after periodic inundation |
| SNOW | snowfield | an area of permanent snow and ice forming the accumulation area of a glacier |
| TRB | tribal area | a tract of land used by nomadic or other tribes |

**P.**

|  |  |  |
| --- | --- | --- |
| PPL | populated place | a city, town, village, or other agglomeration of buildings where people live and work |
| PPLA | seat of a first-order administrative division | seat of a first-order administrative division (PPLC takes precedence over PPLA) |
| PPLC | capital of a political entity |  |
| PPLCH | historical capital of a political entity | a former capital of a political entity |
| PPLF | farm village | a populated place where the population is largely engaged in agricultural activities |
| PPLH | historical populated place | a populated place that no longer exists |
| PPLL | populated locality | an area similar to a locality but with a small group of dwellings or other buildings |
| PPLQ | abandoned populated place |  |
| PPLR | religious populated place | a populated place whose population is largely engaged in religious occupations |
| PPLS | populated places | cities, towns, villages, or other agglomerations of buildings where people live and work |
| PPLW | destroyed populated place | a village, town or city destroyed by a natural disaster, or by war |

**R.**

|  |  |  |
| --- | --- | --- |
| OILP | oil pipeline | a pipeline used for transporting oil |
| PRMN | promenade | a place for public walking, usually along a beach front |
| PTGE | portage | a place where boats, goods, etc., are carried overland between navigable waters |
| RD | road | an open way with improved surface for transportation of animals, people and vehicles |
| RDA | ancient road | the remains of a road used by ancient cultures |
|  |
| RJCT | railroad junction | a place where two or more railroad tracks join |
| RR | railroad | a permanent twin steel-rail track on which freight and passenger cars move long distances |
| RRQ | abandoned railroad |  |
| ST | street | a paved urban thoroughfare |
| STKR | stock route | a route taken by livestock herds |
| TNL | tunnel | a subterranean passageway for transportation |
| TNLN | natural tunnel | a cave that is open at both ends |
| TNLRD | road tunnel | a tunnel through which a road passes |
| TNLRR | railroad tunnel | a tunnel through which a railroad passes |
| TNLS | tunnels | subterranean passageways for transportation |
| TRL | trail | a path, track, or route used by pedestrians, animals, or off-road vehicles |

**S.**

|  |  |  |
| --- | --- | --- |
| UNIV | university | An institution for higher learning with teaching and research facilities constituting a graduate school and professional schools that award master's degrees and doctorates and an undergraduate division that awards bachelor's degrees. |
| ZOO | zoo | a zoological garden or park where wild animals are kept for exhibition |
| TMPL | temple(s) | an edifice dedicated to religious worship |
| SNTR | sanatorium | a facility where victims of physical or mental disorders are treated |
| RSRT | resort | a specialized facility for vacation, health, or participation sports activities |
| RECG | golf course | a recreation field where golf is played |
| PYR | pyramid | an ancient massive structure of square ground plan with four triangular faces meeting at a point and used for enclosing tombs |
| PRN | prison | a facility for confining prisoners |
| PMPO | oil pumping station | a facility for pumping oil through a pipeline |
| PAL | palace | a large stately house, often a royal or presidential residence |
| MKT | market | a place where goods are bought and sold at regular intervals |
| HTL | hotel | a building providing lodging and/or meals for the public |
| HSP | hospital | a building in which sick or injured, especially those confined to bed, are medically treated |
| HSPC | clinic | a medical facility associated with a hospital for outpatients |
| CTRF | facility center | a place where more than one facility is situated |
| CTRM | medical center | a complex of health care buildings including two or more of the following: hospital, medical school, clinic, pharmacy, doctor's offices, etc. |
| CTRR | religious center | a facility where more than one religious activity is carried out, e.g., retreat, school, monastery, worship |
| CTRS | space center | a facility for launching, tracking, or controlling satellites and space vehicles |
| BUSTN | bus station | a facility comprising ticket office, platforms, etc. for loading and unloading passengers |
| BUSTP | bus stop | a place lacking station facilities |
| BANK | bank | A business establishment in which money is kept for saving or commercial purposes or is invested, supplied for loans, or exchanged. |
| AIRB | airbase | an area used to store supplies, provide barracks for air force personnel, hangars and runways for aircraft, and from which operations are initiated |
| AIRF | airfield | a place on land where aircraft land and take off; no facilities provided for the commercial handling of passengers and cargo |
| AIRH | heliport | a place where helicopters land and take off |
| AIRP | airport | a place where aircraft regularly land and take off, with runways, navigational aids, and major facilities for the commercial handling of passengers and cargo |
| AIRQ | abandoned airfield |  |
| AMTH | amphitheater | an oval or circular structure with rising tiers of seats about a stage or open space |
| ANS | ancient site | a place where archeological remains, old structures, or cultural artifacts are located |

### Utilizarea API-ului

Geonames pune la dispoziţia developerilor o serie de librării pentru a uşura accesul la baza de date a serviciului web , librării dezvoltate pentru cele mai populare limbaje de programare :[[8]](#footnote-8)

* Java Client
* Objective –C
* GeoNames Ruby
* GeoNames API Ruby
* Durpal mode
* Geopy – Geocoding Toolbox pentru Python
* Perl GeoNames Client
* Zend Framework PHP
* .NET WFC
* .NET 3.5
* jeoQuery – jQuery pentru GeoNames
* Geode pentru NodeJS
* NodeJS streaming parser
* Node-geonames-Client

### Tool-uri :

* GeoNames csv file to shp file convertor
* GeoNames couchbase importer

### Plugins

* Opensearch plugin pentru Firefox si IE7
* Mozzila Plugin
* Geocode plugin for Geomaias

### Frameworks

* **Gisgraphy –** un framework open srouce pentru date de tip GeoNames
* **NGeoNames –** framework pentru pachetul .NET

### Funcţionalităţi – funcţiile / metodele oferite

|  | **WebService** | **XML** | **JSON** | **RDF** | **CSV** | **TXT** | **RSS** | **KML** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | [astergdem](http://www.geonames.org/export/web-services.html#_blank) | [XML](http://api.geonames.org/astergdem?lat=50.01&lng=10.2&username=demo&style=full&type=XML#_blank) | [JSON](view-source:http://api.geonames.org/astergdemJSON?formatted=true&lat=50.01&lng=10.2&username=demo&style=full#_blank) |  |  | [TXT](http://api.geonames.org/astergdem?lat=50.01&lng=10.2&username=demo&style=full#_blank) |  |  |
| 2 | [children](http://www.geonames.org/export/place-hierarchy.html#_blank) | [XML](http://api.geonames.org/children?geonameId=3175395&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/childrenJSON?formatted=true&geonameId=3175395&username=demo&style=full#_blank) |  |  |  |  |  |
| 3 | [cities](http://www.geonames.org/export/JSON-webservices.html#_blank) | [XML](http://api.geonames.org/cities?north=44.1&south=-9.9&east=-22.4&west=55.2&lang=de&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/citiesJSON?formatted=true&north=44.1&south=-9.9&east=-22.4&west=55.2&lang=de&username=demo&style=full#_blank) |  |  |  |  |  |
| 4 | [contains](http://www.geonames.org/export/place-hierarchy.html#_blank) | [XML](http://api.geonames.org/contains?geonameId=2746385&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/containsJSON?formatted=true&geonameId=2746385&username=demo&style=full#_blank) |  |  |  |  |  |
| 5 | [countryCode](http://www.geonames.org/export/web-services.html#_blank) | [XML](http://api.geonames.org/countryCodeXML?lat=47.03&lng=10.2&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/countryCodeJSON?formatted=true&lat=47.03&lng=10.2&username=demo&style=full#_blank) |  |  | [TXT](http://api.geonames.org/countryCode?lat=47.03&lng=10.2&username=demo&style=full#_blank) |  |  |
| 6 | [countryInfo](http://www.geonames.org/export/web-services.html#_blank) | [XML](http://api.geonames.org/countryInfo?lang=it&country=DE&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/countryInfoJSON?formatted=true&lang=it&country=DE&username=demo&style=full#_blank) |  | [CSV](http://api.geonames.org/countryInfoCSV?lang=it&country=DE&username=demo&style=full#_blank) |  |  |  |
| 7 | [countrySubdivision](http://www.geonames.org/export/web-services.html#_blank) | [XML](http://api.geonames.org/countrySubdivision?lat=47.03&lng=10.2&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/countrySubdivisionJSON?formatted=true&lat=47.03&lng=10.2&username=demo&style=full#_blank) |  |  |  |  |  |
| 8 | [earthquakes](http://www.geonames.org/export/JSON-webservices.html#_blank) | [XML](http://api.geonames.org/earthquakes?north=44.1&south=-9.9&east=-22.4&west=55.2&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/earthquakesJSON?formatted=true&north=44.1&south=-9.9&east=-22.4&west=55.2&username=demo&style=full#_blank) |  |  |  |  |  |
| 9 | [extendedFindNearby](http://www.geonames.org/export/web-services.html#_blank) | [XML](http://api.geonames.org/extendedFindNearby?lat=46.2022156980098&lng=6.14569187164307&username=demo&style=full#_blank) |  |  |  |  |  |  |
| 10 | [findNearby](http://www.geonames.org/export/web-services.html#_blank) | [XML](http://api.geonames.org/findNearby?lat=48.865618158309374&lng=2.344207763671875&fclass=P&fcode=PPLA&fcode=PPL&fcode=PPLC&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/findNearbyJSON?formatted=true&lat=48.865618158309374&lng=2.344207763671875&fclass=P&fcode=PPLA&fcode=PPL&fcode=PPLC&username=demo&style=full#_blank) |  |  |  |  |  |
| 11 | [findNearbyPlaceName](http://www.geonames.org/export/web-services.html#_blank) | [XML](http://api.geonames.org/findNearbyPlaceName?lat=47.3&lng=9&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/findNearbyPlaceNameJSON?formatted=true&lat=47.3&lng=9&username=demo&style=full#_blank) |  |  |  |  |  |
| 12 | [findNearbyPostalCodes](http://www.geonames.org/export/web-services.html#_blank) | [XML](http://api.geonames.org/findNearbyPostalCodes?postalcode=8775&country=CH&radius=10&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/findNearbyPostalCodesJSON?formatted=true&postalcode=8775&country=CH&radius=10&username=demo&style=full#_blank) |  |  |  |  |  |
| 13 | [findNearbyStreets](http://www.geonames.org/maps/us-reverse-geocoder.html#_blank) | [XML](http://api.geonames.org/findNearbyStreets?lat=37.451&lng=-122.18&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/findNearbyStreetsJSON?formatted=true&lat=37.451&lng=-122.18&username=demo&style=full#_blank) |  |  |  |  |  |
| 14 | [findNearbyStreetsOSM](http://www.geonames.org/maps/osm-reverse-geocoder.html#_blank) | [XML](http://api.geonames.org/findNearbyStreetsOSM?lat=37.451&lng=-122.18&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/findNearbyStreetsOSMJSON?formatted=true&lat=37.451&lng=-122.18&username=demo&style=full#_blank) |  |  |  |  |  |
| 15 | [findNearByWeather](http://www.geonames.org/export/JSON-webservices.html#_blank) | [XML](http://api.geonames.org/findNearByWeather?lat=42&lng=-2&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/findNearByWeatherJSON?formatted=true&lat=42&lng=-2&username=demo&style=full#_blank) |  |  |  |  |  |
| 16 | [findNearbyWikipedia](http://www.geonames.org/export/wikipedia-webservice.html#_blank) | [XML](http://api.geonames.org/findNearbyWikipedia?lat=47&lng=9&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/findNearbyWikipediaJSON?formatted=true&lat=47&lng=9&username=demo&style=full#_blank) |  |  |  | [RSS](http://api.geonames.org/findNearbyWikipediaRSS?lat=47&lng=9&username=demo&style=full#_blank) |  |
| 17 | [findNearestAddress](http://www.geonames.org/maps/us-reverse-geocoder.html#_blank) | [XML](http://api.geonames.org/findNearestAddress?lat=37.451&lng=-122.18&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/findNearestAddressJSON?formatted=true&lat=37.451&lng=-122.18&username=demo&style=full#_blank) |  |  |  |  |  |
| 18 | [findNearestIntersection](http://www.geonames.org/maps/us-reverse-geocoder.html#_blank) | [XML](http://api.geonames.org/findNearestIntersection?lat=37.451&lng=-122.18&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/findNearestIntersectionJSON?formatted=true&lat=37.451&lng=-122.18&username=demo&style=full#_blank) |  |  |  |  |  |
| 19 | [findNearestIntersectionOSM](http://www.geonames.org/maps/osm-reverse-geocoder.html#_blank) | [XML](http://api.geonames.org/findNearestIntersectionOSM?lat=37.451&lng=-122.18&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/findNearestIntersectionOSMJSON?formatted=true&lat=37.451&lng=-122.18&username=demo&style=full#_blank) |  |  |  |  |  |
| 20 | [findNearbyPOIsOSM](http://www.geonames.org/maps/osm-reverse-geocoder.html#_blank) | [XML](http://api.geonames.org/findNearbyPOIsOSM?lat=37.451&lng=-122.18&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/findNearbyPOIsOSMJSON?formatted=true&lat=37.451&lng=-122.18&username=demo&style=full#_blank) |  |  |  |  |  |
| 21 | [get](http://www.geonames.org/#_blank) | [XML](http://api.geonames.org/get?geonameId=6295630&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/getJSON?formatted=true&geonameId=6295630&username=demo&style=full#_blank) |  |  |  |  |  |
| 22 | [gtopo30](http://www.geonames.org/export/web-services.html#_blank) | [XML](http://api.geonames.org/gtopo30?lat=47.01&lng=10.2&username=demo&style=full&type=XML#_blank) | [JSON](view-source:http://api.geonames.org/gtopo30JSON?formatted=true&lat=47.01&lng=10.2&username=demo&style=full#_blank) |  |  | [TXT](http://api.geonames.org/gtopo30?lat=47.01&lng=10.2&username=demo&style=full#_blank) |  |  |
| 23 | [hierarchy](http://www.geonames.org/export/place-hierarchy.html#_blank) | [XML](http://api.geonames.org/hierarchy?geonameId=2657896&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/hierarchyJSON?formatted=true&geonameId=2657896&username=demo&style=full#_blank) |  |  |  |  |  |
| 24 | [neighbourhood](http://www.geonames.org/export/web-services.html#_blank) | [XML](http://api.geonames.org/neighbourhood?lat=40.78343&lng=-73.96625&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/neighbourhoodJSON?formatted=true&lat=40.78343&lng=-73.96625&username=demo&style=full#_blank) |  |  |  |  |  |
| 25 | [neighbours](http://www.geonames.org/export/place-hierarchy.html#_blank) | [XML](http://api.geonames.org/neighbours?geonameId=2658434&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/neighboursJSON?formatted=true&geonameId=2658434&username=demo&style=full#_blank) |  |  |  |  |  |
| 26 | [ocean](http://www.geonames.org/export/web-services.html#_blank) | [XML](http://api.geonames.org/ocean?lat=40.78343&lng=-43.96625&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/oceanJSON?formatted=true&lat=40.78343&lng=-43.96625&username=demo&style=full#_blank) |  |  |  |  |  |
| 27 | [postalCodeCountryInfo](http://www.geonames.org/export/web-services.html#_blank) | [XML](http://api.geonames.org/postalCodeCountryInfo?&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/postalCodeCountryInfoJSON?formatted=true&&username=demo&style=full#_blank) |  |  |  |  |  |
| 28 | [postalCodeLookup](http://www.geonames.org/export/web-services.html#_blank) |  | [JSON](view-source:http://api.geonames.org/postalCodeLookupJSON?formatted=true&postalcode=6600&country=AT&username=demo&style=full#_blank) |  |  |  |  |  |
| 29 | [postalCodeSearch](http://www.geonames.org/export/web-services.html#_blank) | [XML](http://api.geonames.org/postalCodeSearch?postalcode=9011&maxRows=10&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/postalCodeSearchJSON?formatted=true&postalcode=9011&maxRows=10&username=demo&style=full#_blank) |  |  |  |  |  |
| 30 | [rssToGeo](http://www.geonames.org/rss-to-georss-converter.html#_blank) |  |  |  |  |  | [RSS](http://api.geonames.org/rssToGeoRSS?feedUrl=http://feeds.reuters.com/reuters/worldNews&username=demo&style=full#_blank) | [KML](http://api.geonames.org/rssToGeoRSS?type=kml&feedUrl=http://feeds.reuters.com/reuters/worldNews&username=demo&style=full#_blank) |
| 31 | [search](http://www.geonames.org/export/geonames-search.html#_blank) | [XML](http://api.geonames.org/search?q=london&maxRows=10&lang=es&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/searchJSON?formatted=true&q=london&maxRows=10&lang=es&username=demo&style=full#_blank) | [RDF](http://api.geonames.org/searchRDF?q=london&maxRows=10&lang=es&username=demo&style=full#_blank) |  |  |  |  |
| 32 | [siblings](http://www.geonames.org/export/place-hierarchy.html#_blank) | [XML](http://api.geonames.org/siblings?geonameId=3017382&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/siblingsJSON?formatted=true&geonameId=3017382&username=demo&style=full#_blank) |  |  |  |  |  |
| 33 | [srtm3](http://www.geonames.org/export/web-services.html#_blank) | [XML](http://api.geonames.org/srtm3XML?lat=50.01&lng=10.2&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/srtm3JSON?formatted=true&lat=50.01&lng=10.2&username=demo&style=full#_blank) |  |  | [TXT](http://api.geonames.org/srtm3?lat=50.01&lng=10.2&username=demo&style=full#_blank) |  |  |
| 34 | [timezone](http://www.geonames.org/export/web-services.html#_blank) | [XML](http://api.geonames.org/timezone?lat=47.01&lng=10.2&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/timezoneJSON?formatted=true&lat=47.01&lng=10.2&username=demo&style=full#_blank) |  |  |  |  |  |
| 35 | [weather](http://www.geonames.org/export/JSON-webservices.html#_blank) | [XML](http://api.geonames.org/weather?north=44.1&south=-9.9&east=-22.4&west=55.2&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/weatherJSON?formatted=true&north=44.1&south=-9.9&east=-22.4&west=55.2&username=demo&style=full#_blank) |  |  |  |  |  |
| 36 | [weatherIcao](http://www.geonames.org/export/JSON-webservices.html#_blank) | [XML](http://api.geonames.org/weatherIcao?ICAO=LSZH&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/weatherIcaoJSON?formatted=true&ICAO=LSZH&username=demo&style=full#_blank) |  |  |  |  |  |
| 37 | [wikipediaBoundingBox](http://www.geonames.org/export/wikipedia-webservice.html#_blank) | [XML](http://api.geonames.org/wikipediaBoundingBox?north=44.1&south=-9.9&east=-22.4&west=55.2&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/wikipediaBoundingBoxJSON?formatted=true&north=44.1&south=-9.9&east=-22.4&west=55.2&username=demo&style=full#_blank) |  |  |  |  |  |
| 38 | [wikipediaSearch](http://www.geonames.org/export/wikipedia-webservice.html#_blank) | [XML](http://api.geonames.org/wikipediaSearch?q=london&maxRows=10&username=demo&style=full#_blank) | [JSON](view-source:http://api.geonames.org/wikipediaSearchJSON?formatted=true&q=london&maxRows=10&username=demo&style=full#_blank) |  |  |  |  |  |
|  | **Total** | **36** | **36** | **1** | **1** | **4** | **2** | **1** |

### Concluzii Geonames API

La momentul actual API-ul pus la dispoziţie de Geonames , prin facilitarea accesului la acesta , suportul oferit , sprijinul comunităţii de developeri este unul din cele mai complete API-uri în regim free , ce are funcţionalitate şi modalităţi de manipulare a informaţiei prin care asigură toate elementele necesare dezvoltării aplicaţiilor de acest tip. Pe lângă puterea de manipulare a informaţiei pe care o ofera API-ul , un mare plus vine şi de la baza de date , care are o mapare la nivel global cu un număr enorm de înregistrări de locaţii , fiind una din bazele de date cele mai complete luând în comparaţie cu alţi furnizori de baze de date de aceeaşi natură .

# Analiza și proiectarea aplicației

## Tipul aplicației

iPhotoTag este o aplicație Android , destinată să funcționeze pe dispozitivele cu sistemul de operare respectiv . Sistemul de operare Android a fost creat în principal pentru dispozitivele mobile de tip touchscreen de la telefone , ceasuri , tablete la televizoare , chiar și masini , care permite prin intermediul interfeței o manipulare naturală a elmentelor de pe ecran . *Gama acestor tipuri de aplicații fiind într-o continuă creștere , ca și statistică mai mult de un milion de aplicații android existau în 2013 , acum într-o continuă creștere , iar ca și statistică în rândul developerilor aplicațiilor pentru dispozitivele mobile , 71% dintre aceștia crează aplicații în Android* .[[9]](#footnote-9)

## Componentele unei aplicații Android [[10]](#footnote-10)

Componentele unei aplicații Android reprezintă blocurile esențiale ale acesteia . Modul de legare al componentelor este descris în manifestul aplicației , anume fișierul AndroidManifest.xml care descrie interacțiunea dintre componente .

Principalele Componente care pot fi utilizate în cadrul unei aplicații Android sunt :

* Activities – dictează UI-ul și se ocupă de interacțiunea utilizatorului cu ecranul dispozitivului
* Services – se ocupă de procesele care rulează în background , de obicei procese time-consuming asociate aplicației
* Broadcast Receivers – se ocupă de comunicarea dintre sistemul de operare Android și aplicații
* Content Providers – în general se ocupă de manipularea datelor , mare parte ce țin de problemele ridicate la managementul bazei de date .

Activities – reprezintă un singur display de ecran cu o interfață grafică , pe scurt spus , se ocupă de interacțiunea cu utilizatorul . Ca și exemplu , o activitate poate afișa o listă a e-mail-urilor necitite , alta este destinată citirii mesajelor , iar alta compunerii și trimiterii de mesaje . O activitate reprezintă și punctul de start al unei aplicații Android și are rolul de „launcher” anologie cu metoda „main” a unui program C/C++ sau JAVA . Activitea „launcher” , de start-up a aplicaţiei este marcată în fişierul de configurare al aplicaţiei Android , în AndroidManifest.xml

Orice activitate va fi implementată ca și subclasă a clasei „Activity” .

public class MainActivity extends Activity {

}

Services – componenta care se ocupă de procesările din background , deseori operații de lungă durată . De exemplu un serviciu poate reda în background o piesă muzicală în timp ce userul se află în cadrul altei aplicații sau poate prelua date de pe internet fără a bloca interacțiunea utilizatorlui în cadrul unei activități .

Un serviciu este implementat ca și subclasa a clasei „Service”

public class MyService extends Service {

}

Broadcast Receivers – sunt sisteme care se ocupă cu monitorizarea schimbului de mesaje dintre aplicații și sistemul de operare sau dintre aplicaţie şi altele . De exemplu o aplicație poate iniția un broadcast pentru a aduce la cunoștință alte aplicații că s-a realizat un download pe dispozitiv , care este disponibil pentru prelucrare și utilizare în cadrul acestora .

Un „broadcast receiver” este implementat ca și sublacsă a clasei BroadcastReceiver și fiecare mesaj este trimis sub forma unui obiect de tip „Intent” .

public class MyReceiver extends BroadcastReceiver {

public void onReceive(context,intent){}

}

### ContentProviders

Un content provider reprezintă componenta care pune la dispoziție resurse , date ale unei aplicații celorlalte aplicații care le solicită .

Un „content provider” este implementat ca și subclasă a clasei ContentProvider și este obligat să aibă implementat un set standard de api-uri care să faciliteze altor aplicații capabilitatea de a iniția , realiza tranzacții .

public class MyContentProvider extends ContentProvider {

public void onCreate(){}

}

### 

### Componente adiționale

* Fragmente – reprezintă o porțiune a unei interfețe grafice din cadrul unei activități
* View – se referă la elemente de UI care sunt afișate pe ecran , cum ar fi : Butoane , TextViews , liste .
* Layouts – reprezintă ierarhii de View-uri care controlează poziția și aspectul componentelor UI pe ecranul aplicației
* Resources – reprezintă o serie de componente externe , cum ar fi : variabile , constante , imagini .
* Manifest – reprezintă fișierul de configurare al aplicației.

## Scopul aplicației

Aplicația iPhotoTag are ca și scop asocierea fotografiilor realizate de dispozitiv cu date de tip geocoding , coordonatele geografice la momentul în care fotografia a fost realizată . Pe baza acestor coordonate geografice înregistrate aplicația surpinde un areal înconjurător și realizează o catalogare a elementelor , obiectivelor din proximitate ca și forme de relief , obiective turistice sau puncte de interes.

## Construcţia aplicaţiei

Activitatea principală a aplicaţiei , „MainActivity , care este şi activitatea marcată în AndroidManifest.xml care are rolul de launcher , afişează utilizatorului meniurile de acces a două module , cât şi a unei secţiuni informative cu privire la modul de utilizare şi instrucţiunile aplicaţiei

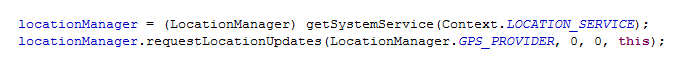


## MainActivity

Este activitatea principală a aplicaţiei , este o exdindere a clasei Activity şi implementează şi interfaţa LocationListener , deoarece la momentul lansării aplicaţiei , aceasta să înceapă să depisteze locaţia dispozitivlui , pentru cazul în care s-ar dori realizarea unei fotografii , ţinând cont că procesul de localizare este unul de durată , circa 2-3 minute este de dorit ca şi localizarea să înceapă odată ce aplicaţia a fost lansată .

Modul de depistare a coordonatelor geografice este la latitudinea utilizatorului , astfel locaţia poate fi obţinută şi prin conexiunea la internet sau prin GPS-ul dispozitivului . Alegerea utilizatorului în funcţie de tipul de localizarea , nu va influenţa rezultatul final , deoarce coordonatele geografice ale fotografii sunt înregistrate în momentul fotografierii , iar requesturile aferente , pentru maparea arealului înconjurător pot fi realizate oricand este posibilă conectarea la internet .

Pentru depistarea locaţiei , am folosit un obiect de tip *LocationManager* , iniţializat şi setat ca să asculte micile variaţii în deplasare ale utilizatorului .



Interfaţa *LocationListener* impune implementarea metodei de monitorizare a variaţilor coordonatelor geografice , anume – onLocationChanged(Location location)

**public** **void** onLocationChanged(Location location) {

*Latitude* = location.getLatitude();

*Longitude* = location.getLongitude();

gpsStatusImage.setImageResource(drawable.*presence\_online*);

}

Prima localizare a coordonatelor geografice , va modifica statusul unui ImageView , indicând dacă poziţia GPS a fost atribuită , odată ce coordonatele geografice sunt atribuite , dacă expirarea unui timp de aproximativ 30 de secunde daca nu mai apar variaţii ale acestor coordonate statusul GPS va fi marcat ca şi inactiv , deoarece modul de localizarea este setat la gradul cel mai amănunţit , precizat în AndroidManifest.XML

*<uses-permission android:name=”android.permission.ACCESS\_FINE\_LOCATION” />*

Statusul GPS va fi marcat inactiv , când aplicaţia determină ca userul nu mai foloseşte aplicaţia , modul de determinare a acestei concluzii legându-se de faptul că până şi cele mai mici mişcari – cm , vor determina un update al coordonatelor locaţiei geografice .

## Modulul camerei de fotografiat[[11]](#footnote-11)

Modulul camerei de fotografiat este creat cu un design foarte simplist , destinat scopului , suprafaţa de afişare a imaginii văzute prin obiectivul dispozitivului conţinând imaginea vizualizată şi un buton pentru a acţiona aparatul .

Pentru acces la camera de fotografiat a dispozitivului am precizat în AndroidManifest.xml permisiunea pentru accces la hardwear-ul dispozitivului , iar pentru a evita complicarea acestui modul în ceea ce priveşte calibrarea camerei foto , am utilizat optiunea de autofocus în momentul realizării fotografiei .

<uses-feature android:name=*"android.hardware.camera"*/>

<uses-permission android:name=*"android.permission.CAMERA"*/>

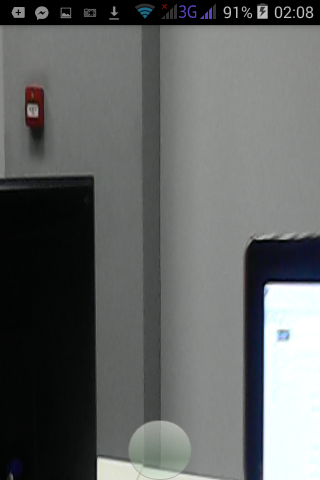
<uses-permission android:name=*"android.hardware.camera.autofocus"*/>

Paşii urmaţi în realizarea modulului :

* Definirea unei activităţi care va avea controlul camerei
* Definirea unei clase de Preview care extinde clasa SurfaceView şi implementează interfaţa SurfaceHolder , pentru a afişa imaginea vizualizată de aparatul dispozitivului în timp real
* Definirea butonului de control , care realizeaza fotografia
* Definirea funcţiei de callback în momentul în care s-a apăsat butonul de realizare al unei fotografii care va da startul unei funcţii asincrone ce va salva imaginea în galerie

Procesele din spatele realizării unei fotografii :

* Este verificată existenţa directorului aplicaţiei – denumit „iPhotoTag” în memoria externă a telefonului , anume pe cardul SD ;
* Se crează o variabila ce va reţine numele fotografiei care este exprimata ca şi timpul în milisecunde a sistemului
* Se salvează coordonatele geografice înregistrate de LocationManager-ul creat şi care a fost pus în stadiul de monitorizare din cadrul activităţii principale în MainActivity
* Se salvează fotografia în folderul ţintă
* Numele fotografiei şi coordonatele geografice sunt înregistrate în *baza de date*[[12]](#footnote-12) a aplicaţiei .



## 

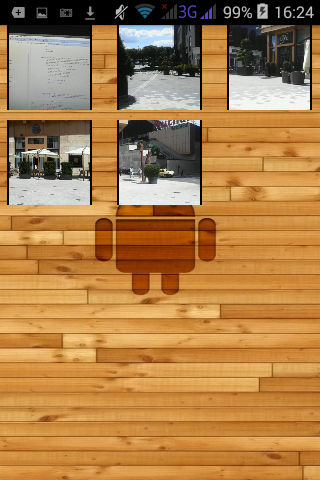
## Modulul Galeriei de imagini

Modulul galeriei de imagini este realizat în activitatea din cadrul aplicaţiei numit GalleryActivity . În momentul accesării galeriei de imagini , dacă acesta va fi primul modul accesat de către utilizator se va asigura faptul ca folderul de stocare al imaginilor destinat aplicaţiei există , iar dacă nu există , va fi creat .

Se vor încărca fotografiile din folderul aplicaţiei , fiecare fotografie fiind decodificată într-un obiect de tip Bitmap , acest obiect va fi mai apoi rotit la dreapta cu 90o din cauza unui bug al aplicaţiilor android care folosesc module proprii de control al camerei de fotografiat , salvarea imaginilor de către module create proprii find ca default de rotire a fotografiei la 90o.

După preluarea fotografiilor şi încărcarea acestora într-un array împreună cu datele asociate din baza de date , latitudine şi longitudine , eventual şi datele referitoare la tag-uri ale fotografiilor ce au fost deja vizualizate şi pentru care s-a realizat requestul de date , fotografiile vor fi afişate în interiorul unui layout de tip GridView care este la rândul lui pus în interiorul unui ScrollView pentru a asigura posibilitatea de scroll , în cazul în care numărul de fotografii încărcate depăşeşte spaţiul ferestrei .

La preluarea imaginilor şi afişarea acestora pe ecran , fiecărei fotografie îi va fi asociat un onClickListener , astfel ca la selecţia unei fotografii aceasta să fie afişată pe tot ecranul în următoarea componentă ce se ocupă strict de datele unei fotografii , anume modulul de afişare şi colectare de date , activitatea PhotoDisplayActivity .



## Modulul de afişare al fotografiei

Partea de management a modulului de afişare al unei fotografii este realizată în activitatea PhotoDisplayActivity . Această activitate este lansată atunci cand utilizatorul face o selecţie asupra unei fotografii din cadrul galeriei de imagini . Activitatea ce ţine de funcţionaliatatea galeriei , anume GalleryActivity practic la selecţia unei fotografii va trimite un Intent către o activitate de tip PhotoDisplayActivity ce va conţine poziţia fotografiei selectate din vectorul fotografiilor încărcate .

PhotoDisplayActivity conţine două elemente de UI , anume o suprafaţă de afişare şi un button – **Areal** – care la accesare va afişa datele geografice asociate fotografiei .



### Funcţionalitatea din background

Odată accesată opţiunea de vizualizare a unei fotografii , în activitatea PhotoDisplayActivity vor fi încărcate datele geografice salvate anterior în baza de date despre acea locaţie , iar dacă fotografia este la prima accesare se vor realiza o serie de requesturi la API-ul de la geonames pentru a prelua datele din arealul înconjurator .

Datele odată obţinute vor fi sotcate în trei vectori

* MyFindNearbyArray
* MyPointsOfInterest
* MyWikipedia

Dacă fotografia este la prima accesare se vor realiza requesturile pentru a obţine datele geografice ale arealului pentru locaţia respectivă , se va face update în baza de date şi se vor popula vectorii enumeraţi mai sus .

Pentru partea de requesturi am creat o clasă care va fi handler al acestor requesturi , clasa GeonamesFunctions . Pentru simplificarea apelurilor la serviciu am utilizat *Volley*[[13]](#footnote-13) , *care este o librărie HTTP care face ca navigarea aplicaţiilor Android mult mai uşoară şi mult mai rapidă* . Clasa GeonamesFunctions cuprinde trei metode :

**public** **void** findNearBy(**double** latitude, **double** longitude)

**public** **void** findNearbyPointsOfInterest(**double** latitude, **double** longitude)

**public** **void** findNearWikipedia(**double** latitude, **double** longitude)

**Prima metodă** – findNearBy – va face un request cu URL –ul :[[14]](#footnote-14)

http://api.geonames.org/findNearbyJSON?lat="+latitude+"&lng="+longitude+"&username=ocristea&radius=1

Parametrii trimişi în URL sunt coordonatele geografice din baza de date asociate pozei , latitudine şi longitudine , al treilea parametru este un nume de cont valid , necesar pentru a avea acces la baza de date geonames , iar ultimul parametru specifică raza de căutare în km .

Răspunsul primit este un obiect de tip JSON care încapsulează de fapt şi de drept un vector de locaţii , urmând ca în interiorul funcţiei să se realizeze parsarea acestuia . Pentru instanţe ale vectorului am creat o clasă şablon – FindNearbyContainer – care va memora obiecte de tipul , aceste obiecte fiind salvate în vectorul – MyFindNearbyArray care este un ArrayList cu cast pentru acest tip de obiecte .

Structura unui obiect din clasa FindNearbyContainer este următoarea :

**public** **class** FindNearbyContainer {

**public** String lng; // memoreaza coordonata longitudine

**public** String lat; // memoreaza coordonata latitudine

**public** String distance; // memoreaza distanta de la //coordonatele trimise ca si parametrii

// la punctul de interes gasit

**public** String geonameId; // id-ul locatiei

**public** String toponymName; // numele toponimului

**public** String name; // numele

**public** String fcode; //featureCode

**public** String fcl; //featureClass

}

**A doua metodă** – findNearbyPointsOfInterest va face un request cu URL-ul :[[15]](#footnote-15)

[http://api.geonames.org/findNearbyPOIsOSMJSON?lat="+latitude+"&lng="+longitude+"&username=ocristea&radius=1](http://api.geonames.org/findNearbyPOIsOSMJSON?lat=%22+latitude+%22&lng=%22+longitude+%22&username=ocristea&radius=1)

Parametrii acestei metode sunt la fel ca şi parametrii primei metode .

Răspunsul primit este tot un obiect de tip JSON care înglobează un vector de intări , pentru acest fel de intrări am definit o clasă şablon – PointOfInterest – cu următoarea structură :

**public** **class** PointOfInterest {

**public** String typeName; // reprezinta tipul punctului de interes

**public** String distance; // distanta de la punct la coordonatele // geografice trimise in request

**public** String name; // denumirea locatiei

**public** String lat; // latitudine

**public** String lng; // longitudine

}

După parsarea intrărilor , acestea vor fi salvate în vectorul – MyPointsOfInterest – care este un array list ce are cast pentru acest tip de date ;

**A treia metodă –** findNearbyWikipedia va face un request cu URL –ul [[16]](#footnote-16)

[http://api.geonames.org/findNearbyWikipediaJSON?lat="+latitude+"&lng="+longitude+"&username=ocristea&lang=en](http://api.geonames.org/findNearbyWikipediaJSON?lat=%22+latitude+%22&lng=%22+longitude+%22&username=ocristea&lang=en)

Parametrii trimişi în acest URL sunt ca şi în celelalte două metode latitudine , longitudine , username , la care se mai adaugă un parametru anume lang , care reprezinta limba de preferinţă în care se doreşte returnat rezultatul .

Răspunsul primit este tot un obiect de tip JSON ce înglobează un vector de intrări de puncte de interes . Pentru parsarea unei intrări am definit o clasă care să aibă rolul de container denumită WikipediaContainer . Structura acestei clase fiind următoarea :

**public** **class** WikipediaContainer {

**public** String title; // titlul locatiei

**public** String summary; // un rezumat al datelor gasite

**public** String distance; // distanta de la coordonate la punct

**public** String feature; // clasa de apartenenta

**public** String wikipediaUrl; // url-ul catre datele complete

**public** String lat; // latitudine

**public** String lng; // longitudine

**public** String rank; // importanta zonala

}

### Vizualizarea datelor geografice

La accesarea butonului – Areal – din activitatea PhotoDisplayActivity se va trimite un Intent către o nouă activitate , anume ArealActivity care de fapt şi de drept este un FragmentActivity pentru ca va îngloba o serie de fragmente *tabbed*[[17]](#footnote-17) , schimbarea acestor fragmente făcându-se prin swipe asupra ecranului .

Primul fragment – *MapFragment*[[18]](#footnote-18) - afişat este cel al localizării pe hartă a locaţiei unde a fost realizată fotografia , într-un fragment a cărui suprafaţă de afişare este un MapView care va prezenta harta oferită prin intermediul API-ului de la Google , anume GoogleMaps .



Pentru a utiliza API-ul Google Maps trebuie specificate în fişierul de configurare al aplicaţiei permisiunile necesare cât şi o cheie de acces :

<uses-permission android:name=*"android.permission.ACCESS\_NETWORK\_STATE"*/>

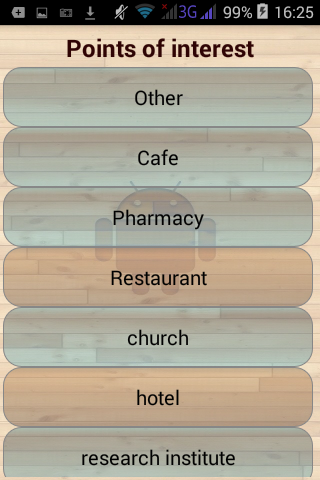
<permission

android:name=*"com.example.iphototag.permission.MAPS\_RECEIVE"* android:protectionLevel=*"signature"*/>

<uses-permission android:name=*"com.example.iphototag.permission.MAPS\_RECEIVE"*/>

<meta-data android:name=*"com.google.android.maps.v2.API\_KEY"* android:value=*"AIzaSyD7MhuwJB7bktjy4MUuM0aw0sR3qZvJUaY"*/>

Al doilea fragment – PointsOfInterestFragment – va afişa o catalogare a punctelor de interes găsite în urma requesturilor .



Layout-ul acestui fragment este de tip ScrollView , deoarce nu exista o structură şi o ordine fixă a butoanelor ce vor apărea , deoarece acestea sunt realizate dinamic în funcţie de ce date au fost gasite în interiorul vectorilor de date geografice , populaţi în urma requesturilor .

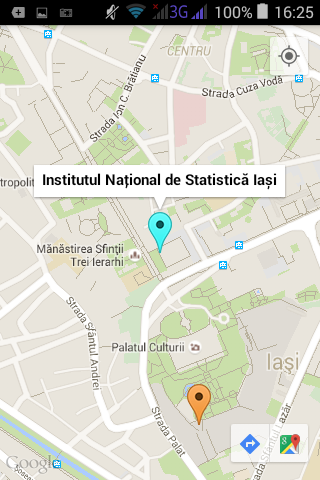
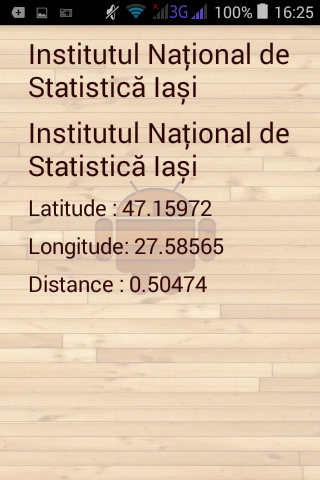
### Catalogarea înregistrărilor din vectorii punctelor de interes .

Catalogarea primului vector – MyFindNearbyArray – constă din simpla grupare a punctelor de interes de aceeaşi natură găsite . De exemplu dacă avem mai multe cafenele găsite într-un areal acestea vor fi grupate sub un singur buton care va realiza afişarea acestora . De precizat că mulţimea acestor puncte de interes depinde de arealul coordonatelor geografice date ca şi parametrii , imaginile fiind doar cu scop reprezentativ , pentru ca aplicaţia catalogheaza o variatate mare de astfel de puncte ca monumente istorice , rezervaţii naturale , obiective turistice şi geografice , ape .

Catalogarea celui de-al doilea vector – MyPointsOfInterest – reuneşte punctele de interes de acelasi tip , realizează o verificare asupra punctelor de interes depistate deja în vectorul MyFindNearbyArray , iar dacă găseşte un match , le înglobează sub aceeasi etichetă , dacă nu va crea o nouă etichetă .

Catalogarea punctelor de interes din cel de-al treilea vector – MyWikipedia – pentru această secţiune a fost creat numai un buton care reuneşte toate datele , vectorul fiind doar la un anumit număr de înregistrări în funcţie de importanţa lor după cum urmează – punctele de interes ce aprţineau claselor administrative au fost eliminate , iar cea de-a doua condiţie este ca fiecare punct de interes selectat să aibă câmpul – rank – adica importanţa acelui punct de interes în regiune mai mare ca 50 . Motivul alegerii acestui prag de 50 pentru câmpul rank a fost luat doar prin simplă observaţie a răspunsului returnat la inserarea manuală a unor coordonatelor geografice ale unor locaţii precunoscute .

Exemple de selecţii :

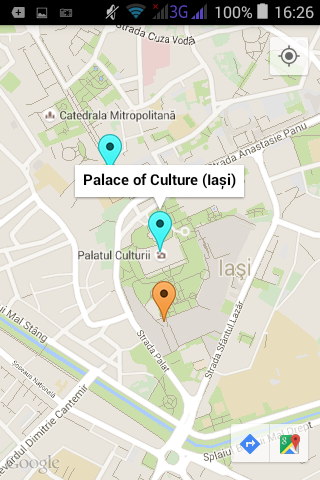
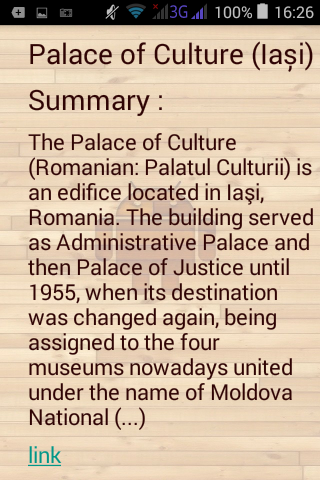
 

### Afişarea rezultatelor obţinute pentru o locaţie

Afişarea detaliilor despre un punct selectat al punctelor de interes marcate în interiorul hărţii se realizează în cadrul activităţii ***MarkerActivity*** . Selectarea unui punct făcându-se în cadrul hărţii prin simplul click asupra unui marker de locaţie .

În funcţie de tipul punctului de interes afişat , sunt două tipare de afişare . Fie doar afişarea simplă a coordonatelor geografice şi a distanţei aproximative de la locaţia din punctul unde fotografia a fost realizată pana la punctul de interes sau afisarea informaţiilor obţinute de pe wikipedia asupra punctelor de interes din areal , anume o denumirea locaţiei , o mică descriere , partea a începutului unui articol , pentru care este pus la dispoziţie link-ul

Exemplu de afişare al unui punct de interes pentru care s-a găsit o înregistrare pe Wikipedia.

# Concluzii

Aplicaţia iPhotoTag destinată realizării tagurilor cu caracter informaţional de tip geocoding îmbină cu succes tehnologiile actuale , astfel îndeplinindu-şi scopul cât şi toate obiectivele propuse : capabilă să realizeze fotografii , să determine coordinatele geografice ale locaţiei unde a fost realizată poza , să surpindă arealul înconjurator al punctelor de interes să realizeze o catalogare şi clasificare a acestora în funcţie de tip , importanţă în zona arealului cuprins , cât şi o căutare de informaţii mult mai detaliată în una din cele mai mari enciclopedii ale internetului – Wikipedia .

### Posibile direcţii viitoare

Ducă am dori să ducem această aplicaţie la un nivel şi mai înalt tehnologic decât cel actual , folosindu-ne de tehnologiile de astăzi orizontul este vast .

Este posibilă realizarea unui sistem de detecţie a obiectului fotografiat cu posibilitatea însuşirii automate a tagului . Avem la dispoziţie posibilitatea găsirii coordonatelor geografice exacte ale locaţiei dispozitivului , iar în ziua de azi se încearcă o standardizare a dispozitivelor cu funcţionalitate de fotografiat care să respecte un standard EXIF al metadatelor pozelor[[19]](#footnote-19). În metadata fotografiei putând fi găsită orientarea dispozitivului , coordonatele GPS şi distanţa focală a dispozitivlui . Având acestea la dispoziţie , în funcţie de dispozitivul care realizează fotografia şi unghiul de înclinare al aparatului într-un plan orizontal cu precizarea că centrul imaginii trebuie sa fie baza obiectului fotografiat , am putea calcula distanţa până la obiect [[20]](#footnote-20), cum avem direcţia de fotografiere , coordonatele geografice cât şi distanţa până la obiect , am putea realiza algoritmic bazându-ne pe datele pe care le avem şi pe arealul punctelor de interes pus la dispoziţie de diferitele servicii web , desigur cu o anumită marjă de eorare , o automatizarea a procesului de tagging .

# Bibliografie

- <https://developers.google.com/maps/documentation/geocoding/>

2 - <http://www.geonames.org/>

3 - <http://www.programmableweb.com/api/geonames>

4 - <http://www.programmableweb.com/api/geonames>

5 - ISO 3166 – este un standard internaţional care defineşte codurile de nume a ţărilor şi a a teritoriilor administarative împărţite în principalele subdiviziuni – judeţe .

6 - <http://www.geonames.org/source-code/javadoc/org/geonames/FeatureClass.html>

7 - codurile aferente tipurilor de clase : <http://www.geonames.org/export/codes.html>

8 - <http://www.geonames.org/export/client-libraries.html>

9 - <https://en.wikipedia.org/wiki/Android_(operating_system)>

10 - Scheletul unei aplicații Android : <http://www.tutorialspoint.com/android/android_application_components.htm>

1 - http://developer.android.com/guide/topics/media/camera.html

2 <http://www.tutorialspoint.com/android/android_sqlite_database.htm>

3 https://developer.android.com/training/volley/index.html

4 - http://www.geonames.org/export/web-services.html#findNearby

5 - http://www.geonames.org/maps/osm-reverse-geocoder.html#findNearbyPOIsOSM

6 - http://www.geonames.org/export/wikipedia-webservice.html#findNearbyWikipedia

7 - https://thepseudocoder.wordpress.com/2011/10/05/android-page-swiping-using-viewpager/

8 - http://stackoverflow.com/questions/19353255/how-to-put-google-maps-v2-on-a-fragment-using-viewpager/25103187#25103187

19 - <http://www.opanda.com/en/pe/help/gps.html>

20 - <http://hyperphysics.phy-astr.gsu.edu/hbase/class/phscilab/imagei.html>

1. - <https://developers.google.com/maps/documentation/geocoding/> [↑](#footnote-ref-1)
2. - <http://www.geonames.org/> [↑](#footnote-ref-2)
3. - <http://www.programmableweb.com/api/geonames> [↑](#footnote-ref-3)
4. - <http://www.programmableweb.com/api/geonames> [↑](#footnote-ref-4)
5. - ISO 3166 – este un standard internaţional care defineşte codurile de nume a ţărilor şi a a teritoriilor administarative împărţite în principalele subdiviziuni – judeţe .

   <https://en.wikipedia.org/wiki/ISO_3166> [↑](#footnote-ref-5)
6. - <http://www.geonames.org/source-code/javadoc/org/geonames/FeatureClass.html> [↑](#footnote-ref-6)
7. - codurile aferente tipurilor de clase : <http://www.geonames.org/export/codes.html> [↑](#footnote-ref-7)
8. - <http://www.geonames.org/export/client-libraries.html> [↑](#footnote-ref-8)
9. - <https://en.wikipedia.org/wiki/Android_(operating_system)> [↑](#footnote-ref-9)
10. - Scheletul unei aplicații Android : <http://www.tutorialspoint.com/android/android_application_components.htm> [↑](#footnote-ref-10)
11. - http://developer.android.com/guide/topics/media/camera.html [↑](#footnote-ref-11)
12. <http://www.tutorialspoint.com/android/android_sqlite_database.htm> [↑](#footnote-ref-12)
13. https://developer.android.com/training/volley/index.html [↑](#footnote-ref-13)
14. - http://www.geonames.org/export/web-services.html#findNearby [↑](#footnote-ref-14)
15. - http://www.geonames.org/maps/osm-reverse-geocoder.html#findNearbyPOIsOSM [↑](#footnote-ref-15)
16. - http://www.geonames.org/export/wikipedia-webservice.html#findNearbyWikipedia [↑](#footnote-ref-16)
17. - https://thepseudocoder.wordpress.com/2011/10/05/android-page-swiping-using-viewpager/ [↑](#footnote-ref-17)
18. - http://stackoverflow.com/questions/19353255/how-to-put-google-maps-v2-on-a-fragment-using-viewpager/25103187#25103187 [↑](#footnote-ref-18)
19. - <http://www.opanda.com/en/pe/help/gps.html> [↑](#footnote-ref-19)
20. - <http://hyperphysics.phy-astr.gsu.edu/hbase/class/phscilab/imagei.html> [↑](#footnote-ref-20)